SynEdit/es

From Free Pascal wiki

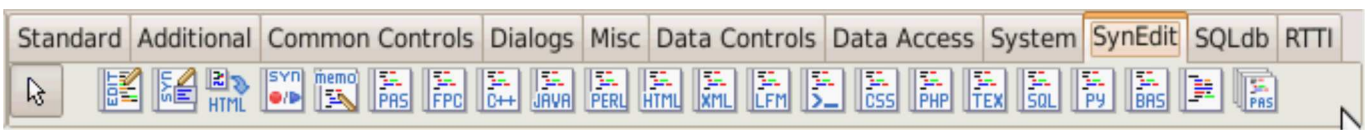
Deutsch (de) | **English (en)** | **Español (es)** | **Français (fr)** | **日本語 (ja)** | **Русский (ru)**

Contents

SynEdit

SynEdit es un paquete de Lazarus que incluye diversos componentes orientados a la implementación de editores de texto para diversos lenguajes y sintaxis. Estos componentes se pueden encontrar dentro de la paleta de componentes en la solapa 'SynEdit'.

Contiene un componente editor de código fuente llamado TSynEdit, varios resaltadores de sintaxis y otros componentes utilizados para la edición de código fuente.



TSynEdit, TSynAutoComplete, TSynExporterHTML, TSynMacroRecorder, TSynMemo, TSynPasSyn, TSynFreePascalSyn, TSynCppSyn, TSynJavaSyn, TSynPerlSyn, TSynHTMLSyn,

TSynXMLSyn, TSynLFMSyn, TSynUNIXShellScriptSyn, TSynCssSyn, TSynPHPSyn, TSynTeXSyn, TSynSQLSyn, TSynPhytonSyn, TSynAnySyn, TSynMultiSyn

Frecuentemente se usa el término "SynEdit" para referirse específicamente al componente TSynEdit (del paquete SynEdit) en lugar de referirse a todo el paquete en si.

Está basado en el proyecto independiente SynEdit [[1] (http://synedit.sourceforge.net/)], tomado en la versión 1.3 a partir del cual fue adaptado y extendido con características tales como soporte UTF-8 y plegado de código (code folding). A pesar de que el SynEdit de Lazarus comparte mucho del código original del SynEdit original, la cantidad de cambios introducidos desde su bifurcación, hacen que el actual SynEdit de Lazarus sea incompatible con el SynEdit original.

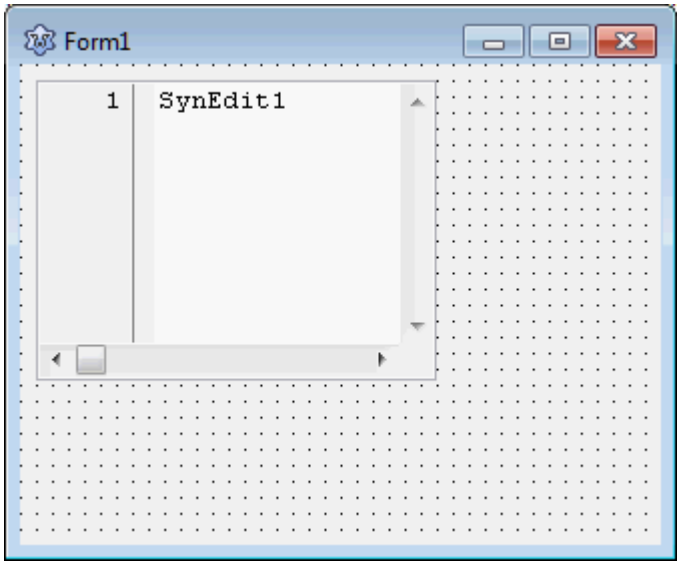
Está licenciado bajo los mismos términos que el SynEdit original (MPL o GPL).

Características

- Viene incluido en la paleta de componentes de Lazarus.
- Es un componente que no requiere de dependencias externas (a diferencia de Scintilla).
- Es de ejecución rápida. Está optimizado para manejar bloques grandes de datos.
- Trabaja en codificación UTF-8.
- Permite implementar coloreado de sintaxis, usando resaltadores predefinidos o creando nuevos resaltadores.
- Soporta las opciones de "Deshacer" y "Rehacer"
- Implementa funciones integradas de Búsqueda/Reemplazo.
- Soporta completado y autocompletado de código.
- Soporta numeración de líneas.
- Soporta plegado de código (folding).
- Soporta marcadores de texto, con íconos.
- Incluye opciones para exportar a 'html' o 'rtf'.
- Soporta selección sencilla en modo columna.
- Permite trabajar con complementos.

Uso de SynEdit

El componente TSynEdit, se agrega como cualquier control desde la paleta de componentes. Una vez en el formulario ya es funcional como control de edición de texto.



El componente TSynEdit se comporta como un control TMemo, pero con característcias avanzadas de presentación. Todo el contenido se mostrará en un solo tipo de letra, y con caracteres mono-espacio (del mismo ancho).

Por defecto tiene habilitadas las funciones del cortapapeles (cortar, copiar, pegar), y las opciones para deshacer y rehacer.

Además incluirá las barras de desplazamiento y un panel vertical (Gutter) con información útil sobre el contenido (como el número de línea).

Inicialmente no incluirá opciones de resaltado excepto el remarcado de los delimitadores paréntesis, llaves y corchetes (brackets).

Gutter

El panel lateral que aparece a la izquierda del control es llamado 'Gutter' o canal. Aquí se muestra información importante del editor, como:

- Marcadores.
- Número de Línea.
- Marcas de cambio.
- Marcas de plegado.

La propiedad 'Gutter', permite acceder a las propiedades del panel lateral. Para mostrar u ocultar el panel lateral, se accede al campo 'Visible':

```
  SynEdit1.Gutter.Visible := False;
```

Para tener una mejor idea de las propiedades que se pueden configurar, es mejor usar el Inspector de Objetos.

Margen Derecho

Al crear el editor, se crea por defecto un margen visible, al lado derecho del contenido del editor.

Este margen es una ayuda visual para limitar el ancho del texto, con fines de no exceder el tamaño de una página para cuando se desee imprimir el contenido.

Para cambiar su posición:

```
SynEdit1.RightEdge:= 100;  //en pixeles
```

Para ocultarlo:

```
SynEdit1.Options := SynEdit1.Options + [eoHideRightMargin];
```

Coordenadas Lógicas y Físicas

La información mostrada en el editor, está distribuida en la pantalla como si fuera una matriz de celdas. Por otro lado la información que se desea mostrar, está por lo general, distribuida como un arreglo de bytes (variables de tipo cadena).

La relación:

(Caracter en pantalla) <---> (Byte de información)

No es una relación de 1 a 1:

- Un caracter en pantalla puede estar representado por más de un byte, (efecto de la codificación UTF-8)
- Un byte de datos puede representar más de un caracter en pantalla (efecto de las tabulaciones).

Para manejar esta falta de correspondencia, SynEdit maneja dos tipos de coordenadas para el cursor de texto (Caret):

- Física: Corresponde a la posición visible del cursor en la pantalla.
- Lógica: Corresponde a la posición del cursor dentro de la cadena de texto real.

La coordenada X lógica y física suelen ser distintas. La coordenada Y lógica y física son siempre iguales (Podría variar en el futuro).

La coordenada física es la posición en la grilla de la pantalla (ignorando los desplazamientos). Esto significa que:

> Las letras "a" y "á" ocupan ambas una celda en pantalla, y se cuentan como un caracter, aún cuando en UTF-8 (la codificación usada por SynEdit) la letra "a" ocupa un byte, y "á" ocupa dos bytes (en coordenadas lógicas tiene dos bytes de ancho). Existen otros caracteres en UTF-8, que pueden ocupar 3 o hasta 4 bytes.
> El caracter de tabulación (#9), por otros lado, ocupa un byte como cualquier otro caracter, pero en la pantalla puede ocupar más de una celda, es decir que físicamente ocupa más de una celda de ancho.

La coordenada lógica está referida a la posición que ocupan los bytes que representan a un caracter, dentro de toda la cadena.

> La letra "a" tiene un byte de ancho, e incrementa la coordenada lógica en 1.
> La letra "á" tiene dos byte de ancho, e incrementa la coordenada lógica en 2.
> El caracter de tabulación tiene un byte de ancho e incrementa la coordenada lógica en 1.

La coordenada X física es siempre contabilizada desde la izquierda del texto, aún si hay desplazamiento en

la visualización.

Para obtener la celda-X actual desplazada en SynEdit se debe hacer:

grid-X-in-visible-part-of-synedit := PhysicalX - SynEdit.LeftChar + 1

Para obtener la celda-Y actual desplazada en SynEdit se debe hacer:

grid-y-in-visible-part-of-synedit := SynEdit.RowToScreenRow(PhysicalY); // incluye plegado

Búsqueda y Reemplazo

Para realizar las operaciones de búsqueda y reemplazo sobre SynEdit, existen dos métodos especiales:

```
function SearchReplace(const ASearch, AReplace: string;
      AOptions: TSynSearchOptions): integer;

function SearchReplaceEx(const ASearch, AReplace: string;
      AOptions: TSynSearchOptions; AStart: TPoint): integer;
```

La función SearchReplaceEx(), es similar a SearchReplace(), excepto que permite definir la posición inicial desde donde se realizará la búsqueda.

Las opciones que se pueden usar en 'AOptions' son:

- ssoMatchCase
- ssoWholeWord
- ssoBackwards
- ssoEntireScope
- ssoSelectedOnly
- ssoReplace
- ssoReplaceAll
- ssoPrompt
- ssoSearchInReplacement
- ssoRegExpr
- ssoRegExprMultiLine
- ssoFindContinue

En búsqueda, la función SearchReplace() puede devolver los siguientes valores:

0 -> Elemento no encontrado 1 -> Se encontró al menos un elemento.

En reemplazo, la función SearchReplace() puede devolver los siguientes valores:

0 -> Elemento no encontrado n -> Se reemplazó “n” elementos

Ejemplo de búsqueda simple:

```
var
  encontrado : integer;
  buscado : string;

...
  buscado := 'texto a buscar';
  encontrado := editor.SearchReplace(buscado,'',[]);
  if encontrado = 0 then
    ShowMessage('No se encuentra: ' + buscado);
...
```

Si se encuentra algún elemento que cumpla el criterio de búsqueda, se selecciona la primera coincidencia. Si es que la selección no es visible, se desplaza el contenido del editor, de modo que la selección se haga visible.

Por defecto, la búsqueda se hace desde la posición del cursor hacia adelante, ignorando las mayúsculas/minúsculas.

Con el siguiente código, se realiza la búsqueda en sentido contrario:

```
  encontrado := editor.SearchReplace(buscado,'',[ssoBackwards]);
```

Para realizar reemplazo, se debe indicar la opción 'ssoReplace':

```
var
  encontrado : integer;
  buscado : string;
begin
  buscado := 'texto a buscar';
  encontrado := editor.SearchReplace(buscado,'texto a reemplazar',[ssoReplace]);
  if encontrado = 0 then
     ShowMessage('No se encuentra: ' + buscado);
...
```

Si es que se encuentra alguna coincidencia, se reemplaza el texto, y se hace visible en el editor.

Remarcado de texto

SynEdit soporta diversos tipos de remarcados para el contenido:

Se puede cambiar el color de fondo de una línea implementando el evento 'OnSpecialLineMarkup'.

```
procedure TForm1.SynEdit1SpecialLineMarkup(Sender: TObject;
                    Line: integer; var Special: boolean;
                    Markup: TSynSelectedColor);
begin
  if Line = 5 then begin //línea a marcar
      Special := True ;
      Markup.Background := clGreen; //color de fondo
  end;
end;
```

Para marcar ciertos bloques de texto, se puede usar el método 'SetHighlightSearch':

```
uses ..., SynEditTypes;
...
SynEdit1.HighlightAllColor.Background := clRed;
SynEdit1.SetHighlightSearch('xyz', [ssoSelectedOnly]) ;
```

Con este método es posible marcar cualquier combinación de caracteres, no solo alfabéticos. Las opciones

que se usan son similares a las opciones de las operaciones de búsqueda/reemplazo.

Para información sobre remarcados complejos, ver:

http://forum.lazarus.freepascal.org/index.php/topic,17415.msg95964.html

http://forum.lazarus.freepascal.org/index.php/topic,17485.msg96510.html

Marcadores de texto (BookMarks)

Los marcadores de texto permiten mostrar un ícono en el panel vertical del editor (Gutter).

El siguiente ejemplo, muestra como crear un marcador de texto y hacerlo visible en la línea 2:

```
uses ... , SynEditMarks;
var
  marcador: TSynEditMark;
...

marcador := TSynEditMark.Create(SynEdit1);  //nuevo marcador
marcador.ImageList:=ImageList1;
marcador.ImageIndex:=1;         //elige su ícono
marcador.Line:=1;               //línea 2
marcador.Visible:=true;
SynEdit1.Marks.Add(marcador);   //agrega el marcador
```

La lista de imágenes ImageList1, debe conteenr el ícono que aparecerá en el panel vertical del editor.

Modificando el texto por código

Para fijar o leer el contenido de SynEdit, se puede acceder a la propiedad 'Text'.

La información del texto que muestra SynEdit, se guarda internamente en un TStrigList, visible en la propiedad 'Lines'.

Así, para acceder a la primera línea de SynEdit, se accede a:

SynEdit1.Lines[0]

Para modificar el contenido de SynEdit, existen diversos métodos:

- Accediendo directamente a 'SynEdit.Lines'. Este método es rápido, porque se modifica directamente a los datos del texto. Sin embargo, este tipo de modificación no guarda registros de las modificaciones hechas de modo que los cambios no se podrán deshacer con la opción 'Undo'.
- Usando métodos como InsertTextAtCaret, TextBetweenPoints o TextBetweenPointsEx. Este tipo de modificación del texto si permite deshacer los cambios. Es rápido, pero puede requerir conocer con precisión, las coordenadas del texto a modificar.
- Usando comandos de edición. Es la forma como se editaría el texto usando teclado. Se hace uso del método ExecuteCommand(). Este método es lento porque hace modificaciones por caracter. También permite deshacer los cambios.

- Usando el portapapeles. Este método está limitado a operaciones de cortado, copiado y pegado, pero se pueden realizar operaciones más elaboradas accediendo directamente al portapapeles para modificar su contenido. Tanbién se pueden deshacer los cambios.

Menus con Duplicar e intercambiar líneas

```
//Duplicar Línea
procedure TForm1.MenuDuplicarLinea(Sender: TObject);
begin
 SynEdit1.Lines.Insert(CustSynEdit().CaretY,CustSynEdit().LineText);
end;

//Intercambiar Línea hacia Arriba
procedure TForm1.MenuLineaArriba(Sender: TObject);
begin
 if CustSynEdit.CaretY -1 >= 1 then
  begin
   CustSynEdit.Lines.Exchange(CustSynEdit.CaretY -1 ,CustSynEdit.CaretY-2);
   CustSynEdit.CaretY:=CustSynEdit.CaretY-1;
  end;
end;

//Intercambiar Línea hacia Abajo
procedure TForm1.MenuLineaAbajo(Sender: TObject);
begin
 if ( CustSynEdit.CaretY + 1 <= CustSynEdit.Lines.Count) then
  begin
   CustSynEdit.Lines.Exchange(CustSynEdit.CaretY-1,CustSynEdit.CaretY);
   CustSynEdit.CaretY:=CustSynEdit.CaretY+1;
  end;
end;
```

Insertar caracteres en la posición actual

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 SynEdit1.CommandProcessor(ecChar,'a', nil); //Inserta a
 SynEdit1.SetFocus;
end;
```

Portapapeles

SynEdit soporta complétamente las operaciones "Cortar", "Copiar" y "Pegar"

Las operaciones con el protapapeles se implementan con los siguientes métodos:

```
   SynEdit1.CopyToClipboard;
   SynEdit1.CutToClipboard;
   SynEdit1.PasteFromClipboard;
```

Para obtener las coordenadas del bloque de selección, usar las propiedades: BlockBegin y BlockEnd.

También se pueden acceder a las acciones del portapapeles, usando comandos de teclado. El siguiente ejemplo, muestra como implementar los atajos por teclado para las acciones 'Cortar', 'Copiar' y 'Pegar':

```
uses
 ...
 SynEdit, SynEditKeyCmds;

procedure TfrmPrincipal.HandleCodigoKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
 if (Shift = [ssCtrl]) then
 begin
   case Key of
   VK_C: synCodigo.CommandProcessor(ecCopy, ' ', nil);
   VK_V: synCodigo.CommandProcessor(ecPaste, ' ', nil);
   VK_X: synCodigo.CommandProcessor(ecCut, ' ', nil);
   end;
 end;
end;
```

Menú con Deshacer y Rehacer

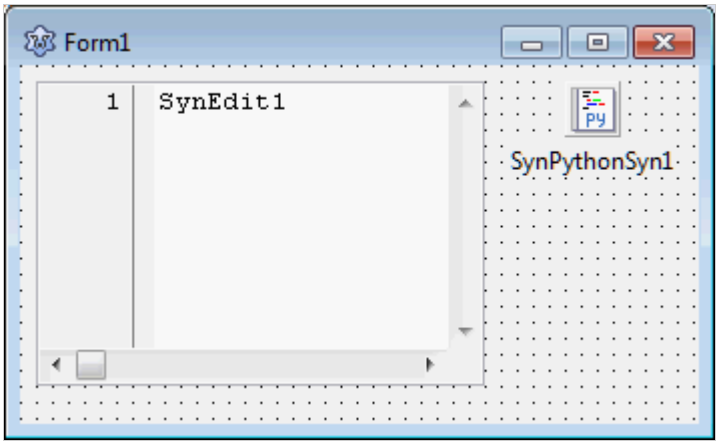
```
procedure TForm1.MenuRehacer(Sender: TObject);
begin
 SynEdit1.Redo;
end;

procedure TForm1.MenuDeshacer(Sender: TObject);
begin
 SynEdit1.Undo;
end;
```

Resaltado de sintaxis

SynEdit soporta el resaltado de sintaxis, usando atributos de texto como el color o el tipo de letra.

Para que SynEdit reconozca una sintaxis en particular, debe estar asociado a un objeto resaltador (Highlighter). Existen diversos componentes con varias sintaxis ya creadas (resaltadores) en la paleta de componentes de Lazarus.



Control SynEdit con resaltador de Python.

Para asociar a un componente SynEdit a un resaltador, se debe hacer que su propiedad "Highlighter", apunte al resaltador de sintaxis. Esto se puede hacer usando el Inspector de Objetos o por código.

Luego, se puede cambair los atributos de cada tipo de token (identificadores, números, palabras reservadas, etc), cambiando las propiedades del resaltador, que sean de tipo "TSynHighlighterAttributes". Existen diversos atributos que se pueden cambiar, como el color del texto, el color de fondo o el color del borde.

También se pueden encontrar resaltadores adicionales en Highlighter for SynEdit (http://bugs.freepascal.org/view.php?id=18248)

Si se quiere un resaltador para una sintaxis nueva, se puede usar alguno de los resaltadores que soportan personalización: SynAnySyn or SynPositionSyn (Ver ejemplos de uso).

Otra opción es crear nuestro propio resaltador de sintaxis. Hay algo de información en inglés en SynEdit_Highlighter. Pero hay más información, y en español en [2] (http://blog.pucp.edu.pe/action.php?action=plugin&name=LinkCounter&type=c&k=20140710-la_biblia_del_synedit_-_rev6.pdf) .

Completado de Código

SynEdit permite implementar la opción de completado de código, que permite mostrar una lista de opciones para completar el texto que se está escribiendo, o se puede completar el texto escrito parcialmente.

Existen 3 complementos de completado para SynEdit:

- TSynCompletion
  - Ofrece una lista de palabras en un menú contextual.
  - Usado en la IDE par el completado de identificadores.
  - Incluido en los ejemplos.
  - Disponible en la paleta de componentes desde la versión 0.9.31
- TSynAutoComplete
  - Reemplaza la palabra actual (si es identificada) con un texto pre-definido. **No** es interactivo ni incluye menú contextual.
  - Incluido en los ejemplos.
  - Disponible en la paleta de componentes.
- TSynEditAutoComplete
  - Módulo Básico de plantillas. **No** es interactivo ni incluye menú contextual.
  - Es usado por la IDE para el completamiento de codigo usando plantillas. La IDE contiene código adicional que extiende sus características.
  - **Not** incluido en los ejemplos.

Los tipos TSynAutoComplete y TSynEditAutoComplete no están muy bien diferenciados y probablemente se fusionen.

Se recomienda ver los ejemplos que vienen con Lazarus.

Completado de Código con SynCompletion

```
Uses
... ...
LCLType,//Para poder usar VK_SPACE
syncompletion;

... ... ....
var
Completion:TSynCompletion;
begin
 Completion:=TSynCompletion.Create(Self);
 Completion.Editor:=SynEdit;
 Completion.ShortCut:=Menus.ShortCut(VK_SPACE, [ssCtrl]);
 Completion.ItemList.add('Arc( )');
 Completion.ItemList.add('Axes( )');
 Completion.ItemList.add('Bezier( )');
 Completion.ItemList.add('Cercle( )');
end;
```

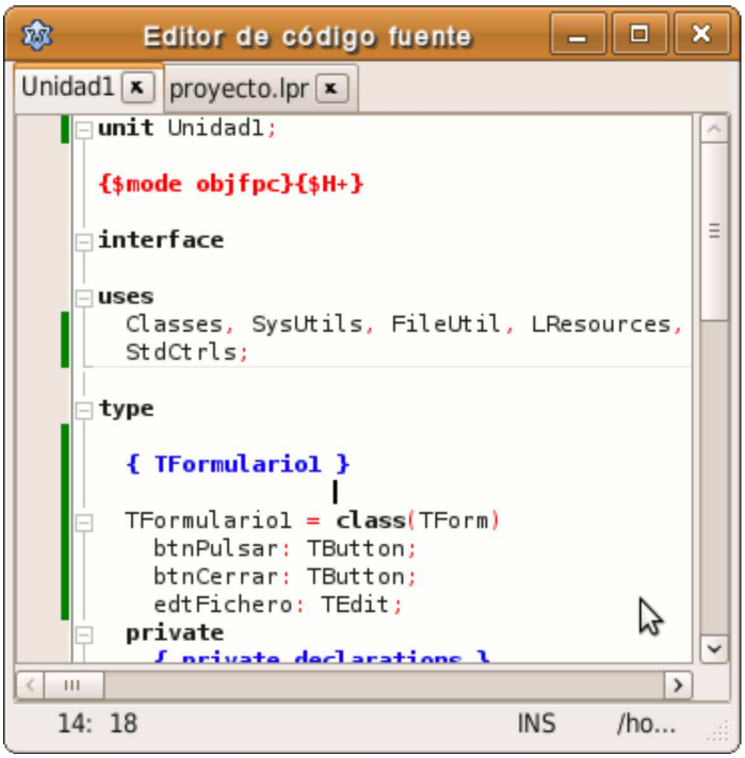
Desplegar TSynCompletion en la posición actual

```
procedure TForm1.Button1Click(Sender: TObject);
var p:TPoint;
 begin
  with  SynEdit1  do
   begin
     P := Point(CaretXPix,CaretYPix + LineHeight);
     P.X:=Max(0,Min(P.X,ClientWidth-Completion.Width));
     P := ClientToScreen(p);
   end;
  Completion.Execute('',p.x,p.y);
 end;
```

SynEdit en el IDE

El componente SynEdit es un paquete que ya viene integrado con Lazarus debido a que lo utiliza el propio IDE. Por lo tanto, el paquete no puede ser removido de la lista de instalación. Es por ello que no existe un fichero .lpk.

Al estar integardo en el IDE, se garantiza su estabilidad y madurez. Además podemos verlo trabajando en el mismo IDE y así conocer sus funcionalidades y potencialidades.



Synedit 2.0.5 port

Existe una versión alterna de SynEdit, portada desde la versión más reciente del original SynEdit 2.0.5:

http://wiki.lazarus.freepascal.org/SynEdit/port

code: https://github.com/rnapoles/

Documentación

Existe amplia información en La Biblia del SynEdit (http://blog.pucp.edu.pe/action.php?action=plugin&name=LinkCounter&type=c&k=20140710-la_biblia_del_synedit_-_rev6.pdf)

Sobre resaltado de sintaxis (en ingles): SynEdit Highlighter

Más desarrollos, discusiones

- RTL (de derecha a izquierda (Right-To-Left)): comenzado por Mazen
- Selección automática de fuente UTF-8: Igual que el caso anterior de monoespaciado, pero con una cadena UTF-8 de fuente, de modo que por ejemplo umlaute se muestra correctamente. Por ahora el

usuario tiene que elegir la fuente correcta.
- Teclas muertas. La mayoría de los teclados soportan pulsar una o más teclas para crear un carácter especial (como en los caracteres acentuados (á,é,í,...) o con el umlaut (http://es.wikipedia.org /wiki/Umlaut)).
- Rediseño del componente SynEdit. El objetivo principal es una presentación y un navegación por el texto más fiable. Un enfoque más modular para permitir una mejor integración de las extensiones, y por controles especializados, para su utilización fuera de Lazarus.