Programando con B4X

Tema 19 – Proyecto Final

Version 1.0, mayo 2021

Autor original: Prokopis Leon

Traducido al español por José Miguel López

Tema 19 – Proyecto Final

10h

Lo que los estudiantes aprenderán
• Usar los conocimientos adquiridos para crear un proyecto final

Estimado Profesor

La solución propuesta abarca todo el conocimiento adquirido durante el curso. Puedes ampliar las preguntas para adaptarlas al nivel de tu clase y al tiempo disponible.

Para aprender cómo se crea una biblioteca escolar puedes leer el libro "Setting Up and Running a School Library by Nicola Baird" aquí en https://files.Eric.Ed. Gov/Fulltext/Ed536911.Pdf

Los nombres de los libros y los autores han sido tomados del Proyecto Gutenberg https://www.gutenberg.org/ebooks/offline_catalogs.html

Los nombres de las editoriales, las fechas de publicación y el nombre de los estudiantes son aleatorios.

Prokopis Leon, pliroforikos[at]gmail.com

Crear una Aplicación de Biblioteca Escolar

Una biblioteca escolar posee una colección de libros que presta a los estudiantes de tres clases. Los libros se colocan en estantes numerados del 1 al 60. Para cada libro, se guarda la siguiente información:

- Código
- Título
- Escritor
- Año de publicación
- Editorial
- Estante

De los estudiantes, se guarda la siguiente información:

- Número de registro
- Nombre
- Apellidos
- Clase
- Teléfono

Construye la aplicación descrita más abajo:

Ficheros

Se proporcionan dos ficheros llamados books.txt y students.txt (dentro de la carpeta de materiales).

Algunos de los elementos del fichero son:

books.txt

```
30;The Life of Abraham Lincoln;Henry Ketcham;1866;New Public publ.;54
31;Christopher Columbus; Mildred Stapley;1954;Cider publ.;43
32;The Adventures of Ferdinand Count Fathom; Tobias Smollett;1982;Orange punl.;32
33;Tales of the Jazz Age;F. Scott Fitzgerald;1944; Gutenberg publ.;5
34;The Old Stone House;Anne March;1904;Orange punl.;50
```

Cada libro posee los siguientes campos para que se distingan unos de otros y que se separan por el carácter ";":

- Id
- Título
- Escritor
- Año
- Editorial
- Estante

students.txt

```
1001;Jude;Segers;A;7900209
1002;Desire;Cid;A;7047635
1003;Madelyn;Pittard;A;9011036
1004;Lorita;Tomczak;A;6677490
1005;Lynwood;Posey;A;9014379
1006;Nella;Felps;A;8423818
```

Cada estudiante posee los siguientes campos para distinguirlos unos de otros y que se separan con el carácter ';':

- Id
- Nombre
- Apellidos
- Clase
- Teléfono

Implementa una función que inserte los anteriores ficheros de texto en dos mapas con nombres mapBooks y mapStudents. La clave será el primer campo que aparece en cada línea de los ficheros que representan el "id" del libro y del estudiante, respectivamente. Los valores de cada ítem se describen más abajo.

Consejo para el profesor

Antes de leer los ficheros tienes que leer todo su contenido como una cadena de texto (string) y luego ir uno a uno comprobando los caracteres para ir colocando los ítems en el array de ítems que usarás para escribir en el mapa. Para comprobar una cadena usa el método **CharAt**(índice).

Ejemplo de Mapa de Estudiantes

Clave	Tipo				
	Id	**Nombre**	**Apellidos**	**Clase**	**Teléfono**
1001	1001	Jude	Segers	A	7900209
1002	1002	Desire	Cid	A	7047635
1003	1003	Madelyn	Pittard	A	9011036

Datos

Tipos

1. Crear el tipo Libro

   El tipo Libro tendrá estos elementos:

   - Id
   - Título
   - Escritor
   - Año
   - Editorial
   - Estante

   Funciones:

   - Insertar Libro
   - Borrar Libro

2. Crea el tipo Estudiante

   Propiedades:

   - Id
   - Nombre
   - Apellidos
   - Clase
   - Teléfono

   Funciones:

   - Insertar Estudiante
   - Borrar Estudiante

Cada estudiante que haya pedido prestado libros posee su propio fichero KVS con el nombre "ID número.dat". Por ejemplo, el estudiante con ID 21 tiene un fichero llamado "21.dat". En el fichero se guardar los libros prestados al estudiante usando un mapa como estructura. Por ejemplo, si has pedido prestado dos libros, deberías tener un mapa con la siguientes estructura:

("11", "03/27/2021")

("14", "04/01/2021")

Donde el primer número es el ID del libro prestado y también la clave del mapa; la fecha se guarda como valor asociado a la clave.

Diseño de la pantalla y Funciones

El programa debe ofrecer un menú central de botones para invocar a la pantalla de gestión adecuada. En concreto:

1. **Libro**
   Incluirá una lista CLV de libros de la biblioteca que son cargados desde el fichero books.txt, así como botones para insertar y borrar libros.
   Cuando se pulse el botón insertar, aparecerá un diálogo para indicar los datos del nuevo libro que será guardado en el fichero books.txt.
   Cuando se pulse el botón Borrar, se borrará el libro elegido de la lista.

2. **Estudiante**
   Incluirá una lista CLV de estudiantes de la escuela que se cargarán del fichero students.txt y botones para borrar e insertar estudiantes.
   Cuando se pulse el botón insertar, se creará un diálogo para pedir los datos del nuevo estudiante y guardarlo en el fichero students.txt.
   Al pulsar el botón borrar, se borrará el estudiante de la lista, junto con los libros que se le han prestado suponiendo que los ha devuelto.

3. **Préstamo**
   La pantalla de préstamo tendrá un ComboBox que contendrá los nombres de los estudiantes una lista CLV de los libros de la biblioteca.
   Habrá un botón "Prestar" que cuando se pulse hará lo siguiente:
   Si hay un fichero del estudiante
   - Cargará el mapa completo del fichero
   - Añadirá la nueva clave con el ID del libro prestado y la fecha del préstamo
   - Guardará de nuevo el mapa

   Si no hay fichero del estudiante
   - Creará el fichero
   - Creará un mapa con la información del préstamo (ID y fecha)
   - Guardará el fichero

Supón que hay tantas copias de cada libro como necesites.

4. **Devolución**
   La pantalla de devolución incluirá un ComboBox con el nombre de los estudiantes y una lista CLV de los libros prestados. Además, habrá un botón "Devolver".
   Si se pulsa el botón "Devolver" se borrará el libro de la lista del estudiante y del fichero del estudiante.
   Si no se le ha prestado ningún otro libro, el programa borrará el fichero también.

Hoja de pistas

1. Cómo declarar un tipo

```
Sub Class_Globals
  Type Estudiante(Nombre  As String, Apellidos  As String, _
                  Dirección As String,  Teléfono  As String)

  Private Estudiante1  As Student
End Sub
```

2. Cómo guardar el contenido de un fichero en una cadena de texto:

```
fichEst = File.ReadString(File.DirAssets, "students.txt")
```

3. Comprobar carácter a carácter una cadena con el contenido de un fichero para crear un array de estudiantes.

```
  Private i  As Int  = 0
  For j = 0  To  ficheroEst. Length-1
    If fichEst.CharAt(j) <> ";" And fichEst.CharAt(j) <> CRLF  Then
      estudiante(i) = estudiante(i) & fichEst.CharAt(j)
    Else if fichEst.CharAt(j) = ";" Then
      i = i + 1
    else if fichEst.CharAt(j) = CRLF  Then
      i = 0
      listaDevuelta.Add(estudiante)
      Private estudiante(5) As String
    End If
  Next
```

4. Convertir una lista de arrays a un mapa:

```
  For i = 0 To lista.Size-1
    Private estud(5) As String
    estud = lista.Get(i)     ' Crear un array de un ítem de la lista
    Private est As Student  ' Crear una variable del tipo estudiante
    est.Initialize
    est.ID = estud(0)   ' Poner cada ítem del array en el tipo
    est.Nombre = estud(1)
    est.Apellidos = estud(2)
    est.Clase = estud(3)
    est.Teléfono = estud(4)
    est.Prestado = 0
    mapaEstudiantes.Put(estud(0), est)   ' Insertar tipo en mapa
  Next
```

5. Colocar espacios en una cadena para incrementar su tamaño hasta un número indicado:

```
Do While cadena1.Length <= 5
    cadena1 = cadena1 & " "
Loop
```

6. Marcar – Desmarcar un ítem de una lista CLV
Cada ítem de una lista se crea dentro de una caja llamada "panel". Puedes fijar su color accediendo a la caja con el método GetPanel(índice) donde el índice es el valor actual de la lista en la que se ha pulsado. Después:

- Si el ítem de la lista se pulsa, el método comprueba el valor del ítem elegido y, si es -1, entonces establece en azul el color de fondo de la línea elegida y guarda en ítemElegido el índice devuelto por el evento _ItemClick.
- Si ítemElegido ya tiene un valor, se elige el color blanco como color de fondo y entonces habría 2 posibilidades:
  - Se haya pulsado sobre un ítem ya elegido, en cuyo caso se desmarca el ítem y ítemElegido se pone a -1
  - Se haya pulsado en otro ítem, con lo que ítemElegido tomará el valor del índice.

```
Private Sub clvLibros_ItemClick (Índice  As Int, Valor  As Object)
  If libroElegido = -1  Then
    Private p  As B4XView  = clvLibros. GetPanel(Índice)
    p.GetView(0). Color = xui. Color_Blue
    libroElegido = Índice
  Else
    Private p  As B4XView = clvLibros. GetPanel(libroElegido)
    p.GetView(0). Color  = xui. Color_White
    If libroElegido = Índice Then
      libroElegido = -1
    Else
      Private p As  B4XView  = clvLibros. GetPanel(Índice)
      p.GetView(0). Color = xui. Color_Blue
      libroElegido = Índice
    End If
  End If
End Sub
```

7. Cargar ítem de un fichero KVS en un mapa

La sentencia se debe ejecutar con Wait For.

```
Wait For(fichEstudiante.GetMapAsync(fichEstudiante.ListKeys))
Complete(mapaEstudiante As Map)
```

El método **GetMapAsync** devuelve un mapa con los ítems. Primero hay que declarar el mapa **mapaEstudiante** e inicializarlo.

8. Guardar los ítems en un fichero KVS de mapas

```
Wait For (fichEstudiante.PutMapAsync(mapaEstudiante))  Complete  (Succes  As
Boolean)
```

9. Borrar un fichero de una carpeta:

```
File.Delete(File.DirTemp, <nombre fichero>)
```