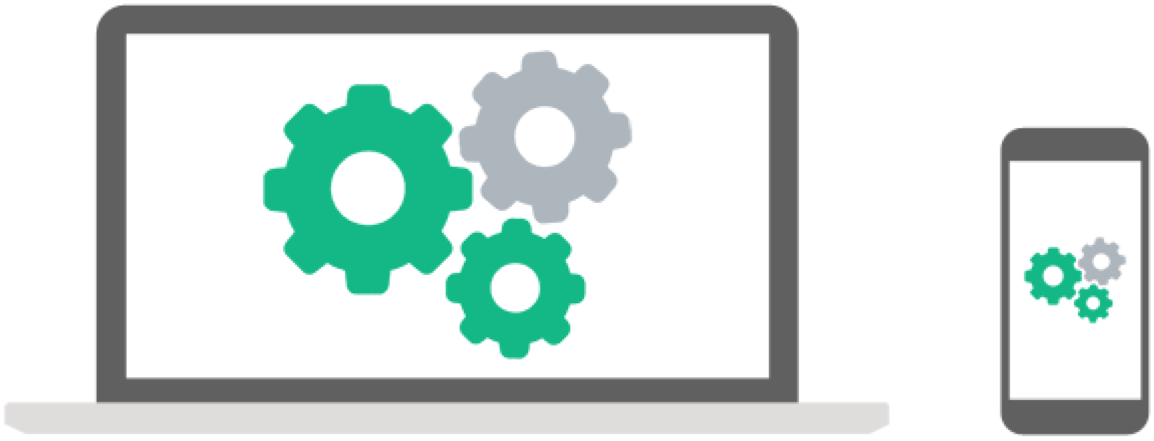
Understand and control the full stack

Holger Klemt, November 2019



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

Many Delphi developers focus on their current platform and know the details of the VCL in general and the components that they have been using frequently for years. Of course, database SQL language is also part of their knowledge.

All the developers we've talked to realize that they need certain solutions, which can be used on a mobile device. But their daily work doesn't allow them to invest that much time on a whole new platform.

Some of them began with solutions based on Delphi's FireMonkey-iOS or Android technologies, but even simple applications created this way require a lot of additional know-how, especially for the iOS market, before the first real application can be used by customers.

One of the biggest disadvantages of such a native Android or iOS app is the very limited possibility to provide an update for the app immediately after a bug has been detected and fixed.
It may take two weeks or longer for the bugfix version to be uploaded from your IDE and made available to the customer on the device again.

Users may still need to work with older versions, even though an update is already available in the App Store.

If the old version still works, the users are lucky, but if a bug destroys data or blocks the use of the app, you are in big trouble and need to quickly find excuses for angry customers, who rightly want to use the app they've paid for.

Not to mention the problems caused by new Google or Apple restrictions, such as the current 64-bit Android requirement or the Electron framework which is blocked by Apple..

Even though the 64-bit Android requirement has been known for a long time, Delphi does not have a working environment for this, even though they are kind enough to offer you access to a beta version.

All the source code you create is completely worthless, if one of the necessary middleware systems between your source code and the running app is not available on your customer's mobile device.

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

In comparison to Lazarus with pas2js, which is 100% open source, TMS WEB Core however offers some parts only as closed source modules, which every developer has to license.

Other solutions such as uniGUI are not comparable to pas2js, as they are unable to run on a mobile device without an active Internet connection.

Each pas2js application is fully compiled in native JavaScript. You can easily store and read data in a so-called local storage system. This simple system is not comparable to a fully functional SQL server.

However simple tasks can be easily implemented with this method, such as working time documentation on mobile devices, mobile access to product data, customer data or anything else you want to store on the system, even if it has no online access.

But how can you transfer the data from your office management system to the mobile devices and return the entries made on the mobile devices to the office management system?

If you look at the different functionalities on the market, you will find dozens of different possibilities where REST-API seems to be the solution for almost everything.

But what does a typical REST server-based application need?

a) A general client application for viewing and editing data. This application should also have the ability to store and read parts of the data in local memory.
b) A part of your client app that knows which APIs are implemented on the REST server and how to call them, what data comes for which request, and where they have to be used in the GUI app mentioned in part a).
c) A REST server that mirrors exactly what you use and need in part b).
d) And definitely 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 will also need some import/export procedures, which may already be implemented on the REST server or directly in the database.

It makes sense to implement basic functions directly in the database, for example, if you want to display your product data with individual prices to all connected customers.

You will definitely not want to implement this at all 5 levels in separate source code.

Too many levels in your application make the application much more complex, and development times explode with each general change, as the above list does not focus on the office management software and reporting system of your FAT client.

A very simple requirement to display the correct product price on the mobile device for a customer, who wants to order spare parts for a particular machine identified by the serial number, is indeed a terrible idea in a multilayer app, if it does not use 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! And from our perspective the answer is very simple: That is exactly how we do it!

A memo control is used to enter valid SQL code into the connected database. The Click Event button sends it from the application over an HTTPS connection and to a web server running Apache and PHP. The PHP engine connects to the Firebird database with a very simple 25-line PHP script and sends the command directly to a stored procedure in the database, where the SQL statement is processed.
The result for selects is returned and, if object permissions have been assigned, other statements such as insert/update/delete are executed in the same way.

A great advantage:

When trying to connect to a database, have you ever had problems with a wrong client library or missing open ports?

This will not happen here, because we simply use the secure HTTPS protocol to transmit commands and results.

You can use the same technology not only from pas2js, but also from any other compiled or interpreted application that supports HTTPS UTL calls, can access them, and can process the results.

What do I need to learn?

For the Lazarus Conference, 29th - 30th November 2019 in Eindhoven, Netherlands, we have prepared a complete 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 complete Lazarus source code, the simple PHP source code, and everything else you need.

And we give you a guarantee! - At the end of the first conference day you will be 100% able to create an application based on this technology.

On the first day of the conference we will start with this project and show you each module used in detail.

We will start with an empty virtual machine, install Firebird, install Apache, install PHP, install the pas2js required and begin to implement the application line by line.

Entering the web development industry today is relatively easy. But to be an all-rounder, as in any other industry, is extremely difficult.
We'll help you get closer to your goal as a full-stack developer! Let's go!

What do I have to buy? What is free and open source?

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

A full IBExpert version, available for €259 plus VAT/sales tax, is used to access the database internals. But everything implemented in the DemoDB can also be accessed using the free IBExpert Personal Edition.

So if you want to prepare yourself to do everything on your own laptop, please download and install the free IBExpert Personal Edition or the IBExpert Developer Studio Edition full version.

All other setups and files will be on the conference visitors' USB drive presented by Blaise Pascal Magazine.

Would you like to attend the conference?

Further information can be found at https://www.lazpro.net. The fee for one day is 55 € and for both days 100 €.

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

The Lazarus factory will also hold a virtual conference if a sufficient number of participants is attained.

The virtual conference is offered as a GoToMeeting session in English, where you can participate from your home or at your workplace with your own computer. View Holger on camera and the presentation on screen at the same time. The content corresponds to the conference topic as described above.

You can also use the GoToMeeting chat system and, if you wish, use your headset and your own webcam to chat with us about details.

The price for this virtual event is 50 € per person and will be charged in advance. The virtual event lasts about 4 hours.

Our virtual event starts at different times depending on the location for the following continents/countries:

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

If you wish to participate, please send an e-mail to sales@ibexpert.biz. You will then receive a list with all available dates for your geographical region. If you prefer a different time than the one foreseen for your region, please let us know in advance in your e-mail.