NEWS · SEPTEMBER 16, 2025

FlexCel is the perfect report generator for any frontend

By Dr. Holger Flick

FlexCel is a powerful library for creating and manipulating Excel files in Delphi and .NET applications. You can export to many image formats as well as PDF files.

The product allows you to create templates in Excel, which makes it easy to design and format reports. You can use all the features of Excel, such as formulas, charts, and conditional formatting. Then, in your backend, you can use FlexCel to populate the templates with data from your database or other sources.

It is currently distributed by TMS Software, a well-known company in the Delphi community. You can find more information about FlexCel on their website. As said, it is available for both Delphi and .NET, so you can use it

in various types of applications.

However, this is a major understatement. FlexCel is not just a tool to create Excel files. It is a comprehensive solution for generating reports in various formats, including PDF, images (PNG, JPEG, BMP, GIF, TIFF), and even HTML.

I have been using FlexCel for Delphi and .NET for many years. It is a fantastic library for generating Excel and PDF reports from templates. I have used it in various projects, from small desktop applications to large web services. In my books you can find real-world examples of how to use FlexCel in Delphi applications to create reports. In my latest book about Modern Delphi Application development, I have a whole chapter dedicated to

FlexCel and reporting as well as in my book about creating Web Services with XData.

Another Use Case

Recently, I have been working on a Next.js project that required generating complex reports that users had to download as PDF files. I decided to use FlexCel for .NET in the backend, and it worked like a charm. I had two alternatives implementing the backend:

- Using Delphi with WebBroker or RAD Server
- Using Delphi with XData

Both these alternatives would have worked, but I decided to go with ASP.NET Core because I wanted to explore the .NET ecosystem as Microsoft has recently made great improvements with .NET Core to compile for any platform, including Linux and macOS. Also, ASP.NET Core Minimal APIs are very easy to use and get started with.

Deployment Scenario

Deployment is very easy. I picked Linux as the operating system to save costs on hosting. A server that is capable of hosting a Linux ASP.NET Core application is much cheaper than a Windows server. The same is obviously true if you want to host a Delphi application on Windows. Thankfully, Delphi comes with a Linux compiler and both XData and FlexCel support Linux.

To isolate the reporting service from the outside world and only to expose the Web Application, I created a Docker Compose setup with two containers:

- One container for the Next.js frontend
- One container for the ASP.NET Core backend

The two containers communicate via a private network. The Next.js application calls the ASP.NET Core backend to generate the reports. The backend uses FlexCel to create the reports and returns them to the frontend, which then serves them to the user.

The Web Application is accessible via HTTPS, and the backend is only accessible from the frontend container. This setup ensures that the backend is not exposed to the internet, which adds an extra layer of security.

Conclusion

FlexCel is a fantastic library for generating reports in various formats. It is easy to use and integrates well with both Delphi and .NET applications. If you need to generate reports in your application, I highly recommend giving FlexCel a try. You can find more information about FlexCel on the TMS Software website.

Need help?

If you need help with FlexCel, Delphi, .NET, or anything else related to software development, feel free to reach out to me. I offer consulting and training services and would be happy to assist you with your project. You can contact me via my website.

INDEPENDENT · AD-FREE · READER-SUPPORTED

Free to read. Not free to make.

Every tutorial, article, and line of example code on Holger's Code is published without a paywall and without ads. This page is about what keeps it that way.

Professional work, published free

The content here is held to the same standard as professional client work: every tutorial is built as a real project, tested against current versions, documented step by step, and revisited when the frameworks move on. A single in-depth tutorial routinely takes days — and that's before the bills behind it: servers, storage, build infrastructure, licenses, and the hardware everything is verified on.

If an article or tutorial here has saved you an afternoon of debugging, taught you a technique, or helped you ship, it created real value for you. Supporting it isn't charity — it's paying for professional work you found useful, at whatever price you choose. And if you can't, keep reading anyway. That's what the free part means.

Ways to support

KO-FI

Buy me a coffee

A one-time tip — no account, no subscription, any amount.

BOOKS & COURSES

Buy the paid work

The paid tier of everything published free here — and you get something lasting in return.

SPREAD THE WORD

Share an article

Send a tutorial to a colleague, cite an article, subscribe to the RSS feed.

Nothing here goes behind a paywall, whether you chip in or not. But if something on this site earned its keep today, this is where to say so.

Support on Ko-fi —