UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES

Diseño de la plataforma tecnológica para Centros de Comunicaciones mediante VoIP

Este Jurado; una vez realizado el examen del presente trabajo ha evaluado su contenido con el resultado: Veinte (20) puntos

J U R A D O E X A M I N A D O R

Firma:
Nombre: Victor Destefano

Firma:
Nombre: José Rafael

Firma:
Nombre: José Luis Díaz G.

REALIZADO POR	*Marieugenia Fernández Mendoza* *Vincenzo Saturno Braggiato*
PROFESOR GUIA	*José Luis Díaz*
FECHA	*24 de Septiembre de 2007*

Índice General

Índice General i

Índice de Figuras ii

Índice de Tablas ii

Apéndice A: Antecedentes de VoIP 1

Apéndice B: Funcionamiento básico del Protocolo SIP 2

Apéndice C: Funciones de los Protocolos RTP y RTCP 6

Apéndice D: Protocolo MGCP 8

Apéndice E: Protocolo IAX 9

Apéndice F: Estándar USB 10

Apéndice G: Configuración de Dirección IP estática en los ATA GrandStream HandyTone-386 12

Apéndice H: Parámetros de Configuración del Teléfono IP Linksys SPA921 13

Apéndice I: Tabla de Comandos de Asterisk 15

Apéndice J: Archivo *sip.conf* 16

Apéndice K: Archivo *extensions.conf* 21

Apéndice L: Ambiente de Desarrollo Boa Constructor 22

Apéndice M: Código Fuente de la Interfaz Gráfica 23

Apéndice N: Tabla de Comandos del Visor CrystalFontz CFA634-NFA-KU 84

Anexo A: Softphone X-lite 85

Anexo B: Tabla de Tarifas de la empresa NGT 86

Índice de Figuras

Figura 15. Modelo de convergencia de servicios en una red SIP 3
Figura 16. Interfaz Web para la configuración del teléfono IP Linksys SPA 921..... 14
Figura 17. Ambiente de Desarrollo Boa Constructor 22
Figura 18. Softphone X-Lite 85

Índice de Tablas

Tabla 6. Lista de algunas opciones permitidas por Asterisk..... 15
Tabla 7. Comandos de Control del visor CrystalFontz CFA634-NFA-KU..... 84

Apéndice A: Antecedentes de VoIP

En sus inicios, el sistema telefónico conmutado público se utilizaba principalmente para el tráfico de voz y muy poco para el transporte de datos. Sin embargo, este último empezó a incrementarse paulatinamente, y para el año 1999 aproximadamente, la cantidad de bits de datos transmitidos igualó a la de bits de voz. Luego en el 2002, el volumen de tráfico de datos se hizo mayor que el de voz, y así continúa creciendo de manera exponencial, mientras que el tráfico de voz solo reflejaba un crecimiento anual de 5%. Como consecuencia de estas cifras, los operadores de redes de conmutación de paquetes se interesaron en transportar voz, sin considerar un problema el hecho de que estas redes están especialmente diseñadas para transportar datos, ya que para el tráfico de voz se requiere un ancho de banda minúsculo. De esta manera nació la telefonía de Internet o VoIP, pues el principal aporte de esta innovación era reducir costos tanto para las operadoras como para los usuarios finales y permitir el envío de llamadas telefónicas a través de las líneas de datos (Tanenbaum, 2003).

Así mismo, con la implantación definitiva del protocolo IP, para la interconexión de redes, y el abaratamiento de los DSP (Digital Signal Processing), que son claves en la compresión y descompresión de la voz, se ha hecho posible el despegue de estas tecnologías y por lo tanto, la transmisión de la voz sobre las redes IP (Huidobro, 2003).

De esta manera surgió y se fue expandiendo VoIP, pero así mismo había que tomar en cuenta ciertos detalles para su final implantación, y es aquí cuando se empezaron a crear y a adaptar protocolos y recomendaciones con la idea de estandarizar y unificar el sistema para lograr un óptimo funcionamiento. A continuación se detallarán los protocolos en los que se fundamentan las redes que utilizan VoIP.

Apéndice B: Funcionamiento básico del Protocolo SIP

SIP tiene una variedad de características como la espera de llamadas, bloque de llamada, codificación y autenticación, así como también posee la capacidad de colocar llamadas de una computadora en un teléfono común, siempre y cuando haya disponible una puerta de enlace entre Internet y el sistema telefónico.

B.1. Direccionamiento

Las direcciones SIP, también llamadas localizadores universales de recursos (URL) SIP, existen en la forma de *usuario@hosts*, similar a una dirección de correo electrónico. Estos URL pueden indicar que el usuario pertenece a un dominio (sip:usuario@dominio), a un determinado computador (sip:usuario@computador), a una dirección IP de un computador (sip:usuario@dirección_IP), o incluso a un número de teléfono (número E.164) accesible a través de una pasarela IP/RTPC (sip:número_teléfono@pasarela).

B.2. Localización de un Servidor

Un cliente puede enviar una petición SIP directamente a un servidor Proxy configurado localmente, o bien a la dirección IP y puerto del correspondiente URL SIP.

B.3. Transacciones SIP

Cuando se ha resuelto el tema de la dirección, el cliente envía una o más peticiones SIP y recibe una o más respuestas desde el servidor especificado. Todas las peticiones y respuestas asociadas con esa actividad están consideradas como parte de una transacción SIP. Para una mayor simplicidad, los campos de cabecera en todos

los mensajes de petición coinciden con los campos de cabecera en todos los mensajes de respuesta.

B.4. Localización de un Usuario

Para los servicios de localización, SIP necesita acomodar la flexibilidad y movilidad de los sistemas finales IP. Estas localizaciones pueden estar registradas con el servidor SIP o con otros fuera del ámbito de SIP. La acción y resultado de localizar a un usuario depende entonces del tipo de servidor SIP que se este utilizando. Un servidor de redirección simplemente devuelve la lista completa de localizaciones y permite que el cliente localice directamente al usuario. Un servidor Proxy puede probar las direcciones en paralelo hasta que la llamada tenga éxito. La convergencia de estos servicios y el funcionamiento del protocolo se ilustran con la figura 15.

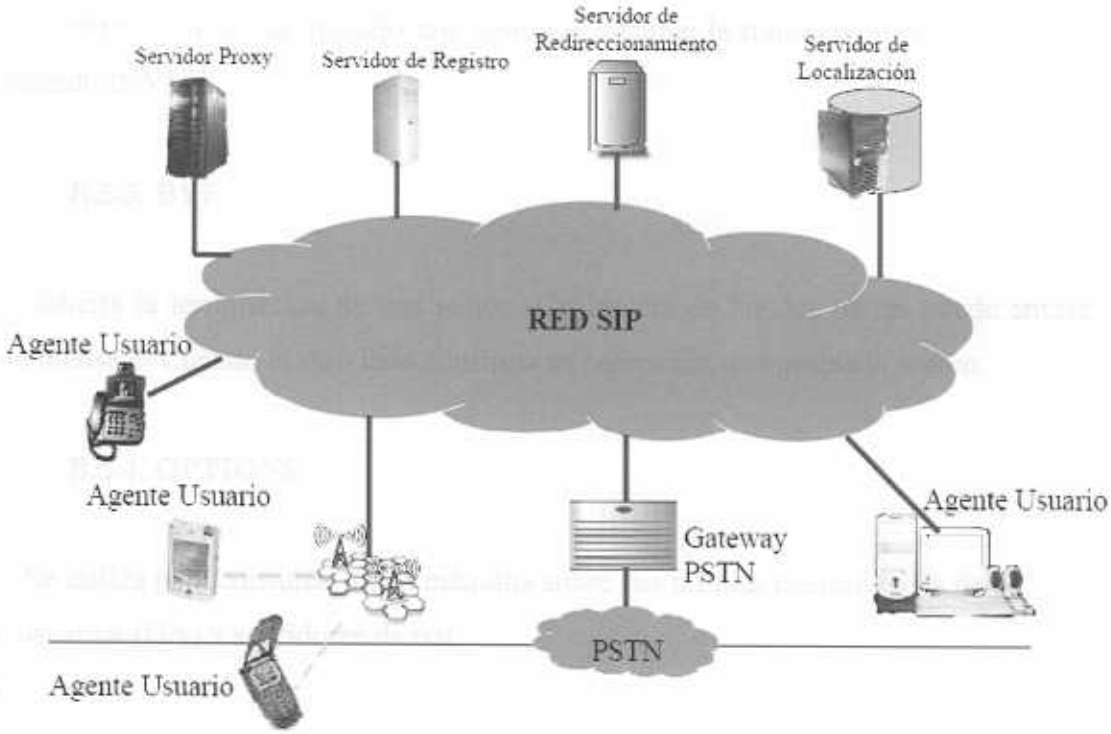

Figura 15. Modelo de convergencia de servicios en una red SIP

B.5. Métodos y mensajes básicos del protocolo SIP

El protocolo SIP se basa en texto y está modelado en HTTP (HyperText Transfer Protocol). Una parte envía un mensaje en texto ASCII que consiste en un nombre de método en la primera línea, seguido por líneas adicionales que contienen encabezados para pasar los parámetros. Estos seis métodos son los siguientes (Tanenbaum, 2003):

B.5.1. INVITE

Solicita el inicio de una sesión. Es un mensaje para invitar al usuario a realizar una conexión o llamada; si se acepta, se responde con un código tipo HTTP.

B.5.2. ACK

Confirma que se ha iniciado una sesión. Concluye la transacción iniciada por el comando INVITE.

B.5.3. BYE

Solicita la terminación de una sesión. Cualquiera de las dos partes puede enviar este mensaje. Cuando el otro lado confirma su recepción, se termina la sesión.

B.5.4. OPTIONS

Se utiliza para consultar a una máquina sobre sus propias posibilidades de agentes de usuarios (UA) y servidores de red.

B.5.5. CANCEL

Cancela una solicitud que está en proceso, es decir es empleado para anular una llamada pendiente, aunque su recepción por parte de un UAS, no garantiza que éste no responda posteriormente, sino que simplemente constituye una sugerencia realizada por el remitente para optimizar el uso de la red.

B.5.6. REGISTER

Informa a un servidor de redirección sobre la ubicación actual del usuario. Estos mensajes proporcionan la localización de un UA a los servidores de registro. En ellos, los UA clientes notifican a los Proxys o los servidores de redirección, la dirección o direcciones en las cuales se encuentra un usuario.

De todos ellos, los cuatro primeros intervienen en el establecimiento de llamadas. El método OPTIONS permite intercambiar información sobre las capacidades de los componentes de un sistema SIP y el método REGISTER es la base para los servicios de localización.

Apéndice C: Funciones de los Protocolos RTP y RTCP

C.1. Funciones de RTP

- Fragmentación: cada paquete RTP contiene un número de secuencia empleado para la detección de perdidas durante el reensamblado del mensaje en recepción.
- Sincronización intramedia: las aplicaciones emplean buffers que utilizan las marcas temporales proporcionadas por RTP para medir el jitter.
- Identificación del tipo de carga: que describe el tipo de codificación que se ha empleado en la generación del paquete, según las variaciones en las condiciones de la red.
- Indicación de trama: RTP utiliza un bit para marcar en informar al receptor el principio y fin de cada una de las tramas.
- Identificación de fuente: para identificar al usuario que generó un determinado paquete.

C.2. Funciones de RTCP

- Realimentación sobre la QoS: los receptores de una sesión emplean RTCP para informar al emisor sobre la calidad de su recepción, incluyendo número de paquetes perdidos y jitter, entre otras.
- Sincronización intermedia: para mejorar los niveles de flexibilidad, el audio y el video suelen transportarse en flujos diferentes que RTCP se encarga de sincronizar en el receptor, incluso en el caso de que los flujos procedan de fuentes distintas.
- Identificación: contienen información de cada participante de la sesión, tal como la dirección de correo electrónico, número de teléfono, nombre, etc.

- Control de la sesión: RTCP permite a un participante indicar que deja la sesión (paquete BYE), así como el intercambio de paquetes cortos entre participantes.

Periódicamente, todos los participantes en una sesión transmiten un paquete con la información citada anteriormente. Estos paquetes se envían a la misma dirección (multicast o unicast) como un flujo RTP pero a un puerto diferente.

Apéndice D: Protocolo MGCP

El protocolo Media Gateway Control Protocol (MGCP) define la comunicación entre los elementos de control de llamadas y las pasarelas de telefonía. Se trata de un protocol que permite a un coordinador central monitorear los eventos que suceden en los teléfonos IP y las pasarelas, así como dar a éstos últimos instrucciones para enviar datos a determinadas direcciones. Es un protocolo de tipo maestro/esclavo, en el que se espera que las pasarelas ejecuten los comandos enviados por agentes de llamada.

D.1. Componentes

Un sistema de MGCP básico esta formado por una o mas pasarelas y, al menos, un agente; el cual será notificado de cualquier evento que ocurra en las pasarelas que controla y, además, enviara comandos a dichas pasarelas. Cada pasarela tiene conectado un teléfono, cuando el llamante descuelga el receptor, la pasarela envía la señal correspondiente al agente, genera el tono de invitación a marcar y recoge los dígitos marcados por el usuario. Estos dígitos son enviados al agente que determina, a partir de esta información, la pasarela destino y le envía los comandos adecuados para que genere los tonos de llamada en el teléfono del abonado llamado. Cuando este ultimo descuelgue el receptor, se establecerá una sesión RTP/RTCP entre las dos pasarelas que permitirá el intercambio de datos.

D.2. Comandos y Señales

Los comandos y señales de MGCP están definidos como paquetes IP, por lo que son independientes del sistema operativo y del lenguaje de programación empleado; de este modo, un agente puede ejecutarse en una plataforma de propósito general conectada a la red. La verdadera potencia de MGCP es la capacidad para definir paquetes, es decir, parámetros de comandos y señales empleados para soportar dispositivos específicos.

Apéndice E: Protocolo IAX

IAX (*Inter Asterisk eXchange)* es uno de los protocolos utilizado por Asterisk, para manejar conexiones VoIP entre servidores Asterisk, y entre servidores y clientes que también utilizan protocolo IAX.

Los objetivos de este protocolo son minimizar el ancho de banda necesario para señalización, y proporcionar soporte interno para transparencia en la traducción de direcciones de red (NAT), todo ello permitiendo la escalabilidad necesaria para futuras mejoras de funcionalidades. No es necesaria ninguna configuración adicional para hacer funcionar IAX para atravesar firewalls NAT.

En lugar de usar Real-time Transfer Protocol (RTP) que emplea muchas cabeceras, IAX utiliza User Datagram Protocol (UDP) sobre un único puerto de Internet (Puerto 4569) para transmitir y recibir la señalización y el medio. Adicionalmente IAX puede triplicar el número de llamadas enviadas a través de un solo megabit cuando se utiliza por ejemplo el codec con compresión G.729, pues al emplearlo a través de protocolo IAX se permiten efectuar al menos 103 llamadas a través de 1M bit de ancho de banda simétrico.

Toda la señalización tiene lugar dentro de una consistente capa 2 de datos. Los tonos multifrecuencia (DTMF) se envían siempre a través del mismo camino como el resto de señalizaciones de datos y de esta forma son retransmitidos de forma fiable al otro extremo. El protocolo IAX, también transmite paquetes de audio con tan sólo 4 bytes de cabecera y los comandos emplean un ancho de banda muy reducido.

Apéndice F: Estándar USB

El Universal Serial Bus (bus universal en serie) fue creado en 1996 por siete empresas: IBM, Intel, Northern Telecom, Compaq, Microsoft, Digital Equipment Corporation y NEC. USB nace como un estándar de entrada/salida de velocidad media-alta, que va a permitir conectar dispositivos que hasta ahora requerían de una tarjeta especial para sacarles todo el rendimiento, lo que ocasionaba un encarecimiento de los productos, ya que obligaban a adquirir una tarjeta para cada dispositivo.

USB proporciona un único conector para solventar casi todos los problemas de comunicación con el exterior, pudiéndose formar una auténtica red de periféricos de hasta 127 elementos casi de cualquier tipo: desde el teclado al modem, pasando por ratones, impresoras, altavoces, monitores, displays, scaners, cámaras digitales, de video, etc., sin necesidad de que nuestro PC disponga de un conector dedicado para cada uno de estos elementos, permitiendo ahorrar espacio y dinero.

Adicionalmente cuenta con la famosa característica PnP (Plug&Play), permitiendo a los dispositivos ser conectados o desconectados al sistema sin necesidad de reiniciar. Cuando se conecta un nuevo dispositivo, el servidor lo enumera y agrega el software necesario para que pueda funcionar. Otras características resaltantes son:

- Dos velocidades de acceso, una baja de 1,5 Mbps para dispositivos lentos como pueden ser joysticks o teclados, y otra alta de 12 Mbps para los dispositivos que necesiten mayor ancho de banda.
- Topología en estrella, lo que implica la necesidad de dispositivos tipo "hub" que centralicen las conexiones.
- Permite suministrar energía eléctrica a dispositivos que no tengan un alto consumo, lo que elimina la necesidad de conectar dichos periféricos a la red

eléctrica, con sus correspondientes fuentes de alimentación, como ahora ocurre por ejemplo con los módems externos.

- Los cables USB están disponibles en una gran variedad de longitudes que va desde los 0,5 hasta 5 metros, lo cual es el máximo de largo de cable permitido por las especificaciones del USB. Sin embargo, para conectar dispositivos a mayores distancias, la soluciones propuestas por el estándar USB son:
  - El uso de Hubs, pudiendo encadenar 5 cables periféricos USB y 4 hubs para obtener un máximo de 25 metros.
  - Una solución más conveniente y económica, es usar los denominados Cables de Extensión Activos, que básicamente son cables de 5 metros de longitud con un puerto de hub integrado. De esta forma, se pueden conectar hasta 4 de estos cables, más un cable periférico USB sencillo de 5 metros para obtener el máximo permitido por el estándar (25 metros).

Apéndice G: Configuración de Dirección IP estática en los ATA GrandStream HandyTone-386

Los pasos a seguir para configurar una dirección IP fija al Adaptador Telefónico ATA GrandStream HandyTone-386, se listan a continuación:

- Inicialmente se debe cambiar al modo "Static IP": descolgando el auricular, presionando cuatro veces la tecla asterisco (*), seguidamente la opción "9".
- Para modificar esta dirección IP de forma manual: se presiona nuevamente cinco veces consecutivas "*" (en este momento la grabadora comunica la IP actual), y finalmente se introduce la dirección deseada marcando los 12 dígitos correspondientes en el teclado del teléfono. Es preciso destacar que si la dirección es, por ejemplo: 172.17.32.7, se debe marcar en el teléfono: 172017032007.

Apéndice H: Parámetros de Configuración del Teléfono IP Linksys SPA921

Para la realización de llamadas de prueba utilizando el teléfono IP Linksys SPA921, los parámetros modificados como administrador del teléfono (Admin Login) en la sección relativa a la extensión 1 (Ext1), fueron:

- Display Name, User ID, Auth ID: se colocó uno de los usuarios que se encuentran registrados en el sip.conf para su autentificación. En este caso *usuario2*.
- Password: la clave respectiva del usuario anteriormente citado (*clave2*).
- Proxy: dirección IP en la que se registra el teléfono, es decir la aplicación Asterisk (*172.17.32.63*).
- Preferred Codec: codificador utilizado, que debe coincidir con alguno de los habilitados en el sip.conf (*G711u*).

Además, la figura 16 muestra la interfaz Web con los parámetros mencionados anteriormente.

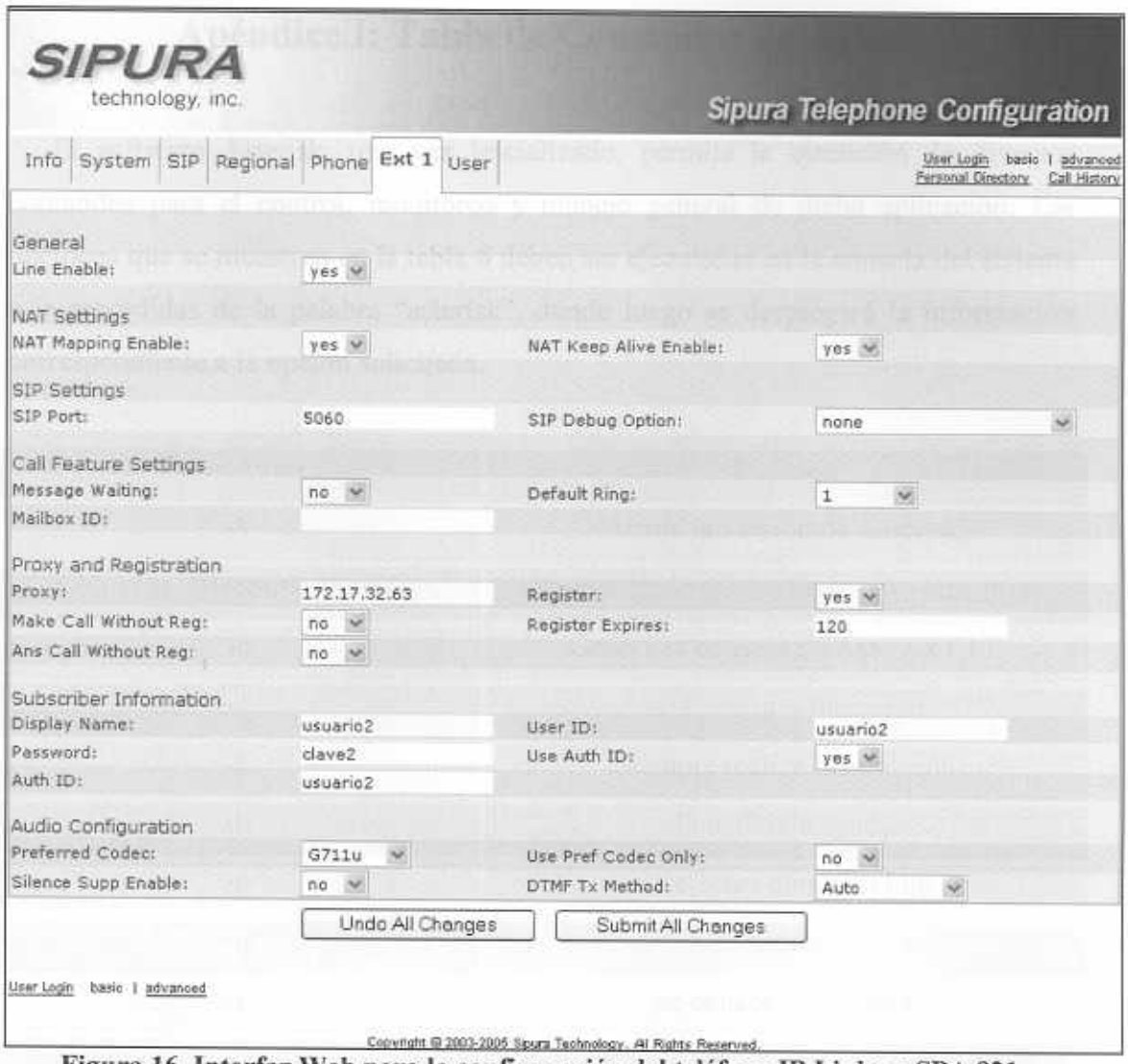

Figura 16. Interfaz Web para la configuración del teléfono IP Linksys SPA 921

Apéndice I: Tabla de Comandos de Asterisk

El software Asterisk, una vez inicializado, permite la ejecución de diversos comandos para el control, monitoreo y manejo general de dicha aplicación. Las opciones que se muestran en la tabla 6 deben ser ejecutadas en la consola del sistema e ir precedidas de la palabra "asterisk", donde luego se desplegará la información correspondiente a la opción solicitada.

Comando	Función
-V	Arroja la versión de Asterisk.
-C <archivodeconfiguración>	Usa el archivo de configuración específico.
-c	Genera la consola de Asterisk CLI.
-f	No Realiza bifurcación.
-F	Siempre realiza bifurcación
-h	Pantalla de ayuda
-p	Se ejecuta como un hilo
-q	Modo
-r	Se conecta a Asterisk
-R	Se conecta a Asterisk e intenta re conexión si se desconecta
-t	Graba archivos de sonido en var/tmp y luego los mueve al sitio donde corresponde.
-T	Muestra la hora en formato [Mes día hh:mm:ss]
-v	Aumentar Verbosidad

Tabla 1. Lista de algunas opciones permitidas por Asterisk

Apéndice J: Archivo *sip.conf*

```
;;;;;;;;; SIP.CONF ;;;;;;;;;

;Contexto general

[general]
port=5060
bindaddr=0.0.0.0
dtmfmode=info
disallow=all
allow=ulaw
allow=g729

[usuario1]
username=usuario1
type=friend
secret=clave1
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60

[usuario2]
username=usuario2
type=friend
secret=clave2
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60

[usuario3]
username=usuario3
type=friend
secret=clave3
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60

[usuario4]
username=usuario4
type=friend
```

```
secret=clave4
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60

[usuario5]
username=usuario5
type=friend
secret=clave5
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60

[usuario6]
username=usuario6
type=friend
secret=clave6
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60

[usuario7]
username=usuario7
type=friend
secret=clave7
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60

[usuario8]
username=usuario8
type=friend
secret=clave8
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
```

```
incominglimit=1
rtptimeout=60

[usuario9]
username=usuario9
type=friend
secret=clave9
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60

[usuario10]
username=usuario10
type=friend
secret=clave10
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60

[usuario11]
username=usuario11
type=friend
secret=clave11
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60

[usuario12]
username=usuario12
type=friend
secret=clave12
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60

[usuario13]
username=usuario13
type=friend
```

```
secret=clave13
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60

[usuario14]
username=usuario14
type=friend
secret=clave14
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60

[usuario15]
username=usuario15
type=friend
secret=clave15
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60

[usuario16]
username=usuario16
type=friend
secret=clave16
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60

[usuario17]
username=usuario17
type=friend
secret=clave17
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
```

```
incominglimit=1
rtptimeout=60

[usuario18]
username=usuario18
type=friend
secret=clave18
qualify=yes
host=dynamic
context=cdc
allow=ulaw
allow=g729
incominglimit=1
rtptimeout=60
```

Apéndice K: Archivo *extensions.conf*

```
;;;;;;;;;;; EXTENSIONS.CONF ;;;;;;;;;;;;;;;

; Variables globales
[globals]
INTER = 8020
PREF2 = 802058
LOCALCALL = 802058212

; Contexto CDC
[cdc]
exten => _00.,1,Dial(SIP/${INTER}${EXTEN:2}@216.72.89.247,140) ; LDI
exten => _02.,1,Dial(SIP/${PREF2}${EXTEN:1}@216.72.89.247,140) ; LDN
exten => _04.,1,Dial(SIP/${PREF2}${EXTEN:1}@216.72.89.247,140) ; Cel
exten => _[2-9]XXXXXX,1,Dial(SIP/${LOCALCALL}${EXTEN}@216.72.89.247,140)
; Locales
```

Apéndice L: Ambiente de Desarrollo Boa Constructor

El Boa Constructor es un ambiente de desarrollo multiplataforma y de fuente abierta, para el lenguaje de programación Python. Además, es un constructor muy amigable para crear Interfaces Gráficas de Usuario (GUI), apoyándose sobre wxPython, que no es mas que un conjunto de librerías avanzadas, igualmente *open source*, que proveen una gran cantidad de objetos como ventanas, diálogos, botones, cajas de texto, etc. En la figura 17 se observan el editor y las herramientas que ofrece el Boa Constructor, el cual se inicia desde la consola con el comando *python Boa.py*.

Figura 17. Ambiente de Desarrollo Boa Constructor

Apéndice M: Código Fuente de la Interfaz Gráfica

M.1. ApCDC.py

```
#!/usr/bin/env python
#Boa:App:BoaApp

import wx

import Frame1

modules ={'ApCDC': [0, '', 'ApCDC.py'],
 'Frame1': [1, 'Main frame of Application', 'Frame1.py'],
 'Frame2': [0, '', 'Frame2.py']}

class BoaApp(wx.App):
    def OnInit(self):
        wx.InitAllImageHandlers()
        self.main = Frame1.create(None)
        self.main.Show()
        self.SetTopWindow(self.main)
        return True

def main():
    application = BoaApp(0)
    application.MainLoop()

if __name__ == '__main__':
    main()
```

M.2. Frame1.py

```
#Boa:Frame:Frame1

import wx
import wx.lib.buttons
from wx.lib.anchors import LayoutAnchors
import wx.lib.masked.timectrl
import Frame2
#import Frame3
import time
import binascii
import string
import os
import threading
import datetime
```

```
def create(parent):
    return Frame1(parent)

[wxID_FRAME1, wxID_FRAME1BUTTON1, wxID_FRAME1BUTTON10,
wxID_FRAME1BUTTON11,
 wxID_FRAME1BUTTON12, wxID_FRAME1BUTTON13, wxID_FRAME1BUTTON14,
 wxID_FRAME1BUTTON15, wxID_FRAME1BUTTON16, wxID_FRAME1BUTTON17,
 wxID_FRAME1BUTTON18, wxID_FRAME1BUTTON19, wxID_FRAME1BUTTON2,
 wxID_FRAME1BUTTON3, wxID_FRAME1BUTTON4, wxID_FRAME1BUTTON5,
 wxID_FRAME1BUTTON6, wxID_FRAME1BUTTON7, wxID_FRAME1BUTTON8,
 wxID_FRAME1BUTTON9, wxID_FRAME1CHECKBOX1, wxID_FRAME1CHECKBOX10,
 wxID_FRAME1CHECKBOX11, wxID_FRAME1CHECKBOX12,
wxID_FRAME1CHECKBOX13,
 wxID_FRAME1CHECKBOX14, wxID_FRAME1CHECKBOX15,
wxID_FRAME1CHECKBOX16,
 wxID_FRAME1CHECKBOX17, wxID_FRAME1CHECKBOX18,
wxID_FRAME1CHECKBOX2,
 wxID_FRAME1CHECKBOX3, wxID_FRAME1CHECKBOX4, wxID_FRAME1CHECKBOX5,
 wxID_FRAME1CHECKBOX6, wxID_FRAME1CHECKBOX7, wxID_FRAME1CHECKBOX8,
 wxID_FRAME1CHECKBOX9, wxID_FRAME1STATICTEXT1,
wxID_FRAME1STATICTEXT2,
 wxID_FRAME1TRAMPA, wxID_FRAME1X1, wxID_FRAME1X10, wxID_FRAME1X11,
 wxID_FRAME1X12, wxID_FRAME1X13, wxID_FRAME1X14, wxID_FRAME1X15,
 wxID_FRAME1X16, wxID_FRAME1X17, wxID_FRAME1X18, wxID_FRAME1X2,
wxID_FRAME1X3,
 wxID_FRAME1X4, wxID_FRAME1X5, wxID_FRAME1X6, wxID_FRAME1X7,
wxID_FRAME1X8,
 wxID_FRAME1X9,
] = [wx.NewId() for _init_ctrls in range(59)]

class MyThread ( threading.Thread ):
    def run(self):
        while 1:
            a=0
            while (a==0):
                ru2=open('/cdc','r')
                for line2 in ru2:
                    senal=line2.find("answered SIP/usuario")
                    senal2=line2.find("exited non-zero on
'SIP/usuario")
                    if (senal >= 0):
                        start=datetime.datetime.now()
                        ini=time.localtime(time.time())
                        horaini=time.strftime("%X",ini)
                        fsalida = open('/dev/usb/tts/0','w')
                        limpiar = binascii.a2b_hex('0c')
                        fsalida.write(limpiar)
                        cdc = binascii.a2b_hex('110800')+'CDC'
                        fsalida.write(cdc)
                        hi = binascii.a2b_hex('110001')+horaini
                        fsalida.write(hi)
                        fsalida.flush()
                        ru2=open('/cdc','r')
```

```
                        ru2data=ru2.readlines()     # busco el
prefijo para el cobro
                        ru2.close()
                        rut=open('/tarifas.txt','r')
                        for linea in rut:
                            valor=0
                            c='/8020'+linea.split("|")[1]
                            for line2 in ru2data:
                                destino=line2.find(c)
                                if (destino >= 0):
                                    valor=linea.split('|')[2]
                                    v=(int(valor)*.0167)
                                    a=1
                                    break
                    elif (senal2 >= 0):
                        ru3=open('/cdc','w')
                        ru3.write(' borrar ')
                        ru3.close()
                        break
            b=0
            elapsed=0
            while(b==0):
                ru3=open('/cdc','r')
                for line3 in ru3:
                    senal3=line3.find("exited non-zero on
'SIP/usuario")
                    if (senal3 >= 0):
                        ru3=open('/cdc','w')
                        ru3.write(' borrar ')
                        ru3.close()
                        b=1
                        break
                    else: # (senal3 < 0):
                        time.sleep(0.3)
                        end=datetime.datetime.now()
                        elapsed=end-start
                        e=elapsed.seconds
                        costo=v*int(e)
                        fsalida=open('/dev/usb/tts/0','w')
                        timev =
binascii.a2b_hex('110002')+str(elapsed)
                        fsalida.write(timev)
                        bb = binascii.a2b_hex('08')

fsalida.write(bb),fsalida.write(bb),fsalida.write(bb)

fsalida.write(bb),fsalida.write(bb),fsalida.write(bb)
                        fsalida.write(bb)
                        costov =
binascii.a2b_hex('110203')+'Costo: '+str(round(costo))+' Bs'
                        fsalida.write(costov)
                        hide = binascii.a2b_hex('04')
                        fsalida.write(hide)
                        fsalida.flush()
```

```
                        fsalida.close()

class Frame1(wx.Frame):
    def _init_ctrls(self, prnt):
        # generated method, don't edit
        wx.Frame.__init__(self, id=wxID_FRAME1, name='',
parent=prnt, pos=wx.Point(155, 52), size=wx.Size(622, 437),
style=wx.DEFAULT_FRAME_STYLE, title='Frame1')
        self.SetClientSize(wx.Size(622, 437))
        self.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL, wx.NORMAL,
False, 'Newspaper'))
        self.Show(False)
        self.SetBackgroundColour(wx.Colour(0, 98, 187))

        self.staticText1 =
wx.StaticText(id=wxID_FRAME1STATICTEXT1, label='CENTRO DE
COMUNICACIONES', name='staticText1', parent=self, pos=wx.Point(77,
0), size=wx.Size(467, 29), style=0)
        self.staticText1.Center(wx.HORIZONTAL)
        self.staticText1.SetFont(wx.Font(16, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Ikarus'))
        self.staticText1.SetBackgroundColour(wx.Colour(0, 0, 0))
        self.staticText1.SetForegroundColour(wx.Colour(255, 255,
255))

        self.button1 = wx.Button(id=wxID_FRAME1BUTTON1,
label='Cabina 1', name='button1', parent=self, pos=wx.Point(40,
48), size=wx.Size(96, 72), style=0)
        self.button1.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button1.SetMinSize(wx.Size(88, 72))
        self.button1.SetBackgroundColour(wx.Colour(255, 255, 255))
        self.button1.Bind(wx.EVT_BUTTON, self.OnButton1Button,
id=wxID_FRAME1BUTTON1)

        self.button2 = wx.Button(id=wxID_FRAME1BUTTON2,
label='Cabina 2', name='button2', parent=self, pos=wx.Point(152,
48), size=wx.Size(96, 72), style=0)
        self.button2.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button2.SetMinSize(wx.Size(27, 21))
        self.button2.SetBackgroundColour(wx.Colour(255, 255, 255))
        self.button2.Bind(wx.EVT_BUTTON, self.OnButton2Button,
id=wxID_FRAME1BUTTON2)

        self.button3 = wx.Button(id=wxID_FRAME1BUTTON3,
label='Cabina 3', name='button3', parent=self, pos=wx.Point(264,
48), size=wx.Size(96, 72), style=0)
        self.button3.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button3.SetMinSize(wx.Size(88, 72))
        self.button3.SetBackgroundColour(wx.Colour(255, 255, 255))
        self.button3.Bind(wx.EVT_BUTTON, self.OnButton3Button,
id=wxID_FRAME1BUTTON3)
```

```
        self.button4 = wx.Button(id=wxID_FRAME1BUTTON4,
label='Cabina 4', name='button4', parent=self, pos=wx.Point(376,
48), size=wx.Size(96, 72), style=0)
        self.button4.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button4.SetMinSize(wx.Size(78, 66))
        self.button4.SetBackgroundColour(wx.Colour(255, 255, 255))
        self.button4.Bind(wx.EVT_BUTTON, self.OnButton4Button,
id=wxID_FRAME1BUTTON4)

        self.button5 = wx.Button(id=wxID_FRAME1BUTTON5,
label='Cabina 5', name='button5', parent=self, pos=wx.Point(488,
48), size=wx.Size(96, 72), style=0)
        self.button5.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button5.SetMinSize(wx.Size(78, 68))
        self.button5.SetBackgroundColour(wx.Colour(255, 255, 255))
        self.button5.SetBackgroundStyle(wx.BG_STYLE_SYSTEM)
        self.button5.Bind(wx.EVT_BUTTON, self.OnButton5Button,
id=wxID_FRAME1BUTTON5)

        self.button6 = wx.Button(id=wxID_FRAME1BUTTON6,
label='Cabina 6', name='button6', parent=self, pos=wx.Point(40,
136), size=wx.Size(96, 72), style=0)
        self.button6.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button6.SetMinSize(wx.Size(50, 72))
        self.button6.SetBackgroundColour(wx.Colour(255, 255, 255))
        self.button6.SetBackgroundStyle(wx.BG_STYLE_SYSTEM)
        self.button6.Bind(wx.EVT_BUTTON, self.OnButton6Button,
id=wxID_FRAME1BUTTON6)

        self.button7 = wx.Button(id=wxID_FRAME1BUTTON7,
label='Cabina 7', name='button7', parent=self, pos=wx.Point(152,
136), size=wx.Size(96, 72), style=0)
        self.button7.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button7.SetMinSize(wx.Size(66, 72))
        self.button7.SetBackgroundColour(wx.Colour(255, 255, 255))
        self.button7.SetBackgroundStyle(wx.BG_STYLE_SYSTEM)
        self.button7.Bind(wx.EVT_BUTTON, self.OnButton7Button,
id=wxID_FRAME1BUTTON7)

        self.button8 = wx.Button(id=wxID_FRAME1BUTTON8,
label='Cabina 8', name='button8', parent=self, pos=wx.Point(264,
136), size=wx.Size(96, 72), style=0)
        self.button8.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button8.SetMinSize(wx.Size(62, 56))
        self.button8.SetBackgroundColour(wx.Colour(255, 255, 255))
        self.button8.SetBackgroundStyle(wx.BG_STYLE_SYSTEM)
        self.button8.Bind(wx.EVT_BUTTON, self.OnButton8Button,
id=wxID_FRAME1BUTTON8)
```

```
        self.button9 = wx.Button(id=wxID_FRAME1BUTTON9,
label='Cabina 9', name='button9', parent=self, pos=wx.Point(376,
136), size=wx.Size(96, 72), style=0)
        self.button9.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button9.SetMinSize(wx.Size(78, 56))
        self.button9.SetBackgroundColour(wx.Colour(255, 255, 255))
        self.button9.SetBackgroundStyle(wx.BG_STYLE_SYSTEM)
        self.button9.Bind(wx.EVT_BUTTON, self.OnButton9Button,
id=wxID_FRAME1BUTTON9)

        self.button10 = wx.Button(id=wxID_FRAME1BUTTON10,
label='Cabina 10', name='button10', parent=self, pos=wx.Point(488,
136), size=wx.Size(96, 72), style=0)
        self.button10.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button10.SetMinSize(wx.Size(72, 38))
        self.button10.SetBackgroundColour(wx.Colour(255, 255,
255))
        self.button10.SetBackgroundStyle(wx.BG_STYLE_SYSTEM)
        self.button10.Bind(wx.EVT_BUTTON, self.OnButton10Button,
id=wxID_FRAME1BUTTON10)

        self.button11 = wx.Button(id=wxID_FRAME1BUTTON11,
label='Cabina 11', name='button11', parent=self, pos=wx.Point(40,
224), size=wx.Size(96, 72), style=0)
        self.button11.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button11.SetMinSize(wx.Size(54, 62))
        self.button11.SetBackgroundColour(wx.Colour(255, 255,
255))
        self.button11.SetBackgroundStyle(wx.BG_STYLE_SYSTEM)
        self.button11.Bind(wx.EVT_BUTTON, self.OnButton11Button,
id=wxID_FRAME1BUTTON11)

        self.button12 = wx.Button(id=wxID_FRAME1BUTTON12,
label='Cabina 12', name='button12', parent=self, pos=wx.Point(152,
224), size=wx.Size(96, 72), style=0)
        self.button12.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button12.SetMinSize(wx.Size(69, 56))
        self.button12.SetBackgroundColour(wx.Colour(255, 255,
255))
        self.button12.SetBackgroundStyle(wx.BG_STYLE_SYSTEM)
        self.button12.Bind(wx.EVT_BUTTON, self.OnButton12Button,
id=wxID_FRAME1BUTTON12)

        self.button13 = wx.Button(id=wxID_FRAME1BUTTON13,
label='Cabina 13', name='button13', parent=self, pos=wx.Point(264,
224), size=wx.Size(96, 72), style=0)
        self.button13.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button13.SetMinSize(wx.Size(58, 62))
```

```
        self.button13.SetBackgroundColour(wx.Colour(255, 255,
255))
        self.button13.SetBackgroundStyle(wx.BG_STYLE_SYSTEM)
        self.button13.Bind(wx.EVT_BUTTON, self.OnButton13Button,
id=wxID_FRAME1BUTTON13)

        self.button14 = wx.Button(id=wxID_FRAME1BUTTON14,
label='Cabina 14', name='button14', parent=self, pos=wx.Point(376,
224), size=wx.Size(96, 72), style=0)
        self.button14.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button14.SetMinSize(wx.Size(66, 50))
        self.button14.SetBackgroundColour(wx.Colour(255, 255,
255))
        self.button14.SetBackgroundStyle(wx.BG_STYLE_SYSTEM)
        self.button14.Bind(wx.EVT_BUTTON, self.OnButton14Button,
id=wxID_FRAME1BUTTON14)

        self.button15 = wx.Button(id=wxID_FRAME1BUTTON15,
label='Cabina 15', name='button15', parent=self, pos=wx.Point(488,
224), size=wx.Size(96, 72), style=0)
        self.button15.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button15.SetMinSize(wx.Size(50, 59))
        self.button15.SetBackgroundColour(wx.Colour(255, 255,
255))
        self.button15.SetBackgroundStyle(wx.BG_STYLE_SYSTEM)
        self.button15.Bind(wx.EVT_BUTTON, self.OnButton15Button,
id=wxID_FRAME1BUTTON15)

        self.button16 = wx.Button(id=wxID_FRAME1BUTTON16,
label='Cabina 16', name='button16', parent=self, pos=wx.Point(40,
312), size=wx.Size(96, 72), style=0)
        self.button16.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button16.SetMinSize(wx.Size(69, 72))
        self.button16.SetBackgroundColour(wx.Colour(255, 255,
255))
        self.button16.SetBackgroundStyle(wx.BG_STYLE_SYSTEM)
        self.button16.Bind(wx.EVT_BUTTON, self.OnButton16Button,
id=wxID_FRAME1BUTTON16)

        self.button17 = wx.Button(id=wxID_FRAME1BUTTON17,
label='Cabina 17', name='button17', parent=self, pos=wx.Point(152,
312), size=wx.Size(96, 72), style=0)
        self.button17.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button17.SetMinSize(wx.Size(62, 59))
        self.button17.SetBackgroundColour(wx.Colour(255, 255,
255))
        self.button17.SetBackgroundStyle(wx.BG_STYLE_SYSTEM)
        self.button17.Bind(wx.EVT_BUTTON, self.OnButton17Button,
id=wxID_FRAME1BUTTON17)
```

```
        self.button19 = wx.Button(id=wxID_FRAME1BUTTON19,
label='Salir', name='button19', parent=self, pos=wx.Point(432,
328), size=wx.Size(120, 48), style=0)
        self.button19.SetBackgroundColour(wx.Colour(255, 255,
255))
        self.button19.SetBackgroundStyle(wx.BG_STYLE_SYSTEM)
        self.button19.SetFont(wx.Font(13, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button19.Bind(wx.EVT_BUTTON, self.OnButton19Button,
id=wxID_FRAME1BUTTON19)

        self.button18 = wx.Button(id=wxID_FRAME1BUTTON18,
label='Cabina 18', name='button18', parent=self, pos=wx.Point(264,
312), size=wx.Size(96, 72), style=0)
        self.button18.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Newspaper'))
        self.button18.SetMinSize(wx.Size(58, 65))
        self.button18.SetBackgroundColour(wx.Colour(255, 255,
255))
        self.button18.SetBackgroundStyle(wx.BG_STYLE_SYSTEM)
        self.button18.Bind(wx.EVT_BUTTON, self.OnButton18Button,
id=wxID_FRAME1BUTTON18)

        self.checkBox2 = wx.CheckBox(id=wxID_FRAME1CHECKBOX2,
label='ocupada', name='checkBox2', parent=self, pos=wx.Point(160,
96), size=wx.Size(96, 23), style=0)
        self.checkBox2.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox2.SetValue(False)
        self.checkBox2.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox2Checkbox, id=wxID_FRAME1CHECKBOX2)

        self.checkBox3 = wx.CheckBox(id=wxID_FRAME1CHECKBOX3,
label='ocupada', name='checkBox3', parent=self, pos=wx.Point(272,
96), size=wx.Size(110, 23), style=0)
        self.checkBox3.SetValue(False)
        self.checkBox3.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox3.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox3Checkbox, id=wxID_FRAME1CHECKBOX3)

        self.checkBox4 = wx.CheckBox(id=wxID_FRAME1CHECKBOX4,
label='ocupada', name='checkBox4', parent=self, pos=wx.Point(384,
96), size=wx.Size(109, 24), style=0)
        self.checkBox4.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox4.SetValue(False)
        self.checkBox4.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox4Checkbox, id=wxID_FRAME1CHECKBOX4)

        self.checkBox5 = wx.CheckBox(id=wxID_FRAME1CHECKBOX5,
label='ocupada', name='checkBox5', parent=self, pos=wx.Point(496,
96), size=wx.Size(117, 23), style=0)
```

```
        self.checkBox5.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox5.SetValue(False)
        self.checkBox5.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox5Checkbox, id=wxID_FRAME1CHECKBOX5)

        self.checkBox6 = wx.CheckBox(id=wxID_FRAME1CHECKBOX6,
label='ocupada', name='checkBox6', parent=self, pos=wx.Point(48,
184), size=wx.Size(110, 24), style=0)
        self.checkBox6.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox6.SetValue(False)
        self.checkBox6.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox6Checkbox, id=wxID_FRAME1CHECKBOX6)

        self.checkBox7 = wx.CheckBox(id=wxID_FRAME1CHECKBOX7,
label='ocupada', name='checkBox7', parent=self, pos=wx.Point(160,
184), size=wx.Size(102, 24), style=0)
        self.checkBox7.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox7.SetValue(False)
        self.checkBox7.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox7Checkbox, id=wxID_FRAME1CHECKBOX7)

        self.checkBox8 = wx.CheckBox(id=wxID_FRAME1CHECKBOX8,
label='ocupada', name='checkBox8', parent=self, pos=wx.Point(272,
184), size=wx.Size(101, 24), style=0)
        self.checkBox8.SetValue(False)
        self.checkBox8.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox8.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox8Checkbox, id=wxID_FRAME1CHECKBOX8)

        self.checkBox9 = wx.CheckBox(id=wxID_FRAME1CHECKBOX9,
label='ocupada', name='checkBox9', parent=self, pos=wx.Point(384,
184), size=wx.Size(96, 24), style=0)
        self.checkBox9.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox9.SetValue(False)
        self.checkBox9.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox9Checkbox, id=wxID_FRAME1CHECKBOX9)

        self.checkBox10 = wx.CheckBox(id=wxID_FRAME1CHECKBOX10,
label='ocupada', name='checkBox10', parent=self, pos=wx.Point(496,
184), size=wx.Size(104, 24), style=0)
        self.checkBox10.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox10.SetValue(False)
        self.checkBox10.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox10Checkbox, id=wxID_FRAME1CHECKBOX10)

        self.checkBox11 = wx.CheckBox(id=wxID_FRAME1CHECKBOX11,
label='ocupada', name='checkBox11', parent=self, pos=wx.Point(48,
272), size=wx.Size(88, 24), style=0)
```

```
        self.checkBox11.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox11.SetValue(False)
        self.checkBox11.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox11Checkbox, id=wxID_FRAME1CHECKBOX11)

        self.checkBox12 = wx.CheckBox(id=wxID_FRAME1CHECKBOX12,
label='ocupada', name='checkBox12', parent=self, pos=wx.Point(160,
272), size=wx.Size(118, 24), style=0)
        self.checkBox12.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox12.SetValue(False)
        self.checkBox12.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox12Checkbox, id=wxID_FRAME1CHECKBOX12)

        self.checkBox13 = wx.CheckBox(id=wxID_FRAME1CHECKBOX13,
label='ocupada', name='checkBox13', parent=self, pos=wx.Point(272,
272), size=wx.Size(96, 24), style=0)
        self.checkBox13.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox13.SetValue(False)
        self.checkBox13.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox13Checkbox, id=wxID_FRAME1CHECKBOX13)

        self.checkBox14 = wx.CheckBox(id=wxID_FRAME1CHECKBOX14,
label='ocupada', name='checkBox14', parent=self, pos=wx.Point(384,
272), size=wx.Size(96, 24), style=0)
        self.checkBox14.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox14.SetValue(False)
        self.checkBox14.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox14Checkbox, id=wxID_FRAME1CHECKBOX14)

        self.checkBox15 = wx.CheckBox(id=wxID_FRAME1CHECKBOX15,
label='ocupada', name='checkBox15', parent=self, pos=wx.Point(496,
272), size=wx.Size(96, 24), style=0)
        self.checkBox15.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox15.SetValue(False)
        self.checkBox15.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox15Checkbox, id=wxID_FRAME1CHECKBOX15)

        self.checkBox16 = wx.CheckBox(id=wxID_FRAME1CHECKBOX16,
label='ocupada', name='checkBox16', parent=self, pos=wx.Point(48,
360), size=wx.Size(88, 24), style=0)
        self.checkBox16.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox16.SetValue(False)
        self.checkBox16.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox16Checkbox, id=wxID_FRAME1CHECKBOX16)

        self.checkBox17 = wx.CheckBox(id=wxID_FRAME1CHECKBOX17,
label='ocupada', name='checkBox17', parent=self, pos=wx.Point(160,
360), size=wx.Size(96, 24), style=0)
```

```
        self.checkBox17.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox17.SetValue(False)
        self.checkBox17.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox17Checkbox, id=wxID_FRAME1CHECKBOX17)

        self.checkBox18 = wx.CheckBox(id=wxID_FRAME1CHECKBOX18,
label='ocupada', name='checkBox18', parent=self, pos=wx.Point(272,
360), size=wx.Size(96, 24), style=0)
        self.checkBox18.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox18.SetValue(False)
        self.checkBox18.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox18Checkbox, id=wxID_FRAME1CHECKBOX18)

        self.checkBox1 = wx.CheckBox(id=wxID_FRAME1CHECKBOX1,
label='ocupada', name='checkBox1', parent=self, pos=wx.Point(48,
96), size=wx.Size(96, 24), style=0)
        self.checkBox1.SetFont(wx.Font(8, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.checkBox1.SetValue(False)
        self.checkBox1.Bind(wx.EVT_CHECKBOX,
self.OnCheckBox1Checkbox, id=wxID_FRAME1CHECKBOX1)

        self.x1 = wx.StaticText(id=wxID_FRAME1X1, label='',
name='x1', parent=self, pos=wx.Point(24, 24), size=wx.Size(32,
24), style=0)
        self.x1.Enable(True)
        self.x1.Show(False)
        self.x1.SetMinSize(wx.Size(0, 0))

        self.x2 = wx.StaticText(id=wxID_FRAME1X2, label='',
name='x2', parent=self, pos=wx.Point(144, 24), size=wx.Size(24,
24), style=0)
        self.x2.SetMinSize(wx.Size(0, 0))
        self.x2.Show(False)

        self.x3 = wx.StaticText(id=wxID_FRAME1X3, label='',
name='x3', parent=self, pos=wx.Point(256, 32), size=wx.Size(24,
24),style=0)
        self.x3.Show(False)

        self.x4 = wx.StaticText(id=wxID_FRAME1X4, label='',
name='x4', parent=self, pos=wx.Point(368, 32), size=wx.Size(24,
24),style=0)
        self.x4.Show(False)

        self.x5 = wx.StaticText(id=wxID_FRAME1X5, label='',
name='x5', parent=self, pos=wx.Point(472, 24), size=wx.Size(24,
32),
                style=0)
        self.x5.Show(False)
```

```
        self.x6 = wx.StaticText(id=wxID_FRAME1X6, label='',
name='x6', parent=self, pos=wx.Point(24, 120), size=wx.Size(24,
35), style=0)
        self.x6.Show(False)

        self.x7 = wx.StaticText(id=wxID_FRAME1X7, label='',
name='x7', parent=self, pos=wx.Point(136, 128), size=wx.Size(32,
32), style=0)
        self.x7.Show(False)

        self.x8 = wx.StaticText(id=wxID_FRAME1X8, label='',
name='x8', parent=self, pos=wx.Point(256, 128), size=wx.Size(32,
32), style=0)
        self.x8.Show(False)

        self.x9 = wx.StaticText(id=wxID_FRAME1X9, label='',
name='x9', parent=self, pos=wx.Point(368, 128), size=wx.Size(32,
32), style=0)
        self.x9.Show(False)

        self.x10 = wx.StaticText(id=wxID_FRAME1X10, label='',
name='x10', parent=self, pos=wx.Point(480, 128), size=wx.Size(32,
32), style=0)
        self.x10.Show(False)

        self.x11 = wx.StaticText(id=wxID_FRAME1X11, label='',
name='x11', parent=self, pos=wx.Point(24, 216), size=wx.Size(32,
33), style=0)
        self.x11.Show(False)

        self.x12 = wx.StaticText(id=wxID_FRAME1X12, label='',
name='x12', parent=self, pos=wx.Point(136, 216), size=wx.Size(32,
32), style=0)
        self.x12.Show(False)

        self.x13 = wx.StaticText(id=wxID_FRAME1X13, label='',
name='x13', parent=self, pos=wx.Point(248, 216), size=wx.Size(32,
32), style=0)
        self.x13.Show(False)

        self.x14 = wx.StaticText(id=wxID_FRAME1X14, label='',
name='x14', parent=self, pos=wx.Point(360, 216), size=wx.Size(32,
24), style=0)
        self.x14.Show(False)

        self.x15 = wx.StaticText(id=wxID_FRAME1X15, label='',
name='x15', parent=self, pos=wx.Point(480, 216), size=wx.Size(32,
32), style=0)
        self.x15.Show(False)

        self.x16 = wx.StaticText(id=wxID_FRAME1X16, label='',
name='x16', parent=self, pos=wx.Point(24, 312), size=wx.Size(32,
32), style=0)
        self.x16.Show(False)
```

```
        self.x17 = wx.StaticText(id=wxID_FRAME1X17, label='',
name='x17', parent=self, pos=wx.Point(144, 312), size=wx.Size(24,
32), style=0)
        self.x17.Show(False)

        self.x18 = wx.StaticText(id=wxID_FRAME1X18, label='',
name='x18', parent=self, pos=wx.Point(256, 312), size=wx.Size(24,
32), style=0)
        self.x18.Show(False)

        self.staticText2 =
wx.StaticText(id=wxID_FRAME1STATICTEXT2, label='Trabajo Especial
de Grado - UCAB 2007
Marieugenia Fern\xe1ndez - Vincenzo Saturno',
              name='staticText2', parent=self, pos=wx.Point(48,
408), size=wx.Size(539, 12), style=0)
        self.staticText2.SetFont(wx.Font(7, wx.SWISS, wx.NORMAL,
wx.NORMAL, False, 'Newspaper'))
        self.staticText2.SetForegroundColour(wx.Colour(255, 255,
255))

        self.trampa = wx.TextCtrl(id=wxID_FRAME1TRAMPA,
name='trampa', parent=self, pos=wx.Point(288, 408),
size=wx.Size(40, 16), style=0, value='')
        self.trampa.Show(False)

    def __init__(self, parent):
        self._init_ctrls(parent)

    def OnButton2Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox2.GetValue()      #toma el valor de check
para luego entrar en la cabina o sino al message dialog
        for line in ru:              #recorre todo el Master.csv
            p=line.split(",")[1]        #particiona para comprobar
el usuario
            if (p=='"usuario2"' and c==True):   #comprobar el
usuario y si ademas hay llamadas en la cabina
                a=2
        ru.close()
        if (a==2):          #entra si hay llamadas con el usuario2
            frame=Frame2.create(self)   #crea la frame2 en cabina2
            frame.Show(True)                     #abre la frame2
            frame.cabina.SetLabel('Cabina 2')
            h=self.x2.GetLabel()                 #toma la hora en
que fue checked la cabina desde el statictext oculto
            frame.prueba.SetLabel(h)             #coloca la hora en
un statictext oculto del frame2
        else:                #entra si no esta checked la checkbox
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 2',
```

```
                'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton1Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox1.GetValue()
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario1"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 1')
            h=self.x1.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 1',
                'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton3Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox3.GetValue()
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario3"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 3')
            h=self.x3.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 3',
                'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
```

```
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton4Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox4.GetValue()
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario4"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 4')
            h=self.x4.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 4',
              'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton5Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox5.GetValue()
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario5"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 5')
            h=self.x5.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 5',
              'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
```

```
            dlg.Destroy()

    def OnButton6Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox6.GetValue()
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario6"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 6')
            h=self.x6.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 6',
              'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton7Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox7.GetValue()
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario7"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 7')
            h=self.x7.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 7',
              'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()
```

```
    def OnButton8Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox8.GetValue()
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario8"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 8')
            h=self.x8.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 8',
              'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton9Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox9.GetValue()
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario9"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 9')
            h=self.x9.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 9',
              'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton10Button(self, event):
        event.Skip()
```

```
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox10.GetValue()
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario10"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 10')
            h=self.x10.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 10',
              'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton11Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox11.GetValue()
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario11"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 11')
            h=self.x11.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 11',
              'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton12Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
```

```
        a=1
        c=self.checkBox12.GetValue()
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario12"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 12')
            h=self.x12.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 12',
              'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton13Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox13.GetValue()
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario13"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 13')
            h=self.x13.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 13',
              'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton14Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox14.GetValue()
```

```
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario14"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 14')
            h=self.x14.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 14',
              'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton15Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox15.GetValue()
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario15"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 15')
            h=self.x15.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 15',
              'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton16Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox16.GetValue()
        for line in ru:
            p=line.split(",")[1]
```

```
            if (p=='"usuario16"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 16')
            h=self.x16.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 16',
              'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton17Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox17.GetValue()
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario17"' and c==True):
                a=2
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 17')
            h=self.x17.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 17',
              'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton18Button(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')
        a=1
        c=self.checkBox18.GetValue()
        for line in ru:
            p=line.split(",")[1]
            if (p=='"usuario18"' and c==True):
                a=2
```

```
        ru.close()
        if (a==2):
            frame=Frame2.create(self)
            frame.Show(True)
            frame.cabina.SetLabel('Cabina 18')
            h=self.x18.GetLabel()
            frame.prueba.SetLabel(h)
        else:
            dlg = wx.MessageDialog(self, 'No hay llamadas en la
Cabina 18',
              'Caption', wx.OK | wx.ICON_INFORMATION)
            try:
                dlg.ShowModal()
            finally:
                dlg.Destroy()

    def OnButton19Button(self, event):
        event.Skip()
        self.Hide()
        pid=os.getpid()
        os.popen("kill -9 "+str(pid))
        pid=os.getpid()

    # ........... CHECKBOX ........... #
    #para validar las llamadas desde la hora en que se hace check
en la checkbox

    def OnCheckBox1Checkbox(self, event):
        event.Skip()
        c=self.checkBox1.GetValue()     # agarra el valor del
checkbox
        if (c==True):
            hora=time.localtime(time.time())    #si esta checked
toma la hora en ese instante
            h=time.strftime('"%Y-%m-%d %X"',hora)   #se convierte
al formato del asterisk
            self.x1.SetLabel(h)                 #se coloca la hora
en un statictext que no se muestra
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
                ru3.write(' borrar ')
                ru3.close()
                MyThread().start()

    def OnCheckBox3Checkbox(self, event):
        event.Skip()
        c=self.checkBox3.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x3.SetLabel(h)
            self.trampa.AppendText('x')
```

```
        i=self.trampa.GetValue()
        if (i=='x'):
            ru3=open('/cdc','w')
            ru3.write(' borrar ')
            ru3.close()
            MyThread().start()

    def OnCheckBox4Checkbox(self, event):
        event.Skip()
        c=self.checkBox4.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x4.SetLabel(h)
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
                ru3.write(' borrar ')
                ru3.close()
                MyThread().start()

    def OnCheckBox5Checkbox(self, event):
        event.Skip()
        c=self.checkBox5.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x5.SetLabel(h)
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
                ru3.write(' borrar ')
                ru3.close()
                MyThread().start()

    def OnCheckBox10Checkbox(self, event):
        event.Skip()
        c=self.checkBox10.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x10.SetLabel(h)
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
                ru3.write(' borrar ')
                ru3.close()
                MyThread().start()

    def OnCheckBox15Checkbox(self, event):
        event.Skip()
```

```
        c=self.checkBox15.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x15.SetLabel(h)
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
                ru3.write(' borrar ')
                ru3.close()
                MyThread().start()

    def OnCheckBox2Checkbox(self, event):
        event.Skip()
        c=self.checkBox2.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x2.SetLabel(h)
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
                ru3.write(' borrar ')
                ru3.close()
                MyThread().start()

    def OnCheckBox6Checkbox(self, event):
        event.Skip()
        c=self.checkBox6.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x6.SetLabel(h)
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
                ru3.write(' borrar ')
                ru3.close()
                MyThread().start()

    def OnCheckBox7Checkbox(self, event):
        event.Skip()
        c=self.checkBox7.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x7.SetLabel(h)
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
```

```
            ru3.write(' borrar ')
            ru3.close()
            MyThread().start()

    def OnCheckBox8Checkbox(self, event):
        event.Skip()
        c=self.checkBox8.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x8.SetLabel(h)
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
                ru3.write(' borrar ')
                ru3.close()
                MyThread().start()

    def OnCheckBox9Checkbox(self, event):
        event.Skip()
        c=self.checkBox9.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x9.SetLabel(h)
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
                ru3.write(' borrar ')
                ru3.close()
                MyThread().start()

    def OnCheckBox11Checkbox(self, event):
        event.Skip()
        c=self.checkBox11.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x11.SetLabel(h)
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
                ru3.write(' borrar ')
                ru3.close()
                MyThread().start()

    def OnCheckBox12Checkbox(self, event):
        event.Skip()
        c=self.checkBox12.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
```

```
        h=time.strftime('"%Y-%m-%d %X"',hora)
        self.x12.SetLabel(h)
        self.trampa.AppendText('x')
        i=self.trampa.GetValue()
        if (i=='x'):
            ru3=open('/cdc','w')
            ru3.write(' borrar ')
            ru3.close()
            MyThread().start()

    def OnCheckBox13Checkbox(self, event):
        event.Skip()
        c=self.checkBox13.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x13.SetLabel(h)
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
                ru3.write(' borrar ')
                ru3.close()
                MyThread().start()

    def OnCheckBox14Checkbox(self, event):
        event.Skip()
        c=self.checkBox14.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x14.SetLabel(h)
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
                ru3.write(' borrar ')
                ru3.close()
                MyThread().start()

    def OnCheckBox16Checkbox(self, event):
        event.Skip()
        c=self.checkBox16.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x16.SetLabel(h)
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
                ru3.write(' borrar ')
                ru3.close()
                MyThread().start()
```

```
    def OnCheckBox17Checkbox(self, event):
        event.Skip()
        c=self.checkBox17.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x17.SetLabel(h)
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
                ru3.write(' borrar ')
                ru3.close()
                MyThread().start()

    def OnCheckBox18Checkbox(self, event):
        event.Skip()
        c=self.checkBox18.GetValue()
        if (c==True):
            hora=time.localtime(time.time())
            h=time.strftime('"%Y-%m-%d %X"',hora)
            self.x18.SetLabel(h)
            self.trampa.AppendText('x')
            i=self.trampa.GetValue()
            if (i=='x'):
                ru3=open('/cdc','w')
                ru3.write(' borrar ')
                ru3.close()
                MyThread().start()
```

M.3. Frame2.py

```
#Boa:Frame:Frame2

import wx
import wx.stc
import Frame1
import Frame3
import wx.grid
import wx.gizmos
import time
import binascii
from wx.lib.anchors import LayoutAnchors

def create(parent):
    return Frame2(parent)

[wxID_FRAME2, wxID_FRAME2CABINA, wxID_FRAME2COSTO,
wxID_FRAME2FECHAHORA,
 wxID_FRAME2IMPRIMIR, wxID_FRAME2LIMPIAR, wxID_FRAME2LISTA2,
 wxID_FRAME2PRECIO, wxID_FRAME2PRUEBA, wxID_FRAME2SALIR,
 wxID_FRAME2STATICTEXT1, wxID_FRAME2TIEMPO, wxID_FRAME2TOTAL,
] = [wx.NewId() for _init_ctrls in range(13)]
```

```
class Frame2(wx.Frame):
    def _init_ctrls(self, prnt):
        # generated method, don't edit
        wx.Frame.__init__(self, id=wxID_FRAME2, name='',
parent=prnt, pos=wx.Point(174, 121), size=wx.Size(604, 415),
style=wx.DEFAULT_FRAME_STYLE, title='Frame2')
        self.SetClientSize(wx.Size(604, 415))
        self.SetForegroundColour(wx.Colour(255, 88, 77))
        self.SetBackgroundColour(wx.Colour(0, 0, 0))

        self.cabina = wx.StaticText(id=wxID_FRAME2CABINA,
label='Cabina', name='cabina', parent=self, pos=wx.Point(254, 0),
size=wx.Size(96, 25), style=0)
        self.cabina.Center(wx.HORIZONTAL)
        self.cabina.SetFont(wx.Font(14, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Ikarus'))
        self.cabina.SetForegroundColour(wx.Colour(255, 255, 255))
        self.cabina.SetBackgroundColour(wx.Colour(255, 255, 255))

        self.total = wx.StaticText(id=wxID_FRAME2TOTAL,
label='Total Bs.', name='total', parent=self, pos=wx.Point(144,
304), size=wx.Size(104, 26), style=0)
        self.total.SetFont(wx.Font(17, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Sans'))
        self.total.SetForegroundColour(wx.Colour(255, 255, 255))

        self.salir = wx.Button(id=wxID_FRAME2SALIR, label='Salir',
name='salir', parent=self, pos=wx.Point(408, 344),
size=wx.Size(120, 48), style=0)
        self.salir.SetBackgroundColour(wx.Colour(255, 255, 255))
        self.salir.SetFont(wx.Font(10, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Sans'))
        self.salir.Bind(wx.EVT_BUTTON, self.OnSalirButton,
id=wxID_FRAME2SALIR)

        self.limpiar = wx.Button(id=wxID_FRAME2LIMPIAR,
label='Limpiar', name='limpiar', parent=self, pos=wx.Point(240,
344), size=wx.Size(120, 48), style=0)
        self.limpiar.SetBackgroundColour(wx.Colour(255, 255, 255))
        self.limpiar.SetFont(wx.Font(10, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Sans'))
        self.limpiar.Bind(wx.EVT_BUTTON, self.OnButton2Button,
id=wxID_FRAME2LIMPIAR)

        self.imprimir = wx.Button(id=wxID_FRAME2IMPRIMIR,
label='Mostrar', name='imprimir', parent=self, pos=wx.Point(80,
344), size=wx.Size(112, 48), style=0)
        self.imprimir.SetBackgroundColour(wx.Colour(255, 255,
255))
        self.imprimir.SetForegroundColour(wx.Colour(0, 0, 0))
        self.imprimir.SetFont(wx.Font(10, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Sans'))
```

```
        self.imprimir.Bind(wx.EVT_BUTTON, self.OnImprimirButton,
id=wxID_FRAME2IMPRIMIR)

        self.fechahora = wx.StaticText(id=wxID_FRAME2FECHAHORA,
label='Fecha - Hora', name='fechahora', parent=self,
pos=wx.Point(64, 48), size=wx.Size(101, 24), style=0)
        self.fechahora.SetForegroundColour(wx.Colour(255, 255,
255))
        self.fechahora.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Sans'))

        self.tiempo = wx.StaticText(id=wxID_FRAME2TIEMPO,
            label='Duraci\xf3n (seg)', name='tiempo',
parent=self, pos=wx.Point(352, 48), size=wx.Size(117, 24),
style=0)
        self.tiempo.SetForegroundColour(wx.Colour(255, 255, 255))
        self.tiempo.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Sans'))

        self.prueba = wx.StaticText(id=wxID_FRAME2PRUEBA,
label='', name='prueba', parent=self, pos=wx.Point(453, 267),
size=wx.Size(27, 21), style=0)
        self.prueba.Show(False)
        self.prueba.SetMinSize(wx.Size(0, 0))

        self.lista2 = wx.TextCtrl(id=wxID_FRAME2LISTA2,
name=u'lista2', parent=self, pos=wx.Point(24, 80),
size=wx.Size(552, 192), style=wx.STATIC_BORDER | wx.VSCROLL |
wx.TE_READONLY | wx.TE_MULTILINE, value='')
        self.lista2.SetBackgroundColour(wx.Colour(255, 255, 255))
        self.lista2.Show(True)
        self.lista2.SetThemeEnabled(False)

        self.staticText1 =
wx.StaticText(id=wxID_FRAME2STATICTEXT1, label='Destino',
name='staticText1', parent=self, pos=wx.Point(232, 48),
size=wx.Size(88, 24), style=0)
        self.staticText1.SetForegroundColour(wx.Colour(255, 255,
255))
        self.staticText1.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Sans'))

        self.precio = wx.TextCtrl(id=wxID_FRAME2PRECIO,
name='precio', parent=self, pos=wx.Point(264, 288),
size=wx.Size(144, 40), style=wx.TE_READONLY | wx.TE_CENTER,
value=u'')
        self.precio.SetFont(wx.Font(14, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Sans'))

        self.costo = wx.StaticText(id=wxID_FRAME2COSTO,
label='Costo (Bs)', name='costo', parent=self, pos=wx.Point(496,
48), size=wx.Size(82, 24), style=0)
        self.costo.SetForegroundColour(wx.Colour(255, 255, 255))
```

```
        self.costo.SetFont(wx.Font(11, wx.SWISS, wx.NORMAL,
wx.BOLD, False, 'Sans'))

    def __init__(self, parent):
        self._init_ctrls(parent)

    def OnSalirButton(self, event):
        event.Skip()
        self.Hide()

    def OnButton2Button(self, event):
        event.Skip()
        self.lista2.SetValue('')
        self.precio.SetValue('')

    def OnImprimirButton(self, event):
        event.Skip()
        ruta='/var/log/asterisk/cdr-custom/'
        ru=open(ruta+'Master.csv','r')      #abre el Master
        x=self.cabina.GetLabel()            #agarra cual es la
cabina que se abre
        pago=0

        if (x=='Cabina 1'):                  #para la cabina1
            for line in ru:                 #recorre el Master.csv
                p=line.split(",")[1]
#particiona y agarra el usuario
                fhast=line.split(",")[8]
#particiona y agarra fecha y hora
                hora1=self.prueba.GetLabel()
#agarra la hora en que se chequeo la cabina desde el frame1
                self.prueba.SetLabel(hora1)                 #esto
no esta haciendo nada
                if (p=='"usuario1"' and fhast>hora1):
#comprobar que las llamadas son de ese usuario, hora asterisk
mayor que hora del check
                    e=line.split(",")[13]
#particona y toma el estado de la llamada
                    if (e=='"FAILED"'):
#mensajes con el estado de la llamada para fallida y no answer
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:      #entra si la llamada fue contestada
                        fh1=line.split(",")[9]              #toma
la fecha y hora
                        fh=fh1.split('"')[1]                #se
eliminan las comillas de los extremos
                        n1=line.split(",")[2]               #toma
el numero discado
                        n=n1.split('"')[1]
```

```
                        t=line.split(",")[12]                    #toma
el tiempo de la llamada

                        # ##### TARIFADOR #############

                        rut=open('/tarifas.txt','r')   #abro la
tabla de codigos
                        for linea in rut:
                            codigo=linea.split("|")[1]
#agarro el codigo
                            d=[]                    #creo un arreglo
                            d=line.split(",")[7]
                            dest8=d[9:-23]          #corto los 1eros 9
y los ultimos 23
                            dest7=d[9:-24]
                            dest6=d[9:-25]
                            dest5=d[9:-26]
                            dest4=d[9:-27]
                            dest3=d[9:-28]
                            dest2=d[9:-29]
                            dest1=d[9:-30]
                            if (codigo==dest8):
                                valor=linea.split("|")[2]
#agarro el costo
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))   #para facturar en segundos
                                c=str(c1)
                            elif (codigo==dest7):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest6):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest5):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest4):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest3):
                                valor=linea.split("|")[2]
```

```
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest2):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest1):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)

                        self.lista2.AppendText(fh+'
'+n+'                '+t1+'                    '+c+'  \n')
                        pago=int(round(pago+c1))
#redondeo el resultado
                        self.precio.SetValue(str(pago))

            self.lista2.AppendText('No hay mas llamadas')

        elif (x=='Cabina 2'):
            for line in ru:
                p=line.split(",")[1]
                fhast=line.split(",")[8]
                hora1=self.prueba.GetLabel()
                if (p=='"usuario2"' and fhast>hora1):
                    e=line.split(",")[13]
                    if (e=='"FAILED"'):
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:
                        fh1=line.split(",")[9]
                        fh=fh1.split('"')[1]
                        n1=line.split(",")[2]
                        n=n1.split('"')[1]
                        t=line.split(",")[12]

                        # ########## TARIFADOR ###########

                        rut=open('/tarifas.txt','r')        #abro
la tabla de tarifas
                        for linea in rut:
                            codigo=linea.split("|")[1]
#agarro el codigo
```

```
                                d=[]
#creo un arreglo
                                d=line.split(",")[7]
                                dest8=d[9:-23]
#corto los leros 9 y los ultimos 23
                                dest7=d[9:-24]
                                dest6=d[9:-25]
                                dest5=d[9:-26]
                                dest4=d[9:-27]
                                dest3=d[9:-28]
                                dest2=d[9:-29]
                                dest1=d[9:-30]
                                if (codigo==dest8):
                                    valor=linea.split("|")[2]
#agarro el costo
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))   #pa facturar en segundos
                                    c=str(c1)
                                elif (codigo==dest7):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest6):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest5):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest4):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest3):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest2):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
```

```
                                    c=str(c1)
                                elif (codigo==dest1):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)

                                self.lista2.AppendText(fh+'
'+n+'                '+t1+'                                '+c+'  \n')
                                pago=int(pago+c1)
                                self.precio.SetValue(str(pago))

                        self.lista2.AppendText('No hay mas llamadas')

                elif (x=='Cabina 3'):
                    for line in ru:
                        p=line.split(",")[1]
                        fhast=line.split(",")[8]
                        hora1=self.prueba.GetLabel()
                        if (p=='"usuario3"' and fhast>hora1):
                            e=line.split(",")[13]
                            if (e=='"FAILED"'):
                                self.lista2.AppendText('Llamada Fallida
\n')

                            elif (e=='"NO ANSWER"'):
                                self.lista2.AppendText('No Contesta \n')
                            elif (e=='"BUSY"'):
                                self.lista2.AppendText('Ocupado \n')
                            else:
                                fh1=line.split(",")[9]
                                fh=fh1.split('"')[1]
                                n1=line.split(",")[2]
                                n=n1.split('"')[1]
                                t=line.split(",")[12]

                                # ########## TARIFADOR ###########

                                rut=open('/tarifas.txt','r')
                                for linea in rut:
                                    codigo=linea.split("|")[1]
                                    d=[]
                                    d=line.split(",")[7]
                                    dest8=d[9:-23]
                                    dest7=d[9:-24]
                                    dest6=d[9:-25]
                                    dest5=d[9:-26]
                                    dest4=d[9:-27]
                                    dest3=d[9:-28]
                                    dest2=d[9:-29]
                                    dest1=d[9:-30]
                                    if (codigo==dest8):
                                        valor=linea.split("|")[2]
                                        t1=t.split('"')[1]
```

```
c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest7):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest6):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest5):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest4):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest3):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest2):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest1):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)

                        self.lista2.AppendText(fh+'
'+n+'                '+t1+'                    '+c+'  \n')
                        pago=int(round(pago+c1))
                        self.precio.SetValue(str(pago))

            self.lista2.AppendText('No hay mas llamadas')

        elif (x=='Cabina 4'):
```

```
            for line in ru:
                p=line.split(",")[1]
                fhast=line.split(",")[8]
                hora1=self.prueba.GetLabel()
                if (p=='"usuario4"' and fhast>hora1):
                    e=line.split(",")[13]
                    if (e=='"FAILED"'):
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:
                        fh1=line.split(",")[9]
                        fh=fh1.split('"')[1]
                        n1=line.split(",")[2]
                        n=n1.split('"')[1]
                        t=line.split(",")[12]

                        # ########## TARIFADOR ###########

                        rut=open('/tarifas.txt','r')
                        for linea in rut:
                            codigo=linea.split("|")[1]
                            d=[]
                            d=line.split(",")[7]
                            dest8=d[9:-23]
                            dest7=d[9:-24]
                            dest6=d[9:-25]
                            dest5=d[9:-26]
                            dest4=d[9:-27]
                            dest3=d[9:-28]
                            dest2=d[9:-29]
                            dest1=d[9:-30]
                            if (codigo==dest8):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest7):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest6):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest5):
```

```
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

cl=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest4):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

cl=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest3):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

cl=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest2):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

cl=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest1):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

cl=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)

                        self.lista2.AppendText(fh+'
'+n+'                '+t1+'                '+c+'  \n')
                        pago=int(round(pago+c1))
                        self.precio.SetValue(str(pago))

            self.lista2.AppendText('No hay mas llamadas')

        elif (x=='Cabina 5'):
            for line in ru:
                p=line.split(",")[1]
                fhast=line.split(",")[8]
                hora1=self.prueba.GetLabel()
                if (p=='"usuario5"' and fhast>hora1):
                    e=line.split(",")[13]
                    if (e=='"FAILED"'):
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:
                        fh1=line.split(",")[9]
                        fh=fh1.split('"')[1]
```

```
                    n1=line.split(",")[2]
                    n=n1.split('"')[1]
                    t=line.split(",")[12]

                    # ##### TARIFADOR #############

                    rut=open('/tarifas.txt','r')
                    for linea in rut:
                        codigo=linea.split("|")[1]
                        d=[]
                        d=line.split(",")[7]
                        dest8=d[9:-23]
                        dest7=d[9:-24]
                        dest6=d[9:-25]
                        dest5=d[9:-26]
                        dest4=d[9:-27]
                        dest3=d[9:-28]
                        dest2=d[9:-29]
                        dest1=d[9:-30]
                        if (codigo==dest8):
                            valor=linea.split("|")[2]
                            t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                            c=str(c1)
                        elif (codigo==dest7):
                            valor=linea.split("|")[2]
                            t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                            c=str(c1)
                        elif (codigo==dest6):
                            valor=linea.split("|")[2]
                            t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                            c=str(c1)
                        elif (codigo==dest5):
                            valor=linea.split("|")[2]
                            t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                            c=str(c1)
                        elif (codigo==dest4):
                            valor=linea.split("|")[2]
                            t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                            c=str(c1)
                        elif (codigo==dest3):
                            valor=linea.split("|")[2]
                            t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
```

```
                                c=str(c1)
                            elif (codigo==dest2):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest1):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)

                        self.lista2.AppendText(fh+'
'+n+'                '+t1+'                    '+c+'  \n')
                        pago=int(round(pago+c1))
                        self.precio.SetValue(str(pago))

                self.lista2.AppendText('No hay mas llamadas')

        elif (x=='Cabina 6'):
            for line in ru:
                p=line.split(",")[1]
                fhast=line.split(",")[8]
                hora1=self.prueba.GetLabel()
                if (p=='"usuario6"' and fhast>hora1):
                    e=line.split(",")[13]
                    if (e=='"FAILED"'):
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:
                        fh1=line.split(",")[9]
                        fh=fh1.split('"')[1]
                        n1=line.split(",")[2]
                        n=n1.split('"')[1]
                        t=line.split(",")[12]

                        # ##### TARIFADOR #############

                        rut=open('/tarifas.txt','r')
                        for linea in rut:
                            codigo=linea.split("|")[1]
                            d=[]
                            d=line.split(",")[7]
                            dest8=d[9:-23]
                            dest7=d[9:-24]
                            dest6=d[9:-25]
                            dest5=d[9:-26]
                            dest4=d[9:-27]
```

```
                                dest3=d[9:-28]
                                dest2=d[9:-29]
                                dest1=d[9:-30]
                                if (codigo==dest8):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest7):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest6):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest5):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest4):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest3):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest2):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest1):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)

                            self.lista2.AppendText(fh+'
'+n+'                '+t1+'                        '+c+'   \n')
```

```
                        pago=int(round(pago+c1))
                        self.precio.SetValue(str(pago))

                self.lista2.AppendText('No hay mas llamadas')

        elif (x=='Cabina 7'):
            for line in ru:
                p=line.split(",")[1]
                fhast=line.split(",")[8]
                hora1=self.prueba.GetLabel()
                if (p=='"usuario7"' and fhast>hora1):
                    e=line.split(",")[13]
                    if (e=='"FAILED"'):
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:
                        fh1=line.split(",")[9]
                        fh=fh1.split('"')[1]
                        n1=line.split(",")[2]
                        n=n1.split('"')[1]
                        t=line.split(",")[12]

                        # ########## TARIFADOR ###########

                        rut=open('/tarifas.txt','r')
                        for linea in rut:
                            codigo=linea.split("|")[1]
                            d=[]
                            d=line.split(",")[7]
                            dest8=d[9:-23]
                            dest7=d[9:-24]
                            dest6=d[9:-25]
                            dest5=d[9:-26]
                            dest4=d[9:-27]
                            dest3=d[9:-28]
                            dest2=d[9:-29]
                            dest1=d[9:-30]
                            if (codigo==dest8):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest7):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest6):
```

```
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest5):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest4):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest3):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest2):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest1):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)

                        self.lista2.AppendText(fh+'
'+n+'                  '+t1+'                    '+c+'  \n')
                        pago=int(round(pago+c1))
                        self.precio.SetValue(str(pago))

            self.lista2.AppendText('No hay mas llamadas')

        elif (x=='Cabina 8'):
            for line in ru:
                p=line.split(",")[1]
                fhast=line.split(",")[8]
                hora1=self.prueba.GetLabel()
                if (p=='"usuario8"' and fhast>hora1):
                    e=line.split(",")[13]
                    if (e=='"FAILED"'):
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
```

```
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:
                        fh1=line.split(",")[9]
                        fh=fh1.split('"')[1]
                        n1=line.split(",")[2]
                        n=n1.split('"')[1]
                        t=line.split(",")[12]

                        # ########## TARIFADOR ###########

                        rut=open('/tarifas.txt','r')
                        for linea in rut:
                            codigo=linea.split("|")[1]
                            d=[]
                            d=line.split(",")[7]
                            dest8=d[9:-23]
                            dest7=d[9:-24]
                            dest6=d[9:-25]
                            dest5=d[9:-26]
                            dest4=d[9:-27]
                            dest3=d[9:-28]
                            dest2=d[9:-29]
                            dest1=d[9:-30]
                            if (codigo==dest8):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest7):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest6):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest5):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest4):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
```

```
                                c=str(c1)
                            elif (codigo==dest3):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest2):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest1):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)

                        self.lista2.AppendText(fh+'
'+n+'                '+t1+'                    '+c+'  \n')
                        pago=int(round(pago+c1))
                        self.precio.SetValue(str(pago))

            self.lista2.AppendText('No hay mas llamadas')

        elif (x=='Cabina 9'):
            for line in ru:
                p=line.split(",")[1]
                fhast=line.split(",")[8]
                hora1=self.prueba.GetLabel()
                if (p=='"usuario9"' and fhast>hora1):
                    e=line.split(",")[13]
                    if (e=='"FAILED"'):
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:
                        fh1=line.split(",")[9]
                        fh=fh1.split('"')[1]
                        n1=line.split(",")[2]
                        n=n1.split('"')[1]
                        t=line.split(",")[12]

                        # ##### TARIFADOR #############

                        rut=open('/tarifas.txt','r')
                        for linea in rut:
                            codigo=linea.split("|")[1]
                            d=[]
```

```
                                d=line.split(",")[7]
                                dest8=d[9:-23]
                                dest7=d[9:-24]
                                dest6=d[9:-25]
                                dest5=d[9:-26]
                                dest4=d[9:-27]
                                dest3=d[9:-28]
                                dest2=d[9:-29]
                                dest1=d[9:-30]
                                if (codigo==dest8):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest7):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest6):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest5):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest4):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest3):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest2):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest1):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]
```

```
cl=round(int(tl)*(int(valor)*.0167))
                                c=str(cl)

                        self.lista2.AppendText(fh+'
'+n+'                  '+tl+'                    '+c+'  \n')
                        pago=int(round(pago+cl))
                        self.precio.SetValue(str(pago))

            self.lista2.AppendText('No hay mas llamadas')

        elif (x=='Cabina 10'):
            for line in ru:
                p=line.split(",")[1]
                fhast=line.split(",")[8]
                hora1=self.prueba.GetLabel()
                if (p=='"usuario10"' and fhast>hora1):
                    e=line.split(",")[13]
                    if (e=='"FAILED"'):
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:
                        fh1=line.split(",")[9]
                        fh=fh1.split('"')[1]
                        n1=line.split(",")[2]
                        n=n1.split('"')[1]
                        t=line.split(",")[12]

                        # ########## TARIFADOR ###########

                        rut=open('/tarifas.txt','r')
                        for linea in rut:
                            codigo=linea.split("|")[1]
                            d=[]
                            d=line.split(",")[7]
                            dest8=d[9:-23]
                            dest7=d[9:-24]
                            dest6=d[9:-25]
                            dest5=d[9:-26]
                            dest4=d[9:-27]
                            dest3=d[9:-28]
                            dest2=d[9:-29]
                            dest1=d[9:-30]
                            if (codigo==dest8):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

cl=round(int(tl)*(int(valor)*.0167))
                                c=str(cl)
                            elif (codigo==dest7):
```

```
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest6):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest5):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest4):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest3):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest2):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest1):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)

                        self.lista2.AppendText(fh+'
'+n+'                '+t1+'                    '+c+'   \n')
                        pago=int(round(pago+c1))
                        self.precio.SetValue(str(pago))

            self.lista2.AppendText('No hay mas llamadas')

        elif (x=='Cabina 11'):
            for line in ru:
                p=line.split(",")[1]
                fhast=line.split(",")[8]
                hora1=self.prueba.GetLabel()
```

```
                if (p=='"usuario11"' and fhast>hora1):
                    e=line.split(",")[13]
                    if (e=='"FAILED"'):
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:
                        fh1=line.split(",")[9]
                        fh=fh1.split('"')[1]
                        n1=line.split(",")[2]
                        n=n1.split('"')[1]
                        t=line.split(",")[12]

                        # ##### TARIFADOR #############

                        rut=open('/tarifas.txt','r')
                        for linea in rut:
                            codigo=linea.split("|")[1]
                            d=[] #creo un arreglo
                            d=line.split(",")[7]
                            dest8=d[9:-23]
                            dest7=d[9:-24]
                            dest6=d[9:-25]
                            dest5=d[9:-26]
                            dest4=d[9:-27]
                            dest3=d[9:-28]
                            dest2=d[9:-29]
                            dest1=d[9:-30]
                            if (codigo==dest8):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest7):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest6):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest5):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
```

```
                                c=str(c1)
                            elif (codigo==dest4):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest3):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest2):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest1):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)

                        self.lista2.AppendText(fh+'
'+n+'                '+t1+'                    '+c+'   \n')
                        pago=int(round(pago+c1))
                        self.precio.SetValue(str(pago))

            self.lista2.AppendText('No hay mas llamadas')

        elif (x=='Cabina 12'):
            for line in ru:
                p=line.split(",")[1]
                fhast=line.split(",")[8]
                hora1=self.prueba.GetLabel()
                if (p=='"usuario12"' and fhast>hora1):
                    e=line.split(",")[13]
                    if (e=='"FAILED"'):
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:
                        fh1=line.split(",")[9]
                        fh=fh1.split('"')[1]
                        n1=line.split(",")[2]
                        n=n1.split('"')[1]
                        t=line.split(",")[12]
```

```
# ##### TARIFADOR #############

rut=open('/tarifas.txt','r')
for linea in rut:
    codigo=linea.split("|")[1]
    d=[] #creo un arreglo
    d=line.split(",")[7]
    dest8=d[9:-23]
    dest7=d[9:-24]
    dest6=d[9:-25]
    dest5=d[9:-26]
    dest4=d[9:-27]
    dest3=d[9:-28]
    dest2=d[9:-29]
    dest1=d[9:-30]
    if (codigo==dest8):
        valor=linea.split("|")[2]
        t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
        c=str(c1)
    elif (codigo==dest7):
        valor=linea.split("|")[2]
        t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
        c=str(c1)
    elif (codigo==dest6):
        valor=linea.split("|")[2]
        t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
        c=str(c1)
    elif (codigo==dest5):
        valor=linea.split("|")[2]
        t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
        c=str(c1)
    elif (codigo==dest4):
        valor=linea.split("|")[2]
        t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
        c=str(c1)
    elif (codigo==dest3):
        valor=linea.split("|")[2]
        t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
        c=str(c1)
    elif (codigo==dest2):
        valor=linea.split("|")[2]
        t1=t.split('"')[1]
```

```
c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest1):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)

                            self.lista2.AppendText(fh+'
'+n+'                   '+t1+'                    '+c+'   \n')
                            pago=int(round(pago+c1))
                            self.precio.SetValue(str(pago))

                self.lista2.AppendText('No hay mas llamadas')

            elif (x=='Cabina 13'):
                for line in ru:
                    p=line.split(",")[1]
                    fhast=line.split(",")[8]
                    hora1=self.prueba.GetLabel()
                    if (p=='"usuario13"' and fhast>hora1):
                        e=line.split(",")[13]
                        if (e=='"FAILED"'):
                            self.lista2.AppendText('Llamada Fallida
\n')
                        elif (e=='"NO ANSWER"'):
                            self.lista2.AppendText('No Contesta \n')
                        elif (e=='"BUSY"'):
                            self.lista2.AppendText('Ocupado \n')
                        else:
                            fh1=line.split(",")[9]
                            fh=fh1.split('"')[1]
                            n1=line.split(",")[2]
                            n=n1.split('"')[1]
                            t=line.split(",")[12]

                            # ##### TARIFADOR #############

                            rut=open('/tarifas.txt','r')
                            for linea in rut:
                                codigo=linea.split("|")[1]
                                d=[]
                                d=line.split(",")[7]
                                dest8=d[9:-23]
                                dest7=d[9:-24]
                                dest6=d[9:-25]
                                dest5=d[9:-26]
                                dest4=d[9:-27]
                                dest3=d[9:-28]
                                dest2=d[9:-29]
                                dest1=d[9:-30]
                                if (codigo==dest8):
```

```
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest7):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest6):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest5):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest4):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest3):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest2):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest1):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)

                            self.lista2.AppendText(fh+'
'+n+'                    '+t1+'                    '+c+'   \n')
                            pago=int(round(pago+c1))
                            self.precio.SetValue(str(pago))

                self.lista2.AppendText('No hay mas llamadas')
```

```
        elif (x=='Cabina 14'):
            for line in ru:
                p=line.split(",")[1]
                fhast=line.split(",")[8]
                hora1=self.prueba.GetLabel()
                if (p=='"usuario14"' and fhast>hora1):
                    e=line.split(",")[13]
                    if (e=='"FAILED"'):
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:
                        fh1=line.split(",")[9]
                        fh=fh1.split('"')[1]
                        n1=line.split(",")[2]
                        n=n1.split('"')[1]
                        t=line.split(",")[12]

                        # ##### TARIFADOR #############

                        rut=open('/tarifas.txt','r')
                        for linea in rut:
                            codigo=linea.split("|")[1]
                            d=[]
                            d=line.split(",")[7]
                            dest8=d[9:-23]
                            dest7=d[9:-24]
                            dest6=d[9:-25]
                            dest5=d[9:-26]
                            dest4=d[9:-27]
                            dest3=d[9:-28]
                            dest2=d[9:-29]
                            dest1=d[9:-30]
                            if (codigo==dest8):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest7):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest6):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
```

```
                                c=str(c1)
                            elif (codigo==dest5):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest4):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest3):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest2):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest1):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)

                        self.lista2.AppendText(fh+'
'+n+'                '+t1+'                        '+c+'   \n')
                        pago=int(round(pago+c1))
                        self.precio.SetValue(str(pago))

            self.lista2.AppendText('No hay mas llamadas')

        elif (x=='Cabina 15'):
            for line in ru:
                p=line.split(",")[1]
                fhast=line.split(",")[8]
                hora1=self.prueba.GetLabel()
                if (p=='"usuario15"' and fhast>hora1):
                    e=line.split(",")[13]
                    if (e=='"FAILED"'):
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:
```

```
                    fh1=line.split(",")[9]
                    fh=fh1.split('"')[1]
                    n1=line.split(",")[2]
                    n=n1.split('"')[1]
                    t=line.split(",")[12]

                    # ##### TARIFADOR #############

                    rut=open('/tarifas.txt','r')
                    for linea in rut:
                        codigo=linea.split("|")[1]
                        d=[]
                        d=line.split(",")[7]
                        dest8=d[9:-23]
                        dest7=d[9:-24]
                        dest6=d[9:-25]
                        dest5=d[9:-26]
                        dest4=d[9:-27]
                        dest3=d[9:-28]
                        dest2=d[9:-29]
                        dest1=d[9:-30]
                        if (codigo==dest8):
                            valor=linea.split("|")[2]
                            t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                            c=str(c1)
                        elif (codigo==dest7):
                            valor=linea.split("|")[2]
                            t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                            c=str(c1)
                        elif (codigo==dest6):
                            valor=linea.split("|")[2]
                            t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                            c=str(c1)
                        elif (codigo==dest5):
                            valor=linea.split("|")[2]
                            t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                            c=str(c1)
                        elif (codigo==dest4):
                            valor=linea.split("|")[2]
                            t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                            c=str(c1)
                        elif (codigo==dest3):
                            valor=linea.split("|")[2]
                            t1=t.split('"')[1]
```

```
c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest2):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest1):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)

                        self.lista2.AppendText(fh+'
'+n+'                '+t1+'                    '+c+'  \n')
                        pago=int(round(pago+c1))
                        self.precio.SetValue(str(pago))

            self.lista2.AppendText('No hay mas llamadas')

        elif (x=='Cabina 16'):
            for line in ru:
                p=line.split(",")[1]
                fhast=line.split(",")[8]
                hora1=self.prueba.GetLabel()
                if (p=='"usuario16"' and fhast>hora1):
                    e=line.split(",")[13]
                    if (e=='"FAILED"'):
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:
                        fh1=line.split(",")[9]
                        fh=fh1.split('"')[1]
                        n1=line.split(",")[2]
                        n=n1.split('"')[1]
                        t=line.split(",")[12]

                        # ##### TARIFADOR #############

                        rut=open('/tarifas.txt','r')
                        for linea in rut:
                            codigo=linea.split("|")[1]
                            d=[]
                            d=line.split(",")[7]
                            dest8=d[9:-23]
                            dest7=d[9:-24]
                            dest6=d[9:-25]
```

```
                                dest5=d[9:-26]
                                dest4=d[9:-27]
                                dest3=d[9:-28]
                                dest2=d[9:-29]
                                dest1=d[9:-30]
                                if (codigo==dest8):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest7):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest6):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest5):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest4):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest3):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest2):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
                                elif (codigo==dest1):
                                    valor=linea.split("|")[2]
                                    t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                    c=str(c1)
```

```
                        self.lista2.AppendText(fh+'
'+n+'                '+t1+'                '+c+'  \n')
                        pago=int(round(pago+c1))
                        self.precio.SetValue(str(pago))

            self.lista2.AppendText('No hay mas llamadas')

        elif (x=='Cabina 17'):
            for line in ru:
                p=line.split(",")[1]
                fhast=line.split(",")[8]
                hora1=self.prueba.GetLabel()
                if (p=='"usuario17"' and fhast>hora1):
                    e=line.split(",")[13]
                    if (e=='"FAILED"'):
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:
                        fh1=line.split(",")[9]
                        fh=fh1.split('"')[1]
                        n1=line.split(",")[2]
                        n=n1.split('"')[1]
                        t=line.split(",")[12]

                        # ##### TARIFADOR #############

                        rut=open('/tarifas.txt','r')
                        for linea in rut:
                            codigo=linea.split("|")[1]
                            d=[]
                            d=line.split(",")[7]
                            dest8=d[9:-23]
                            dest7=d[9:-24]
                            dest6=d[9:-25]
                            dest5=d[9:-26]
                            dest4=d[9:-27]
                            dest3=d[9:-28]
                            dest2=d[9:-29]
                            dest1=d[9:-30]
                            if (codigo==dest8):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest7):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
```

```
                                c=str(c1)
                            elif (codigo==dest6):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest5):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest4):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest3):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest2):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest1):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)

                        self.lista2.AppendText(fh+'
'+n+'                '+t1+'                    '+c+'  \n')
                        pago=int(round(pago+c1))
                        self.precio.SetValue(str(pago))

            self.lista2.AppendText('No hay mas llamadas')

        elif (x=='Cabina 18'):
            for line in ru:
                p=line.split(",")[1]
                fhast=line.split(",")[8]
                hora1=self.prueba.GetLabel()
                if (p=='"usuario18"' and fhast>hora1):
                    e=line.split(",")[13]
                    if (e=='"FAILED"'):
```

```
                        self.lista2.AppendText('Llamada Fallida
\n')
                    elif (e=='"NO ANSWER"'):
                        self.lista2.AppendText('No Contesta \n')
                    elif (e=='"BUSY"'):
                        self.lista2.AppendText('Ocupado \n')
                    else:
                        fh1=line.split(",")[9]
                        fh=fh1.split('"')[1]
                        n1=line.split(",")[2]
                        n=n1.split('"')[1]
                        t=line.split(",")[12]

                        # ##### TARIFADOR #############

                        rut=open('/tarifas.txt','r')
                        for linea in rut:
                            codigo=linea.split("|")[1]
                            d=[] #creo un arreglo
                            d=line.split(",")[7]
                            dest8=d[9:-23]
                            dest7=d[9:-24]
                            dest6=d[9:-25]
                            dest5=d[9:-26]
                            dest4=d[9:-27]
                            dest3=d[9:-28]
                            dest2=d[9:-29]
                            dest1=d[9:-30]
                            if (codigo==dest8):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest7):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest6):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest5):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]

c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest4):
                                valor=linea.split("|")[2]
```

```
                                t1=t.split('"')[1]
c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest3):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]
c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest2):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]
c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)
                            elif (codigo==dest1):
                                valor=linea.split("|")[2]
                                t1=t.split('"')[1]
c1=round(int(t1)*(int(valor)*.0167))
                                c=str(c1)

                        self.lista2.AppendText(fh+'
'+n+'                '+t1+'                    '+c+'  \n')
                        pago=int(round(pago+c1))
                        self.precio.SetValue(str(pago))

            self.lista2.AppendText('No hay mas llamadas')
        ru.close()
```

Apéndice N: Tabla de Comandos del Visor CrystalFontz CFA634-NFA-KU

El visor seleccionado para el presente Trabajo Especial de Grado es el CrystalFontz CFA634-NFA-KU de puerto USB y pantalla LCD. A través de la tabla 7 se exponen los principales comandos para su control y configuración.

COMANDO (Código ASCII)	FUNCION
001	Colocar cursor al inicio
003	Restaurar el display
004	Ocultar cursor
005	Mostrar cursor debajo de la línea
006	Mostrar cursor de bloque vertical
008	Borrar
010	Baja el cursor a la siguiente fila
011	Borrar carácter donde se encuentra el cursor
012	Limpiar el display
014	Control del color del fondo de pantalla
015	Control del contraste de la pantalla
017	Posicionar el cursor (columna y fila)
026	Resetear display
000 / 016	Reservadas

Tabla 2. Comandos de Control del visor CrystalFontz CFA634-NFA-KU

Anexo A: Softphone X-lite

La figura 18 muestra el terminal virtual para VoIP tipo Softphone, modelo X-Lite, apropiado al utilizar el protocolo SIP. Permite realizar y recibir llamadas, y funciones adicionales tales como: remarcado, grabado de números, silencio, altavoz, llamada en espera, etc. Su configuración contempla el nombre de usuario, contraseña, dirección IP, tipo de servidor y plan de marcado, entre otras.

Figura 18. Softphone X-Lite

Anexo B: Tabla de Tarifas de la empresa NGT

```
COD_TARIFA|COD_PAIS_CIUDAD|VALOR|MONEDA|UD_TIEMPO|TIPO_TARIFA|
VOIPTPUBLD0|1|105|VEB|MIN|RET|
VOIPTPUBLD0|1204|280|VEB|MIN|RET|
VOIPTPUBLD0|1242|1398|VEB|MIN|RET|
VOIPTPUBLD0|1242357|1398|VEB|MIN|RET|
VOIPTPUBLD0|1242359|1398|VEB|MIN|RET|
VOIPTPUBLD0|1242457|1398|VEB|MIN|RET|
VOIPTPUBLD0|1246|1398|VEB|MIN|RET|
VOIPTPUBLD0|124623|1398|VEB|MIN|RET|
VOIPTPUBLD0|1246240|1398|VEB|MIN|RET|
VOIPTPUBLD0|1246241|1398|VEB|MIN|RET|
VOIPTPUBLD0|1246250|1398|VEB|MIN|RET|
VOIPTPUBLD0|1250|280|VEB|MIN|RET|
VOIPTPUBLD0|1264|1398|VEB|MIN|RET|
VOIPTPUBLD0|1264235|1398|VEB|MIN|RET|
VOIPTPUBLD0|1264772|1398|VEB|MIN|RET|
VOIPTPUBLD0|1268|1398|VEB|MIN|RET|
VOIPTPUBLD0|1268464|1398|VEB|MIN|RET|
VOIPTPUBLD0|1268723|1398|VEB|MIN|RET|
VOIPTPUBLD0|1268724|1398|VEB|MIN|RET|
VOIPTPUBLD0|1268725|1398|VEB|MIN|RET|
VOIPTPUBLD0|1268727|1398|VEB|MIN|RET|
VOIPTPUBLD0|1268728|1398|VEB|MIN|RET|
VOIPTPUBLD0|1268729|1398|VEB|MIN|RET|
VOIPTPUBLD0|1268773|1398|VEB|MIN|RET|
VOIPTPUBLD0|1284|1398|VEB|MIN|RET|
VOIPTPUBLD0|128444|1398|VEB|MIN|RET|
VOIPTPUBLD0|1284496|1398|VEB|MIN|RET|
VOIPTPUBLD0|1284499|1398|VEB|MIN|RET|
VOIPTPUBLD0|1289|280|VEB|MIN|RET|
VOIPTPUBLD0|1305|105|VEB|MIN|RET|
VOIPTPUBLD0|1306|280|VEB|MIN|RET|
VOIPTPUBLD0|1340|525|VEB|MIN|RET|
VOIPTPUBLD0|1345|1398|VEB|MIN|RET|
```