

**SISTEMA AUTOMATIZADO PARA EL CONTROL DE
ASISTENCIA POR MEDIO DE RECONOCIMIENTO FACIAL**

MAURICIO RIGOBERTO MARTÍNEZ ROMERO

**UNIVERSIDAD DE EL SALVADOR
FACULTAD MULTIDISCIPLINARIA ORIENTAL
DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA**

**SISTEMA AUTOMATIZADO PARA EL CONTROL DE ASISTENCIA
POR MEDIO DE RECONOCIMIENTO FACIAL**

MAURICIO RIGOBERTO MARTÍNEZ ROMERO MR13052

**PROYECTO DE FIN DE CICLO PARA LA ASIGNATURA ALGORITMOS
GRÁFICOS**

**DOCENTE:
ING. LUDWIN HERNÁNDEZ**

UNIVERSIDAD DE EL SALVADOR
FACULTAD MULTIDISCIPLINARIA ORIENTAL
DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA

INTRODUCCIÓN

En el primer capítulo se muestra el título del proyecto a realizar, la situación problemática, se hace un planteamiento del problema dando una solución a la problemática, posteriormente se elabora una justificación, se detallan las delimitaciones para realizar este proyecto y los objetivos que queremos lograr, el objetivo principal es elaborar un sistema automatizado para el control de asistencia del personal administrativo por medio de reconocimiento facial.

El segundo capítulo consiste en la teoría de todos los elementos a utilizar para realizar el sistema, que va desde el hardware utilizado (Raspberry PI 3) hasta el software (Python, Wxpython (para el diseño gráfico), OpenCV, SQLite, Raspbian Jessie).

En el tercer capítulo se habla de las personas a cargo de este proyecto (Ingeniero a cargo de la asignatura y estudiantes), se hace un presupuesto de todo el hardware y software implementado, también se muestra un cronograma de actividades, detallando el inicio del proyecto hasta el final.

Finalmente se presentan los anexos, incluyendo un tutorial de

como realizar el sistema de reconocimiento facial y uno con el funcionamiento de dicho sistema.

1. PROBLEMA

1.1 Título descriptivo del proyecto

"Sistema automatizado para el control de asistencia por medio de reconocimiento facial."

1.2 Situación problemática

En la universidad de El Salvador Facultad Multidisciplinaria Oriental (UES-FMO), es necesario llevar un registro de la hora de entrada y salida del personal administrativo que labora en la universidad, la manera en que se lleva a cabo este proceso hace que se pierda tiempo y se da el caso que pueden alterar las horas de entrada y salida por el mismo personal.

1.3 Planteamiento del problema

El control de asistencia y permanencia de los empleados, en la

cual se llevan a cabo los siguientes controles: Descuentos a los empleados por llegadas tardes, realizar descuentos por días no laborados sin justificación alguna, realizar descuentos por tomar días de licencia que no correspondían, compensación por horas extras laboradas, etc. Todas estas decisiones se toman al tener los informes de cada empleado en el mes, y para llegar a esto se necesita realizar las funciones de control de asistencia y permanencia de los empleados, y el control de licencias. Es por esto que es necesario que toda la información este integrada y actualizada en un sistema automatizado, para tomar decisiones acertadas que no le causen perjuicio tanto la universidad como al empleado. Por esta razón el grupo de trabajo propone la solución del problema en un sistema automatizado para el control de asistencia por medio de detección de rostro.

1.4 Enunciado del problema

En la Universidad de El Salvador Facultad Multidisciplinaria Oriental (UES-FMO), en el proceso de obtención de asistencia se genera una pérdida de tiempo para el personal administrativo e incongruencias en el registro de la misma.

1.5 Justificación

Debido a la problemática que se presenta al momento de registrar la hora de entrada y salida del personal administrativo de la UES-FMO, se tiene la necesidad de automatizar este proceso implementando un sistema de reconocimiento facial.

El registro de entrada y salida es importante para llevar un mejor control de asistencia, le pone fin a los retrasos, salidas temprano y regresos tarde de la hora de comer.

Realizar este proceso manualmente hace que se pierda tiempo y que no se obtengan datos acertados ya que se puede alterar la hora en que el empleado llega o sale de su área de trabajo.

Al desarrollar este sistema se logrará llevar un mejor registro de la asistencia y ayudará a optimizar el tiempo que se emplea en obtener estos datos.

El registro se obtendrá con el reconocimiento facial a la hora que el empleado entre y salga de su área de trabajo, a esta información tendrá acceso un encargado que será el mismo que administrará el sistema.

1.6 Delimitaciones

1.6.1 Lugar o espacio

Se tiene un espacio físico reducido para la implementación,

así como la necesidad de infraestructuras eléctricas o similares.

1.6.2 Tiempo

Se cuenta con un periodo de tiempo limitado para la culminación del sistema.

1.6.3 Teorías

Para realizar el sistema se basó en ejemplos de reconocimiento facial, algunos hechos por otros estudiantes.

http://www.ucsp.edu.pe/archivos/revistadeinvestigacion/4_Proyecto_de_reconocimiento_de_Rostros.pdf

1.7 Objetivos

1.7.1 Objetivo general

- Elaborar un Sistema automatizado para el control de asistencia del personal administrativo por medio de reconocimiento facial.

1.7.2 Objetivos específicos

- Analizar y elaborar el diseño detallado de los elementos que intervendrán en el sistema y que permita la manipulación de ellos.

- Utilizar un algoritmo de de detección de rostros en la librería Open Source Computer Vision Library (OpenCV).
- Implementar el Sistema de asistencia automatizado de reconocimiento facial en UES-FMO.
- Informar al personal administrativo los beneficios que tiene el uso del sistema de asistencia automatizado con detección de rostro.

2. FUNDAMENTACIÓN TEÓRICA

2.1 Raspberry Pi 3

La Raspberry Pi 3 es una placa base con unas dimensiones tan llamativas como sus especificaciones y la larga lista de aplicaciones que este pequeño artilugio puede tener.

Mejorando en todos los aspectos a los modelos anteriores, la Raspberry Pi 3 se puede convertir en casi cualquier cosa, desde un mini pc, hasta un servidor de datos, centro de ocio o la base de un robot. Así de dinámica es esta pequeña placa que por fin incorpora conexiones inalámbricas.

El nuevo modelo de Raspberry, el Pi 3, ensambla en su circuito un chipset Broadcom BCM2387 con cuatro núcleos ARM Cortex-A53 a 1.2 GHz. Dicho procesador es capaz de mover con soltura videojuegos y aplicaciones, además de disponer de una gran potencia de procesamiento para otros tipos de tareas. La GPU encargada de los gráficos es la Broadcom VideoCore IV, una solución Dual Core compatible con Open GL ES 2.0 y OpenVG que permite llegar a resoluciones Full HD con soltura.

El 'giga' de RAM DDR2 que incorpora le permite mover con soltura bastantes sistemas operativos, como Windows 10 IoT Core y la

distribución Ubuntu Snappy, versiones Slim de los sistemas operativos a los que emulan.

Si seguimos enumerando las especificaciones de la Raspberry Pi, disponemos de 4 puertos USB para ampliar el dispositivo con todo tipo de periféricos, que a día de hoy son muchos.

Entre otras de las mejoras que encontramos en el modelo Pi 3 se cuenta la llegada de la conectividad Wi-Fi y Bluetooth integradas.

La entrada *HDMI* permite la visualización de contenidos en alta definición en un gran número de dispositivos, como pueden ser pantallas, ordenadores portátiles o televisores.

Tiene unas dimensiones de 85 x 56 x 17 mm, perfecta utilizarla como base de un kit de robótica, una estación meteorológica, consola retro, un NAS, etc.

Raspberry Pi 3 especificaciones

•Procesador:

- Chipset Broadcom BCM2387.meteorológica
- 1,2 GHz de cuatro núcleos ARM Cortex-A53

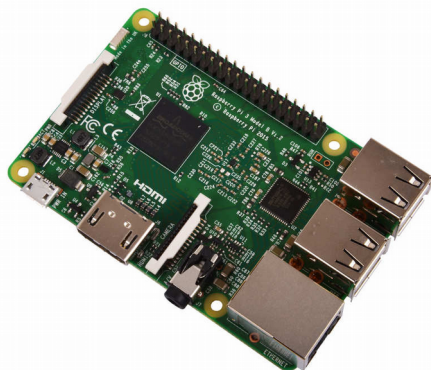
•GPU

- Dual Core VideoCore IV ® Multimedia Co-procesador. Proporciona Open GL ES 2.0, OpenVG acelerado por hardware, y 1080p30 H.264 de alto perfil de decodificación.
- Capaz de 1 Gpixel / s, 1.5Gtexel / s o 24 GFLOPs con el filtrado de texturas y la infraestructura DMA

- RAM:1GB LPDDR2.**

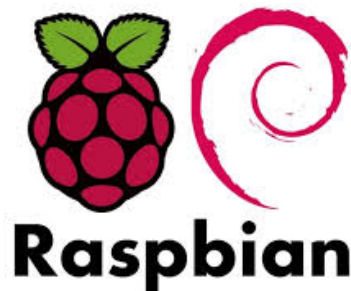
- Conectividad**

- Ethernet socket Ethernet 10/100 BaseT
- 802.11 b / g / n LAN inalámbrica y Bluetooth 4.1 (Classic Bluetooth y LE)
- Salida de vídeo
 - HDMI rev 1.3 y 1.4
 - RCA compuesto (PAL y NTSC)
- Salida de audio
 - jack de 3,5 mm de salida de audio, HDMI
 - USB 4 x Conector USB 2.0
- Conector GPIO
 - 40-clavijas de 2,54 mm (100 milésimas de pulgada) de expansión: 2x20 tira
 - Proporcionar 27 pines GPIO, así como 3,3 V, +5 V y GND líneas de suministro
- Conector de la cámara de 15 pines cámara MIPI interfaz en serie (CSI-2)
- Pantalla de visualización Conector de la interfaz de serie (DSI) Conector de 15 vías plana flex cable con dos carriles de datos y un carril de reloj
- Ranura de tarjeta de



memoria Empuje / tire Micro SDIO

2.2 Raspbian Jessie



El sistema operativo Raspbian está basado en Debian Linux, y las diferentes versiones de Debian. Las versiones recientes de Raspbian se han basado en Debian Wheezy, pero Raspbian ahora se ha actualizado a la nueva versión estable de Debian, que se llama Jessie.

Hay modificaciones en el sistema subyacente de mejorar el rendimiento y la flexibilidad, especialmente en lo que se refiere al control de los procesos del sistema, y como con cualquier actualización, hay numerosas correcciones de errores y ajustes. Y, al mismo tiempo que la actualización a Jessie, se ha añadido un montón de cambios y mejoras en la interfaz de usuario de

escritorio.

2.3 SQLite



SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña (~275 kiB)² biblioteca escrita en C. SQLite es un proyecto de dominio público creado por D. Richard Hipp.

A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

La biblioteca puede ser usada desde programas en C/C++, aunque están disponibles enlaces para Tcl y muchos otros

lenguajes de programación interpretado.

Python incluye soporte para SQLite nativamente desde la versión 2.5 incorporado en la Biblioteca Estándar como el módulo sqlite3.

2.4 Python



Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y

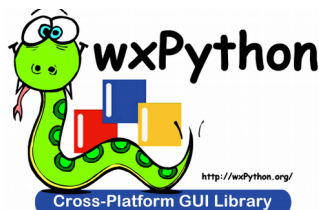
programación funcional. Otros paradigmas están soportados mediante el uso de extensiones.

Python usa tipado dinámico y conteo de referencias para la administración de memoria.

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

2.5 WxPython



WxPython es un binding de la biblioteca gráfica wxWidgets para el lenguaje de programación Python. La biblioteca wxWidgets se caracteriza por ser multiplataforma, por lo que su uso junto a Python permite el desarrollo rápido de aplicaciones gráficas multiplataforma.

2.6 WxFormBuilder



WxFormBuilder es una aplicación de diseño GUI de código abierto para wxWidgets kit de herramientas, lo que permite la creación de aplicaciones multiplataforma. Una interfaz optimizada, fácil de usar permite un desarrollo más rápido y más fácil mantenimiento de software. Está escrito en C ++.

2.7 OpenCV



OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Se utiliza en sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

Open CV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estérea y visión robótica.

El proyecto pretende proporcionar un entorno de desarrollo fácil

de utilizar y altamente eficiente. Esto se ha logrado realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multinúcleo. OpenCV puede además utilizar el sistema de primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel específicas para procesadores Intel.

Posee una estructura modular, estando agrupadas la mayoría de funciones de la biblioteca en los siguientes módulos:

- CORE: donde se definen las estructuras de datos básicas que utilizan el resto de módulos.
- IMGPROC: módulo de procesamiento de imágenes, donde encontraremos funciones para el filtrado lineal/no lineal, transformaciones afines, conversión del espacio de color, histogramas....
- VIDEO: modulo de análisis de video que incluye algoritmos para la estimación del movimiento, extracción del fondo y seguimiento de objetos.
- CALIB3D: Algoritmos básicos de visión multiple como calibración de camaras estereo, correspondencia o reconstrucción 3D.
- FEATURES2D: detectores de características, descriptores y comparadores.
- OBJDETECT: detección de objetos e instancias de las clases

predefinidas, como por ejemplo: caras, ojos, gente, coches....

- HighGUI: todo lo relacionado a la interfaz grafica de OpenCV y las funciones que permiten importar imágenes y video.
- GPU: algoritmos acelerados por hardware para distintos modulos OpenCV.

2.8 Python y OpenCV

OpenCV-Python es una biblioteca de enlaces Python diseñados para resolver los problemas de visión por ordenador.

Python es un lenguaje de programación de propósito general iniciado por Guido van Rossum que se hizo muy popular muy rápidamente, principalmente debido a su simplicidad y legibilidad del código. Permite al programador para expresar ideas en menos líneas de código sin reducir la legibilidad.

En comparación con lenguajes como C / C ++, Python es más lento. Dicho esto, Python puede ampliarse fácilmente con C /C++, lo que nos permite escribir código computacionalmente intensivas en C /C++ y Python crear contenedores que pueden ser utilizados como módulos de Python. Esto nos da dos ventajas: primero, el código es tan rápido como el código original C / C++ (ya que es el código real C++ de trabajo en el fondo) y segundo, que es más fácil de código en Python que C /C++. OpenCV-Python es un envoltorio de Python para la implementación del original OpenCV C++.

OpenCV-Python hace uso de Numpy, que es una biblioteca altamente optimizada para operaciones numéricas con una sintaxis de estilo MATLAB. Todas las estructuras de matriz OpenCV se convierten a partir de matrices y NumPy. Esto también hace que sea más fácil de integrar con otras bibliotecas que utilizan Numpy como SciPy y matplotlib.

¿Cómo funciona el reconocimiento facial de OpenCV?

Para legos: OpenCV utiliza un conjunto de “bloques” para reconocer formas llamados Haar-like features. Por ejemplo, algunas de estas formas pueden ser:



a)

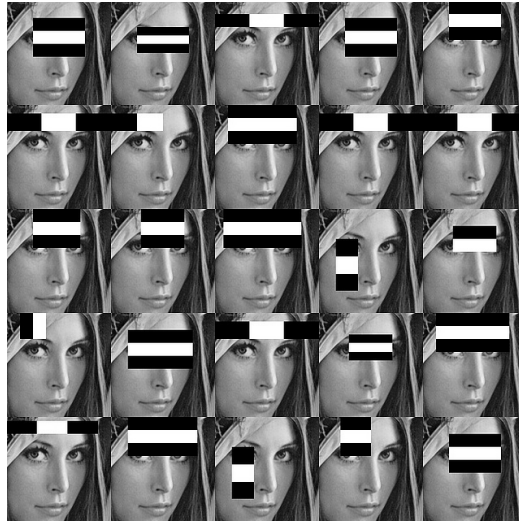


b)



c)

El algoritmo busca en la imagen combinaciones de estos patrones. Por ejemplo, si queremos detectar un rostro, como es en nuestro caso, el algoritmo buscará en la imagen la combinación de estos bloques que, si se juntan, se aproximan a un rostro:



Este sistema tiene un porcentaje de aciertos bastante alto, aunque su éxito también dependerá del tipo de cámara utilizada, la iluminación de la sala, etc.

El código es una variante del código oficial para detectar rostros en imágenes estáticas.

#Ejemplo de deteccion facial con OpenCV y Python

```
import numpy as np
import cv2
```

```
#cargamos la plantilla e inicializamos la webcam:
```

```
face_cascade =
```

```
cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')
```

```
cap = cv2.VideoCapture(0)
```

```
while(True):
```

```
    #leemos un frame y lo guardamos
```

```
    ret, img = cap.read()
```

```
    #convertimos la imagen a blanco y negro
```

```
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
    #buscamos las coordenadas de los rostros (si los hay) y
```

```
#guardamos su posicion
faces = face_cascade.detectMultiScale(gray, 1.3, 5)

#Dibujamos un rectangulo en las coordenadas de cada rostro
for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(125,255,0),2)

#Mostramos la imagen
cv2.imshow('img',img)

#con la tecla 'q' salimos del programa
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
cap.release()
cv2.destroyAllWindows()
```

Al ejecutar el código, se debería ver nuestro rostro dentro de un recuadro de color verde.

3. ASPECTOS ADMINISTRATIVOS





3.1 Recurso Humano

Son 2 entes los que participan en la realización de este proyecto, ingeniero a cargo de la asignatura “Algoritmos Gráficos” y estudiantes que cursan dicha asignatura, los aspectos administrativos son dirigidos por recursos humanos dividiendo las funciones entre: ingeniero que se encarga de orientar y brindar información y los estudiantes que son los responsables de programar, documentar, diseñar, testear el sistema, gracias a ello es posible la culminación del presente proyecto.

3.2 Presupuesto

HARDWARE		
Producto	Descripción	Precio

	<p>Raspberry PI 3</p>	<p>\$66.00</p>
	<p>Fuente de alimentación de 5V 2.5A</p>	<p>\$9.99</p>
	<p>Cable HDMI</p>	<p>\$3.00</p>
	<p>Micro usb de 32 gb</p>	<p>\$17.00</p>

	<p>Pantalla LCD de 19 pulgadas</p>	<p>\$90.00</p>
	<p>Teclado inalámbrico</p>	<p>\$27.99</p>
	<p>Ratón inalámbrico</p>	<p>\$9.99</p>
	<p>WebCam</p>	<p>\$30.00</p>

SOFTWARE	
Licencia python	\$0.00
Raspbian Jessie	\$0.00
OpenCV	\$0.00

Presupuesto total	
HARDWARE	\$253.97
SOFTWARE	\$0.00
TOTAL	\$253.97

3.3 Cronograma

[illegible]

4. Referencias

1. http://www.pccomponentes.com/raspberry_pi_3_modelo_b.html
2. <https://www.raspberrypi.org/blog/raspbian-jessie-is-here/>
3. <https://es.wikipedia.org/wiki/Python>
4. <https://es.wikipedia.org/wiki/OpenCV>
5. <https://geekytheory.com/opencv-en-linux/>
6. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_setup/py_intro/py_intro.html
7. <https://es.wikipedia.org/wiki/SQLite>
8. <http://robologs.net/2014/05/26/reconocimiento-facial-con-opencv-webcam/>
9. <http://www.solvetic.com/tutoriales/article/2307-desarrollo-de-aplicaciones-con-python-y-wxformbuilder/>
10. <http://www.mclarenx.com/2015/02/03/raspberry-pi-paso-1-instalar-raspbian/>
11. <http://www.mclarenx.com/2015/02/04/raspberry-pi-paso-2->

[primer-arranque-raspi-config/](#)

12. http://turing.iimas.unam.mx/~ivanvladimir/es/content/teach/curso_aprendizaje_automatico_s4.html

ANEXOS

Tutorial para la implementación del sistema de reconocimiento facial

Instalación de Raspbian

1. Descargar la imagen del sistema operativo

Entrar en la web de descargas oficial de Raspberry Pi <https://www.raspberrypi.org/downloads/> buscamos Raspbian y le damos a “Download ZIP”.

Una vez descargado el ZIP, hay que descomprimirlo para obtener el fichero .img que hay dentro.

2. Instalar Raspbian en la tarjeta microSD

La tarjeta debe de ser al menos de 8GB o más. Instalando raspbian con el sistema operativo GNU/Linux.

Antes de meter la tarjeta en el PC, que no hace falta que esté formateada porque este proceso lo hará, vamos a listar los dispositivos que tenemos conectados:

`df -h`

Después introducimos la tarjeta en el ordenador, bien mediante un lector de tarjetas o un adaptador USB y volvemos a hacer:

```
df -h
```

Habr  una l nea nueva que empezar  con la ruta del nuevo dispositivo, deber  ser algo as :

```
/dev/sdc1
```

Variar  en funci n de cada sistema, pero casi seguro que empezar  por /dev/sd seguido por una letra y un n mero, en el ejemplo ser  c1. Si por casualidad has introducido una tarjeta SD particionada, aparecer n varias l neas nuevas pero todas con la misma ra z, algo tipo:

```
/dev/sdc1
```

```
/dev/sdc2
```

Ahora vamos a desmontar la unidad para poder modificarla:

```
umount /dev/sdc1
```

Si hay varias particiones de tu sdc, hay que desmontarlas todas.

Por fin vamos a copiar la imagen del sistema operativo en nuestra tarjeta microSD. El comando es el siguiente:

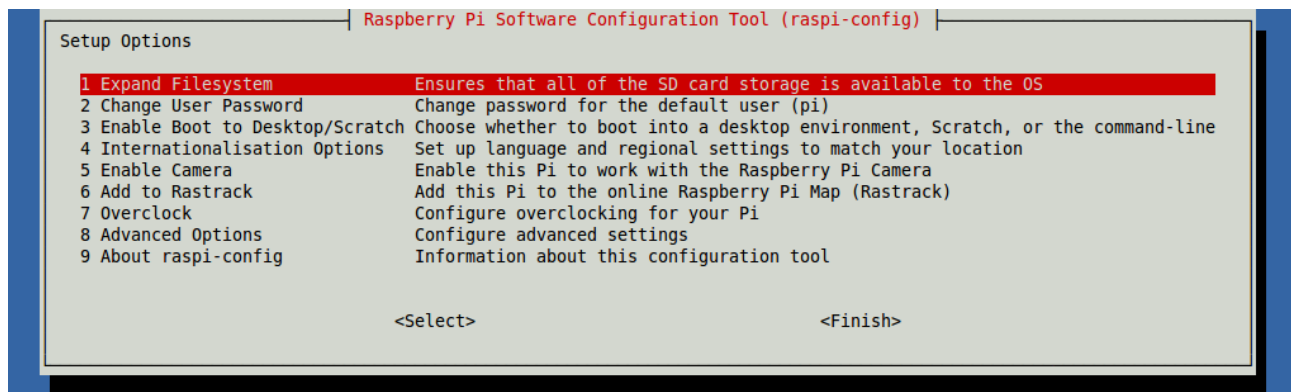
```
dd if=/ruta/descarga/zip/2015-01-31-wheezy-raspbian.img o  
f=/dev/sdc bs=1M
```

Donde if es la ruta donde est  el .img que descomprimimos del ZIP, of es el nombre del dispositivo que desmontamos antes (sin n mero de partici n) y bs es el tama o de los bloques a copiar. Este punto es importante. Normalmente con un tama o de bloque

de 4M sería suficiente, pero se recomienda que se copie en bloques de 1M para mayor seguridad. Eso sí, tarda muchísimo más, y lo peor de todo es que el comando `dd` no muestra absolutamente ninguna información del progreso de la copia, así que nos toca esperar a que termine sin ningún indicativo de nada.

Introducir la tarjeta en la RasPi3, conéctala a una pantalla LCD con un cable HDMI y a la red eléctrica con un cargador de móvil microUSB de 2A. Para este paso también es fundamental enchufar un teclado [USB](#).

Una vez todo conectado, cambia el input de la TV al puerto HDMI dónde hayas conectado la Raspberry Pi y verás que aparecerá una pantalla como esta:



Es exactamente la misma pantalla que aparecerá en el futuro si alguna vez queremos reconfigurar algo y ejecutamos el comando:

`sudo raspi-config`

La manera de manejarse por esta pantalla es bastante intuitiva y sólo nos hacen falta las flechas (moverse entre opciones), el tabulador (ir a los botones de abajo), el intro (acceder) y la barra espaciadora (marcar opción). Hay pocas cosas importantes, pero vamos a repasarlas todas y voy comentando las que son fundamentales.

1. Expand Filesystem

Sirve para poder ocupar todo el espacio de la microSD. Con esta opción expandimos el sistema de ficheros para poder utilizar toda la capacidad de la tarjeta, y así instalar todo tipo de programas que necesitemos sin problemas.

2 Change User Password

El usuario por defecto es pi, con la contraseña raspberry. Si queremos cambiarla es aquí donde se hace.

3 Enable Boot to Desktop/Scratch

No hace falta tocar nada. Por defecto la Raspberry Pi arranca en la típica consola negra con letras blancas, es recomendable dejarlo así, pero tienes otras 2 opciones más. Escritorio: inicia un escritorio de toda la vida con su menú de programas y demás (útil si vamos a usar la RPi como un ordenador normal). Scratch: es un entorno de programación que facilita el aprendizaje autónomo, una herramienta muy útil para crear juegos sencillos y aprender a programar sin conocimientos previos.

4 Internationalisation Options

El sistema viene por defecto en inglés y con disposición americana de teclas.

4.1 Change Locale

Cambiar idioma.

4.2 Change Timezone

Cambiar zona horaria.

4.3 Change Keyboard Layout

Elige el que se adapte al teclado que vayas a utilizar, pero en general seleccionando uno genérico en nuestro idioma debería ser suficiente. Además pregunta varias cosas de teclas especiales, si no sabes por donde van los tiros, déjalo por defecto.

5 Enable Camera

Por si no lo sabías, se puede conectar una cámara a la Raspberry Pi gracias al GPIO, como supongo que no es el caso, no tocar nada aquí.

6 Add to Rastrack

Esto agrega la posición GPS de nuestra Raspberry Pi a un mapa mundial que se puede consultar aquí: rastrack.co.uk. No sirve absolutamente para nada, simple curiosidad.

7 Overclock

Aumentar la velocidad de procesamiento.

8 Advanced Options

8.1 Overscan

Si vas a conectar la RPi a un monitor o un televisor antiguo puede que veas unos bordes negros que no deberían estar, con esta opción se quitan.

8.2 Hostname

Es el nombre que tendrá tu Raspberry Pi a ojos del resto de la red.

8.3 Memory Split

Es la cantidad de memoria que le damos a la GPU (a los gráficos). Es una memoria compartida entre el sistema y los gráficos.

8.4 SSH

Es el protocolo que vamos a utilizar para conectarse a la Raspberry Pi 3 desde cualquier otro ordenador o desde nuestro smartphone para poder modificar, actualizar o instalar lo que queramos en ella sin necesidad ni de estar físicamente delante ni de tener conectado ningún teclado por USB a la misma.

8.5 SPI

Esto es un módulo que sirve para utilizar periféricos que no vamos a usar, no hace falta activarlo.

8.6 Audio

Por defecto viene configurado por HDMI, que es lo que queremos generalmente.

Actualiza raspi-config, la pantalla donde estamos, para tener las últimas opciones de configuración.

9 About raspi-config

Explica lo que es el raspi-config, típico “about”, pura información sin importancia.

Con el tabulador nos posicionamos sobre los botones de abajo y con los cursores nos movemos hasta “Finish”, al pulsar intro nos preguntará si queremos reiniciar, diremos que sí para que coja los cambios. Cuando arranque de nuevo, ya no saldrá la ventana de configuración sino que nos aparece la consola pidiendo usuario y contraseña, obviamente los ponemos.

Instalación de SQLite3 para la Base de Datos

1. sudo apt-get install sqlite3

Configuración para la interfaz gráfica

1. Abrir el navegador Iceweasel >> Herramientas >> SQLite Manager

2. Crear una base de datos con el nombre “db_registro”

Base de Datos >> Nueva Base de Datos

3. Crear las tablas con sus respectivas columnas

- Tabla “login”

Columnas (3)

Column ID	Name	Type	Not Null	Default Value	Primary Key	
0	Id	INTEGER	1	null	1	
1	usuario	VARCHAR(20)	0	null	0	
2	contrasenia	VARCHAR(20)	0	null	0	

Nombre Tipo No Null Por Defecto

 ▼ ☐

- Tabla “usuarios”

Columnas (9)

Column ID	Name	Type	Not Null	Default Value	Primary Key	
0	id_usuario	INTEGER	1	null	1	
1	nombre	VARCHAR(100)	0	null	0	
2	apellido	VARCHAR(100)	0	null	0	
3	direccion	VARCHAR(200)	0	null	0	
4	dui	VARCHAR(12)	0	null	0	
5	nit	VARCHAR(24)	0	null	0	
6	cargo	VARCHAR(50)	0	null	0	
7	imagen	VARCHAR(50)	0	null	0	
8	telefono	VARCHAR(16)	0	null	0	

Nombre Tipo No Null Por Defecto

 ▼ ☐

- sqlite_sequence

Columnas (2)

Column ID	Name	Type	Not Null	Default Value	Primary Key	
0	name		0	null	0	
1	seq		0	null	0	

Instalación de python, wxpython, wxformbuilder.

Desde una ventana de terminal escribimos los siguientes comandos:

1. sudo apt-get update
2. sudo apt-get python2.7
3. sudo add-apt-repository -y ppa:wxformbuilder/wxwidgets
4. sudo apt-get update

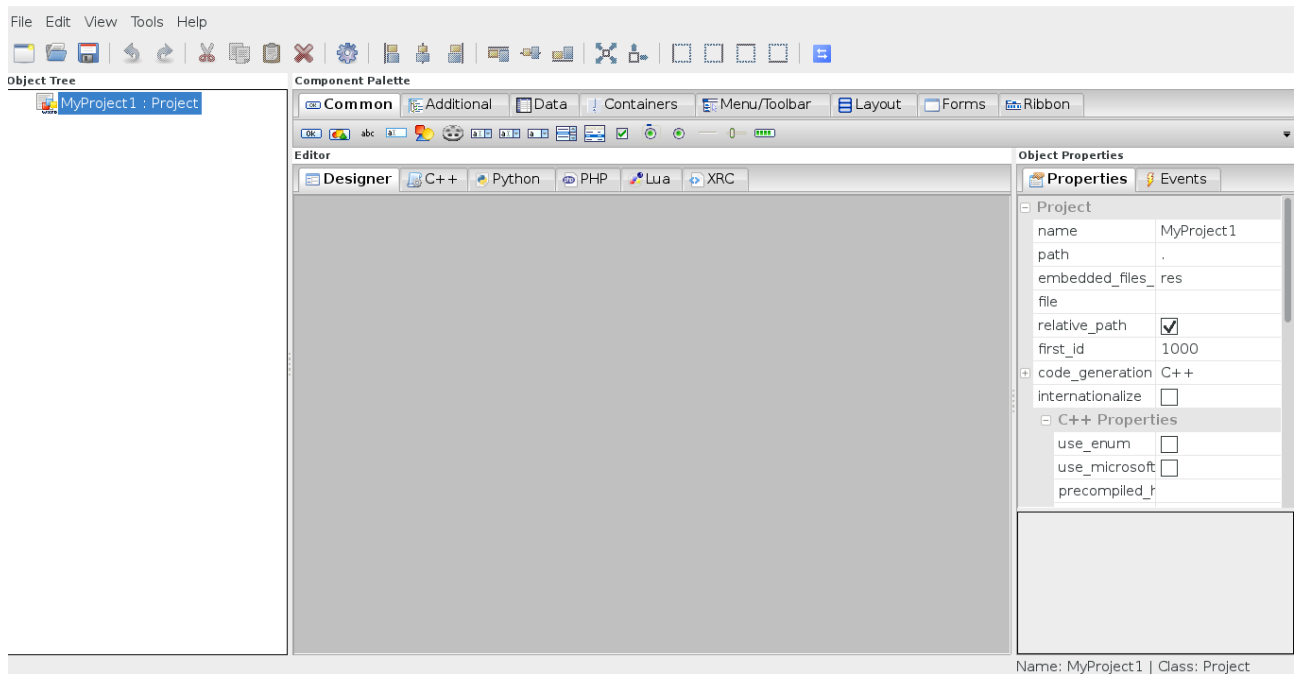
5. `sudo apt-get install libwxgtk3.0-0 libwxgtk-media3.0-0`
6. `sudo add-apt-repository -y ppa:wxformbuilder/release`
7. `sudo apt-get update`
8. `sudo apt-get install wxformbuilder`

Luego debemos añadir wxpython como variable de entorno para poder utilizarlo desde cualquier directorio:

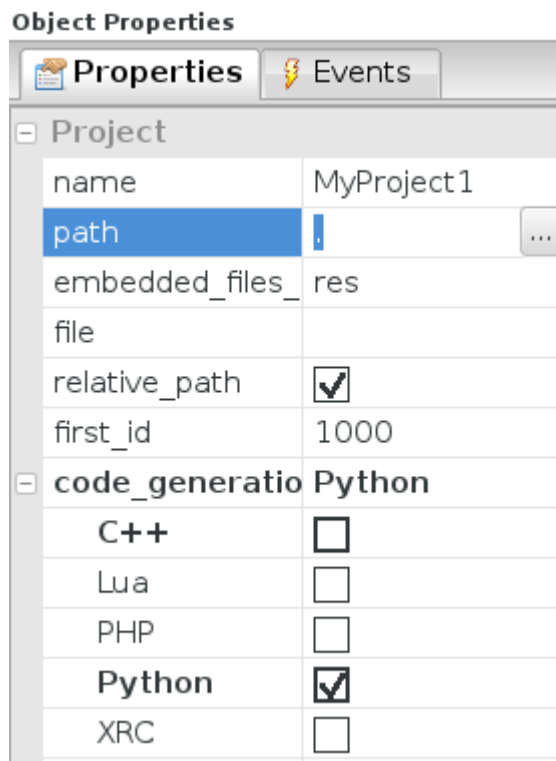
```
export PYTHONPATH="$PYTHONPATH":/usr/lib/python2.7/dist-packages/wx-2.8-gtk2-unicode/
```

A continuación abrimos wxFormbuilder desde el menú principal:

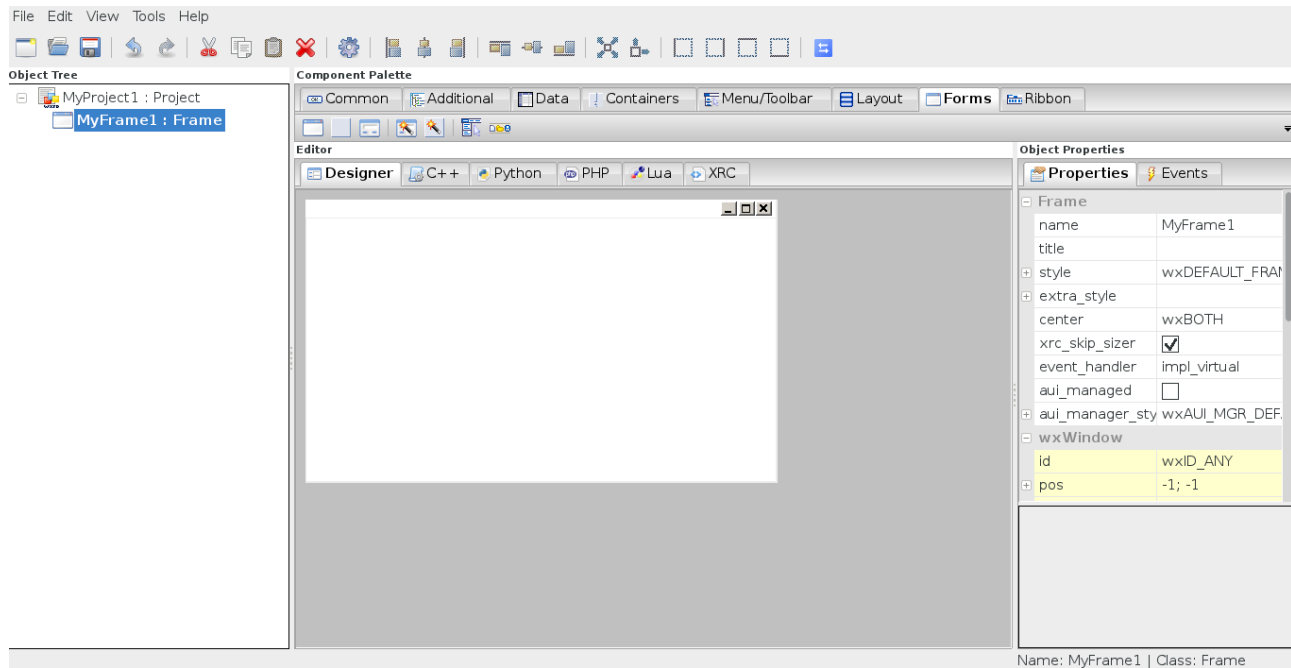
WxFormBuilder presenta un escritorio con una columna izquierda donde se ubicaran el proyecto o pantalla y componentes que vayamos utilizando, en el centro la pestaña de diseño y una pestaña por cada lenguaje, a la derecha tendremos las propiedades tanto del proyecto como de los componentes que utilizemos.



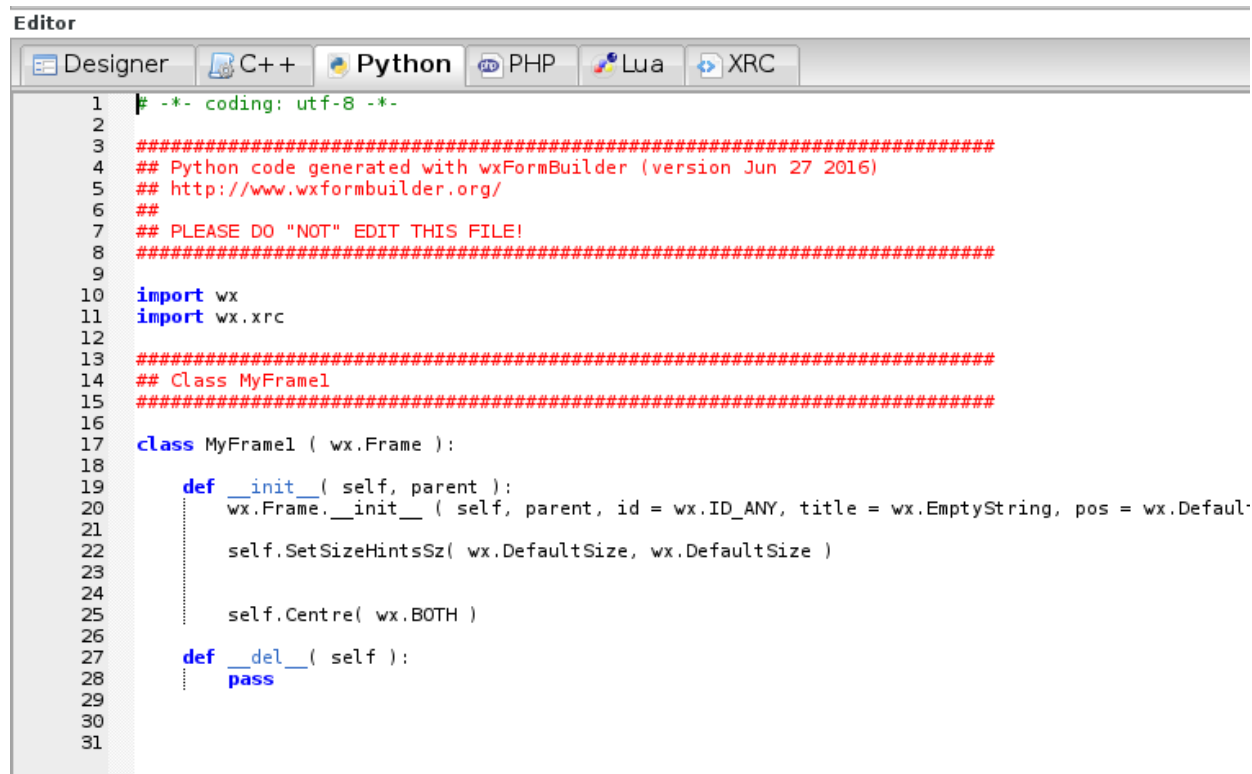
Lo primero que deberemos configurar sera el proyecto, para ello hacemos un clic en el nombre del proyecto y luego vamos a las propiedades donde asignaremos un nombre y el lenguaje que utilizaremos.



A continuación vamos a la pestaña Form y añadimos un formulario que sera el contenedor de la pantalla.



Luego desde las propiedades podremos modificar el nombre del formulario y el titulo, además podremos modificar muchas opciones como el color de fondo, el tamaño, el tipo de ventana y muchos mas. Podemos ir a la pestaña Python y ver como se va generando el código.



The image shows a code editor window titled "Editor". The toolbar includes buttons for "Designer", "C++", "Python", "PHP", "Lua", and "XRC". The Python tab is active, displaying the following code:

```
1  # -*- coding: utf-8 -*-
2
3  #####
4  ## Python code generated with wxFormBuilder (version Jun 27 2016)
5  ## http://www.wxformbuilder.org/
6  ##
7  ## PLEASE DO "NOT" EDIT THIS FILE!
8  #####
9
10 import wx
11 import wx.xrc
12
13 #####
14 ## Class MyFrame1
15 #####
16
17 class MyFrame1 ( wx.Frame ) :
18
19     def __init__( self, parent ) :
20         wx.Frame.__init__( self, parent, id = wx.ID_ANY, title = wx.EmptyString, pos = wx.DefaultPosition, size = wx.DefaultSize, style = wx.DEFAULT_FRAME_STYLE )
21
22         self.SetSizeHintsSz( wx.DefaultSize, wx.DefaultSize )
23
24         self.Centre( wx.BOTH )
25
26     def __del__( self ) :
27         pass
28
29
30
31
```

Para generar el código en un archivo, primero deberemos guardar el proyecto desde el menú File > Save As, y lo guardamos como ejemplo.fbp

A continuación vamos a la opción del menú File > Generate Code, luego vamos al directorio donde guardamos el archivo del proyecto y veremos el archivo noname.py

Este archivo noname.py contiene el código Python generado con el diseño de la interfaz, podemos renombrar el archivo a ejemplo.py

A continuación debemos añadir el código para que se muestre este diseño cuando se ejecuta la aplicación. Para ello abrimos el archivo y añadimos el siguiente código debajo quedándonos de la siguiente manera:

```

import wx

1.import wx.xrc
2.
3.class MiForm ( wx.Frame ):
4.
5.def __init__( self, parent ):
6.wx.Frame.__init__( self, parent, id = wx.ID_ANY, title =
u"Ejemplo01 - Tutorial ", pos = wx.DefaultPosition, size =
wx.Size( 500,300 ), style = wx.DEFAULT_FRAME_STYLE|
wx.TAB_TRAVERSAL )
7.self.SetSizeHintsSz( wx.DefaultSize, wx.DefaultSize )
8.
9.self.Centre( wx.BOTH )
10.
11.def __del__( self ):
12.Pass
13.
14.#Fin código del diseño de Formulario
15.
16.## Código que muestra la aplicación al ejecutarse
17.app = wx.App(False)
18.frame = MiForm (None)
19.frame.Show(True)
20.app.MainLoop()

```

Luego desde una ventana de terminal vamos al directorio de la aplicación y ejecutamos “python ejemplo.py”

A continuación vamos a wxFormbuilder y comenzaremos a diseñar la pantalla. Los componentes se distribuyen en la pantalla mediante Layout y grillas.

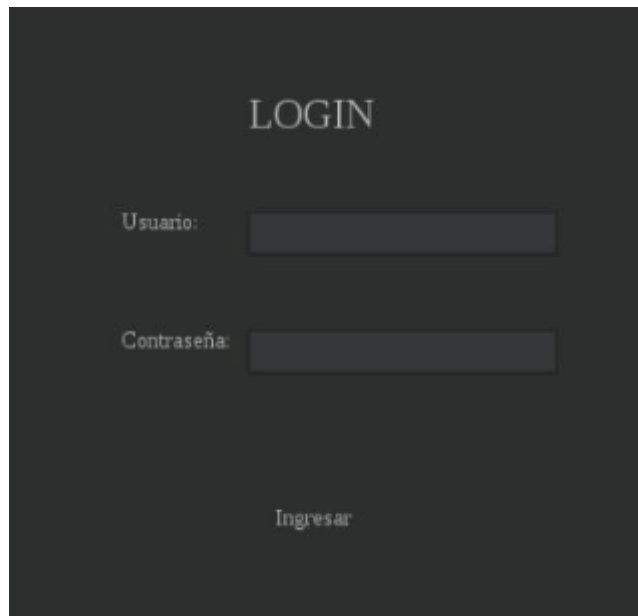
Para más información de como diseñar formularios puede acceder a este enlace <http://www.solvetic.com/tutoriales/article/2307-desarrollo-de-aplicaciones-con-python-y-wxformbuilder/>

Instalación de OpenCV (Para poder tomar fotografías).

1. sudo apt-get install libopencv-*
2. sudo apt-get install python-opencv
3. sudo apt-get install python-numpy

Para el sistema se necesitan los siguientes formularios:

- Login



LOGIN

Usuario:

Contraseña:

Ingresar

Código del formulario login

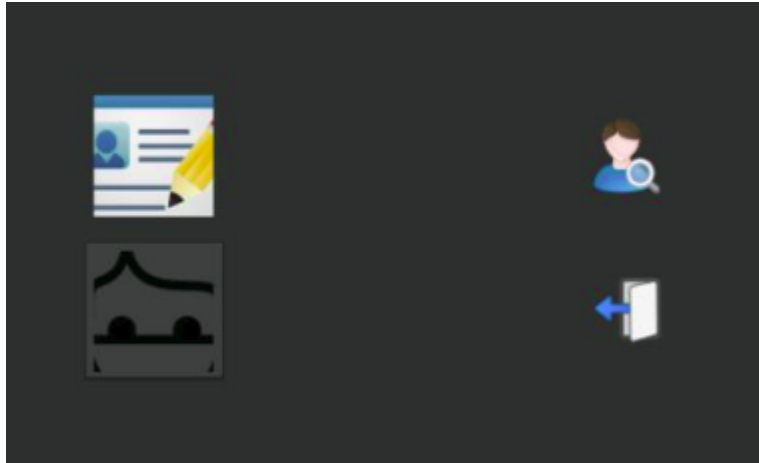
```
# dentro del evento __init__
#conectamos a la base de datos
self.db =
sqliteclass.Database("db_registro.sqlite")
#evento ejecutado al cerrar el formulario
def __del__( self ):
    pass
# evento para validar la cuenta del usuario que inicia
sesión
def ingresar(self, event):
```

```

        #conectamos a la base de datos y solicitamos (si
existen), los datos del usuario
        sql="""SELECT contrasenia FROM login WHERE
login.usuario=:nombre"""
        data_param={'nombre':self.txtnom.GetValue()}
        typesql='S'
        rows=self.db.query(sql,data_param,typesql)
        # si existe el usuario hacemos la validacion de la
contraseña
        if len(str(rows))>2:
            for row in rows:
                # si el usuario existe y la contraseña es
correcta
                if str(row[0]) == self.txtpass.GetValue():
                    self.frm_menu = menu.Menu(self)
                    self.frm_menu.Show(True)
                else:
                    #si la contraseña ingresada es
incorrecta damos un mensaje
                    self.lblerror.SetLabel("Contraseña
incorrecta")
            else:
                # si el usuario no existe damos un mensaje
                self.lblerror.SetLabel("El usuario no existe")

```

- Menú



Código del
formulario menu

```
# funcion para ajustar las imagenes
def redimension(self, ruta):
    img = wx.Image(ruta, wx.BITMAP_TYPE_ANY)
    W = img.GetWidth()
    H = img.GetHeight()

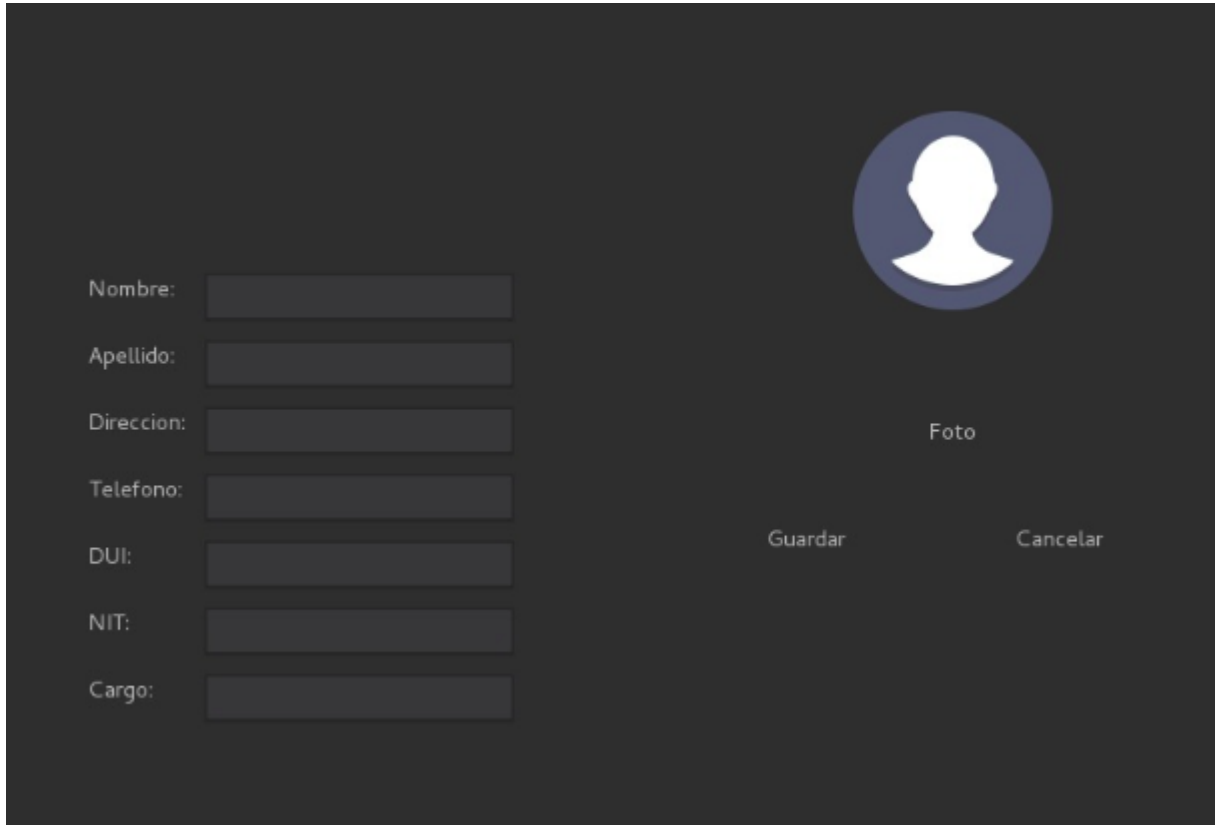
    if W > H:
        NewW = self.anch
        NewH = self.anch * H / W
    else:
        NewH = self.anch
        NewW = self.anch * W / H
    img = img.Scale(NewW, NewH)
    return img

# cerrar el formulario
def salir(self, event):
    self.Close()

# abrir el formulario de busqueda, edicion y
eliminacion de registros
def busqueda(self, event):
    self.frm_busqueda = busqueda.Busqueda(self)
    self.frm_busqueda.Show(True)

# abrir el formulario de registro
def registro(self, event):
    self.frm_registro = usuarios.Registro(self)
    self.frm_registro.Show(True)
```

- Registro de Usuarios



Nombre:

Apellido:


Dirección:

Teléfono:

DUI:

NIT:

Cargo:



Foto

Guardar Cancelar

```
# este bloque de codigo va dentro de def __init__  
#=====   
# Añadiendo imagen a Bitmap  
#=====   
  
self.ruta="orig_frame.png"  
  
if os.path.exists(self.ruta):  
    os.remove(self.ruta)  
    self.ruta="imgs/blank_img.png"  
    print "Habia foto, pero se a eliminado"  
  
else:  
    self.ruta="imgs/blank_img.png"
```

```

        print "NO existia imagen de usuario en el
sistema"

        img = wx.Image(self.ruta, wx.BITMAP_TYPE_ANY)

        self.ancho = 150
        self.alto = 200

        #=====

        # dentro de def __init__
        # agregada la conexion a la base de datos
        self.db =
sqliteclass.Database("db_registro.sqlite")
    def __del__( self ):
        self.padre.m_button2.Enable()
        # evento para cerrar el formulario
    def Cancelar( self, event ):
        self.padre.m_button1.Enable()
        self.Close()

        # evento para tomar una foto del nuevo usuario
--posterior
    def Foto(self,event):
        pass

        # guardar los datos de la persona leidos de los campos
de texto en la base de datos
    def Guardar(self,event):
        print "guardar"
        #obtenemos los datos de los campos de texto
        nombre = self.txt_nombre.GetValue()
        apellido = self.txt_apellido.GetValue()
        direccion = self.txt_dir.GetValue()
        telefono = self.txt_tel.GetValue()
        dui = self.txt_oui.GetValue()
        nit = self.txt_nit.GetValue()
        cargo = self.txt_cargo.GetValue()
        imagen = "blank_img.png"
        # insertamos los datos en la base de datos
        query = """INSERT INTO usuarios (nombre, apellido,
direccion, dui, nit, cargo, telefono, imagen) values
(:nombre, :apellido, :direccion, :dui, :nit, :cargo,
:telefono, :imagen);"""

```

```

        param = {'nombre':nombre, 'apellido':apellido,
'direccion':direccion, 'telefono':telefono, 'dui':dui,
'nit':nit, 'cargo':cargo, 'imagen':imagen}
        typesql = 'I'
        self.db.query(query, param, typesql)
        self.padre.m_button1.Enable()
        self.Close()

```

- Búsqueda

Busqueda:

Id	Nombre	Apellido	Dirección	Teléfono	DUI	NIT	Cargo

Modificar Eliminar

```

#codigo dentro de def __init__

#=====
# Añadiendo imagen a Bitmap

#=====

```

```

self.ruta="orig_frame.png"

if os.path.exists(self.ruta):
    os.remove(self.ruta)
    self.ruta="imgs/blank_img.png"
    print "Habia foto, pero se a eliminado"

else:
    self.ruta="imgs/blank_img.png"
    print "NO existia imagen de usuario en el
sistema"

img = wx.Image(self.ruta, wx.BITMAP_TYPE_ANY)

self.ancho = 150
self.alto = 200

#=====
#Codigo dentro de def __init__
self.padre = parent
self.padre.m_button1.Disable()
#Conexion a la base de datos
self.db =
sqliteclass.Database("db_registro.sqlite") #Instanciar la
conexion a la Bd.
#establecemos el tamaño maximo de la imagen
self.PhotoMaxSize = 150
#cargamos los datos a la tabla por defecto
self.cargar()

def __del__( self ):
    self.padre.m_button1.Enable()
#cargamos los datos en la lista
def cargar(self):
    #evento para cargar datos de la bd a la lista de 2
maneras todos si el ctrl texto esta vacio
    #o dependiendo de la busqueda con like asi muestra
los resultados
    self.tab_busqueda.DeleteAllItems() # quita los
renglones de la lista
    cadena_buscar=self.txtbusqueda.GetValue()
    if cadena_buscar!="":
        self.prod="%" +str(cadena_buscar)+"%"

```

```

        sql="SELECT * FROM usuarios WHERE nombre
LIKE ?"
        data_param=self.prod
        typesql='SL'

self.rows=self.db.query(sql,data_param,typesql)
    else:
        sql="SELECT * FROM usuarios"
        data_param=''
        typesql='S'

self.rows=self.db.query(sql,data_param,typesql)
self.row_count = 0
#al tener el cursor se van insertando las columnas
for row in self.rows:

    self.tab_busqueda.InsertStringItem(self.row_count,
str(row[0]))

    self.tab_busqueda.SetStringItem(self.row_count,1,
str(row[1]))

    self.tab_busqueda.SetStringItem(self.row_count,2,
str(row[2]))

    self.tab_busqueda.SetStringItem(self.row_count,3,
str(row[3]))

    self.tab_busqueda.SetStringItem(self.row_count,4,
str(row[8]))

    self.tab_busqueda.SetStringItem(self.row_count,5,
str(row[4]))

    self.tab_busqueda.SetStringItem(self.row_count,6,
str(row[5]))

    self.tab_busqueda.SetStringItem(self.row_count,7,
str(row[6]))
        if self.row_count % 2:

self.tab_busqueda.SetItemBackgroundColour(self.row_count,
"#EBEBEB")

```



```

        else:

self.tab_busqueda.SetItemBackgroundColour(self.row_count,
"#C4C4C4")

        self.row_count += 1
        # evento al modificarse el texto de la caja de busqueda
def Busqueda( self, event ):
    self.cargar()
    # evento para el entrenamiento --posterior
def Entrenamiento(self, event):
    pass
    # evento para salir del formulario
def Cancelar( self, event ):
    self.padre.m_button1.Enable()
    self.Close()
    # evento al seleccionar un elemento del listado de
registros cargados
def Seleccionar(self, event):
    self.seleccionado = ''
    self.id_seleccionado = ''
    self.seleccionado =
self.tab_busqueda.GetFocusedItem() #traer la posicion del
indice
    self.id_seleccionado =
self.tab_busqueda.GetItemText(self.seleccionado)#traer el
texto del primera columna segun la posicion del indice
    self.BuscarImagen(self.id_seleccionado)
    #continuamos con la busqueda y carga de información al
seleccionar un elemento de la lista
def BuscarImagen( self, id_seleccionado):
    self.ruta = 'imgs/blank_img.png'
    self.VerImagen(self.ruta)
    cadena_buscar =
{'id_seleccionado':str(id_seleccionado)}
    if cadena_buscar!="":
        sql="""SELECT * FROM usuarios WHERE
usuarios.id_usuario=:id_seleccionado"""
        data_param=cadena_buscar
        typesql='S'
        self.rows=0

self.rows=self.db.query(sql,data_param,typesql)
        if self.rows>0:

```

```

        for row in self.rows:
            if row[7] is None: #row[8] es la
posicion del cursor del campo imagen
                self.ruta='imgs/blank_img.png'
                #self.Limpiar()
            else:
                self.ruta = 'imgs/'+row[7]
                self.txtid.SetValue(str(row[0]))
                self.txtnom.SetValue(str(row[1]))

                self.txtape.SetValue(str(row[2]))

self.txtdireccion.SetValue(str(row[3]))

self.txtphone.SetValue(str(row[8]))
                self.txtdui.SetValue(str(row[4]))

                self.txtnit.SetValue(str(row[5]))

self.txtcargo.SetValue(str(row[6]))
                self.VerImagen(self.ruta) #Llamar al
metodo para ver la imagen
# metodo para ver la imagen
def VerImagen(self,path):
    if path=='':
        pass
    else:
        self.ruta=path
        #self.ruta="imgs/blank_img.png"
        print self.ruta
        img = wx.Image(self.ruta, wx.BITMAP_TYPE_ANY)
        #Escala la imagen preservando la relacion de
aspecto, es decir no deformarla visualmente
        W = img.GetWidth() #ancho de img
        H = img.GetHeight() #alto de img
        if W > H:
            NewW = self.PhotoMaxSize
            NewH = self.PhotoMaxSize * H / W
        else:
            NewH = self.PhotoMaxSize
            NewW = self.PhotoMaxSize * W / H
        img = img.Scale(NewW,NewH)

```

```

        self.imagen.SetBitmap(wx.BitmapFromImage(img))
        self.Refresh()
        # evento que actualiza los datos del registro
        seleccionado
        def Actualizar( self, event ):
            param =
            {'nombre':self.txtnom.GetValue(), 'apelm':self.txtape.GetValue(), 'direcm':self.txtdireccion.GetValue(), 'telefm':self.txttelefono.GetValue(), 'duim':self.txtdui.GetValue(), 'nitm':self.txtnit.GetValue(), 'cargom':self.txtcarga.GetValue(), 'id_seleccionado':self.id_seleccionado}
            sql=""" UPDATE usuarios SET nombre=:nombre,
            apellido=:apelm, direccion=:direcm, telefono=:telefm,
            dui=:duim, nit=:nitm, carga=:cargom WHERE
            id_usuario=:id_seleccionado"""
            typesql='U'
            self.db.query(sql,param,typesql)
            self.cargar()
            self.limpiar()
        # evento que elimina el registro seleccionado
        def Eliminar(self, event):
            id_seleccionado = str(self.txtid.GetValue())
            if id_seleccionado != '':
                caja_dialogo = wx.MessageDialog(None,
                '¿Realmente desea eliminar este registro?', 'Eliminar',
                wx.YES_NO | wx.ICON_QUESTION)
                if caja_dialogo.ShowModal() == wx.ID_YES:
                    sql="""DELETE FROM usuarios WHERE
                    id_usuario=:id_seleccionado"""

            data_param={'id_seleccionado':id_seleccionado}
            typesql='D'

            self.rows=self.db.query(sql,data_param,typesql)
            self.cargar()
            self.limpiar()
            caja_dialogo.Destroy()
        # funcion que limpia los cuandros de texto asi como
        reestablece la imagen, para luego cargar nuevamente el
        listado
        def limpiar(self):
            self.txtid.SetValue("")
            self.txtnom.SetValue("")
            self.txtape.SetValue("")

```

```
self.txtdireccion.SetValue("")
self.txtphone.SetValue("")
self.txtdui.SetValue("")
self.txtnit.SetValue("")
self.txtcargo.SetValue("")
self.ruta='imgs/blank_img.png'
self.VerImagen(self.ruta)
```