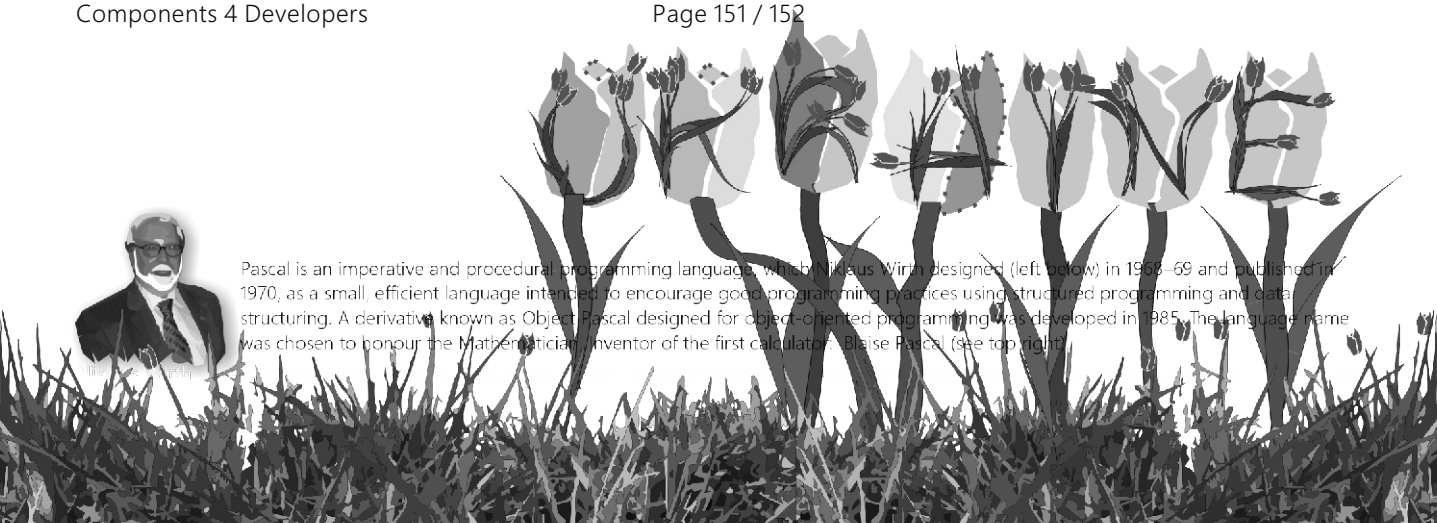
Code with Geo Location
FastReport: New Version
Using the Table Object in Table Reports
The Ceva project
The distance of diagonals in a cube
Creating Electron apps with PAS2JS
Installing FPC Trunk versions
SKIA for Delphi the next level
Quantum Cryptography
Tested: The new version of Database Workbench
PAS2JS samples for creating your new website
Anonymous functions Explained for Delphi and Lazarus

CONTENT

ARTICLES

ADVERTISERS



Pascal is an imperative and procedural programming language, which Niklaus Wirth designed (left below) in 1968–69 and published in 1970, as a small, efficient language intended to encourage good programming practices using structured programming and data structuring. A derivative known as Object Pascal designed for object-oriented programming was developed in 1985. The language name was chosen to honour the Mathematician, Inventor of the first calculator, Blaise Pascal (see top right).

Stephen Ball
http://delphiaball.co.uk
@DelphiABall

Dmitry Boyarintsev
dmitry.living @ gmail.com

Michaël Van Canneyt,
michael @ freepascal.org

Marco Cantù
www.marcocantu.com
marco.cantu @ gmail.com

David Dirkse
www.davdata.nl
E-mail: David @ davdata.nl

Benno Evers
b.evers @ everscustomtechnology.nl

Bruno Fierens
www.tmssoftware.com
bruno.fierens @ tmssoftware.com

Holger Flick
holger @ flixments.com

Mattias Gärtner
nc-gaertnma@netcologne.de

Max Kleiner
www.softwareschule.ch
max @ kleiner.com

John Kuiper
john_kuiper @ kpnmail.nl

Wagner R. Landgraf
wagner @ tmssoftware.com

Vsevolod Leonov
vsevolod.leonov@mail.ru

Andrea Magni www.andreamagni.eu
andrea.magni @ gmail.com
www.andreamagni.eu/wp

Paul Nauta PLM Solution Architect
CyberNautics
paul.nauta @ cybernautics.nl

Kim Madsen
www.component4developers.com

Boian Mitov
mitov @ mitov.com

Jeremy North
jeremy.north @ gmail.com

Detlef Overbeek - Editor in Chief
www.blaisepascal.eu
editor @ blaisepascal.eu

Howard Page Clark
hdpc @ talktalk.net

Heiko Rompel
info @ rompelsoft.de

Wim Van Ingen Schenau -Editor
wisone @ xs4all.nl

Rik Smit
rik @ blaisepascal.eu

Bob Swart
www.eBob42.com
Bob @ eBob42.com

B.J. Rao
contact @ intricad.com

Daniele Teti
www.danieleteti.it
d.teti @ bittime.it

Anton Vogelaar
ajv @ vogelaar-electronics.com

Danny Wind
dwind @ delphicompany.nl

Jos Wegman / Corrector / Analyst

Siegfried Zuhr
siegfried @ zuhr.nl

Editor - in - chief
Detlef D. Overbeek, Netherlands Tel.: Mobile: +31 (0)6 21.23.62.68
News and Press Releases email only to **editor@blaisepascal.eu**

Editors
Peter Bijlsma, W. (Wim) van Ingen Schenau, Rik Smit

Correctors
Howard Page-Clark, Peter Bijlsma

Subscriptions (2019 prices)

	Internat. excl. VAT	Internat. incl. 9% VAT	Shipment
Printed Issue ±60 pages	€ 155,96	€ 250	€ 80,00
Electronic Download Issue 60 pages	€ 64,20	€ 70	———
Printed Issue inside Holland (Netherlands) 60 pages	———	€ 250,00	€ 70,00

Subscriptions can be taken out online at www.blaisepascal.eu or by written order, or by sending an email to office@blaisepascal.eu
Subscriptions can start at any date. All issues published in the calendar year of the subscription will be sent as well.
Subscriptions run 365 days. Subscriptions will not be prolonged without notice. Receipt of payment will be sent by email.
Subscriptions can be paid by sending the payment to:
ABN AMRO Bank Account no. 44 19 60 863 or by credit card or Paypal
Name: Pro Pascal Foundation-Foundation for Supporting the Pascal Programming Language (Stichting Ondersteuning Programeertaal Pascal)
IBAN: NL82 ABNA 0441960863 BIC ABNANL2A VAT no.: 81 42 54 147 (Stichting Programmeertaal Pascal)
Subscription department
Edelstenenbaan 21 / 3402 XA IJsselstein, The Netherlands
Mobile: + 31 (0) 6 21.23.62.68 office@blaisepascal.eu

Member and donator of WIKIPEDIA
Member of the **Royal Dutch Library**

From your editor

This is a very special edition: it contains 152 pages.
I had to create it like this because a lot of new things were arriving, but to document that all is quite a task - to get it done within a certain time frame - especially Anonymous functions and the article about Pas2JS needed a lot of time until we (the author and me) were satisfied.
So that is why this issue has so many titles.
I am quite sure you will like it. I have been looking for a real good explanation of the subject of anonymous
functions to be well explained even for starters. I started it myself but after some time I found out that my knowledge of the background of this was to small and so I asked Michael van Canneyt to help.

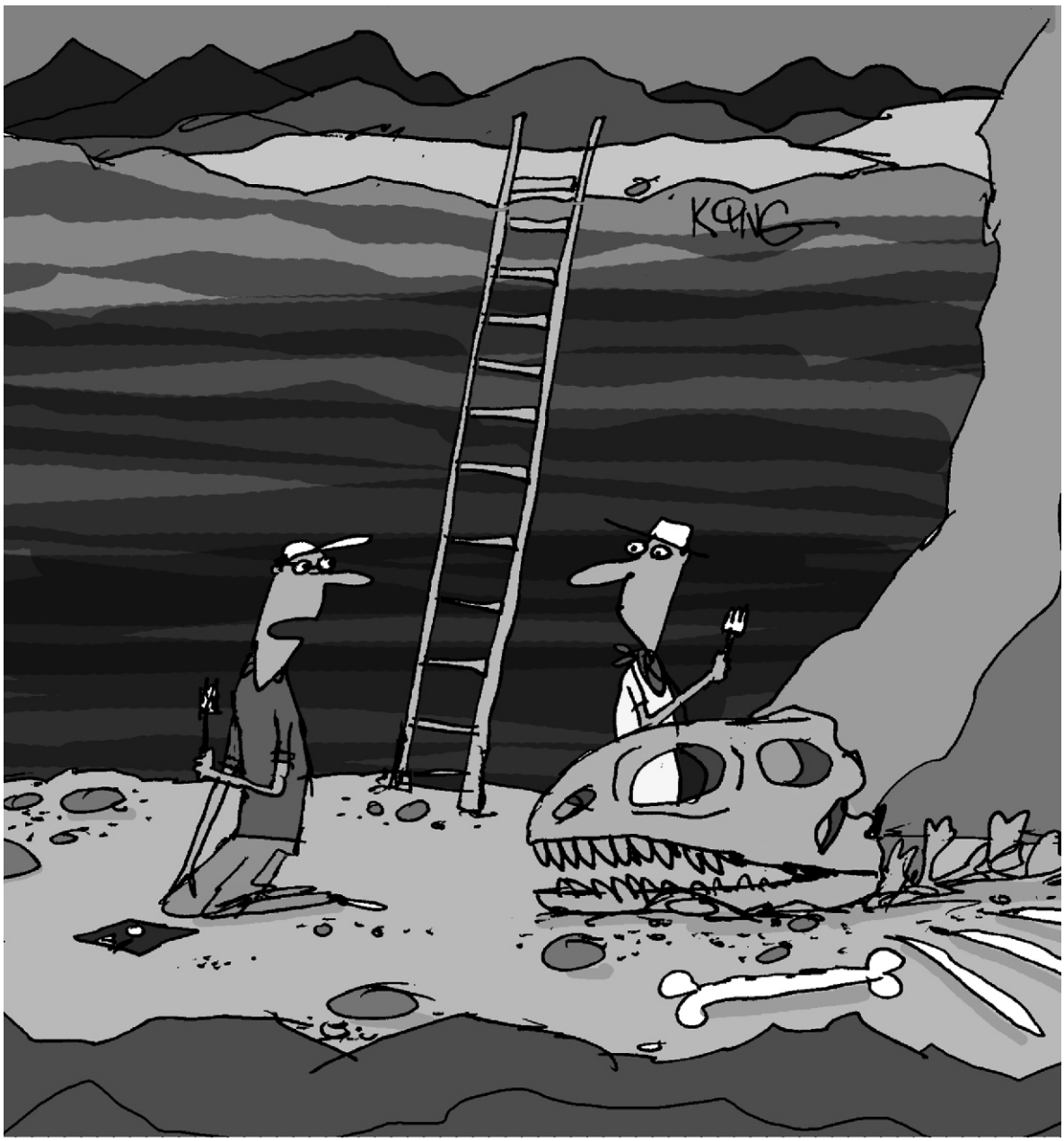
That was a real good shot: it took him again a week to get it all together and I am curious about your critics on this article. Just let me know. In all cases I did some research on the subject I found technical
advices, but no clear comprehensible description.

Pas2JS is a very special framework which will make Lazarus even more useful. Webassembly has now been added and we have some big surprises for the future of Lazarus: the look and feel of Lazarus will be enormously upgraded so you will be capable to make any component or drawing colourful and agile as it would be on the web. Even moving items will be possible. So you can change the IDE of Lazarus to look like you would want it. We will announce the details in the next Issue 105.

Hope you have nice readings and some new insights...

Detlef

From our Technical advisor: Cartoons from Jerry King



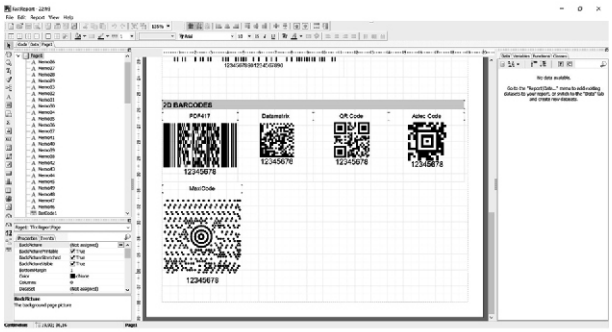
"Forget about that thing. I found something really old! It's a floppy disk."

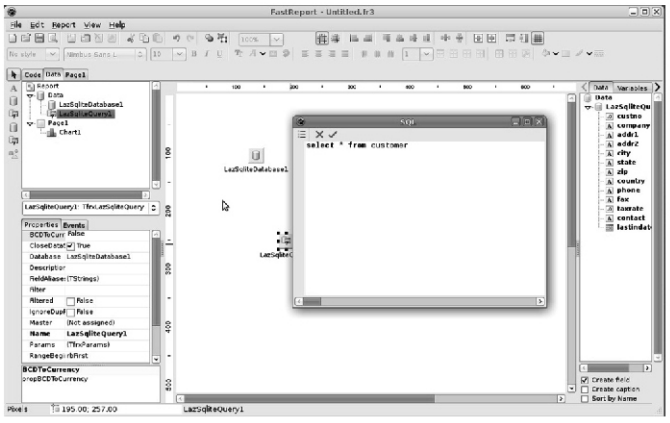
FastReport VCL is a modern solution for integrating Business Intelligence for Delphi

It is a set for generating reports and documents. It provides a visual template designer, access to the most popular datasources, a reporting engine, preview, export filters to 30+formats and deployment to clouds, web, e-mail, and print.

The latest version - FastReport VCL 2022.2 - includes:

- Support for Delphi 7-XE8, C++Builder 2005-XE8, Embarcadero Rad Studio 11 and Lazarus
- Extended interactive functionality
- Connection to 20+ databases
- Export in 45 Data Formats
- 40 types of bar codes
- Support for 48 Languages
- Cloud service integration





TRY FOR FREE:
http://fast.report/5562c

GEOCODING

AUTHOR: MAX KLEINER In the beginning there was nothing, which exploded. — Terry Pratchett

D11 starter expert

Geocoding is the processing of single addresses into geographic coordinates and can be performed by a number of free online web sites. Many of these same sites can also perform reverse geocoding, or the use of geographic coordinates to determine addresses or place names. The web site links below provide access to free single address or place name geocoding services. I want to show two of them, a commercial and a free one.

Just like every actual house has its address (*which includes the number, the name of the street, city*), every single point on the surface of earth can be specified by the latitude and longitude coordinates (**lat & lng**).

For example you see the location **Bonnaud** on the map above, we convert them to lat and long:

```
Coords: lat 46.621 lng 5.434 :
     Bonnaud, Val-Sonnette, Lons-le-Saunier, Jura, Bourgogne-Franche-Comté,
     France métropolitaine, 39190, France importance: 0.7350
```

Therefore, by using latitude and longitude we can specify virtually any point on earth, also like Delphi:

Geographic coordinates of Delphi, Greece
Latitude: 38°28'45" N
Longitude: 22°29'36" E
Elevation above sea level: 560 m = 1837 ft

OPENSTREETMAP

So how I did this? With the **OpenStreetMap API Nominatim**.
The **OSM Nominatim** is a search engine for **OpenStreetMap** data. From this site you may search for a name or address (*like Route de Bonnaud, Bonnaud, France*), or look up place names by geographic coordinates. Each result will link to details page where you can inspect what data about the object is saved in the database and investigate how the address of the object has been computed (*URI and JSON for example):*

```
function TAddressGeoCodeOSM(faddress: string): string;
var
  url, res, display: string;
  jo, location: TJSONObject;
  urlid: TIduri; windown: TWinApiDownload;

begin
  urlid:= TIdURI.create('');
  url:= urlid.URIEncode('https://nominatim.openstreetmap.org/search? format=json&q='+fAddress);
  windown:= TWinApiDownload.create;
  windown.useragent:= 'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1';
  windown.url:= url;
  windown.download1(res);

  //windown.OnWorkStart
  StrReplace(res,'[{','{');
  jo:= TJSONObject.create4(res);
  try
     if jo.getString('place_id') <> ''
     then display:= jo.getstring('display_name');

     result:= Format('Coords: lat %2.3f lng %2.3f :%s importance: %2.4f',
     [jo.getdouble('lat'),jo.getdouble('lon'),display,
     jo.getdouble('importance')]);
  except
     writeln('E: '+ExceptiontoString(exceptiontype,exceptionparam));
  finally
     jo.Free;
     urlid.free;
     windown.free;
  end;
end;
```

I use an **URI,** a **JSON** and a **HTTPGet** object and call the function:

```
writeln('res back_: '+TAddressGeoCodeOSM('Bonnaud, France'));
```

You should in particular verify that you have set a custom HTTP referrer or HTTP user agent (windown.useragent:=) that identifies your application, and that you are not overusing the service with massive bulk requests. Otherwise you get following message:

```
<p>You have been blocked because you have violated the<a
href="https://operations.osmfoundation.org/policies/nominatim/">usage policy</a>

of OSM's Nominatim geocoding service. Please be aware that OSM's resources are
limited and shared between many users. The usage policy is there to ensure that the
service remains usable for everybody.</p>
```

In case you missed the user agent, I passed it already with the constructor:

```
constructor TWinApiDownload.Create;
begin
  inherited;
  fUserAgent:= 'Mozilla/5.001(windows; U; NT4.0; en-US;) Gecko/25250101';
  fProgressUpdateInterval:= 100;
  fCachingEnabled:= True;
  fStop:= False;
  fActive:= False;
end;
```

So with **OpenStreetMap** the call is easy just to get an `https:// and user agent to provide.` **OpenStreetMap (OSM)** is a collaborative project to create a free editable map of the world. It is built by a community of mappers that contribute and maintain **OSM** data.
This **REST** endpoint is also used to reverse geocode, this is, generate an address from a latitude and a longitude. This is the format of the request:

```
with TWinApiDownload.create do
  begin
    Useragent:= 'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1';
    Url:= 'https://nominatim.openstreetmap.org/reverse?lat='
    +floats(flat)+'&lon='+floats(flong)+'&zoom=10&format=json';
    Download1(rest);

    //writeln(rest)
    with TJSONObject.create4(rest) do begin
    writeln('display_name: '+getstring('display_name'));
    free;
  end;
  free; //ApiDown
end;
```

The double with statement is functional programming, the rest variable is a stringstream in Download1(rest); and we free the 2 objects immediately.

A geocoding service (*also called an address locator*) is a program that allows for a user to input a batch of data contained in a table, search for matches as compared to a reference table, and output the result in a map or **GIS** layer format.

Now we can turn our geocoding coordinates in a map:

```
https://www.latlong.net/c/?lat=46.617&long=5.430
```

Two attributes in **OSM** are of interest, the zoom and the importance. Renderers can't display on a map in each zoom level, so they have to make a selection. The more important an object is, the earlier (*in zooming in*) the object should be rendered. Also the this key "importance" with a value which refers to the area this object is important for, e.g. regional or urban:

```
res back_: Coords: lat 46.62084 lng 5.43424 :Bonnaud, Val-Sonnette, Lons-le-Saunier,
Jura, Bourgogne-Franche-Comté, France métropolitaine, 39190, France importance:
0.7350
```

(46.617, 5.430) Lat & Long Map

The latitude 46.617 and longitude 5.430 shown on map.

GPS coordinates: 46° 37' 1.2000" N 5° 25' 48.0000" E

By the way you can simply open the map
in maXbox with the command:

```
OpenWeb('https://www.latlong.net/c/?lat=46.617&long=5.430');
```

Currently there are many options to choose from when geocoding batches of address data.
Lets turn our interest to commercial Services. We all know **Google** or the **Google Maps Platform:**
This **Geocoding API** can be used to geocode worldwide addresses after obtaining an **API** key.

You will need to enter a credit card to set up a billing account with **Google,** and they will give you $200/month credit for the first 12 months (*view their pricing structure*).
They will not charge you until you give them permission to. You may also apply for additional educational credit up to $250/month. In case you don't have a valid billing:
Exception:

```
TJSONObject["status--\u0020You\u0020must\u0020enable\u0020Billing\u0020on\
u0020the\u0020Google\u0020Cloud\u0020Project\u0020at\u0020https\
u003A//console.cloud.google.com/project/_/billing/enable\u0020Learn\u0020more
\u0020at\u0020https\u003A//developers.google.com/maps/gmp-get-started"] not
found.
```

GEOCODIO

Geocoding (also known as forward geocoding) allows you to convert one or more addresses into geographic coordinates (*i.e. latitude and longitude*). **Geocoding** will also parse the address and append additional information (*e.g. if you specify a zip code, Geocodio will return the city and state corresponding zip code as well*).

Geocodio supports geocoding of addresses, cities and zip codes in various formats. **Geocodio** provides bulk geocoding and reverse lookup services through a **REST API.** The **API** is able to process a single address, as well as handle bulk requests of up to 10,000 addresses. The results are returned with an accuracy score indicating the confidence **Geocodio** has in the accuracy of the result. It is also able to parse addresses into individual components.

So I started with **Geocodio** `https://www.geocod.io/docs/#geocoding` and registered to get an **API** Key. **Geocodio's RESTful API** allows you to perform forward and reverse geocoding lookups.
They support both batch requests as well as individual lookups, but and this is the disadvantage, for the moment only for USA and Canada.
The base **API** url is `https://api.geocod.io/v1.7/.`
All HTTP responses (*including errors*) are returned with JSON-formatted output.
For the code I tested with Python and maXbox with an Wininet and JSON Object:

```
var Url,API_KEY, source: string;
jo, locate: TJSONObject; urlid: TIdURI; fLat,fLong: double;

begin
   urlid:= TIdURI.create('');
   API_KEY:='785b4141b.....'; //get your own one please
   if fcountry <> '' then
      Url:= urlid.URLEncode('https://api.geocod.io/v1.7/geocode?q='+
         fAddr+'&country='+fcountry+'&api_key='+API_KEY) else
      Url:= urlid.URLEncode('https://api.geocod.io/v1.7/geocode?q='+
         fAddr+'&api_key='+API_KEY);
      jo:= TJSONObject.Create4(wdc_WinInet_HttpGet2(Url,Nil));
   try
      jo.getString('input')
      locate:= jo.getJSONArray('results').getJSONObject(0).getJSONObject('location');
      source:= jo.getJSONArray('results').getJSONObject(0).getString('source');

      //geometry.getJSONObject('coordinates');
      fLat:= locate.getDouble('lat')
      fLong:= locate.getDouble('lng');
      result:=Format('Coordinates: lat %2.3f lng %2.3f :%s',[flat,flong,source]);
   except
      Xraise(Exception.Create(jo.getString('error')));
   finally
      jo.Free;
      urlid.free;
   end;
end;
```

You can test your REST-Call direct in the browser with:

```
https://api.geocod.io/v1.7/geocode?
q=cologne&api_key=785b4141b&#8230...;
```

Or you choose a map one tester online:

```
https://www.geoapify.com/geocoding-api
```

One note to the code function above. I pass and address optional the country (to prevent name conflicts) to get the coordinates from the location:

```
writeln(TAddressGeoCoding4('Ontario','Canada'));
```

In most cases, the standard output format would be used. In certain situations, it can however be beneficial to work with a JSON structure that is specifically designed for your use case.

The interesting part is the line `jo:= TJSONObject.Create4(wdc_WinInet_HttpGet2(Url,Nil));` with the constructor to pass the httpget2 which returns the json result in one line to the `TJSONObject`.

If a **JSON** object is posted, you can specify a custom key for each element of your choice.
This can be useful to match queries up with your existing data after the request is complete.
If using a JSON array, results are guaranteed to be returned in the same order as they are requested.
The Geocodio API implements the concept of "warnings".
This is meant to assist and guide developers when implementing our API.

But I checked also the error as **Json** field in the try except catch.
To prevent misleading, inconsistent or duplicated locationslike Cologne, Paris or Bern all over the world we should also check the source as identity authentication too.

If a city is provided without a state, **Geocodio** will automatically guess and add the state based on what it is most likely to be. **Geocodio** also understands shorthand's for both streets and cities, for example NYC, SF, etc., are acceptable city names.

Each geocoded result is returned with an accuracy score (`like the importance in OpenStreetMap`), which is a decimal number ranging from 0.00 to 1.00. This score is generated by an internal **Geocodio** engine based on how accurate the result is believed to be.
The higher the score, the better the result. Results are always returned ordered by accuracy score.

CONCLUSION

Geocoded locations expressed in latitude, longitude coordinates can be obtained one at a time in web maps such as **OSM, Geocodio, Bing Maps or Google Maps**.

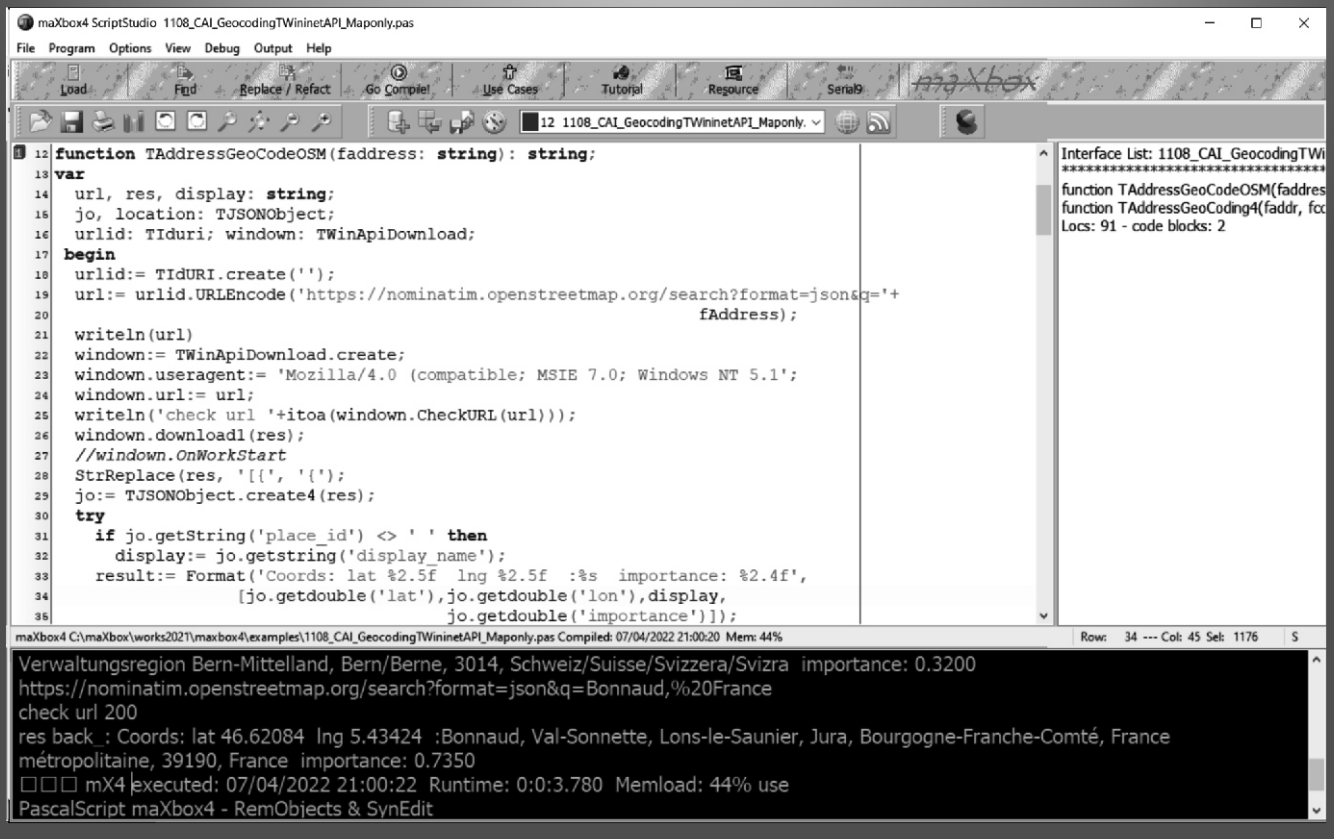
Reverse geocoding is the process of converting latitude and longitude into a street address. The relative ease of geocoding and resulting accuracy can vary widely depending on a number of factors. What is the nature or the source of the data? How accurate is it and what format is it in? What geocoding technique will be used like **REST** or batch or scripts? Determining a geocoding strategy that best suits a particular need is not always clear.

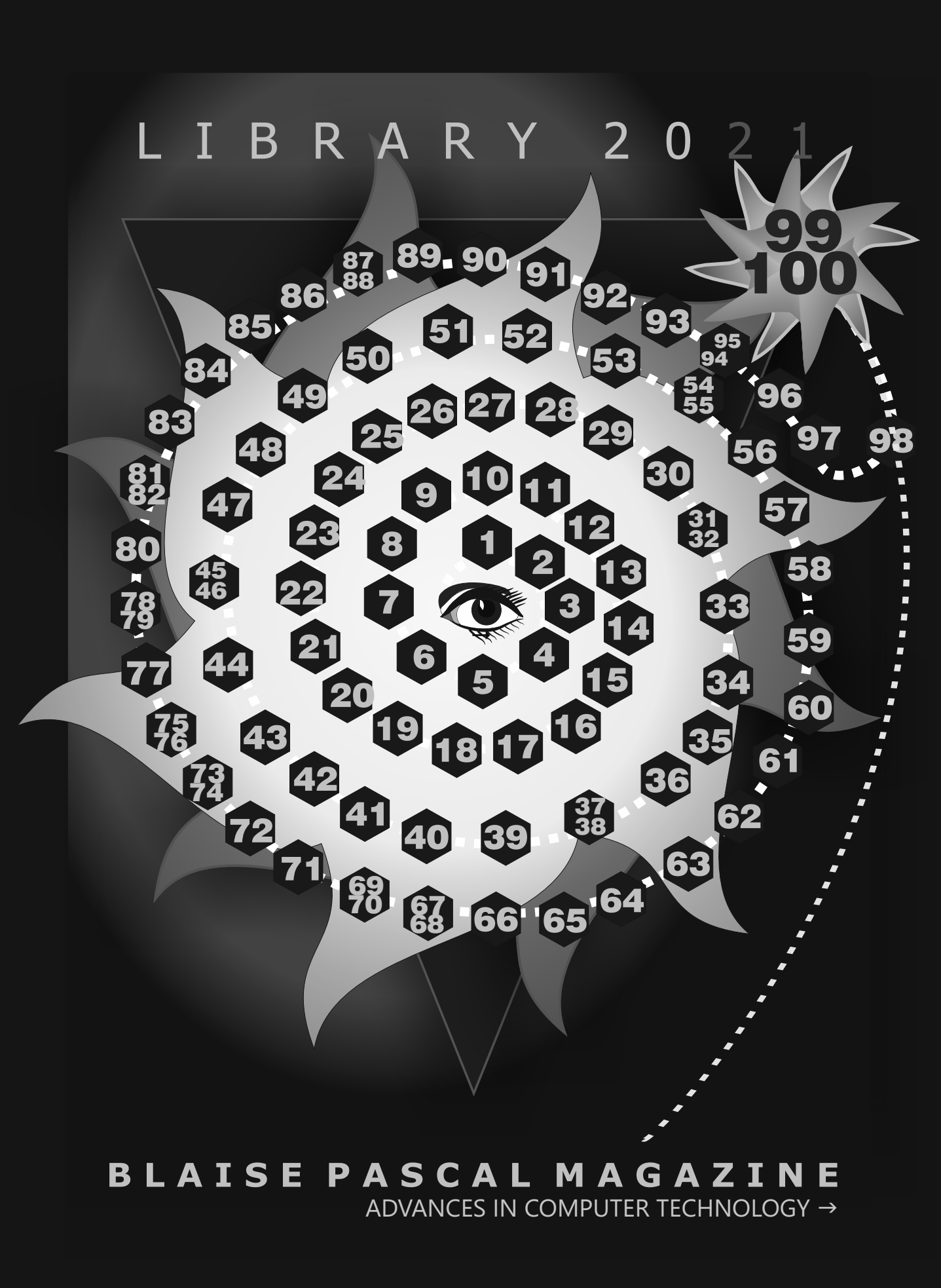
The key to confidently geocoding data lies in understanding the reference table of the **GIS** which the data is being matched to, how a match is found, and the resulting spatial accuracy.

`Scriptname: 1108_CAI_GeocodingTWininetAPI_Maponly.pas`

GEOCODING TOPICS AND SCRIPT

- `https://www.geoapify.com/geocoding-api`
- `https://www.latlong.net/`
- `https://www.openstreetmap.org/`
- `https://sourceforge.net/projects/maxbox/files/Examples/EKON/EKON26/1108_CAI_GeocodingTWininetAPI_Maponly.pas/download`
- `https://www.countrycoordinate.com/search/?keywords=bern%2C+switzerland`



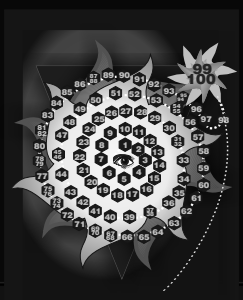

LIBRARY 2021
99
100
BLAISE PASCAL MAGAZINE
ADVANCES IN COMPUTER TECHNOLOGY →


LIBRARY 2019
LIBRARY 2020
LIBRARY 2021
SKIA4DELPHI PROJECT
LIBRARY 2022
BLAISE PASCAL MAGAZINE
TO THE NEXT LEVEL→

LibraryForm

Show article | Go to issue Nr | Issue | Search

ID	IssueNr	Author	Article	PDF	PageNr
854	98	Michael van Canneyt	GIT continued: Contributing		50
855	98	Fast Reports	Fast Reports: Uploading objects to FastReport using the http and https protocols		66
856	98	Detlef Overbeek	Recover Icons on your Windows desktop		75
857	98	Embarcadero	RAD Studio 11		76
858	99_100	TMS and Dr Holger Flick	New Books: Hands on with Delphi - a Preview		80
859	99_100	Max Kleiner	Image detection with Pascal and artificial intelligence		7
860	99_100	Detlef Overbeek	Speaking Sports Clock		63
861	99_100	Michael van Canneyt	GIT continued: Branching and partial commits for Lazarus and Delphi		36
862	99_100	Bob Swart	Testing IntraWeb Applications with Delphi		145
863	99_100	Paulo César Botelho Barbosa	Skia for Delphi		30



Show Thumbnails | Page | Jump to page


Page 30


Page 31


Page 32


Page 33


Page 34


Page 35


Page 36


Page 37


Page 38


Page 39


Page 40


Page 41

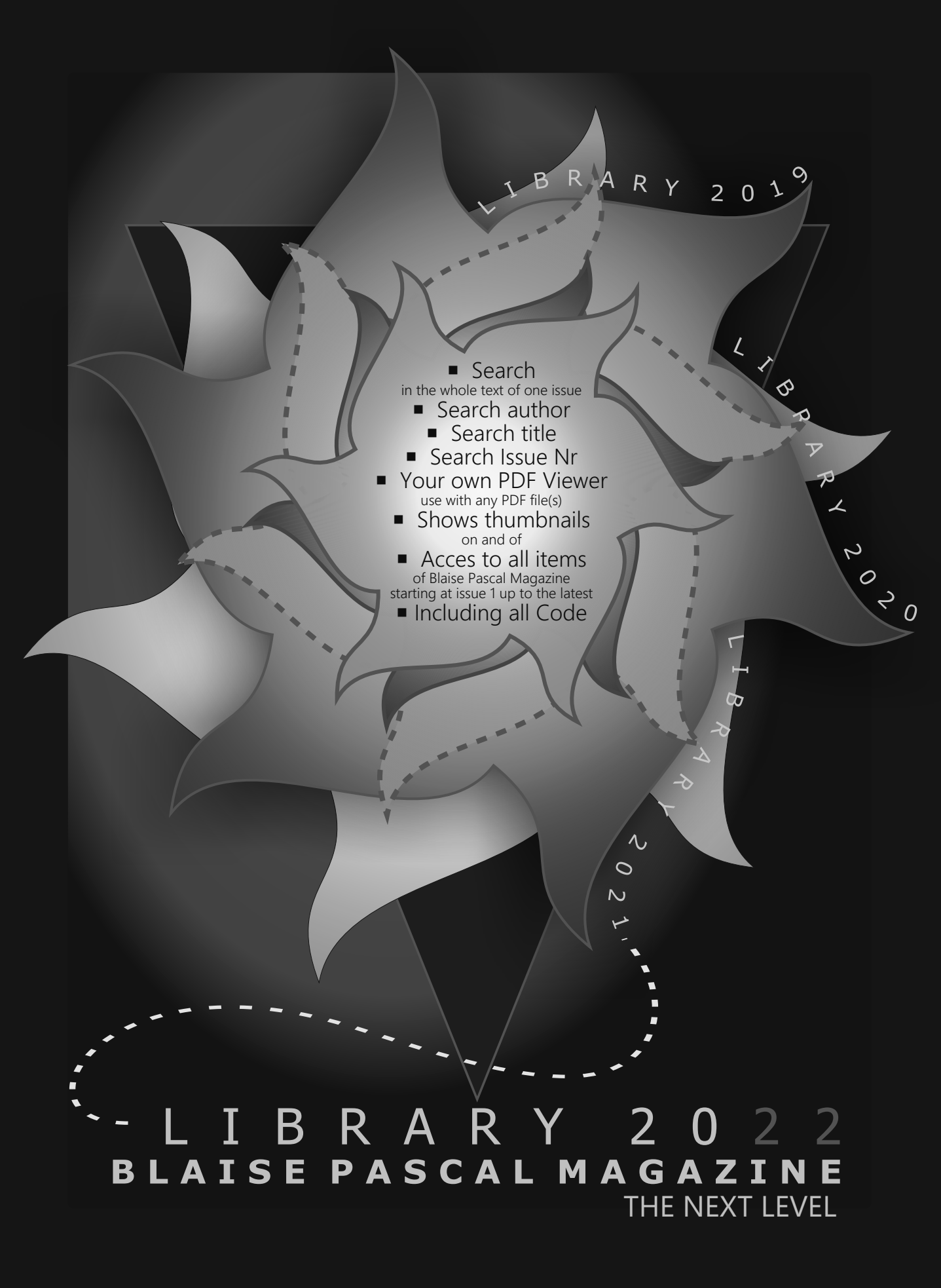

Page 42


Page 43


Page 44


Page 45


Page 46

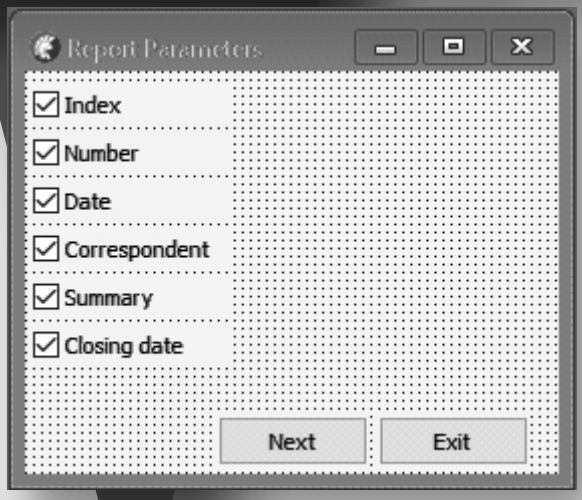

LIBRARY 2019
LIBRARY 2020
LIBRARY 2021
■ Search
in the whole text of one issue
■ Search author
■ Search title
■ Search Issue Nr
■ Your own PDF Viewer
use with any PDF file(s)
■ Shows thumbnails
on and of
■ Acces to all items
of Blaise Pascal Magazine
starting at issue 1 up to the latest
■ Including all Code
LIBRARY 2022
BLAISE PASCAL MAGAZINE
THE NEXT LEVEL

D11 starter expert

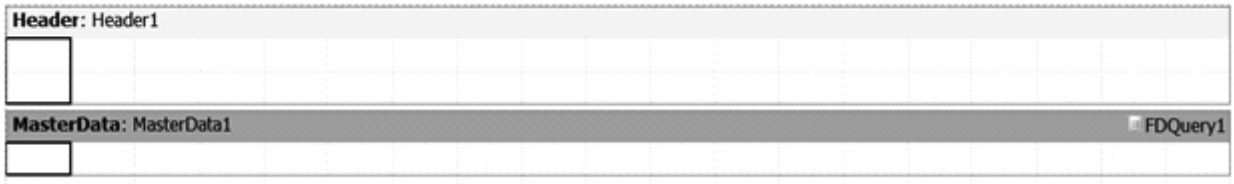
One of the important innovations of Fastreport is tables (TfrxTableObject component), which allows you to build tabular reports, I usually use it when I need to build a report with a variable number of columns.
For example, we have documents that include the following details:

- Index
- Number
- Date
- Correspondent
- Summary
- Date of completion

The user wants to print them as a table. However, in some cases, the user may not want to print all the details, so we should give him a choice of choosing document details for printing.
Let's create a new report, add a dialog form to the report, and place the names of the details on the form for selection.



In the "Data" section of the report, we place the data source, and on the report print page, we place a couple of bands: Header and MasterData, to which we link the data source. On the Header, we place the first TfrxTableObject (here we will have the table header), and on the MasterData we place the second TfrxTableObject (here we will have the table data). We'll get something like this:



Then let's write the following code:

```
procedure NextButtonOnClick(Sender: TfrxComponent);
var
  oColumn :TfrxTableColumn;
  oCell   :TfrxTableCell;
  nCol    :Integer;
  nCou            :Integer;

  nWidth  :Extended;
  nEndMax :Extended;

begin

  nCol := 0;

  TableObject1.DeleteColumn(0);
  TableObject2.DeleteColumn(0);

  if CheckBox1.Checked then begin

    oColumn := TableObject1.CreateNewColumn(nCol);
    oColumn.Tag := 10;
    oColumn.Width := GetWidth(oColumn.Tag)*fr1cm;
    oCell := TableObject1.Cells(nCol, 0);
    oCell.Memo.Text := 'Index';
    oCell.Font.Style := fsBold;
    oCell.HAlign := haCenter;
    oCell.VAlign := vaCenter;

    oColumn := TableObject2.CreateNewColumn(nCol);
    oColumn.Tag := 10;
    oColumn.Width := GetWidth(oColumn.Tag)*fr1cm;
    oCell := TableObject2.Cells(nCol, 0);
    oCell.Memo.Text := '[FDQuery1."INDEXNUMBER"]';
    oCell.HAlign := haCenter;
    oCell.VAlign := vaCenter;

    Inc(nCol);

  end; { if }
```

```
if CheckBox2.Checked then
begin

  oColumn := TableObject1.CreateNewColumn(nCol);
  oColumn.Tag := 10;
  oColumn.Width := GetWidth(oColumn.Tag)*fr1cm;
  oCell := TableObject1.Cells(nCol, 0);
  oCell.Memo.Text := 'Number';
  oCell.Font.Style := fsBold;
  oCell.HAlign := haCenter;
  oCell.VAlign := vaCenter;

  oColumn := TableObject2.CreateNewColumn(nCol);
  oColumn.Tag := 10;
  oColumn.Width := GetWidth(oColumn.Tag)*fr1cm;
  oCell := TableObject2.Cells(nCol, 0);
  oCell.Memo.Text := '[FDQuery1."DOCNUMBER"]';
  oCell.HAlign := haCenter;
  oCell.VAlign := vaCenter;

  Inc(nCol);
end; { if }

if CheckBox3.Checked then
begin

  oColumn := TableObject1.CreateNewColumn(nCol);
  oColumn.Tag := 10;
  oColumn.Width := GetWidth(oColumn.Tag)*fr1cm;
  oCell := TableObject1.Cells(nCol, 0);
  oCell.Memo.Text := 'Date';
  oCell.Font.Style := fsBold;
  oCell.HAlign := haCenter;
  oCell.VAlign := vaCenter;

  oColumn := TableObject2.CreateNewColumn(nCol);
  oColumn.Tag := 10;
  oColumn.Width := GetWidth(oColumn.Tag)*fr1cm;
  oCell := TableObject2.Cells(nCol, 0);
  oCell.Memo.Text := '[FDQuery1."DOCDATE"]';
  oCell.HAlign := haCenter;
  oCell.VAlign := vaCenter;

  Inc(nCol);

end; { if }

if CheckBox4.Checked then
begin

  oColumn := TableObject1.CreateNewColumn(nCol);
  oColumn.Tag := 20;
  oColumn.Width := GetWidth(oColumn.Tag)*fr1cm;
  oCell := TableObject1.Cells(nCol, 0);
  oCell.Memo.Text := 'Correspondent';
  oCell.Font.Style := fsBold;
  oCell.HAlign := haCenter;
  oCell.VAlign := vaCenter;

  oColumn := TableObject2.CreateNewColumn(nCol);
  oColumn.Tag := 10;
  oColumn.Width := GetWidth(oColumn.Tag)*fr1cm;
  oCell := TableObject2.Cells(nCol, 0);
  oCell.Memo.Text := '[FDQuery1."CORRESPONDENT"]';
  oCell.HAlign := haCenter;
  oCell.VAlign := vaCenter;

Inc(nCol);

end; { if }
```

```
  if CheckBox5.Checked then
  begin

    oColumn := TableObject1.CreateNewColumn(nCol);
    oColumn.Tag := 40;
    oColumn.Width := GetWidth(oColumn.Tag)*fr1cm;
    oCell := TableObject1.Cells(nCol, 0);
    oCell.Memo.Text := 'Summary';
    oCell.Font.Style := fsBold;
    oCell.HAlign := haCenter;
    oCell.VAlign := vaCenter;

    oColumn := TableObject2.CreateNewColumn(nCol);
    oColumn.Tag := 10;
    oColumn.Width := GetWidth(oColumn.Tag)*fr1cm;
    oCell := TableObject2.Cells(nCol, 0);
    oCell.Memo.Text := '[FDQuery1."DOCCONTENT"]';
    oCell.HAlign := haCenter;
    oCell.VAlign := vaCenter;

    Inc(nCol);

  end; { if }

  if CheckBox6.Checked then
  begin

    oColumn := TableObject1.CreateNewColumn(nCol);
    oColumn.Tag := 10;
    oColumn.Width := GetWidth(oColumn.Tag)*fr1cm;
    oCell := TableObject1.Cells(nCol, 0);
    oCell.Memo.Text := 'Closing date';
    oCell.Font.Style := fsBold;
    oCell.HAlign := haCenter;
    oCell.VAlign := vaCenter;

    oColumn := TableObject2.CreateNewColumn(nCol);
    oColumn.Tag := 10;
    oColumn.Width := GetWidth(oColumn.Tag)*fr1cm;
    oCell := TableObject2.Cells(nCol, 0);
    oCell.Memo.Text := '[FDQuery1."ENDDATE"]';
    oCell.HAlign := haCenter;
    oCell.VAlign := vaCenter;

    Inc(nCol);

  end; { if }

  nWidth := 0;

  for nCou := 0 to TableObject1.ColumnCount-1 do
    nWidth := nWidth+TableObject1.Columns[nCou].Width;

  nEndMax := ((nMax*fr1cm) - nWidth)/TableObject1.ColumnCount;

  for nCou := 0 to TableObject1.ColumnCount-1 do begin
    TableObject1.Columns[nCou].Width := TableObject1.Columns[nCou].Width + nEndMax;
    TableObject2.Columns[nCou].Width := TableObject1.Columns[nCou].Width;
  end; { for }

end;

function GetWidth(nPercent :Integer) :Extended;
begin
  Result := (nMax*nPercent)/100;
end;

begin
  nMax := 19.00;
end.
```

First, in this code, we create columns for the header and for the data depending on the selected details on the dialog page, and we set the width of the columns as a percentage of the maximum width of our table. In the end, we give free space in the table to the remaining columns if the user has not selected any columns.

Here is what we get if the user has selected all columns:

Index	Number	Date	Correspondent	Summary	Date of
1	1	09/02/2029	JOHN	for signature verification	0
2	1	09/02/2029	WILL	for another signature	0

And this is how it will look like if not all columns are selected:

Number	Date	Correspondent	Summary
1	09/02/2029	JOHN	for signature verification
1	09/02/2029	WILL	for another signature verification

As you can see, the remaining columns stretched in width themselves.

In this example, all columns are created dynamically, supposedly because the user may want to hide any of them, which is why initially we do not have a single column in the table. If some details should always be in the report, then we can create their designed columns in the table, and put only dynamic details into the preprint dialog.

https://www.blaisepascalmagazine.eu/product-category/special-offer/

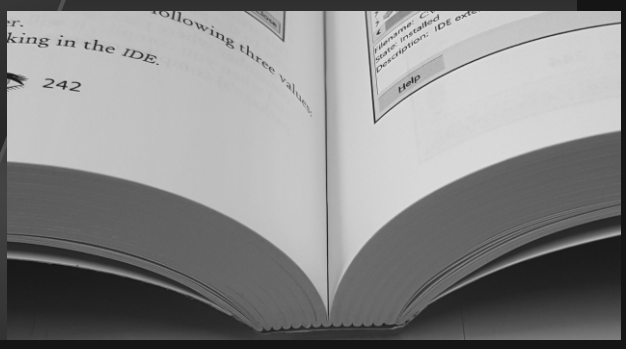
Lazarus Handbook Pocket + Subscription + PDF

- English
- Printed black & white
- 2 Volumes
- Sewn +2 ribbons
- PDF included
- 934 Pages
- Weight: 2kg
- Shipment included
- Extra protected
- Including 40 Example projects and extra programs

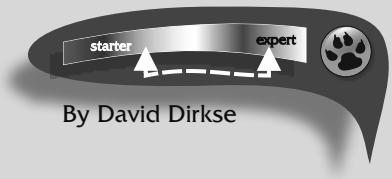
- **BlaisePascalMagazine PDF viewer included for Windows, Lazarus & Mac**

€ 90

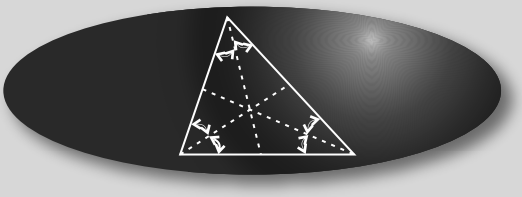
Blaise Pascal Magazine believes it is important to use natural resources in a responsible manner.
The production of this book has been done using paper that has been confirmed not to have caused forest destruction.



printed on FSC paper
https://fsc.org/en/forest-management-certification



By David Dirkse



Tested on Delphi 3/7/ Alexandria11.1 and Lazarus

ABSTRACT

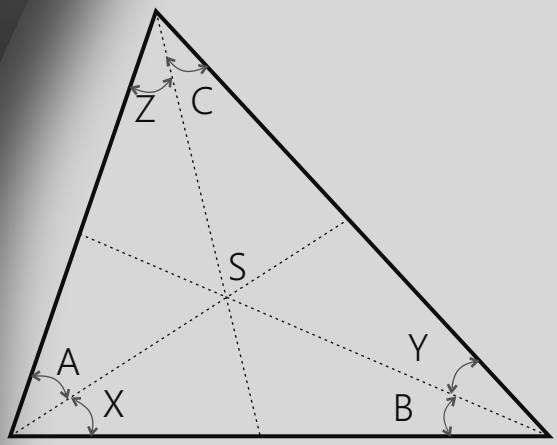
Ceva's theorem is a theorem about triangles in plane geometry.
Given a triangle ABC, let the lines AO, BO and CO be drawn from the vertices to a common point O (not on one of the sides of ABC), to meet opposite sides at D, E and F respectively.
(*The segments AD, BE, and CF are known as cevians.*)
Then, using signed lengths of segments,
The theorem is often attributed to **Giovanni Ceva**, who published it in his 1678 work *De lineis rectis*. But it was proven much earlier by Yusuf Al-Mu'taman ibn Hud, an eleventh-century king of Zaragoza. (*Saragossa*)

INTRODUCTION TO THE CEVA PROJECT

The **CEVA** project
Recently I received the following problem and request for help:
For integer angles `A,B,C,X,Y,Z` find solutions of the equation
`Sin(A).sin(B).sin(C) = sin(X).sin(Y).sin(Z)`

For A,B,C being different from X,Y,Z.
Angles are expressed in degrees ranging 1 to 89.

This equation originates from the **CEVA** lemma:
If the dotted lines in picture at the right have one point (S) in common, the equation is true.



More important is the opposite CEVA lemma:
If the equation is true, the lines through the corners of a triangle intersect at one point.

Proofs are not presented here.

The sender attached a **Pascal** program with the complaint that the process was extremely slow.
It took several minutes to reach the first solution.
No way to find all solutions within hours.
Reason was mainly that for each case all calculations were made again and again.
No results were saved and reused.

USING LOOKUP TABLES TO SAVE TIME

Delphi calculates sine (cosine, tangent) values from angles expressed in radians.
The conversion factor from degrees to radians is pi/180.
(2*pi radians make 360 degrees)
So this constant is saved:

```
const degtoRad : double = pi/180;
```

Next a table containing all sine values is generated:

```
var sinetable : array[1..89] of double; //for angles 1..89
...
procedure makesinetable;
var i : byte;
begin
 for i := 1 to 89 do sinetable[i] := sin(i*DegtoRad);
end;
```

Above procedure is called at create time, so once.
Next a table is generated that contains all sine products.
Also this table holds the three angles:

```
const maxproduct = 121484; //(89-1+3)!/((89-1)!3!)-1
type TProduct = record
          A1,A2,A3 : byte;      //angles
          v : longInt;          //sines product *1e9
        end;

var producttable : array[0..maxproduct] of TProduct;
....

procedure makeproductTable;
var i,j,k : byte;
  n : longInt;
  P : double;
begin
   n := 0;
   for i := 1 to 89 do
   for j := i to 89 do
   begin
      P := sinetable[i]*sinetable[j];
      for k := j to 89 do
      with producttable[n] do
         begin
           A1 := i; A2 := j; A3 := k;
           v := round(P*sinetable[k]*1e9);
           inc(n);
         end;
   end;
   // showmessage('n='+inttostr(n-1)); //check
end;
```

The highest entry in the producttable is 121484 (maxproduct).
This number may be obtained in two ways:

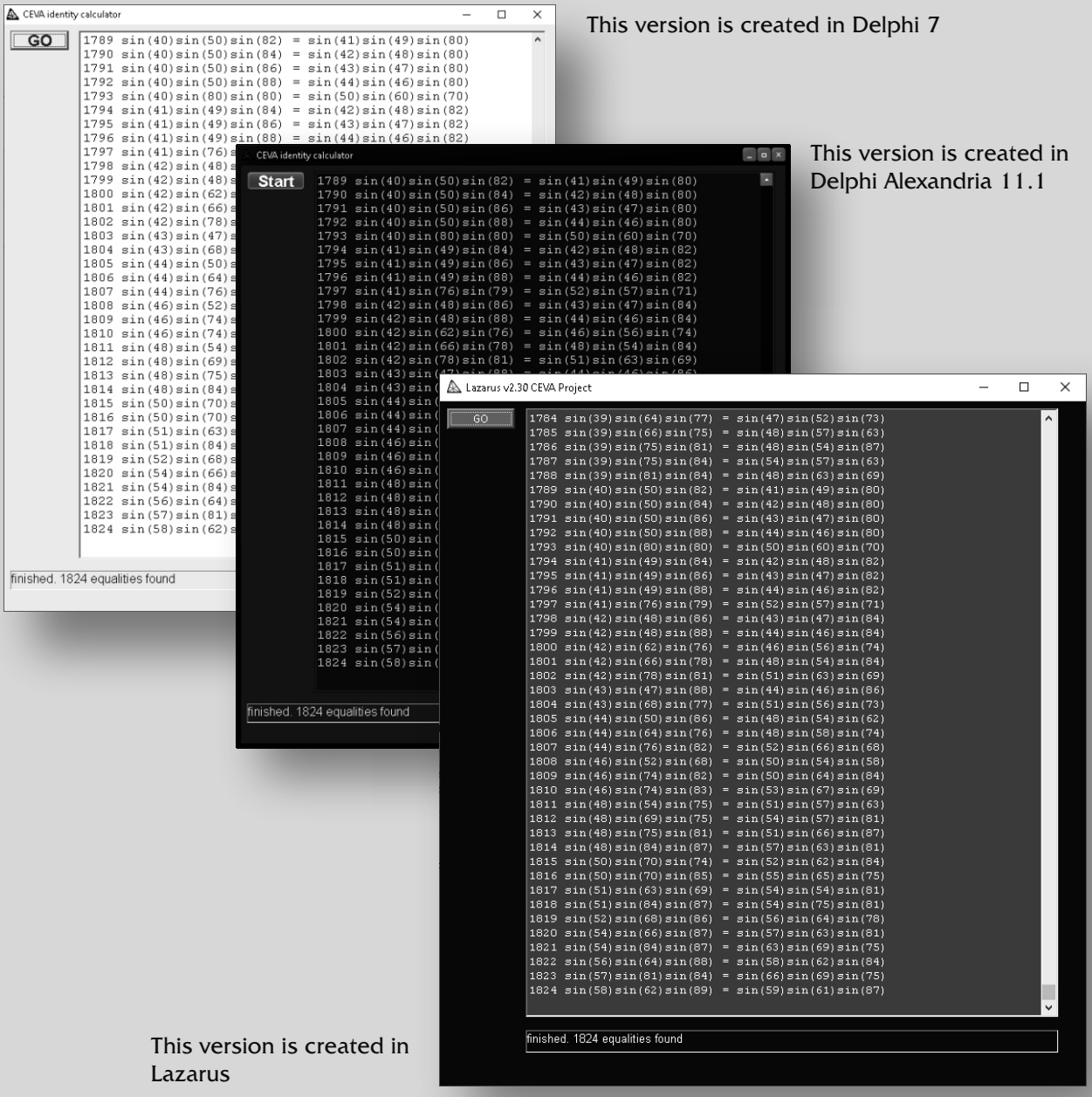
1. Empirically using the showmessage procedure (see code).
2. From the combinatorics rule `C(N,k) =(N-1+k)!/((N-1)!k!)`
   Where k choices are made from N elements and
   - Elements may be reselected
   - The order of elements is not important.

I call this number of combinations the Zoo case:
"How many zoos of k animals may be formed from N species"?
In this case k=3, N=89.

NOTE that the product of sines is stored as integer value.
Originally the sine product is a floating point value between 0 and 1.
Multiplication by 100.000.000 rounds this values at 9 digit accuracy.
This provides for a faster equality check.
The search process start by clicking the GO button.
Results are save in a Tmemo component.

```
procedure TForm1.GObtnClick(Sender: TObject);
const st ='sin(';
   sp = ')';
var i,j : longInt; count : word; A1,A2,A3 : byte;//angles
   B1,B2,B3 : byte;//angles   s : string;
begin
  memo1.Clear;
  count := 0;
  msglabel.Caption := '';
  for i := 0 to maxproduct-1 do
  begin
    A1 := producttable[i].A1;
    A2 := producttable[i].A2;
    A3 := producttable[i].A3;
    msglabel.Caption := 'searching '+ inttostr(A1);
    for j := i+1 to maxproduct do
    if producttable[i].v = producttable[j].v then
    begin
      B1 := producttable[j].A1;
      B2 := producttable[j].A2;
      B3 := producttable[j].A3;
      if (A1 <> B1) and (A1 <> B2) and (A1 <> B3) and
      (A2 <> B1) and (A2 <> B2) and (A2 <> B3) and
      (A3 <> B1) and (A3 <> B2) and (A3 <> B3) then
      begin
        inc(count);
        s := Intstring(4,count)+' ';
        s := s + st + intstring(2,A1) + sp;
        s := s + st + intstring(2,A2) + sp;
        s := s + st + intstring(2,A3) + sp + ' = ';
        s := s + st + intstring(2,B1) + sp;
        s := s + st + intstring(2,B2) + sp;
        s := s + st + intstring(2,B3) + sp;
        memo1.lines.add(s);
      end;
    end;
  end;
  s := 'finished. ';
  s := s + inttostr(count)+' equalities found';
  msglabel.caption := s;
end;
```


Some results are listed below:



This version is created in Delphi 7

This version is created in Delphi Alexandria 11.1

This version is created in Lazarus

This concludes the description.

FIN FIN

https://www.blaisepascalmagazine.eu/product-category/special-offer/

Lazarus Handbook PDF + Subscription

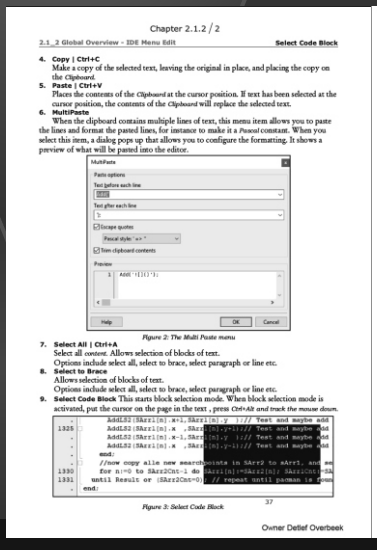
- English
- Black & white edition
- 934 Pages
- Signed with your own name
- Electronic indexed
- Index of 240 pages
- List of chapters is completely clickable for moving to the page
- Including 40 Example projects and extra programs

- **BlaisePascalMagazine PDF viewer included for Windows, Lazarus & Mac**

LAZARUS HANDBOOK
FOR PROGRAMMING WITH FREE PASCAL AND LAZARUS

€ 85

PDF File
Including electronic Index


Chapter 2.1.2 / 2

2.1_2 Global Overview - IDE Menu Edit — Select Code Block

4. **Copy | Ctrl+C**
Make a copy of the selected text, leaving the original in place, and placing the copy on the *Clipboard*.
5. **Paste | Ctrl+V**
Places the contents of the *Clipboard* at the cursor position. If text has been selected at the cursor position, the contents of the *Clipboard* will replace the selected text.
6. **MultiPaste**
When the clipboard contains multiple lines of text, this menu item allows you to paste the lines and format the pasted lines, for instance to make it a *Pascal* constant. When you select this item, a dialog pops up that allows you to configure the formatting. It shows a preview of what will be pasted into the editor.

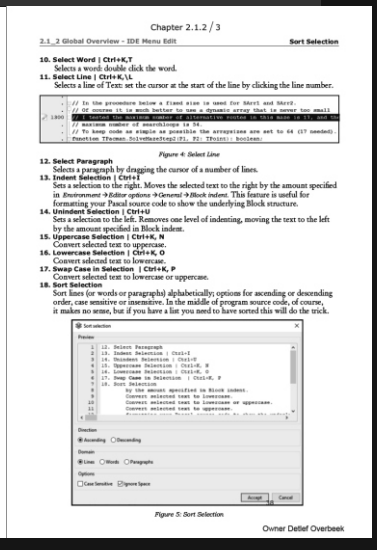
Figure 2: The Multi Paste menu

7. **Select All | Ctrl+A**
Select all content. Allows selection of blocks of text.
Options include select all, select to brace, select paragraph or line etc.
8. **Select to Brace**
Allows selection of blocks of text.
Options include select all, select to brace, select paragraph or line etc.
9. **Select Code Block** This starts block selection mode. When block selection mode is activated, put the cursor on the page in the text , press *Ctrl+Alt and track the mouse down.*

Figure 3: Select Code Block

37

Owner Detlef Overbeek


Chapter 2.1.2 / 3

2.1_2 Global Overview - IDE Menu Edit — Sort Selection

10. **Select Word | Ctrl+K,T**
Selects a word: double click the word.
11. **Select Line | Ctrl+K,\L**
Selects a line of Text: set the cursor at the start of the line by clicking the line number.

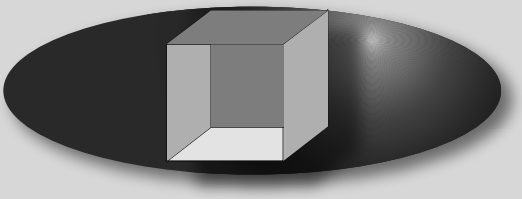
Figure 4: Select Line

12. **Select Paragraph**
Selects a paragraph by dragging the cursor of a number of lines.
13. **Indent Selection | Ctrl+I**
Sets a selection to the right. Moves the selected text to the right by the amount specified in *Environment→Editor options→General→Block indent*. This feature is useful for formatting your Pascal source code to show the underlying Block structure.
14. **Unindent Selection | Ctrl+U**
Sets a selection to the left. Removes one level of indenting, moving the text to the left by the amount specified in Block indent.
15. **Uppercase Selection | Ctrl+K, N**
Convert selected text to uppercase.
16. **Lowercase Selection | Ctrl+K, O**
Convert selected text to lowercase.
17. **Swap Case in Selection | Ctrl+K, P**
Convert selected text to lowercase or uppercase.
18. **Sort Selection**
Sort lines (or words or paragraphs) alphabetically; options for ascending or descending order, case sensitive or insensitive. In the middle of program source code, of course, it makes no sense, but if you have a list you need to have sorted this will do the trick.

Figure 5: Sort Selection

Owner Detlef Overbeek

By David Dirkse



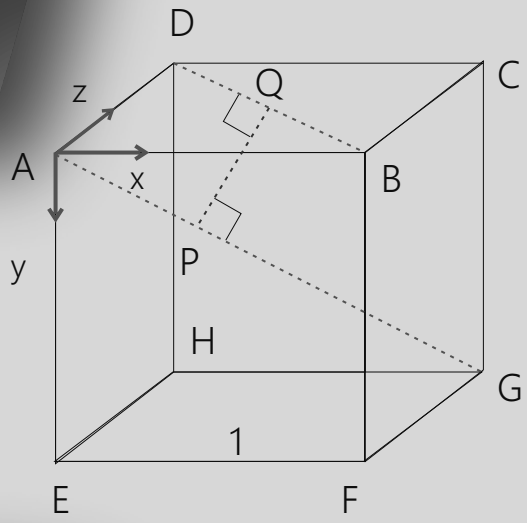
D11

Tested on Delphi 3/7/ Alexandria11.1 and Lazarus

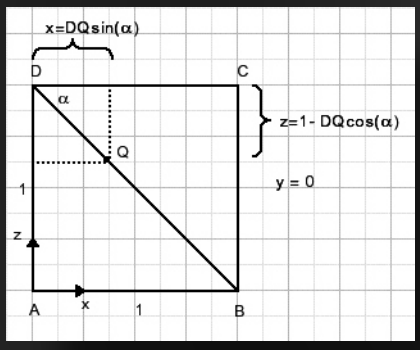
INTRODUCTION

In picture above **PQ** is the distance of diagonals **AG** and **BD**. This article at right describes two methods to find the minimal length of **PQ**.
First with a **Delphi** program by varying distances **AP** and **DQ** and second by straight calculation

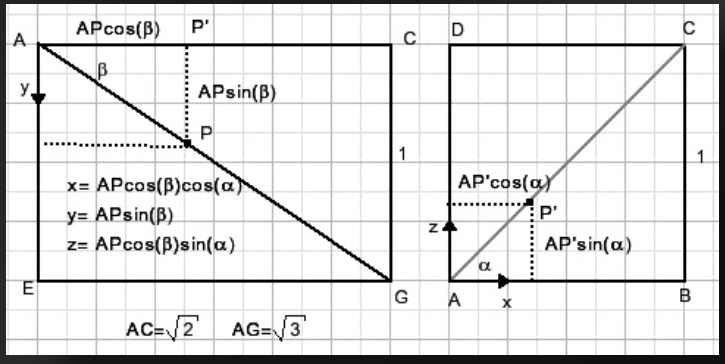
The numerical approach.
The positions of **P** on **AG** and **Q** on **BD** are defined by their **(x,y,z)** coordinates.
A is the origin (0,0,0).

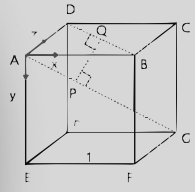


Point Q:

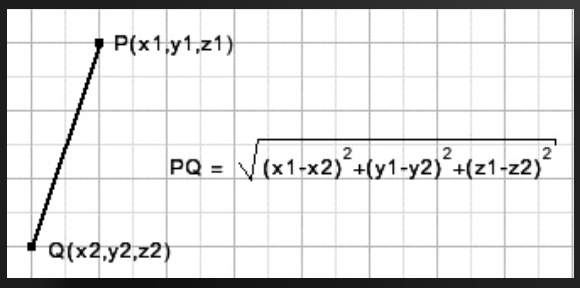


Point P:

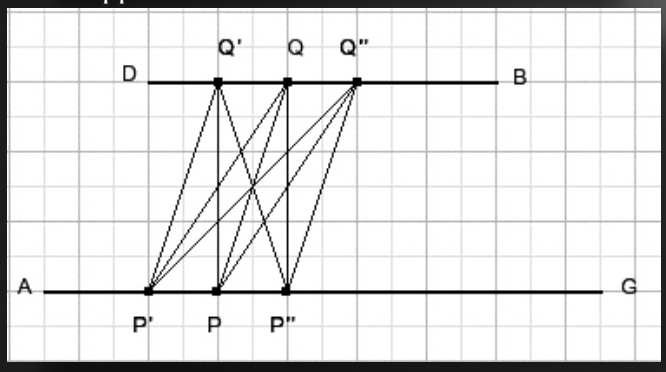




Distance PQ:



Successive approximation



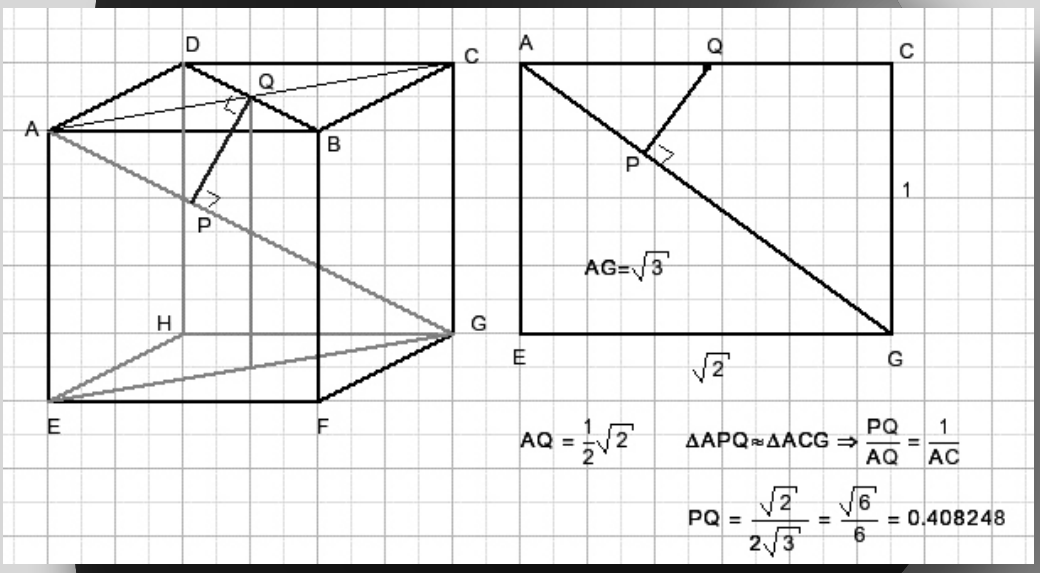
`AP'= AP-delta; AP''=AP+delta.`
`DQ'=DQ-delta; DQ''=DQ+delta.`
The initial value of `AP=0 , DQ=0, delta=0.1`
All distances `P'Q', P'Q,….P''Q''` are calculated and the smallest value is saved. If the smallest `PQ` was obtained for `P'and Q''` then `AP` is set to `AP'`and `DQ` is set to `DQ''`.
All distances are recalculated with the same value of delta.
If `PQ` was the smallest distance (so `delta=0`) then delta is reduced by 50%.
This process repeats until delta becomes very small.

PROGRAM DESCRIPTION

Pressing the GO button initializes `AP,DQ`, delta and executes a for loop `I := 1..9` to calculate the distance between `APm, DQm` which are the `AP, DQ` values +/- delta.
The smallest values are stored in `LAP, LDQ`. Ldistance.
i is stored in mi.
`PQ` is calculated at `i = 5` (no delta value added).
If `mi = 5` then `delta := delta/2`.
The repeat…until loop continues until `delta < 0.000001`
At form creation time the sine and cosine values are calculated. (*called sina, cosa, sinb, cosb*)
The approximation process is recorded in memo1 (`TMemo`).

Function `floatstring` makes a string of double value v with the decimal point aligned.
Function Intstring makes a string of integer value v of fixed length and right alignment.
The string holding the `AP,DQ, PQ` values is then added to the memo.
The `memo1` font is `courier new`, which has fixed character widths.

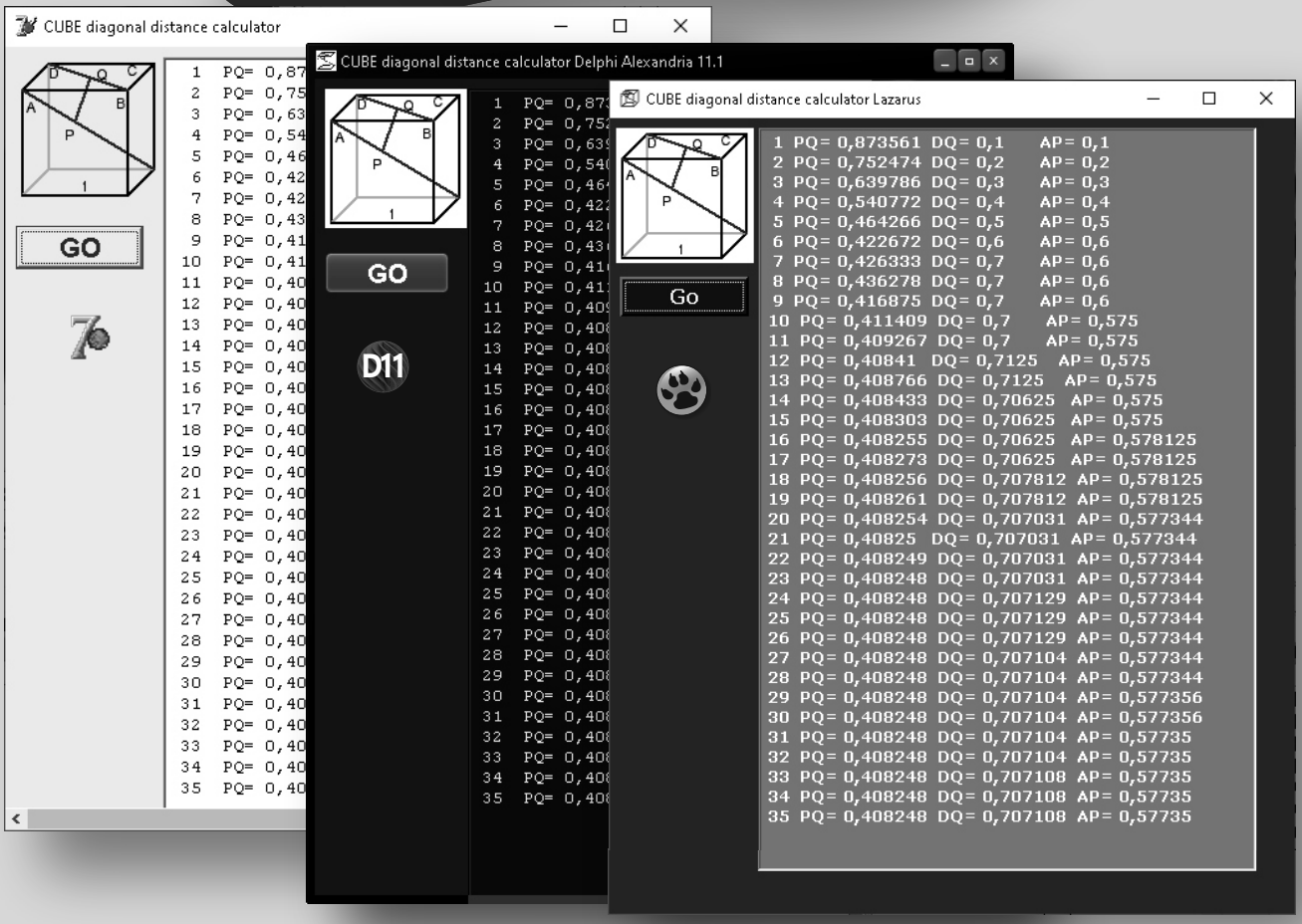
The analytical approach
Planes AEGC and DHFB are perpendicular.
So, BD is perpendicular to PQ.



$$AQ = \frac{1}{2}\sqrt{2} \qquad \Delta APQ \approx \Delta ACG \Rightarrow \frac{PQ}{AQ} = \frac{1}{AC}$$

$$PQ = \frac{\sqrt{2}}{2\sqrt{3}} = \frac{\sqrt{6}}{6} = 0.408248$$

This value is consistent with what was found by the numerical approach.



Lazarus Handbook Pocket

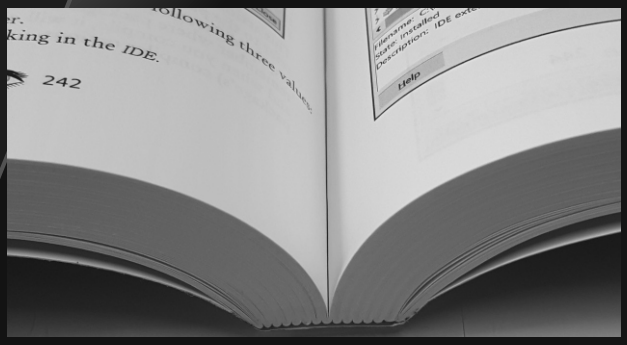
- English
- Printed black & white
- 2 Volumes
- PDF included
- 934 Pages
- Weight: 2kg
- Shipment included
- Extra protected
- Including 40 Example projects and extra programs

Blaise Pascal Magazine believes it is important to use natural resources in a responsible manner. The production of this book has been done using paper that has been confirmed not to have caused forest destruction.

€ 50

- **BlaisePascalMagazine PDF viewer included for Windows, Lazarus & Mac**

printed on FSC paper
https://fsc.org/en/forest-management-certification



Since Lazarus 2.3 and pas2js 2.2.1.

ABSTRACT

In this article we create a simple Lazarus Electron App. (*This can also be done for Delphi of Course*). Just for this goal **Mattias Gärtner** created all the necessary items for Lazarus. The installation of it all is quite easy and shown in this article. In the next Issue we will present the necessity for the Project Group

(Lazarus Factory has sponsored this project.)

INTRODUCTION

Since Lazarus 2.3 and pas2js 2.2.1 it is possible to create an **Electron Web App**.

To be able to create a simple web app we first need to install some extra programs you need to be able to create your **Electron App**:

NodeJS and **Electron.**

For **NodeJS** you can download the package:

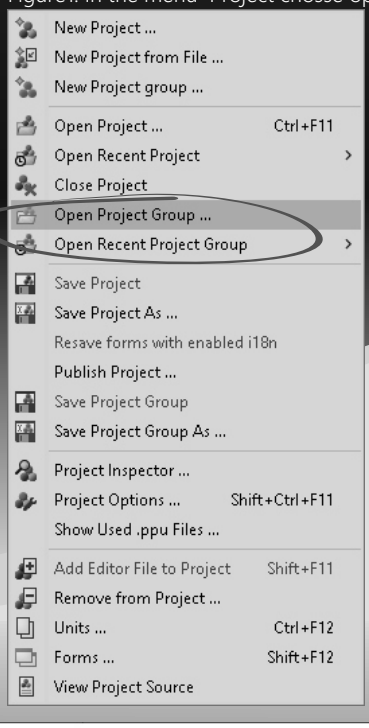
`https://nodejs.org/en/download/`

In **Issue Nr 102** (*Page 89*) was an introduction to Electron.

Mattias Gärtner has created the new ability to create an Electron Project Group (See Figure 2:)

Keep in mind: you will always need to compile each of the projects itself before you can run the final product.

Figure1: In the menu Project chosse open project group



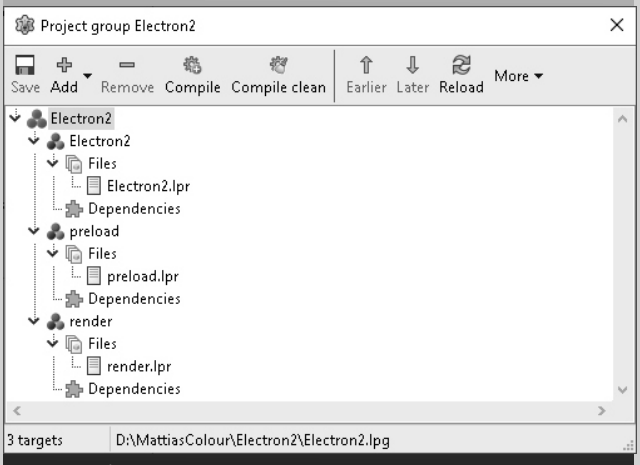


Figure2: The project group

First you need to install node.js.

Windows, MacOS:

`https://nodejs.org/en/download/`

there some easy to use installers:

`Windows Installer (.msi)32-bit or 64-bit or a Windows Binary (.zip)`

`.msi` aims towards the filename extension of Windows Installer packages.

Just double click on it to get it started.

For Debian, Ubuntu, you need to write:
sudo apt install nodejs npm
After this all you need to ckeck that “node” and “npm” works:
node -v
npm -v
it answers wit the version number

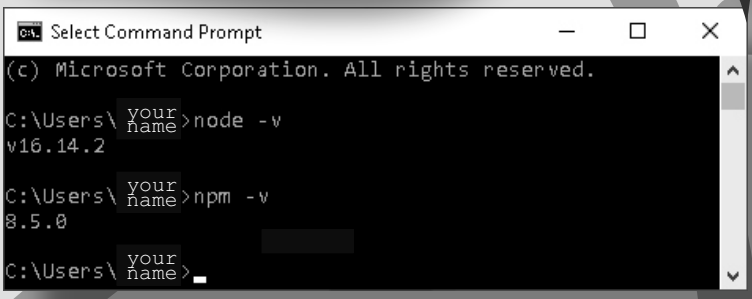


Figure 3: Test, showing the version number

Create your own (whatever or **mynodejs**) directory and install **electron** there:

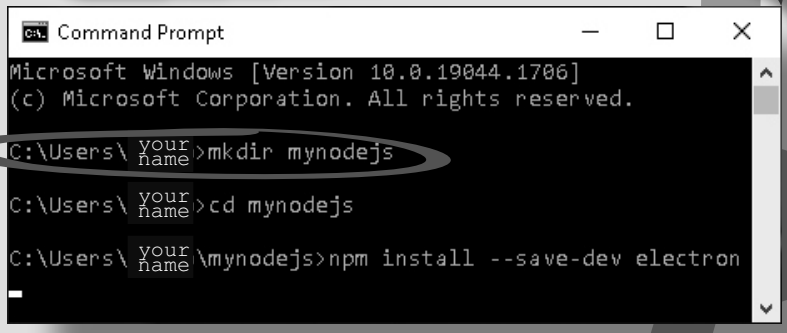


Figure 4: creating the directory

This should create a sub folder node_modules containing electron:

`C:\Whatever\mynodejs>node_modules\electron\dist\electron.exe`
To show how I handled it, here is the stucture:
`d:\ElectronLaz>node_modules\electron\dist\electron.exe`
The first part (red circle)The second part (*light blue*) is exactly the same.

Figure1: Droste effect

Set the full path to this executable in Lazarus **Tools → Options → Pas2JS →** and then **"Path of electron executable"**.

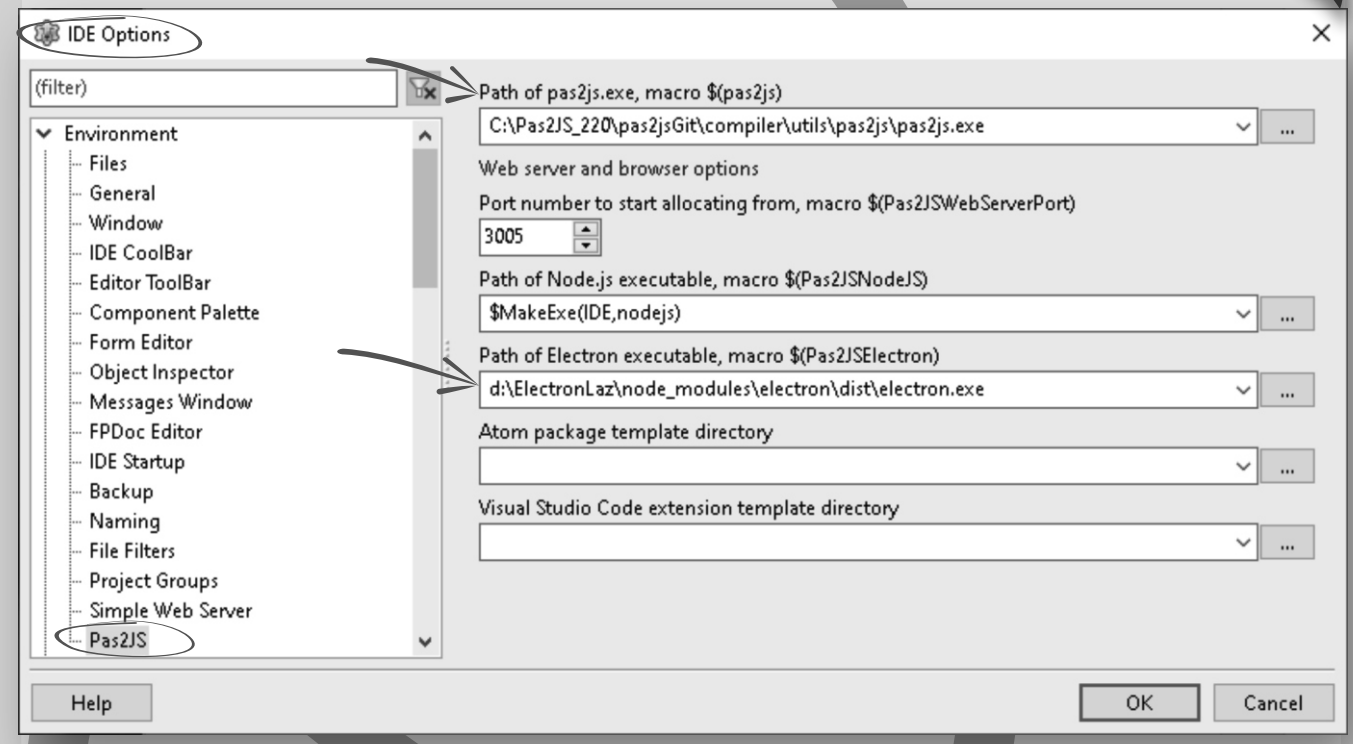


Figure 5: IDE Options

You find here the path to **pas2js.exe** and the port number and the path for **NodeJs**

LAZARUS ELECTRON WEB APPLICATION

Install pas2jsdgn package. For more information *see page 5 of this article.*
Now the next step has to be done:
you can choose from:

File → New → Project → Electron Web Application
or
Project New Project → Electron Web Application.

Eventually you can even create your own new **project group**, but that is out of scope.

Now there will be created a new project type **Electron Web Application**, which is like a **Web Browser Application**, except this wizard creates three projects, the `preload.lpi`, the `render.lpi` and `YourApp.lpi`, and a `project group`.

After the options dialog it asks for the file name of the webapp program source filename (usually a `*.lpr` or `*.pas`). Choose some name, you can later rename it. The **project group window** helps you to switch easily and compile all projects.

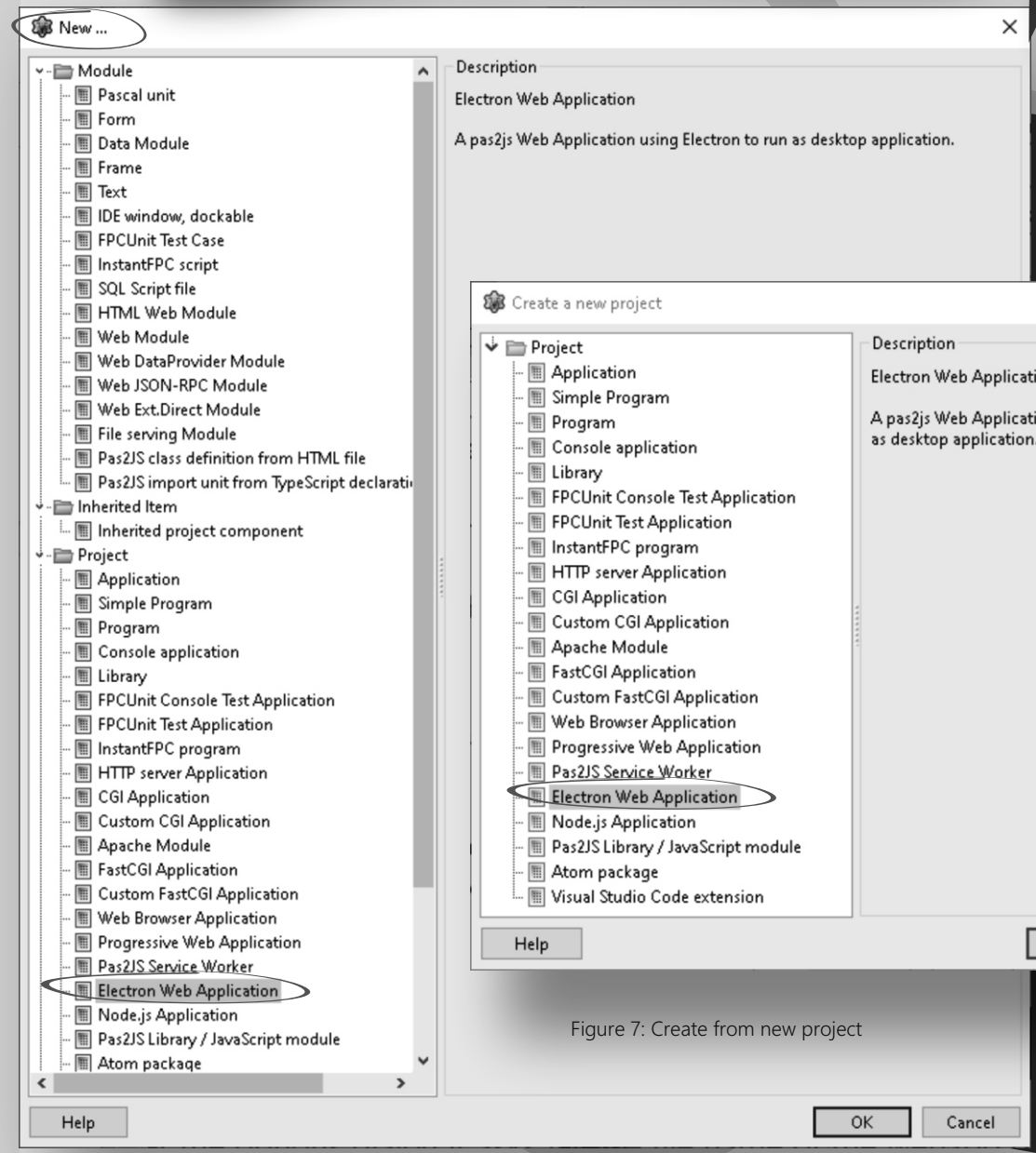


Figure 6: Select from New

Figure 7: Create from new project

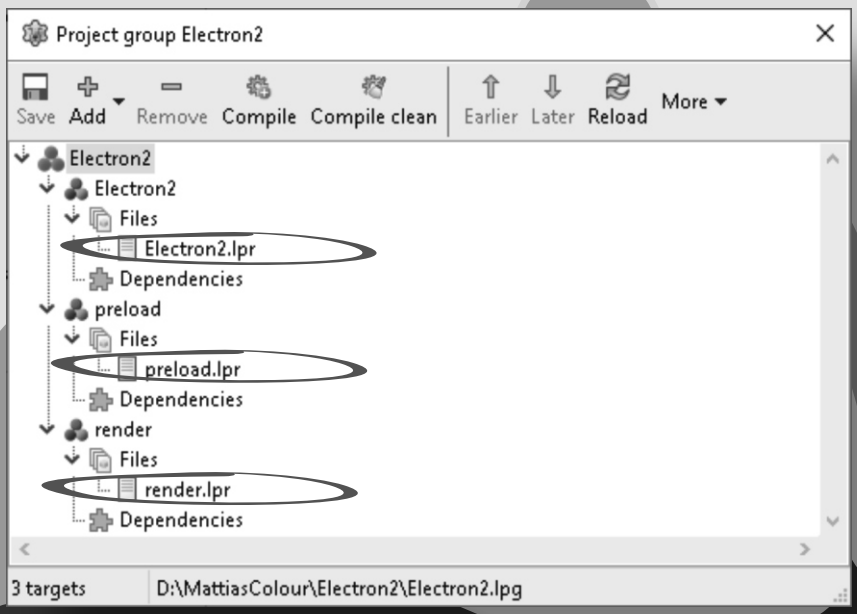


Figure 8: Overview of the Electron Group

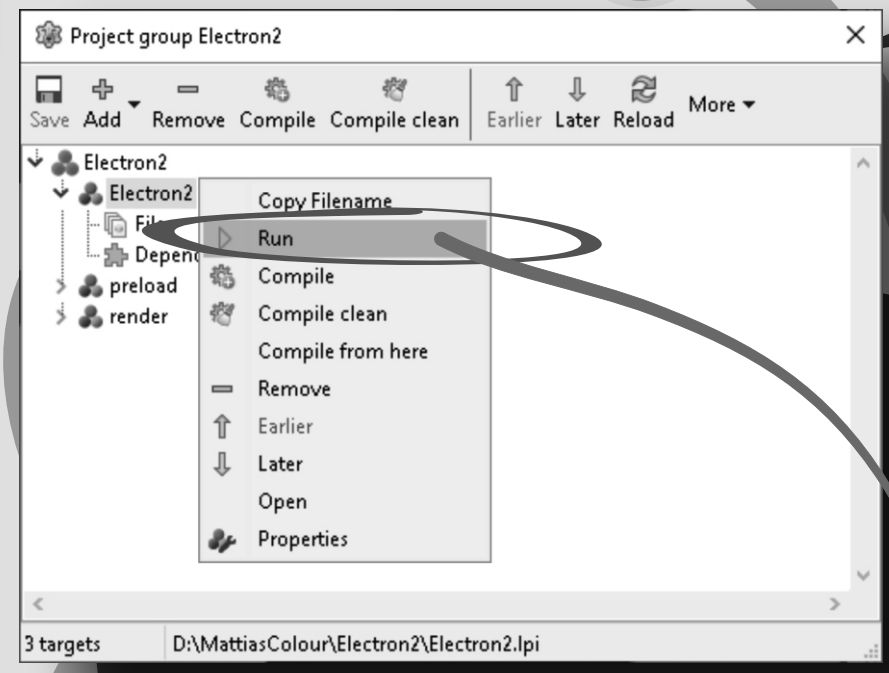
To run it you must compile all three projects to create the three needed js files.

```
program Electron2;

{$mode objfpc}

uses
  js, nodejs, node.fs, libelectron;

Procedure createWindow(event : TEvent; accessibilitySupportEnabled : boolean);
Var  opts : TBrowserWindowConstructorOptions;  win : TBrowserWindow;
begin
  opts:=TBrowserWindowConstructorOptions.new;
  opts.width:=800;
  opts.height:=600;
  opts.webPreferences:=TWebPreferences.New;
  opts.webPreferences.preload:=NJS_Path.join(__dirname,'preload.js');
  win:=Electron.TBrowserWindow.new(opts);
  win.loadFile('index.html');
end;

begin
  electron.app.addListener('ready',@CreateWindow);
end.
```

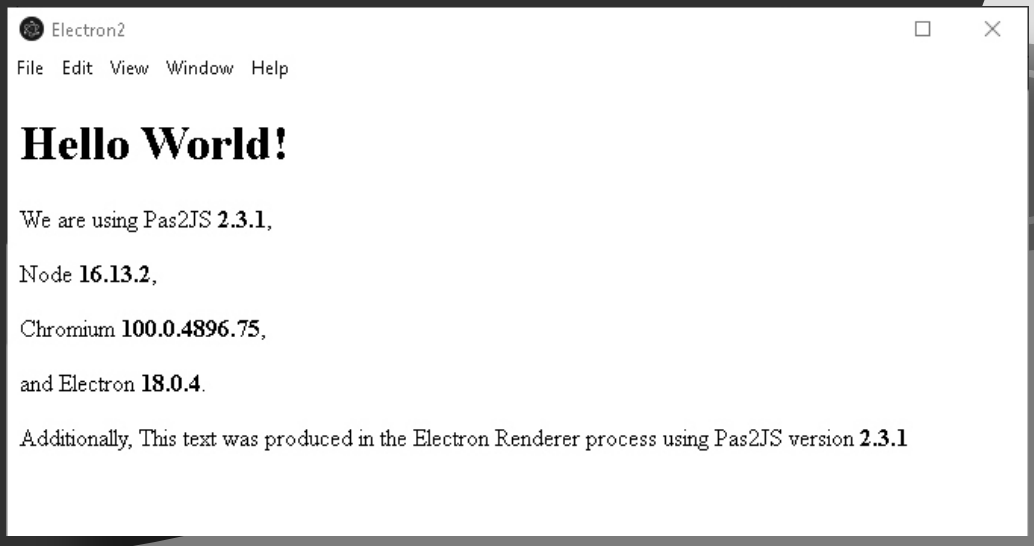
```
program preload;

{$mode objfpc}

uses
  JS, Classes, SysUtils, Web, libelectron, nodejs;

procedure DoRun(event : TJSEvent);

  procedure ReplaceText(const aSelector, aText : String);
  var el : TJSHTMLElement;
  begin
    el:=TJSHTMLElement(Document.getElementById(aSelector));
    if Assigned(el) then el.InnerText:=aText;
  end;

begin
  ReplaceText('pas2js-version', {$i %PAS2JSVERSION%});
  ReplaceText('chrome-version', String(TNJSProcess.versions['chrome']));
  ReplaceText('electron-version', String(TNJSProcess.versions['electron']));
  ReplaceText('node-version', String(TNJSProcess.versions['node']));
end;

begin
  console.log('preload environment start');
  console.log(electron);
  console.log('preload environment done');
  window.addEventListener('DOMContentLoaded', @DoRun);
end.
```

```
program render;

{$mode objfpc}

uses
  JS, Web;

var
  el : TJSHTMLElement;

begin
  el:=Document.getHTMLElementById('renderertext');
  el.innerHTML:='This text was produced in the Electron Renderer process using'
    +' Pas2JS version <b>'+{$i %PAS2JSVERSION%}+'</b>';
end.
```

To be able to run the **Electron.app** You need to „Right Click“ on Electron2 and then Run. Its easy but different to what we are used at.



And now the magic happens: here is the Electron App



INSTALLING PAS2JSDGN PACKAGE

Requires Lazarus 1.9+ and pas2js

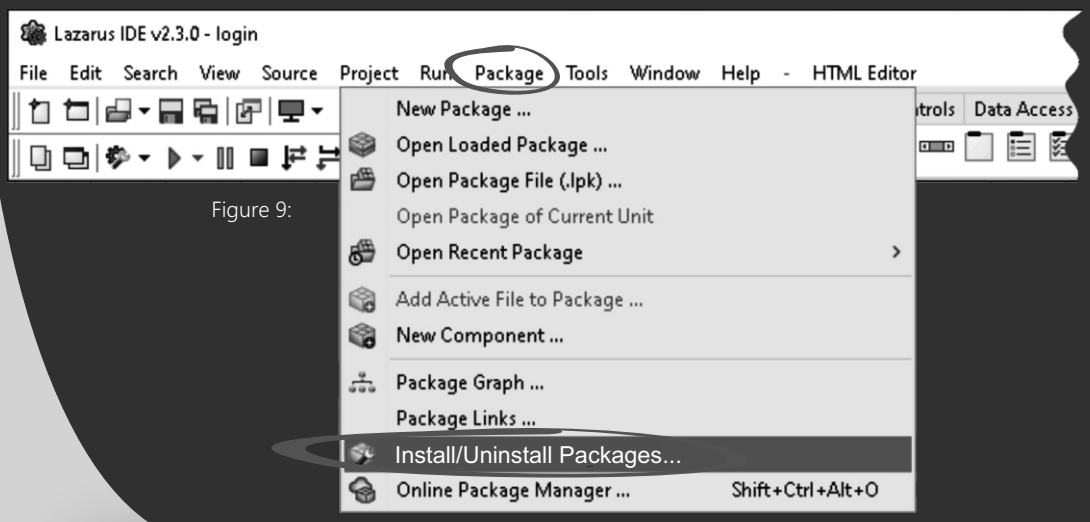


Figure 9:

❶ Click on menu Package - Install/Uninstall Packages

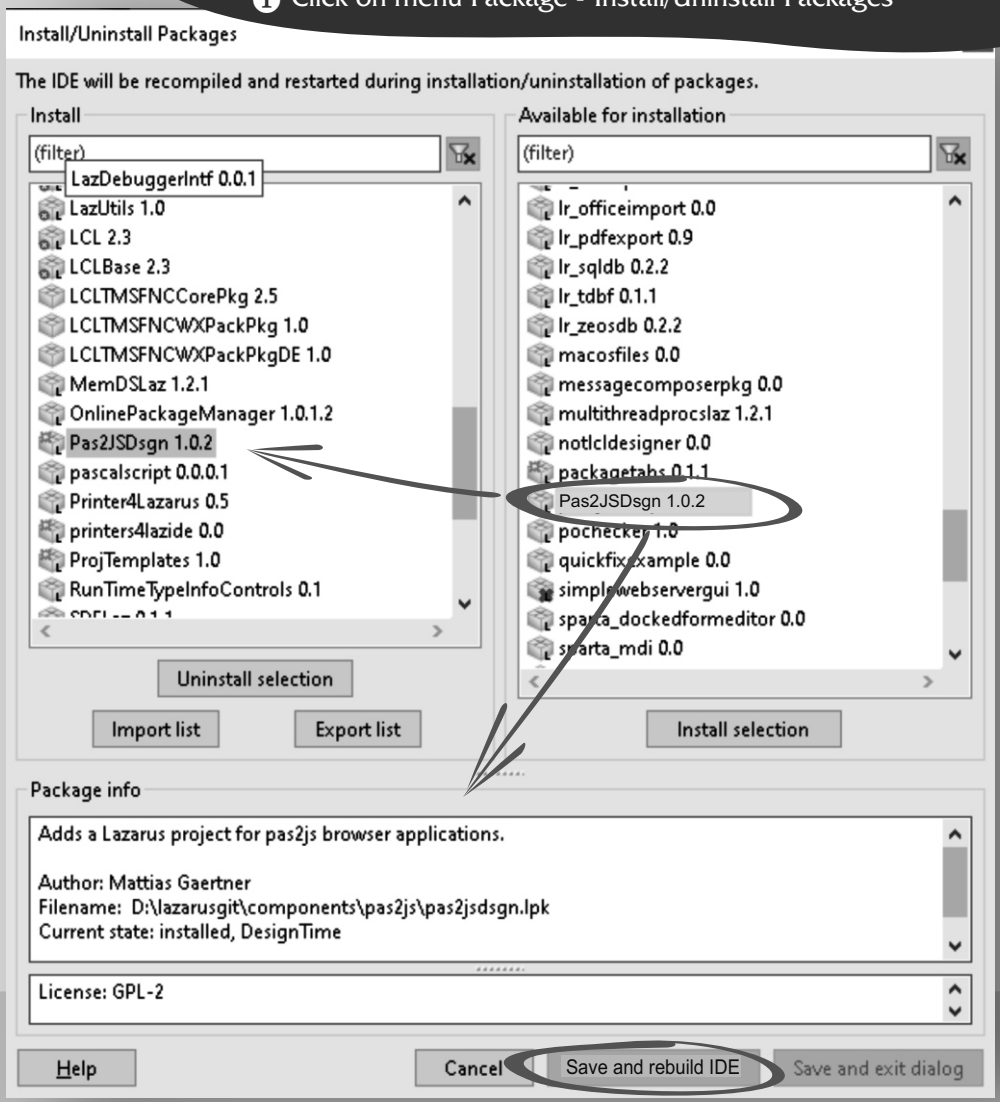
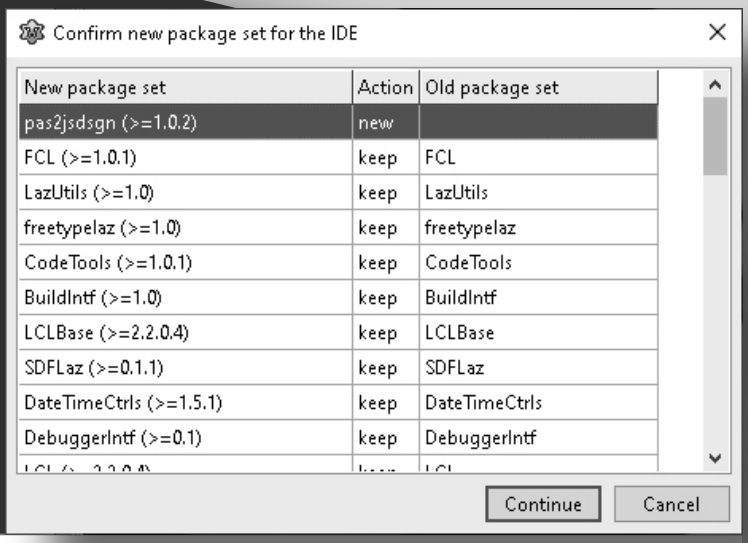
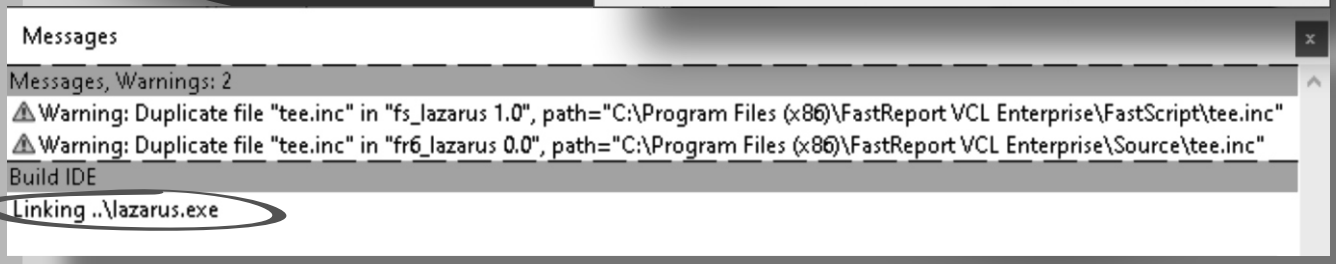


Figure 10:

❷ Choose the package **`pas2jsdsgn`** then restart the IDE.
Press **save and rebuild** *(Fig 10)*
You will be shown this extra window.

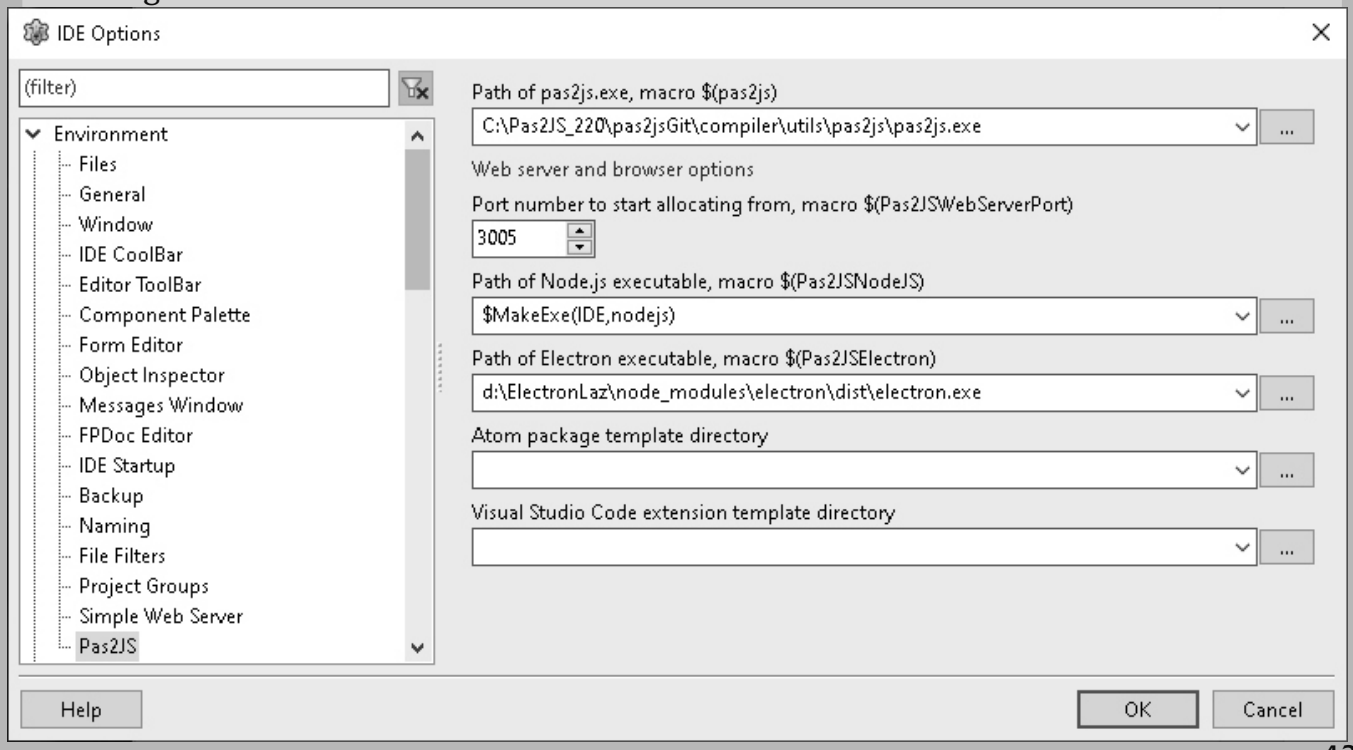
❸ Continue *(Fig 11)*
Its only a confirmation: so now you press continue and after that you will see the next window: figure 12 on the next page of this article.





GLOBAL OPTIONS

In the **`Tools/Options`**... dialog, the **Pas2JS** section allows you to set some options that affect the IDE integration. It looks like this:



These settings are available:

- **Path of pas2js**
  This is the compiler binary used when setting up a new project.
  It will be searched in the **PATH** if no absolute path is set.
  Corresponds to `IDE macro Pas2JSJS`.

  Change Path of **pas2js** to
  `C:\lazarus\pas2js\bin\win64-x86_64\pas2js.exe`
  or your own path.

- **Path of simpleserver** (**Lazarus 2.3**+ configure this in **Simple Web Server**)
  This is the webserver that is started when a project is run that needs a webserver.
  This is by default the **simpleserver** application from the **FPC** project.
  The program can be found in the FPC sources in the directory
  `fpcsrc\packages\fcl-web\examples\simpleserver` and has to be compiled manually.
  You may also use any other webserver,
  as long as it accepts the -p option to set the path,
  and it serves files from the directory in which it was started.

Change `$MakeExe(IDE,simpleserver)` to `C:\lazarus\pas2js\bin\compileserver.exe`

- **Port number to start allocating from**
  Every time you create a new webserver project,
  a new port number is allocated for the webserver.
  (*you can still edit this in the new project dialog*).

- **Browser to use when opening HTML page**
  (**Lazarus 2.3**+ configure this in **Simple Web Server**)
  The IDE will use this browser to open your HTML page.
  It will be searched in the PATH if no absolute path is set.
  Corresponds to IDE macro Pas2JSBrowser.

- **Node.js executable**
  The IDE will use this **Node.js** executable to start a **Node.js** project.
  It will be searched in the **PATH** if no absolute path is set.
  Corresponds to **IDE** macro **Pas2JSNodeJS.**

- **Electron** executable (since **Lazarus 2.3**)
  The IDE will use the electron.exe to start an Electron project.
  It will be searched in the PATH if no absolute path is set.
  Corresponds to IDE macro Pas2JSElectron.

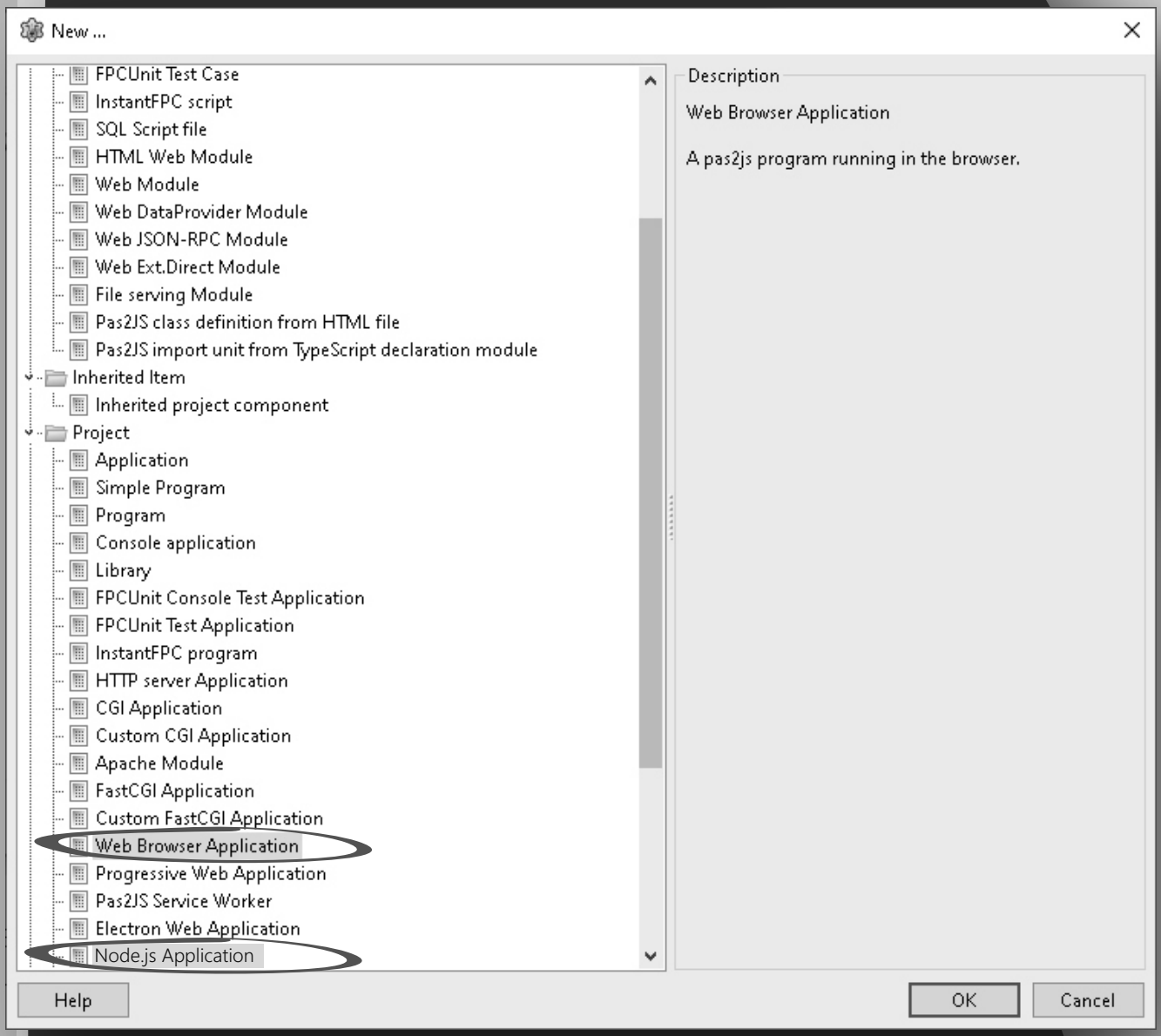
- **Atom package template directory**
  The IDE will use some files in this directory when creating an Atom package skeleton.
  There is no default for this.

- **Visual Studio Code extension template directory**
  The IDE will use some files in this directory when creating a VS Code extension skeleton.
  There is no default for this.

PROJECT WIZARDS

The **pas2js** support is in the `pas2jsdsgn.lpk` package, which you can find in the components/pas2js directory. It registers 2 wizards in the **'New project'** dialog:

Web Browser Application
Node.js Application



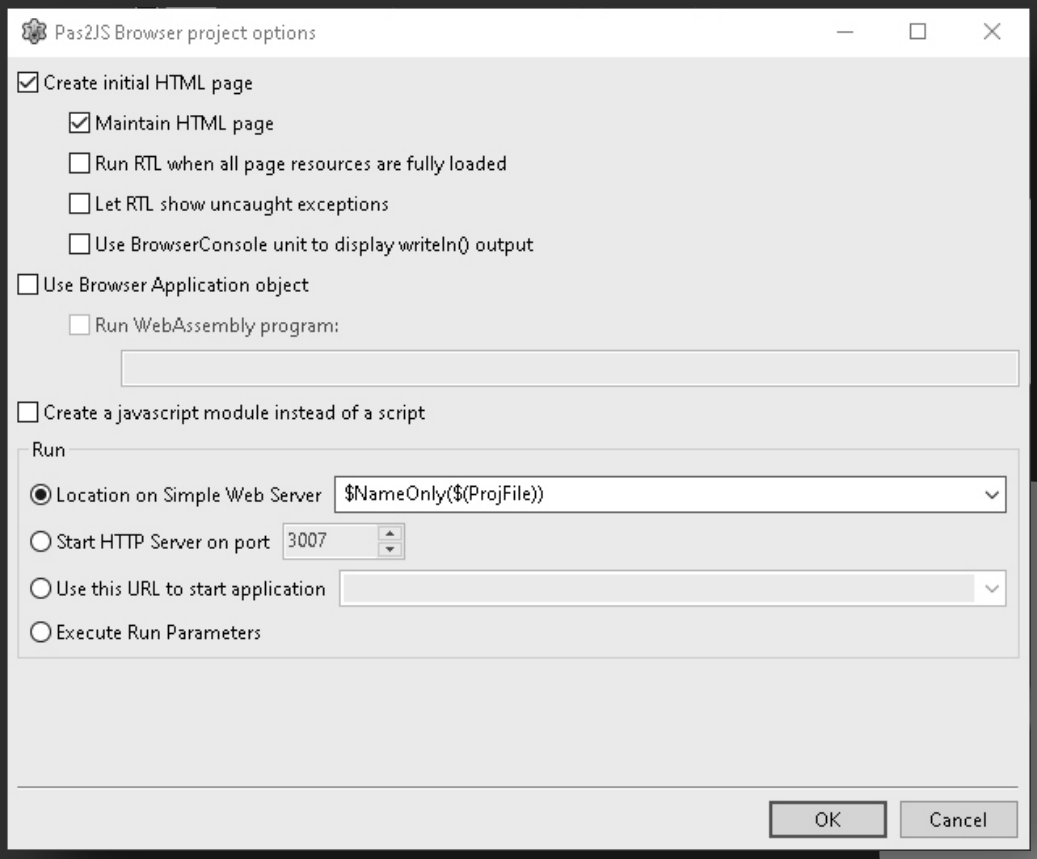
Besides creating an initial project source, both options will also:

- Set up the compiler command for compiling with **pas2js**
- change the 'Run' command so the 'Run without debugging' option works: it will open a project in the browser or run it with **nodejs.**

PROJECT WIZARDS

The **pas2js** support is in the `pas2jsdsgn.lpk` package, which you can find in the `components/pas2js` directory. It registers 2 wizards in the 'New project' dialog:

Web Browser Application
Node.js Application



THESE OPTIONS ARE AVAILABLE:

- **Create initial HTML page.**
  This is self-explanatory, the **IDE** will generate a template HTML page which includes all that is necessary to run the **pas2js** code.
  The page is included in the project description, so you can open and edit it from within the **IDE.**

  - **Maintain HTML page.**
    If you change the name of the project, the **IDE** will update the references in the **HTML** File. (*all your changes will be lost*)

  - **Run RTL**
    when all page resources are fully loaded:
    By default the script tag that starts the ball rolling will call `rtl.run()`.

THESE OPTIONS ARE AVAILABLE: CONTINUATION

- Checking this option changes the script, so the `rtl.run` is run in the HTML `document.onReady` event instead.
  This is necessary if your code contains startup code that references elements in the HTML.
  The elements will only be available after the `onReady` event.

- **Let RTL show uncaught exceptions:**
  catch exceptions and show a dialog instead of normal browser behavior, which needs the developer tools to see an uncaught exception.

- **Use Browser Console unit.**
  Checking this will simply include the browserconsole unit in the uses clause.
  This hooks into the system units writeln command: any output will be appended to the HTML. the generated HTML has a div with the correct ID to which the output is appended.

- **Use Browser Application object.**
  This changes the code to use the TBrowserApplication object.
  This is a TCustomApplication descendent which offers support for query parameters etc. as if they were command-line parameters.

  - **Run webassembly program**

- **Run**
  What to do on Run
  - **Start HTTP server on port**
    the IDE starts a browser at this port and opens a browser with the URL of the html file
  - **Use this URL to start application**
    the IDE opens a browser with this URL.
  - **Execute Run parameters**
    default Lazarus behavior

PROGRESSIVE WEB APPLICATION

Since Lazarus 2.3 and pas2js 2.2.1

The **"PWA" Progressive Web Application** is like a **Web Browser Application**, except this wizard creates two projects, the `ServiceWorker.lpi` and `YourPWA.lpi`, a project group, a manifest and some icons.
After the options dialog it asks for the file name of the webapp program source filename (*usually a* `*.lpr` *or* `*.pas`).

To run it you must compile both projects to create the two needed js files.
The project group window helps you to switch easily and compile the projects.

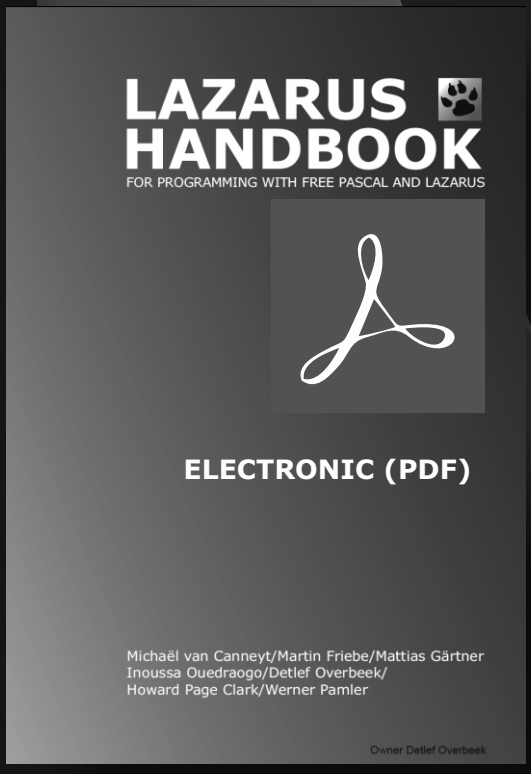
FIN FIN

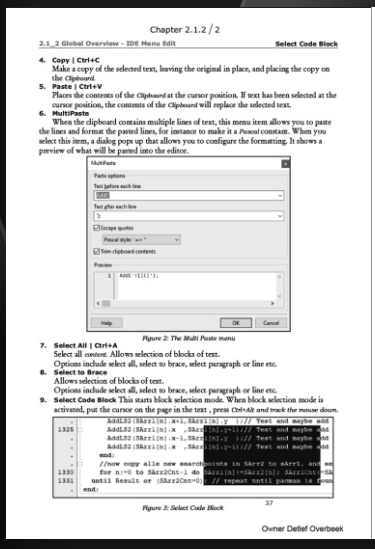
Lazarus Handbook PDF

- English
- Black & white edition
- 934 Pages
- Signed with your own name
- Electronic indexed
- Index of 240 pages
- List of chapters is completely clickable for moving to the page
- Including 40 Example projects and extra programs

- BPM PDF viewer included for Windows, Lazarus & Mac

€ 40




Chapter 2.1.2 / 2

2.1_2 Global Overview - IDE Menu Edit — Select Code Block

4. **Copy | Ctrl+C**
   Make a copy of the selected text, leaving the original in place, and placing the copy on the *Clipboard*.
5. **Paste | Ctrl+V**
   Places the contents of the *Clipboard* at the cursor position. If text has been selected at the cursor position, the contents of the *Clipboard* will replace the selected text.
6. **MultiPaste**
   When the clipboard contains multiple lines of text, this menu item allows you to paste the lines and format the pasted lines, for instance to make it a *Pascal* constant. When you select this item, a dialog pops up that allows you to configure the formatting. It shows a preview of what will be pasted into the editor.

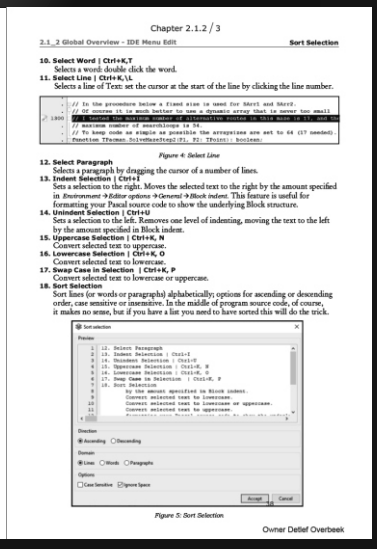
Figure 2: The Multi Paste menu

7. **Select All | Ctrl+A**
   Select all content. Allows selection of blocks of text.
   Options include select all, select to brace, select paragraph or line etc.
8. **Select to Brace**
   Allows selection of blocks of text.
   Options include select all, select to brace, select paragraph or line etc.
9. **Select Code Block** This starts block selection mode. When block selection mode is activated, put the cursor on the page in the text , press *Ctrl+A3 and track the mouse down.*

Figure 3: Select Code Block

37

Owner Detlef Overbeek


Chapter 2.1.2 / 3

2.1_2 Global Overview - IDE Menu Edit — Sort Selection

10. **Select Word | Ctrl+K,T**
    Selects a word: double click the word.
11. **Select Line | Ctrl+K,\L**
    Selects a line of Text: set the cursor at the start of the line by clicking the line number.

Figure 4: Select Line

12. **Select Paragraph**
    Selects a paragraph by dragging the cursor of a number of lines.
13. **Indent Selection | Ctrl+I**
    Sets a selection to the right. Moves the selected text to the right by the amount specified in *Environment→Editor options→General→Block indent*. This feature is useful for formatting your Pascal source code to show the underlying Block structure.
14. **Unindent Selection | Ctrl+U**
    Sets a selection to the left. Removes one level of indenting, moving the text to the left by the amount specified in Block indent.
15. **Uppercase Selection | Ctrl+K, N**
    Convert selected text to uppercase.
16. **Lowercase Selection | Ctrl+K, O**
    Convert selected text to lowercase.
17. **Swap Case in Selection | Ctrl+K, P**
    Convert selected text to lowercase or uppercase.
18. **Sort Selection**
    Sort lines (or words or paragraphs) alphabetically; options for ascending or descending order, case sensitive or insensitive. In the middle of program source code, of course, it makes no sense, but if you have a list you need to have sorted this will do the trick.

Figure 5: Sort Selection

Owner Detlef Overbeek

QUANTUM CRYPTOGRAPHY

By Detlef Overbeek

starter expert

ABSTRACT

This article is for your information.
The future is closer and closer. So now is the time.
Thanks to the development of quantum computers, it is likely that in the future **public key cryptography** will be compromised thanks to the "**algorithm of Shor**". **Shor's** algorithm is a quantum computer algorithm for finding the **prime factors of an integer.** (*Prime factorization or integer factorization of a number is breaking a number down into the set of prime numbers which multiply together to result in the original number*). Developed in 1994 by the American mathematician **Peter Shor.**

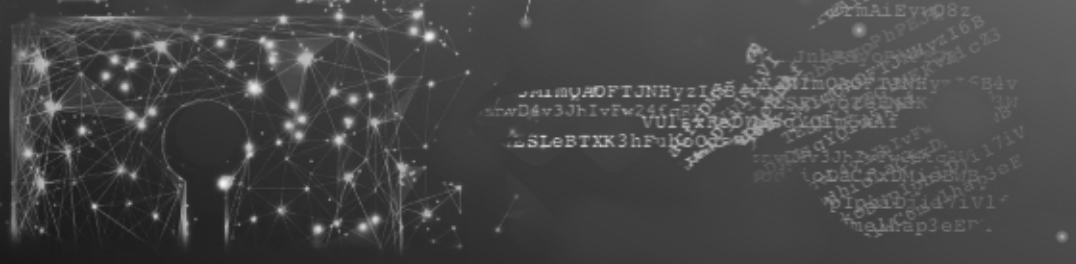
As a result, it is imperative that these crypto systems are replaced by quantum-resistant algorithms, also known as post-quantum cryptography (**PQC**) algorithms.
The **PQC** research field has grown tremendously over the past two decades, resulting in the creation of a wide variety of algorithms that are expected to be resilient to quantum attacks.
These **PQC** algorithms are selected and standardized by various standardization bodies.
But even with the guidance of these important efforts, the danger has not passed: there are billions of old and new devices that need to be migrated to the **PQC suite** of algorithms, leading to a transition process spanning decades and taking into account aspects such as security, algorithm performance, ease of secure implementation, etc.

INTRODUCTION

Even though current, publicly known, experimental quantum computers lack processing power to break any real cryptographic algorithm, many cryptographers are designing new algorithms to prepare for a time when quantum computing becomes a threat.
This work has gained greater attention from academics and industry through the **PQCrypto** conference series since 2006 and more recently by several workshops on **Quantum Safe Cryptography** hosted by the **European Telecommunications Standards Institute (ETSI)** `https://www.etsi.org/` and the **Institute for Quantum Computing** `https://uwaterloo.ca/institute-for-quantum-computing`

In contrast to the threat quantum computing poses to current public key algorithms such as used in **https or secure email**, most current symmetric cryptographic algorithms and hash functions are considered to be relatively secure against attacks by quantum computers.
While the quantum Grover's algorithm does speed up attacks against symmetric ciphers, doubling the key size can effectively block these attacks. Thus post-quantum symmetric cryptography does not need to differ significantly from current symmetric cryptography.

MANY OF TODAY'S CRYPTO SYSTEMS WILL BE BROKEN IF LARGE ENOUGH AND LOOP FAULT-TOLERANT (LFT) QUANTUM COMPUTERS ARE BUILT.



*1 FAULT-TOLERANCE

The principle of fault-tolerance is to **design specific protocols to realize the detection and correction of errors** *according to a quantum error correcting code in such a way that they can work even when their own components are noisy.*

The notion of fault tolerance is crucial since encoding quantum information in a quantum error correcting code and frequently performing error correction increases the size of the total system and the number of operations realized on it.
When considering every element to be noisy, this increase in size, a priori, increases the number of places where errors can occur. It is therefore non-trivial to design protocols which overall have a net decrease of the error rate when comparing bare, un-encoded systems, and logical encoded ones.

From : Delft University of Technology:Fault-tolerant quantum computation Theory and practice by Christophe Vuillot,
https://research.tudelft.nl/en/publications/fault-tolerant-quantum-computation-theory-and-practice

WIKIPEDIA

post quantum cryptography

(In cryptography, post-quantum cryptography (sometimes referred to as **quantum-proof, quantum-safe or quantum-resistant***) refers to cryptographic algorithms (usually public-key algorithms) that are thought to be secure against a cryptanalytic attack by a quantum computer.*

The problem with currently popular algorithms is that their security relies on one of three hard mathematical problems: the integer factorization problem, the discrete logarithm problem or the elliptic-curve discrete logarithm problem. A sufficiently powerful quantum computer running **Shor's algorithm** *will solve the integer factorization problem.*
(See page 6 of this article)

In recent decades, the field of CRYPTOGRAPHY has evolved from an obscure set of rudimentary scrambling techniques to a mature, formal science. With the help of better **cryptographic techniques**, a series of **crypto-analysis techniques** have emerged. One of these **crypto-analysis techniques** is related to quantum computers and threatens the foundations of the security guarantees that cryptography pursues.

This is a challenge in itself.
One option is a solution derived from extensive discussions within the established core practices in the information security and cryptography communities.

We present a series of actionable recommendations to organizations:
from outlining the reasons why **they should develop a robust strategy t**o start the migration to post-quantum cryptosystems now and increasing awareness and understanding of **PQC,** to an analysis of its computational resources. that these new crypto-systems will require.
Taking critical steps now will be beneficial to mitigate the future shortcomings of rushing poorly planned countermeasures down the road. The structure of this article is relevant to industry and government.

POST QUANTUM CRYPTOGRAPHY

In general terms, cryptography is the study of mathematical techniques that help determine information policy. This policy specifies who may transmit, read, and edit digital information of key organizations.

Some common uses are:
❶ security against eavesdroppers
❷ enforcing read and write access to data
❸ message verification.

Collectively, the problem is that one is dependent on the persistence of certain mathematical problems. To ensure that a cryptosystem is secure, is it necessary to demonstrate that cracking such a cryptosystem, is at least as difficult as solving a math problem, which is considered **INSOLUBLE** to anyone who has no knowledge of a piece of secret information, henceforth known as a key.

Apart from implementation errors, the hardness of this problem is the most important security guarantee of the cryptosystem, and if the hardness is disproved by a crypto analysis technique, the cryptosystem is considered defective.

THE QUANTUM THREAT TO TRADITIONAL CRYPTOGRAPHY

In **number theory**, integer factorization is the decomposition of a composite number into a product of smaller integers. If these factors are further restricted to prime numbers, the process is called **prime factorization** and the **discrete logarithm problem**.

When the numbers are sufficiently large, no efficient, non-quantum integer factorization algorithm is known. However, it has not been proven that no efficient algorithm exists. The presumed difficulty of this problem is at the heart of widely used algorithms in cryptography such as **RSA.**

Many areas of mathematics and computer science have been brought to bear on the problem, including elliptic curves,(*see page 7/16 of this article*) algebraic number theory, and quantum computing.

WIKIPEDIA

(**Discrete logarithms** *are quickly computable in a few special cases.*
However, no efficient method is known for computing them in general.
Several important algorithms in **public-key cryptography,** *such as* **ElGamal** *(In cryptography, the* **ElGamal** *encryption system is an asymmetric key encryption algorithm for public-key cryptography*)
Discrete logarithms base their security on the assumption that the discrete logarithm problem over carefully chosen groups has no efficient solution).

The security of the cryptosystem depends on the (in)solubility and the degree of difficulty.

In 1994, the mathematician **"Peter Shor"** devised a quantum algorithm: see next page.

In cryptography, the **ElGamal encryption system**
is an asymmetric key encryption algorithm for public-key cryptography which is based on the Diffie–Hellman key exchange. It was described by Taher Elgamal in 1985. **ElGamal** *encryption is used in the free* **GNU Privacy Guard software,** *recent versions of* **PGP**, *and other cryptosystems.*
The **Digital Signature Algorithm (DSA)** *is a variant of the* **ElGamal signature scheme,** *which should not be confused with* **ElGamal encryption**.

WIKIPEDIA

Shor's algorithm is a quantum computer algorithm for finding the prime factors of an integer. It was developed in **1994** by the American mathematician Peter Shor.

On a quantum computer, to factor an `integer N`, Shor's algorithm runs in polynomial time, meaning the time taken is polynomial in log N, the size of the integer given as input.

Specifically, it takes quantum gates of order `O (( log N)2( log log N)( log log logN))` using fast multiplication, thus demonstrating that the **integer factorization problem** can be efficiently solved on a quantum computer and is consequently in the complexity class **BQP bounded-error quantum polynomial time*1**
This is almost **exponentially** faster than the most efficient known classical factoring algorithm, the general number field sieve, which works in sub-exponential time:
`O(e1.9(log N)1/3(log log N)2/3.`
The efficiency of Shor's algorithm is due to the efficiency of the quantum Fourier transform, and modular exponentiation by repeated squarings. *(In quantum computing, the quantum Fourier transform (QFT) is a linear transformation on quantum bits, and is the quantum analogue of the discrete Fourier transform).*

If a quantum computer with a sufficient number of qubits could operate without succumbing to quantum noise and other quantum-decoherence phenomena, then **Shor's algorithm** could be used to break public-key cryptography schemes, such as

- The **RSA** scheme
- The **Finite Field Diffie-Hellman** key exchange
- The **Elliptic Curve Diffie-Hellman** key exchange

RSA is based on the assumption that factoring large integers is computationally intractable.
As far as is known, this assumption is valid for classical (non-quantum) computers;
no classical algorithm is known that can factor integers in „polynomial time“,
***1** Which means the time complexity is the computational complexity that describes the amount of computer time it takes to run an algorithm.

Shor's algorithm shows that factoring integers is efficient on an ideal quantum computer,
so it may be feasible to defeat RSA by constructing a large quantum computer.

It was also a powerful motivator for the design and construction of quantum computers, and for the study of new quantum-computer algorithms. It has also facilitated research on new **cryptosystems** that are secure from quantum computers, collectively called **POST-QUANTUM CRYPTOGRAPHY.(PQC)**

,

Peter Shor speaking after receiving the 2017 Dirac Medal from the ICTP.
https://www.youtube.com/watch?v=J7HeDX_7Heg&t=7075

WIKIPEDIA

A **quantum algorithm** *in quantum computation is an algorithm that runs on a realistic model for quantum computation.*
A classical (or non-quantum) algorithm consists of a finite set of instructions, or a step-by-step procedure for solving a problem, where each step or instruction can be executed on a classical computer. Also, a quantum algorithm is a step-by-step procedure, where each of the steps can be performed on a quantum computer.

A*lthough all* **classical algorithms** *can also be run on a quantum computer, the term* **quantum algorithm** *is most often used for algorithms that appear to be* **inherently quantum** *or that make use of an essential feature of quantum computation, such as* **superposition or quantum entanglement** *(See Blaise 62 page 9/ 10 for a comlpeet article about this) which promised an exponential acceleration for* **integer factorization** *and discrete logarithm discovery over* **non-quantum algorithms,** *theoretical allowing a quantum computer to crack the majority of currently used public-key cryptosystems.*
Which means; many of today's crypto systems will be broken if large enough and loop -fault-tolerant (LFT) quantum computers are built.

WIKIPEDIA

QUANTUM SUPERPOSITION
is a fundamental principle of quantum mechanics.
It states that, much like waves in classical physics, any two (or more) quantum states can be added together ("superposed") and the result will be another valid quantum state; and conversely, that every quantum state can be represented as a sum of two or more other distinct states. Mathematically, it refers to a property of solutions to the Schrödinger equation; since the Schrödinger equation is linear, any linear combination of solutions will also be a solution. An example of a physically observable manifestation of superposition is interference peaks from an electron wave in a double-slit experiment. Another example is a quantum logical qubit state, as used in quantum information processing, which is a linear superposition of the "basis states" | 0 > and | 1 >. Here | 0 > is the Dirac notation for the quantum state that will always give the result 0 when converted to classical logic by a measurement. Likewise | 1 > is the state that will always convert to 1.)

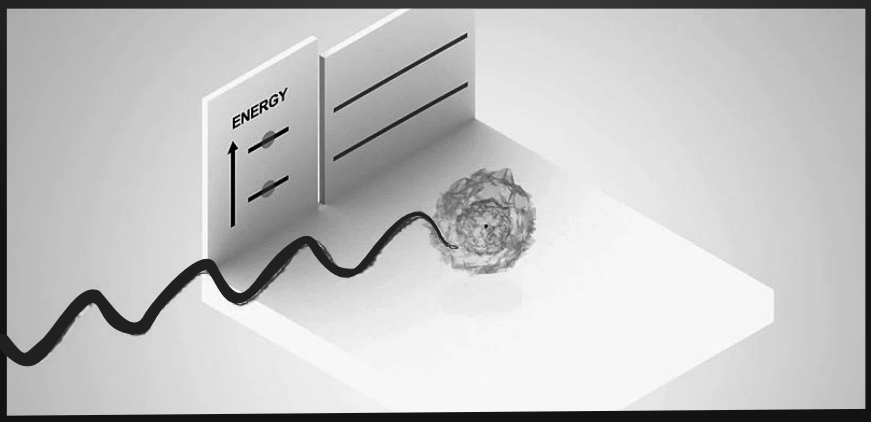


Figure 1. The green line shows the "wave" which means the superposition changing through time. The two red circles show the superpositions of the Qubit: the outer ring is active state, the inner is the inactive state. They can be active and inactive at the same time.

WIKIPEDIA

*(**ENTANGLEMENT** is a physical phenomenon that occurs when pairs or groups of particles are generated or interact in ways such that the quantum state of each particle cannot be described independently of the others, even when the particles are separated by a large distance—instead, a quantum state must be" described for the system as a whole.)*

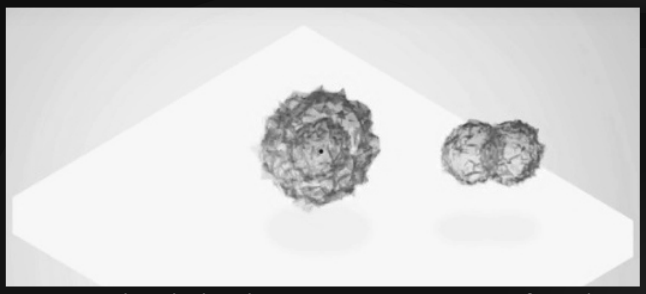


Figure 5: The Bloch sphere is a representation of a qubit, the fundamental building block of quantum computers.

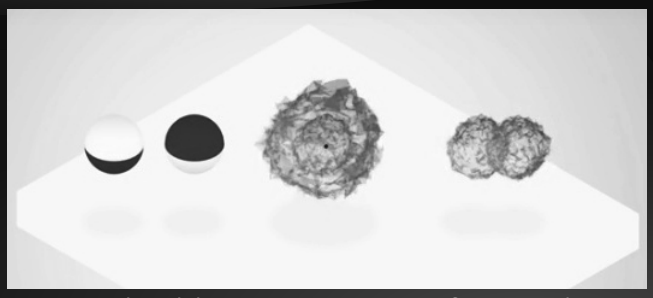


Figure 6: This ability to a superposition of state applies to any quantum particle. For example a molecule, a photon or a spin , (a small magnetic effect carried by electrons).

Each additional qubit doubles the power of a quantum computer, so when **Google AI Quantum** announced quantum supremacy in late 2019, their experiment was run on a processor of just 53 qubits. The number of perfect qubits required to break RSA-2048, with **RSA (Rivest-Shamir-Adleman)** being the cryptosystem, and 2048 the most commonly used parameter set, is estimated to be around 20 million. when using necessary quantum error correction in a loop fault tolerant system. However with 4000 perfect noise-free qubits we would also be able to break RSA-2048. "We emphasize that action is urgent, despite the technical challenges this development poses.

Consequently, new cryptographic primitives are needed to maintain the security of communications and information storage in the face of quantum threats.

While there are quantum-based cryptographic techniques that are secure against quantum computers, a substantial advantage of **PQC** over all quantum alternatives is that **PQC** schemes can be connected to any conventional communication infrastructure or contemporary devices.

PQC TRANSITION TIMELINE

This makes it possible to make a number of recommendations to organizations, about the process and timeline along which the **PQC transition** should take place.

The **so-called store-now-decrypt-later (SNDL)** attack is already an active threat. It equates to adversaries now capturing, storing, and later decrypting valuable encrypted information once **LFT** quantum computers are available. *(See page 2 of this article:Fault tolerance)*

The **SNDL** attack assumes that this information will remain valuable in the future.

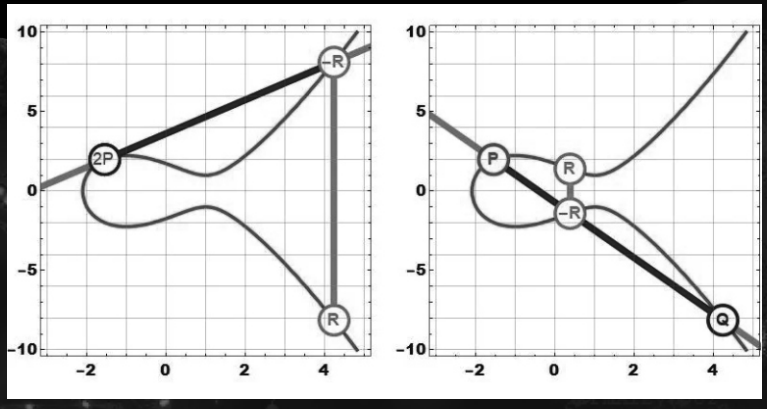
The **second quantum threat** refers to the ability to
Breaking **RSA** and **Elliptic Curve Cryptography (ECC)**,
the two most widespread public key algorithms for encrypting information today that can be broken using the **Shor algorithm**.
This could allow attackers to forge digital **RSA** and **ECC** signatures and poses a risk to systems relying on them, such as:

- **safe web browsing,**
- **zero trust architectures and**
- **cryptocurrencies.**

Elliptic-curve cryptography (ECC)
is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC allows smaller keys compared to non-EC cryptography (based on plain Galois fields) to provide equivalent security.[1]

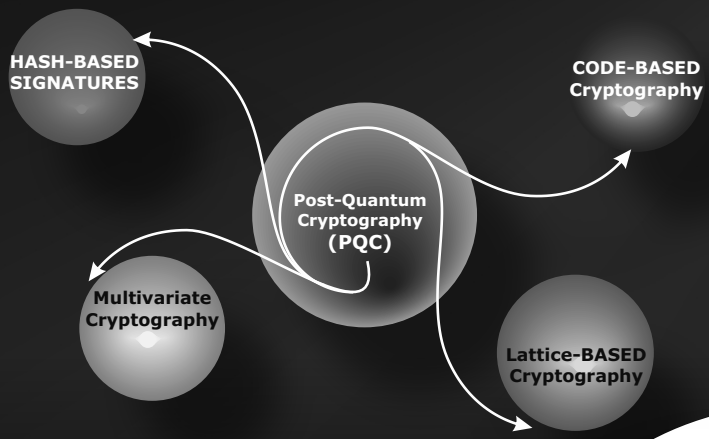
Elliptic curves are applicable for key agreement, digital signatures, pseudo-random generators and other tasks. Indirectly, they can be used for encryption by combining the key agreement with a symmetric encryption scheme. Elliptic curves are also used in several integer factorization algorithms based on elliptic curves that have applications in cryptography, such as Lenstra elliptic-curve factorization.

https://www.allaboutcircuits.com/technical-articles/elliptic-curve-cryptography-in-embedded-systems/

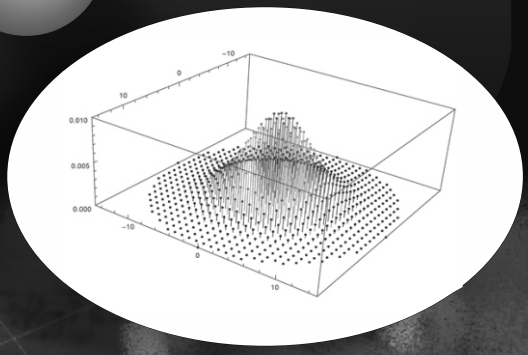


There has already been a natural convergence of standardization bodies and organizations to **PQC.**
That's why it should be recommend that organizations looking to protect their systems and users from quantum attacks use **PQC** as their primary quantum security strategy.

Organizations that have not yet started integrating **PQC** into their systems should start their efforts now. Organizations and enterprises with sensitive data with a time value greater than five years should immediately consider **PQC.** The **SNDL** attack is already executable making the risk very high. It is extremely important to create a security-first mindset.



Lattice-based cryptography *is the generic term for constructions of cryptographic primitives that involve lattices, either in the construction itself or in the security proof.*
`https://homepages.cwi.nl/~dadush/teaching/lattices-2018/`



TIMING OF MIGRATION TO PQC

Speaking of quantum attacks, it's natural to wonder when the **PQC** transition should begin. Since **LFT** quantum computers are not yet available, it is now important to start the **PQC** transition.

STORE-NOW-DECRYPT-LATER

The **SNDL** attack threatens information that is now encrypted using quantum-vulnerable cryptography. Such encrypted data, often sent over the public internet infrastructure, can be collected, stored indefinitely and decrypted in the future once the adversary has access to an **LFT quantum computer.**

There are important **trade secrets** (*and keys to them in our case*), **medical records, national security documents** etc that can be kept for several decades and must be kept confidential for a longer period of time. For that reason, the **SNDL** attack is one of the main arguments for not delaying the initiation of the transition any further.

FAR HORIZON PROJECTS

Another reason that **PQC** is of immediate importance concerns projects that are now being designed and planned, but have a long lifespan (e.g. several decades).

Vehicles are a good example of this: many cars, planes, trains and ships now in production will soon need this.

In some cases they will contain modules where one cryptosystem can be swapped for another, but this is not the case for all modules.

This is especially true when application-specific hardware is used to implement cryptography and remains immutable over the life of the product.

Critical national infrastructure projects are another example where high availability is essential and upgrading the cryptography software or hardware represents an unacceptable cost.

Since LFT quantum computers are not yet available, it is now important to start the PQC transition.

TIME IS VERY PRECIOUS

ECC was proposed as early as the 1980' s and despite being much more efficient (in space and speed, depending on parameters) than RSA13,14,15,16, it took more than two decades before it was finally widely adopted.
Hash functions are also another example of cryptographic tools that took a long time to be adopted from the start.
For example, the **NIST SHA-3** competition was announced as early as 2007, the winner was announced in 2012, and SHA-3 is still not widely accepted in 2021. Therefore, cryptography transitions (even much simpler than the **PQC** transition) usually take several years.
The **PQC** transition is more complex, as many of the approaches are relatively new and the performance of many candidates is significantly worse than current algorithms.

ACCEPTANCE PQC STANDARDS

One of the reasons for starting this transition now is the series of announcements about PQC standardization.
In 2019, **NIST** published the stateful hash-based signature standard. As a result, the cryptography community now has a clearer direction for which primitives are likely to form the backbone of the PQC suite of standardized cryptosystems.

Moving early gives time to iron out bugs, train staff and prepare appropriately for what will be a lengthy process.

RECOMMENDATIONS TO ORGANIZATIONS AND DEVELOPERS RELATING TO THE URGENCY TO ACTION

We encourage anyone interested in protecting their systems and users from quantum attacks to begin planning their **PQC** transition strategy right away.

We warn against a sense of complacency.

PQC STANDARDIZATION

These efforts are led by US-based **NIST,** the International Standards Organization (ISO), the **Internet Engineering Task Force (IETF)**, and the European Telecommunications Standards Institute **(ETSI)**.

STANDARDIZATION OF STATEFUL HASH-BASED SIGNATURES

Stateful **hash-based signatures (HBS)** are digital signature schemes whose security relies solely on the security of hash functions. This is an advantage compared to other digital signature schemes, both from a quantitative and qualitative point of view.

"HBS security" relies on the security of hash functions, while other signature schemes rely on the security of hash functions plus some other supposedly major problems.

Remember, the fewer security assumptions, the better for any cryptosystem.
From a qualitative point of view, hash functions are among the most studied topics in cryptology, meaning their security properties are well understood, including their expected resistance to quantum attacks (*assuming the correct processing sizes are used*).

The statefulness property means that the signer must maintain a state between signature generation. In practice, the state is an increasing counter. Reusing the same state would break the security of the system.

This creates implementation challenges for some applications, but not all.
For example, code signing applications seem well suited to **stateful HBS schemes**, as the signer is on the server side, and so should be able to manage the state well. There are also stateless **HBS** schemes too, but they are relatively less efficient than their statefull counterparts.

Given their optimal security features and acceptable performance metrics (usually less efficient than **RSA** and **ECC,** but not by a large margin), stateful **HBS** have already been standardized by multiple standardization bodies, being the first **PQC** standards available for widespread adoption.

The **IETF** has published stateful HBS request-for-comments **(RFCs),** which are often adopted as informal standards by industry players.

NIST is conducting two standardization efforts related to PQC.
NIST has standardized the same schemes for which the IETF published RFCs, namely the XMSS and LMS schemes, and their multitree variants.
In addition, the ISO and the IEC are also very active in this area. ISO/IEC JTC1 SC27 Work Group 2 has begun work on the first draft of the ISO PQC standard, ISO/IEC 14888-4 on stateful HBS algorithms.

THE NIST PQC PROJECT

NIST has long been active in shaping cybersecurity best practices. The widely used **AES algorithm (Advanced Encryption Standard)**24 and, more recently, the SHA-3 algorithm are examples of these initiatives. In 2016, as the quantum threat loomed, **NIST** launched a process to standardize public key **PQC** algorithms.

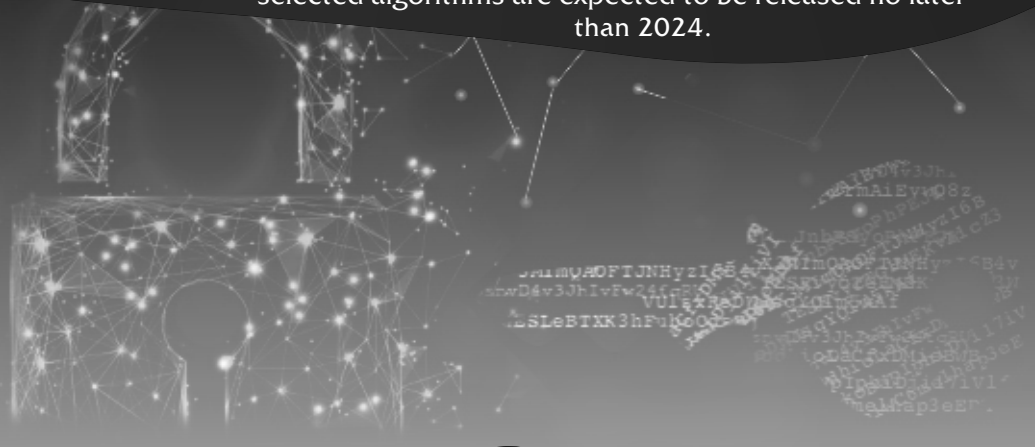
Since the Beginning of the Process **NIST** recognized that this particular process would be significantly more complex than the processes for **AES** and **SHA-3**. One reason for this is that the requirements for public key cryptography and digital signatures are more complex than symmetric cryptography. The process takes into account two cryptographic functionalities:
stateless digital signature and asymmetric encryption and key encapsulation mechanisms.
The evaluation criteria for this process are primarily focused on security,
then on practical considerations of efficiency and performance, and as a last priority they take into account other factors such as intellectual property claims and easy secure implementation.

During the standardization process, **NIST** released benchmark results to illustrate potential performance gaps between the candidates. Here we make a few observations. Isogenies
(In mathematics, in particular, in algebraic geometry, an isogeny is a morphism of algebraic groups (*a.k.a. group varieties*) that is surjective and has a finite kernel.
`https://en.wikipedia.org/wiki/Isogeny`) are extremely space-efficient with small public keys and ciphertexts, but have poor speed performance.

OTHER PQC-RELATED STANDARDIZATION EFFORTS

The **IETF,** which was responsible for drafting the transport layer security protocol **(TLS)** widely used for secure web browsing, has made several ongoing efforts to integrate post-quantum primitives in different protocols, for example in **TLS31** and Internet standard for key exchange **(IKE)32**.
The intent is to combine **RSA** and **ECC-**based and **PQC** schemes, providing a stepping stone to **PQC** without running the risk of massive vulnerabilities inherent in relatively new cryptography.

The **NIST PQC** project is nearing the end of its third round and the standards for the selected algorithms are expected to be released no later than 2024.



RECOMMENDATIONS FOR CRYPTO-AGILITY

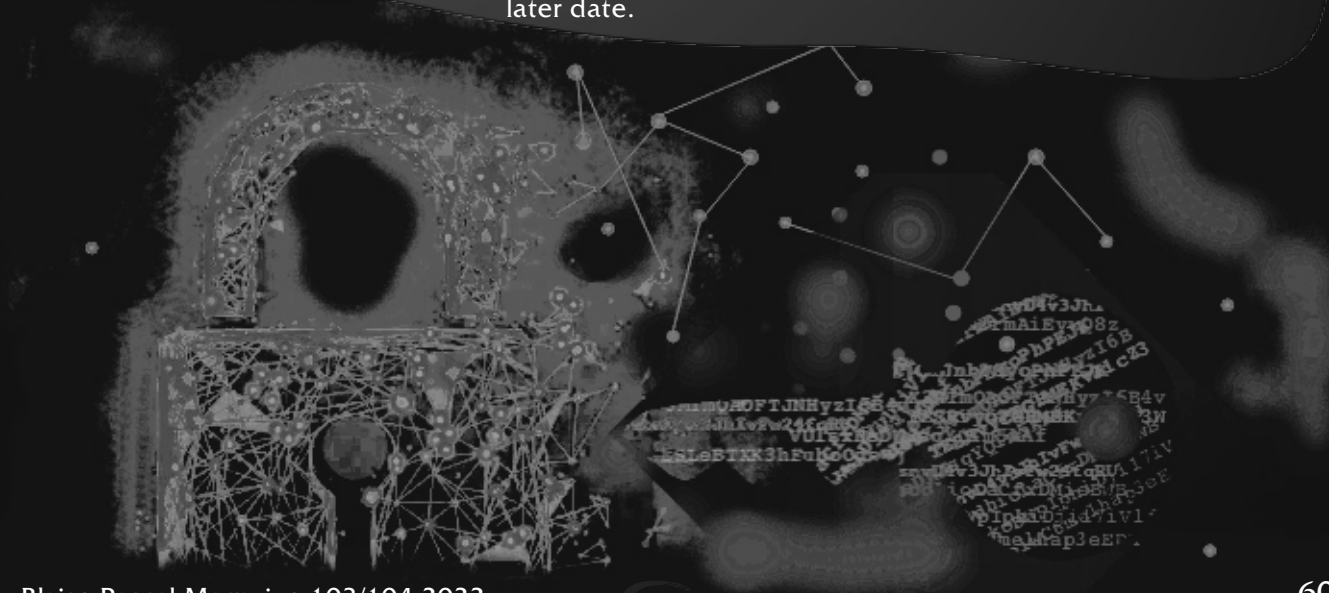
To develop a holistic (**The whole is greater than the sum of its parts**) approach to infrastructure security in the face of post-quantum migration, companies and organizations need to take steps towards crypto-agility.
This is because changing cryptographic algorithms, or layering with existing cryptographic algorithms, requires more than the direct work itself. There are also differences in key sizes, encrypted file sizes, and signature lengths. This latter set of attributes is likely to have broader implications for application and infrastructure software, protocol specifications and application program interfaces, and the standards that define them.
Adapting the infrastructure to accommodate such considerations will be a significant part of the migration to **PQC.** Moreover, despite the efforts of the new algorithm authors and reviewers, there is the potential for some degree of continuous change in algorithms, practices, or specific parameters which in turn can affect the broader system configuration.

In preparing to implement these changes, businesses and organizations should plan for crypto-agility, especially to apply layers of abstraction to centrally managed toolkits and services that minimize the effort for any subsequent changes.
Likewise, the implementation of such tooling should also include capabilities for the application or infrastructure users of such tools to handle custom data formats and sizes.

Crypto-agility should be embedded in all standards currently being developed, e.g. 6G should be inherently crypto-agile and **PQC** compliant; and industry-specific regulators in critical infrastructure sectors should urgently begin planning sector coordination to mitigate systemic risk.

PRIORITIZING STRATEGY

As for this prioritization of efforts, we must first identify the cryptographic schemes that are most at risk. In the context of the immediate need for confidentiality, key exchange algorithms are most at risk. This is because the outcome of a key exchange procedure can be captured to be broken later (**SNDL** attack).
Digital signatures, however, require an online adversary (that is, the adversary must be able to forge signatures at the time of signing). Meanwhile, some systems are difficult to update, but do not absolutely require immediate quantum-proof confidentiality, such as vehicle communications. In these cases, secure digital signatures should be a first step, which can then be used to push updates to key exchange algorithms at a later date.



HYBRID ALGORITHMS

Rather than replace existing algorithms with relatively less studied post-quantum alternatives, the scientific community came up with a simple and effective approach.
This approach consists of combining both a traditional algorithm and a post-quantum algorithm in a single mechanism.

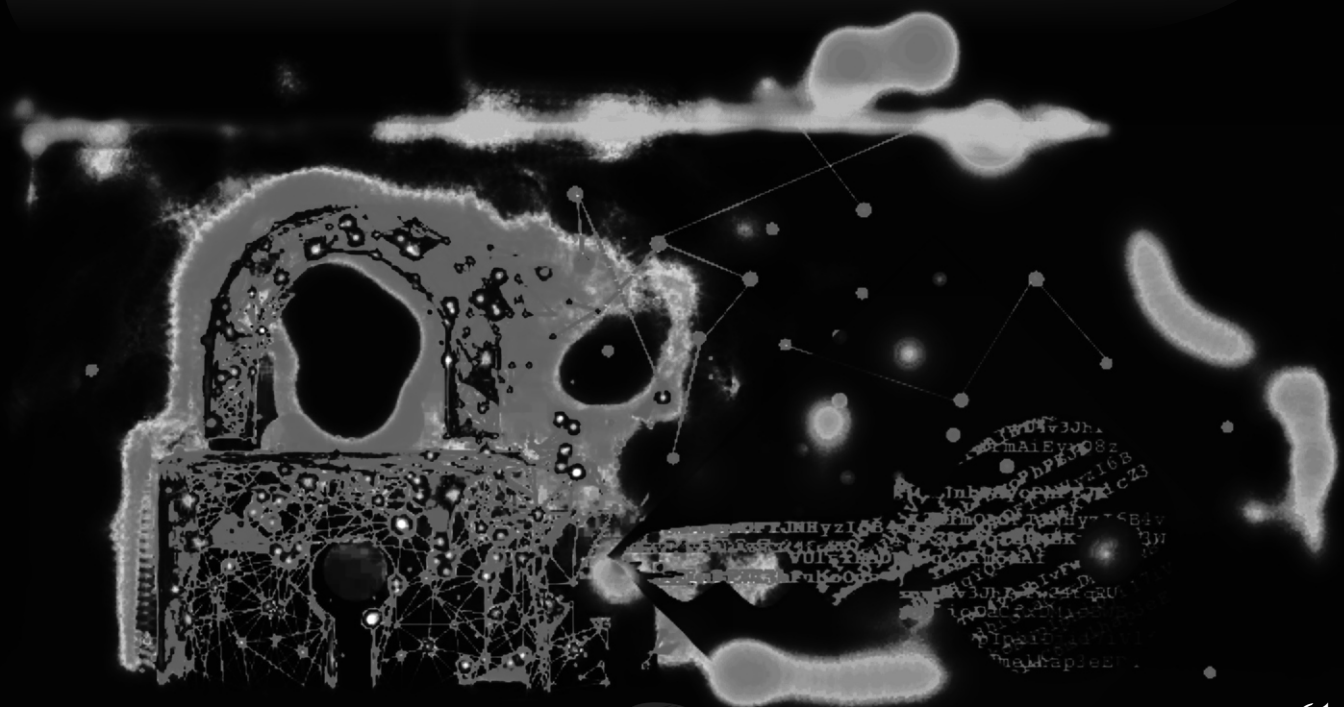
When done correctly, the security of the overall system is capped lower by the stronger of the two **cryptosystems** that make up the **hybrid system**.
In other words, even if the **PQC** algorithm is subsequently identified as flawed, the security provided by the classical scheme is still guaranteed.
In this way, security is only potentially increased in this transition, never diminished.

To combine the key exchange algorithms, one uses each algorithm (a **PQC** and a **CLASSIC)** to generate a single shared secret.

Then these two secrets can be combined to produce a single symmetric key.

In this way, an opponent who wants to attack the system will necessarily have to break through **BOTH CLASSICAL** and **PQC** schemes.

A remaining question is how to combine these two shared secrets.



PQC RESOURCES

Part of planning for the transition is educating both the underlying theory of the hard problems that security presents for these new schemes, and mastering software implementations to analyze their integration with enterprise network infrastructure.

Good theoretical resources include an **overview of lattice cryptography** to better understand primitives related to lattice cryptography, and the book **Post-Quantum Cryptography**, which gives a broader explanation of the field, more relevant to those seeking schemes in the alternate track for the third round, want to understand.
`https://link.springer.com/book/10.1007/978-3-540-88702-7`

There are a number of places where you can obtain software implementations of the **NIST** schemas.

The most comprehensive repository of these is **Liboqs46**,
from the **Open Quantum Safe project**.
Other resources include **BoringSSL47** and **Tink48**, which implement **NTRU-HRSS** and **X25519** in hybrid mode (*collectively referred to as* **CECPQ2**).

For highly optimized implementations, consider **SUPERCOP49**,
which has become the de facto benchmarking tool for **PQC** schemes.

Crypto hardware for PQC is less readily available and is an area that seems to require more research in the future.
Relevant and up-to-date material can be found at a number of conferences
with different focus areas and technical levels.
To name a few conferences on the **International Association of Cryptological Research (IACR)** calendar. There are:

Post-Quantum Cryptography,
Real World Cryptography,
Public-Key Cryptography, and
Cryptographic Hardware and **Embedded Systems.**

SUMMARY

In 2021, more than half of the experts surveyed believed that the probability of an LFT quantum computer breaking through factorization and discrete logarithm-based cryptography within 15 years was greater than 50% (ref. 50), so time is short. to switch to **PQC,** especially considering that even today's private data could be compromised by tomorrow's quantum computers as a result of the **SNDL attack,** and cryptographic hardware is now deployed that is expected to remain in the field for many decades to come.

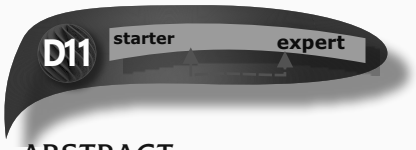
Due to the diversity of schemes, the need to immediately switch from one algorithm to another upon a successful attack, and the requirement to increase connectivity between systems, crypto-agility must be at the forefront of designing new protocols.

Work on the transition doesn't have to wait for full **NIST** standards, as hybrid cryptography allows practitioners to securely deploy quantum-proof schemes without compromising current security levels. The **U.S. government** released a memorandum on the transition to quantum-resistant cryptographic protocols in early 2022, setting a strong example for both public and private organizations worldwide, and signalling that preparatory work for the transition should begin as soon as possible.

Steps organizations can now take to mitigate the threat posed by quantum computing include: building a crypto inventory of where all public key cryptography exists within their infrastructure and products; preparing a roadmap to initiate the transition once the standards are published; experimenting with different families of **PQC algorithms,** to measure performance, explore different approaches (e.g. hybrid), assess interdependencies, and so on; and ensuring that systems are crypto-agile, that is, ready to move to **PQC** with minimal cost and time.
None of these tasks is a trivial matter, and they can all be started now, helping you responsibly work towards a more secure future for organizations, systems, and users.

This article is based on Nature **https://doi.org/10.1038/s41586-022-04623-2**
or **https://www.nature.com/articles/s41586-022-04623-2**

The datasets analyzed in the report are available from SUPERCOP at **https://bench.cr.yp.to/supercop.html.**
Source data is provided with this document.



ABSTRACT

In this article I want to make an easy entrance for all of you to create Special objects in Delphi and to show you example code of some of them and where to find it. This is not only meant for your VCL Programs. You could use it in FMX as well as on the web. It is NOT a complete overview, but enables you to create your own objects and or code. At the end of the article you will see a number of small apps that will help to find the right way to do it and to fool around a bit.

INSTALLATION

There are some special things you should read before you start running the installation. It is your choice how to run it but the easiest way is also the quickest way. I have a habit of looking at all the specifications and paths, simply to understan what is happening where etc.

You can do the installation of Skia4Delphi in 2 ways:

Figure1:

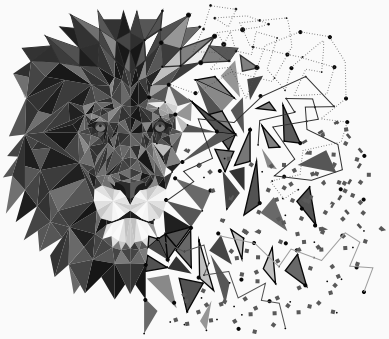


Figure2: An SVG file Editing it will let you add colors

❶ SETUP

(recommended, its the quickest and most simple)

Open your Delphi and go to The Package manager at the right top corner of your IDE.
Set the category to “All”.
Write **Skia** in the small search area. And you will see it appear. (See below).

Skia is 100 % cross platform:

Android ARM-32 (armeabi-v7a)	watchOS ARM-32 (armv7-k)
Android ARM-64 (arm64-v8a)	watchOS ARM-64 (arm64)
Linux 32-Bit (x86)	watchOS Simulator (x86)
Linux 64-Bit (x86_64)	Windows 32-Bit (x86)
Linux ARM-32 (arm)	Windows 64-Bit (x86_64)
Linux ARM-64 (arm64)	Windows ARM-64 (arm64)
macOS 64-Bit (x86_64)	Windows (UWP) 32-Bit (x86)
macOS ARM-64 (arm64)	Windows (UWP) 64-Bit (x86_64)
Mac Catalyst ARM-64 (arm64)	Windows (UWP) ARM-32 (arm)
iOS ARM-64 (arm64)	Windows (UWP) ARM-64 (arm64)
iOS Simulator (x86_64)	Tizen 32-Bit (x86)
tvOS ARM-64 (arm64)	Tizen ARM-32 (armel)
tvOS Simulator (x86_64)	Fuchsia (wasm)

Figure3: The found Package install automatically

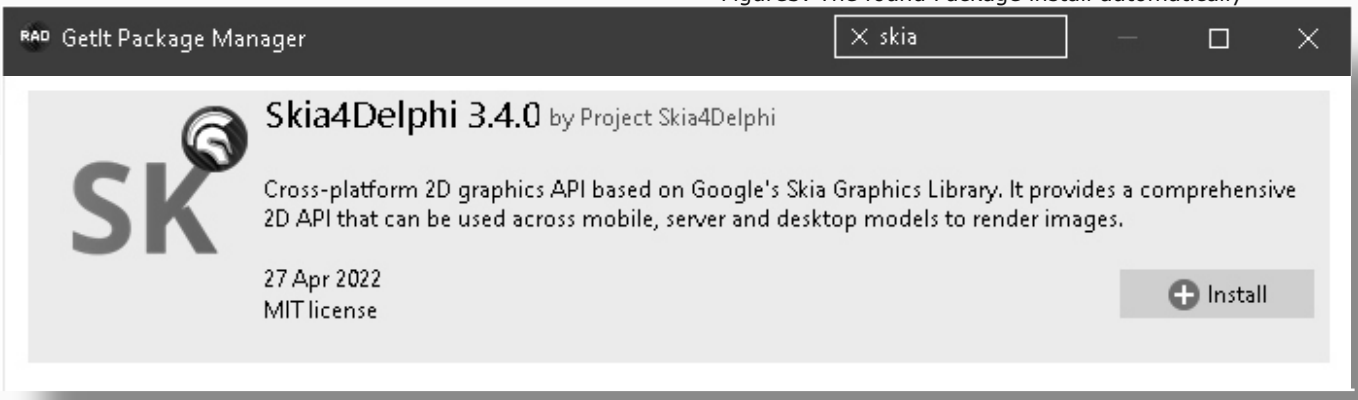


Figure 4 Running

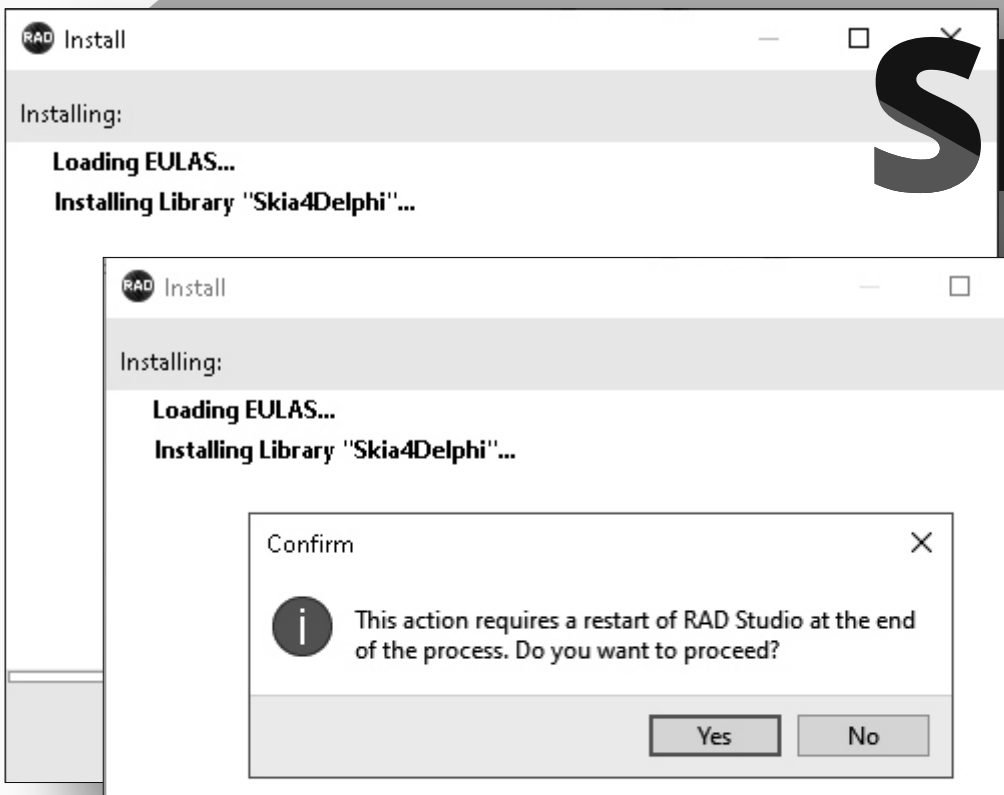


Figure 5 Running

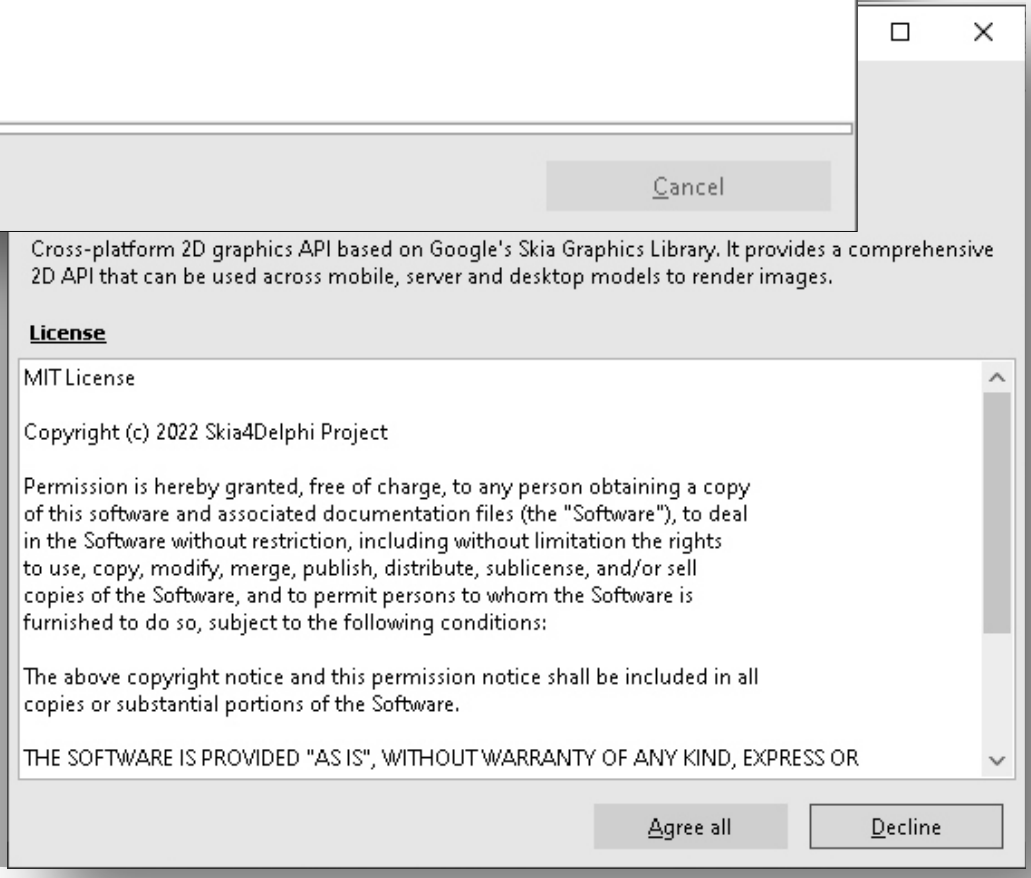


Figure 5
Mit License

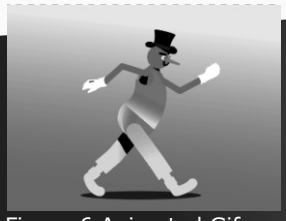


Figure 6 Animated Gif

The MIT License is a permissive free software license originating at the Massachusetts Institute of Technology (MIT) in the late 1980s.
As a permissive license, it puts only very limited restriction on reuse and has, therefore, high license compatibility.

The MIT License is compatible with many copyleft licenses, such as the **GNU General Public License (GNU GPL)**.
Any software licensed under the terms of the **MIT License** can be integrated with software licensed under the terms of the GNU GPL. Unlike copyleft software licenses, the MIT License also permits reuse within proprietary software, provided that all copies of the software or its substantial portions include a copy of the terms of the MIT License and also a copyright notice.[9][11] As of 2020, the MIT License was the most popular software license found in one analysis,[12] continuing from reports in 2015 that the MIT License was the most popular software license on GitHub.[13]

Notable projects that use the MIT License include the X Window System, Ruby on Rails, Nim, Node.js, Lua, and jQuery. Notable companies using the MIT License include Microsoft (.NET Core), Google (Angular), and Meta.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,

FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Figure 7
Loading...

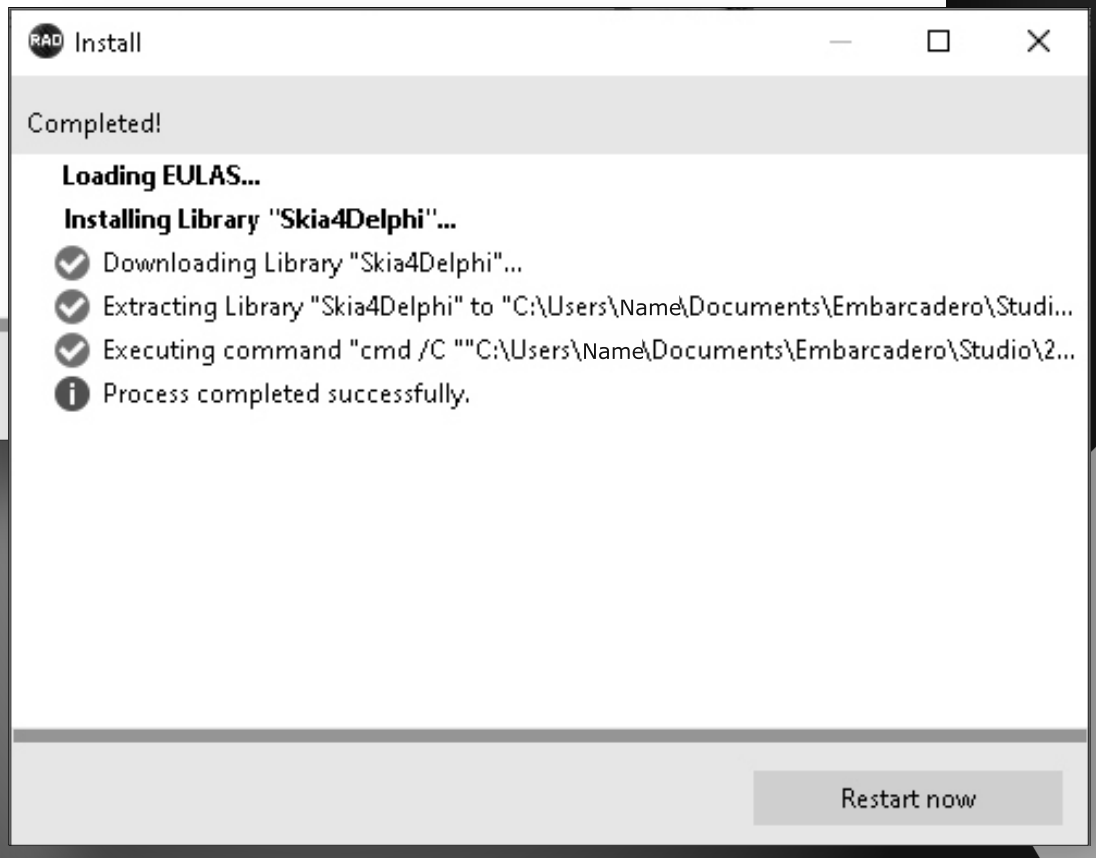
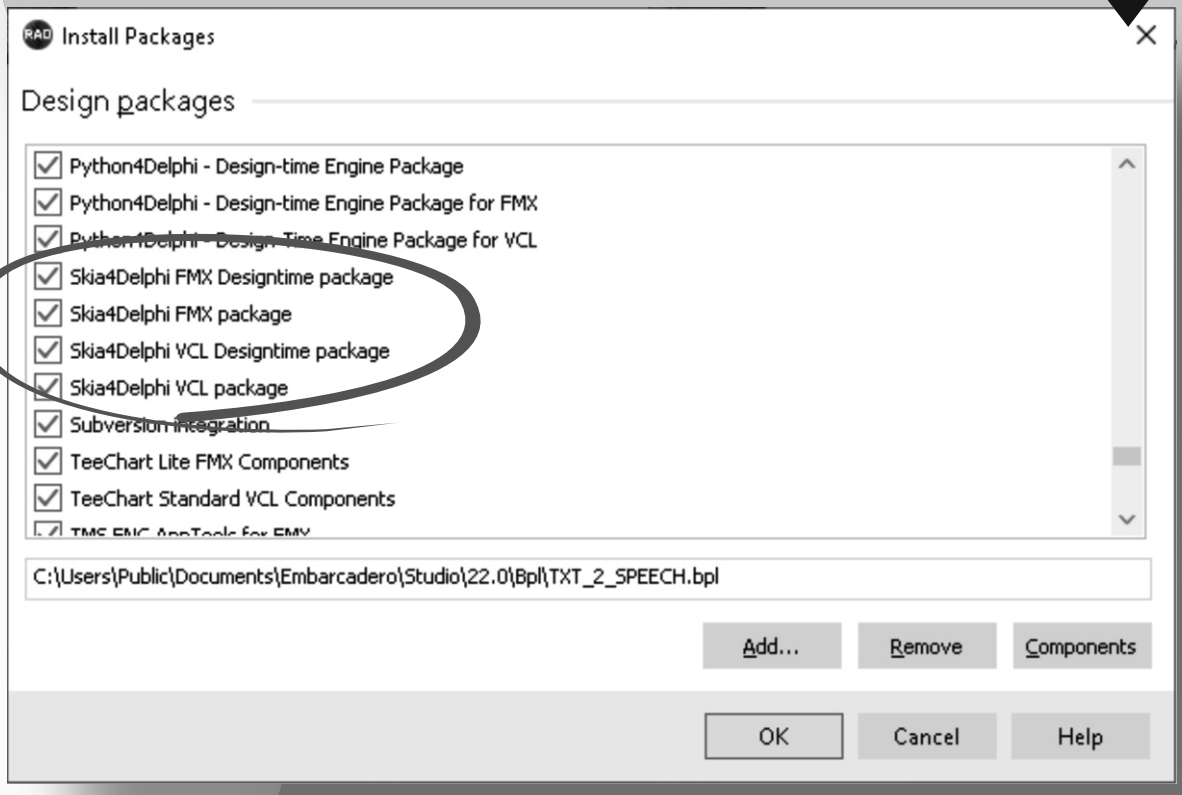


Figure 8: Completed

After restarting you can find the path in your Delphi environment where your Skia has landed:
In my case:
`c:\Users\edito\Documents\Embarcadero\Studio\22.0\CatalogRepository\Skia4Delphi-3.4.0\`

I think it is worth while to look at all the possible cheks:
If want to see a list of thecomponents installed components go to: (In Delphi)
Component → Install Packages and you will see the result (*Figure 9 on page 5 - top of this article)*. In *Figure 10* youll find the location of the demos and the complet project.
The next inspection can be done (page 6) the environment variables:
Tools → Options → Environment Variables.

Figure 9: The packages found the list is alfabetic



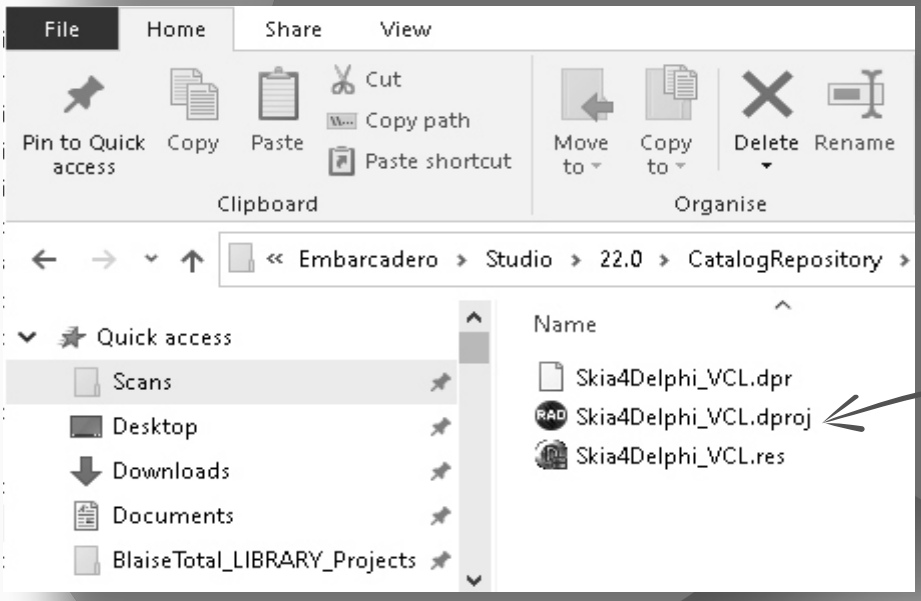
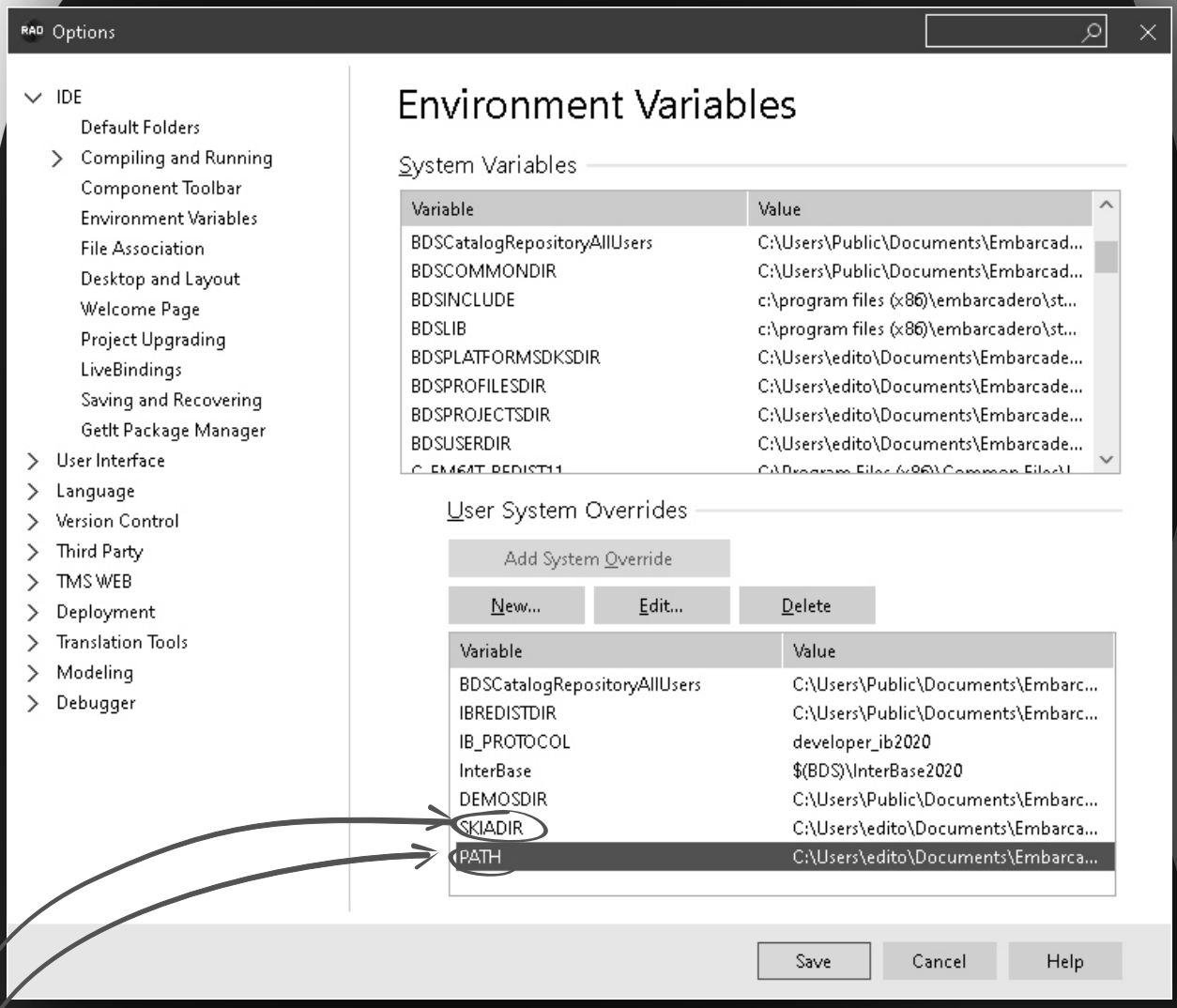


Figure 10:
The complete project address

Figure 10: The complete project address




```
C:\Users\edito\Documents\Embarcadero\Studio\22.0\CatalogRepository\
Skia4Delphi-3.4.0\Library\RAD Studio 11 Alexandria\Win32\Release\Bpl;

C:\Users\edito\Documents\Embarcadero\Studio\22.0\CatalogRepository\
Skia4Delphi-3.4.0\Binary\Win32\Release;$(PATH)
```

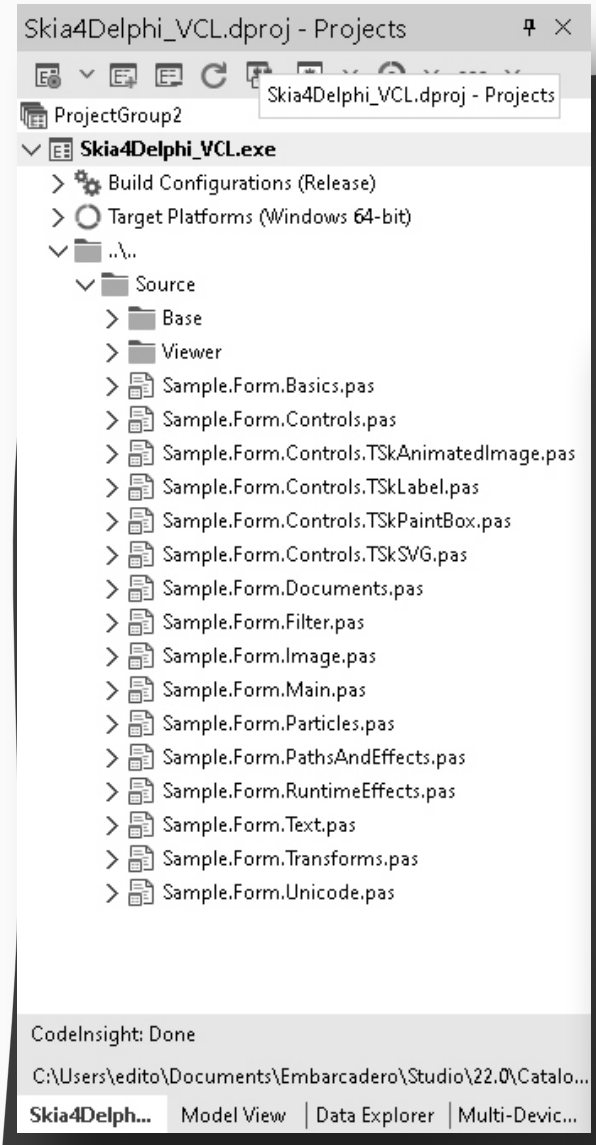
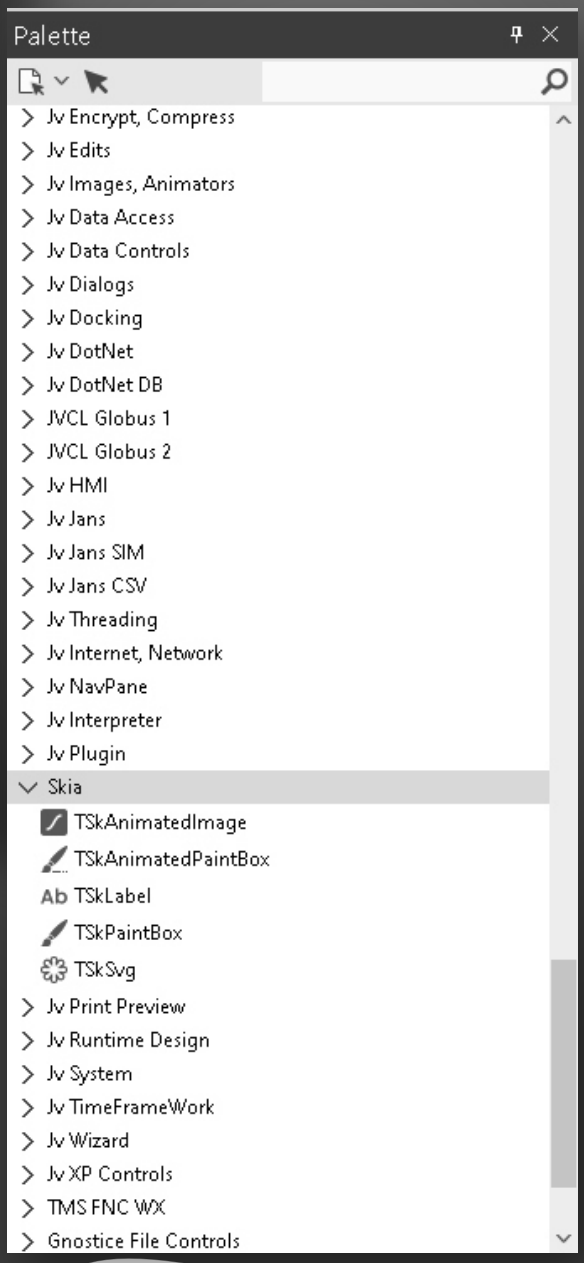


Figure 11: VCL project

By opening each of the units, you gain insight into a huge number of application examples, including tested and perfectly working code.

Just try to play with it, it will give you new perspectives and maybe even new ideas which makes the user interface much nicer and therefore better.

Figure 12:The avialble components for Skia



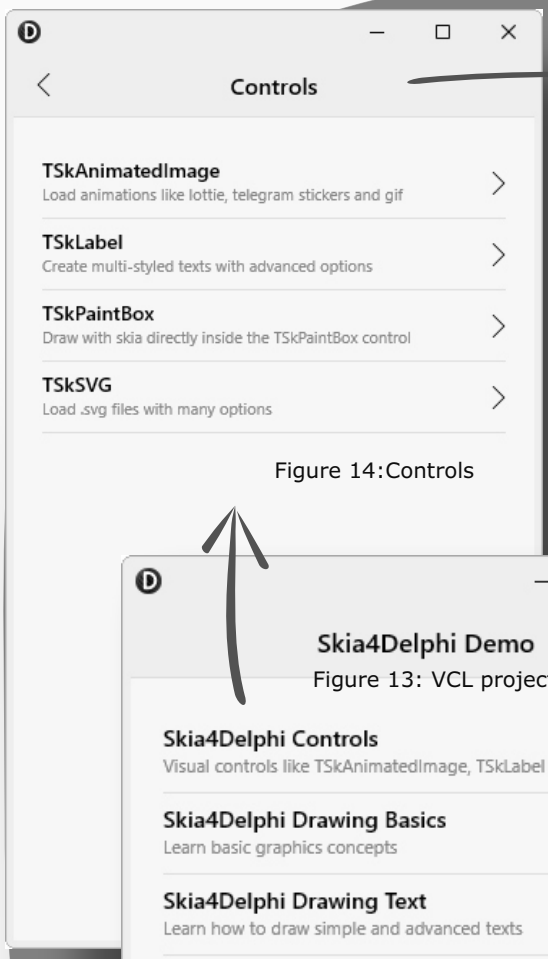


Figure 14:Controls

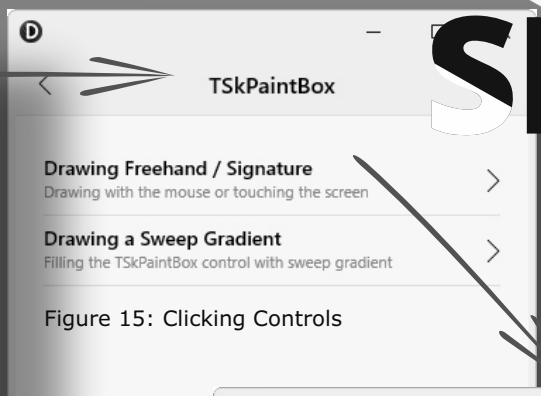


Figure 15: Clicking Controls

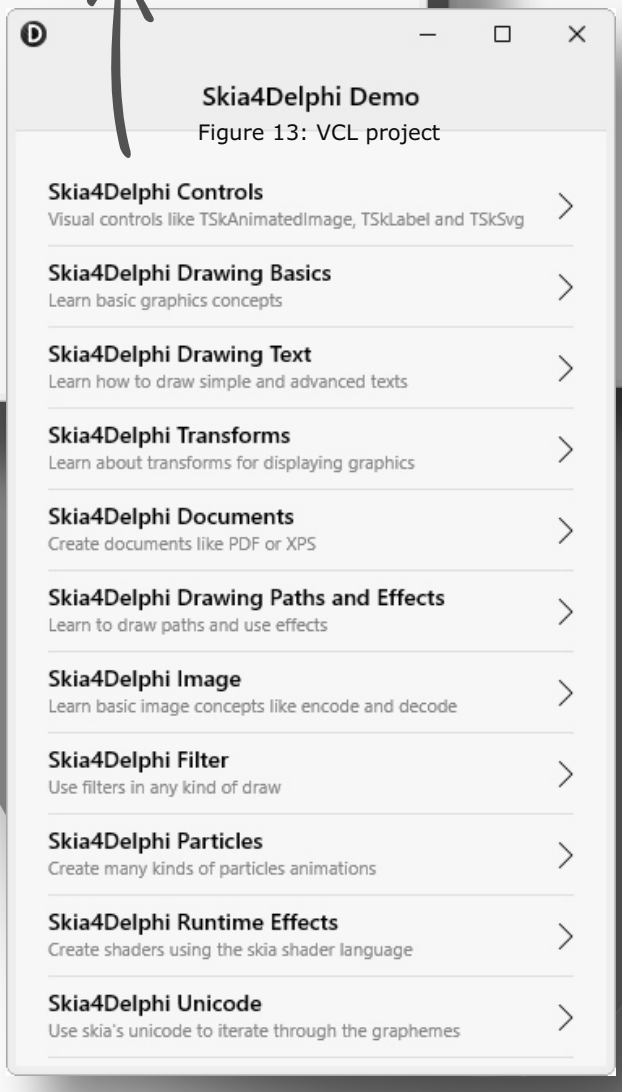


Figure 13: VCL project

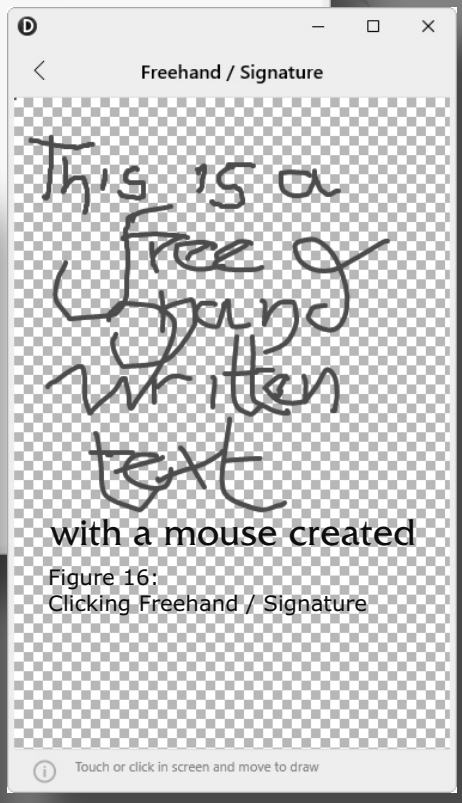


Figure 16:
Clicking Freehand / Signature

After compiling and running you get this overview:
On the next pages (9/10 and 11) you will see the code that belongs to this specific project

Delphi Unit: Sample.Form.Controls.TSK.PaintBox Part 1

```
{*************************************************************************}
{                                                                         }
{                       Skia4Delphi                                       }
{                                                                         }
{ Copyright (c) 2011-2022 Google LLC.                                     }
{ Copyright (c) 2021-2022 Skia4Delphi Project.                            }
{                                                                         }
{ Use of this source code is governed by a BSD-style license that can be  }
{ found in the LICENSE file.                                              }
{                                                                         }
{*************************************************************************}
unit Sample.Form.Controls.TSkPaintBox;

interface

  {$SCOPEDENUMS ON}

uses
  { Delphi }
  ystem.Classes, System.Types, System.UITypes, Vcl.Graphics, Vcl.Controls, Vcl.Forms,
  Vcl.ExtCtrls,

  { Skia }
  Skia, Skia.Vcl,

  { Sample }
  Sample.Form.Base;

type
  TfrmTSkPaintBox = class(TfrmBase)
    pnlFreehand: TPanel;
    lblFreehandDescription: TSkLabel;
    lblFreehandTitle: TSkLabel;
    svgFreehandArrow: TSkSvg;
    pnlFreehandLine: TPanel;
    pnlSweepGradient: TPanel;
    lblSweepGradientDescription: TSkLabel;
    lblSweepGradientTitle: TSkLabel;
    svgSweepGradientArrow: TSkSvg;
    pnlSweepGradientLine: TPanel;
     procedure pnlFreehandClick(Sender: TObject);
    procedure pnlSweepGradientClick(Sender: TObject);
  private
    procedure OnSweepGradientDraw(ASender: TObject; const ACanvas: ISkCanvas; const ADest: TRectF;
        const AOpacity: Single);
  public
    { Public declarations }
  end;

implementation

uses
  { Sample }
  Sample.Form.Viewer.Control;

  {$R *.dfm}

type
  IFreehandRender = interface
  end;

  TFreehandRender = class(TInterfacedObject, IFreehandRender)
  strict private
    FCurrentPath: ISkPath;
    FOldPaths: TArray<ISkPath>;
    FPathBuilder: ISkPathBuilder;
    FPressed: Boolean;
  public
    procedure OnDraw(ASender: TObject; const ACanvas: ISkCanvas; const ADest: TRectF; const AOpacity: Single);
    procedure OnMouseDown(ASender: TObject; AButton: TMouseButton; AShift: TShiftState; X, Y: Integer);
    procedure OnMouseLeave(ASender: TObject);
    procedure OnMouseMove(ASender: TObject; AShift: TShiftState; X, Y: Integer);
    procedure OnMouseUp(ASender: TObject; AButton: TMouseButton; AShift: TShiftState; X, Y: Integer);
 end;

var
 FFreehandRender: IFreehandRender;.
```

```
{ TfrmTSkPaintBox }

procedure TfrmTSkPaintBox.OnSweepGradientDraw(ASender: TObject;
 const ACanvas: ISkCanvas; const ADest: TRectF; const AOpacity: Single);
var
 LPaint: ISkPaint;
begin
 LPaint := TSkPaint.Create;
 LPaint.Shader := TSkShader.MakeGradientSweep(ADest.CenterPoint, [$FFFCE68D, $FFF7CAA5, $FF2EBBC1,
$FFFCE68D]);
 ACanvas.DrawPaint(LPaint);
end;

procedure TfrmTSkPaintBox.pnlFreehandClick(Sender: TObject);
begin
 FFreehandRender := TFreehandRender.Create;
 ChildForm<TfrmControlViewer>.Show('Freehand / Signature', 'Touch or click in screen and move to draw',
  function (): TControl
  var
   LPaintBox: TSkPaintBox absolute Result;
  begin
   LPaintBox := TSkPaintBox.Create(nil);
   LPaintBox.Align := alClient;
   LPaintBox.OnDraw := TFreehandRender(FFreehandRender).OnDraw;
   LPaintBox.OnMouseDown := TFreehandRender(FFreehandRender).OnMouseDown;
   LPaintBox.OnMouseMove := TFreehandRender(FFreehandRender).OnMouseMove;
   LPaintBox.OnMouseUp := TFreehandRender(FFreehandRender).OnMouseUp;
   LPaintBox.OnMouseLeave := TFreehandRender(FFreehandRender).OnMouseLeave;
  end);
end;

procedure TfrmTSkPaintBox.pnlSweepGradientClick(Sender: TObject);
begin
 ChildForm<TfrmControlViewer>.Show('Sweep Gradient', '',
  function (): TControl
  var
   LPaintBox: TSkPaintBox absolute Result;
  begin
   LPaintBox := TSkPaintBox.Create(nil);
   LPaintBox.Align := alClient;
   LPaintBox.OnDraw := OnSweepGradientDraw;
  end);
end;

{ TFreehandRender }

procedure TFreehandRender.OnDraw(ASender: TObject; const ACanvas: ISkCanvas;
 const ADest: TRectF; const AOpacity: Single);
var
 LPaint: ISkPaint;
 LPath: ISkPath;
begin
 ACanvas.Save;
 try
  ACanvas.ClipRect(ADest);
  LPaint := TSkPaint.Create(TSkPaintStyle.Stroke);
  LPaint.AntiAlias := True;
  LPaint.Color := TAlphaColors.Royalblue;
  LPaint.SetPathEffect(TSkPathEffect.MakeCorner(50));
  LPaint.StrokeCap := TSkStrokeCap.Round;
  LPaint.StrokeWidth := 4;

  for LPath in FOldPaths do
   ACanvas.DrawPath(LPath, LPaint);
  if Assigned(FPathBuilder) and not Assigned(FCurrentPath) then
   FCurrentPath := FPathBuilder.Snapshot;
  if Assigned(FCurrentPath) then
   ACanvas.DrawPath(FCurrentPath, LPaint);
 finally
  ACanvas.Restore;
 end;
end;
```

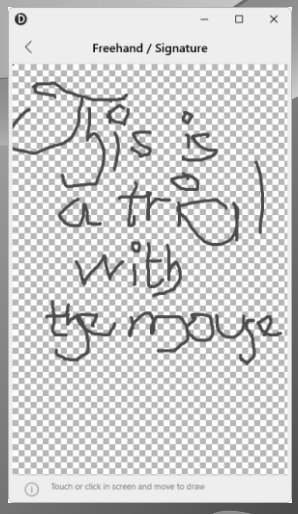
Delphi Unit:
Sample.Form.Controls.TSK.PaintBox
Part 2

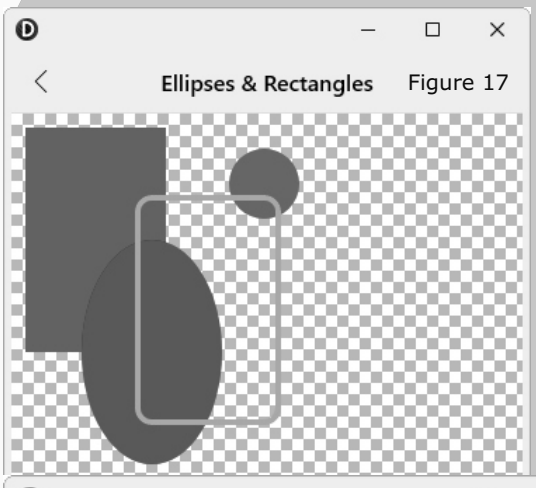
```
procedure TFreehandRender.OnMouseDown(ASender: TObject; AButton: TMouseButton;
  AShift: TShiftState; X, Y: Integer);
begin
  FPressed := True;
  FPathBuilder := TSkPathBuilder.Create;
  FPathBuilder.MoveTo(X, Y);
  FCurrentPath := nil;
end;

procedure TFreehandRender.OnMouseLeave(ASender: TObject);
begin
  if Assigned(FPathBuilder) then
  begin
    if FCurrentPath = nil then
      FCurrentPath := FPathBuilder.Snapshot;
    FOldPaths := FOldPaths + [FCurrentPath];
    FPathBuilder := nil;
    FCurrentPath := nil;
  end;
end;

procedure TFreehandRender.OnMouseMove(ASender: TObject; AShift: TShiftState; X,
  Y: Integer);
begin
  if FPressed and Assigned(FPathBuilder) then
  begin
    FCurrentPath := nil;
    FPathBuilder.LineTo(X, Y);
    (ASender as TSkPaintBox).Redraw;
  end;
end;

procedure TFreehandRender.OnMouseUp(ASender: TObject; AButton: TMouseButton;
  AShift: TShiftState; X, Y: Integer);
begin
  OnMouseLeave(ASender);
end;

end.
```

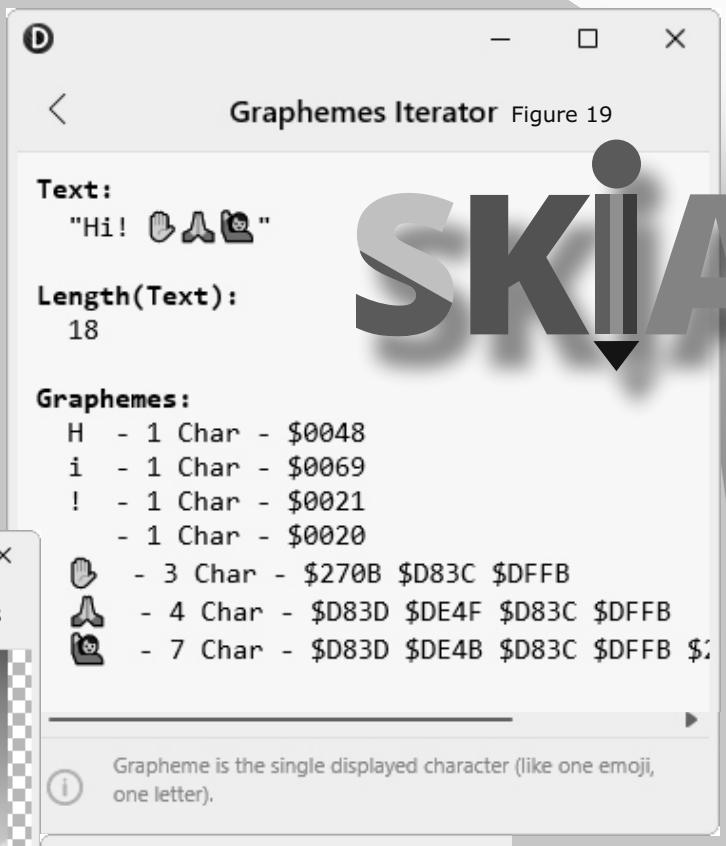
SKIA

Delphi Unit:
Sample.Form.Controls.TSK.PaintBox
Part 3 (Last part)

If you cant read my scribbels:
This is a trial with the mouse.




Ellipses & Rectangles Figure 17


Graphemes Iterator Figure 19
Text:
"Hi! 👋🙏🙋"
Length(Text):
18
Graphemes:
H - 1 Char - $0048
i - 1 Char - $0069
! - 1 Char - $0021
- 1 Char - $0020
👋 - 3 Char - $270B $D83C $DFFB
🙏 - 4 Char - $D83D $DE4F $D83C $DFFB
🙋 - 7 Char - $D83D $DE4B $D83C $DFFB $
Grapheme is the single displayed character (like one emoji, one letter).


Common Color Filter Figure 18
Color filter
Grayscale
Grayscale
Black and White
Technicolor
Sharpen
LSD
Night Vision
Luminance to Alpha
Invert

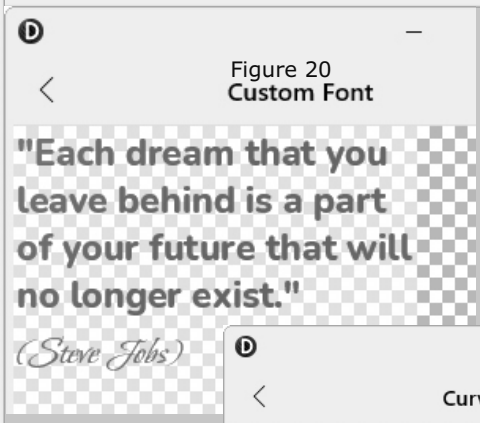
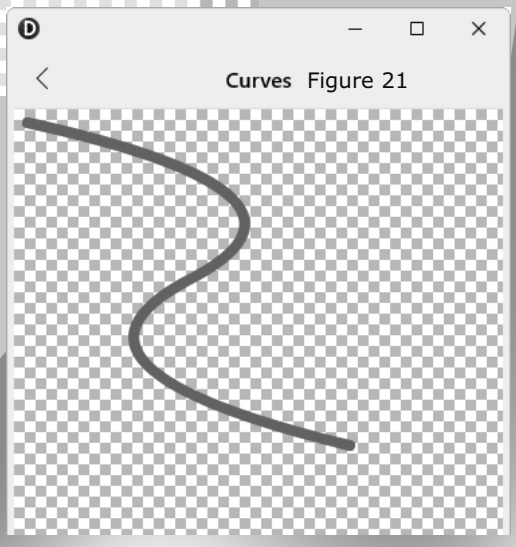


Figure 20
Custom Font
"Each dream that you leave behind is a part of your future that will no longer exist."
(Steve Jobs)


Curves Figure 21

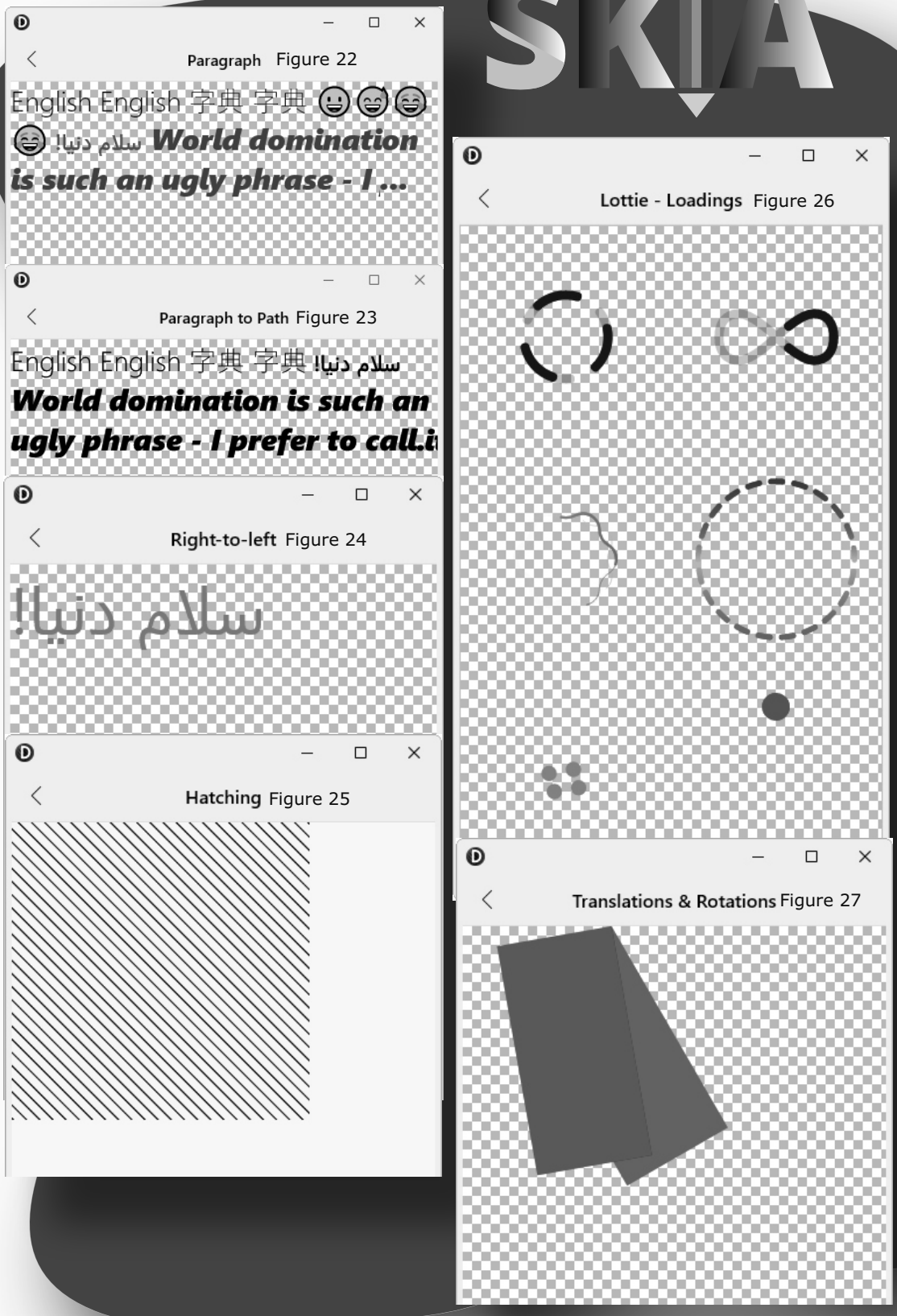
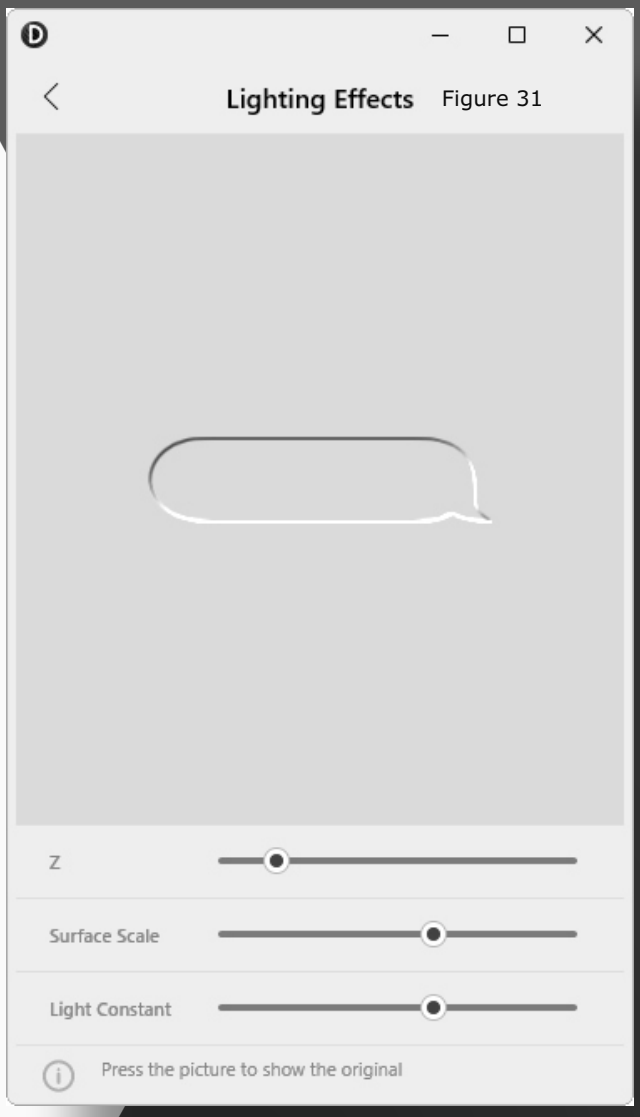
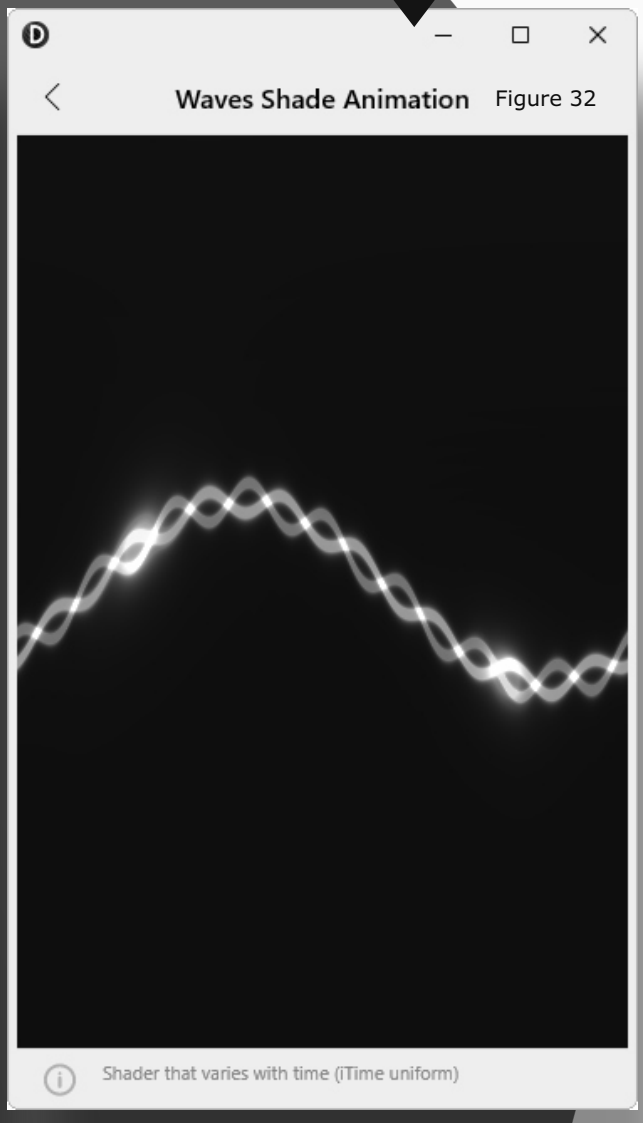

Paragraph Figure 22
English English 字典 字典 😀😅😄
😄 سلام دنیا! World domination
is such an ugly phrase - I ...
Paragraph to Path Figure 23
English English 字典 字典 سلام دنیا!
World domination is such an
ugly phrase - I prefer to call.i
Right-to-left Figure 24
سلام دنیا!
Hatching Figure 25
Lottie - Loadings Figure 26
Translations & Rotations Figure 27

Figure 30 The walking pinocchio:
a beautifull animated Gif Image.
To be found at
Skia4Delphi Controls - TSkAnimatedImage


Lighting Effects
Figure 31
Z
Surface Scale
Light Constant
Press the picture to show the original


Waves Shade Animation
Figure 32
Shader that varies with time (iTime uniform)

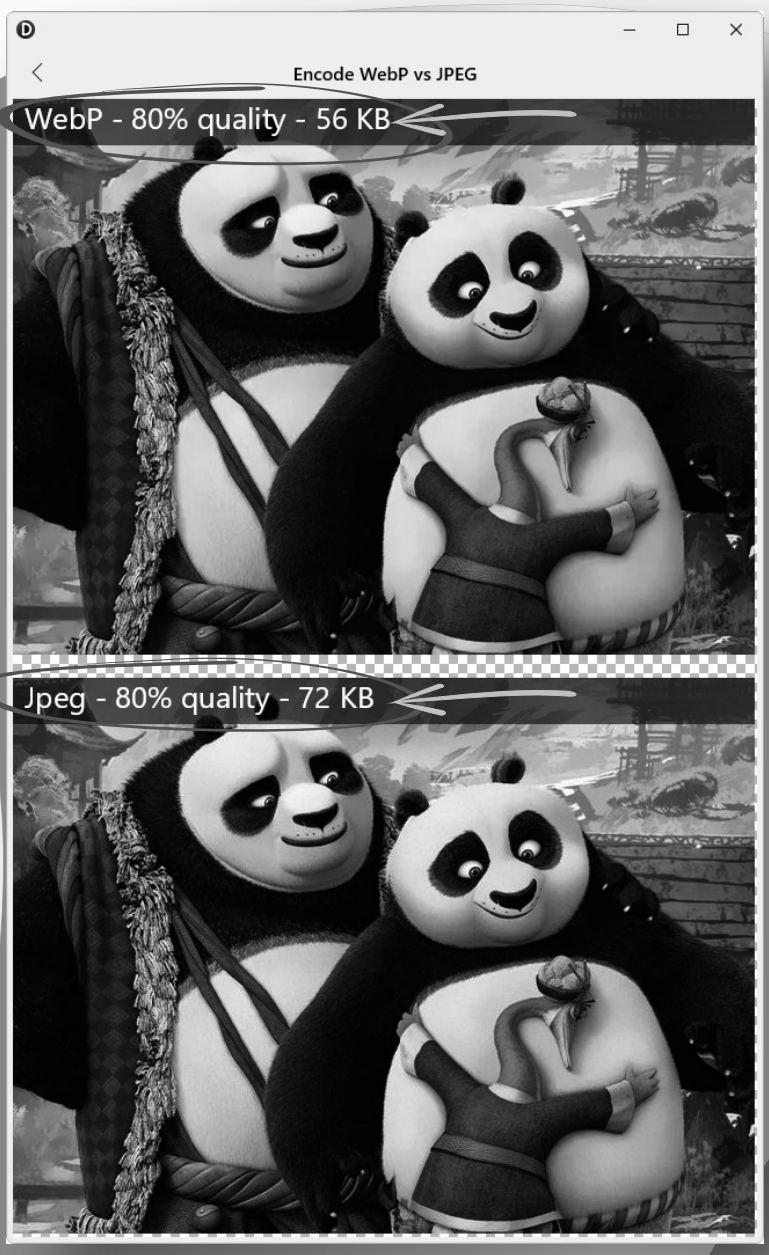
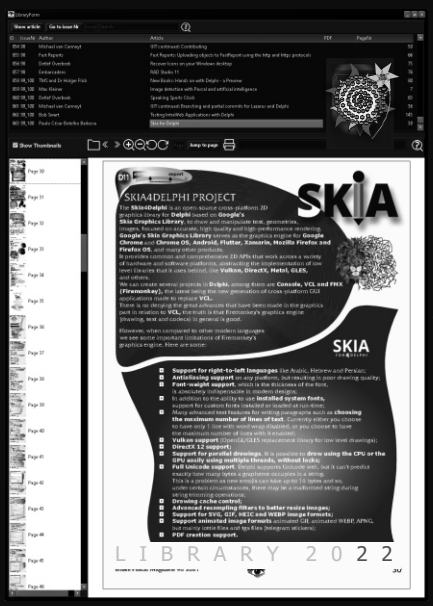


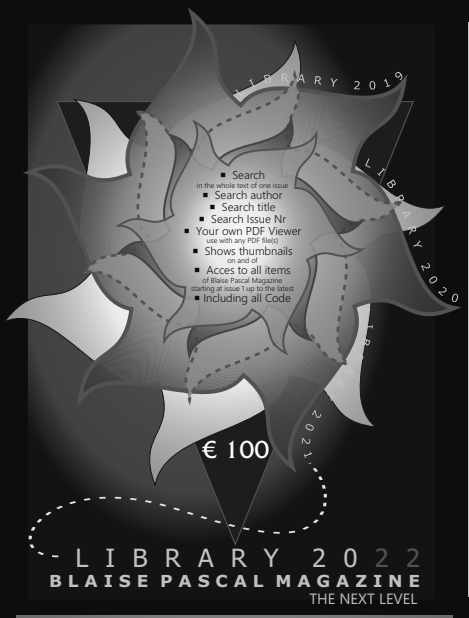
Figure 33
Notice the size of the file and the comparable percentage

WIKIPEDIA

WebP is an image file format developed by **Google** intended as a replacement for **JPEG, PNG,** and **GIF** file formats. **WebP** yields files that are smaller for the same quality, or of higher quality for the same size. It supports both lossy and lossless compression, as well as animation and alpha transparency.

Google announced the **WebP** format in September 2010, and released the first version of its supporting library in April 2018.





NEW DEVELOPED

2 LIB STICK PROGRAM

SUPER OFFER

Normal Price € 271 € 150

LAZARUS HANDBOOK

FOR PROGRAMMING WITH FREE PASCAL AND LAZARUS

3 including 30 example projects

934 PAGES

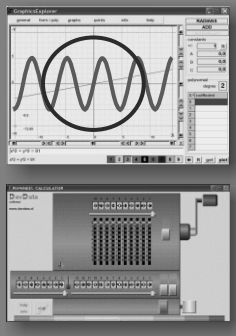
4 including 19 example projects

LEARN TO PROGRAM USING LAZARUS

HOWARD PAGE-CLARK

DAVID DIRKSE

including 50 example projects



5

BLAISE PASCAL MAGAZINE

www.blaisepascal.eu

COMPUTER (GRAPHICS) MATH & GAMES IN PASCAL

1

Code with Geo Location By Max Kleiner
Database Workbench: a great new Version / better than ever before By Detlef Overbeek
Artificial Intelligence By Danny Wind
The new approach for Qubits By Detlef Overbeek
Pascal for Quantum Computers? By Detlef Overbeek
PAS2JS samples for creating your new website By Michaël van Canneyt
PAS2JS Electron
FastReport: New Version By Michael Philippenko
Fast Report: Using the Table Object in Table Reports By Michael Philippenko
Creating and implementing Icons for Lazarus By Detlef Overbeek
Skia for Delphi the next level

UKRAINE

1. One year Subscription
2. The newest LIB Stick - including Credit Card USB stick
3. Lazarus Handbook - Personalized - PDF including Code
4. Book Learn To Program using Lazarus PDF including 19 lessons and projects
5. Book Computer Graphics Math & Games book + PDF including ±50 projects

2

https://www.blaisepascalmagazine.eu/product-category/special-offer/

INSTALLING FPC TRUNK VERSIONS

DETLEF OVERBEEK

ABSTRACT

This article is meant to help you create or update your trunk version of FPC installed in your Lazarus version.

INSTALLING FOR THE FIRST TIME:

In Windows Explorer or Total commander (*or any other filesystem on your computer*) you can right-click with your mouse on **Git Clone** (*Figure 1*). I have used the path: `C:\lazarus\fpc\3.3.1\source`.
The directory `3.3.1` and `source` needs to be created by hand.

You can't do it wrong because the first time **Git Clone** is available and disappears when you're done.
A new window appears (*Figure 2*):
Next to the **URL** you type in the complete address:
`https://gitlab.com/freepascal.org/fpc/source.git`

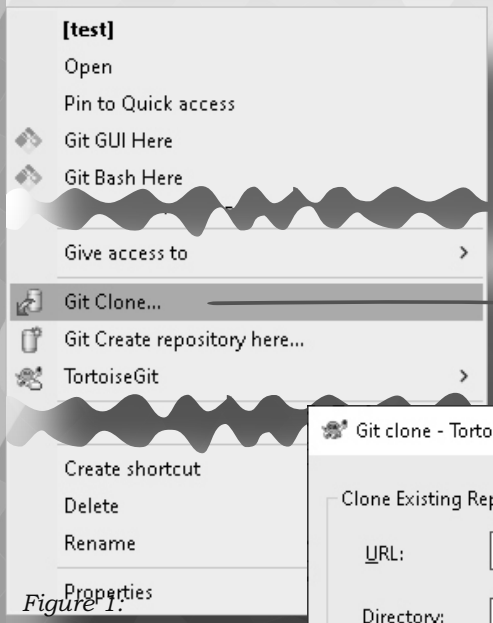


Figure 1:

The directory where it will be installed is:
`C:\lazarus\fpc\3.3.1\source`
Click OK.
Yours can be differernt if you want that.
In (*Figure 2*) you see how that looks like.

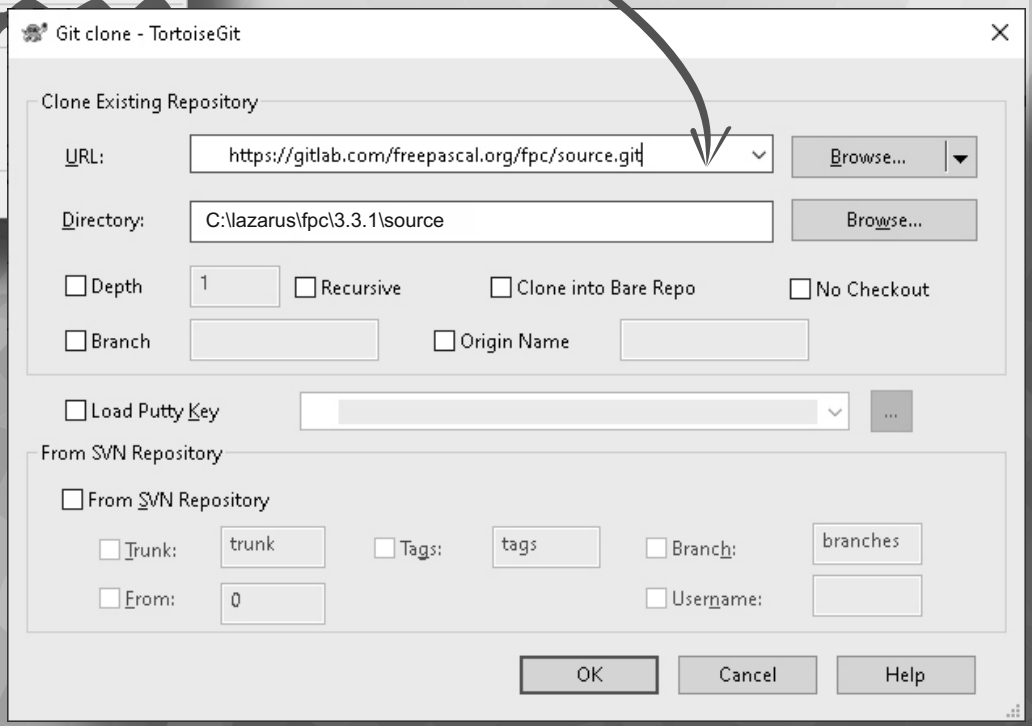


Figure 2:

INSTALLING FOR THE FIRST TIME:

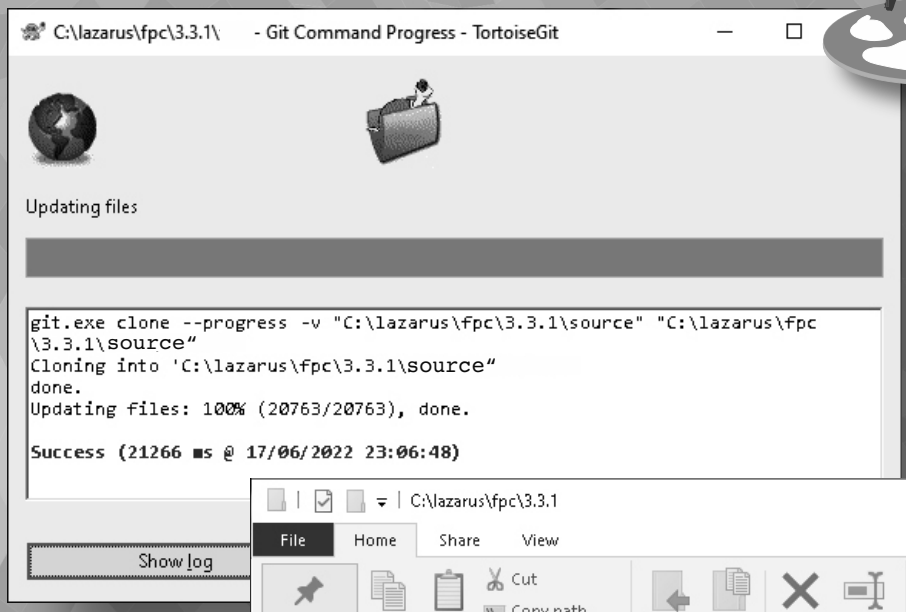


Figure 3:

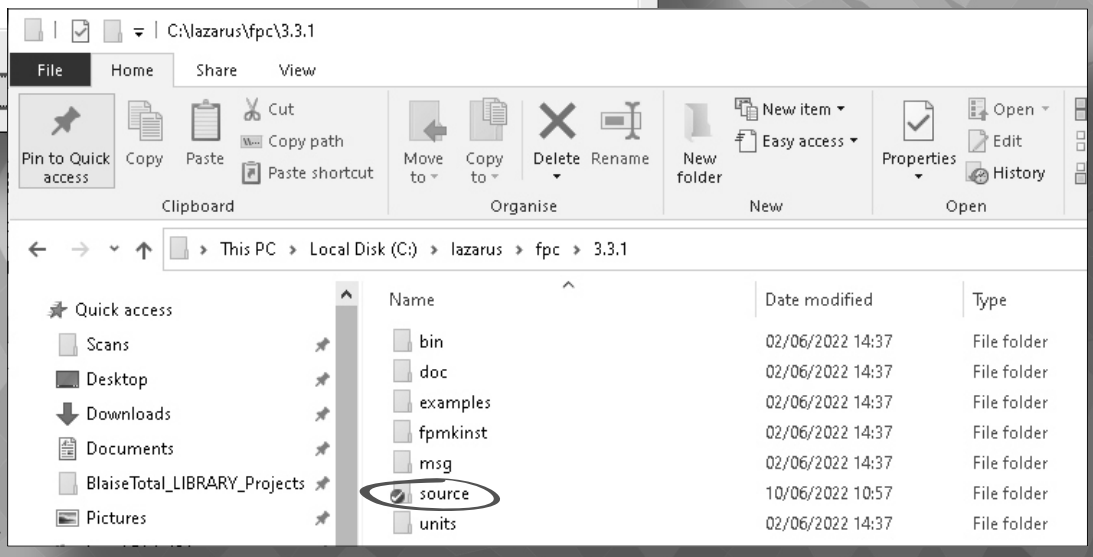


Figure 4:

Now the installation will be created in the directory "source" (*as Figure 3 and 4 shows, - made in Windows File Explorer.*)
You can see the directory structure in the next *Figure 5*, made with **Total Commander,** which I think is still the best File Explorer for Windows I know.

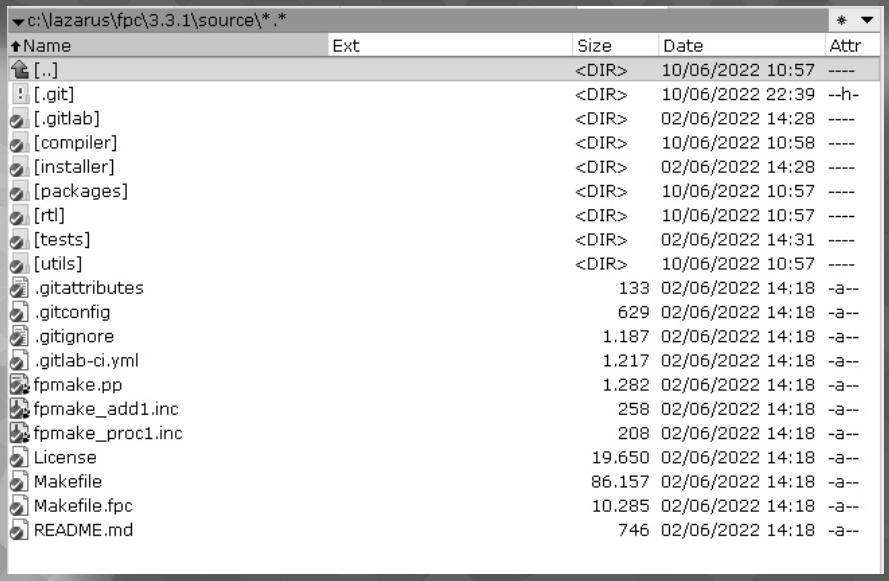


Figure 5:

THE COMPILATION

The compilation is the second step which I describe here so you can run this in an easy way. For the future this will be created with the help of a wizard but now you will have to do it the old fashioned way:
Open the command prompt window and go to the directory where your sources are available.

```
c:\lazarus\fpc\3.3.1\source\
```

If you do not have the `c:\lazarus\fpc\3.3.1\` directory you need to create it first. See the structure Figure 4 on page 2

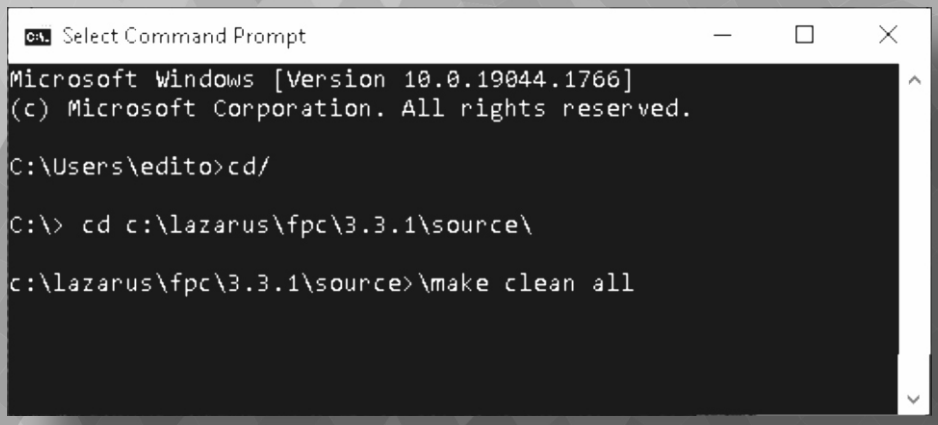


Figure 6:

See figure 6 to compile **FPC** type after the prompt: **`make clean all`**
(*to make sure all files are updated*)
Have a look at **Read this first on the next page**
This will take a while, up to several minutes depending on your computer speed. Have a look at the **on the next page**

Once it is compiled we have to do the last step:

- Launching **Lazarus** and recompiling it.
  But before you do so you need to make sure you that the directory
  `C:\Lazarus\fpc\3.2.2\bin\x86_64-win64\fpc.exe`
  (*in our case as example so for you it could be different*)
  is added to your **`path environment variable`**.

INSTALLING FOR THE FIRST TIME:

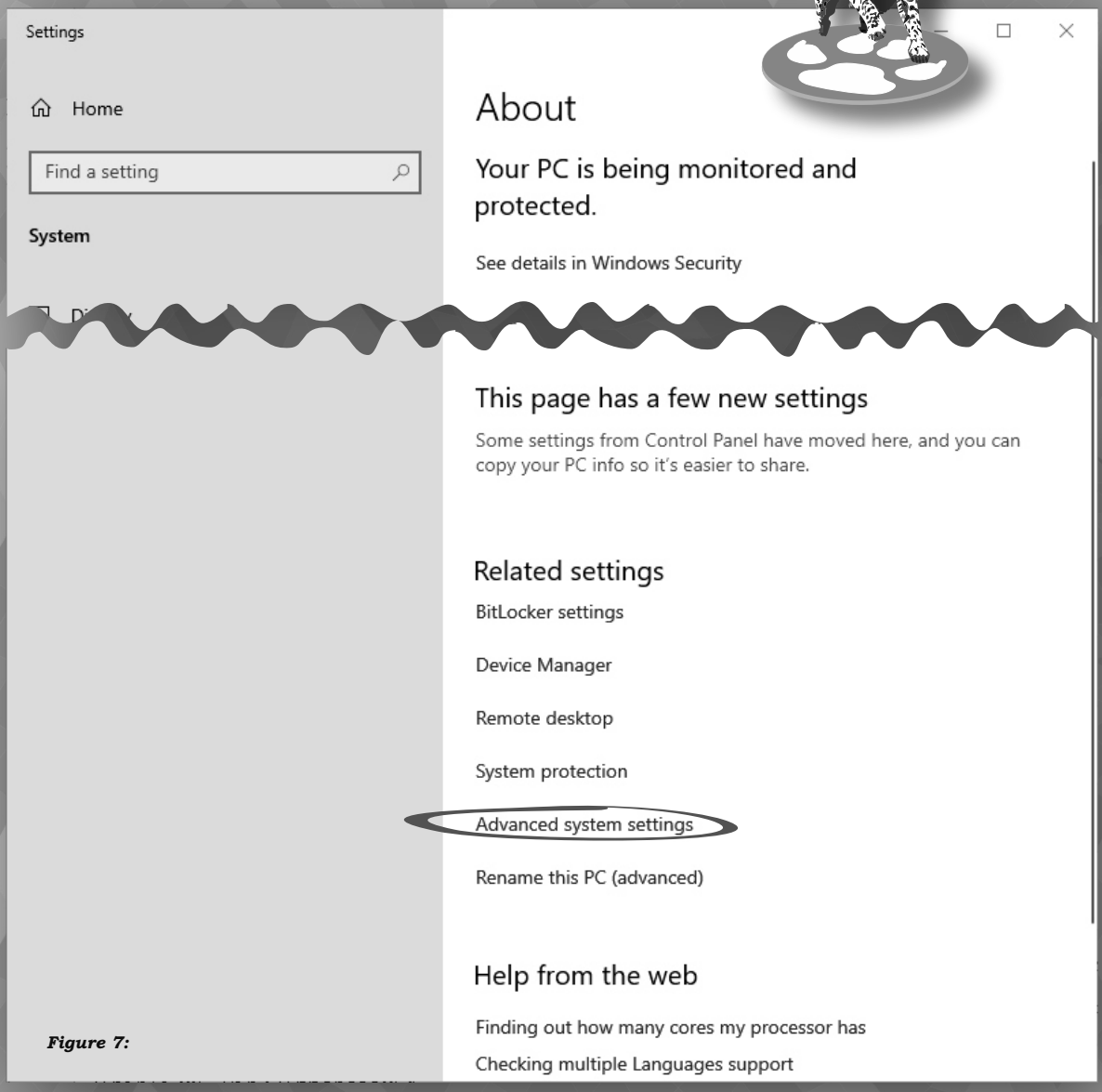


Figure 7:

READ THIS FIRST

Here is an example: If you need to change your path so that the correct **`make.exe`** is found, you can either do it temporarily or permanently.

- To change it **temporarily** (*it is in effect only until you close the current command prompt*) to find **`make.exe`** in `C:\Lazarus\fpc\3.2.2\bin\x86_64-win64\` at a command prompt type:
  `C:\Source> set PATH=``C:\Lazarus\fpc\3.2.2\bin\x86_64-win64;`**`%PATH%`**

- To change it permanently, open the **About Page** of **Windows** then click on**:**
  Advanced system settings → System properties. Now choose **Environment Variables** in the **System Properties dialog**.
  (*see Figure 7 at top and Figure 8 and 9 of the next page of this article*)
  Here you will probably need to change the Path that appears under "**System variables**" so that `C:\Lazarus\fpc\3.2.2\bin\x86_64-win64;` occurs first.

INSTALLING FOR THE FIRST TIME:

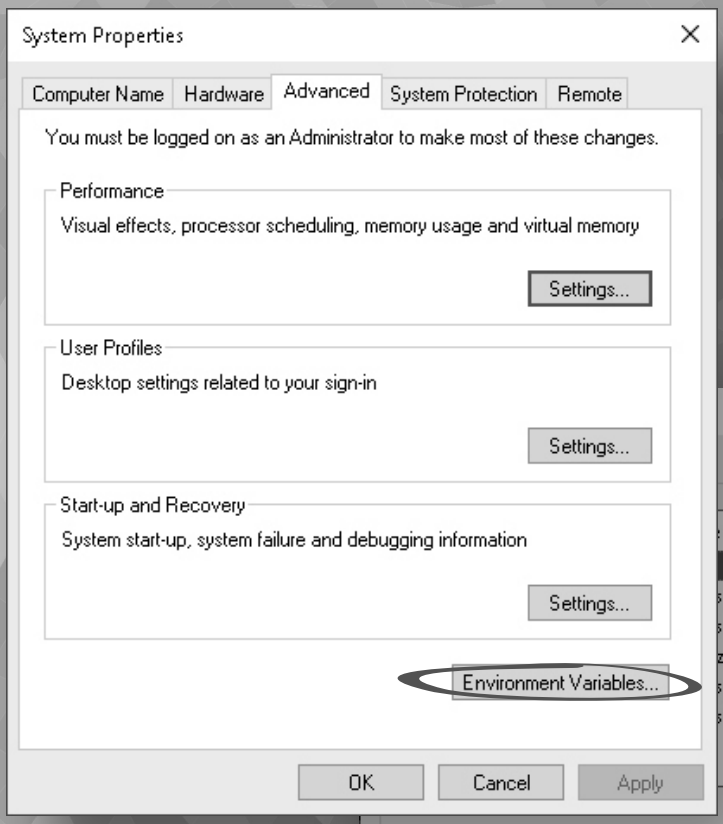


Figure 8:

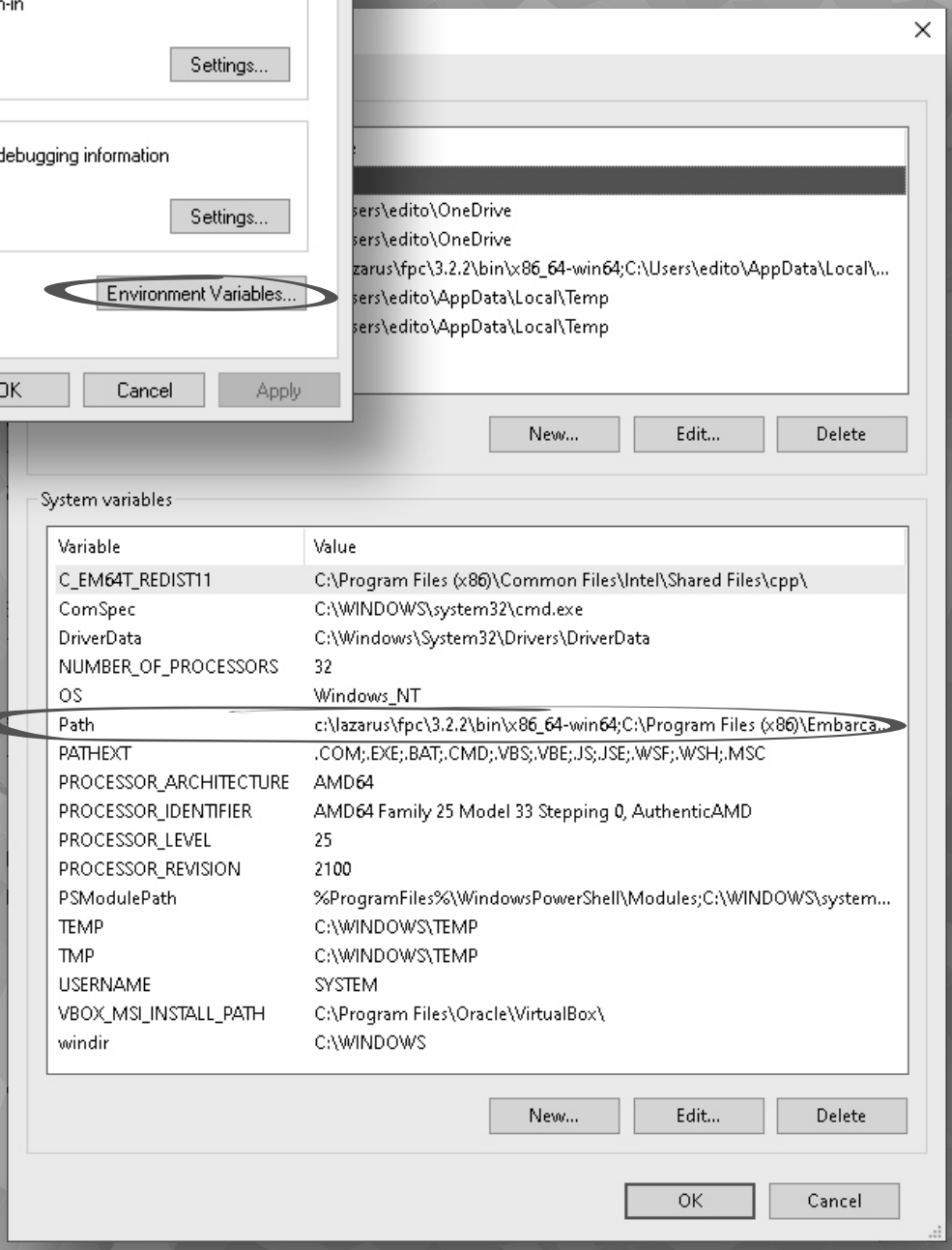


Figure 9:

INSTALLING FOR THE FIRST TIME:

Figure 10:

The installation **PREFIX** in the instructions that follow is totally dependent on the directory in which you installed FPC. Lazarus installs and uses a default location of `C:\lazarus\fpc\3.2.2` and the FPC 3.2.2 release files are placed in `C:\lazarus\fpc\3.2.2.`

For the last **Lazarus version 2.2.0, FPC** was automatically installed `$Path($(CompPath))\make.exe`

In the trunk version(**Lazarus** 2.3.0) `C:\lazarus\fpc\3.2.2\bin\x86_64-win64\make.exe` which is the original version and works with the environment variable.

`Installing_an_FPC_release_with_the_Windows_installer`

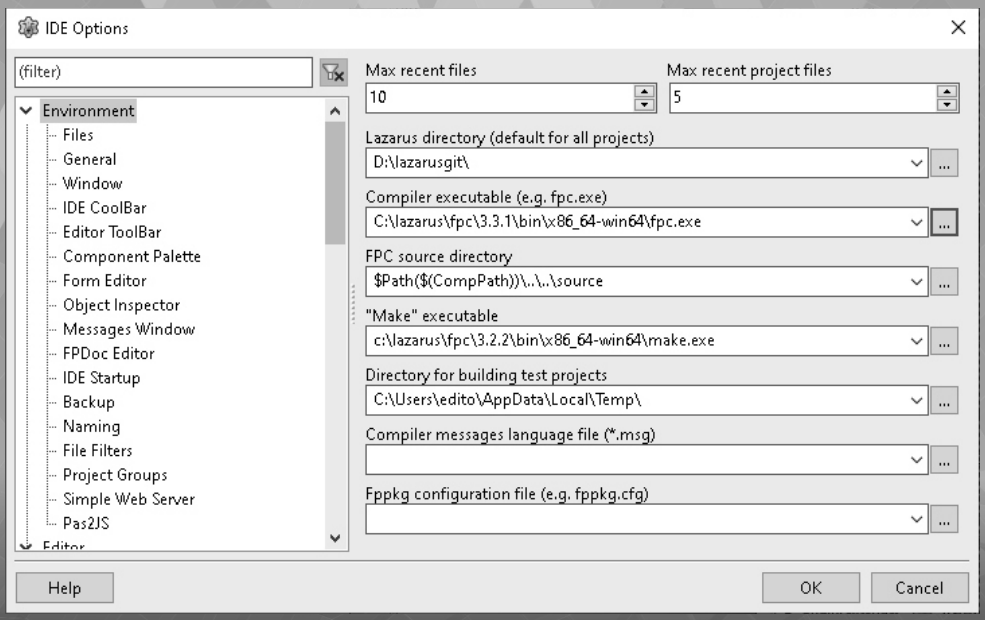


Figure 11:
The Ide Options show all the relevant paths

SYNCHRONIZING FROM GIT:

SYNCHRONIZING FROM GIT

The second possibility - you have installed **Git** already in **Lazarus/FPC** - is to **synchronize** from GIT.

The Drop down menu appears after right-clicking in your File Manager and you should choose **Git Sync ...** as you can see in figure 10. *Continues on the next page*

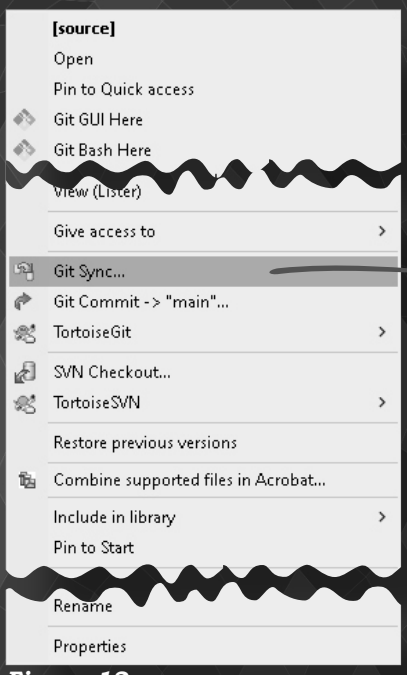


Figure 12:

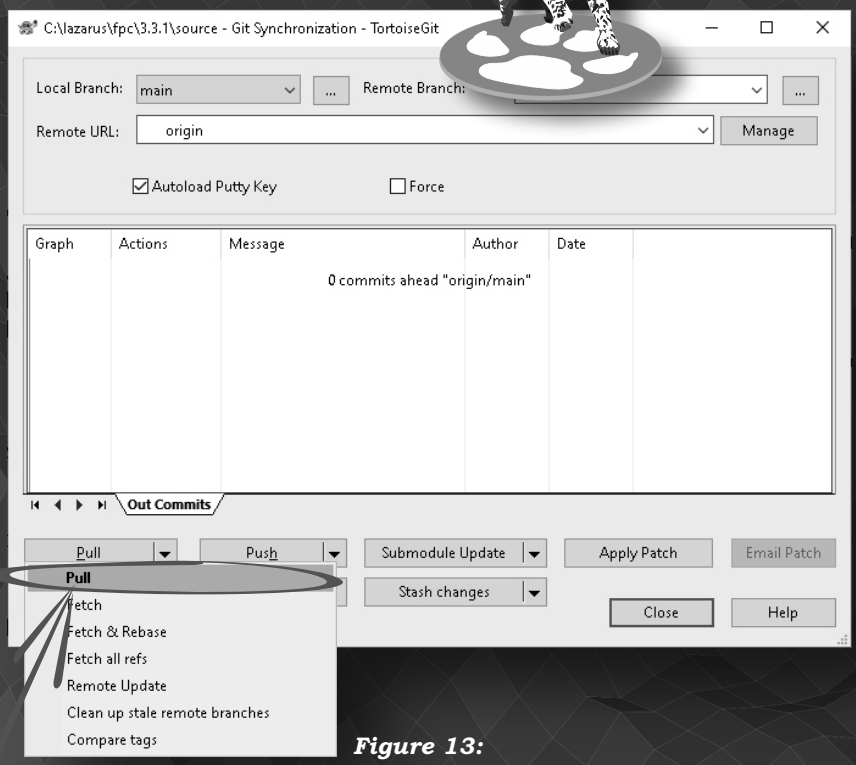


Figure 13:

After clicking the item **Git Sync** (*Figure 12*) you will see a form like figure 13 shows: The **GIT Synchronization – Tortoise Git.** At the bottom there is button "**Pull**" which if activated shows a drop down menu. Choose **Pull**.

In the window of Figure 14 you can notice the changes that have been downloaded.

NOTE: YOU NEED TO COMPILE FPC AGAIN AND AFTER THAT COMPILE THE LAZARUS VERSION.

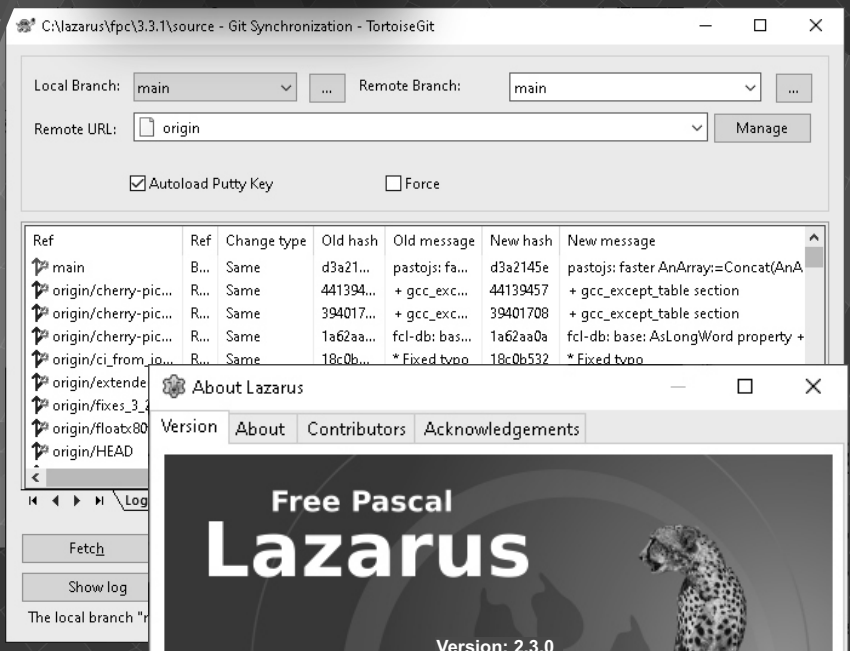


Figure 14:

Figure 15:
Check the version number in the about box

ABSTRACT

Database Workbench, a database development tool has created the next major version 6 and is already available.
I write about this tool because it is my favourite for this kind of Job:
it is a development tool with support for **Oracle, SQL Server, MySQL, MariaDB, PostgreSQL, Firebird, InterBase and NexusDB.**
For those of you who are not yet familiar with the tool, I start here an introduction that will show you some extra ordinary capabilities it has.
Keeping in mind what some programs cost:
this is extremely cheap – especially for the quality you get.

INTRODUCTION

Database Workbench is a database development tool for novices and professionals alike.
It covers a range of tools, from basic design features, with conceptual to physical diagramming tools, to database object creation and modification. ➜

It can generate test data, run queries or debug stored procedures.
Migrate **metadata** *1 from one system to another, to exporting or moving data.

It has a very special user interface which looks clean and is consistent.
It has **High DPI** *2 support and **Multithreading** *3 where possible and includes a lot of tools to speed up the development.
Some of these tools are extremely helpful special for novices to the subject.

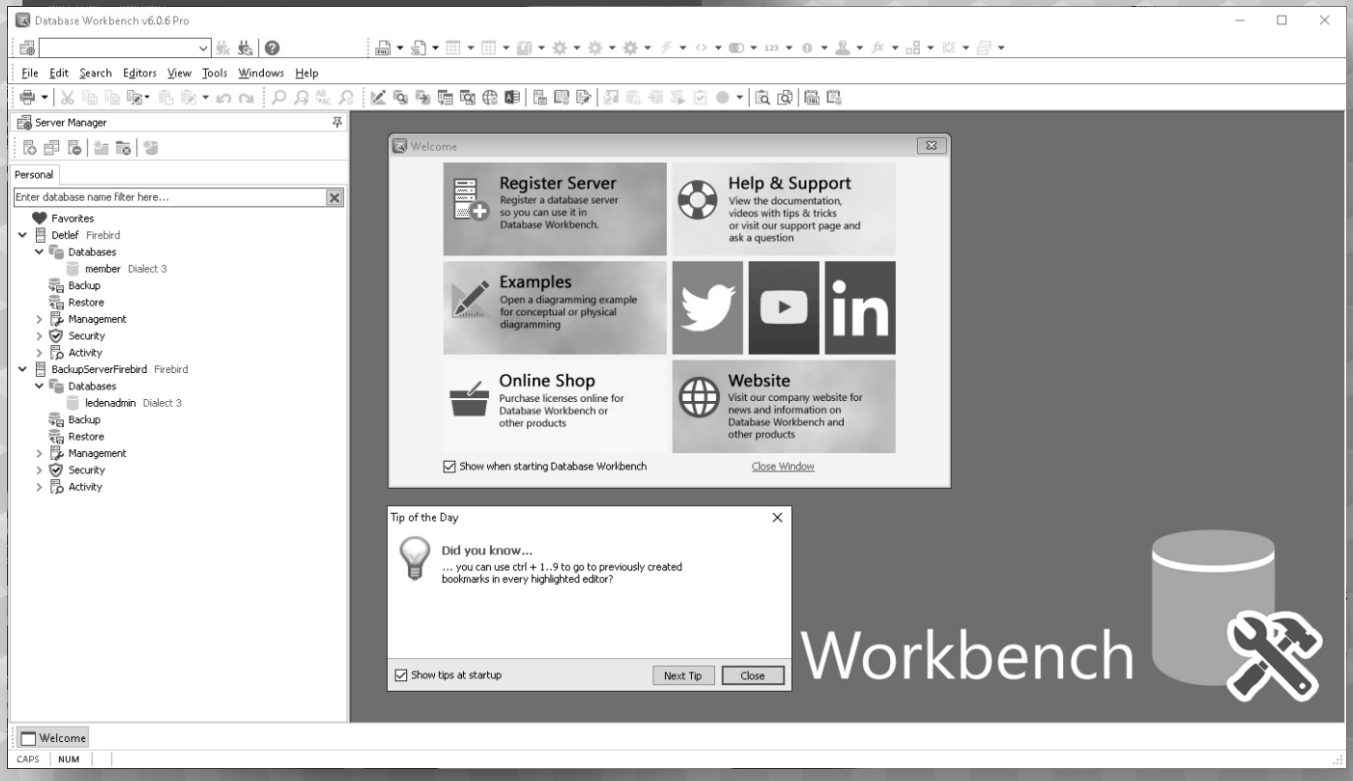


Figure 1. Overview of the IDE

WikipediA

1 *META DATA*
Metadata *is data that provides information about other data, but not the content of the data such as the text of a message or the image itself. Metadata is not strictly bounded to one of these categories, as it can describe a piece of data in many other ways.*

2 *HIGH DPI*
(High)DPI (Dots Per Inch) *is the relation between size in pixels and the actual display size. Here dot can be an equivalent for pixel in printing terminology. Applications can either use pixel sizes, or take into account the actual display size. In this second case, sizes are given in points. Most of today operating systems use default DPI set to 96 and allow to change it to higher value manually. The physical DPI can be determined from display through the* **EDID** *protocol from physical size data and actual resolution.* **But the physical DPI is not used automatically by the system** *so if you connect video output to monitor with different size, then screen resolution and visual size of controls are* **not automatically changed.** *Usually DPI is presented as one value but it can be different for horizontal and vertical axes if pixel is not square. In addition to basic application DPI awareness you can add own DPI options to your application to allow users to set custom per application DPI to overcome wrong system DPI setting.*

3 *MULTITHREADING*
Multithreading is the ability of a program or an operating system to enable more than one user at a time without requiring multiple copies of the program running on the computer. Multithreading can also handle multiple requests from the same user.

THREE DIFFERENT EDITIONS AND MODULAR DATABASE-SUPPORT
Version 6 introduces the Enterprise edition. There are already the Basic and Pro Version. The differences:

BASIC
Is mostly for limited demand is practically only to be used with:
* Commercial database development
* Ability to register more than 2 servers
* Ability to register/use more than 3 databases per server
* Ability to use multiple database systems

PRO (Professional)
Has anything but not the special items of team server

ENTERPRISE (for teams)
Connect to Database Workbench TeamServer

Database Workbench licenses are available with support for a single database system or for multiple database systems.
You can even add an additional database support at a later stage if required.

UPDATES, UPGRADES & SUPPORT
You will receive free updates and support for a year, additional update and support periods can be purchased later at a small fee, see each edition-section above for details.

This website address
https://www.upscene.com/database_workbench/editions#featurenotes
lists a feature matrix that shows the differences between the editions.
There are various pricing models for the three editions and for the separate database modules:
there's always a combination that has what you need.

LETS START

Database Workbench is a complete database-development tool that can help you to create your database from scratch, making visual documentation (structures) to the final goal: a complete database.
Even the creation can be done visual.

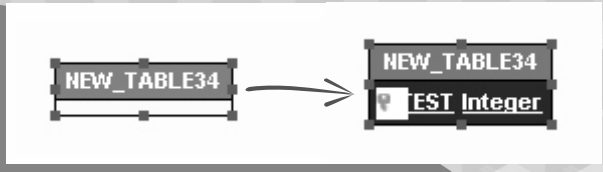
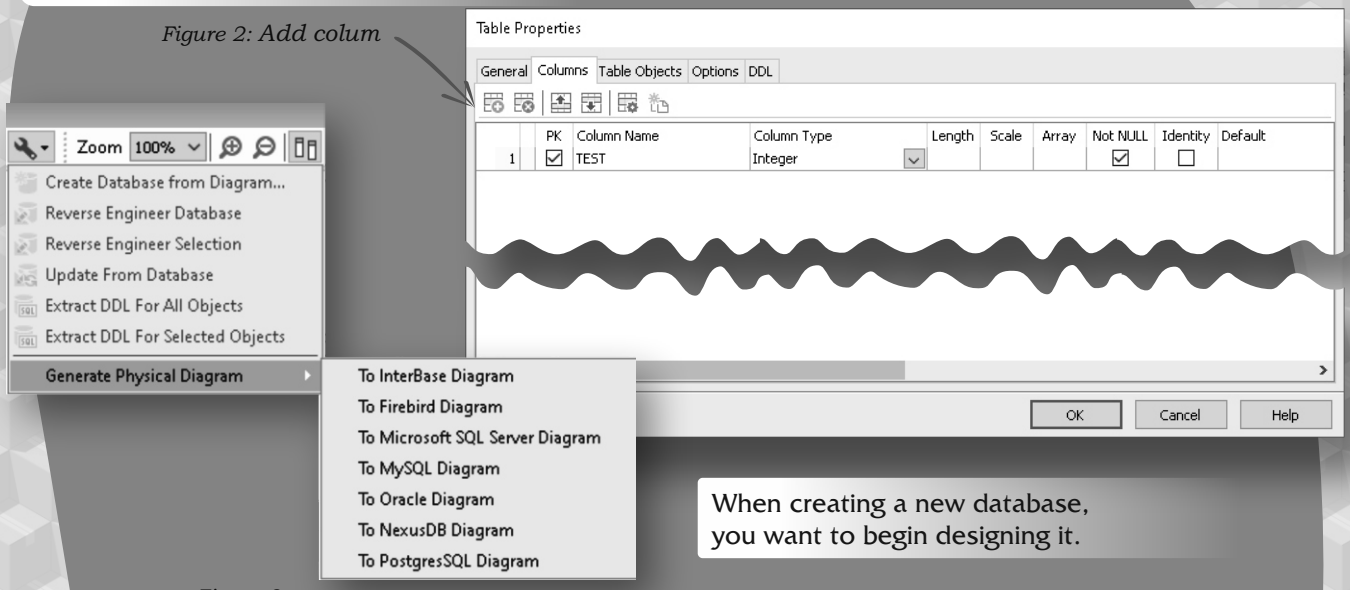


Figure 4: you evn can colorize

Figure 2: Add colum



When creating a new database, you want to begin designing it.

Figure 3: Generate a physical diagram

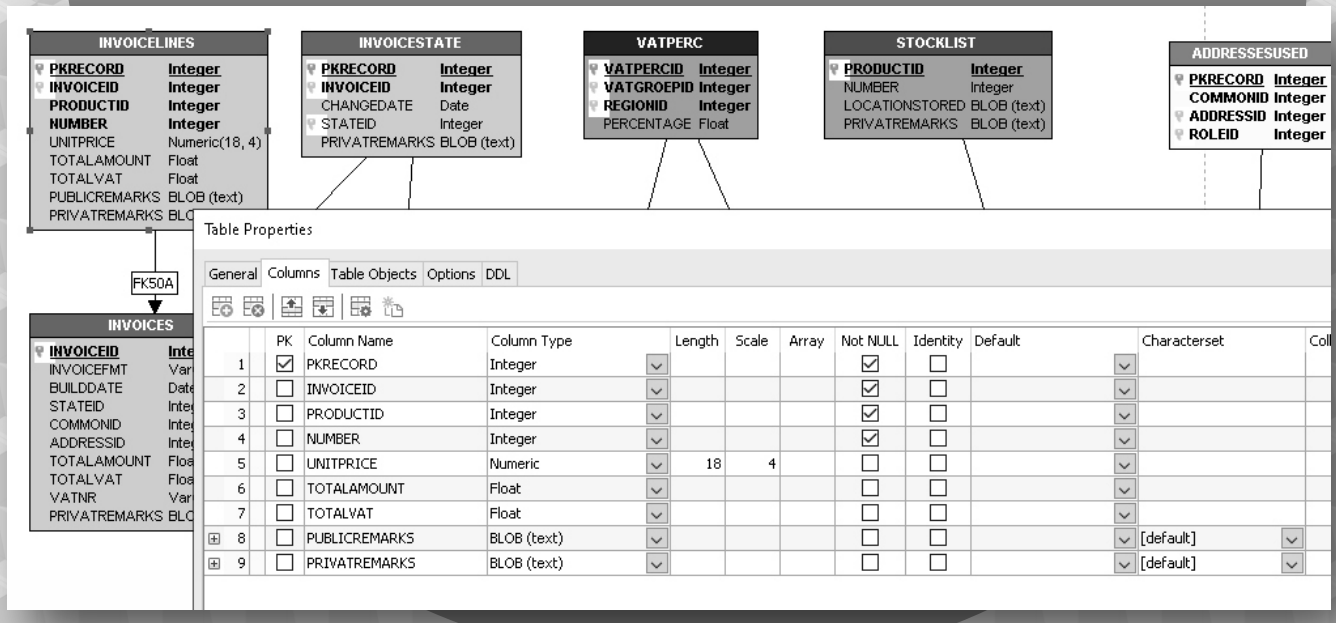


Figure 5: A n example view of table

The diagramming tools allow you to do just that:
keep on modifying your database design without restrictions.
This can be done on **two levels**, the **conceptual model** and the **physical model**. I will show you how.
The conceptual model is somewhat more abstract, it's an **"entity-relationship model" (ER model**).
You don't design tables, but **'entities'** with **'attributes'.** Entities can have relationships between them with different **cardinalities.**
A relationship can be optional, **`required, one-to-one, one-to-many or many-to-many.`**

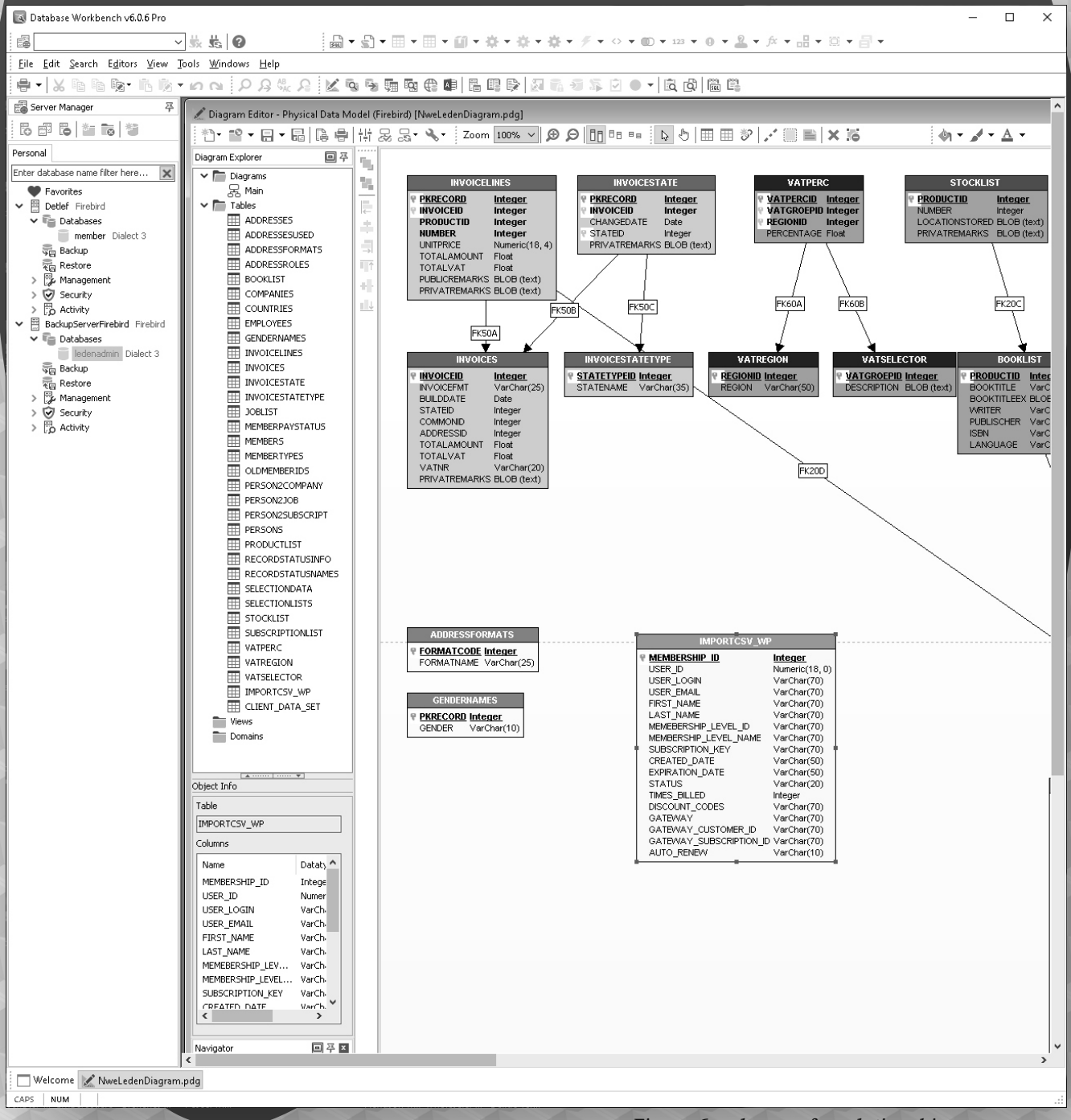


Figure 6: color use for relationships

ER MODEL

An **entity–relationship model** (or **ER model**) *describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of* **entity types** *(which classify the things of interest) and specifies relationships that can exist between entities (instances of those entity types).*

In software engineering, an **ER model** *is commonly formed to represent things a business needs to remember in order to perform* **business processes**.
Consequently, the ER model becomes an abstract data model, that defines a **data-structure** *or* **information-structure** *which can be implemented in a database, typically a relational database.*
Entity–relationship modelling *was developed for database and design by* **Peter Chen**.
An ER model is usually **the result of systematic analysis** *to define and describe what is important to processes in an area of a business.*
It does not define the business processes; it only presents a business data schema in graphical form.
It is usually drawn in a graphical form as boxes (entities) that are connected by lines (relationships) which express the associations and dependencies between entities. (See figure 6 at page 94)
An ER model can also be **expressed in a verbal form**, *for example: one building may be divided into zero or more apartments, but one apartment can only be located in one building.*

Entities may be characterized not only by relationships, but also by additional properties (attributes) (See figure 6 which include identifiers called **"primary keys".**
Diagrams created to represent attributes as well as entities.

An ER model is typically implemented as a database.
In a simple **relational database implementation**, *each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a relational database a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity. (See figure 13 at page 100)*
So far the lesson about Entities.

Once you've created your database, or if you have an existing database, or when you want to create your database by hand, Database Workbench has database object editors.
These object editors have a consistent layout for all different types of database objects, like tables, indices, stored procedures and so on.

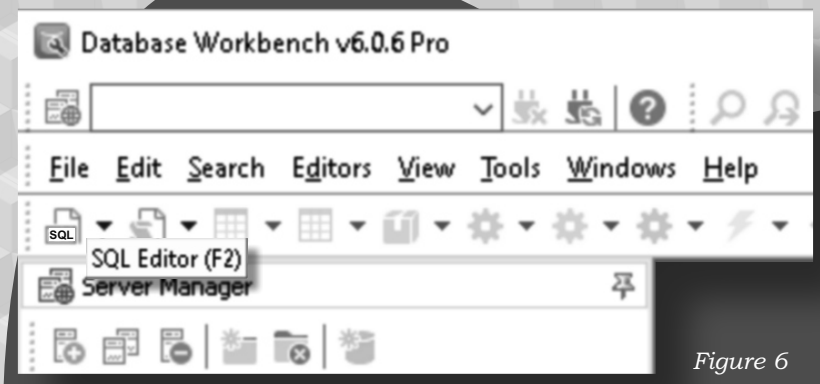


Figure 6

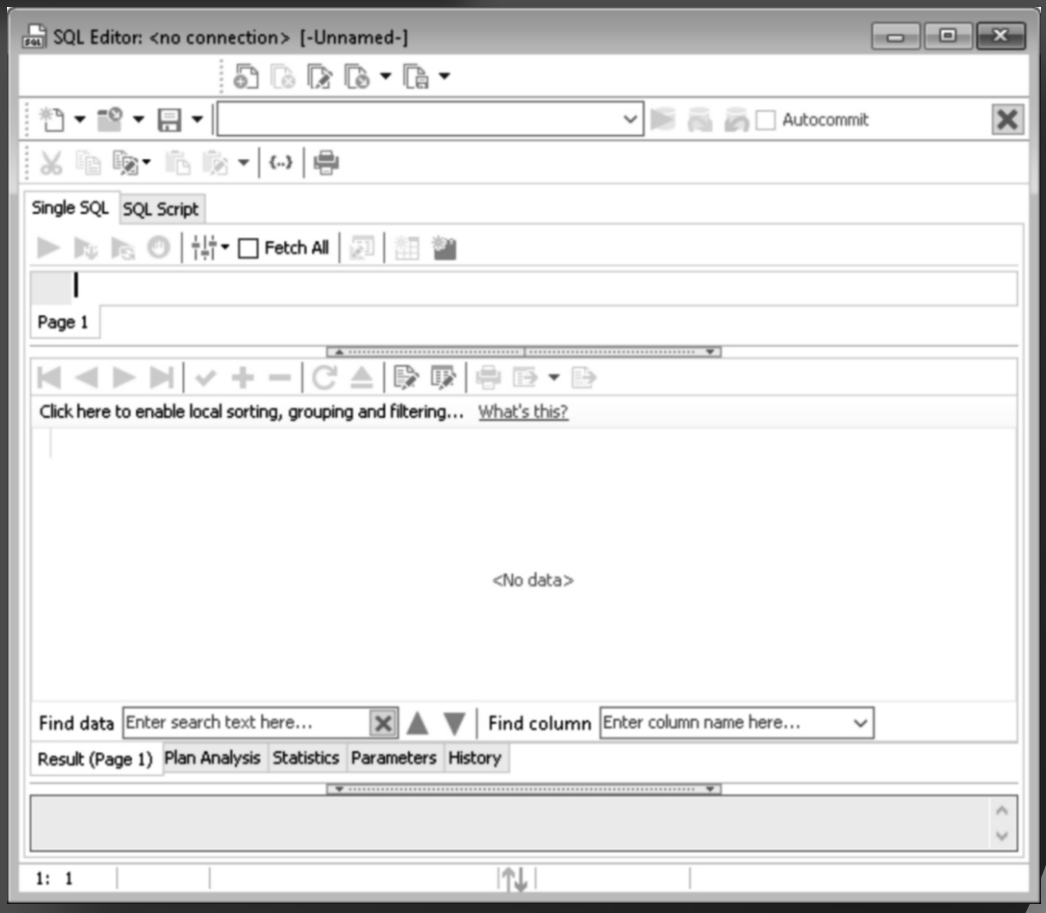


Figure 7

At the top is the toolbar, besides New, Copy, Save, Drop.
There's also a button to retrieve the SQL statement to create or modify the object.
Depending on the object type, there can be additional buttons, like Execute, for example.

Another tab holds the object SQL statement.

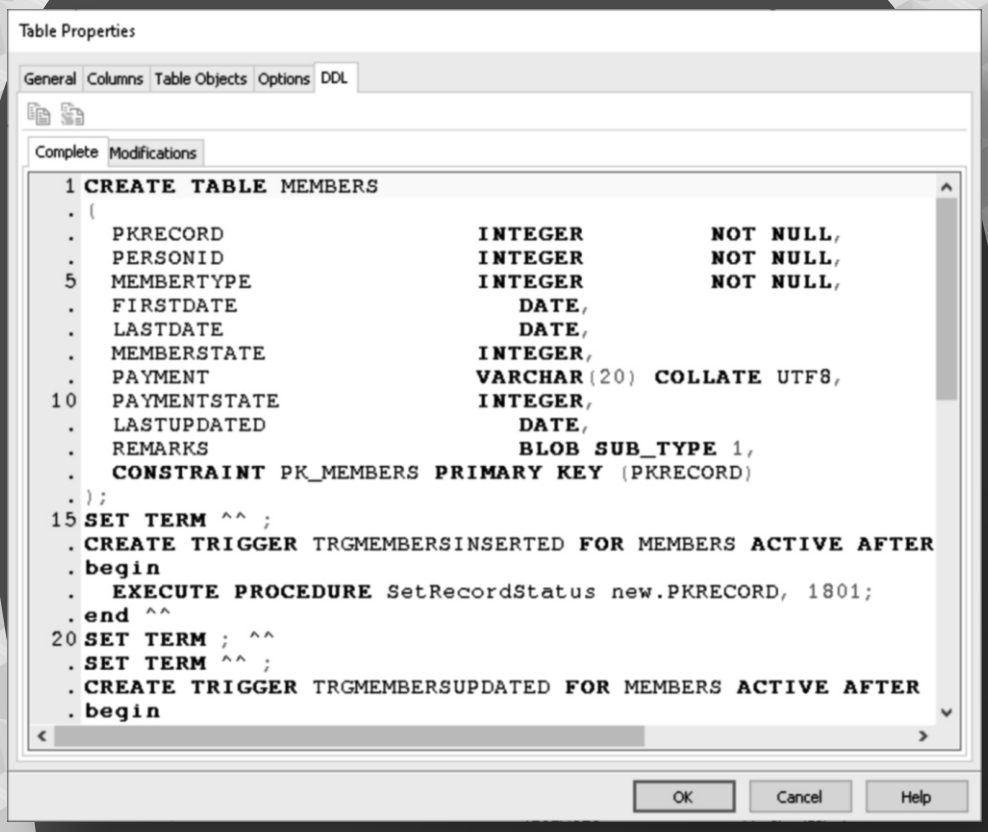


Figure 9: create table

If supported by the database system, there's a tab that shows object dependencies. Each object can have entries in a **to-do list inside Database Workbench.** Depending on the object type, other tabs can be available, like data (*for tables and views*) or stored procedures results.

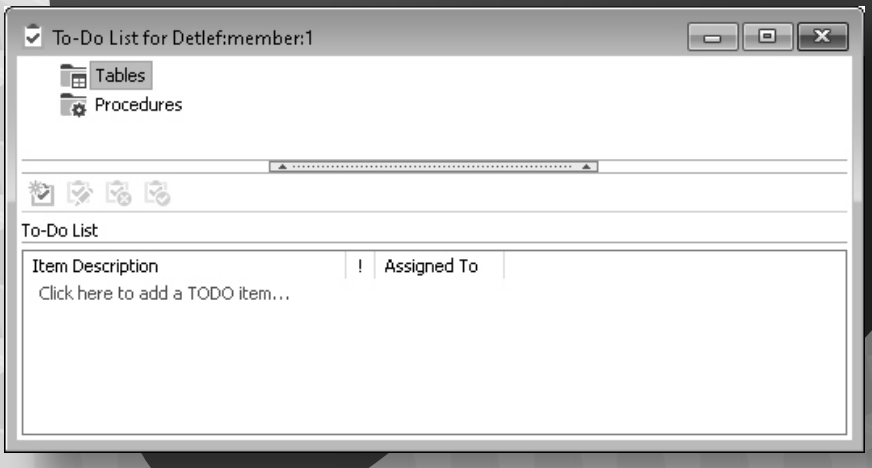


Figure 10: create a to-do

With these object editors, you can create or modify your database objects in a user interface that looks alike for the different database systems.
There's no need to remember exact data types, the exact possible options or the exact database system specific SQL syntax. Create, click and save.
And there you go, you've got your table, your constraint and your index.

Figure 11:
Crete the fieldnames and then create table

MODERN CODE EDITORS

All code editors in Database Workbench support modern editor features: code insight, code staples, code collapse and code templates.

You can also open an object editor for a specific object right from the code and system tables have different colors.

The code insight is called **"SQL Insight"** when it completes your code and **"Parameter Insight"** when displaying hints when calling routines.

SQL Insight is code context sensitive.

When writing **SQL,** a list of available tables is only useful in a particular context, eg the `FROM` clause of your statement. The same goes for columns, not useful in a `FROM` clause, but it is in a `SELECT` clause. Additionally, **SQL** Insight offers "join completion", when writing the `JOIN` clause, it will use database metadata for foreign key constraints to help you finish your **SQL** statement.

Code Insight is available in all code editors, from normal `SQL`, to **Triggers** or **Stored Procedures**. When entering a function name and opening parenthesis, **Parameter Insight** will kick in and show the parameter names and types, this is available for database system built in functions and for stored procedures or stored functions.

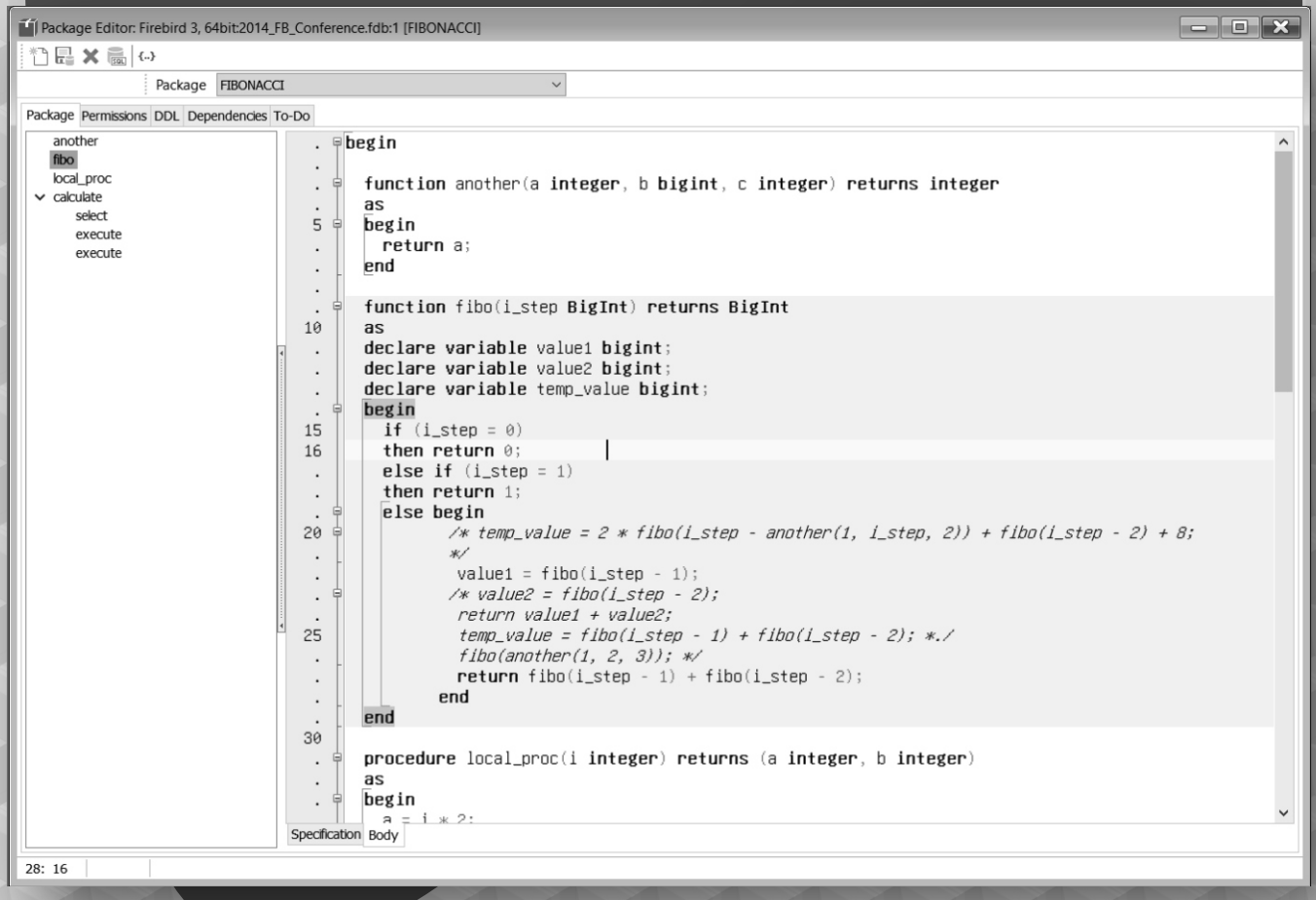


Figure 12: table DDL (code collapse, staples, highlighting visible)

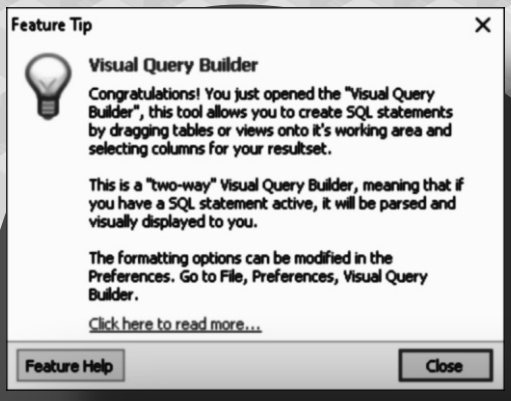
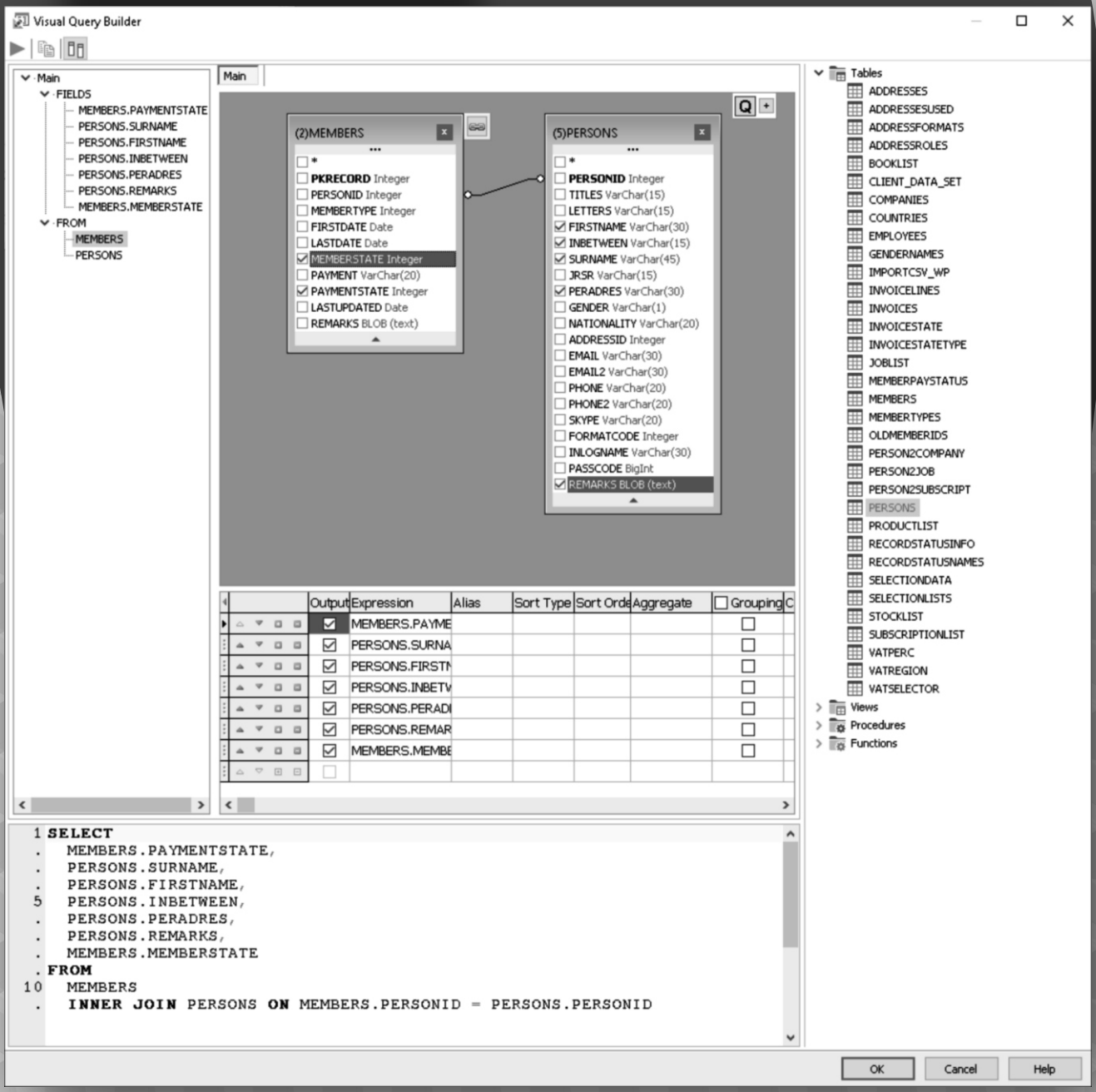


Figure 13: creating through visual Qery Builder



In stored code editors, code staples and code collapse is available for easier code navigation and with packages, which can include multiple functions and procedures, a package navigator helps you to go to specific routines more easily.

With the proper data, you are now able to test your queries, views and stored procedures. Testing stored procedure code can be a tedious task because of the lack of feedback while running the code on the database server.
Oracle and PostgreSQL support a debugger API that can be used to provide step-by-step debugging. Database Workbench includes a visual debugger for these database systems.
But there's more, for InterBase, Firebird and MySQL, Database Workbench supports a debugger using emulation of the stored routine code and it works quite well.

TESTING AND DEBUGGING

After design and additional development comes testing and debugging.
Obviously, a new database needs data to do so. Database Workbench includes tools to import and export data, either via an intermediate file or directly using the Data Pump.
Alternatively, a simple test-data-generator is included. You can easily create fake data including dates, addresses, e-mail, people or company names.

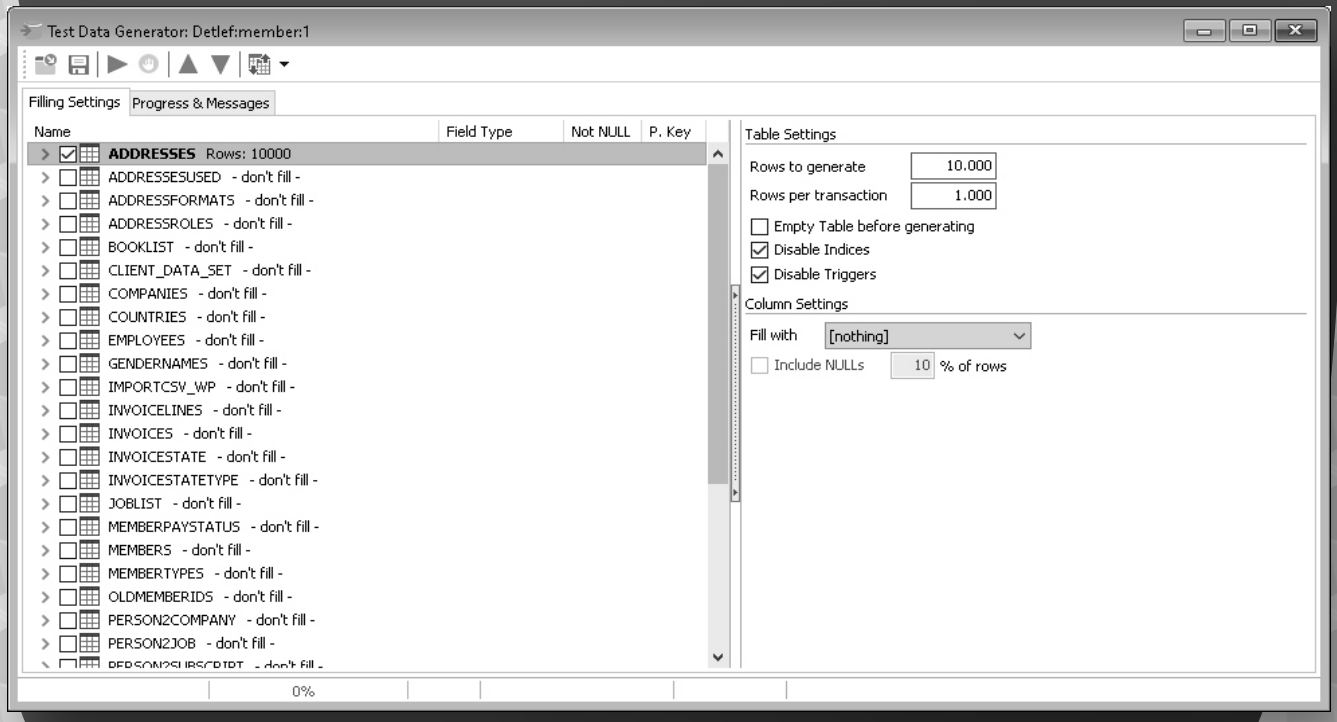


Figure 14: What and how to be tested

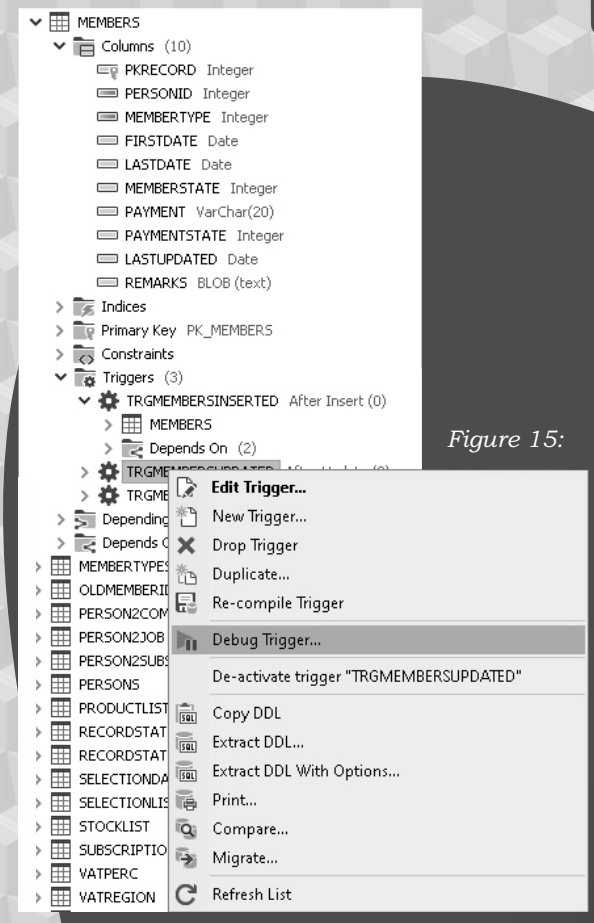
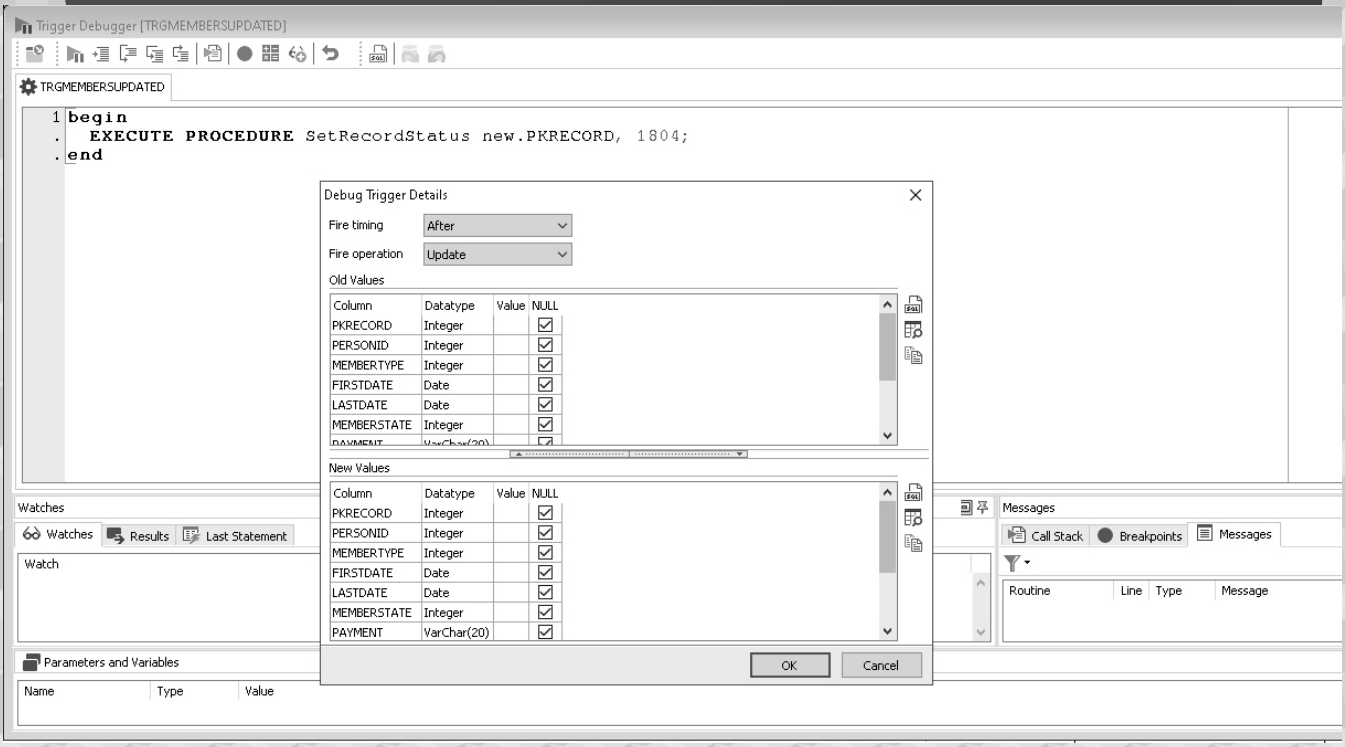


Figure 15:

With the debugger you can seecan the statements in the stored routine code one by one and follow your code logic.
You can check variables for their values, modify those values and continue the routine.
You can set breakpoints for the debugger to pause on and then continue from there.

The availability of a stored routine debugger is a big help when developing database applications that want to use code running inside the database server.



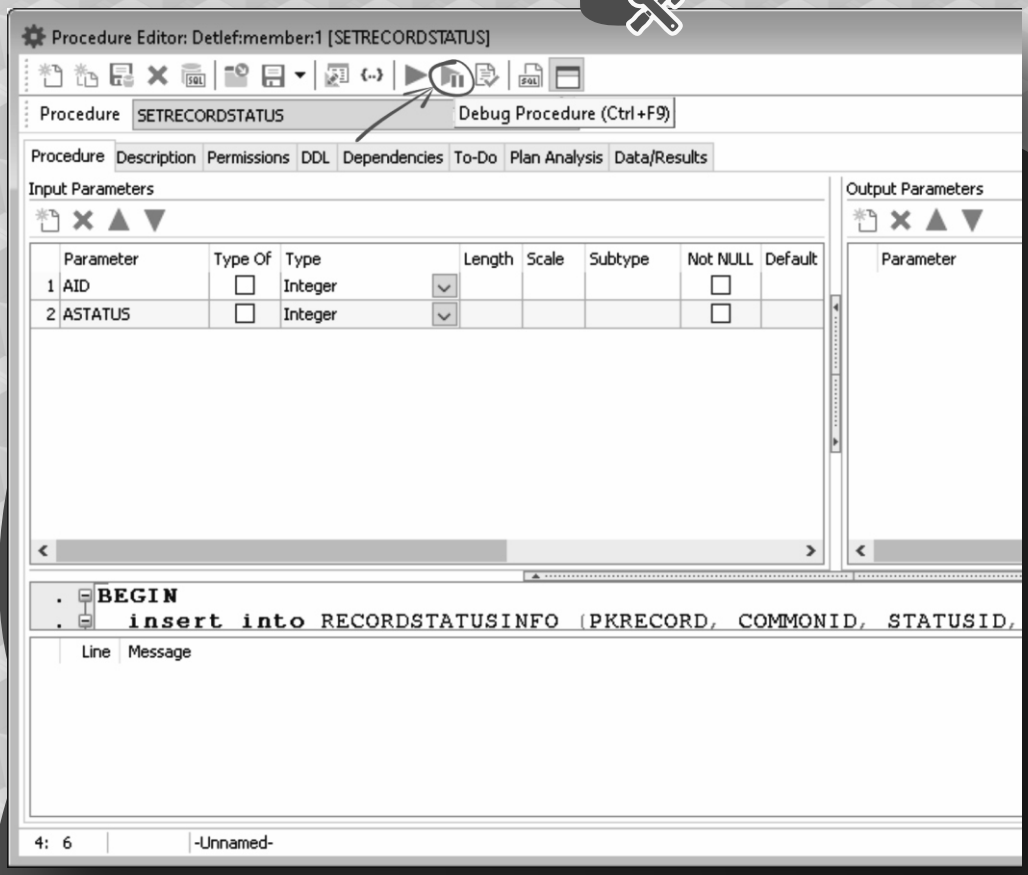


Figure 16: here you find the button to debug your procedure

PRODUCTIVITY TOOLS

Database Workbench offers several productivity tools. Each database you use has a **Workspace** available in which you can create folders with a custom set of database objects, for example to group objects that are related, especially useful for databases with a large number of tables and other items. You can also design database specific printable reports and store them in the Workspace, so you can easily activate and print the reports later.

The **SQL Editor** has a history page that holds a number of your last performed SQL statements. This is very useful and avoids having to enter the same SQL statement in multiple work sessions. Alternatively, you can store SQL snippets re-usable code in the Catalogs.

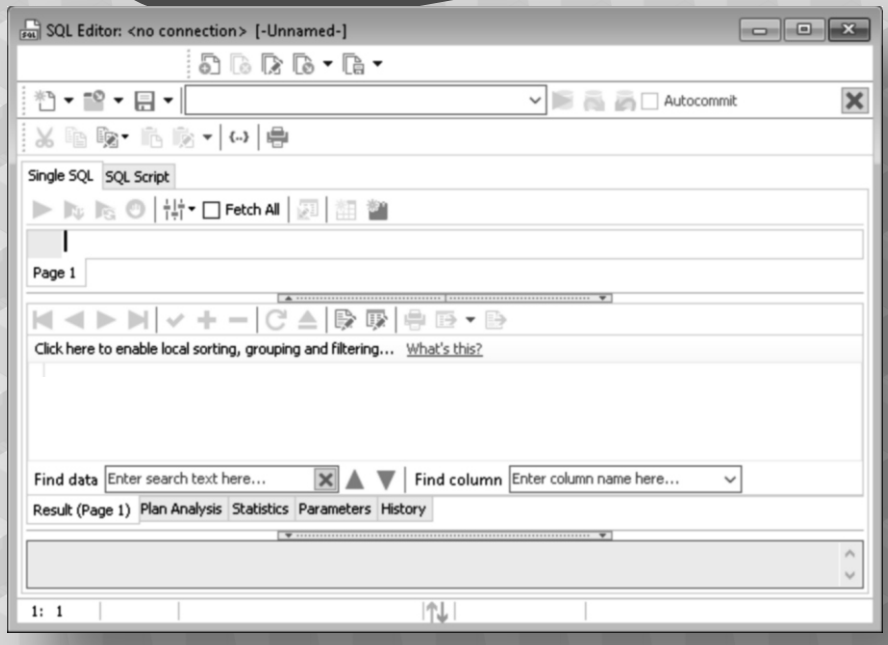


Figure 18:

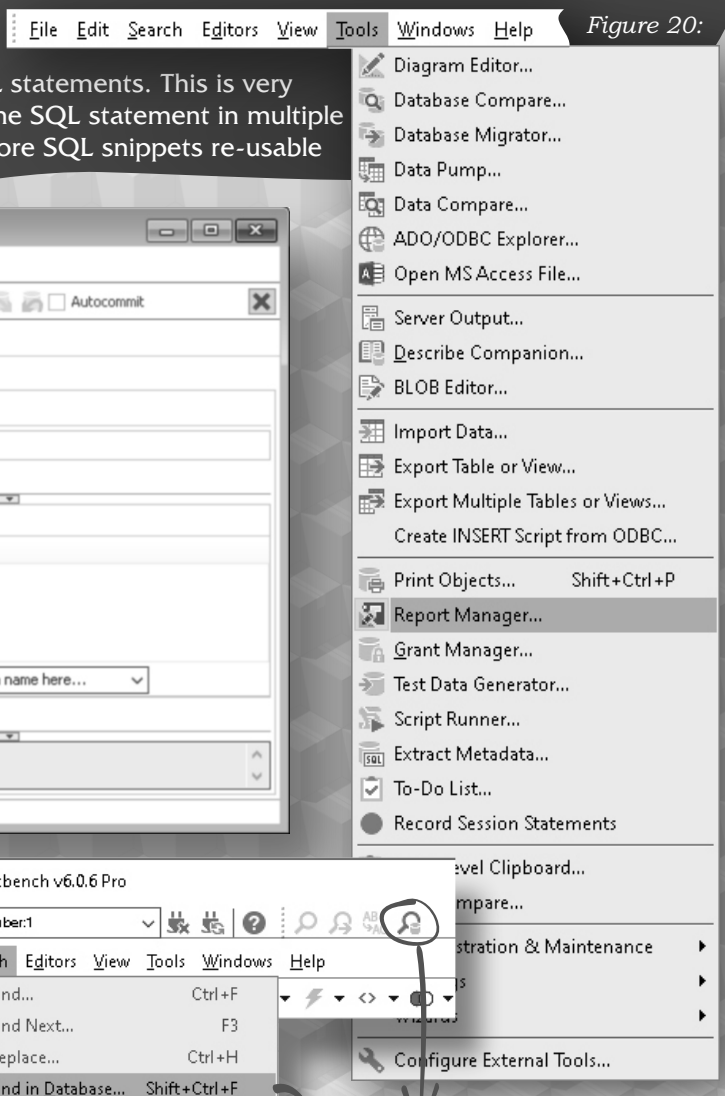


Figure 20:

Figure 19:

With the **Search Database tool** you can search object names, table, index and constraint column names, and the source code of objects.
This is a powerful tool to check to see where an object is used, or a specific piece of code can be found, especially if your database system doesn't support dependency tracking.

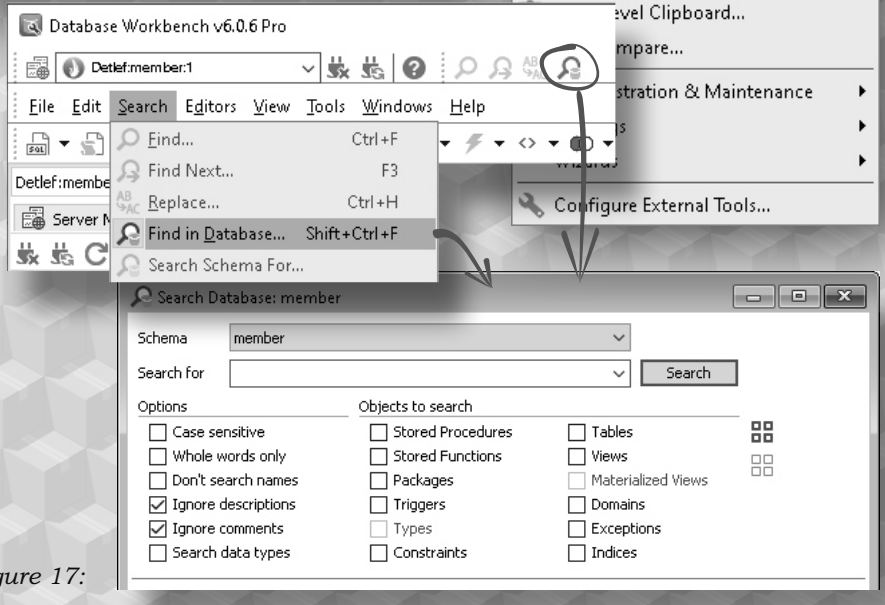


Figure 17:

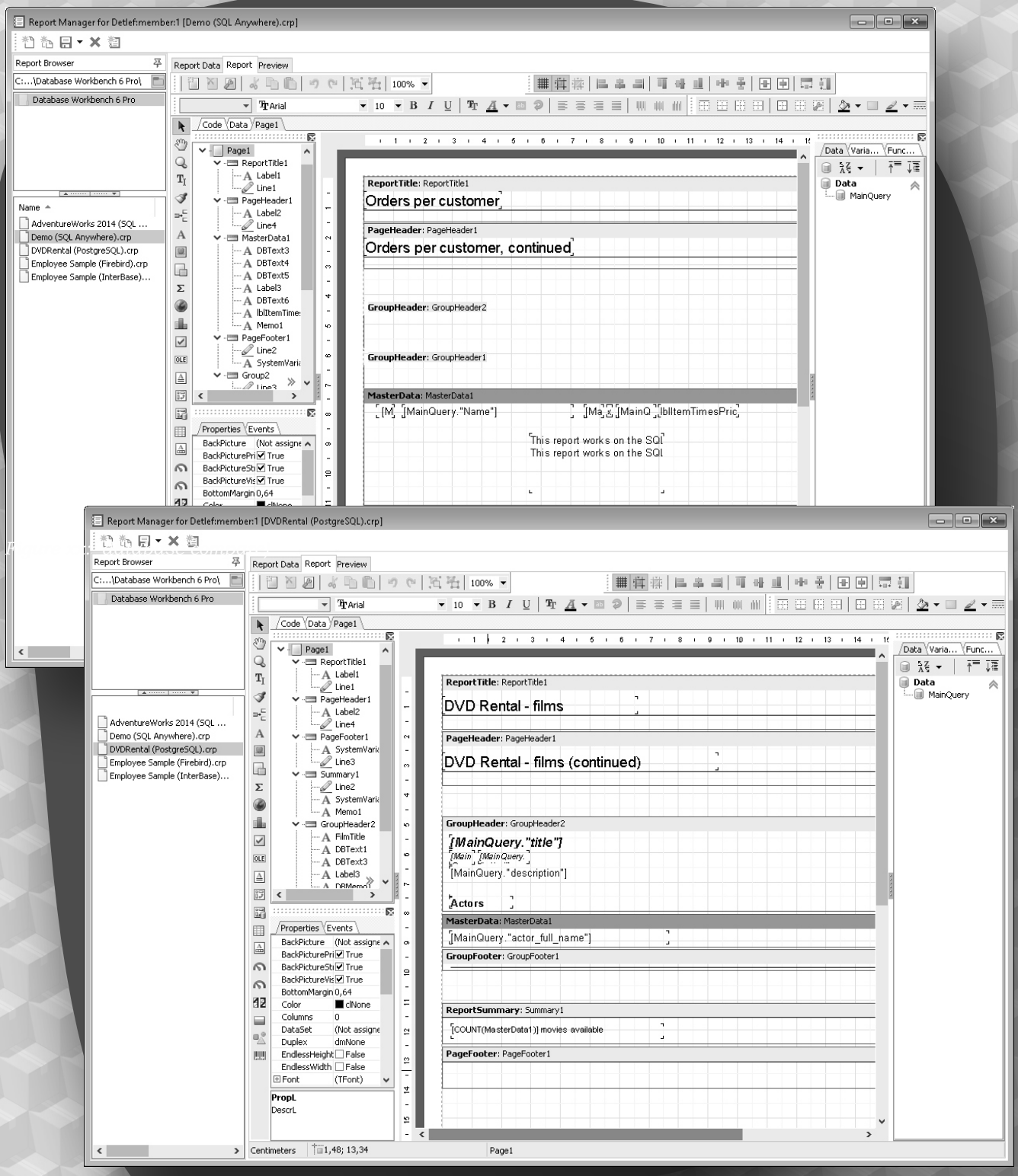


Figure 21: Create your own reports....

NEW:
THE ENTERPRISE EDITION WITH ITS TEAMSERVER

By default, **Basic, Pro** and **Enterprise** store your registered servers, databases and other configuration items in your Windows profile.

The Pro edition allows you to store this in the roaming profile, so you'll have the same set of servers and databases on other Windows machines you use to do your work. The Enterprise edition adds the **TeamServer** service, running on an external machine. Besides your personal set of servers and databases, this adds a centralized storage for shared servers and databases for all developers in your team.

For shared database, the Workspace can include shared folders as well. These are available besides your personal database specific Workspace folders and are the same for all developers using this database. This includes team based to-do lists. TeamServer also stores your workspaces, includes team based to-do lists and adds additional SQL and Code Catalogs, available for all connected users.

You van even add databases to the **TeamServer Version Control System (VCS)** via the Workspace. Once you've done this, you can add database objects to the VCS, and team members can't just modify these database objects: they are protected. You have to 'check out' an objects, make your changes and then check it back in. The SQL for the changes will be recorded automatically, so there's a record of all changes between versions of the database. You can also compare the current database version against recorded versions in the VCS.

CROSS-DATABASE DEVELOPMENT

Although lots of people use **Database Workbench** with support for a **single database** system, it really shines when you use **multiple brands of databases**. These applications offer a consistent user interface so you don't have to get used to multiple, vendor specific, applications for each database system.

But there is more. **Database Workbench** includes specific cross-database tools, like the **Database Compare** and **Database Migrator.** The **Database Compare** allows you to **compare meta data from one database to another,** even if these are on different types of servers. For example, if your applications can use either a **Firebird** or **MySQL** database and you want to make sure your development database on one systems is in the same state as on the other, you can use the **Database Compare tool** for this. ➜

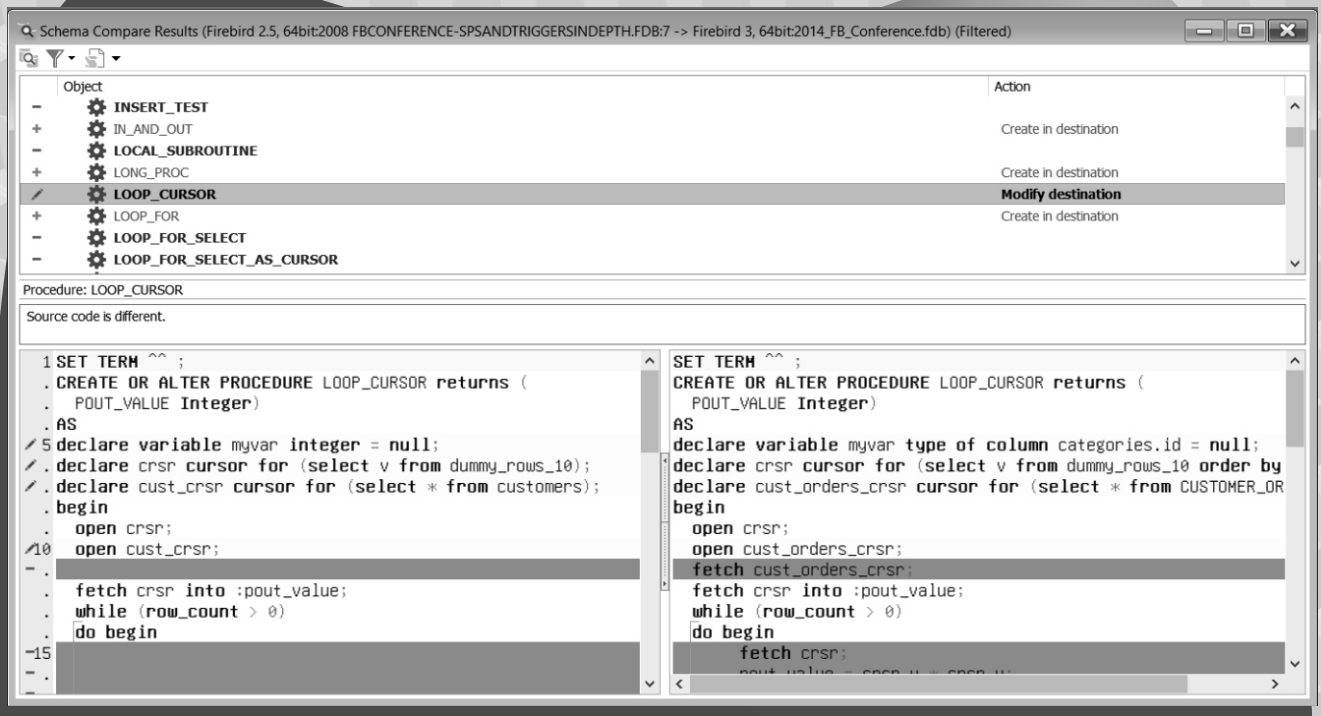


Figure 22: database compare)

Compare tables, constraints, indices and the structure of source code objects like stored procedures and views from one database to another.

Data types will be converted and matched, for example, some database systems support the **SINGLECHAR** data type which, well, is just a single character. **Firebird** doesn't have this as a built in type, so when comparing, this will be converted to **CHAR(1)** and be considered equal.

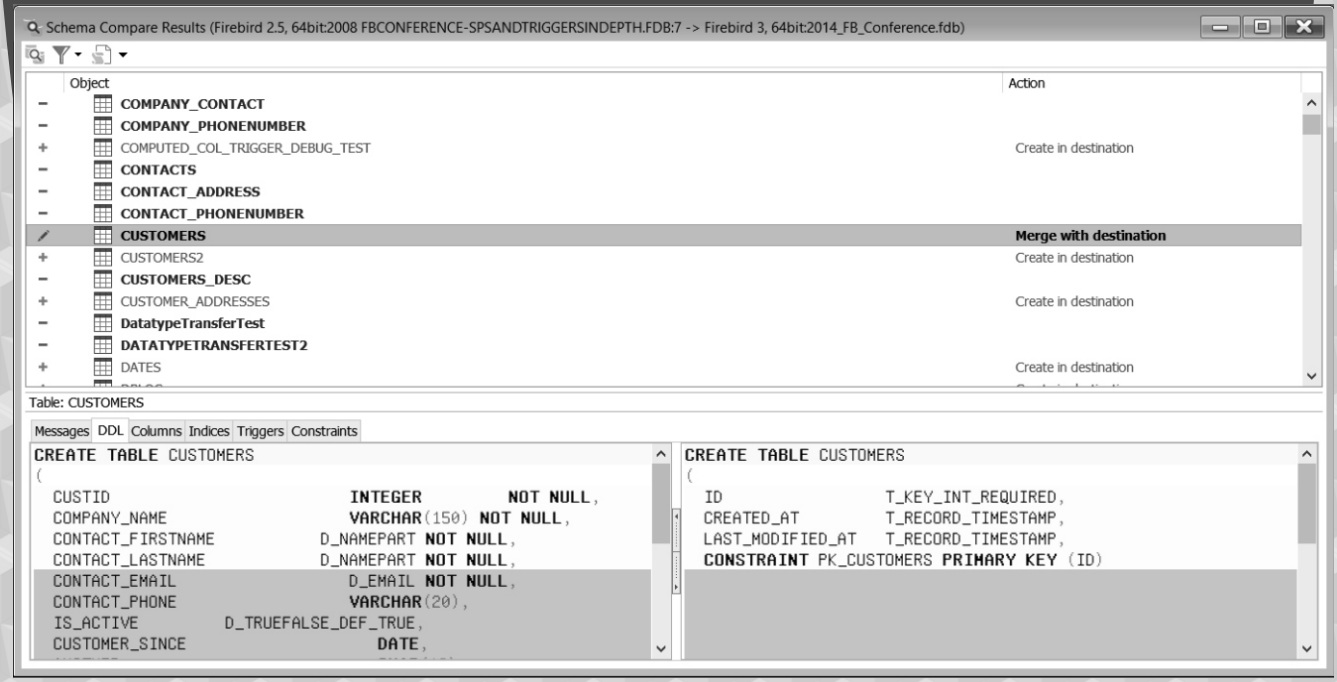


Figure 23: database compare

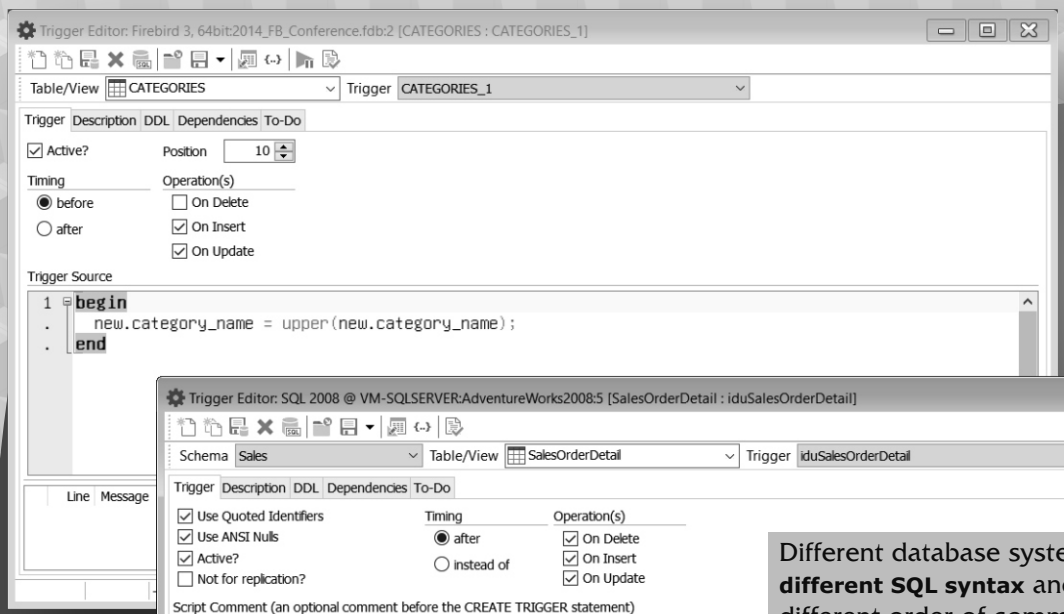


Figure 24

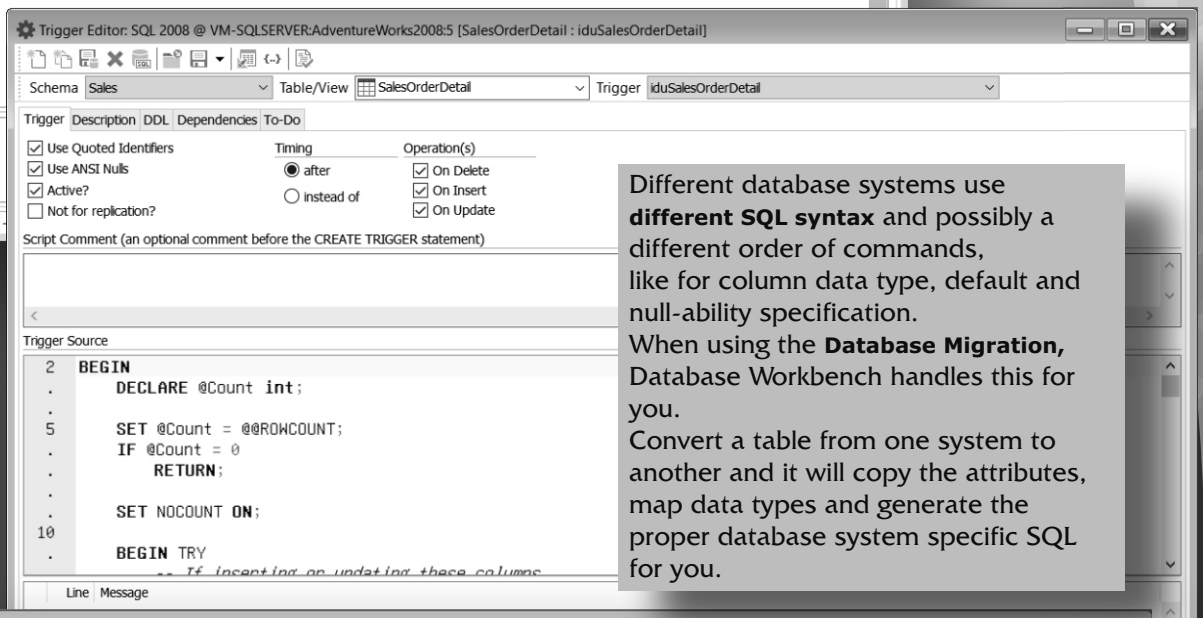
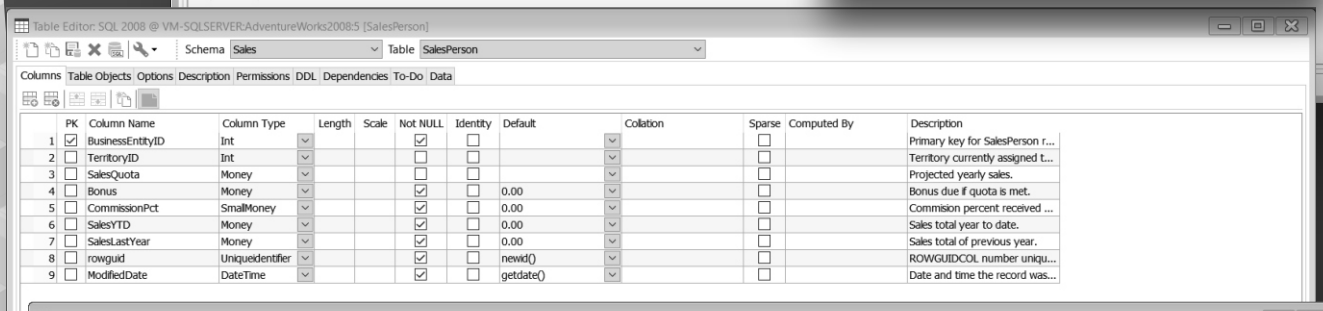
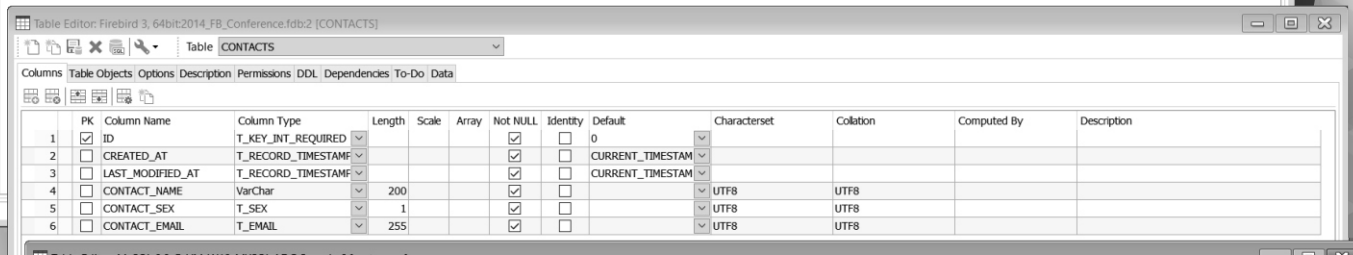


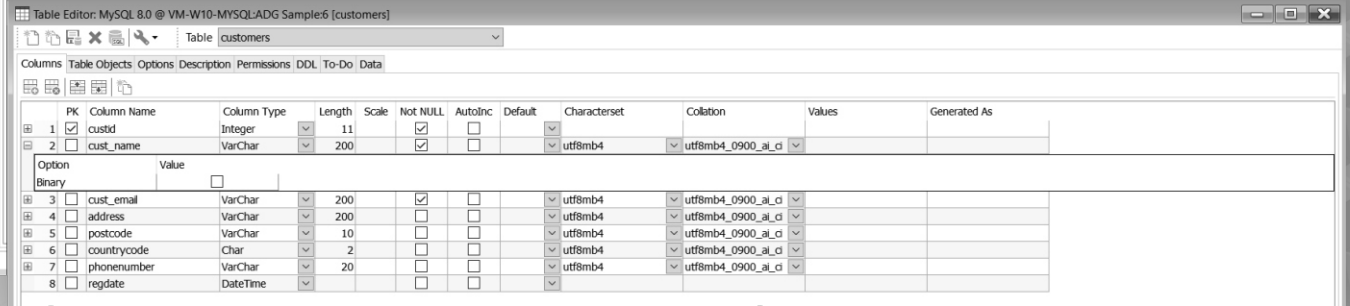
Figure 25:

Different database systems use **different SQL syntax** and possibly a different order of commands, like for column data type, default and null-ability specification.
When using the **Database Migration,** Database Workbench handles this for you.
Convert a table from one system to another and it will copy the attributes, map data types and generate the proper database system specific SQL for you.

Figure 26:







DATABASE DOCUMENTATION

When working on large databases and database applications, documenting your database (meta data) is important. You can do that all with this version.

Reverse engineering your database and arranging sections of it into diagrams can certainly help to make the structure of your database clear. I have tried that several times and that is especially for starters a great help. It takes away a lot of annoyance especially if you are not experienced.

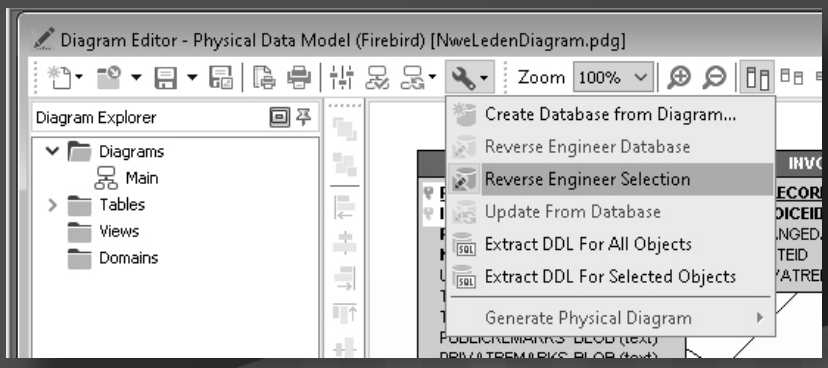


Figure 27:

From one database you can create a main diagram with sub-diagrams all in a single file. Tables can be grouped in sections and foreign key relationships can be made visible easily.

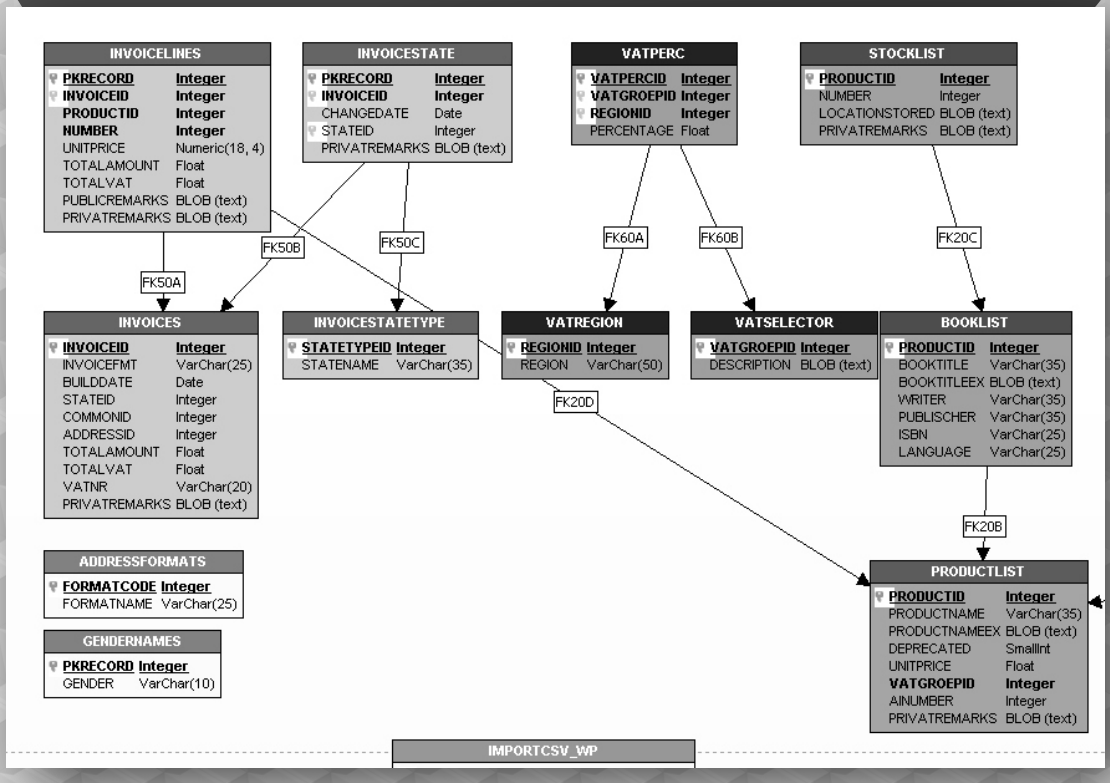


Figure 28: Image diagrams with groups and relationships

Another method of documentation is the ability **to print meta data objects** in lists, like tables with their columns, indices, triggers and constraints. Or stored procedures with parameters and source code. These reports can include object descriptions. This can be useful when checking code or when having to look up objects without access to the database.

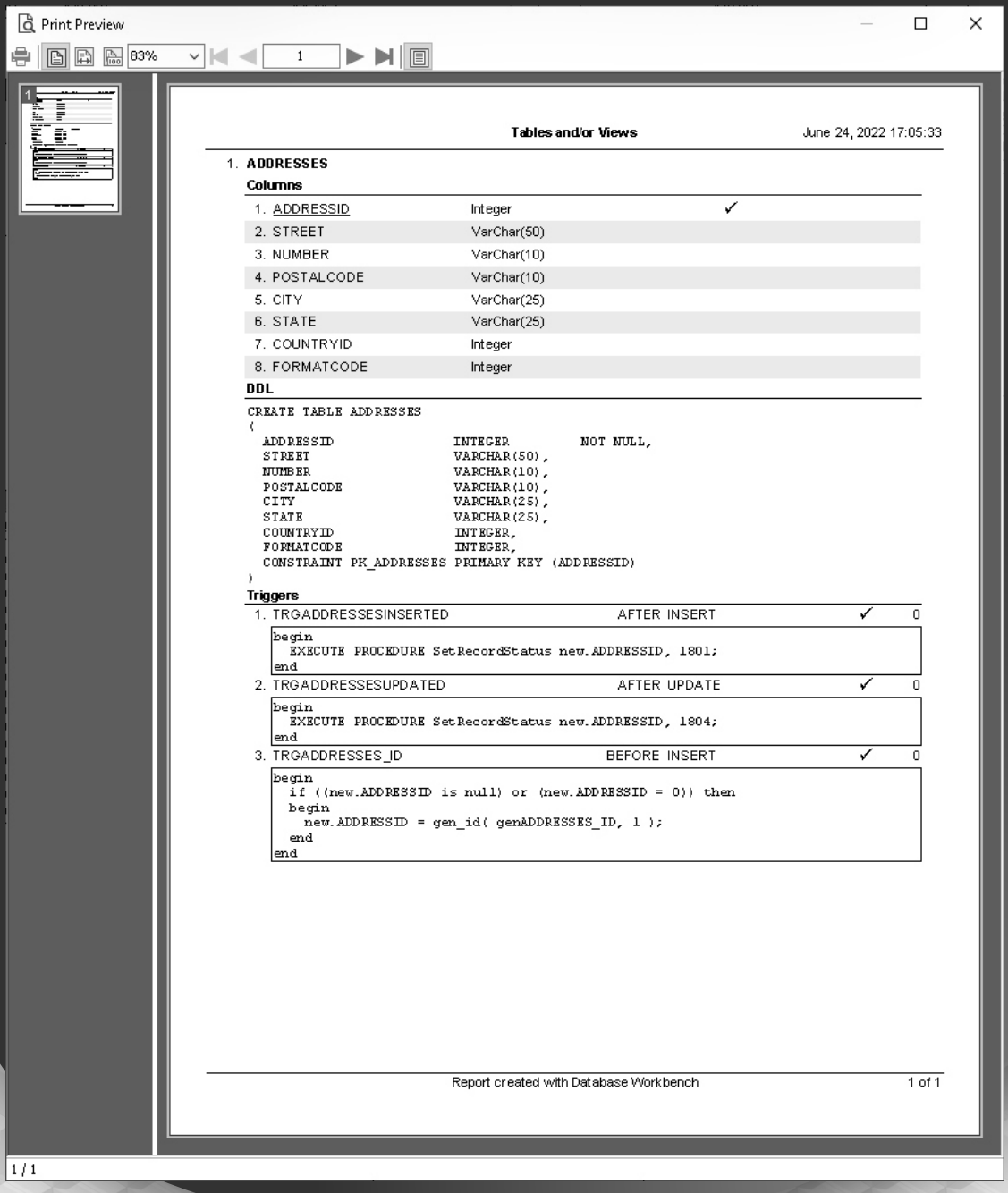


Figure 29:

CONCLUSION

Database Workbench is a complete tool,
created by database developers for database developers.
You can clearly see that a lot of love and craftsmanship has been applied here.
It feels like a second skin.
It has grown from an InterBase specific development tool to an application with support for multiple popular database systems. It is by no means a "generic" tool:
each module includes support for specific features of the database system it was created for.

From start to finish, from design to documentation, from creation to debugging, you do have very much useful tools within **Database Workbench** to make your development a lot easier.

With the **pricing** depending on the database systems you want supported, and the different editions with sets of features, there's a suitable and absolutely affordable version for everyone.
The company is **Upscene** and the **Database Workbench** information can be found at:

www.upscene.com/database_workbench/

I think it is a must-have!

Detlef

barnsten

Delphi & C++Builder are the best development tools on the market to design and develop modern, cross-platform native apps and services. Also for Windows 11! It's easier than ever to create stunning, high performing apps for Windows, macOS, iOS, Android and Linux Server (Linux Server is supported in Delphi Enterprise or higher), using the same native code base. Share visually designed UIs across multiple platforms that make use of native controls and platform behaviors, and leverage powerful and modern languages with enhancements that help you code faster.

Buy directly in the webshop or ask us for a quote.

Delphi 11.1 Alexandria Professional

Delphi

€1.699,00

MOST SOLD!

Delphi 11.1 Alexandria Enterprise

Delphi

€3.999,00

Delphi 11.1 Alexandria Architect

Delphi

€6.499,00

RAD Studio 11.1 Alexandria Professional

RAD

€2.999,00

RAD Studio 11.1 Alexandria Enterprise

RAD

€4.999,00

RAD Studio 11.1 Alexandria Architect

RAD

€6.999,00

* Included in Update Subscription:

https://www.barnsten.com/nl/promotions/

PAS2JS PART 4

ABSTRACT

In this article we explore how to read and write data from a database using **Pas2JS** and **Free Pascal**.

INTRODUCTION

Many websites use data from a database to serve content to users. It can be any kind of data, but most often it will be some SQL-based database: **MySQL, PostgreSQL, MS-SQL server** or any other. However, the browser does not allow direct connections to such a database:
It only supports the **HTTP** protocol.
For this reason, database access must be handled by a server process that resides between the browser and the database.
HTTP does not provide a protocol to access data. So, a protocol that makes use of **HTTP** must be devised to access data.

In the previous articles, *(Issue Nr 101 Page 4)* the **RPC protocol (Remote Procedure Call)** was introduced to execute methods on the server.

This could be used to access data:
introduce some methods to fetch and update data on the server.
The most common protocol for data access these days is not **RPC** based, it is a mechanism called **REST. REST** is not a real protocol.
It is a series of practices that translate to 2 basic rules:

❶ All data is addressable using a **HTTP** URL.
❷ the **HTTP** verbs can be used to get, create, update or delete data.

Most **REST** implementations use **JSON** (See wiki 1)to encode the data for transport over **HTTP.** This is because **JSON** is lightweight and, as a subset of Javascript, it can be handled natively by the browser.
So we have 2 elements in our story so far:
an **SQL database** and a **REST protocol. Free Pascal** uses **SQLDB** to access **SQL** databases. To allow access to SQL databases a series of components exist which, taken together, form the **SQLDBRestBridge** framework.
Basically, **SQLDBRestBridge** forms a bridge between the **SQL** database and the browser, using a **REST** approach and the **SQLDB** components.
It is a set of components that is integrated in the **FCL-Web technology** that exists on the server:
whatever technology you use to embed your server logic in, your **REST server** will function the same.
Once the **SQLDBRestbridge** is set up, the data can be accessed in the browser using a regular `TDataset` descendant.
Updates are handled using the well-known `ApplyUpdates` call.

Representational state transfer (REST) *is a software architectural style that was created to guide the design and development of the architecture for the* **World Wide Web**.

REST *defines a set of constraints for how the architecture of the Web, should behave.*
The **REST architectural style** *emphasises the scalability of interactions between components, uniform interfaces, independent deployment of components, and the creation of a layered architecture to facilitate caching components to reduce user-perceived latency, enforce security, and encapsulate legacy systems.*

REST *has been employed throughout the software industry and is a widely accepted set of guidelines for creating stateless, reliable web APIs. A web API that obeys the REST constraints is informally described as* **RESTful. RESTful** *web APIs are typically loosely based on HTTP methods to access resources via URL-encoded parameters and the use of JSON or XML to transmit data.*

In a **RESTful Web service, r***equests made to a resource's* **URI** *elicit a response with a payload formatted in* **HTML, XML, JSON**, *or some other format.*

For example, the response can confirm that the resource state has been changed. The response can also include hypertext links to related resources. The most common protocol for these requests and responses is **HTTP.**
It provides operations **(HTTP methods)** *such as* `OPTIONS`, `GET`, `POST`, `PUT`, `PATCH` *and* `DELETE`.
By using a stateless protocol and standard operations, **RESTful systems** *aim for fast performance, reliability, and the ability to grow by reusing components that can be managed and updated without affecting the system as a whole, even while it is running.*

The goal of **REST** *is to increase performance, scalability, simplicity, modifiability, visibility, portability, and reliability.*
This is achieved through following **REST** *principles such as a client–server architecture, statelessness, cache-ability, use of a layered system, support for code on demand, and using a uniform interface.*
These principles must be followed for the system to be classified as **RESTful.**

JSON (JavaScript Object Notation) *is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays (or other serializable values). It is a common data format with diverse uses in electronic data interchange, including that of web applications with servers.*

JSON *is a language-independent data format. It was derived from* **JavaScript**, *but many modern programming languages include code to generate and parse* **JSON-format** *data.* **JSON** *filenames use the extension* **.json.**

Douglas Crockford *originally specified the* **JSON** *format in the early 2000s.*
He and **Chip Morningstar** *sent the first* **JSON** *message in April 2001.*

2 SQLDBRESTBRIDGE FEATURES

The **SQLDBRestbridge** system exposes a **REST** interface for one or more databases in a HTTP server. It has the following main features:

1. Connect to multiple databases. Any database type supported by **SQLDB** can be accessed.
2. Have more or one resource schemas.
3. Various in/output formats are supported: **JSON, XML** (*in multiple flavours:* `TClient-Dataset`, **Access format (ADO)**, *and* **custom XML**), **CSV**. It is easy to add your own format.
4. A descriptive Metadata resource is available : all resources are described.
5. Simple URL scheme:
   - `/REST/ResourceName` the list of all resources named ResourceName
   - `/REST/ResourceName/ID` A single resource named ResourceName
   - `/REST/ResourceName?CustomerID=123` the list of all resources named `ResourceName` with property CustomerID equal to 123.
6. Complete configuration of filtering of resource lists. Any field can be configured tobe filtered on, custom filters can be added too.
7. You can control the list of fields in the output using optional query strings. Either specify fields to include or fields to exclude from the output.
8. Paging of results is supported, when possible using the native mechanisms of the SQL database, simulated when needed.
9. CORS support.
10. Authentication can be handled out-of-the box or completely customized. Simple authentication (the username and password are in one of the database tables) can be handled with no code at all, only an SQL statement is needed.
11. Optional: allow the client to specify an SQL statement.
12. Optional: Connections can be managed through a resource as well. This allows you to set up a single general-purpose binary which can be configured to connect to any database at runtime.
13. Optional: reverse engineer a database to expose all tables as REST resources.
14. Optional: Save and load a configuration from an .ini file.

3 SERVER-SIDE SQLDBRESTBRIDGE COMPONENTS

SQLDBRestbridge can be installed in the lazarus IDE using the lazsqldbrest package. Once installed, 4 components appear on the component palette, see figure 1 on page 3. The main component is implemented in the SQLDBRestbridge unit:

Figure 1: The SQLDBRestBridge components on the fpWeb tab

3 SERVER-SIDE SQLDBRESTBRIDGE COMPONENTS

- **`TSQLDBRestDispatcher`**
  This component is responsible for handling the **HTTP** request.
  It takes care of error handling, running queries and transforming data into a transportable format: **JSON, XML, CSV** etc. It holds the Database connection-definitions *(more than one connection is possible)* and has lots of properties that describe what the **REST** interface looks like to the clients.

  `TSQLDBRestSchema`
  This component defines available resources for the dispatcher:
  a resource is essentially a view (*an* **SQL** *select statement*) on a database, together with **SQL** statements that can be used to update the data. The schema can be edited visually in the **IDE** or with a stand-alone program. There are also some methods to import a schema directly from a database.
  The dispatcher can have one or more schemas attached to it
  (*They can be specified in the Schemas property*), and when a client requests a resource, it will look in each of the schemata for the definition of the resource.

- **`TSQLDBRestBusinessProcessor`**
  Although both the `TSQLDBRestDispatcher` and the `TSQLDBRestSchema` components have lots of options to customize and finetune the behaviour of the **REST** bridge, sometimes you will need to handle some cases specially.
  The `TSQLDBRestBusinessProcessor` component lets you implement a set of event handlers that allow you to control the behaviour of the **REST** bridge at the level of a single resource: for instance fine-grained access control, check input etc. It follows that you can have many business processors attached to the **REST** dispatcher: one for each defined resource.

- **`TRestBasicAuthenticator`**
  Is a component that allows you to implement **BASIC http** authentication.
  You can do this with an event handler, or you can set up an **SQL** statement that will return a **user ID** based on a **username/password.**
  All these components have lots of properties. They will not be discussed in detail here, it would lead too far. The properties are discussed in the **FPC Wiki** pages:
  `https://wiki.freepascal.org/SQLDBRestBridge#Available_Components`

4 CLIENT-SIDE SQLDBRESTBRIDGE COMPONENTS

In the browser application, there are 2 components that must be used, they are implemented in the **`sqldbrestdataset`**.

`TSQLDBRestConnection`
Basically, this component specifies the location where the `SQLDBRestBridge` server process is listening. It has some properties that describe the server configuration: the name of the metadata resource, the connections resource name.
Optionally a username and password can be specified.

`TSQLDBRestDataset`
This is an actual dataset which can be used to get the data of a resource.
You must specify the connection, the name of the resource and optionally an ID if you want to get a single resource, or add some filters if you want to get a selection of all resources.

This works mostly as you would expect it to work in a native client/server database application using `TClientDataset`. There are some caveats, as the browser works asynchronously, we will describe this later on.

5 A SAMPLE APPLICATION

To demonstrate the use of `SQLDBRestBridge` components, we will create a small blogging application. It has 3 tables: one for users, one for articles and one for comments. The **SQL** for these tables looks like this:

```
create sequence seq_users;
create table users (
    u_id bigint not null default nextval('seq_users'),
    u_firstname varchar(50) NOT NULL,
    u_lastname varchar(50) NOT NULL,
    u_login varchar(127) not null,
    u_password varchar(127) not null,
    constraint pk_users primary key (u_id)
);
create sequence seq_articles;
create table articles (
    a_id bigint not null default nextval('seq_articles'),
    a_created_on timestamp not null default CURRENT_TIMESTAMP,
    a_author_fk bigint not null,
    a_title varchar(255) not null,
    a_text text not null,
    constraint pk_articles primary key (a_id)
);
create sequence seq_comments;
create table comments (
    c_id bigint not null default nextval('seq_comments'),
    c_created_on timestamp not null default CURRENT_TIMESTAMP,
    c_article_fk bigint not null,
    c_author_fk bigint not null,
    c_text text not null,
    constraint pk_comments primary key (c_id)
)
```

The **SQL** is for a **PostgreSQL** database, but for other databases the **SQL** will look similar, the use of a sequence may vary somewhat.The server will expose a **RPC** service to allow a user to log in and out (*we will reuse the code presented in the previous articles for this*), and a **REST** service using `SQLDBRest-Bridge`.

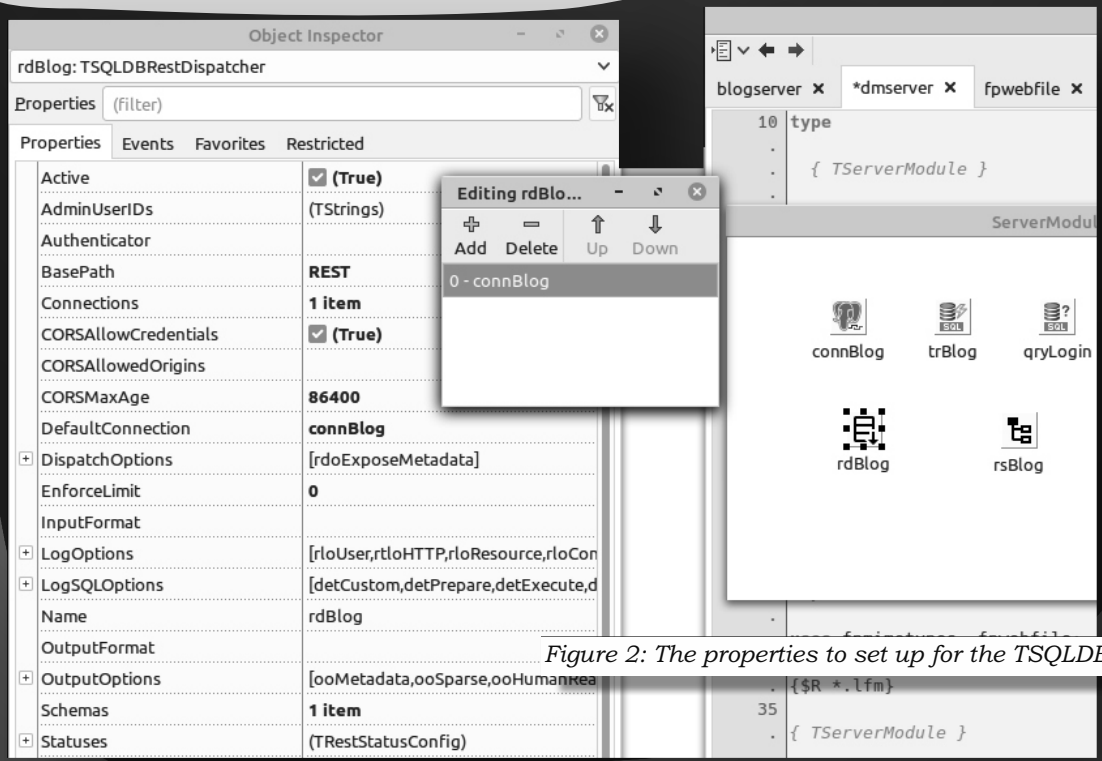


Figure 2: The properties to set up for the TSQLDBRestDispatcher

6 THE SERVER APPLICATION

The server application is a simple HTTP application. It will serve the HTML files to the browser, and host the RPC and REST services. There are 2 modules in the application:

❶ **`ServerModule`** a simple data module with an SQLDB Database connection component, and the **`SQLDBRestBridge`** components.
❷ **`dmRPC`** a simple FPWeb RPC module which offers the RPC API to authenticate a user.

The **RPC** service will not be discussed here, it is essentially the same as presented in the previous articles in this series, but the 2-factor authentication code has been stripped. The **REST** functionality is achieved by dropping 2 components on the ServerModule:

❶ a **`TSQLDBRestDispatcher`** component (*we'll name it* **`rdBlog`**)
❷ a **`TSQLDBRestSchema`** component (*we'll name it* **`rsBlog`**)

In the **`rdBlog`** component, we need to set up several properties (*see figure 2 on page 5*)

`Connections`
this is a property which defines the **SQLDB** connections to the databases you want to expose. Each collection item has a unique name and contains the standard **`DatabaseName`**, **`HostName`**, **`UserName`** and **`Password`** parameters needed to set up a connection. You can also specify an existing connection component in the **`SingleConnection`** property of the collection item: in that case this connection component will be used to connect to the database.

`DefaultConnection` must be set to the default connection.

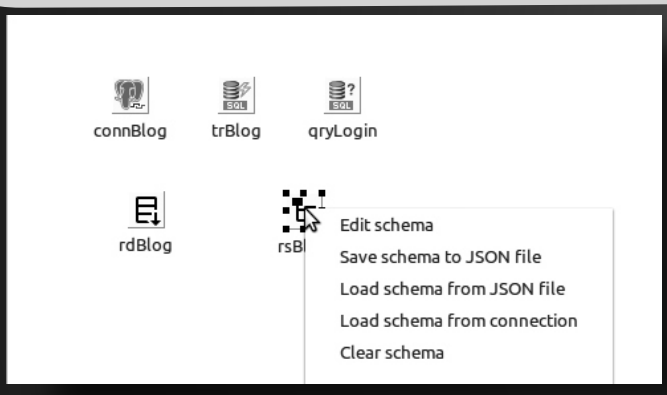


Figure 3:
Starting the schema designer for the TSQLDBRestSchema

`Schemas`
A collection of schemas to serve to clients. Here you specify the schema component **`rsBlog`** which has been dropped on the datamodule.
`BasePath`
This is the first component of the URL which will be checked to determine if a **HTTP** request must be handled by the **`SQLDBRESTBridge`**. The **RPC** path is used to identify **RPC** requests, so **REST** is a logical choice for the **REST** requests.
To set up the schema with the resource definitions, there are 2 possibilities:

❶ Use the stand-alone schema editor: This is a tool that can be used to edit schema files. The tool is distributed with **lazarus,** but you will need to build it yourself; the sources are in the **`components/sqldbrest/editor`** directory. The schema file (*an ordinary JSON file*) can be loaded into the schema component in the designer or at runtime.
❷ Use the component editor menu to edit the schema right in the IDE.

6 THE SERVER APPLICATION

We'll use the latter approach, *see figure 3 on page 6*.
When started, the main form of the designer is divided into three parts.

❶ At the left is a tree view with the available connections and the tables of each connection.

❷ In the middle, a list of defined resources with their fields.
Each resource will be available through the **REST** interface.

❸ At the right, there is a pane which will change depending on the selection in the resources tree. It can be the resource definition (*SQL statements, allowed operations etc.*) or the definition of a field or an overview.

At the top is a menu bar which allows you to **add/edit/delete** connections, resources and fields.

To start a schema, the simplest is to drag and drop a table from the left pane to the middle pane: When you do this, a resource corresponding to the table contents will be defined with all the fields of the table and some default **SQL** statements (*empty, as a matter of fact*) and default properties.

You can see the result when we drag the users table from the `connBlog` connection to the middle pane in figure 4 on page 8. This is in fact what would happen if you allow the `SQLDBRestBridge` to reverse-engineer a database at runtime.

If you check the **SQL** statements for the resource, you will find they are empty:
this is because the `SQLDBRestbridge` will generate the statements at runtime from the table name and the field definitions. It needs of course to know the primary key field, but normally it can detect this from the table definition.

While `SQLDBRestBridge` can cope with primary keys that span multiple fields, it complicates things somewhat, and it is recommended to have a primary key consisting of a single field.

If the defaults are not to your liking, you can adapt the **SQL,** and you can ask the schema editor to generate a default **SQL** statement for you, which you can then modify at will.

Alternatively, you can do everything manually:
Press the **'New resource'** button in the menu bar, specify a resource name and enter a table name and some **SQL** statements.
You can set for each resource what **REST** operations are allowed. It is not recommended to disable the **OPTIONS** operation:
this is used by the **CORS** support of the browser to discover (*using a preflight request*) whether a `POST` request can be sent or not.

Again, the schema editor can generate "Update SQL statements" for you if you specified the table name (*it is not mandatory to specify a table name*).

Once the Select **SQL** statement is created, you can use the '**Update Fields**' button on the **Fields tab** to generate the field definitions. For each field in your resource, you can specify the following things:➜

6 THE SERVER APPLICATION

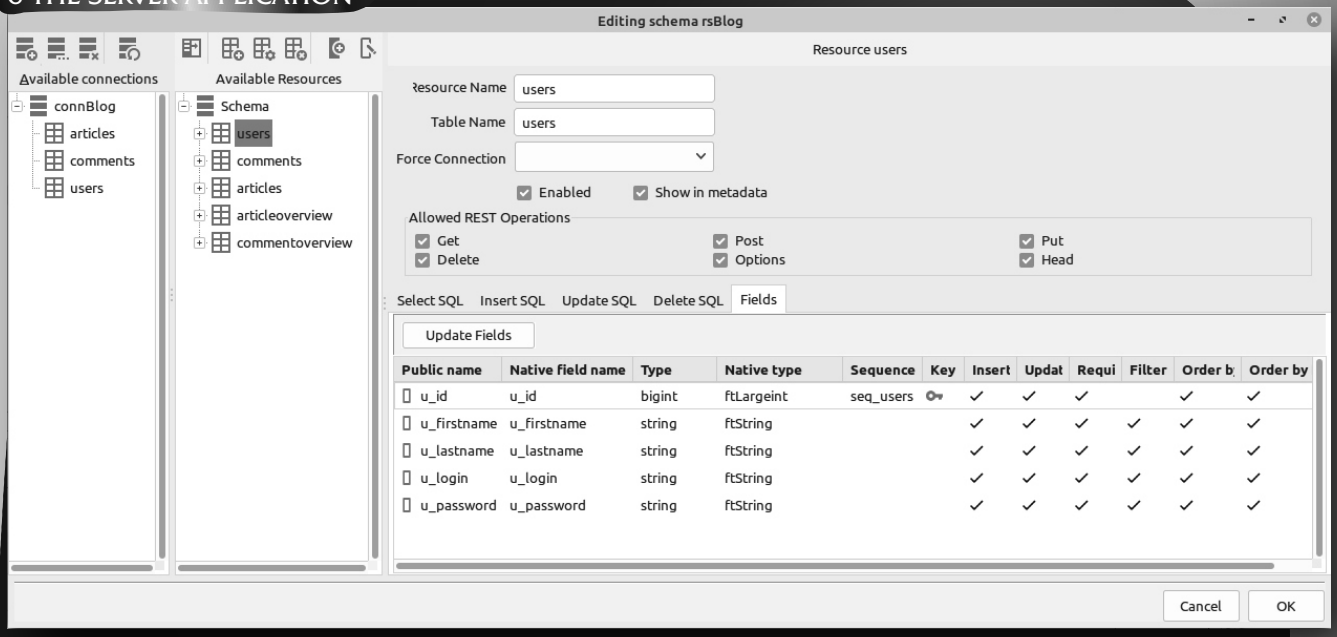


Figure 4: A default schema

❶ The type of field that the client will receive - since the data is transferred over JSON or some other text format, logically the list of available client types is more limited than the available field types in **SQLDB.**

❷ Whether the field must be specified when inserting/updating and if it is required or not.

❸ Whether the field can be used in sorting or filtering operations. The client can request filters and sort order for its result, but for some fields this could result in lengthy operations for the DB server, so it makes sense to disallow it for some fields.

❹ whether the field is part of the primary key.

❺ The name of a database sequence that must be used to generate a field value.

Once the schema is defined, the server is almost ready to go.

If you set the 'Active' property of the `TSQLDBRestDispatcher` component to `True` then the component will register the **HTTP** routes in fpweb's request router system. You can do this also manually with the `RegisterRoutes` method.

In some cases it may be necessary to register the routes manually: the order of route definitions matters and an automatic system such as the form loading mechanism does not necessarily respect the order you expect or need.
As an example, the `OnCreate` event of the `ServerModule` can have the following code:

6 THE SERVER APPLICATION

```
procedure TServerModule.DataModuleCreate(Sender: TObject);
Var
ClientDir : String;
begin
ClientDir:=ExtractFilePath(ParamStr(0))+'../client/';
TSimpleFileModule.BaseDir:=ExpandFileName(ClientDir);
TSimpleFileModule.RegisterDefaultRoute;
TSimpleFileModule.IndexPageName:='index.html';
mimetypes.LoadKnownTypes;
rdBlog.RegisterRoutes;
end;
```

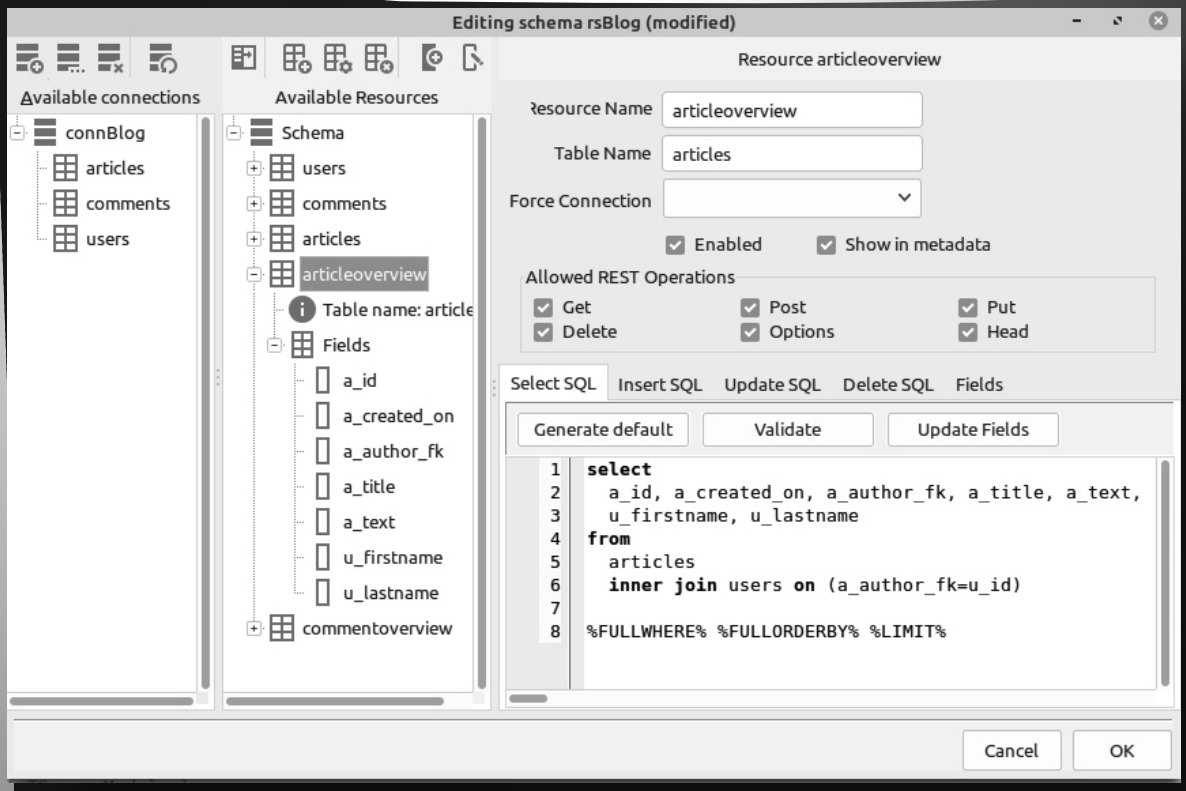


Figure 5: A custom schema

With all this, our REST server is ready to go. It can be tested easily with the browser or the curl or wget command-line utilities, by querying the following URL:

`http://localhost:8080/REST/metadata`

For example with wget you can do:
`wget -q -O - http://localhost:8080/REST/metadata`
If all went well, then you can expect output similar to the one *shown in figure 6 on page 11*.
You can see that there are 2 parts in the **JSON** response: **`metadata`**, which contains the field definitions for the requested resource. **`Data`** contains the actual data for the resource.

The presence of the metadata in a REST response can be controlled by a property of the **`TSQLDBRestDispatcher`** component. You can also test actual data by testing one of the resources: **`http://localhost:8080/REST/articles`**
You can also try to post data with tools like Postman in the browser or the curl and wget command-line tools.

When the server tests work to your satisfaction, it is time to turn to the client application in the browser.

7 THE CLIENT APPLICATION

The pas2js client application to demonstrate the SQLDBRestBridge is a simple site with 3 pages:

❶ A login page. This is essentially the login page presented in the previous articles. On successful login, the list of articles is shown.

❷ An overview page with the list of Blog articles. A logged in user can create a new article.

❸ A detail page with the details of an article. When the user is logged in, (s)he can comment if (s)he is not the author of the article. The author of an article can change the text.

The login page will not be discussed as it is the same as in the previous articles, except to note that:

- In the case of a successful login, the display will switch to the list of articles.
- The 2-Factor authentication has been removed.

In the servermodule (*which already holds the* **RPC Client** *from the Login example*) we now add a `TSQLDBRestConnection`:

```
Property RestConnection : TSQLDBRestConnection Read FRestConnection;
```

And we set it up in the constructor:

```
FRestConnection:=TSQLDBRestConnection.create(Self);
FRestConnection.BaseURL:=ServerURL+'REST/';
```

First we'll start with the article list page. We start obviously by drafting the HTML which will show the articles. We need 4 things:

1. A dropdown to filter on author.
2. A tag with a message to show if no articles are found.
3. A div in which to show the articles.
4. A button to allow a logged-in user to start a new article.

```
<div class="section">
  <div id="divSearch" class="block">
    <div class="field is-horizontal">
      <div class="field-label is-normal">
        <label class="label" for="selAuthor">Select author</label>
      </div>
      <div class="field-body">
        <div class="field">
          <div class="control">
            <div class="select is-link">
              <select id="selAuthor">
              </select>
            </div> <!-- .select -->
          </div> <!-- .control -->
        </div> <!-- .field -->
      </div> <!-- .field-body -->
    </div> <!-- field -->
  </div> <!-- .block -->
<div id="divNoArticles" class="block">
  <div class="notification is-warning is-light">
    Alas, no articles were found...
  </div>
</div>
<div id="divArticleList" "class="block">
</div>
  <div id="divNewArticle" class="block is-hidden">
    <a href="#/article/new" class="button is-primary">New article</a>
  </div>
</div>
```

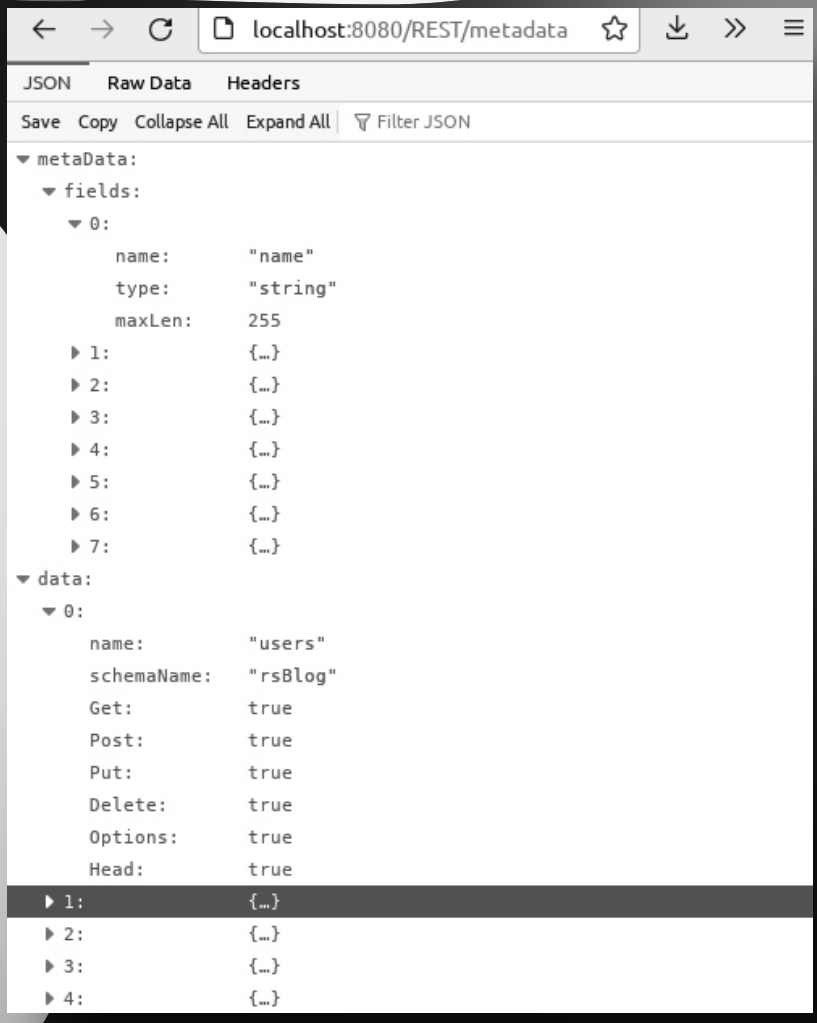


Figure 6: Metadata returned by the server

The above is not a lot of **HTML,** but it is sufficient to do what is needed.
With the **IDE** wizard to create a 'form' class from this **HTML** we can now create a form definition, which we'll name **`TArticleListForm`**, and which will be a descendent of **`TBaseForm`** so we can register a route.
We need to override the **`FormRoutes`** and **`FormHTMLName`** to match the URL we want and the **`articles.html`** filename from which we generated the class:

```
class function TArticleListForm.FormRoutes: TStringDynArray;
begin
  Result:=['/articles/'];
end;

class function TArticleListForm.FormHTMLFileName: String;
begin
  Result:='articles.html';
end;
```

In the constructor we call **`SetupDatasets`** which - not surprisingly - will set up the datasets we need in this form. We need 2 datasets: one for the authors (**`rdsAuthors`**) to fill the dropdown with **`authors`**, and one for displaying the blog articles (**`rdsArticles`**).

7 THE CLIENT APPLICATION

Setup of these datasets is straightforward:

```
procedure TArticleListForm.SetupDatasets;
begin
  rdsAuthors:=TSQLDBRestDataset.Create(Self);
  With rdsAuthors do
  begin
    Connection:=Server.RestConnection;
    ResourceName:='users';
    AfterOpen:=@DoOpenAuthors;
    Load;
  end;
```

The `Connection` and ResourceName are set up, and an `AfterOpen` event is set up. The last step is calling `Load`. This will fetch data from the server, and when the data is returned, it will set it up and then call `Open`.

Why not do this using the Open call and instead introduce a new call Load? Loading data from the server is an asynchronous process, and by putting this in a separate call, this is made explicit.

The `Open` method of `TDataset` is expected to work synchronously: when it returns, the data is expected to be there, enabling code flow like this:

```
With aDataset do
  begin
    Open;
    While not EOF do
    begin
      // Do stuff
      Next;
    end;
  end;
```

Because loading data works asynchronously, this flow would not be possible if the implementation fetched data in the `Open`.

Note that the `Open` method is still functional in the **Pas2JS** `TDataset`: For a `JSONDataset` you can fetch data using another mechanism, set the data in the `Rows` property and call `Open`.

The second dataset fetches the articles:

```
rdsArticles:=TSQLDBRestDataset.Create(Self);
With rdsArticles do
  begin
    Connection:=Server.RestConnection;
    ResourceName:='articleoverview';
    AfterOpen:=@DoOpenArticles;
    Params.AddParam('sort',true).AsString:='a_created_on desc
    Params.AddParam('a_author_fk',False);
  end;
  dsArticles:=TDatasource.Create(Self);
  dsArticles.Dataset:=rdsArticles;
  rdsArticles.Load;
  end;
```

Here a little more is done: a datasource is attached to the dataset - it will become clear why shortly - and 2 parameters are set up.
The parameters of a `TSQLDBRestDataset` are attached to the URL when the data is fetched: The above will result in an URL like this:

```
http://localhost:8080/REST/articleoverview?sort=a_createdon%20desc
```

Note that the parameter `a_author_fk` is disabled and is not visible in the **URL.**
When the parameter `a_author_fk` is enabled and set to value 3, the **URL** will look like this:

```
http://localhost:8080/REST/articleoverview?sort=a_createdon%20desc&a_author_fk=3
```

The `DoOpenAuthors` event handler is called when the authors have been fetched and the dataset is opened. In it, we will fill the dropdown with a list of authors the user can choose from.

This routine simply loops over the available records and constructs the **HTML** needed to fill the `SELECT` html element. It uses a template which is filled with a `Format` statement.

```
procedure TArticleListForm.DoOpenAuthors(DataSet: TDataSet);
Const
  ItemTemplate = '<option value="%d">%s</option>'+sLineBreak;
var
  aHTML,aAuthor : String;
  FID,FFirstName,fLastname : TField;
begin
  aHTML:=Format(ItemTemplate,[-1,'All authors']);
  FID:=Dataset.FieldByName('u_id');
  FFirstName:=Dataset.FieldByName('u_firstname');
  FLastName:=Dataset.FieldByName('u_lastname');
  While not Dataset.EOF do
    begin
      aAuthor:=FFirstName.AsString+' '+fLastname.AsString;
      if (FID.AsInteger=Server.UserInfo.ID) then
      aAuthor:='You ('+aAuthor+')';
      aHTML:=aHTML+Format(ItemTemplate,[FID.asInteger,aAuthor]);
      Dataset.Next;
    end;
  selAuthor.InnerHTML:=aHTML;
end;
```

There is little magical about this routine.
If you prefer, you can of course also create the **HTML** elements in code:

```
procedure TArticleListForm.DoOpenAuthors(DataSet: TDataSet);
var  aAuthor : String;
     FID,FFirstName,fLastname : TField;
     Function MakeOption(aID : NativeInt;
                const aName : String) : TJSHTMLOptionElement;
  begin
    Result:=TJSHTMLOptionElement(Document.createElement('option'));
    Result.innerText:=aName;
    Result.value:=IntToStr(aID);
  end;
begin
  selAuthor.InnerHTML:='';
  SelAuthor.append(MakeOption(-1,'All authors'));
  FID:=Dataset.FieldByName('u_id');
  FFirstName:=Dataset.FieldByName('u_firstname');
  FLastName:=Dataset.FieldByName('u_lastname');
  While not Dataset.EOF do
    begin
      aAuthor:=FFirstName.AsString+' '+fLastname.AsString;
      if (FID.AsInteger=Server.UserInfo.ID) then
      aAuthor:='You ('+aAuthor+')';
      selAuthor.append(MakeOption(FID.asInteger,aAuthor));
      Dataset.Next;
    end;
end;
```

The routines do not differ much, but the second routine is a little safer, since the compiler will check the statements: all is done using the declared **Javascript HTML** classes.
The list of articles can be constructed in a similar way, but we will do things differently. The dbbwebwidgets unit contains a component (`TDBLoopTemplateWidget`) that, given a datasource, will construct a **HTML** fragment by looping over the records in a dataset and for every record fill a template with the values of the fields, and append it to the html. Basically, it does the loop above. No coding is involved except setting up the component. This is of course why the articles dataset has a datasource attached to it in the code above.
The `TDBLoopTemplateWidget` is set up in the `SetupTemplates` method:

```
procedure TArticleListForm.SetupTemplates;
Const
  articleTemplate =
  '<article class="media">' +
  '  <div class="media-content">' +
  '    <div class="content">' +
  '      <p>' +
  '        <strong>{{u_firstname}} {{u_lastname}}</strong>' +
  '          <small>{{a_created_on}}</small>' +
  '        <br>' +
  '        <a href="#/article/{{a_id}}">' + '
           {{a_title}}' +
  '        </a>' +
  '      </p>' +
  '    </div>' +
  '  </div>' +
  '</article>';
begin
  ltwArticleList:=TDBLoopTemplateWidget.Create(Self);
  ltwArticleList.Datasource:=dsArticles;
  ltwArticleList.ItemTemplate:=articleTemplate;
  ltwArticleList.ParentID:=divArticleList.ID;
end;
```

The method is called from the constructor of the form. Since it is data-aware, it will do its work as soon as the dataset is opened. The template is quite simple, it uses the bulma media classes, which are exactly meant for this kind of content:

```
<article class="media">
  <div class="media-content">
    <div class="content">
      <p>
        <strong>{{u_firstname}} {{u_lastname}}</strong>
          <small>{{a_created_on}}</small>
        <br>
        <a href="#/article/{{a_id}}">
          {{a_title}}
        </a>
      </p>
    </div>
  </div>
</article>
```

The placeholders in double braces {{ }} will be replaced with the contents of the fields from the dataset. There is also an event that can be used to get values for placeholders which are not the name of a field, so basically any content can be inserted in the template.
The result of this can be *seen in figure 7 on page 16.*

When the user selects an author to filter the list of articles, the 'change' event is triggered by the **HTML.** The following code is inserted in the event handler that was automatically generated from the **HTML.**

The code makes clear why we added a disabled parameter to the dataset when creating the **`rdsArticles`** dataset, because at this point we enable the parameter depending on whether an author was selected, or the 'all authors' option was selected (*this option was given the value -1*)

```
procedure TArticleListForm.DoSelectAuthor(Event: TJSEvent);
Var aAuthorID : Integer;
begin
  aAuthorID:=StrToIntDef(selAuthor.value,-1);
  With rdsArticles.ParamByName('a_author_fk') do
    begin
      AsInteger:=aAuthorID;
      Enabled:=(aAuthorID<>-1);
    end;
  rdsArticles.Close;
  rdsArticles.Load;
end;
```

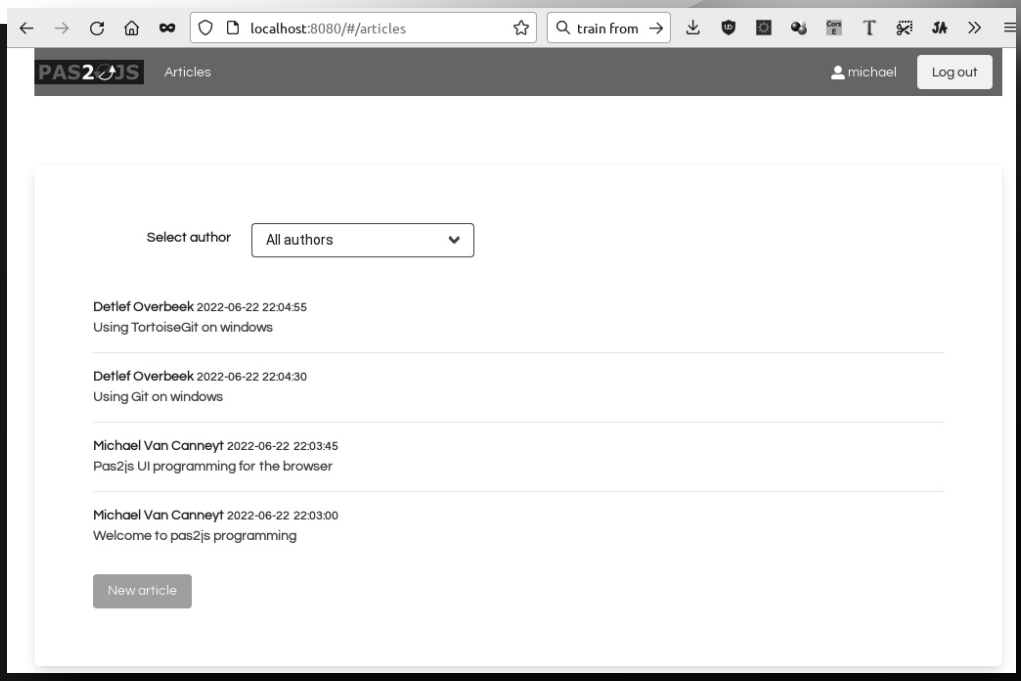


Figure 7: The full article list

After setting the parameters, the dataset is closed and reopened. Since the **`TDBLoopTemplateWidget`** component is data-aware, it will automatically re-render the list of articles.
Finally, in the **`AfterOpen`** event of the articles, we show or hide the 'no articles' message on our page depending on whether the dataset is empty or not:

```
procedure TArticleListForm.DoOpenArticles(DataSet: TDataSet);
begin
  if Dataset.IsEmpty then
    divNoArticles.classList.Remove('is-hidden')
  else
    divNoArticles.classList.Add('is-hidden');
end;
```

The result of this little routine can be *seen in figure 8 on page 16*.
At the bottom of the page there is a button to create a new article.
This button should only be visible when the user is logged in.
This is done in the constructor of the page:

```
constructor TArticleListForm.create(aOwner : TComponent);
begin
  inherited;
  BindElements;
  BindElementEvents;
  SetupDatasets;
  SetupTemplates;
  if Server.UserInfo.ID>0 then
    divNewArticle.classList.remove('is-hidden');
end;
```

Looking at the HTML template for an article, we can see that the article title functions as a hyperlink: when the user clicks on it, the url `#/article/{{a_id}}` is opened.

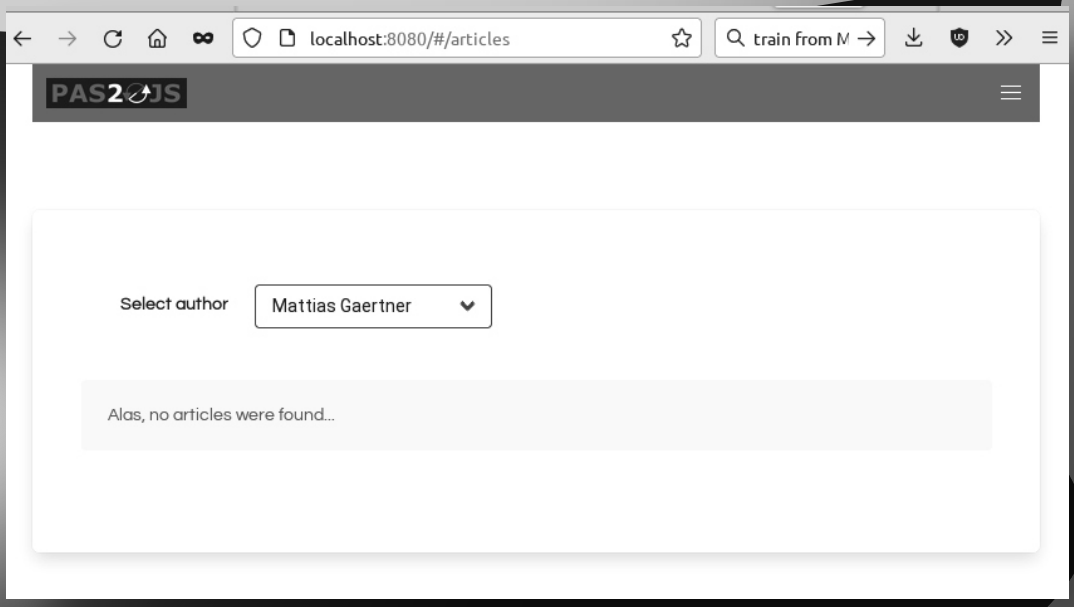


Figure 8: Some people do not like to write blogs

8 EDITING DATA

This is the second page that we add to our application: the article detail page. The page is a little special in the sense that it must handle 2 cases: The logged-in user is the author of the article (*in which case he can edit it*) or the user can only see the article.

There is also the possibility that the asked-for article no longer exists. The page must also handle that situation and deal with it gracefully.

The top of the page contains a block that is shown if the article does not exist. It is initially hidden. Below it is the section that either displays the title and author, or allows to edit the title of the author: which part will be shown is determined in code.

8 EDITING DATA

```
<div class="block is-hidden" id="divNoArticles">
<h1 class="title is-3">Article not found</h1>
<p>You can return to <a href="#/articles">
    the list of articles</a>
    and select another article.</p>
</div>
  <!-- title and date etc. -->
  <div class="block">
  <div id="divTitle" >
    <h1 class="title is-3" id="lblTitle"></h1>
    <h5 class="title is-5">
      <strong id="lblAuthor">Author</strong>
      <small id="lblDate">Date/small>
    </h5>
  </div>
  <div id="divEditTitle"
       class="field is-horizontal is-hidden">
    <div class="field-label is-normal">
      <label class="label" for="edtTitle">Title</label>
    </div>
    <div class="field-body">
      <div class="field">
        <div class="control">
          <input id="edtTitle"
                type="text"
                class="input"
                placeholder="Type your title here">
        </div> <!-- .control -->
      </div> <!-- .field -->
    </div> <!-- .field-body -->
  </div> <!-- field -->
</div> <!-- .block -->
```

After the title, there is a block where the article will be shown, together with a 'Save' button, which is initially hidden. Note that for the article content, there are no separate edit and display blocks. The reason for this will become clear soon.

```
  <div id="divArticleBlock"
       class="block"
       style="min-height: 40vh">
  <div id="divArticle" class="content">
     <h5>This will be a long rant</h5>
     But it will end well...
  </div>
</div> <!-- .block -->

<div class="block">
  <button id="btnSave"
     class="button is-primary is-hidden"
     _click_="HandleSavePost">Save post</button>
</div>
```

The last part of the page is where comments can be shown, and where a logged-in user can add a comment:

```
<div id="divComments">
   <div class="block">
     <h1 class="is-title is-5">Comments</h1>
   </div>
   <div id="divExistingComments" class="block">
   </div>
   <div class="divAddComment">
     <div>
       <div id="editcomment" style="min-height: 10em;"></div>
     </div>
    <div>
     <button id="btnAddComment"
         _click_="HandleSaveComment"
         class="button is-primary">Add comment</button>
    </div>
   </div>
</div>
```

8 EDITING DATA

The list of comments will again be rendered by a **TDBLoopTemplateWidget** component. Again, we use the 'File-New' wizard to create a class definition for this **HTML** file. We start by correcting the formroutes method so it returns a route that has a parameter for the article ID.

```
class function TArticleForm.FormRoutes: TStringDynArray;
begin
  Result:=['/article/:ArticleID'];
end;
```

This route matches the URL that was specified in the article overview. In the constructor, we again add code to set up the datasets and the **TDBLoopTemplateWidget** component:

```
procedure TArticleForm.SetupDatasets;
begin
  rdsArticle:=TSQLDBRestDataset.Create(Self);
  rdsArticle.Connection:=Server.RestConnection;
  rdsArticle.ResourceName:='articles';
  rdsArticle.AfterOpen:=@DoAfterArticleOpen;
  rdsComments:=TSQLDBRestDataset.Create(Self);
  rdsComments.Connection:=Server.RestConnection;
  rdsComments.ResourceName:='commentoverview';
  With rdsComments.Params.AddParam('c_article_fk') do
    begin
      DataType:=ftLargeInt;
      Enabled:=true;
    end;
  dsComments:=TDatasource.Create(Self);
  dsComments.Dataset:=rdsComments;
end;
```

Note that the data is not yet loaded in this routine.
Note also the datasource, needed to render the comments.
The code to set up the comment renderer is very similar to the one for the article list:

```
procedure TArticleForm.SetupTemplates;
Const
  CommentTemplate =
  '  <article class="media">' +
  '    <div class="media-content">' +
  '      <div class="content">' +
  '        <p>' +
  '          <strong>{{u_firstname}} {{u_lastname}}</strong>' +
  '            <small>{{c_created_on}}</small>' +
  '          <br>' +
  '            {{c_text}}' +
  '        </p>' +
  '    </div>' +
  '  </div>' +
  '</article>';
begin
  ltwComments:=TDBLoopTemplateWidget.Create(Self);
  ltwComments.Datasource:=dsComments;
  ltwComments.ItemTemplate:=CommentTemplate;
  ltwComments.ParentID:=divExistingComments.id;
end;
```

When the web router creates the form and shows it, it will call the 'ShowRoute' method of the form. We must override this method and add code to handle the value of the **ArticleID** parameter: After recording the value of the parameter, we must distinguish between the value **New** and a numerical value. In the former case, we must append to the dataset. In the latter case, we must fetch the data for the article with the given **ID.**

8 EDITING DATA

```
procedure TArticleForm.ShowRoute(const aURL: String; aRoute: TRoute;
  aParams: TStrings);
Var
  aID : String;
begin
  Inherited;
  aID:=aParams.Values['ArticleID'];
  if SameText(aID,'New') then
    begin
      FArticleID:=-1;
      rdsArticle.Params.AddParam('a_author_fk',true).asInteger:=-1;
    end
  else
    begin
      FArticleID:=StrToIntDef(aID,-1);
      rdsArticle.ResourceID:=IntToStr(FArticleID);
    end;
  rdsArticle.Load();
  if (FArticleID>0) then
    begin
      rdsComments.Params[0].asLargeInt:=FArticleID;
      rdsComments.Load();
    end
end;
```

When loading an existing article, we can simply set the `ResourceID` to the ID of the requested article. The `ResourceID` value will be appended to the URL. When creating a new article, we must also load the dataset, because during the load operation, the structure of the dataset (which fields to create) is fetched from the server in the metadata property of the returned data.

But we cannot set resourceID to -1 or any other non-existing value, because doing so will result in a **404 ERROR** code: asking for a nonexistent resource results in a **404**, and the dataset will not be opened, instead a load error is reported.
So instead we load using a filter that is guaranteed to give an empty result.
(*there is another approach possible, which will be discussed in a future contribution*)
The last piece of code simply opens the comments dataset, so the comments will be rendered.
The logic of this page continues in the `AfterOpen` event of the article dataset.
Here there are 4 possibilities:

❶ An article was requested but not found.
❷ A new article was requested (and no data is incoming) so we must edit it.
❸ There is data for an article, and we must allow to edit it.
❹ There is data for an article, and we must simply display it.

We distinguish these cases by showing and hiding some parts of the page. We start with the case that the dataset is empty. What to do depends on whether an article is expected (*the FArticleID is positive*) or not: ➜

8 EDITING DATA

```
procedure TArticleForm.DoAfterArticleOpen(DataSet: TDataSet);
Var  IsEdit : Boolean;
     S : String;
begin
  if Dataset.IsEmpty then
    begin
    if FarticleID>0 then
      begin
        HideEl(divArticle);
        HideEl(divTitle);
        HideEl(divEditTitle);
        HideEl(divComments);
        ShowEl(divNoArticles);
      end
    else
      begin
        // Appending, show known data:
        lblAuthor.InnerText:=Server.UserInfo.Login;
        S:=FormatDateTime('yyyy-mm-dd hh:nn',Now);
        lblDate.InnerText:=S;
        edtTitle.value:='';
        divArticle.InnerHTML:='';
        SetupForEdit(True);
        Dataset.Append;
      end;
end
```

The first part of the if statement treats the case when an article was asked, but the server does not have it.
The second part handles the case when we need to append. The `SetupForEdit` takes care of `showing/hiding` the edit controls.
In case there is data, we must decide whether we allow the user to edit:

```
else
  begin
    isEdit:=Dataset.FieldByname('a_author_fk').AsInteger=
                Server.UserInfo.ID;
    SetupForEdit(isEdit);
    ShowData(Dataset);
    if IsEdit then
      Dataset.Edit;
  end;
  With Dataset.FieldByname('a_id') do
    ProviderFlags:=ProviderFlags+[pfInKey];
end;
```

Note that when editing is allowed, the dataset is put in edit or insert mode.
The last statement is needed to set up the key field of the dataset: without this, the client will not know which field is the key field. (*a change is planned in* **SQLDBRESTBridge** *which removes the need for this statement*)
The data is actually shown in the showdata method. It simply copies the data from the dataset to the various html tags:

```
procedure TArticleForm.ShowData(Dataset : TDataset);
var  S : string;
begin
  With Dataset do
    begin
      S:=FieldByName('u_firstname').AsString + ''
          + FieldByName('u_lastname').AsString;
      lblAuthor.InnerText:=S;
      S:=FormatDateTime('yyyy-mm-dd hh:nn',
      FieldByName('a_created_on').AsDateTime);
      lblDate.InnerText:=S;
      S:=FieldByName('a_title').AsString;
      edtTitle.value:=S;
      lblTitle.InnerText:=S;
      S:=FieldByName('a_text').AsString;
      divArticle.InnerHTML:=S;
    end;
end;
```

8 EDITING DATA

The last piece in the puzzle is the **SetupForEdit** routine, which simply shows or hides/disables some elements.

```
procedure TArticleForm.SetupForEdit(IsEditing : Boolean);
begin
  if IsEditing then
    begin
      HideEl(divComments);
      HideEl(divTitle);
      ShowEl(divEditTitle);
      tinyEditor.transformToEditor(divArticle);
      ShowEl(btnSave);
    end
  else
    begin
      HideEl(divEditTitle);
      ShowEl(divComments);
      btnAddComment.disabled:=(Server.UserInfo.ID<=0);
      if not btnAddComment.disabled then
        begin
          editcomment.style.setProperty('min-height','10em;');
          tinyEditor.transformToEditor(editcomment);
        end;
    end;
end;
```

Note the following statement:

```
tinyEditor.transformToEditor(divArticle);
```

This activates the **TinyEditor** Javascript component. This is a small javascript function that transforms a **html** element into an editor with basic editing capabilities such as bulleted lists, headings, indent etc. It can be downloaded from github:

https://github.com/fvilers/tiny-editor

With this, we're all set to display an article, as can be seen in figure 9 on page 22
When editing, the save button must save the data, this is handled in the 'click' event handler of the button. Note that the dataset was already set in edit or insert mode

```
procedure TArticleForm.HandleSavePost(Event: TJSEvent);
begin
  SaveData(rdsArticle);
  rdsArticle.Post;
  rdsArticle.ApplyUpdates;
end;
```

There is nothing special about this: **SaveData** will transfer the contents of the **HTML** tags to the dataset. The **Post** and **APplyUpdates** have the same purpose as in any other dataset. For completeness, we present here **SaveData**

```
procedure TArticleForm.SaveData(Dataset : TDataset);
begin
  With Dataset do
    begin
      FieldByName('a_created_on').AsDateTime:=Now;
      FieldByName('a_text').AsString:=divArticle.InnerHTML;
      FieldByName('a_title').AsString:=edtTitle.Value;
    FieldByName('a_author_fk').AsInteger:=Server.UserInfo.ID;
  end;
end;
```

The tinyeditor component directly edits the inner html of the tag it was assigned to, so the actual **HTML** content is directly available for us.
Similarly, the button to save a comment does the same: ➜

8 EDITING DATA

```
procedure TArticleForm.HandleSaveComment(Event: TJSEvent);
begin
   With rdsComments do begin
     Append;
     FieldByName('c_text').AsString:=editcomment.innerHTML;
     FieldByName('c_createdon').AsDatetime:=Now;
     FieldByName('c_author_fk').AsDateTime:=Server.UserInfo.ID;
     Post;
     ApplyUpdates;
   end;
end;
```

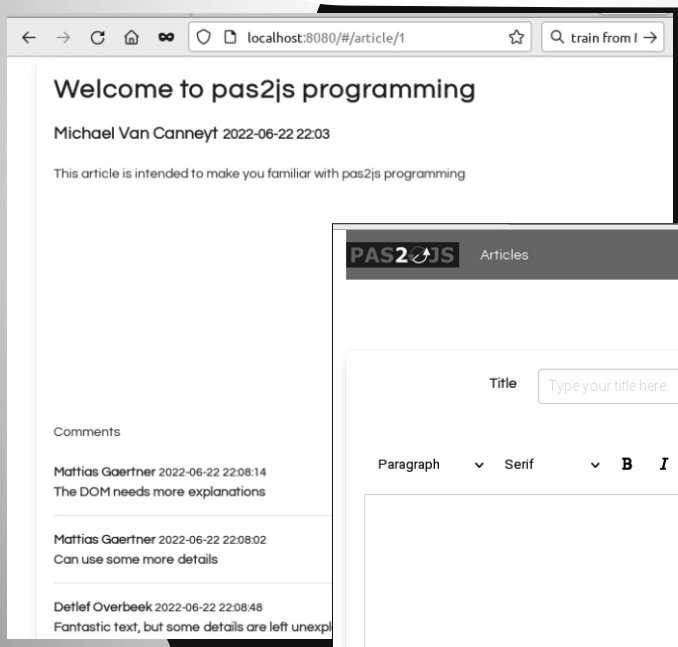


Figure 9: A displayed article and comments

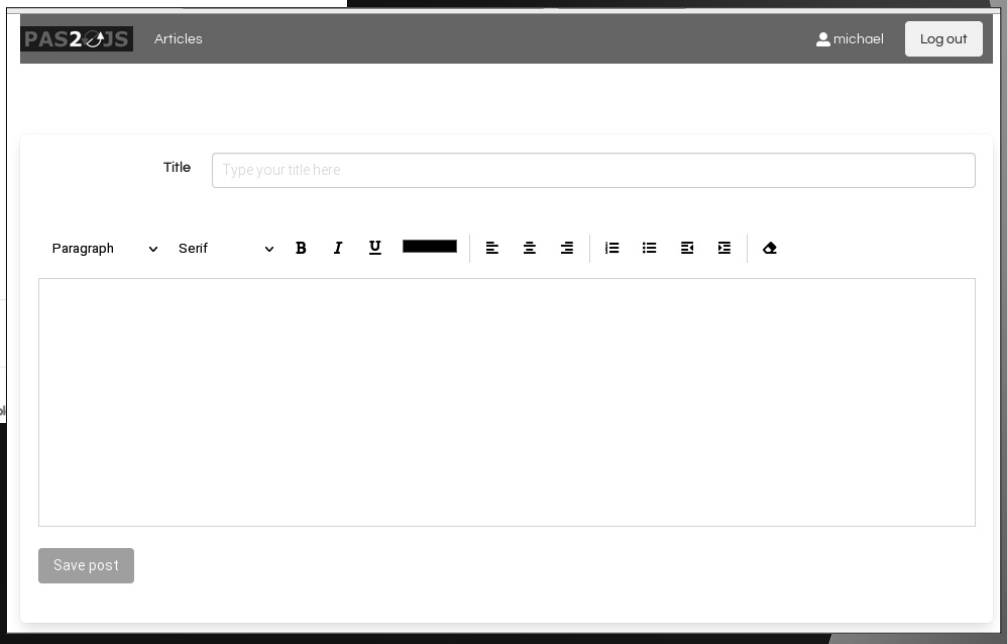


Figure 10: An article in edit mode

And that's it. An article in edit mode is shown in figure 10 on page 22

9 CONCLUSION

In this article we've shown that displaying and editing data is not different in a **Pas2JS** application than it is in a classic 3-tier desktop application. We didn't pay a lot of attention to error handling and did not handle security at all.

Also, in this article, we've done everything manually: Create the dataset and other components, transfer data between **HTML** and dataset. However, **Pas2JS** is progressing and we can do better: drag and drop of components, data-aware components that automatically transfer data between data and **HTML** elements. This is the subject of a new contribution.

ABSTRACT

Since some weeks, support for **Anonymous functions** has been introduced in **Free Pascal.** In this article, we'll take a look at this new and long-awaited feature.

❶ INTRODUCTION

Delphi knows anonymous functions since quite some time now. It is one area in which the language compatibility of Free Pascal with Delphi was lacking.
Scooter Software sponsored the development of this language feature.
A programmer with the name **Blaise** (*coincidence ?*) developed it for them.
It has been available for their private use since some time, and last year, they shared their code with the **Free Pascal** team.
Sven Barth took it upon himself to integrate and adapt the code for the main compiler branch.
Some weeks back, he announced general availability of this feature, much to the joy of many users.

With the exception of some small corner cases, the workings of **Anonymous functions** in **Free Pascal** are the same as in Delphi. But **Free Pascal** offers some extra functionalities, which we'll describe below.

❷ WHAT ARE ANONYMOUS FUNCTIONS ANYWAY ?

Well, as the name implies, **anonymous functions** are functions that do not have a name. There is of course more to it than that, and we'll examine the consequences of this feature in detail.
In pascal, all methods and procedures are called by name, like this:

```
MyInterestingFunction(10,20);
```

Here, `MyInterestingFunction` is the name of a function which is defined prior to this piece of code.
But if an **anonymous function** does not have a name, how can you call it?
The answer is simple: by assigning it to a procedural type variable and executing this procedural variable. The above example could be rewritten as follows with a procedural type since the days of ordinary pascal:

```
Type
    TMyProcedureType = procedure(X,y : integer);
var
   P : TMyProcedureType;
begin
  P:=@MyInterestingFunction;
  P(10,20);
end;
```

The effect of this code is the same.
The advantage is that you can assign different functions (for instance various color interpolation functions) to a variable of procedural type. Since it is a variable, you can also pass around this function variable to other functions.

As you can see, the variable is assigned using the name of the function. And here is where anonymous functions come into play: what if you could, instead of the name, simply type the declaration of the function you wish to assign ?

Anonymous functions allow just this.
You can now type:

```
{$mode objfpc}
{$modeswitch anonymousfunctions}

Type
TMyProcedureType = procedure(x,y : integer);

procedure doit;
var  doPlus : Boolean;
  P : TMyProcedureType = procedure (x,y : integer)
begin
  if DoPlus then
    writeln(X,'+',Y,'=',X+Y)
  else
    writeln(X,'-',Y,'=',X-Y)
end;

begin
  for doPlus:=False to True do
    P(10,20);
end;

begin
  Doit;
end.
```

```
{$mode objfpc}
{$modeswitch anonymousfunctions}
Type
      TMyProcedureType = procedure(x,y : integer);
var
  P : TMyProcedureType;
begin
  P:=procedure (x,y : integer)
begin
  writeln(X,'+',Y,'=',X+Y);
end;
  P(10,20);
end.
```

There are 2 things to note about this program:

❶ The anonymousfunctions modeswitch.
This enables the use of anonymous functions.
❷ This code will not compile in Delphi.
Delphi requires a special procedural variable type for this to work. But Free Pascal allows the above code to work because the anonymous function does not reference any symbols from outside the function scope.

Additionally, there are 3 things to note about the anonymous function:
❶ There is no name
- which was the whole point, of course.
❷ There is no semicolon after the procedure or function header:
the header is followed by the function body.
❸ The anonymous function must appear in the statements of the surrounding block.

The last point means it is not possible to do this:

So it is not possible to initialize a variable with an anonymous function. The assignment must happen in the statement block.

Pascal has always known local procedures:
in local procedures, you can access symbols of the surrounding **`function`**,
as shown in the following example, where the variable **`doplus`** declared outside of **`DoPrint`** is accessed inside **`DoPrint`**:

```
procedure DoIt;
var
  doPlus : Boolean;

procedure DoPrint (x,y : integer);
begin
  if DoPlus then
     writeln(X,'+',Y,'=',X+Y)
  else
     writeln(X,'-',Y,'=',X-Y)
end;

begin
  for doPlus:=False to True do
  DoPrint(10,20);
end;

begin
  DoIt;
end.
```

This works, and will print (as expected) the following:
10-20=-10
10+20=30
But if we want to do this with an anonymous function like in the below:

```
{$mode objfpc}
{$modeswitch anonymousfunctions}

Type
    TMyProcedureType = procedure(x,y : integer);

procedure DoIt;
var
  P : TMyProcedureType;
  doPlus : Boolean;
begin
  P:=procedure (x,y : integer)
    begin
      if DoPlus then
        writeln(X,'+',Y,'=',X+Y)
      else
        writeln(X,'-',Y,'=',X-Y)
    end;

  for doPlus:=False to True do
    P(10,20);
end;

begin
  DoIt;
end.
```

Then the Free Pascal compiler will complain:

```
ex5.pp(13,6) Error: Incompatible types:
got "anonymous procedure(LongInt;LongInt);"
expected "<procedure variable type of
procedure(LongInt;LongInt);Register>"
ex5.pp(26,4) Fatal: There were 1 errors compiling module, stopping
```

Which is strange, because it was possible to use the variable `doplus` in the local procedure, so why not in the anonymous function?
The reason for this failure to compile is that we wish to assign the procedure to a variable.
The local procedure `DoPrint` also cannot be assigned to a variable:

```
{$mode objfpc}

Type
   TMyProcedureType = procedure(x,y : integer);

procedure DoIt;
var
   P : TMyProcedureType;
   doPlus : Boolean;

   Procedure DoPrint(x,y : integer);
   begin
      if DoPlus then
        writeln(X,'+',Y,'=',X+Y)
      else
        writeln(X,'-',Y,'=',X-Y)
   end;
begin
   P:=@DoPrint;
   for doPlus:=False to True do
      P(10,20);
end;

begin
   DoIt;
end.
```

The compiler will complain:

```
ex6.pp(23,6) Error: Incompatible types:
got
"<address of procedure(LongInt;LongInt) is nested;Register>"
expected
"<procedure variable type of procedure(LongInt;LongInt);Register>"
ex6.pp(31) Fatal: There were 1 errors compiling module, stopping
Fatal: Compilation aborted
Error: /usr/local/bin/ppcx64 returned an error exitcode
```

That is because a procedural variable is - behind the scenes - simply a pointer to the address of the function. But in the above case, this pointer is not sufficient to allow calling the function: the function needs to know the address of the `doPlus` variable to do its work, and this information cannot be present in simply one pointer.

In the example above, the compiler could probably still decide that the variable `P` lives only in the procedure `DoIt`, and allow it anyway. In general, the function variable could be passed on to other routines or in some other manner still be 'alive' when `DoIt` exits. This reason is similar to the reason why you cannot assign a normal procedure to an event handler that is declared with the '`Of Object`': the information about the object instance would be missing.

❸ CLOSURES: FUNCTION REFERENCES

This brings us to closures and function references. We speak of a closure when a function is defined and used together with some elements from its environment: in the above case the variable `DoPlus` is needed for the functioning of the anonymous function.
So, how to solve the above? How can we use DoPlus from the environment in our anonymous function?

The answer is a function reference. A new type of procedural variable is now possible:

```
{$ModeSwitch functionreferences}
Type
    TMyProcedureType = reference to procedure(x,y : integer);
```

NOTE that you need a mode switch for the compiler to accept this definition in `ObjFPC` mode. When using a function reference, a 'Reference to procedure' type of variable, the compiler knows that it must be prepared to capture the environment of the function assigned to the procedural variable.

You can declare variables (or fields) directly as usual:

```
var
  P : reference to procedure(x,y : integer);
```

NOTE: THIS IS NOT POSSIBLE IN DELPHI. in Delphi you need to define a type (*see later in this article*).

Generics will of course also work:

```
Type
    Generic TMyProcedureType<t> = reference to procedure(x,y : T);
```

Armed with this new type, we now can change our program to:

```
{$mode objfpc}
{$modeswitch anonymousfunctions}
{$ModeSwitch functionreferences}

Type
  TMyProcedureType = reference to procedure(x,y : integer);

procedure doit;
var
  P : TMyProcedureType;
  doPlus : Boolean;
begin
  P:=procedure (x,y : integer)
  begin
    if DoPlus then
      writeln(X,'+',Y,'=',X+Y)
`   else
      writeln(X,'-',Y,'=',X-Y)
  end;
    for doPlus:=False to True do
      P(10,20);
end;

begin
    Doit;
```

CLOSURES: FUNCTION REFERENCES

And now it will compile and run.
Whenever **P** is used, the compiler will capture the environment (*in this case the variable DoPlus*) and pass it on to the called routine.
Note the 2 modeswitches at the start of the program:

```
{$modeswitch anonymousfunctions}
{$ModeSwitch functionreferences}
```

We are using 2 different features:
anonymous functions, and function references.
To demonstrate that this is actually so, here is the same program, using a local procedure, no anonymous function:

```
{$mode objfpc}
{$modeswitch functionreferences}

Type
     TMyProcedureType = reference to procedure(x,y : integer);
procedure doit;
var
  P : TMyProcedureType;
  doPlus : Boolean;

  Procedure DoPrint(x,y : integer);
  begin
    if DoPlus then
      writeln(X,'+',Y,'=',X+Y)
    else
      writeln(X,'-',Y,'=',X-Y)
  end;

begin
P:=@DoPrint;
for doPlus:=False to True do
P(10,20);
end;

begin
  Doit;
end.
```

You can assign a local procedure to a 'reference to procedure'.
Note that this code is not portable, as this construct is *(unfortunately)* not possible in **Delphi.**

In the above examples, the variables **P** and **DoPlus** are still used within the scope of the procedure **DoIt**.
So, to show that they are actually used outside the scope of **DoIt**,
we will change the program somewhat:

CLOSURES: FUNCTION REFERENCES

```
{$mode objfpc}
{$modeswitch anonymousfunctions}
{$ModeSwitch functionreferences}

Type
    TMyProcedureType = reference to procedure(x,y : integer);

Procedure ExecProcedure(A,B : integer;aProc : TMyProcedureType);
begin
  aProc(4*A,2*B)
end;

procedure doit;
var
  doPlus : Boolean;
  P : TMyProcedureType;

begin
  P:=procedure (x,y : integer)
  begin
    if DoPlus then
      writeln(X,'+',Y,'=',X+Y)
    else
      writeln(X,'-',Y,'=',X-Y)
    end;

    for doPlus:=False to True do
      ExecProcedure(10,20,P);
end;

begin
  Doit;
end.
```

The following output is produced:
40-40=0
40+40=80
Demonstrating that the current value of **DoPlus** is actually used when executing **ExecProcedure**,
while **DoPlus** is not defined in the scope of **ExecProcedure**.
The procedural variable **P** is actually superfluous, it can be omitted:

```
{$mode objfpc}
{$modeswitch anonymousfunctions}
{$ModeSwitch functionreferences}

Type
  TMyProcedureType = reference to procedure(x,y : integer);

Procedure ExecProcedure(A,B : integer;aProc : TMyProcedureType);
begin
  aProc(4*A,2*B)
end;

procedure doit;
var
  doPlus : Boolean;
begin
  for doPlus:=false to True do
    ExecProcedure(10, 20, procedure (x,y : integer)
                          begin
                            if DoPlus then
                              writeln(X,'+',Y,'=',X+Y)
                            else
                              writeln(X,'-',Y,'=',X-Y)
                          end
                        );
end;

begin
  Doit;
end.
```

CLOSURES: FUNCTION REFERENCES

Again, this also works with a local procedure, without anonymous functions:

```
{$mode objfpc}
{$modeswitch functionreferences}

Type
  TMyProcedureType = reference to procedure(x,y : integer);

Procedure ExecProcedure(A,B : integer;aProc : TMyProcedureType);
begin
  aProc(4*A,2*B)
end;

procedure doit;
var
  doPlus : Boolean;
  Procedure DoPrint(x,y : integer);
  begin
    if DoPlus then
      writeln(X,'+',Y,'=',X+Y)
    else
      writeln(X,'-',Y,'=',X-Y)
  end;

begin
  for doPlus:=false to True do
    ExecProcedure(10,20,@DoPrint);
end;

begin
  Doit;
end.
```

The above examples are of course very simple.
More complex code can be thought of:
threads could be started in the local procedure,
the procedural variable can be passed on to other routines.

When using a 'reference to procedure', it is not necessary to assign a local procedure or anonymous function to it.
It is of course possible to assign normal procedures or methods to a function reference:

```
{$mode objfpc}
{$modeswitch functionreferences}

Type
  TMyProcedureType = reference to procedure(x,y : integer);

Procedure ExecProcedure(A,B : integer;aProc : TMyProcedureType);
begin
  aProc(4*A,2*B)
end;

var
  doPlus : Boolean;

Procedure DoPrint(x,y : integer);
begin
  if DoPlus then
    writeln(X,'+',Y,'=',X+Y)
  else
    writeln(X,'-',Y,'=',X-Y)
end;

begin
  for doPlus:=false to True do
    ExecProcedure(10,20,@DoPrint);
end.
```

And for methods, the same applies: they can be assigned to references to procedures.

❹ 3 TYPES OF PROCEDURAL REFERENCES

In Object Pascal, we now have 3 types of procedural variables at our disposal:

```
Type
  TMyProc = procedure;
  TMyMethod = procedure of object;
  TMyReferenceProc = Reference to Procedure;
```

One might ask: why 3 different types ? Which one must I use ? Why not use a single type that is usable for all cases: calling a plain procedure, a method or an anonymous function ?

After all, if a 'reference to procedure' is usable for all 3 cases, why not simply use that ?

The answer to these questions is two fold:

❶ Historical reasons and backwards compatibility: Originally, in **Pascal** *(and Turbo Pascal)* only the procedural type existed. It was a simple pointer to an address to be called.
With Delphi and the introduction of classes, the 'procedure of object' appeared, and under the hood, it was implemented differently (*2 pointers are needed instead of 1*).
For backwards compatibility, the simple procedure reference had to be kept.
With closures, even more information must be stored, and it depends on the actual function: this is not compatible to the previous 2 types, which again need to be kept for backwards compatibility.

❷ Performance. a "Reference to procedure" tells the compiler that the environment of the function assigned to a referenced function must be captured.
This capturing happens internally by creating a temporary instance of an interface and a class to back it up.
This class is allocated on the heap, as it is an interface, there is reference counting involved etc: It implies lot of overhead which in many cases is simply not necessary and slows down the code.

These 2 reasons show why we now have 3 kinds of procedural types.

If you have an API which makes use of event handlers, you could consider converting the event handlers to 'reference to procedure'. If you are sure that your current API is not used by someone who uses the internal workings of the compiler to work with your API (*i.e. uses the pointers directly)* you will most likely not break existing code.

But before you rush to convert your APIs to enable anonymous functions, consider whether it is actually useful, because due to the need to capture the environment, there is a nonnegligible performance penalty involved, both in **Delphi** and **FPC.**

❺ CAPTURING THE ENVIRONMENT

So, why use function references and anonymous functions ? When you need to capture the environment to be available at a later time, then it makes sense to use function references.

In the above examples, all that was captured was the value of a local variable. But not only local variables are captured: also arguments to the outer function `(DoIt)` can be captured, or the value of `Self` in case of a method.

The following (fictitious) form sets an event handler for a logger class to show log messages. If the logger class can be called from various threads, the actual showing of the log message must be done in a synchronize routine. Function references (and anonymous functions) allow to do this in a straightforward manner:

```
procedure TMainForm.SetupLog;
begin
  MyLogger.OnLog:=@DoGlobalLog
end;

procedure TMainForm.DoLog(Const Msg : String);
begin
  Memo1.Lines.Add(Msg);
end;

procedure TMainForm.DoGlobalLog(Sender : TObject; Const Msg : String);
begin
  TThread.Queue(TThread.Current,procedure
  begin
    DoLog(Msg)
  end
 );
end;
```

When `TThread.Queue` notices that the current thread is the main thread, it will execute the anonymous function in `DoGlobalLog` at once, so it is "part" of the `DoGlobalLog` routine. At that point, the `Msg` parameter will still be available.
When called from another thread, the `TThread.Queue` call `DoGlobalLog` will actually queue the call to `DoLog` and will return at once, after which `DoGlobalLog` will also exit.
At a later time, when the queued procedure is actually executed in the main thread, the `Msg` parameter to `DoGlobalLog` must still be available. This is what the capture process does: it has saved the value of `Msg` so it is available when the queued anonymous function is executed.

❺ CAPTURING THE ENVIRONMENT

We can of course use a local procedure for this as well:

```
procedure TMainForm.DoGlobalLog(Sender : TObject; Const Msg : String);

Procedure WriteLog;
begin
  DoLog(Msg)
end;

begin
  TThread.Queue(TThread.Current,@WriteLog);
end;
```

To make it clear what the compiler does for you, we'll implement the same functionality without function references. We need somehow to save the message so it is available when the queued call is executed, as well as the form `Self` pointer. The way to do this is to store them in an object, and to let the object call the form `DoLog` method.
Here is such an object:

```
Type
   TLogTask = class
     FForm : TMainForm;
     FMsg : String;
     Constructor Create(aForm : TMainForm; const aMsg : String);
     Procedure Invoke;
   end;
```

The constructor is not very exciting, it stores the main form and message parameters:

```
Constructor TLogTask.Create(aForm : TMainForm; const aMsg : String);
begin
  FForm:=aForm;
  FMsg:=aMsg;
end;
```

The real work happens in the `Invoke` method. In that method, the saved message and form reference are used to actually log the message:

```
Procedure TLogTask.Invoke;
begin
  FForm.DoLog(FMsg);
  Destroy;
end;
```

Note that the `Invoke` calls the destructor `Destroy` : After the message was written, the object must destroy itself, or we would have created a huge memory leak.
Armed with this object, our `DoGlobalLog` routine now becomes:

```
procedure TMainForm.DoGlobalLog(Sender : TObject; Const Msg : String);
var
  aTask : TLogTask;
begin
  aTask:=TLogTask.Create(Self,Msg);
  TThread.Queue(TThread.Current,@aTask.Invoke);
end;
```

The first line creates the `TLogTask` object, passing it the `Self` pointer and the `Msg` message parameter. The second line queues the call to `WriteLog`. Since the reference to `aTask` is lost when `DoGlobalLog` exits, the `TLogTask` object needs to destroy itself.

Another approach would be to store it in an object list and let it remove itself from the list.

The above is actually performing exactly the same tasks as the capturing process, but manually;
It gives you an idea of the work the compiler does for you: behind the scenes it creates a class with fields to store all references, and a method to do the work. It uses an interface to handle automatic reference counting - so the object destroys itself when the method has been invoked, whereas the simple case above can use a manual call to destroy: in more complex scenarios it becomes a little more difficult to decide when to free the object.

6 INTERFACES

It was hinted at before, that the internal implementation of 'reference to' procedures uses interfaces. In Delphi, this is an implementation detail which is opaque to the user. In free pascal, this is made explicit. The definition

```
Type
  TMyProcedureType = reference to procedure(x,y : integer);
```

Is in fact equivalent to the definition of a reference-counted interface with a single Invoke method which has the same signature as the reference type:

```
Type
  TMyProcedureType = interface(IInterface)
    procedure Invoke(x,y : integer); stdcall; overload;
  end;
```

You can actually use the procedural 'reference to' type as an interface, i.e. you can declare a class with it:

```
Type
  TMyProcedureType = reference to procedure(x,y : integer);

  TMyClass = Class(TInterfacedObject,TMyProcedureType)
    procedure Invoke(x,y : integer);
  end;
```

Because the `Invoke` procedure is declared with 'Overload' it means you can include multiple such implicit interfaces in your class.

Why is this useful? It allows for more control over the capturing process. In the first place, sometimes you may wish to capture more environment than the compiler will capture automatically.
You can use this to avoid declaring local variables to provide an environment.

In the following example, an interface is constructed and used as the callback:

```
{$mode objfpc}
{$modeswitch functionreferences}

Type
    TMyProcedureType = reference to procedure(x,y : integer);

Procedure ExecProcedure(A,B : integer;aProc : TMyProcedureType);
begin
  aProc(4*A,2*B)
end;

Type
  TMyPrecisionImpl = class(TInterfacedObject, TMyProcedureType)
      Prec: LongInt;
      DoPlus : Boolean;
      procedure Invoke(X,Y: integer);
  end;

procedure TMyPrecisionImpl.Invoke(X,Y : Integer);
begin
  if DoPlus then
    writeln(X:Prec,'+',Y:prec,'=',(X+Y):Prec)
  else
    writeln(X:prec,'-',Y:prec,'=',(X-Y):prec)
end;

procedure DoIt;
var
  P : TMyPrecisionImpl;
  F : TMyProcedureType;
begin
  P:=TMyPrecisionImpl.Create;
  F:=P;
  P.Prec:=3;
  P.Doplus:=True;
  ExecProcedure(10,20,F);
  P.Prec:=5;
  P.DoPlus:=False;
  ExecProcedure(10,20,f);
  F:=Nil;
end;

begin
  DoIt;
end.
```

❻ INTERFACES

A second advantage is that because you have the possibility to use an interface, it means you can also control the lifetime of the interface and can hence improve performance: The above function constructs and destroys the interface only once. if the above function was written with an anonymous function, 2 instances of a compiler-generated interface would be constructed and destroyed.

Since the definition of an API determines whether or not a 'reference to' procedure is used, this mechanism can be used to work around some of the drawbacks that this implies.

The use of an interface as the backing mechanism has some consequences:

- it can be implemented by a class (*as demonstrated above*)
- Inheritance can be used.
- It is a reference counted interface and hence a managed type (So in most cases you class will descend from `TInterfacedObject`).
- The interface has **RTTI** associated with it.
- The `$M` directive applies to it.
- Contrary to a normal interface, it has no valid GUID and hence cannot be used in `QueryInterface`.

❼ DELPHI COMPATIBILITY

FPC's anonymous functions implementation is compatible to Delphi: if it works in Delphi, it will work in Free Pascal. In the previous paragraphs, we've shown that more is possible:

- Anonymous functions can be assigned to variables of regular procedural types. In Delphi, this is not allowed.
- Local procedures can be assigned to function references. This is also not allowed in Delphi.
- You can actually make use of the fact that behind the scenes, a closure is implemented using an interface.
- In Free Pascal you can declare a variable using an anonymous reference function type:

```
var
  P : reference to procedure;
```

In Delphi this is not allowed, you must explicitly declare a type.

```
Type
  TRefProc = reference to procedure;
var
  P : TRefProc ;
```

There is a small corner case where Free Pascal takes a different approach than Delphi.

```
procedure A;
begin
  Writeln('We are in procedure A');
end;

procedure B;
begin
  Writeln('We are in procedure B');
end;

Type
  TProc = reference to procedure;

procedure Test;
var
  p: TProc;
  p2: procedure;
begin
  p2:=A;
  p:=p2;
  p();
  p2:=B;
  p();
end;
```

When compiled with Delphi, this generates the following output:

```
We are in procedure A
We are in procedure B
```

While, when compiled with FPC, it generates the following output:

```
We are in procedure A
We are in procedure A
```

The reason for this is the use of temporary variables during assignments. Delphi generates code that amounts to the following:

```
procedure Test;
var
  p: TProc;
  p2: procedure;
begin
  p2 := A;
  p := procedure
    begin
      p2();
    end;
  p();
  p2:=B;
  p();
end;
```

While Free Pascal generates code equivalent to the following (*note the* `tmp` *variable*):

```
procedure Test;
var
  P : Tproc;
  p2, tmp: procedure;
begin
  p2:=A;
  tmp:=p2;
  p := procedure
     begin
       tmp();
     end;
  p();
  p2:=B;
  p();
end;
```

The example is contrived and unlikely to occur in practice, but it is worth mentioning.

❽ A WORD ON READABILITY

We've shown that in **Free Pascal,** you can use a local procedure instead of an anonymous function when passing it as a 'reference to' procedural type.
Whether you use a local procedure or an anonymous function is a matter of taste: the effect
is the same, the only difference is the readability of the code.
As a more elaborate example, take the following piece of code (taken from a Pas2JS RTL unit):

```
Procedure DoFetchURL(URL : String);

  function doOK(response : JSValue) : JSValue;
  var
    Res : TJSResponse absolute response;
  begin
    Result:=Null;
    If (Res.status<>200) then
      begin
        DoLoadError(Format(SErrUnknownError,[URL,res.StatusText]));
      end
    else
      Res.text._then( @LoadLanguageJson );
  end;

  function doFail(response{%H-} : JSValue) : JSValue;
  begin
    Result:=Null;
    DoLoadError(Format(SErrFailedToLoadURL,[URL]));
  end;

begin
  Window.Fetch(URl)._then(@DoOK,@DoFail).catch(@DoFail);
end;
```

And compare it with the following code which is functionally the same, but uses anonymous functions instead:

```
Procedure DoFetchURL(URL : String);
begin
  Window.Fetch(URl)._then(
     function doOK(response : JSValue) : JSValue
     var
       Res : TJSResponse absolute response;

     begin
       Result:=Null;
       If (Res.status<>200) then
         begin
           DoLoadError(Format(SErrUnknownError,[URL,res.StatusText]));
         end
       else
         Res.text._then( @LoadLanguageJson );
     end,
     function (response{%H-} : JSValue) : JSValue;
     begin
       Result:=Null;
       DoLoadError(Format(SErrFailedToLoadURL,[URL]));
     end).catch(
         function (response{%H-} : JSValue) : JSValue;

         begin
           Result:=Null;
           DoLoadError(Format(SErrFailedToLoadURL,[URL]));
         end
     );
end;
```

Note that the `doFail` function is duplicated when using anonymous functions, so there may be some advantages to using locally named procedures.

Luckily, in **Free Pascal,** every programmer can use what he likes most.

9 CONCLUSION

With the long-awaited arrival of anonymous functions, **Free Pascal** is again closer to **Delphi** compatibility. It does more than that and improves the **Delphi** implementation in some details. We've attempted to show the reasoning behind the implementation, and tried to explain the benefits (*and disadvantages*) of some of the constructs. To make use of this functionality, you need the main branch compiler: this functionality is not available in any release. How to get and use the main branch compiler has been shown in other articles.

DONATE FOR UKRAINE AND GET A FREE LICENSE AT:

https://components4developers.blog/2022/02/26/donate-to-ukraine-humanitarian-aid/

(Just click)

COMPONENTS 4 DEVELOPERS

If you are from Ukrainian origin you can get a free Subscription for Blaise Pascal Magazine, we will also give you a free pdf version of the Lazarus Handbook.

You need to send us your Ukrainian Name and Ukrainian email address (that still works for you), so that it proofs you are real Ukrainian.

please send it to editor@blaisepascal.eu and you will receive your book and subscription

BLAISE PASCAL MAGAZINE

DONATE FOR UKRAINE AND GET A FREE LICENSE AT:

https://components4developers.blog/2022/02/26/donate-to-ukraine-humanitarian-aid/
(Just click)

KBMMW PROFESSIONAL AND ENTERPRISE EDITION V. 5.19.00 RELEASED!

- **RAD Studio XE5 to 11 Alexandria supported**
- Win32, Win64, Linux64, Android, IOS 32, IOS 64 and OSX client and server support
- Native high performance 100% developer defined application server
- Full support for centralized and distributed load balancing and failover
- Advanced ORM/OPF support including support of existing databases
- Advanced logging support
- Advanced configuration framework
- Advanced scheduling support for easy access to multithread programming
- Advanced smart service and clients for very easy publication of functionality
- High quality random functions.
- High quality pronouncable password generators.
- High performance LZ4 and Jpeg compression
- Complete object notation framework including full support for YAML, BSON, Messagepack, JSON and XML
- Advanced object and value marshalling to and from YAML, BSON, Messagepack, JSON and XML
- High performance native TCP transport support
- High performance HTTPSys transport for Windows.
- CORS support in REST/HTML services.
- Native PHP, Java, OCX, ANSI C, C#, Apache Flex client support!

kbmMemTable is the fastest and most feature rich in memory table for Embarcadero products.

- **Easily supports large datasets with millions of records**
- **Easy data streaming support**
- **Optional to use native SQL engine**
- **Supports nested transactions and undo**
- **Native and fast build in M/D, aggregation/grouping, range selection features**
- **Advanced indexing features for extreme performance**
- New I18N context sensitive internationalization framework to make your applications multilingual.
- New ORM LINQ support for Delete and Update.
- Comments support in YAML.
- New StreamSec TLS v4 support (by StreamSec)
- Many other feature improvements and fixes.

Please visit
http://www.components4developers.com
for more information about kbmMW

- High speed, unified database access (35+ supported database APIs) with connection pooling, metadata and data caching on all tiers
- Multi head access to the application server, via REST/AJAX, native binary, Publish/Subscribe, SOAP, XML, RTMP from web browsers, embedded devices, linked application servers, PCs, mobile devices, Java systems and many more clients
- Complete support for hosting FastCGI based applications (PHP/Ruby/Perl/Python typically)
- Native complete AMQP 0.91 support (Advanced Message Queuing Protocol)
- Complete end 2 end secure brandable Remote Desktop with near realtime HD video, 8 monitor support, texture detection, compression and clipboard sharing.
- Bundling kbmMemTable Professional which is the fastest and most feature rich in memory table for Embarcadero products.

COMPONENTS 4 DEVELOPERS

EESB, SOA,MoM, EAI TOOLS FOR INTELLIGENT SOLUTIONS. kbmMW IS THE PREMIERE N-TIER PRODUCT FOR DELPHI / C++BUILDER