# Recomendaciones Para el Desarrollo de Aplicaciones P2P Utilizando el Protocolo SIP

## ANEXO B
## PREPARACIÓN DEL ENTORNO DE DESARROLLO Y CREACIÓN DE PROYECTOS


**Wilson Yecit Ortiz Sánchez**

Director: Ing. Javier Alexander Hurtado

**Universidad del Cauca**
**Facultad de Ingeniería Electrónica y Telecomunicaciones**
**Departamento de Telemática**
Línea de Investigación en Ingeniería de Sistemas Telemáticos
Popayán, Enero de 2007

**TABLA DE CONTENIDO**

## LISTA DE FIGURAS

# ANEXO B PREPARACION DEL ENTORNO DE DESARROLLO Y CREACIÓN DE PROYECTOS

En este documento se describen algunos pasos y sugerencias para obtener un IDE que contenga los componentes necesarios para el desarrollo de aplicaciones móviles SIP *pear-to-pear* y lo básico para iniciar la implementación de estas aplicaciones. Para la muestra, solo se describen los pasos necesarios de algunos IDEs existentes, que se pretende sirvan de referencia a la hora de realizar esta tarea.

## B1 DESARROLLOS SIP EN J2ME

La descripción de los pasos para preparar un Entorno de Desarrollo e implementar aplicaciones SIP con J2ME, se hicieron con algunas versiones disponibles de herramientas para el sistema operativo Windows. Si se utilizan versiones diferentes de estas herramientas los pasos a seguir pueden variar, pero la mecánica de obtener un IDE completo que contenga los componentes necesarios para desarrollar aplicaciones SIP con J2ME y la forma de crear una aplicación será similar. A continuación se describen los requerimientos necesarios para dos IDEs conocidos.

### B1.1 Instalación de las Herramientas Necesarias para Trabajar SIP sobre J2ME con Eclipse

A continuación se describen brevemente los pasos mas importantes para la instalación y configuración del IDE Eclipse para desarrollar aplicaciones móviles J2ME con SIP.

1. Primero instalar un JRE o JDK java igual o superior al necesitado por eclipse, por ejemplo JRE1.4.2_10.
2. Instalar eclipse siguiendo los pasos que presenta el asistente de instalación. Por ejemplo Eclipse 3.1.
3. Instalar un plug-in para desarrollar en J2ME, por ejemplo EclipseME.
4. Instalar por lo menos un JDK de emulación de dispositivos móviles que contenga soporte para SIP, por ejemplo: nptsdk_jme_v3_0, nptsdk_jme_v4_0_beta o Nokia Prototype SDK 4.0 Beta for JME. También es posible instalar un JDK sin soporte SIP y

luego instalar un plug-in SIP para J2ME como el Nokia_SIP_Plug-in_4[1].0_for_Series_60. Todos estos SDKs y el plug-in se pueden descargar del sitio del forum de Nokia (www.forum-nokia.com).

La versión actual del Wireless Toolkit (versión 2.2) no tiene soporte SIP, pero realizándole algunos cambios se puede utilizar para compilar y empaquetar MIDlets que utilizan el API SIP (Ver: 1.9 Uso del Wireless Toolkit con SIP).

Ahora Eclipse ya cuenta con lo necesario para desarrollar aplicaciones Móviles J2ME que usen SIP.

### B1.2 Crear un Proyecto SIP J2ME en Eclipse

Asumiendo que se tiene instalado el entorno de desarrollo Eclipse con todos los paquetes necesarios para poder generar proyectos J2ME con SIP (Ver Instalación de las herramientas necesarias para trabajar SIP J2ME con Eclipse), se realizan los siguientes pasos:

1. Para abrir el asistente de creación de proyectos seleccionar: *Archivo/Nuevo/Proyecto.* El asistente despliega la interfaz mostrada en la figura 1.

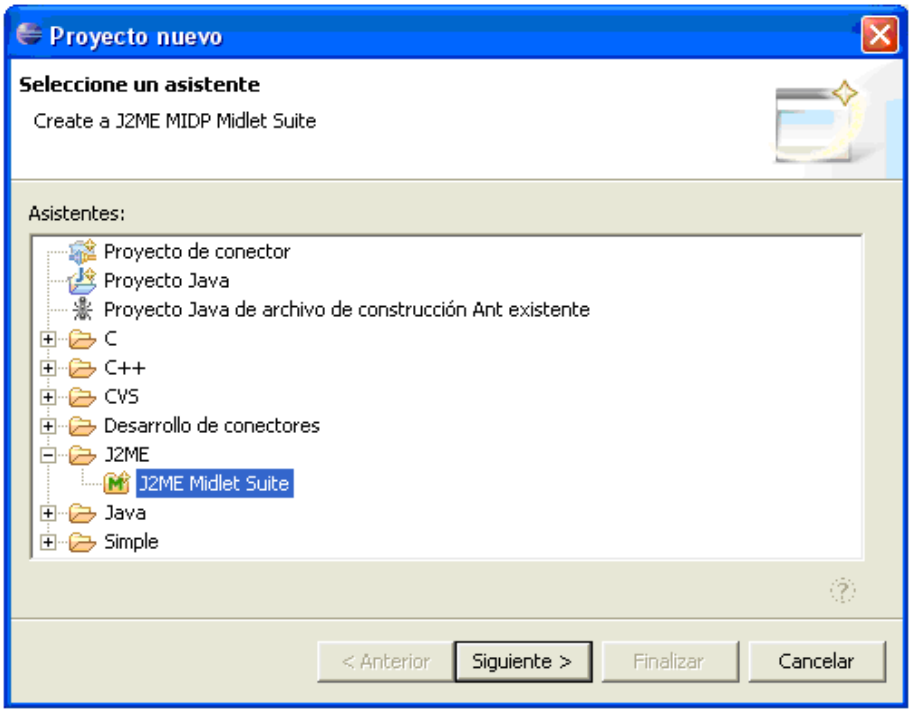


Figura 1. Nuevo proyecto SIP J2ME en Eclipse.

2. En la ventana de *Proyecto Nuevo*, abrir la carpeta *J2ME* del árbol de asistentes, seleccionar *J2ME Midlet Suite* y pulsar el botón *Siguiente>.*
3. En la ventana de creación de nuevo proyecto J2ME que se muestra en la figura 2 entrar un nombre de proyecto (en el ejemplo Prueba) y Pulsar el botón *Siguiente>.*

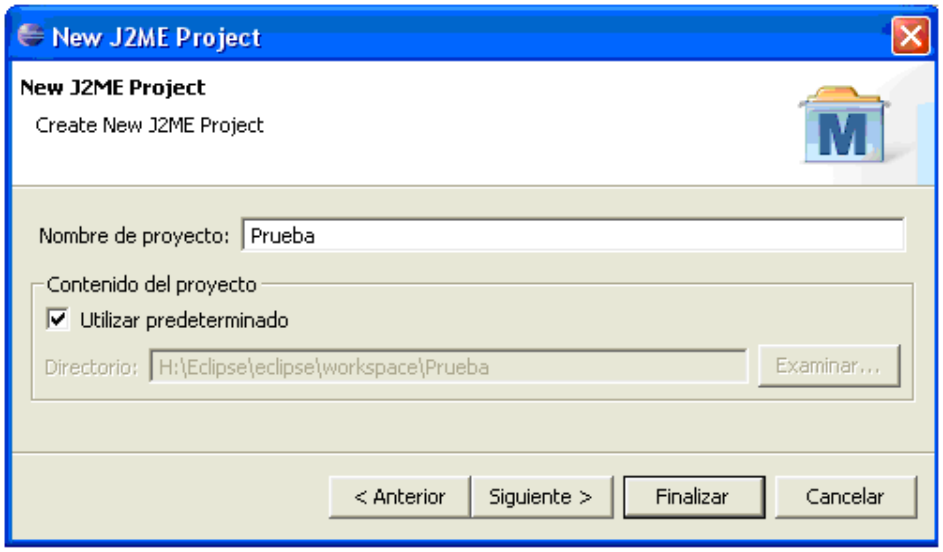


Figura 2. Nombre del proyecto SIP J2ME en Eclipse.

4. En la ventana de Definición de Plataforma que se muestra en la figura 3, elegir la plataforma correcta para el proyecto, en este caso "Prototype_4_0_Beta_S60_MIDP_Emulator_Platform".

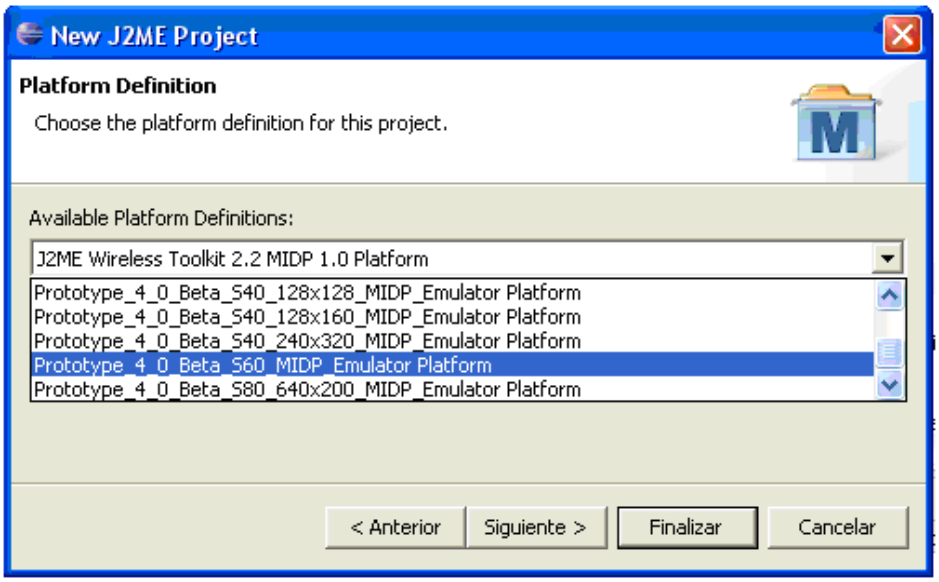


Figura 3. Plataformas del proyecto SIP J2ME en Eclipse.

5. Pulsar el botón *Finalizar,* y el proyecto “Prueba” es creado. La carpeta del proyecto creado, puede ser visualizada en el explorador de paquetes, como se muestra en la figura 4.

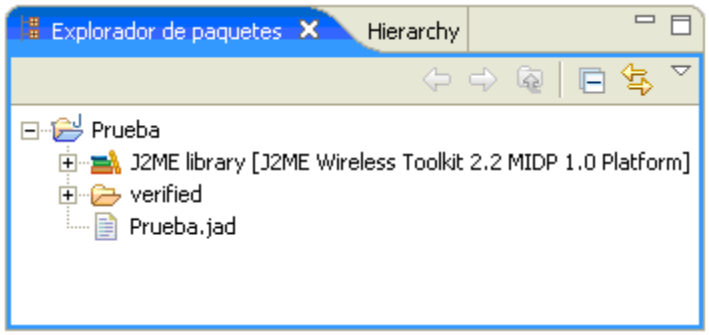


Figura 4. Proyecto J2ME creado en Eclipse.

Finalmente el asistente crea el proyecto “Prueba” y genera los archivos necesarios para este. Ahora solo resta crear una Midlet y las clases necesarias, como los pasos son iguales solo se mostrará el proceso de creación de la Midlet.

1. Se da clic derecho con el ratón sobre la carpeta del proyecto en el explorador de paquetes y se selecciona: “Nuevo/Otros…” como se muestra en la figura 5.

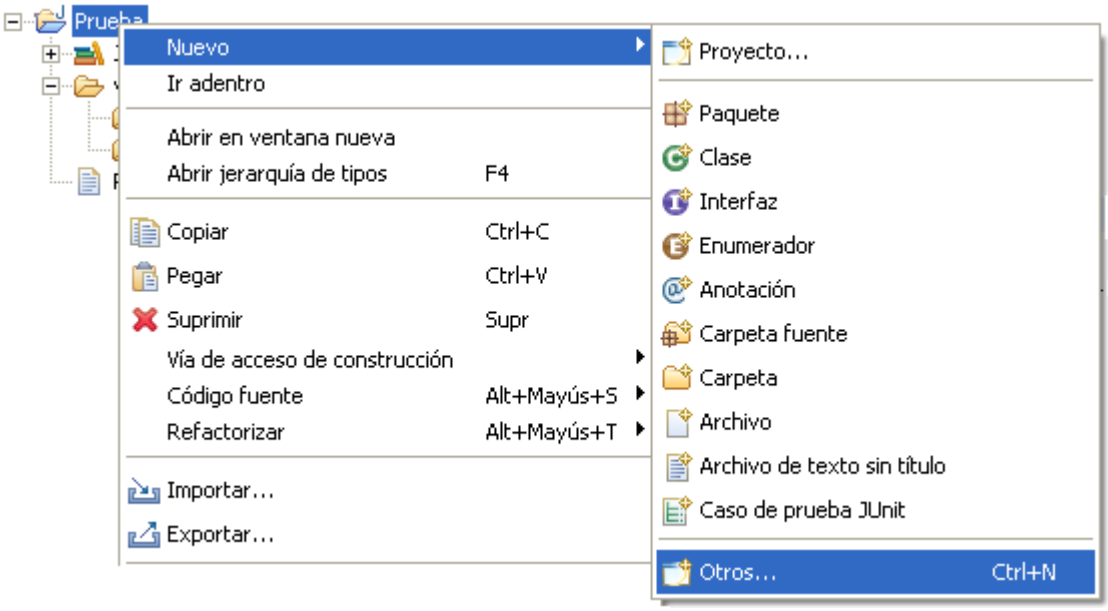


Figura 5. Creación de una Midlet en Eclipse

2. En figura 6 se muestra la ventana desplegada, en esta, para el caso de una midlet se selecciona la carpeta *J2ME* y el asistente *J2ME Midlet* y pulsar *Siguiente>*.

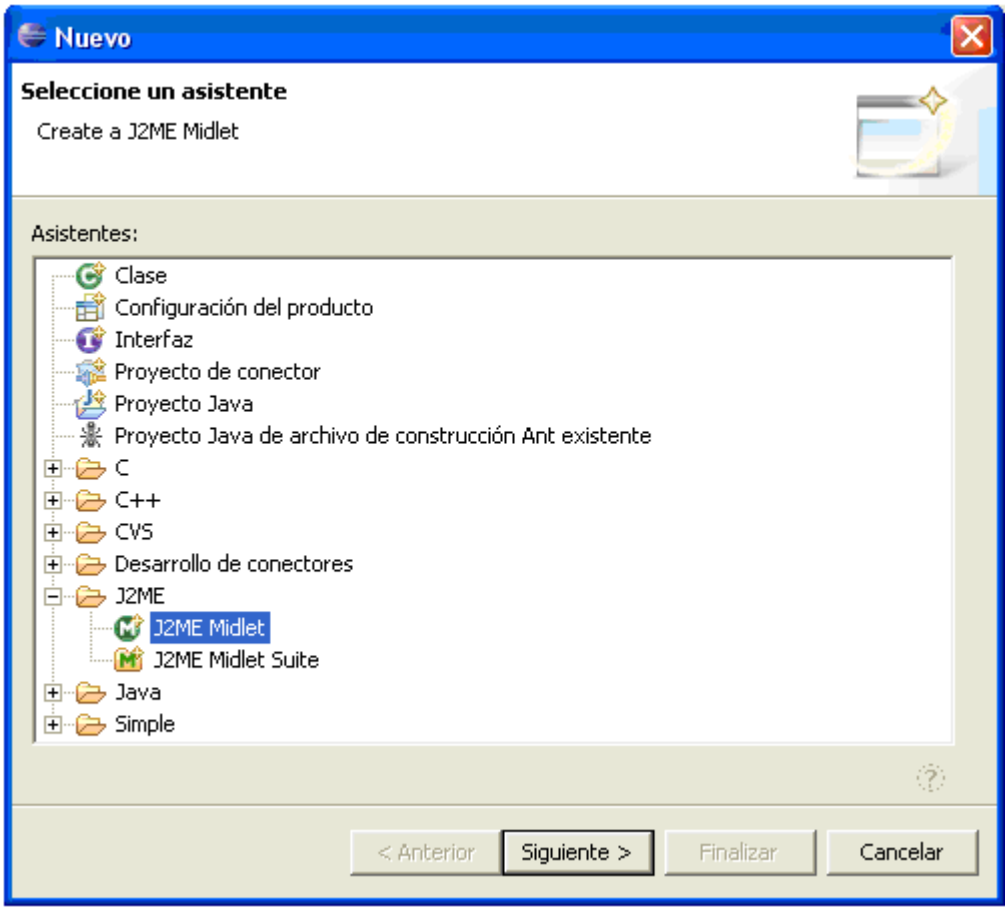


Figura 6. Selección de un asistente en Eclipse.

3. En la ventana desplegada que se muestra en la figura 7 se ingresa el nombre de la Midlet, en este caso Midlet_prueba y se termina la creación de la Midlet con el botón *Finalizar*.

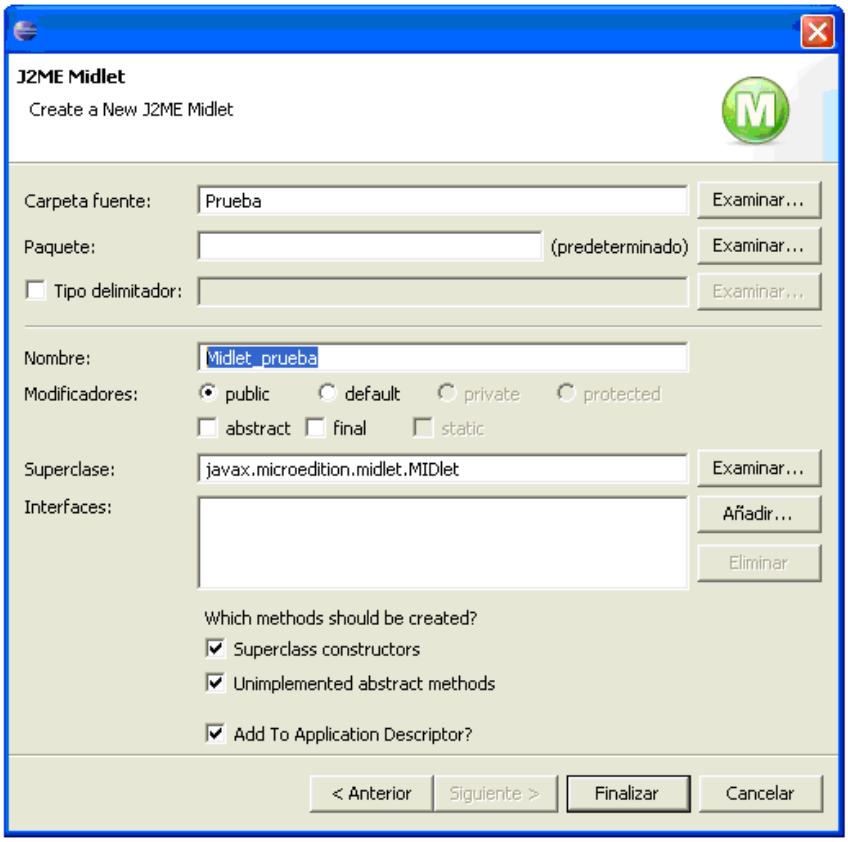


Figura 7. Nombre de la Midlet en Eclipse.

4. La Midlet queda creada con su constructor y los métodos principales: destroyApp, pauseApp y stopApp que pueden ser visualizados en el explorador de paquetes como se puede ver en la figura 8.

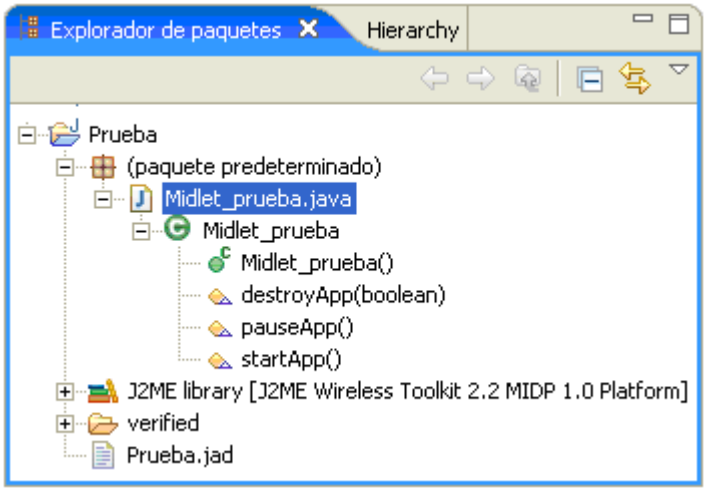


Figura 8. Midlet creada en Eclipse.

### B1.3 Emular un Proyecto J2ME con SIP en Eclipse

Para correr aplicaciones SIP J2ME creadas sobre Eclipse por primera vez se debe iniciar por "construir" la aplicación, para esto se elige la opción *Construir todo* del menú desplegable *Proyecto* como se muestra en la figura 9.

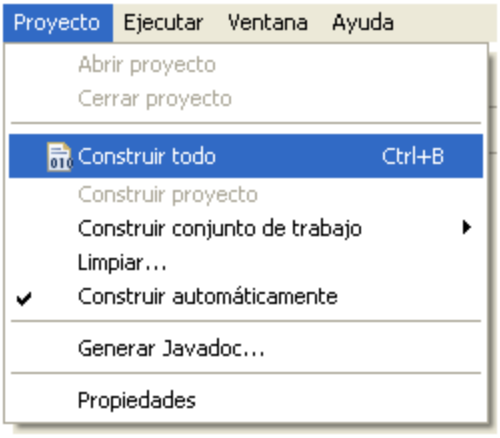


Figura 9. Construcción de la aplicación.

Ahora se procede a crear una configuración de ejecución desde la opción "Ejecutar" que se encuentra en el menú desplegable *Ejecutar* como se muestra en la figura 10 o dando clik derecho sobre la midlet se encuentra la opción *Ejecutar como,* que abre un sub-menú, el cual contiene esta opción, como se muestra en la figura 11.

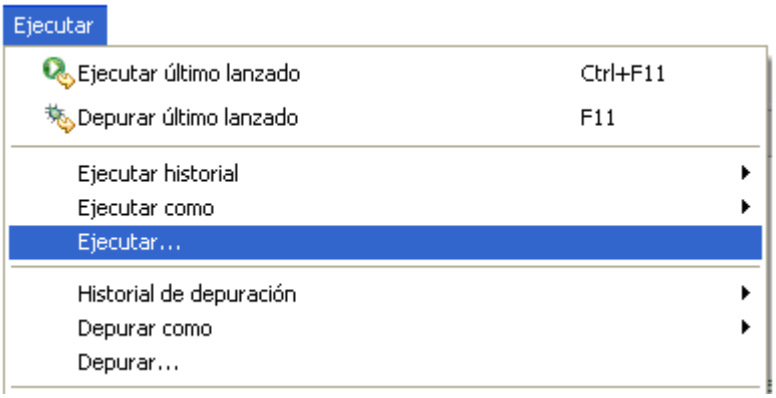


Figura 10. Opción Ejecutar desde el menú desplegable Ejecutar.

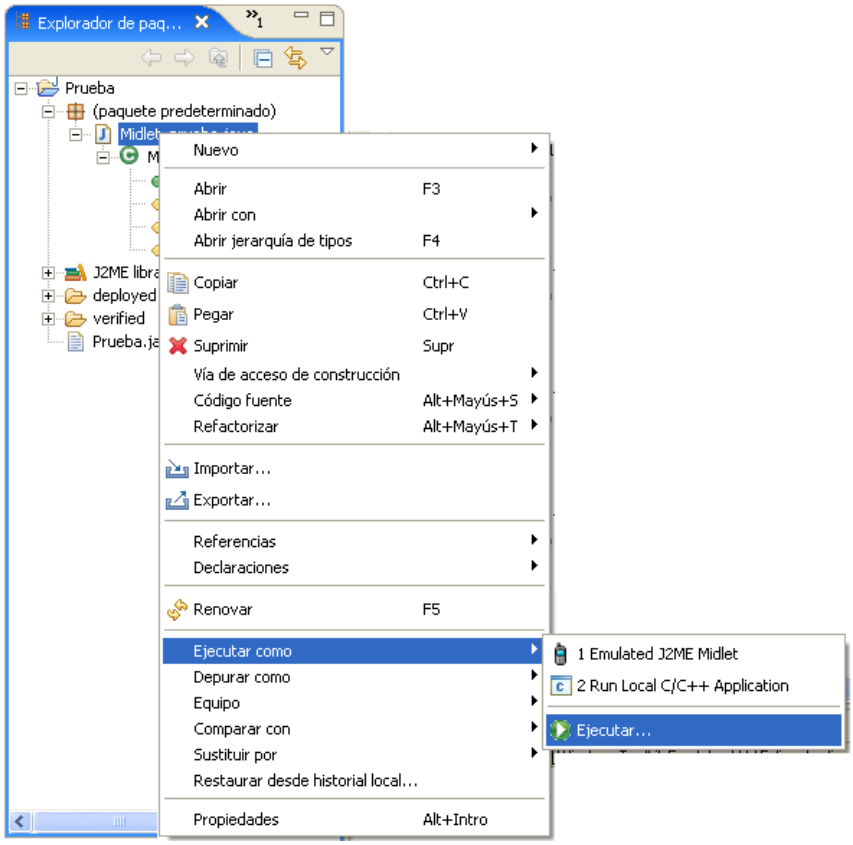


Figura 11. Opción Ejecutar desde la Midlet.

Después de seleccionar *Ejecutar* se abre la ventana de creación, gestión y ejecución de configuraciones la cual por defecto contiene la ultima configuración utilizada y que puede pertenecer a otro proyecto. Por esto se debe crear por lo menos una nueva configuración para el proyecto actual. Esta interfaz se muestra en la figura 12.

Para crear una nueva configuración se selecciona el botón *Nueva* y el asistente adiciona una nueva configuración con el nombre “configuración_nueva”. Si no se desea este nombre de configuración, este puede ser cambiado en el campo correspondiente.

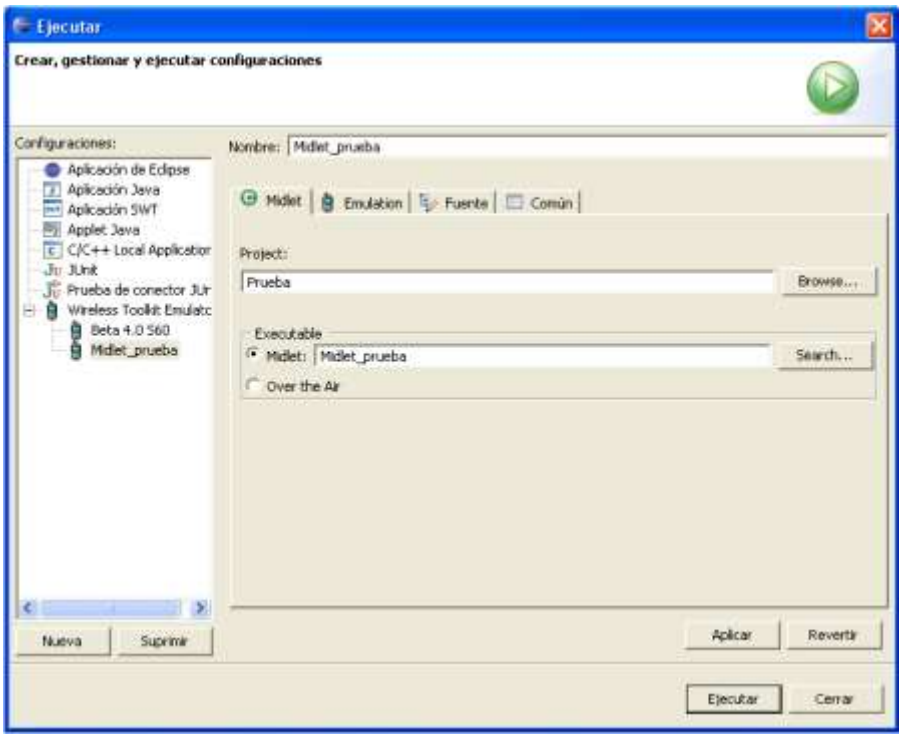


Figura 12. Ventana de creación, gestión y ejecución de configuraciones.

Ahora teniendo activo el tab Midlet se ingresa o escoge el nombre del proyecto que se desea emular (En el ejemplo es “Prueba”) sobre el campo “Proyecto” y en el campo “Ejecutable” se ingresa o escoge el nombre de la Midlet. En el tab “Emulación” se puede confirmar que el emulador sea el correcto (uno para la serie 60 que soporte SIP) o se modifica eligiendo uno de la lista desplegable “Definición de plataforma”. Por ultimo se guarda la configuración con el botón *Aplicar* y se comienza la emulación con *Ejecutar.* La creación de esta configuración sirve para todas las veces que se desee realizar la emulación, por lo tanto las próximas emulaciones podrán ser iniciadas desde cualquier opción “Ejecutar” disponible, como la que se encuentra en la barra de herramientas, que comienza la emulación con la configuración ya creada, sin necesidad de abrir la ventana de creación, gestión y ejecución de configuraciones. El emulador desplegado se puede visualizar en la figura 13.


Figura 13. Emulación de una aplicación J2ME en Eclipse.

## B1.4 Instalación de las Herramientas Necesarias para Trabajar SIP J2ME con JBuilder

A continuación, se describen brevemente los pasos mas importantes, para la instalación y configuración del IDE JBuilder, para desarrollar aplicaciones móviles J2ME con SIP.

1. Instalar una versión del JBuider siguiendo los pasos que presenta el asistente.
2. Si la versión instalada del JBuilder no contiene soporte para J2ME instalar el paquete J2ME adicional para la versión del JBuilder instalada.
3. Instalar por lo menos un JDK de emulación de dispositivos móviles que contenga soporte para SIP, por ejemplo: nptsdk_jme_v3_0, nptsdk_jme_v4_0_beta o Nokia Prototype SDK 4.0 Beta for JME; también es posible instalar un JDK sin soporte SIP y luego instalar un plug-in SIP para J2ME. Todos estos SDKs y el plug-in se pueden descargar de las páginas del forum de Nokia (http://forum.nokia.com).

La versión actual del Wireless Toolkit (versión 2.2) no tiene soporte SIP, pero realizándole algunos cambios se puede utilizar para compilar y empaquetar MIDlets que utilizan el API SIP (Ver: 1.9 Uso del Wireless Toolkit con SIP).

Adicionar el JDK de emulación de dispositivos móviles que contiene el soporte SIP, a los SDKs del JBuilder de la siguiente manera:
Seleccionar Herramientas/configurar JDKs como se muestra en la figura 14.

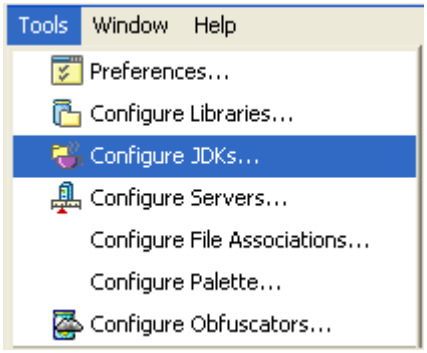


Figura 14. Configuración de JDKs en JBuilder.

En la ventana de Configuración de JDKs seleccionar el botón *Nuevo* como se muestra en la figura 15.

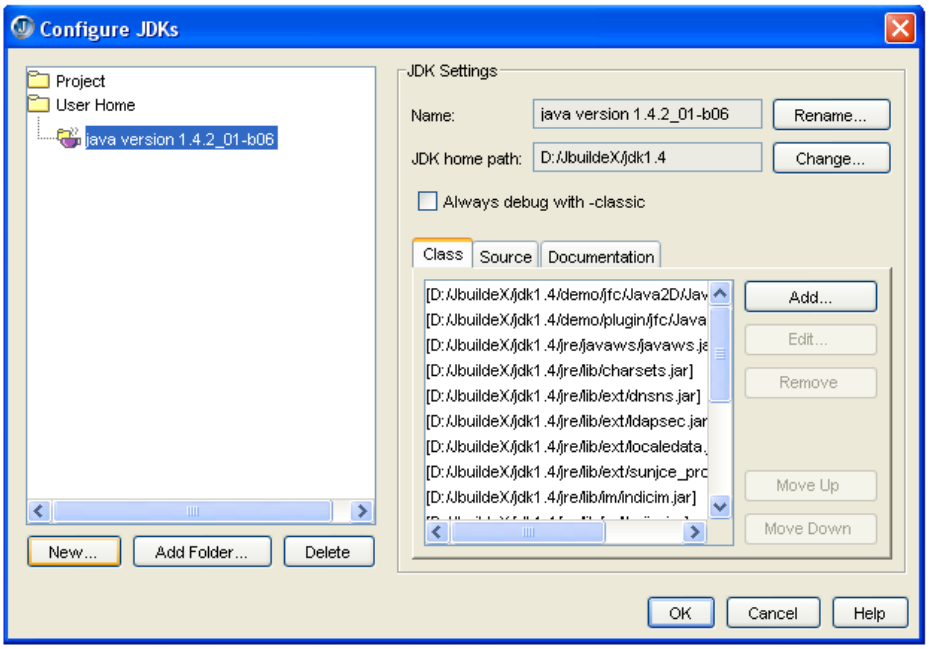


Figura 15. Adición de un JDK en JBuilder.

Luego en la ventana del asistente ingresar la ruta del JDK en el campo “Dirección principal del JDK existente” o seleccionar el botón “…” que permite realizar esta operación de una forma más cómoda. La figuras 16  y 17muestran la interfaces correspondientes a este paso.

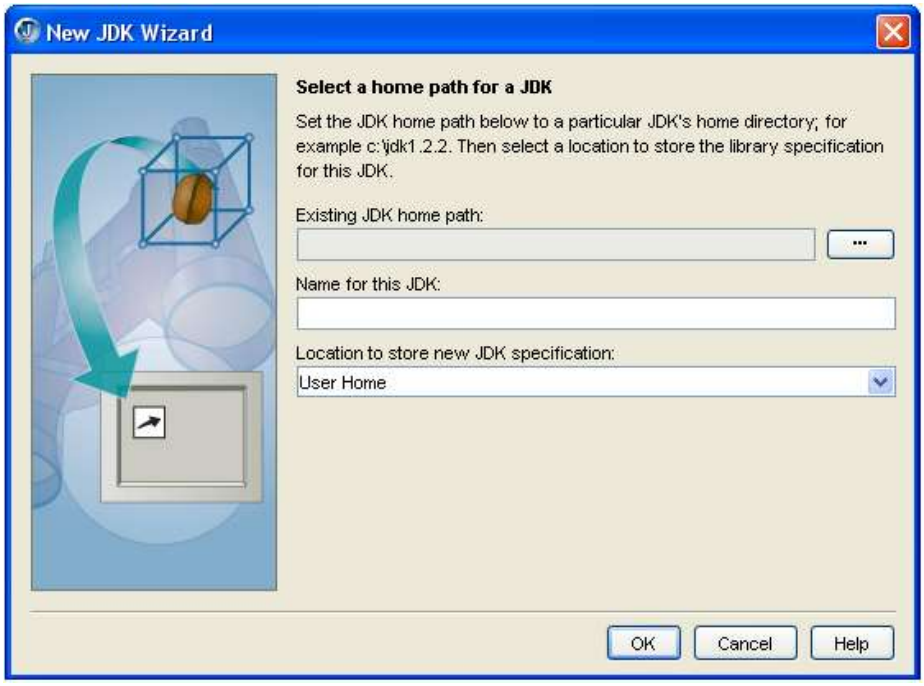


Figura 16. Ingreso de ruta de JDK en JBuilder.

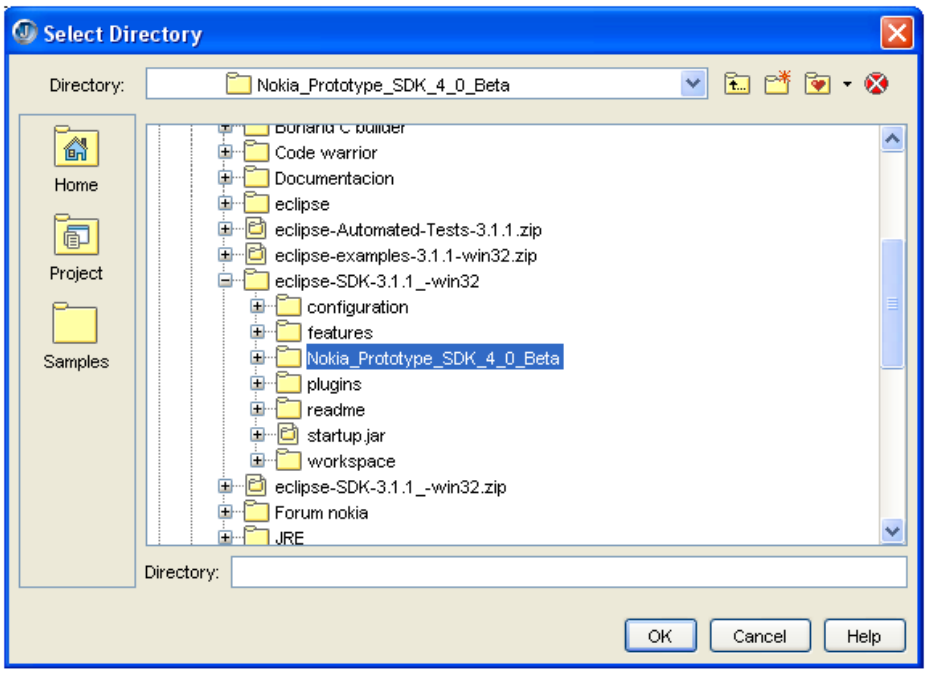


Figura 17. Selección de JDK en JBuilder.

Seleccionar el botón “OK” y el JDK ha quedado adicionado al JBuilder como se muestra en la figura 18.

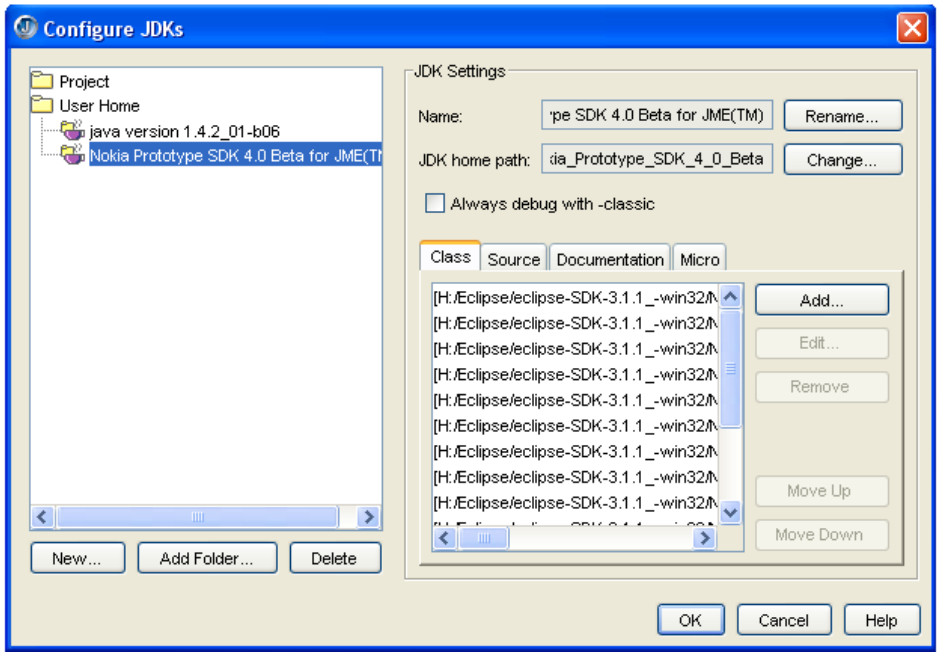


Figura 18. JDK adicionado en JBuilder.

Como el JDK emula varios dispositivos de diferentes series (40,60 y 80) y el que interesa para los desarrollos con SIP es el de la serie 60, la selección se realiza de la siguiente forma: En la misma ventana de configuración, se selecciona el JDK “Nokia Prototype SDK 4.0 for JME” y en el lado de configuración se elige la pestaña “Micro” como se muestra en la figura 19.

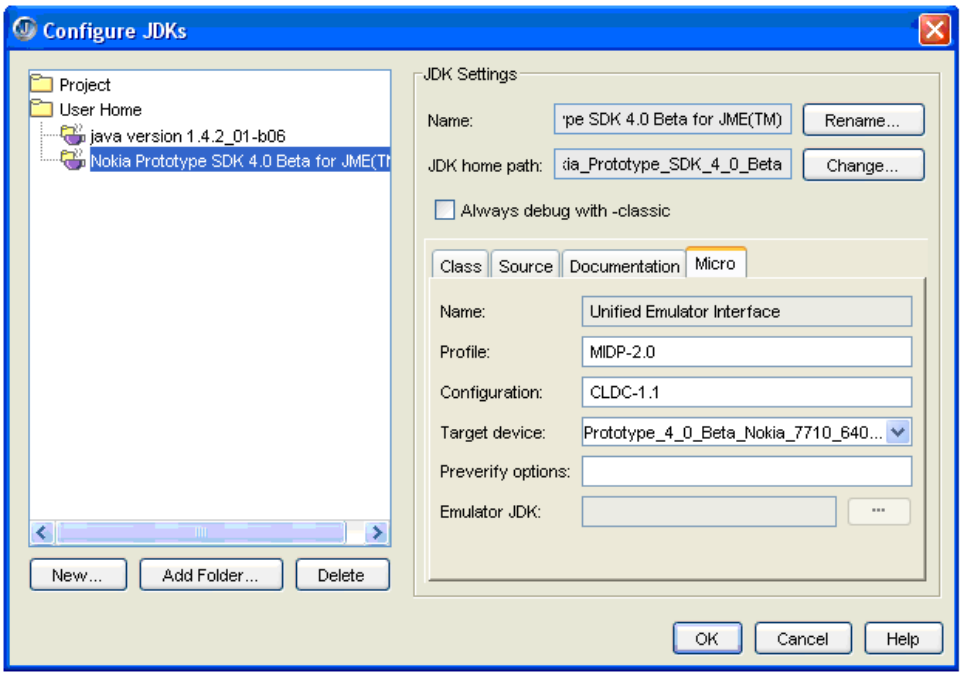


Figura 19. Configuración del emulador en JBuilder.

En el ítem dispositivo de destino se selecciona "Prototype_4_0_Beta_S60_MIDP_Emulator" y luego el botón de "OK" como se muestra en la figura 20.

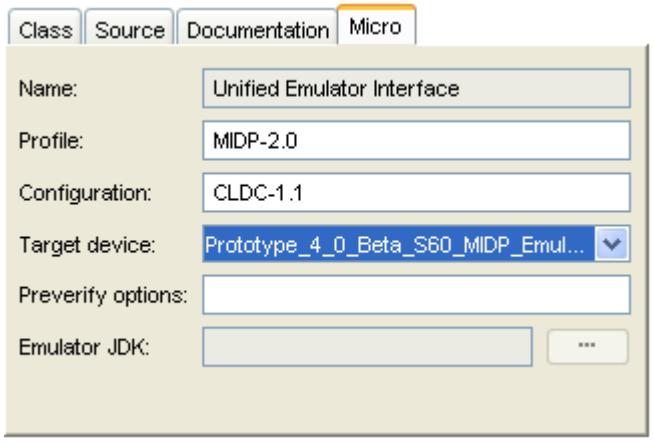


Figura 20. Selección de emulador en JBuilder.

El IDE JBuilder presenta otra forma para seleccionar un emulador. Que es mediante la selección de la opción *Preferencias de emulación* del menú desplegable *Herramientas*. Esta forma no produce un cambio real sobre la selección del emulador. Es decir que sea cual sea el emulador elegido desde la opción Preferencias de emulación, el JBuilder

emulará la aplicación de acuerdo al ítem elegido en “Dispositivo de destino” de la pestaña “Micro”

Ahora el JBuilder ya cuenta con lo necesario para desarrollar aplicaciones Móviles J2ME que usen SIP.

### B1.5 Crear un Proyecto SIP J2ME en JBuilder

Asumiendo que se tiene instalado el entorno de desarrollo JBuilder con todos los paquetes necesarios para poder generar proyectos J2ME con SIP (Ver Instalación de las herramientas necesarias para trabajar SIP J2ME con JBuilder), se realizan los siguientes pasos:

1. Para abrir el asistente de creación de proyectos seleccionar: *Archivo/Nuevo Proyecto.*
2. En la ventana desplegada del asistente ingresar el nombre del proyecto en el campo “*Nombre*” y si se desea se puede modificar el directorio del proyecto, desde la opción “*Directorio*” como se muestra en la figura 21. Luego se selecciona *Siguiente>*

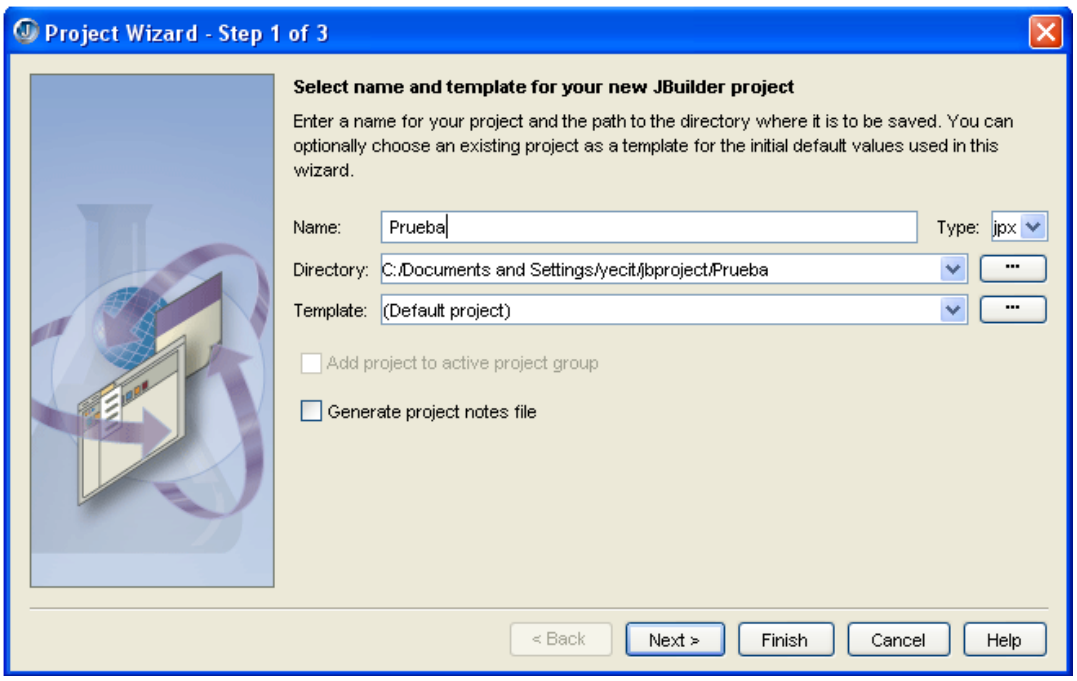


Figura 21. Nombre del proyecto SIP J2ME en JBuilder.

3. En la ventana de “Rutas del proyecto” se elige el JDK para el proyecto. La interfaz correspondiente a este paso se muestra en al figura 22.

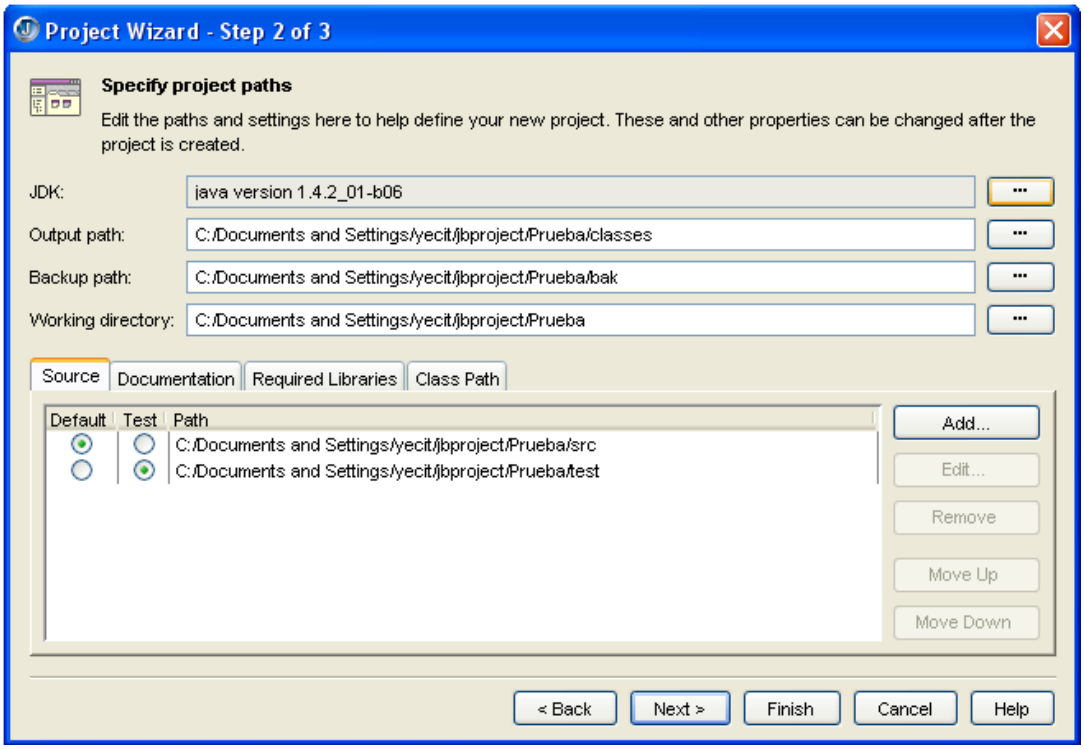


Figura 22. Rutas del proyecto SIP J2ME en JBuilder.

4. En este caso se elige el JDK “Nokia Prototype SDK 4.0 Beta for JME(TM)” y se elige el botón *OK* como se muestra en la figura 23.

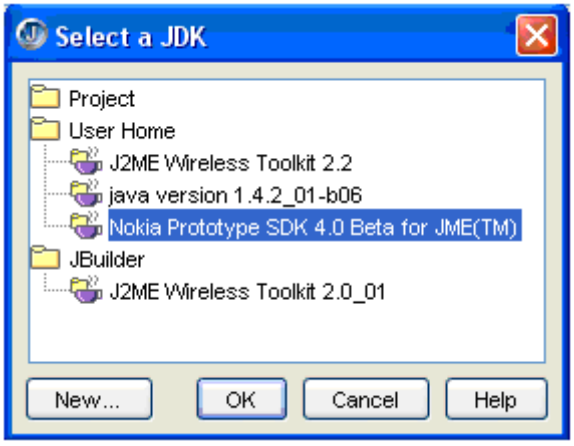


Figura 23. JDK del proyecto SIP J2ME en JBuilder.

5. Por ultimo se termina la creación del proyecto con el botón *Finalizar*.

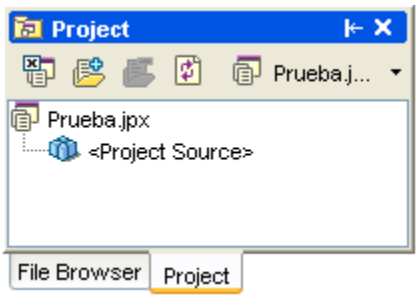


Figura 24. Proyecto SIP J2ME creado en JBuilder.

Finalmente el asistente crea el proyecto “Prueba” y genera los archivos necesarios para este como se muestra en la figura 24. Ahora solo resta crear una Midlet y las clases necesarias. A continuación se listan los pasos necesarios para crear una Midlet.

1. Se selecciona *Archivo/Nuevo* y el IDE abre el asistente para la creación de a Midlets y Displayables como se muestra en la figura 25.

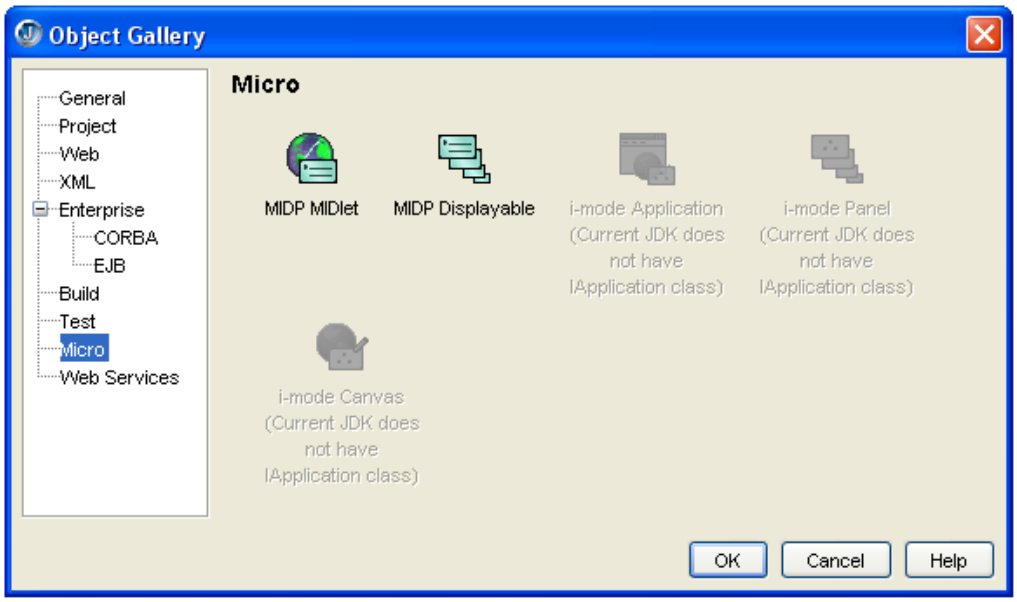


Figura 25. Asistente para la creación de a Midlets y Displayables.

2. A continuación se selecciona MIDP MIDlet y luego el botón OK, mostrándose la interfaz de la figura 26.


Figura 26 Interfaz para ingresar los detalles de la Midlet.

3. En la ventana de “Detalles de la Midlet” del asistente se ingresa el nombre de esta. Luego se elige *Siguiente>.*

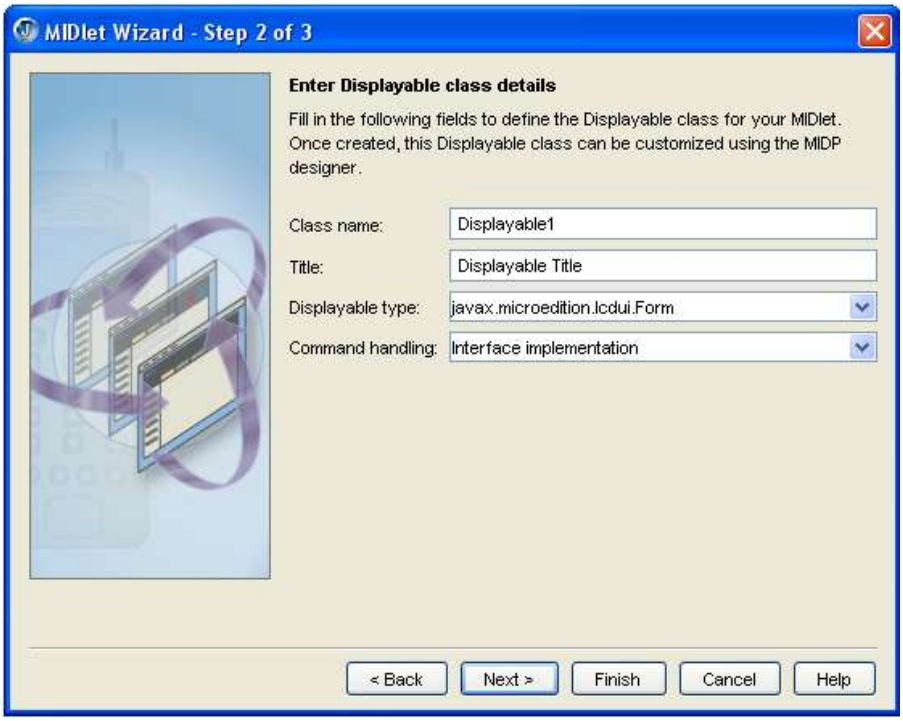


Figura 27. Interfaz para ingresar los detalles del displayable.

4. En la ventana “Detalles de la clase Displayable” se ingresa el nombre de la clase, el titulo que presentará ésta clase cuando este en ejecución y se escoge el tipo de Displayable. Luego terminamos la creación de la Midlet con el botón *Finalizar*. Esta interfaz se muestra en la figura 27.

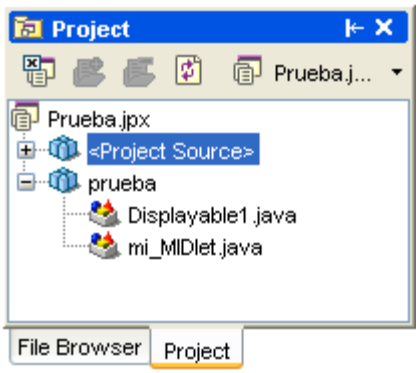


Figura 28. Proyecto creado en el JBuilder.

La Midlet y el Displayable que esta lanzará quedan creados y la ventana de proyectos muestra el contenido del proyecto, como se muestra en la figura 28. Para generar Displayables adicionales se siguen los pasos 1 y 4 de la creación de una Midlet, pero en la primera ventana del asistente se escoge Displayable en lugar de MIDlet.

### B1.6 Emular un Proyecto J2ME con SIP en JBuilder

Asumiendo que todas las herramientas se encuentran correctamente instaladas para este entorno, para correr las aplicaciones SIP J2ME creadas sobre el JBuilder solo basta elegir la opción “Run Project” que se encuentra en el menú desplegable “Run” o que se activa con la tecla función F9. Otras maneras de ejecutar la emulación es dando clic derecho con el ratón sobre la Midlet del proyecto y seleccionando la opción “Micro Run using” nombre_Midlet o con el icono “Run” de la barra de Ejecución/depuración. La figura 29 muestra el emulador desplegado.


Figura 29. Emulación de una aplicación J2ME en JBuilder.

### B1.7 Configuración de Multi-Emulador en el ”Nokia Prototype Sdk 4.0 Beta For JME”

Esta operación permite tener dos o más emuladores ejecutándose en el mismo PC y ejecutando la misma o diferentes aplicaciones SIP, sin producir conflicto debido al uso del puerto predefinido para SIP.

El emulador “nokia prototype sdk 4.0 beta for jme” soporta el API para J2ME JSR-180 que proporciona la habilidad de comunicación usando SIP, lo que permite la creación de MIDlets de comunicación que usen el protocolo. Por defecto, el API SIP permite la comunicación entre emuladores que sean instalados en equipos de trabajo separados. Si

se necesita usar SIP para comunicación entre emuladores instalados en el mismo equipo de trabajo, se debe configurar el archivo: internal.config de la siguiente forma:

- Abrir el archivo internal.config que se encuentra en la siguiente ruta:
  <SDK_Home>\devices\<Device>\lib\internal.config
  Nota: “SDK_Home” indica el directorio raíz de instalación del SDK y “device” indica el emulador elegido de la serie 60, en este caso:
  “Prototype_4_0_Beta_S60_MIDP_Emulator”.

- Elimine la definición del número de puerto de uso por defecto en la línea:
  com.nokia.phone.ri.sip.local_port: <remover este numero>

Al eliminar el número de puerto, cada interfaz de emulación podrá tomar un puerto diferente, evitando el conflicto que se genera al ejecutar dos interfaces de emulación.
Nota: Cada interfaz de emulación es diferente e independiente de las demás; es decir, si por ejemplo si en la segunda interfaz desplegada se almacenó alguna información en su Record Store como un número de dirección SIP, en adelante solo será posible recuperar esta dirección desde el segundo terminal de emulación desplegado. [1]

### B1.8 Configuración de uso de Recursos de los SDKs de la Serie 60 de Nokia

Esta operación facilita eliminar los molestos mensajes de autorización desplegados por el emulador, al usar algunas funciones restringidas y permitir el uso de otras funciones que se encuentran bloqueadas, como el acceso a lectura y escritura del Record Store y el acceso a los archivos del dispositivo (imágenes, de audio, video y otros) entre otras. Dos de estos mensajes se muestran en la figura 30.

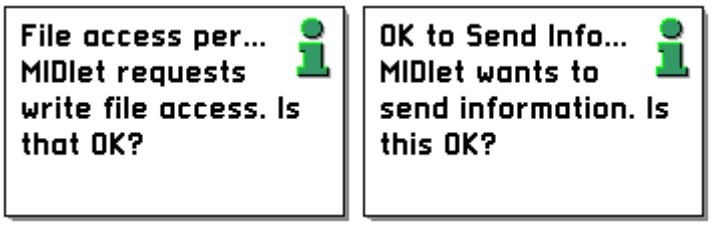


Figura 30. Mensajes de solicitud de autorización a las funciones restringidas.

El SDK para la serie 60 de Nokia, entre sus capacidades incluye la emulación de 6 modos o niveles de operación, los cuales pretenden realizar un funcionamiento similar a como lo

hace un dispositivo real mediante un comportamiento simulado, limitando y permitiendo el uso de ciertos recursos del dispositivo mediante permisos de acceso a las aplicaciones. Algo análogo a lo que hacen las últimas versiones del sistema operativo Windows con la instalación de drivers, a los cuales se les pide un certificado de autenticidad para evitar la instalación de software “malicioso”.

Los niveles existentes son los siguientes:

Nivel de Vida Real. En este nivel los Midlets son tratados de la misma forma a como son manejados por los dispositivos reales. El Midlet gana el acceso a las funciones protegidas según la información del certificado. Midlets MIDP 1.0 y sin firmar siempre serán tratadas como No Confiables. Si la firma del certificado no coincide al certificado raíz en el SDK la Midlet es rechazada.

Nivel Fabricante. En este nivel cada Midlet es tratado como un Midlet creado por Nokia. Se da acceso a las funciones protegidas de acuerdo con el Midlet.

Nivel confiar en una tercera parte. En este nivel un Midlet se trata como uno certificado por un tercer participante.

Nivel de desconfiado. En este nivel el acceso del MIDlet a las funciones protegidas está significativamente reducido y el Midlet se trata como sin firmar.

Nivel personalizado. Es un funcionamiento simulado dónde el usuario puede establecer los permisos para que las funciones protegidas sean soportadas. Cuando se establece este nivel, el emulador no trabaja como un dispositivo real, ya que las restricciones ya no están manejadas por el emulador.

Para cambiar los niveles de seguridad del emulador y permitir el acceso de las aplicaciones desarrolladas a los recursos, se debe seguir los siguientes pasos:

- Abrir el archivo security_config.xml que se encuentra en la siguiente ruta: <SDK_Home>\devices\<Device>\bin\ security_config.xml
- Buscar en las primeras líneas el nombre de dominio que por defecto se encuentra establecido como “untrusted” y cambiarlo a “manufacturer” por ejemplo. <domain name="manufacturer"/>
- Reiniciar el entorno para que los nuevos permisos sean cargados en el emulador.

Con esto las aplicaciones desarrolladas serán tratadas como si fueran producidas por Nokia y se les pondrá ningún tipo de restricción. Pero si lo que se desea es mantener el

nombre dominio y solo permitir el acceso a una o algunas funciones se puede hacer lo siguiente:

- Buscar la función a la cual se le van a modificar los permisos en las etiquetas <function name=" ">. Entre las funciones están:
  Network Access
  Auto-start
  Messaging
  Connectivity
  Multimedia Rec.
  Location
  Landmark
  no group
  Read User Data Access
  Write User Data Access
  Smart Card Communicatio
  Authentication
- Modificar los permisos cambiando los números de las etiquetas del dominio actual (untrusted) teniendo en cuenta que entre menor sea el número establecido menor será el nivel de seguridad (el nivel de seguridad maneja números entre 0 y 16).
  Por ejemplo para permitir el acceso a la lectura y escritura en el Record Store del emulador cambiamos la etiqueta "untrusted mode" de las funciones Read User Data Access y Write User Data Access de la siguiente manera:
  <untrusted mode="1"/>
- Reiniciar el entorno para que los nuevos permisos sean cargados en el emulador.
  Con esto se le permitirá a las aplicaciones emuladas acceder en lectura y escritura al Record Store.

### B1.9 Uso del Wireless Toolkit con SIP

Hasta el momento el SUN Wireless Toolkit no soporta el API JSR 180. Pero con algunos cambios esta puede ser usada para compilar y empaquetar MIDlets que utilizan el API SIP.

Es importante avisar que con este cambio no se puede utilizar la emulación en los terminales del Wireless Toolkit, ya que las MIDlets que usan el API JSR180 no pueden ejecutarse con el Wireless Toolkit.

Los siguientes pasos muestran como el SUN Wireless Toolkit puede ser modificado, para que soporte la compilación de MIDlets que usan el JSR180. Se asume que el Wireless Toolkit esta instalado en el directorio C:WTK22. La clave de estos pasos es incluir las clases del JSR 180 en la librería midpapi20.jar del WTK. Esto realmente puede hacerse de varias formas, a continuación se ilustran los pasos de una:

- Abrir c:\WTK21\lib\midpapi20.jar con un descompresor de archivos, por ejemplo winzip.
- Abrir c:\<SIPA_ROOT>\lib\sipa1_0_1.jar con otro descompresor, winzip.
- Copiar las clases de sipa1_0_1.jar y péguelas en midpapi.jar
- Cerrar los archivos

El Wireless Toolkit esta listo para compilar MIDlets, recordando que no se puede usar el termina (RI emulador JSR 180) ya que no soporta todas las APIs disponibles en el WTK. Para asegurarse de que no se usaran las APIs que no son soportadas, se puede modificar el proyecto desde: Project / Settings / selección de APIs como se muestra en la figura 31. La idea es incluir solo las APIs del MIDP 2.0 y el CLDC 1.1 (o 1.0) del WTK.

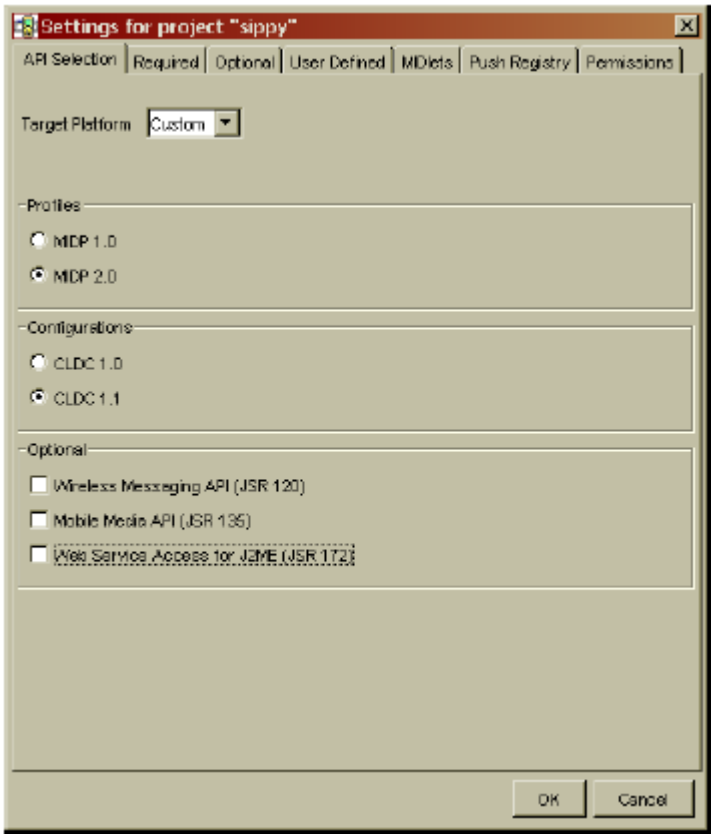


Figura 31Interfaz para configurar el proyecto.

Ahora se puede usar normalmente las herramientas 'Build' y Project / Package / create package del WTK. El permiso javax.microedition.io.Connector.sip debe ser aun adicionado manualmente en el archivo JAD.

La MIDlet empaquetada puede ser ejecutada directamente en el emulador RI JSR180 desde el directorio project/bin con el siguiente comando de línea de comandos:

c:\<SIPA_ROOT>\midp\bin\sipa-midp –classpath c:\WTK21\apps\my_project\bin my_project.jar MyMidlet [2].

## B2 DESARROLLOS SIP EN C++

La descripción de los pasos para preparar un Entorno de Desarrollo e implementar aplicaciones SIP en C++ se hicieron con algunas versiones disponibles de herramientas para el sistema operativo Windows. Si se utilizan versiones diferentes de estas herramientas los pasos a seguir pueden variar, pero la mecánica de obtener un IDE

completo que contenga los componentes necesarios para desarrollar aplicaciones SIP en C++ y la forma de crear una aplicación será similar. A continuación se describen los requerimientos necesarios para dos IDEs conocidos.

### B2.1 Instalación de las Herramientas Necesarias para Trabajar SIP con C++ en Eclipse

A continuación se describen brevemente los pasos mas importantes para la instalación y configuración del IDE Eclipse para desarrollar aplicaciones móviles en C++ con SIP.

- Revisar las especificaciones de la versión de Eclipse a instalar. Por ejemplo, para Eclipse 3.1 las especificaciones recomiendan tener instalado Microsoft® Windows® XP (con Service Pack 2) o Microsoft® Windows® 2000 (con Service Pack 4) para los equipos con sistema operativo Windows.
- Instalar un JDK o JRE de Java con versión igual o superior a la requerida por Eclipse. Para la versión de Eclipse elegida se instalo el JRE de Java 1.4.2_10
- Instalar Eclipse siguiendo los pasos del asistente de instalación.
- Instalar Active Perl. Se instalo ActivePerl 5.6.1
- Instalar Carvide para Eclipse. Durante la instalación Carvide se agrega a Eclipse y crea un acceso directo para entrar al entorno de desarrollo Eclipse con Carvide. El acceso directo tiene un nuevo icono con el nombre de Carbide.c++. La versión instalada de Carvide es la 2.0
- Instalar los SDKs de emulación de la serie 60 de Nokia con soporte SIP o un SDK sin soporte SIP y luego el plug-in SIP. Por ejemplo se pueden instalar los SDKs: S60-SDK-0548-3.0-f.3.215f, s60_2nd_sdk_fp3 y s60_sdk_v2_0. Los SDKs durante su instalación detectan la presencia del Carvide y se adicionan como SDKs de trabajo, por lo cual no es necesaria ninguna configuración adicional. [3]

Ahora Eclipse ya cuenta con lo necesario para desarrollar aplicaciones Móviles con C++ que usen SIP.

## B2.2 Crear un Proyecto SIP en C++ con Eclipse

Asumiendo que se tiene instalado el entorno de desarrollo Eclipse con todos los paquetes necesarios para poder generar proyectos en C++ con SIP (Ver Instalación de las herramientas necesarias para trabajar SIP con C++ en Eclipse), se siguen los siguientes pasos:

- Para abrir el asistente de creación de proyectos seleccionar: *Archivo/Nuevo/Proyecto C++ Symbian OS* como se muestra en la figura 32.

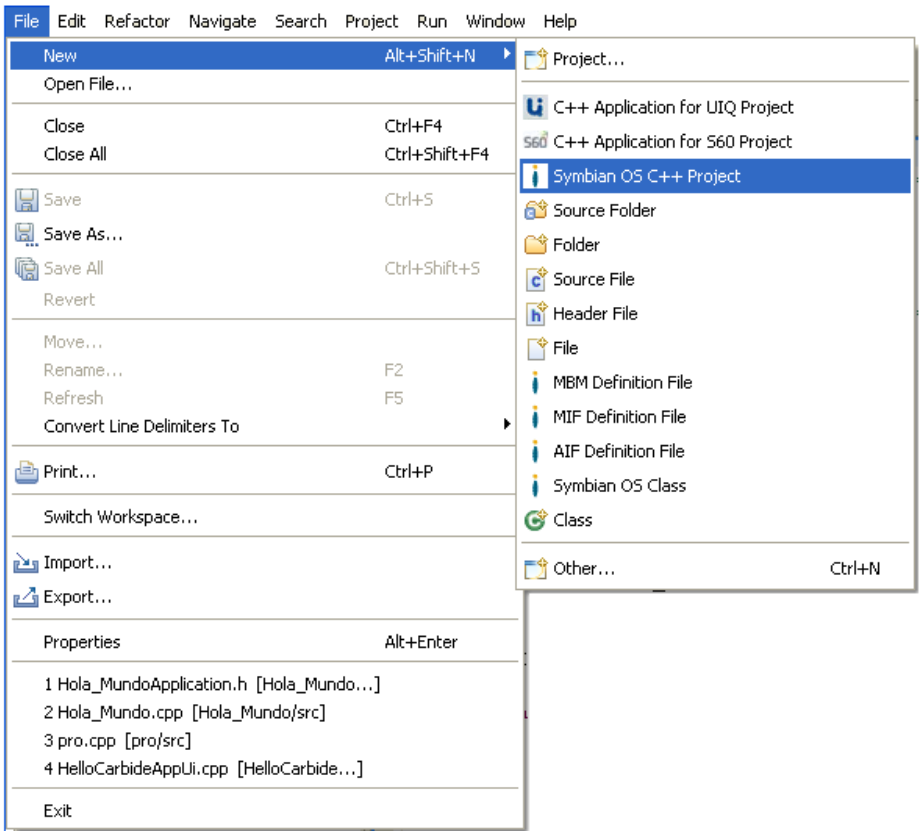


Figura 32. Creación de un proyecto en C++ para Symbian OS sobre Eclipse.

- El asistente para la creación de proyectos en C++ para Symbian OS mostrada en la figura 33 se abre y solicita el ingreso del nombre del proyecto. Después de ingresar el nombre se elige el botón *Siguiente>.*

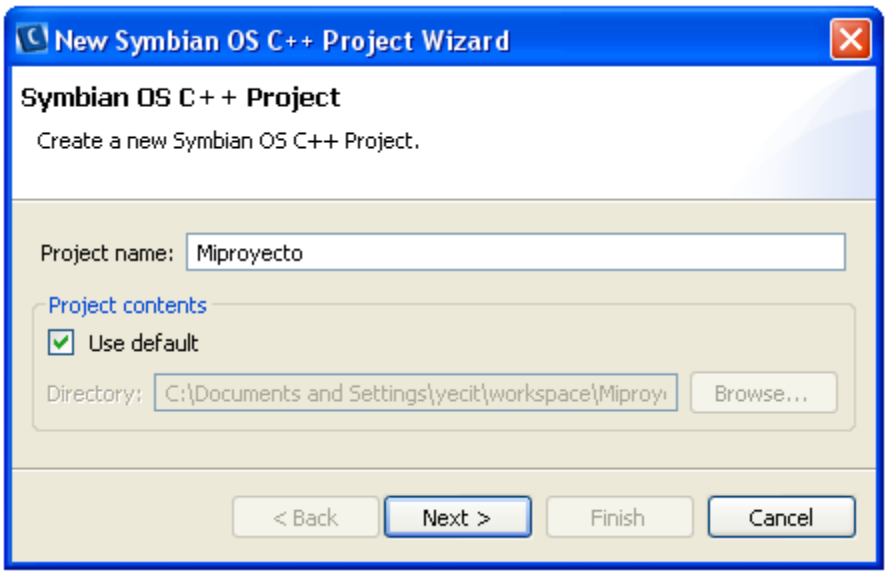


Figura 33. Asistente para la creación de proyectos en C++ para Symbian OS.

- En la pantalla de plantillas mostrada en la figura 34 se elige “S60 3.x GUI Application” de “Symbian Executable Project”. Se elige el botón *Siguiente>.*

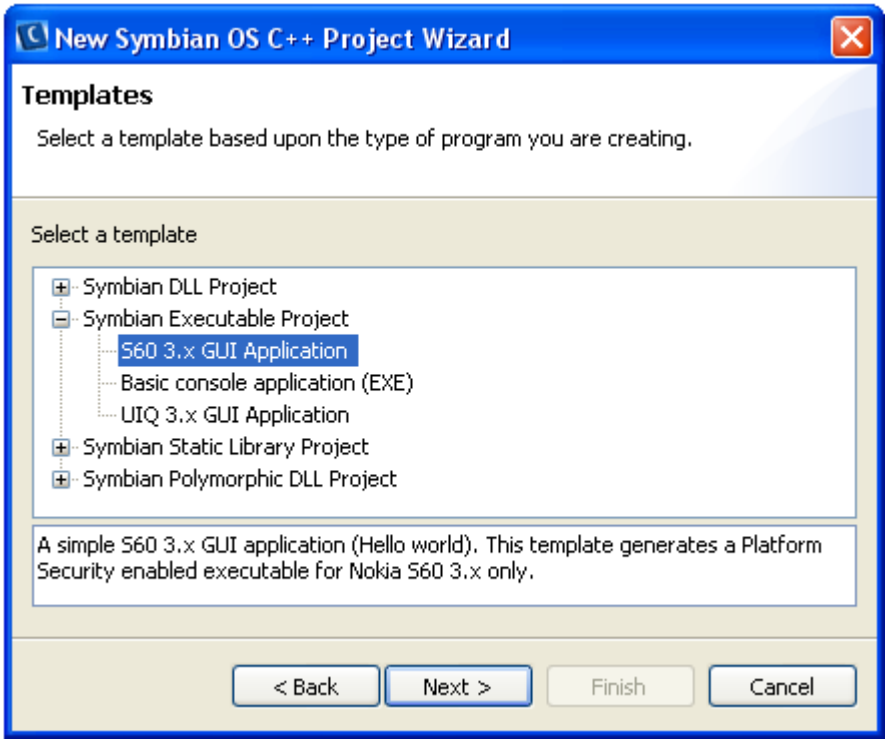


Figura 34. Plantillas de aplicación.

- Ahora el asistente presenta los SDKs disponibles para llevar a cavo el proyecto. Se elige el instalado para la serie 60 de Nokia (el SDK S60-SDK-0548-3.0-f.3.215f instalado aparece como S60_3rd). Se da *Finalizar* para completar la creación del proyecto. Esta interfaz se muestra en la figura 35.

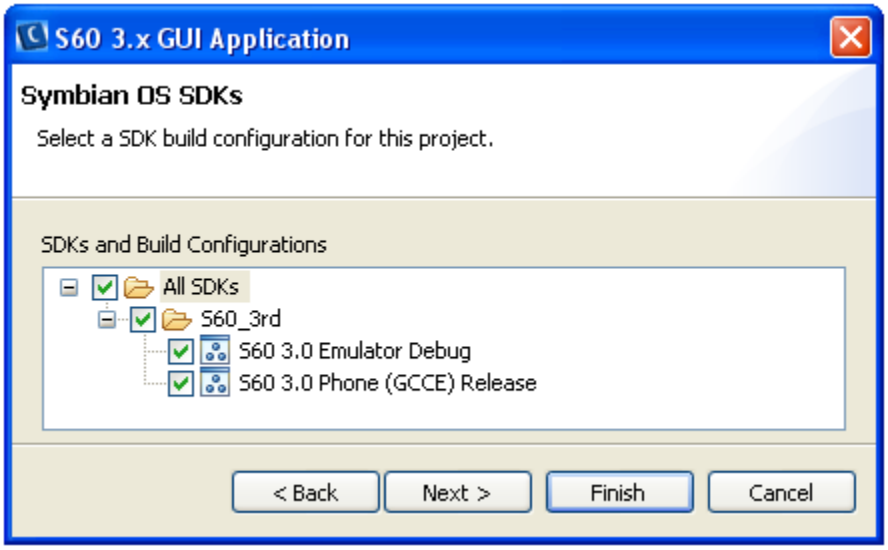


Figura 35. Ventana de presentación de los SDKs instalados.

El proyecto queda creado con todas sus carpetas y archivos necesarios, que pueden ser visualizados en la ventana de proyectos C/C++ como se muestra en la figura 36. [4]

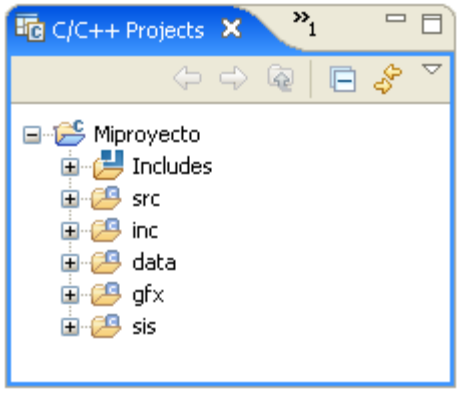


Figura 36. Ventana de proyectos C/C++

### B2.3 Emular un Proyecto SIP en C++ con Eclipse

Para correr aplicaciones C++ con SIP creadas sobre Eclipse por primera vez, se debe iniciar por "construir" la aplicación, para esto se elige la opción *Construir todo* del menú desplegable *Proyecto* como se muestra en la figura 37.

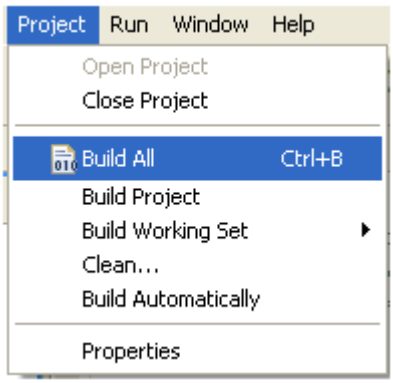


Figura 37. Construcción de la aplicación.

Si la aplicación queda correctamente construida, agregara dos elementos mas al explorador de proyectos C/C++ como los que se muestra en la figura 38.

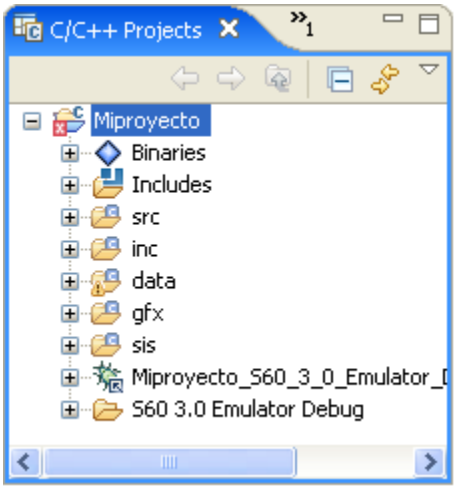


Figura 38. Explorador de proyectos C/C++ después de construir la aplicación.

Ahora se procede a crear una configuración de ejecución desde la opción "Ejecutar" que se encuentra en el menú desplegable *Ejecutar* o dando clik derecho del ratón sobre el proyecto, en la ventana de proyectos C/C++ se encuentra la opción *Ejecutar como,* que abre un sub-menú, el cual contiene esta opción como se muestra en las figuras 39 y 40.

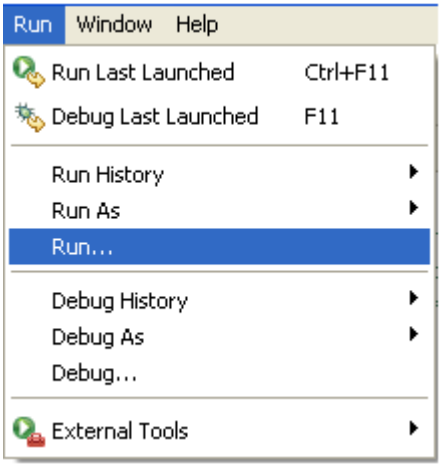


Figura 39. Opción Ejecutar desde el menú desplegable Ejecutar.

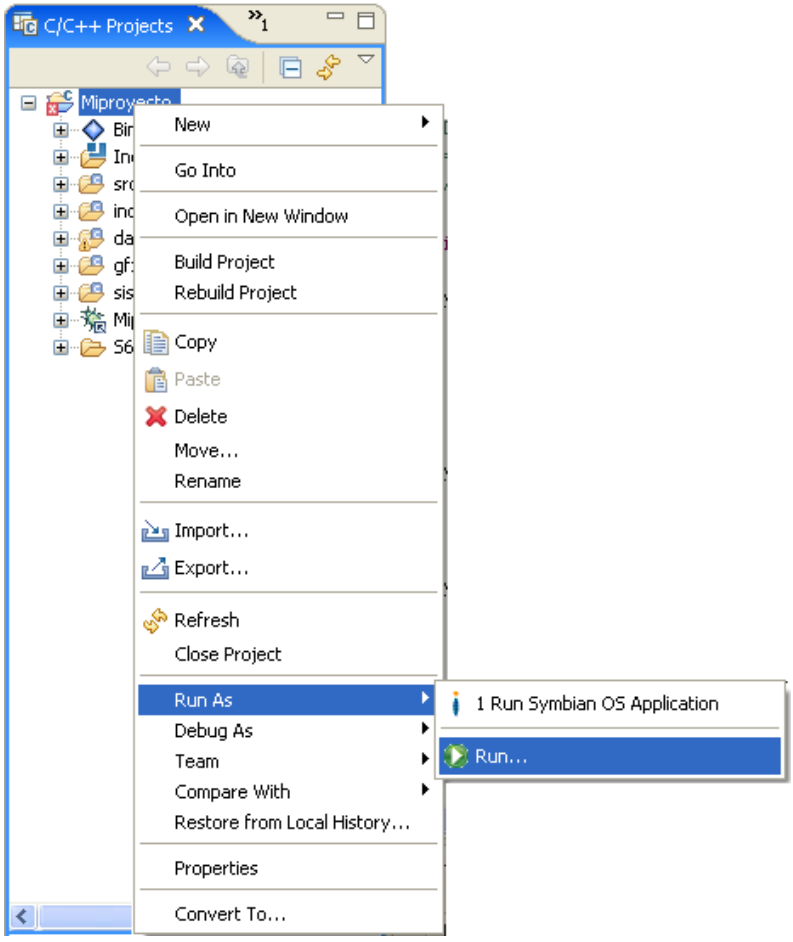


Figura 40. Opción Ejecutar desde el proyecto.

Después de seleccionar *Ejecutar* se abre la ventana de creación, gestión y ejecución de configuraciones que se muestra en la figura 41, la cual por defecto y si existe, contiene la ultima configuración utilizada y que puede pertenecer a otro proyecto. Por esto se debe crear por lo menos una nueva configuración para el proyecto actual. Se selecciona el botón *Nueva* y el asistente adiciona una nueva configuración con el nombre "configuración_nueva". Si se desea el nombre de la configuración puede ser cambiado en el campo correspondiente.

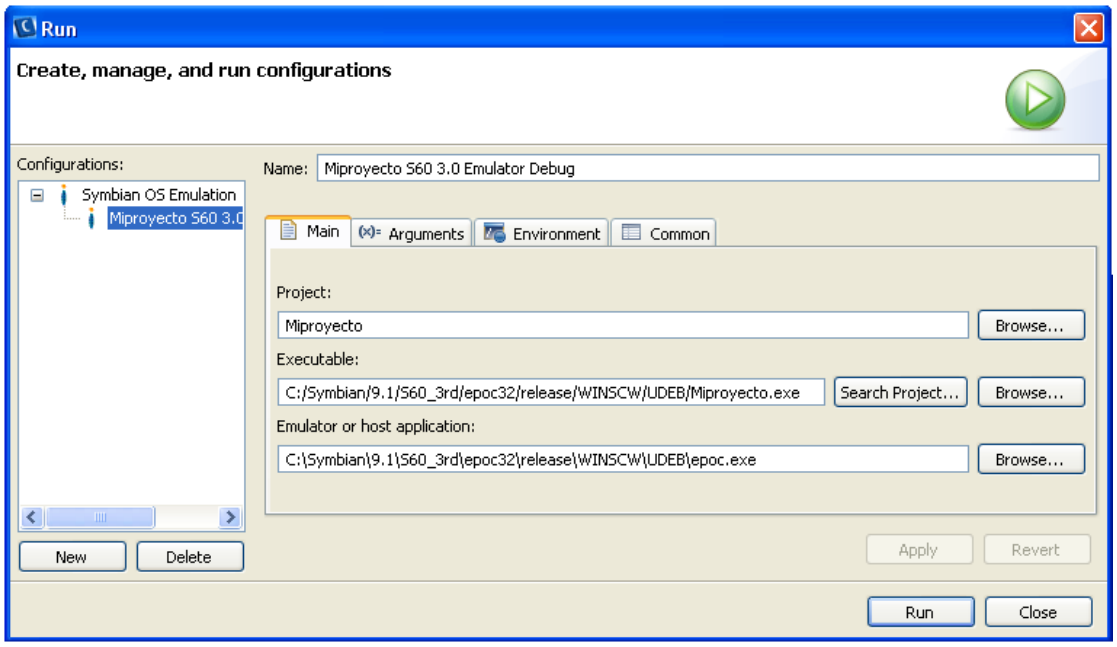


Figura 41. Ventana de creación, gestión y ejecución de configuraciones.

Ahora teniendo activo el tab "Principal" se ingresa o escoge el nombre del proyecto que se desea emular (En el ejemplo es "Miproyecto") sobre el campo "Proyecto" y en el campo "Ejecutable" se ingresa o escoge el nombre de la Midlet. En el tab "Emulación" y en el campo "Ejecutable" debe tener el mismo nombre del proyecto mas la extensión .exe. Por ultimo se guarda la configuración con el botón *Aplicar* y se comienza la emulación con *Ejecutar.* La creación de esta configuración sirve para todas las veces que se desee realizar la emulación, por lo tanto las próximas emulaciones podrán ser iniciadas desde cualquier opción "Ejecutar" disponible, como la que se encuentra en la barra de herramientas, que comienza la emulación con la configuración ya creada sin necesidad de abrir la ventana de creación, gestión y ejecución de configuraciones. La figura 42 muestra el emulador ya lanzado.

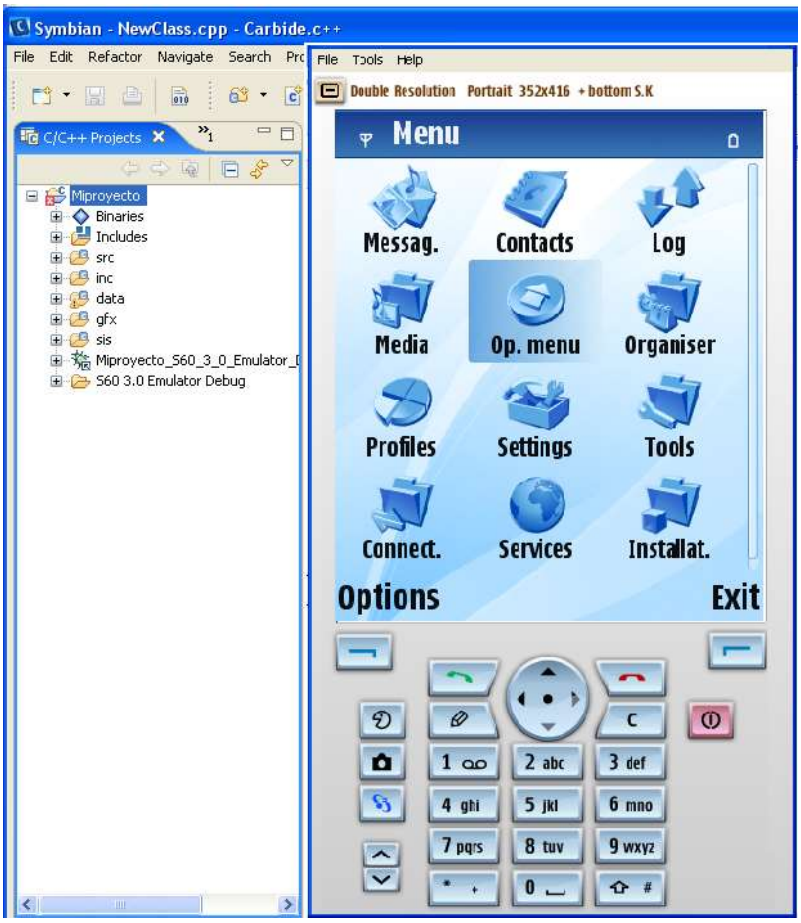


Figura 42. Emulación de una aplicación C++ en Eclipse.

## B2.4 Instalación de las Herramientas Necesarias para Trabajar SIP con C++ en Visual Studio .NET

A continuación se describen brevemente los pasos mas importantes para la instalación y configuración de Visual Studio .NET para desarrollar aplicaciones móviles C++ con SIP.

- Instalar un JDK o JRE de Java con versión igual o superior a la requerida por el (los) JDKs de emulación. Se instaló el JRE de Java 1.4.2_10
- Instalar WinPcap (Requerimiento de Active Perl 5.6.1). Por ejemplo WinPcap 3.1.
- Instalar Active Perl. Se instaló ActivePerl 5.6.1
- Instalar Visual Studio .NET siguiendo el asistente de instalación (Generalmente requiere igual o superior a la 2003), solo se necesita J# .NET, C# y C++.
- Instalar Carvide 1.0 para Visual Studio. [5]

- Instalar los SDKs de emulación de la serie 60 de Nokia con soporte SIP o un SDK sin soporte SIP y luego el plug-in SIP. Por ejemplo se pueden instalar los SDKs: S60-SDK-0548-3.0-f.3.215f, s60_2nd_sdk_fp3 y s60_sdk_v2_0.

Ahora Visual Studio .NET ya cuenta con lo necesario para desarrollar aplicaciones Móviles que usen SIP.

### B2.5 Crear un Proyecto en C++ con SIP en Visual Studio .NET

Para crear un nuevo proyecto en Visual Studio .NET para Symbian se ejecutan los siguientes pasos:
Se selecciona *Archivo/Nuevo proyecto* como se muestra en la figura 43.

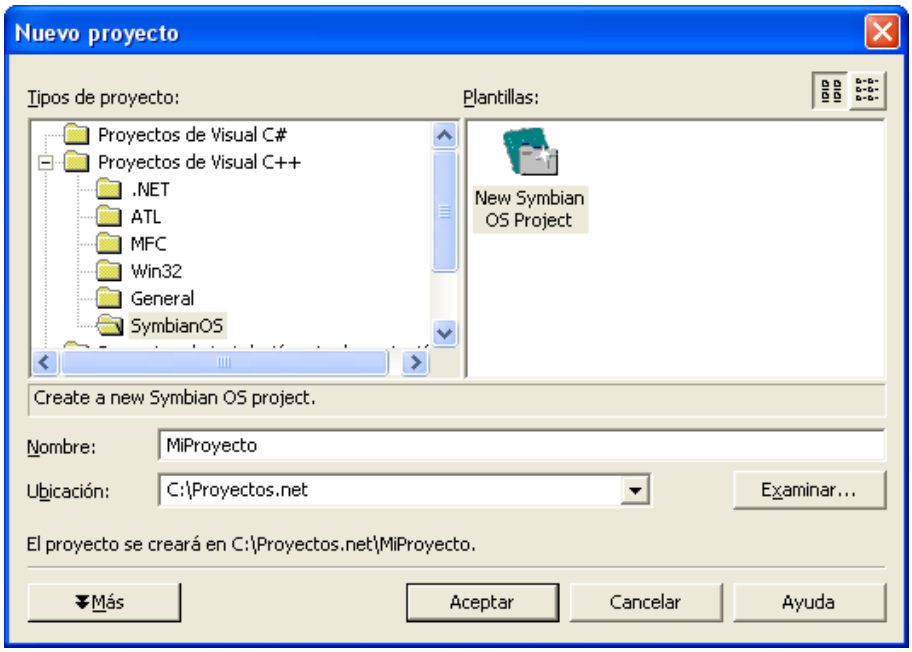


Figura 43. Creación de un proyecto en visual Studio .NET.

En la ventana de nuevo proyecto se ingresa el nombre del proyecto y la ruta de ubicación del directorio principal del proyecto. Luego se elige “Aceptar”, como se muestra en la figura 43.

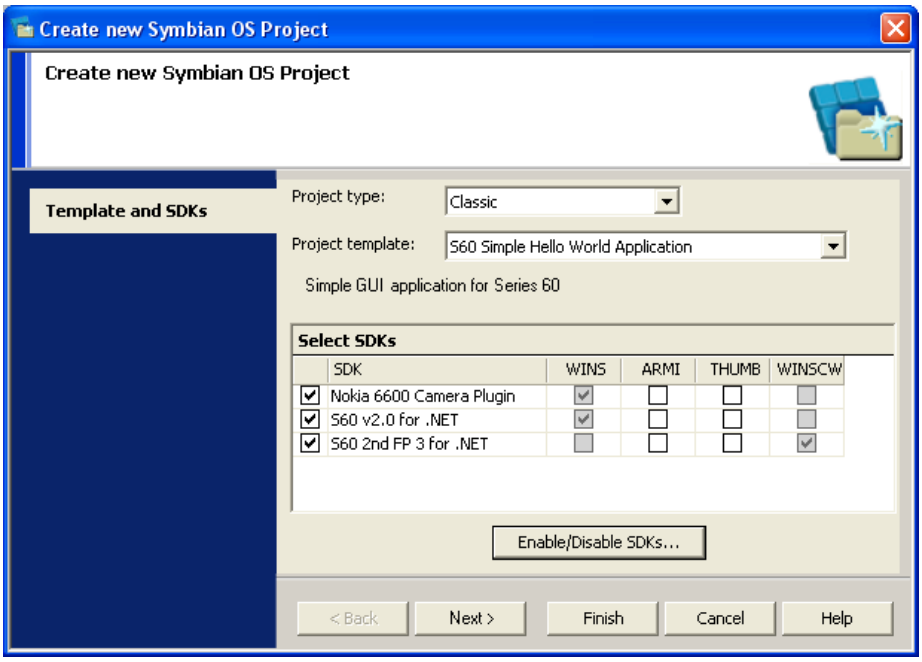


Figura 44. Interfaz para la elección del tipo de proyecto, plantilla y SDKs adicionales.

El asistente presenta una ventana para elegir el tipo de proyecto y la plantilla, estas dos se eligen como se muestra en la figura 44. En la misma ventana se puede elegir los SDKs de soporte adicional que se tengan instalados, por ejemplo el plugin de la cámara.

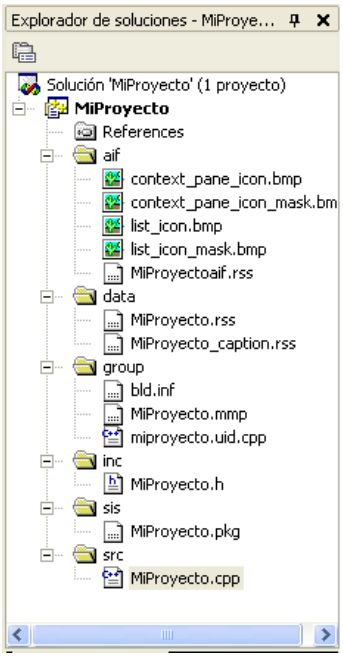


Figura 45. Árbol de archivos de un proyecto creado en Visual C++ .NET.

Finalmente se elige el botón finalizar y el asistente cierra la ventana de creación del proyecto y muestra el árbol de archivos creados como se muestra en la figura 45.

### B2.6 Emular un Proyecto en C++ con SIP en Visual Studio .NET

Para emular un proyecto C++ en Visual Studio .NET, solo basta con elegir la opción iniciar del menú desplegable, depurar o presionar la tecla F5 como se muestra en la figura 46.

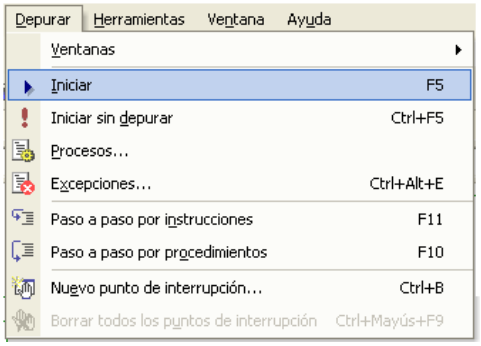


Figura 46. Opción Iniciar para emular un proyecto.

Luego de iniciar la emulación se desplegará la interfaz de un equipo emulado como se muestra en la figura 47. [6]


Figura 47. Emulador lanzado.

## B3 SERVIDORES PROXY SIP

A continuación, se presentan algunos servidores Proxy SIP que pueden ser utilizados para trabajar las aplicaciones móviles SIP P2P desarrolladas.

### B3.1 SIP Server Emulator

Es un emulador de servidor SIP que viene como herramienta adicional al instalar cualquiera de los SDKs para la plataforma de la serie 60. Puede actuar como un servidor Proxy o como un servidor de registro para una lista seleccionada de dominios o direcciones IP. Cuando el emulador de servidor SIP es iniciado, se listan dos dominios predeterminados y pueden ser agregados o eliminados nuevos dominios, sobre los cuales actúa como servidor.

El emulador escucha mensajes SIP en los puertos TCP y UDP seleccionados. El puerto predefinido para ambos protocolos es el 5060, pero puede ser modificado.

El emulador de Servidor SIP puede actuar en dos modos:

En modo Proxy y Registrador: en el cual actúa como un servidor de registro SIP y servidor Proxy. En este modo registra las direcciones SIP de los UAs y responde a sus peticiones.

En el modo de respuesta Manual: el emulador de servidor SIP responde a los mensajes SIP entrantes con la respuesta seleccionada en lugar de remitir los mensajes automáticamente. Las respuestas manuales pueden ser elegidas de una lista preestablecida de respuestas. Un conjunto de escenarios o situaciones con sus respuestas SIP típicas están disponibles para su elección. Cada escenario de respuesta SIP contiene cadenas de mensajes con uno o más mensajes SIP; por ejemplo un escenario de error podría contener los mensajes “100 que se esta Intentando” y “404 no Encontrado”. Entre los escenarios disponibles que se encuentran en el emulador están: *Bad request, Method not allowed, Ok, Ok trying busy here, Ok trying not found, Ok trying ringo k,Server internal error, service unavailable, trying busy everywere, trying busy here, trying decline, trying does not exist anywere, trying forbidden,trying fordware ring ok, trying gone, trying insufficient bandwith, trying moved permanently, trying not acceptable, trying not found, trying request timeout, trying ring ok y trying unauthorized.* [6]

La figura 48 muestra la interfaz del SIP Server.

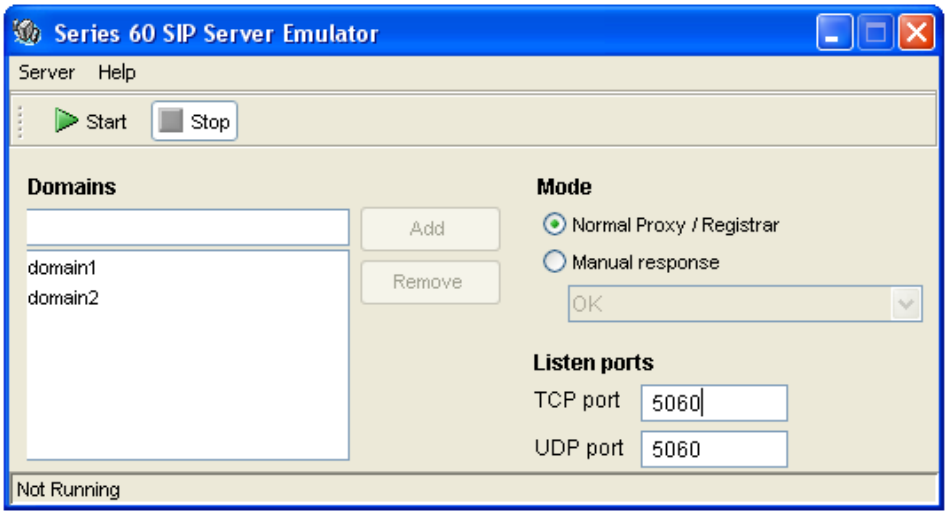


Figura 48. Interfaz del SIP Server.

### B3.2 Ondo SIP Server

Es un servidor SIP que proporciona capacidades de registro, Proxy, NAT trasversal y autenticación. Es un software creado por Brekeke, que entrega una versión de prueba totalmente operativa. Funciona sobre un navegador Web y presenta un completo conjunto de características importantes y opciones de configuración, que pueden ser de mucha ayuda a la hora de realizar un desarrollo SIP. El Servidor OnDO SIP, tiene las siguientes funciones principales:

#### B3.2.1 Ruteo de llamadas

El servidor dirige las solicitudes SIP de un UA u otros servidores SIP, a la dirección SIP URI más apropiada basada en los datos de registro. Especificando la asignación de la ruta de llamada establecida en el plan de marcación, también se puede priorizar la ruta de llamada. Si la asignación de ruta es resuelta exitosamente en el servidor, un usuario puede establecer una llamada incluso cuando la dirección SIP URI final es desconocida para el llamante. Usando expresiones regulares, se puede crear un plan de marcación que analiza las cabeceras o las direcciones IP de los paquetes SIP de las llamadas.

### B3.2.2 Registro

El OnDO SIP Servidor recibe las solicitudes de Registro de los UAs SIP, y actualiza su base de datos. La SIP URI de la solicitud de registro se agrega a la base de datos de registro, como dirección de usuario.

### B3.2.3 NAT Trasversal

Cuando el llamante y el llamado se localizan en diferentes redes, el Servidor OnDO SIP puede conectar las llamadas volviendo a escribir los paquetes SIP apropiadamente. Es común tener una dirección IP local y privada dentro de un entorno LAN, lo que hace necesario contar con el servicio NAT trasversal, para cuando un usuario local necesita establecer una conexión con otro usuario de la red IP global (Internet). Dependiendo de la situación, el Servidor OnDO SIP puede relevar los paquetes RTP para prevenir perdidas de voz o de datos de los medios de comunicación. La característica NAT trasversal ofrecida en el Servidor OnDO SIP soporta: servidor y UAs SIP localizados dentro del mismo cortafuego y UAs SIP localizados en el otro lado de un cortafuego de una red remota. La figura 19 muestra la interfaz del servidor Ondo SIP.


Figura 49. Interfaz del Ondo SIP Server.

### B3.3 NIST-SIP Server

Es un Servidor Proxy SIP de fuente abierta creado por el Instituto Nacional de Estándares y Tecnología (NIST) para ser usado principalmente como herramienta de desarrollo o como ejemplo de implementación SIP sobre JAVA (JAIN SIP 1.1). El Proxy también funciona como servidor de registro SIP y servidor de presencia SIP.

Entre las capacidades con las que cuenta el servidor están:

#### B3.3.1 Servidor de Presencia

El Proxy puede actuar como un servidor de presencia y puede procesar solicitudes de notificación y suscripción. Si este parámetro es desactivado, el servidor simplemente remitirá las solicitudes a la siguiente decisión de asignación de ruta apropiada.

#### B3.3.2 Procesamiento de Cabeceras de Ruta y Registro de Ruta

El Proxy puede agregar y procesar cabeceras de ruta y de registro de ruta especificadas en una solicitud, esto permite el uso de múltiples Proxys en el establecimiento de una sesión.

#### B3.3.3 Ingreso de Registros

Permite especificar un conjunto de usuarios SIP que serán ingresados en el servidor en el momento de su inicio. Los usuarios a registrarse se ingresan en el archivo de configuración "registrations.xml" localizado en el directorio de configuración del servidor.

#### B3.3.4 Autenticación

El Proxy puede autenticar todas las solicitudes que recibe. El método de autenticación predefinido es DIGEST y es el único soportado hasta el momento.

#### B3.3.4 Capacidad de Responder

El Servidor Proxy esta en capacidad de responder a las solicitudes de invitación que reciba.

La interfaz principal del servidor NIST se muestra en la figura 50.


Figura 50. Interfaz del Servidor SIP NIST.

## BIBLIOGRAFIA


[1] Forum Nokia. Nokia Prototype SDK 4.0 Beta for JME™ User's Guide Version 1.7 for Windows. Septiembre de 2005. http://sigit.no-ip.info/japidocs/nokiasdk/ Nokia_Prototype_SDK_UsersGuide.pdf

[2] Nokia SIP API for J2ME Getting Started with RI Version 1.0.1. Diciembre de 2004. www.forum.nokia.com/ME_Developers_Library/GUID-9D6A8CE9-C288-463B-AF37-CB9B42C39636.pdf

[3] Nokia. S60 rd Edition SDK for Symbian OS: SIP Developer User's Guide Version 1.02. Diciembre de 2005. [Unidad donde se instaló el SDK3.0]\Symbian\9.1\S60_3rd\S60Doc\ s60_3_0_cpp_sip_usersguide_1_02.pdf

[6] Nokia. S60 3rd Edition: Simulation PSY Configurator User's Guide Version 2.0. Octubre de 2005. . [Directorio de instalación del SDK3.0]\Symbian\9.1\S60_3rd\S60Doc\ s60_simulation_psy_configurator_users_guide.pdf

[4] Nokia. Series 60 2nd Edition SDK for Symbian OS, Supporting Feature Pack 3 For C++ Getting Started Guide for CodeWarrior IDE Version 1.0. Mayo de 2005. http://sw.nokia.com/id/f38a75cc-a68c-45b6-96fe-963a15be8143/ S60_3_0_CPP_SDK_CW_GettingStarted_1.0.pdf

[5] Nokia. Carbide.vs Installation Guide for Visual Studio. Octubre de 2005. http://sw.nokia.com/id/72b419b2-5e73-4d95-8f37-614ee3f94f03/ Carbide_vs_201_InstallationGuide

[6] Nokia. S60 Platform SDKs for Symbian OS, for Java MIDP http://sw.nokia.com/id/70ec47ef-3fc4-4374-a03d-8c39836fd7c3/DS_S60_MIDP_SDK.pdf