UNDERSTAND AND CONTROL THE FULL STACK

Holger Klemt, November 2019

What is from my point of view the biggest problem with web applications created by typical Delphi developers?

Most Delphi developers' main focus is on their current platform and they understand in depth the details of the VCL in general and the components that they have often been using for years. And for sure, the database SQL language is also part of their knowledge.

A lot of developers I have spoken to realise that they need specific solutions useable on a mobile device, but their daily work does not allow them to spend so much time on a completely new platform. Some of them started with solutions based on Delphi´s FireMonkey iOS or Android technology, but even a simple application created this way requires a lot of additional know-how, especially for the iOS market, before the first real world application can be used by customers.

One of the biggest disadvantages of such a native Android or iOS app is also the very limited possibility to immediately deliver an update for the app after a bug has been detected and fixed. Users might still work with their old version, even though the app store already has a new version available, sometimes 2 weeks or more after your bugfix version was uploaded from your IDE. If the old version still works, you are lucky, but if a bug destroys data or blocks use of the app altogether, you are in big trouble, and quickly need to find excuses for angry customers, who just want to use the app they paid for.

And this is not to mention the problems caused by new Google or Apple restrictions, for example the current requirement for 64-bit Android, or the Electron framework which is blocked by Apple.

Even though the 64-bit Android requirement has already been well-known for a very long time, Delphi does not have a working environment for this, even when they are so kind to offer you access to a beta version. All the source code you created is completely worthless, when one of the required middleware systems between your source code and the running app on your customer's mobile device is not available.

One of the solutions currently on the market as an add-on for pure JavaScript-based applications direct from your Pascal source code is ps2js, which is also used by TMS WEB Core, when it is integrated in Delphi or in the Lazarus IDE.

But compared to Lazarus with pas2js, which is 100% open source, TMS WEB Core also offers some parts only as closed source modules, which need to be licensed for each developer.

Other solutions like uniGUI are not comparable to pas2js, because they are unable to run on a mobile device without an active internet connection.

Any pas2js application is compiled completely into native JavaScript, and you can easily store and read data in a system called local storage, which is a very basic system and not comparable to a fully functional SQL Server, however simple tasks can be easily implemented based on this method, for example working time documentation on the mobile device, mobile access to product data, customer data or everything else you want to store on the system, even when it has no online access.

But how do you bring the data from your back office system to the mobile devices, and send any entry made on the mobile device back to the back office system?

If you check out the market and how others do this work you will see dozens of different possibilities, where currently REST-API servers seems to be the solution for almost everything.

But what does a typical REST server based application need?

a) A general client application that allows the display and editing of data, which should still have the possibility to store and read parts of the data in local storage.

b) A part in your client app, that knows which APIs on the REST server are implemented and how they need to be called, which data will come for which request and where they have to be used in the GUI app mentioned in part a).

c) A REST server that is an exact mirror of what you use and need in part b).

d) For sure a database where the data for the REST server is stored.

e) If your back office software is much more complicated than your mobile apps, you need also some import/export procedures, perhaps implemented already in the REST server or directly in the database.

Implementing basic functions directly inside the database makes sense for example, when you try to display product data with individual prices for each connected customer. You will definitely not want to implement this on all 5 levels in separate source codes.

So too many layers in your application makes any application much more complex, and development times explodes on any general change, since the focus in the above list is not on your fat client back office software and reporting system etc.

So a very simple requirement to display the correct product price for a customer who wants to order spare parts for a specific machine, which he has identified by serial number on the mobile device, is in fact, in a multilayer application, a horrible idea, when it is not using a proper architecture with only as many layers as are really required.

Why not use direct SQL statements from your web application source code?

Good question, from our perspective the answer is very easy: We do it exactly in that way!

What is the difference? Take a look at our demo application. A memo control is used to enter any valid SQL code on the connected database and the button click event sends it from the application over an HTTPS connection to a web server running with Apache and PHP. The PHP engine connects in a very simple 25-line PHP script to the Firebird database and sends the command directly to a stored procedure in the database, where the SQL statement is processed and, if rights to all objects are granted, will bring back the result for selects or simply execute any other insert/update/delete statement in the same way.

Big advantage: Have you ever had problems with a wrong client library or missing open ports when you try to connect to any database? This will not happen here since we simply use the secure HTTPS protocol to transmit commands and results.

You can also use the same technology not just from pas2js, it can also be accessed from any other compiled or interpreted application that supports HTTPS UTL calls and process the results.

What do I need to learn for this?

For the Lazarus Conference, 29th-30th November 2019 in Eindhoven, Netherlands, we have prepared a full set of source code to show how everything works. The complete source code comprises approximately 600 lines of code including all metadata objects used in the Firebird database, the full Lazarus source code, the simple PHP source code and everything else you need.

And we give you a guarantee: on day one of the conference we start with this project and show you in detail every module used. We will start with an empty virtual machine, install Firebird, install Apache, install PHP, install the pas2js required and start to implement the application line by line. At

the end of day one, you are 100% able to create exactly the same or a similar application based on the same technology. As we said: Understand and control the full stack.

What do I have to buy or pay for this?

What is free and open source?

Firebird 3	yes	www.firebirdsql.org
Apache	yes	www.apache.org
PHP	yes	www.php.org
DemoDB Firebird Source code	yes	on conference visitors' USB drive
DemoAPP Lazarus aps2js Source code	yes	on conference visitors' USB drive

For accessing the database internals we will use an IBExpert full version, which is available for 259€ excl. VAT, but anything implemented in the DemoDB can also be accessed using the free IBExpert Personal Edition. So if you want to be prepared for doing everything on your own laptop, please download and install the IBExpert Personal Edition or the IBExpert Developer Studio Edition. All other setups and files will be on the conference visitors' USB drive, presented by Blaise Pascal Magazine.

You want to attend the conference?

See https://www.lazpro.net for details, the fee for one day 55€, both days cost 100€.

You have no time to attend or distance is too far away?

Please let us know that you are interested. The Lazarus factory will also hold a virtual conference when we have enough attendees. This is simply a GoToMeeting session that you can attend at home on your own computer and you will see Holger on the cam and on the screen where he presents the same content again.

The price for the virtual event will be 50€ per person and take around 4 hours. You can also use the GoToMeeting chat system and, if you wish, also use your headset and own webcam to chat with us about details.

Depending on the location where you come from, it will typically start at:

07:00 CET	Eastern countries, Asia, Japan, Australia etc.
11:00 CET	Europe, Africa
15:00 CET	USA, Canada, South America

After we received your e-mail that you wish to attend, we will send a list with available dates when the event will happen. I you live in eastern countries, but prefer to attend another time frame that proposed, it is not a problem.