

A WEB-BASED FINANCIAL PLANNING APPLICATION FOR ENTERPRISE BUDGETING AND INVESTMENT MANAGEMENT: A DESIGN AND IMPLEMENTATION CASE STUDY

Ondřej Pavlík¹, Oldřich Trenz¹

¹Department of Informatics, Faculty of Business and Economics, Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1, 613 00 Brno, Czech Republic

DOI: https://doi.org/10.11118/ecos2026_026

ABSTRACT

A mid-sized company that tripled its number of organizational units over five years outgrew its spreadsheet-based budgeting. Centre managers planned in their own Excel files, the finance department merged them by hand, and errors surfaced only after the fact. This paper reports the design and implementation of an on-premise web application that sits between the company ERP system and the centre managers, replacing the manual consolidation with a single validated planning workflow. The work follows a standard software-engineering process: analysis of the existing routine, requirements (22 functional, 12 non-functional), a make-or-buy assessment of three commercial vendors, UI design in Figma, implementation, testing, and deployment. The application is built on ASP.NET Core (.NET 8) with Razor Pages, a Microsoft SQL Server database, SOAP integration with the ERP, integrated Windows authentication, and Power BI reporting embedded through OAuth 2.0. None of the three vendors covered the on-premise requirement, customization, and integration at once, so a tailored solution was developed and is now in production. We describe the architecture, the integration and authentication decisions, and five concrete problems solved during development. No quantitative performance metrics were measured; the contribution is the engineering design.

Keywords: financial planning, ASP.NET Core, ERP integration, on-premise deployment, Power BI, Razor Pages

JEL Code: C88



This article is licensed under CC BY-SA 4.0 Attribution-ShareAlike 4.0 International license.
<https://creativecommons.org/licenses/by-sa/4.0/deed.en>

1 INTRODUCTION

Budgeting in a small company often starts as a single shared spreadsheet. It works until the company grows. The firm behind this case study tripled its number of organizational units in five years (Pavlik, 2025), and the spreadsheet routine did not scale with it. Each centre manager planned in a separate Excel file, the files arrived in different layouts, and the finance department merged them manually into a company-wide plan. Mistakes in a single cell propagated into the consolidated figures and were usually caught only during a later review, when correcting them meant another round of e-mails.

The aim of the work was to design and implement a web application for managing and planning corporate budgets and investments that acts as an intermediary between the company ERP system and the centre managers. It was to replace the manual consolidation with a centralized overview and validate entries as they are made, rather than after submission. Two constraints shaped every later decision: the system had to run fully on-premise, and it had to fit the company's existing Microsoft-based infrastructure instead of forcing new processes onto the staff.

1.1 Why not a commercial product

The cost of bad financial data is a recurring theme in the management-information literature: data that cannot be trusted forces manual rework and erodes the value of every report built on top of it (Redman, 2008; Laudon and Laudon, 2020). Centralizing the planning data was therefore the first goal, ahead of any new feature. The make-or-buy question came next. Romero *et al.* (2021) show that financial web applications are commonly assembled on top of a business-intelligence layer rather than written from scratch, which is one argument for buying.

Three commercial vendors active on the Czech market were assessed against the company's key requirements, together with the planning module of the existing ERP. The comparison was qualitative; it is reproduced in Section 3 as Table 1. None of the options met the on-premise requirement, the demand for deep customization, and integration with the existing systems at the same time. Cloud dependence was the most frequent disqualifier. The ERP planning module was too rigid to model the company's centre structure. This led to the decision to build a tailored solution, which the requirements analysis (Sommerville, 2015; Gorbachenko, 2021) then turned into 22 functional and 12 non-functional requirements.

2 METHODOLOGY

The project followed a sequential software-engineering process with an iterative testing phase: analysis of the current process, requirements definition, UI design, implementation, testing, and deployment. The non-functional requirements fixed the technology before the first line of code, because the firm already ran a Microsoft stack and Power BI licences and wanted to keep them.

The backend is an ASP.NET Core application on .NET 8, built as a monolith with Razor Pages and server-side rendering and hosted by the integrated Kestrel server (Freeman, 2023; Karpenko, 2025). A monolith was a deliberate choice for a single internal tool with one team; the trade-offs against a service-oriented split are well documented (Richards and Ford, 2020; Newman, 2021; Bass *et al.*, 2012), and none of the drivers for splitting applied here. Persistent data lives in Microsoft SQL Server. The schema was designed first as a conceptual model abstracted over four main sections

of the system and then as a physical ER diagram. Database rows are mapped onto domain models through a dedicated mapping layer, and the services are wired together with dependency injection.

Integration with the ERP uses a SOAP API with scheduled synchronization. The SOAP client is configured dynamically rather than from a fixed endpoint, so the same code reaches different ERP environments without recompilation, and the channel is secured for the data it carries. Sensitive configuration is kept out of source control with User Secrets, and the application logs through Serilog.

Authentication reuses the company's Active Directory. With integrated Windows authentication, a user who is already signed in to the domain reaches the application without logging in again. Reporting is handled by Power BI, embedded under OAuth 2.0 in the Service Principal mode, where the application authenticates as itself and requests an embedded token carrying an Effective Identity so that each manager sees only the figures for their own centres. The deployment topology, including the link to the reporting service, is shown in Figure 1.

The frontend stays inside the same Razor Pages application. Layout uses Tailwind CSS with Heroicons, and the dynamic parts are plain JavaScript, without a separate framework. The interface was designed first in Figma, where the colour scheme, logo, wireframes, and clickable prototypes were checked with users before implementation; Figure 3 shows one of the resulting screens, the

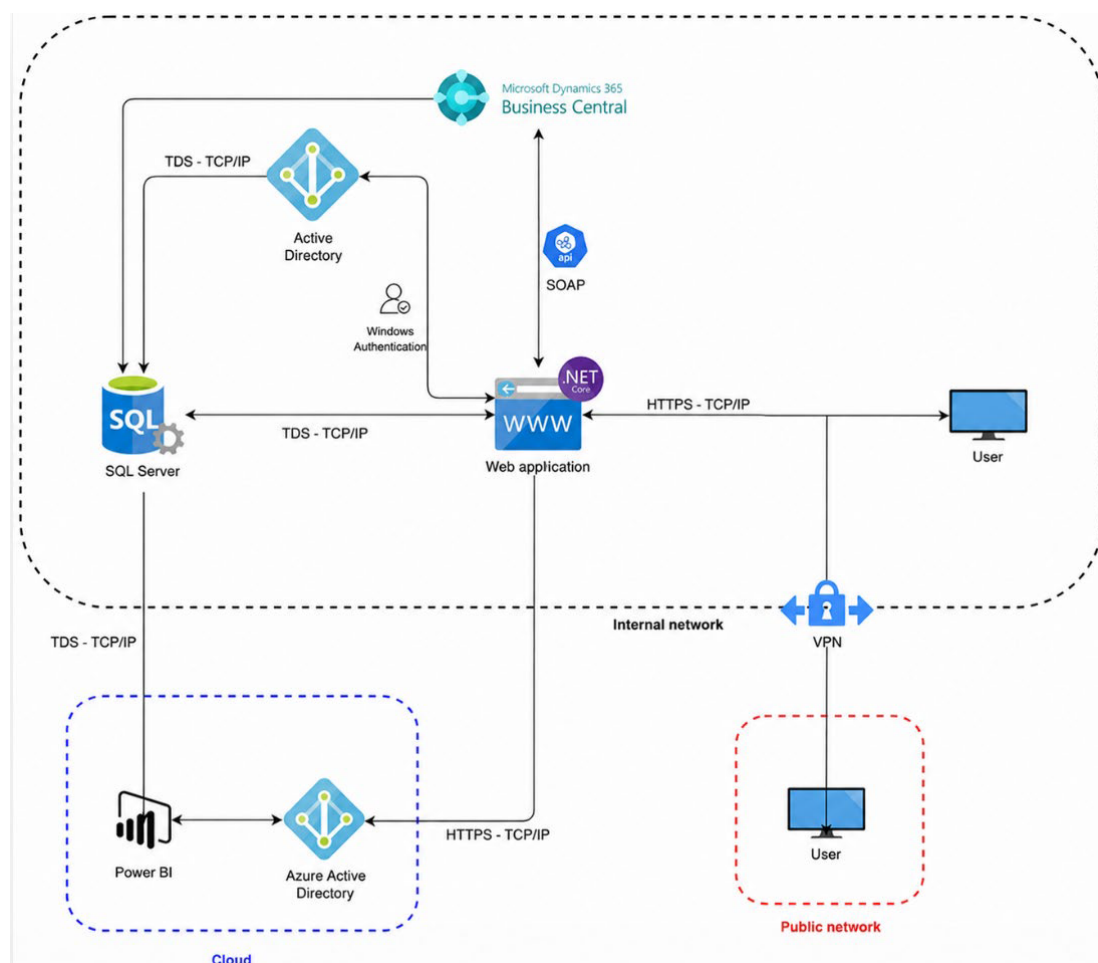


Fig. 1: Deployment architecture of the financial-planning web application integrating the ERP system, SQL Server, Active Directory and Power BI across the internal network and VPN (adopted from Pavlík, 2025).

investment-creation form. Reporting needed two Power BI variants, because paginated reports render differently from interactive ones and require their own embedding path.

The application is deployed on-premise on IIS, installed with the .NET Core Runtime and the Hosting Bundle and run from a dedicated application pool with in-process hosting. Historical data from the old spreadsheets was migrated during rollout, and every module was first released to a separate UAT (User Acceptance Testing) server. Testing combined MSTest unit tests on the key components, use-case scenarios that replay real planning situations, and isolated checks of the API against unexpected inputs. The thesis reports no test counts, coverage, or timing figures, and none are claimed here (Patton, 2002).

3 RESULTS

The outcome of the project is a working, deployed application, so the results are described qualitatively. Table 1 documents the vendor assessment that led to the in-house decision. It is the single table in the thesis and reports ordinal judgements, not measurements; no row is a winner, because every commercial option was rejected.

The tailored application replaced the Excel-based routine in production. Centre managers now enter their plans through the cost-planning screen shown in Figure 2, the entries are validated on the spot, and the consolidated view is built from the database instead of by hand. The application reads source data from the Business Central database and uploads planned data back to Business Central over SOAP, signs users in through their domain accounts, and presents the financial statements through embedded Power BI. The company reported that the time saved in some areas was higher than it had expected; that statement was not quantified in the thesis, and no figure is attached to it here.

Five concrete problems were solved during development, and they are part of the result because they shaped the working system. The default .NET Core limit on the number of form fields was reached by the largest plans and had to be raised explicitly. Loading large plans was slow until the database indexing was reworked (Mukherjee, 2019). Power BI paginated reports did not render through the interactive embedding path and needed a separate one. Logging silently failed on IIS until the application pool was given write permissions to the log directory. The frontend constraints rendered correctly only in Chromium and required extra work for Safari and Firefox. None of these is dramatic on its own; together they are most of the distance between a prototype and a tool people use daily.

Tab. 1 Qualitative comparison of the three assessed commercial vendors against the client's key requirements (adopted from Pavlík, 2025)

Criterion	Sophia Solutions	TARGETTY	GIST
On-premise operation	Partly supported	Possible, mostly not	Yes
Customization flexibility	Low	Medium	Very high
Integration with existing systems	Standard connectors + API	Strong MS-ecosystem integration	Any custom interface
Implementation speed	Medium	Fast	Long
Initial price	Higher	Medium	High
Cloud dependence	Optional	Yes	Optional

Tab. 2 Annual cost-planning screen of the deployed application, showing planned expense items per accounting period (adopted from Pavlík, 2025)

Tab. 3 Investment-creation form with depreciation parameters and reinvestment milestones (adopted from Pavlík, 2025)

4 DISCUSSION

The decisive constraint was not technical but organizational: the company wanted to keep its processes and its infrastructure. That single requirement explains most of the design. It ruled out the cloud-leaning vendors in Table 1, it made integrated Windows authentication the natural sign-in mechanism, and it made Power BI the obvious reporting tool, since the licences and the staff's familiarity with it were already there. A greenfield project would likely have chosen differently; here, fitting the existing environment mattered more than picking the theoretically best component.

The monolith deserves a second look, because microservices are often treated as the default for new systems. For one internal application maintained by one team, the usual reasons to split (independent scaling, separate release cycles, multiple teams) were absent, and a split would have added deployment and operational cost for no return. The Power BI choice is more debatable. It was right given the existing licences, but it ties the reporting to a paid product and to an embedding model that, as the paginated-report problem showed, is not uniform across report types. A company without those licences might reasonably build its own reporting instead.

We should be honest about what this work can and cannot claim. It shows that a tailored on-premise application can replace a manual budgeting process and run in production, and it documents the decisions and the obstacles in enough detail to be reusable. It does not show by how much the new process is faster or less error-prone, because those quantities were never measured. The qualitative statement that time savings exceeded expectations is the client's, not a result of this study, and treating it as a number would overstate the evidence. The most useful part for a reader facing a similar problem is probably the make-or-buy reasoning together with the list of five concrete obstacles, which are easy to underestimate at the planning stage.

5 CONCLUSION

A web application for corporate budgeting and investment planning was designed, implemented, and deployed in a company's own infrastructure, replacing a manual, spreadsheet-based consolidation that no longer scaled with the firm's growth. The on-premise requirement, combined with the need for customization and ERP integration, was the reason no commercial product fit and a tailored solution on ASP.NET Core, SQL Server, and Power BI was built instead. For practitioners weighing the same decision, the lesson is that infrastructure fit and account integration can outweigh feature checklists, and that the unglamorous problems (form-field limits, indexing, report embedding, IIS permissions, cross-browser rendering) are where much of the implementation effort actually goes. The work was not evaluated with quantitative metrics; that is its main limitation. Useful next steps are clear and were identified during the project: ADFS account federation for single sign-on to Power BI, a possible in-house reporting layer to drop the Power BI dependency, form-based or OAuth 2.0 authentication for access from outside the local network, a React frontend to lift the interactivity limits of Razor Pages, and a CI/CD pipeline, for instance on Azure DevOps (Manolov *et al.*, 2025), to make releases routine.

Acknowledgements

This paper was supported by the project CZ.02.1.01/0.0/0.0/16_017/0002334 Re-search Infrastructure for Young Scientists, which is co-financed from the Operational Programme Research, Development and Education.

Anthropic Claude was used to summarize the text of the final thesis (Pavlik, 2025) and transform it into the form of this compact paper. The paper was then reviewed and fine-tuned by the author. The author is responsible for the correctness of the final text.

REFERENCES

- BASS, L., CLEMENTS, P., KAZMAN, R. 2012. *Software Architecture in Practice*. 3rd ed. SEI Series in Software Engineering. Upper Saddle River, NJ: Addison-Wesley Professional. ISBN 978-0-321-81573-6
- FREEMAN, A. 2023. *Pro ASP.NET Core 7*. 10th ed. Manning Publications. ISBN 1-63343-782-5
- GORBACHENKO, P. 2021. What are Functional and Non-Functional Requirements and How to Document These. *Enkonix*. <https://enkonix.com/blog/functional-requirements-vs-non-functional/>
- KARPENKO, M. 2025. Innovative Solutions for Enhancing System Performance on the .NET Platform. *International Journal of Computer*. 54(1), 21-28. <https://ijcjournal.org/index.php/InternationalJournalOfComputer/article/view/2354>
- LAUDON, K., LAUDON, J. 2020. *Management Information Systems: Managing the Digital Firm*. 16th ed. Harlow: Pearson. ISBN 978-0-13-519179-8
- MANOLOV, V. *et al.* 2025. Practical Comparison Between the CI/CD Platforms Azure DevOps and GitHub. *Future Internet*. 17(4), 153. <https://doi.org/10.3390/fi17040153>
- MUKHERJEE, S. 2019. *Indexes in Microsoft SQL Server*. ResearchGate. https://www.researchgate.net/publication/331889027_Indexes_in_Microsoft_SQL_Server
- NEWMAN, S. 2021. *Building Microservices*. 2nd ed. Sebastopol, CA: O'Reilly Media. ISBN 978-1-492-03402-5
- PATTON, R. 2002. *Software Testing (in Czech)*. Brno: Computer Press. ISBN 80-7226-636-5
- PAVLÍK, O. 2025. *Web application for financial planning* [in Czech]. Master's thesis, head: TRENZ, O. Brno: Mendel University in Brno.
- REDMAN, T. C. 2008. *Data Driven: Profiting from Your Most Important Business Asset*. Boston, MA: Harvard Business Review Press. ISBN 978-1-4221-1912-9
- RICHARDS, M., FORD, N. 2020. *Fundamentals of Software Architecture: An Engineering Approach*. 1st ed. Sebastopol, CA: O'Reilly Media. ISBN 978-1-4920-4345-4.
- ROMERO, C. *et al.* 2021. Web Application Commercial Design for Financial Entities Based on Business Intelligence. *Computers, Materials & Continua*. 67(3), 3177-3188 <https://doi.org/10.32604/cmc.2021.014738>
- SOMMERVILLE, I. 2015. *Software Engineering*. 10th ed. Harlow: Pearson. ISBN 978-0-13-394303-0

Contact information

Ondrej Pavlik: e-mail: 77449@node.mendelu.cz

Oldrich Trenz: e-mail: oldrich.trenz@mendelu.cz