Linux™ para las ciencias biomédicas

NICOLÁS RIGALLI
ALFREDO RIGALLI

CENTRO UNIVERSITARIO DE
ESTUDIOS MEDIOAMBIENTALES

LINUX PARA LAS CIENCIAS BIOMÉDICAS

DE PRINCIPIANTE A EXPERTO

Modulo 5

Diseño de aplicaciones utilizando Gambas

AUTORES

Alfredo Rigalli

Nicolás Francisco Rigalli

Centro Universitario de Estudios Medioambientales

Facultad de Ciencias Médicas

Universidad Nacional de Rosario

2021

Rigalli, Alfredo

Línux para las ciencias biomédicas : de principiante a experto / Alfredo Rigalli ; Nicolás Francisco Rigalli ; ilustrado por Natalia Trost. - 1a ed - Rosario : Alfredo Rigalli, 2021.

Libro digital, PDF

Archivo Digital: descarga

ISBN 978-987-86-8420-8

1. Sistemas Operativos. 2. Ciencias de la Información. I. Rigalli, Nicolás Francisco II. Trost, Natalia, ilus. III. Título.

CDD 005

Prefacio

Hace una decena de años tomé por primera vez contacto con el sistema operativo LINUX y rápidamente pude ver sus beneficios. Sin embargo la sensación de soledad no deja de invadirnos en ciertas circunstancias, que en mi caso fueron rápidamente disipadas gracias a la ayuda de mi hijo que ya se había convertido en usuario autodidacta de este sistema. Sin duda una gran tranquilidad para el usuaria es saber que tras la puerta hay alguien dispuesto a ayudarnos y solucionar los problemas. No mucho más tarde me di cuenta que en el mundo hay miles o millones de usuarios con la misma filosofía.

Transitando el mundo de las ciencias biomédicas, me he encontrado muchos usuarios de otros sistemas operativos y otras aplicaciones, que se daban cuenta de las ventajas del sistema, pero presentaban el mismo temor que había sufrido en mis comienzos. Así, podría decir que surgió la idea de esta obra, con el objetivo de que este libro se convierta en una puerta para el usuario iniciado, seguro que si golpea en ella tendrá una ayuda o al menos toda la intención de hacerla. La idea de esta obra es también convertir al usuario principiante en experto, de manera que cada uno se transforme en una ayuda, en un punto de referencia para nuevos usuarios.

Es abundante la información sobre Linux, sus aplicaciones y propuestas de soluciones a problemas que se plantean en la red. Sin embargo, no abundan las propuestas que permitan guiar desde el inicio al interesado, mostrando los pasos necesarios para poner en marcha la computadora con el nuevo sistema operativo. Este libro, que consideramos tomo 1, aporta lo mínimo indispensable para poder comenzar a trabajar, un breve detalle de sus aplicaciones y accesorios más comunes y las mínimas tareas para poder administrar los archivos.

La obra se halla dividida en clases en lugar de capítulos, dado que la misma es el material de referencia del curso “Linux para las ciencias biomédicas. De principiante a experto”, dictado por el departamento de docencia del Centro Universitario de Estudios Medioambientales de la Facultad de Ciencias Médicas de la Universidad Nacional de Rosario. A su vez cada clase está asistida con un vídeo al que se puede acceder directamente desde el texto a través de un link y dispone de ejercitación de autoevaluación, también accesible para el alumno.

Los autores esperamos que este libro se transforme en esa ayuda permanente que necesitamos los usuarios de cualquier sistema operativo y que cada nuevo usuario se multiplique en otros principiantes.

Alfredo Rigalli

Tabla de contenidos

Clase 1

En este módulo veremos una introducción a la programación en el entorno Linux. De ninguna manera este módulo intenta transformar al usuario de Linux en un programador experto, sino que el objetivo del módulo es mostrar algunos caminos disponibles en los que el usuario puede crear sus propias aplicaciones o rutinas. Existen diferentes herramientas y lenguajes de programación y muchos cursos de altísimo nivel y calidad referidos a este tema. En este curso nos concentraremos en la programación de tareas a través de la terminal y en el uso básico de una aplicación de programación visual, conocida como Gambas. También veremos a modo de demostración, scripts construidos en R, que funcionan aproximadamente como un software y cuya eficiencia es destacable. La programación en R es parte de otro curso que forma parte de la oferta del CUEM.

La pregunta que nos podemos hacer es ¿Por qué querer hacer una aplicación?. La respuesta es sencilla. Si bien la tecnología nos provee de herramientas cada vez más sofisticadas y efectivas, enfrentamos situaciones en que la misma no alcanza para cumplir con nuestros objetivos. Pongamos un ejemplo sencillo. Supongamos que debemos hacer un experimento con animales en que éstos deben recibir ciertos tratamientos a diferentes horas del día. Como es común con el trabajo con animales, los tratamientos llevan su tiempo, por lo que la aplicación de los tratamientos se pueden desfasar. Así, podemos tener que un animal recibe una inyección a las 9:32 y el otro no podrá hacerlo al mismo tiempo y quizás por la dinámica del experimento y la destreza de los operarios el otro reciba la misma inyección a las 9:37. Para evitar equivocaciones seremos cuidadosos en la organización y registro de las actividades de cada animal, pero es claro que el control del tiempo es clave.

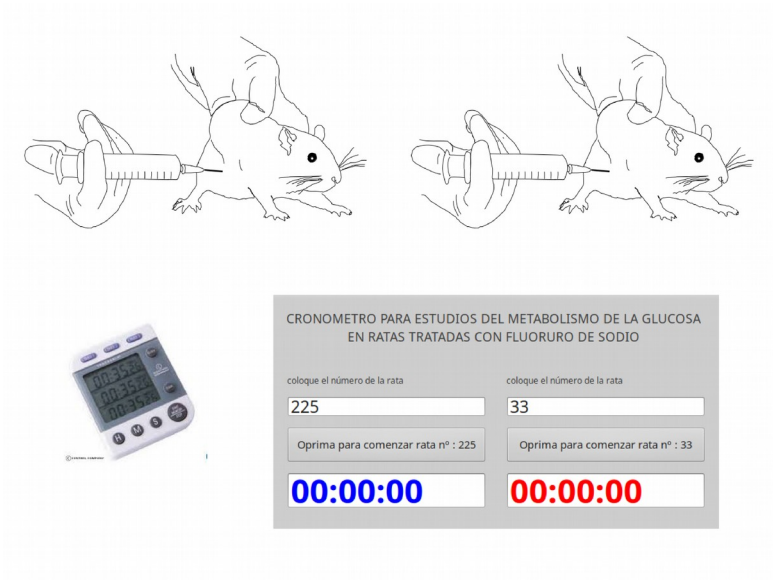


Figura 1

Como vemos en la Figura 1, la tecnología nos provee de cronómetros con 2, 3 y más cronómetros. Podríamos así utilizar el cronómetro superior para la primer rata y el inferior para la segunda. También podríamos construir nuestro propio cronómetro a través de herramientas de programación. La ventaja de "nuestra aplicación" es que podremos agregar detalles que nos permitan realizar el procedimiento con más seguridad. Si quisiéramos trabajar con 70 animales, podríamos hacerlo, construyendo una aplicación que tenga 70 cronómetros, y para cada animal podríamos programar las tareas, haciendo que "la aplicación" nos avise con un sonido o un cambio de color.... o lo que

deseemos, qué tenemos que hacer, sobre que animal y en que momento. Es común en los tiempos que corren que cuando se deben realizar este tipo de actividades, cada miembro del laboratorio aporta su celular, controlando cada uno con un celular y su cronómetro. Sin duda, construir "nuestra aplicación" aportará no solo organización al trabajo, sino un buen aspecto estético.

Instalación de Gambas

La aplicación Gambas actualmente está en su versión 3. Como siempre tenemos más de una mecanismo. Podremos hacerlo desde el centro de software, desde terminal o Synaptic.

Desde centro de Software

Para instalar Gambas 3 desde el centro de software debe abrir el centro de software y buscar Gambas. Para ello oprima la lupa y escriba gambas en la ventana. Si está disponible hallará una respuesta como muestra la Figura 2. En el caso mostrado, nos indica que la aplicación está en el centro de software y además está instalada.

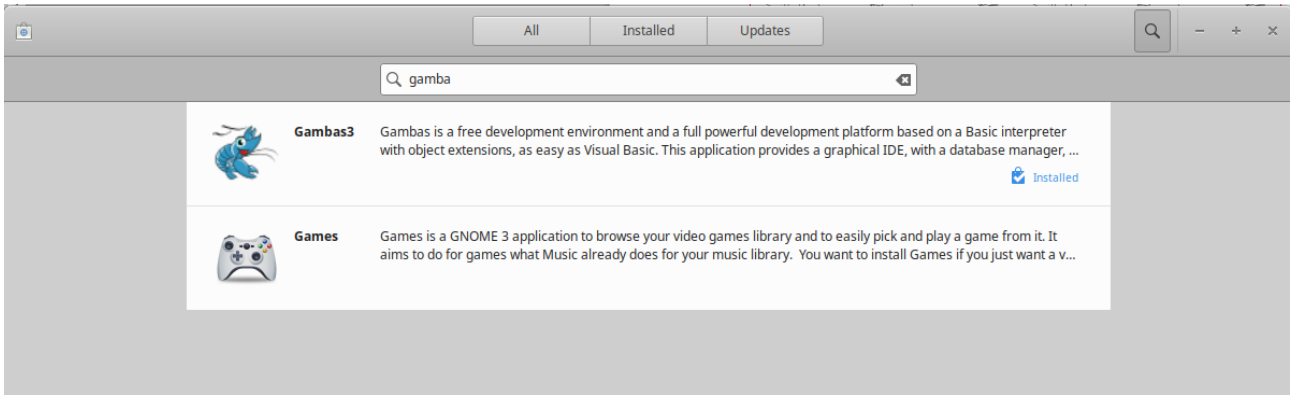


Figura 2

Si al buscar Gambas, no le muestra el icono y su descripción, deberá bajar la aplicación de un repositorio. Para esto desde el mismo centro de software, diríjase al icono del vertice superior izquierdo y elija Software & Updates, Figura 3

Figura 3

Elija la pestaña Other softwares y oprima Add, que le proporcionará una ventana para colocar el repositorio de donde descargar Gambas, figura 4

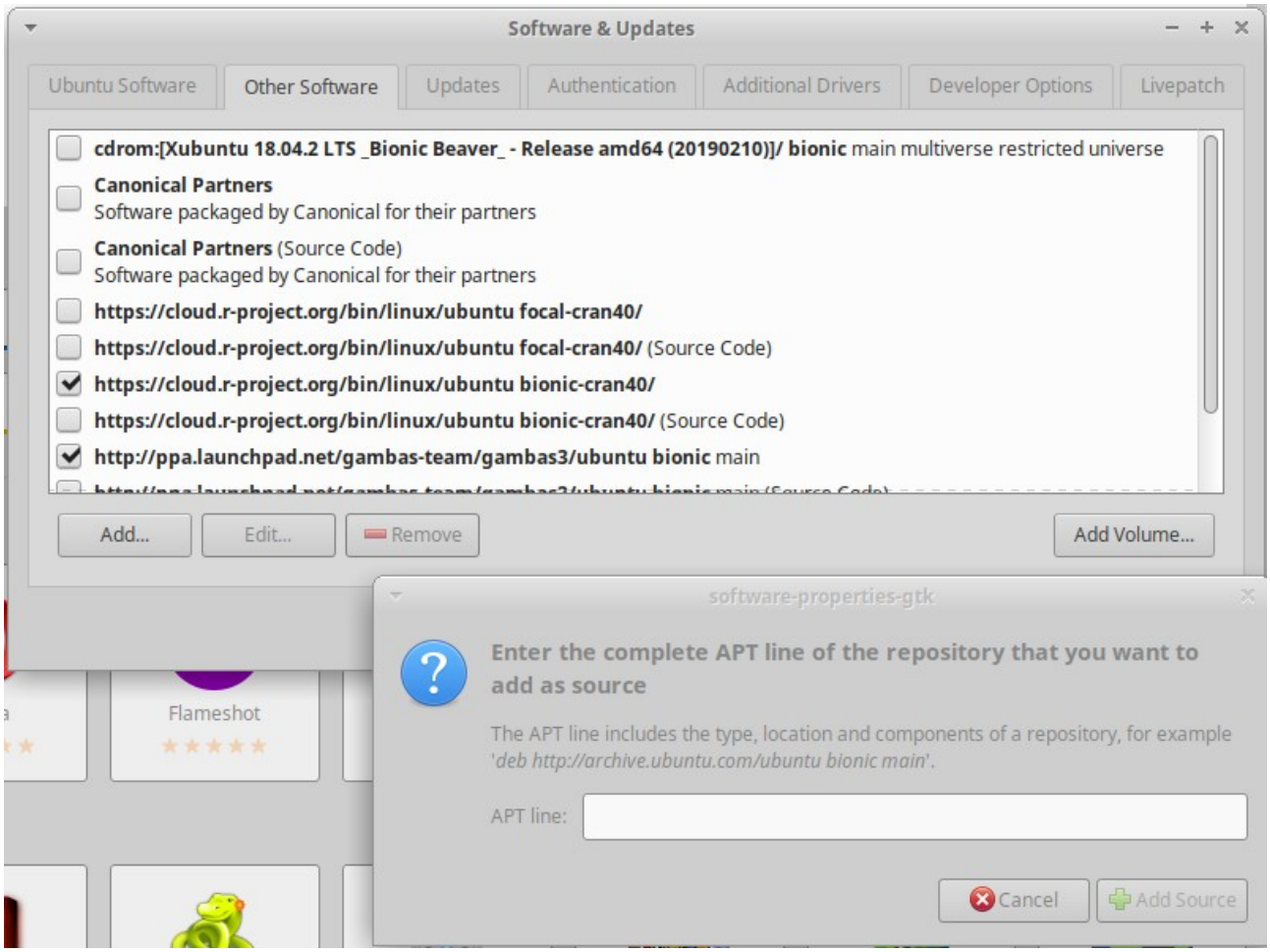


Figura 4

Luego en APT line coloque la siguiente linea: `ppa:gambas-team/gambas3`

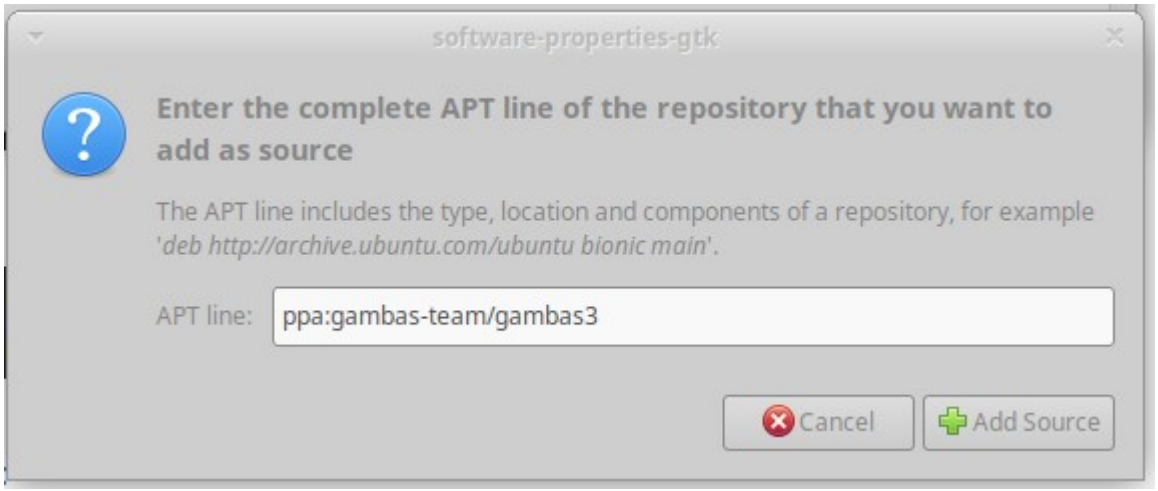


Figura 5

Luego retorne a la página principal del centro de software y oprima Updates. Si luego busca Gambas nuevamente, debería aparecerle el ícono y de allí realizar la instalación.

Desde Synaptic

Synaptic podemos arrancarlos desde el menú de aplicaciones.

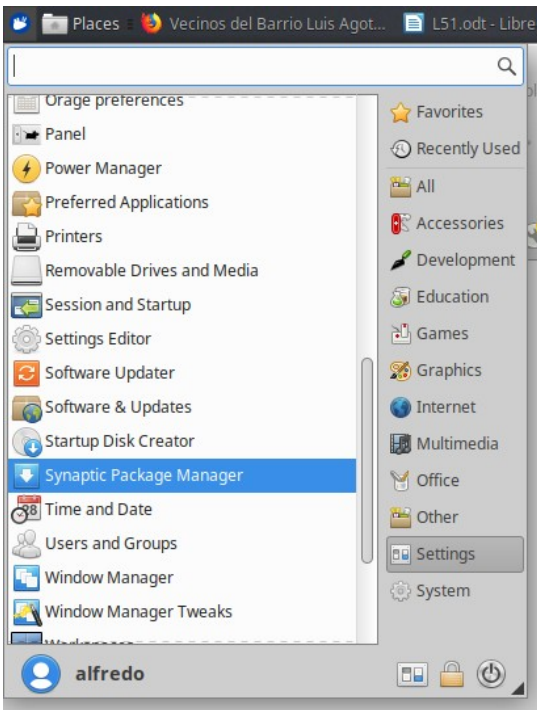


Figura 6

o bien desde una termina con el comando

sudo synaptic

Una vez en Synaptic, colocamos buscar Gambas y marcamos para instalación

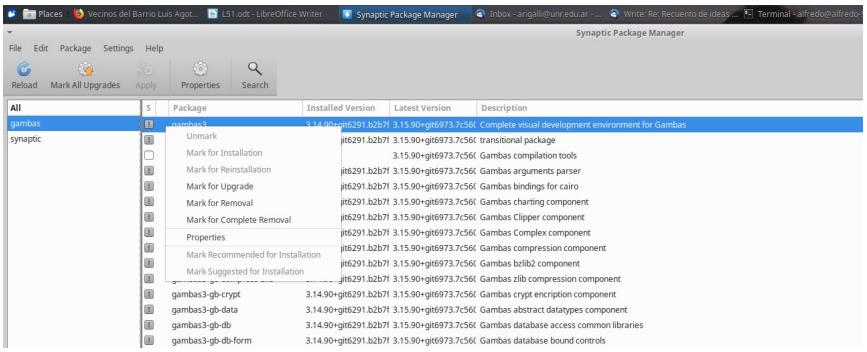


Figura 7

luego oprimimos Apply y quedará disponible.

Hallará Gambas 3 en el menú de aplicaciones, normalmente dentro de Developments, Figura 8

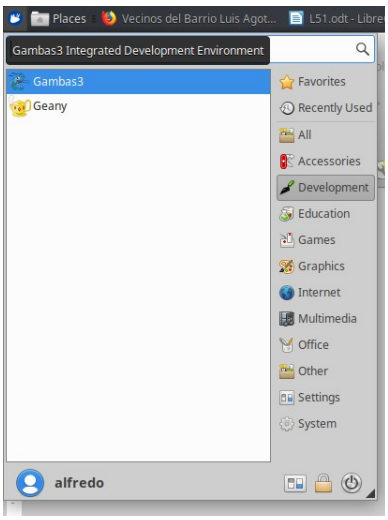


Figura 8

Arrancando Gambas

Desde el menú de aplicaciones, Figura 8, hacemos click en Gambas 3 y se abrirá el programa mostrando la Figura 9.

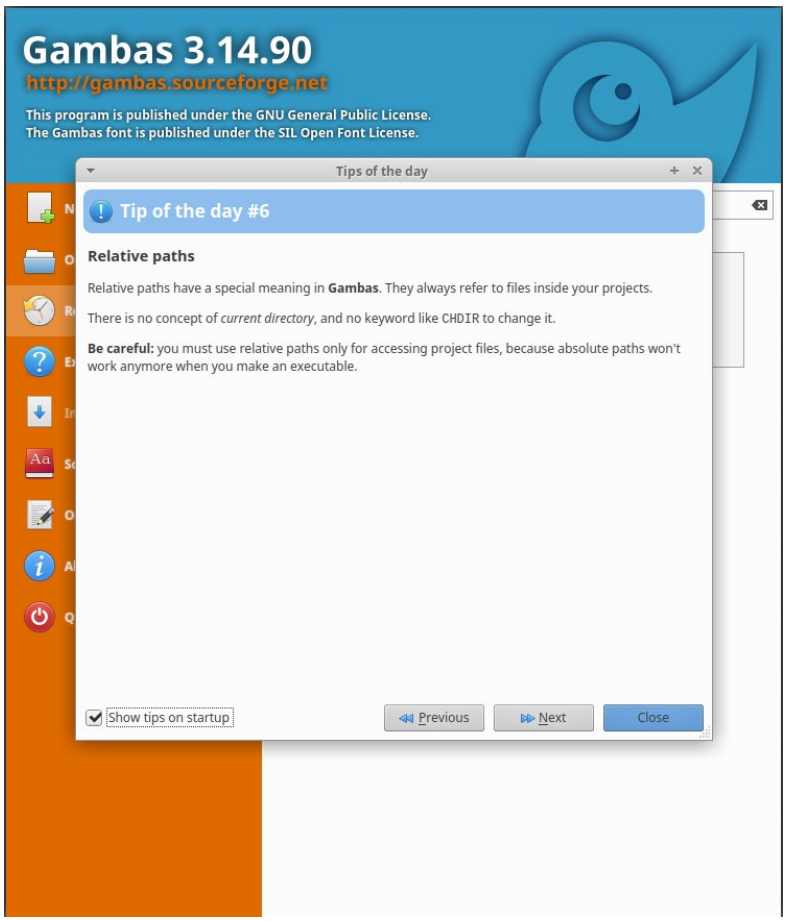


Figura 9

Al inicio nos mostrará algunos tips de programación, que por ahora podemos obviar. Si lo deseamos desactivar de manera permanente desmarcamos a la izquierda abajo Show tips on startup. Así pasamos a la aplicación, que por cierto es muy amplia.

Figura 10

Arranca con el foco en Recent projects. Pero como podemos ver tenemos varias opciones:

New project, que será para crear un nuevo software.

Open project, que nos permite abrir el proyecto que deseamos que puede estar dentro de Recent projects o no.

Examples: que nos muestra ejemplos muy variados. Para ello deberemos descargarlos, Figura 11

Figura 11

Iniciar un proyecto

Para iniciar un proyecto hacemos click sobre New project. Se abrirá una ventana que nos permite elegir el tipo de proyecto, Figura 12. Existe una variedad de proyectos, pero en este curso no centraremos en Graphical application, que nos permite realizar un software con una interfaz gráfica como el que mostramos para los cronómetros. Por supuesto se pueden realizar prácticamente cualquier tipo de aplicaciones a partir de él.

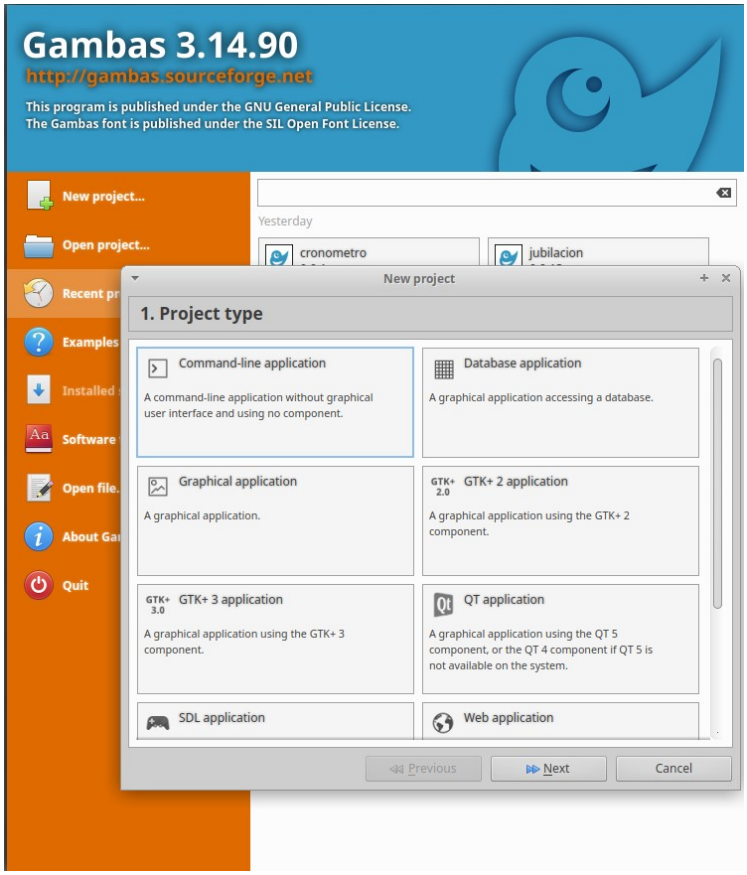


Figura 12

Una vez marcado Graphical application, oprimimos Next y nos pedirá un directorio donde alojar el proyecto. Puede ser el que desee, en el ejemplo elegimos L51, haciendo referencia al curso linux módulo 5 clase 1, Figura 13.

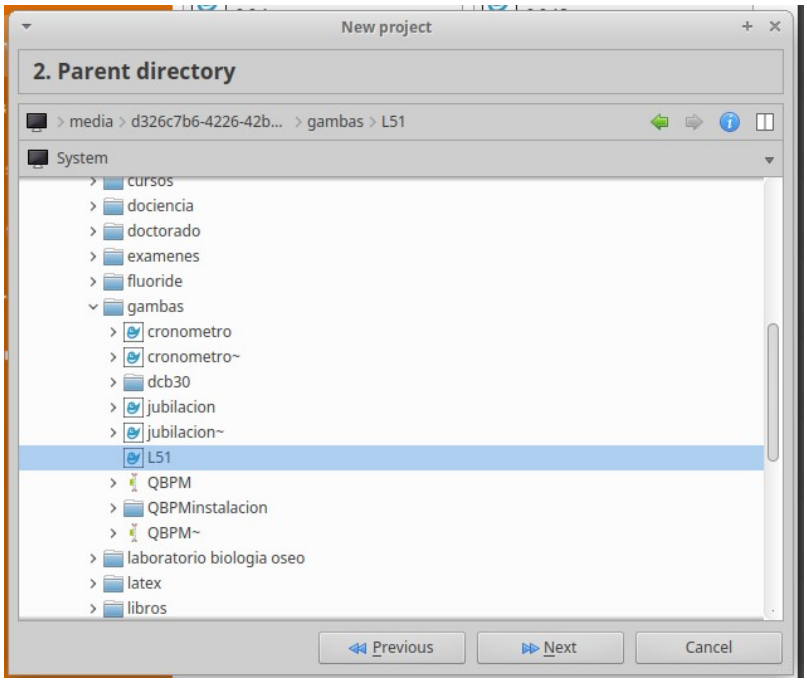


Figura 13

Oprimimos Next y nos pedirá un nombre de proyecto y titulo de la aplicación. El último será el que mostrará cuando corramos la aplicación que desarrollaremos. En ese caso llamamos al proyecto CursoLinuxL51 y como título de la aplicación IngresoEgreso, en alusión a lo que haremos con ella, Figura 14.

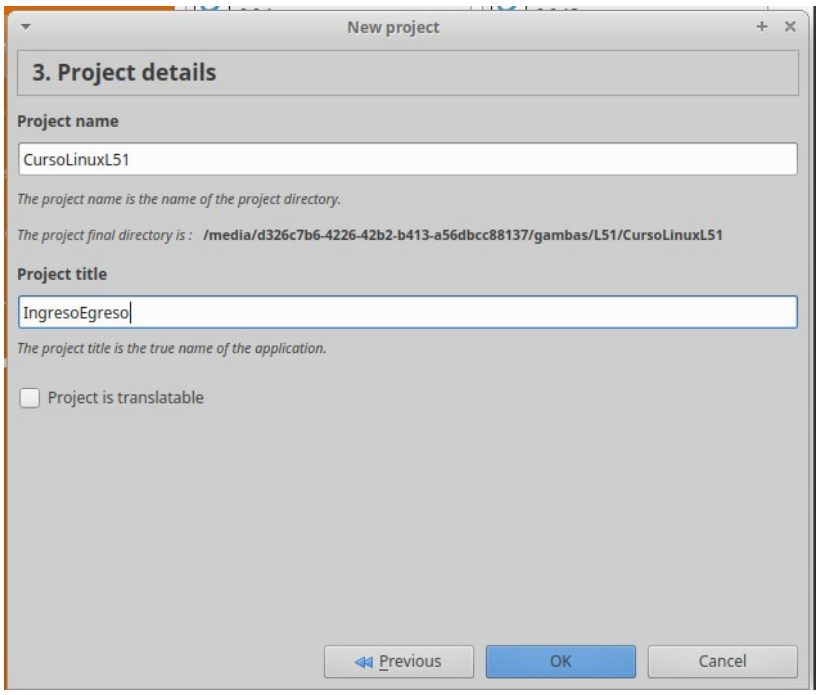


Figura 14

Oprimimos OK y accederemos a una ventana donde deberemos comenzar a programar nuestra aplicación, Figura 15.

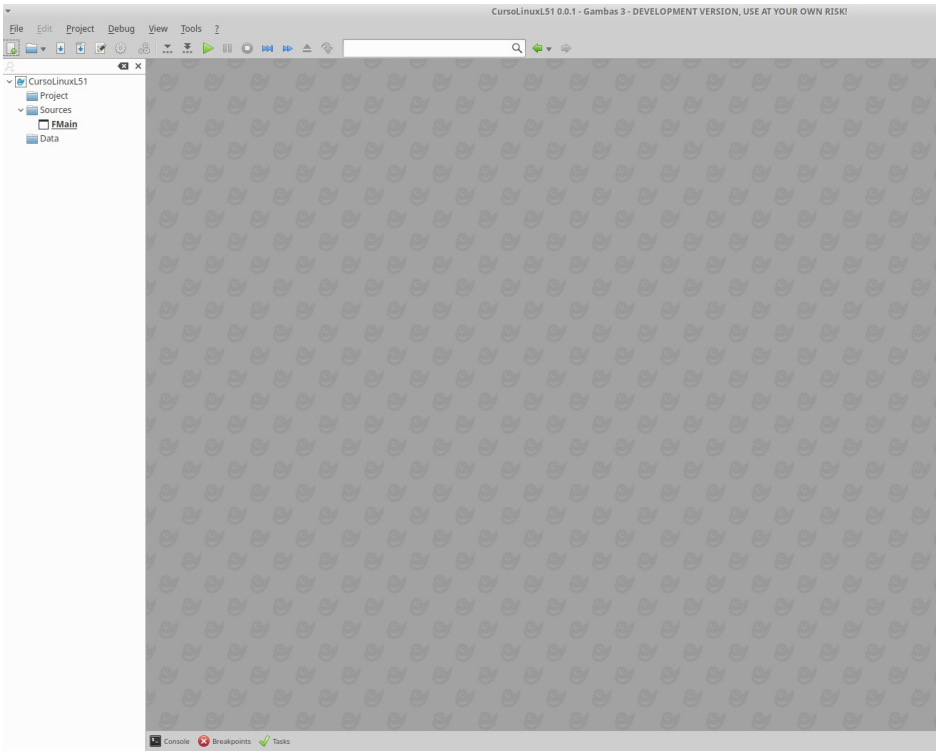


Figura 15

Nuestro primer proyecto será un software con interfaz visual que nos permita introducir cantidad de una sustancia gastada en un laboratorio, la cantidad de sustancia ingresada al stock y el saldo de la misma. Para ello crearemos en nuestro proyecto 3 formularios. Llamamos formulario a un espacio donde diseñamos lo que veremos cuando ejecutemos la aplicación y desde donde ejerceremos las acciones. Llamaremos a estos formularios

controlador: a través de este formulario accedemos al software y de acá nos podremos dirigir a los formularios ingresos y egresos

ingreso: en este formulario colocaremos la droga que se adquiere y se suma al stock.

egreso: en este formulario colocaremos la droga que se gasta.

FMain: es un formulario con el que arranca cada software.

Crear un formulario

Para crear un formulario colocamos el puntero del mouse sobre Sources, en la ventana de la izquierda, oprimimos botón derecho y elegimos New y luego Form, Figura 16

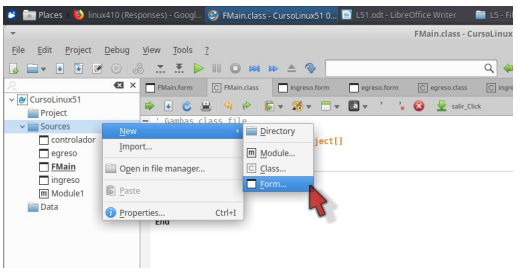


Figura 16

Se abrirá una ventana en la que ponemos el nombre del formulario, por ejemplo: controlador, Figura 17

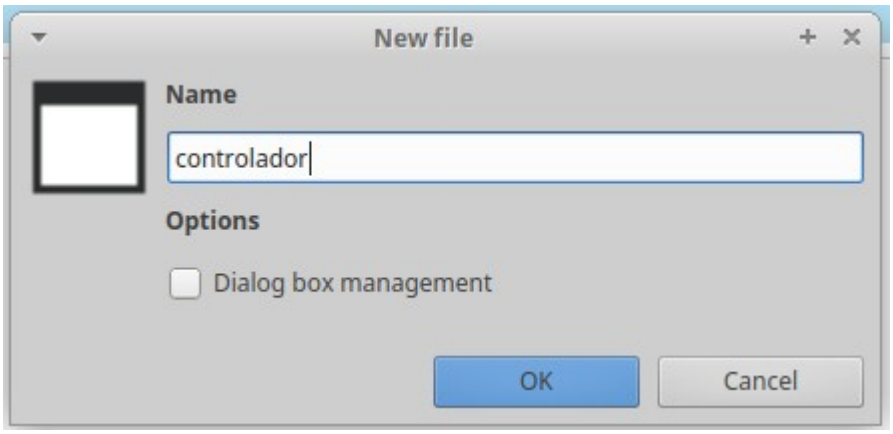


Figura 17

Arrancar y detener el software

Cuando tengamos terminado nuestro software, diseñaremos un ícono y éste tendrá un lugar más en nuestro menú de aplicaciones, como las otras aplicaciones. Pero mientras diseñamos la aplicación, desearemos ir probando el mismo. Para ello en la barra de herramientas tenemos dos botones que nos permiten:

Iniciar el software :  . Si el software no está en funcionamiento, está verde el icono de encendido y deshabilitado los comandos de pausar y detener.

Detener el software:  . Si el software está en funcionamiento, estarán habilitados los botones de pausar y detener

Es importante distinguir que estos botones permiten controlar el encendido y apagado del software que estamos diseñando. Para cerrar Gambas, no son útiles estos botones y de desear hacerlo se hará como con cualquier aplicación.

Diseño de formularios

Comenzaremos por un sencillo diseño de un formulario. Colocaremos tres objetos en el formulario FMain

1- Label de bienvenida

2- Ingresar a la aplicación

3- salir de la aplicación

Para ello marcamos con el puntero del mouse la herramienta de lista y manteniendo apretado el botón izquierdo la arrastramos al formulario. Para nuestro fin colocamos dos botones y un label, Figura 18.

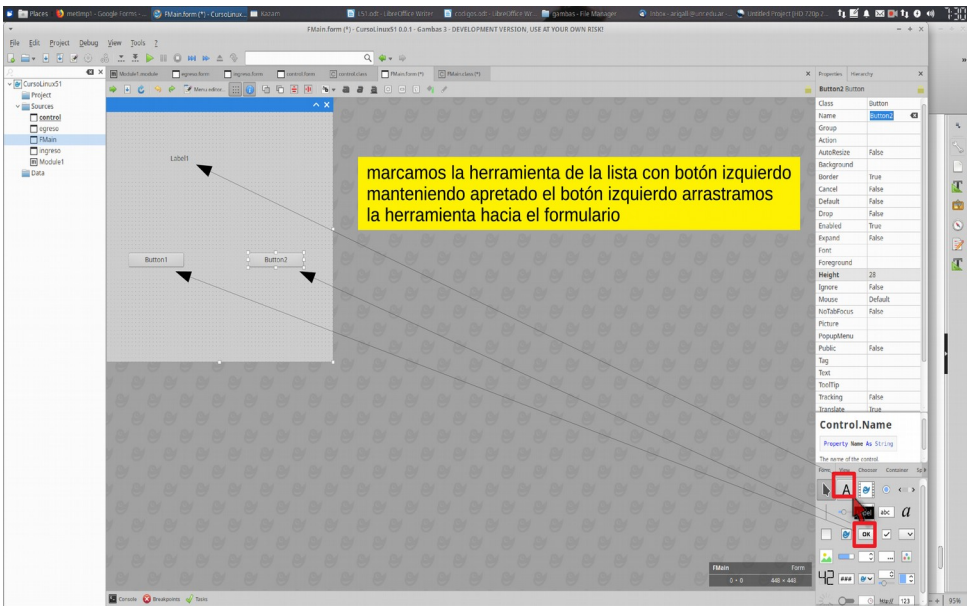


Figura 18

Cambiamos las propiedades de los objetos introducidos. Por ahora solo cambiaremos el texto que figura en el formulario. Para esto en la lista de Propiedades, modificamos el texto visible. Primero hacemos click en el objeto del formulario y veremos que se actualiza la lista de propiedades a dicho objeto. Allí escribiremos el texto que queremos que se vea en el atributo Text y por una cuestión de orden también le cambiaremos el nombre. Usted elegirá el mejor y más claro mecanismo con el avance del tiempo. Pero para comenzar sugerimos que el nombre del objeto y el texto visible sean coincidentes en lo posible, llamaremos en este caso a ambos: ingresar. Para ello hacemos click en ls cuadros de texto a la derecha de Name y Text y colocamos: ingresar. De la misma manera haremos para Bienvenido y salir

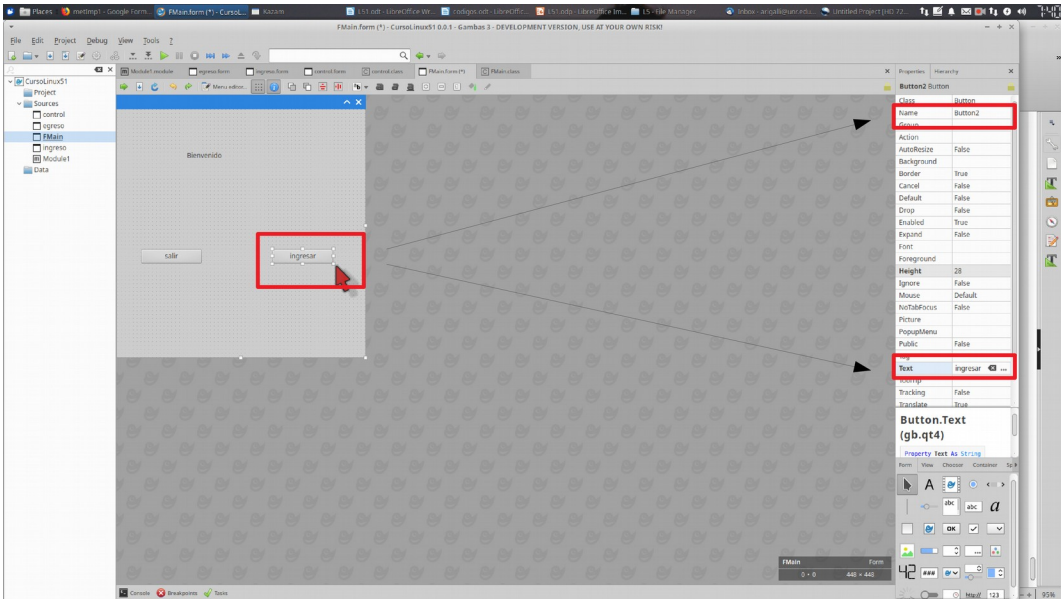


Figura 19

Salir del software

Para salir de nuestro software tenemos un botón llamado salir que debemos programar. Para ello hacemos click con el botón derecho del mouse sobre el botón y se abrirá un menú con diversas opciones. Nos interesa en este momento Event, en donde podremos elegir ante que evento reaccionará el botón. La opción más sencilla para un botón es Click y es la que elegiremos, pero como puede ver, hay muchas opciones que iremos utilizando.

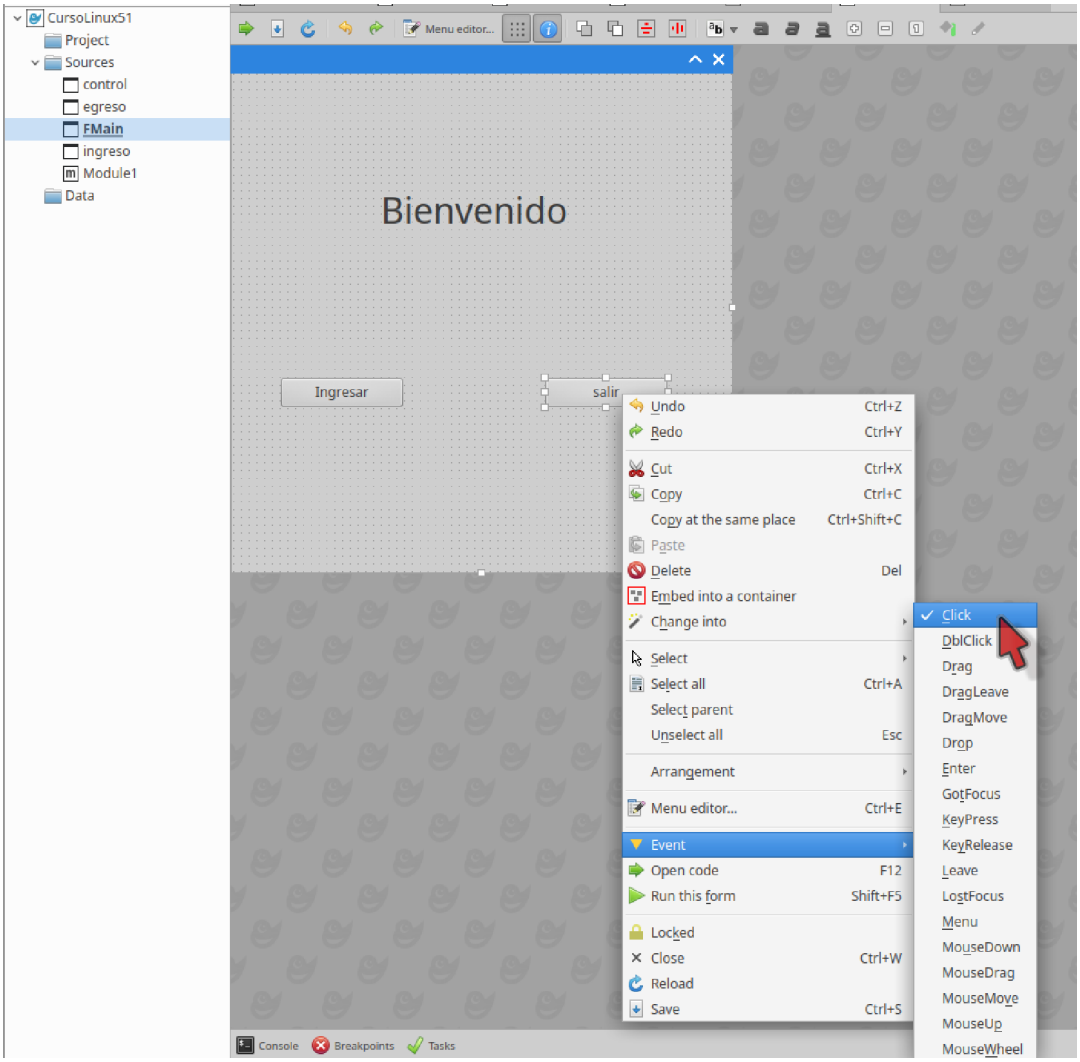


Figura 20

Al marcar Click nos llevará al formulario o clase en que podemos escribir el código que nos permita salir. El cursor del mouse quedara debajo del texto

"Public Sub ingresar_Click()"

allí comience a escribir quit, antes de finalizar Gambas le mostrará la función a utilizar.

Para finalizar un programa la sentencia es: quit.

Podemos escribirla completa o bien hacer click sobre la opción que nos propone gambas, Figura 21.

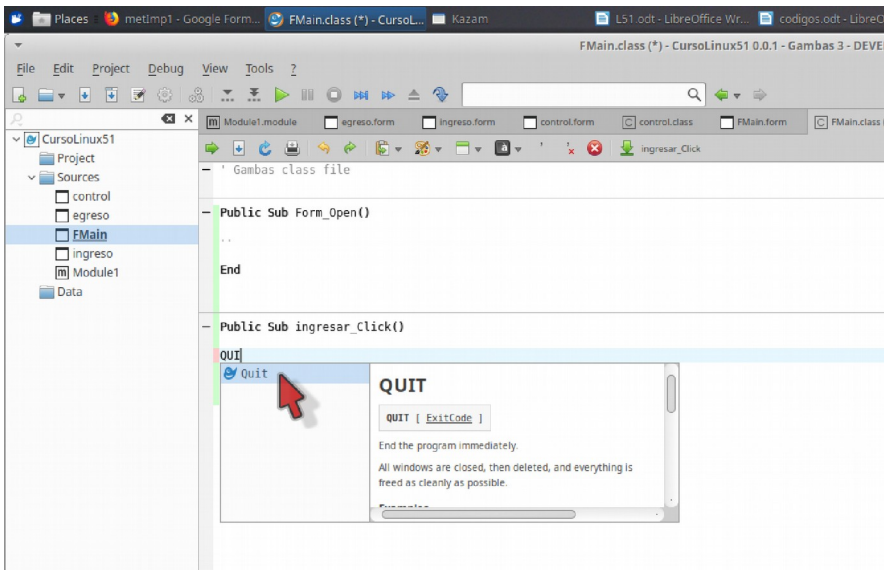


Figura 21

Así nos queda ya programada la salida del software. En la Figura 22, puede observar como queda programada la salida del software. La imagen muestra el código correspondiente a todo el formulario. La parte que nos interesa en este punto

Public Sub salir_Click()

  Quit

End

Así, nos irán quedando en esa ventana todos los eventos programados para el formulario FMain. De la misma manera será para todo el programa. Puede distinguir en el código varias cosas:

1- Public **Sub salir_ Click**()

hace referencia a que es una **Sub**rutina que se ejecutará, al hacer **Click** en el boton **salir**

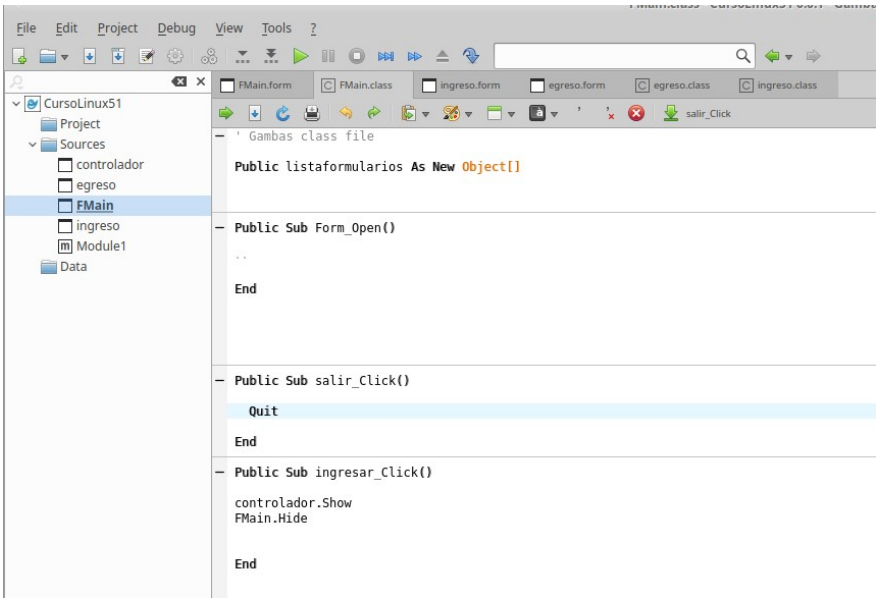


Figura 22

Si arrancamos el software desde el menú de tareas de Gambas, ya no necesitaremos apagarlo desde el menú, sino que oprimiendo el botón salir deberá terminar.

Pasar de un formulario a otro

Para pasar de un formulario a otro y luego volver al inicial las rutinas se repiten. Veremos el caso para ir del formulario **FMain** al formulario **controlador** y luego regresar a **FMain**.

Lo que deseamos es que al oprimir el botón ingresar del Formulario FMain se oculte éste y se pueda ver controlador. Como ya hemos visto,

1. posicionamos el mouse sobre el botón ingresar,
2. oprimimos botón derecho y elegimos Event
3. allí seleccionamos Click
4. se abrirá le ventana FMain.class, con los códigos y el mouse quedará en el evento: Public Sub ingresar _ Click()
5. allí escribiremos que se muestre el formulario controlador, con el siguiente código: controlador.Show. La acción Show a continuación del nombre del formulario, será interpretada como una orden para mostrar ese formulario
6. a continuación programamos la orden para que se oculte FMain.
7. Fmain.Hide. La acción Hide, oculta FMain, por lo que al oprimir el botón Ingresar, dejaremos de ver FMain y pasaremos a ver controlador.

Diseñemos ahora el formulario controlador. Este formulario tendrá tres botones

Ingreso sustancia: que nos llevará al formulario ingreso

Egreso sustancia: que nos llevará cuando lo oprimamos al formulario egreso

Salir: nos retornará a Fmain. Esta acción es la que programaremos. De la misma manera que venimos haciendo:

1. click derecho sobre el botón a programa
2. elegimos Event y Click. Así llegaremos a controlador.class
3. en la subrutina salircontrolador_Click, programamos mostrar FMain y ocultar controlador, Figura 23

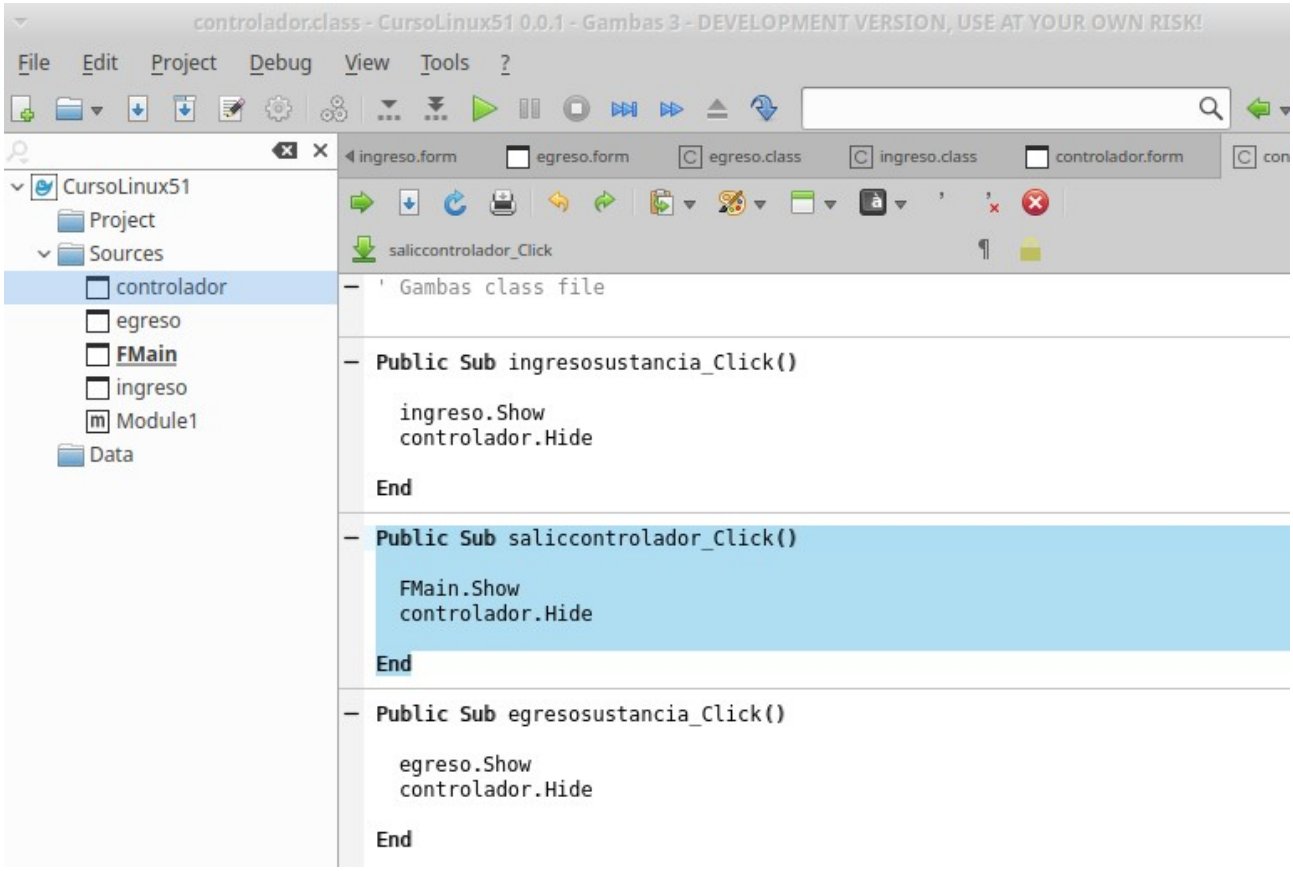


Figura 23

De esta manera cuando oprimamos el botón Salir del formulario controlador, ocultaremos éste y mostraremos nuevamente FMain.

De la misma manera programaremos el pasaje y retorno entre el formulario controlador y egreso, así como entre controlador e ingreso.

Clase 2

En la primer clase instalamos Gambas y comenzamos nuestro primer desarrollo. Creamos un proyecto de nombre CursoLinux51 y cuyo título es IngresoEgreso. La Figura 24 muestra la ventana en la que hicimos estas acciones.

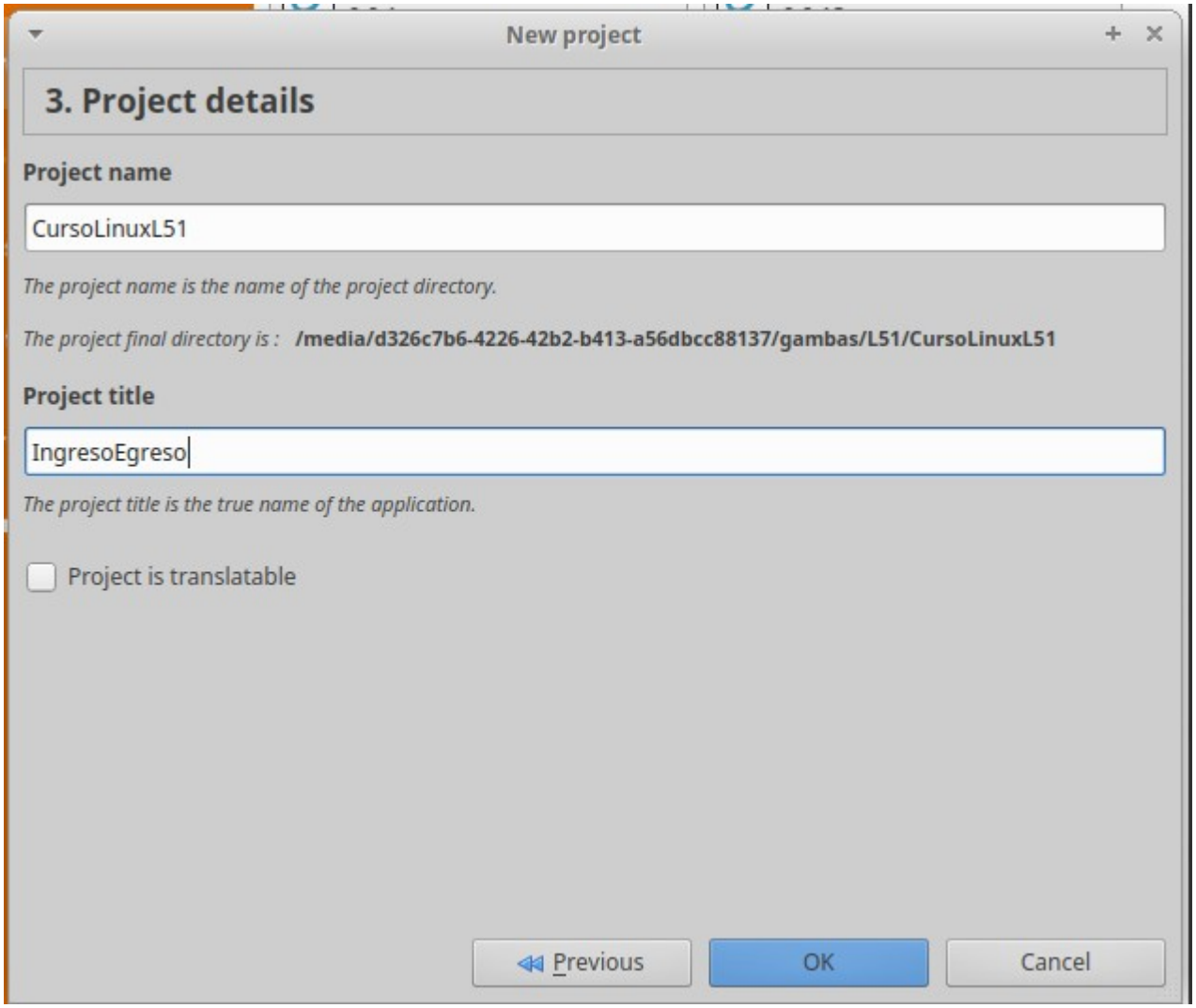


Figura 24

El proyecto está formado por cuatro formularios y su función será registrar cuanta droga se gasta o se compra, pudiéndonos informar de la cantidad disponible. Los formularios son

controlador: a través de este formulario accedemos al software y de acá nos podremos dirigir a los formularios ingresos y egresos

ingreso: en este formulario colocaremos la droga que se adquiere y se suma al stock.

egreso: en este formulario colocaremos la droga que se gasta.

FMain: es un formulario con el que arranca cada software.

Obviamente la aplicación que estamos desarrollando bien podría ser llevada adelante, con más facilidad y efectividad con una planilla de cálculo e inclusive con un block de nota de una computadora o celular. Pero la intensión es comprender los principios básicos de programación.

Este software será tan poco eficiente en su funcionamiento, que ni siquiera tendrá la capacidad de almacenar los valores ingresados, egresados y el saldo disponible de la sustancia. Además solo servirá para una sustancia.

Cuadros de diálogo

Los cuadros de diálogo nos permite introducir alertas al tomar una decisión, pudiendo concretar la acción planeada o volver hacia atrás. Si bien existen diferentes tipos veremos dos que pueden ser útiles a la hora de comenzar

1- cuadros de alerta y avance: nos aparece el cuadro despertando nuestra atención, pero solo nos permite seguir adelante. A este cuadro lo llamaremos cuadroOK

2- cuadro de alerta con aceptación o rechazo de la opción: nos permite realizar la acción o quedar en la situación actual. A este cuadro lo llamaremos cuadroSiNO

Veremos estos dos ejemplos en el formulario Fmain.

Cuadro con una opción

Colocaremos el cuadroOK en el botón ingresar del formulario Fmain. Este cuadro nos avisará que entraremos al software, pero solo tendremos la posibilidad de aceptar la acción. Por supuesto usted podría colocar cualquier otro cuadro de dialogo. Deseamos que el cuadro se active cuando oprimimos el botón y debe hacerlo antes de mostrar el formulario controlador.

El código que crea y llama estos cuadros es

```
Message.Info("texto que se desea ver en el cuadro", "OK")
```

En "texto que se desea ver en el cuadro" usted colocará lo que desea que aparezca. "OK" es lo que aparece en el botón

En nuestro caso utilizaremos

Message.Info("Usted ingresará al Software de administración de cloruro de sodio", "oprima para seguir")

Como deseamos que se ejecute antes de pasar al formulario controlador, lo colocaremos en el evento click del botón ingresar y antes de las acciones ya existentes, quedando el código de la manera

```
Public Sub ingresar_Click()
Message.Info("Usted ingresará al Software de administración de cloruro de sodio", "oprima para seguir")
controlador.Show
FMain.Hide
End
```

Así, al oprimir ingresar nos mostrará el cuadro de dialogo con el texto "Usted ingresará al Software de administración de cloruro de sodio" con un solo botón sobre el que podremos hacer click o bien oprimir la tecla enter. Cuando el botón está resaltado en azul, funciona con click del mouse o enter. Simultáneamente al imprimir ingresar el software se posición en el evento click y luego de éste ejecutará las dos acciones siguientes que son mostrar el formulario controlador y ocultar Fmaim, Figura 25.

Figura 25

Cuadro con dos botones

En el botón salir de FMain colocaremos un cuadro de diálogo con botones de aceptar y rechazar. Cuando hagamos click sobre este botón nos mostrará un cuadro con dos botones y un texto, su forma general es

```
IF Message.Question("¿Desea salir de la página?", "Si", "No") = 2 THEN
STOP EVENT
ELSE
acción a realizar
END IF
```

En nuestro caso dentro del botón salir dentro del evento click colocaremos el código del cuadroSiNo. Este tiene dos partes. Una luego the THEN que se ejecuta si oprimimos No y que finaliza la acción iniciada retornando al punto en que nos hallábamos. La otra acción está luego de ELSE y se ejecuta si oprimimos el botón Si. Por supuesto los textos de los botones Si y No de los botones pueden modificarse. También es modificable el orden de los botones. En el formato general de este botón =2 indica que la acción STOP EVENT (que nos permite no ejecutar la acción siguiente) lo hará con el segundo botón, es decir el que dice "No". Pero podríamos modificar esto

colocando =1 y la acción STOP EVENT se ejecutaría con el primer botón, es decir el que en este caso dice Si. Para nuestro caso el código quedará

```
Public Sub salir_Click()
    If Message.Question("¿Desea abandonar la aplicación?", "Si", "No") = 2 Then
        Stop Event
    Else
        Quit
    End If
End
```

Al oprimir salir, nos mostrará el cuadro de diálogo y simultáneamente el software estará ejecutando la acción del evento click del botón salir, Figura 26. Esperando una decisión. Si oprimimos el segundo botón, es decir el que dice No, volvemos a Fmain. Si oprimimos el primer botón (que dice Si) saldremos de la aplicación.

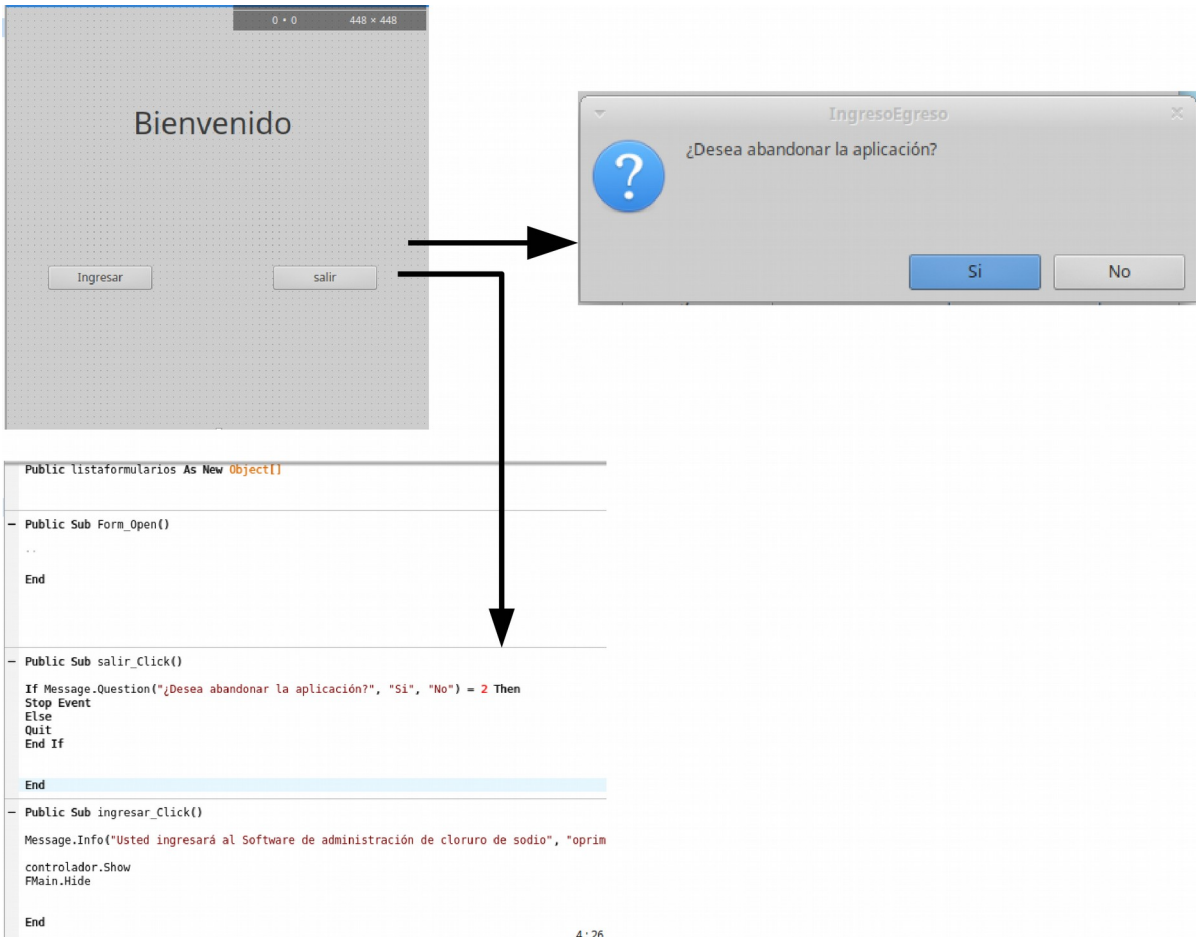


Figura 26

Algunas herramientas en manejo de objetos

Tooltip

Esta herramienta nos permite colocar una explicación a un objeto, de manera que al posicionar el puntero del mouse sobre el objeto nos de una explicación. Suele ser útil en aplicaciones con numerosos objetos, ya que nos brinda información de qué debemos hacer. Veremos en este caso colocar un tooltip en el botón ingresar. Justamente en este caso no sería útil, ya que es obvio que es el botón para ingresar a la aplicación, pero si es útil para entender el concepto. Para colocar el tooltip marcamos el objeto de manera que quede seleccionado (queda enmarcado en 4 delimitadores del objeto) y en la ventana Properties buscamos la propiedad Tooltip, hacemos click sobre los tres punts y se abrirá una ventana donde escribiremos lo que desamos que nos muestre la aplicación cuando el mouse pase sobre el botón. En este caso introduciremos el texto: "Oprima este botón para ingresar a la aplicación" o el texto que deseamos, Figura 27.

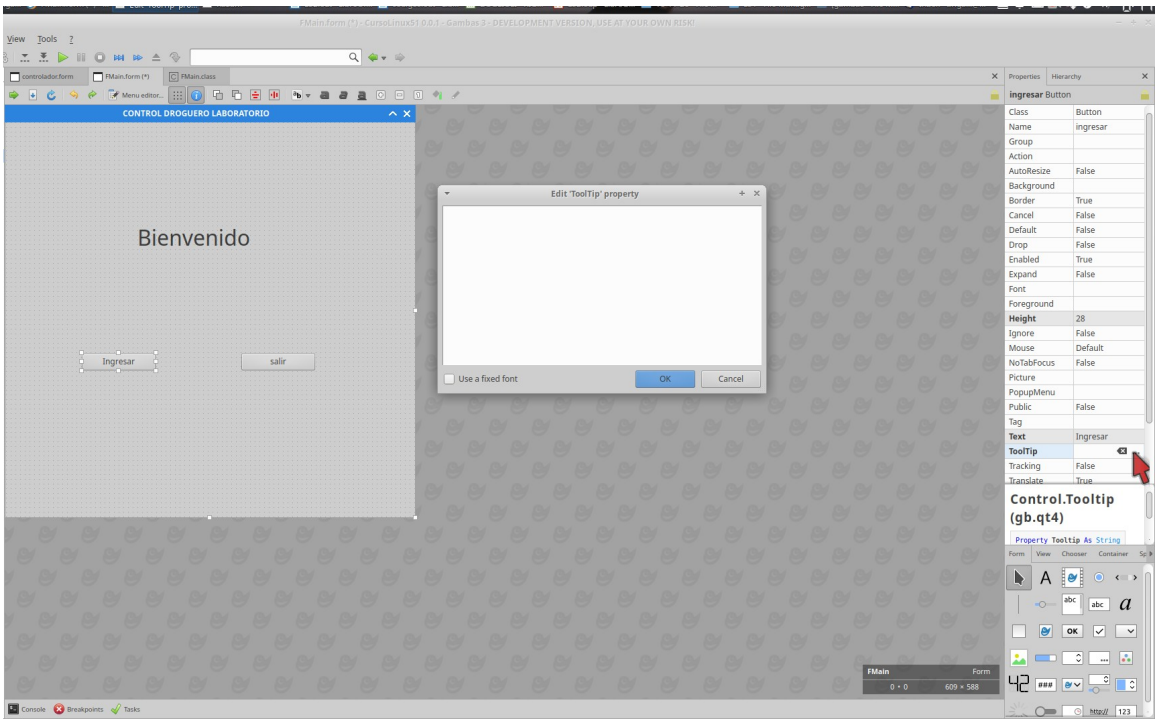


Figura 27

al oprimir estar ejecutando el software y posicionar simplemente el puntero del mouse sobre el botón tendremos en un cuadro negro la explicación introducida como tooltip, Figura 28.

Figura 28

Enabled

La propiedad Enabled es importante durante la ejecución de aplicaciones. Esta propiedad toma dos valores TRUE o FALSE. Si el valor está en TRUE, el objeto podrá ejecutar los eventos: Click, Dblclick, etc. Si está en FALSE, estará deshabilitado y por más que hagamos click no se ejecutará. Es una propiedad muy importante, porque ciertas ejecuciones en ciertos momentos de la aplicación pueden estar prohibidos. Para establecer esta propiedad, hay que marcar el objeto, ir a propiedades, buscar Enabled y elegir del menú desplegable TRUE o FALSE, Figura 29

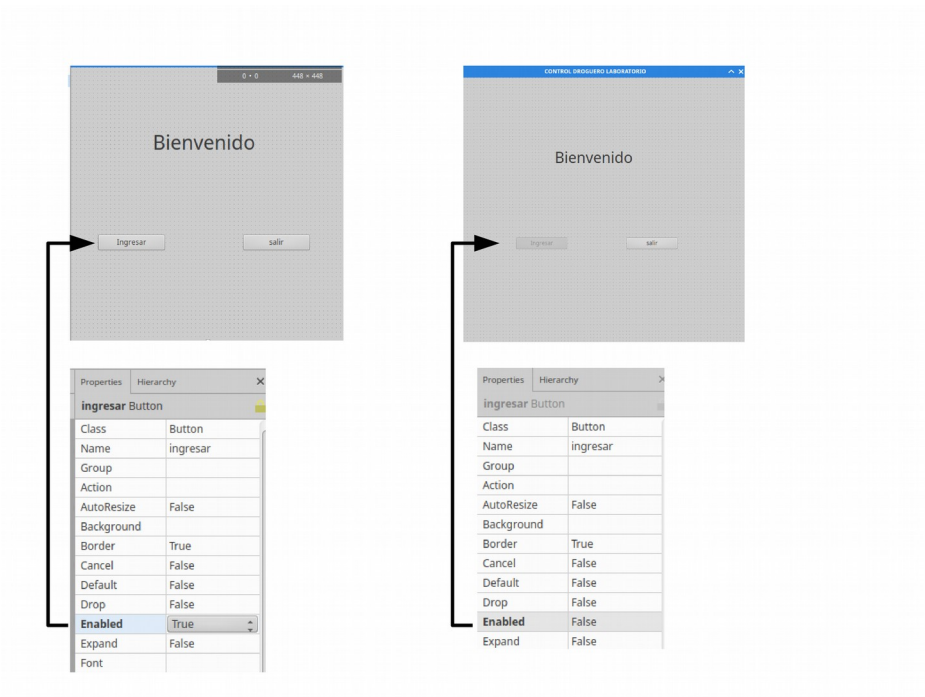


Figura 29

Mouse

La propiedad mouse de un objeto permite elegir diferentes formas del puntero para el mouse, que será adoptada por éste al pasar por el objeto. En nuestro ejemplo para el botón ingresar elegimos la

opción Arrow de la propiedad mouse. Como se puede ven en la Figura 30, el mouse tiene una forma común en cualquier otra parte del formulario pero toma el color rojo y de mayor tamaño al pasar sobre el botón ingresar. Se pueden elegir diferentes formas para diferentes objetos del formulario. De todas maneras más que estético es una cuestión de funcionamiento, indicando cada formato algo que deseamos tener en cuenta.

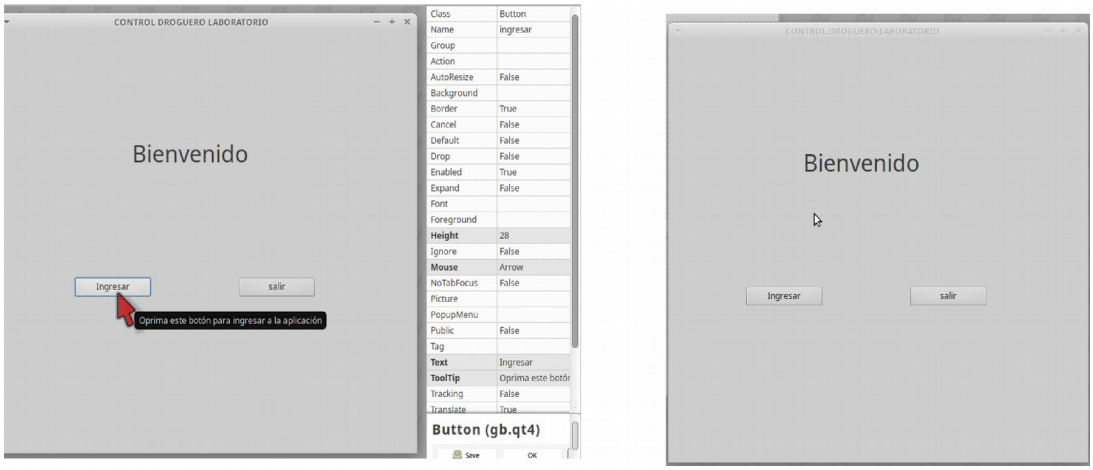


Figura 30

Cuadros de texto

Los cuadro de texto son objetos en los que podemos escribir en tiempo de ejecución de nuestro software. Dado que nuestro software tiene dos formularios, uno para ingresar la cantidad de cloruro de sodio comprado y otro para lo gastado, haremos el ejemplo con el formulario ingreso. Para colocar el cuadro de texto, nos posicionamos en el formulario y elegimos de la ventana herramienta a TextBox, hacemos click con el botón izquierdo y manteniendo apretado arrastramos la herramienta dentro del formulario, Figura 31. Una vez allí podemos modificar nombre del objeto y las demás propiedades como venimos viendo: text, Mouse, Tooltip, Enabled, y muchas otras que iremos viendo a lo largo del curso.

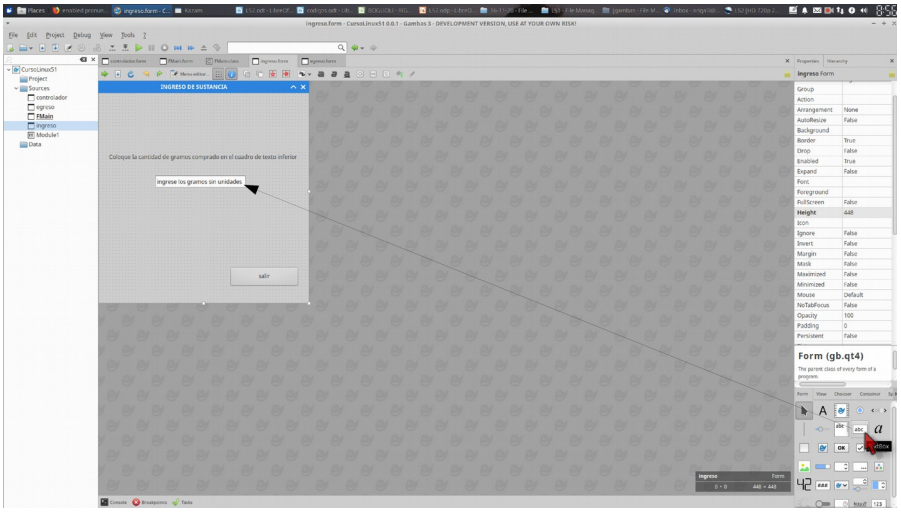


Figura 31

Definición de variable

Una variable es un objeto de nuestro software que podrá tomar valores a lo largo de la ejecución. Existen diferentes tipos de variables que se diferencian por el contenido, la memoria que consumen y las operaciones que podemos hacer. La lista siguiente muestra a título informativo las mismas

Name	Description	Memory size	Default value
`Boolean`	True or False	1 byte	False
`Byte`	0 ... 255	1 byte	0
`Short`	-32768 ... +32767	2 bytes	0
`Integer`	-2147483648 ... +2147483647	4 bytes	0
`Float`	Like the *double* datatype in *C*	8 bytes	0.0
`Date`	Date and time, each stored in an *integer* .	8 bytes	Null
`String`	A reference to a variable length string.	4 bytes	Null
`Variant`	Any datatype.	12 bytes	Null
`Object`	A anonymous reference to an object.	4 bytes	Null

Comenzaremos trabajando con algunas de ellas por ser las mas comunes. Las variables de tipos Integer, permiten manejar números enteros que como vemos pueden oscilar entre numeros negativos y positivos

-2147483648 ... +2147483647

El tamaño de memoria que ocupan es 4 bytes.

Si necesitaramos números que no excedan 32768, ya sea en el rango positivo o negativo nos convendría la variables de tipo Short, que aunque se superpone con Integer, el tamaño de memoria es de 2 bytes.

Las variables Byte solo abarcan números de 0 a 255, pero ocupa 1 byte.

A modo de ejemplo, si en nuestro software manejaramos edades de personas en años, sabemos que son enteros positivos y que rara vez supera los 100 años. Podríamos utilizar variables Byte, Short o Integer, pero la short sería de elección.

Cuando manejamos variables que tienen decimales, la variable a utilizar es de tipo Float, que como podemos ver utiliza memoria de 8 bytes.

Cuando declaramos una variable, en principio la podemos declarar para un evento de un objeto, para todo un formulario o para todo el software.

Definición para un evento

Supongamos que declaramos una variable que llamaremos compra, que alojará los gramos comprados de cloruro de sodio y esa variable se inicializará con el valor comprado que colocamos en el cuadro de texto con el valor ingresado.

Para declarar la variable en un evento de un objeto se utiliza la palabra DIM delante de la variable. En el caso de la Figura 32, el objeto en que definiremos la variable compra es Button1, es decir el botón con el rótulo "ingresar el dato" y la declaración la haremos dentro del evento doble click (DblClick). A su vez dentro del mismo evento inicializaremos la variable con el valor que introdujimos previamente en el TextBox que llamamos textoingreso.

Es importante ir distinguiendo que la variable solo existe para el software dentro de ese evento del objeto Button1 y no podrá ser utilizada en ninguna otra parte de la aplicación.

La Figura 32 muestra el formulario en tiempo de diseño, si hacemos click con el botón derecho sobre el botón Button1, elegimos Event y de allí DblClick se abre el class del formulario ingreso con los códigos y quedará posicionado el cursor del mouse en el objeto y evento deseado. Allí declaramos la variable como

```
Dim compra as Float
```

Float es el tipo de variable que hemos elegido para la variable compra.

Luego inicializamos compra con el valor del TextBox textoingreso con el siguiente código

```
compra = textoingreso.Text
```

y a continuación mostramos el valor de la variable compra en el Label2

```
Label2.Text = compra
```

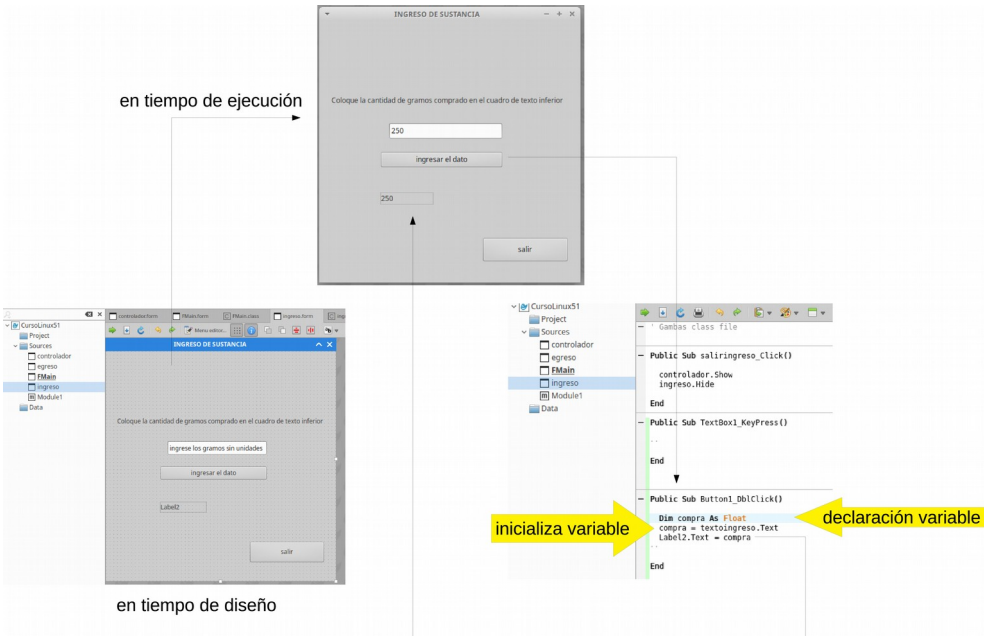


Figura 32

Definición para un formulario

La declaración de una variable para un formulario se hace en la ventana class del formulario y se coloca la declaración a continuación de 'Gambas class file'. A diferencia de la declaración de una variable para un evento que se hacia con DIM, en este caso se declara con PRIVATE. Por supuesto, si se define para un formulario no puede definirse nuevamente en un evento. Veremos entonces la misma variable "compra", pero para el formulario. En la Figura 33 se muestra la forma de declarar la variable para un formulario. Estando en el formulario ingreso, en tiempo de diseño, hacemos doble click sobre el formulario y se abrirá el formulario ingreso class, donde escribimos los códigos. La declaración de la variable compra se ha realizado al inicio del formulario y se utilizó PRIVATE. Por otra parte esa variable tomará el valor que esté en el TextBox que llamamos textoingreso, pero eso ocurrirá al hacer doble click sobre el botón Button1 que tiene el rótulo

"ingrese el dato". Simultáneamente al hacer doble click sobre Button1, en el Label2 se mostrará el valor.

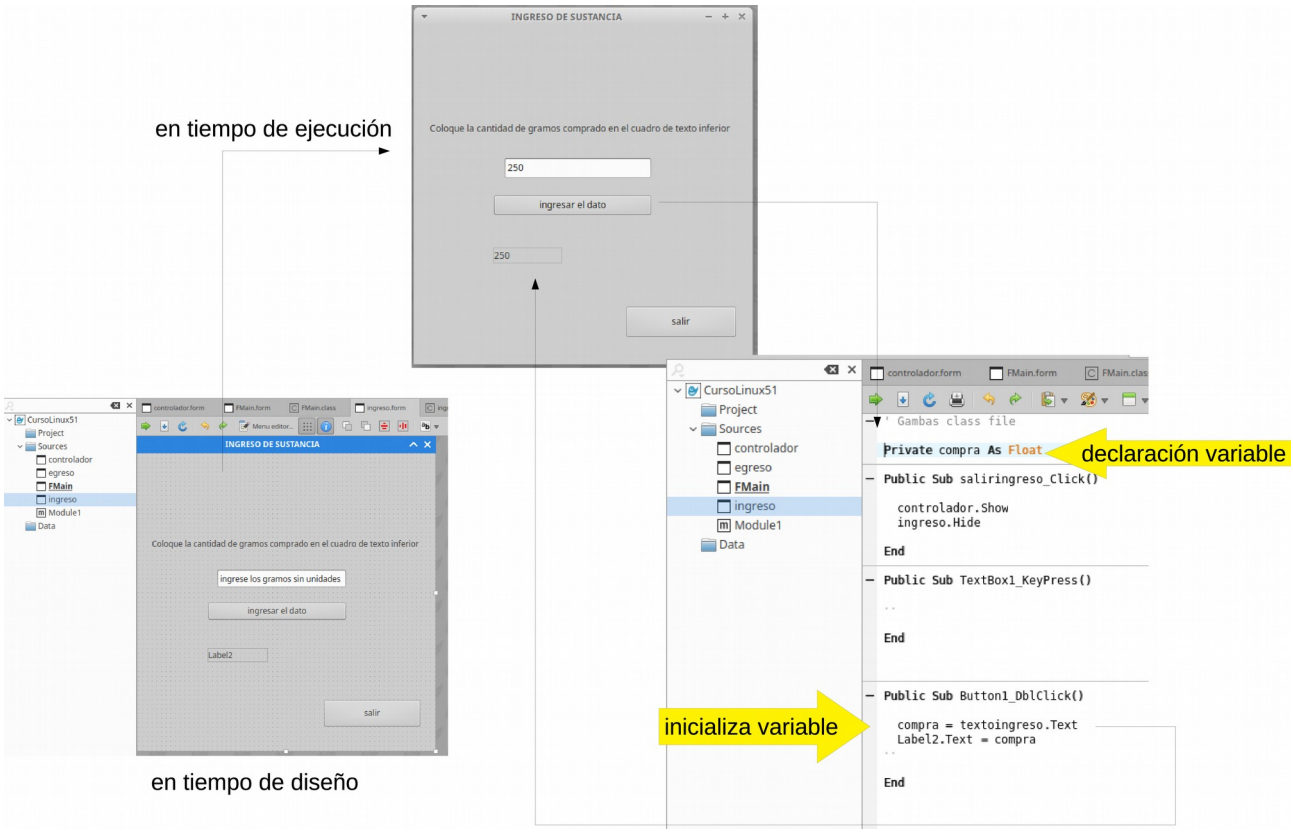


Figura 33

Si se deseara utilizar el valor de la variable compra, 250 en el ejemplo de la Figura 33, se podría hacer asociado a cualquier evento de cualquier objeto del formulario ingreso. Por supuesto no podrá utilizarse el valor en cualquiera de los otros formularios de la aplicación que estamos desarrollando. Para lograr esto deberíamos definir la variable de forma global que veremos próximamente.

Clase 3

Continuaremos en esta clase con el desarrollo de la aplicación titulada IngresoEgreso. El proyecto está formado por cuatro formularios y su función será registrar cuanta droga se gasta o se compra, pudiéndonos informar de la cantidad disponible. Los formularios son

controlador: formulario de acceso a los formularios ingresos y egresos

ingreso: formulario para registro de adquisición de la droga.

egreso: formulario de registro de gasto.

FMain: formulario de arraque de la aplicación.

Diseño de formularios

A menudo tenemos formularios parecidos por lo cual podemos acelerar el diseño copiando objetos entre ellos o bien importando formularios del mismo u otro proyecto. Veremos a continuación agregado de objetos y cambios de propiedades de ellos en tiempo de diseño.

Comentar líneas de código

Es una buena práctica de programación ir comentando los códigos desarrollados, ya que aunque sean claros en el momento de escribirlo, muchas veces deberemos destinar tiempo extra cuando volvamos sobre ellos con el fin de hacer alguna modificación.

Si se desea introducir un comentario en la class file se debe anteponer un apóstrofo (') a la línea de texto. Por ejemplo, luego de declarar una variable podemos colocar:

' la variable compra tomará el valor de la cantidad de gramos que adquirimos de cloruro de sodio en un día. Esta línea no será ejecutada y aparecerá en un color diferente al resto, Figura 34.

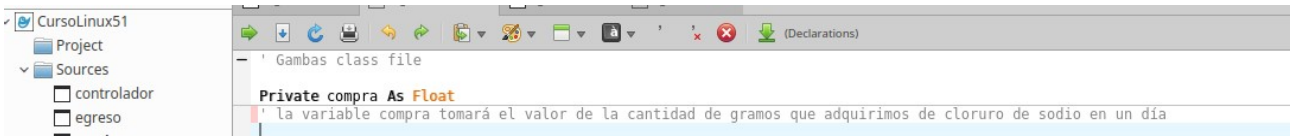


Figura 34

El apóstrofo también es útil para desactivar alguna línea de código que está funcionando mal y no hallamos el defecto. Al anteponer el apóstrofo a una línea de código, ésta no se ejecutará al correr la aplicación.

SetFocus

Esta propiedad de un objeto nos permite hacer foco sobre él, de manera que el el objeto quede preparado para una dada acción. Por ejemplo si en el formulario ingreso, en la acción Form_Open colocamos el código

```
textoingreso.SetFocus
```

cuando ingresemos al formulario durante la ejecución, el puntero del mouse quedará posicionado y titilando en el textBox donde debemos colocar la cantidad de droga comprada.

Timer

Aprovechando la introducción de SetFocus, veremos la utilización de un timer. Al ejecutar el formulario ingreso, con SetFocus, posicionamos el cursor en el TextBox: textoingreso. Con un timer podemos hacer muchas cosas pero para comprender su uso, lo que haremos es que una vez que

ejecutamos ingresar, deje unos segundos el texto, para saber que hacer y luego lo borre para que podamos escribir.

Para introducir un timer, nos dirigimos a la lista de herramientas y dentro del rubro Special, hallaremos el timer, identificado por un reloj: 

Como con cualquier herramienta, hacemos click sobre ella y lo arrastramos al formulario, pudiendo depositarla en cualquier lado ya que si bien la veremos en tiempo de diseño no aparecerá en tiempo de ejecución.

Haciendo doble click sobre él escribiremos la acción a desarrollarse cuando se active el timer. En nuestro caso queremos que el texto que figura en el TextBox de Name: textoingreso, desaparezca luego de 5 segundos. El timer se programa en milisegundos, por lo cual en la propiedad delay del objeto Timer1 colocaremos 5000, Figura 35.

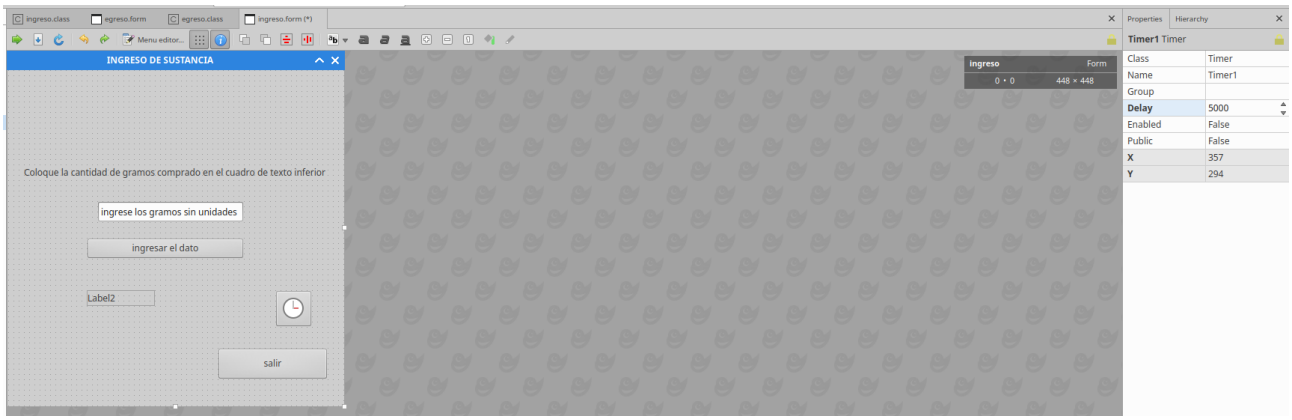


Figura 35

Así cuando se inicie el timer, luego de 5 segundos el texto que figura, desaparecerá y quedará listo para escribir. Por supuesto se podría hacer esto sin necesidad del uso del timer.

El timer se hallará inactivo, como muestra la propiedad Enabled= False. Por lo tanto habrá que indicar que se active el timer. Deseamos que se active al iniciarse el formulario, para ello hacemos doble click sobre el formulario y accederemos a la class file, en el evento

Form_Open()

allí colocaremos el código

Timer1.Star

Todas aquellas acciones que coloquemos dentro del formulario Form_Open se ejecutarán al cargarse el formulario. Si deseamos que se active con cualquier otro evento, colocaremos ese código en otro objeto o evento, Figura 36.

Ahora debemos programar el timer. Haciendo doble click sobre el timer accedemos

Timer1-Timer()

y allí colocaremos lo que deseamos que ocurra que será borrar el texto de textoingreso

textoingreso.text = ""

Al colocar la propiedad text con dos comillas, indica que no tendrá texto, Figura 36.

```
Public Sub Form_Open()

  textoingreso.SetFocus
  Timer1.start

End

Public Sub Timer1_Timer()

  textoingreso.text = ""

End
```

Figura 36

Introducción de imágenes

Para introducir una imagen elegimos el objeto PictureBox de la barra de herramientas y la arrastramos al formulario, Figura 37. Si bien veremos otros usos, en este caso solo la utilizaremos para mostrar una imagen y hacer más atractivo nuestro software. Veremos más adelante una utilización más fructífera de estos objetos.

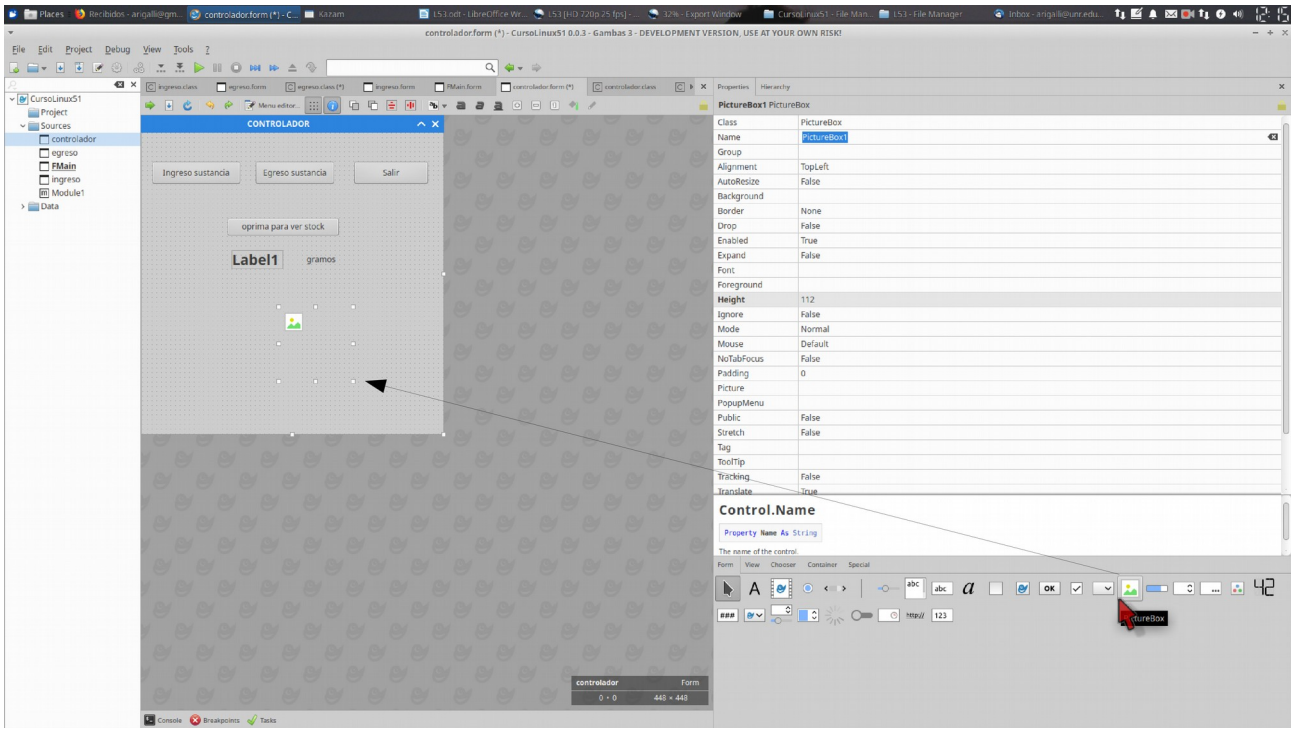


Figura 37

Una vez el PictureBox en el formulario hacemos click sobre él y de la lista de propiedades elegimos Picture, hacemos click sobre los tres puntos que se hallan a la derecha de la línea (...) y elegimos la imagen. Así quedará incluida en nuestro formulario. Esta imagen será estática y solo decorativa, por

el momento, Figura 38. Es importante para que muestre la imagen completa que la propiedad Stretch se halle en el valor True. Cuando arranquemos el software tendremos la siguiente presentación

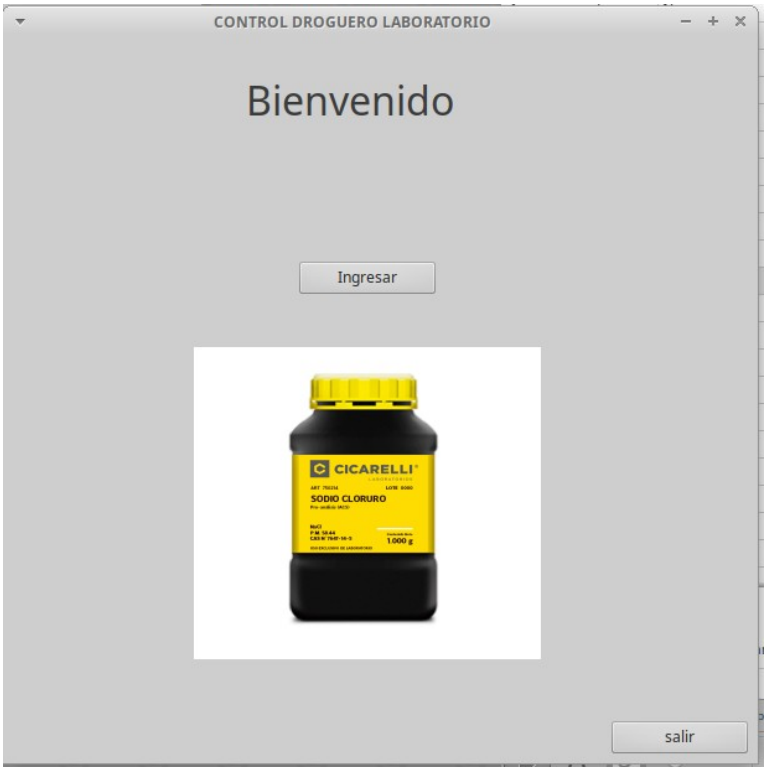


Figura 38

Declaración de variables

Una variable es un objeto de nuestro software que podrá tomar valores a lo largo de la ejecución. Como hemos visto y recordamos existen diferentes tipos de variables que se diferencian por el contenido, la memoria que consumen y las operaciones que podemos hacer. La lista siguiente muestra a título informativo las mismas

Name	Description	Memory size	Default value
`Boolean`	True or False	1 byte	False
`Byte`	0 ... 255	1 byte	0
`Short`	-32768 ... +32767	2 bytes	0
`Integer`	-2147483648 ... +2147483647	4 bytes	0
`Float`	Like the *double* datatype in *C*	8 bytes	0.0
`Date`	Date and time, each stored in an *integer* .	8 bytes	Null
`String`	A reference to a variable length string.	4 bytes	Null
`Variant`	Any datatype.	12 bytes	Null

`Object`	A anonymous reference to an object.	4 bytes	Null

Las variables se deben declarar, es decir asignar un nombre y un tipo. Esta acción puede llevarse dentro de diferentes partes del software.

Declaración dentro de un evento

Si la variable se define dentro de un evento de un objeto, por ejemplo al hacer click sobre un botón. Esta variable existirá para el software solo dentro de ese evento y objeto. Cualquier acción (click, dobleclick, getfocus, etc) en otro objeto o inclusive una acción sobre el mismo botón pero que no sea en click, será ignorada por la aplicación. Estas variables se declaran como

DIM nombreDeLaVariable as TipoDeVariable

Este tema ya fue estudiado en clases anteriores.

Declaración para todo un formulario

Si la variable se define para todo un formulario, se debe declarar en las primeras líneas del class file del formulario y es válida para cualquier objeto o evento de éstos dentro del formulario en cuestión, con el evento que se elija (click, dobleclick, etc). Estas variables se declaran con

Private nombreDeLaVariable as TipoDeVariable

Este tema fue desarrollado en clases anteriores.

Recordando, tenemos ya declarada una variable ***compra*** dentro de formulario ingreso y toma el valor de la cantidad de cloruro de sodio que adquirimos en un día. Por otro lado declaramos la variable ***gasto*** dentro del formulario egreso que registrará el gasto de cloruro de sodio. Estas variable la definimos para todo el formulario , por lo tanto la declaramos dentro de sus formularios respectivos con

Private compra as Float

Private gasto as Float

Declaración de variables para más de un formulario

Si necesitamos declarar una variable que tome valores en más de un formulario, la definiremos dentro de un formulario en particular, en función de nuestra preferencia. Para nuestro caso declararemos una variable que llamaremos ***stock*** que llevará como valor la diferencia entre la variable ***compra*** (declarada y utilizada en el formulario ingreso) y la variable ***gasto*** (declarada y utilizada en el formulario egreso)

Declaramos la variable stock en el formulario controlador, a continuación de gambas Class file. Para la declaración de este tipo de variables utilizamos la palabra Public. Como forma general para definir estas variables se utiliza

Public nombreDeLaVariable as TipoDeVariable.

en nuestro caso ser, Figura 39

Public stock as Float

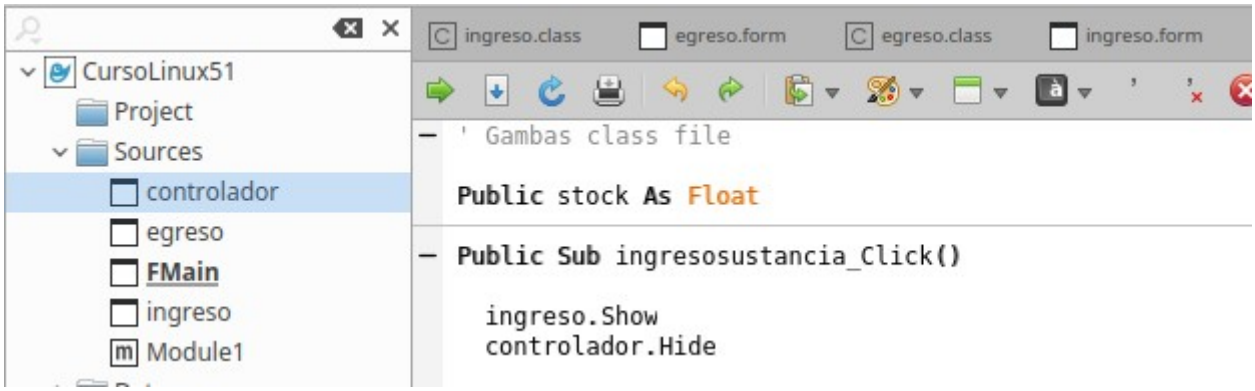


Figura 39

Esa variable que tomará la sumatoria de compra y gasto, la mostraremos en el formulario controlador con un botón y en un label, Figura 40. El botón lleva el texto "oprima para ver stock" y al oprimirlo en el label1 mostrará la cantidad de gramos del stock

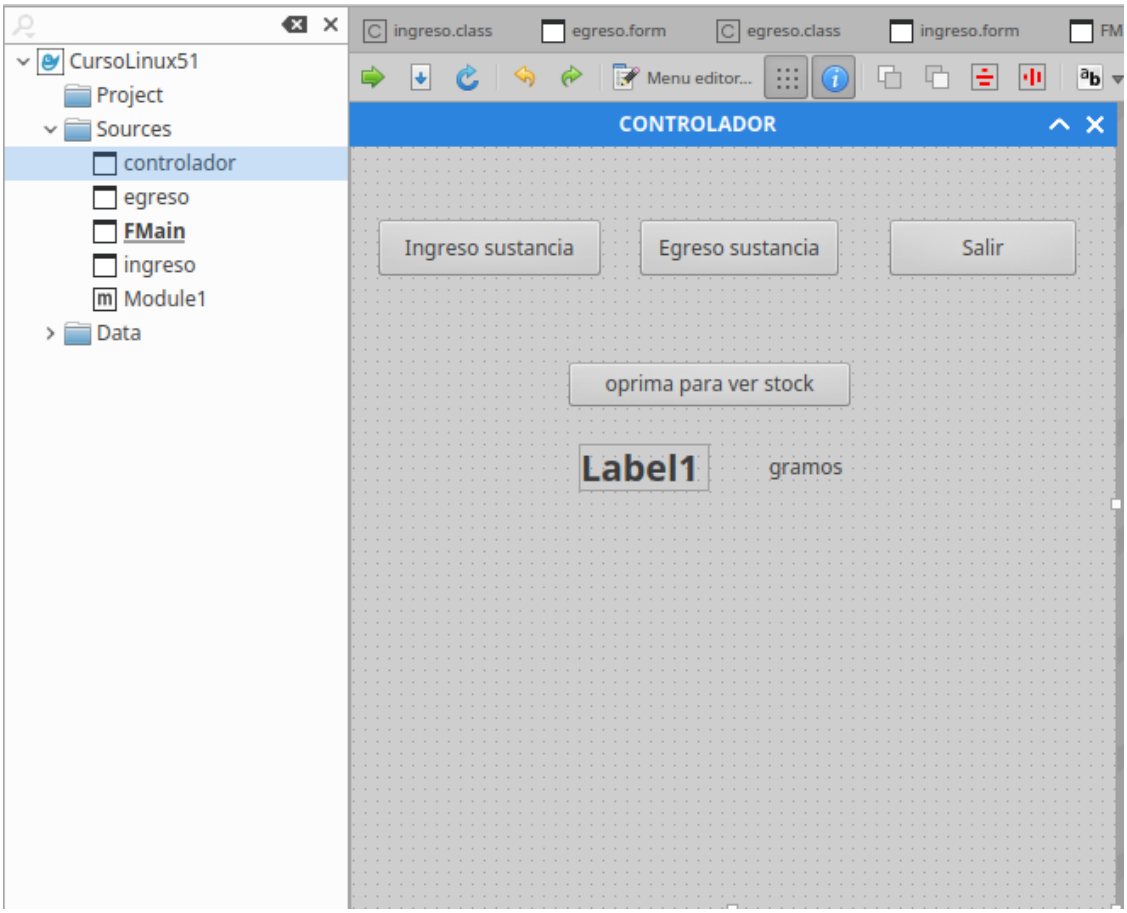


Figura 40

Veamos ahora como programamos el botón, para que muestre los gramos del stock. Cuando se recurre a una variable global como lo es ***stock***, en lugar de colocar solo el nombre de la variable, se le antepone el nombre del formulario donde fue declarada. En este caso, estamos trabajando en el mismo formulario en que fue declarada. Entonces al label1, en su propiedad Text, colocamos el valor que tenga la variable stock con el siguiente código, Figura 41

Label1.text= controlador.stock

```
Public Sub Form_Open()
..
End

Public Sub mostrarstock_Click()
  Label1.text = controlador.stock
End
```

Figura 41

Así al ejecutar nuestro programa y oprimir el botón "oprimir para ver stock" tendremos la presentación de la Figura 42, indicándonos que tenemos 500 g en stock.

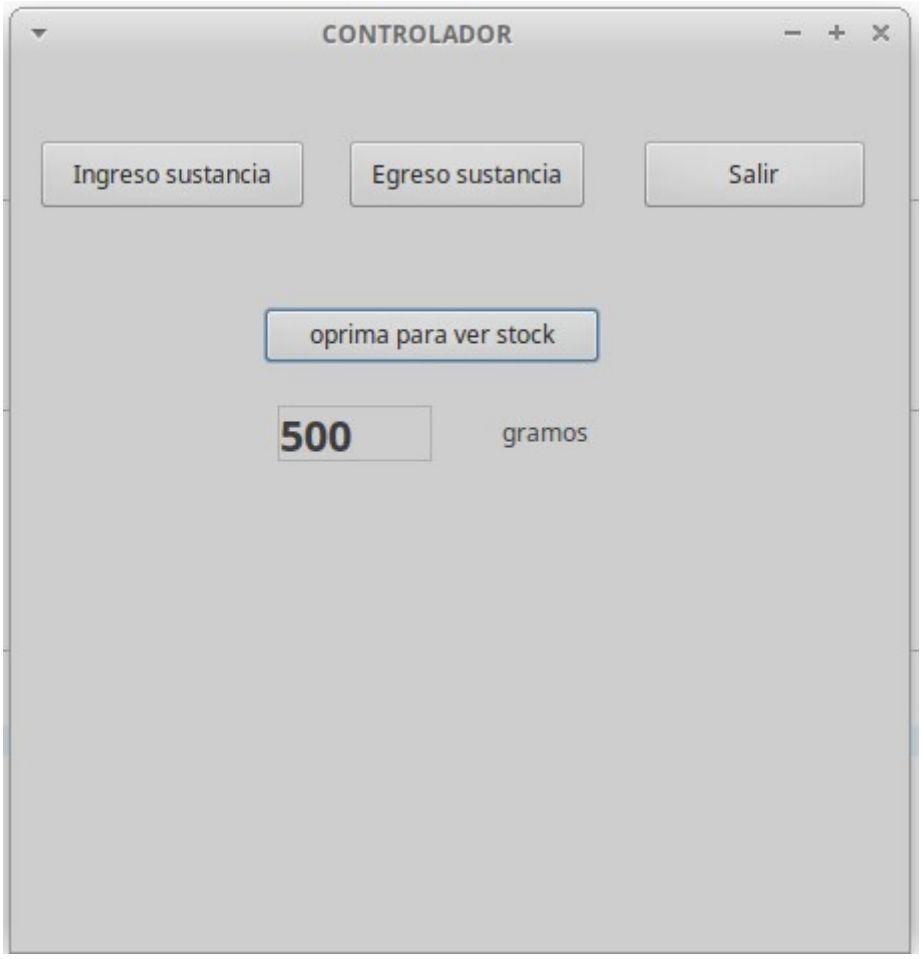


Figura 42

Veremos ahora como vincular la variable ***stock*** con las variables ***compra*** y ***gasto***. Cuando realicemos una compra, el valor de gramos comprados deberá sumarse a stock y esto lo haremos desde el formulario ingreso. Esta acción deberá ocurrir cuando colocamos los gramos comprados y oprimimos el boton "ingresar el dato". Esto lo logramos con el código de la Figura 43

```
Public Sub Button1_Click()

  compra = textoingreso.Text
  controlador.stock = controlador.stock + compra

End
```

Figura 43

Como vemos en la primer linea asignamos a la variable compra el valor del TextBox llamado textoingreso. Luego ese valor de la variable compra se suma al valor que almacena la variable stock. Como está variable la hemos declarada en el formulario controlador, se utiliza controlador.stock para invocarla. Veremos más adelante que este proceso podría optimizarse, con códigos más sencillos, pero no es en este momento el objetivo de la clase.

Lo mismo haremos con la variable gasto del formulario egreso, salvo que en este caso, la variable gasto se restará a stock.

Control de flujo: estructura If Then Else

Un detalle importante que tenemos que tener en cuenta es que no podemos gastar más droga que la que tenemos, por lo que introduciremos un código de control de flujo. Este código realizará la resta si el valor de gasto es menor o igual al valor de la variable stock, pero nos avisará que no se puede retirar la cantidad de droga si el valor de la variable gasto es mayor que el valor de la variable stock. Introducimos entonces en el formulario egreso un label2, donde nos aparecerá el mensaje que no tenemos la cantidad suficiente, en el caso que se cumpla la segunda condición Figura 44.

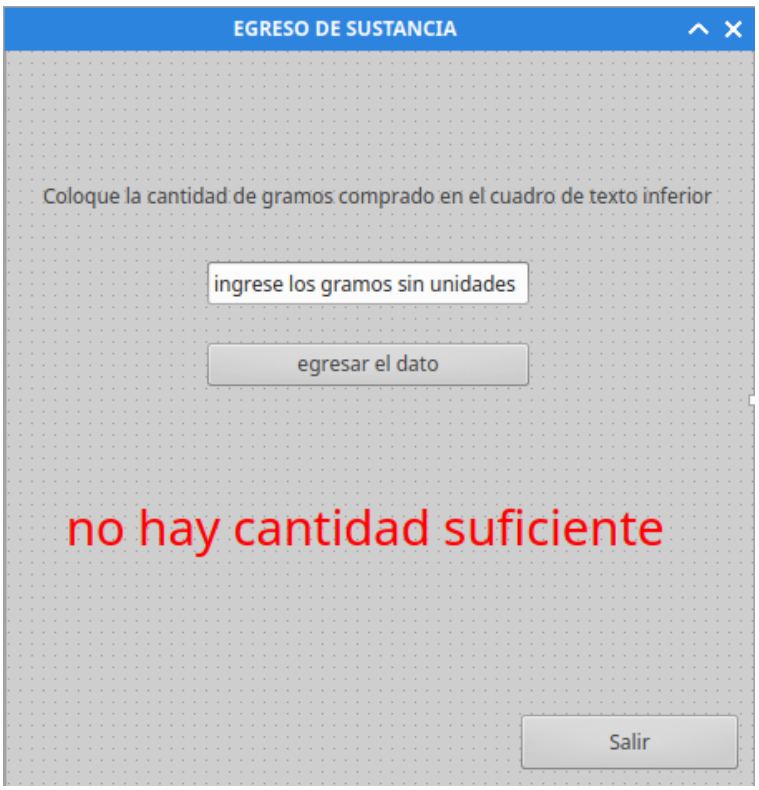


Figura 44

En el evento click del botón con el texto "egresar el dato" tendremos el siguiente código, Figura 45

```
Public Sub Button1_Click()

  gasto = textoegreso.Text
  If gasto < controlador.stock Then
  controlador.stock = controlador.stock - gasto
  Else
  Label2.visible = True
  Label2.text = "No hay cantidad suficiente"
  Endif
..
..
..
..

End
```

Figura 45

Como podemos ver al hacer click en el botón la variable gasto toma el valor del TextBox: textoegreso. Luego se evalúa si el valor de la variable gasto < stock. De cumplirse esta condición, resta al valor de stock el valor egresado, que se halla en la variable gasto. En cambio si gasto > stock, hará visible el label2 y mostrará el texto "No hay cantidad suficiente".

Creación de archivo ejecutable

Hasta ahora hemos corrido nuestra aplicación desde Gambas. Ahora construiremos un programa ejecutable que no necesita la aplicación Gambas. Para hacer la aplicación, estando en el software en desarrollo, oprimimos de la barra de herramientas el ícono "Make executable": 

Al oprimir el ícono, se abre una ventana donde pondremos el nombre de nuestra aplicación como muestra la Figura 46. Las aplicaciones llevan un nombre seguido de .gambas

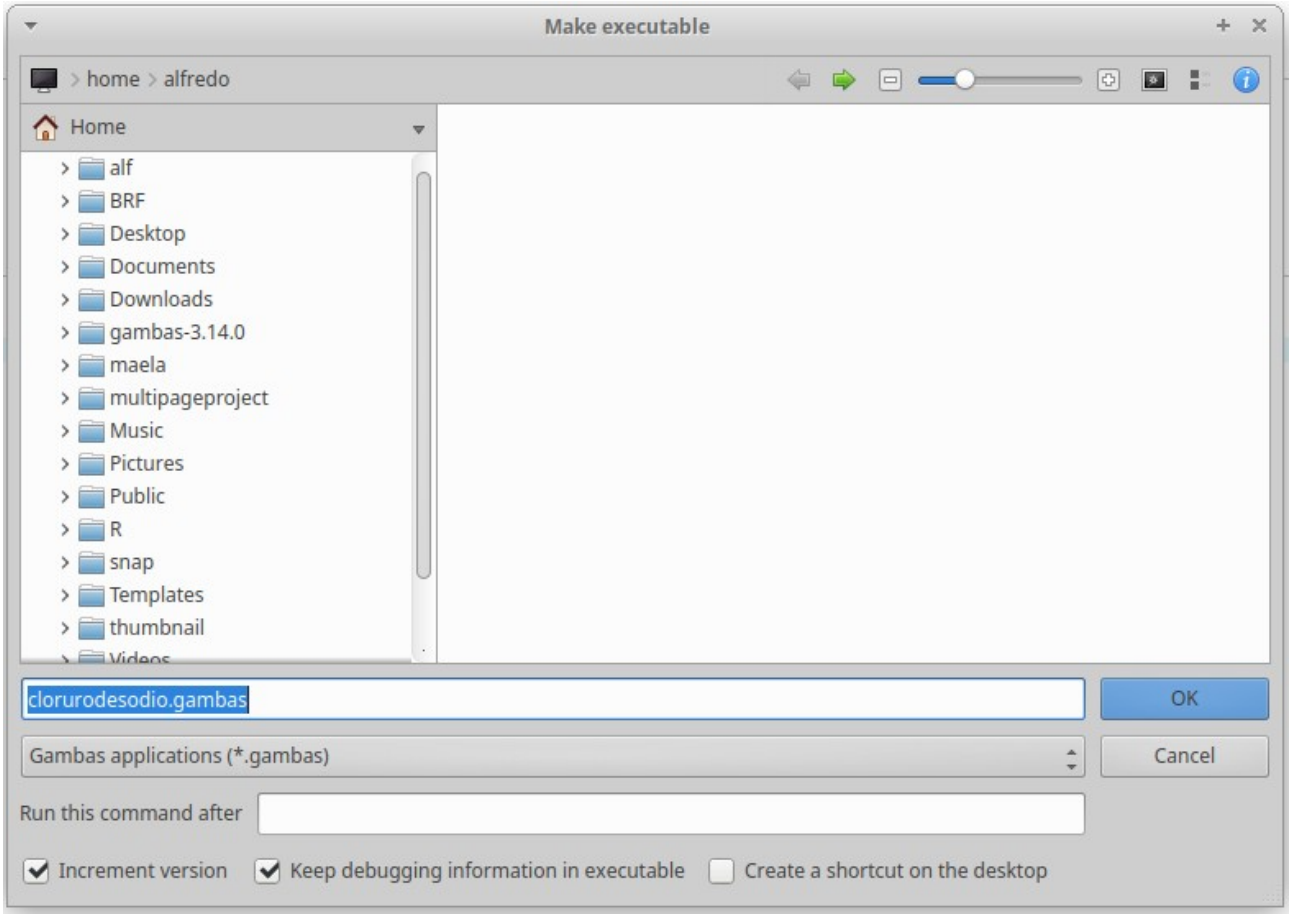


Figura 46

Asó se creará un archivo, para este caso llamado clorurodesodio.gambas, que es nuestra aplicación y que podremos colocar en cualquier carpeta de la computadora, Figura 47. Al hacer doble click sobre el arrancará la aplicación diseñada

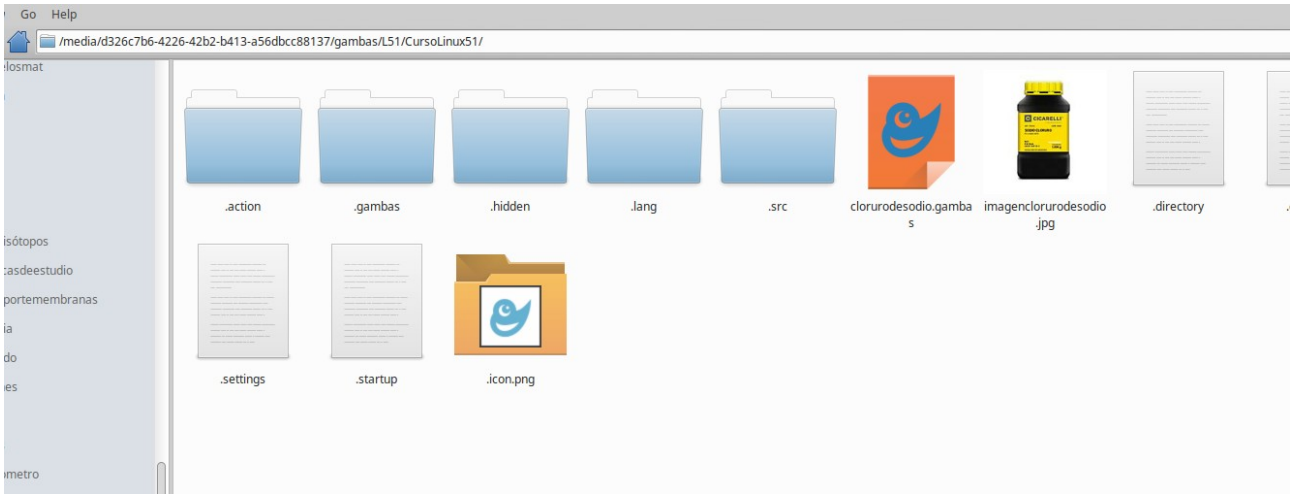

Go Help
/media/d326c7b6-4226-42b2-b413-a56dbcc88137/gambas/L51/CursoLinux51/
.action
.gambas
.hidden
.lang
.src
clorurodesodio.gambas
imagenclorurodesodio.jpg
.directory
.settings
.startup
.icon.png

Figura 47

Clase 4

En esta clase iniciaremos una nueva aplicación gráfica y asignaremos al proyecto el nombre "CursoLinux52" y el título de la aplicación será "preguntasrespuestas". Se trata de una aplicación que me permitirá evaluar conceptos de un área del conocimiento a través de preguntas sugeridas y ns servirá para introducir nuevos objetos. En esta y en las sucesivas clases se introducirán los nuevos elementos de programación no describiendose en el texto, aunque si en el vídeo, el funcionamiento y propiedades de objetos conocidos

RadioButton

El radioButton es un objeto que puede tener dos valores True o False y cuyo valor podemos verlo en la aplicación porque al estar en True, el radioButton tiene un círculo en su interior y si está en valor False se muestra vacío. Si en un dado espacio de un Form tenemos varios RadioButton, no podrán estar todos en False y solo podrá haber uno en el valor True, Figura 48.

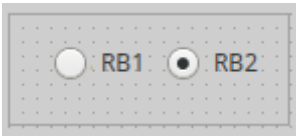


Figura 48

Son muy útiles cuando tenemos que marcar opciones que son mutuamente excluyentes, como las de nuestra aplicación: elegir Verdadero o Falso respecto de una proposición.

El valor True o False, es la propiedad Value, que puede fijarse desde la pestaña properties. En la Figura 49 tenemos marcado un RadioButton del Form preguntasglucidos y en las propiedades hemos fijado Value en True. Por esta razón el mismo tiene en su interior un punto.

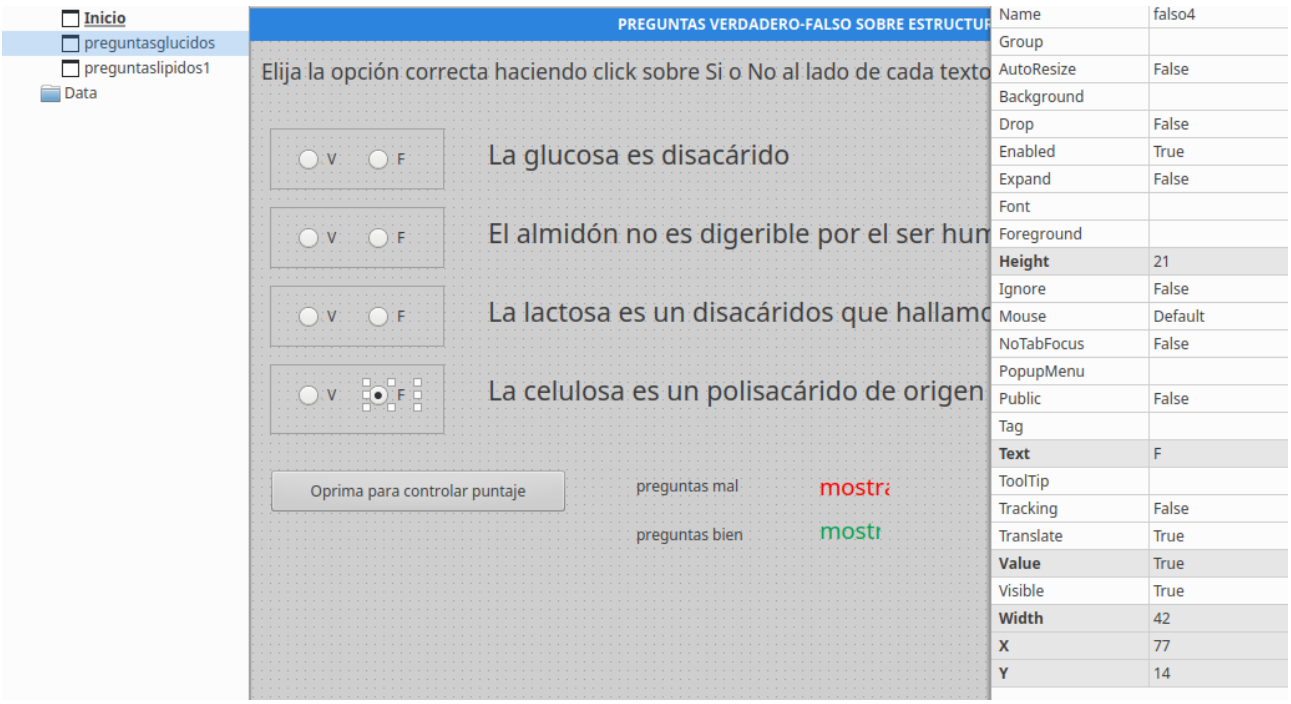


Figura 49

Una opción muy util es fijar sus valores de arranque cuando ejecutemos la aplicación. Es común que cuando tenemos que fijar valores de alguno objeto al arrancar un formulario, dicha orden la coloquemos dentro del evento Open del Form que contiene el objeto.

En el Form de la Figura 49, tenemos 8 RadioButton. Si los ocho estuvieran en el Form, solo uno podría tomar el valor True, pero nosotros necesitamos que de a pares podamos elegir en función del valor de verdad de la proposición teórica que estamos analizando. Para ello introducimos un nuevo objeto que es el contenedor y que discutiremos inmediatamente a continuación de este ítem.

Como deseamos que la aplicación al arrancar muestre la opción verdadero (V) y falso (F) sin marcar, dentro del mismo contenedor colocamos otro RadioButton, que lo dejaremos en valor True y lo colocaremos fuera del área que mostrará el contenedor. En la Figura 50, se muestra el cuarto contenedor de arriba hacia abajo en un tamaño que no se mostrará en tiempo de ejecución. El contenedor lo ajustamos luego al valor deseado y lo mismo hacemos con los cuatro contenedores.

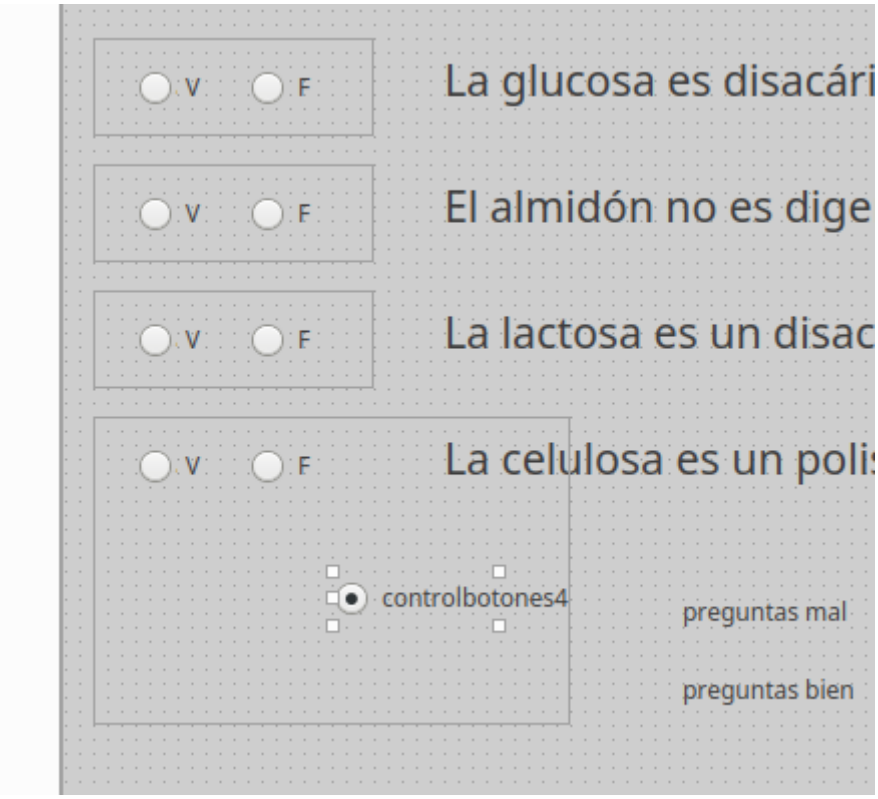


Figura 50

Contenedores

Los contenedores son marcos que alojan objetos y que no se pueden mover libremente dentro de todo el formulario sino dentro de ese marco. Son muy útiles para alojar objetos del tipo RadioButton como hemos visto recientemente.

Su uso exige que primero coloquemos el contenedor en el formulario y luego dentro de él depositemos los radioButton, como muestra la Figura 50. En ese caso hay 4 contenedores y cada uno de ellos con 3 RadioButton.

A un contenedor como los utilizados en la Figura 50, conocidos como objetos de la clase Panel, podemos asociar eventos como a cualquier objeto. En la Figura 51, vemos marcado el Panel4 del Form preguntasglucidos y desplegado con el botón derecho del mouse los eventos que podríamos asociar.

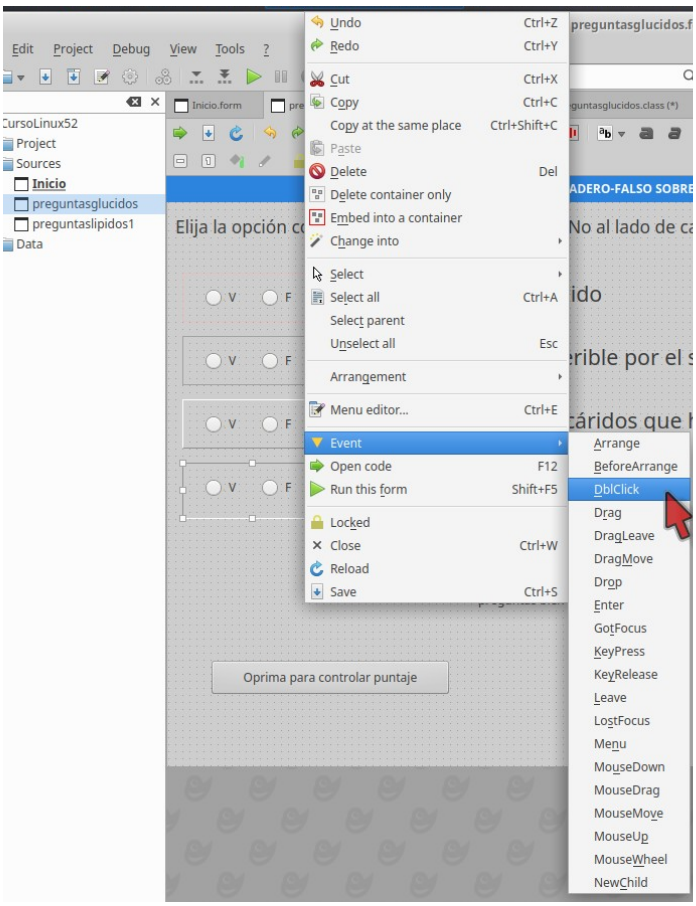


Figura 51

Una propiedad sencilla pero útil para mostrar nuestras aplicaciones, en el caso de los Panel es la propiedad Border. Existen diversas opciones que se fijan desde la pestaña Properties en tiempo de diseño o bien por código en tiempo de ejecución. En la Figura 51 se muestran cuatro opciones para el borde el contenedor, cuyo objetivo es solo estético.

Fijar propiedades de objetos en tiempo de diseño o ejecución

Como hemos visto los objetos tienen propiedades que se pueden fijar desde la pestaña properties. Cuando fijamos un o más propiedades desde este sitio decimos que "estamos fijando sus propiedades en tiempo de diseño" ya que lo hacemos mientras "diseñamos" el software. Pero las propiedades pueden modificarse en tiempo de corrida.

En nuestra aplicación hemos construido un formulario que tiene preguntas sugeridas del tipo verdadero o falso. Luego de leer cada proposición, el usuario deberá elegir la opción V o F, pudiendo ser acertada o no su respuesta. Si eligió la opción correcta el software la considera correcta y suma una unidad a una variable local del Form a la que llamamos contadorbien, definida al inicio del formulario como un entero y si eligió la opción incorrecta será contada como una pregunta mal y sumará una unidad a la variable contadormal, declarada en el mismo sitio, Figura 52.

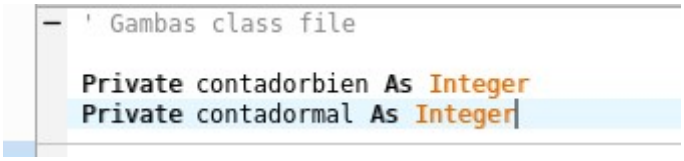


Figura 52

La primera proposición "La glucosa es un disacárido" es conceptualmente falsa. Por lo tanto, el usuario, supuestamente un estudiante de química biológica deberá decidir si marcar el botón V o F, cuyos nombres son verdadero1 y falso1 respectivamente. Si marca el radioButton verdadero1, claramente estará mal y si marca falso1 estará bien.

Nuestro formulario tiene un objeto de tipo **Button** de Name **correccion** y que tiene el texto: **oprima para controlar puntaje.** En el evento Click del button correcion tenemos el código de la Figura 53. Podemos observar que al hacer Click sobre el botón correccion se ejecutan numerosas instrucciones. En primer lugar vemos estructura del tipo If Then Else. Analicemos la primera. Dijimos que la primer proposición, cuyo nombre es proposicion1 es falsa. Entonces la estructura If Then Else analiza si el radiobutton verdero1 está en valor False (que sería la respuesta correcta), ya que si éste está en False implica que el radiobutton falso1 esta en True. Entonces si verdadero1 es falso, suma un punto a la variable contadorbien, que es una variable que irá contando nuestros aciertos en este formulario y simultáneamente a la proposicion1 de cambia el valor de la propiedad Foreground, que es el color. En este caso el código indica que el texto se pondrá de color verde. En el vídeo puede ver una forma sencilla de conocer estos códigos de colores. Ahora, si verdadero1 en lugar de estar en su valor False, que es la respuesta correcta, estuviera en valor True. La estructura If then Else, pasaría a la parte Else y sumaría una unidad al contadormal que lleva la cuenta de nuestros errores y a la proposicion1 le cambia el color a rojo.

En este caso estamos haciendo una cambio de propiedades en tiempo de corrida.

```
Public Sub correccion_Click()

  If verdadero1.value = False Then
    contadorbien = contadorbien + 1
    proposicion1.Foreground = &H0000A04D&

    Else
    contadormal = contadormal + 1
    proposicion1.Foreground = &HFF0000&
    explicacion1.text = "La glucosa es monosacárido"
  Endif

  If verdadero2.value = False Then
    contadorbien = contadorbien + 1
     proposicion2.Foreground = &H0000A04D&
    Else
    contadormal = contadormal + 1
    proposicion2.Foreground = &HFF0000&
    explicacion2.text = "Existen enzimas en humanos que hidrolizan el almidón"
  Endif

    If verdadero3.value = True Then
    contadorbien = contadorbien + 1
     proposicion3.Foreground = &H0000A04D&
    Else
    contadormal = contadormal + 1
    proposicion3.Foreground = &HFF0000&
    explicacion3.text = "Efectivamente la lactosa está formada por glucosa y galactosa y se halla en varios lacteos"
  Endif

     If verdadero4.value = True Then
    contadorbien = contadorbien + 1
     proposicion4.Foreground = &H0000A04D&
    Else
    contadormal = contadormal + 1
    proposicion4.Foreground = &HFF0000&
    explicacion4.text = "la celulosa es un polisacarido de sostén vegetal"
  Endif

  mostrarmal.Text = contadormal
  mostrarbien.text = contadorbien

  Inicio.contadorgeneralbien = Inicio.contadorgeneralbien + contadorbien
  Inicio.contadorgeneralmal = Inicio.contadorgeneralmal + contadormal
```

Figura 53

Por otra parte vemos también que en tiempo de corrida cambiamos la propiedad text del objeto explicacion1, asignándole una explicación: "La glucosa es monosacárido". El cambio de la propiedad text del objeto explicacion1 solo se produce si el alumno eligió la opción incorrecta.

Cualquier propiedad de cualquier objeto puede modificarse tanto en tiempo de diseño, como en tiempo de ejecución. La fijación en tiempo de diseño es la que nos permite fijar **COMO** se verá el formulario en el momento que se ejecute la acción Show.

En el proyecto que estamos diseñando hemos modificado en tiempo de ejecución la propiedad text de algunos Labels, Figura 54. En este caso vemos que asignamos a la propiedad text del Label1 un valor que resulta de unir un string ("PREGUNTAS BIEN: ") con el valor que toma una variable global del proyecto: contadorgeneralbien, que está declarada en el Form Inicio

```
Public Sub Form_Show()

Label1.text = "PREGUNTAS BIEN:   " & Inicio.contadorgeneralbien
Label2.text = "PREGUNTAS MAL:   " & Inicio.contadorgeneralmal

End
```

Figura 54

Ambas estructuras se separan por el operador &. Es importante recordar que las cadenas de caracteres o strings, deben ir entre comillas y las variables sin ellas. Asi, si la variable contadorgeneralbien tuviera el valor 5, cuando se ejecute el evento Show del Formulario en cuestión, en el label1 se verá el texto: PREGUNTAS BIEN: 5.

Startup class

Cuando ejecutamos la aplicación ya sea durante nuestro trabajo de diseño directamente desde Gambas o bien ya desde nuestra aplicación, arrancará un formulario, el que se llama Startup class. Al comenzar a diseñar un software tenemos siempre un formulario que se llama Fmain y que es el Startup class por defecto. Pero podemos borrar este formulario o bien asignar a otro formulario ese rol. Para saber cual es el Startup class, tenemos que mirar el listado de formularios y el que se halla subrayado lo es. Para nuestro caso es el Form Inicio, Figura 55.

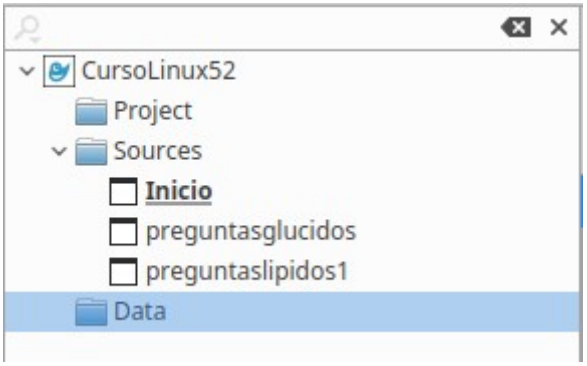


Figura 55

Si deseamos cambiar el startup Class, hacemos click derecho sobre el form de interés y marcamos Startup Class, Figura 56.

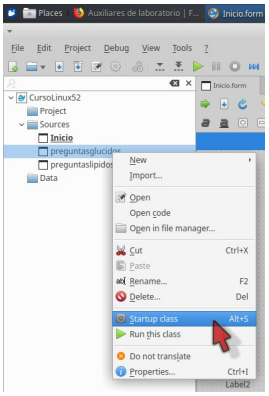


Figura 56

Por supuesto, puede ocurrir que no funcione correctamente nuestro software, ya que quizás algunas variables globales se declararon en otros formularios que en condiciones normales enfrentamos primero y ahora no lo haremos, al haber cambiado el orden de inicio.

Importar formulario

Al diseñar nuestro software, muchas veces tenemos formularios que tienen el mismo formato. Tal es el caso de la aplicacion que estamos diseñando. Tenemos un Form que tiene preguntas sobre glúcidos y luego tendremos otros de igual diseño donde cambiarán solo las proposiciones y su

grado de verdad. Entonces nos conviene utilizar el mismo formulario. Para ello nos posicionamos con el mouse en Source e Import, y se abrirá una ventana de archivos. Buscamos nuestro proyecto, y dentro de la carpeta .src hallaremos los forms de la aplicación. Marcamos el Form deseado y oprimimos OK, quedando incluido en la lista de Forms del proyecto, Figura 57.

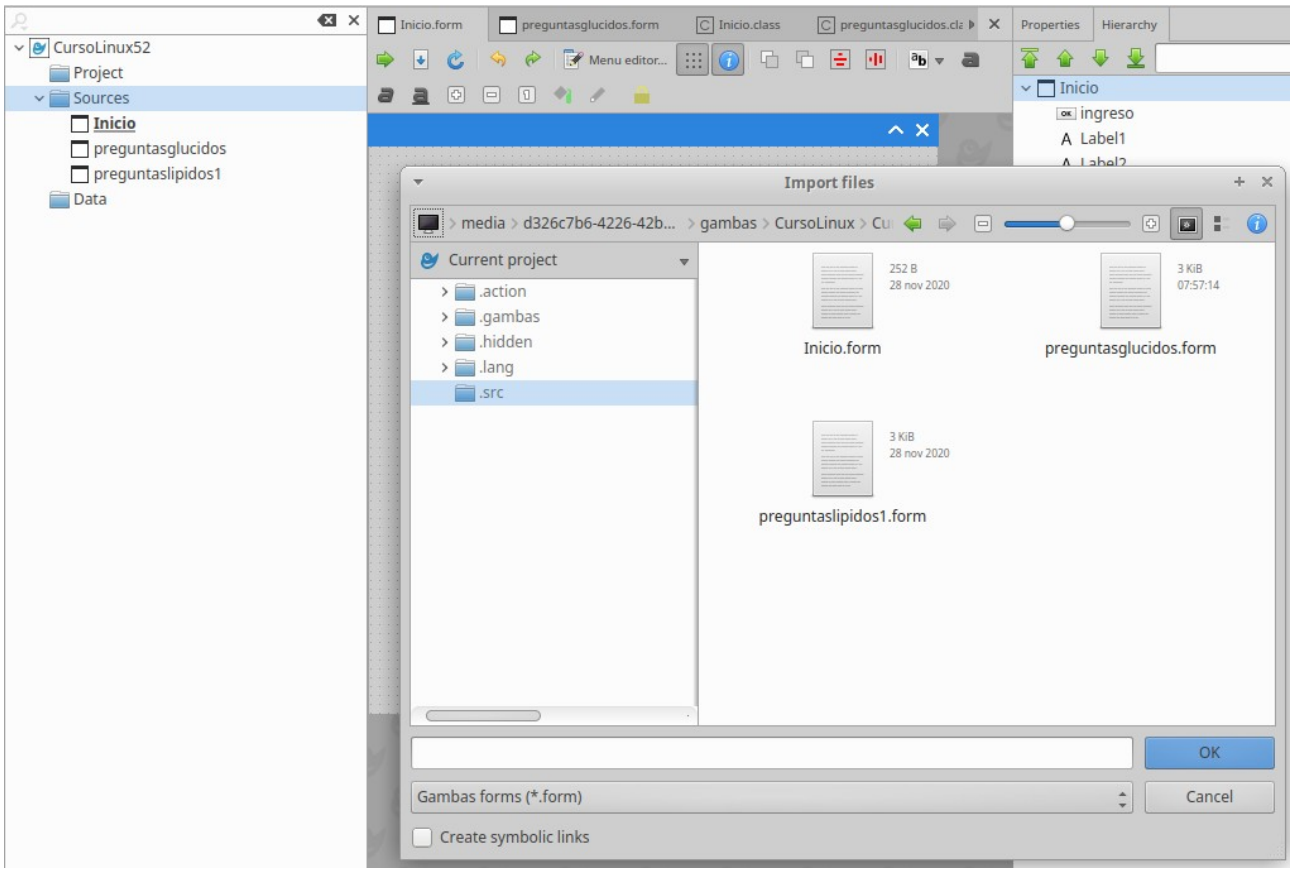


Figura 57

Clase 5

En esta clase iniciaremos una nueva aplicación gráfica, utilizando como base la aplicación CursoLinux52 y asignaremos al proyecto el nombre "CursoLinux55" y el título de la aplicación será "PreguntasQuimica". Se trata de una aplicación que me permitirá evaluar conceptos de un área del conocimiento a través de preguntas sugeridas y ns servirá para introducir nuevos objetos. En esta y en las sucesivas clases se introducirán los nuevos elementos de programación.

ComboBox

Los objetos ComboBox son listados que los se pueden desplegar y hacer elecciones. En primer lugar veamos como cargar los ítems del listado. Lo haremos a través de código y que la carga se haga en tiempo de ejecución, cuando el formulario que lo contiene se abra. Para incorporar un objeto a un formulario, como en todos los casos, lo arrastramos desde la barra de herramienta y configuraremos algunas cosas básicas desde Properties.

La propiedad Sorted la colocamos en el valor TRUE, de esta manera la lista mostrará los ítems en orden alfabético, independiente de como fueron introducidos, Figura 58.

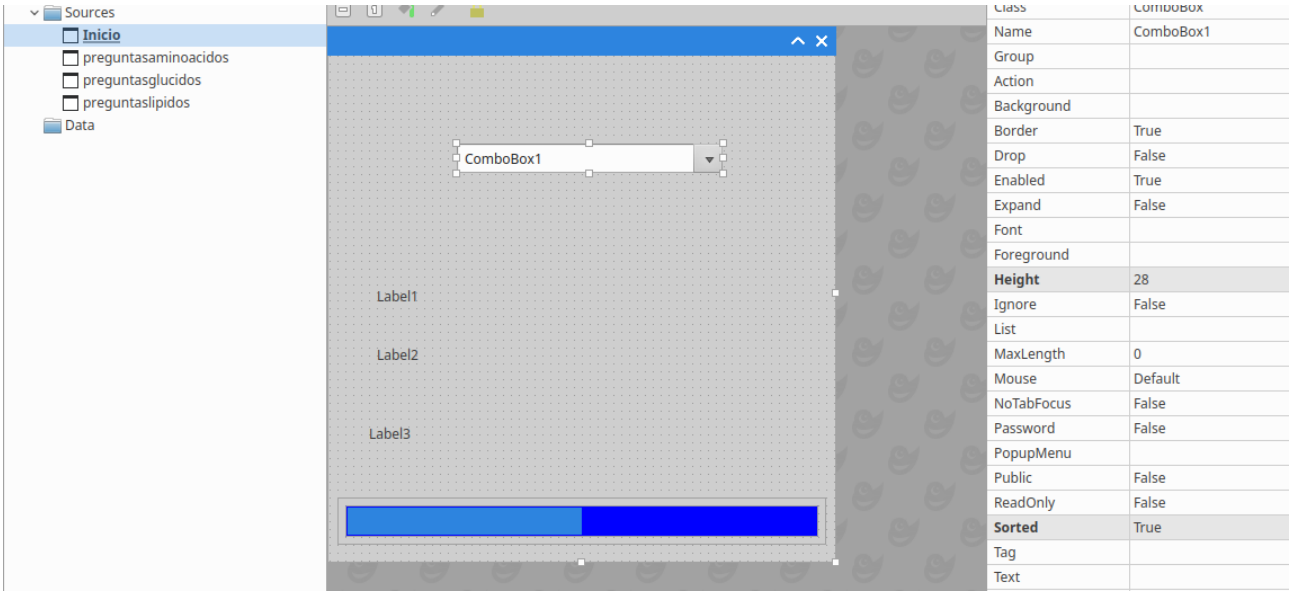


Figura 58

La carga de los ítems como dijimos se hará durante la ejecución y la hacemos con el evento Open del formulario Inicio. Es decir, que cuando arranque el software, el primer formulario que se abre es el Inicio por ser el Startup Form. Como podemos ver en la Figura 59, además de otras instrucciones, hay tres instrucciones con la forma general

ComboBox1.Add("un texto")

Así, tenemos tres líneas con los textos Glúcidos, Lípidos y Aminocidos.

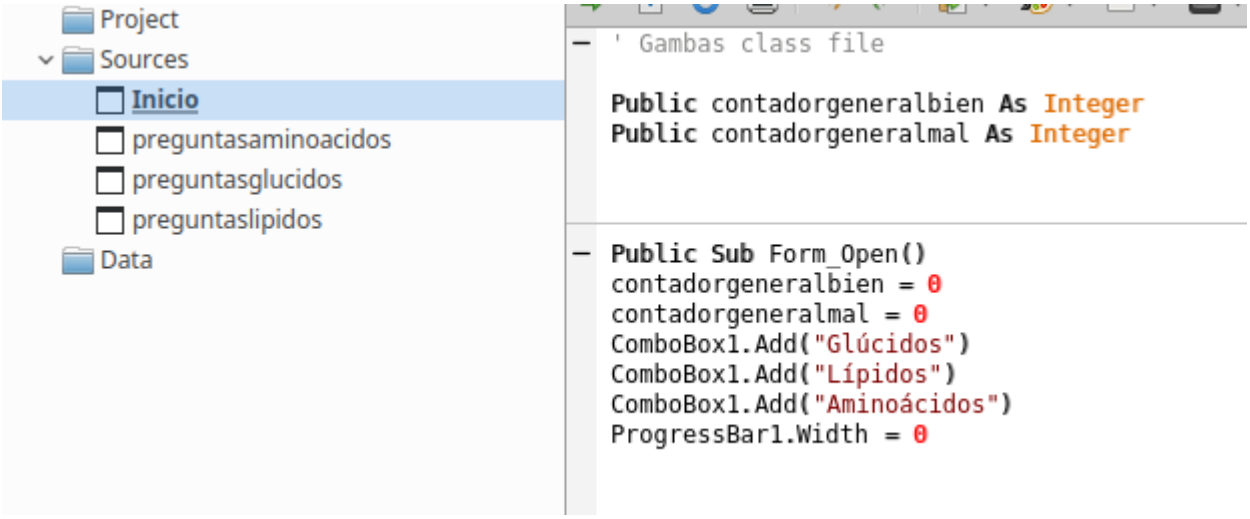


Figura 59

Al correr el software tendremos nuestro ComboBox1 con esos textos, ordenados en orden alfabético (Figura 60) ya que la propiedad Sorted fue seteada en True. Una variable muy útil en los ComboBox es la variable Index. Esta variable es un número entero que indica la posición de cada ítem en el listado. Así, los Index para Aminoácidos, Glucidos y Lípidos serán 0, 1 y 2, respectivamente. Debe notar que el Index comienza desde 0.

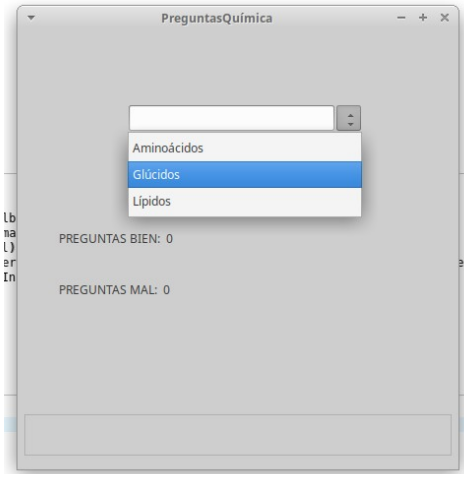


Figura 60

Ahora debemos programar que cuando ejecutemos algún evento del ComboBox1, se muestren los formularios correspondientes. Para esto introduciremos un nuevo control de flujo, que es la estructura Select Case.

Los ComboBox como todos los objetos usados en gambas tienen muchos eventos, pero casi siempre el mejor evento es el click con el botón izquierdo del Mouse. A este evento lo conocemos ya como Click. Como ya hemos hecho en otros casos hacemos click con el botón derecho sobre el ComboBox, elegimos even y de allí Click, Figura 61

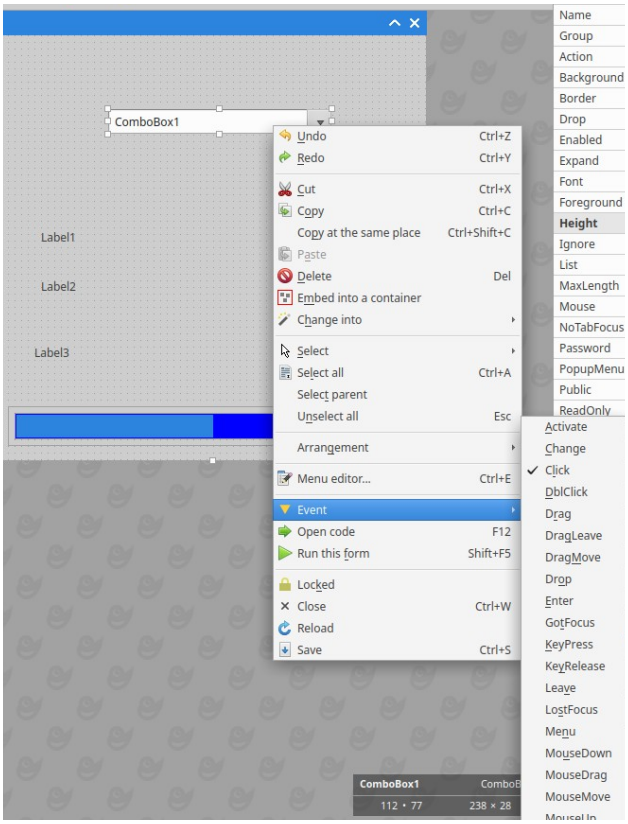


Figura 61

Ya dentro del evento Click colocamos la estructura Select Case, Figura 62.

Estructura Select Case

La Figura 62 muestra la estructura para este caso. En primer lugar describamos su funcionamiento. Esta estructura de control de flujo tiene una variable que en este caso llamamos índice, que fue declarada como número entero. Esa variable se declara al hacer click en el ComboBox1.

A continuación esa variable, se inicializa con el valor del Index del ComboBox1 sobre el que hicimos click. Ejemplo: Si desplegamos el ComboBox1 y hacemos click sobre Glúcidos, cuyo Index es 1, la variable indice, tomará el valor 1.

La estructura Select Case comienza realmente en la línea

Select Case indice

que nos indica que realizará una acción según el valor de la variable indice. Debajo podemos observar

Case 0

    Inicio.Hide

preguntasaminoacidos. Show

Lo que indica que si hubieramos hecho click sobre el primer elemento del ComboBox1: Aminoácidos, la variable indice tomaría el valor 0 y se ocultaría el formulario Inicio, pasando a mostrarse el formulario preguntasaminoácidos.

```
Public Sub ComboBox1_Click()
Dim indice As Integer
indice = ComboBox1.Index

Select Case indice
  Case 0
    Inicio.Hide
    preguntasaminoacidos.Show
  Case 1
    Inicio.Hide
    preguntasglucidos.Show
  Case 2
    Inicio.Hide
    preguntaslipidos.Show
End Select

End
```

Figura 62

TextLabel

El objeto TextLabel es similar en casi todos sus aspectos al objeto Label. La principal diferencia es que permite escribir en varias líneas.

En la Figura 63 se muestran 4 TextLabel, que reciben un texto luego de controlar el examen. En algunos de ellos figura: "su repuestas fue correcta" que entra en una línea y en otros, cuando la pregunta fue respondida de manera errónea, figura la respuesta correcta, que como vemos en la pregunta 3 ocupa más de una línea.

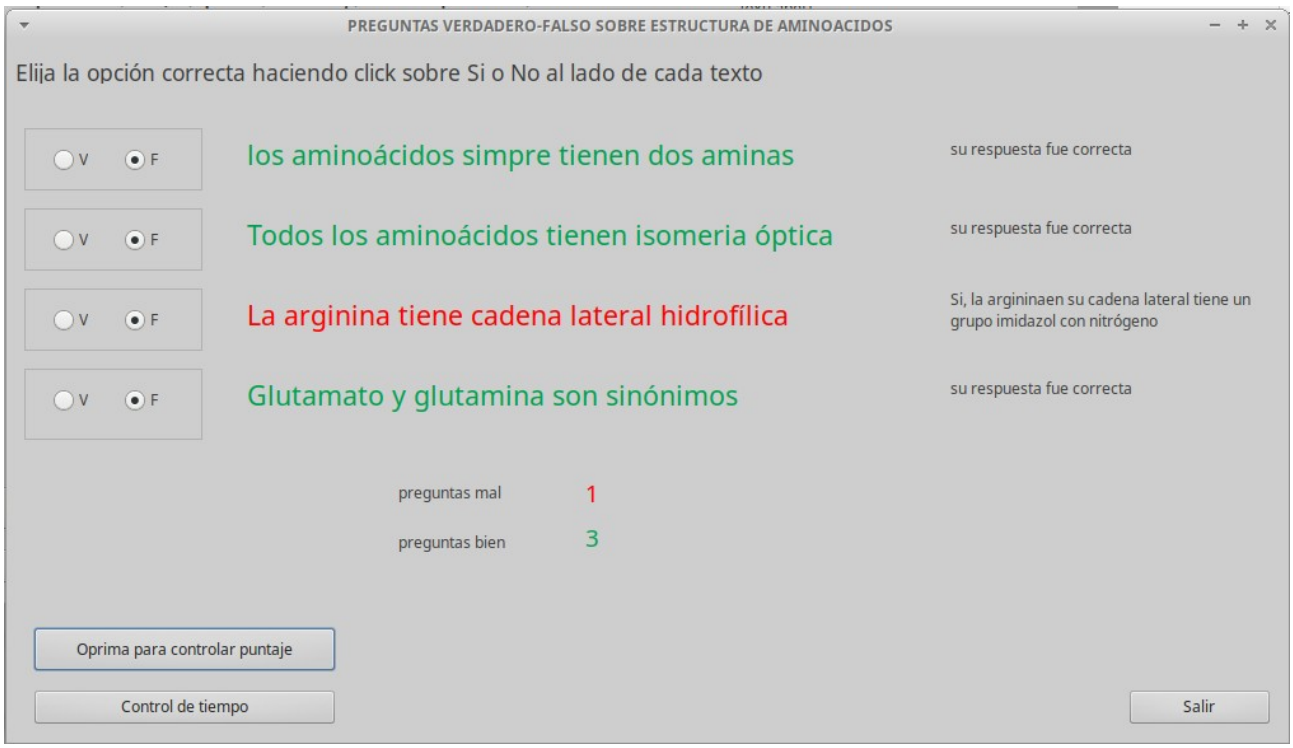


Figura 63

Los TextLabels tienen las mismas propiedades que los Labels y para asignarle un texto puede hacerse desde la propiedad Text de la ventana properties, en tal caso ese texto se verá al iniciar la aplicación. Si la propiedad Text en la ventana Properties queda vacía, no se mostrará texto al iniciar el formulario de la aplicación.

El texto por supuesto puede asignarse por código en el tiempo de diseño y apareceré cuando se ejecute el mismo en tiempo de ejecución. En la Figura 64, se muestra la asignación de un texto al TextLabel1. Esta asignación se halla dentro de una estructura de control de flujo de tipo IF THEN ELSE, que asigna al TextLabel1 un texto u otro, según la propiedad value del objeto verdadero1 que es un RadioButton

```
Public Sub correccion_Click()

  If verdadero1.value = False Then
    contadorbien = contadorbien + 1
    proposicion1.Foreground = &H0000A04D&
    TextLabel1.text = "su respuesta fue correcta"

    Else
    contadormal = contadormal + 1
    proposicion1.Foreground = &HFF0000&
    TextLabel1.text = "NO, la mayoría tiene un solo amino"
  Endif
```

Figura 64

Progress bar

Los objetos ProgressBar, permiten mostrar una barra cuya longitud puede ser variable durante la ejecución de un software. Es común utilizarla para mostrar el avance de un proceso. En la aplicación que nos hallamos desarrollando utilizaremos dos ProgressBar, una para mostrar el avance de nuestra aplicación y otra para mostrar el avance del tiempo.

Nuestra aplicación tiene un formulario con el que inicia, llamado Inicio. Allí tendremos el ProgressBar1.

Cuando iniciamos la aplicación, el ProgressBar1 tendrá longitud 0, y se irá sumando un 33% de su longitud al realizar cada ejercitación, Figura 65. Como tenemos tres formularios, al finalizar la ejercitación de los tres, la barra llegará a su longitud máxima

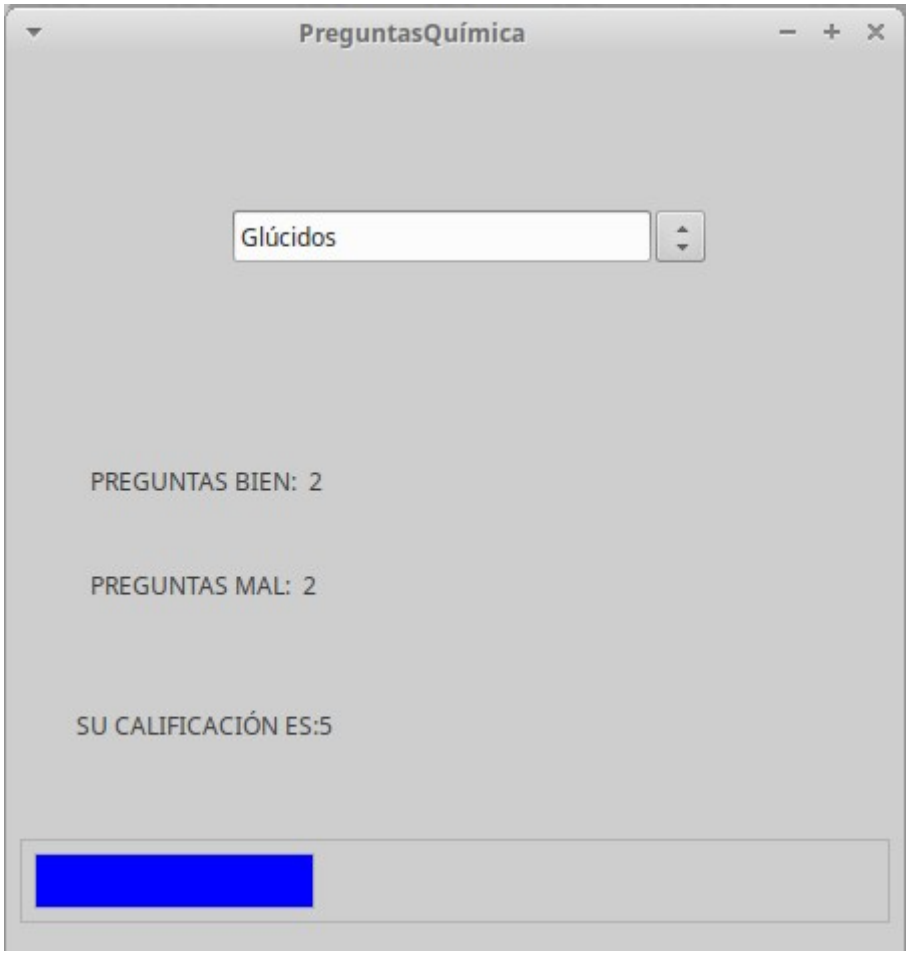


Figura 65

Como todo objeto se arrastra desde la barra de herramientas al formulario y luego se fijan sus propiedades.
Las propiedades más importantes que veremos son

Background: es el color de la barra la que fijaremos en color azul. Para ello pueden seleccionarse desde los ... que se hallan a final de la línea de la propiedad o bien escribiendo Blue. Podría ser cualquier otro color.

Label: lo colocaremos en False y elimina el texto que podría aparecer dentro de la barra.

Width: es el largo de la barra, que irá creciendo a lo largo de la ejecución. En general es conveniente colocar el valor 0 en Properties.Width, si la barra comienza desde ese valor. O bien podemos asignarla a través de código, por ejemplo dentro del evento Open del formulario, como hemos hecho en nuestro caso, Figura 66.

```
Public Sub Form_Open()
contadorgeneralbien = 0
contadorgeneralmal = 0
ComboBox1.Add("Glúcidos")
ComboBox1.Add("Lípidos")
ComboBox1.Add("Aminoácidos")
ProgressBar1.Width = 0


End
```

Figura 66

Cada formulario de ejercitación que hacemos (recuerde que tenemos tres: glúcidos, lípidos y aminoácidos) los contadores de preguntas bien y mal respondidas de cada formulario, toman valores acordes a nuestras respuestas y sus valores se suman a dos contadores de preguntas bien y mal respondidas de todo el software. Estos son contadorgeneralbien y contadorgeneralmal. La aplicación completa tiene 12 preguntas (4 por formulario) por lo tanto la suma de contadorgeneralbien y contadorgeneralmal debe dar 12 al haber recorrido los tres formularios. Y en esta situación el ProgressBar1 debería llegar a su máxima longitud. Y lo debería mostrar cuando volvemos a mostrar el formulario Inicio. Por ello en el evento Show del formulario Inicio colocaremos el código que lo hace. En este evento además nos muestra las preguntas totales bien y mal en los label1 y label2, respectivamente, Figura 67.

En el label3 además nos muestra la calificación que surge de dividir la preguntas bien del contadorgeneralbien por la suma de los contadorgeneralbien y contadorgeneralmal. El resultado se multiplica por 10 para que la nota quede es una escala de 0-10.

Allí mismo colocamos el código para que el ProgressBar1 tome una longitud acorde al número de preguntas respondidas, no importa si bien o mal. Hagamos un análisis de este código.

El código es

ProgressBar1.Width = 420 * (Inicio.contadorgeneralbien + Inicio.contadorgeneralmal) / 12

El número 420 es la longitud máxima que debería tomar cuando terminemos la ejercitación de 12 preguntas y lo fijamos por diseño del objeto en el formulario Inicio. Este valor lo multiplicamos por

la suma de contadorgeneralbien y contadorgeneralmal. Esta suma tomará el valor cero antes de haber realizado una ejercitación, dará 4 al finalizar un formulario, 8 al finalizar el segundo y 12 al finalizar el tercero. Estos valores se dividen por 12, por lo que 420 será multiplicado por valores que van entre 0 y 1, que corresponden a no haber realizado ningún ejercicio o todos.

```
Public Sub Form_Show()

Label1.text = "PREGUNTAS BIEN:  " & Inicio.contadorgeneralbien
Label2.text = "PREGUNTAS MAL:  " & Inicio.contadorgeneralmal
If (Inicio.contadorgeneralbien + Inicio.contadorgeneralmal) > 0
Label3.text = "SU CALIFICACIÓN ES:" & (Inicio.contadorgeneralbien * 10 / (Inicio.contadorgeneralbien + Inicio.contadorgeneralmal))
ProgressBar1.Width = 420 * (Inicio.contadorgeneralbien + Inicio.contadorgeneralmal) / 12

Endif
End
```

Figura 67

Cuando la barra esté completa, en el formulario inicio la suma de las respuestas bien y mal deberá dar 12, Figura 68. En cambio si solo realizamos una ejercitación, la barra tendrá una longitud del 33% y la suma de bien y mal debe dar 4, Figura 65.

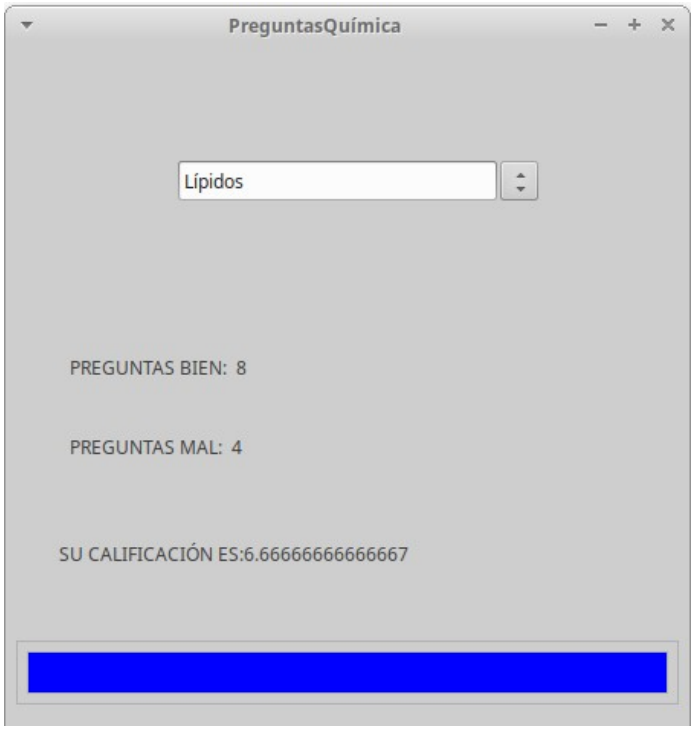


Figura 68

InputBox

InputBox es un objeto que nos permite introducir desde el teclado un valor de una variable. Por supuesto esto ya lo hemos hecho desde TextBox, pero este objeto es interesante porque aparece como una nueva ventana.

Su programación es sencilla y se puede colocar dentro de cualquier evento de cualquier objeto. En nuestro caso lo introducimos dentro de un Button, para que se pueda seleccionar el tiempo máximo asignado para responder las preguntas de un formulario

En el formulario preguntasaminoacidos, tenemos un botón Button2, que en su evento Click colocamos el InputBox, Figura 69. Al hacer click en ese botón una variable b, que hemos definido para todo el formulario como Integer tomará el valor que ingresemos por el InputBox. Además dentro del InputBox tenemos entre paréntesis textos, los dos primeros aparecen en el cuerpo del InputBox y el tercero quedó sin texto, "". En esta situación en la barra del inputBox se mostrará el nombre del formulario. Puedo variar según necesidad el uso de estos tres textos.

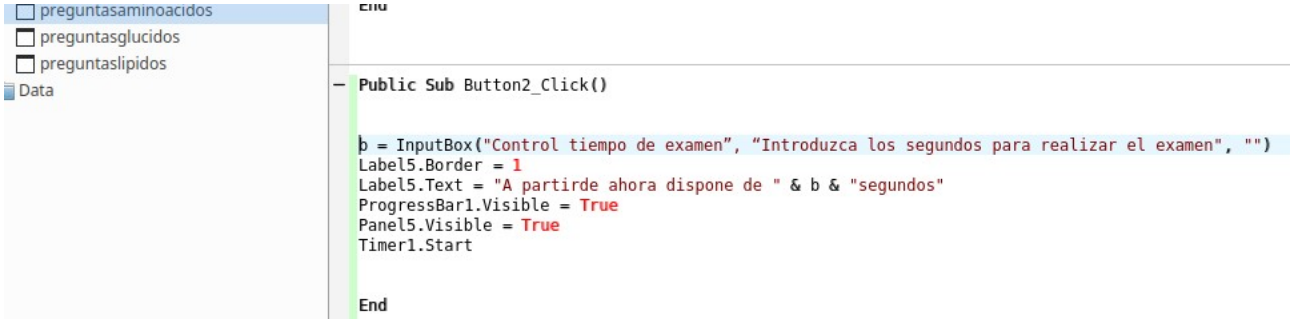


Figura 69

Cuando hagamos Click sobre el Button2, se mostrará el inputBox y quedará esperando el ingreso del valor de la variable. La segunda línea de código de Button2: Label5.Border=1, Figura 69, no se ejecutará hasta que se oprima OK en InputBox

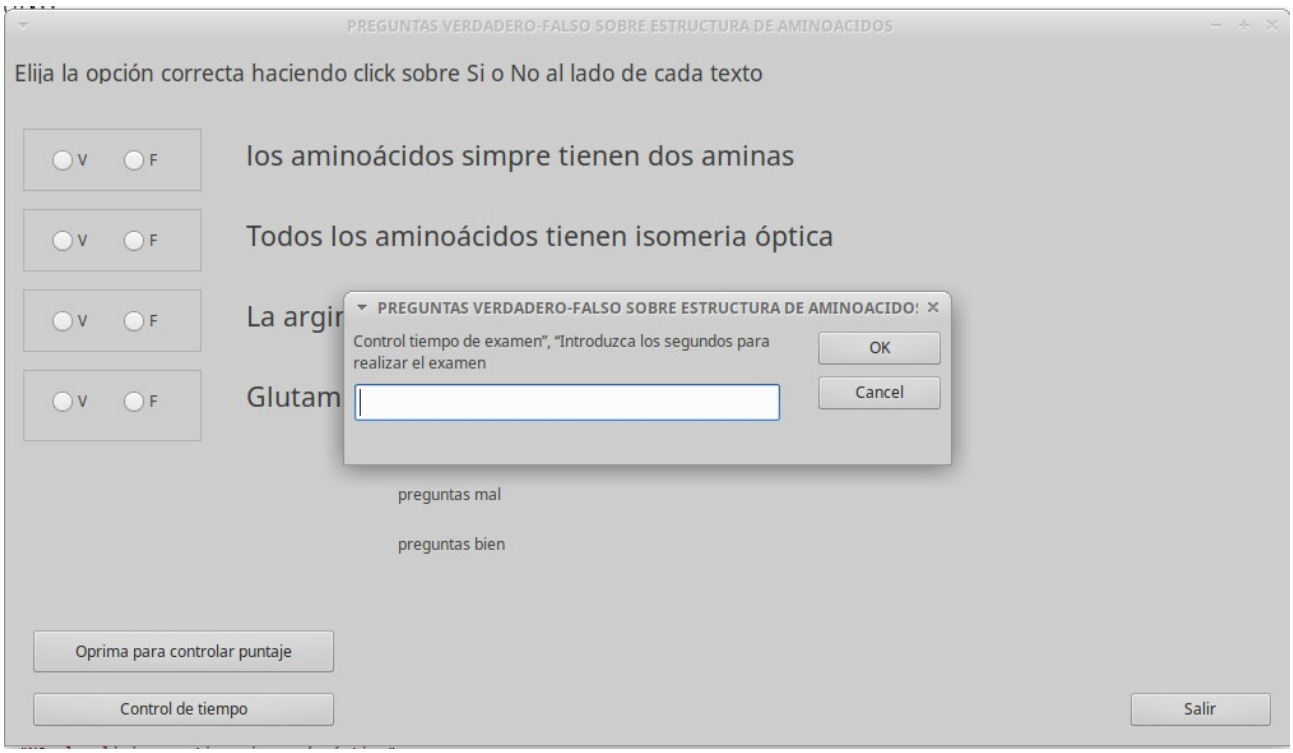


Figura 70

Cuando oprimamos OK, el flujo del software seguirá su curso, cambiando el borde del Label5, haciendo visible ProgressBar1 y Panel5 y poniendo en marcha el Timer1, Figura 69.

Clase 6

En esta clase iniciaremos una nueva aplicación gráfica, utilizando como base la aplicación CursoLinux52 y asignaremos al proyecto el nombre "CursoLinux56" y el título de la aplicación será "UsoBasesDatos". Se trata de una aplicación que me permitirá evaluar conceptos de un área del conocimiento a través de preguntas sugeridas y nos servirá para introducir nuevos objetos y los conceptos básicos para utilizar bases de datos en nuestras aplicaciones.

La diferencia de esta aplicación con la anterior es que en esta el usuario podrá trabajar en días diferentes y almacenar su avance en la ejercitación.

Bibliotecas necesarias.

Asegúrese que tenga instaladas las bibliotecas

gambas3.gb.db

gambas3.gb.db.sqlite2

gambas3.gb.db.sqlite3

A través de synaptic será lo más sencillo. Ingresando a synaptic puede buscarlas de a una o simultáneamente, pero deberá tener una presentación similar a la de la Figura 71, en la que se resaltaron las bibliotecas imprescindibles para el manejo de bases de datos

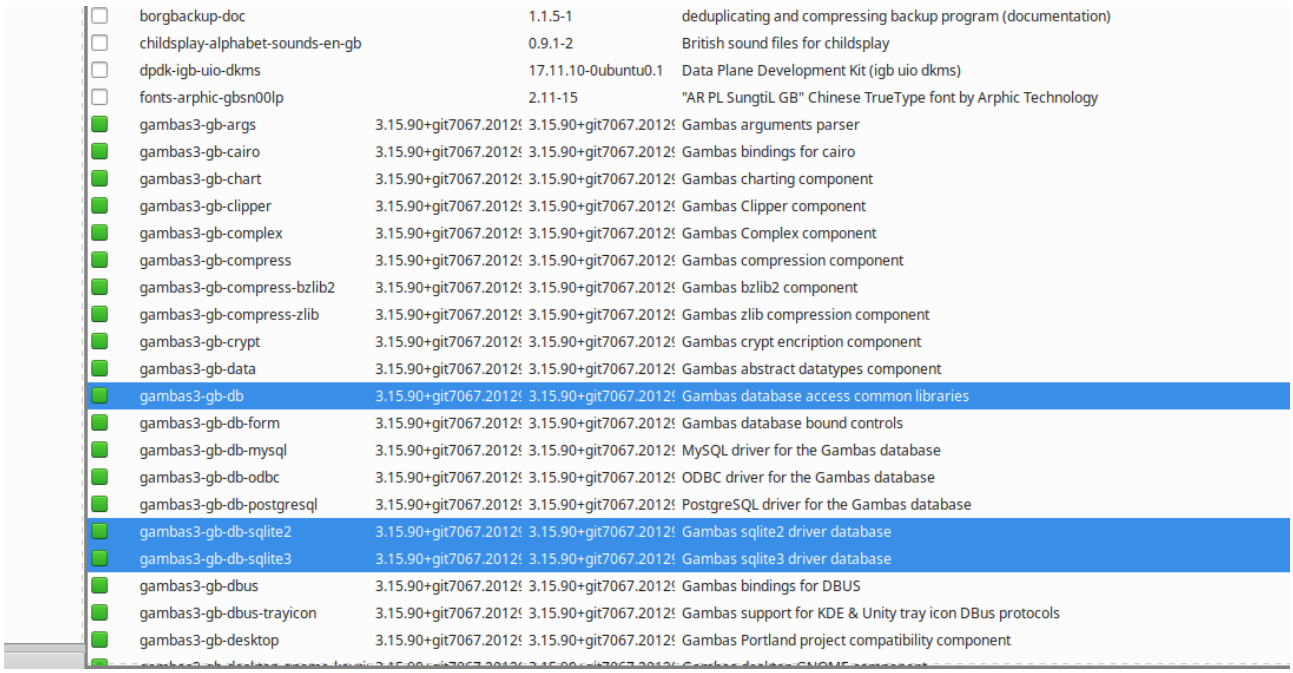


Figura 71

Si no tuviera las bibliotecas marcadas como muestra la Figura 71, márquelas para instalación y oprima Apply. Luego de comprobada la instalación correcta de las bibliotecas vamos a nuestra aplicación en Gambas 3 y del menú de tareas ejecutamos

Project > Properties > Components

Llegaremos a la tabla de la Figura 72. Allí, en el ítem Components, debemos comprobar que las bibliotecas

gb.db

gb.sqlite3

estén instaladas para el proyecto. Como verá hay bibliotecas marcadas y otras no. Si alguna o las dos mencionadas no lo estuvieran, márquelas y luego oprima OK. Si no aparece en la lista gb.db.sqlite2, no se preocupe.

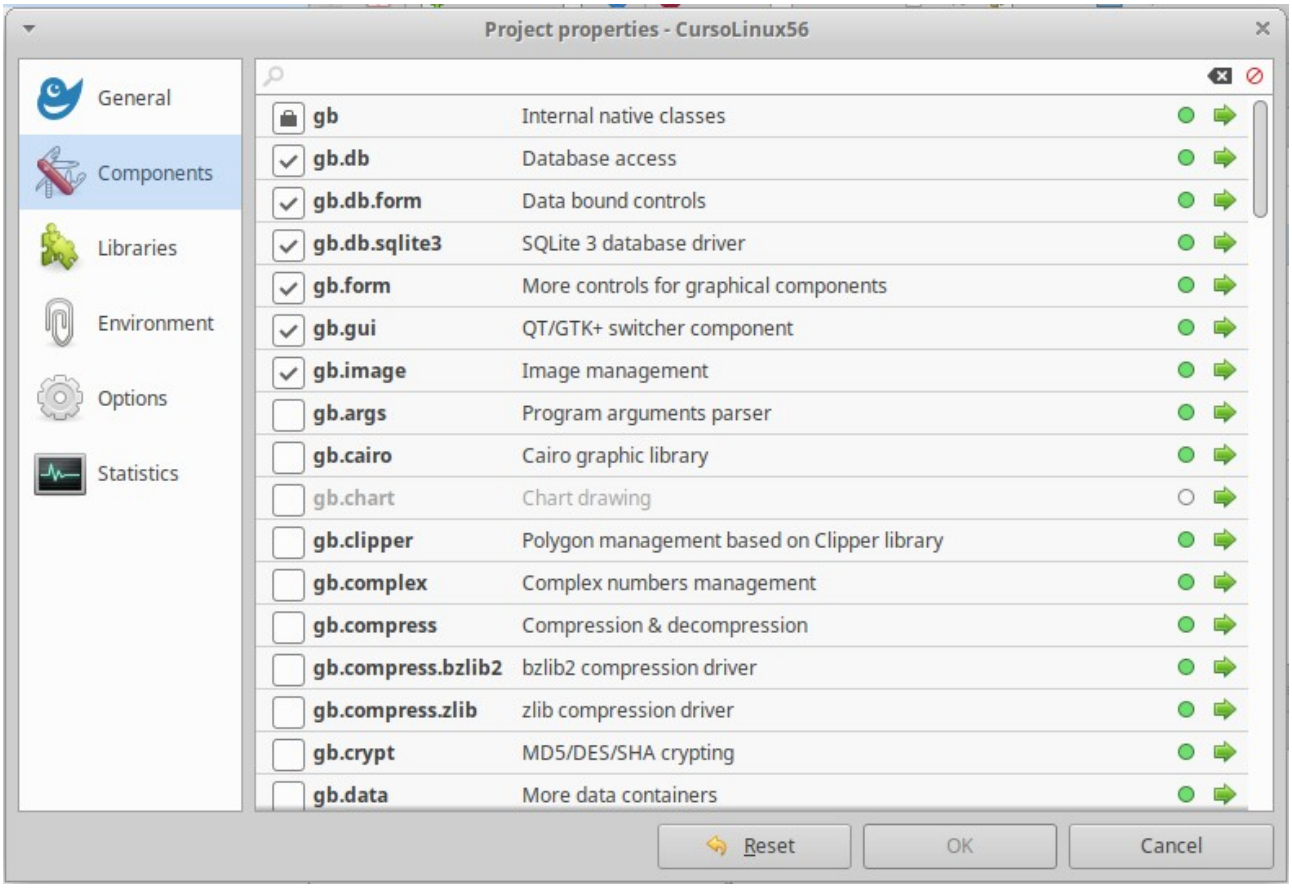


Figura 72

Luego de este proceso estamos en condiciones de crear nuestra base de datos, que en lo haremos directamente desde nuestra aplicación.

Creación base de datos

Si cumplimos con la instalación de las bibliotecas gb.db y gb.db.sqlite3, en nuestro proyecto tendremos un nuevo elemento que se llama Connections, Figura 73.

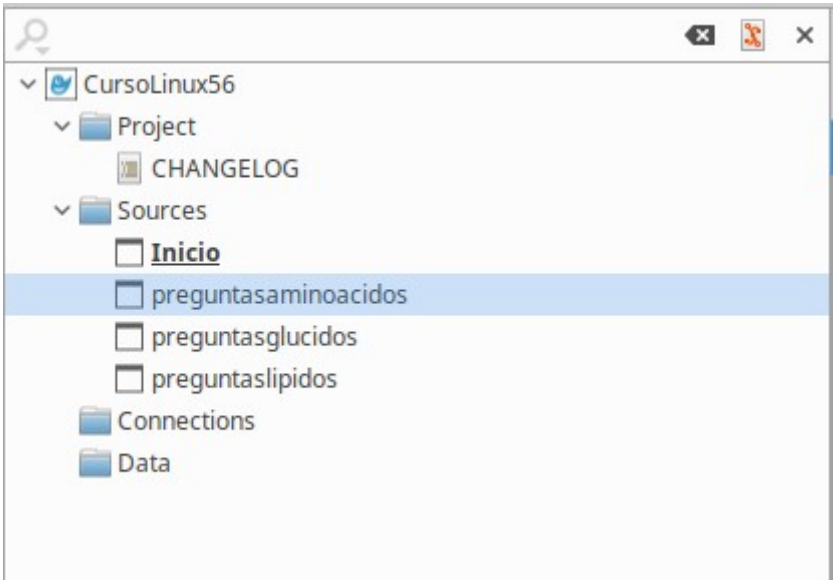


Figura 73

Hacemos click con el botón derecho del mouse sobre Connections y elegimos NewConnection
Se abrirá la ventana de la Figura 74, donde completaremos ciertos datos

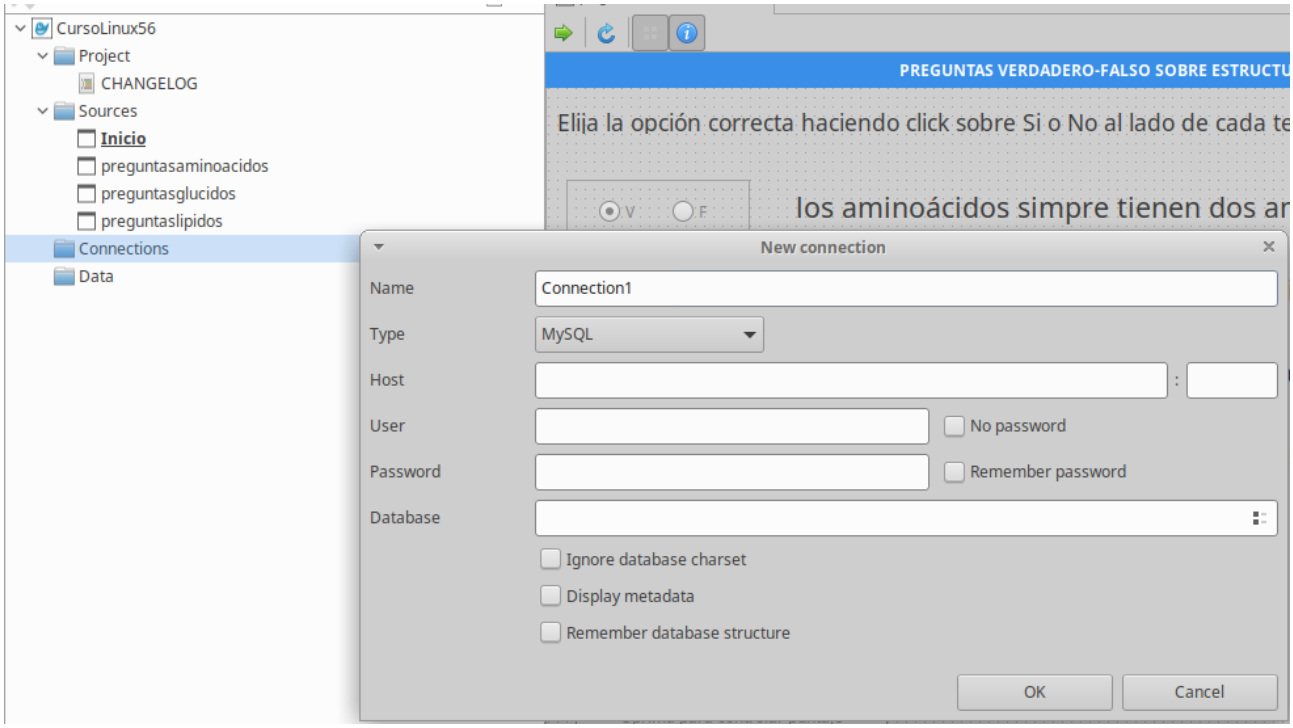


Figura 74

En Name, dejaremos Connection1, pero podríamos cambiarlo si lo deseamos
En Type, desplegamos el menú y elegimos SQLite, allí cambiará la ventana a la presentación de la Figura 75

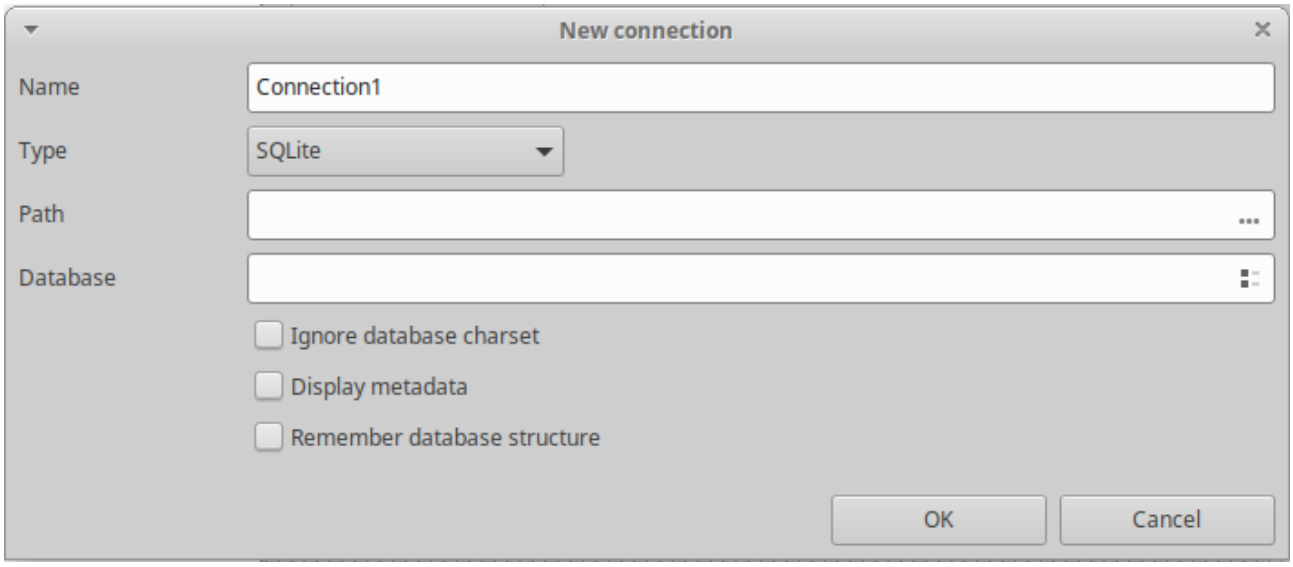


Figura 75

Luego indicamos donde ubicaremos la base de datos, es decir el Path. En nuestro caso elegiremos el mismo directorio donde está la aplicación. Para ello haga click en los tres puntos del final del campo Path y podrá buscar el mismo. Podría ubicarla si lo desea en otro sitio de su computadora, aunque no será muy operativo tener su aplicación dispersa en su unidades de almacenamiento. Lo recomendable es hacerlo en el mismo directorio que está la aplicación y éste podemos sacarlo desde el menú de tareas

Project > Properties > Components

En el ítem General podemos observar Location, donde se halla el path necesario

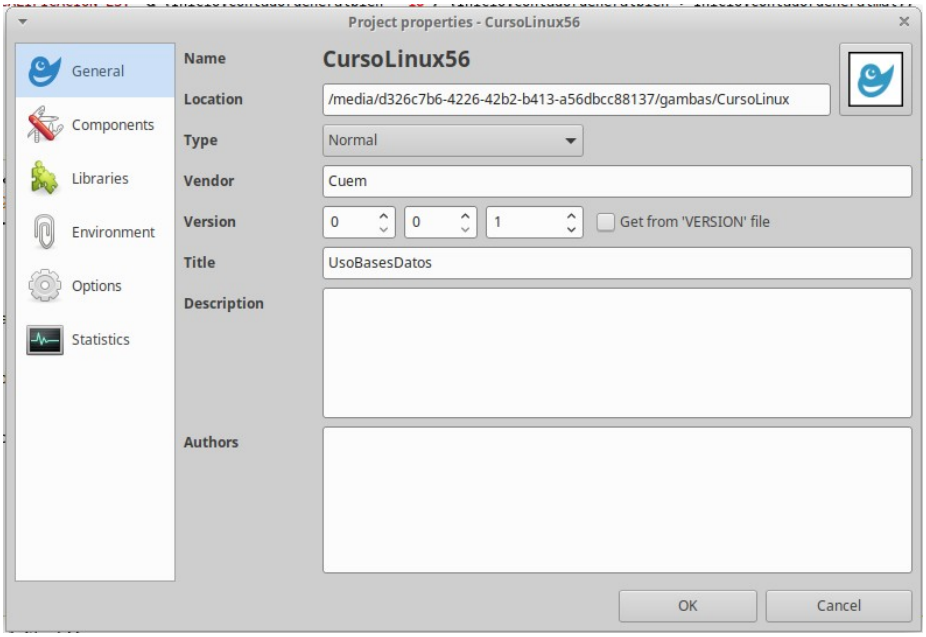


Figura 76

Finalmente vamos al campo Database. Haremos click sobre los íconos de la derecha del campo y cambiará nuevamente la ventana, apareciendo un botón "Createdatabase", Figura 77. En el campo Database, colocaremos el nombre de la base de datos: datosquimica.

Debajo hallará una ventana más grande donde se listarán las bases de datos de su aplicación, en este caso está vacío ya que es la primer base de datos

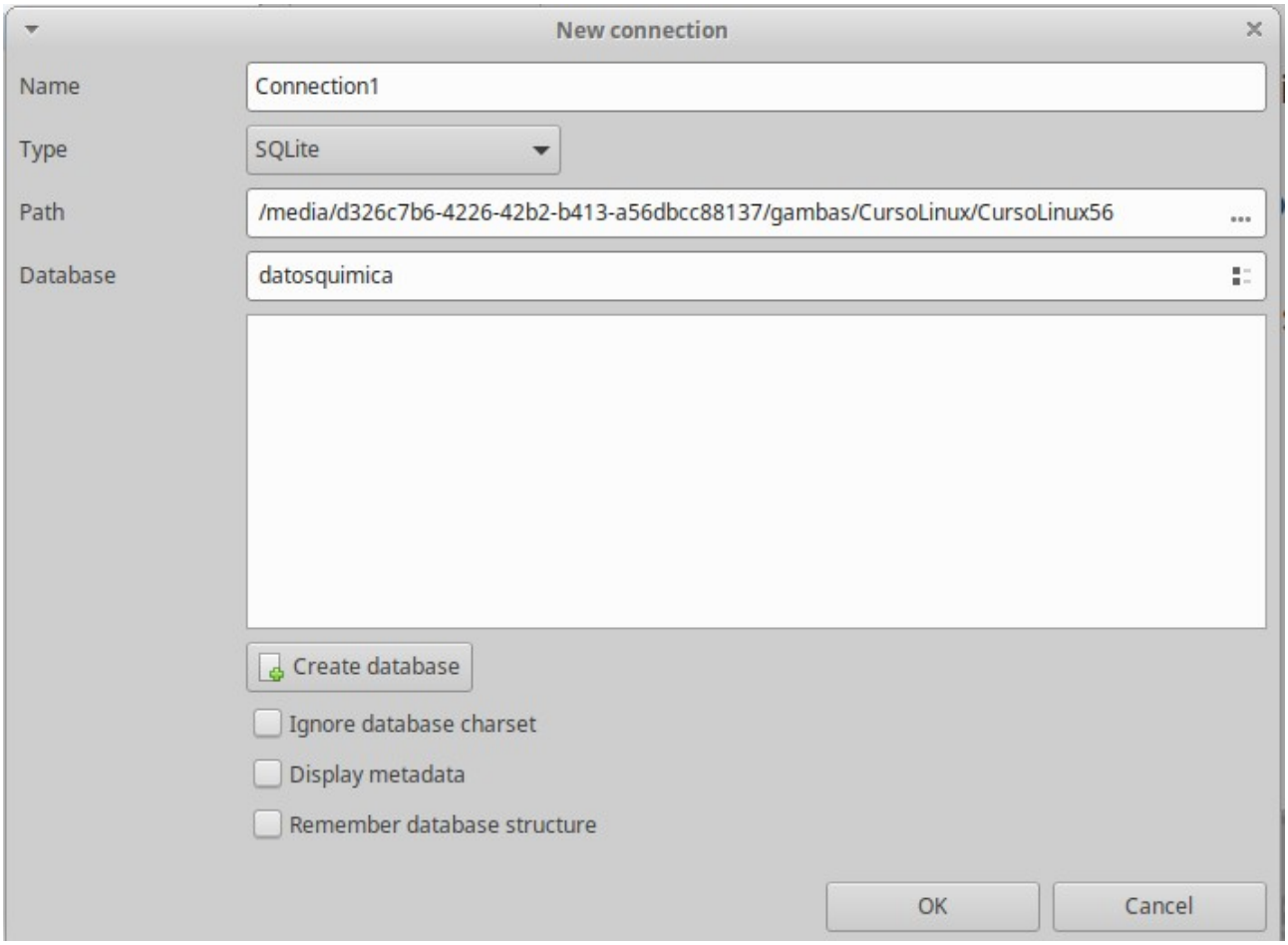


Figura 77

Oprimimos Create database, con lo que finalizamos este proceso y pasaremos a configurar las tablas y campos de la base de datos: datosquimica. Oprimimos OK y seguimos adelante, Figura 78. Podemos ver en el listado, ya incorporada a nuestra base de datos.

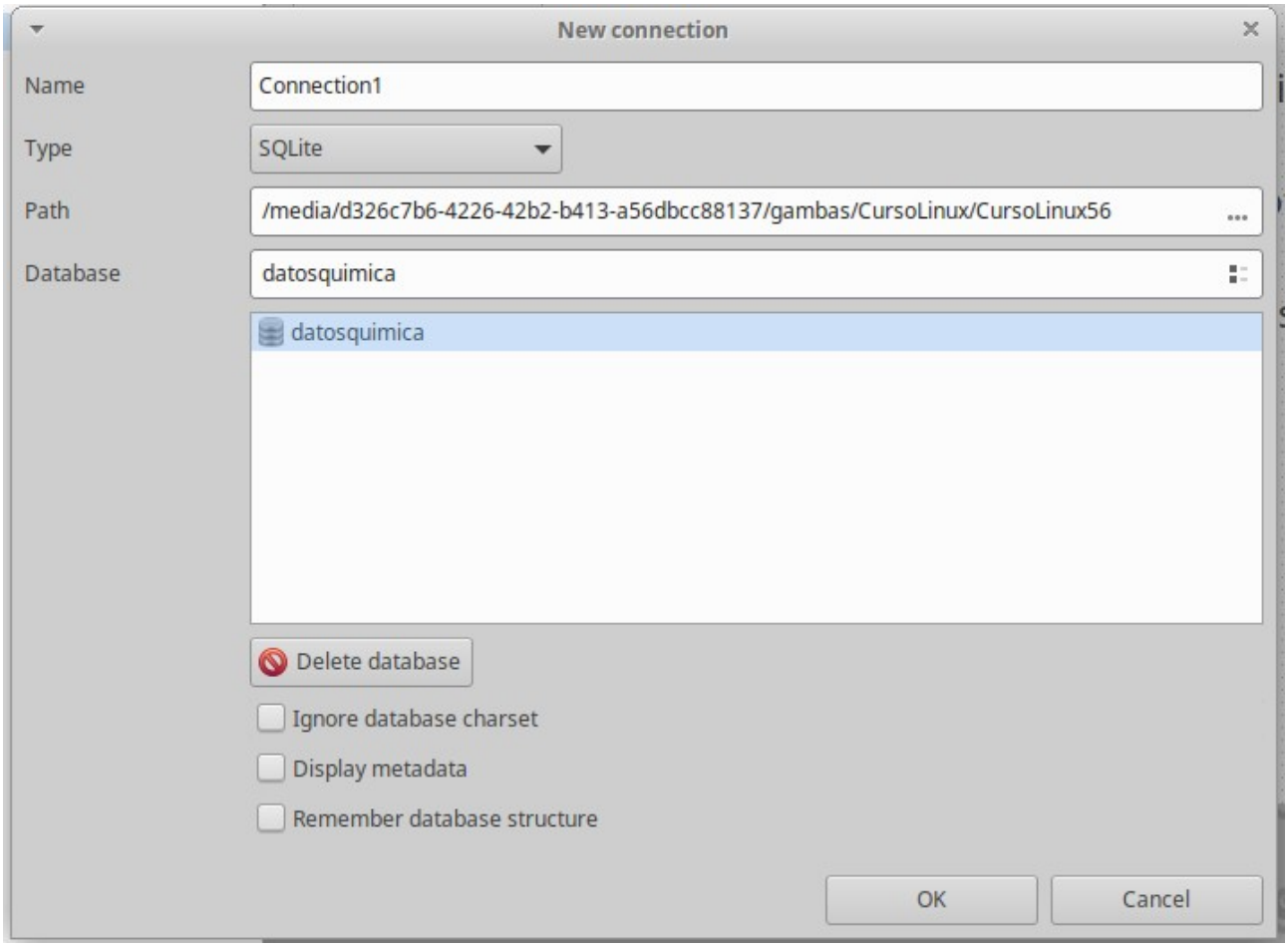


Figura 78

Pasaremos a una presentación, como muestra la Figura 79. A la derecha tenemos una solapa Conenction1.connection, hacemos click en ella para obtener la presentación deseada

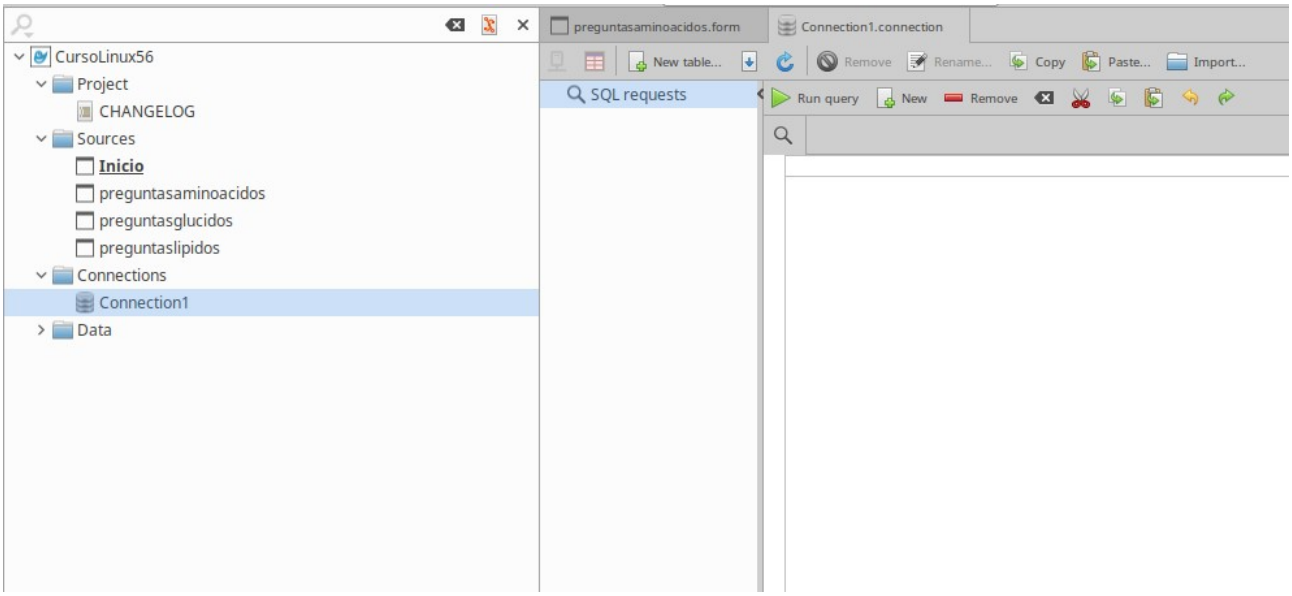


Figura 79

Hacemos click sobre SQL requests y oprimimos New table, se abrirá un cuadro de ingreso de datos, donde colocaremos el nombre que elegimos (pero podría ser otro) que es en este caso: ejercicios. Oprimimos OK y seguimos adelante con la definición de los campos

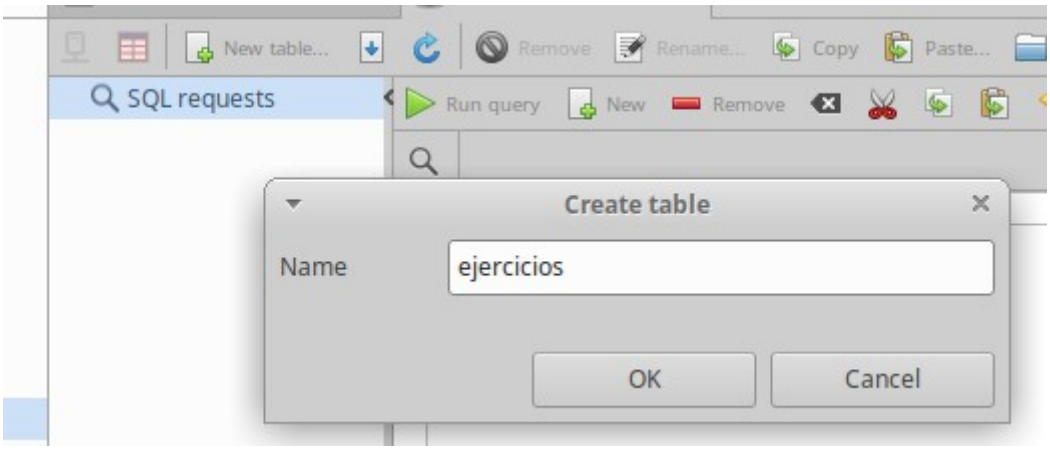


Figura 80

Luego de oprimir OK, cambiará la presentación de la pantalla pasando a la que se muestra en la Figura 81

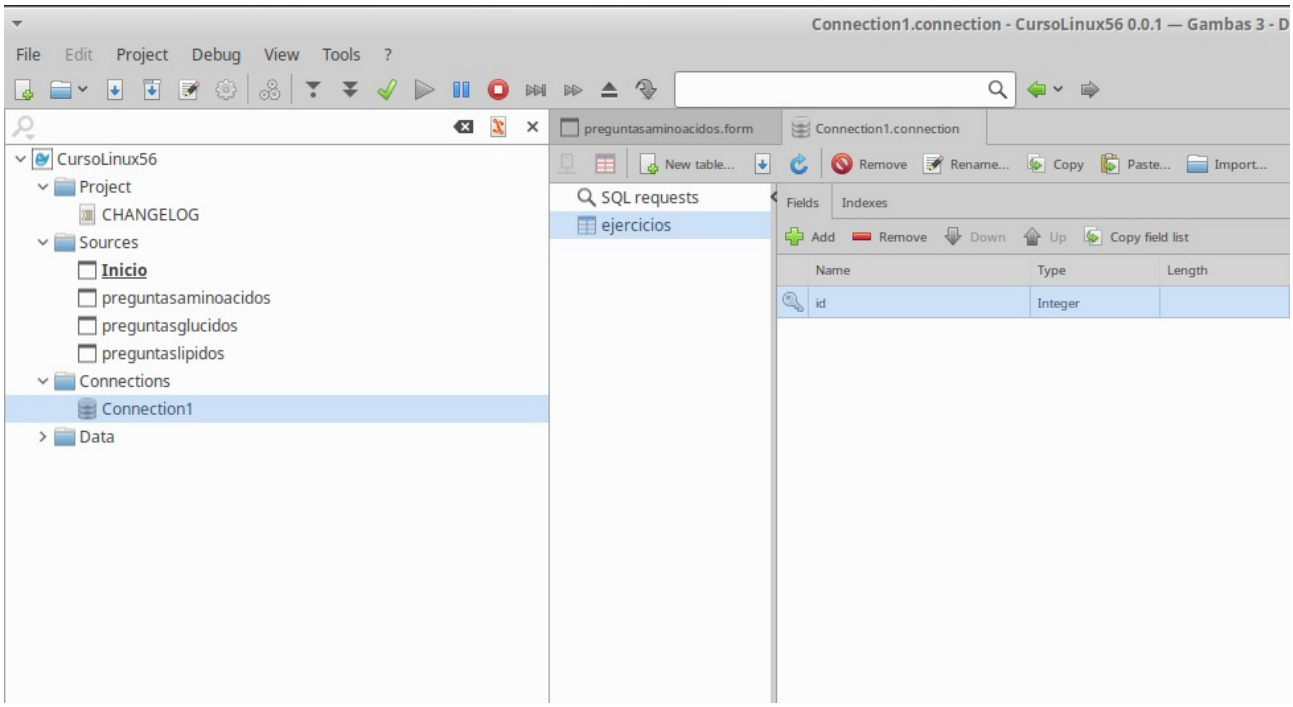


Figura 81

Antes de continuar analicemos lo que deseamos que se pueda almacenar en la base de datos. Los ítems siguientes muestran la información que nos proponemos almacenar:

1. nombre del usuario
2. documento
3. si hizo o no los ejercicios de aminoácidos.

4. si hizo o no los ejercicios de glúcidos.
5. si hizo o no los ejercicios de lípidos.
6. la nota de la ejercitación sobre aminoácidos.
7. la nota de la ejercitación sobre glucidos.
8. la nota de la ejercitación sobre lípidos.

Es decir debemos tener 8 campos para alojar los datos de cada registro. Cada campo tendrá la información de los 8 ítems anteriores y cada usuario será un registro de la base de datos. Para que no haya confusiones y sobre-escritura de datos, situación que puede producirse cuando existen dos usuarios con el mismo nombre, colocaremos el campo 2, documento como campo que no puede tener datos repetidos. Así, si un usuario desea ingresar aunque tenga el mismo nombre no será problema ya que seguramente no lo será su documento.

Cada campo almacenará una variable de un tipo adecuado que detallamos a continuación

1. nombre del usuario: este campo tendrá una variable de tipo string, es decir una cadena de caracteres, por ejemplo Juan Perez.
2. documento: tendrá una variable de tipo integer, pudiendo almacenar: 152369741. Si deseáramos que se colocara con punto, deberíamos utilizar una variable de tipo string.
3. si hizo o no los ejercicios de aminoácidos. Tendrá una variable Booleana, es decir su valor podrá ser True o False
4. si hizo o no los ejercicios de glúcidos. Tendra una variable Booleana, es decir su valor podrá ser True o False
5. si hizo o no los ejercicios de lípidos. Tendra una variable Booleana, es decir su valor podrá ser True o False
6. la nota de aminoácidos. Para este campo y los dos siguientes la variable será un integer, pudiendo almacenar notas de 0-10
7. la nota de glúcidos
8. la nota de química

Para crear los campos hacemos click sobre la tabla que hemos creado y elegimos la solapa Fields. Allí oprimimos el botón Add identificado por una cruz verde, que nos permitirá agregar un campo. Cuando hagamos esto aparecerá un listado y una ventana para colocar en la columna Name el nombre del campo. Podríamos dejar Field1, pero le colocaremos "nombre" como acordamos. A la derecha tendremos un listado desplegable donde elegimos string. En la columna "Length" dejamos unlimited, Figura 82. Podríamos elegir una longitud máxima de caracteres si lo deseáramos. Pero evitaremos inconvenientes en este momento. No importará el largo del nombre del usuario.

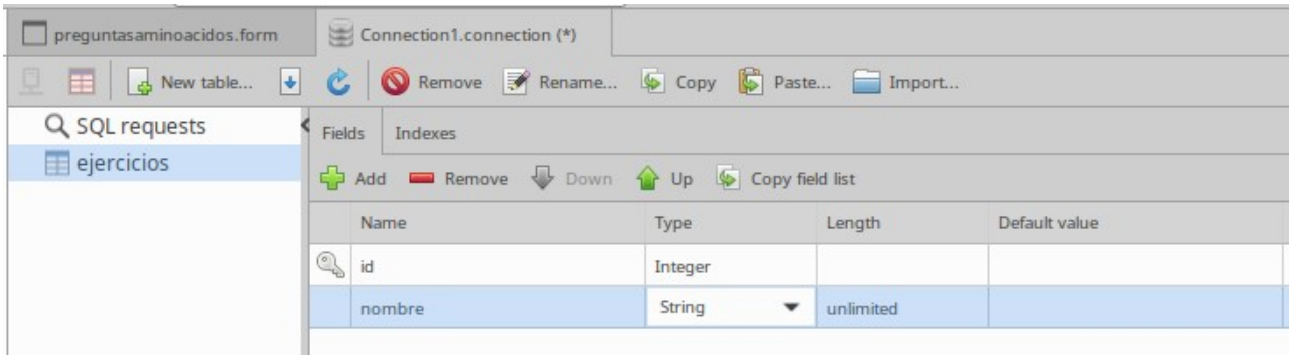


Figura 82

Introducimos luego el campo "documento", para el que elegiremos Type= integer y haremos doble click sobre la primer columna que tiene una llave de manera que este campo pueda tener registros no repetidos. Luego de esta acción quedará una llave a la izquierda del nombre del campo: documento, Figura 83.

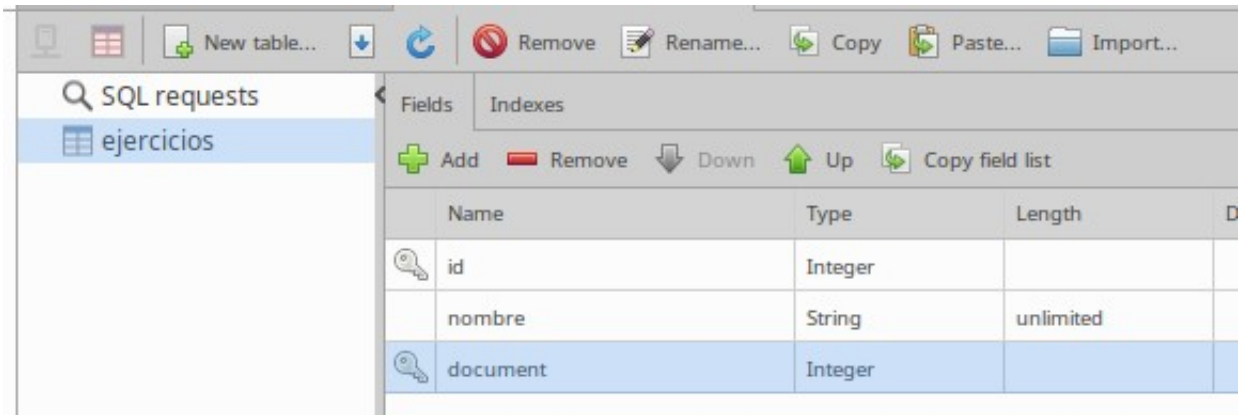


Figura 83

procedemos luego con los campos aminoacidos, glucidos y lipidos que son booleanos, quedándonos luego del agregado como muestra la Figura 84

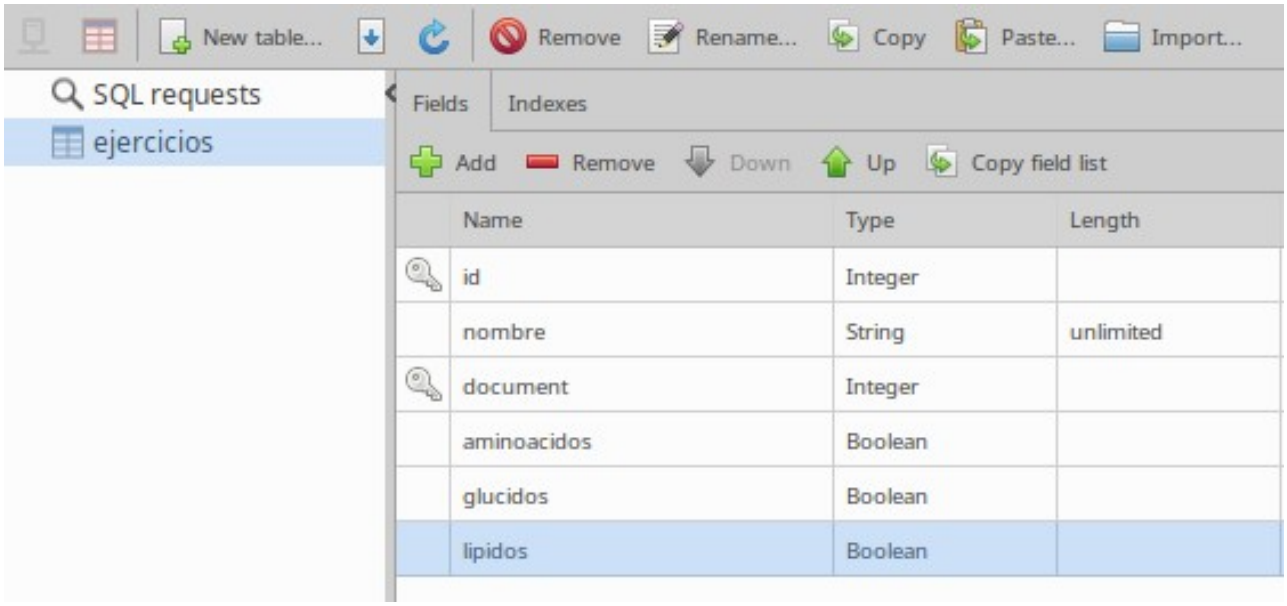


Figura 84

Finalmente agregamos los campos notaaminoacidos, notaglucidos, notalipidos, que serán de tipo integer. Luego de agregarlos tendremos una presentación como la Figura 85.

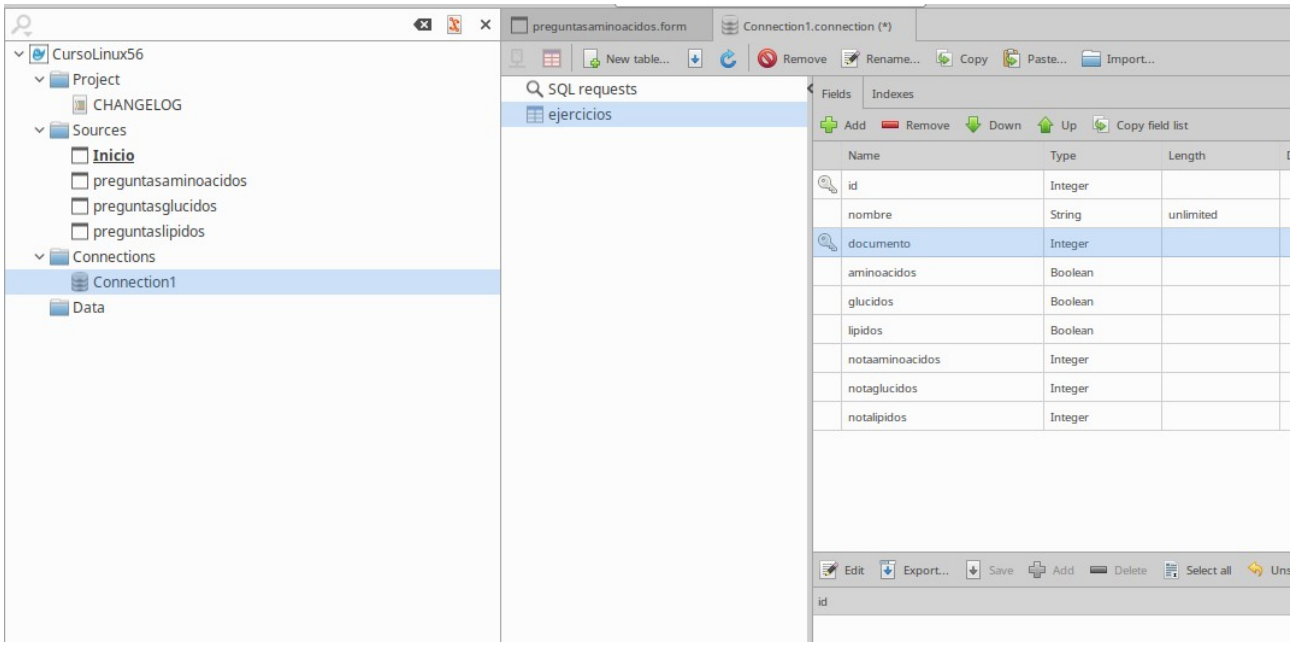


Figura 85

Gambas por defecto coloca un campo id que tiene una llave, podemos eliminarlo, marcándolo y oprimiendo Remove. Grabamos la tabla oprimiendo el botón Reload:  . Asi obtendremos la presentación final de nuestra tabla ejercicios con los campos necesarios, Figura 86.

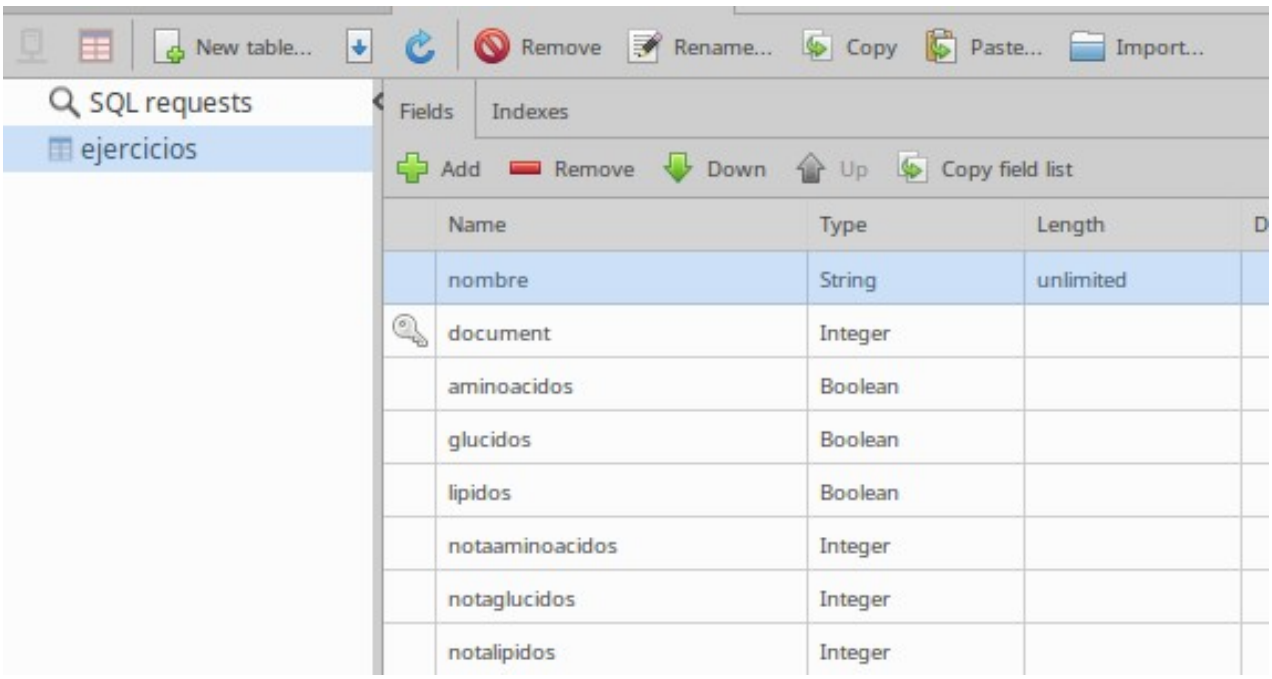


Figura 86

Introducción de registros de la base de datos

Si bien la base de datos se irá llenando a medida que los alumnos utilicen la aplicación. Veremos ahora como cargar algunos registros a la base de datos. desde el sitio que estamos trabajando. Cargaremos dos alumnos

Juan Perez

José Garcia

y supondremos que han realizado algunos ejercicios

Para lograrlo, vamos a la parte inferior y oprimimos Edit, y se abrirán los campos para completar. Los campos nombre, documento, notasglucidos, notaslipidos y notasaminoacidos debemos completarlos. Aquellos campos de notas que no se han realizado le pondremos cero y a los otros inventaremos una nota. En los campos aminoacidos, glucidos y lipidos seleccionamos True o False según haya o no realizado la ejercitación. Al agregar el primer alumno: Juan Perez, oprimimos Add, identificado por la flecha verde. Distinga que hay dos botones Add: uno superior que es para agregar campo y otro inferior para agregar registro. En este último caso nos hallamos trabajando. Luego de cargar los datos de Juan Perez y oprimir Add tendremos la presentación de la Figura 87.

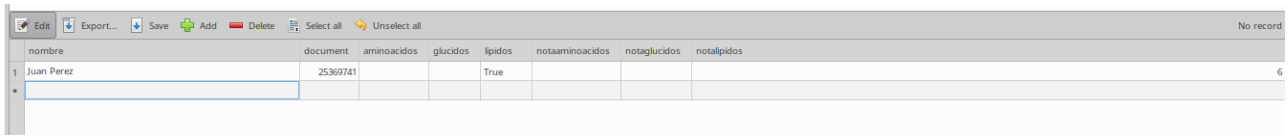


Figura 87

Agreguemos a José Garcia y supongamos que introducimos sus datos, utilizando el mismo número de documento que Juan Perez. Notaremos que cuando oprimamos Add, no nos permitirá guardar el registro, dado que documento no puede ser repetido, Figura 88. Colocamos entonces otro número.

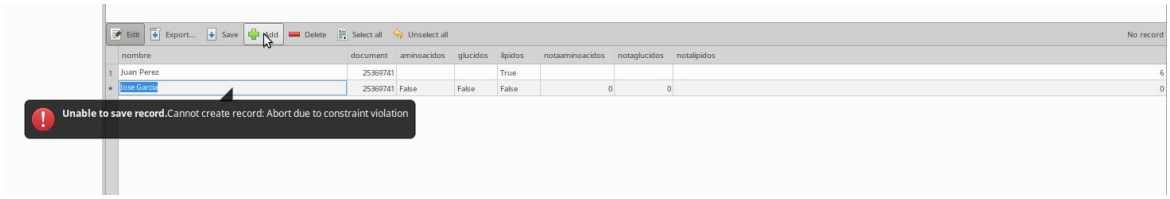


Figura 88

Finalmente colocamos Save para guardar los registros.

Si en este punto usted cierra Gambas, cuando vuelva a abrir la aplicación, debería tener toda la información de su tabla y los registros incorporados.

Incorporación de la base de datos a la aplicación gráfica

En primer lugar reorganizamos el formulario Inicio de la aplicación para que en la parte superior queden los objetos de selección de ejercitación y seguimiento de la misma en cada sesión y en la parte inferior los registros de datos.

Para ello introducimos un separador de ambas partes e incorporamos una serie de objetos:

ComboBox2 donde podremos ver los registros ya cargados

CheckBox1, 2 y 3: que mostrará con un tick si la ejercitación fue o no realizada

Un grupo de Labels que mostrarán algún texto y la nota de cada ejercitación

La aplicación nos ha quedado según muestra la Figura 89.

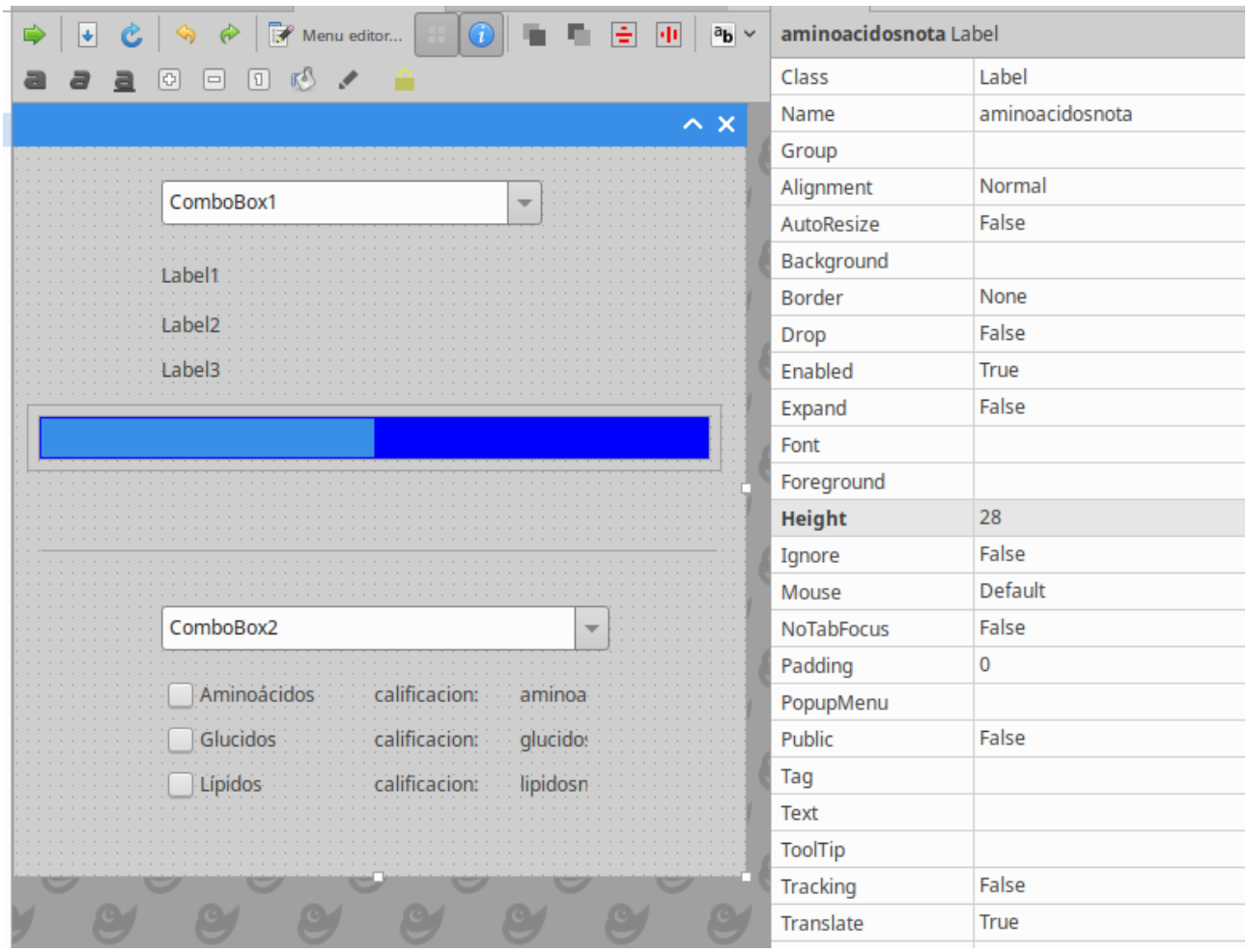


Figura 89

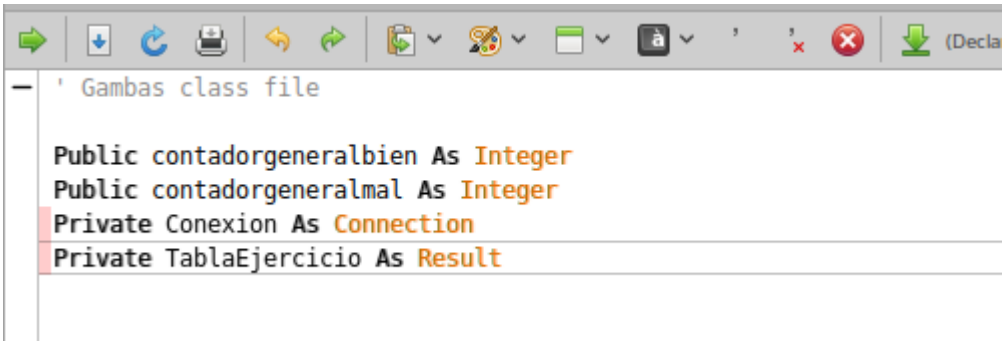
Lo que deseamos ahora es que cuando se inicie la aplicación además de permitirnos hacer una ejercitación, nos muestre la base de datos con los registros incorporados. Por supuesto que en la clase siguiente veremos cómo un alumno puede ingresar, hacer los ejercicios, registrarse y cargar la o las ejercitaciones realizadas. Por ahora nos conformaremos con menos.

Para lograr este objetivo debemos abrir la conexión a la base de datos para lo que utilizaremos algunas estructuras y nuevas y otras conocidas. No buscaremos explicación a ciertos códigos sino solo nos concentraremos en qué hace cada uno y su necesidad.

En el formulario class del form Inicio debemos definir algunas variables locales del formulario, Figura 90.

La variable Conexion, es la variable que se conecta con la base de datos y es del tipo Connection

La variable TablaEjercicio, recibirá los datos de los registros de la base de datos que hemos definido, a la que habíamos llamado basequímica



```
' Gambas class file

Public contadorgeneralbien As Integer
Public contadorgeneralmal As Integer
Private Conexion As Connection
Private TablaEjercicio As Result
```

Figura 90

En el evento Open del form Inicio, en el que ya existían códigos que se ejecutaban al abrirse el formulario, agregaremos códigos que mostramos en la Figura 91. Hemos dejado indentado las línea de código introducidas para programar la base de datos

```
Public Sub Form_Open()
    Dim res As Integer
    Dim reg As Integer

contadorgeneralbien = 0
contadorgeneralmal = 0
ComboBox1.Add("Glúcidos")
ComboBox1.Add("Lípidos")
ComboBox1.Add("Aminoácidos")
ProgressBar1.Width = 0


    Conexion = New Connection
    Conexion.Type = "sqlite3"
    Conexion.Host = "/media/d326c7b6-4226-42b2-b413-a56dbcc88137/gambas/CursoLinux/CursoLinux56"  'directorio donde está la base de datos
    Conexion.Name = "datosquimica"    'nombre de la base de datos
    Try Conexion.Open()
        If Error Then
        Message.Error("Error al conectar a la base de datos.")
        Conexion = Null
        Else
        TablaEjercicio = Conexion.Exec("Select * from ejercicios")
        If TablaEjercicio.Available Then MostrarCampos
        'MostrarCampos función que carga los campos en textbox
        End If
    'el código siguiente carga el campo apellido de la base datos en el DataCombo1
    res = TablaEjercicio.Count »'cuenta los registros de TablaEjercicio
    While reg < res
    ComboBox2.Add(TablaEjercicio!nombre)
    TablaEjercicio.MoveNext
    reg = reg + 1
    Wend
End
```

Figura 91

declaramos localmente en el evento dos variables de tipo Integer, que son res y reg. Recuerde que puede utilizar el nombre que desee. Veremos su uso más adelante.

Luego siguen los códigos

Conexion= New Connecion

Inicializa la variable Conexion como una nueva variable del tipo Connection

Conexion.Type: "sqlite3"

Asigna a la variable Conexion el tipo de base de datos, que en nuestro caso es del tipo sqlite3

Las dos líneas siguientes son donde está alojada la base de datos y cual es el nombre de ella

Conexion.Host= "/media/d326 ....."

Conexion.Name= "datosquímica"

Es probable que a esta altura del desarrollo se haya olvidado de algunas cosas. Podemos buscarla en el proyecto en el Item Connections > Connetion1 > properties. Se abrirá una ventana como muestra la Figura 92. Allí vemos el Path, que necesitamos para Conexion.Host y en Database, hallamos el nombre

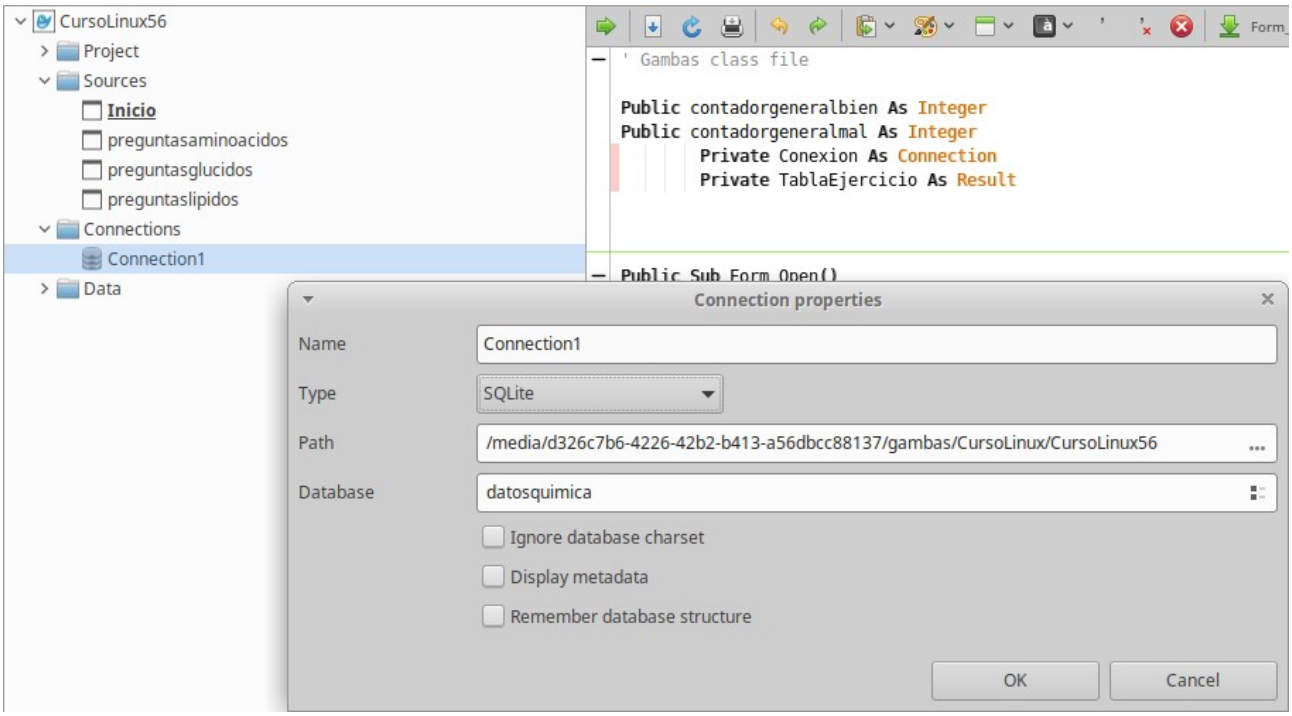


Figura 92

Las líneas, las analizaremos en conjunto y para organización las numeraremos

```
1 Try Conexion.Open()
2         If Error Then
3         Message.Error("Error al conectar a la base de datos.")
4         Conexion = Null
5         Else
6         TablaEjercicio = Conexion.Exec("Select * from ejercicios")
7         If TablaEjercicio.Available Then MostrarCampos
8         'MostrarCampos función que carga los campos en textbox
9         End If
```

Linea 1: Try Conexion.Open() abrirá la base de datos

Línea 2: estructura IF THEN ELSE.

Línea 3: Si hay un error nos mostrará "Error al conectar a la base de datos."

Línea 6: Si no hay error a la variable TablaEjercicio le asignará todos los campos de la tabla ejercicio. Esto se expresa con el texto: "Select * from ejercicios". El * indica todos los campos.

Línea 7: si la variable TablaEjercicio está disponible, ejecutará una rutina que llamamos "MostrarCampos", que veremos más adelante.

La última parte del código en el evento Open es

```
1       res = TablaEjercicio.Count  'cuenta los registros de TablaEjercicio
2       While reg < res
3       ComboBox2.Add(TablaEjercicio!nombre)
4       TablaEjercicio.MoveNext
5       reg = reg + 1
6       Wend
```

Acá tenemos una estructura WHILE-WEND para control de flujo. Esta estructura es un buble que se ejecutará mientras ocurra algo.

Fuera de la estructura WHILE WEND en la línea 1, inicializamos la variable res con el número de registros de la variable TablaEjercicio, que recordemos es una tabla que recibió campos y registros de la base de datos: datosquimica.

La Linea 2 inicia el bucle y dice que MIENTRAS reg sea menor que res, ejecutará los comandos de las líneas 3-5.

La variable res tendrá ahora el valor 2, porque el número de registros de nuestra base de datos es 2. Recuerde que incorporarmos a Juan Perez y José Garcia. La variable reg tiene el valor 0, ya que no la hemos inicializado.

La línea 2 dice: While reg < res ....

reg=0

res=2

se cumple la condición, entonces se ejecuta la línea 3, que lo que hace es agregar al ComboBox2, el contenido del campo nombre del primer registro de la TablaEjercicio.

En la linea 4 se mueve al segundo registro (MoveNext) de TablaEjercicio en búsqueda del segundo registro

En la línea 5 a la variable reg le suma 1, por lo que ahora tomará el valor 1.

Se choca con la sentencia Wend y regresa a la línea 2, evaluando si reg < res. Recordemos reg vale 1, y res vale 2. SE cumple la condición por lo tanto ejecuta nuevamente las líneas 3-5, quedando ahora reg en el valor 2. Cuando vuelva a la línea 2 no se cumplirá la condición reg < res, entonces finaliza el bucle.

Así el ComboBox2 quedará cargado con los nombres de los usuarios de la ejercitación.

Ahora pasemos a analizar el código del rutina MostrarCampos que llamamos en la línea 7, explicada anteriormente, Figura 93

```
Public Sub MostrarCampos()
        CheckBox1.Value = TablaEjercicio["aminoacidos"]
        CheckBox2.Value = TablaEjercicio["glucidos"]
        CheckBox3.Value = TablaEjercicio["lipidos"]
        aminoacidosnota.Text = TablaEjercicio["notaaminoacidos"]
        glucidosnota.Text = TablaEjercicio["notaglucidos"]
        lipidosnota.Text = TablaEjercicio["notalipidos"]
End
```

Figura 93

Al ejecutarse esta rutina, que se activa desde el evento Open del form Inicio, los CheckBox1, 2 y 3 tomarán valores True o False dependiendo que en la base de datos los campos correspondientes a aminoácidos, glúcidos y lípidos tengan ese valor.

La forma de asignar una valor al checkBox, como se ve tiene siempre el mismo formato. En la propiedad Value del CheckBox colocamos el contenido del campo correspondiente de la variable TablaEjercicio.

De la misma manera en los Labels aminoacidosnota, glucidosnota y lipidosnota, en su propiedad Text colocaremos los valores de los campos correspondientes.

La última parte del código es el que corresponde a la programación del evento Click del ComboBox2. Deseamos que cuando hagamos click sobre un nombre del ComboBox2, la variable TablaEjercicio se mueva al registro con ese nombre. Vemos este código en la Figura 94

```
Public Sub ComboBox2_Click()

    TablaEjercicio = Conexion.EXEC("SELECT * FROM ejercicios WHERE nombre = '" & ComboBox2.text & "'")
    MostrarCampos

End
```

Figura 94

Recordemos

La variable TablaEjercicio tiene los valores de los campos de la base de datos datosquímica. En este caso tiene dos registros: Juan Perez y José Garcia.

El código de la Figura 94 ahora a tabla ejercicio le asignará solo un registro a través de la función Conexion.EXEC. Dentro del paréntesis vemos la siguiente sentencia

"SELECT * FROM ejercicios WHERE nombre = '" & ComboBox2.text & "'"

Que se lee así: Seleccione todos los campos de la base de datos ejercicios donde el campo nombre toma el valor del texto seleccionado en el ComboBox2.

Una vez seleccionado solo ese registro, ejecutará la rutina MostrarCampos, es decir llenará los datos de los checkBox y Labels con los ejercicios realizados por el alumno y la nota de ellos.

Clase 7

Recordemos los pasos seguidos en la clase anterior

1- creamos una base de datos, una tabla y en ella definimos los campos. Además desde el administrador incorporamos dos registros

En nuestro caso la base de datos tiene el nombre **datosquímica** y la tabla el nombre **ejercicios.** Haciendo click sobre Connection1 y luego click con el botón derecho podemos elegir Open o Properties que nos darán acceso a la tabla, sus campos y registros, o el nombre y ubicación de la base de datos Figura 95.

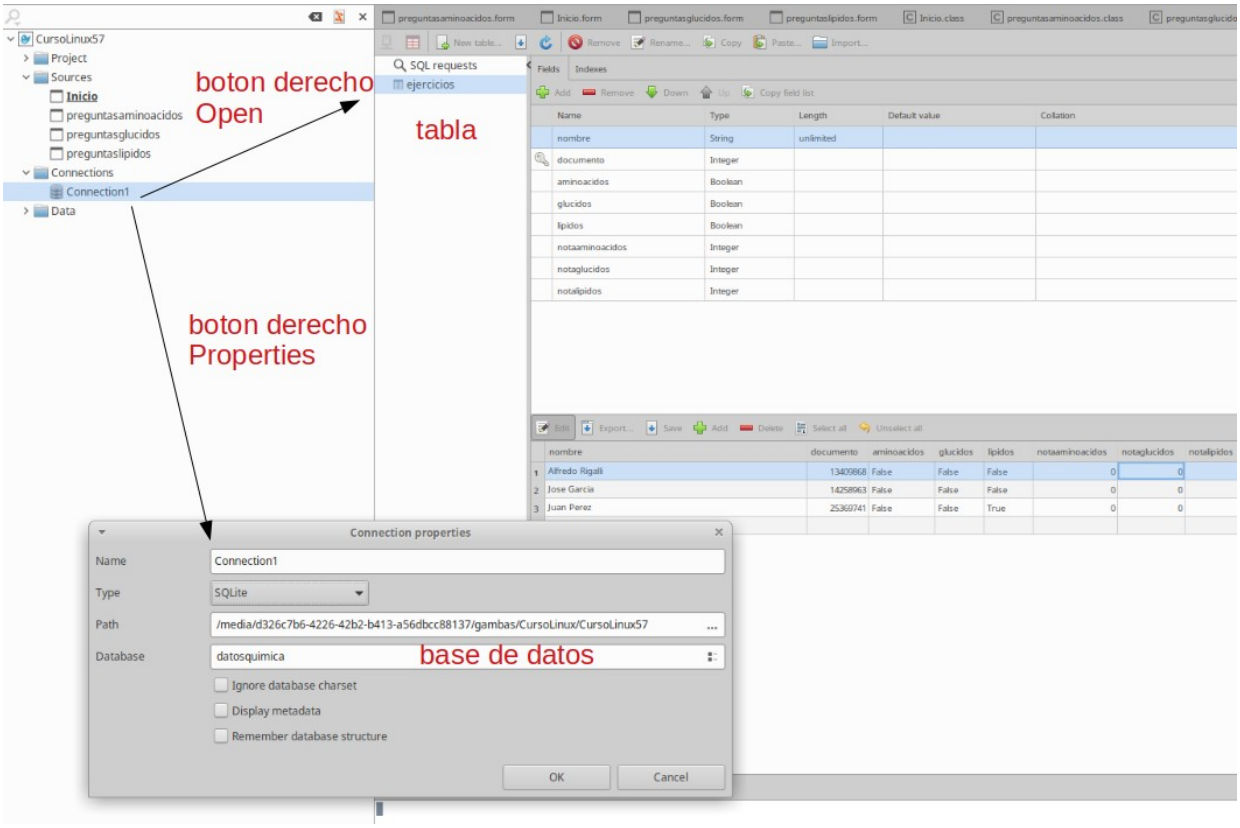


Figura 95

2- Incorporamos la base de datos a nuestra aplicación y desde allí accedimos a los registros.

Definimos dos variables del formulario, una que establece la conexión con la base y otra que alojará los registros de la base

```
Private Conexion As Connection
Private TablaEjercicio As Result
```

Dentro del evento Open del formulario Inicio colocamos el siguiente código, con el que inicializamos la variable TablaEjercicio con los datos de nuestra base de datos. Además de declararon variables que serán utilizadas en el proceso de mostrar los datos. En la tabla siguiente se muestra el código a la izquierda y se da una aclaración a cada línea de ser necesario

código	interpretación

Public Sub Form_Open()
Dim res As Integer
Dim reg As Integer
Conexion = New Connection
Conexion.Type = "sqlite3"
Conexion.Host = "/media/d326c7b6-4226-42b2-b413-a56dbcc88137/gambas/CursoLinux/CursoLinux57"
Conexion.Name = "datosquimica"
Try Conexion.Open()
If Error Then
Message.Error("Error al conectar a la base de datos.")
Conexion = Null
Else
TablaEjercicio = Conexion.Exec("Select * from ejercicios")
If TablaEjercicio.Available Then MostrarCampos
End If
	las siguientes instrucciones se ejecutarán al abrirse el formulario Inicio
	Se declara la variable res que será utilizada más adelante
	Se declara la variable reg que será utilzada más adelante
	se crea una nueva conexión
	se indica el tipo de base de dato
	se indica el directorio en que está la base de datos. El mismo de la aplicación en este caso
	se da el nombre de la base de datos
	se inicia el proceso de conexión
	Si hubiera algun error en el proceso se mostraría "Error al conectar a la base de datos"
	no realiza conexión
	en caso que no hubiera error ejecuta lo siguiente
	Inicializa la variable TablaEjercicio con todos los campos de la tabla ejercicio
	Si la variable TablaEjercicio queda disponible ejecuta la rutina "MostrarCampos, que cargará los valores de los campos de cada registros en TextBox y Labels, para que podamos ver los valores.

3- Cargamos los nombres de cada registro en un ComboBox. El código tiene un bucle que en cada vuelta carga el campo nombre de cada persona ComboBox2. Este nombre se halla en la variable TablaEjercicio que aloja los datos de la tabla ejercicios de la base de datos datosquímica. Este código se halla a continuación del anterior y se ejecuta al abrir el formulario inicio

código	interpretación
res = TablaEjercicio.Count While reg < res ComboBox2.Add(TablaEjercicio!nombre) TablaEjercicio.MoveNext reg = reg + 1 Wend	inicializa la variable res con el número total de registro de la base de datos Inicia un bucle que se ejecutará mientra la variable reg sea menor que el valor de la variable res, que fue inicializa al principio del evento Open agrega al ComboBox2 el campo nombre de la base de datos se mueve al registro siguiente de la base de datos a la variable reg le suma una unidad vuelve al principio del bucle y evalua si reg<res. Si lo es ejecuta el buble nuevamente, si no lo es sale del bucle y sigue con la retina

4- Al hacer click en el ComboBox2 sobre el nombre de uno de los usuarios, se selecciona solo ese registro y se ejecuta la rutina MostrarCampos, que carga los datos en los CheckBox y los labels mostrándonos que ejercicios se hallan realizados y cuales no.

código	interpretación
Public Sub ComboBox2_Click() TablaEjercicio = Conexion.EXEC("SELECT * FROM ejercicios WHERE nombre = '" & ComboBox2.text & "'") MostrarCampos End	Se ejecutará el código siguiente al hacer click en un nombre del ComboBox2 Inicializa la variable TablaEjercicio solo con el registro de la base de datos cuyo nombre es el que figura en el ComboBox2 Ejecuta la rutina MostrarCampos, que se halla en el formulario inicio

Incorporación de nuevo registro

Veremos como ingresar un nuevo registro, es decir incorporar el nombre y el documento de un nuevo usuario de nuestra aplicación.

Los otros campos se irán completando a medida que se realicen las ejercitaciones. Incorporamos a nuestra aplicación un objeto Button, al que le colocamos el text: Nuevo, Figura 96. Recuerde que los objetos como este del tipo Button tiene un Name: en este caso Button6 y un texto, que en este caso se visualiza en la interfaz gráfica. La elección de Name y Text depende del usuario, pueden coincidir o no. Si no se asigna la propiedad texto, mostrará en la interfaz el Name.

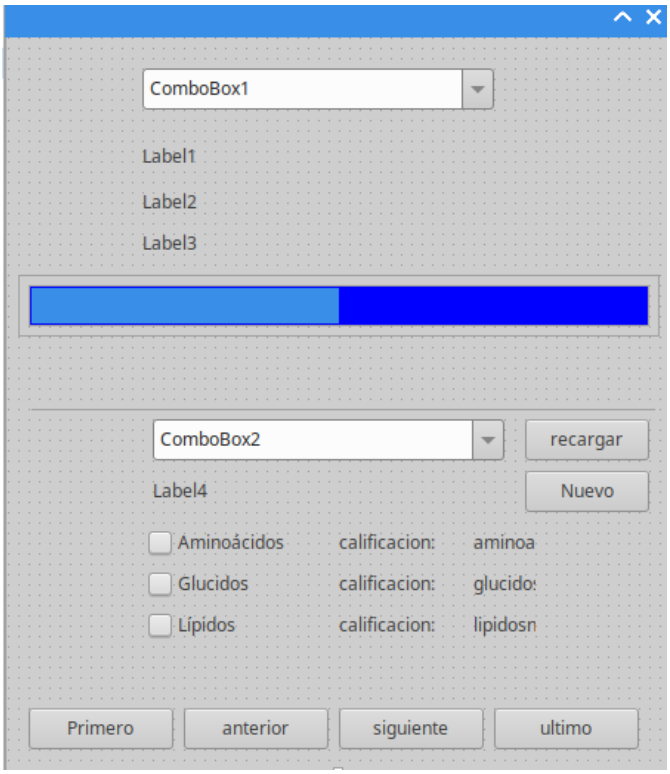


Figura 96

Cuando hagamos click en el Button6: Nuevo, nos permitirá incorporar un nuevo registro. Analicemos el código, en el que utilizaremos InputBox, para ingresar los datos.

codigo	explicacion
Public Sub Button6_Click()	
Dim Nuevo As Result	declaración variable Nuevo como tipo Result
Dim res As Integer	res tomará el número de registros de la base de datos
Dim reg As Integer	reg es un contador que tomará valores desde 0 a res
Nuevo = Conexion.Create("ejercicios")	Crea un nuevo registro en la tabla ejercicios
Nuevo["documento"] = InputBox("Ingrese documento sin puntos”, “", 0)	Al campo documento le asigna el valor introducido por un inputbox
Nuevo["nombre"] = InputBox("Ingrese nombre y apellido”, “", 0)	al campo nombre de la tabla ejercicios le asigna el nombre introducido por el inputbox
Nuevo["aminoacidos"] = False	al campo aminoacidos lo llena con False ya que no se hicieron aun los ejercicios de ese tema
Nuevo["glucidos"] = False	De igual manera con el campo glucidos
Nuevo["lipidos"] = False	De igual manera con el campo lipidos
Nuevo["notaaminoacidos"] = 0	notaaminoacido se carga con 0

Nuevo["notaglucidos"] = 0	igual para notaglucidos
Nuevo["notalipidos"] = 0	igual para notalípidos
Nuevo.Update	Se actualiza la tabla Nuevo
TablaEjercicio = Conexion.Exec("Select * from ejercicios")	Selecciona todos los campos de la tabla ejercicio
ComboBox2.Clear	Borra todos los nombres del ComboBox2
res = TablaEjercicio.Count	asigna a res el número de registros de TablaEjercicio
While reg < res	Inicia un bucle que ocurre mientras reg<res
ComboBox2.Add(TablaEjercicio!nombre)	carga al ComboBox2 el primer nombre de TablaEjercicio
TablaEjercicio.MoveNext	Se mueve al segundo registro
reg = reg + 1	suma una unidad al reg
Wend	regresa a While y evaluá si reg<res. Si lo es ejecuta nuevamente la rutina, sino deja de hacerlo.
End	

Es decir que al oprimir el Button6: Nuevo, se carga un nuevo registro y se cargan nuevamente los nombre el el ComboBox2, incluyendo el nuevo registro.

De esta manera tendremos un usuario nuevo. A continuacion veremos como cargar información en los otros campos de un usuario. Por ejemplo: si un usuario utiliza la aplicación por primera vez, se registrará oprimiendo Nuevo. De esta manera quedará incluido en la base de datos. Cuando realice ejercicios y en base a como está diseñada la aplicación, el usuario debería seleccionar su nombre del ComboBox2. De no hacerlo su tarea realizada será asignada al registro que esté seleccionado, que normalmente es el primer registro de la base de datos. En el mencionado proceso de pulido y optimización se deben tener en cuenta estos detalles. Veremos al finalizar esta clase una forma sencilla para evitar éste, una de las mil formas de fallar.

Actualización de campos

Deseamos que cuando un usuario realice un ejercicio en su registro se coloque True en el campo correspondiente a la ejercitación realizada (aminoacidos, glucidos o lipidos) y se coloque la nota en el campo correspondiente (notaaminoacidos, notaglucidos, notalipidos).

Tenemos varios pasos que listamos y explicamos brevemente

1- Declaramos dos variables que nos indicarán la ejercitación realizada y almacenarán temporariamente la nota de cada ejercitación. Estas variables las llamaremos nota y tema.

2- Inicializamos la variable tema con un número que nos indique que ejercitación haremos y actualizamos el campo correspondiente.

3- Inicializamos la variable nota con el valor que surge de las preguntasbien y mal respondidas en cada ejercitación

4- con los datos de la variable nota actualizaremos el campo correspondientes a la nota.

Paso 1. En la Figura 97, vemos la declaración de variables del class file del formulario Inicio. Tenemos dos variables

nota, del tipo integer, que en el momento de realizar una corrección de algunas de las ejercitaciones, tomará el valor de la calificación.

tema, del tipo intege, que tomará un valor diferente para cada ejercitación y que coincidirá con el indice del ComboBox1, donde se listan los ejercicios.

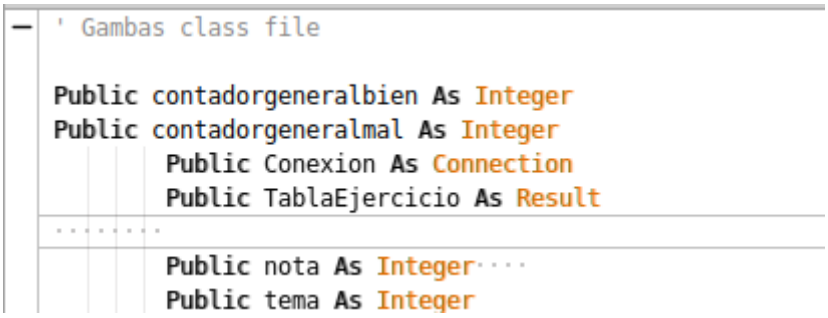


Figura 97

Por el momento solo las hemos declarado.

Paso 2. La inicialización de la variable tema se hará al elegir del ComboBox1 del formulario Inicio, el tema a ejercitar. En la Figura 98 se muestra el código que se ejecuta al hacer click sobre un ítem del ComboBox1. Este evento ya tenía códigos asignados en las aplicaciones de las clases anteriores y nos permitían acceder a los formularios de cada ejercitación. Para hacerlo utilizamos la estructura de control de flujo del tipo SELECT CASE.

Debemos recordar que tenemos una variable local del evento, a la que llamamos indice y que toma el valor del indice del ComboBox, que no es otra cosa que el orden en que lo vemos, y que toma valores de 0 en adelante. Si hacemos click en el primer ítem del ComboBox1, indice tomará el valor 0. Si lo hicieramos en el segundo ítem tomaría el valor 1 y así sucesivamente. La estructura Select Case ejecutará Case 0, Case 1 o Case 2, segun hagamos click en el primero (aminoácidos), el segundo ítem (glúcidos) o el tercero (lípidos)

Focalicemos el análisis en uno de ellos

```
Public Sub ComboBox1_Click()
Dim indice As Integer
Dim Nuevo As Result

indice = ComboBox1.Index

Select Case indice
  Case 0
    Nuevo = Conexion.Edit("ejercicios", "documento= " & TablaEjercicio["documento"])
    Nuevo["aminoacidos"] = True
    Nuevo.Update
    tema = indice
    Inicio.Hide
    preguntasaminoacidos.Show

  Case 1
    Nuevo = Conexion.Edit("ejercicios", "documento= " & TablaEjercicio["documento"])
    Nuevo["glucidos"] = True
    Nuevo.Update
    tema = indice
    Inicio.Hide
    preguntasglucidos.Show
  Case 2
    Nuevo = Conexion.Edit("ejercicios", "documento= " & TablaEjercicio["documento"])
    Nuevo["lipidos"] = True
    Nuevo.Update
    tema = indice
    Inicio.Hide
    preguntaslipidos.Show
End Select



End
```

Figura 98

La estructura que se describe dentro de Case 0, que correspondería si hubieramos elegido realizar la ejercitación de aminoácidos, se analiza a continuación

Código	Estructura
Case 0	El código siguiete se ejecutará solo si elegimos el primer ítem del ComboBox1
Nuevo = Conexion.Edit("ejercicios", "documento= " & TablaEjercicio["documento"])	Edita el registro de la TablaEjercicio donde el documento toma el nombre del usuario en uso
Nuevo["aminoacidos"] = True	Al campo aminoacido le asigna el valor True, indicando así que ejecutó esa ejercitación
Nuevo.Update	
tema = indice	la variable tema se incializa con el valor del indice, que en este caso sería : 0 e indica que estamos haciendo los ejercios de aminoácidos
Inicio.Hide	Oculta el formulario Inicio
preguntasaminoacidos.Show	Nos muestra el formulario preguntasaminoacidos

Paso 3. Realizamos la ejercitación correspondiente, en este caso aminoácidos. Al finalizar oprimiremos el botón "Oprima para controlar puntaje", que nos contará las preguntas bien y mal respondidas, asignado dichos valores a las variables locales preguntasbien y preguntasmal, respectivamente.

Cuando oprimamos el botón "Salir" (cuyo Name es Button1) , la variable nota tomará el valor de la calificación y como es una variable global disponible para todos los formularios, se llevará ese dato, Figura 99.

```
Public Sub Button1_Click()
  Inicio.nota = contadorbien * 10 / (contadorbien + contadormal)
  preguntasaminoacidos.Close
  Inicio.Show

End
```

Figura 99

A continuación analizamos dicho código

Public Sub Button1_Click()
  Inicio.nota = contadorbien * 10 / (contadorbien + contadormal)
  preguntasaminoacidos.Close
  Inicio.Show
End
	las instrucciones siguiente se ejecutarán solo si hacemos click en Button1
	La variable nota se inicializa con la calificación en una escala de 0-10
	se cierra el formulario preguntasaminoacidos
	se vuelve a mostrar el formulario Inicio

Paso 4. Luego de ejecutar Salir de cada formulario de ejercitación, volvemos al formulario Inicio con dos variables inicializadas:

tema: que nos indica qué hemos ejercitado

nota: qué calificación hemos obtenido

Al producirse el evento Show del formulario Inicio, se deberá actualizar el campo con la nota de la ejercitación correspondiente. En la Figura 100, resaltado se ve el código que especificamente realizará lo deseado. Las otras partes corresponden a diseños anteriores que nos permitia mostrar la nota y un ProgressBar.

```
Public Sub Form_Show()
Dim Nuevo As Result

Label1.text = "PREGUNTAS BIEN:  " & Inicio.contadorgeneralbien
Label2.text = "PREGUNTAS MAL:  " & Inicio.contadorgeneralmal
    If (Inicio.contadorgeneralbien + Inicio.contadorgeneralmal) > 0
    Label3.text = "SU CALIFICACIÓN ES:" & (Inicio.contadorgeneralbien * 10 / (Inicio.contadorgeneralbien + Inicio.contadorgeneralmal))
    ProgressBar1.Width = 420 * (Inicio.contadorgeneralbien + Inicio.contadorgeneralmal) / 12
    Nuevo = Conexion.Edit("ejercicios", "documento= " & TablaEjercicio["documento"])
    If (tema = 0) Then Nuevo["notaaminoacidos"] = Inicio.nota
    If (tema = 1) Then Nuevo["notaglucidos"] = Inicio.nota
    If (tema = 2) Then Nuevo["notalipidos"] = Inicio.nota
    Nuevo.Update
    Endif

End
```

Figura 100

A continuación analizamos el código específico de este sector.

código	explicación
Nuevo = Conexion.Edit("ejercicios", "documento= " & TablaEjercicio["documento"])	Edita el registro del usuario
If (tema = 0) Then Nuevo["notaaminoacidos"] = Inicio.nota	si la variable tema tuviera el valor 0, al campo notaaminoacidos le asigna la nota
If (tema = 1) Then Nuevo["notaglucidos"] = Inicio.nota	Si tema tuviera el valor 1, la nota la pondría en notaglucidos
If (tema = 2) Then Nuevo["notalipidos"] = Inicio.nota	si tema tuviera el valor 2, la nota la colocaría en notalipidos
Nuevo.Update	Actualiza la base de datos.

De esta manera cuando un usuario realiza una ejercitación, quedarán los datos almacenados y visible en los labels y textBox correspondientes. En la Figura 101, se puede ver para el usuario Juan Perez, que ha realizado las ejercitaciones Aminoácidos y Lípidos con calificaciones 10 y 6 respectivamente.

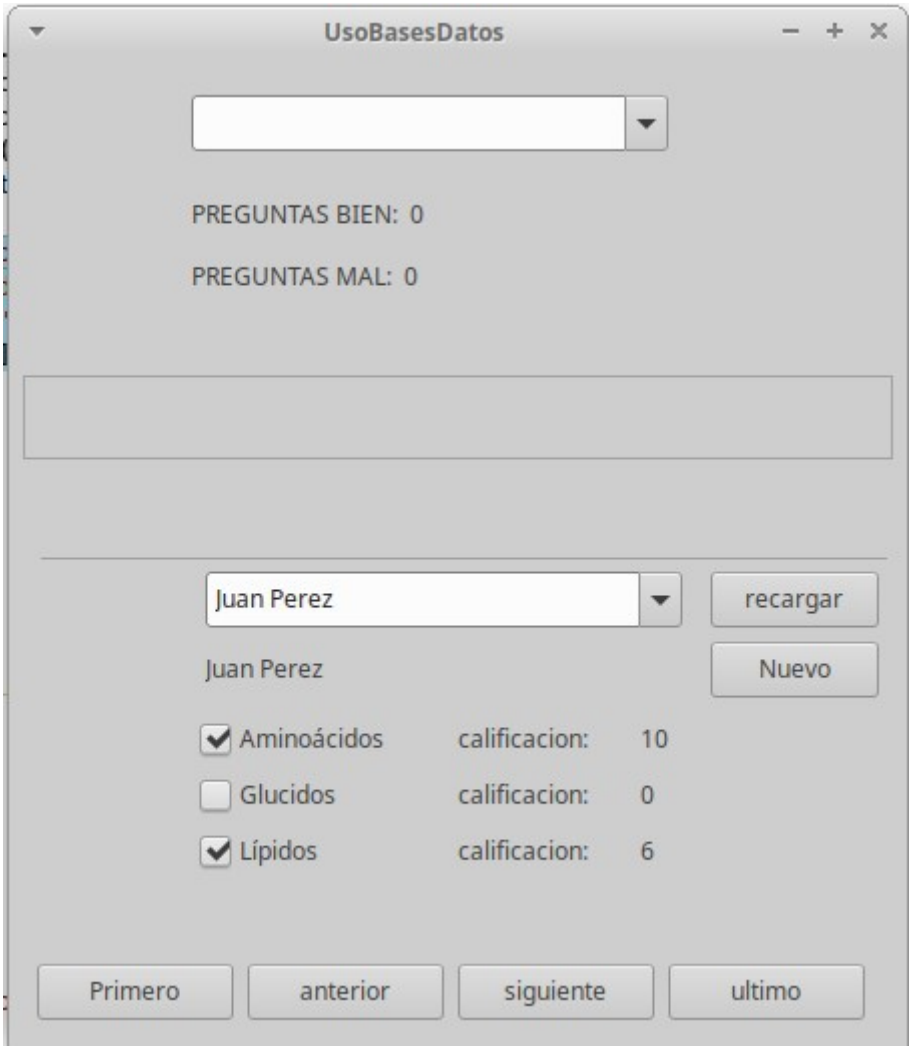


Figura 101

Control de posible errores

Como dijimos hay una sola forma de funcionar y miles de fallar. Veremos solo una a título ilustrativo. Un error previsible es que un usuario entre a la aplicación y haga una ejercitación. Si cualquier usuario hace una ejercitación, los datos de la misma serán cargados a Juan Perez, que es el usuario que queda seleccionado por defecto, según programamos, como se puede ver en la Figura 101.

La solución sencilla es impedir al usuario que elija de ComboBox1 la ejercitación si primero no eligió su usuario o se registro como "Nuevo".

Un recurso sencillo es adjudicar al ComboBox1 la Property: Enabled=False. De esta manera cuando se ejecute la aplicación ese ComboBox no estará accesible, aunque si visible, pero con un color gris característico de los objetos con Enabled=False, Figura 102.

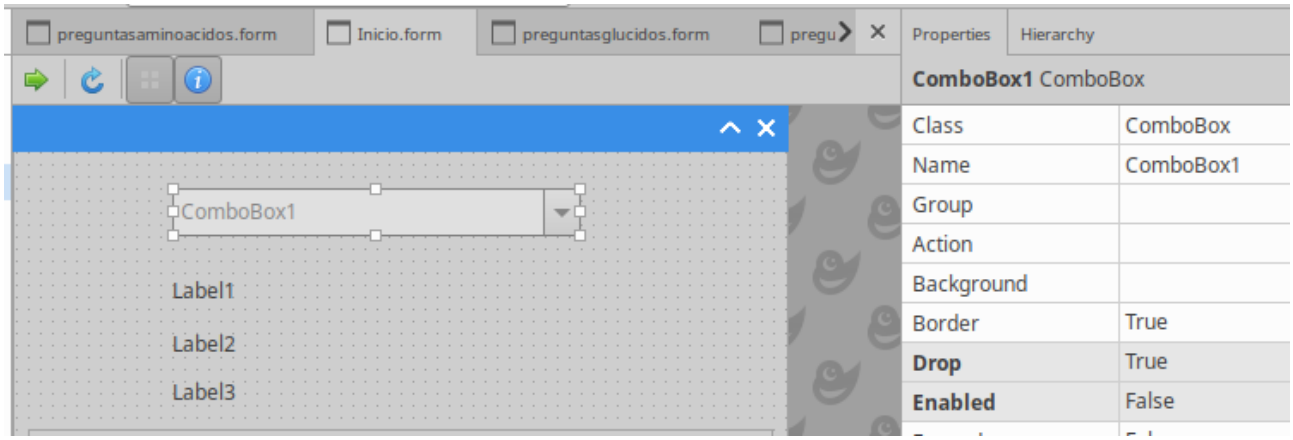


Figura 102

Entonces al ingresar el usuario tiene solo algunas posibilidades: Elegir un usuario del ComboBox2, agregarse como "Nuevo", moverse por la base de datos con los botones: primero, último, siguiente, anterior o salir, Figura 101.

Si elige un usuario del ComboBox1, programaremos que el ComboBox1, recién allí se active. Para ello agregamos la modificación de su propiedad Enabled, pasandola a valor True, Figura 103.

```
Public Sub ComboBox2_Click()

    TablaEjercicio = Conexion.EXEC("SELECT * FROM ejercicios WHERE nombre = '" & ComboBox2.text & "'")
    MostrarCampos
    ComboBox1.Enabled = True

End
```

Figura 103

Veremos al hacer click en el ComboBox2, que el ComboBox1, que estaba deshabilitado, ahora estará accesible, Figura 104.

Figura 104

Clase 8

En esta clase seguiremos con la aplicación que venimos desarrollando pero le incorporaremos algunas mejoras y agregados. A la aplicación desarrollada en la clase anterior y que llamamos CursoLinux57, la grabaremos como CursoLinux58.

Recuerde que para ello debe abrir la aplicación CursoLinux57 y elegir del menú de Gambas

File > save project as

se abrirá una ventana en que elige el directorio donde quiere grabar la nueva aplicación y se creará una carpeta con el nuevo desarrollo. En la Figura 105 vemos que grabaremos el nuevo proyecto con el nombre CursoLinux58, dentro de la carpeta CursoLinux. Esta acción creará automáticamente una nueva carpeta con el nombre CursoLinux58

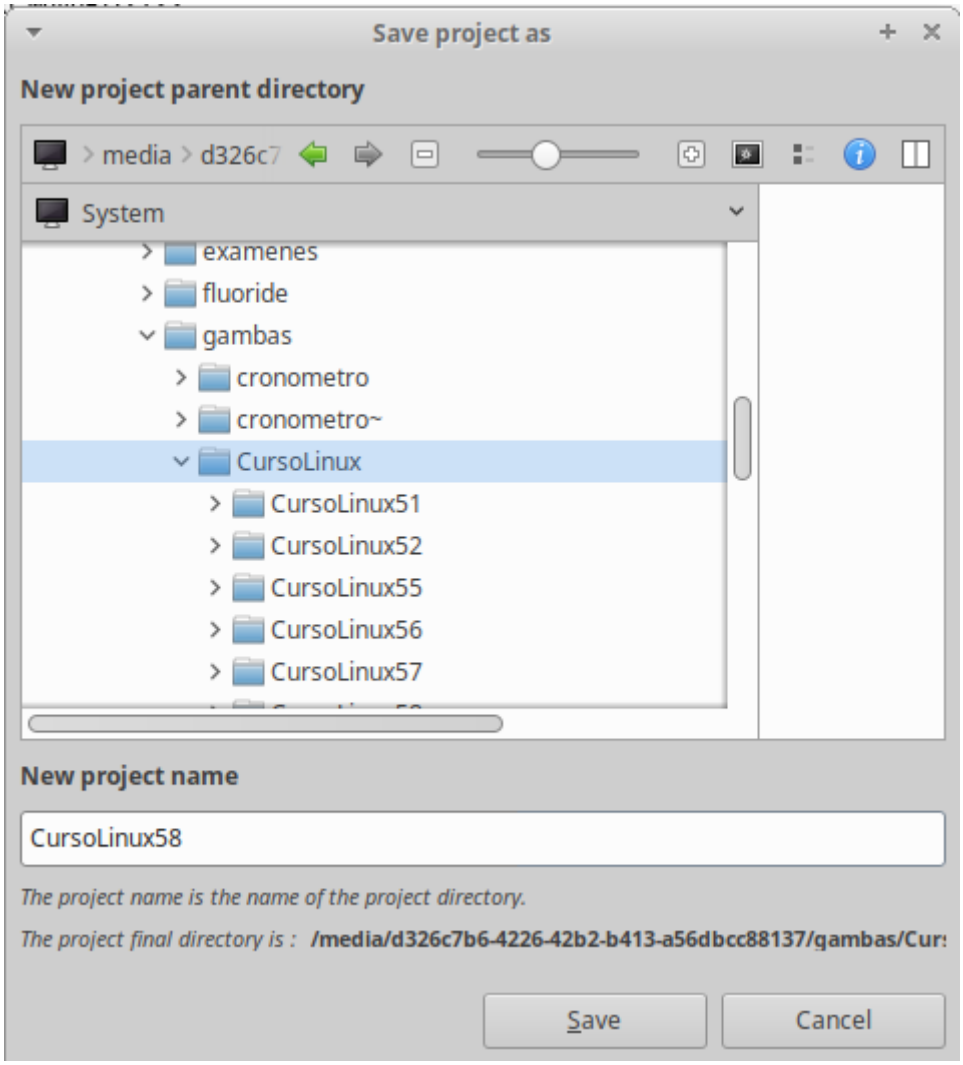


Figura 105

Ahora introduciremos algunas mejoras en esta aplicación

Insertar Menú de aplicación

Hasta el momento hemos manejado las acciones a través de objetos de tipo Button, ComboBox, RadioButton, etc. Ahora introduciremos menú desplegables como los que estamos acostumbrados a utilizar en todas las aplicaciones. Luego utilizaremos este menu para introducir nuevos temas de la clase.

Insertaremos un sencillo menú, en el formulario Inicio y con el ejemplo planteado usted podrá luego alcanzar sus propios objetivos. Posicionado en el formulario, para insertar un menú oprimimos el botón "Menu Editor" que se halla en la barra de tareas del formulario, Figura 106

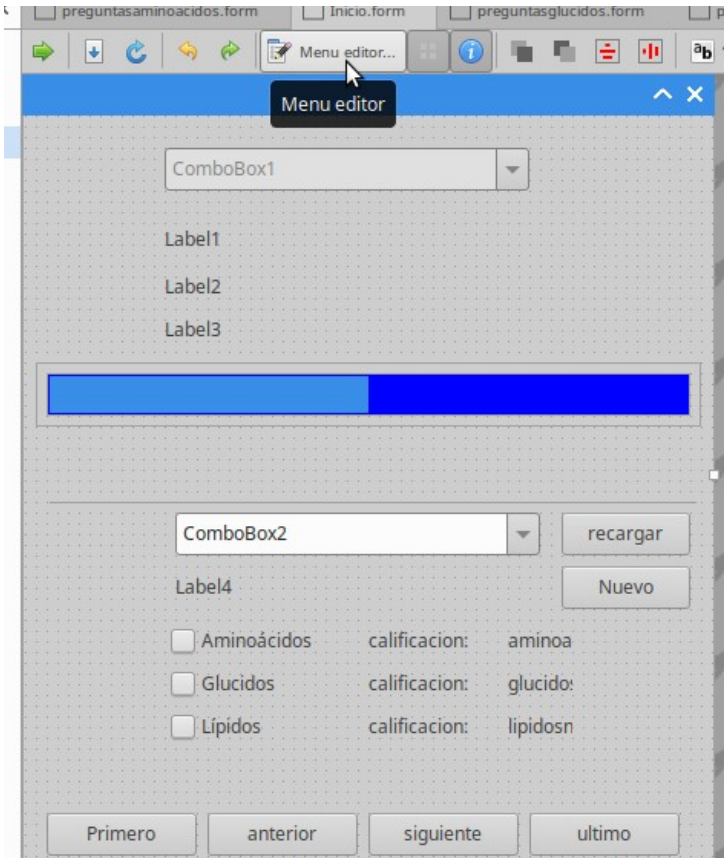


Figura 106

al realizar esa acción se abrirá un ventana como muestra la Figura 107

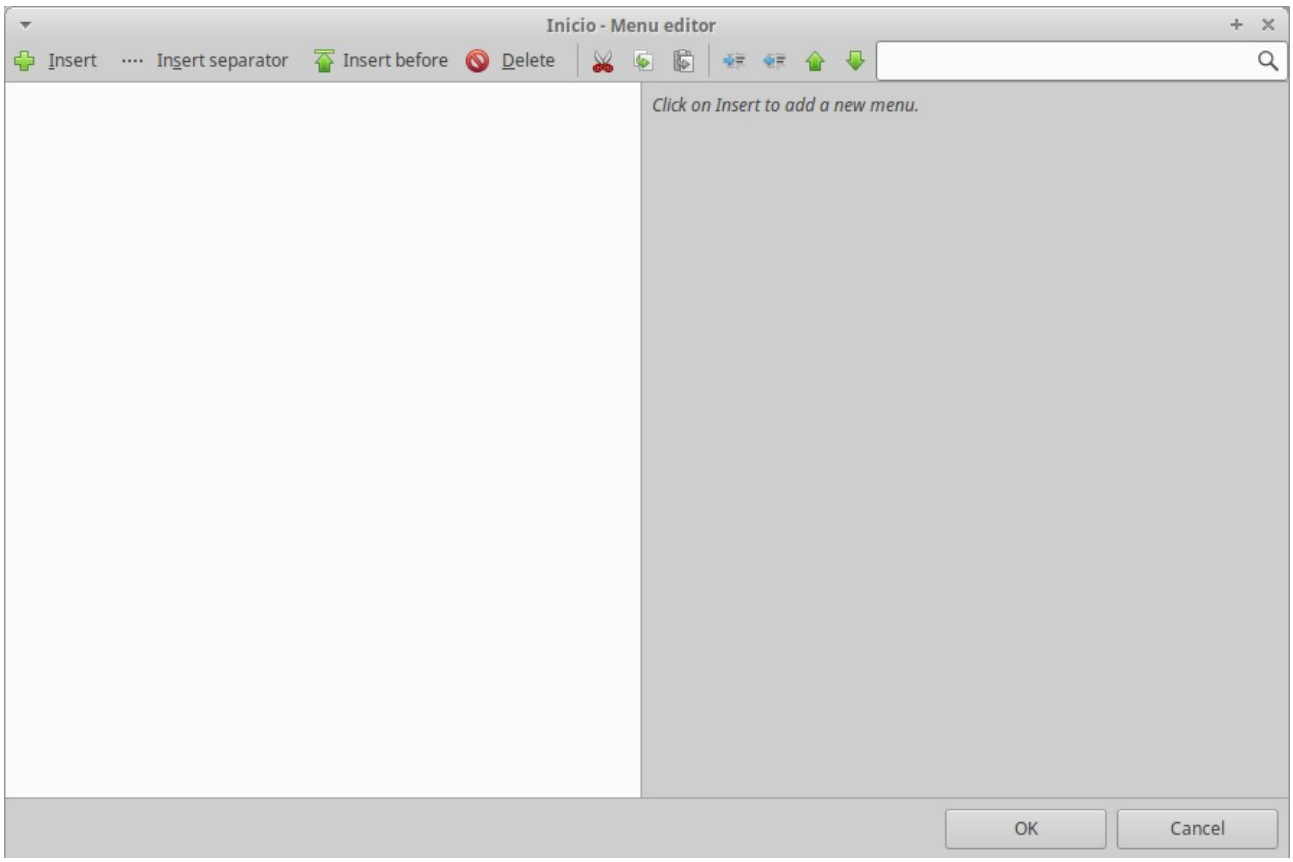


Figura 107

Para agregar un menú oprimimos "Insert" que se indica con un signo más de color verde. Incluiremos menú y submenús.

Por un lado incluiremos un menú con la siguiente estructura

Edición Base Datos

y dentro de este ítem incluiremos dos subítems

Insertar nuevo registro

Eliminar un registro

En otro item del menú que llamaremos

Acciones

que incluirá dos acciones

desconectar base de datos

Salir

Para esto incluimos oprimiendo Insert, cada uno de estos items. Al hacerlo verá que tiene dos sitios a llenar obligatorios: Name y Caption. Si no lo llena quedan con un nombre y caption por defecto. Name es el nombre con que se llamará para las rutinas y Caption, lo que veremos en el menú. En la Figura 108, dejamos el primer ítem con el Name: Menu1 pero en caption colocamos: Edición Base Datos.

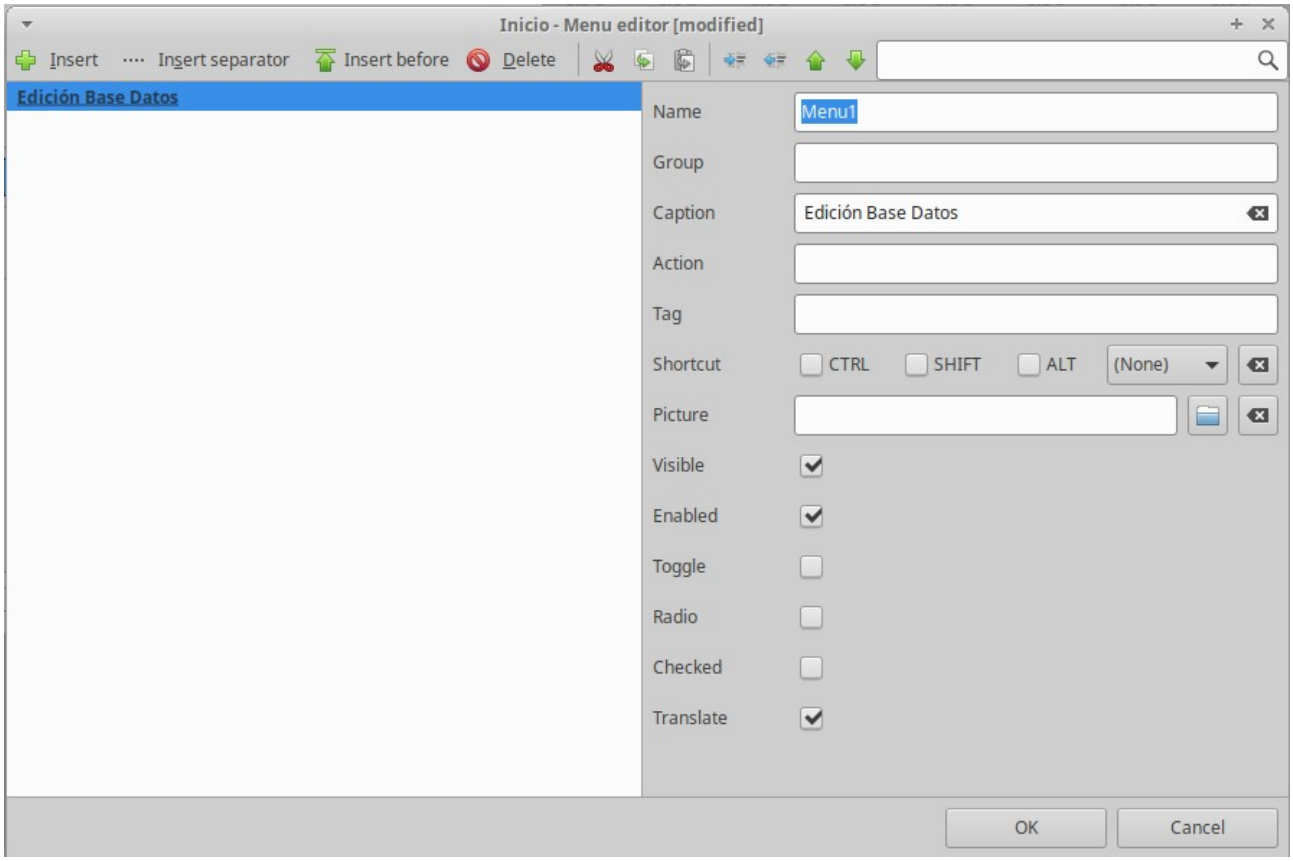


Figura 108

Así seguimos agregando los otros ítems, hasta que nos quedará lo que mostramos en la Figura 109

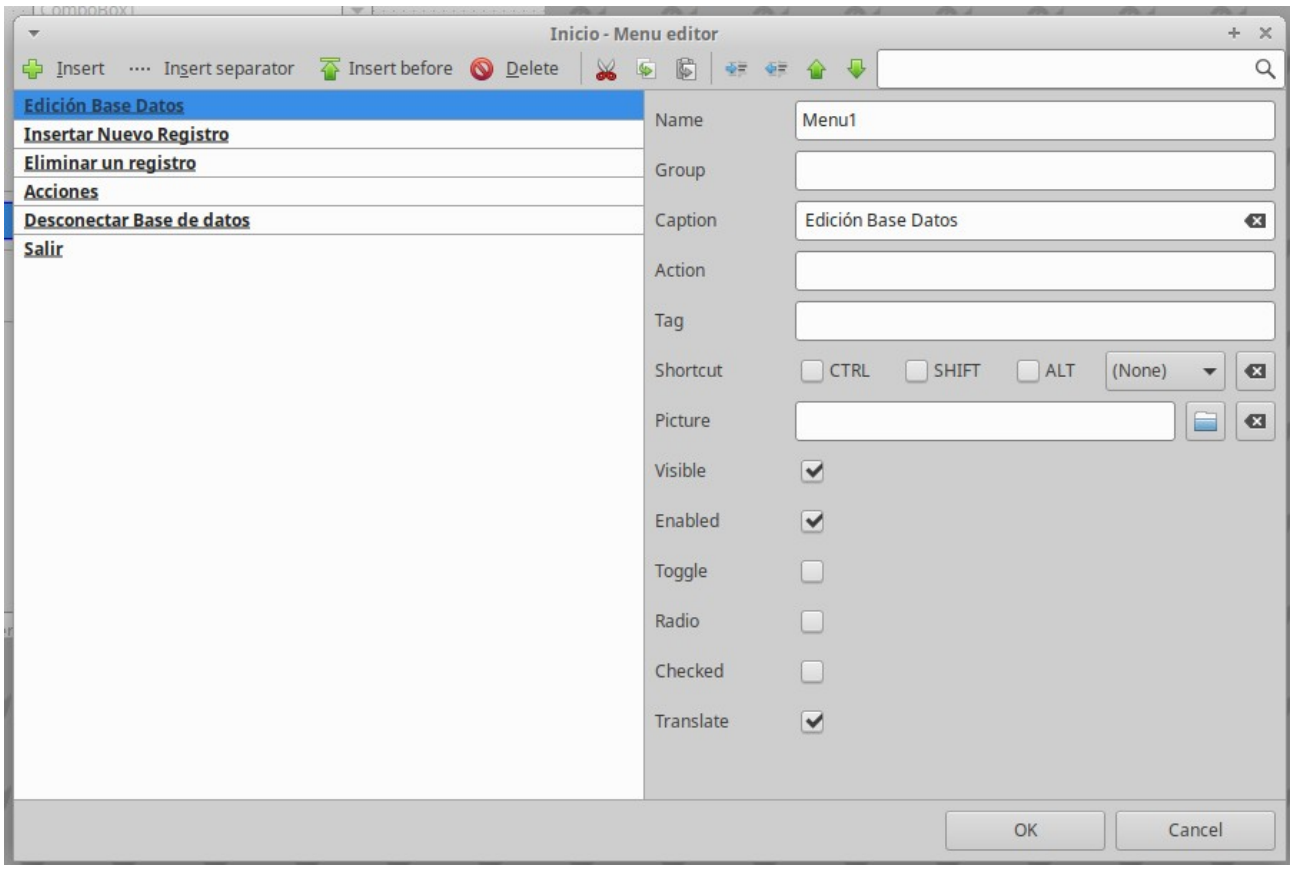


Figura 109

Ahora deseamos que Insertar nuevo Registro y Eliminar un registro queden como submenús de Edición Base Datos y por otro lado, Desconectar Base de datos y Salir, como submenús de Acciones. Para esto marcamos el menú, por ejemplo Insertar Nuevo Registro y oprimimos del menú la flecha azul hacia la derecha que indica indentar y lo pasará automáticamente a submenú, Figura 110.

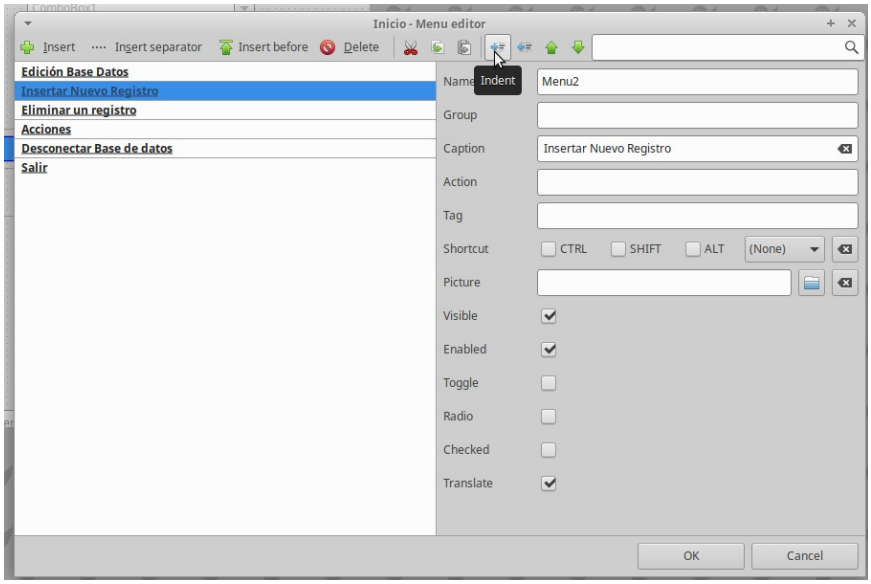


Figura 110

Luego de proceder de la misma manera con todos los items nos quedará configurado como muestra la Figura 111

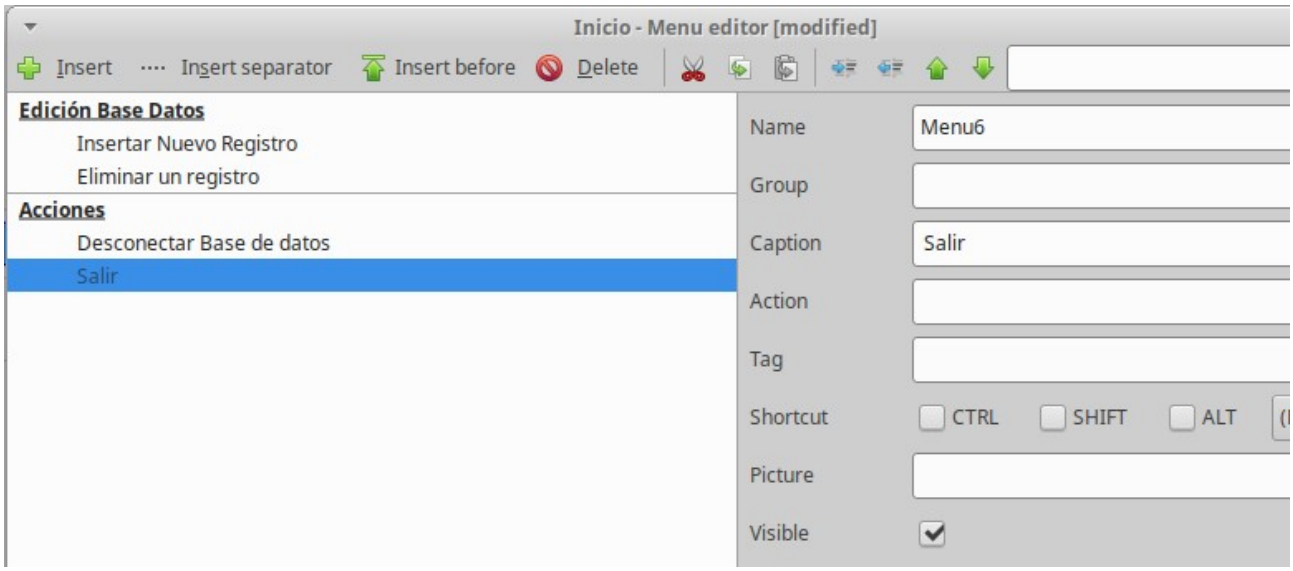


Figura 111

y nuestro formulario mostrará la siguiente forma, si hacemos click con el botón izquierdo del ratón sobre uno de los items

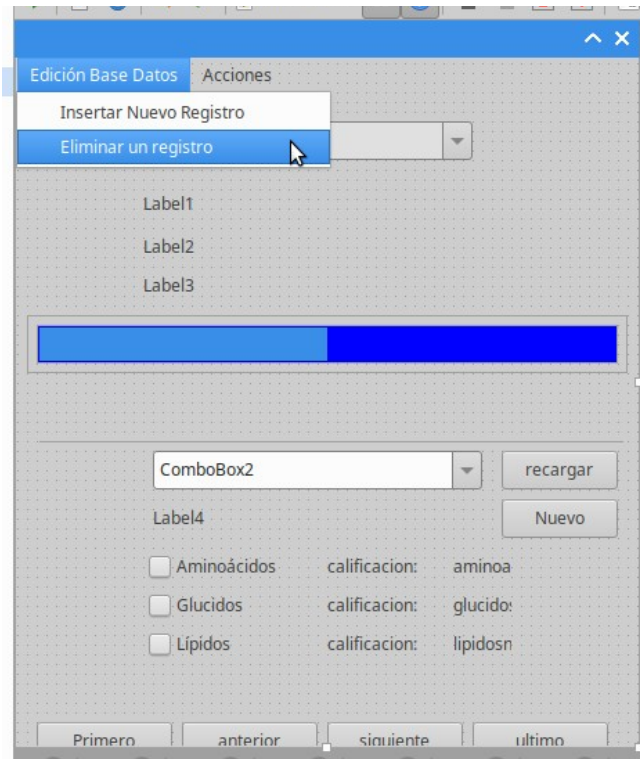


Figura 112

Ahora debemos programar para que cuando elijamos un ítem, ejecute una acción. Comenzaremos programando el submenú Insertar Nuevo REgistro. Esta acción fue programada en la clase anterior y el código necesario se encuentra en el Form Inicio, dentro del evento click del Button Nuevo. Asi que podemos hacer doble click sobre el Button Nuevo, acción que nos llevara al código para insertar un nuevo menu. Lo marcamos y lo copiamos al portapapeles con Ctrl + C, como muestra la Figura 113.

```
End

Public Sub Button5_Click()

  TablaEjercicio = Conexion.Exec("Select * from ejercicios")
   TablaEjercicio.MoveFirst()
  MostrarCampos
End

Public Sub Button6_Click()
Dim Nuevo As Result
Dim res As Integer
Dim reg As Integer

Nuevo = Conexion.Create("ejercicios")
Nuevo["documento"] = InputBox("Ingrese documento sin puntos", "", 0)
Nuevo["nombre"] = InputBox("Ingrese nombre y apellido", "", 0)
Nuevo["aminoacidos"] = False
Nuevo["glucidos"] = False
Nuevo["lipidos"] = False
Nuevo["notaaminoacidos"] = 0
Nuevo["notaglucidos"] = 0
Nuevo["notalipidos"] = 0
Nuevo.Update
TablaEjercicio = Conexion.Exec("Select * from ejercicios")
ComboBox2.Clear
res = TablaEjercicio.Count »'cuenta los registros de TablaEjercicio
            While reg < res
            ComboBox2.Add(TablaEjercicio!nombre)
            TablaEjercicio.MoveNext
            reg = reg + 1
            Wend

End

Console  Breakpoints  Notes
```

Figura 113

y este código lo pegamos dentro del item Insertar nuevo registro, del menú. Para ello hacemos click sobre el item, que nos llevará al class form, Figura 114

```
Public Sub Menu2_Click()

End
```

Figura 114

entre Public Sub Menu2 y End pegamos el código copiado, que deberá quedar como muestra la Figura 115

```
Public Sub Menu2_Click()

  Dim Nuevo As Result
Dim res As Integer
Dim reg As Integer

Nuevo = Conexion.Create("ejercicios")
Nuevo["documento"] = InputBox("Ingrese documento sin puntos", "", 0)
Nuevo["nombre"] = InputBox("Ingrese nombre y apellido", "", 0)
Nuevo["aminoacidos"] = False
Nuevo["glucidos"] = False
Nuevo["lipidos"] = False
Nuevo["notaaminoacidos"] = 0
Nuevo["notaglucidos"] = 0
Nuevo["notalipidos"] = 0
Nuevo.Update
TablaEjercicio = Conexion.Exec("Select * from ejercicios")
ComboBox2.Clear
res = TablaEjercicio.Count »'cuenta los registros de TablaEjercicio
.............
            While reg < res
            ComboBox2.Add(TablaEjercicio!nombre)
            TablaEjercicio.MoveNext
            reg = reg + 1
            Wend


End
```

Figura 115

En el menú Acciones y los submenús Salir y Desconectar base datos colocaremos de la misma manera el código correspondiente.

Para salir del programa, el código es: Quit

Para desconectar la base de datos: conexion.Close. Figura 116

```
Public Sub Menu6_Click()

  Quit

End

Public Sub Menu5_Click()

  Conexion.Close

End
```

Figura 116

Recuerde que conexion es una variable declarada al principio del documento y corresponde a una variable del tipo Connection.

Si desea agregar un ícono a un menú, marcamos el menú deseado, salir en el caso de la Figura 117. Luego hacemos click en el icono de carpeta a la derecha de "Picture", que desplegará una ventana con los íconos. Haremos doble-click sobre el deseado y se incorporará su nombre a la ventana y al menú. Luego debemos oprimir OK.

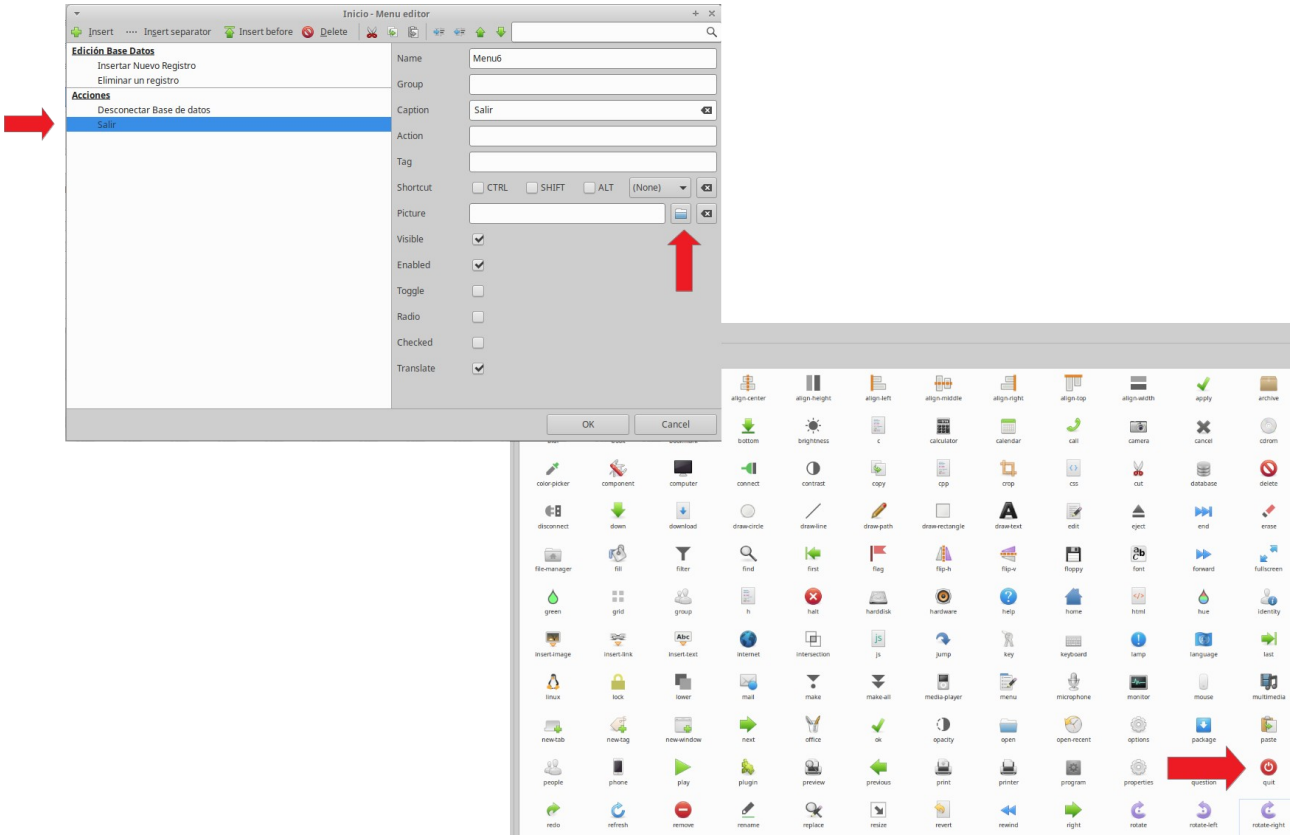


Figura 117. Las flechar rojas indican los sitios a clickear

Eliminación de registro de una base de datos

La eliminación de un registro es una acción necesaria en manejo de base de datos. En nuestro caso, si un alumno deja de ser parte de nuestra clase, podríamos eliminarlo. Por supuesto que también hay otras opciones, como por ejemplo agregar un campo que tenga como opciones activo-pasivo. Podríamos programar que nos muestre todos los alumnos, solo los activos o los pasivos. Recuerde que en programación todo es posible.

Veamos el código para eliminar un registro

código	interpretación
Public Sub Menu3_Click()	esta rutina se activa al hacer click sobre el menú eliminar un registro cuyo name es Menu3
Dim eliminado As Result	declaramos solo para este evento la variable eliminado
Dim res As Integer	declaramos para este evento la variable res
Dim reg As Integer	declaramos para el evento la variable reg
If Message.Question("¿Desea eliminar el registro?", "Si", "No")	Al hacer click en el menú nos aparecerá un cuadro de diálogo

= 1 Then	dándonos la posiblidad de eliminar o no un registro.
eliminado = Conexion.edit("ejercicios", "documento=" & TablaEjercicio["documento"])	Si oprimimos Si, la variable eliminado toma los datos del registro a eliminar
eliminado.Delete	Lo elimina
TablaEjercicio = Conexion.Exec("Select * from ejercicios")	carga en TablaEjercicio todos los campos de la tabla ejercicico
End If	
ComboBox2.Clear	limpia el ComboBox2, para volver a cargar los nombres
res = TablaEjercicio.Count	la variable res se inicializa con el número de registros de la tabla
While reg < res	Inicial un bucle que ocurrirá mientra reg sea menor que res
ComboBox2.Add(TablaEjercicio!nombre)	agrega al ComboBox2 el primer nombre de la base de datos
TablaEjercicio.MoveNext	se mueve al siguiente registro
reg = reg + 1	le suma una unidad a la variable reg
Wend	y retorna a la primer linea del bucle While evaluando si reg<res
	Si lo es inicia el bucle nuevamente. Si no lo es termina y llega a
End	End

Este código lo pegamos dentro del submenú, Eliminar un registro. De la misma manera que procedimos para Insertar nuevo registro, hacemos click sobre el submenú, que nos llevará al class form, que nos quedará como muestra la Figura 118

```
Public Sub eliminarRegisto_Click()

Dim eliminado As Result
Dim res As Integer
Dim reg As Integer

If Message.Question("¿Desea eliminar el registro?", "Si", "No") = 1 Then
eliminado = Conexion.edit("ejercicios", "documento=" & TablaEjercicio["documento"])
eliminado.Delete
TablaEjercicio = Conexion.Exec("Select * from ejercicios")
End If

ComboBox2.Clear
res = TablaEjercicio.Count »'cuenta los registros de TablaEjercicio
.............
            While reg < res
            ComboBox2.Add(TablaEjercicio!nombre)
            TablaEjercicio.MoveNext
            reg = reg + 1
            Wend


End
```

Figura 118

Captura de errores

Como dijimos todo lo que funciona tiene una única forma de hacerlo y miles de fallar. Disminuir las fallas es un trabajo progresivo, en el que se avanza rápido al principio corrigiendo aquellas cosas previsibles. Por ejemplo en la aplicación CursoLinux58, tenemos cuatro botones para movernos en la base de datos. Pruebe los mismos. Si oprimimos Ultimo o Primero nos lleva al último o primer registro. Posicinesé entonces en el primero y oprima sucesivamente el botón siguiente. Este botón nos mueve al siguiente registro. Es previsible que si lo oprimimos al llegar al último, se produzca un error. Compruébelo. El problema del error es que nos saca de la aplicación.

Algunos errores son anticipables y corregibles a través de código. Por ejemplo en este caso, podríamos hacer que el botón Siguiente pase a enabled.False, al llegar al final. No sería una tarea nada difícil. Inténtelo.

Otro error previsible es que un usuario intente registrarse dos veces. Como la base de datos tiene el documento como campo único, donde no puede haber dos registros con el mismo valor. Al introducir un valor igual a uno que existe dará un error. Esto no es sencillo de corregir por código y es más sencillo hacerlo con un capturador de errores. Estos recursos, nos muestras que hay un error, pero la aplicación no se detiene. Veamos el código introducido para evitar que un error cuelgue el software. Este código lo incluimos en insertar un nuevo registro, ya que allí se ingresa el documento. Este código lo hemos desarrollado en la clase anterior y ahora tendrá el agregado

codigo	explicacion
Public Sub insertarRegisto_Click()	se activa el código siguiente al hacer click en el submenu Insertar nuevo registro
Dim Nuevo As Result	Se declaran variables necesarias para la ejecución de esta rutina
Dim res As Integer	
Dim reg As Integer	
Nuevo = Conexion.Create("ejercicios")	se crea un nuevo registro
Nuevo["documento"] = InputBox("Ingrese documento sin puntos", "", 0)	a través de inputBox ingresamos el documento
Nuevo["nombre"] = InputBox("Ingrese nombre y apellido", "", 0)	a través de InputBox ingresamos el nombre
Nuevo["aminoacidos"] = False	se asigna False al campo aminoácidos
Nuevo["glucidos"] = False	False a glucidos
Nuevo["lipidos"] = False	False a lipidos
Nuevo["notaaminoacidos"] = 0	se asigna 0 a la notaaminoacidos
Nuevo["notaglucidos"] = 0	0 a notaglucido
Nuevo["notalipidos"] = 0	0 a notalipido
Try Nuevo.Update	Se antepone Try a Nuevo.Update, que es el punto donde se produce el error
If Error Then	Si se produce el error por documento repetido
Message.Error("Número de documento repetido")	aparece un mensaje de error, oprimimos OK y seguieremos
Endif	
TablaEjercicio = Conexion.Exec("Select * from ejercicios")	se seleccionan todos los campos de la tabla ejercicios
ComboBox2.Clear	sigue el código conocido.
res = TablaEjercicio.Count 'cuenta los registros de TablaEjercicio	
While reg < res	
ComboBox2.Add(TablaEjercicio!nombre)	

TablaEjercicio.MoveNext reg = reg + 1 Wend End	

Software de instalación

Llegamos al último paso de desarrollo de la aplicación. Esto es al momento en que tenemos que hacer los paquetes para distribuirla. Estos paquetes podrán ser enviados, subidos a la web, distribuidos en pendrive, etc, de manera que los usuarios lo puedan instalar en sus computadoras.

Gambas hace el trabajo por nosotros siguiendo los pasos que se detallan.

1- Abra la aplicación de la cual quiere construir los paquetes, en nuestro caso CursoLinux58. En el menú de tareas diríjase a

Project > make > Installation package, Figura 119

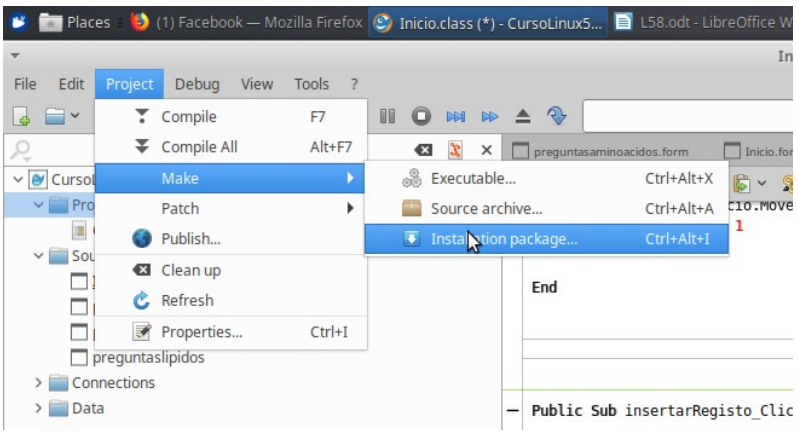


Figura 119

Se abrirá una ventana, en la que podrá cargar diversos datos. Algunos son cargados por defecto. Otros pueden quedar vacíos. Como podemos ver en la Figura 120, asigna un Package name, Package version. Nombre y dirección de quien lo mantiene, una descripción del software y otros detalles. Oprima Next finalizada esta ventana

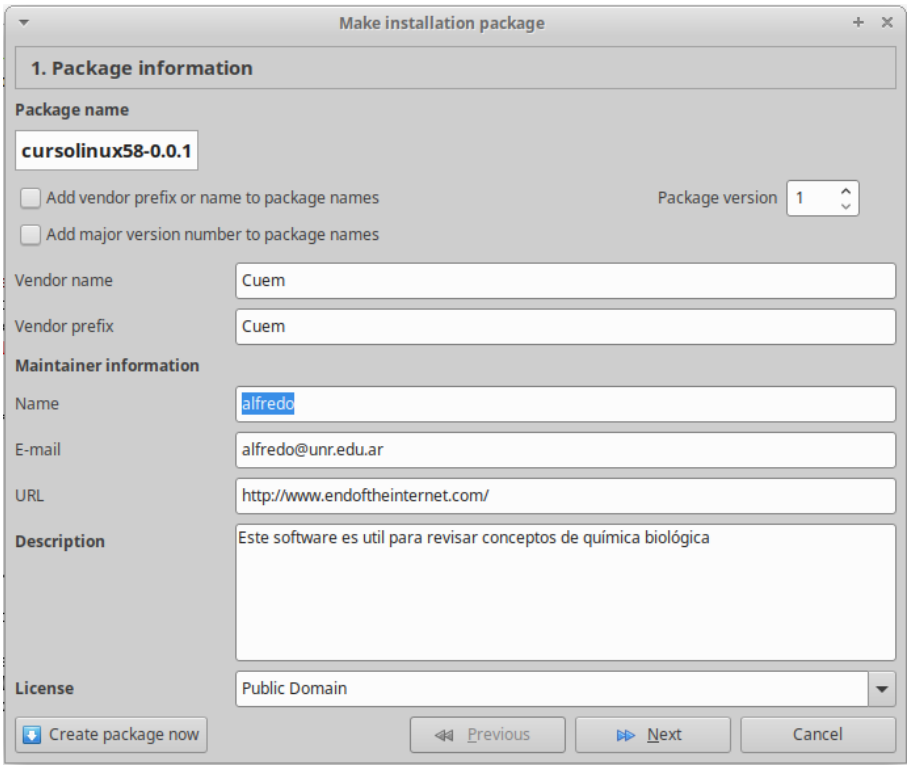


Figura 120

pasamos así a la segunda ventana. En el primer cuadro de texto coloque los detalles de esta edición del software, en la de abajo queda un registro cronológicos de las anteriores versiones de la aplicación, Figura 121. Completado oprima Next

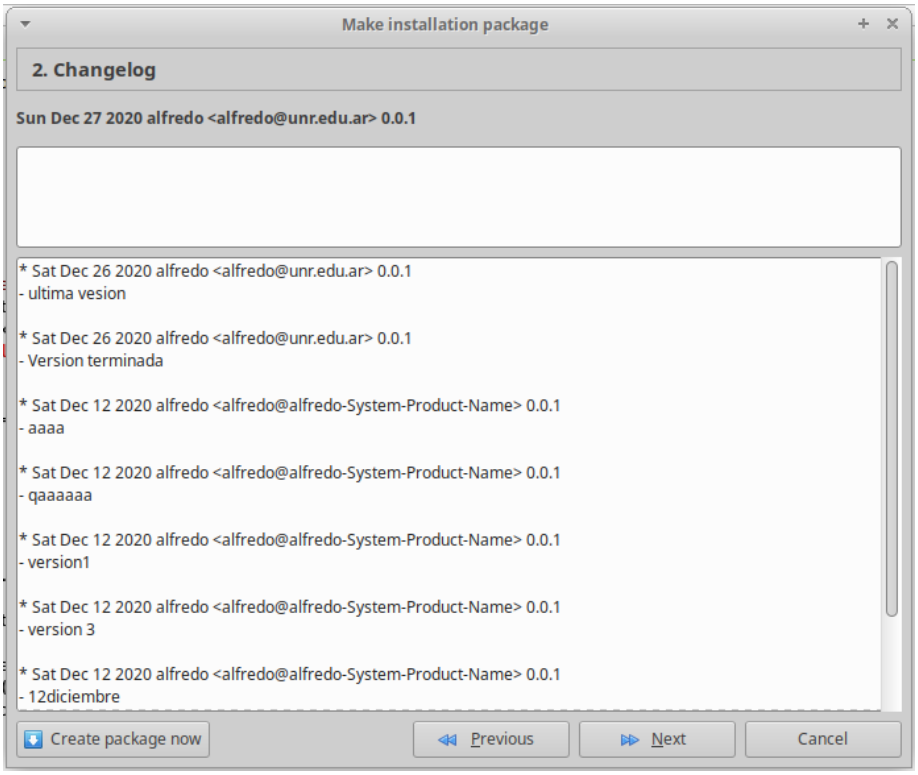


Figura 121

elija en la proxima ventana la distribución de Linux para la cual está preparando la aplicación, Figura 122. Oprima Next

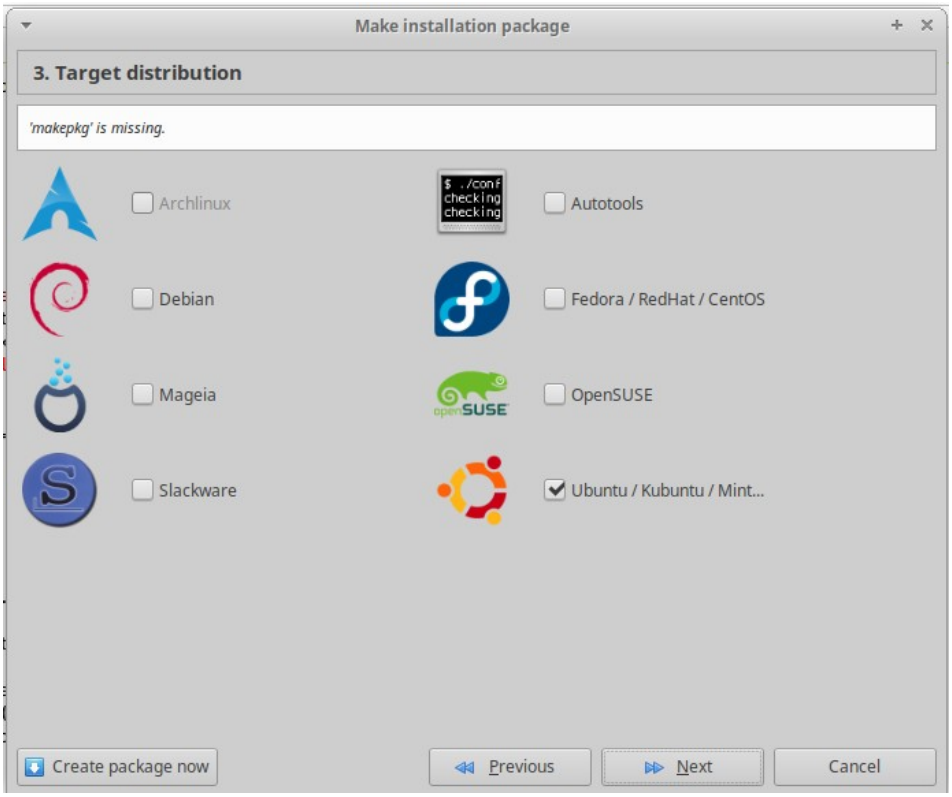


Figura 122

en la próxima ventana elija el grupo en que desea incluir su aplicación, Figura 123

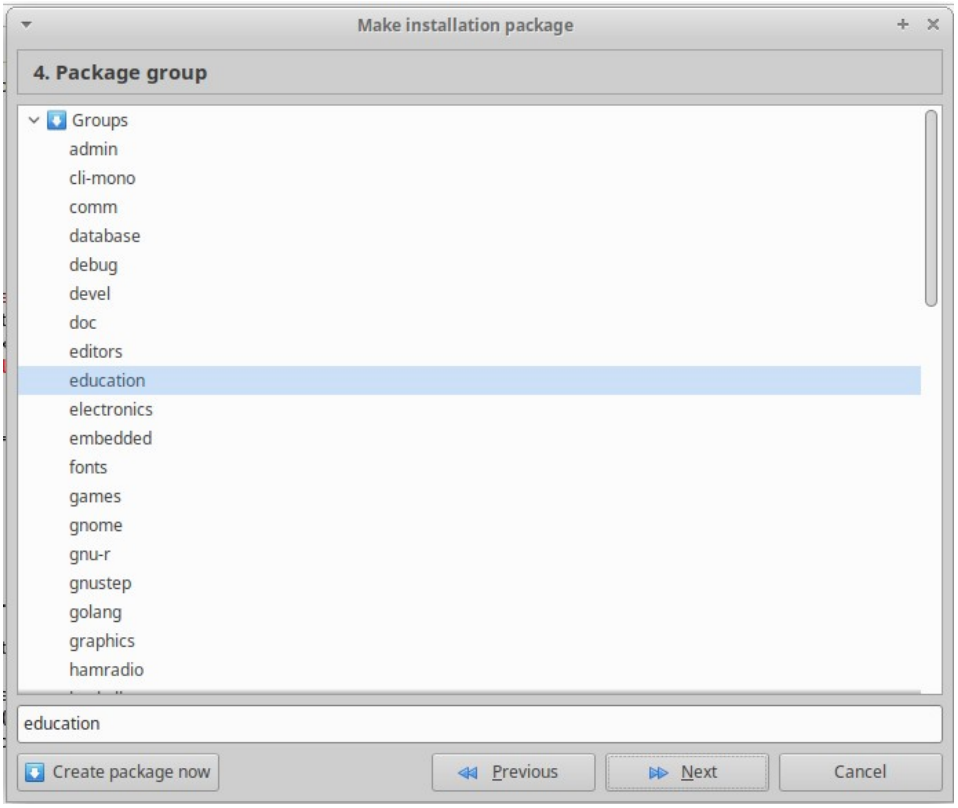


Figura 123

oprima Next y pasará a la ventana de a la elegirá en que parte del menú de aplicaciones será incluida la aplicación, Figura 124.

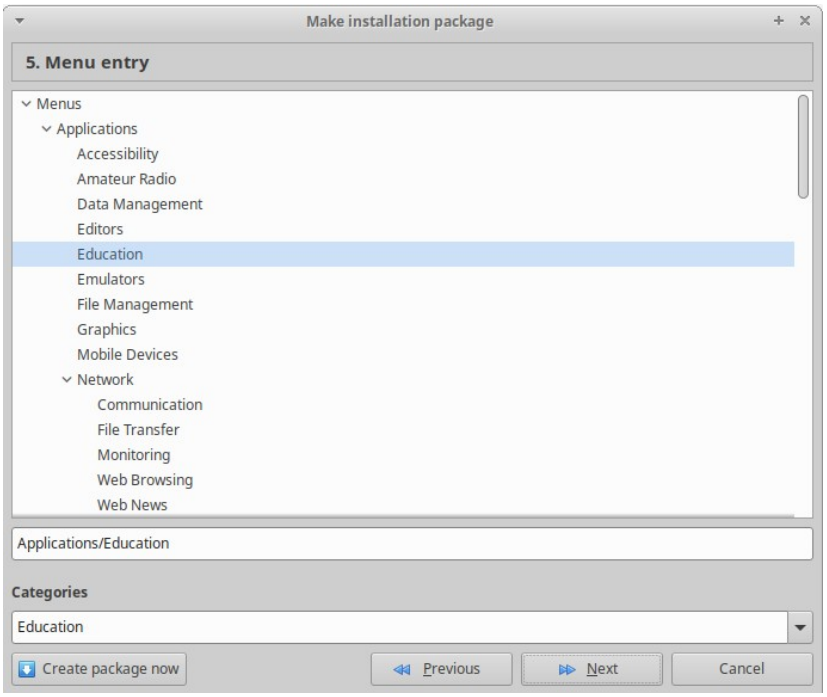


Figura 124

Oprimimos Next en el paso 6 y en el paso 7, en este caso no tenemos nada ya que no tenemos paquetes o dependencias. En el paso 8, si tuvieramos algun archivo anexo al software deberíamos subirlo, Figura 125. En la primer aplicacion que realizamos incluimos una figura. La misma debería subirse en este sitio, si existiera en esta aplicacion.

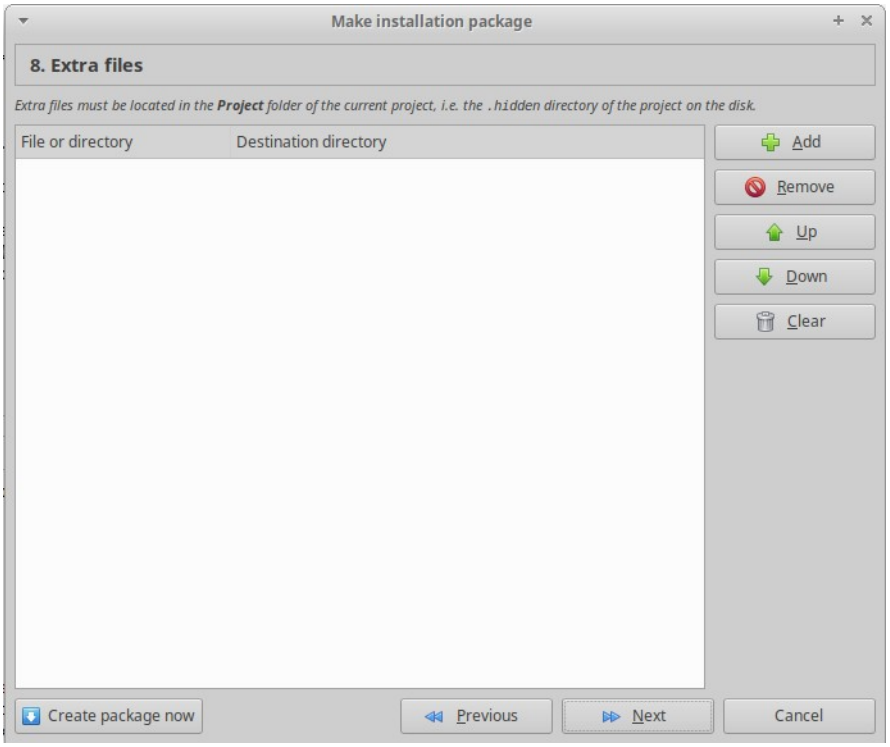


Figura 125

En el paso siguiente indicamos el directorio donde se ubicarán los paquetes de instalación. En nuestro caso lo haremos dentro de un directorio que se llame instalación Linux58, para ello hacemos click derecho en la carpeta corresponidente y elegimos crear directorio

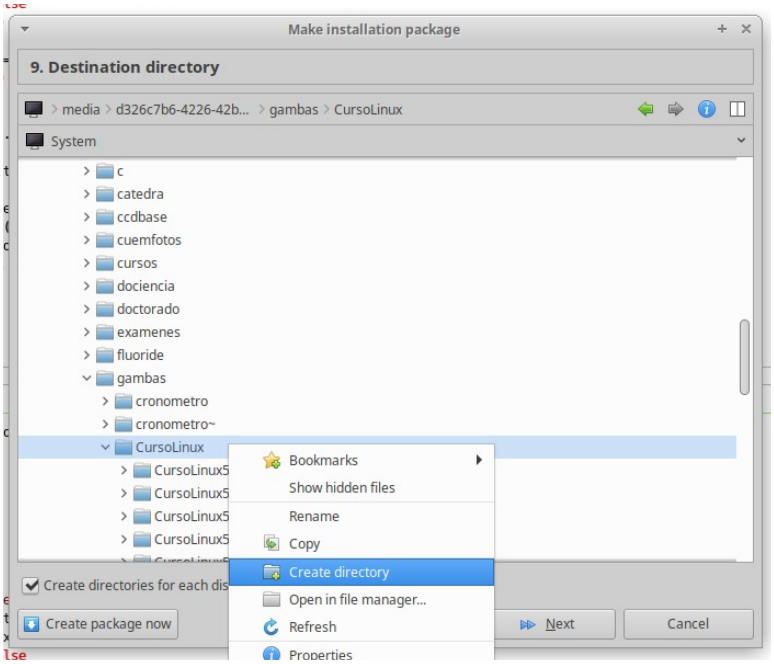


Figura 126

Llegamos al último paso, que es la creación del paquete.

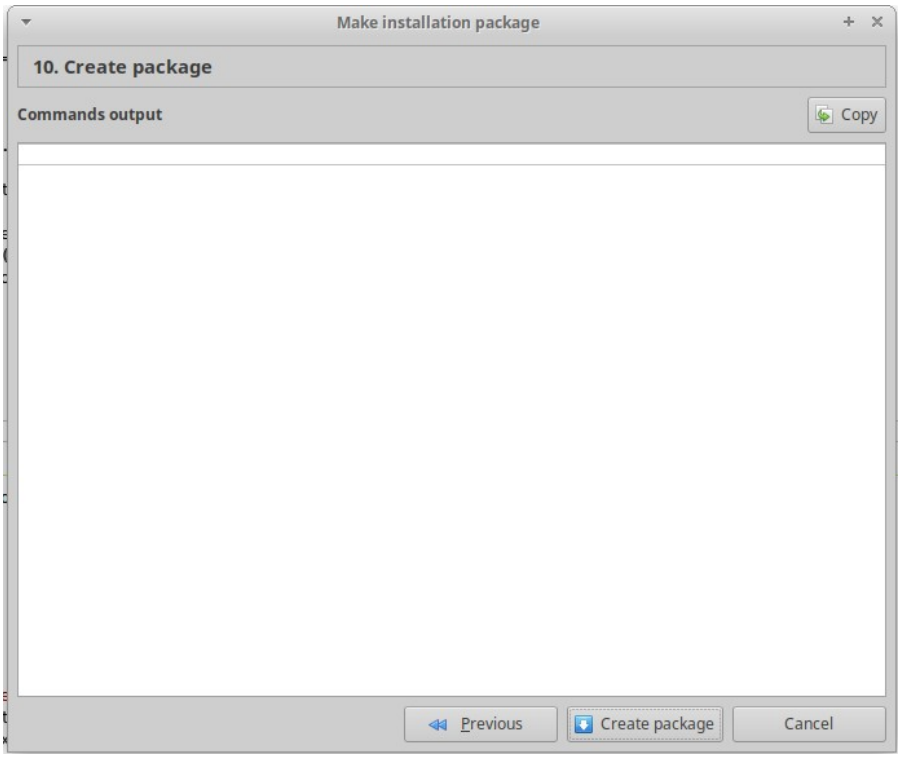


Figura 127

Oprimimos Create package y comenzará a detallarse pasos en la ventana, Figura 127. Si la creacción es exitosa, al finalizar se expresará que la creacion fue exitosa, Figura 128.

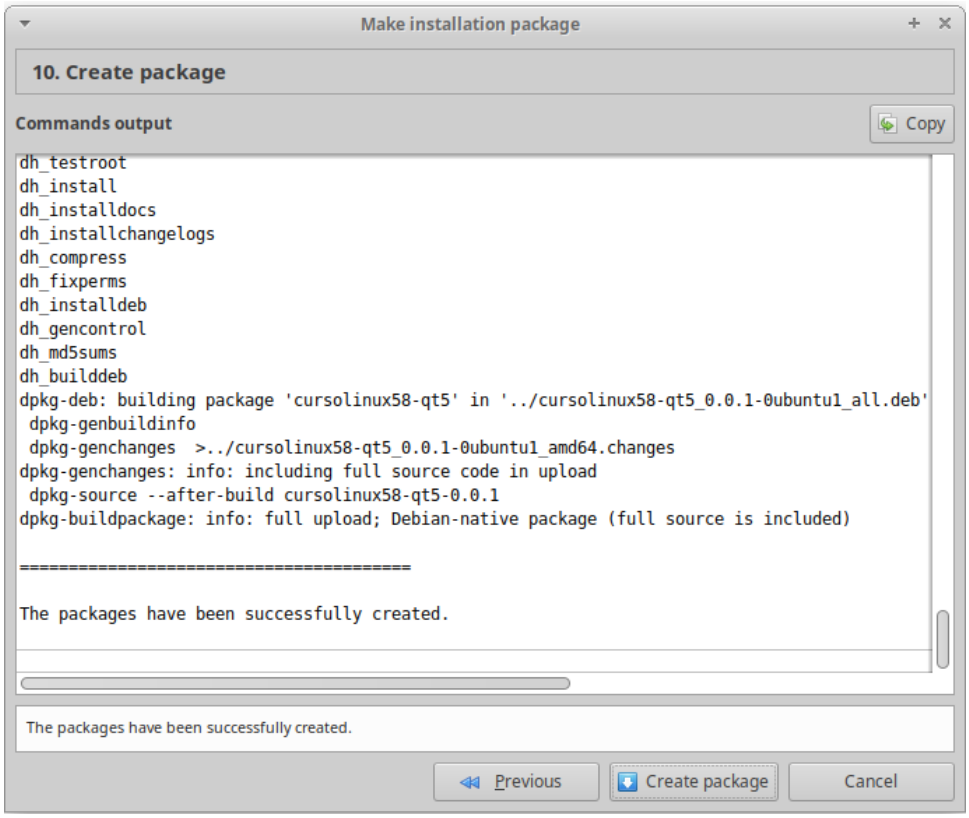


Figura 128

Una vez finalizado el proceso, hallaremos en la carpeta definida durante el proceso, los paquetes de instalación, Figura 129.

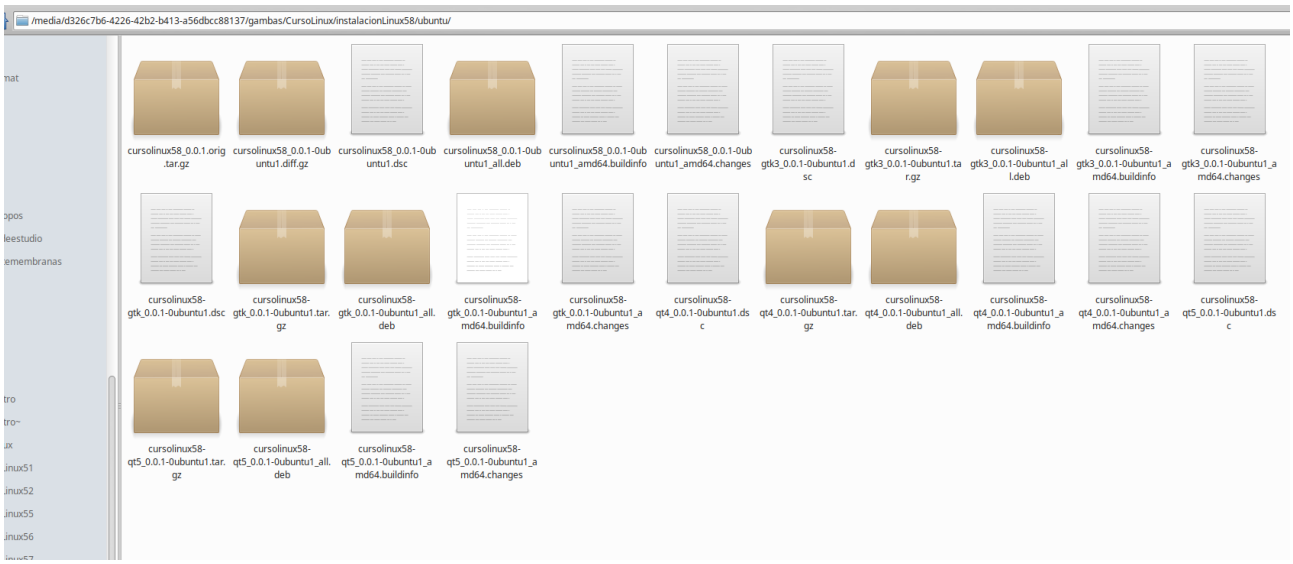


Figura 129

Para la instalación, supongamos en otra o en la misma computadora, debemos hacer doble click sobre el archivo cursolinux58_0.0.1-0ubuntu1_all.deb. De esta manera se abrirá el centro de software donde oprimimos Install y el proceso se hará automáticamente, Figura 130. La aplicación se hallará en el menú Education, como le indicamos durante el proceso

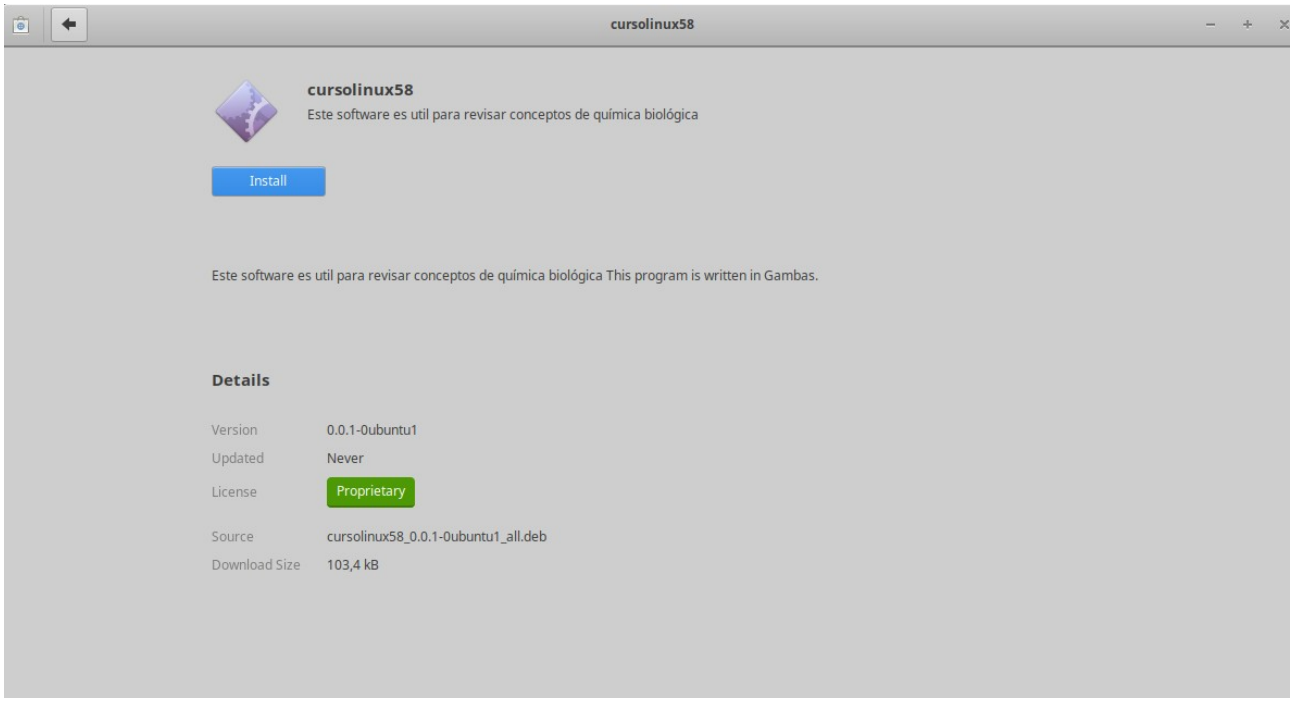


Figura 130

Puede ocurrir que la instalación desde el centro de software no se pueda realizar, entonces será más efectivo desde la terminal. Abrimos una terminal, nos posicionamos en el directorio donde se halla el paquete .deb mencionado y ejecutamos el siguiente código.

```
~$ sudo dpkg -i cursolinux58_0.0.1-0ubuntu1_all.deb
```

Si ocurriera algún error, allí podrá visualizarlo y corregirlo. El error más común es que falte alguna biblioteca que se podrá instalar desde synaptic

Luego busque en el menú de aplicaciones el software, haga click sobre él. Se deberá abrir y podrá ejecutarlo normalmente.

También podrá abrir la aplicación directamente desde una consola, escribiendo el nombre del programa

```
~$ cursolinux58
```

y ejecutando enter

En ambos casos se accederemos a la aplicación, ahora sí, sin posibilidad de edición o cambios.

Clase 9

Como ya hemos visto a lo largo del curso, el uso de la terminal o consola permite ejecutar todas las acciones que nos puede demandar el trabajo. Sin embargo algunas aplicaciones con interfaz más amigable reemplazan las aplicaciones de consola. Ejemplos sencillos de estos son:

1- Podemos cambiar de un directorio a otro utilizando el comando cd en la consola, pero sabemos que es más fácil desde un administrador de archivos hacer doble click con el botón izquierdo sobre el directorio que nos hallamos y de allí meternos dentro del mismo o volver hacia atrás en el árbol de directorios.

2- Podemos crear un directorio nuevo utilizando el comando mkdir, o bien desde la aplicación File Manager podemos crearlo simplemente haciendo click derecho con el mouse y elegir Create New Folder.

En la Figura 131 ilustramos ambos ejemplos

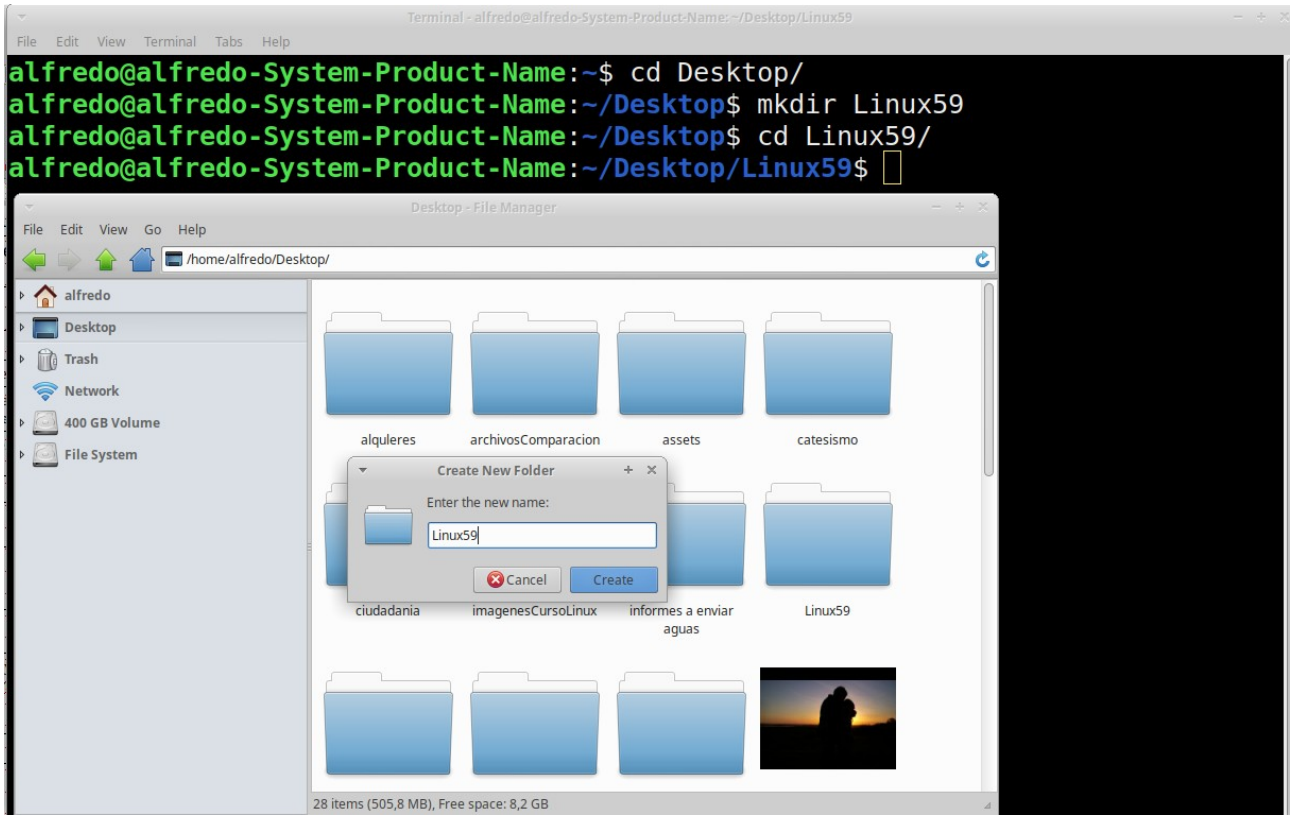


Figura 131

Una posibilidad que también nos da la consola es ejecutar una serie de acciones, que se conoce como script. Conociendo programación en Gambas, será más sencillo crear una aplicación de Gambas que ejecute una aplicación de consola.

En esta clase iniciaremos una nueva aplicación gráfica y asignaremos al proyecto el nombre "CursoLinux59" y el título de la aplicación será "AutomatizarComando". Se trata de una aplicación que nos permitirá ejecutar automáticamente comandos de consola sobre los archivos dentro de un directorio que seleccionaremos de una lista.

El objetivo de este programa será ejecutar un comando de consola sobre cada uno de los archivos en un directorio seleccionado, tanto dentro del directorio como subdirectorios.

Específicamente buscaremos reducir todas las imágenes en un árbol de directorios al 50% de su tamaño, agregando convertido al final del nombre. Por supuesto que este objetivo es solo a fines de explicación del mecanismo, pero podrá ser implementado para infinidad de tareas que como siempre dependerán del tiempo disponible, la necesidad y la imaginación. Para esto utilizaremos comandos disponibles en la consola de Linux. Gambas incorpora una forma sencilla de ejecutar procesos de consola a través de la sentencia SHELL, pudiendo ingresar y leer la salida de estos comandos.

Lo primero que realizaremos es ejecutar un comando de consola para obtener las lista de archivos dentro del directorio seleccionado en DirChooser y mostraremos esta lista en el ListView que hemos agregado. Luego, leyendo la lista llenada previamente en el ListView ejecutaremos un comando sobre cada elemento de esa lista.

Veamos primero la descripción los elementos mencionados

DirChooser

Como todos los objetos de la interfaz gráfica, los insertamos en nuestro formulario arrastrándolo desde la lista de herramientas. En este caso lo hallaremos dentro de la solapa Chooser, Figura 132

Los objetos DirChooser muestra los directorios presentes en el sistema y permite seleccionar un directorio.

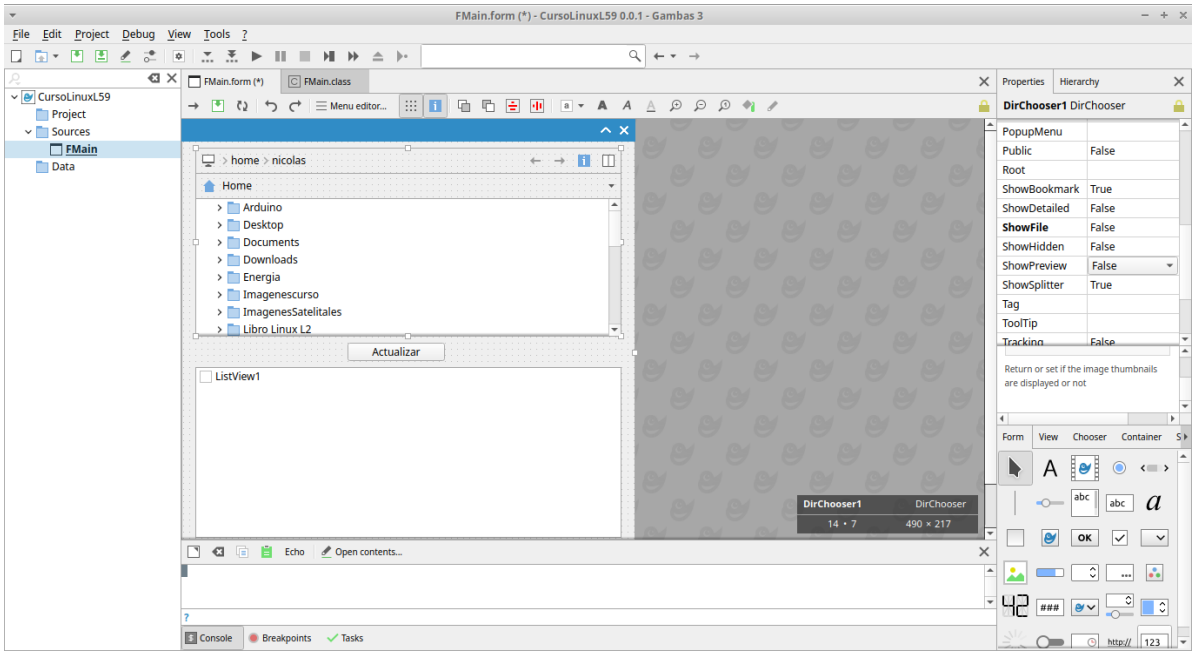


Figura 132: Ventana de la aplicación con DirChooser

Podemos dividir el objeto en dos partes si queremos que muestre los archivos dentro de un directorio poniendo TRUE en la propiedad ShowFile. A su vez podremos indicar que muestre vista previa de imágenes poniendo TRUE en la propiedad ShowPreview. En la Figura 132 se puede observar la ventana de la aplicación con el objeto DirChooser con ShowFile en FALSE, mostrando la lista de directorios del sistema. Al cambiar a TRUE tanto ShowFile como ShowPreview DirChooser cambia a la forma de la Figura 133 donde se separa en 2 el objeto, mostrando una columna con los archivos y mostrando una miniatura del archivo si es una imagen. El tamaño de la

zona dedicada al árbol de directorios y a los archivos puede cambiarse ubicando el mouse sobre la línea que divide las dos partes.

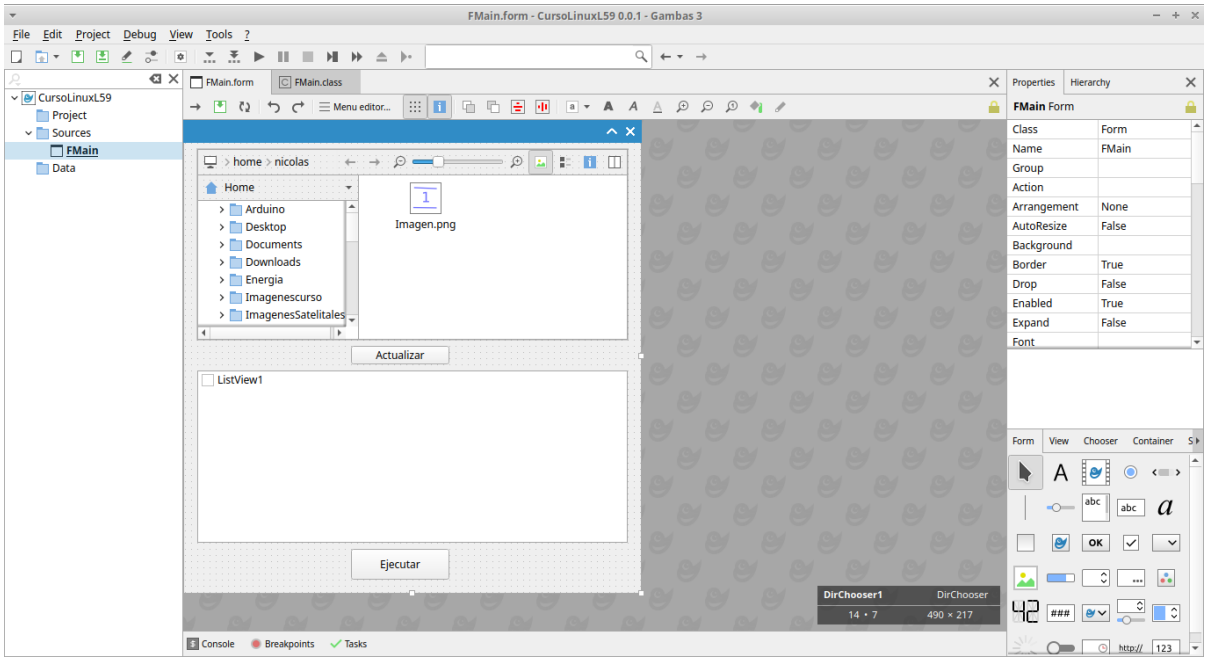


Figura 133: DirChooser con ShowFiles y ShowPreview en TRUE

El directorio que se encuentra seleccionado puede obtenerse leyendo la propiedad SelectedPath. En nuestra aplicación hemos definido una variable para el formulario con el nombre ruta

Private ruta as String

Dentro del evento click del botón Actualizar hemos colocado la sentencia

ruta = DirChooser1.SelectedPath

entonces la variable ruta tomará el valor correspondiente al arbol de directorios.

ListView

El objeto ListView brinda similares utilidades a los ComboBox vistos en clases anteriores. Ambos objetos brinda una lista de elementos de texto seleccionables indexados. En el caso de ListView, este objeto tiene un cursor interno que puede moverse mediante los métodos MoveFirst (mueve al primer elemento)

ListView1.MoveFirst()

Otros métodos son MoveAbove (mueve al elemento superior al actual) y MoveBelow (mueve al elemento inferior al actual) entre otros. Los métodos devuelven el valor TRUE si la lista está vacía en el caso de MoveFirst, si no se realizó un movimiento debido a que no había elemento arriba en el caso de MoveAbove, abajo en el caso de MoveBelow. Estos métodos resultan de utilidad para conocer cuándo se está en el último elemento.

Para limpiar la lista se utiliza el método Clear

ListView1.Clear()

Para agregar elementos a la lista se utiliza el método Add.

ListView1.Add(i, proceso.ReadLine())

En la Figura 134 se observa a la izquierda un ListView vacío y a la derecha el mismo ListView luego de agregarle líneas. La forma de agregar elementos es igual que para el agregado de elementos en ComboBox. En esta clase el agregado de elementos se hace a través de la lectura de archivos de un directorio elegido con FileChooser. En clases anteriores cuando llenamos el ComboBox, lo hicimos directamente indicando los elementos entre las instrucciones del programa.

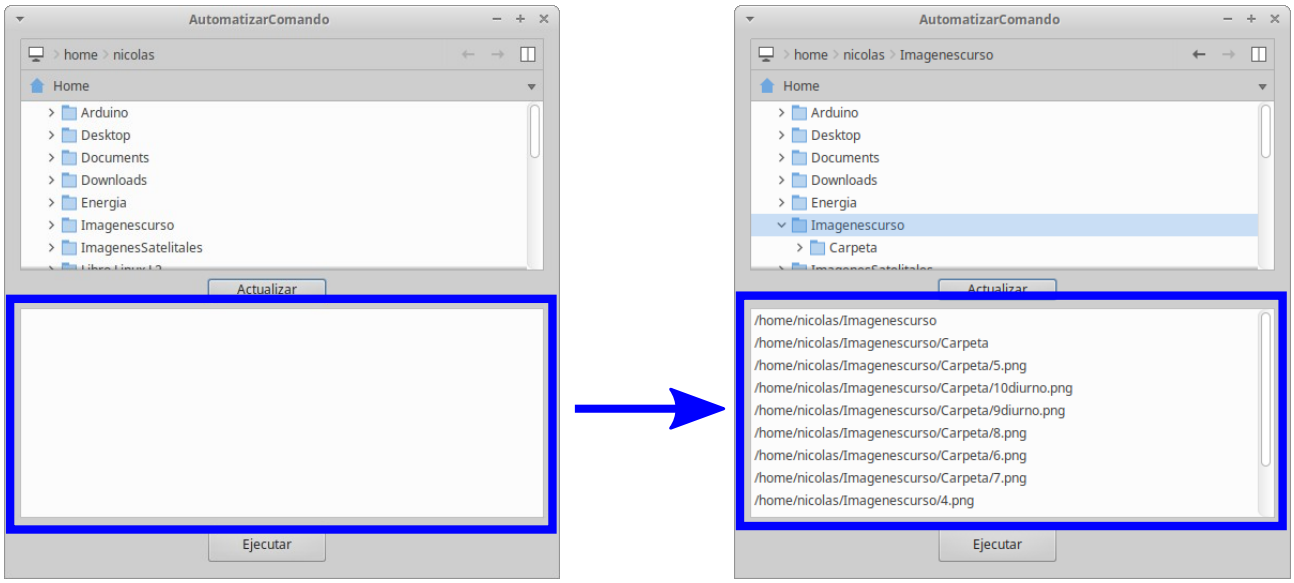


Figura 134: ListView vacío y con elementos

Obtención de lista de archivos

La obtención de la lista de archivos la asociaremos al evento Click del botón Actualizar. Para ello utilizaremos la función de Gambas SHELL y el comando de consola find. El comando find permite buscar archivos en un árbol de directorios, cada uno de los archivos encontrados los incorporaremos como líneas en el ListView. Veremos una forma de realizarlo en el código mostrado más abajo

código	interpretación

Public Sub Actualizar_Click()
  Dim busqueda As String
 	
  Dim proceso As Process
 	
  Dim linea As String
 	
  Dim i As Integer
 	
  ruta = DirChooser1.SelectedPath
  ListView1.Clear()
  busqueda = "find " & ruta
	Esta rutina se ejecuta al realizar click sobre el objeto Actualizar
	String donde almacenaremos al comando a ejecutar
	
	Esta variable almacena el proceso de sistema que luego ejecutaremos.
	Cada línea devuelta por el proceso la guardaremos temporalmente en este string
	Usaremos el entero i para el llenado del ListView
	
	Obtengo la ruta del DirChooser con su propiedad SelectedPath
	Limpio la lista
	Construyo el comando a ejecutar

proceso = Shell busqueda For Read	Ejecuto el comando con la función SHELL y le indico que quiero sacar la salida del comando con “For Read”
i = 0	Procedo a llenar el listview desde el índice inicial 0
While (Not proceso.Eof)	Mientras no termine la salida del proceso (Eof significa End of File)
ListView1.Add(i, proceso.ReadLine())	Agrego un elemento a ListView1 con la siguiente línea de salida del proceso
i += 1	Paso al elemento siguiente de ListView1
Wend	Termino el bucle de ejecución
End	Termina la función

Al incorporar el código en el evento Click del botón y ejecutar la aplicación si hacemos click sobre el botón Actualizar veremos una lista como la de la figura 135. Podemos observar que la salida del comando de consola Find devuelve todas las rutas, no solo las de archivos. Esto lo solucionaremos al ejecutar el comando sobre cada archivo pero podríamos acotar los resultados dentro de Gambas procesando cada ReadLine o dentro del comando guardado en búsqueda.

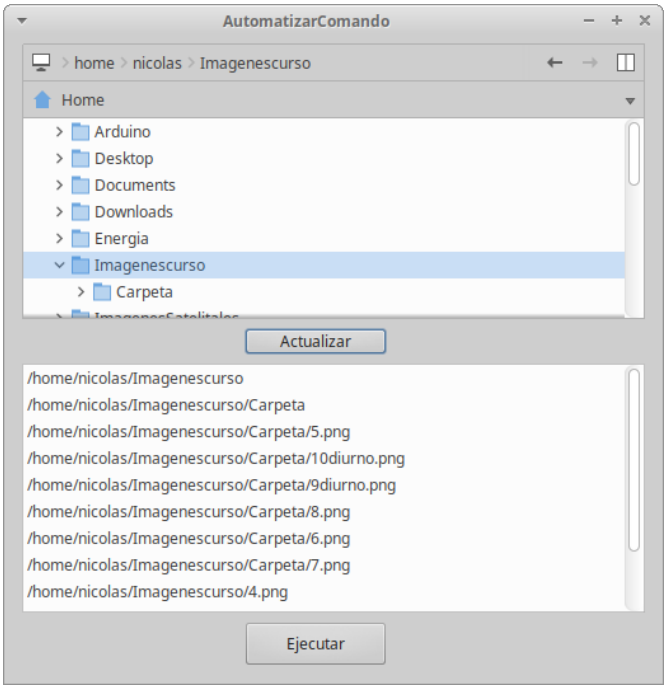


Figura 135: Comando de obtención de lista de archivos ejecutado

Ejecución de comando

Para ejecutar el comando de consola que redimensionará las imágenes utilizaremos el mismo entorno SHELL que utilizamos anteriormente, el comando de consola en este caso será convert. El comando convert permite realizar diversas operaciones sobre imágenes, entre ellas redimensionar. Construiremos el comando correspondiente a cada línea que corresponda a un archivo de imagen

entre los listados en el ListView. Asociaremos la función descripta debajo al evento Click del botón Ejecutar

código	interpretación
Public Sub Ejecutar_Click()	Esta rutina se ejecuta al realizar click sobre el botón Ejecutar
Dim comando As String ' comando a realizar a cada archivo	String donde almacenaremos al comando a ejecutar
Dim noHayElementoSiguiente As Boolean	Esta variable valdrá TRUE si el ListView tiene elementos luego del que estamos procesando
Dim rutacompleta As String Dim extension As String ' extension del archivo Dim archivo As String ' ruta del archivo sin la extension	Los strings rutacompleta, extension y archivo guardarán la ruta completa, la extensión y la ruta junto con el nombre sin la extensión respectivamente del archivo siendo procesado
ListView1.MoveFirst()	Como voy a usar el ListView como lista de archivos ubico el cursor de ListView en su primer elemento
Repeat	Voy a repetir las sentencias dentro de este bloque mientras se cumpla la condición en Until
rutacompleta = ListView1.Item.Text	Obtengo el texto del elemento seleccionado en el ListView, este corresponde a una de las rutas encontradas anteriormente
archivo = Left(rutacompleta, -4)	Recorto los últimos 4 caracteres que serian la extensión y el punto
extension = Right(rutacompleta, 3)	Conservo los últimos 3 caracteres que indican la extensión del archivo
If extension == "png"	Si la extensión es la adecuada, en este caso .png
' convert input -resize 50% Output comando = "convert " & rutacompleta & " -resize 50% " & archivo & "convertido.png"	Construyo el comando que sigue la forma “convert archivoentrada -resize 50% archivo salida”
Shell comando ' ejecuto el comando	Ejecuto el comando
Endif noHayElementoSiguiente = ListView1.MoveBelow()	Muevo el cursor del ListView al elemento siguiente y guardo su resultado para saber si este era el último elemento.
Until (noHayElementoSiguiente)	El lazo de control Repeat….Until repite la sentencias hasta que la condición en Until sea verdadera. En este caso se repite hasta que no haya elementos en el ListView
End	

Si apretamos primero el botón actualizar con el directorio deseado seleccionado y luego apretamos el botón ejecutar aparecerán los nuevos archivos redimensionados. Una forma de comprobarlo es apretar luego nuevamente el botón actualizar. Veremos que el ListView mostrará los nuevos archivos con el nombre modificado como se observa en la figura 136 donde se encuentra marcado en azul los nuevos archivos convertidos.

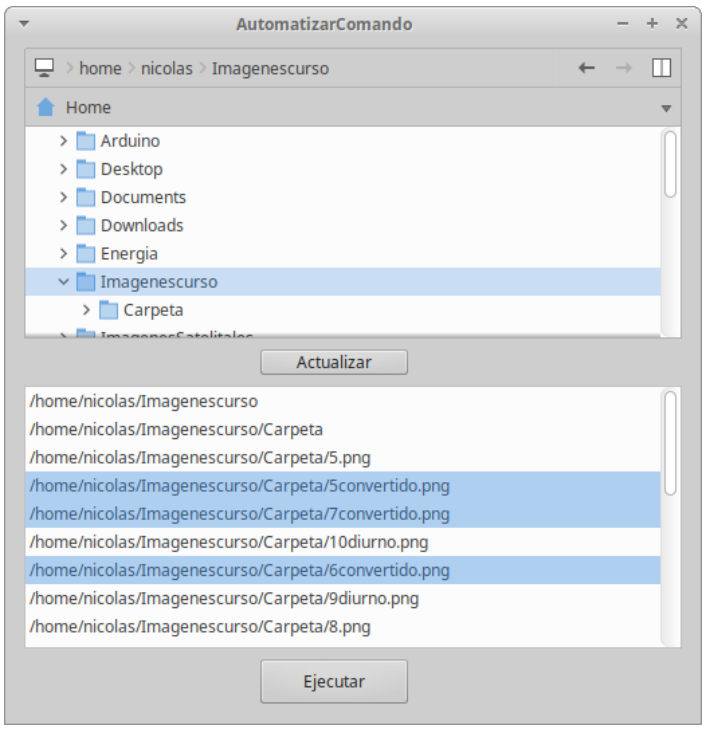


Figura 136: ListView actualizado luego de ejecutar el comando sobre los archivos

Algunas líneas finales

Hemos construidos algunas aplicaciones gráficas y con manejo de bases de datos. Sin duda ellas son solo un punto de partida para comenzar proyectos más ambiciosos y aplicados a sus necesidades. Verá en el desarrollo posterior la enormidad de recursos que Gambas propone y podrá imaginar también lo que será en el futuro.

Al realizar estas aplicaciones hemos utilizado algunos términos que no definimos claramente en principio, orientándonos principalmente a su uso, aceptando más de un término y acción.

Sin duda lo que hace sencilla la programación en Gambas son sus objetos que podemos arrastrar al interior de nuestros formularios y asignarle a eventos de ellos códigos que realizan las tareas requeridas. Cada uno de estos objetos tienen **propiedades**, **métodos** y **eventos**.

Propiedades

Las propiedades básicas para cada aplicación la hemos elegido durante tiempo de diseño a través de la solapa Properties. La Figura 137 muestra seleccionado el objeto DirChooser1 del formulario FMain y se ha fijado la propiedad Height en el valor 231. Como ya hemos visto también las propiedades pueden fijarse en tiempo de ejecución. Si dentro de nuestro código por ejemplo en el evento Open del formulario Form incluyéramos la línea

DirChooser1.Height= 230

Al ejecutarse el evento correspondiente, cambiaría la altura del objeto DirChooser

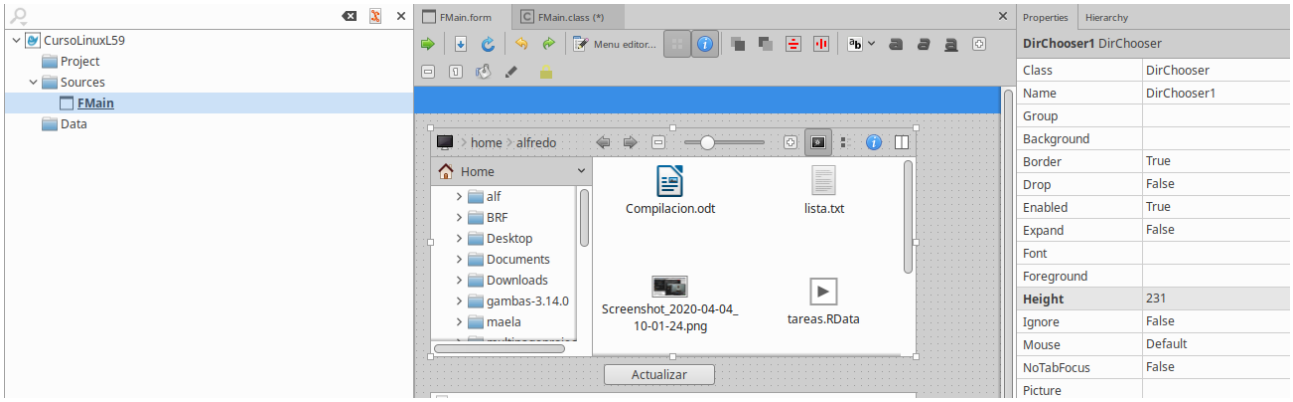


Figura 137

El listado de propiedades lo podemos ver en la solapa Properties, pero también podemos verlo y consultarlo en la parte inferior de la misma ventana. Si es muy pequeña pude arrastrar la linea separadora, manteniedo apreptado el botón del mouse ventana. En la Figura 138 vemos seleccionado el botón Actualizar del Form Fmain y a la derecha la solapa Properties, con la ayuda sobre este tipo de objeto (Button, en este caso). Abajo pùede ver también la lista de Properties. Haciendo click sobre cualquiera de ella tendrá una valiosa ayuda que le permitirá sacar mucho más rédito de cada objetco

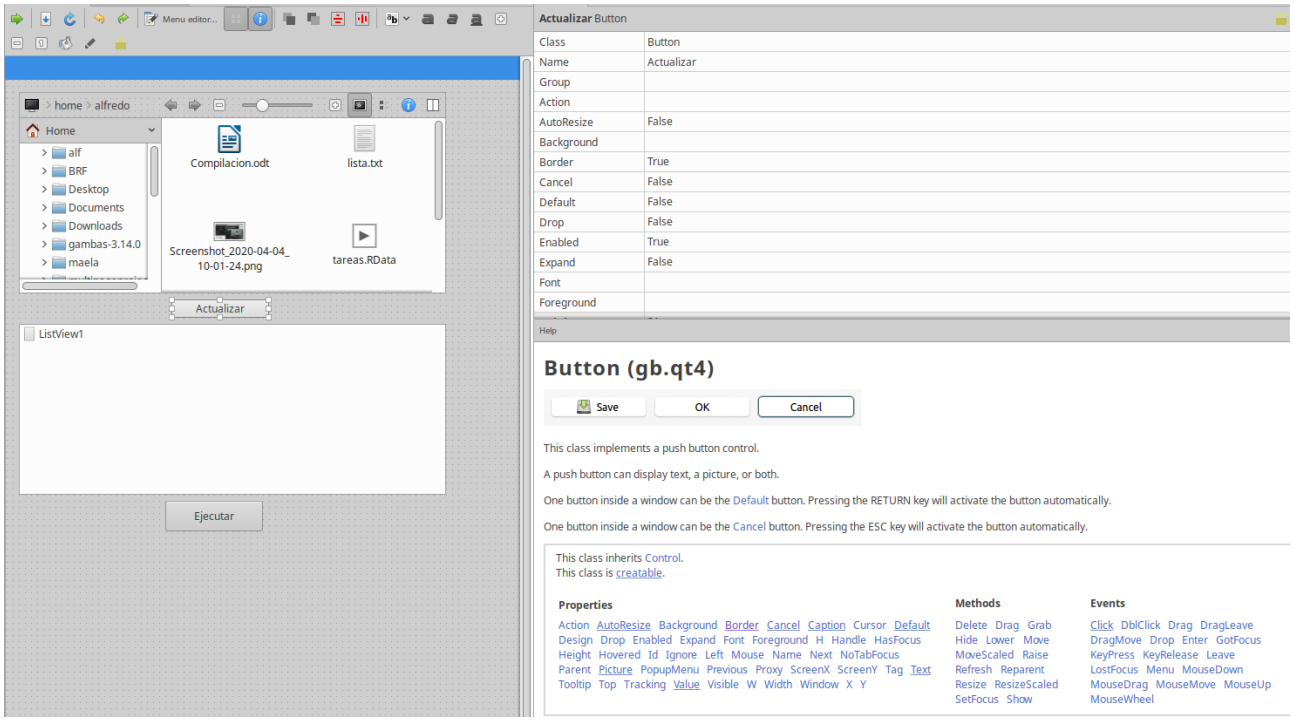


Figura 138

Eventos

Los eventos son procedimientos que se ejecutan sobre un objeto en particular, por ejemplo: click con el botón izquierdo del Mouse, doble-click, etc. La lista de eventos es muy grande pero por comodidad y simplicidad hemos utilizado habitualmente click y doble-click. Para conocer el lista de eventos tenemos dos mecanismos. Uno es marcar el objeto en el formulario y hacer click derecho, eligiendo la opción Event. En la Figura 139 vemos la selección del evento Drag del objeto Actualizar

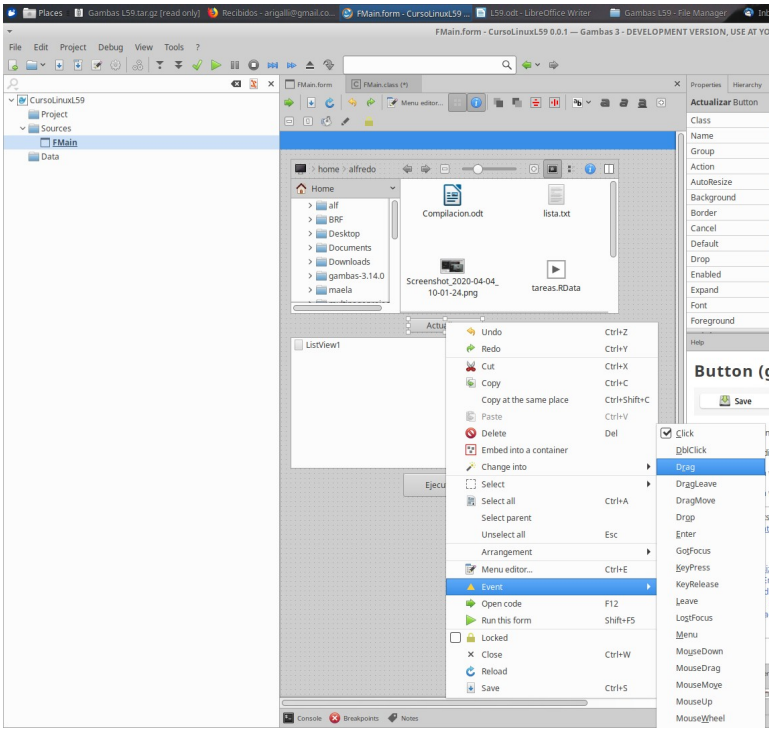


Figura 139

El listado de eventos de cada objeto también lo puede consultar en la ventana inferior de la solapa Properties, Figura 140. Haciendo click sobre cualquiera de ellas accederá a una ayuda que le permitirá conocer cada evento y hacer la elección más conveniente para cada objeto y aplicación. Un dato interesante es que a un mismo objeto podemos asociarle más de un Evento y que se ejecute cada uno con instrucciones y comandos específicos. Tal fue el caso en nuestras aplicaciones del evento Open y Show para un dado formulario

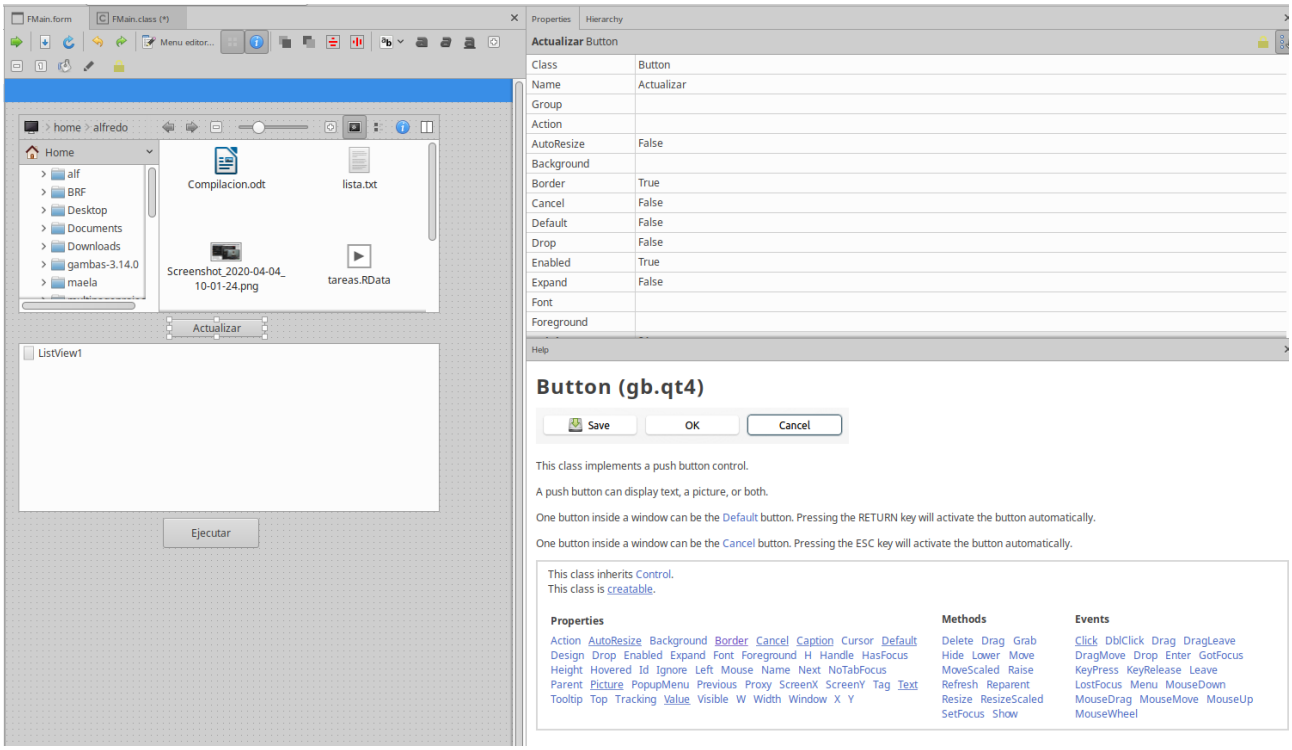


Figura 140

Métodos

Los métodos son acciones que podemos hacer con un dado objeto. En el caso de los ComboBox y que también es válido para los objetos ListView, son los eventos Clear y Add, que hemos utilizado. El primero vacía la lista de ítems y el segundo agrega ítems a ambos listados. Otro evento fue SetFocus, que lo utilizamos para orientar el puntero del mouse a un determinado objeto. Por ejempli si deseamos que cuando se abre un formulario, el puntero del mouse se posicione en un comboBox y además éste se vacie, tendriamos un código

Public Sub Form_Open()

ComboBox1.SetFocus

ComboBox1.Clear

El listado de los Métodos, también lo hallaremos para cada objeto debajo de la pestaña Properties. En la Figura 141 se muestran los métodos para un objeto de tipo Button. Haciendo click sobre cualquier de ellos hallará instrucciones.

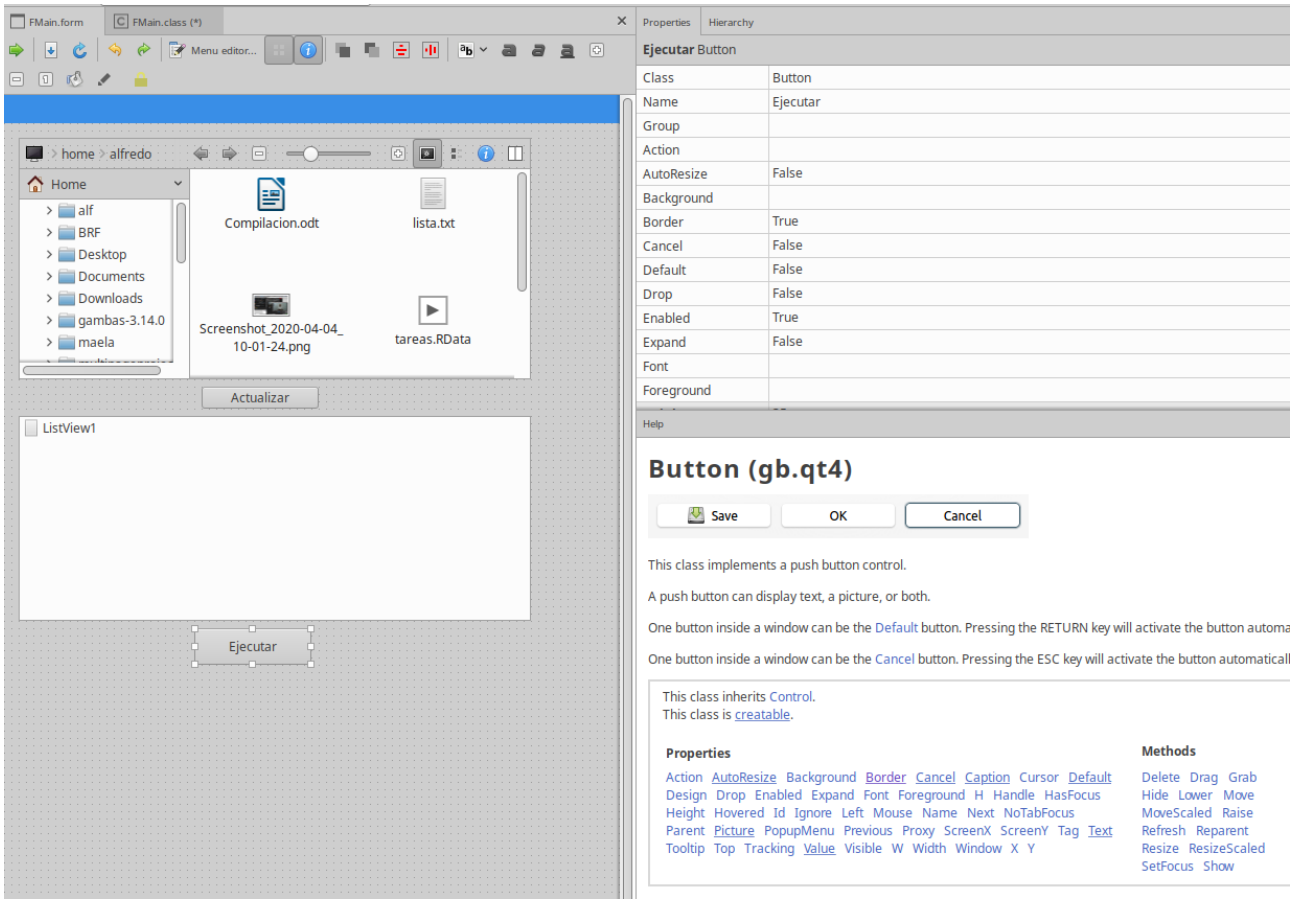


Figura 141

Creemos que con los elementos aportados en este módulo se puede hacer una aplicación para ejecución en el entorno Linux, prácticamente sin limitaciones. El trabajo cotidiano, sostenido e imaginativo permitirá ampliar las posibilidades y optimizar sus aplicaciones.

@cuem_unr
/estudiosmedioambientales
arigalli@unr.edu.ar

ISBN 978-987-86-8420-8