UNIVERSIDAD ANÁHUAC
VINCE IN BONO MALUM

Metodología para el desarrollo de aplicaciones en la plataforma: Linux PDA

Guillermo Prieto Daza

Tesis presentada para la obtención del grado de
Maestro en Ingeniería
(Tecnologías de la Información)

Centro de Alta Dirección en Ingeniería y Tecnología
Universidad Anáhuac
México
Enero de 2005

Dedicado a

A mi hermano Alberto (q.e.p.d.)

A mis padres.

A la comunidad de software libre.

Una metodología integral para el desarrollo de aplicaciones en la plataforma Linux PDA

Guillermo Prieto Daza

Entregado para la obtención del grado de Maestro en Ingeniería
Enero 2005

Resumen

El trabajo describe el modelo teórico-práctico y conceptual de una metodología para el desarrollo de aplicaciones en la plataforma Linux PDA. Esta secuencia de métodos ilustran todo el proceso desde la configuración de la PDA con Linux integrado para desarrollo, la instalación de servicios y aplicaciones disponibles en la comunidad de software libre, la instalación de paquetes de desarrollo para la PC de escritorio, y, finalmente la programación e instalación de las aplicaciones en ésta PDA.

Declaración

Ninguna parte de esta tesis se ha enviado a ningún lugar para la obtención de ningún grado académico. Toda la tesis representa mi trabajo a menos que se exprese lo contrario.

[1]GPL - General Public Licenses, en Español Licencia Pública General, emitida por la FSF(Free Software Fundation). Mayor información se puede obtener del sitio http://www.fsf.org

Reconocimientos

Agradezco a mi hermana Guadalupe quien indirectamente me proporcionó los medios para adquirir el equipo y libros necesarios para ésta tesis. Agradezco además a mis compañeros y amigos Martha Abarca, Martha Alicia Hernandez, Alicia Bernal y agregados de la FEFA con quienes forme un gran equipo de trabajo.

Índice general

. Resumen IV

. Declaración V

. Reconocimientos VI

1. Justificación 1

1.1. Objetivos 2

2. Antecedentes 4

2.1. Ambiente Linux en Crecimiento 4

2.2. ¿ Es conveniente Linux para las PDAs ? 5

2.3. Orígenes (principios) de Linux sobre PDAs 5

2.4. La catedral y el bazar 6

3. Selección del Hardware 7

3.1. Dispositivos del entorno Linux integrado 7

3.2. Asistentes Personales Digitales (PDAs) 8

3.3. Cuadros comparativos de PDAs 13

3.4. Selección del modelo Sharp - Zaurus 14

3.4.1. ¿ Qué pasa con la metodología si se selecciona otro modelo de PDA ? 16

4. Entornos Gráficos de Trabajo(IDE) 17

4.1. Implementaciones provistas por el kernel de handheld.org 17

4.1.1. Century Software Microwindows 18
4.1.2. Qt-Palmtop 19
4.1.3. Transvirtual Technology's PocketLinux 20
4.1.4. Opie 21
4.2. ¿ A donde vamos a partir de aquí ? 22
4.3. Selección de entorno gráfico (IDE) 22
4.3.0.1. Opie, una buena ópcion 22
4.3.0.2. Selección de Qtopia como entorno de trabajo 22

5. Herramientas de Desarrollo. 26
5.1. Que es Open PDA ? 26
5.2. Qt 27
5.3. Jeode edición PDA 29
5.3.1. Jeode en la PDA Zaurus 30
5.3.2. PersonalJava 31
5.3.3. Perfil CDC de J2ME suscesor de PersonalJava 31
5.3.4. Controladores JDBC 32
5.3.4.1. Sintaxis URL para Base de datos de JDBC 33
5.4. Php sobre Apache 34
5.4.1. ¿Qué es PHP? 34
5.4.2. ¿ Qué se puede hacer con PHP ? 34
5.5. Aplicaciones comerciales de Rápido Desarrollo (RADS) para Java 37
5.5.1. Metrowerks CodeWarrior Development Studio para Linux Integrado . 37
5.5.2. Metrowerks CodeWarrior Wireless Studio para Java 38
5.5.3. Borland JBuilder 38
5.6. Herramientas front-end de almacenamiento y acceso a bases de datos 39
5.6.1. PortaBase 39
5.6.1.1. Características 39
5.6.2. JCards 40

5.6.2.1. Caracteristicas 40
5.6.3. TkcRekall, una base de datos comercial 41
5.6.3.1. Características 41
5.7. Bases de datos para la Zaurus 42
5.7.1. Mysql 43
5.7.2. Postgresql 43
5.7.3. Sybase iAnywhere 44

6. VII. Metodología 45
6.1. Cuadros sinópticos de la metodología 46
6.2. Si se cambia el modelo de PDA, ¿ funciona la metodología ? 49
6.3. Configuración básica de la PDA Zaurus 51
6.3.1. Actualización del ROM para el modelo Zaurus SL-5000 51
6.3.1.1. Actualización del ROM desde Linux 51
6.3.2. Aplicaciones consideradas esenciales 52
6.3.2.1. Terminal 52
6.3.2.2. Ligas simbólicas a librerías 53
6.3.2.3. Utilerías para ARM /bin 56
6.3.2.4. Openssh 56
6.3.2.5. VNC Server 58
6.3.2.6. PDF viewer 59
6.3.2.7. Librerías 59
6.3.3. Base de Datos Mysql 59
6.3.3.1. Instalación de Mysql 59
6.3.3.2. Arrancando MySQL 60
6.3.3.3. Instalación de PhpMyAdmin, Software de administración para Mysql 60
6.3.3.4. Instalación del controlador JDBC 61

6.3.4. Servidor Web Apache con PHP versión 4 61
6.3.4.1. Instalación de Apache con PHP 62
6.3.4.2. Arranque, reinicio y detención del servidor Apache con php 63
6.3.5. Redes y comunicaciones 63
6.3.5.1. Samba 63
6.3.6. Otras aplicaciones 64
6.3.6.1. Advance File Manager 64
6.3.6.2. HTML@Z 64
6.3.6.3. Rotate 64
6.3.6.4. Cute Icons 64
6.3.7. Configuración del puerto IrDa para impresión. 65
6.3.8. Recomendaciones al finalizar la configuración de la PDA 65
6.3.8.1. Creación de memoria swap 65
6.3.8.2. Respaldo 66
6.3.8.3. Restauración 68
6.3.8.4. Organizar iconos 69
6.4. Configuracion básica de la PC de escritorio 69
6.4.1. Necesidad de un equipo de escritorio ó laptop. 69
6.4.2. Instalación de la base de datos Mysql en la PC de escritorio 69
6.4.3. PhpMyAdmin - Software de administración para Mysql 71
6.4.3.1. Instalación de phpMyAdmin en la PC de Escritorio 72
6.4.3.2. Configuración de phpMyAdmin
72
6.4.3.3. Acceso a PhpMyAdmin en la red 73
6.4.4. Configuracion básica para programación en Qt. 74
6.4.4.1. Instalación del SDK - QPE 74
6.4.4.2. Instalación del compilador multi-plataforma (Cross-Compiler Setup) 74
6.4.4.3. Variables de entorno del compilador multiplataforma . . . 75

6.4.4.4. Configuración de tmake.conf para compilación ARM de la Zaurus. 76
6.4.4.5. Prueba del compilador con aplicación de ejemplo 77
6.4.4.6. Empleo de qvfb en X11 78
6.4.4.7. Qt Designer para aplicaciones ARM en la Zaurus. 79
6.4.4.8. Herramienta tmake 79
6.4.5. Configuracion básica para programación en PersonalJava. 82
6.4.5.1. Instalación de herramientas básicas para desarrollo en PersonalJava 82
6.4.5.2. Instalación del plugin de Java en el navegador Mozilla . . . 83
6.4.5.3. Instalación del controlador para acceso a Mysql 83
6.4.6. Configuración básica para programación en PHP. 84
6.4.6.1. Instalación de PHP con conexión a MySQL. 84
6.4.6.2. Configuración de PHP 84
6.4.6.3. Integración de PHP con Apache. 85
6.4.6.4. Configuración MIME 85
6.4.6.5. Reinicio del servidor web Apache 86
6.5. Desarrollo de aplicaciones en Qt 87
6.5.1. Usando la documentacion de referencia de Qt. 87
6.5.2. El primer programa multi-plataforma en Qt - holamundo.cpp 88
6.5.2.1. Código fuente - holamundo.cpp 88
6.5.2.2. Definición de variables de entorno para compilación según plataforma deseada. 89
6.5.2.3. Creación del archivo de proyecto - holamundo.pro 89
6.5.2.4. Compilación de holamundo.cpp 90
6.5.2.5. Corriendo el programa holamundo 90
6.5.3. Otro programa multi-plataforma - qtscribble. 90
6.5.3.1. Código fuente - qtscribble.cpp 90

6.5.3.2. Definiendo variables de entorno para compilación según plataforma deseada. 92
6.5.3.3. Creación de archivo de proyecto - qtscribble.pro 92
6.5.3.4. Compilación de qtscribble.cpp 92
6.5.3.5. Ejecutando qtscribble 93
6.5.4. Programa multi-plataforma con acceso a base de datos mysql. 93
6.5.4.1. Creación de base de datos *courses* 93
6.5.4.2. Código fuente dblistview.cpp 94
6.5.4.3. Definiendo variables de entorno para compilación según plataforma deseada. 94
6.5.4.4. Creación de project file dblistview.pro 94
6.5.4.5. Compilación dblistview.cpp 94
6.5.5. Programa para impresión de códigos de barra vía IrDa 95
6.5.6. Programa para transmisión de archivos vía IrDa 96
6.5.7. Instalación de aplicaciones Qt en la Zaurus 97
6.5.7.1. El paquete ipkg 98
6.5.7.2. Creación de la estructura de directorios para el paquete .ipk 98
6.5.7.3. Descripción de archivos requeridos para crear el paquete .ipk 99
6.5.7.4. Consideraciones especiales para el rom V3 102
6.5.7.5. Creación del paquete ipk 103
6.5.7.6. El script ipkg 103
6.5.7.7. Transfiriendo archivos a la Zaurus 104
6.5.7.8. Instalación del paquete ipk en la Zaurus 105
6.5.8. Creación de paquete iPKG utilizando la herramienta mkipks 105
6.5.8.1. Lista de archivos para la creación de paquete ipk para la aplicación ir.cpp 105
6.5.8.2. Contenido de archivos para creación del paquete .ipk . . . 105
6.5.9. Creación del paquete de instalación ir_1.0.0_arm.ipk 107
6.5.9.1. Librerías requeridas por el programa *ir* 108

6.6. Desarrollo de aplicaciones en PersonalJava 108
6.6.1. Runtime de PersonalJava en la Zaurus 109
6.6.2. Consideraciones escenciales previas al desarrollo en PersonalJava . . 109
6.6.2.1. Lista de paquetes PersonalJava 109
6.6.2.2. Paquetes opcionales soportados por la Zaurus 110
6.6.2.3. JDBC un conducto a MySQL. 110
6.6.3. Programación de aplicaciones en PersonalJava 111
6.6.3.1. Primer programa Hola PersonalJava 112
6.6.3.2. Creación de un Applet llamado holaPJavaApplet.java . . . 115
6.6.3.3. Interacción con MySQL en PersonalJava 117
6.6.3.4. Programa ir.java para impresión de códigos de barra vía IrDa 121
6.6.4. Instalación de aplicaciones PersonalJava en la Zaurus 122
6.6.4.1. Herramienta ipkg-build 123
6.6.4.2. Creación de directorios para el paquete .ipk 123
6.6.4.3. Listado de archivos necesarios para crear el paquete .ipk . 124
6.6.4.4. Descripción de archivos necesarios para crear el paquete .ipk 124
6.6.4.5. Contenido de los archivos requeridos para crear el paquete .ipk 124
6.6.4.6. Creación del paquete holaPJava_1.0_arm.ipk con la herramienta ipkg-build 125
6.7. Desarrollo de aplicaciones en PHP 126
6.7.1. Programación de aplicaciones 126
6.7.1.1. Inicio y termino de un bloque con instrucciones en PHP . . 126
6.7.1.2. Primer programa PHP info 127
6.7.1.3. Combinación de código de HTML y PHP en un mismo archivo 128
6.7.1.4. Interacción con Mysql usando PHP 128
6.7.1.5. Programa de impresión de código de barras en PHP 134
6.7.2. Instalación de aplicaciones PHP en la Zaurus 135

6.8. Internacionalización (I18N) de una aplicación 136
6.8.1. Introducción 136
6.8.2. Configuración de la variable Language en Linux integrado 137
6.8.3. Aspectos para i18n de una aplicación 139
6.8.3.1. Que aspectos de i18n pueden cubrir las aplicaciones en la Zaurus 140
6.8.3.2. Unicode en la Zaurus 140
6.8.3.3. Soporte en la Zaurus para el lenguaje Chino 140
6.8.4. i18n en Qt/integrado 141
6.8.4.1. Unicode 141
6.8.4.2. Traducción de aplicaciones 141
6.8.5. i18n en PersonalJava 143
6.8.5.1. Localización de mensajes en PersonalJava 143
6.8.6. i18n en PHP 146
6.8.7. Localización de mensajes en PHP 148
6.8.7.1. Localización de mensajes con la función *sprintf*. 149
6.8.8. Localización de imágenes 149
6.8.9. Localización de inclusión de archivos 150

7. Conclusiones **153**

. Apéndice **160**

A. Introduccion **160**
A.1. Resumen de la Catedral y el Bazar 160

B. Desarrollo de aplicaciones **167**
B.1. Código fuente programas en Qt 167
B.1.1. Código fuente programa qtscribble.cpp 167
B.1.2. Creación de la base de datos *courses* 179
B.1.3. Código fuente programa qlistview.cpp 180

B.1.4. Código fuente programa para Impresión de códigos de barra vía IrDa 181
B.1.5. Código fuente de programa de transmisión vía IrDa 184
B.1.5.1. Include ir.h 186
B.1.5.2. Archivo project file (.pro) para compliación de ir.cpp 186
B.1.5.3. Archivo main.cpp de aplicación ir.cpp 187
B.2. Código fuente de aplicaciones en Java 187
B.2.1. Código fuente programa JDBCTest.java 187
B.2.2. Código fuente programa `JDBCInserta.java` 190
B.2.3. Código fuente programa Ir.java 193
B.3. Codigo fuente aplicaciones en Php 201
B.3.1. Script de PHP adivina.php 201
B.3.2. Código fuente script select.php 202
B.3.3. Código fuente script ir.php 202

C. Internacionalización de una aplicación 206
C.1. Localización de Mensajes en PersonalJava 206
C.1.1. Programa l10n.java 206
C.2. Localización de mensajes en PHP 216
C.2.1. Programa locale.php 216

D. Comandos importantes en Linux integrado. 218
D.0.2. Para arrancar Linux en la PDA sin cargar Qtopia. 218
D.0.3. Para sacar el pipeline 218
D.0.4. Comando ipkg 218
D.0.5. Usuarios en la Zaurus 219
D.0.6. Ligas simbólicas 219

Índice de figuras

3.1. Sharp Zaurus Series SL-C7XX 8
3.2. Sharp Zaurus SL-5x00 9
3.3. Compaq iPaq 9
3.4. Yopy 9
3.5. Q-Reader 10
3.6. 4P DAT500 10
3.7. Filewalker 10
3.8. GSPDA v-2002 10
3.9. HanGil 11
3.10. Informat Kaii 11
3.11. Mizi 11
3.12. Royal 11
3.13. SK Telecom 12
3.14. Softfield VR3 12
3.15. aml m7100 12
3.16. Sharp Zaurus Serie SL-5500 15

4.1. Entornos Gráficos 18
4.2. Century Software 19
4.3. Qtopia 19
4.4. Pocket Linux 20
4.5. Opie logo 21

4.6. Opie en C-7xx 23
4.7. Opie en ipaq 23
4.8. Stack Qt. 25
4.9. Arquitectura Sharp Zaurus SL-5500 25

5.1. Open PDA 26
5.2. Qt C++ 27
5.3. Qt Stack 28
5.4. Java logo 29
5.5. Esmertec 30
5.6. Jeode 31
5.7. Uso estimado de PHP en abril del 2004 34

6.1. Add/Remove Software 54
6.2. Install packages 55
6.3. backup-restore 67
6.4. Backup Media 68
6.5. Corriendo holamundo 91
6.6. Programa Qtscribble 93
6.7. Programa para impresión de códigos de barra en Qt 95
6.8. Impresora Cameo 3 96
6.9. Código de barra impreso 96
6.10. Programa impresión vía IR 97
6.11. Estructura directorios para iPKG 99
6.12. Ventana ”Details” 102
6.13. Programa holaPJava.class 115
6.14. Applet holaPJavaApplet.class 117
6.15. Programa JDBCTest.class 118
6.16. Programa JDBCTest.class en conección con otro servidor 119
6.17. Programa JDBCinserta.class 121

6.18. Impresión vía IrDa en PersonalJava 122
6.19. Jerarquía de directorios para Jeode 123
6.20. Programa adivina.php 129
6.21. Forma Insertar Datos 132
6.22. script select.php 133
6.23. Programa ir.php 134
6.24. Qtopia en Alemán 138
6.25. Wecker en Alemán 139
6.26. Programa l10n.class en Alemán 144
6.27. Programa l10n.class en Inglés 145
6.28. Programa l10n.class en Español 146
6.29. Qtopia en Chino 147
6.30. Navegador en Chino 148
6.31. localizacion de mensajes 149
6.32. Localización de mensajes con sprintf 151

Índice de cuadros

3.1. Dispositivos con Linux integrado 8
3.2. Cuadro Comparativo PDAs SHARP 13
3.3. Cuadro Comparativo otras PDAs 14

5.1. Detalle controlador JDBC para MySQL 33

6.1. Resumen configuración básica de la PDA 47
6.2. Resumen configuración básica de la PC de escritorio 47
6.3. Resumen pasos para desarrollo en Qt 48
6.4. Resumen pasos para desarrollo en PersonalJava 48
6.5. Resumen pasos para desarrollo en PHP 48
6.6. Resumen pasos para internacionalización de una apliación 49
6.7. Archivo `control` para el paquete iPKG 100
6.8. Comandos para crear paquete ir_1.0.0_arm.ipk 107
6.9. creación del paquete ir_1.0.0_arm.ipk 108
6.10. Paquetes PersonalJava 109
6.11. Paquetes adicionales Jeode 110
6.12. Código fuente programa holaPJava.java 113
6.13. Código fuente applet holaPJavaApplet.java 116
6.14. Código archivo HTML para el applet holaPJavaApplet.class 116
6.15. Lista de archivos y ubicación en Jeode 124
6.16. Etiquetas PHP 127
6.17. Script PHP de conexión a MySQL 130

6.18. Script PHP para crear una tabla 131
6.19. Forma HTML para Insertar Datos 131
6.20. Script PHP para inserción de datos 132
6.21. Código del script PHP que utiliza la función sprintf 150
6.22. Código del script PHP para localización de imágenes 151
6.23. Código del script de PHP para localización de inclusión de archivos 152

Capítulo 1

Justificación

Dado el rápido surgimiento del entorno Linux como la mayor "tercera alternativa" de sistema operativo para los dispositivos personales de bolsillo además de Palm y Pocket PC, y a la ausencia de publicaciones que conjunten puntos concretos y sintetizados para el desarrollo de aplicaciones para la plataforma Linux PDAs, se abre un espacio para presentar esta metodología como la introducción a una nueva serie de dispositivos Linux que exploran la historia, el estado, las arquitecturas alternativas, y los progresos futuros en Linux PDAs y los dispositivos handheld.

¿Por qué el entorno Linux? Así como las distribuciones de Linux para la PC de escritorio y los servidores, los productos basados en un entorno Linux raramente consisten exclusivamente del sistema operativo[1]. El entorno Linux enfoca el software comercial y de la comunidad de código abierto (open source community) en una solución confiable para el dispositivo de bolsillo "Handheld" de una manera estandarizada, fiable y comercial. Los productos que corren en un entorno Linux, tales como PDAs y teléfonos inteligentes, integran el sistema operativo, las aplicaciones y los servicios asociados al dispositivo.

Los usuarios de un dispositivo de bolsillo (Handheld) no necesitan utilizar Linux, necesitan hacer una llamada telefónica, encontrar una dirección, ver una película, registrar unos

[1] Integran además aplicaciones de información personal (PIM), programas de oficina y de comunicaciones, entre otros.

datos ó corregir un documento al encontrase en tránsito. De hecho, las mejores puestas en práctica ó implementaciones del entorno Linux en los dispositivos handheld no le notifican al usuario final del subyacente sistema operativo Linux que su dispositivo utiliza. La plataforma Linux PDA alcanza esta meta aparentemente combinando lo mejor posible el middleware y las aplicaciones, de una manera fácil e intuitiva.

El futuro de la computación móvil está en las manos con los dispositivos de bolsillo "handheld". La plataforma Linux PDA, es la base de una solución completa de software tanto para los asistentes personales digitales como para los teléfonos inteligentes "Smart Phones", que permite resaltar los innovadores diseños de hardware con el mejor software disponible de productividad, entretenimiento y de información para la alta dirección.

Linux PDA debe significar - Entendible. Los fabricantes de hardware que construyen hoy la mejor clase de PDAs no pueden afrontar el construir proyectos basados en una misteriosa caja negra binaria. Los ingenieros saben que ninguna herramienta de desarrollo de características intensivas puede reemplazar lo práctico de las soluciones de código abierto de Linux para las PDAs. La alta dirección que requiere entender y controlar el destino de sus proyectos para sus handhelds, prefiere una solución de un entorno de trabajo abierto "open framework", a cerrarse sólo en implementaciones binarias.

Con soporte a nivel mundial en Linux, Java, y las herramientas de desarrollo nativas, no se necesita ser un "guru" de Linux para tener el trabajo hecho en tiempo y dentro del presupuesto.

1.1. Objetivos

1. Presentar el entorno Linux PDA como una solución real e inmediata de calidad para dispositivos de bolsillo, definiendo como una solución de calidad a aquella que satisface por completo las expectativas del consumidor.

2. Enumerar y describir el potencial de ésta plataforma de alto nivel tecnológico que está más allá de la comparación con las limitadas tecnologías ofrecidas por los grandes

consorcios de software; mostrar que Linux PDA ofrece excelente interoperabilidad y un conveniente ambiente de desarrollo.

3. Mostrar las distintas opciones de hardware disponibles en el mercado para esta plataforma, y poder seleccionar el hardware óptimo para la aplicación a desarrollar.

4. Resaltar las distintas herramientas de desarrollo que para esta plataforma permitirán visualizar un mundo informático independiente de las herramientas tradicionales de los grandes consorcios mundiales del software.

5. Visualizar los distintos entornos de desarrollo para Linux PDA definiendo las características y requerimientos de cada uno de estos, mostrando las capas que componen todo el entorno arriba del sistema operativo y que permitirán la ejecución de la aplicación.

6. Ejemplificar con el desarrollo aplicaciones las herramientas de desarrollo seleccionadas.

En resumen, presentar una metodología para desarrollo de aplicaciones para la plataforma Linux PDA que permita el análisis y la toma de decisión respecto del hardware y las herramientas de desarrollo más convenientes para satisfacer las necesidades del usuario de los asistentes digitales personales PDAs.

Capítulo 2

Antecedentes

2.1. Ambiente Linux en Crecimiento

Por muchos años, los defensores de Linux han predecido que Linux se convertirá en un factor importante en el entorno del mercado. Además de sus virtudes como un sistema operativo moderno y completo, resulta que otro factor muy importante en el entorno de los sistemas es que es económico de reproducir ó duplicar. Otros, que vienen de una escuela de desarrollo más tradicional han sido mas escépticos y han contrastado los viejos y más espartanos ambientes de desarrollo como VxWorks, QNX, ó Lynx, versus los requerimientos de Linux.

Hasta hace poco, esto era un debate estéril. Sí, la inclusión de características maduras y estándares de redes y de interfaces gráficas del usuario han hecho a Linux apropiado para el entorno de sistemas de alto nivel. Y sí, su tamaño lo hace inapropiado para los sistemas de bajo nivel. Pero como alguien predijo cuando el debate inicio, dos corrientes han removido por completo el factor de los requerimientos al seleccionar Linux como el entorno de sistemas.

Primero simplemente la RAM, ROM, Flash, y el poder de procesamiento necesitados por Linux se han hecho más económicos y rápidos a una taza mayor de la que Linux pudiera consumir con nuevas características añadidas. La segunda son las mejoras que ha tenido el mismo Linux. Empezando desde un kernel específico para x86 con poca modularidad,

Linux a progresado hasta convertise en una arquitectura independiente y ha obtenido una clara modularidad. La serie del kernel 2.4 resulta especialmente bien diseñada, haciendo factible el añadir ó remover componentes para optimizarlo para una plataforma en particular.

2.2. ¿ Es conveniente Linux para las PDAs ?

En ningún área esta implementación es más importante que para las PDAs, un mercado para el que Linux se observa asombradamente apropiado. Aquí hay un mercado en donde la conectividad con la PC de escritorio es importante, asi que un soporte de red estable y maduro es una necesidad. La presencia de una interfaz gráfica del usuario virtualmente define lo que una PDA es, así que el soporte de ésta interfaz también es importante.

Por el otro lado, el mercado es lo suficientemente sensible al precio para que muchos fabricantes encuentren oneroso el pagar regalías a los grandes consorcios mundiales de software por el uso de un sistema operativo.

En resumen, todo parace apuntar hacia un entorno de Linux como la plataforma actual para los cada vez más sofisticados PDAs.

2.3. Orígenes (principios) de Linux sobre PDAs

Linux en la Compaq iPAQ H3800 es probablemente el mayor primer adelanto. La iPAQ es una PDA de alto nivel que es normalmente distribuida con un sistema operativo distinto al Linux. Sin embargo, Compaq esta patrocinando un admirable esfuerzo para portar Linux a la iPaq que se encuentra la siguiente dirección: www.handhelds.org

La astucia del acercamiento de Compaq difícilmente puede ser exagerada. Ellos han hecho demasiado fácil para cualquiera el participar en el esfuerzo de desarrollo y han mostrado los resultados a todos de manera abierta. Parece ser que han enviado gratuitamente dispositivos a los mejores conocedores de Linux y del código abierto, con el resultado de que algunos de ellos se han mostrado interesados en participar en este esfuerzo de desarrollo. Este movimiento de marketing ha tenido por seguro dos resultados: primero, Linux progresa

rápidamente en la iPAQ; y segundo, el mercado seguirá encajándose en las iPAQ tanto rápido como Compaq las vaya produciendo.

2.4. La catedral y el bazar

El resumen de este ensayo (ver anexo A.1) escrito por Eric Random se incluye como ilustración del potencial de desarrollo que tiene la comunidad de código abierto en Linux y como han abrazado este estilo de desarrollo compañías tan grandes como Netscape.

En mi opinión, la catedral y el bazar nos muestra el potencial de la comunidad de software libre, la cual ha logrado crear tanto sistemas operativos como aplicaciones muy potentes, completas, seguras y de bajo costo que le permiten a Linux satisfacer las necesidades de la industria y los gobiernos. Como ejemplo podemos mencionar la decisión del gobierno Alemán[1] en el año de 2002, de desechar el sistema operativo de Microsoft de la mayoría de sus equipos administrativos, basándose en el potencial que ofrece la comunidad de software libre, a la cual, cabe mencionar que Alemania ha contribuido ofreciendo varias distribuciones de Linux como Knoppix y SUSE.

[1] En un movimiento intencionado a incrementar la seguridad y reducir costos, el Gobierno Alemán dice que cambiará la mayoría de sus equipos administrativos con el sistema operativo de Microsoft, por el sistema operativo de la comunidad de software libre: Linux. Ver [Alemania cambia a Linux]

Capítulo 3

Selección del Hardware

3.1. Dispositivos del entorno Linux integrado

Introducción a una nueva serie de dispositivos Linux que explora la historia, el estado, las arquitecturas alternativas, y los progresos futuros de Linux PDAs y los dispositivos de bolsillo (handheld).

La gente se pregunta: “Toda esta plática acerca del Entorno de Linux que sale como un cohete, suena bien, ¿ pero hay algunas compañías que realmente estén surtiendo productos reales con Linux ? y de ser así, ¿ cuándo van a empezar a posicionarse en el Mercado ?”

La respuesta es: Por supuesto, ¡se están diseñando productos reales con el entorno de Linux y muchos! Algunos ya se están enviando a los consumidores, otros se están surtiendo en grandes lotes, y muchos más se encuentran en distintas etapas de desarrollo, de hecho desde el año 2003 comenzó una gran aparición de estos.

A continuación se presenta un sumario de algunos de los dispositivos del entorno Linux. Hay que tener en cuenta que la lista que aquí aparece representa solo una pequeña muestra del *iceberg* real del entorno de Linux, y que nuevos dispositivos basados en el entorno Linux están apareciendo continuamente. El cuadro 3.1 muestra un panorama de dispositivos con Linux integrado.

De todas estas categorías nos enfocaremos a los asistentes digitales personales ó PDAs,

		
Asistentes Personales Digitales	Teléfonos móviles y basados sobre IP	Dispositivos Audio / Video
		
PCs tipo Tablet	Gateways, servers y access points	Otros dispositivos inteligentes

Cuadro 3.1: Dispositivos con Linux integrado

obteniendo de igual manera una lista pequeña de ese *iceberg* de dispositivos basados en Linux, resaltando que cada día salen al mercado nuevos dispositivos para esta plataforma.

3.2. Asistentes Personales Digitales (PDAs)

La siguiente es una lista recopilada de PDAs de diferentes marcas y características existentes en el mercado .

Sharp Zaurus Series SL-C7xx -- Sharp llama a los modelos de PDAs SL-C7xx "la primera PDA del mundo con una resolución VGA a color (640x480 pixeles) pantalla LCD." Las PDAs SL-C7xx también incluyen un teclado único integrado QWERTY similar al teclado normal de una notebook(aunque mas pequeño). Al igual que los modelos Zaurus anteriores, la plataforma de software esta basada en Linux junto con la máquina virtual de java JVM para ofrecer una mayor portabilidad del software y el soporte de desarrollo de código abierto a nivel mundial "world-wide open source"

Figura 3.1: Sharp Zaurus Series SL-C7XX

Sharp Zaurus SL-5x00 – La primera PDA con Linux de Sharp para el Mercado mundial presentando un entorno de sistema operativo Linux complementado por un entorno de Qt y Java para ejecución de aplicaciones. La SL-5500 tiene un procesador Intel StrongARM a 206 Mhz con 64MB de memoria RAM y 16MB para almacenamiento tipo FLASH, mientras que el modelo SL-5600 usa un procesador Intel XScale a 400 Mhz y cuenta con 32MB de memoria RAM y 64MB para almacenamiento tipo flash. Los dos modelos tienen pantalla de cristal líquido a color TFT con resolución de 320 x 240 pixeles, e incluyen un teclado miniatura integrado QWERTY (visible al mover la cubierta deslizable), están equipadas con ranuras CompactFlash y SD para expansión, y la interfaz IrDA y USB.

Figura 3.2: Sharp Zaurus SL-5x00

Compaq iPAQ – aunque no este equipada por default con el entorno de Linux, existen muchos proyectos ya para el desarrollo de implementaciones de Linux en la iPaq, incluído uno de los promotores de Compaq en el sitio handhelds.org. La iPAQ da 240 x 320 pixeles a color LCD y tiene un procesador Intel StrongARM a 206MHz con 32MB de RAM y 16MB de memoria flash. La interfaz externa es vía IrDA, serial (sync/async), USB, y PCMCIA. Varias distribuciones de Linux que soportan la iPAQ están disponibles.

Figura 3.3: Compaq iPaq

G.Mate Yopy -- la Yopy integra las funciones de acceso a email e internet (via teléfono celular), grabadora digital de voz, reproductor de MP3 (audio) y MPEG (video), juegos, además de un set completo de aplicaciones para PDA. El modelo YP3000 tiene una pantalla a color QVGA (240 x 320 pixeles), con procesador Intel StrongARM a 206MHz equipado con 64MB de RAM y 16MB de Flash ROM, intefases RS232 serial y USB. El siguiente modelo YP3500 tiene una pantalla backlight y más memoria (128MB RAM, 32MB Flash), mientras que la YP3700 tiene un socket integrado para CompactFlash.

Figura 3.4: Yopy

Q-Reader Ebook – este dispositivo electrónico de lectura con Linux tiene como objetivo el Mercado de la educación Chino. Además de las capacidades de lectura, las distintas funciones del dispositivo incluyen conectividad al Web, email, y algunas otras características del PDA, incluyendo soporte para los lenguajes Inglés y Chino.

Figura 3.5: Q-Reader

4P DAT500 handheld – este es un handheld para trabajo *industrial*, que pretende usarse en aplicaciones para puntos de venta y manejo de inventarios movibles. En adición a todas las funciones esperadas de una computadora tipo Palm (interfaz gráfica, touchscreen, reconocimiento de escritura), la DAT500 es muy confiable y mas resistente, y ha pasado muchas pruebas para acreditar estándares de seguridad de la aviación comercial. La unidad incluye lector de código de barras, lector de tarjeta de crédito, y una impresora.

Figura 3.6: 4P DAT500

Invair Filewalker – Tecnologías Invair (Stuttgart, Alemania) presentó su nueva PDA basada en Linux en la expo CeBIT 2002 en Hannover, Alemania. El dispositivo esta diseñado para operar con una mano, pesa solo 0.2 libras, y es pequeña(aunque un poco ancho), midiendo 3.4 x 2.2 x .74 pulgadas. Esta basada en un procesador Intel StrongARM con reloj a 133MHz e incluye 32MB de SDRAM y 16MB de Flash ROM, tiene una pantalla en escala de grises con resolución de 160 x 240 pixeles, y provee interfaz IrDA y puerto USB además de un splot de expansión SD y MMC .

Figura 3.7: Filewalker

GSPDA V-2002 – la PDA con Linux para el Mercado Chino basada en una procesador Intel StrongARM a 206 Mhz con 32MB de RAM y 32MB de memoria Flash, además puertos USB, audio in/out, BlueTooth integrado, y una ranura de expansión tipo SD.

Figura 3.8: GSPDA v-2002

HanGil C3224 multimedia PDA – basada en un procesador Intel StrongARM SA-1110 a 206 Mhz de 32-bits, con 16MB de RAM y 16MB de memoria Flash. La pantalla es de 3.8-pulgadas a color Œ VGA (240 x 320 pixeles) TFT LCD, con touchpad, y el dispositivo ofrece puerto USB, IrDA, y RS232C integrado, así como una ranura para memoria CompactFlash.

Figura 3.9: HanGil

Infomart Kaii – esta nueva PDA basada en Linux fue creada para satisfacer un Mercado entre los PDAs de largo alcance y alto costo de marca Pocket PC y los de corto alcanze y bajo costo como Palm. El dispositivo intenta ofrecer software de alto nivel compatible con Zaurus, mediante el empleo de una plataforma similar. A diferencia del PDA Zaurus, este utiliza un procesador Hitachi SH3 a 160MHz y posee un teclado en pantalla en vez de uno físico. Incluye 32 a 128 Mb RAM junto con 32 MB de ROM(o Flash) dependiendo de la versión.

Figura 3.10: Informat Kaii

Mizi Linux PDA – este modelo de PDA se basa en un procesador StrongARM SA-1110 a 206 Mhz y tiene una pantalla LCD con resolución de 240 x 320 pixeles TFT . Incluye 32M de RAM y 32MB de memoria Flash, un slot CompactFlash, interfaz USB y puerto IrDA, radio digital interno FM, y una bocina incluida.

Figura 3.11: Mizi

Royal Lin@x -- Royal presentó un prototipo de su nueva PDA basada en Linux en Las Vegas, NV en Enero del 2002. El dispositivo esta basado en un procesador Intel StrongARM a 206 MHz con 32MB de RAM y 16MB de memoria flash. Tiene una pantalla a color LCD TFT con resolución de 320 x 240 pixeles, y está equipada con un spot de expansión CompactFlash, y puertos USB, IrDA, y RS232. El stack del software está basado en la suite de software para PDA de Century Software's PIXIL, que incluye la interfaz gráfica del usuario Microwindows .

Figura 3.12: Royal

SK Telecom IMT2000 WebPhone – este dispositivo es una combinación entre teléfono celular + PDA. Tiene una pantalla de 4-pulgadas LCD y una video cámara integrada. Las funciones de PDA utilizan un procesador StrongARM SA1110 a 206MHz ,y contiene 32MB de RAM + 32MB de memoria tipo flash. El sistema operativo es de una compañía llamada PalmPalm's Tynux que utiliza un entorno Linux, con soporte para la interfaz gráfica del usuario de Qt/Embedded y el navegador Opera. Un microprocesador separado controla las funciones de teléfono celular.

Figura 3.13: SK Telecom

Softfield VR3 – la VR3 de Softfield Technologies (originalmente introducida por Agenda Computing) es una computadora tipo hanheld con funciones completas, una pantalla con resolución de 160 x 240 pixeles LCD. Está basada en un procesador NEC VR4181 MIPS a 66MHz de 32-bits, y tiene 8MB o 16MB de memoria RAM y 16MB de almacenamiento tipo Flash. Para I/O, el dispositivo ofrece un puerto serial RS232 así como una interfaz infrarroja IrDA .El sistema operativo de la VR3 se llama Linux-VR.

Figura 3.14: Softfield VR3

Terminal de colección de datos AML M7100 – este dispositivo handheld para el Mercado de Retail ó detallista posee un procesador StrongARM SA-1110, equipado con 16MB de RAM y 4MB de Flash ROM. Sus interfaz de entrada y salida (I/O) incluyen entradas múltiples, de scanner, RS232, USB, infrarojo (IrDA), y una interfaz de red IEEE 802.11b por radiofrecuencia.

Figura 3.15: aml m7100

3.3. Cuadros comparativos de PDAs

El cuadro 3.2 muestra un comparativo de PDAs marca Sharp, y el cuadro 3.3 muestra un comparativo de otras marcas de PDAs, indicándose el precio aproximado en Diciembre del 2004.

Características			
	Sharp SL-6000	Sharp SL-C3000	SL-5600
Resolución	VGA	VGA	240x320 pixeles
Procesador	XScale 400 mhz	XScale 416 mhz	XScale 400 mhz
Entorno Gráfico	Qtopia	Qtopia	Qtopia
Red Inhalám.	Integrada	Slot CF	Slot CF
JVM	J2ME	J2ME	Jeode-PJava
Memoria RAM	64MB	64MB	64MB
Memoria FLASH	64MB	16MB	16MB
Ranuras	1 CF,1 SD	1 SD, 1 CF	1 CF, 1 SD
IrDa	Incluído	Incluído	Incluído
Micrófono	Incluído	Incluído	Incluído
Bocina mono	Incluída	Incluída	Incluída
Salida audífonos	Incluída	Incluída	Incluída
Disco duro	N/A	4GB	N/A
Precio USD	$ 500.00	$ 735.00	$ 300.00

Cuadro 3.2: Cuadro Comparativo PDAs SHARP

Características			
	G.Mate Yopy 3700	Nec VR3	Compaq iPaq 3800
Resolución	240x320 pixeles	160x240 mono	240x320 pixeles
Procesador	Strong ARM 206 Mhz	NEC VR4 66mhz	Strong ARM 206 mhz
Entorno Gráfico	X window	Linux VR OS	Qtopia instalable
Red Inhalám.	Integrada	N/D	N/D
JVM	N/D	N/D	Jeode-PJava
Memoria RAM	128MB	16MB	32MB
Memoria FLASH	32MB	16MB	16MB
Ranuras	1 CF,1 MMC	N/D	1 SD
IrDa	Incluído	Incluído	Incluído
Micrófono	Incluído	N/D	Incluído
Bocina mono	Incluída	N/D	Incluída
Salida audífonos	Incluída	N/D	Incluída
Precio USD	$ 400.00	$ 135.00	$ 300.00

Cuadro 3.3: Cuadro Comparativo otras PDAs

3.4. Selección del modelo Sharp - Zaurus

Se selecciona el modelo Zaurus de Sharp como la PDA para desarrollo en la plataforma Linux (la figura 3.16 muestra la PDA Sharp Zaurus series SL-5500). Esta PDA se seleccionó por ser la más apropiada para el desarrollo de aplicaciones así como las más versátil. Entre sus características se enumeran las siguientes:

Figura 3.16: Sharp Zaurus Serie SL-5500

1. Contiene un teclado integrado que facilita la escritura de comandos y captura de datos
2. Cuenta con una ranura para tarjetas de memoria secure digital lo que permite almacenamiento muy superior a la memoria con que originalmente viene este equipo
3. También tiene una ranura para tarjetas compact flash en donde se puede instalar una tarjeta de red inhalámbrica.
4. El entorno gráfico de esta PDA, Qtopia, cuenta con soporte de herramientas para desarrollo de aplicaciones en C++ de la compañía Trolltech.
5. Esta PDA incluye en el ROM un runtime de Personal Java lo que permite la ejecución de programas en Java.
6. Cuenta con el navegador Opera para que permite ejecutar aplicaciones en PHP [1]

Más adelante en el capítulo 5 se describirán las herramientas de desarrollo.

[1] Se requiere la instalación del servidor web Apache con PHP para Linux integrado, el cual está disponible para PDAs con procesadores StrongARM ó XScale.

3.4.1. ¿ Qué pasa con la metodología si se selecciona otro modelo de PDA ?

Esta metodología funciona para PDAs con Linux integrado de diversas marcas, pero con la restricción de que aplica sólo para aquellas que cumplan los requerimientos de software y ciertas características que se describen en el punto 6.2, más adelante en la metodología.

Capítulo 4

Entornos Gráficos de Trabajo(IDE)

4.1. Implementaciones provistas por el kernel de handheld.org

Actualmente todos los esfuerzos para implementar Linux en la iPAQ están, hasta donde parece, concentrándose en la portabilidad para el kernel de Linux de handhelds.org. Pero las implementaciones en la interfaz gráfica del usuario contenidas en este kernel son más diversas. La implementación de handhelds.org contiene una línea directa para la portabilidad de sistema X-Window en la implementación en los XFree86 hacia los procesadores StrongARM y XScale usados por las PDAs. Al menos otros cuatro grupos proveen implementaciones alternativas para la interfaz gráfica del usuario ó GUI. Century software provee sus herramientas de desarrollo Microwindows, Trolltech tiene su entorno QT Palmtop, Transvirtual tiene la distribución Pocket Linux y el proyecto Opie (Open Palmtop Integrated Environment) el cual es una bifuración del entorno de trabajo de Qtopia desarrollado originalmente por Trolltech. Opie es un entorno de trabajo con interfaz gráfica de usuario basado en la comunidad de código abierto diseñado para PDA y otros dispositivos corriendo Linux. Se le da mantenimiento por un grupo de personas esparcidos por el mundo y convencidos por la filosofía de código abierto. Opie tiene usos entendidos y mejorados con respecto del original de Qtopia, lo que lo convierte en la interfaz gráfica de usuario gratuita más sofisticada para PDAs y dispositivos con Linux integrado.

Figura 4.1: Entornos Gráficos

La figura 4.1 muestra algunos de los entornos gráficos de trabajo. Cada una de estas implementaciones se construyeron sobre la base provista por el kernel de handhelds.org. A continuación se analiza cada una de ellas.

4.1.1. Century Software Microwindows

Como otras herramientas que serán discutidas del entorno operativo para PDA, Century Software comenzó con el kernel, las utilerías y compilador multiplataforma de handhelds.org. A esto le aunó un número de componentes interactivos que juntos forman el entorno de trabajo de la Interfaz Gráfica de Usuario. Century software en la más fina tradición del software libre ha juntado este entorno de varios componentes previamente existentes. Microwindows provee de 2 APIs para compatibiliad con 2 entornos de trabajo las cuales permiten emular Windows y X-like GUIs

Arriba de esta emulación la siguiente capa es FLTK (Fast Light Tool Kit) lo cual provee funcionalidad moderna en GUIs sin generar engrosamiento y soporta graficos en 3D a través de OpenGL. FLTK provee el conjunto normal de widgets que una GUI debe tener: casillas de texto, casilla de verificación, menus, y otros. Además de portar FLTK hacia la API enfocada hacia Nano-X, se trabajo en modificar su modo de verse ó look and feel para ser mas de 2D, el cual creen (en concordancia con Palm y otros) resulta ser mas apropiado para las PDAs.

Figura 4.2: Century Software

Figura 4.3: Qtopia

4.1.2. Qt-Palmtop

Los desarrolladores de la herramienta Qt de Trolltech, liberaron su entorno Qt palmtop. Mucha gente recuerda la historia de Qt, especialmente en el contexto de KDE, el entorno de desarrollo escrito con Qt.

Mientras Qt en si se enfoca a ambientes Windows y X, el entorno Qt Palmtop esta basado en Qt/Integrado de Trolltech, una versión de Qt intencionada para aplicaciones de tamaño pequeño. Al igual que microwindows no necesita tener el entorno X presente. El entorno de Trolltech es conceptualmente mucho más simple que el de Century Software. La aplicación es escrita para al API de Qt; luego Qt/Embedded interactua directamente con las funciones de

Figura 4.4: Pocket Linux

I/O de Linux. Este es probablemente el entorno ideal para desarrolladores que deseen utilizar la API de Qt. La ausencia de sobre posición de capas hacen de Qt/Embedded el entorno más compacto para correr programas basados en Qt.

4.1.3. Transvirtual Technology's PocketLinux

PocketLinux forma el tercer mayor entorno de desarrollo poveeyendo soporte a la iPAQ. Transvirtual fue fundada por los creadores de Kaffe, la máquina virutal de Java, una de las mejores implementaciones de JVM. Al igual que los otros entornos PocketLinux fue creado sobre el Linux portable de handhelds.org para la iPAQ. La arquitectura de Transvirtual puede ser simplemente sumarizada como Kaffe implementado en Linux. Su mercado puede ser identificado mas claramente y sucintamente que el de Trolltech: Desarrolladores de Java.

Para aquellos que usan Java, la atracción hacia PocketLinux es clara. PocketLinux es proveído para la compañia lider de software libre de Java, así que uno puede esperar buen soporte y un cometido de largo término hacia el éxito del producto.

La presentación de PocketLinux se espera con escepticismo por muchos, particularmente por sistemas interconstruídos de viejos tiempos, algunos de los cuales nunca se han ajustado a la idea de usar un compilador, muchos menos un interprete de byte-codes, en un sistema integrado.

Como uno puede esperar de una plataforma Java pura, PocketLinux tiene un elegante modelo de programación. Pantallas de interfaz de usuario son desplegadas usando XML, luego los eventos asociados con los widgets que fueron definidos en XML son procesados en Java. Las aplicaciones se heredan de una clase de aplicaciones base llamada BaseApplication,

Figura 4.5: Opie logo

la cual provee un manejo por default de eventos. Asi pues las aplicaciones de usuario deben contemplar el manejo de los eventos que no son los que se incluyen por default.

4.1.4. Opie

Opie (Open Palmtop Integrated Environment) es completamente un entorno de trabajo basado en código abierto para PDAs y otros dispositivos que corren Linux.

El Opie es una bifurcación del entorno Qtopia de Trolltech. Opie en sus principios emergió como la interfaz gráfica de usuario gratuita más sofisticada para dispositivos y PDAs interconstruídas con Linux.

Opie tiene un sofisticado entorno de trabajo de información personal así como muchos otras aplicaciones productivas, capacidades extendidas para multimedia, herramientas de red y comunicación así como soporte multilenguaje para más de una docena de ellos.

Basada en los estándares comúnes de la industria como XML, Obex, IrDa, etc. Todos los Opies son capaces de interactuar con gran rango de dispositivos desde teléfonos celulares hasta servicios de fondo de servidor. Opie esta altamente optimizada para dispositivos móviles y trata de soportar al usuario con accesos directos y un fácil manejo.

No importa si uno desea organizar cuestiones de la vida diaria, mantenerse al día leyendo noticias ó si se requiere de una terminal móvil de acceso a internet, Opie provee todas las capacidades necesarias para el uso cotidiano.

4.2. ¿ A donde vamos a partir de aquí ?

Por ahora, parece claro que los entornos gráficos disponibles para las PDAs corriendo Linux se están volviendo tan diversos como las opciones que se ofrecen para las computadoras de escritorio. Esto forma un espectro, de ésos que proporcionan diversidad y opciones a aquellos que ofrecen un entorno más estructurado (de fuentes eficientes) con las herramientas de soporte favoritas.

El entorno por defecto proveído por handhelds.org es el menos estructurado, siendo un canal directo para Linux y X para la iPAQ. Las hackers tradicionales de Linux de seguro favorecerán esta distribución debido a que provee un entorno familiar para ellos y les permite usar sus herramientas preferidas. El entorno de Century Software es más flexible, proporcionando X pero usando herramientas de código abierto tradicionales con las que muchos desarrolladores ya están familiarizados. Algo más estructurado es lo que ofrece Trolltech; este esencialmente encierra a los desarrolladores en la API de Qt; en intercambio ellos tienen una API bien diseñada y un espacio de uso muy eficiente. Finalmente, Transvirtual provee Java en el entorno de la handheld. PocketLinux provee una madura y estable máquina virtual Java JVM para aquellos que deseen usar Java.

4.3. Selección de entorno gráfico (IDE)

4.3.0.1. Opie, una buena ópcion

El entorno Opie resulta bastante atractivo, aun que no cuenta con las herramientas de sincronización hacia el desktop a programas basados en Windows como el Outlook, además de no contar con el emulador Java - Jeode de Insignia dentro de la instalación del ROM. Las figura 4.6 muestra el entorno Opie en un equipo serie C-7xx y la figura en 3.3 la compaq ipaq.

4.3.0.2. Selección de Qtopia como entorno de trabajo

Qtopia es el primer entorno para aplicaciones desarrollado para Linux integrado. Este incluye PIMs (Manejo de Información Personal), productividad de negocios, aplicaciones de

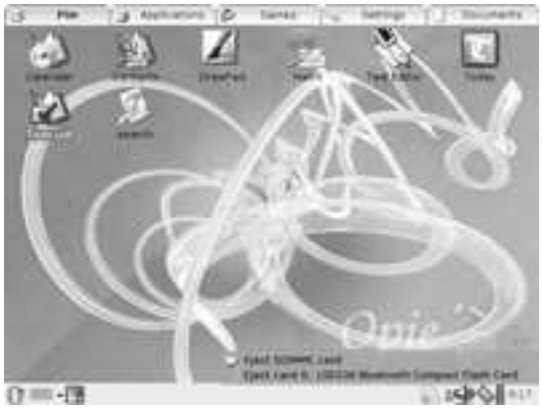

Figura 4.6: Opie en C-7xx

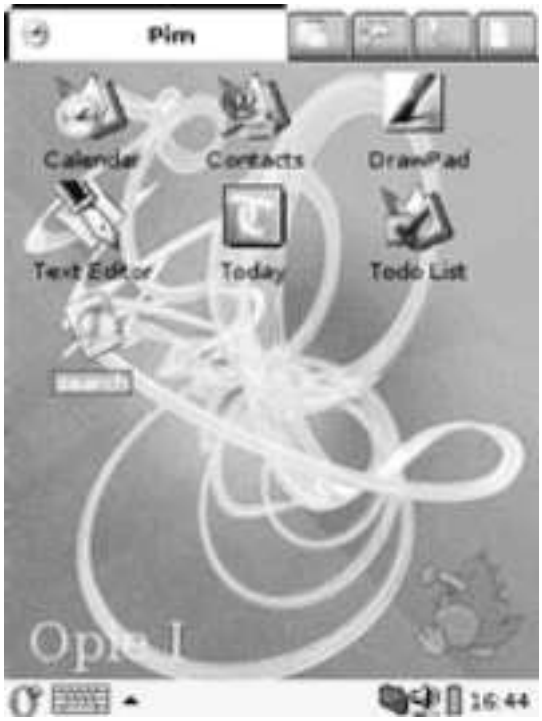

Figura 4.7: Opie en ipaq

internet, juegos y utilerías. Las siguientes son algunas características de Qt, por las que fué seleccionado como entorno gráfico de trabajo a utilizar:

- Applicaciones: Qtopia está basado en Qt, el cual tiene una comunidad de 150,000 desarrolladores. Sus continuas innovaciones de aplicaciones tanto en software libre y comercial han resultado en alrededor de 1,000 aplicaciones de Qtopia hacia el año 2003.

- Qt/Integrado: Qtopia está construído con Qt/embedded, el puerto integrado de Linux del entorno de aplicaciones multiplataforma C++, Qt. El código fuente de Qtopia y Qt/Embedded se ofrecen, y no hay licencia de restricción de uso, así que los fabricantes de hardware tienen la libertad de innovar y crear dispositivos originales para el mercado.

- Qt/Integrado cuenta con herramientas de rápido desarrollo.

- Qt/Embedded Virtual Framebuffer: El framebuffer(marco-búfer) virtual permite que los programas diseñados para Qt/Integrado puedan ser programados en la computadora de escritorio sin tener que cambiar entre consolas y X11. El marco-búfer virtual en sí esta incluído en la versión 2.3.1 de Qt.

- Qt Designer: Qt Designer es una herramienta visual que crea diseños e implementa la interfaz de usuario de una manera muy simple. Qt desginer está incluído también en la version 2.3.1.

- El sistema operativo: Qtopia corre sobre Linux integrado.

- Hardware para Qt: Qtopia corre sobre Linux integrado, el cual soporta las arquitecturas StrongARM, Xscale, PowerPC, x86, MIPS, y Dragonball.

- Foot Print de Qt: Qt/Integrado utiliza 3800Kb, Qtopia 5000Kb y el navegador Opera 2600 Kb, un total de 11 Mb.

Esta figura 4.8 muestra el stack de Qtopia en el dispositivo y la figura 4.9 la arquitectura de la PDA Sharp Zaurus modelo SL-5500 basada en este ambiente gráfico.

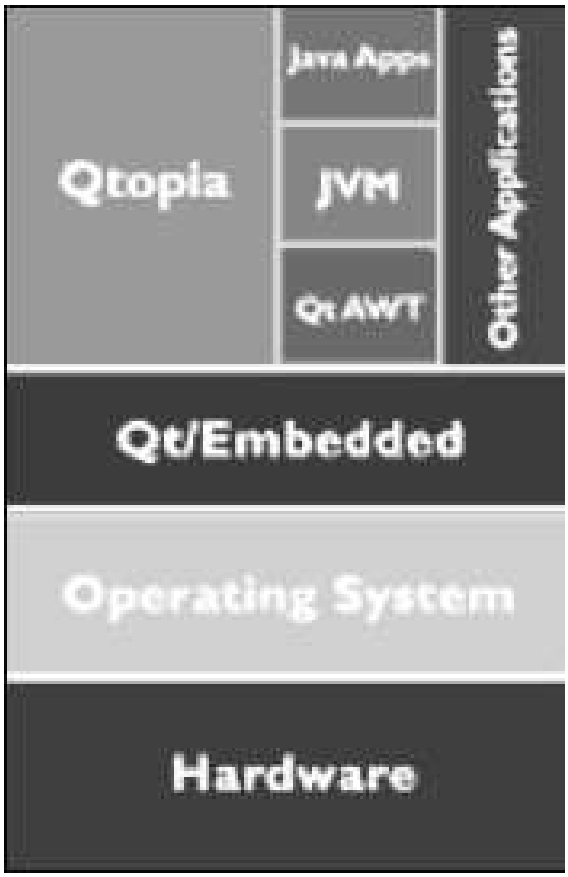

Figura 4.8: Stack Qt.

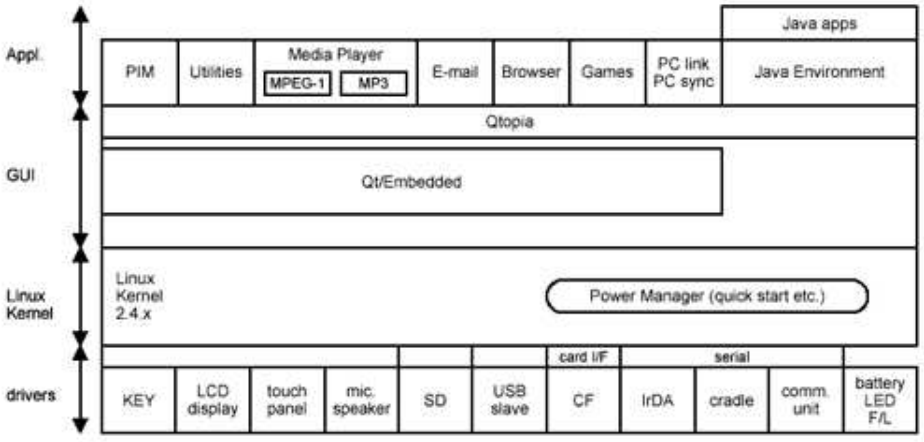

Figura 4.9: Arquitectura Sharp Zaurus SL-5500

Capítulo 5

Herramientas de Desarrollo.

5.1. Que es Open PDA ?

La plataforma OpenPDA es una solución completa de software para teléfonos inteligentes y asistentes personales digitales PDAs. OpenPDA resalta los innovadores diseños de hardware con el mejor software disponible para productividad, entretenimiento y sistemas de información personal. Desde juegos en el PDA hasta transacciones seguras para negocios, la plataforma OpenPDA se puede crear y configurar para satisfacer cualquier necesidad de los consumidores. La figura 5.1 muestra una ilustración de ésta plataforma, y se describe a continuación:

Figura 5.1: Open PDA

- Applications - Aplicaciones soportando Navegador HTML 4.0, CSS, XML, WML, ECMA, DOM, Redes y Unicode
- Java Runtime - Personal Java y J2ME
- GUI Framework - Entorno de Trabajo Gráfico de Qtopia. Un entorno de trabajo con interfaz gráfica de usuario completos para los dispositivos tipo hand held.
- Embedded Linux OS - Sistema Operativo con Linux Integrado. Toda la funcionalidad de un sistema operativo comercial pero sin el costo.
- Target Hardware - Hardware enfocado para ARM, ARM X-Scale, MIPS, SuperH and X86

5.2. Qt

Figura 5.2: Qt C++

Qt es un entorno de trabajo de C++, multiplataforma, que permite a los desarrolladores escribir una aplicación que correrá nativamente en Windows, Linux/Unix,Mac OS X, y Linux integrado con una simple recompilación. Es elegante, intuitivo y completamente orientado a objetos. Mayor información se puede obtener de la página trolltech `www.trolltech.com`. La figura 5.3 muestra el stack de capas de un equipo corriendo aplicaciones de Qt, y se muestra que al correr nativamente, no existen capas de emulación ni máquinas virtuales.

Para desarrollar en Qt solo es necesario aprender una sola API para escribir aplicaciones que correrán casi en cualquier lado. Qt tiene un set completo de widgets estándar, y permite escribir controles personalizados. Encapsula cuatro diferentes APIs de plataforma-específica,

Figura 5.3: Qt Stack

y los APIs para manejo de archivos, redes, manejo de procesos, cadenas (threading), acceso a base de datos, etc.

- Conexión a Bases de Datos

La versión 3 de Qt ofrece una API para escribir aplicaciones de bases de datos que es independiente tanto del sistema operativo subyacente y la base de datos seleccionada. Escrita en SQL, esta API aísla de las tares específicas de la base de datos.

La versión GPL de Qt tiene drivers nativos para MySQL, PostgreSQL y ODBC. La versión comercial tiene además drivers para Oracle, Sybase, y DB2.

- Sincronización Automática

Qt 3 tiene una nueva librería de controles conocedores de bases de datos que proveen sincronización automática entre la Interfaz Gráfica de Usuario y la base de datos. Estas clases incluyen widget de un navegador de datos basado en formas (para navegación y edición) y un visor de datos basado en formas (povee de formas de solo-lectura). Este entorno de trabajo puede ser extendido mediante el uso de editores de campos personalizados, permitiendo por ejemplo, a una tabla de datos el uso de widgets personalizados para edición en el lugar. También se puede usar el mecanismo de Qt `signal/slot` para extender y y customizar estos widgets, y para incluir validaciones propias de datos.

Existe una clase en la versión 3 de Qt llamada QSqlCursor , para aquellos que no gusta programar SQL puro, la cual envuelve instrucciones de `insert, update y delete` en una interfaz de C++ amigable.

Figura 5.4: Java logo

Finalmente , el módulo de base de datos de Qt es totalmente integrado con Qt Designer para un rápido diseño WYSIWYG de GUIs de aplicaciones de bases de datos.

5.3. Jeode edición PDA

Algunas PDAs (como la Sharp-Zaurus series SL-5XXX) tienen implementado el entorno de ejecución de Java *Jeode* . Mediante la implementación de Java, una plataforma estándar y abierta, se pueden crear aplicaciones usando las herramientas de desarrollo existentes para Java. La versión Personal Java está diseñada específicamente para dispositivos móviles de alto nivel integrados. Como PersonalJava esta rediseñado para dispositivos integrados basándose en el estandar de Java para la PC, permite a los desarrolladores utilizar muchos de los recursos de Java diseñados para ésta. Como ejemplo en la PDA de Sharp modelo Zaurus SL-6000 se puede correr la configuración de dispositivos conectados a la plataforma Java Micro Edición (J2ME). El perfil personal de J2ME es el sucesor de Personal Java. Mientras la aplicación Java es diseñada y desarrollada para mantener ciertas reglas, los desarrolladores pueden mantener cierta compatibilidad entre PersonalJava y el perfil personal de J2ME.

A diferencia de otros proveedores de la Máquina Virtual Java (JVM) que han implementado capas propietarias de software para gráficos, el paquete de herramientas de ventanas abstractas (AWT - Abstract Windowing Toolkit) de Jeode ha sido implementado para manejar el sistema nativo de ventanas de cada plataforma. Este acercamiento arquitectónico preserva el ver y sentir *look-and-feel* familiar de ese entorno; habilita el uso de fuentes plataforma-específicos y soporte de lenguajes (como métodos editores de entrada para Japonés, Koreano, etc.), y asegura integración con teclados virtuales. Esmertec ha implementado además una completa implementación del protocolo de la Interfaz Nativa de Java (JNI) que permite a

Figura 5.5: Esmertec

los desarrolladores el soportar funcionalidad plataforma-expecífica (ej. Lectores de códigos de barra, etc.) a través de clases Java.

Para soportar aún más las necesidades de la comunidad de las PDAs, Esmertec ha hecho mejoras significantes en el tiempo de arranque del motor de arranque de Jeode EVM por medio de la implementación de librerías de clases pre-cargadas.

Esmertec, figura 5.5 ofrece el primer entorno JVM en el mercado que provee soporte de plugins para correr applets de Java dentro del navegador de internet - Pocket Internet Explorer. Jeode además soporta otros navegadores populares para PDAs incluyendo Espial Escape, Netclue Clue, Opera 5 para Linux y Tenik WorldTALK.

5.3.1. Jeode en la PDA Zaurus

Una gran PDA es más que un buen hardware. Un factor clave para el éxito ó fracaso de un dispositivo son los componentes en el stack del software y que también trabajen estos en conjunto. La figura 5.6 ilustra el stack del software de una PDA en donde Joede encaja en la mezcla.

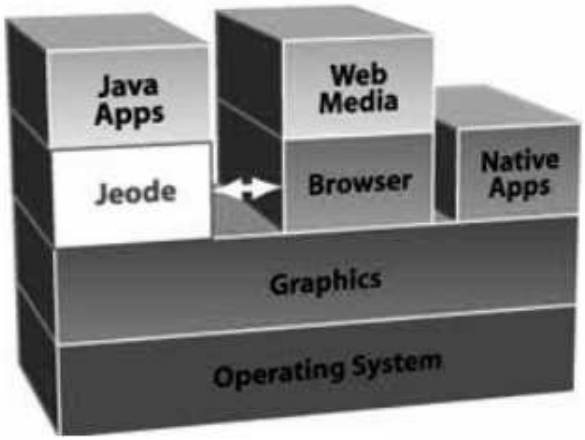

Figura 5.6: Jeode

5.3.2. PersonalJava

El entorno de Personal Java (PJAE) está basado en la API de PersonalJava, la cual difiere de J2SDK y JRE es su especificación. Aunque PersonalJava este creado con la API de Java como base, PJAE esta organizado por los subconjuntos específicos del dispositivo. La Máquina Virtual PersonalJava es entonces una versión compactada de la familiar Máquina Virtual Java (JVM), para poder encajar en la limitada capacidad de memoria de éstos dispositivos portables.

PJAE incluye librerías de clases que soportan tanto a la Máquina Virtual Personal Java como a la API de PersonalJava. Aunque cada instalación de PJAE esta hecha a la medida para cada dispositivo en específico, la interoperabilidad de aplicaciones y/o applets dentro de equipos con PJAE está garantizada siempre y cuando el código soporte la API. Pero si se incluye esta API, las librerías específicas del dispositivo, applets y aplicaciones con métodos específicos, la interoperabilidad no está garantizada.

5.3.3. Perfil CDC de J2ME suscesor de PersonalJava

PersonalJava fué descontinuado e integrado como un perfil dentro de J2ME, el perfil Connected Device Configuration (CDC) . Éste es un entorno de trabajo con bases estándares

para construir y distribuir aplicaciones para una variedad de dispositivos móviles. CDC está diseñado para escenarios de productos con recursos comprometidos, típicamente 2 Mb de RAM y 2.5 MB de ROM para el entorno de aplicaciones Java. CDC está además basado en compatiblidad con las APIs estandares de J2SE. La familia CDC incluye los siguientes perfiles Foundation Profile, Personal Basis Profile y Personal Profile.

J2ME está organizado en distintos perfiles dentro de los que se encuentran Connected Device Configuration (CDC) y Connected Limited Device Configuration (CLDC), para soportar dispositivos para el consumidor de bajo costo, PDAs de poco alcanze y teléfonos portátiles.

5.3.4. Controladores JDBC

Los controladores JDBC[1] proveen clases que implementan la interfaz JDBC. JDBC define cuatro tipos de controladores, y cada una tiene una mezcla variada de ejecutables sistema operativo dependientes y código de Java como se indica a continuación:

Bridge (tipo 1) JDBC-ODBC tecnología de puente. Una mezcla de código Java y sistema operativo-dependiente con el que JDBC accesa a base de datos mediante ODBC.

Native (tipo 2) Ejecutables sistema operativo dependientes y Java. Una mezcla de código Java y sistema operativo-dependiente con el que JDBC accesa a la base de datos a través de código propietario de proveedor.

Network (tipo 3) Middleware de Red y Java. Una mezcla de código Java y código basado en un servidor sistema operativo-dependiente donde el JDBC accesa a la base de datos a través del middleware.

Thin (tipo 4) 100 % Java. Un controlador Java puro que no requiere de código sistema operativo-dependiente, lo que hace al controlador 100 % portable.

[1] JDBC Java Database Connection

5.3.4.1. Sintaxis URL para Base de datos de JDBC

La siguiente es la sintaxis general de una URL de base de datos JDBC:

```
URL:jdbc:subprotocol:subname
```

jdbc Protocolo de comunicación

subprotocol Nombre del controlador

subname Referencia controlador-específica a la base de datos. El elmento subname en esta sintaxis se puede tornar algo complicada, y su formato depende del controlador que se utilize.

El cuadro 5.1 describe el controlador para MySQL detallando el archivo, la clase que debe cargarse y la sintaxis del URL de la base de datos para el controlador, indicando también las propiedades soportadas.

Descargar de:	http://www.mysql.com
Archivo:	mysql-connector-java-2.0.14-bin.jar
Controlador:	com.mysql.jdbc.Driver
Sintaxis URL:	jdbc:mysql://[nombrehost][:puerto]/
	[nombre bd][?param1=valor1][&param2=valor2]

Propiedades	**soportadas:**
User	Password
autoReconnect	maxReconnects
initialTimeout	maxRows
UseUnicode	characterEncoding
relaxAutocommit	ultraDevHack
capitalizeTypeNames	profileSql

Cuadro 5.1: Detalle controlador JDBC para MySQL

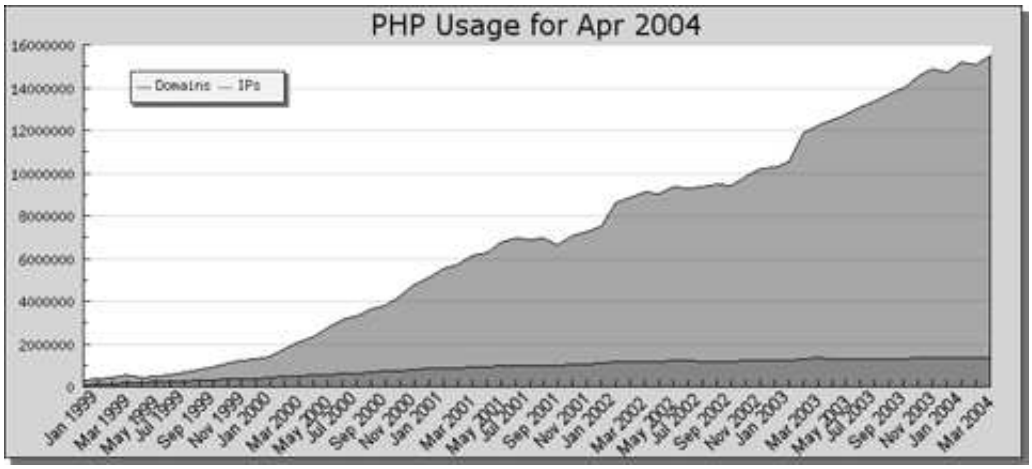

Figura 5.7: Uso estimado de PHP en abril del 2004

5.4. Php sobre Apache

5.4.1. ¿Qué es PHP?

PHP es un proyecto de la fundación de software Apache `www.apache.org`. PHP es un lenguaje de scripts muy usado de propósito general que está especialmente hecho para desarrollo en el Web y que puede ser integrado hacia HTML.

PHP viene incluído en el paquete de instalación del servidor web Apache disponible para las PDAs con Linux integrado. En el punto 6.3.4.1 se explica como se pude instalar y de donde se puede descargar. El servidor web Apache con PHP NO viene incluido en el ROM de las PDAs con Linux.

La figura 5.7 muestra la gráfica del aumento que ha tenido PHP en servidores web.

5.4.2. ¿ Qué se puede hacer con PHP ?

PHP puede hacer cualquier cosa que se pueda hacer con un script CGI, como procesar la información de formularios, generar páginas con contenidos dinámicos, ó enviar y recibir cookies. Y esto no es todo, se puede hacer mucho más. Existen tres campos en los que se usan scripts escritos en PHP.

1. Scripts del lado del servidor. Este es el campo más tradicional y el principal foco de tra-

bajo. Se necesitan tres cosas para que esto funcione. El intérprete PHP (CGI ó móulo), un servidor web y un navegador. Es necesario correr el servidor web con PHP instalado. El resultado del programa PHP se puede obtener a través del navegador, conectándose con el servidor web.

2. Scripts en la línea de comandos. Se puede crear un script PHP y correrlo sin ningún servidor web o navegador. Solamente se necesita el intérprete PHP para usarlo de esta manera. Este tipo de uso es ideal para scripts ejecutados regularmente desde cron (en Unix ó Linux). Estos scripts también pueden ser usados para tareas simples de procesamiento de texto.

3. Escribir aplicaciones de interfaz gráfica. Probablemente PHP no sea el lenguaje más apropiado para escribir aplicaciones gráficas, pero si se conoce bien PHP, y se quisieran utilizar algunas características avanzadas en programas clientes, se puede utilizar PHP-GTK para escribir dichos programas. También es posible escribir aplicaciones independientes de una plataforma. PHP-GTK es una extensión de PHP, no disponible en la distribución principal, ni tampoco en la distribución para PDAs con Linux Integrado.

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux integrado y Linux, muchas variantes Unix (incluyendo HP-UX, Solaris y OpenBSD), Mac OS X, RISC OS y seguramente algunos más. PHP soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros. PHP tiene módulos disponibles para la mayoría de los servidores, para aquellos otros que soporten el estándar CGI, PHP puede usarse como procesador CGI.

De modo que, con PHP se tiene la libertad de elegir el sistema operativo y el servidor de que mas guste. También se tiene la posibilidad de usar programación procedimental ó programación orientada a objetos. Aunque no todas las características estándares de la programación orientada a objetos están implementadas en la versión actual de PHP, muchas librerías y aplicaciones grandes están escritas íntegramente usando programación orientada a objetos.

Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz vía web para una base de datos es una tarea simple con PHP. Las siguientes bases de datos están soportadas actualmente[2]:

Adabas D, Ingres, Oracle (OCI7 and OCI8), dBase, InterBase, Ovrimos, Empress, FrontBase, PostgreSQL, FilePro (read-only), mSQL, Solid Hyperwave, Direct MS-SQL, Sybase, IBM DB2, MySQL, Velocis, Informix, ODBC, Unix dbm.

También se cuenta con una extensión DBX de abstracción de base de datos que permite usar de forma transparente cualquier base de datos soportada por la extensión. Adicionalmente, PHP soporta ODBC (el Estándar Abierto de Conexión con Bases de Datos), asi que se puede conectar a cualquier base de datos que soporte tal estándar.

PHP también cuenta con soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros. Se pueden crear sockets puros. PHP soporta WDDX para el intercambio de datos entre lenguajes de programación en web. Y hablando de interconexión, PHP puede utilizar objetos Java de forma transparente como objetos PHP y la extensión de CORBA puede ser utilizada para acceder a objetos remotos.

PHP tiene unas características muy útiles para el procesamiento de texto, desde expresiones regulares POSIX extendidas ó tipo Perl hasta procesadores de documentos XML. Para procesar y acceder a documentos XML, soporta los estándares SAX y DOM. Puede utilizar la extensión XSLT para transformar documentos XML.

Si se usa PHP en el campo del comercio electrónico, se encontrarán muy útiles las funciones Cybercash, CyberMUT, VeriSign Payflow Pro y CCVS para los programas de pago.

Para terminar, cuenta con muchas otras extensiones muy interesantes, las funciones del motor de búsquedas mnoGoSearch, funciones para pasarelas de IRC, utilidades de compresión (gzip, bz2), conversión de calendarios, traducción, etc.

[2]La versión de PHP para Linux integrado soporta únicamente MySQL, hasta el momento.

5.5. Aplicaciones comerciales de Rápido Desarrollo (RADS) para Java

5.5.1. Metrowerks CodeWarrior Development Studio para Linux Integrado

Metrowerks ha soportado a la comunidad Linux por años, como el proveedor de la herramienta de desarrollo comercial mejor vendida para Linux. Combinada con más de 10 años de experiencia en herramientas de desarrollo integradas Metrowerks, potenciando a desarrolladores con una suite completa de herramientas de desarrollo para aumentar la productividad y mejorar la calidad de las aplicaciones de software.

La edición de `CodeWarrior Development Studio Embedded Linux` simplifica el desarrollo de aplicaciones diseñadas para correr en sistemas operativos con Linux integrado como nunca antes. Codewarrior cuenta con un ambiente desarrollo (IDE) altamente integrado, visual e intuitivo, tal que los desarrolladores para sistemas Linux pueden evitar las limitantes de consumo de tiempo que genera el estilo de desarrollo de línea de comandos de GNU. ¿ El resultado? Reducción de tiempo de desarrollo, incremento en la productividad y mejoramiento de la calidad.

La edición para Linux Integrado permite a los desarrolladores de software el crear, compilar, vincular(link), depurar (debug) y distribuir aplicaciones en la plataforma final apuntada, sin tener que salirse del entrono de CodeWarrior. Las caracteríticas de la edición para Linux Integrado son una amigable Interfaz de Usuario que incluye un potente manejador de proyectos, duprador, editor y motor de búsqueda que ayudan a aumentar la velocidad en el desarrollo de aplicaciones.

Otro característica es que, a diferencia de las herramientas hospedadas en Linux, Codewarrior Studio para Linux Integrado es hospedado en sistemas basados en Windows.

5.5.2. Metrowerks CodeWarrior Wireless Studio para Java

CodeWarrior Wireless Studio 7 es lo último (en el año 2003) en herramientas de desarrollo para crear aplicaciones de clase mundial que apuntan a las plataformas J2 Edición eStandar (J2SE), Java 2 Micro Edición (J2ME) y PersonalJava para teléfonos inteligentes, PDAs, cajas set-top y otros dispositivos con Java-habilitado. Mayor información de CodeWarrior puede obtener en `http://www.metrowerks.com`.

CodeWarrior Wireless Studio provee de un Entorno Integrado de Desarrollo (IDE) abierto y adaptable, un manejador de proyectos y un constructor de sistemas de primera línea, navegador de clases y un sistema de navegación para código, editor de texto con terminación de código, depurador(debugger), emulador, soporte para colaboración en equipo y desarrollo WYSIWYG arrastra y suelta (drag and drop).

Siendo la seguna compañía que licencio la tecnología Java de Sun, y la primera en soportar comprehensivamente J2ME y PersonalJava, Metrowerks esta enfocada en proveer las herramientas líderes de desarrollo y los servicios para hacer existoso a los desarrollores de equipos móviles.

Desarrollar aplicaciones para la tecnología Java incluída en las PDAs ó los teléfonos inteligentes es muy diferente que desarrollar para una aplicación empresarial de la plataforma J2EE, el desarrollador requiere de un conjunto de herramientas distintas. Es por eso que CodeWarrior Wireless Studio fue diseñado específicamente para proveer a los desarrolladores con la herramienta para desarrollo en J2ME y PersonalJava.

5.5.3. Borland JBuilder

JBuilder es un RAD comercial que acelera el desarrollo en Java, siendo este un entorno de desarrollo multiplataforma ya que corre en diversos sistemas operativos, entre los cuales se encuentra Linux. Borland JBuilder ofrece desarrollo en el profile J2ME para dispositivos móviles como PDAs, además de un amplio espectro de desarrollo en Java incluyendo JavaBeans, Java Server Pages, Web Services, XML, aplicaciones móviles, Swing, aplicaciones de bases de datos, entre otros.

5.6. Herramientas front-end de almacenamiento y acceso a bases de datos

Una opción primaria de almacenamiento consiste en utilizar el estandard XML. Alternativamente existen otras utilerías como Portabase, Jcard y TckRekall para almacenamiento de datos en archivos planos.

5.6.1. PortaBase

Portabase es un pequeño programa de base de datos para crear, explorar y editar tablas personalizadas de datos. Originalmente diseñada para la PDA Sharp Zaurus, ahora también puede correr en Linux/Unix y aplicaciones de Windows. Los usos típicos son inventarios, listas de compras, TODO lists, manejo de contraseñas, Álbunes de fotos, etc. PortaBase es software libre bajo licencia de GNU General Public License (GPL).

5.6.1.1. Características

- Columnas tipo String, Integer, Decimal, Boolean, Note (multi-line text), Date, Time, Calculation, Sequence, Image, y Enum.
- Agregar, editar y eliminar renglones de datos
- Filtrado de renglones desplegados usando un conjunto de condiciones
- Ordenar renglones en base a una combinación de columnas, cada una en orden ascendente y descendente
- Botones de navegación para las páginas, con un numero de renglones por página customizado.
- Agregar, eliminar, rearreglar y renombrar columnas en cualquier momento.
- Especificar valores pro default de columnas.

- Vista sumaria de estadísticas por columnas(total, promedio, min, max, etc.)
- Especificar valores por defecto para columnas
- Importación y exportación de datos de CSV y archivos XML.
- Encriptamiento de archivos de datos
- Soporte Unicode
- Selección de cualquier fuente disponible para utilizarse en la aplicación
- Colores de segundo plano alternativos especificados por el usuario
- Widget de calculadora simple para entrada de datos numéricos
- Barra de herramienta y menú configurables

5.6.2. JCards

5.6.2.1. Caracteristicas

Jcards es una aplicación Java gratuita diseñada para manejar y almacenar datos en cualquier formato que el usuario defina. En lugar de tener multiples aplicaciones para manejar contraseñas, base de datos musicales, listas de compras, etc, Jcards permite al usuario definir una base de datos que contenga los campos que necesiten y customizarlo de varias maneras. Esta se realiza a través de una interfaz fácil de usar que no requiere de conocimiento de SQL; de hecho Jcards en un front-end en una base de datos de archivos planos.

Jcards fue inicialmente diseñada para correr en la PDA Sharp Zaurus SL-5x00, pero también correrá en cualquier PocketPC PDA que tenga JVM instalado. El paquete de instalación llama `jcards_5.0-1_arm.ipk` se encuentra en el sitio jcards `http://www.nsydenham.net/java/JCards/`

5.6.3. TkcRekall, una base de datos comercial

tkcRecall es un puerto de la vista frontal ***front-end*** para la Sharp Zaurus que viene de la base de datos Rekall para la PC de escritorio. Cualquier cosa que se puede hacer en la PC de escritorio se puede hacer en la Zaurus, limitándose solo por la capacidad de la misma. Existe una versión recortada de tkcRecall que es tkcRecall lite, la cual restringe el uso de archivos en formato XBase(dbf) exclusivamente.

Esta base de datos NO está licenciada bajo GPL.

5.6.3.1. Características

- Acceso a bases de datos MySQL y PostgreSQL en la red, y a MySQL directamente en la Zaurus
- Acceso a bases de datos en la Zaurus en formato Xbase. Permite la creación de base de datos simples almacenadas en la Zaurus
- Diseño, vista y actualización de tablas. tkcRecall incorpora una herramienta para diseño de tablas que permite ver y actualizar los datos almacenados en las tablas
- Diseño y presentación de formas. Soporta formas y fomas/subformas. Se pueden diseñar y ejecutar formas en la Zaurus, con un rango de controles desde campos de texto hast imágenes y texto con formato. Las formas puede recuperar datos directamente de las tablas, a través de texto-libre de consultas de SQL ó de consultas creadas con el diseñador gráfico de consultas(*querys*).
- Diseño y uso de consultas (*querys*). tkcRecall incluye un diseñador de consultas el cual puede ser usado para crear y ver consultas (*querys*) de múltiples tablas. Dependiendo del servidor permite la funcionalidad de hacer *joins*, *group by* y *having*, *where* y *ordering*. Estas *consultas* se pueden utilizar en las formas.

- Importación y exportación. Los datos pueden se movidos en archivos y tablas usando el copiador de tkcRecall. El copiador puede mover datos de uno de varios recursos (archivos, tablas y consultas SQL) hacia otro de varios destinos (archivos, tablas y texto con formato XML). Por ejemplo se pueden copiar datos desde y hacia una base de datos en la red RDBMS y una base datos Xbase dentro de la Zaurus.

- Secuencia de comandos. tkcRecall esta programada usando el lenguaje Piton, e incorpora el interprete de Piton 2.1.1- Se pueden escribir secuencias de comandos de Piton para automatizar y extender las funcionalidades de la base de datos utilizando, y se pueden acceder los módulos que son portados a la Zaurus.

- Reportes. Esta versión de tkcRecall no incluye diseño de reportes. La impresión no está soportada.

Una base de datos mínima viene incluida en el paquete de instalación. Debe ser instalado en RAM, y debe crear archivos en `/home/tmp/DB/Text`. El archivo de la base de datos tkcReKall es `/home/tmp/Text/Text.rlk`. Este usa la librería Xbase, asi que puede correr completamente en la Zaurus. Para empezar hay que abrir esta base de datos, seleccionar la pestaña de *formas*, abrir la opción *Files* y hacer un doble clic en *MainForm*.

Respecto del runtime

No es necesario comprar un runtime para cada aplicación que se disribuye, solo se necesita comprar una y se puede distribuir el run-time con las aplicaciones gratuitamente, sin importar si se adquirió la versión *estandar* ó la *lite*. El costo del runtime es de aproximadamente $100.00 USD

5.7. Bases de datos para la Zaurus

De pequeñas dimenciones pero con grandes características propias de un DMBS[3] existen actualmente, para la Zaurus, dos poderosas bases de datos bajo licencia GPL: **PostgreSQL** y

[3]Manejador de Bases de Datos, en Inglés Data Base Management System

MySQL; así como algunas comerciales como **iAnywhere** de Sybase.

5.7.1. Mysql

Mysql, una de las bases de datos con licencia GPL más conocidas en ambientes Linux la cual cuenta ahora con una versión para la Zaurus. ¿ Por qué alguien querría una base de datos de alto poder como Mysql en una PDA? La principal razón es la poderosa combinación del servidor Apache+Php con Mysql, lo cual abre un largo campo de aplicaciones para ser usadas en la Zaurus. Estas posibilidades incluyen cosas como programas contables, inventarios, agendas, etc., todos accesados mediante el navegador Opera el cual viene incluído en el ROM de la Zaurus. Este paquete de Mysql para Zaurus fue obtenido de la distribución debian y editado para correr en la Zaurus.

5.7.2. Postgresql

Desde el incio del proyecto POSTGRES en 1986 en la Universidad de California, los desarrolladores de PostgreSQL han enfatizado tanto la implementación de las ideas más recientes en el diseño y estándares de bases de datos, como el permitir a los usuarios el realizar tareas complejas dentro de la misma base de datos. Muchos llaman a PostgreSQL la “base de datos administradora de base de datos”, por que ésta maximiza el uso de SQL y ofrece controles de integridad de datos, procedimientos en la base de datos, y teoría relacional para hacer el trabajo dentro de PostgreSQL, en vez de utilizar un código middleware.

La arquitectura de PostgreSQL y sus características *Objeto-Relacionales* hacen a las extensiones modulares de la base de datos funcionalmente sencillas y accesibles. PostgreSQL es extensible, flexible e intuitivo. La versión 7.4.2 de PostgreSQL para la Zaurus se puede descargar del sitio:

`http://www.paipai.org/~humorum/pukiwiki.php?ZaurusPostgreSQL`

5.7.3. Sybase iAnywhere

La base de datos iAnywhere de Sybase, quizá inadvertidamente, ha provocado un impacto mayor en la PDAs basadas en procesadores ARM con Linux integrado a través del vínculo con el gigante de la electronica Sharp Electronics. Ésta relación muestra a la subsidiaria móvil de Sybase trabajando con el gigante de la electrónica Japonesa para alentar a los de desarrolladores a construir aplicaciones basadas en Java para las PDAs con Linux integrado como la Zaurus SL-5500, utilizando la plataforma ”iAnywhere m-Business” que en Español se puede nombrar como Negocios Móviles iAnywhere.

Esta base de datos NO es software libre por lo que es necesario adquirir la licencia para poder utilizarla. El sitio oficial de Sybase es `http://www.sybase.com`

Capítulo 6

VII. Metodología

Existen sólamente 10 tipos de personas, los que entienden binarios y los que no.

6.1. Cuadros sinópticos de la metodología

El resumen de la metodología se divide en seis cuadros sinópticos que indican los pasos que se consideran indispensables para cumplir con la configuración mínima para desarrollo de la PDA y la PC de escritorio; así como los pasos mínimos para desarrollo en los diferentes lenguajes, indicando además, el tiempo que se demoran aproximadamente. El cuadro 6.1 contiene el resumen de los pasos necesarios para la configuración básica de la PDA y el cuadro 6.2 para la configuración básica de la PC de escritorio. El cuadro 6.3 contiene el resumen de los pasos relacionados con el desarrollo de aplicaciones en Qt, el cuadro 6.4 con el desarrollo de aplicaciones en PersonalJava, el cuadro 6.5 con el desarrollo de aplicaciones en PHP. Finalmente el cuadro 6.6 indica los pasos necesarios para la internacionalización de una aplicación.

La siglas de identificación para cada paso se definen de la siguiente manera:

iMin Indispensable para la configuración mínima de la PDA y la PC de escritorio.

iQt Indispensable para el desarrollo de aplicaciones en Qt.

iPJ Indispensabe para el desarrollo de aplicaciones en PersonaJava.

iPHP Indispensable para el desarrollo de aplicaciones en PHP.

i18n Indispensable para internacionalización de una aplicación.

	Configuración básica PDA	Tiempo	Indispensable
1.	Actualización del ROM	1 Hr.	iMin[1]
2.	Terminal	10 min	iMin
3.	Ligas simbólicas a librerías	10 min	iPHP
4.	Comandos escenciales Linux /bin	10 min	iMin
5.	Openssh	20 min	iMin
6.	VNC Server	15 min	iMin
7.	Librerías	10 min	iMin
8.	Instalación de MySQL	20 min	iMin
9.	Arrancando MySQL	3 min	iMin
10.	Instalación del controlador JDBC	10 min	iPJ
11.	Instalación de Apache con PHP	20 min	iPHP
12.	Arranque y reinicio servidor Apache	3 min	iPHP
13.	Configuración del puerto IrDa	10 min	iMin
14.	Creación de memoria swap	10 min	iMin
15.	Respaldo	5 min	iMin
16.	Restuaración	5 min	iMin

Cuadro 6.1: Resumen configuración básica de la PDA

	Config. básica PC escritorio	Tiempo	Indispensable
17.	Instalación de MySQL	15 min	iMin
	Configuración básica para Qt		
18.	Instalación SDK de Qt	20 min	iQt
19.	Instalación compilador multi-plataforma (C++)	20 min	iQt
20.	Variables de entorno del compilador	3 min	iQt
21.	Configuración tmake.conf	3 min	iQt
22.	Qtdesigner para aplicaciones ARM	3 min	iQt
	Configuración básica para Java		
23.	Instalación herramientas básicas	20 min	iPJ
24.	Instalación plugin de Java	3 min	iPJ
25.	Instalación del controloador para Mysql	5 min	iPJ
	Configuración básica para PHP		
26.	Instalación de PHP	10 min	iPHP
27.	Configuración PHP	3 min	iPHP
28.	Integración de PHP con Apache	3 min	iPHP
29.	Reinicio del servidor web Apache	3 min	iPHP

Cuadro 6.2: Resumen configuración básica de la PC de escritorio

	Desarrollo en Qt	Tiempo	Indispensable
30.	Código fuente holamundo.cpp	10 min	iQt
31.	Definición de variables de entorno	5 min	iQt
32.	Creación del archivo de proyecto holamundo.pro	5 min	iQt
33.	Compilación holamundo.cpp	3 min	iQt
	I*nstalación de aplicaciones Qt*		
34.	Creación de la estructura de directorios para el paquet ipk	10 min	iPJ
35.	Transfiriendo el paquete ipk a la Zaurus	5 min	iPJ
36.	Instalando el paquete ipk en la Zaurus	3 min	iPJ

Cuadro 6.3: Resumen pasos para desarrollo en Qt

	Desarrollo en PersonalJava	Tiempo	Indispensable
37.	Lista de paquetes PersonalJava	5 min	iPJ
38.	Paquetes opcionales soportados por la Zaurus	5 min	iPJ
39.	JDBC un conducto a MySQL	10 min	iPJ
40.	Primer programa Hola PersonalJava	5 min	iPJ
41.	Creación de un Applet holaPJavaApplet.class	10 min	iPJ
42.	Interactuando con MySQL en PersonalJava	15 min	iPJ
43.	Programa en Java para impresión vía IrDa	25 min	iPJ
	Instalación de aplicaciones Personal Java		
44.	Creación de directorios para el paquete ipk	10 min	iPJ
45.	Contenido de los archivos requeridos para crear el paquete ipk	5 min	iPJ
46.	Creación del paquete ipk con la herramienta ipkg-build	3 min	iPJ

Cuadro 6.4: Resumen pasos para desarrollo en PersonalJava

	Desarrollo en PHP	Tiempo	Indispensable
47.	Primer programa PHP info	5 min	iPHP
48.	Combinando código HTML y PHP en un mismo archivo	5 min	iPHP
49.	Interactuando con MySQL usando PHP	10 min	iPHP
50.	Programa de impresión de código de barras en PHP	5 min	iPHP
51.	Instalación de aplicaciones PHP en la Zaurus	10 min	iPHP

Cuadro 6.5: Resumen pasos para desarrollo en PHP

	Internacionalización de una aplicación	Tiempo	Indispensable
52.	Configuración de la variable Language en Linux integrado	5 min	i18n
53.	Instalación de fonts para otro idioma	5 min	i18n
54.	Traducción de aplicaciones	20 min	i18n
55.	Localización de mensajes en PersonalJava	10 min	i18n
56.	Localización de mensajes en PHP	10 min	i18n
57.	Localización de imágenes en PHP	10 min	i18n

Cuadro 6.6: Resumen pasos para internacionalización de una apliación

6.2. Si se cambia el modelo de PDA, ¿ funciona la metodología ?

En primer lugar, ésta metodología aplica para las PDAs con Linux integrado con procesadores StrongARM ó XScale, y el kernel versión 2.4 ó mayor. En segundo lugar ésta metodología se refiere al desarrollo de aplicaciones en tres diferentes plataformas ó lenguajes los cuales son C++, Java y PHP. Para el desarrollo en alguno de estos tres lenguajes se requiere que la PDA cumpla con alguno de los siguientes requerimientos de software para cada lenguaje en específico como se indica a continuación:

Entorno gráfico de trabajo: Qtopia ->para ejecutar aplicaciones desarrolladas en Qt.

Runtime: PersonalJava ->para ejecutar aplicaciones desarrolladas en Java.

Navegador ->para ejecutar aplicaciones desarrolladas en PHP.

Existen diversos entornos gráficos de trabajo, de los cuales, la PDA Zaurus utiliza el entorno gráfico de trabajo llamado Qtopia, así que para obtener compatibilidad al 100 % de ésta metodología con otro modelo de PDA se requiere que se utilize el mismo entorno gráfico (Qtopia), ya que para la programación en C++ se utilizó Qt, una herramienta de Trollltech,

que aunque funciona para varias plataformas, en el caso de las PDAs con Linux integrado está diseñado para el entorno gráfico Qtopia, así que la metodología para desarrollo en Qt se limita sólo a aquellas PDAs con el entrono gráfico Qtopia y otros compatibles como el caso del entorno gráfico Opie.

De no utilizarse Qtopia, la metodología funcionará parcialmente con Java y PHP bajo las siguientes premisas:

1. Para otros modelos de PDAs con Linux integrado que tienen incluído (ó que se pueda instalar) el runtime de PersonalJava ó J2ME, la portabilidad que ofrece Java en sí, permite que la metodología funcione en cuanto al desarrollo y ejecución de las aplicaciones, aunque algunos pasos como la creación de paquetes de instalación no sería el mismo debido a un posible cambio en la estructura de directorios y ubicación de archivos ejecutables.

2. Quizá el desarrollo de aplicaciones más portable entre distindos modelos de PDAs con Linux integrado se puede ver en PHP, ya que casi cualquier PDA con Linux integrado cuenta con un navegador lo cual permite la ejecución de scripts de PHP una vez instalado el servidor web Apache, del cual sólo se tiene el requerimiento para su instalación del tipo de procesador StrongARM ó XScale con el kernel 2.4 ó mayor, independiente del entorno gráfico de trabajo.

6.3. Configuración básica de la PDA Zaurus

Esta seccción muestra la secuencia lógica de instalación de las aplicaciones y servicios necesarios para configurar la PDA Zaurus para desarrollo.

6.3.1. Actualización del ROM para el modelo Zaurus SL-5000

La actualización del ROM versión 3.1 se encuentra dentro del archivo compactado **5500v31u.zip**, el cual se puede descargar de MyZaurus `http://www.myzaurus.com`, el sitio oficial de Sharp. Para descompactarlo se utiliza el programa `unzip`, ejemplo:

```
#unzip 5500v31u.zip
```

La versión original del ROM de éste modelo tiene algunas desventajas y no contiene herramientas (programas) de respaldo ni de restauración. Las aplicaciones PIM del modelo SL-5500 tienen algunas deficiencias en cuanto a la sincronización de la información hacia el desktop con el programa de correo electrónico Outlook; la aplicación TodoList esta muy recortada y la agenda no funciona muy bien.

Resulta entonces esencial actualizarle el ROM.

6.3.1.1. Actualización del ROM desde Linux

Para la actualización del ROM desde Linux se requiere de una tarjeta de memoria tipo CF (Compact Flash) ó una tarjeta tipo SD (Secure Digital), en la que se deben copiar los archivos `Ospack` y `Setup` que se ubican en la carpeta `/updaterom/` dentro del archivo comprimido `5500v31u.zip` mencionado en el punto 6.3.1. Los archivos deben ser copiados en el directorio `raíz(root)->/`. Ya con estos archivos copiados en el directorio raíz de una tarjeta de memoria CF (ó SD) se deben realizar los siguientes pasos:

1. Insertar la tarjeta de memoria CF (ó SD) card en la ranura correspondiente en la Zaurus.

2. Conectar el adaptador de corriente a la Zaurus y verificar que este conectado supliendo de corriente a la Zaurus. AVISO: El adaptador de corriente DEBE ser utilizado cuando se realice la actualización del ROM.

3. Abrir el compartimiento de la batería, pero no removerla.

4. Deteniendo y presionando la teclas [C] y [D] en el teclado de la Zaurus, oprimir el botón de FULL RESET (utilizando la plumita). El botón FULL RESET esta localizado en la parte inferior derecha del compartimiento de la batería. Una vez presionados estos tres artículos, un LED verde(con el icono simbólico de correo electrónico) y un LED naranja (con el símbolo de batería impreso) se encenderán. Cuando estos leds esten prendidos , el proceso de actualización del ROM estara en proceso.

   AVISO: No se debe abortar, ó desconectar la alimentación de corriente mientras los LEDs están encendidos. Abortar ó desconectar la corriente durante el proceso de actualización podría causar fallas ó daños a la Zaurus

- Despues de aproximadamente 3 minutos el LED verde se apagara. Al apagarse se necesita reinicializar el equipo presionando el botón FULL RESET utilizando la plumita.

  Nota: Después de la actualización del ROM (el LED se apaga), pero antes de ejecutar el FULL RESET, la Zaurus podría desplegar mensajes de texto deslizables. Se pueden ignorar y proceder a ejecutar el FULL RESET.

Por último se debe cambiar el switch a Operación Normal(Normal Operation) y colocar la tapa de la batería.

6.3.2. Aplicaciones consideradas esenciales

6.3.2.1. Terminal

Terminal de la consola de Linux. Esencial para ejecución de comandos, instalación de paquetes, servicios, etc.

El paquete ipk de instalación de la terminal está incluído en el archivo compactado **5500v31u.zip** de actualización del ROM 3.10 en el directorio \5500v31u\Applications. El archivo se llama qpe-terminal_1.5.0-3_arm.ipk. Para instalarlo es necesario:

1. Copiar el archivo `qpe-terminal_1.5.0-3_arm.ipk` a la Zaurus, en la tarjeta de memoria flash tipo SD ó CF.

2. Seleccionar en la pestaña de Herramientas y la opción `Add/Remove Software` como se indica en la figura 6.1.

3. Presionar el botón `Install packages` que se muestra en la figura 6.2. El programa de instalación detectará aquellos paquetes de instalación (ipkg) que no estén instalados. Dentro de estos se debe localizar el que dice `qpe-terminal_1.5.0-3_arm.ipk` y presionarlo con un tap (click con la pluma ó puntero)

4. El color del botón a la izquierda del nombre pasará de color rojo a azul. Esto es el indicador de que el programa ya está instalado. El icono de la terminal aparece por defecto en la pestaña `Applications`

Desde la terminal instalada se pueden instalar los `paquetes.ipk` en la línea de comandos utilizando la instrucción `ipkg`. La sintaxis de ésta instrucción se puede consultar el apéndice D. En la instalación de diversos paquetes es necesario entrar como **root** para obtener de los permisos de ejecución. La sintaxis de cómo entrar como **root** se incluye también en éste mismo apéndice D.

6.3.2.2. Ligas simbólicas a librerías

En ocaciones las aplicaciones creadas en Qt para la Zaurus requieren de la librería *libqtopia.so.1*, pero esta librería en la Zaurus se encuentra como *libqpe.so.1* así que es mandatorio si se van a crear aplicaciones en Qt, crear una liga simbólica a esta librería con los siguientes comandos dentro de la terminal de la Zaurus (como instarlar la terminal se indica en el punto 6.3.2.1):

```
su // para obtener permisos de root
cd /home/QtPalmtop/lib  // para accesar al dir de librerias
ln - s libqpe.so.1 libqtopia.so.1  // para crear la liga simbólica
```

Figura 6.1: Add/Remove Software

Figura 6.2: Install packages

6.3.2.3. Utilerías para ARM /bin

Existen comandos necesarios para que algunos scripts de configuración de diversas aplicaciones se ejecuten correctamente. Estos scripts de configuración ó de arranque de servicios necesitan ciertos programas que por defecto no vienen en la Zaurus. Se debe descargar de killefiz `www.killefiz.de/zaurus` el archivo compactado `armutils.tar`. Una vez descomprimido el archivo, se deben copiar los comandos a la Zaurus en el directorio:

```
/home/root/usr/bin.
```

En ocaciones es necesario crear algunas ligas simbólicas para compatiblidad con los nombres y versiones de los comandos. Por ejemplo, existe en este archivo `armutils.tar` un comando llamado `mawk`, pero algunos scripts (ej: `apachectl`) se refieren a este comando con el nombre `awk`. Para esto se debe crear una liga simbólica con el comando:

```
ln -s awk mawk.
```

6.3.2.4. Openssh

Socket secure host - permite conectarse a equipos locales y remotos de manera segura. De Openssh tenemos:

- el servidor: `openssh-server-3`
- el cliente: `openssh-client-3`
- las utilerias: `openssh-addon_3`

Después de descargar estos archivos es necesario ejecutar los siguientes pasos desde la consola:

1. Crear llave ssh con el programa:

```
ssh-keygen
Aparece .../.ssh/id_dsa <enter>
pass: <enter>
again: <enter>
```

2. Crear un password para el usuario `zaurus`, ya que por defecto no tiene password este usuario. Esto se hace utilizando el comando:

```
passwd zaurus <enter>
```

3. Verificar la entrada al servidor de manera local de ssh con el comando:

```
ssh localhost (La primera vez mostrara un fingerprint a lo que se responde
        yes ( <enter> ).
Aparece : zaurus@localhost y solicita el password del usuario zaurus.
```

4. Verificar la entrada desde otra computadora en la red.

```
Conectarse con: ssh zaurus@direccion_ip
```

5. Agregar la llave creada localmente al archivo authorized_keys con los siguientes comandos:

```
cd
cd .ssh
cat id_dsa.pub >> authorized_keys
```

6. Verificar localmente (no deberá pedir password).

```
Ejecutar: ssh localhost
```

7. Generar la llave en una maquina remota y enviarla al servidor mediante sftp. Ejecutar:

```
cd
cd .ssh
sftp zaurus@direccion_ip
cd .shh
put id_dsa.pub <usuario.maquina>.pub
```

8. Agregar la llave enviada abriendo la terminal en la Zaurus de manera similar al punto 5, y ejecutar los siguientes comandos:

```
cd
cd .ssh
cat <usuario.maquina>.pub >> authorized_keys
```

9. Verificar de manera similar al punto 6 (no deberá de pedir password)

```
ssh localhost
```

6.3.2.5. VNC Server

El servidor Virtual Network Connector (VNC) permite que otro equipo en la red pueda tomar el control de la interfaz gráfica de la Zaurus (ó de cualquier otro equipo) accesándolo por medio de su dirección IP y utilizando en el cliente el programa `vncviewer` para el puerto 5900, ó por medio de algún navegador de internet [2] en el puerto 5800. Si se toma como ejemplo que la Zaurus tiene la dirección IP 192.168.1.8, entonces se podrá accesar de las siguientes dos maneras, la primera con el programa vncviewer y la seguna con la ayuda de el navegador de internet el cual puede ser Netscape, Mozilla ó IE.

```
vncviewer 192.168.1.8:0
```

ó en la URL del navegador:

[2] El navegador deberá tener instalado el plugin de Java.

```
http://192.168.1.8:5800
```

El servidor de VNC aunque se encuentra en la página de killefiz.de/zaurus, se recomienda descargar la versión directamente del fabricante que es **SDG Systems** (`www.sdgsystems.com`). En este sitio se encuentra una versión de instalación para la Zaurus modelo SL-5500 que es el archivo `fbvncserver-sl5500_0.9.4_arm.ipk` y otra para la Zaurus modelo SL-5600 que es el archvio `fbvncserver-sl6000_0.9.5_arm.ipk`. Una vez descargado éste archivo, se debe copiar en la Zaurus e instalarse con el comando `ipkg` desde la terminal.

6.3.2.6. PDF viewer

Una aplicación muy útil para consultar manuales y documentos en formato PDF. El paquete de instalación se llama `qpe-qpdf_1.5.0-20020618_arm.ipk`, y se puede bajar de killefiz `www.killefiz.de/zaurus`. Se debe copiar el archivo en la Zaurus e instalarse con el comando `ipkg` desde la terminal.

6.3.2.7. Librerías

En ocasiones, algunas librerías requeridas por las aplicaciones que se instalan no se encuentran en el internet en la versión solicitada, si no en una más nueva. Para corregir esto se deben crear, desde la terminal, ligas simbólicas de la versión nueva a la vieja. Ej:

```
ln -s lib0.4 lib0.5
```

6.3.3. Base de Datos Mysql

6.3.3.1. Instalación de Mysql

Para la instalación de Mysql es necesario el paquete `mysql_3.22.32-2_arm.ipk`, el cual se puede descargar de killefiz `www.killefiz.de/zaurus`. Una vez descargado éste archivo en la Zaurus, se abre la terminal y se instala éste paquete con el comando `ipkg`[3]. Es

[3]

necesario entrar como `root` para realizar esta instalación. El demonio `mysqld` corre como usuario `deamon`, por lo que una vez instalado el paquete, el directorio `/var/lib/mysql` deberá tener como ***owner*** al usuario `daemon`. El cambio de ***owner*** se hace con los siguientes comandos desde la terminal:

`su-` // para entrar como super usuario (root)

`#cd /var/lib` // para cambio de directorio

`#chown -R deamon mysql` // le cambia de owner a deamon al directorio mysql. Con el parámetro -R (que significa Recursive) incluye el cambio a todo lo que se encuentre dentro del directorio mysql.

6.3.3.2. Arrancando MySQL

Para ejecutar y detener mysql se deben ejecutar los siguientes scripts con permisos de **root**:

`/usr/bin/mysqlstart` // inicio del demonio

`/usr/bin/mysqlstop` // terminación de ejecución del demonio

6.3.3.3. Instalación de PhpMyAdmin, Software de administración para Mysql

PhpMyAdmin es un software de administración para MySql basado en PHP. La información de donde descargar éste paquete, que versión se recomienda utilizar, y como se configura ésta aplicación se especifica más adelante en el punto 6.4.3. En esta sección se especificará únicamente su instalación.

Por cuestiones de espacio en memoria se recomienda descomprimir el archivo `phpMyAdmin-2.5.6.tar` en la tarjeta de memoria tipo flash, Secure Digital, puede ser a nivel raíz dentro del directorio `/mnt/card/` para que al descomprimirse el archivo, se cree el

`#ipkg install mysql_3.22.32-2_arm.ipk`

directorio:

`/mnt/card/phpMyAdmin-2.5.6/`

Después, se debe crear una liga simbólica al directorio que contiene phpMyAdmin, con los siguientes comandos:

```
cd /home/www/htdocs/
```

para cambiarse al directorio de inicio del servidor Apache de la Zaurus.

```
ln -s /mnt/card/phpMyAdmin-2.5.6/ phpmyadmin
```

para crear la liga simbólica.

6.3.3.4. Instalación del controlador JDBC

Para realizar una conexión a la base de datos MySQL desde PersonalJava es necesario, al igual que en la PC de escritorio (punto 6.4.5.3), instalar el controlador JDBC en la Zaurus. En la Zaurus no es necesario copiar el controlador dentro del directorio del runtime de PersonalJava, y, debido a las limitaciones de memoria para almacenamiento de nuevos programas, se recomienda copiar este controlador en la tarjeta de memoria Secure Digital SD. El archivo del controlador para la conexión a MySQL es **el mismo** que el de la PC de escritorio, `com.mysql.jdbc.Driver` y también viene incluído en el archivo comprimido (.jar) llamado `mysql-connector-java-2.0.14-bin.jar`, el cual se puede descargar del sitio de MySql que es `http://www.mysql.com`

Para instalar el controlador es necesario copiar éste archivo en la Zaurus, de preferencia en la tarjeta de memoria secure digital (SD) para evitar saturar la memoria, en algún directorio creado manualmente para este propósito como por ejemplo: `/mnt/card/jdbc/`

6.3.4. Servidor Web Apache con PHP versión 4

De pequeñas dimensiones pero muy potente ya que la versión Apache para Zaurus contiene además de PHP las librerías de acceso a la base de datos Mysql.

6.3.4.1. Instalación de Apache con PHP

El archivo ipk de instalación se encuentra en el sitio chuop.net en la siguiente URL choup.net `http://www.choup.net/v2/index.php?id=pda` y se llama:

```
apache-1.3.27-php-4.2.3_0.1_arm.ipk.
```

Es necesario entrar como `root` para instalar el paquete con el comando ipkg. Una vez instalado el paquete es necesario realizar los siguientes pasos:

1. Instalar comandos awk y less, ver punto 6.3.2.3

2. Editar el archivo: `/home/www/bin/apachectl` para modificar la variable HTTPD, y poner el directorio correcto en donde se encuentra el programa httpd

```
HTTPD=/home/www/bin/httpd
```

3. Modificar la variable PIDFILE

```
PIDFILE=/home/www/logs/httpd.pid
```

4. Verificar que exista el archivo `/etc/hosts` y que al menos contenga la siguiente línea

```
127.0.0.1 localhost localhost.localdomain
```

5. Configurar extensiones en php editando el archivo `/home/www/conf/httpd.conf`

```
AddType application/x-httpd-php .php .html
```

6. El directorio de inicio de Apache en la Zaurus es

   `/home/www/htdocs`

   Se puede crear un documento HTML con el nombre `index.html` y guardarlo en éste directorio para que se cargue al accesar el servidor Apache de la Zaurus a través del navegador desde la URL:

```
http://localhost
```

6.3.4.2. Arranque, reinicio y detención del servidor Apache con php

Dentro del direcorito `/home/www/bin/` se encuentra el script `apachectl` que se encarga de levantar, detener ó reinicar el demonio de httpd.

`./apachectl start` // inicia el servidor Apache

`./apachectl stop` // detiene el servidor Apache

`./apachectl status` // muestra el estatus del servidor

6.3.5. Redes y comunicaciones

6.3.5.1. Samba

La herramienta para conectarse a equipos con otros sistemas operativos. El sitio oficial de samba es: samba `www.samba.org`.

- Arranque de samba. En la Zaurus, dentro del directorio `/home/etc/rc.d/init.d/` se encuentra el script de arranque del demonio llamado `samba` . Se ejecuta simplemente con: `./samba start` [4]

- Configurando samba. El archivo de configuración de samba se llama `smb.conf` y se encuentra en `/home/root/usr/lib/samba/`. Este archivo debe de contener al menos estas dos líneas, una para el indicar el grupo de trabajo y otra que indique el manejo de contraseñas encriptadas, ya que Windows así lo requiere. Ej:

```
workgroup = GRUPO_TRABAJO
encrypt passwords = yes
```

- Listando recursos compartidos de la Zaurus. Para listar los recursos compartidos del host se puede utilizar en el dektop el comando smbclient. Ej: `smbclient -L zaurus`. Es necesario indicar en el desktop en que dirección IP se encuentra la Zaurus, esto se indica en el archivo `/etc/hosts`, el cual deberá contener la siguiente línea:

[4] El arranque de samba abre el puerto :901

```
192.168.0.14  zaurus
```

en donde `192.168.0.14` es la dirección IP asignada en la red a la Zaurus.

6.3.6. Otras aplicaciones

6.3.6.1. Advance File Manager

Advance File Manager es un programa que permite explorar el sistema de archivos de la Zaurus con accesos directos a la memoria tipo flash secure digital (SD) y compact flash (CF). Para su instalación se requiere únicamente del paquete: `qtopia-advancedfm_1.0_arm.ipk` que se puede descargar desde `http://www.my-zaurus.narod.ru/`

6.3.6.2. HTML@Z

HTML@Z, un editor de html para Zaurus. Para su instalación de necesita descargar el archivo `htmlatzaurus_0.2.2_arm.ipk` del sitio `www.killefiz.de/zaurus` e instalarlo con el comando ipkg.

6.3.6.3. Rotate

Con esta aplicación se puede rotar el display de manera horizontal, muy conveniente cuando se leen archivos PDF ó manuales en general. Se requiere instalar un plugin que se puede descargar de `http://my-zaurus.narod.ru/`; el archivo se llama `rotate_0.0.2_arm.ipk`

6.3.6.4. Cute Icons

Estos son unos iconos un poco más bonitos que los originales del ROM de Sharp. Para instalarlos, se debe descargar el archivo `cool-icons_0.0.1_arm.ipk` desde el sitio `http://my-zaurus.narod.ru/` e instalarse con el comando ipkg.

6.3.7. Configuración del puerto IrDa para impresión.

En el costado izquierdo de la Zaurus se encuentra el puerto infrarojo (IrDa). Este puerto no es muy potente y puede alcanzar sólo algunos centimetros cuando más. Más adelante se mostrarán aplicaciones que utilizarán el puerto infrarojo para la impresión de códigos de barrra desde varias plataformas de desarrollo como Qt en el punto 6.5.5, PHP en el punto 6.6.3.4 y Java en el punto 6.6.3.4. Por el momento, en esta sección únicamente se procederá con la creación del nodo para que este listo para ser utilizado por estas aplicaciones.

Para configurar la Zaurus para impresión por el puerto Infrarojo se necesita crear un nodo que apunte hacia el dispositivo IrDa. Para realizar esto se debe entrar a la terminal y ejecutar las siguientes instrucciones:

```
#su  // para obtener permisos de root
#mknod /dev/irlpt0 c 161 16    // crea el nodo del dispositivo
#cd /dev
#chown zaurus irdevlpt0  // cambia de owrner al nodo para que
                    éste pueda ser utilizado por el usuario -zaurus-
```

Las instrucciones anteriores crean el dispositivo llamado *irlpt0*, en el cual se pueden enviar archivos de texto desde la terminal utilizando el comando `cat`:

```
cat ruta/archivo > /dev/irlpt0
```

6.3.8. Recomendaciones al finalizar la configuración de la PDA

6.3.8.1. Creación de memoria swap

Al terminar con la instalación del software antes mencionado, la PDA quedará con muy poco espacio para correr los servicios instalados, por lo que resulta esencial proveerla de memoria adicional para cumplir con los requerimientos demandados por el software y el usuario. Aunque se puede utilizar la memoria Compact Flash CF para este propósito, se recomienda el uso de la memoria Secure Digital SD ya que la ranura para CF resulta más útil

para otro tipo de dispositivos como tarjetas de red inhalámbrias, lectores de códigos de barra, modems, etc. A continuación se enumeran los pasos para la creación de memoria adicional por 32 Megabytes para swap dentro de un archivo llamado [`swapfile`]:

1. Abrir la terminal y cambiar a usuario root como el comando: `su`
2. Cambiarse de directorio hacia la tarjeta de memoria SD ej: `# cd /mnt/card`
3. Crear un archivo para memoria swap con el comando dd ej: `# dd /if/dev/zero of=swapfile bs=1024 count=32768`
4. Inicializar el archivo swapfile creado en paso anterior con el comando mkswap, ej: `mkswap -v1 swapfile`
5. Activar las particiones y/o archivos de memoria swap con el comando `# swapon -a`
6. Verificar que el archivo este siendo utilizado para swap con el comando `# free`
7. Para detener el uso de la memoria swap se utiliza el comando swapoff, ej: `# swapoff swapfile`

AVISO IMPORTANTE

Si se desea extraer la tarjeta de memoria flash Secure Digital (SD) es necesario desactivar el archivo de memoria swap previamente a la extracción ya que de lo contrario el equipo podría congelarse y el sistema de archivos de la tarjeta de memoria SD podría dañarse permanentemente.

6.3.8.2. Respaldo

Se recomienda una vez terminada la configuración de la PDA realizar un respaldo en tarjeta de memoria flash, ya sea SD (SecureDigital) ó CF (Compact Flash). Se requieren de 16 a 26 MB libres dependiendo del número de aplicaciones instaladas.

Este respaldo se realiza con la utilidad *Backup/Restore* ubicada en la pestaña *Settings* como se muestra en la figura 6.3. Es necesario especificar el tipo de medio, memoria SD ó CF en donde se desea realizar el respaldo como se muestra en la figura 6.4. El nombre del archivo se genera también en automático en un formato *año-mes-día-hora-minuto* en que se realiza éste proceso, el programa de respaldo determinará automáticamente si hay espacio suficiente. Para que comience el proceso se presiona el botón *Start*.

Figura 6.3: backup-restore

Figura 6.4: Backup Media

El archivo creado de respaldo contiene TODA la información de la PDA, no solo del PIM, sino toda la instalación tanto de aplicaciones como de datos.

6.3.8.3. Restauración

El proceso de restauracón es similar al de respaldo, a diferencia de seleccionar la opción `Restore`. Se debe especificar en que medio se encuentra el respaldo SD ó CF, seleccionar el archivo de restauración y presionar el botón `start` para iniciar éste proceso.

Es muy importante que mientras se están ejecutando estos procesos no se desconecte de la corriente eléctrica el equipo, y/o que cuente con batería suficiente para terminar este proceso completamente, de lo contrario el medio de respaldo e incluso la Zaurus, podría dañarse.

6.3.8.4. Organizar iconos

La versión 3.10 del ROM de Sharp Zaurus tiene la opción para personalizar la ubicación de los iconos de los programas en las pestañas que Qtopia despliega en la Zaurus (*applications / games /settings /documents*) con el programa `tab setting` que se encuentra en la pestaña `settings`. Resulta entonces conveniente, una vez que se tengan instaladas las nuevas aplicaciones, crear pestañas nuevas para agruparlas. Como ejemplo, se puede crear una pestaña llamada `Network` para ubicar aquí todas las aplicaciones de red (VNC Server, Terminal, Opera browser, EMail).

6.4. Configuracion básica de la PC de escritorio

6.4.1. Necesidad de un equipo de escritorio ó laptop.

Debido a las limitaciones de la Zaurus en cuanto a la capacidad del equipo, el espacio en memoria, la velocidad del procesador, el tamaño de la pantalla, el sistema de ventanas y la incomodidad del teclado se requiere de un equipo desktop para el desarrollo de aplicaciones. Además, los paquetes de programación no estan diseñados para instalarse y correr en Linux integrado.

6.4.2. Instalación de la base de datos Mysql en la PC de escritorio

La idea de instalar MySql es para propósitos de desarrollo y depuración dentro de la PC de escritorio, sin tener que accesar a MySql en la Zaurus durante éste proceso. Para instalar MySql se necesita: **a)** instalar el paquete rpm, **b)** dar de alta un usuario llamado `mysql`, **c)** cambiar el **owner** del directorio de instalación y, por último, **d)** levantar el demonio.

a) Instalación del paquete rpm con MySQL.

En el disco 2 de la distribución de RedHat Versión 9.0 se encuentra el archivo que contiene la versión 3.23 de MySQL (dentro del directorio `[cdrom-path]\RedHat\RPMS`):

```
mysql-server-3.23.54a-11.i386.rpm,
```

Para instalar el rpm es necesario abrir una sesión en la terminal con permisos de *root*, cambiarse a este directorio dentro del cd-rom (`[cdrom-mount-path]\RedHat\RPMS`) y ejecutar el comando: `rpm -ivh mysql-server-3.23.54a-11.i386.rpm`. El archivo de configuración que se llama `/etc/my.cnf` debe contener las siguientes líneas:

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
[mysql.server]
user=mysql
basedir=/var/lib
[safe_mysqld]
err-log=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

b) Dar de alta el usuario llamado *mysql*

En una ventana de la terminal con permisos de *root*, se debe ejecutar el comando :

```
# adduser mysql /home/mysql
```

c) Cambiar el owner al directorio de Mysql

El directorio de mysql es `/var/lib/mysql`. Éste y todos los archivos dentro del directorio deben tener como ***owner*** a un usuario llamado `mysql` (previamente dado de alta). Para realizar ésta tarea es necesario abrir la terminal con permisos de ***root*** y ejecutar los siguientes comandos:

```
cd /var/lib          // Para cambiarse al directorio Raíz de Mysql
$chown -R mysql mysql //Este comando cambia el owner
                           -R significa Recursivamente.
```

d) Levantar el demonio MySQL.

Para levantar el demonio, se necesita abrir la terminal con permisos de ***root***, y ejecutar los siguientes comandos:

```
#cd /etc/rc.d/init.d
#mysqld start          //Para iniciar el demonio
```

e) Para detener el demonio MySQL

Se utiliza el mismo script que se usa para arrancarlo, pero con la opción `stop`, ej:

```
#mysqld stop
```

6.4.3. PhpMyAdmin - Software de administración para Mysql

Existe un paquete llamado PhpMyAdmin el cual facilita la administración de la base de datos MySQL tanto el desktop como en la Zaurus. Es necesario instalar la versión *2.2.7-pl1* de phpMyAdmin con soporte para MySQL versión 3.22 y no una mayor a ésta, como MySQL versión 3.23. Esto debido a que la versión disponible por el momento de *MySQL* **para la Zaurus** es la **3.22**. Es mandatorio que el servidor web Apache tenga instalado el módulo de PHP (ver punto 6.4.6.1), ya que éste software está programado en PHP. A continuación se describen los pasos para instalar phpMyAdmin.

1. Descargar del sitio `http://www.phpmyadmin.net` el archivo

```
phpMyAdmin-2.2.7-pl1-php.tar.gz
```

2. Copiar este archivo dentro en el directorio de inicio de documentos html, ej:

```
/var/www/html/
```

3. Descomprimir el archivo con el comando

```
#tar xvf phpMyAdmin-2.2.7-p11-php.tar.gz
```

4. Configurar el archivo de inicio, especificando las variables de configuración correspondientes para cada caso. Ver punto 6.4.3.2.

```
config.inc.php
```

6.4.3.1. Instalación de phpMyAdmin en la PC de Escritorio

Para instalar phpMyAdmin en la PC de escritorio con Red Hat 9.0 se debe descomprimir el archivo
`phpMyAdmin-2.5.6.tar` dentro del directorio: `/var/www/html/`
Esto crea el directorio `/var/www/html/phpMyAdmin-2.5.6/`
Para facilitar la sintaxis de acceso a phpMyAdmin-2.5.6 se recomienda crear una liga simbólica en el directorio `/var/www/html/` con el comando:

```
ln -s /var/www/html/phpMyAdmin-2.5.6/ phpmyadmin
```

6.4.3.2. Configuración de phpMyAdmin

El archivo de configuración se llama `/phpMyAdmin-2.5.6/config.inc.php`, y debe especificar como mínimo las siguientes variables:

Dirección y path

`$cfg[PmaAbsoluteUrl]=http://direccion_ip_Zaurus/phpmyadmin` Dirección y path en donde se encuentra instalado phpMyAdmin.En esta variable se especifica la dirección IP (ó hostname)del equipo que tiene instalado phpMyAdmin, ya sea la dirección IP del desktop ó de la Zaurus. Si se hace de manera local se puede indicar *localhost*.

Nombre del host

`$cfg[Servers][$i][host] = localhost` Esta variable contiene la dirección IP (ó host-name) del equipo que tiene instalado MySQL. Para accesar Mysql localmente se indica *localhost,* si se va a accesar Mysql en la Zaurus desde el desktop, se debe indicar la dirección IP de la Zaurus.

Nombre del usuario

`$cfg[Servers][$i][user] = zaurus` En esta variable se debe indicar algún ***usuario de Mysql***

Contraseña

`$cfg[Servers][$i][password] = (Password del usuario de Mysql)` En esta variable se debe indicar el password o contraseña del ***usuario de Mysql***

6.4.3.3. Acceso a PhpMyAdmin en la red

PhpMyAdmin es una aplicación basada en PHP, ésta se puede instalar y ejecutar tanto en el desktop como en la Zaurus. Si se instala *phpMyAdmin* en la PC de escritorio, se puede correr PhPMyAdmin a través del navegador desde la Zaurus y accesar la base de datos *Mysql* instalada en la PC de escritorio, ej: `http://ip_PC_escritorio/phpmyadmin`. Si se instala *phpMyAdmin* en la Zaurus se puede correr phpMyAdmin a través del navegador desde el desktop y accesar la base de datos *Mysql* instalada en la Zaurus, ej: `http://ip_Zaurus/phpmyadmin`.

6.4.4. Configuracion básica para programación en Qt.

La PC de escritorio para desarrollo debe tener una distribución de Linux ya precargada, preferentemente una que nativamente soporte paquetes de RPM, por ejemplo Red Hat, SuSE, Mandrake o Caldera. Tambien se pueden utilizar distribuciones como Slackware y Debian, pero se necesita usar una utilidad de conversion de RPM como "alien" para soportar el formato RPM.

6.4.4.1. Instalación del SDK - QPE

El desarrollo nativo de la Zaurus se realiza utilizando C++ y Qt de TrollTech. QPE viene con un marco búfer (frame buffer) virtual llamado **qvfb**, que permite probar aplicaciones sobre X11 sin necesidad de tener una Zaurus. Para ejecutar aplicaciones en la Zaurus(y en qvfb) se necesita ligar la aplicación hacia QPE en vez de Qt.

Para desarrollar en Qt se necesita obtener el Kit de Desarrollo de Software (SDK) de TrollTech, ya sea la version GPL ó la comercial. El paquete GPL para Redhat V9 se llama `qtopia-free-1.7.0-2rh9.i386.rpm` y se puede descargar del stitio de Trolltech en `http://www.tolltech.com/download/qtopia/index.html`
Por default, RPM instala el Kit de Desarrollo de Software (SDK) de Qtopia en el directorio `/opt/Qtopia/`.

6.4.4.2. Instalación del compilador multi-plataforma (Cross-Compiler Setup)

Varias distribuciones ahora usan GCC 3.2 ó mayor, como su compilador por default. Esto significa que la compilación para el desktop no estará habilitada para ligar los archivos SDK ya que éstos fueron creados usando GCC 2.95. Para arreglar este problema se necesitará instalar la version 2.95 en la distribución de Linux que se está utilizando. Es necesario descargar los siguientes paquetes de la red de internet:

`gcc-cross-sa1100-2.95.2-0.i386.rpm` (gcc compilaldor para la arquitectura ARM)

`binutils-cross-arm-2.11.2-0.i386.rpm` (utilerías binarias para arquitectura ARM)

`glibc-arm-2.2.2-0.i386.rpm` (GNU C librerías para arquitectura ARM)

`linux-headers-arm-sa1100-2.4.6-3.i386.rpm` (linux header files para arquitectura ARM)

Para RedHat V 9 En el CD #1 , en los archivos RPMS, se debe encontrar e instalar lo siguiente:

`compat-gcc-c++-7.3-2.96.118`

`compat-libstdc++-devel-7.3-2.96.118`

`compat-libstdc++-7.3-2.96.118`

`compat-gcc-7.3-2.96.118`

Cada uno de los archivos RPM necesitarán ser instalados desde la línea de comandos:

```
rpm -Uvh filename.rpm
```

Por defecto, RPM instala la cadena de herramientas ARM en el directorio `/opt/Embedix/`. Según la documentación de Qtopia se debe realizar una copia del archivo `tmake.conf` hacia `/opt/Qtopia/tmake/lib/qws/linux-x86-g++`, pero en Redhat V9 la instalación ya esta lista para correr como esta, por lo que **no se debe** copiar este archivo.

6.4.4.3. Variables de entorno del compilador multiplataforma

Después de que el SDK y el compilador (toolchain) son instalados, se recomienda crear dos archivos secuenciales (batch files) en el directorio de inicio(homedir). Uno para configurar las variables de entorno para compilar versiones para x86 de aplicaciones Zaurus (usando qvfb), y otro para configurar las variables de entorno para realizar compilación nativa para el procesador ARM de la Zaurus.

- El archivo secuencial #1, `dev-x86-qpe.sh` debe contener las siguientes líneas:

```
export QPEDIR=/opt/Qtopia
export QTDIR=/opt/Qtopia
export PATH=$QTDIR/bin:$PATH
```

```
export TMAKEPATH=/opt/Qtopia/tmake/lib/qws/linux-generic-g++
export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
echo "Entorno alterado para desarrollo en x86 de Sharp Zaurus"
```

- El archvio secuencial #2, `dev-arm-qpe.sh` debe contener las siguientes líneas:

```
QPEDIR=/opt/Qtopia/sharp
QTDIR=/opt/Qtopia/sharp
PATH=${QTDIR}/bin:${QPEDIR}/bin:${CROSSCOMPILE}:${PATH}
TMAKEPATH=/opt/Qtopia/tmake/lib/qws/linux-sharp-g++/
#LD_LIBRARY_PATH=${QTDIR}/lib:${LD_LIBRARY_PATH}[5]
export QPEDIR QTDIR PATH TMAKEPATH LD_LIBRARY_PATH
echo "Entorno alterado para desarrollo en ARM de Sharp Zaurus"
```

Cuando se desee compilar y probar aplicaciones x86 se deberá ejecutar el siguiente comando desde el directorio de inicio (home):

```
#source dev-x86-qpe.sh
```

Cuando se desee compilar y probar aplicaciones ARM en la Zaurus se deberá ejecutar:

```
#source dev-arm-qpe.sh
```

6.4.4.4. Configuración de tmake.conf para compilación ARM de la Zaurus.

La versión GPL del Kit de Desarrollo de Software (SDK) de TrollTech para Redhat V9 que se instaló tiene un pequeño error, que no permite la compilación para el procesador ARM de la Zaurus. Para corregirlo es necesario reconfigurar el archivo:

```
/opt/Qtopia/tmake/lib/qws/linux-sharp-g++/tmake.conf
```

A la línea número 52 que originalmente dice `TMAKE_LIBS =` hay que cambiarla por `TMAKE_LIBS =-ljpeg -luuid` . Esto por que al cargar las librerías jpeg y uuid, el compilador para ARM marca error de librerías inexistentes y se interrumpe impidiendo la compilación.

[5] Esta línea DEBE estar comentada ó simplemente no ser incluída. Aunque en la documentación de Qtopia se menciona que se debe escribir, en la práctica, el incluirla generá que el shell se interrumpa.

6.4.4.5. Prueba del compilador con aplicación de ejemplo

El directorio `$QPEDIR/examples/application` incluye el código fuente para crear una aplicación de prueba[6].

1. Primero, se ejecuta el script de configuración de variables de entorno para x86 en el directorio de inicio (home directory) desde una sesión de shell con el siguiente comando:
   `#source dev-x86-qpe.sh`

2. Siguiente, dentro de la misma sesión pero dentro del directorio
   `/opt/Qtopia/examples/application/`
   se debe ejecutar el siguiente comando para crear el Makefile:
   `#tmake -o Makefile example.pro .`

3. Para crear el programa ejecutable ó la aplicación, se ejecuta el siguiente comando dentro de ése mismo directorio:
   `#make`

4. Para ejecutar la aplicación creada en X11 este necesario utilizar qvfb. El punto 6.4.4.6 explica su uso.

Alternativamente, para construir una versión para *ARM* que se ejecute en la *Zaurus*, hay que realizar los siguientes pasos:

1. Primero , ejecutar el script de configuración de entorno de variables desde una ventana de shell en el directorio de inicio: `#source dev-arm-qpe.sh`

2. Si existe un *Makefile* se recomienda removerlo del directorio de ejemplo con el comando rm: `#rm Makefile` *(NO es esencial hacer esto).*

3. Ejecutar el comando : `#make clean` , desde el directorio
   `/opt/Qtopia/examples/application/` para eliminar los viejos archivos temporales que se crearon en la configuración para x86.

[6] Se recomienda crear una copia de este directorio previo a la prueba del compilador.

4. Ejecutar: `# tmake example.pro >Makefile`, de nuevo para crear los Makefiles necesarios para la compilación para ARM.

5. Para crear el programa ejecutable para ARM, ejecutar : `# make`, dentro del mismo directorio.

6. Una vez que se compiló, esta listo para ejecutarse. Necesita ser copiado a la Zaurus y ejectuarse desde la consola ó terminal con el comando:`# ./example`

7. Si el programa no se ejecuta debido a la ausencia de la librería *libqtopia.so.1,* el punto 6.3.2.2 indica como crearla.

Para mayor información acerca de Qt, QtDesigner, como crear el archivo aplicacion.pro, qmake, tmake y la mayoría de la preguntas relacionadas con Qt y la documentación del SDK, se puede consultar el sitio web de Trolltech en trolltech `http://doc.trolltech.com`

La documentación incluída en el SDK de Qtopia (incluyendo qpe) esta localizada en la PC en: `/opt/Qtopia/doc/index.html` , y también esta disponible en el sitio de Trolltech `http://doc.trolltech.com`, para la versión en línea actualizada.

6.4.4.6. Empleo de qvfb en X11

Las aplicaciones pueden ser probadas y ejecutadas dentro de X11 usando qvfb. Para esto se deben realizar sólo 2 pasos:

1. Ejecutar el comando: `./qvfb &`, para lanzar un display simulando al de la Zaurus. Cualquier aplicación Qt/Embedded compilada para x86 que se ejecute ahora será desplegada en esta ventana.

2. Despues, hay que ejecutar: `./example -qws` para correr la aplicación de ejemplo en modo de servidor.

Alternativamente, la aplicación puede instalarse en `qpe` para simular el entorno actual de Qtopia y correr la aplicación ejemplo en una manera de no servidora (non-server) en X11.

Para instalar la aplicación de ejemplo, desde el directorio:
`/opt/Qtopia/examples/application/`, entrando como root `su` para obtener privilegios de instalación se deben realizar los siguientes pasos :

```
cp example.desktop $QPEDIR/apps/Applications
cp Example.png $QPEDIR/apps/Applications
cp exmaple $QPEDIR/bin/
cp example.html $QPEDIR/help/html/
exit # no se necesita más ser root
```

Despues, para correr qpe :

```
qvfb &
qpe
```

La aplicación de ejemplo aparecerá en la pestaña *Applications* con el icono color rojo de signo de interrogación.

6.4.4.7. Qt Designer para aplicaciones ARM en la Zaurus.

Qt designer es una herramienta WYSIWYG para desarrollo de aplicaciones en Qt. Ésta aplicación genera archivos con extensión .ui, los cuales son archivos de texto en formato XML. Algunas distribuciones de Linux como RedHat 9.0 incluyen Qt designer, pero el archivo XML que generan no es compatible con el SDK de Qt para procesadores ARM por lo que se debe utilizar la versión de Qt Designer que viene por defecto con el SDK para ARM aunque ésta no sea tan nueva. Este programa se arranca desde el shell una vez cargadas las variables de entorno (punto 6.4.4.3), con el comando:

```
#designer
```

6.4.4.8. Herramienta tmake

Tmake es una herramienta de Trolltech, fácil de usar de usar que permite mantener *make-files* para proyectos de software. Puede ser una tarea muy laboriosa y molesta el querer

manejar los makefiles de manera manual, especialmente si se desarrolla para mas de una plataforma ó se usa más de un compilador. Tmake automatiza y facilita este proceso y asi se puede invertir mas tiempo en desarrollo y no en hacer *makefiles*.

La principal motivación que genero el desarrollo de tmake fue el mucho tiempo invertido en el mantenimiento de los makefiles para Qt, el kit de herramientas Graficas (GUI) para multi-plataforma. Qt soporta cerca de 15 sabores ó tipos de Unix, Windows y alrededor de 15 diferentes compiladores de C++.

Tmake esta escrito en Perl y requiere que se tenga instalado una version 5 ó mayor de Perl. El uso basico de tmake no requiere conocimiento de perl, pero si se conoce perl se puede extender tmake y escribir plantillas (templates) propias para makefiles.

Tmake es un software gratuito y esta permitido su uso, copia, modificación y distribución, tanto del software como de su documentación. En la versión de Qt viene incluído el programa `tmake`.

Instalación de tmake

La versión SDK de Qt para procesadores ARM ya incluye el programa tmake. Para instalar tmake es necesario contar con una version 5 ó mayor de Perl y realizar lo siguiente:

1. Desempaquetar tmake del archivo con extensión tar.gz para Unix

2. Configurar la variable de entorno TMAKEPATH hacia los directorios que contienen los archivos de plantillas 'template files'.

3. Agregar el directorio tmake/bin al PATH. Aquí estan unos ejemplos:

```
Unix Bourne shell:
TMAKEPATH=/local/tmake/lib/linux-g++
PATH=$PATH:/local/tmake/bin
export TMAKEPATH PATH
Unix C shell:
setenv TMAKEPATH /local/tmake/lib/linux-g++
```

```
setenv PATH $PATH:/local/tmake/bin
```

El nombre del directorio de la plantilla ó template tiene la forma *plataforma-compilador* y contiene un archivo de configuración para la plataforma(*tmake.conf*) y los archivos de plantillas de tmake *(tmake template files*).

La plataformas soportadas son: AIX, Data General, FreeBSD, HPUX, SGI Irix, Linux, NetBSD, OpenBSD, OSF1/DEC, SCO, Solaris, SunOS, Ultrix, Unixware y Win32.

Se puede encontrar la plataforma-compilador deseada en el directorio `tmake/lib`.

En Unix `tmake` requiere que **Perl** este instalado en el directorio `/usr/bin`. Si la versión de perl esta en algún otro lado, se debe cambiar la primera línea de tmake ó cear un pequeño script que invoque tmake con la ubicación correcta de **Perl**.

Comenzando...

Vamos a asumir que se tiene una pequeña aplicación de Qt que consiste en un header file de C++ y dos archivos fuentes. Primero se necesita crear el archivo de proyecto, p.e. `hello.pro` que contenga las siguientes líneas:

```
HEADERS = hello.h
SOURCES = hello.cpp main.cpp
TARGET = hello
```

Después se debe ejectar tmake para crear el Makefile:

```
tmake hello.pro -o Makefile
```

Y finalmente correr *make.* Esto crea el programa `hello`.

```
make
```

Es necesario configurar la variable de entorno TMAKEPATH , antes de ejecutar `tmake`.

6.4.5. Configuracion básica para programación en PersonalJava.

6.4.5.1. Instalación de herramientas básicas para desarrollo en PersonalJava

Las herramientas básicas necesarias para el desarrollo de aplicaciones en Java son:

- El compilador Java
- El intérprete de Java, ó entorno de ejecución Java
- La herramienta de empaquetamiento de Java

Para la instalación de éstas herramientas es mandatorio tener permisos de **root**. El SDK[7] de Java incluye el compilador `javac`, el intérprete `java` y la herramienta de empaquetamiento `jar`. Éste kit de desarrollo para Java *J2SDK* versión estandar 1.4.2_xx[8] se puede descargar del sitio de Sun `http://java.sun.com/j2se/1.4.2/download.html`. El archivo a descargar para RedHat es *j2sdk-1_4_2_xx-linux-i586rpm-bin*. Una vez descargado el archivo se deben realizar los siguientes pasos y comandos:

1. Cambiarle el permiso de ejecución al archivo descargado:
   `chmod +x j2sdk-1_4_2_xx-linux-i586rpm-bin`
2. Ejecutar el programa binario:
   `./j2sdk-1_4_2_xx-linux-i586rpm-bin`
3. Ver y aceptar los términos de la licencia de *Sun*
4. Instalar el rpm:
   `rpm -ivh j2sdk-1_4_2_xx-linux-i586.rpm`
5. Se debe crear una liga simbólica al directorio en donde se instaló Java:

   `ln -s /usr/local/java /usr/java/j2re_1.4.2_xx/`

[7] SDK - Software Development Kit en Inglés, ó Paquete de Desarrollo de Software en Español.
[8] xx -se refiere a la versión de la revisión del paquete.

Definiendo las variables de entorno de Java

Para definir las variables de entorno de Java es necesario editar el archivo `/etc/profile` y agregarle las siguientes tres líneas:

```
JAVA_HOME = /usr/local/java
export JAVA_HOME
PATH=$PATH:$JAVA_HOME/bin
```

6.4.5.2. Instalación del plugin de Java en el navegador Mozilla

Para correr applets en la PC de escritorio es necesario instalar el plugin de Java para el navegador. Instalar el plugin de Java para el navegador es tan simple como copiar la librería `libjavaplugin.so` en el directorio para *plugins*. Ejemplo, para instalar el plugin Java en el navegador mozilla versión 1.2.1 se deben ejecutar lo siguientes comandos:

1. Para localizar el plugin de Java[9]

   ```
   [# find / -name "libjavaplugin*"]
   ```

2. Cambiarse a usuario root

   ```
   $ su -
   ```

3. Cambiarse al directorio de Mozilla en donde se deben instalar los plugins

   ```
   # cd /usr/lib/mozilla-1.2.1/plugins
   ```

4. Crear dentro del directorio de plugins una liga simbólica que apunte al archivo del plugin de Java

   ```
   # ln -s /usr/java/j2re_1.4.2_06/jre/plugin/i386/ns610/libjavaplugin_oji.so .
   ```

6.4.5.3. Instalación del controlador para acceso a Mysql

Para accesar a la base de datos Mysql en la PC de escritorio es necesario instalar el controlador JDBC que define la conexión a MySql. Este controlador se llama *com.mysql.jdbc.Driver*

[9] El comando entre corchetes[] es opcional y se debe ejecutar si se desconece la ubicación del archivo del plugin de Java.

y viene incluído en el archivo `mysql-connector-java-2.0.14-bin.jar`, el cual se puede descargar de MySQL `http://www.mysql.com`

Su instalación es muy sencilla, únicamente es necesario copiar este archivo dentro del directorio de librerías externas del Runtime del entorno de Java con el comando `cp`:

```
cp mysql-connector-java-2.0.14-bin.jar  /usr/local/java/jre/lib/extw/
```

6.4.6. Configuración básica para programación en PHP.

Para instalar PHP en el desktop se requiere instalar el paquete de instalación (.rpm) que contiene PHP, configurar los archivos de inicio, integrar PHP con Apache y reiniciar el servidor web Apache. El sitio oficial de PHP es `http://www.php.net`.

6.4.6.1. Instalación de PHP con conexión a MySQL.

En el disco 2 de la distribución de RedHat Versión 9.0, se encuentra el archivo `[cdrom-mount-path]\RedHat\RPMS\php-mysql-4.2.2-17.i386.rpm` el cual contiene una versión de PHP lista para conectarse a MySQL.

Para instalarlo es necesario abrir una sesión en la terminal con permisos de **root**, cambiarse al directorio dentro del cd-rom (`[cdrom-mount-path]\RedHat\RPMS`) y ejecutar el comando:

`rpm -ivh php-mysql-4.2.2-17.i386.rpm.`

6.4.6.2. Configuración de PHP

El paquete de instalación de PHP crea el archivo para configuración llamado `/etc/php.ini` Es necesario editar este archivo y modificar los siguientes parámetros:

```
save_mode_allowed_env_vars = PHP_
```

si esta directiva esta vacía, PHP dejará al usuario modificar cualquier variable de entorno.

```
max_execution_time = 30
```

número de segundos máximos para ejecución de un script, esto es con el fin de evitar que algun script pueda entrar en un loop infinito.

`engine = on`

habilita el script de PHP en Apache

`register_global = off`

off = utiliza variables seguras, on = Utiliza variables NO seguras

`extension_dir = /usr/lib/php4`

directorio en donde residen la extensiones (módulos) que se pueden cargar.

6.4.6.3. Integración de PHP con Apache.

En el archivo de configuración de Apache `httpd.conf` es necesario agregar la siguiente línea (si la línea ya existe pero se antepone el símbolo #, éste debe removerse para activarla):

```
LoadModule php4_module modules/libphp4.so
```

Esta línea le dice al servidor web Apache que utilize la librería objeto compartida que fue creada en el proceso de instalación de PHP (libphp4.so).

6.4.6.4. Configuración MIME

Como último paso, es necesario indicar la configuración MIME[10] para que el servidor Web Apache reconozca las extensiones de archivos de PHP, para lo que se deben agregar en el archivo `/etc/httpd.conf` las siguientes líneas:

```
AddType application/x-httpd-php .php .phtml .html
AddType application/x-httpd-php-source .phps
```

En resumen, para que PHP funcione con Apache el archivo `httpd.conf` debe contener las siguientes líneas:

```
LoadModule php4_module modules/libphp.4so
AddType application/x-httpd-php .php. phtml .html
AddType application/x-httpd-php-source .phps
```

[10] MIME en Inglés Multipropouse Internet Mail Extension, un sistema estandar para determinar el tipo de aplicación relacionada con un archivo en base a su extensión.

6.4.6.5. Reinicio del servidor web Apache

Para que el servidor web Apache tome la instalación de PHP es necesario reiniciar el servicio desde la terminal (con permisos de **root**), con el comando:

```
#service httpd restart
```

6.5. Desarrollo de aplicaciones en Qt

Bueno, llego la hora de ensuciarse un poco las manos con algo de código real. Por supuesto que el primer programa será el ejercicio tradicional `Hola mundo`, después un programa pequeño pero completo que mostrará algunos menús y funciones para pintar, el siguiente programa mostrará la conexión a base de datos y por último una aplicación para impresión de códigos de barra a través del puerto infrarojo IrDa hacia una impresora portátil modelo *Cameo 3* de la compañía *Zebra Technologies*.

Como programar en Qt y C++ están fuera del alcance de esta tesis. Como referencia al respecto se puede consultar la bibliografía, ver [Qt programming].

6.5.1. Usando la documentacion de referencia de Qt.

Qt viene con documentación en formato HTML, y simplemente es excelente. Esta es la principal razón de por que no se incluye en esta tesis. No es difícil encontrarle la forma de acceder la documentación, pero si toma un poco de tiempo el llegar a acomodarse con esta. Esta explicación ayudará a encontrale el modo un poco más rapido.

Existen dos formas de usar la documentación de referencia. Primero, se puede usar cualquier navegador, ya que la documentación esta en formato HTML plano. Segundo, también se puede usar el programa Qt Assistant, el cual esta incluído con Qt. El asistente tiene la ventaja de que indexa toda la documentación, lo que hace más fácil la búsqueda de algo de lo que no se tiene ni idea de donde buscarlo. Del otro lado, quizá se este más acostumbrado a manejar el navegador, y sus características de *Favoritos (bookmarks)* son un poco mejores que las que ofrece el asistente de Qt.

Para leer la documentación, hay que arranacar el navegador. Como Qt no usa ningun truco especial de HTML, y por que tampoco se basa en Java ó Javascript, se puede usar cualquier navegador como por ejemplo Konqueror, Netscape, IE, Lynx u Opera.

Solo hay que apuntar el navegador hacia el archivo index.html que se encuentra en el directorio `doc/html` dentro de la instalacion de Qt. Para obtener una idea de cómo este organizada la clase de Qt, se elige la opcion *Annotated Classes* en la seccion *API Referente*.

Si se da clic en este link, se presenta en una lista de todas las *clases públicas (public class)* de Qt con solo una línea de descripción de lo que cada una hace. Después de trabajar por algún tiempo con Qt, se habrá dominado esta información, por lo que se pretenderá elegir simplemente la *Lista de todas las clases (All Classes List).* Esta lista no contiene anotaciones y despliega un mayor número de clases en una sola pantalla, pero por el momento habrá que concentrarse en las *clases anotadas (Annotated Classes).* Existe también la opción *Clases Principales (Main Classes)* que muestran una lista de las clases que los desarrolladores de Qt piensan que son especialmente importantes. De cualquier forma, esta lista pudiera no mostrar lo que alguien en particular necesite, así que se sugiere mejor acostumbrarse a la lista completa.

6.5.2. El primer programa multi-plataforma en Qt - holamundo.cpp

`holamundo.cpp`, este programa es muy simple, crea una pequeña ventana que dice *Hola Mund*o, y contiene código que se verá frecuentemente en programas de Qt. Para crear un programa en Qt se necesita escribir el código fuente y compilarlo.

6.5.2.1. Código fuente - holamundo.cpp

Con la ayuda de un editor de texto se debe crear un archivo llamado `holamundo.cpp`, y anotar dentro de éste las siguientes líneas de código:

```
#include <qpe/qpeapplication.h>
#include <qlabel.h>
int main( int argc, char ** argv )
{
  QPEApplication myapp( argc, argv );
  QLabel* mylabel = new QLabel( "Hola, Mundo", 0 );
  mylabel->resize( 120, 30 );
  myapp.setMainWidget( mylabel );
  mylabel->show();
```

```
  return myapp.exec();
}
```

6.5.2.2. Definición de variables de entorno para compilación según plataforma deseada.

Para compilar para x86 se debe ejecutar desde el directorio de inicio(home dir) el comando:

```
#source dev-x86-qpe.sh
```

Para compilar para ARM se debe ejecutar desde el directorio de inicio(home dir) el comando:

```
#source dev-arm-qpe.sh
```

6.5.2.3. Creación del archivo de proyecto - holamundo.pro

Para compilar el programa `holamundo.cpp`, es necesario crear un archivo `holamundo.pro` (project file) y pasarle la tarea de generación de archivos de compilación al programa *tmake* incluido con *Qt*. El archivo `holamundo.pro` deberá contener las siguientes líneas:

```
TEMPLATE = app
#CONFIG = qt warn_on debug
CONFIG = qt warn_on release
HEADERS =
SOURCES = holamundo.cpp
INCLUDEPATH += $(QPEDIR)/include
DEPENDPATH += $(QPEDIR)/include
LIBS += -lqpe
INTERFACES =
TARGET = holamundo
```

6.5.2.4. Compilación de holamundo.cpp

Una vez creado el archivo `holamundo.pro`, se deben ejecutar los siguientes comandos:

```
# tmake -o Makefile holamundo.pro
# make
```

Listo!

6.5.2.5. Corriendo el programa holamundo

El programa `holamundo` esta listo para ejecutarse. Si se compilo para X86 podrá correr en la PC de escritorio pero, como se mencionó en el punto 6.4.4.6 se debe utilizar un frame buffer virtual (*qvfb*). Si se compiló para ARM hay que copiar el programa ejecutable `holamundo` a la Zaurus y correrlo desde la terminal `./holamundo` La figura 6.5 muestra el programa en ejecución.

6.5.3. Otro programa multi-plataforma - qtscribble.

Programa gráfico `qtscribble.cpp`. Este programa, mas avanzado, además de que contiene un menú, permite dibujar en la pantalla con distintos colores, y también permite guardar lo que tenemos dibujado en la pantalla hacia un archivo predeterminado llamado *`test.gif`*. También permite cargar este archivo con la opción Load. Sería insteresante renombrar algun *archivo gif* con el nombre *`test.gif`* para cargarlo en la pantalla. Cabe mencionar que este programa cuenta con un control adicional de barras de scroll que permiten deslizar la pantalla a una dimensión de 1000 x 1000 pixeles aumentando así la funcionalidad de la pantalla *limitada* por default a 240 x 320 pixeles en la Zaurus.

6.5.3.1. Código fuente - qtscribble.cpp

Con la ayuda de un editor de texto se debe crear un archivo con las líneas de código que vienen en el anexo B.1.1 y llamarlo `qtscribble.cpp`.

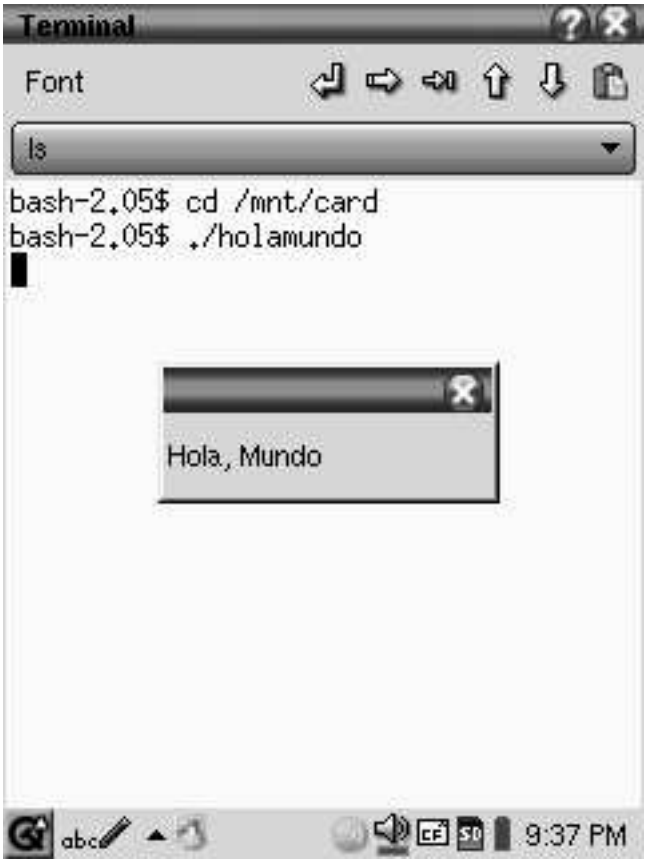

Figure 6.5: Corriendo holamundo

6.5.3.2. Definiendo variables de entorno para compilación según plataforma deseada.

Ver 6.5.2.2

6.5.3.3. Creación de archivo de proyecto - qtscribble.pro

Para compilar el programa `qtscribble.cpp`, es necesario crear primero el archivo `qtscribble.pro` (*project file*) y pasarle la tarea de generación de archivos de compilación al programa *tmake* incluído con *Qt*. Como nota importante las líneas `#CONFIG` y `CONFIG` tienen la opcion `warn_off`, para que los *warnings* NO impidan que se compile el programa. El archivo `qtscribble.pro` debe contener las siguientes líneas:

```
TEMPLATE = app
#CONFIG = qt warn_off debug
CONFIG = qt warn_off release
HEADERS =
SOURCES = qtscribble.cpp
INCLUDEPATH += $(QPEDIR)/include
DEPENDPATH += $(QPEDIR)/include
LIBS += -lqpe
INTERFACES =
TARGET = qtscribble
```

6.5.3.4. Compilación de qtscribble.cpp

Una vez creado el archivo `qtscribble.pro`, se deben ejecutar los siguientes comandos:

```
# tmake -o Makefile qtscribble.pro
# make
```

Listo!

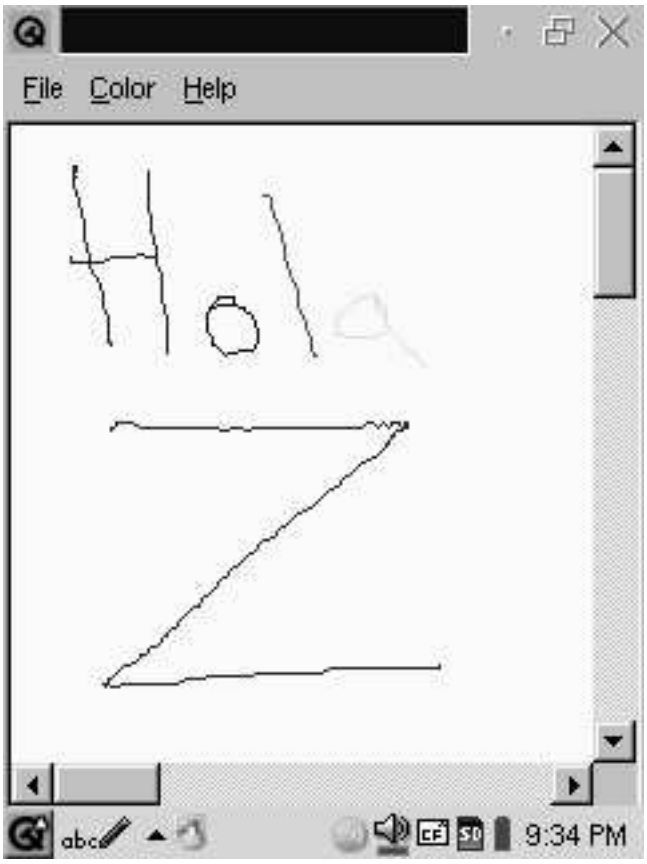

Figure 6.6: Programa Qtscribble

6.5.3.5. Ejecutando qtscribble

Para ejecutar el programa se debe copiar el programa ejecutable `qtscribble` a la Zaurus y correrlo desde la terminal `./qtscribble` La figura 6.6 muestra el programa en ejecución.

6.5.4. Programa multi-plataforma con acceso a base de datos mysql.

6.5.4.1. Creación de base de datos *courses*

Antes de crear el programa, necesitamos una base de datos llamada `courses`. Esta se puede crear desde el programa de acceso a la base de datos que se llama *`mysql`*, y desde ahí agregar la información.

En el anexo B.1.2 se listan los comandos de SQL para la creación de la base de datos *courses.db*

6.5.4.2. Código fuente dblistview.cpp

Un programa que accesa a la base datos *courses* en *MySql*. Este programa genera una vista con cuatro campos de la base de datos. El proceso de creación y compilación es el mismo que el de los programas anteriores. Con la ayuda de un editor de texto se debe crear un archivo llamado `dblistview.cpp`, y anotar dentro de éste las líneas de código que vienen en el anexo B.1.3

6.5.4.3. Definiendo variables de entorno para compilación según plataforma deseada.

Ver 6.5.2.2

6.5.4.4. Creación de project file dblistview.pro

Para compilar el programa `dblistview.cpp`, es necesario primero crear el archivo `dblistview.pro` el cual deberá contener las siguientes líneas:

```
TEMPLATE = app
#CONFIG = qt warn_off debug
CONFIG = qt warn_off release
HEADERS =
SOURCES = dblistview.cpp
INCLUDEPATH += $(QPEDIR)/include
DEPENDPATH += $(QPEDIR)/include
LIBS += -lqpe
INTERFACES =
TARGET = dblistview
```

6.5.4.5. Compilación dblistview.cpp

Una vez creado el archivo `dblistview.pro`, se deben ejecutar los siguientes comandos:

```
# tmake -o Makefile dblistview.pro
```

```
# make
```

Listo!

6.5.5. Programa para impresión de códigos de barra vía IrDa

Es mandatorio crear primero el nodo de impresión como se muestra en el punto 6.5.5. Para crear un programa en Qt que imprima archivos en el nodo creado, no se necesitan cargar las librerías para enviar archivos por IrDa, únicamente es necesario leer el archivo y redireccionarlo al nodo creado. La figura 6.7 muestra el programa en ejecución, la figura 6.8 muestra la impresora Cameo 3 de Zebra technologies, la figura 6.9 muestra la impresión generada por el programa, y el apéndice B.1.4 muestra el código fuente de este programa.

Figura 6.7: Programa para impresión de códigos de barra en Qt

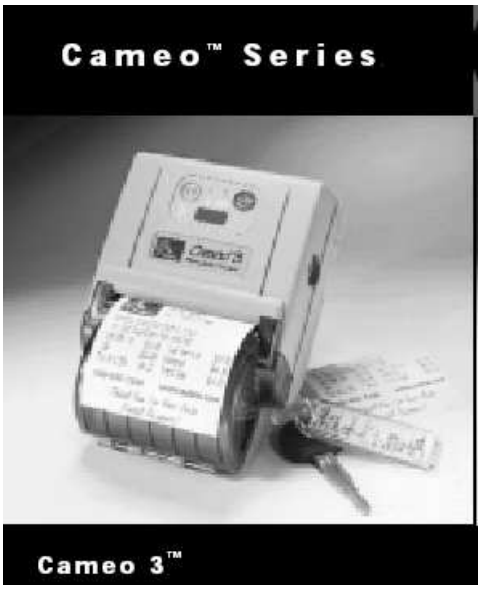

Figura 6.8: Impresora Cameo 3

Figura 6.9: Código de barra impreso

6.5.6. Programa para transmisión de archivos vía IrDa

Usar el puerto Infrarojo es tan simple como crear un objeto y llamar a la función `send` (enviar), ejemplo:

```
Ir *pruebaIr = new Ir(this,"TestIrObject");
pruebaIr->send("/proc/uptime","Uptime file","text/plain");
```

Para transmitir desde la Zaurus a través del puerto IRDA, se necesita una pequeña rutina en Qt con las instrucciones que se muestran en la apéndice B.1.5. Esta pequeña aplicación

permite la transmisión de un texto ó de un archivo.

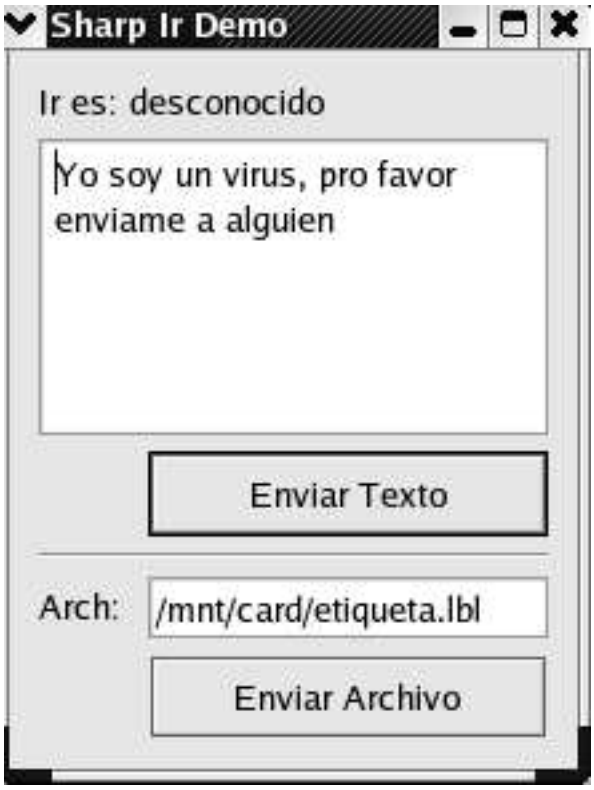

Figura 6.10: Programa impresión vía IR

Nota importante Al usar la función *send* [11] para enviar datos por puerto IrDa a otra Zaurus , es necesario configurar el *mimetype*[12] si el archivo no tiene una extensión ya que de lo contrario la transferencia IrDa interrumpirá a la Zaurus que recibe el archivo y causará que se efectué un reinicio ó reboot.

6.5.7. Instalación de aplicaciones Qt en la Zaurus

Para instalar las aplicaciones de Qt en la Zaurus es necesario crear un paquete de instalación conocido como "ipkg". Para crear el paquete `ipkg`, se debe crear la estrcutura del

[11] void send(const QString & fn, const QString & description, const QString & mimetype = QString::null)
[12] mimetype - los *mimetypes* permiter especificar las extensiones relacionadas con las aplicaciones.

directorio en la PC de escritorio, algunos archivos que son necesarios, y ubicar éstos del directorio apropiado.

6.5.7.1. El paquete ipkg

iPKG es un sistema de manejo de paquetes muy liviano. Esta diseñado para instalaciones Linux con severas limitaciones de almacenamiento como las PDAs. Mayor información acerca de `ipks` se puede encontrar en el sitio de Handhelds.org `http://www.handhelds.org`.

Un paquete `.ipk` es básicamente un archivo compactado de formato "gzip tar" que contiene los siguientes 3 miembros:

./data.tar.gz: Contiene los archivos actuales que pertenecen a este paquete. El contenido del directorio será extraído a la raíz `/` (**root dir**) cuando se instale el `ipk`. Si se requeire, debe contener entradas como `./usr` y `./etc` como directorio a nivel raíz.

./control.tar.gz: Contiene meta datos y comandos (scripts) para el paquete. Debe de contener un archivo llamado `control` (ver la sección 6.5.7.3 para detalles sobre éste archivo). Y puede contener además los siguientes archivos: `conffiles, preinst, postinst, prerm, postrm`.

./debian-binary: Este archivo actualmente es ignorado por ipkg. De cualquier manera, en todos los ipkgs actuales este es un archivo de texto con una simple línea: `2.0`

6.5.7.2. Creación de la estructura de directorios para el paquete .ipk

Para crear un paquete .ipk primero se necesita crear la estructura de directorios en la PC de escritorio con la misma estructura de directorios que existe en la Zaurus. La figura 6.11 muestra la estructura de directorios que se debe crear en la PC de escritorio.

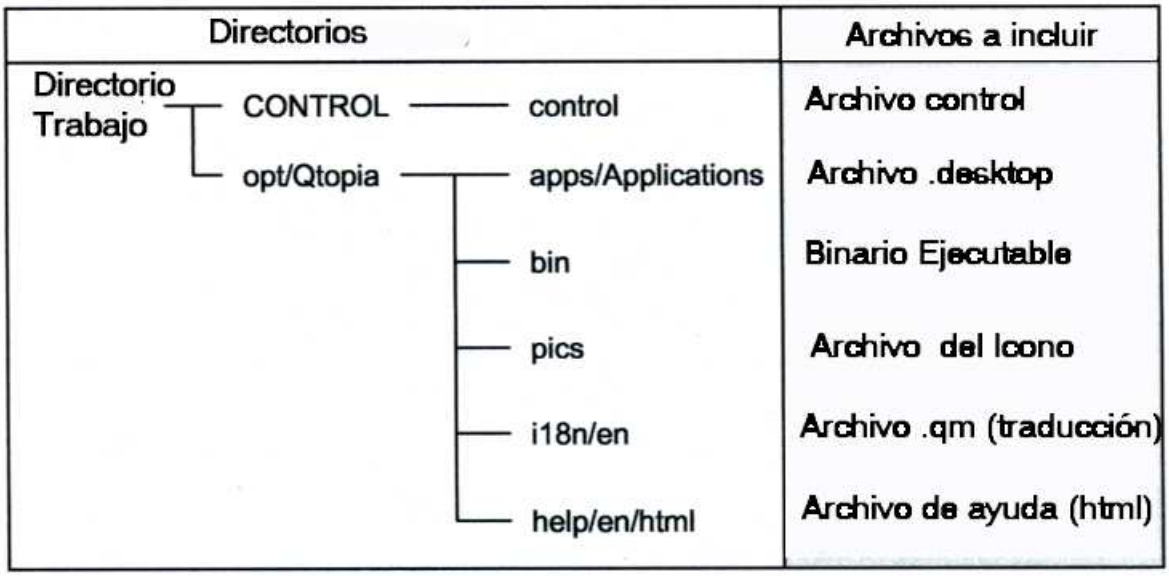

Figura 6.11: Estructura directorios para iPKG

6.5.7.3. Descripción de archivos requeridos para crear el paquete .ipk

Archivo control

El archivo de control es un archivo que describe y especifica los detalles y contenidos de un paquete ipk. En la Zaurus la aplicación para agregar y remover aplicaciones ("Add/Remove Software") usa la información en este archivo para instalar la aplicación. El cuadro 6.7 lista las entidades que componen el archivo control, las que son marcadas con "M-" significa que son **Mandatorias**.

M-Package: El nombre del paquete que debe de concordar con la expresión regular [a-z0-9.+-]\+

Installed-Size: Describe aproximadamente el tamaño del programa binario en orden de bytes.

Files: Archivos incluídos en el paquete incluyendo directorio

Priority: Esta entidad debe de contener uno entre: requiered, standard, important

Section: La categoría que mejor encaje con este tipo de paquete:

- Games
- Multimedia (Gráficos, video/audio/imágenes or reproductor)
- Communications (Mensajes instantáneos, email, etc)
- Setting (cualquiera que modifique el sistema)
- Utilities (por lo general aplicaciones pequeñas)
- Applications (Cualquiera que no entre en lo arriba mencionado)

M-Maintainer: Esta entidad debe ser el nombre y dirección de email de la persona responsable de mantener el paquete, (no necesariamente el autor del programa)

M-Architecture: Esta entidad debe especificar al menos un dígito y debe concordar con la expresión [a-zA-Z0-9.+]*. La versión puede contener revisión emparejada arrastrada "-fam![0-9]\+". Ésta revisión debe ser incrementada cada vez que el paquete cambie pero no la versión (ej. una mini modificación). Este puede ser inicializado, ó simplemente omitido, cada vez que la versión es incrementada.

Depends: Esta entidad indica paquetes adicionales que también deben ser instalados para que éste paquete trabaje. Los paquetes deben ser listados en una sola línea, separados por comas.

M-Descripción: Esta entidad debe ser una pequeña descripción (80 caracteres) del programa. También puede una descripción más larga en líneas subsecuentes, (cada una indentada por un caracter de espacio sencillo). Líneas en blanco en la descricpión larga pueden ser indicadas por una línea que consista de un caracter de espacio seguida por un punto, ej " ."

Cuadro 6.7: Archivo `control` para el paquete iPKG

En el punto 6.5.8.2 se muestra un ejemplo del archivo control para la instalación del programa de impresión via Irda.

Archivo desktop

El archivo desktop es usado para definir el icono y la información que será dada al lanzador de aplicaciones Qtopia desktop. las siguientes entidades deben ser incluídas en este archivo:

[desktop entry]

Comment = Breve explicación de ésta aplicación

Exec = Nombre del archivo del programa, ó nombre del script de ejecución

Icon = Nombre del archivo del icono de la aplicación

Type = Tipo de paquete de instalación. La mayoría de los programas deben decir "Application"

Name = Nombre de la aplicación para ser desplegada en el desktop

CanFastload = 1 para mostrar la casilla de verificación ó 0 para ocultarla al desplegar la ventana de detalles "Details" por la aplicación de iconos presiona_y_deten "tap-and-hold"

HidePrivilege = 1 para ocultar la casilla de verificación ó 0 para mostrarla al desplegar la ventana de detalles "Details" por la aplicación de iconos presiona_y_deten "tap_and_hold"

En el punto 6.5.8.2 se muestra un ejemplo del archivo desktop para la creación del paquete de instalación del programa para impresión vía IrDa.

Archivo icono

Se debe contar con un archivo que sea el icono de la aplicación en formato PNG[13]

[13] Formato PNG - Portable Network Graphics

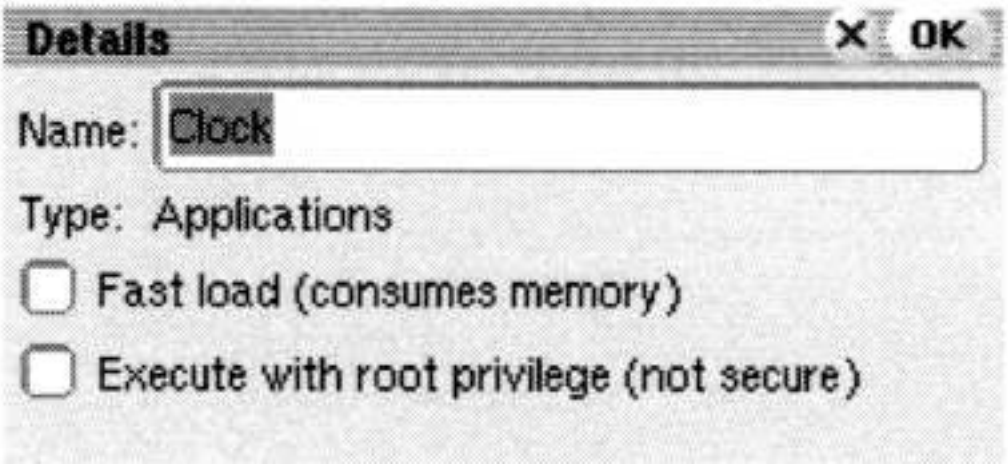

Figura 6.12: Ventana ”Details”

Archivo de ayuda

Para la ayuda del programa se debe crear un programa en html plano, de pequeñas dimensiones y preferentemente diseñado para una pantalla de 240x320 pixeles.

Archivo del programa

Por último algo obvio, el archivo del programa ejecutable compilado para ARM.

6.5.7.4. Consideraciones especiales para el rom V3

El Rom Versión 3.1 de la Zaurus incorpora una capacidad llamada “QuickExec”(Rápida ejecución) ó “FastLoad” (Cargar rápido) para minizar el tiempo requerido para invocar la aplicación.

Habilitar la opción “QuickExec” ó “FastLoad” puede ser configurado por la ventana “Details” desplegada por el icono de la aplicación tap_and_hold (presiona_y_sostén). La entidad `CanFastLoad =` puede mostrar u ocultar esta casilla. Si esta función es habilitada, la Zaurus mantendrá *en memoria cacheó de rápido acceso,* aún si la aplicación es cerrada por el usuario, pero ésto consume memoria. La figura 6.12 muestra la ventana ”Details”

6.5.7.5. Creación del paquete ipk

Una vez que se ha creado la estructura de directorios ilustrada en la subsección 6.6.4.2 y se han ubicado todos los archivos en el directorio correspondiente, se puede crear el archivo .ipk de instalación.

Existen entre otras, dos herramients de ayuda en la creación de paquetes ipkg. La primera se llama "ipkg_build" (ver punto 6.6.4.1). La segunda herramienta se llama "mkipks" y viene incluída en el SDK de Qt que se menciona en el punto 6.4.4.1. Para la creación del paquete de instalación del programa `ir.cpp` (para impresión vía IrDa) se utilizará ésta última.

Herramienta mkipks

En el punto 6.5.8 se muestra el ejemplo de la creación del paquete de instalación ipk para el programa de impresión vía IrDa.

6.5.7.6. El script ipkg

Si se necesita, el paquete puede incluir algunos scripts que serán envueltos por el sistema de mantenimiento del paquete. Existen cuatro posibles tiempos en los que el script correrá: justo antes de que sea instalado el paquete, justo después de que el paquete es instalado, justo antes de que el paquete sea removido, y justo después de que el paquete fue removido.

preinst Un archivo de script que puede ser ejecutado antes de la instalación del ipk

postinst Un archivo de script que puede ser ejecutado después de la instalación del ipk

prerm Un archivo de script que puede ser ejecutado antes de la desinstalación del ipk

postrm Un archivo de script que puede se ejecutado después de la desinstalación del ipk

Estos scripts debe de ser ubicados en el directorio CONTROL. Estos scripts deben regresar 0 si se ejecutaron con éxito (un valor distinto de cero impedirá que el paquete se instale – lo

cual puede ser útil en situaciones raras). El script no debe asumir que tty[14] está disponible de tal forma que quizá no pueda preguntarle al usuario (“hacer prompt”).

Cada script debe ser definido como cualquier otro archivo de script. El siguiente es un ejemplo de un script de post-instalación `postinst` el cual crea un directorio llamado "`data`" accesible por privilegios de usuario dentro del directorio `/home/`. Este archivo de post-instalación podría ser incluído en la aplicación *ipk* para que el directorio sea creado junto con la instalación de la aplicación. Se debe verificar que la variable PKG_ROOT este marcada hacia la raíz del paquete de instalación y se pueda usar para referir al contenido de los paquetes dentro de sus ubicaciones instaladas.

```
#!/bin/sh
mkdir /home/data
chmod +w /home/data
```

6.5.7.7. Transfiriendo archivos a la Zaurus

Existen muchas formas de tranferir archivos, entre la cuales estan:

- Copiar el paquete a una tarjeta de memoria secure digital (SD) ó compact flash (CF)
- Copiar el paquete utilizando la red con `sftp`[15] , ej:

```
sftp root@ip_zaurus   // para conectarse via ftp como root
cd /mnt/card    // cambio de dir a la tarjeta de memoria sd
put paquete.ipk   //copia el paquete
```

- El programa de sincronización que viene con la Zaurus

[14] tty se refiere a la terminal del usuario

[15] Previo a ejeuctar `sftp` se debe estar ubicado en el directorio en donde se tiene el paquete *ipk* en la PC de escritorio

6.5.7.8. Instalación del paquete ipk en la Zaurus

Se puede instalar el paquete de dos maneras, utilizando el programa para Agregar/Quitar programas *"Add/Remove Software"* que se encuentra en la pestaña *"Settings"* en Qtopia, ó de manera manual desde la terminal con los comandos:

```
$su // para obtener permisos de root
#ipkg install paquete.ipk // instala el paquete
```

6.5.8. Creación de paquete iPKG utilizando la herramienta mkipks

Esta subsección muestra un ejemplo para la creación del programa de instalación de la aplicación *ir.cpp* que se creó en el punto 6.5.5, la cual realiza la impresión de códigos de barra utilizando el puerto IrDa, en una impresora modelo Cameo3 de Zebra Technologies.

6.5.8.1. Lista de archivos para la creación de paquete ipk para la aplicación ir.cpp

ir	programa ejecutable
ir.control	Archivo control
ir.desktop	Archivo desktop
ir.html	Archivo de ayuda
Ir.png	icono del programa

A continuación se muestra el contenido de cada uno de estos.

6.5.8.2. Contenido de archivos para creación del paquete .ipk

Contenido del archivo ir.control

El archivo ir.control debe contener las siguientes líneas:

```
Files: bin/ir apps/Applications/ir.desktop pics/Ir.png help/html/ir.html
Priority: optional
```

```
Section: qpe/applications
Maintainer: Guillermo Prieto <gprieto_98@yahoo.com>
Architecture: $CPU_ARCH
Arch: $DEVICE_ARCH
Version: 1.0.0
License: Public Domain
Description: Ir barcode printing program
Un ejemplo de impresion de codigos de barra
mediante el puerto IrDa en una impresora Zebra Cameo 3.
```

Contenido del archivo desktop

El archivo ir.desktop debe contener las siguientes líneas:

```
[Desktop Entry]
Comment=Un programa de impresion de codigos de barra en Zebra Cameo 3
Exec=ir
Icon=Ir
Type=Application
Name=ir
```

Contenido del archivo ir.html

El archivo ir.html es simplemente un archivo de ayuda ó información en formato html, ejemplo:

```
<html>
<h1>ir</h1>
<p>Este es el archivo de ayuda del programa ir.
<p>Para usar esta aplicacion:
<ol>
<li>Presione el icono <img width=12 height=12 src=Ir.png> en Qtopia.
```

```
export QPEDIR=/opt/Qtopia/sharp
export QTDIR=/opt/Qtopia/sharp
export PATH=$QTDIR/bin:/usr/local/arm/bin:$PATH
export TMAKE=/opt/Qtopia/tmake/lib/qws/linux-sharp-g++
```

Table 6.8: Comandos para crear paquete ir_1.0.0_arm.ipk

```
<li>Escriba un numero de 11 digitos referente al numero de codigo
    de barra.
<li>Presione el Boton Imprimir codigo.
<li>Si desea imprimir desde un archivo. Especifique la ruta y nombre
    del mismo.
<li>Presione el Boton Imprimir desde Archivo.
</ol>
No olvide poner el puerto infrarojo en vista directa con el puerto
de la impresora Zebra Cameo 3!
```

icono del programa

El archivo Ir.png es el icono del programa 

6.5.9. Creación del paquete de instalación ir_1.0.0_arm.ipk

Para crear el paquete de instalación `ir_1.0.0_arm.ipk` utilizando la herramienta `mkipks` se debe en primer lugar ubicarse en el directorio de trabajo en el cual se creó y compiló la aplicación (`cd /dir_de_trabajo`). Se entiende que los archivos mencionados en la lista del punto 6.5.8.1 se encuentran en este mismo directorio. Después es necesario checar las variables de entorno para ARM que se mencionaron en el punto 6.4.4.3, el cuadro 6.8 muestra de nuevo estas variables. Una vez configurado todo el entorno la creación el ipk se realiza con los comandos que se muestran en el cuadro 6.9

El archivo resultante *ir_1.0.0_arm.ipk* puede ser instalado en la Zaurus!

```
su -m# permisos de root se requieren para instalar
//la opción -m preserva el entorno y no inicializa variables
cp ir.desktop $QPEDIR/apps/Applications
cp Ir.png $QPEDIR/pics
cp ir $QPEDIR/bin
cp ir.html $QPEDIR/help/html
mkipks ir.control
exit # no se necesita mas ser root
```

Table 6.9: creación del paquete ir_1.0.0_arm.ipk

6.5.9.1. Librerías requeridas por el programa *ir*

El programa `ir` requiere de la librería libqtopia.so.1, consulte el punto 6.3.2.2 referente a este archivo.

6.6. Desarrollo de aplicaciones en PersonalJava

Antes de comenzar a desarrollar en PersonalJava es necesario conocer como ejecutar el entorno Java en la Zaurus y tener en cuenta algunas consideraciones sobre el manejo del estandar de PersonalJava.

Como programar en Java está fuera del alcance de esta tesis; como referencia al respecto se pueden consultar los libros mencionados en la bibliografía: Ver [Java in a Nutshell], y [JDBC Pocket].

6.6.1. Runtime de PersonalJava en la Zaurus

El Runtime de Java viene incluido con el ROM de Qtopia, y se llama `evm`. Por ejemplo, para ejecutar el programa en Java `holajava.class` se usa la siguiente sintaxis desde la línea de comandos en la terminal de la Zaurus:

```
evm holajava holajava.class
```

ó, si la clase es pública:

```
evm holajava
```

6.6.2. Consideraciones escenciales previas al desarrollo en PersonalJava

Como se mencionó anteriormente en la sección 5.3, la Zaurus maneja el estandar de PersonalJava. Para desarrollar aplicaciones para esta plataforma es necesario tener en cuenta los paquetes incluídos en PersonalJava que son compatibles para este entorno. PersonalJava ofrece compatibilidad con J2SE en la versión 1.1, así que al compilar es necesario indicar como objetivo esta versión con la opcion `-target 1.1`.

6.6.2.1. Lista de paquetes PersonalJava

Los siguientes son los paquetes que están incluídos en PersonalJava. Al desarrollarse aplicaciones en Java para la PDA Zaurus se debe tener en consideración los paquetes básicos que soporta el entorno de ejecución de PersonalJava para la series SL de la Zaurus, los cuales se ilustran en el cuadro 6.10.

java.applet	java.io	java.security.spec
java.awt	java.lang	java.text
java.awt.datatransfer	java.lang.reflect	java.text.resources
java.awt.event	java.net	java.util
java.awt.image	java.security	java.util.jar
java.peer	java.security.cert	java.util.zip
java.beans	java.security.interfaces	

Cuadro 6.10: Paquetes PersonalJava

6.6.2.2. Paquetes opcionales soportados por la Zaurus

La implemetación de Jeode soporta los paquetes opcionales que se muestran en el cuadro 6.11

java.math	java.rmi.dgc	java.rmi.server
java.rml	java.rmi.registry	java.sql

Cuadro 6.11: Paquetes adicionales Jeode

6.6.2.3. JDBC un conducto a MySQL.

Más adelante, en el punto 6.6.3.3 se mostrarán aplicaciones que se conecten a MySQL usando JDBC para insertar y realizar consultas de datos utilizando el controlador JDBC correspondiente. En el punto 5.3.4 se detallan los controladores JDBC y la subsección 5.3.4.1 muestra la sintaxis y propiedades del controlador JDBC para MySQL.

Como primer instancia es necesario tener corriendo MySQL (ver punto 6.3.3.2), en segundo lugar es necesario tener registrado dentro de MySQL un usuario con ó sin contraseña, y por último es necesario conocer el nombre de la base de datos a la que se realizará la conexión.

Al utilizar el manejador de controladores JDBC para establecer una conexión se deben conocer tres cosas:

- El nombre de clase del controlador JDBC
- La sintaxis URL de la base de datos del controlador JDBC
- La información de conexión a la base de datos

Nombre de la clase del controlador JDBC

Para cargar el controlador JDBC, se utiliza el método:

```
class.forName(driverClassName)
```

Las siguientes líneas de código muestran la utilización de éste método en Java:

```
String controlador = "com.mysql.jdbc.Driver";
try {
 Class.forName(controlador).newInstance();
}
catch (exception e) {
 Systemout.println("No se puede encontrar el controlador");
 System.exit();
}
```

Definición de la URL de la base de datos

En la URL de la base de datos se especifica el nombre de la base de datos a la que se desea conectarse, en este caso la base de datos creada dentro de MySQL se llama `testJDBC`, y contiene registrado a un usuario llamado `zaurus` sin contraseña. La definición de la URL de la base de datos que se muestra a continuación almacena ésta información en una variable tipo *String* llamada *`url`*:

```
String url;
url = "jdbc:mysql://127.0.0.1/test?user=zaurus&password=";
```

Información de la conexión a MySQL con JDBC

Una vez cargado el controlador y definida la URL se puede realizar la conexión utilizando el método `getConnection()` como se muestra a continuación[16]

```
Connection con=DriverManager.getConnection(url, usr, pass)
```

6.6.3. Programación de aplicaciones en PersonalJava

El primer programa será el ejercicio tradicional `Hola PersonalJava`, de ahí se pasará a crear un applet, se continuará con dos programas con conexión a la base de datos MySql

[16]Contenido de las variables (tipo String) `usr = "zaurus", pass=""`):

y por útlimo la impresión de códigos de barra utilizando el puerto IrDa hacia una impresora portátil modelo *Cameo 3* de la compañía *Zebra Technologies.*

6.6.3.1. Primer programa Hola PersonalJava

`holapjava.java`, este programa es muy simple, crea una pequeña ventana que dice *Hola PJava*, y contiene código que se verá frecuentemente en programas de Java. Para crear un programa en Java se necesita escribir el código fuente y compilarlo. El cuadro 6.12 contiene el código fuente del programa holaPJava.

```
/************ holaPJava *************************/
import java.awt.*;
import java.awt.event.*;
class holaPJava implements ActionListener {
  void init () {
    Frame f = new Frame("Hola");
    Label label = new Label("Hola PJava en la Zaurus"
                                        ,Label.CENTER);
    Button b = new Button("Salir");
    f.setSize(240,280);
    f.setLayout( new BorderLayout() );
    b.addActionListener(this);
    f.addWindowListener( new WindowAdapter () {
      public void windowClosing(WindowEvent e) {
        System.exit(0);
      }
    });
    f.add(label, BorderLayout.CENTER);
    f.add(b, BorderLayout.SOUTH);
    f.show();
  }
  public void actionPerformed(ActionEvent e) {
    System.exit(0);
  }
  public static void main(String[] argv) {
    holaPJava hj = new holaPJava();
    hj.init();
  }
}
/************ holaPJava Fin del codigo************************/
```

Table 6.12: Código fuente programa holaPJava.java

Compilando HolaPJava.java

A diferencia de los programas en Qt, los programas en Java no se necesitan compilar para una plataforma específica, siempre y cuando se tomen en consideraciónlas clases soportadas por cada una de éstas (ver punto 6.6.2 para PersonalJava), así que el mismo compilador para una aplicación que corre en la PC de escritorio con J2SE, funciona para una aplicación

en la Zaurus con PersonalJava. El compilador de Java es el programa llamado `javac`. Para compilar el programa `holapjava.java` se utiliza la siguiente instrucción:

```
javac -target 1.1 holaPJava.java
```

La compilación del programa genera dos nuevos archivos, el programa compilado y un archivo de definición (`holaPJava.class, holaPJava$1.class`), ambos archivos se deben copiar a la Zaurus.

Transferencia de los archivos holaPJava* a la Zaurus

Para tranferir los archivos *holaPJava.class* y *holaPJava$1.class* se usa el mismo procedimiento del punto 6.5.7.7

Ejecutar el programa holaPJava.class en la Zaurus

Para correr el programa en la Zaurus desde la terminal se utiliza el comando:

```
evm holaPJava
```

La figura 6.13 muestra el programa en ejecución.

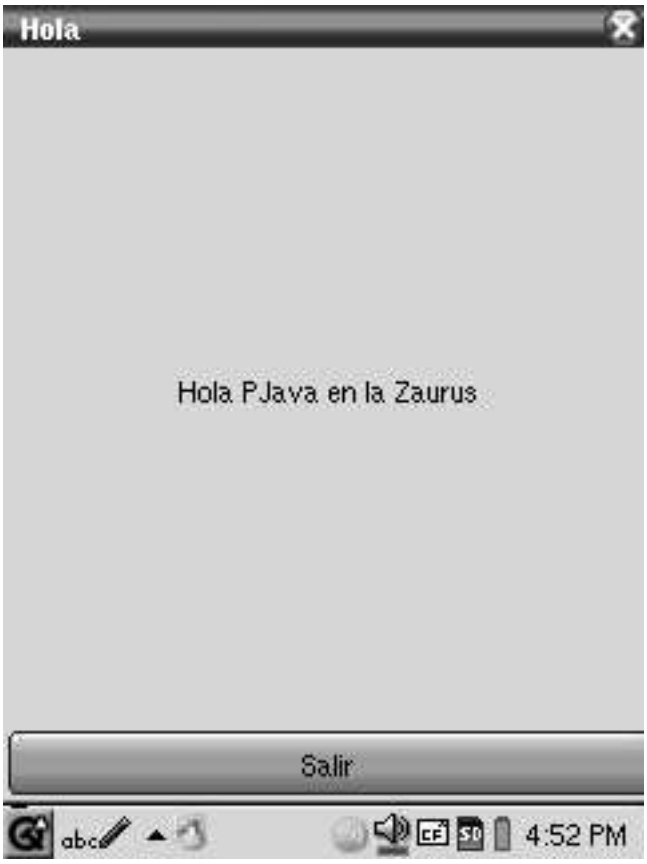

Figura 6.13: Programa holaPJava.class

6.6.3.2. Creación de un Applet llamado holaPJavaApplet.java

Un applet es una extensión de la clase java.applet.APPLET. El siguiente ejemplo es la creación de un applet similar al primer programa holaPJava con la diferencia que para éste programa se debe crear, además del programa compilado en Java, un archivo en HTML cuyo propósito será cargar el applet.

El cuadro 6.13 contiene el código fuente del applet HolaPJavaApplet. El cuadro 6.14 contiene el código del archivo en html `holaPJavaApplet.html`

```
import java.awt.Graphics;
import java.applet.Applet;
public class holaPJavaApplet extends Applet {
  public void paint ( Graphics g ) {
    g.drawString("Hola Java Applet en la Zaurus",10,50);
  }
}
```

Cuadro 6.13: Código fuente applet holaPJavaApplet.java

```
<HTML>
<HEAD>
<TITLE> Hola Applet de Java </TITLE>
</HEAD>
<BODY>
Salida de Hola Applet de Java
<HR>
<APPLET CODE="holaPJavaApplet.class" WIDTH=230 HEIGHT=260>
</APPLET>
</BODY>
</HTML>
```

Cuadro 6.14: Código archivo HTML para el applet holaPJavaApplet.class

Compilación del Applet holaPJavaApplet

Para compilar el applet de Java se utiliza la misma instrucción que se usa para compilar una aplicación normal de Java .

```
javac -target 1.1 holaPJavaApplet.java
```

La compilación de este Applet genera un solo archivo `holaPJavaApplet.class`. Este archivo y el archivo en html `holaPJavaApplet.html` (que no se compila) se deben copiar a la Zaurus.

Transferencia de los archivos holaPJavaApplet* a la Zaurus

Para tranferir los archivos *holaPJavaApplet.class* y *holaPJavaApplet.html* se usa el mismo procedimiento del punto 6.5.7.7

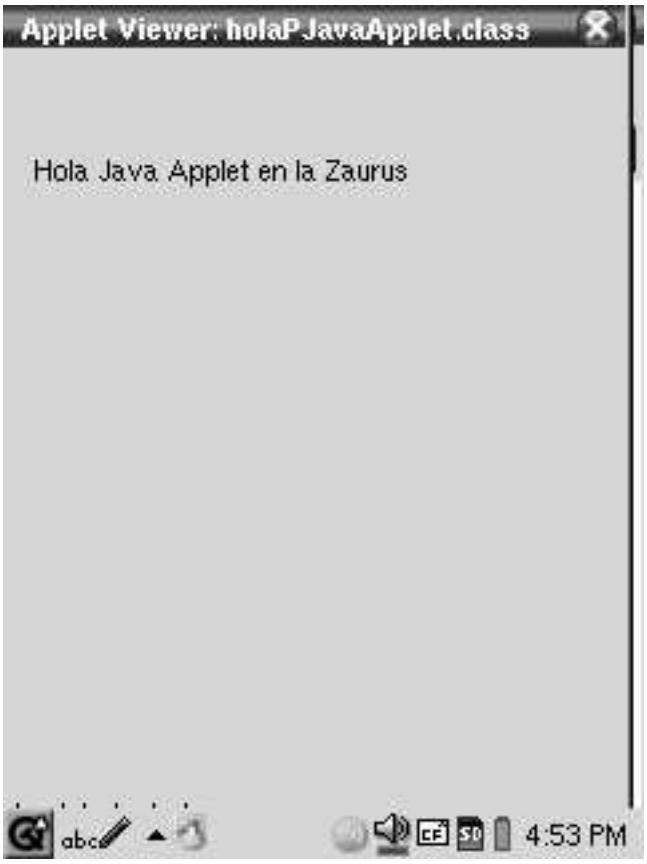

Figura 6.14: Applet holaPJavaApplet.class

Ejecución del Applet en la Zaurus

Para correr el Applet holaPJavaApplet en la Zaurus se utiliza el siguiente comando:

```
evm -appletviewer holaPJavaApplet.html
```

Aquí el archivo en html carga el Applet, y además especifica las dimensiones de la ventana. La figura 6.14 muestra el applet en ejecución.

6.6.3.3. Interacción con MySQL en PersonalJava

Esta subsección muestra programas en PersonalJava que interactuan con la base de datos Mysql en la Zaurus.

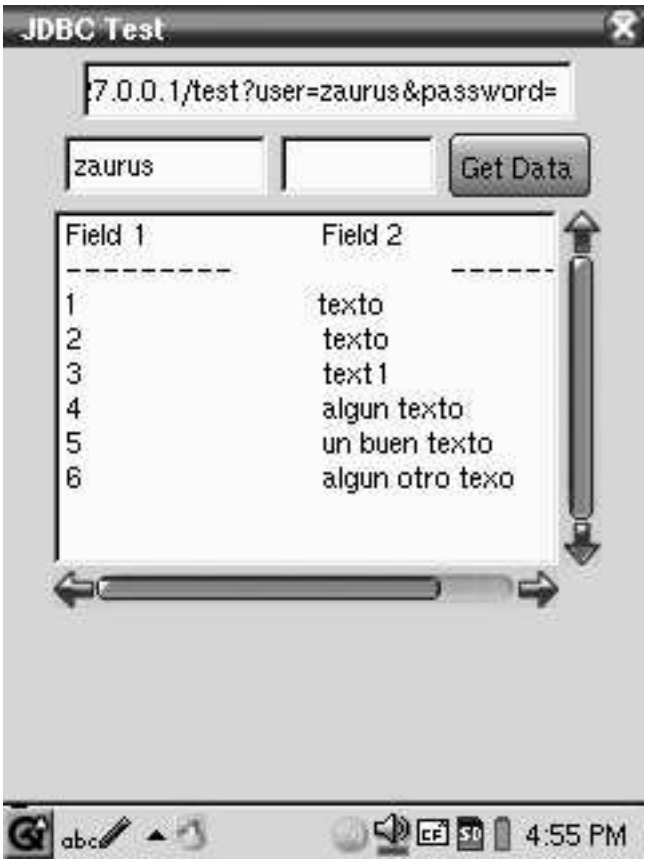

Figura 6.15: Programa JDBCTest.class

Programa *JDBCTest.java* para consultas de SQL en el servidor MySQL de la Zaurus

El programa compilado `JDBCTest.class` realiza consultas a la base de datos MySQL instalada en la Zaurus. La figura 6.15 muestra el programa en ejecución, y el apéndice B.2.1 muestra el código fuente de éste programa.

Programa JDBCTest.java para consultas de SQL en el servidor MySQL de otro equipo

El mismo programa compilado `JDBCTest.class` permite realizar consultas a la base de datos MySQL instalada en **otro** equipo en la red, para ésto, únicamente es necesario cambiar la dirección IP por la del equipo en donde se encuentra el servidor de MySQL. La figura 6.15 muestra el programa en ejecución en la Zaurus, el cual realiza una conección al servidor MySQL **de otro equipo** en la red, ubicado en la dirección IP `192.168.0.1`

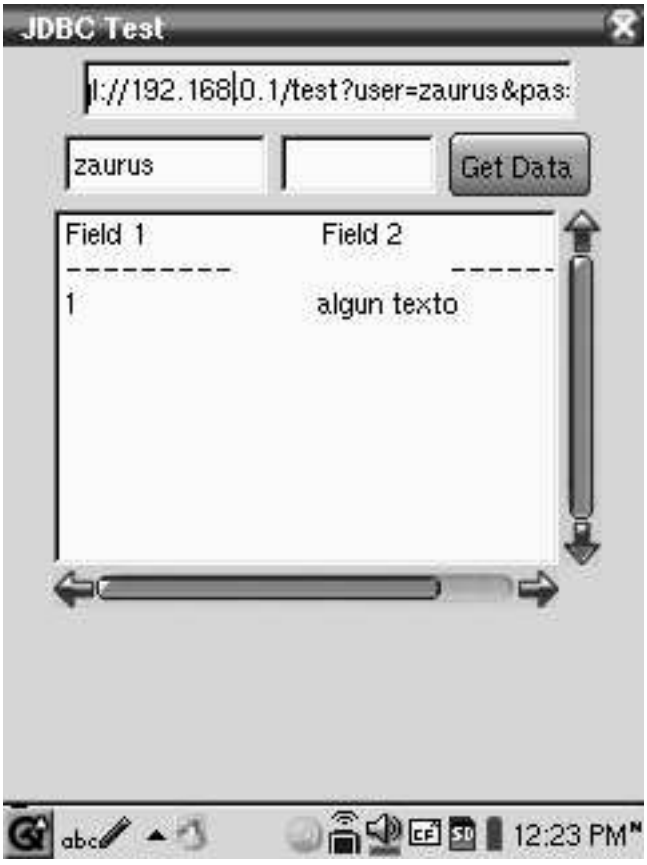

Figura 6.16: Programa JDBCTest.class en conección con otro servidor

Como ubicar el controlador JDBC al ejecutar el programa JDBCTest.class

Para ejecutar éste programa en la Zaurus es necesario utilizar el parámetro `-cp` *(class path)*, en la línea de comandos al invocar el runtime de Java: `evm`. Ésto es requerido para que el runtime de Java localize la clase que contiene el **controlador JDBC** para MySQL, y la clase que en sí, contiene al programa `JDBCTest`, ej[17]:

```
evm -cp    "/mnt/card/jdbc/mysql-connector-java-2.0.14-bin.jar:/mnt/card/java-apps" JDBCTest
```

Programa JDBCinserta.java para inserción de datos con SQL

El programa compilado JDBCinserta.class realiza la inserción de renglones en la tabla `tablaPrueba` de la base de datos `test`.

Este programa ejecuta la inserción *(Insert)* a través de una instrucción de SQL precompilada que permite ejecutar sentencias de SQL de una manera más eficiente y fácil de codificar. De una conexión se obtiene, utilizando el método *`preparedStatement()`*, el llamado `PreparedStatment` ó SQL precompilado.

Un `PreparedStatement` es más eficiente ya que se prepara una vez y se puede utilizar en múltiples ocaciones. Además resulta más fácil de codificar con éste, ya que libera al programador de escribir complejas concatenaciones de *Strings* y de usar métodos de formateo de datos propietarios para formular sentencias del SQL. En vez, se utiliza el signo de interrogación (?) para reservar un lugar para cada parámetro de entrada y programáticamente se especifican los valores para cada lugar reservado previo a la ejecución de la sentencia de SQL. La figura 6.17 muestra el programa en ejecución y el apéndice B.2.2 muestra el código fuente del programa `JDBCinserta.java`

[17] Este ejemplo asume que el controlador JDBC para MySQL se localiza en el directorio `/mnt/card/jdbc/` y que el programa `JDBCTest` se ubica en el directorio `/mnt/card/java-apps/`.

Figura 6.17: Programa JDBCinserta.class

Ejecutar el programa compilado JDBCinserta.class

La siguiente sintaxis ejecuta el programa JDBCinserta desde la terminal[18]:

```
evm -cp "/mnt/card/jdbc/mysql-connector-java-2.0.14-bin.jar:/mnt/card/java-apps" JDBCinserta
```

6.6.3.4. Programa ir.java para impresión de códigos de barra vía IrDa

Es mandatorio crear primero el nodo de impresión como se muestra en el punto 6.5.5. El programa `ir.java`, del cual se muestra el código en el apéndice B.2.3, permite imprimir etiquetas con códigos de barra especificados por el usuario a través del puerto IrDa de la Zaurus en una impresora portátil modelo Cameo-3 de Zebra Technologies Inc. La figura 6.18

[18]Este ejemplo asume que el controlador JDBC para MySQL se localiza en el directorio `/mnt/card/jdbc/` y que el programa `JDBCInserta` se ubica en el directorio `/mnt/card/java-apps/`.

muestra el programa en ejecución y la figura 6.9 muestra la impresión obtenida (la misma que se obtuvo con el programa Ir.cpp en Qt).

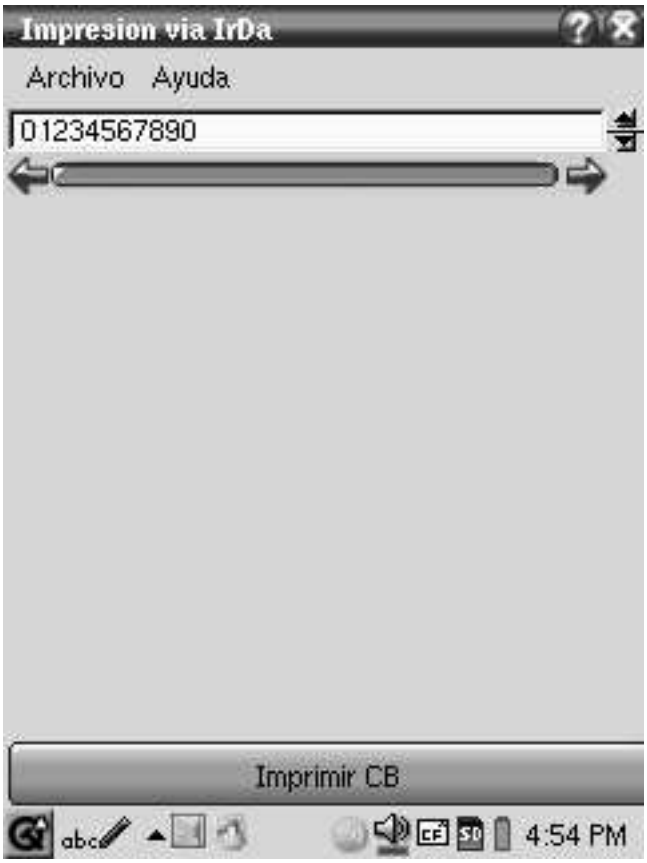

Figura 6.18: Impresión vía IrDa en PersonalJava

6.6.4. Instalación de aplicaciones PersonalJava en la Zaurus

Para instalar las aplicaciones PersonalJava en la Zaurus se debe crear el paquete ipk al igual que con las aplicaciones en Qt, solo que con algunas variantes que incluyen la creación de un script que ejecute el programa ó el Applet en Java y la reubicación del programa ejecutable dentro de otro directorio. Los siguientes pasos muestran la manera de crear un paquete ipk para instalar una aplicación en PersonalJava.

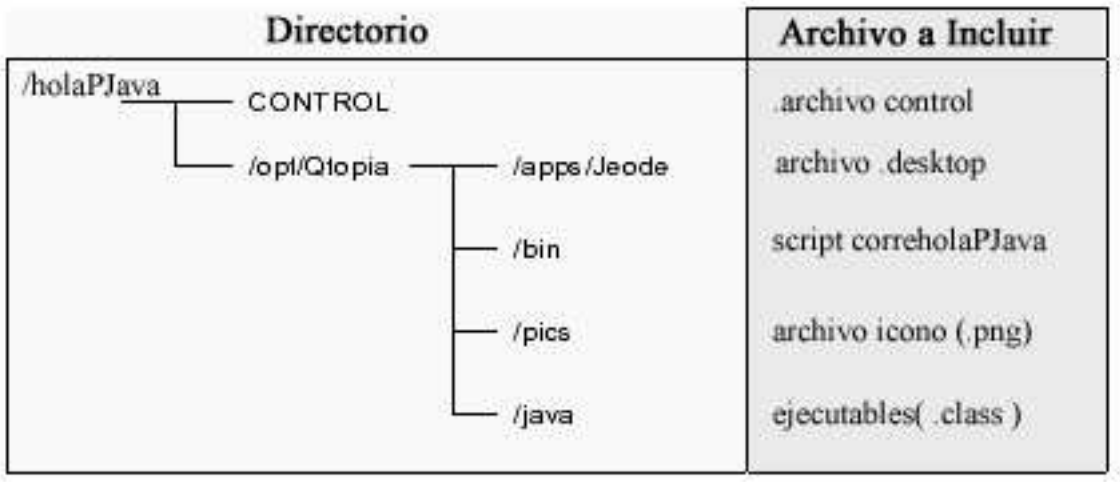

Figura 6.19: Jerarquía de directorios para Jeode

6.6.4.1. Herramienta ipkg-build

La herramienta **ipkg-build** se puede descargar de la siguiente URL: `http://ipkgfind.handhelds.ord/details.phtml?package=ipkg-build`. Una vez que se descarga ésta herramienta del Web, la cual es en realidad un *shell script*[19], se puede extraer en algún directorio que este incluído en el *PATH*[20] como p.e. `/usr/bin`. A continuación se muestra el uso básico de la herramienta:

$ ipkg-build (directorio de trabajo donde los directorios para el ipk son creados)

6.6.4.2. Creación de directorios para el paquete .ipk

Dentro del directorio de trabajo(ej: `holaPJava/`) se debe crear la jerarquía apropiada de directorios y copiar los archivos en los directorios correspondientes. La figura 6.19 muestra la estructura de directorios dentro de éste directorio de trabajo.

[19] shell script - secuencia de comandos en archivo de proceso secuencial que se ejecuta en una sesión de la terminal del sistema operativo

[20] PATH ó Ruta de búsqueda

Nombre del Archivo	Ubicación
holaPJava.control	holaPJava/CONTROL/
holaPJava.desktop	holaPJava/opt/Qtopia/apps/Jeode/
correholaPJava.sh	holaPJava/opt/Qtopia/bin/
holaPJava.png	holaPJava/opt/Qtopia/pics/
holaPJava.class, holaPJava$1.class	holaPJava/opt/Qtopia/java

Cuadro 6.15: Lista de archivos y ubicación en Jeode

6.6.4.3. Listado de archivos necesarios para crear el paquete .ipk

El cuadro 6.15 muestra la lista de archivos y el lugar donde se deben ubicar (copiar). Se asume que se desea instalar el programa `holaPJava.class` y que el directorio de trabajo es `holaPJava/`

6.6.4.4. Descripción de archivos necesarios para crear el paquete .ipk

La descripción de éstos archivos es la misma que la especificada en el punto 6.5.7.3, para la creación del paquete .ipk de una aplicación en Qt

6.6.4.5. Contenido de los archivos requeridos para crear el paquete .ipk

Archivo holaPJava.control

```
Package: java-hola
Installed-Size: 3k
FileName: ./java-hola_1.0_arm.ipk
Section: java
Version: 1.0
Architecture: 1.0
Maintainer: gprieto_98@yahoo.com
Description: Ejemplo de Prueba. Este es un simple ejemplo.
```

Archivo holaPJava.desktop

```
[Desktop Entry]
Comment=Un programa HolaPJava
Exec=correholaPJava
Icon=holaPJava
Type=Application
Name=holaPJava
HidePrivilege = 1
```

Archivo correholaPJava.sh

```
. /home/QtPalmtop/bin/installdir.sh # configura la variable INSTALLDIR
$QPEDIR/bin/evm -XappName=correholaPJava -cp $INSTALLDIR/java holaPJava
```

Archivo holaPJava.png

Este archivo es el icono del programa 

6.6.4.6. Creación del paquete holaPJava_1.0_arm.ipk con la herramienta ipkg-build

Una vez creada la estructura del directorio ilustrada en el punto 6.6.4.2, y localizados todos los archivos necesarios en el directorio correspondiente, se puede crear el paquete de instalación `holaPJava_1.0_arm.ipk` con la instrucción `ipkg-build` que se mencionó en el punto 6.6.4.1. La sintaxis será la siguiente:

```
ipkg-build holaPJava
```

6.7. Desarrollo de aplicaciones en PHP

6.7.1. Programación de aplicaciones

A diferencia de Qt ó Java, la programación en PHP se maneja en base a líneas de comandos[21] que son ejecutados en el navegador e interpretados por el motor ó, módulo de PHP configurado para el servidor web Apache. Las aplicaciones de PHP no se necesitan compilar, únicamente es necesario configurar los paŕametros de PHP en el archivo de inicio como se especificó en el punto 6.4.6.2. PHP es un módulo dentro del servidor web Apache , así que el navegador Opera en la Zaurus interpreta el HTML y puede recibir una respuesta en HTML del script de PHP una vez procesado por el módulo correspondiente, ó funcionar de manera similar a un programa cgi[22] escrito el Perl[23]. Se llaman `scripts` ya que son interpretados al momento de la ejecución ó carga de la página web.

Como programar en PHP está fuera del alcance de esta tesis. Como referencia al respecto se puede consultar la bibliografía. Ver [PHP Libro1], [PHP Libro 2]

6.7.1.1. Inicio y termino de un bloque con instrucciones en PHP

Al escribir comandos en PHP se debe informar al motor ó módulo de PHP en el servidor Apache que se le requiere que ejecute estos comandos. De no hacerse así, el código PHP que se escribe será mal interpretado como código de HTML[24] y será mostrado tal como se escribió en el navegador. El código de PHP se delimita con etiquetas especiales que marcan el inicio y el fin del bloque de código de PHP. El cuadro 6.16 muestra cuatro de éstas etiquetas delimitadoras.

De las etiquetas del cuadro 6.16 únicamente dos (*estandar* y *script*) están garantizadas para trabajar en cualquier configuración. La etiqueta *corta* se utiliza más para XML, así que si se utiliza con el motor de PHP, éste podría confundirse. Si se planea usar XML es necesario

[21] también llamados scripts
[22] cgi - common gateway interface - programa de interfaze común.
[23] perl - lenguaje de programación
[24] HTML significa Hyper Text Markup Language

Estilo de etiqueta	Etiqueta de inicio	Etiqueta de término
Etiqueta estandar	<?php	?>
Etiqueta corta	<?	?>
Etiqueta ASP[25]	<%	%>
Etiqueta script	<SCRIPT LANGUAGE='php'>	</SCRIPT>

Table 6.16: Etiquetas PHP

dehabilitar este tipo de etiqueta dentro del archivo de configuración de php (ver punto 6.4.6.2) en la variable `short_open_tag = Off;`

6.7.1.2. Primer programa PHP info

Así como en C++ se creó un primer programa llamado `holamundo.cpp`, en PHP se ilustrará un primer script que despliegue la versión de PHP que se tiene . Este script es muy sencillo, se puede nombrar `hola.php` y deberá contener las siguientes líneas:

```
<?
phpinfo();
?>
```

Ejecutar scripts de PHP

De nuevo, para ejecutar un script de PHP se debe tener instalado el servidor Apache con PHP (ver punto 6.3.4.1) y éste debe de estar corriendo (ver punto 6.3.4.2).

Primero, es necesario copiar el script deseado(ej: `hola.php`), dentro del directorio de inicio de Apache, ver punto 6.3.4.1. Segundo, desde el navegador de internet *Opera* en la Zaurus se indica en la URL la dirección IP de la Zaurus ó el hostname (ej: `localhost`), y *nombre del archivo* del script de PHP (ej: `http://localhost/hola.php`). En el punto 6.16 se mencionan los pasos para que el servidor web reconozca los archivos con extension .php de la misma manera que reconoce los archivos con extensión *.htm* y *.html*

6.7.1.3. Combinación de código de HTML y PHP en un mismo archivo

En ocaciones se puede necesitar incluir código de análisis y procesamiento en la misma forma escrita en HTML. Esta combinación puede ser útil si se necesita presentar al usuario la misma forma más de una vez. Sin duda, se puede tener más flexibilidad escribiendo toda la página dinámicamente, pero la idea es mostrar las ventajas de PHP. Entre más HTML estandar se pueda dejar en las páginas, los diseñadores de páginas web tendrán menos de que ocupares y en tanto sea posible se recomienda crear funciones que puedan ser llamadas dentro del código de HTML para que puedan ser re-usadas en otros proyectos.

El siguiente ejemplo es un script que toma un número capturado en la forma y le dice al usuario si el número que escribió es mayor ó menor a un número previamente definido.

La figura 6.20 muestra el script en ejecución. El código fuente del script se puede consultar en el apéndice B.3.1

6.7.1.4. Interacción con Mysql usando PHP

PHP es un conducto a MySQL. En esta sección se verá como conectarse a MySQL usando PHP y como insertar y seleccionar datos a través de scripts de PHP.

Conexión a MySQL con PHP

Como primer instancia es necesario tener corriendo MySQL en una ubicación a donde el navegador se pueda conectar, ver punto 6.3.3.2. En segundo lugar es necesario tener registrado dentro de MySQL un usuario con contraseña, y por último es necesario conocer el nombre de la base de datos a la que se realizará la conexión. Esto se puede hacer utilizando la herramienta PhpMyAdmin que se menciona en el punto 6.4.3.

- `mysql_connect()`

La función `mysql_connect()` es la primera función que se debe llamar cuando se utiliza un script de PHP para conectarse a MySQL. Sin una conexión abierta no se puede accesar a la base de datos. La sintaxis básica para la conexión es:

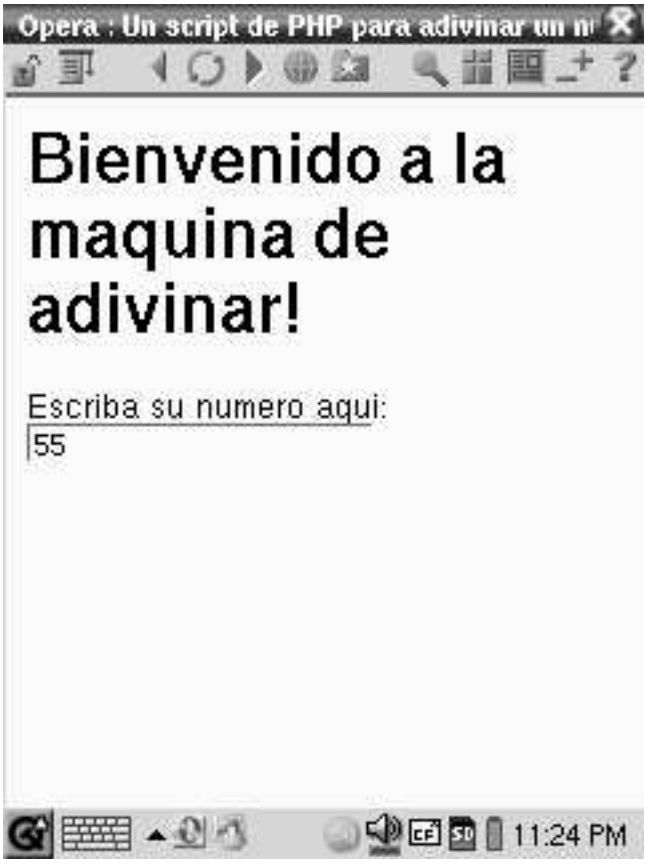

Figura 6.20: Programa adivina.php

```
<?php
$conn = mysql_connect("localhost","zaurus","");
echo $conn;
mysql_close($conn);
?>
```

Cuadro 6.17: Script PHP de conexión a MySQL

```
mysql_connect("hostname","username","password")
```

Esta función regresa un *índice de conexión* si es que ésta es exitosa y regresa `false` si la conexión falla. El cuadro 6.17 muestra un script de conexión simple, que asigna el valor del índice de conexión a una variable llamada `$conn`, y muestra el valor de ésta variable como prueba de la conexión. Se asume que existe en la base de datos MySQL de la Zaurus el usuario *"zaurus"* y que su password es *"zaurus"*. La conexión se cierra cuando termina el script pero se puede cerrar explícitamente usando la función `mysql_close()`.

Ejecución de consultas y comandos de SQL

La función `mysql_query()` en PHP es usada para enviar comandos de SQL a MySQL. Si el comando SQL resulta existoso, un índice de resultados es regresado. Si ocurre un error la función regresa *falso*. Para saber en que base de datos se va a realizar la consulta se utiliza la función `mysql_select_db()` la cual tiene la siguiente sintaxis: `mysql_select_db(database name, connection index)`; entonces, para conectarse a la base de datos llamada *`testDB`*, primero se usa `mysql_connect()`, después `mysql_select_db()` y, para ejecutar el comando de SQL `mysql_query()`. El cuadro 6.18 muestra un script de PHP el cual crea la tabla *`tablaPrueba`* dentro de la base de datos *`testDB`*.

Forma para inserción de datos

Para insertar renglones en una tabla se creó una forma en HTML que se llama a si misma y que inserta renglones cada vez que se presiona el botón ***Enviar***. El cuadro 6.19 muestra el código de la forma en HTML la cual carga un script de PHP al ser presionado el botón

```
<?php
// abrir la conexión
$conn = mysql_connect(”localhost”,”zaurus”,”zaurus”);
// seleccion de la base de datos a utilizar
mysql_select_db(”BDdePrueba”,$conn);
// creación del comando de SQL
$sql = ”CREATE TABLE tabladePrueba (identif int not null primary
         key auto_increment, campoTexto varchar(75))”;
// ejecutar el comando de SQL
$result = mysql_query($sql, $conn) or die(mysql_error());
// mostrar el indentificador del resultado
echo $result;
?>
```

Cuadro 6.18: Script PHP para crear una tabla

ENVIAR; el cuadro 6.20 muestra el script PHP, el cual genera la inserción de renglones en la tabla `tabladePrueba`; y la figura 6.21 muestra la forma corriendo en el navegador Opera de la Zaurus.

```
<HTML>
<HEAD>
<TITLE>Forma Insertar</TITLE>
</HEAD>
<BODY>
<FORM ACTION=”inserta.php” METHOD=POST>
<P> Texto para agregar a la tabladePrueba:<br>
<input type=text name=”campoTexto” size=30>
<p><input type=submit name=”Enviar”
         value=”Inserta registro”></p>
</FORM>
</BODY>
</HTML>
```

Cuadro 6.19: Forma HTML para Insertar Datos

```
<?php
// abrir la conexión
$conn = mysql_connect(”localhost”,”zaurus”,””);
// seleccion de la base de datos
mysql_select_db(”BDdePrueba”,$conn);
// creación de comando SQL
$sql = ”INSERT INTO tablaPrueba values (”,’$_POST[campoTexto]’)”);
// ejecución del comando de SQL
if( mysql_query($sql, $conn)) {
echo ”registro agregado!”;
} else {
echo ”algo salió mal”;
}
?>
```

Cuadro 6.20: Script PHP para inserción de datos

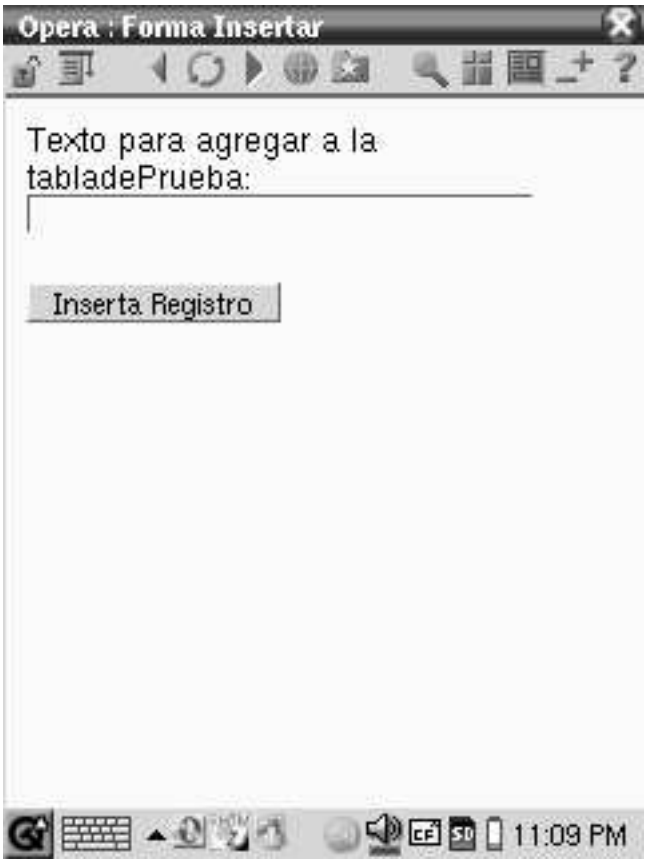

Figura 6.21: Forma Insertar Datos

Recuperando datos con PHP

Para recuperar datos se utiliza el muy conocido comando `SELECT`. Los valores recuperados por la consulta con el comando `SELECT` se almacenan en una `$variable`. La función de PHP llamada `mysql_num_rows()` regresa el número de registros obtenidos por la consulta, y la función `mysql_fetch_array($variable)` regresa el arreglo de datos contenido dentro de la `$variable`. En el apéndice B.3.2 se muestra el código fuente del script de PHP para consultar los datos en pantalla de la tabla *tablaPrueba*, y la figura 6.22 muestra el resultado de éste script en el navegador Opera de la Zaurus.

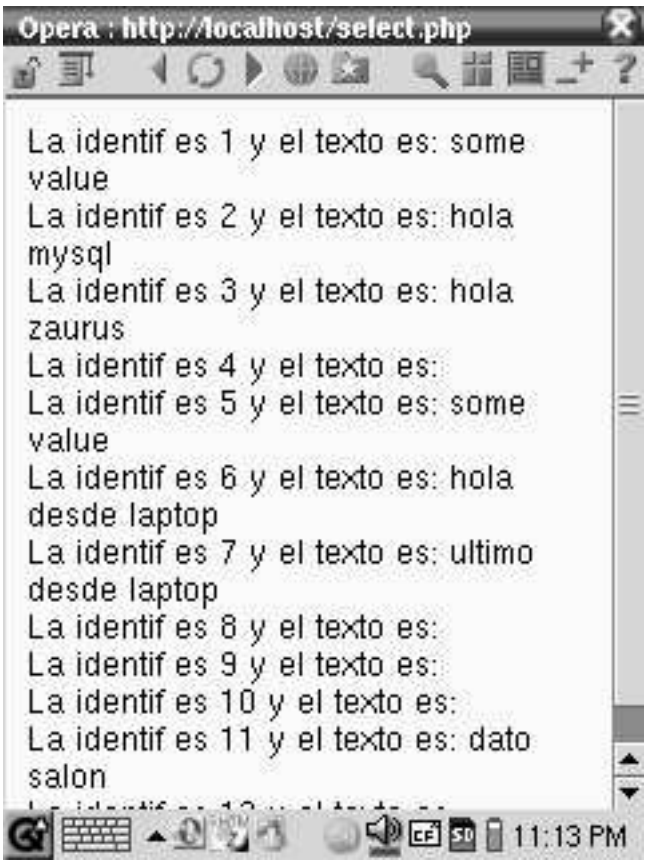

Figura 6.22: script select.php

Para obtener una lista completa de funciones se puede consultar la sección de MySQL en el manual de PHP en la siguiente URL `http://www.php.net/manual/en/ref.mysql.php`

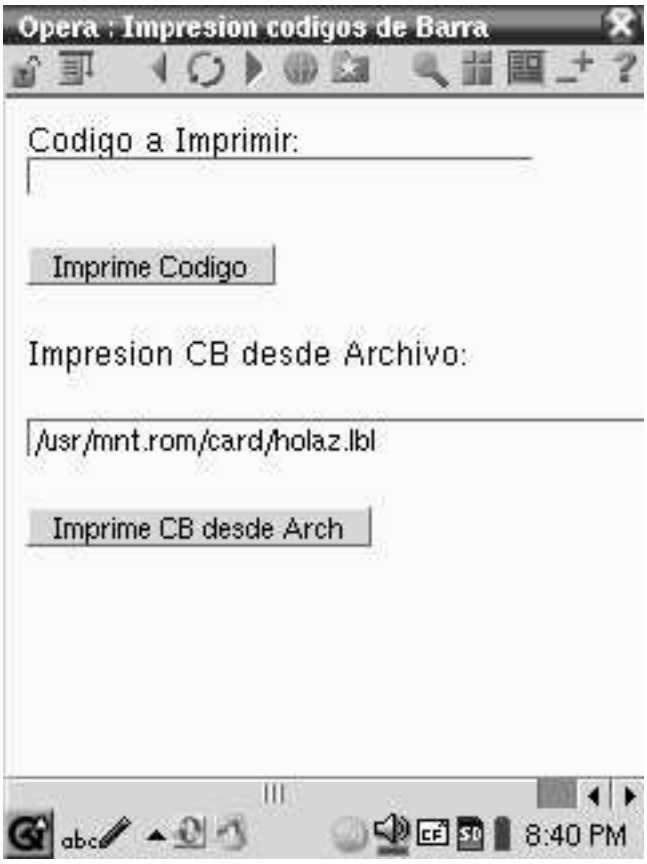

Figura 6.23: Programa ir.php

6.7.1.5. Programa de impresión de código de barras en PHP

Para imprimir desde PHP a través del puerto IrDa de la Zaurus es necesario cambiar el *owner* al nodo de impresión `/dev/irlpt0` ya que al ejecutarse scripts de PHP, el navegador Opera utiliza el usuario `nobody`, por lo que se requieren permisos de usuario para que *`nobody`* pueda realizar la impresión. El cambio de owner de éste nodo se mencionó en el punto 6.3.7, pero ésta vez se debe poner como owner al usuario `nobody`. El apéndice B.3.3 muestra el código fuente de este script y la figura 6.23 muestra el programa en ejecución.

6.7.2. Instalación de aplicaciones PHP en la Zaurus

Para instalar aplicaciones PHP la Zaurus , únicamente es necesario copiar los scripts dentro del directorio de inicio de Apache (ver punto 6.3.4.1); y para accesarlos y/o ejecutarlos[26] desde el navegador Opera en la Zaurus se debe entrar a la siguiente URL:

```
http://localhost/nombre_del_script.php
```

[26]Para ejecutar un script de PHP se requiere que el servidor Apache este corriendo, ver punto 6.3.4.2

6.8. Internacionalización (I18N) de una aplicación

6.8.1. Introducción

Internacionalización (comunmente abreviado i18n[27]) es el proceso de tomar una aplicación diseñada para una localidad y re-estructurarla para que pueda se usada en varias diferentes localidades; también se define como el proceso de crear un programa lo suficientemente flexible para que se ejecute correctamente en cualesquier localidad. Localización (comunmente abreviado l10n[28]) es el proceso de agregar soporte para una nueva *localidad* hacia una aplicación internacionalizada.

Un *localidad* es un grupo de parámetros que describe el formato de texto y modismos de lenguaje en una area en particular en el mundo. Los parámetros están divididos en seis categorías:

LC_COLLATE Estos parámetros controlan el sorteo de texto: que letras van antes y después de otras en orden alfabético.

LC_CTYPE Estos parámetros controlan el mapeo entre letras mayúsculas y minúsculas además de que caracteres caen en las diferentes clases de caracteres, como los caracteres alfanuméricos.

LC_MONETARY Estos parámetros describen el formato preferido de información de la moneda, como que caracter se usa como punto decimal y como indicar importes negativos.

LC_NUMERIC Estos parámetros describen la información del formato numérico preferido, como la forma de agrupar números y que caracter se usa como separador de miles.

LC_TIME Estos parámetros describen la información del formato preferido de tiempo y fecha, como los nombres de los meses y días, y el usar horas de tiempo de 24- ó 12-

[27]La palabra ”internacionalizacion” (en lngles *internationalization*) tiene 18 letras en la primera ”i” y la útlima ”n”.

[28]La palabra localización (en Inglés localization) tiene 10 letras en la primera ”l” y la ”n”.

LC_MESSAGES Esta categoría contiene mensajes de texto usados por aplicaciones que necesitan desplegar información en múltiples lenguajes.

Existe también una metacategoría, **LC_ALL**, que abarca todas estas categorias.

Una *localidad* representa una región cultural, política y geográfica, definida frecuentemente por un leguaje, que es representado por un código estandar de dos letras pero en ocaciones, el lenguaje solo no es suficiente para especificar una localidad única, así que el país es agregado a la especificación. El nombre de una localidad generalmente tiene tres componentes. El primero, una abreviación que indica el lenguaje, y es mandatorio, como ejemplo, *"en"* para Inglés ó *"pt"* para Portugés. Después, seguido de un guión bajo viene un especificador opcional del país, para distinguir entre dos países que hablan diferentes versiones del mismo idioma. Como ejemplo, *"en_US"* para Inglés de Estados Unidos de América y *"en_GB"* para el Inglés Británico, ó *"pt_BR"* para el Portugés Brazileño y *"pt_PR"* para el Portugés de Portugal. Finalmente, seguido de un punto, viene un especificador opcional del juego de caracteres. Como ejemplo, *"zh_TW.Big5"* para el Chino Taiwanés usando el juego de caracteres Big5. Para obtener una lista de las localiades que soporta la PC de escritorio se utiliza el comando `locale -a`.[29]

6.8.2. Configuración de la variable Language en Linux integrado

En linux integrado, específicamente en la Zaurus, se encuentra el archivo de configuración `locale.conf` en el directorio `/home/zaurus/Settings/`, este archivo cuenta con las siguientes líneas:

```
[Language]
Language = us
[Location]
Timezone = America/MexicoCity
```

[29] Este comando `locale -a` no esta incluído en Linux integrado.

De este archivo podemos obtener la variable `Language`, que si se configura en lugar de `Language = us` como `Language = de`, obtendremos el entorno de Qtopia con algunos nombres en Alemán como se muestra en la figura 6.24. Las mayoría de las aplicaciones siguen en idioma Inglés excepto por la aplicación *Wecker* (en Inglés *AlarmClock*) y la razón es que esta aplicación está programada con soporte de traducción de lenguajes (ver punto 6.8.4.2) y maneja tres idiomas: Inglés, Alemán y Ruso. En la Zaurus, dentro del directorio `/home/QtPalmtop/i18n/` se encuentran dos sub-directorios `de/` y `ru/` que contienen el archivo altamente comprimido de traducción de lenguaje de ésta aplicación, llamado en ambos casos `qpealarmclock.qm`. No existe este archivo para traducción en Inglés ya que éste es el lenguaje nativo de la aplicación. La figura 6.25 muestra la aplicación *Wecker* (en Español *Despertador*) corriendo en idioma Alemán.

Figura 6.24: Qtopia en Alemán

Figura 6.25: Wecker en Alemán

6.8.3. Aspectos para i18n de una aplicación

Dentro de la internacionalización de una aplicación existen tres aspectos que deben ser considerados:

1. Un programa debe ser capaz de leer, escribir y manipulizar texto localizado.

2. Un programa debe adaptarse a las reglas locales cuando se despliegan fechas y horas, formato de números y sorteo de cadenas de caracteres.

3. Un programa debe desplegar todo el texto visible al usuario en el lenguaje local. La traducción de mensajes que un programa despliega es siempre una de las principales tareas en la localización de una aplicación.

6.8.3.1. Que aspectos de i18n pueden cubrir las aplicaciones en la Zaurus

A través de la programación de aplicaciones, de los tres aspectos mencionados en el punto 6.8.3, en la Zaurus se pueden cubrir sólo el segundo y el tercero, y el primero de manera parcial.

Para que la programación de aplicaciones pueda cubrir el primer aspecto que es la capacidad de leer, escribir y manipulizar texto localizado en ocaciones es necesario que ésta capacidad venga desde el ROM instalado en la Zaurus que por defecto no soporta el juego de caracteres de una localidad específica como el Chino ya que los caracteres no pueden ser desplegados correctamente en pantalla (aparecen cuadros) tanto en las aplicaciones como en el navegador. En el caso del ROM en Inglés de la Zaurus, las aplicaciones pueden leer, escribir y manipular el juego de caracteres Unicode que se menciona en el punto 6.8.3.2, y existe además la posibilidad de instalar otros fonts ó tipos de caracteres como por ejemplo el Ruso. Para lenguajes como el Chino ver el punto 6.8.3.3.

6.8.3.2. Unicode en la Zaurus

En Unicode cada caracter ocupa dos bytes. Los caracteres Unicode soportados por defecto en el ROM de la Zaurus comprenden del \u0020 al \u007E los cuales son equivalentes al ASCII y a los caracteres ISO8859-1 - Latin-1 comprendidos entre el 0x20 al 0x7E; y del \u00A0 al \u00FF que son idénticos al juego de caracteres ISO8859-1 comprendidos entre el 0xA0 al 0xFF. Este juego de caracteres se puede consultar directamente en la Zaurus en las opciones de escritura de barra de tareasm seleccionando la opción *Unicode*.

6.8.3.3. Soporte en la Zaurus para el lenguaje Chino

Si se requiere utilizar un lenguaje como el Chino es necesario instalar el ROM con soporte para Chino. El ROM para la Zaurus en Chino se puede descargar del sitio `http://zaurus.cis92.net/chinese_rom/1.0`. La instalación de éste se describe el el punto 6.3.1.

6.8.4. i18n en Qt/integrado

Qt/integrado soporta completamente Unicode, el juego de caracteres internacional. Los desarrolladores pueden libremente mezclar Árabe, Inglés, Japonés, Ruso y cualquier otro lenguaje soportado por Unicode, en sus aplicaciones. Qt/integrado además ofrece herramientas para soportar traducción de aplicaciones para ayudar a la compañías a alcanzar mercados internacionales.

6.8.4.1. Unicode

Qt usa la clase **Qstring** para almacenar cadenas de caracteres Unicode. **Qstring** reemplaza el crudo `const char *;` constructores y operadores se proveen para manejar conversión entre **QString** y `const char *`. Los programadores pueden copiar QStrings por valor sin pena alguna, ya que el uso de **QString** implicitamente comparte para reducir el uso de memoria. Qt también provee **QCString** para eficientemente almacenar cadenas ASCII de caracteres.

Qt provee una poderosa máquina de interpretación de texto Unicode para todo el texto que es desplegado en pantalla, desde la más simple etiqueta hasta el mas sofisticado editor de texto. Esta interpreta la mayoría de los sistemas de escritura del mundo, incluyendo Árabe, Chino, Inglés, Griego, Japonés, Koreano, Latino y Vietnamiés.

Conversiones de y hacia diversas codificaciones y juegos de caracteres son manejados por las subclases QTextCode. La versión 3.0 de Qt soporta 37 diferentes codificaciones, incluyendo BIG5 and GBK para Chino, EUC-JP, JIS y Shift-JIS para Japonés y KOI8-R para Ruso. Pueden ser compilados como parte de las librerías ó como plugins.

6.8.4.2. Traducción de aplicaciones

Qt provee herramientas y funciones para ayudar a los desarrolladores a proveer aplicaciones en los lenguages nativos de sus clientes.

Para hacer a una cadena de caracteres traducible, simplemente se envuelve en una llamada a la función `tr()` (lee traducción)

```
saveButton->setText( tr("Save") );
```

`tr()` intenta reemplazar la cadena de caracteres literal (ej: *"Save"*) por su traducción, si es que esta se encuentra. Por ejemplo, el Inglés puede ser usado como lenguaje fuente y el Chino como lenguaje traducido ó viceversa. El argumento enviado a `tr()` es convertido al Unicode de la codificación por defecto de la aplicación.

La sintaxis general de `tr()` es:

```
Context::tr("texto fuente","comentario")
```

El contexto (*"Context"*) es el nombre de subclase de QObject. Por lo general es omitido, y en tal caso la clase que contiene la llamada a `tr()` es usado como el contexto. El *"texto fuente"* es el texto a traducir. El *"comentario"* es opcional; junto con el contexto, éste provee información adicional para traducciones humanas.

Las traducciones son almacenadas en objetos QTranslator, archivos `.qm` (Qt Message File) los cuales usan mapeo de memoria. Cada archivo .qm contiene las traducciones para un lenguaje en particular. El lenguaje puede ser cambiado al momento de la ejecución; cualesquier diálogo creado usando Qt Designer puede retraducirse a si mismo al correr sin ninguna provisión especial.

Qt provee tres herramientas para preparar archivos traducidos: *lupdate*, *Qt Linguistic* y `lrelease`.

1. lupdate extrae todo, los tres (*contexto, texto fuente, comentario*) del código fuente, incluyendo archivos *.ui* creados con *Qt Designer*, y genera un archivo *.ts* (Translation Source [30]). El archivo *.ts* es leible por humanos.

2. Los traductores usan Qt Linguistic para proveer traducciones del texto fuente en el archivo *.ts*

3. `lrelease` es la herramienta que genera archivos altamente comprimidos `.qm` los cuales son usados en dispositivos integrados como la **Zaurus**.

[30] En español: Fuente de Traducción

6.8.5. i18n en PersonalJava

Java también soporta localización en la clase `java.util.locale`. Para la tarea del manejo de reglas locales y convenciones en áreas como el formateo de fechas y horas, Java inlcuye el paquete `java.text` que define clases para ayudar con esta tarea. Estas definiciones están incluídas en la version 1.3 y mayores, y la versión 1.1 que ofrece compatibilidad con PersonalJava no las incluye, por lo que el aspecto principal de i18n que PersonalJava puede cubrir es la capacidad de desplegar el texto visible al usuario local con localización de mensajes e imágenes.

6.8.5.1. Localización de mensajes en PersonalJava

Para localización de mensajes en PersonalJava se puede utilizar el manejo de arreglos, de manera muy similar a la que se verá en localización de mensajes en PHP en el punto 6.8.7. El siguiente programa en Java `l10n.java` lee la variable `language` definida en el archivo `locale.conf` de la Zaurus y permite presentar un menú en tres idiomas: Inglés, Alemán y Español. También acepta como argumento el lenguage `sp`, `en` ó `de` con lo que el programa deja de tomar en cuenta la variable leída del archivo `locale.conf`. El apéndice C.1.1 muestra el código fuente del programa y las figuras 6.26, 6.27 y 6.28 muestran el programa en ejecución en tres idiomas; el primer gráfico muestra el programa en el idioma Alemán (se obtiene el parámetro *de* desde el archivo `locale.conf`), el segundo gráfico muestra el programa en idioma Inglés (se especificó el parámetro `en` desde la línea de comandos) y en el tercer gráfico se ejecutó con el parámetro *sp* desde la línea de comandos para obtener la versión en Español.

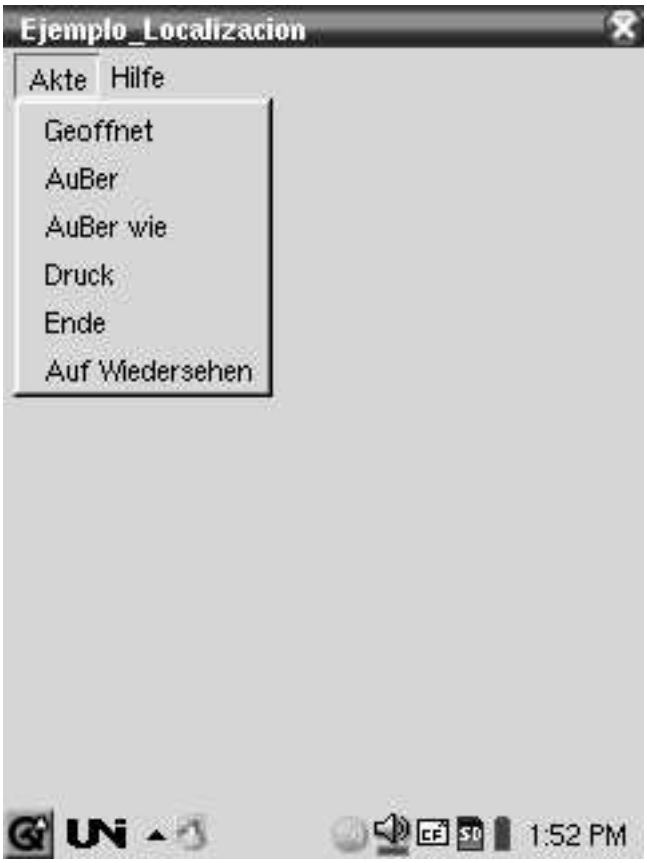

Figura 6.26: Programa l10n.class en Alemán

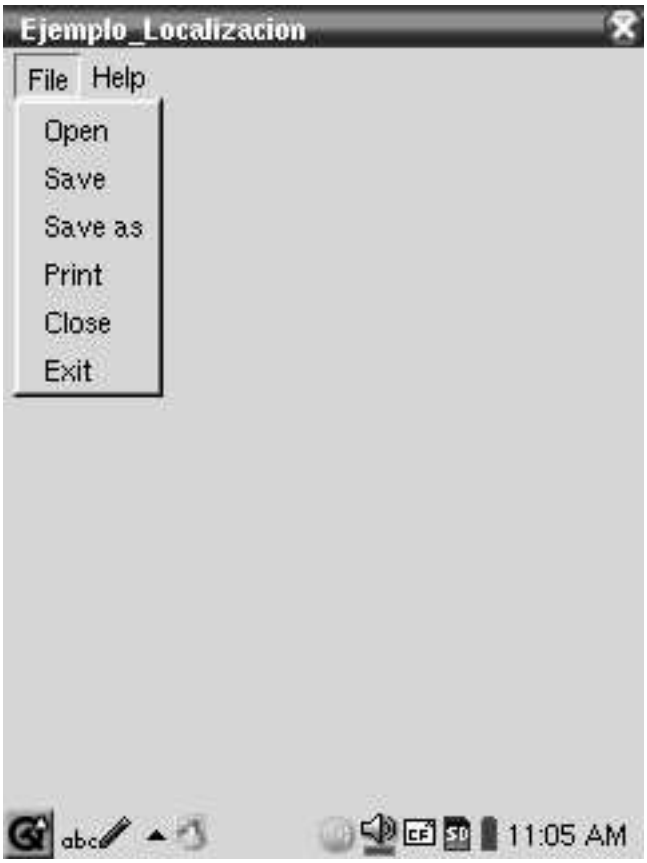

Figura 6.27: Programa l10n.class en Inglés

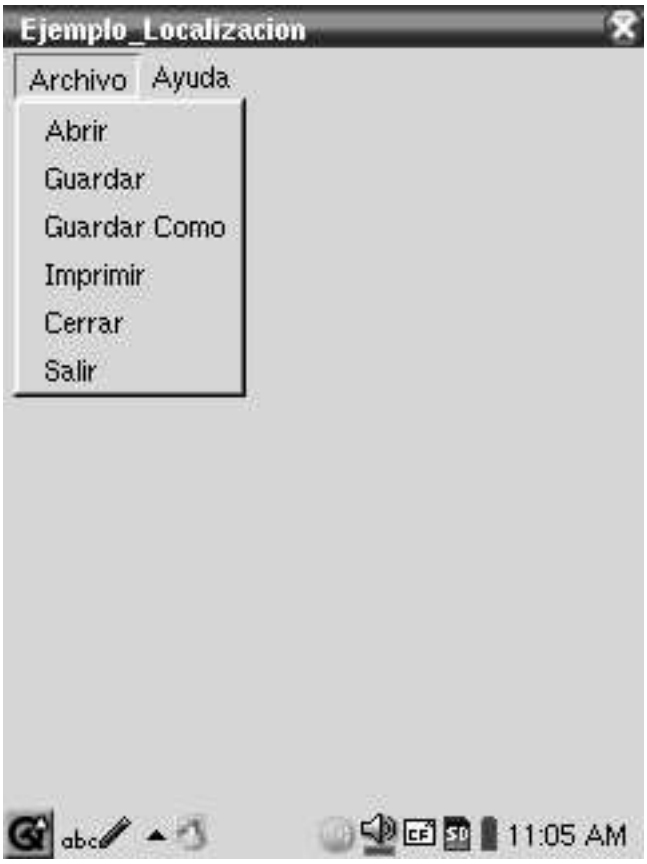

Figura 6.28: Programa l10n.class en Español

6.8.6. i18n en PHP

PHP es un intérprete que corre sobre el servidor web Apache y para accesar scripts de PHP se necesita de un navegador, en el caso de la Zaurus se cuenta con el navegador Opera. Si se requiere usar lenguajes como el Chino se necesita actualizar el ROM de la Zaurus por uno que soporte chino, y así el navegador podrá reconocer, por defecto, ese tipo de caracteres. La figura 6.29 muestra el menú de Aplicaciones de Qtopia en Chino y la figura 6.30 muestra el navegador Opera abriendo páginas en el mismo idioma.

Figura 6.29: Qtopia en Chino

Figura 6.30: Navegador en Chino

Si bien PHP no soporta todos los lenguajes, si se puede hacer lozalización de mensajes e imágenes con aquellas localidades que son del mismo juego de caracteres ISO8859-1-Latin-1 como el Español, Inglés Americano, Inglés Británico, etc. Los siguientes puntos muestran la manera de hacer scripts de PHP con localización.

6.8.7. Localización de mensajes en PHP

El apéndice C.2.1 muestra el código de un script de PHP con localización de mensajes predefiniendo el idioma Inglés Británico en la variable `$LANG=en_GB` definida dentro del mismo script. La figura 6.31 muestra el resultado de la ejecución del script en el navegador Opera de la Zaurus.

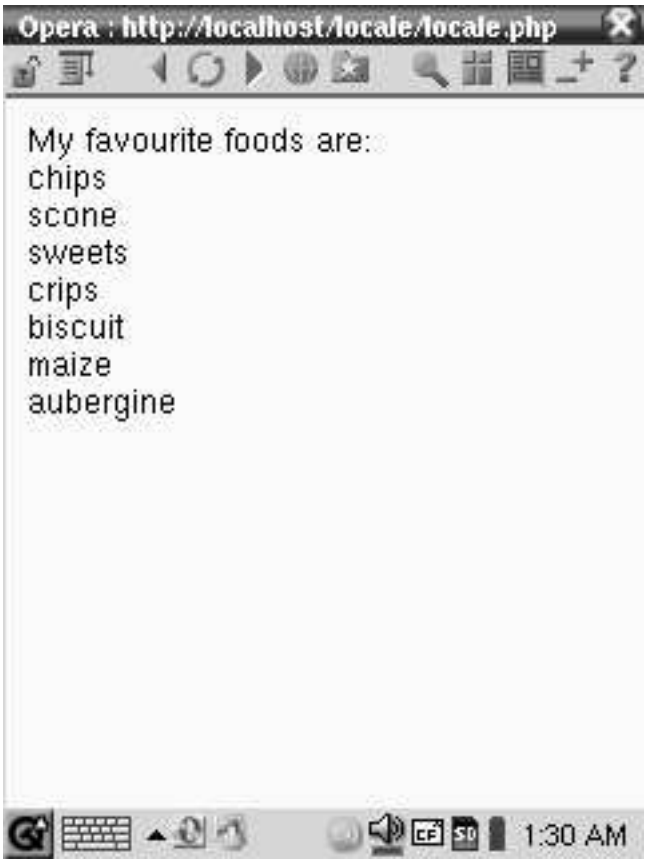

Figura 6.31: localizacion de mensajes

6.8.7.1. Localización de mensajes con la función *sprintf*.

El cuadro 6.21 muestra el código de un script de PHP con localización de mensajes con la utilización de la función *sprintf*. La figura 6.32 muestra la ejecución del script en el navegador Opera de la Zaurus.

6.8.8. Localización de imágenes

Si se desea desplegar imágenes que tienen texto dentro de ellas en un lenguaje apropiado para la localidad se requiere crear un directorio de imágenes para cada localidad que se desea soportar, además de un directorio de imágenes globales para imágenes que no tienen información local-específica dentro de ellas. Como siguiente paso se deben crear copias de cada imagen local-específica en el directorio apropiado local-específico. Las imágenes deben

```
<?php
  $mensajes = array(
    'en_US' => array('I am X years and Y months old.' =>
      'I am %d years and %d months old.'),
    'es_MX' => array('I am X years and Y months old.' =>
      'Tengo %2$d meses y %1$d a&ntildeos.') );
function msg($s) {
  global $LANG;
  global $mensajes;
  if( isset($mensajes[$LANG][$s])) {
    return $mensajes[$LANG][$s];
  } else {
    error_log("error I10n $LANG, mensajes: $s");
  }
}
$LANG = 'es_MX';
print sprintf(msg('I am X years and Y months old.'),12,7);
?>
```

Cuadro 6.21: Código del script PHP que utiliza la función sprintf

tener el mismo nombre en los diferentes directorios. El cuadro 6.22 muestra el código de una función de selección de imágenes.

6.8.9. Localización de inclusión de archivos

Para incluir archivos locales específicos en la páginas se puede modificar dinámicamente la ruta de inclusión una vez que se haya determinado la localización específica. El cuadro 6.23 muestra el código de para incluir esta ruta.

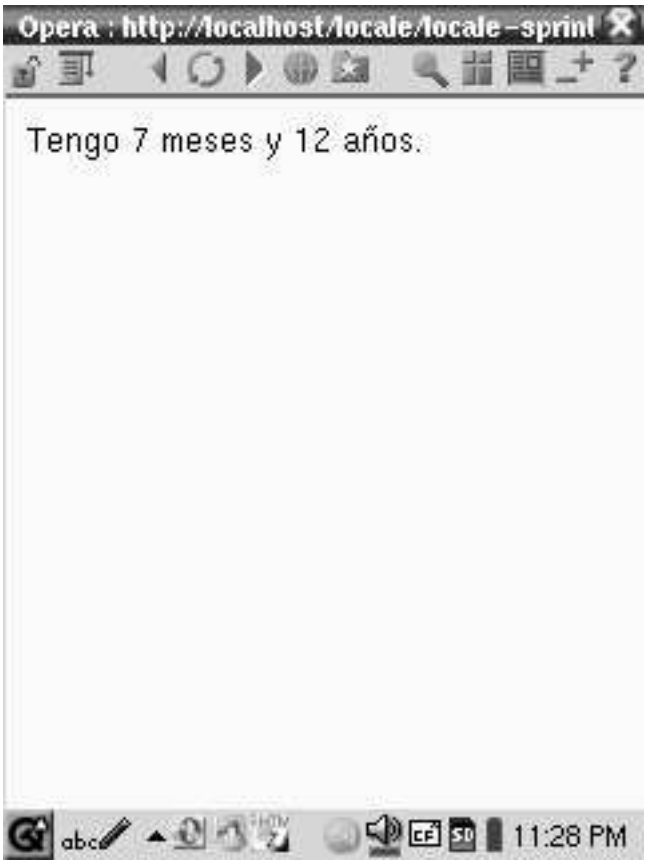

Figura 6.32: Localización de mensajes con sprintf

```
<?php
$path_base_imagenes = '/usr/local/www/imagenes';
$url_base_imagenes = '/imagenes';
function img($f) {
  global $LANG;
  global $path_base_imagenes;
  global $url_base_imagenes;
  if (is_readable("$path_base_imagenes/$LANG/$f")) {
    print "$url_base_imagenes/$LANG/$f";
  } elseif (is_readable("$path_base_imagenes/global/$f")) {
    print "$url_base_imagenes/global/$f";
  } else {
    error_log("Error i10n: LANG: $LANG, imagen: '$f'");
  }
}
?>
```

Cuadro 6.22: Código del script PHP para localización de imágenes

```
<?php
  $base = '/usr/local/php-include';
  $LANG = 'es_MX';
  $include_path = ini_get('include_path');
  ini_set('include_path',"$base/$LANG:$base/global:$include_path");
?>
```

Cuadro 6.23: Código del script de PHP para localización de inclusión de archivos

Capítulo 7

Conclusiones

En conclusión, ésta metodología explica detalladamente los pasos necesarios para el desarrollo aplicaciones para la plataforma Linux PDA, lo cual permite a los desarrolladores de software de habla hispana optimizar la curva de aprendizaje que pudiera ser de meses a tan solo unas cuantas horas, razón por la cual, podría llamarse también a ésta metodología "Aprendiendo a desarrollar para la plataforma Linux PDA en 24 horas".

Si bien es cierto que ésta metodología no abarca el desarrollo de aplicaciones para todos los modelos de PDAs con Linux, también es necesario tomar en consideración que el mayor proveedor de estos equipos y líder en el mercado de PDAs con Linux integrado es Sharp; que el entorno gráfico más utilizado y mejor soportado es Qtopia; y que además los desarrolladores y fabricantes de software como Sybase, IBM y otros apuntan hoy en día a la tecnología Java; por lo que podemos concluir que ésta metodología cubre el mayor porcentaje de las opciones para desarrollo disponibles en PDAs con Linux integrado.

Ésta metodología demuestra que no se necesita ser un *"Guru"* de Linux para utilizar una de éstas PDAs, ó para desarrollar aplicaciones móviles para las mismas. Ésta metodología es una puerta hacia aquella caja negra con Linux integrado que permite a los desarrolladores conocer éste mundo de PDAs, quienes en ocaciones por falta de conocimiento creen que por que tiene Linux es difícil ó incompatible, y peor aún, piensan que es mejor ofrecer una solución en PDAs con otros sistemas operativos. El mercado de PDAs con Linux integrado no es muy grande, y no lo será si los desarrolladores desconocen el potencial de estos equipos,

los cuales cuentan con un sistema operativo que supera por mucho a otros ofrecidos por los grandes consorcios de software.

Durante la introducción de ésta tesis se mencionó entrecomillado que Linux ofrecía en PDAs la "tercera mayor alternativa", si bien es cierto esto, se menciona entrecomillado por que Linux integrado ocupa la "tercera" posición en el mercado en cuestión de ventas de PDAs únicamente. Ésta tesis demuestra que Linux integrado en las PDAs no es la "tercera", sino la primera y mejor alternativa de sistema operativo, empezando por que es un sistema abierto, que ofrece una serie de aplicaciones desarrolladas por la comunidad de software libre que permiten a las PDAs con Linux ofrecer más servicios a menor costo. Ésta tésis ayuda en la toma de decisiones, en cuestión de selección de una PDA con Linux integrado y de las soluciones que ésta puede ofrecer. Se muestra que estos equipos además de contar con el PIM estandar de cualquier PDA, permiten la instalación de una serie de aplicaciones y servicios (la mayoría de ellos de la comunidad de software libre) que permiten entregar proyectos dentro del tiempo y presupuesto requeridos, ofreciendo ahorros siginficativos para las compañías.

Este tesis aporta un conocimiento detallado de las PDAs con Linux integrado que sirve a la comunidad en general, pero especialmente a la de desarrolladores, y les permite conocer todos los por menores de este entorno operativo en cuestión de desarrollo, con ejemplos de desarrollos en varios lenguajes como C++, Java y PHP. Representa una cooperación a la comunidad de software libre de habla hispana, siendo una guía y un llamado a los desarrolladores de estas regiones a sumarse al esfuerzo de posicionar a Linux como una alternativa distinta a las ofrecidas por los grandes monopolios del software.

Linux integrado es muy bien recibido en países como China, que intenta hacer de Linux el sistema operativo estandar en toda la nación, en Alemania, quienes adoptaron Linux en el ministerio del interior y más de quinientas agencias del gobierno, en Japón, de donde reside el mayor impulsor y fabricante de éstas PDAs - Sharp Electronics, y varios países más de Europa, Asia, Oceanía e inclusive en algunas regiones de los Estados Unidos. Ésta tesis representa entonces una estimulación e impulso al uso de PDAs con Linux integrado en México y otros países de habla hispana bajo la premisa de que si en países del primer

mundo como Alemania la implementación de Linux ha respresentado ahorros de cientos de millones de dólares americanos, en países como México en donde el presupuesto de egresos del Gobierno y las empresas es limitado, los desarrollladores y profesionales en Tecnologías de la Información estamos obligados y comprometidos con nuestra sociedad y debemos tener en mente a Linux como la primera alternativa al ofrecer soluciones tanto en el sector público como privado, para que las empresas y el gobierno tengan además de sistemas operativos de mayor calidad, ahorros muy significativos que les permitan destinar sus recursos financieros hacia otras áreas que los demandan, y no en pago de regalías por licencias, sobre todo de los servicios que se compran en pago por usuario, uno de los mas grandes rubros de ahorro con sistemas Linux. Para dar un ejemplo la instalación de la base de datos MySQL en ésta tesis no requiere del pago de regalías ni por el software ni por el número de usuarios que la accesen, limitándose éste únicamente por la capacidad del equipo (velocidad del procesador, memoria disponible, etc). Linux integrado entonces, permite adquirir equipos más potentes y de mayores características de hardware ya que se omite el costo del sistema operativo, de varias aplicaciones y servicios, así como de las herramientas de desarrollo.

Linux integrado ya es considerado por los grandes consorcios comerciales como IBM, Oracle y Sybase, quienes ofrecien hoy en día sus productos para esta plataforma. IBM ofrece aplicaciones de conexión a *WebSphere*, y Sybase, su base de datos móvil *iAnywhere*, por citar algunos ejemplos.

Bibliografía

[Alemania cambia a Linux] Fuente: http://www.dw-world.de/english/0,3367,1431_A_568696,00.html

[PHP Libro1] Julie C. Meloni: Teach Youself PHP, Mysql and Apache in 24 Hours. SAMS©, 2002.

[PHP Libro 2] David Sklar y Adam Trachtenberg: PHP cookbook. O'REILLY©, 2002.

[Qt programming] Matthias Kalle Dalheimer: Programming with Qt. Segunda edición. O'REILLY©, 2002.

[Java in a Nutshell] David Flanagan: Java Examples in a Nutshell. Tercera edición. O'REILLY©, 2004.

[JDBC Pocket] Donald Bales: JDBC Pocket Reference. Primera edición. O'REILLY©, 2003.

[JDBC y JAVA] George Reese: Programación de bases de datos con JDBC y JAVA. Anaya Multimedia (O'REILLY©), 2001

[Aprendiendo JDBC] Ashton Hobbs: Aprendiendo programación para bases de datos con JDBC. Prentice Hall, 1998

[teachyourself Java] Laura Lemay y Rogers Cadenhead: Teach Yourself JAVA 2 in 21 days. Segunda edición. SAMS, 2001.

[HTML] Jennifer Niederst: HTML Pocket Reference. Segunda edición. O'REILLY©, 2002.

[RHCE] Dave Egan: Red Hat Certified Engineer Linux Study Guide. Mc Graw Hill, 2000.

Referencia de URLs

1. Relacionadas con selección de Hardware

www.linuxdevices.com Revista de equipos con Linux que ofrece varios artículos para las PDAs.

2. Relacionadas con Qt

`http://www.tolltech.com/download/qtopia/index.html` Qtopia 1.7.0 GPLS SDK Linux for Red Hat 9

3. Relacionadas con Java

http://www.sun.com Java SDK

4. Otras URLs para descargar aplicaciones y paquetes para la Zaurus.

http://www.killefiz.de Zaurus software index

http://www.openzaurus.com Opie Rom y otras aplicaciones

http://www.rpmfind.net Un sitio para encontrar software y librerias para Linux.

http://www.myzaurus.com Actualizaciones de ROM y otras aplicaciones.

http://www.tahallah.demon.co.uk/programming/prog.html

http://www.ibiblio.org Central de Linux . Para buscar *howto's*

http://www.zaurus.com/dev/tools/qtctools/htm Cross compiler

http://www.sourceforge.net Sitio oficial de proyectos de opensource.

http://my-zaurus.narod.ru/ Aplicaciones Opie como advance filemanager, rotation

http://www.tldp.org The Linux Documentation Project - Sitio para documentación de proyectos Linux

http://www.linux-mag.com Artículos sobre PostgreSQL.

Apéndice A

Introduccion

A.1. Resumen de la Catedral y el Bazar

Resumen del ensayo **La catedral y el bazar** escrito por *Eric Random*.

Analizo un proyecto de software de dominio público desarrollado con éxito, fetchmail, que se realizó como una prueba deliberada de algunas teorías sorprendentes sobre ingeniería de software sugeridas por la historia de Linux. Presento estas teorías en términos de dos estilos de desarrollo completamente distintos, el modelo ‘”cathedral”, aplicable a la mayor parte de los desarrollos realizados en el mundo del software comercial, frente al modelo ”bazar”, más propio del mundo Linux. Muestro que estos modelos se derivan de puntos de partida opuestos sobre la naturaleza del proceso de depuración del software. Prosigo manteniendo a partir de la experiencia Linux la hipótesis de que ”Dado un número suficiente de ojos, todos los errores son irrelevantes”, sugiriendo analogías productivas con otros sistemas autocorrectores integrados por agentes autónomos, y concluyo realizando alguna exploración de las consecuencias de este punto de vista sobre el futuro del software.

El documento original en Inglés se encuentra en la siguiente URL:
http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ , y la traducción en español se puede obtener de
http://es.tldp.org/Otros/catedral-bazar/cathedral-es-paper-00.html#toc11

El resumen que a continuación se presenta, menciona los capítulos del ensayo y cita las reglas escritas en cada uno de estos.

La Catedral y el Bazar

Linux es subversivo. ¿Quién habría pensado hace trece años (1991) que un sistema operativo de clase-mundial podría unirse como por arte de magia a partir del trabajo aficionado y parcial de miles de programadores dispersos por todo el planeta, conectados solamente por los filamentos tenues del Internet?

Linux volcó mucho de lo que se pensaba. Se había predicado durante años el evangelio Unix de herramientas pequeñas, de prototipos rápidos y de la programación evolutiva. Pero también creía que existía una cierta complejidad crítica por encima de la cual era preciso recurrir a un enfoque más centralizado y planificado desde el principio. Se creía que el software más importante (los sistemas operativos o las herramientas realmente grandes tales como Emacs) necesitaban ser construidas al modo de las catedrales, ser cuidadosamente ensamblados por magos o pequeñas bandas de hechiceros trabajando en un espléndido aislamiento, sin que hubiera lugar al lanzamiento de versiones de prueba antes de que hubiera llegado el momento.

El estilo de desarrollo de Linus Torvalds - lanzar versiones de prueba enseguida y a menudo, delegar cuanto sea posible, estar abierto hasta el punto de resultar promiscuo - resultó una verdadera sorpresa. Nada que ver con la silenciosa y reverente construcción de una catedral – la comunidad Linux, por contra, parecía semejarse a un gran bazar bullicioso con diferentes agendas y enfoques (adecuadamente reflejado por los depósitos de software Linux, que admitían contribuciones de cualquiera) del cual solo parecía posible que emergiera un sistema coherente y estable mediante una sucesión de milagros.

El hecho de que este estilo bazar parecía funcionar, y bien, produjo una auténtica conmoción y era difícil comprender porqué el mundo Linux no solo no se desmoronaba en medio de una colosal confusión sino que parecía ir de logro en logro a una velocidad difícil de imaginar para los constructores de catedrales.

El mail debe llegar

Se genera en primera instancia una necesidad de un programa para manejar el correo electrónico.

1. Todo buen trabajo de software comienza rasguñando una inquietud personal del desarrollador.

2. Los buenos programadores saben que escribir. Los grandes saben que re-escribir (y re-usar).

3. ”Planear el desechar alguno; de cualquier manera, se hará”.

4. Si tienes la actitud correcta, interesantes problemas te encontraran.

5. Cuando se pierde interés en un programa, el último deber es ponerlo en manos de un sucesor competente.

La importancia de tener usuarios

Los usuarios son el recurso más importante para el desarrollo de software abierto

6. Tratar a tus usuarios como colaboradores es el camino menos complicado para mejorar con rapidez, y depurar eficazmente un programa

Libera Pronto, Libera continuamente

Se dice en la comunidad de software abierto que liberar pronto evita desperdicio de tiempo de trabajo de programadores en reparar errores que ya han sido arreglados por alguien mas, así que, liberando pronto se pueden todos percatar de los errores ya corregidos.

7. Libera pronto. Libera frecuentemente. Y escucha a tus usuarios.

8. Dada una base lo suficientemente grande de probadores y colaboradores, casi cualquier problema se identificará con rapidez y el arreglo será obvio para alguien.

O, menos formalmente, ”Dado bastantes globos oculares, todos los insectos de programación o errores son minimizados.(Con un número de ojos suficiente, todos los errores son naderías). Eric Random le llamo "La ley de Linus".

Cuantos globos oculares provocan Complejidad

La complejidad de tener muchos colaboradores repercute en que se descubren muchos errores.

¿ Cuando una Rosa deja de serlo ?

En este sección menciona la complejidad al entrar a modificar un código fuente de un programa que otra persona hizo, así que no siempre el tener el código fuente (una rosa) es algo fácil de asimilar. Un principio general que los buenos programadores deben tener en mente:

9. Estructuras de datos inteligentes asociadas a un código torpe funcionan mucho mejor que la alternativa opuesta.

Fred Brooks, en el Capítulo 11: "Ensñame tu código y mantén ocultas tus estructuras de datos, y me seguirás engañando. Muéstrame tus estructuras de datos y normalmente no necesitaré que me enseñes tu código: resultará evidente."

El pago de estas simples medidas fue inmediato. Desde el principio del proyecto, se obtuvieron reportes de errores de una calidad que muchos desarrolladores desearían, y muchas veces con buenos arreglos adjuntados. Lo cual lleva a lo siguiente:

10. Si se tratan a la gente que te ayuda a depurar como si fueran el recurso más valioso, responderán convirtiéndose en eso precisamente.

Eric Random obtuvo muchos colaboradores pero a finales de mayo de 1997 la lista comenzaba a perder a miembros de un máximo de cerca de 300 por una razón interesante: 'fetchmail' les funciona tan bien que no necesitaban seguir ya su evolución. Quizás ésta es parte del ciclo vital normal de un proyecto maduro en el estilo del bazar.

Popclient se convierte en Fetchmail

Eric random recibió de Harry Hochheiser el código para re-enviar correo al puerto del smtp de la máquina del cliente. Eric Random vislumbro casi inmediatamente que una confiable implementación de esta característica haría que el resto de otros medios de reparto de

correo quedaran obsoletos. Este concepto de re-enviar Smtp fue la más grande rentabilidad conseguida del consciente intento de emular los métodos de Linus.

11. Algo mejor que tener buenas ideas es reconocer las buenas ideas de los usuarios. Y en ocasiones esta última es la mejor en términos absolutos.

Otra lección generalizada para todo tipo de diseños es:

12. Con frecuencia, las soluciones más impactantes e innovadoras vienen de darse cuenta que el concepto del problema esta equivocado.

No hay que dudar en desechar características cuando se pueden hacer sin perder efectividad, Antoine de Saint-Exupory dijo:

13. '"La perfección (en diseño) no se obtiene cuando no hay nada mas que agregar, sino cuando no hay nada mas que desechar."

Ahora era el tiempo para el cambio de nombre. El nuevo diseño se renombro como fetchmail.

Fetchmail Crece

Fetchmail se convierte en una herramienta necesaria para muchos usuarios ya que el resultado fue de todas formas bastante trascendental – de hecho, fue el tipo de éxito que todo 'hacker' ansía lograr. Y eso hacía que se tuvieran que hacer estándares aún más exigentes.

Cualquier herramienta debe ser útil de una cierta manera, pero una verdadera gran herramienta nos lleva a usos nunca esperados.

15. Al escribir software que actúe como puerta de enlace ("gateway software") de cualquier clase, tomate la molestia de disturbar el flujo de datos lo menos posible y *nunca* elimines información a menos que el destinatario te fuerce a hacerlo.

Algunas lecciones más apartir de Fetchmail

El lenguaje empleado (keywords), y la seguridad del programa.

16. Cuando el lenguaje de tu programa no está cerca de volverse completo, una sintaxis más dulce puede ser su amigo.

17. La seguridad en un sistema resulta tan fiable como sea su secreto. Cuídese de pseudo-secretos.

Condiciones previas necesarias para el estilo del Bazar

Es muy necesario dar a los colaboradores un programa ejecutable junto con el código para que ensayen y lo prueben. Además el coordinador deberá de tener la capacidad para reconocer las buenas ideas de diseño de los demás.

El contexto social de la comunidad de código abierto

La historia de Unix nos debe de preparar para lo que se esta aprendiendo con Linux. Es decir, que aunque la programación sea una actividad solitaria, los auténticos logros surgen de la puesta en común de la atención y la capacidad intelectual de comunidades enteras. Aquel que dependa tan sólo de su cerebro al desarrollar un sistema va estar siempre en desventaja frente al que sepa cómo crear un ambiente abierto y en evolución en el cual la búsqueda de errores y las mejoras se confíen a cientos de personas.

18. Para resolver un problema interesante, comienza por encontrar uno que sea interesante para ti.

Respecto de la discusión de Gerald Weinberg del libro ”The Psychology Of Computer Programming” sobre "la programación sin ego", Weinberg hizo notar que en aquellos lugares en que los programadores no desarrollan un sentido de la propiedad sobre su propio código, y animan a los demás a buscar errores y posibles mejoras, el desarrollo progresa a una velocidad dramáticamente superior a la habitual.

19: Si el coordinador de un proyecto tiene a su disposición un medio de comunicación al menos tan potente como Internet, y sabe como conducir a la gente sin coaccionarla, muchas cabezas son inevitablemente mejor que una.

Epilogo: Netscape Abraza el Bazar

Es extraño sentir que realizas algo para hacer historia.

14th February 2005

El 22 de Enero de 1998 aproximadamente siete meses después de publicado La Catedral y el Bazar, Netscape Communications, Inc. anuncia planes de liberar el código de software Netscape Communicator. Eric Hahn, vice-presidente ejecutivo y jefe de tecnología de Netscape envió el siguiente correo electrónico a Eric Random ”En nombre de todos en Netscape, le quiero agradecer por ayudarnos a llegar a este punto en primer lugar. Sus ideas y escritos fueron una inspiración fundamental para nuestra decisión”.

Apéndice B

Desarrollo de aplicaciones

B.1. Código fuente programas en Qt

B.1.1. Código fuente programa qtscribble.cpp

```
#include <qapplication.h>
#include <qdialog.h>
#include <qfiledialog.h>
#include <qmenubar.h>
#include <qmessagebox.h>
#include <qpainter.h>
#include <qpixmap.h>
#include <qpopupmenu.h>
#include <qscrollview.h>
#include <qwidget.h>
#include <qcursor.h>
enum MenuIDs
{
COLOR_MENU_ID_BLACK,
COLOR_MENU_ID_RED,
```

```
COLOR_MENU_ID_BLUE,
COLOR_MENU_ID_GREEN,
COLOR_MENU_ID_YELLOW
};
/**
*A class that lets the user draw with the mouse. The
*windows knows how to redraw itlself.
*/
class ScribbleArea : public QWidget
{
Q_OBJECT // necessary beacuse ScribbleArea contains slots
public:
ScribbleArea();
~ScribbleArea();
public slots:
void setColor( QColor );
void slotLoad( const QString& );
void slotSave( const QString& );
protected:
virtual void mousePressEvent( QMouseEvent* );
virtual void mouseMoveEvent( QMouseEvent* );
virtual void paintEvent( QPaintEvent* );
virtual void resizeEvent( QResizeEvent* );
private slots:
void slotClearArea();
private:
QPoint _last;
QColor _currentcolor;
QPixmap _buffer;
```

```
QPopupMenu* _popupmenu;
};
class ScribbleWindow : public QWidget
{
Q_OBJECT
public:
ScribbleWindow();
~ScribbleWindow();
private slots:
void slotAbout();
void slotAboutQt();
void slotColorMenu( int );
void slotLoad();
void slotSave();
signals:
void colorChanged( QColor );
void load( const QString& );
void save( const QString& );
protected:
virtual void resizeEvent( QResizeEvent* );
private:
QMenuBar* _menubar;
QPopupMenu* _filemenu;
QPopupMenu* _colormenu;
QPopupMenu* _helpmenu;
QScrollView* _scrollview;
ScribbleArea* _scribblearea;
};
#include "qtscribble.moc"
```

```
/**
* The constructor. Initializes the member variables, and creates a
* pop-up menu that will pop up when the right mouse button is
* clicked
*/
ScribbleArea::ScribbleArea()
{
//initialize member variables
_currentcolor = black;
// dont blank the window before repainting
setBackgroundMode( NoBackground );
// create a popup menu
_popupmenu = new QPopupMenu();
_popupmenu->insertItem( "&Clear", this, SLOT( slotClearArea() ) );
}
/**
* The destructor.Deletes the pop-upmenu.
*/
ScribbleArea::~ScribbleArea()
{
delete _popupmenu;
}
/**
* Thsi slot sets the current color for the drawing area. It will be
* conected with the color Changed( QColor ) signal from the ScribbleWindow.
*
*/
void ScribbleArea::setColor( QColor new_color )
{
```

```
_currentcolor = new_color;
}
/**
* This slot clears the drawing area by filling the off-screen buffer with
* white , and then ncopying it over to the window.
*/
void ScribbleArea::slotClearArea()
{
// fill the off-screen buffer with plain white
_buffer.fill( white );
}
/**
* This method does the actual loading. It relies on QPixmap ( and the
* underlying I/O machinery ) to determine the filetype.
*/
void ScribbleArea::slotLoad( const QString& filename )
{
if( !_buffer.load( filename ) )
QMessageBox::warning( 0,"Load error", "Could not load file" );
repaint(); // refresh the window
}
/**
* this mehtod does the actual saving. We hard-code the file type as
* BMP. Unix users might want to replace this file type with XPM.
*/
void ScribbleArea::slotSave( const QString& filename )
{
if( !_buffer.save( filename, "XPM" ) )
QMessageBox::warning( 0, "Save error", "Could not save file " );
```

```
}
/**
* This virtual method is called when the pointer is on the window
* and the user presses the mouse button. When the right mouse button
* is pressed, it pops up a previously constructed pop-up menu. Otherwise,
* it just records the position of the mouse at the time of the click.
*/
void ScribbleArea::mousePressEvent( QMouseEvent* event )
{
if( event->button() == RightButton )
_popupmenu->exec( QCursor::pos() );
else
{
_last = event->pos(); // retrieve the coordinates from the event
}
}
/**
* This virtual method is called whenever the user moves the mouse
* while the mouse button is pressed. If we had called
* setMouseTracking( true ) before, this method would also be called
* when the mouse was moved without any button pressed. We know that
* we haven't, and thus don't have to check whether any buttons are
* pressed.
*/
void ScribbleArea::mouseMoveEvent( QMouseEvent* event )
{
// create a QPainter object for drawing onto the window
QPainter windowpainter;
// and other QPainter object for drawing into off-screen pixmap
```

```
QPainter bufferpainter;
// start painting
windowpainter.begin( this ); //this painter paints into the window
bufferpainter.begin( &_buffer ); // and this one in the buffer
// set a standard pen with the currently selected color
windowpainter.setPen( _currentcolor );
bufferpainter.setPen( _currentcolor );
// draw a line in both the window and the buffer
windowpainter.drawLine( _last, event->pos() );
bufferpainter.drawLine( _last, event->pos() );
// done with painting
windowpainter.end();
bufferpainter.end();
// remember the current mouse position
_last = event->pos();
}
/**
* This virtual method is called whenever the widget needs
* painting. Such as when it has been obscured and then revealed again.
*/
void ScribbleArea::paintEvent( QPaintEvent* event )
{
// copy the image from the buffer pixmap to the window
bitBlt( this, 0, 0, &_buffer );
}
/**
* This virtual method is called whenever the window is resized. We
* use it to make sure that the off-screen buffer is always the same
* size as the window.
```

```
* To retain the original drawing, it is first copied
* to a temporary buffer. After the main buffer has been resized and
* filled with white, the image is copied from the temporary buffer to
* the main buffer.
*/
void ScribbleArea::resizeEvent( QResizeEvent* event )
{
QPixmap save( _buffer );
_buffer.resize( event->size() );
_buffer.fill( white );
bitBlt( &_buffer, 0, 0, &save );
}
ScribbleWindow::ScribbleWindow()
{
/* The next few lines build up the menu bar. We first create the menus
* one by one, then add them to the menu bar. */
_filemenu = new QPopupMenu; // create the file menu
_filemenu->insertItem( "&Load", this, SLOT( slotLoad() ) );
_filemenu->insertItem( "&Save", this, SLOT( slotSave() ) );
_filemenu->insertSeparator();
_filemenu->insertItem( "&Quit", qApp, SLOT( quit() ) );
_colormenu = new QPopupMenu; // create de color menu
_colormenu->insertItem("B&lack", COLOR_MENU_ID_BLACK );
_colormenu->insertItem("&Red", COLOR_MENU_ID_RED );
_colormenu->insertItem("&Blue", COLOR_MENU_ID_BLUE );
_colormenu->insertItem("&Green", COLOR_MENU_ID_GREEN );
_colormenu->insertItem("&Yellow", COLOR_MENU_ID_YELLOW );
QObject::connect( _colormenu, SIGNAL( activated( int ) ),
this, SLOT( slotColorMenu( int ) ) );
```

```
_helpmenu = new QPopupMenu; // create the help menu
_helpmenu->insertItem("&About QtScribble", this, SLOT( slotAbout() ) );
_helpmenu->insertItem( "About &Qt", this, SLOT( slotAboutQt() ) );
_menubar = new QMenuBar( this ); // create a menu bar
_menubar->insertItem( "&File", _filemenu );
_menubar->insertItem( "&Color", _colormenu );
_menubar->insertSeparator();
_menubar->insertItem( "&Help", _helpmenu );
/** We create a QScrollView and ScribbleArea. The ScribbleArea will
* be managed by scroll view */
_scrollview = new QScrollView( this );
_scrollview->s<?php
$num_to_guess = 42;
$message = "";
if (!isset($_POST[guess])) {
$message = "Bienvenido a la maquina de adivinar!";
} elseif ($_POST[guess] > $num_to_guess) {
$message = "$_POST[guess] es my grande! Intente un numero menor";
} elseif ($_POST[guess] < $num_to_guess) {
$message = "$_POST[guess] es muy pequenio! Intente un numero mas grande";
} else { // must be equivalent
$message = "Bien hecho!";
}
?>
<html>
<head>
<title>Un script de PHP para adivinar un numero</title>
</head>
<body>
```

```
<h1>
<?php print $message ?>
</h1>
<form action="<?php print $_SERVER[PHP_SELF] ?>" method="POST">
Escriba su numero aqui: <input type="text" name="guess">
</form>
</body>
</html>etGeometry( 0, _menubar->height(), width(), height() - _menubar->height
_scribblearea = new ScribbleArea();
_scribblearea->setGeometry( 0, 0, 1000, 1000 );
_scrollview->addChild( _scribblearea );
QObject::connect( this, SIGNAL( colorChanged( QColor ) ), _scribblearea, SLOT(
QObject::connect( this, SIGNAL( save( const QString& ) ), _scribblearea, SLOT(
QObject::connect( this, SIGNAL( load( const QString& ) ), _scribblearea, SLOT(
}
ScribbleWindow::~ScribbleWindow()
{
}
void ScribbleWindow::resizeEvent( QResizeEvent* event )
{
/* When the whole window is resized, we have to rearrage the geometry
* in the ScribbleWindow as well. Note that the ScribbleArea does not need
* to be changed */
_scrollview->setGeometry( 0, _menubar->height(), width(), height() - _menubar-
}
void ScribbleWindow::slotAbout()
{
QMessageBox::information( this, "About QtScribble 4","This is the QtScribble 4
}
```

```
void ScribbleWindow::slotAboutQt()
{
QMessageBox::aboutQt( this, "About Qt" );
}
void ScribbleWindow::slotColorMenu( int item )
{
switch( item )
{
case COLOR_MENU_ID_YELLOW:
emit colorChanged( yellow );
break;
case COLOR_MENU_ID_RED:
emit colorChanged( darkRed );
break;
case COLOR_MENU_ID_BLUE:
emit colorChanged( darkBlue );
break;
case COLOR_MENU_ID_GREEN:
emit colorChanged( darkGreen );
break;
case COLOR_MENU_ID_BLACK:
emit colorChanged( black );
break;
}
}
/**
* This is the slot for the menu item File/Load. It opens a
* Qfiledialog to ask the user for a filename, then emits a save()
* signal with the filename as a parameter
```

```
*/
void ScribbleWindow::slotLoad()
{
/** Open a file dialog for loading. The default directory is the
* current directory, the filter *.bmp.
*/
QString filename = "test.gif";
//QString filename = QFileDialog::getOpenFileName(".","Images(*.gif)", this );
/** This is the save equivalent to slotLoad(). Again, we just ask for a
*filename and emit a signal.
*/
void ScribbleWindow::slotSave()
{
/*Open a file dialog for saving. The default directory is the
* current directory, the filter *.bmp.
*/
QString filename = "test.gif";
//QString filename = QFileDialog::getSaveFileName(".","Images(*.gif)", this);
if( !filename.isEmpty() )
emit save( filename );
}
int main( int argc, char* argv[] )
{
QApplication myapp( argc, argv );
ScribbleWindow* mywidget = new ScribbleWindow();
mywidget->setGeometry( 50, 50, 400, 400 );
myapp.setMainWidget( mywidget );
mywidget->show();
return myapp.exec();
```

}. Ejecutando el siguiente comando se tendrï¿½todo el directorio incluï¿½o en

B.1.2. Creación de la base de datos *courses*

```
### create the table structure for the course registration
#Drop the database and create it a new
DROP DATABASE courses;
CREATE DATABASE courses;
USE courses;
#Create the courses table
CREATE TABLE courses
{
id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(50) NOT NULL,
description TEXT,
start_date DATE NOT NULL,
end_date DATE NOT NULL,
location VARCHAR(50) NOT NULL
};
CREATE INDEX courses_name_index ON courses (name);
USE courses;
INSERT INTO courses
( name, description, start_date, end_date, location )
values
 ( 'HTML para Principiantes', 'Una introducción a HTML',
   '2001-10-08', '2001-10-12','Moscow, Russia' );
INSERT INTO courses
 (name, description, start_date, end_date, location )
Values
```

```
('HTML para avanzados', 'Todo lo que se debe saber de
  HTML','2002-05-03','2002-05-07','Stockholm,Sweden');
```

B.1.3. Código fuente programa qlistview.cpp

```
#include <qlistview.h>
#include <qsqldatabase.h>
#include <qsqlquery.h>
int main( int argc, char* argv[] )
{
QApplication app( argc, argv );
//Connect to the database
QSqlDatabase* courseDB = QSqlDatabase::addDatabase( "QMYSQL3" );
if( courseDB ) {
courseDB->setHostname( "localhost" );
courseDB->setDatabaseName( "courses" );
courseDB->setUserName( "zaurus" );
courseDB->setPassword( "zaurus");
if( courseDB->open() ) {
// Database could be opened
// Create a QListView as the toplevel widget.
QListView courseLV;
app.setMainWidget( &courseLV );
courseLV.addcolumn( "Name" );
courseLV.addcolumn( "Location" );
courseLV.addcolumn( "Start Date" );
courseLV.addcolumn( "End Date" );
courseLV.show();
// create a query and execute it
```

```
QSqlQuery query( "SELECT name, location, start_date,end_date FROM courses" );
if( query.isActive() ) {
while( query.next() ) {
new QListViewItem( &courseLV,
query.value(0).toString(),
query.value(1).toString(),
query.value(2).toString(),
query.value(3).toString() );
}
}
return app.exec();
} else
qDebug( "Could not open database " );
} else
qDebug( "Could not activate database driver" );
}
```

B.1.4. Código fuente programa para Impresión de códigos de barra vía IrDa

```
#include "ir.h"
#include <qlabel.h>
#include <qmultilineedit.h>
#include <qlineedit.h>
#include <qpushbutton.h>
#include <qmessagebox.h>
#include <qdatastream.h>
#include <qfile.h>
#include <qtextstream.h>
```

```
IrTest::IrTest(QWidget *parent, const char *name) : IrBase(parent, name, true)
  connect(sendTextButton, SIGNAL(clicked()), this,    SLOT(sendText()));
  connect(sendFileButton, SIGNAL(clicked()), this, SLOT(sendFile()));
  irStatus->setText("Especifique el numero del CB");
}
void IrTest::sendText(){
  if(textToSend->text().length() == 0 ||  textToSend->text().length() != 11 ){
    QMessageBox::critical(this, "Mensaje", "Por favor escriba un numero de 11
  return;
  }
  QFile b("/usr/mnt.rom/card/bc_ejemplo.lbl");
  if ( b.open(IO_ReadWrite) ) { // file opened successfull
    QTextStream u( &b ); // use a text stream
    u << "! 0 200 200 400 1\r\n";
    u << "LABEL\r\n";
    u << "CONTRAST 0\r\n";
    u << "TONE 0\r\n";
    u << "SPEED 5\r\n";
    u << "PAGE-WIDTH 600\r\n";
    u << "BAR-SENSE\r\n";
    u << ";// PAGE 0000000006000400\r\n";
    u << "T90 5 3 21 296 Hola Mundo\r\n";
    u << "T 5 3 157 89 Desde la ZAURUS\r\n";
    u << "BT 0 2 10\r\n";
    u << "B UPCA 2 1 90 164 176 " << textToSend->text() << "\r\n";
    u << "BT OFF\r\n";
    u << "BOX 24 90 129 308 1\r\n";
    u << "LINE 279 322 279 338 21\r\n";
    u << "BOX 262 314 322 350 1\r\n";
```

```
    u << "FORM\r\n";
    u << "PRINT\r\n";
    b.close();
  }
  QFile f("/usr/mnt.rom/card/bc_ejemplo.lbl");
  if ( f.open(IO_ReadOnly) ) { // arch abierto con exito
    QDataStream t( &f ); // uso de un stream para texto
    QFile p( "/dev/irlpt0" );
    if( p.open( IO_WriteOnly ) ) {
      QDataStream q( &p );
      while ( !t.eof() ) {
        q << t;
      }
      f.close();
      p.close();
      QMessageBox::critical(this, "Mensaje", "Impresion finalizada." ,QString(
    }
  }
}
void IrTest::sendFile(){
  if(!QFile::exists(fileLocation->text())){
    QMessageBox::critical(this, "Message", "El archivo no existe." ,QString("O
    return;
  }
  QFile f( fileLocation->text() );
  if ( !f.open( IO_ReadOnly ) ) {
      QMessageBox::critical(this, "Mensaje", "El archivo no se pudo abrir." ,Q
      return;
  } else {
```

```
    QDataStream t( &f );
    QFile p( "/dev/irlpt0" );
    if( p.open( IO_WriteOnly ) ) {
      QDataStream q( &p );
      while ( !t.eof() ) {
      q << t;
    }
    f.close();
    p.close();
    QMessageBox::critical(this, "Mensaje", "Impresion finalizada." ,QString("O
  }
}
}
```

B.1.5. Código fuente de programa de transmisión vía IrDa

```
#include "ir.h"
#include <qlabel.h>
#include <qmultilineedit.h>
#include <qlineedit.h>
#include <qpushbutton.h>
#include <qmessagebox.h>
#include <qfile.h>
#include <qtextstream.h>
IrTest::IrTest(QWidget *parent, const char *name) : IrBase(parent, name, true)
connect(sendTextButton, SIGNAL(clicked()), this, SLOT(sendText()));
connect(sendFileButton, SIGNAL(clicked()), this, SLOT(sendFile()));
Ir test;
if(!test.supported()){
```

```
irStatus->setText("Ir is: Not present");
this->setEnabled(false);
}
else
irStatus->setText("Ir is: Present");
}
void IrTest::sendText(){
if(textToSend->text().length() == 0){
QMessageBox::critical(this, "Message", "Please enter text" ,QString("Ok") );
return;
}
QFile f("/tmp/example.txt");
if ( f.open(IO_ReadWrite) ) { // file opened successfull
QTextStream t( &f ); // use a text stream
t << textToSend->text();
f.close();
}
Ir *file = new Ir(this, "IR");
//connect(file, SIGNAL(done ( Ir * ir )), this, SLOT(sent( Ir * ir )));
file->send("/tmp/example.txt", textToSend->text(),"text/plain");
}
void IrTest::sendFile(){
if(!QFile::exists(fileLocation->text())){
QMessageBox::critical(this, "Message", "File doesn't exist." ,QString("Ok") );
return;
}
Ir *file = new Ir(this, "IR");
//connect(file, SIGNAL(done(Ir*)), this, SLOT(sent( Ir * )));
file->send(fileLocation->text(), fileLocation->text(),"text/plain");
```

```
}
void IrTest::sent(Ir *){
QMessageBox::critical(this, "Message", "Ir sent." ,QString("Ok") );
}
```

B.1.5.1. Include ir.h

```
#ifndef IRTest_H
#define IRTest_H
#include "irbase.h"
#include <qlist.h>
#include <qpe/ir.h>
class IrTest : public IrBase {
Q_OBJECT
public:
IrTest(QWidget *parent=0, const char *name=0);
protected slots:
void sendText();
void sendFile();
void sent(Ir * ir);
private:
QList<Ir> itemsSent;
};
#endif
TEMPLATE = app
```

B.1.5.2. Archivo project file (.pro) para compliación de ir.cpp

```
#CONFIG = qt warn_on debug
CONFIG = qt warn_on release
```

```
HEADERS = ir.h
SOURCES = main.cpp ir.cpp
INCLUDEPATH += $(QPEDIR)/include
DEPENDPATH += $(QPEDIR)/include
LIBS += -lqpe
INTERFACES = irbase.ui
TARGET = ir
```

B.1.5.3. Archivo main.cpp de aplicación ir.cpp

```
#include "ir.h"
#include <qpe/qpeapplication.h>
int main(int argc, char **argv)
{
QPEApplication app(argc, argv);
IrTest window;
window.showMaximized();
return window.exec();
}
```

B.2. Código fuente de aplicaciones en Java

B.2.1. Código fuente programa JDBCTest.java

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class JDBCTest extends Frame implements ActionListener{
  TextField server, user, passwd;
```

```
Button b1;
TextArea data;
public JDBCTest(){
  super("JDBC Test");
  setSize(240, 280);
  setLayout(new FlowLayout());
  server=new TextField("jdbc:mysql://127.0.0.1/test?
                        user=zaurus&password=", 20);
  add(server);
  user=new TextField("zaurus", 8);
  add(user);
  passwd=new TextField("", 6);
  passwd.setEchoChar('*');
  add(passwd);
  b1=new Button("Get Data");
  b1.addActionListener(this);
  add(b1);
  data=new TextArea(10, 30);
  add(data);
  addWindowListener(new WindowAdapter(){
    public void windowClosing(WindowEvent e){
      dispose();
      System.exit(0);
    }
  });
  show();
}
public void getData() throws SQLException{
  String url, usr, pass;
```

```
    String f1, f2, f3;
    try{
      Class.forName("com.mysql.jdbc.Driver");
    }catch(ClassNotFoundException e){
      System.out.println("Could not find driver!");
      System.exit(1);
    }
    url=server.getText();
    usr=user.getText();
    pass=passwd.getText();
    Connection con=DriverManager.getConnection(url, usr, pass);
    Statement stm=con.createStatement();
    ResultSet rs=stm.executeQuery("SELECT identif, campoTexto
                                   FROM tablaPrueba");
    while(rs.next()){
      f1=rs.getString(1);
      f2=rs.getString(2);
      data.append("\n"+f1+"\t\t"+f2);
    }
  }
  public void actionPerformed(ActionEvent e){
    data.replaceRange("Field 1\t\tField 2",0,49);
    data.append("\n---------\t\t---------");
    try{
      getData();
    }catch(SQLException ex){
      data.append("\nError connecting to database:");
      data.append("\n"+ex.getMessage());
    }
```

```
  }
  public static void main(String args[]){
    new JDBCTest();
  }
}
```

B.2.2. Código fuente programa `JDBCInserta.java`

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class JDBCInserta extends Frame implements ActionListener{
  TextField texto;
  Button b1;
  TextArea data;
  Label etiqueta;
  int rows = 0;
  public JDBCInserta(){
    super("JDBC Inserta");
    setSize(240, 280);
    setLayout(new FlowLayout());
    TextField("jdbc:mysql://127.0.0.1/test?
              user=zaurus&password=", 20);
    etiqueta = new Label("Escriba el texto a insertar:");
    add(etiqueta);
    texto=new TextField("algun texto", 18);
    add(texto);
    b1=new Button("Inserta Datos");
    b1.addActionListener(this);
```

```
    add(b1);
    data=new TextArea(10, 30);
    add(data);
    addWindowListener(new WindowAdapter(){
    public void windowClosing(WindowEvent e){
      dispose();
      System.exit(0);
    }
  });
  show();
}
public void insertaDatos() throws SQLException{
  String usr, pass;
  String f1, f2;
  String url="jdbc:mysql://127.0.0.1/test?
              user=zaurus&password=";
  String text1 = texto.getText();
  PreparedStatement pstmt = null;
  try{
    // initialize & load driver
    Class.forName("com.mysql.jdbc.Driver");
  }catch(ClassNotFoundException e){
    System.out.println("Could not find driver!");
    System.exit(1);
  }
  Connection con=DriverManager.getConnection(url);//, usr, pass);
  try {
    pstmt = con.prepareStatement(
            "INSERT into tablaPrueba ( "+
```

```
            " identif,"+
            " campoTexto)"+
            " values ( "+
            " ”,"+
            " ?) ");
            }
  catch (SQLException e) {
    System.err.println(e);
  }
  pstmt.setString(1,text1);
  rows = 0;
  rows=pstmt.executeUpdate();
  pstmt.close();
  if( rows == 1 ) {
    data.append("\nRegistro Insertado");
  }
}
public void actionPerformed(ActionEvent e){
  data.replaceRange("Field 1\t\tField 2",0,59);
  data.append("\n---------\t\t---------");
try{
  insertaDatos();
}catch(SQLException ex){
  data.append("\nError inserting in database:"+rows+"inserted.");
  data.append("\n"+ex.getMessage());
  }
}
public static void main(String args[]){
  new JDBCInserta();
```

```
}
}
```

B.2.3. Código fuente programa Ir.java

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
final public class ir extends WindowAdapter
  implements ActionListener {
  // Main Frame
  private Frame mainFrame;
  private static TextArea area;
  //Menu items -- file --
  private Menu fileMenu;
  private MenuItem exitMI;
  private MenuItem openMI;
  private MenuItem saveMI;
  private MenuItem saveasMI;
  private MenuItem printMI;
  private MenuItem closeMI;
  //Menu items -- help --
  private Menu helpMenu;
  private MenuItem aboutMI;
  // Panel
  private Panel mainPanel;
  //Window Size
  private int windowWidth = 240;
```

```
private int windowHeight = 320;
private int titleHeight = 13;
private int taskbarHeight = 19;
private int margin = 2;
public ir (String[] menu_prin, String[] menu_arch, String[]
  menu_ayuda) {
  // Create frame
  mainFrame = new Frame("Impresion via IrDa");
  Button b = new Button("Imprimir CB");
  area = new TextArea("01234567890",1,2);
  mainFrame.setSize(240,280);
  mainFrame.setLayout( new BorderLayout() );
  b.addActionListener(this);
  mainFrame.add(area, BorderLayout.NORTH);
  mainFrame.add(b, BorderLayout.SOUTH);
  // Create menu bar
  MenuBar mb = new MenuBar();
  mainFrame.setMenuBar(mb);
  //File Menu
  fileMenu = new Menu(menu_prin[0]);
  openMI = new MenuItem(menu_arch[0]);
  saveMI = new MenuItem(menu_arch[1]);
  saveasMI = new MenuItem(menu_arch[2]);
  printMI = new MenuItem(menu_arch[3]);
  closeMI = new MenuItem(menu_arch[4]);
  exitMI = new MenuItem(menu_arch[5]); //"Exit"
  exitMI.addActionListener(this);
  fileMenu.add(openMI);
  fileMenu.add(saveMI);
```

```
    fileMenu.add(saveasMI);
    fileMenu.add(printMI);
    fileMenu.add(closeMI);
    fileMenu.add(exitMI);
    mb.add(fileMenu);
    //Help Menu
    helpMenu = new Menu(menu_prin[1]);
    aboutMI = new MenuItem(menu_ayuda[0]); //"Acerca");
    aboutMI.addActionListener(this);
    helpMenu.add(aboutMI);
    mb.add(helpMenu);
    //setup panel
    mainPanel = new Panel();
    mainFrame.add(mainPanel);
    mainPanel.setVisible(true);
    // setup frame
    mainFrame.addWindowListener(this);
    mainFrame.setSize((windowWidth - margin + 2 ), (windowHeight -
                      titleHeight - taskbarHeight - margin * 2 ));
    mainFrame.setVisible( true );
    mainFrame.setResizable(false);
    mainFrame.show();
  }
  // Event handling
  // windows event
  public void windowClosing (WindowEvent we) {
    if(we.getWindow() == mainFrame) {
      System.exit(0);
    }
```

```
}
// action event
public void actionPerformed (ActionEvent ae) {
  Object o = ae.getSource();
  if (o instanceof MenuItem) {
    MenuItem mi = (MenuItem)o;
    if (mi == exitMI) {
      System.exit(0);
    } else if (mi ==aboutMI) {
      new l10nAbout();
    } else if (mi == printMI) {
      try { convert();
      } catch (Exception e) {
        System.err.println(e);
        System.exit(1);
      }
    }
  } else if( o instanceof Button ) {
    Button bi = (Button) o;
    if( o == bi ) {
      try { convert();
      } catch (Exception e) {
        System.err.println(e);
        System.exit(1);
      }
    }
  }
}
public static String locale_en_Z() throws IOException
```

```
{
  StringTokenizer st1;
  String quote = "";
  FileReader infile = new FileReader("locale.conf");
  BufferedReader buff = new BufferedReader(infile);
  boolean eof = false;
  boolean siguiente = false;
  while( !eof ) {
    String line = buff.readLine();
    if( line == null )
      eof = true;
    else {
      if( siguiente ) {
        quote = line;
        siguiente = false;
      }
      line = line.trim();
      if( line.equals("[Language]") ) siguiente = true;
    }
  }
  buff.close();
  st1 = new StringTokenizer( quote, "=");
  st1.nextToken();
  return st1.nextToken();
}
public static void main (String args[]) {
  String[][] menu_prin = { {"Archivo","Ayuda"}, {"File","Help"},
                           {"Akte","Hilfe"} };
  String[][] menu_arch = { {"Abrir","Guardar","Guardar Como",
```

```
                          "Imprimir","Cerrar","Salir"},
                         {"Open","Save","Save as",
                          "Print","Close","Exit"},
                         {"Geoffnet","AuBer","AuBer wie",
                          "Druck","Ende","Auf Wiedersehen"} };
String[][] menu_ayuda = { {"Acerca"}, {"About"}, {"Uber"} };
int arreglo; // 0 = sp, 1 = en, 2 = de
String localiza = "sp";
try { localiza = locale_en_Z(); }
catch (IOException e) {
  System.err.println(e);
  return;
}
localiza = localiza.trim();
if( args.length == 1 ) localiza = args[0];
  arreglo = 0; // sp por defecto
  if(localiza.equals("sp")) arreglo = 0;
  else if(localiza.equals("en")) arreglo = 1;
  else if(localiza.equals("de")) arreglo = 2;
  ir foo = new ir(menu_prin[arreglo], menu_arch[arreglo],
                  menu_ayuda[arreglo]);
}
public static void convert() throws IOException,
  UnsupportedEncodingException {
  try {
    String bc;
    char[] bc_user = new char[11];
    String texto = "";
    InputStream in;
```

```
OutputStream out;
String infile = "holaz.lbl";
String outfile = "/dev/irlpt0";
int ubicacion, larea;
if( infile != null) in = new FileInputStream(infile);
  else return;
if( outfile != null) out = new FileOutputStream(outfile);
  else return;
Reader r = new BufferedReader( new
                    InputStreamReader(in));
Writer w = new BufferedWriter( new
                    OutputStreamWriter(out));
bc = area.getText();
larea = bc.length();
for( int j = 0; j < larea; j++ )
  bc_user[j] = bc.charAt(j);
  texto = "! 0 200 200 400 1\n\r";
  texto += "LABEL\n\r";
  texto += "CONTRAST 0\n\r";
  texto += "TONE 0\n\r";
  texto += "SPEED 5\n\r";
  texto += "PAGE-WIDTH 600\n\r";
  texto += "BAR-SENSE\n\r";
  texto += ";// PAGE 0000000006000400\n\r";
  texto += "T90 5 3 21 296 Hola Mundo\n\r";
  texto += "T 5 3 157 89 Desde la ZAURUS\n\r";
  texto += "BT 0 2 10\n\r";
  texto += "B UPCA 2 1 90 164 176 ";
  ubicacion = texto.length();
```

```
        texto += bc;
        texto += " \n\r";
        texto += "BT OFF\n\r";
        texto += "BOX 24 90 129 308 1\n\r";
        texto += "LINE 279 322 279 338 21\n\r";
        texto += "BOX 262 314 322 350 1\n\r";
        texto += "FORM\n\r";
        texto += "PRINT\n\r";
        char[] buffer = new char[4096];
        int len;
        while( (len = r.read(buffer)) != -1 ) {
          for( int k = 0; k < larea; k++ )
            buffer[ubicacion+k] = bc_user[k];
            w.write(buffer, 0, len);
            System.err.print(buffer[ubicacion]);
            ubicacion++;
            System.err.print(buffer[ubicacion]);
          }
          r.close();
          w.close();
        } catch (Exception e) {
          System.err.println(e);
          System.exit(1);
        }
      }
    }
```

B.3. Codigo fuente aplicaciones en Php

B.3.1. Script de PHP adivina.php

```
<?php
$num_to_guess = 42;
$message = "";
if (!isset($_POST[guess])) {
  $message = "Bienvenido a la maquina de adivinar!";
} elseif ($_POST[guess] > $num_to_guess) {
  $message = "$_POST[guess] es my grande! Intente un numero
             menor";
} elseif ($_POST[guess] < $num_to_guess) {
  $message = "$_POST[guess] es muy pequenio! Intente un numero
              mas   grande";
} else { // must be equivalent
  $message = "Bien hecho!";
}
?>
<html>
<head>
<title>Un script de PHP para adivinar un numero</title>
</head>
<body>
<h1>
<?php print $message ?>
</h1>
<form action="<?php print $_SERVER[PHP_SELF] ?>" method="POST">
Escriba su numero aqui: <input type="text" name="guess">
</form>
```

```
</body>
</html>
```

B.3.2. Código fuente script select.php

```
<?php
// abrir la conexión
$conn = mysql_connect(”localhost”,”zaurus”,”zaurus”);
// seleccionar la base de datos a utilizar
mysql_select_db(”DBdePrueba”,$conn);
// crear el comando de SQL
$sql = ”SELECT * FROM tabladePrueba”;
// ejecutar el comando de SQL
$resultado = mysql_query($sql,$conn) or die (mysql_error());
// ir a través de cada registro en el conjunto de resultados y
//desplegar el dato
while ($nuevo_arreglo = mysql_fetch_array($result)) {
  // dar un nombre a los campos
  $varIdent  = $nuevo_arreglo[‘identif‘];
  $varTexto = $nuevo_arreglo[‘campoTexto‘];
  // enviar los resultados a la pantalla
  echo ”La identif es $varIdent y el texto es $varTexto <br>”;
}
?>
```

B.3.3. Código fuente script ir.php

```
<?php
if( $_POST[submit] == "Imprime Codigo" ) {
  if( $_POST[cb_numero] != "" ) {
```

```
    $fh = fopen('/dev/irlpt0','w') or die($php_errormsg);
    fputs($fh,"! 0 200 200 400 1\r\n");
    fputs($fh,"LABEL\r\n");
    fputs($fh,"CONTRAST 0\r\n");
    fputs($fh,"TONE 0\r\n");
    fputs($fh,"SPEED 5\r\n");
    fputs($fh,"PAGE-WIDTH 600\r\n");
    fputs($fh,"BAR-SENSE\r\n");
    fputs($fh,";// PAGE 0000000006000400\r\n");
    fputs($fh,"T90 5 3 21 296 Hola Mundo\r\n");
    fputs($fh,"T 5 3 157 89 Desde la ZAURUS\r\n");
    fputs($fh,"BT 0 2 10\r\n");
    fputs($fh,"B UPCA 2 1 90 164 176 ");
    fputs($fh,"$_POST[cb_numero]\r\n");// incluye
                    // el numero del codigo de barra
    fputs($fh,"BT OFF\r\n");
    fputs($fh,"BOX 24 90 129 308 1\r\n");
    fputs($fh,"LINE 279 322 279 338 21\r\n");
    fputs($fh,"BOX 262 314 322 350 1\r\n");
    fputs($fh,"FORM\r\n");
    fputs($fh,"PRINT\r\n");
    fflush($fh);
    fclose($fh) or die($php_errormsg);
  }
} else {
  if( $_POST[submit] == "Imprime CB desde Arch" ) {
    if( $_POST[cb_arch] != "" ) {
      if( file_exists($_POST[cb_arch]) ) {
        $fr = fopen($_POST[cb_arch],'r') or die($php_errormsg);
```

```
        $cb = fread($fr,filesize($_POST[cb_arch]));
        fclose($fr);
        $fh = fopen('/dev/irlpt0','w') or die($php_errormsg);
        fputs($fh,$cb."\r\n");
        fflush($fh);
        fclose($fh) or die($php_errormsg);
      } else {
        echo "El archivo especificado no existe";
      }
    }
  }
}
?>
<HTML>
<HEAD>
<TITLE>Impresion codigos de Barra</TITLE>
</HEAD>
<BODY>
<FORM name = "form1" METHOD="POST" ACTION="
            <?php print $_SERVER[PHP_SELF] ?>" >
<P>Codigo a Imprimir:<br>
<input name="cb_numero" type=text id="cb_numero"
     size=30 value="<?php print $cb_numero ?>">
<p><input type="submit" name="submit"
          value="Imprime Codigo"></p>
<p>Impresion CB desde Archivo:</p>
<input type=text name="cb_arch" size=40
           value="/usr/mnt.rom/card/holaz.lbl">
<p><input type="submit" name="submit"
```

```
            value="Imprime CB desde Arch"></p>
    </FORM>
    </BODY>
    </HTML>
```

Apéndice C

Internacionalización de una aplicación

C.1. Localización de Mensajes en PersonalJava

C.1.1. Programa l10n.java

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.io.*;
final public class l10n extends WindowAdapter
  implements ActionListener {

  private Frame mainFrame;
  //Menu items -- file --
  private Menu fileMenu;
  private MenuItem exitMI;
  private MenuItem openMI;
  private MenuItem saveMI;
  private MenuItem saveasMI;
  private MenuItem printMI;
```

```
private MenuItem closeMI;
//Menu items -- help --
private Menu helpMenu;
private MenuItem aboutMI;
// Panel
private Panel mainPanel;
//Window Size
private int windowWidth = 240;
private int windowHeight = 320;
private int titleHeight = 13;
private int taskbarHeight = 19;
private int margin = 2;
public l10n (String[] menu_prin, String[] menu_arch, String[]
             menu_ayuda) {
  // Create frame
  mainFrame = new Frame("Ejemplo_Localizacion");
  // Create menu bar
  MenuBar mb = new MenuBar();
  mainFrame.setMenuBar(mb);
  //File Menu
  //String menulabel = ("Exit");
  fileMenu = new Menu(menu_prin[0]);
  openMI = new MenuItem(menu_arch[0]);
  saveMI = new MenuItem(menu_arch[1]);
  saveasMI = new MenuItem(menu_arch[2]);
  printMI = new MenuItem(menu_arch[3]);
  closeMI = new MenuItem(menu_arch[4]);
  exitMI = new MenuItem(menu_arch[5]); //"Exit");
  exitMI.addActionListener(this);
```

```
    fileMenu.add(openMI);
    fileMenu.add(saveMI);
    fileMenu.add(saveasMI);
    fileMenu.add(printMI);
    fileMenu.add(closeMI);
    fileMenu.add(exitMI);
    mb.add(fileMenu);
    //Help Menu
    helpMenu = new Menu(menu_prin[1]);
    aboutMI = new MenuItem(menu_ayuda[0]); //"Acerca");
    aboutMI.addActionListener(this);
    helpMenu.add(aboutMI);
    mb.add(helpMenu);
    //setup panel
    mainPanel = new Panel();
    mainFrame.add(mainPanel);
    mainPanel.setVisible(true);
    // setup frame
    mainFrame.addWindowListener(this);
    mainFrame.setSize((windowWidth - margin + 2 ), (windowHeight -      titleH
    mainFrame.setVisible( true );
    mainFrame.setResizable(false);
  }
  // Event handling
  // windows event
  public void windowClosing (WindowEvent we) {
    if(we.getWindow() == mainFrame) {
      System.exit(0);
    }
```

```
    }
    // action event
    public void actionPerformed (ActionEvent ae) {
    Object o = ae.getSource();
    if (o instanceof MenuItem) {
      MenuItem mi = (MenuItem)o;
    if (mi == exitMI) {
      System.exit(0);
    } else if (mi ==aboutMI) {
      new l10nAbout();
    }
  }
}
public static String locale_en_Z() throws IOException
{
  StringTokenizer st1;
  String quote = "";
  FileReader infile = new FileReader("locale.conf");
  BufferedReader buff = new BufferedReader(infile);
  boolean eof = false;
  boolean siguiente = false;
  while( !eof ) {
    String line = buff.readLine();
    if( line == null )
      eof = true;
    else {
      if( siguiente ) {
        quote = line;
        siguiente = false;
```

```
        }
        line.trim();
        if( line.equals("[Language]") ) siguiente = true;
      }
    }
    buff.close();
    st1 = new StringTokenizer( quote, "=");
    st1.nextToken();
    return st1.nextToken();
  }
  public static void main (String args[]) {
    String[][] menu_prin = { {"Archivo","Ayuda"}, {"File","Help"},
                             {"Akte","Hilfe"} };
    String[][] menu_arch = { {"Abrir","Guardar","Guardar Como",
                              "Imprimir","Cerrar","Salir"},
                             {"Open","Save","Save as",
                              "Print","Close","Exit"},
                             {"Geoffnet","AuBer","AuBer wie",
                              "Druck","Ende","Auf Wiedersehen"} };
    String[][] menu_ayuda = { {"Acerca"}, {"About"}, {"Uber"} };
    int arreglo; // 0 = sp, 1 = en, 2 = de
    String localiza = "sp";
    try { localiza = locale_en_Z(); }
    catch (IOException e) {import java.awt.*;
  import java.awt.event.*;
  import java.util.*;
  import java.io.*;
  final public class l10n extends WindowAdapter
    implements ActionListener {
```

```
private Frame mainFrame;
//Menu items -- file --
private Menu fileMenu;
private MenuItem exitMI;
private MenuItem openMI;
private MenuItem saveMI;
private MenuItem saveasMI;
private MenuItem printMI;
private MenuItem closeMI;
//Menu items -- help --
private Menu helpMenu;
private MenuItem aboutMI;
// Panel
private Panel mainPanel;
//Window Size
private int windowWidth = 240;
private int windowHeight = 320;
private int titleHeight = 13;
private int taskbarHeight = 19;
private int margin = 2;
public l10n (String[] menu_prin, String[] menu_arch, String[]
             menu_ayuda) {
  // Create frame
  mainFrame = new Frame("Ejemplo_Localizacion");
  // Create menu bar
  MenuBar mb = new MenuBar();
  mainFrame.setMenuBar(mb);
  //File Menu
```

```
//String menulabel = ("Exit");
fileMenu = new Menu(menu_prin[0]);
openMI = new MenuItem(menu_arch[0]);
saveMI = new MenuItem(menu_arch[1]);
saveasMI = new MenuItem(menu_arch[2]);
printMI = new MenuItem(menu_arch[3]);
closeMI = new MenuItem(menu_arch[4]);
exitMI = new MenuItem(menu_arch[5]); //"Exit");
exitMI.addActionListener(this);
fileMenu.add(openMI);
fileMenu.add(saveMI);
fileMenu.add(saveasMI);
fileMenu.add(printMI);
fileMenu.add(closeMI);
fileMenu.add(exitMI);
mb.add(fileMenu);
//Help Menu
helpMenu = new Menu(menu_prin[1]);
aboutMI = new MenuItem(menu_ayuda[0]); //"Acerca");
aboutMI.addActionListener(this);
helpMenu.add(aboutMI);
mb.add(helpMenu);
//setup panel
mainPanel = new Panel();
mainFrame.add(mainPanel);
mainPanel.setVisible(true);
// setup frame
mainFrame.addWindowListener(this);
mainFrame.setSize((windowWidth - margin + 2 ), (windowHeight -      titleH
```

```
    mainFrame.setVisible( true );
    mainFrame.setResizable(false);
  }
  // Event handling
  // windows event
  public void windowClosing (WindowEvent we) {
    if(we.getWindow() == mainFrame) {
      System.exit(0);
    }
  }
  // action event
  public void actionPerformed (ActionEvent ae) {
  Object o = ae.getSource();
  if (o instanceof MenuItem) {
    MenuItem mi = (MenuItem)o;
  if (mi == exitMI) {
    System.exit(0);
  } else if (mi ==aboutMI) {
    new l10nAbout();
  }
 }
}
public static String locale_en_Z() throws IOException
{
  StringTokenizer st1;
  String quote = "";
  FileReader infile = new FileReader("locale.conf");
  BufferedReader buff = new BufferedReader(infile);
  boolean eof = false;
```

```
    boolean siguiente = false;
    while( !eof ) {
      String line = buff.readLine();
      if( line == null )
        eof = true;
      else {
        if( siguiente ) {
          quote = line;
          siguiente = false;
        }
        line.trim();
        if( line.equals("[Language]") ) siguiente = true;
      }
    }
    buff.close();
    st1 = new StringTokenizer( quote, "=");
    st1.nextToken();
    return st1.nextToken();
  }
  public static void main (String args[]) {
    String[][] menu_prin = { {"Archivo","Ayuda"}, {"File","Help"},
                             {"Akte","Hilfe"} };
    String[][] menu_arch = { {"Abrir","Guardar","Guardar Como",
                              "Imprimir","Cerrar","Salir"},
                             {"Open","Save","Save as",
                              "Print","Close","Exit"},
                             {"Geoffnet","AuBer","AuBer wie",
                              "Druck","Ende","Auf Wiedersehen"} };
    String[][] menu_ayuda = { {"Acerca"}, {"About"}, {"Uber"} };
```

```
    int arreglo; // 0 = sp, 1 = en, 2 = de
    String localiza = "sp";
    try { localiza = locale_en_Z(); }
    catch (IOException e) {
      System.err.println(e);
      return;
    }
    localiza = localiza.trim();
    if( args.length == 1 ) localiza = args[0];
      arreglo = 0; // sp por defecto
    if(localiza.equals("sp")) arreglo = 0;
    else if(localiza.equals("en")) arreglo = 1;
    else if(localiza.equals("de")) arreglo = 2;
    l10n foo = new l10n(menu_prin[arreglo], menu_arch[arreglo],
                        menu_ayuda[arreglo]);
    //System.err.println( args[0] );
  }
      System.err.println(e);
      return;
    }
    localiza = localiza.trim();
    if( args.length == 1 ) localiza = args[0];
      arreglo = 0; // sp por defecto
    if(localiza.equals("sp")) arreglo = 0;
    else if(localiza.equals("en")) arreglo = 1;
    else if(localiza.equals("de")) arreglo = 2;
    l10n foo = new l10n(menu_prin[arreglo], menu_arch[arreglo],
                        menu_ayuda[arreglo]);
    //System.err.println( args[0] );
```

```
}
```

C.2. Localización de mensajes en PHP

C.2.1. Programa locale.php

```
<?php
$mensajes = array(
  'es_MX' => array(
    'Mi comida favorita es' => 'Mi comida favorita es',
    'Papas a la francesa' => 'Papas a la frances',
    'biscocho' => 'biscocho',
    'dulces' => 'dulces',
    'papas fritas' => 'papas fritas',
    'galletas' => 'galletas',
    'maiz' => 'maiz',
    'berenjena' => 'berenjena'
  ),
  'en_GB' => array(
    'Mi comida favorita es' => 'My favourite foods are',
    'Papas a la francesa' => 'chips',
    'biscocho' => 'scone',
    'dulces' => 'sweets',
    'papas fritas' => 'crips',
    'galletas' => 'biscuit',
    'maiz' => 'maize',
    'berenjena' => 'aubergine'
  ),
  'en_US' => array(
```

```
    'Mi comida favorita es' => 'My favorite foods are',
    'Papas a la francesa' => 'french fries',
    'biscocho' => 'biscuit',
    'dulces' => 'candy',
    'papas fritas' => 'potato chips',
    'galletas' => 'cookies',
    'maiz' => 'corn',
    'berenjena' => 'eggplant'
  )
);
function msg($s) {
  global $LANG;
  global $mensajes;
  if( isset($mensajes[$LANG][$s])) {
    return $mensajes[$LANG][$s];
  } else {
    error_log("error I10n $LANG, mensajes: $s");
  }
}
$LANG = 'en_GB';
print msg('Mi comida favorita es').":<br>";
print msg('Papas a la francesa')."<br>";
print msg('biscocho')."<br>";
print msg('dulces')."<br>";
print msg('papas fritas')."<br>";
print msg('galletas')."<br>";
print msg('maiz')."<br>";
print msg('berenjena')."<br>";
?>
```

Apéndice D

Comandos importantes en Linux integrado.

D.0.2. Para arrancar Linux en la PDA sin cargar Qtopia.

Presionar / cuando marca 5..4..3..2..1

D.0.3. Para sacar el pipeline

Presionar Shift+SpaceBar = |

D.0.4. Comando ipkg

El comando ipkg sirve para instalar paquetes desde archivos con extension .ipk, lgunos ejemplos son:

ipkg paquete.ipk // para instalar el paquete

ipkg status | grep apache // indica donde esta instalado el programa

ipkg files paqueteipkgsinlaextensionarmniipk // lista los archivos y directorios de instalacion del paquete

D.0.5. Usuarios en la Zaurus

Existen dos usuario por default : zaurus y root, por defecto se entra como - zaurus

Para cambiar a usuario root:

- su -<enter>

D.0.6. Ligas simbólicas

Crear una liga para que cuando un prog. busque la libreria0.9.6 cargue la 0.9.7

- ln -s libcrypto.so.0.9.7 libcrypto.so.0.9.6

Para que corra un programa en vez de otro hacemos una liga

- ln -s links lynx