Capítulo 6

Kit de desarrollo de software SDK

En el siguiente capítulo se describe qué es un SDK, para qué se utiliza, y los diferentes tipos de SDKs disponibles para el sistema operativo Symbian. Al igual que en el Capítulo cuarto, se han usado las webs de Symbian [9] y de Forum-Nokia [12], y los manuales Getting Started on Symbian OS [8] y S60 Platform: Introductory Guide [13].

Un software development kit (SDK) es un conjunto de herramientas de desarrollo que permiten a un programador crear aplicaciones para un sistema concreto. Es como una interfaz de programación de aplicaciones o API creada para permitir el uso de cierto lenguaje de programación. También puede incluir hardware sofisticado para comunicarse con un determinado sistema embebido. Las herramientas más comunes incluyen soporte para la detección de errores de programación como un entorno de desarrollo integrado y otras utilidades. Los SDKs frecuentemente incluyen códigos de ejemplo y notas técnicas de soporte.

6.1 ¿Qué SDK se debe usar?

Los teléfonos basados en Symbian OS pueden tener un aspecto distinto y emplear diferentes interfaces de usuario (UI). Para escribir aplicaciones para una UI en concreto se debe seleccionar un SDK adecuado.

Figura 22. Modelos de teléfonos con diferentes interfaces de usuario.

Las principales plataformas para Symbian OS son:

- Nokia Series 60

  La interfaz de usuario ha sido diseñada para smartphones que poseen displays pequeños y donde la inserción de datos de usuario es llevada a cabo por medio del teclado numérico. Nokia basó la Serie 60 en el patrón de diseño conocido como Pearl, aunque Nokia realizó modificaciones significativas. Esta serie es popular para smartphones de bajo coste. Nokia ha concedido licencias a otros fabricantes de smartphones.

La mayoría de los teléfonos con Symbian OS emplean la interfaz de usuario S60. Esta ha sido revisada varias veces por lo que habrá que elegir la combinación adecuada de SDK S60 y Symbian OS:

- S60 3rd Edición Feature Pack 2 – Symbian OS v9.3

- S60 3rd Edición Feature Pack 1 – Symbian OS v9.2
- S60 3rd Edición – Symbian OS v9.1
- S60 2nd Edición Feature Pack 3 – Symbian OS v8.1
- S60 2nd Edición Feature Pack 2 – Symbian OS v8.0a
- S60 2nd Edición Feature Pack 1 – Symbian OS v7.0s
- S60 2nd Edición – Symbian OS v7.0s
- S60 1st Edición – Symbian OS v6.1

El SDK usado es S60 3rd Edition FP1 y puede descargarse desde la web de Forum-Nokia [12].

- UIQ

Este sistema tiene su origen en el patrón de diseño de Symbian conocido como Quartz. Está diseñado para smartphones con pantalla táctil VGA y sin teclado. Para la inserción de datos por parte del usuario cuenta con un teclado por pantalla y sistemas de reconocimiento de escritura. Sony Ericsson y Motorola usan UIQ. También existen varias versiones:

- UIQ 2.0 – Symbian OS v7.0
- UIQ 2.1 – Symbian OS v7.0
- UIQ 3.0 – Symbian OS v9.1

Hay otros SDKs disponibles pero son menos usados, como la plataforma Nokia Series 80, que fue diseñada basándose en la plataforma Cristal para teléfonos con pantalla VGA, teclado y botones con funciones dinámicas a lo largo del lado derecho de la pantalla.

6.2 Plataforma Nokia Series 60

La plataforma S60 es una propuesta construida para smartphones [13]. Soporta una pantalla en color y un interfaz intuitivo, e incorpora comunicaciones de vanguardia y tecnologías de dispositivo de seguridad que interactúan y responden con rapidez.

Con el SDK se pueden desarrollar e implementar aplicaciones de la plataforma S60 en el ordenador y el SDK incluye un emulador que imita las funciones de un smartphone, con el que se puede ver y probar las aplicaciones antes de instalar a un dispositivo real.

Figura 23. Vista del emulador de S60.

El SDK proporciona las herramientas necesarias, interfaces de programación de aplicaciones (API), y la documentación para que se puedan desarrollar nuevas aplicaciones S60 escritas en C + +.

En el apartado anterior se detallaron las diferentes versiones de S60 que existen. Se verán a continuación las nuevas características que se incluyeron en la tercera edición del SDK de S60, ya que es la versión usada en este proyecto:

- Nuevo kernel: S60 3rd edición se basa en Symbian OS v9.x, que cuenta con un nuevo núcleo en tiempo real llamado EPOC. EPOC era el nombre original para el sistema operativo Symbian. El nuevo kernel permite a los fabricantes crear dispositivos con una arquitectura de un solo chip, reducir facturas de materiales, y ofrecer los dispositivos S60 a un usuario de tipo medio.

- Nuevo binario: Symbian OS v9.x se basa en un nuevo binario, la interfaz binaria de aplicación (ABI) para la arquitectura ARM. Este binario ofrece importantes mejoras de rendimiento para aplicaciones que se ejecutan en los dispositivos S60. Este cambio produce una ruptura con todas las versiones

anteriores del sistema operativo Symbian y la plataforma S60. En consecuencia, las aplicaciones para versiones anteriores de la plataforma S60 necesitarán recompilar, como mínimo, antes de que se puedan ejecutar en S60 3 ª edición.

- Mayor seguridad: S60 3rd Edition introduce una plataforma de seguridad en Symbian OS, que proporciona una serie de API que requieren una solicitud para ser certificadas antes de poder acceder a las API. Estas características ofrecen protección contra el software malintencionado.

Las aplicaciones que se crea con el S60 SDK están diseñadas para ejecutarse en la plataforma S60 siendo un paquete completo de aplicaciones, con interfaz de usuario y herramientas de desarrollo basado en la tecnología Symbian OS. La arquitectura de la plataforma se muestra a continuación:

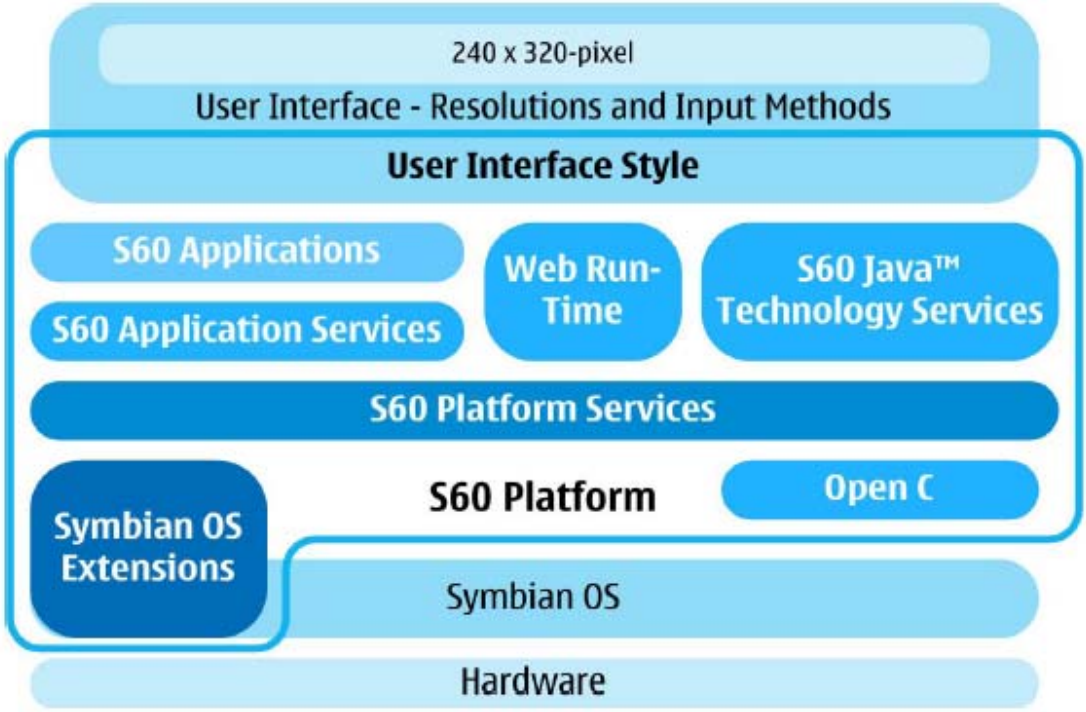

Figura 24. Diagrama de la arquitectura de S60.

El S60 de interfaz de usuario (IU), ha sido específicamente diseñado para un uso fácil con una sola mano. Desde el punto de vista del usuario, probablemente la característica más importante de la plataforma es su interfaz de usuario.

A continuación se describen algunas de las partes que aparecen en la arquitectura de la plataforma S60:

- Las extensiones del Symbian OS: son un conjunto de capacidades que permiten a la plataforma S60 interactuar con el hardware del dispositivo, como por ejemplo, funciones de alerta por vibración, iluminación del dispositivo o el estado de carga de la batería.

- Servicios de la plataforma S60: los fundamentales servicios que ofrece la plataforma incluyen:

    - Servicios de aplicación de framework: proporcionan las capacidades básicas para el lanzamiento de aplicaciones y servidores, gestión de estados persistentes, componentes de la interfaz de usuario.

    - Servicios de framework de interfaz de usuario: proporcionan el aspecto concreto de la interfaz de usuario y la manipulación de eventos de la IU.

    - Servicios gráficos: proporcionan capacidades para la creación de gráficos que se puedan mostrar en la pantalla.

    - Ubicación servicios: permiten que la plataforma S60 sea consciente de la ubicación de un dispositivo.

    - Servicios basados en la web: proporcionan servicios para establecer conexiones e interactuar con funcionalidades basadas en la web, incluyendo navegación, descarga de archivos o mensajería.

    - Servicios multimedia: proporcionan la capacidad para reproducir audio y vídeo, así como apoyo para el streaming y el reconocimiento de voz.

    - Servicios de comunicación: prestación de soporte a comunicaciones de área local y ancha, que van desde la tecnología Bluetooth a las llamadas de voz.

- Servicios de aplicación en S60: son un conjunto de capacidades que ofrecen las funcionalidades básicas para determinadas aplicaciones S60. Estos servicios son utilizados por las aplicaciones embebidas S60. Estos incluyen:

    - Servicios de aplicación PIM: proporcionan los elementos fundamentales de aplicaciones PIM, incluyendo contactos, calendario y tareas de gestión, así como funciones asociadas al Bloc de notas y al reloj.

    - Servicios de aplicaciones de mensajería: prestación de soporte para los varios tipos de mensajes, como el servicio de mensajes cortos (SMS), servicio de mensajes multimedia (MMS), correo electrónico, mensajes BIO (mensajes inteligentes), y de mensajería instantánea (IM).

    - Servicios de aplicaciones del navegador: proporcionan las capacidades para ver el contenido de la Web.

6.3 Estructura de aplicaciones gráficas en S60

La estructura de la interfaz gráfica de usuario proporciona funcionalidades de ejecución y de interfaz de usuario en las aplicaciones. Consta de tres niveles para facilitar la portabilidad entre estilos:

- Uikon: consiste en varias librerías que proporcionan servicios para la ejecución de aplicaciones, servicios de estado y gestión de eventos.
- Eikon: es un estándar que en la plataforma S60 ha sido implementado por Nokia. Junto con Avkon suministran la apariencia de IU S60.
- Avkon: proporciona una arquitectura de clases base, y controles específicos para usar en las aplicaciones. Estos controles son componentes representados en pantalla, como menús, diálogos, listas de selección,…

La base de Symbian OS para la interfaz de usuario es Uikon. Una de las más importantes bibliotecas es eikcore.dll, que contiene el esquema de clases de la interfaz de usuario como CEikApplication, CEikAppUI y CEikonEnv.

Tanto S60 como UIQ incluyen otras bibliotecas adicionales a la plataforma para proporcionar controles específicos, en S60 se tiene Avkon y en UIQ Qikon. Cada una de ellas tiene componentes específicos para sus plataformas pero debido a que tienen una base común, Uikon, también tienen a menudo APIs muy similares.

Al crear una aplicación de IU se necesitan clases heredadas de las clases bases que son proporcionadas por cada una de las plataformas, para poder hacer esto es necesario incluir los correspondientes ficheros de cabeceras y las bibliotecas donde se encuentran.

Independientemente de la plataforma elegida para obtener una interfaz de usuario gráfica se tiene que derivar de cuatro clases específicas, como también se ve en [14]:

- Application: esta clase sirve para definir las propiedades de la aplicación, y también para fabricar un documento nuevo y vacío. Un objeto de esta clase es lo primero en ser creado al iniciar una aplicación. Se usa el UID del objeto para identificar la aplicación.

- Document: un document representa el modelo de datos para la aplicación. Si la aplicación está basada en ficheros el document es el responsable en almacenar y restaurar los datos de la aplicación. Incluso si la aplicación no está basada en ficheros debe tener una clase tipo document aunque esa clase no haga mucho más que crear la App UI.

- App UI: la App UI es totalmente invisible. Esta crea la App view para manejar la interfaz gráfica. Además proporciona los medios para procesar comandos que deben ser generados por ejemplo por los elementos del menú del teléfono.

- App view: es un controlador cuyo propósito es mostrar los datos de la aplicación por pantalla y permitir al usuario interactuar con estos.

La siguiente tabla muestra la relación entre las clases de Symbian OS y S60:

Class	Generic Uikon class	S60 (Avkon)	S60 header file
Application	`CEikApplication` (which derives from `CApaApplication`)	`CAknApplication`	`aknapp.h`
Document	`CEikDocument` (which derives from `CApaDocument`)	`CAknDocument`	`akndoc.h`
Application UI	`CEikAppUi`	`CAknAppUi`	`aknappui.h`
View	`CCoeControl`	`CCoeControl`	`coecntrl.h`

Figura 25. Equivalencia entre clases S60 y Symbian.

A continuación se verán con más detalle cada una de las clases, aunque primero se ven dos funciones que tienen que ser implementadas por todas las aplicaciones Symbian OS para obtener una GUI. El propósito de estas líneas es crear una nueva aplicación:

```
LOCAL_C CApaApplication* NewApplication()
        {
        return new Cproyecto2Application;
        }

GLDEF_C TInt E32Main()
        {
        return EikStart::RunApplication(NewApplication);
        }
```

En la primera función se crea un nuevo ejemplo de la clase application, se devolverá una muestra o Null si no se puede crear, y la segunda función es el punto de entrada a la aplicación. Estas funciones se encuentran en un fichero .cpp con el nombre de la aplicación.

- La clase Application:

Suponiendo que se ha podido crear una muestra de la clase application, entonces se llamará a AppDllUid() lo que proporciona otro controlador de la identidad de la aplicación y con esto se tiene todo lo necesario para pasar a crear un objeto de la clase document.

```
CApaDocument* CnameprojectApplication::CreateDocumentL()
        {
        return CnameprojectDocument::NewL(*this);
        }

TUid CnameprojectApplication::AppDllUid() const
        {
        return KUidnameprojectApp;
        }
```

❖ La clase document:

Esta clase se construye y devuelve en la función CreateDocumentL que como se ha visto anteriormente es llamada en la clase application.

Cuando la aplicación se basa en ficheros, la clase document debe almacenar y recuperar la información necesaria mediante los métodos StoreL ( ) y RestoreL ( ), que están vacíos por defecto. En caso de no usar ficheros, lo único que se tendría que hacer es crear la primera fase del constructor de la clase App UI.

```
CEikAppUi* CnameprojectDocument::CreateAppUiL()
        {
        return new (ELeave) CnameprojectAppUi;
        }
```

❖ La clase Application UI:

La interfaz gráfica empieza en App UI, con dos funciones principales:

- Obtener los comandos de la aplicación.
- Distribuir las teclas correspondientes a los controles, incluyendo la vista principal de la aplicación, que es propiedad de App UI.

Un comando es simplemente una instrucción sin parámetros o alguna información acerca de dónde viene, que debe ejecutar el programa.

La segunda fase del constructor es un método importante, se muestra a continuación:

```
void CnameprojectAppUi::ConstructL()
        {
        BaseConstructL(CAknAppUi::EAknEnableSkin);
        iAppView = CnameprojectAppView::NewL(ClientRect());
        }
```

En el constructor se ha creado el objeto view y no hay que olvidar borrarlo en el destructor:

```
CnameprojectAppUi::~CnameprojectAppUi()
        {
        if (iAppView)
                {
                delete iAppView;
                iAppView = NULL;
                }
        }
```

La siguiente función se encarga de manejar eventos procedentes de diferentes fuentes, como barras de menús, o de teclas.

```
void CnameprojectAppUi::HandleCommandL(TInt aCommand)
        {
        switch (aCommand)
                {
                case EEikCmdExit:
                case EAknSoftkeyExit:
                        Exit();
                        break;

                case ECommand:
                        {
                        iAppView->Function();
                        }
                        break;
                default:
                        Panic(EnamepprojectUi);
                        break;
                }
        }
```

En un menú real o en el emulador se puede ver de la siguiente forma:

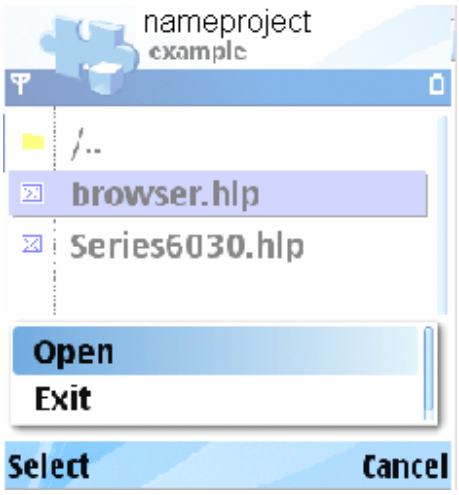

Figura 26. Vista del emulador.

Al elegir la opción "Open" se ejecutarán las instrucciones que se hayan programado en la función "Function ( )".

❖ La clase Application View

Esta clase es la interfaz tangible con la que los usuarios interactúan. En una aplicación sencilla se necesita una única clase view, que deriva de la clase base de Uikon, CCoeControl. Mientras que si la aplicación es más compleja serán necesarias más de una view, y estas deben ser creadas y destruídas en la clase App UI.

El principal objetivo de la Application View es mostrar por pantalla el texto que se haya programado. A continuación se muestra el código necesario para hacerlo.

Se crea una ventana, se ajusta su tamaño y se activa:

```
void CnameprojectAppView::ConstructL(const TRect& aRect)
        {
        CreateWindowL();
        SetRect(aRect);
        ActivateL();
        }
```

Función encargada de dibujar o escribir en la ventana, después de unos instantes se limpia la ventana:

```
void CnameprojectAppView::Draw(const TRect& /*aRect*/) const
        {
        CWindowGc& gc = SystemGc();
        TRect drawRect(Rect());
        gc.Clear(drawRect);

        }
```

Segunda fase de la construcción del objeto, la primera se realiza en App UI:

```
CnameprojectAppView* CnameprojectAppView::NewL(const TRect& aRect)
        {
        CnameprojectAppView* self = CnameprojectAppView::NewLC(aRect);
        CleanupStack::Pop(self);
        return self;
        }

CnameprojectAppView* CnameprojectAppView::NewLC(const TRect& aRect)
        {
        CnameprojectView* self = new (ELeave) CnameprojectAppView;
        CleanupStack::PushL(self);
        self->ConstructL(aRect);
        return self;
        }
```

6.4 Firmado de aplicaciones en S60

El firmado de aplicaciones es necesario en todas las aplicaciones S60 a partir de la 3ª edición. Sin firmar una aplicación esta no puede ser instalada en el terminal móvil.

Existen diferentes métodos a la hora de firmar aplicaciones Symbian. Los prerrequisitos necesarios para las diferentes opciones de firmado son los siguientes:

	Publisher ID Required	Independent Testing Required	IMEI Restrictions	For Commercial Distribution?
Open Signed Online	NO	NO	YES	NO
Open Signed Offline	YES	NO	YES	NO
Express Signed	YES	NO	NO	YES
Certified Signed	YES	YES	NO	YES

Figura 27. Requisitos para firmar aplicaciones.

El servicio de firma de aplicaciones usado es Open signed Online, accesible desde la página web de Symbian Signed proporcionada en [15]. Únicamente se pueden firmar de forma gratuita las aplicaciones que tengan algunas de las capacidades mostradas en la figura. Además de las capacidades, es necesario introducir el IMEI del teléfono para en el cual se instalará la aplicación.

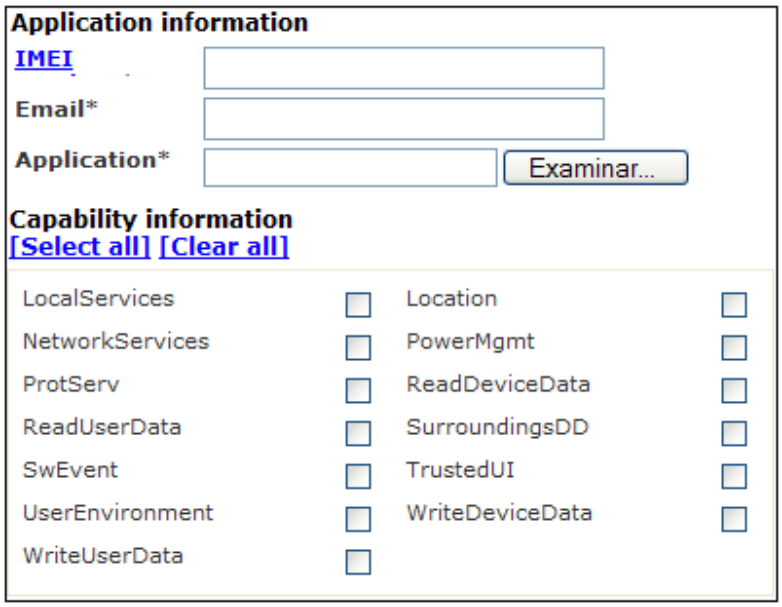

Figura 28. Información requerida por Symbian Signed.

Durante la fase de creación de la aplicación se le asignan las capacidades necesarias, en la aplicación diseñada en este proyecto han sido requeridas las siguientes capacidades:

- NetworkServices: capacidad requerida para marcar un número de teléfono o enviar un mensaje de texo.

- ReadUserData: acceso de lectura a los datos del usuario.

- ReadDeviceData: acceso de lectura a los datos del sistema.

Una vez introducidos todos los campos y un código de seguridad, se recibe un correo electrónico de confirmación en la cuenta que se ha proporcionado y un segundo e-mail con la aplicación firmada.

6.5 SDK S60 tercera edición

En este proyecto se ha requerido el diseño de una aplicación para smartphones que permita obtener la intensidad de la señal recibida en el terminal desde la estación base.

Para ello, se eligió el sistema operativo Symbian. En Symbian se pueden usar varios SDKs, entre los que destacan la plataforma S60 respaldada por Nokia y la plataforma UIQ respaldada por Sony Ericsson.

Tanto en S60 como en UIQ se podría haber diseñado la aplicación, sin embargo se ha elegido S60, ya que UIQ parece estar en declive.

Dentro de S60 se tienen varias versiones, como se ha visto al principio del capítulo. Es en la tercera edición donde aparece la biblioteca requerida para obtener el valor de la señal, por este motivo es la versión utilizada en el desarrollo del proyecto.