Programación Symbian SO. ¿!Misión imposible!?

Symbian: S.O. Para dispositivos móviles

Temario

- Historia Symbian.
- Opciones de Programación de Symbian OS.
- Python y Pys60
- Redes GSM y Localización.
- GPS y Python.

Historia Symbian SO.

- 1980-1997 : Psiloc, epoc32.

Historia Symbian SO.

- 1980-1997 : Psiloc, epoc32.
- 1998 : Se crea Symbian.

Historia Symbian SO.

- 1980-1997 : Psiloc, epoc32.
- 1998 : Se crea Symbian.
- 1998 – 2008 : Consolidación en el mercado.

Historia Symbian SO.

- 1980-1997 : Psiloc, epoc32.
- 1998 : Se crea Symbian .
- 1998 – 2008 : Consolidación en el mercado.
- 2008 : Nokia(47.9%), Ericcson (15.6%), SonyEriccson(13.1%),Panasonic(10.5%), Siemens(8.4%),Samsung(4.5%).

Historia Symbian SO.

- Octubre 2008 : Se crea la Fundación Symbian.

Historia Symbian SO.

- Octubre 2008 : Se crea la Fundación Symbian.

  - Nokia compra el 52 % de las acciones que le faltaban. Y crea la Fundación Symbian.

Historia Symbian SO.

- Octubre 2008 : Se crea la Fundación Symbian.

  - Nokia compra el 52 % de las acciones que le faltaban.
  - Comprometen su cooperación Sony Ericcson, Motorola ,Samsung, Vodafone…

Historia Symbian SO.

- Octubre 2008 : Se crea la Fundación Symbian.

  - Nokia compra el 52 % de las acciones que le faltaban.

  - Cooperaran Sony Ericcson , Motorola ,Samsung …

  - Se promete que en un plazo de 2 años Symbian SO. se convertirá en un proyecto Open Source.

Opciones de Programación en Symbian OS.

- C++.
  - Api complicada .
  - Curva de aprendizaje lenta.
  - Mayor tiempo de desarrollo.
  - Orientado a desarrolladores con gran expertiz en C++.

Opciones de Programación en Symbian OS.

- Python.
  - Multiplataforma.
  - Open Source.
  - Lenguaje interpretado de scripts.
  - Acceso a todas las funcionalidades del teléfono
  - Lenguaje oficial en google.

Opciones de Programación en Symbian OS.

- Para Symbian SO de la s60 3º edición. Se utiliza PyS60, la versión de Python para celulares.

Hello world!

- C++

bld.inf

```
PRJ_MMPFILES
helloworld.mmp
```

helloworld.mmp

```
TARGET helloworld.exe
TARGETTYPE exe
SOURCEPATH ..\src
UID 0
SOURCE helloworld.cpp
USERINCLUDE ..\inc
SYSTEMINCLUDE \epoc32\include
LIBRARY euser.lib
```

Hello world!

- C++

helloworld.cpp

```
#include <e32base.h>
#include <e32cons.h>
LOCAL_D CConsoleBase* gConsole;
// main function
LOCAL_C void MainL()
{
_LIT(KHelloWorldString, "Hello World\n");
gConsole->Printf(KHelloWorldString);
}
// Console Harness
LOCAL_C void ConsoleMainL()
{
_LIT(KConsoleTitle,"Hello World!");
gConsole = Console::NewL(KConsoleTitle, TSize(KConsFullScreen,
KConsFullScreen));
MainL();
}
// EPOC's main entry point
GLDEF_C TInt E32Main()
{
ConsoleMainL();
User::After(5000000);
return 0;
}
```

Hello World!

- Python(PyS60)

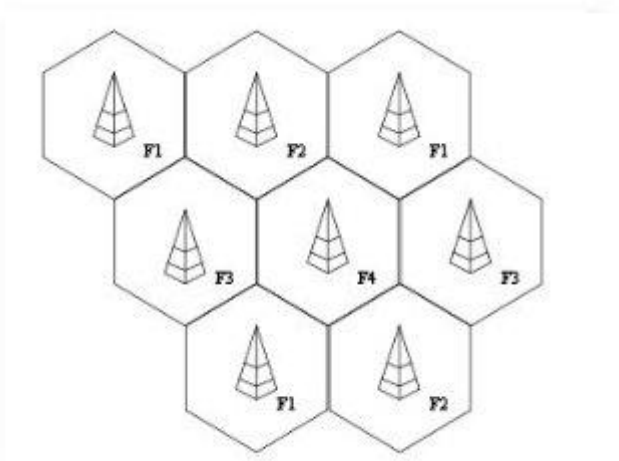
  hello.py

  ```
  import appuifw
  appifw.note(u’Hello World!’)
  ```

  Solo 2 líneas!!

Redes GSM y Localización.

- La celda es la unidad geográfica de la red.



Redes GSM y Localización

Redes GSM y Localización.

- En Pys60 existe el modulo location que entrega algunas parámetros de la celda a la que un equipo esta conectado en determinado momento.

Redes GSM y Localización.

- Country (MCC)
- Mobile Network Code (MNC)
- Location Area Code (LAC)
- Cell id (cellid)

Ejemplo: gsm.py

```
import appuifw , e32 , location
def gsm_location() :
       (mcc, mnc, lac , cellid) = location.gsm_location()
       print u"MCC: " + unicode(mcc)
       print u"MNC: " + unicode(mnc)
       print u"LAC: " + unicode( lac )
       print u"Cell_id : " + unicode(cellid)
       print " "

def quit() :
    app_lock.signal()
    appuifw.app.set_exit()

app_lock = e32.Ao_lock()
appuifw.app.title = u" Location"
appuifw.app.exit_key_handler = quit
appuifw.app.menu = [(u"Obtener_Localizacion ", gsm_location) ]
app_lock.wait()
```

GPS y PyS60.

- En PyS60 existe el modulo:
  - Positioning.
  - Trabaja en equipos que tienen un GPS interno como el Nokia N95.

Ejemplo: gps.py

```
import e32, appuifw, positioning


def gps_init():
      global gps_data
      gps_data = {
      'satellites': {'horizontal_dop': 0.0,
      'used_satellites': 0, 'vertical_dop': 0.0,
      'time': 0.0,'satellites': 0, 'time_dop':0.0},
      'position': {'latitude': 0.0, 'altitude': 0.0,
      'vertical_accuracy': 0.0, 'longitude': 0.0,
      'horizontal_accuracy': 0.0},
      'course': {'speed': 0.0, 'heading': 0.0,
      'heading_accuracy': 0.0, 'speed_accuracy': 0.0}
      }
      try:
            positioning.select_module(positioning.default_module())
            positioning.set_requestors([{"type":"service","format":"application","data":"gps_app"}])
            positioning.position(course=1,satellites=1,callback=gps, interval=200000000,partial=0)
            e32.ao_sleep(3)
      except:
            appuifw.note(u'Problem with GPS','error')
```

```
def gps(event):
      global gps_data
      gps_data = event

def gps_stop():
      #this function stops GPS
      try:
            positioning.stop_position()

      except:
            appuifw.note(u'Problem with GPS','error')

def quit() :
      app_lock.signal()
      appuifw.app.set_exit()


#testing
gps_init()
count = 0
while count < 10:
      count = count + 1
      sat = gps_data['satellites']['used_satellites']
      pos_lat = gps_data['position']['latitude']
      pos_long = gps_data['position']['longitude']
      speed = gps_data['course']['speed']
      print u"Satelite: " + unicode(sat)
      print u"Latitud: " + unicode(pos_lat)
      print u"Longitud: " + unicode(pos_long)
      print " "
      e32.ao_sleep(1)

appuifw.app.menu = [(u" Parar ", gps_stop)]

appuifw.app.exit_key_handler=quit
```