SILESIAN UNIVERSITY OF TECHNOLOGY PUBLISHING HOUSE

SCIENTIFIC PAPERS OF SILESIAN UNIVERSITY OF TECHNOLOGY 2024
ORGANIZATION AND MANAGEMENT SERIES NO. 213

ARCHITECTURAL DECISIONS SUPPORTING THE MANAGEMENT OF PUBLIC UTILITY BUILDINGS

Ireneusz J. JÓŹWIAK[1*], Rafał KRUSZYNA[2], Alicja M. KRUPA[3], Jan SWITANA[4]

[1] Wroclaw University of Science and Technology, Faculty of Computer Science and Management; ireneusz.jozwiak@pwr.edu.pl, ORCID: 0000-0002-2160-7077
[2] Wroclaw University of Science and Technology, Faculty of Computer Science and Management; rafalkruszyna10@gmail.com, ORCID: 0009-0008-3053-486X
[3] Wroclaw University of Science and Technology, Faculty of Architecture; a.joozwiak@wp.pl, ORCID: 0000-0002-8796-2549
[4] Wroclaw University of Science and Technology, Faculty of Electronics, Photonics and Microsystems, switana.jv@gmail.com, ORCID: 0000-0001-7188-8199
*Correspondence author

Purpose: The aim of the article is to present the concept of an integrated system for managing cultural institutions.
Design/methodology/approach: The analysis is based on the specification of requirements and making architectural decisions to support management.
Findings: Supporting the management of cultural institutions' activities.
Originality/value: A detailed specification of requirements has been proposed for the functioning of cultural institutions, and the development of mobile and web applications.
Keywords: cultural institution, software architecture, management.
Category of the paper: Research paper.

1. Introduction

There is no doubt that in recent times we have witnessed the digitization of various areas of our lives. The rapid advancement of technology has permeated every facet of our daily routines, from how we communicate and work to how we manage and interact with public services. However, these innovations do not always fully meet users' expectations. The discrepancy between technological potential and actual user satisfaction leads to the continuous search for new solutions that can bridge this gap effectively. Many social institutions currently operate with incomplete or inefficient applications and systems, which hinders their ability to fully harness the benefits of digitization. This paper presents the concept of anintegrated system designed to support the management of public utility facilities, with a particular focus on cultural institutions.

Several areas of cultural institutions' operations were considered in this study, such as communication with the local community, the range of activities offered by the institutions, and integration with social media. Effective communication with the local community is paramount for cultural institutions to remain relevant and engaged with their audience. This includes promoting events, gathering feedback, and fostering a sense of community ownership and participation.

A detailed specification of requirements has been proposed to address these needs. This specification encompasses the technical, functional, and user experience aspects necessary for the successful implementation of the integrated system. After specifying the requirements, it was necessary to make architectural decisions that would align with the project's goals and constraints. The task set by the user is the simultaneous development of a mobile and web application.

2. Scope of the project

The functioning of cultural institutions encompasses three primary areas: communication, reservation, and system control. Each of these areas is integral to the efficient management and operation of such institutions, ensuring they can effectively engage with the public, manage their resources, and maintain smooth operational workflows.

Within the scope of communication, several functionalities must be considered to enhance the interaction between cultural institutions and their audience:

a) functionality that enables the publication of new information about planned activities,
b) the possibility of integrating the system with social media platforms such as Facebook and Instagram,
c) sending email notifications to users about news and changes in cultural institutions.

Within the reservation system, several aspects are critical for managing user interactions and participation in cultural activities:

a) sign-ups for activities offered by cultural institutions,
b) ticket reservations,
c) participation in events and activities.

System control is another critical area, involving the management of user profiles, permissions, reservations, and payment statuses.

As part of the conducted research aimed at understanding the current standards of cultural institutions, a survey among individuals interested in the functioning of cultural institutions was proposed.

Based on the conclusions drawn from the surveys, it was crucial to develop a flexible communication environment that meets the expectations of institutions operating in various settings. An interesting finding from the surveys is that traditional methods of information storage still dominate; besides computer databases, paper documentation remains prevalent. It is noted that online technologies will soon dominate, as they offer advantages in terms of efficiency, accessibility, and data security. This shift towards digitalization underscores the need for an integrated system that can seamlessly transition institutions from traditional to modern methods of information management.

The paper proposes a system that will significantly streamline the operations of cultural institutions. This system is designed to enhance communication, simplify reservation processes, and provide robust system control, thereby meeting the diverse needs of cultural institutions.

3. Choice of technology

A key issue is the choice of technology. It is proposed to develop both a mobile application and a web application. This dual-platform approach is essential for reaching users on their preferred devices, whether they are accessing the system from their smartphones or desktops.

The choice of backend technology was crucial. The backend was implemented using Kotlin (Jemerov, Isakova, 2018) and the Spring Boot framework (Walls, 2015). Kotlin was selected because it is designed to interoperate seamlessly with Java, offering modern language features while maintaining compatibility with existing Java libraries and frameworks. This interoperability ensures that the development team can leverage existing Java expertise and resources, while also benefiting from Kotlin's concise syntax and enhanced safety features, such as null safety and extension functions. The Spring Boot framework was chosen for its comprehensive suite of features that simplify the development of production-ready applications. Its dependency management capabilities, built-in transaction support, and extensive ecosystem of modules and extensions provide a robust foundation for building scalable and maintainable backend services.

For the visual component, TypeScript (Vanderkam, 2020) was chosen. TypeScript, a statically typed superset of JavaScript, has become increasingly popular and now dominates over 66% of projects (TypeScript, 2022). Its strong typing system helps catch errors early in the development process, enhancing code quality and maintainability. Additionally, the Angular Material library (Bampakos, Deeleman, 2023) was utilized to accelerate implementation. Angular Material provides a set of reusable, well-tested, and accessible UI components based on Google's Material Design specifications. This choice not only speeds up the development process by providing pre-built components but also ensures a consistent and user-friendly interface across the application.

The API specification was implemented according to the OpenAPI standard (Gough et al., 2020). The OpenAPI standard, formerly known as Swagger, provides a framework for defining and documenting APIs in a language-agnostic manner. This standardization facilitates the use of external code generation tools, which can automatically generate client libraries, server stubs, and API documentation from the API definition. This approach enhances the efficiency of development and integration processes, allowing for more seamless collaboration between frontend and backend teams and ensuring that the API remains well-documented and easy to use.

Currently, mobile application users predominantly use two operating systems, Android and iOS, which account for 99% of the market (TypeScript, 2022; Stasiewicz, 2013). Therefore, the project was based on these systems. For the mobile application development, technologies such as Kotlin Multiplatform Mobile (KMM) for Android and Swift for iOS were considered. However, to optimize development efforts and ensure a consistent user experience across both platforms, a cross-platform framework like Flutter or React Native could be utilized. These frameworks allow for the development of a single codebase that can run on both Android and iOS, significantly reducing development time and maintenance efforts while ensuring a native-like performance and user experience.

4. Software Architecture

Architectural decisions regarding the management support system for cultural institutions include several key elements (Gąbka et al., 2023):

1) Shared backend.
   To ensure simplicity of implementation and ease of maintenance, a monolithic backend architecture was chosen. This model eliminates the need for a complex microservices structure. A monolithic architecture allows for all the backend logic to be contained within a single codebase, simplifying the development and deployment processes. This approach reduces the overhead associated with managing numerous microservices, such as network latency, service discovery, and inter-service communication complexities.
2) Shared relational database.
   The use of a single relational database is crucial for effective data management. This approach facilitates the maintenance of information consistency and enables efficient query processing. Relational databases, such as MySQL, are well-suited for handling structured data and supporting complex queries, transactions, and relationships between data entities. The relational model provides robust support for data integrity and consistency through features like foreign keys and constraints.

3) Storing photos in the file system.

   To optimize the transfer and retrieval of large files, it was decided to store photos for mobile and web clients in the file system. This allows for resource savings in the database, focusing on storing essential information for business functionality. Storing large media files directly in the database can lead to performance bottlenecks and increased storage costs. By using the file system for these files, the system can leverage the file system's inherent efficiency in handling large binary objects. This approach also simplifies backup and recovery processes and can make scaling storage more straightforward.

4) Architecture visualization.

   The model developed by Simon Brown (Brown, 2017) moves away from VML in favor of a less formal method of visualizing architecture. This architecture visualization model proposes representing the architecture in four levels. The following levels are distinguished (Sokół, 2023):

   a) System Context Diagram.

      The The goal here is to present the operating environment of the system. The System Context Diagram provides a high-level overview of how the system interacts with external entities, such as users, external systems, and other stakeholders. This diagram helps stakeholders understand the boundaries and interfaces of the system, highlighting how it fits within the larger ecosystem.

   b) Container diagram.

      Separate containers that are part of the system are shown here. The Container Diagram breaks down the system into its major components or "containers", each responsible for a specific part of the application's functionality. These containers could represent web applications, mobile apps, databases, and external services. This level of detail helps in understanding the modular architecture of the system, facilitating easier management and scaling of individual components. It also aids in identifying how different parts of the system communicate and depend on each other.

   c) Component diagram.

      The Component Diagram delves deeper into each container, detailing the internal components and their interactions. This level of visualization is crucial for developers as it outlines the specific classes, modules, or services within a container and how they collaborate to fulfill the container's responsibilities. By providing a detailed view of the internal structure, the Component Diagram facilitates better design and implementation practices, ensuring that each component is well-defined and that their interactions are clearly understood.

d) Code.
   Finally, a class diagram representing the code is created. This can be a UML class diagram. The Code level provides the most detailed view, illustrating the actual classes, methods, and their relationships within the codebase. This diagram is essential for developers during the implementation phase, as it ensures that the architectural design is accurately translated into the actual code.

Figure 1 shows one of the ways to visualize architecture based on a container diagram.

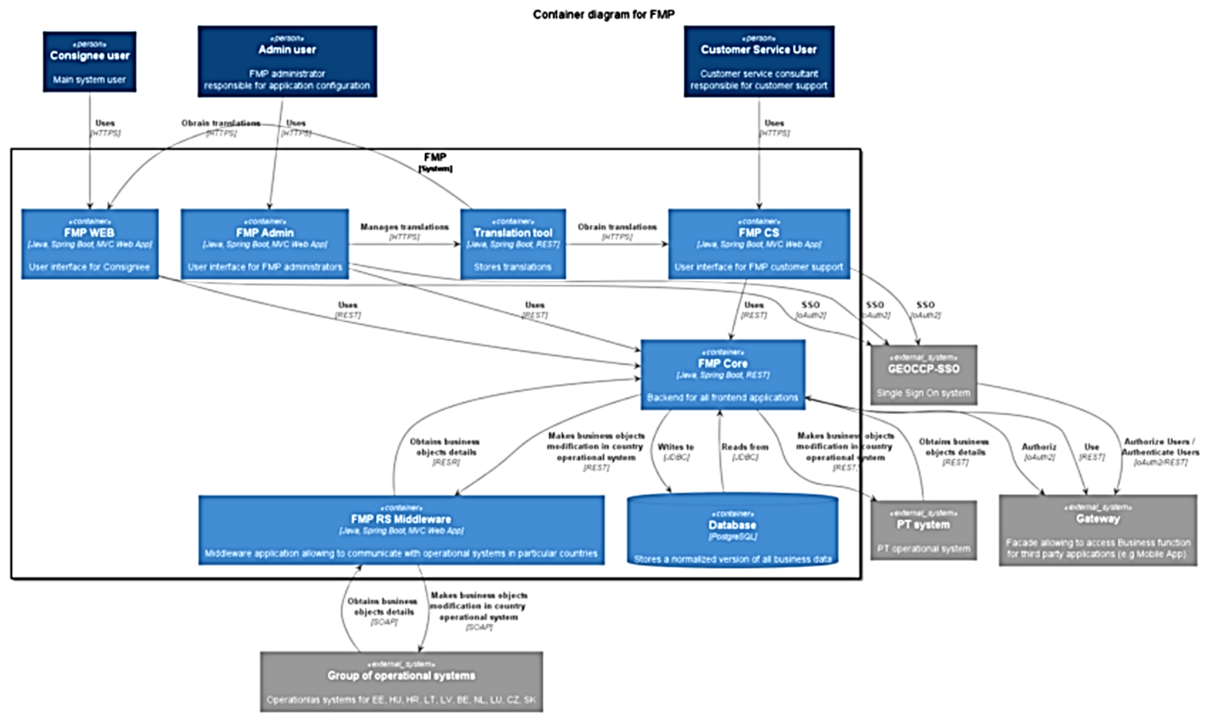
Container diagram for FMP
«person»
Consignee user
Main system user
«person»
Admin user
FMP administrator
responsible for application configuration
«person»
Customer Service User
Customer service consultant
responsible for customer support
Uses
[HTTPS]
Obrain translations
[HTTPS]
Uses
[HTTPS]
Uses
[HTTPS]
FMP
[System]
«container»
FMP WEB
[Java, Spring Boot, MVC Web App]
User interface for Consignee
«container»
FMP Admin
[Java, Spring Boot, MVC Web App]
User interface for FMP administrators
Manages translations
[HTTPS]
«container»
Translation tool
[Java, Spring Boot, REST]
Stores translations
Obrain translations
[HTTPS]
«container»
FMP CS
[Java, Spring Boot, MVC Web App]
User interface for FMP customer support
Uses
[REST]
Uses
[REST]
Uses
[REST]
SSO
[oAuth2]
SSO
[oAuth2]
SSO
[oAuth2]
«container»
FMP Core
[Java, Spring Boot, REST]
Backend for all frontend applications
«external_system»
GEOCCP-SSO
Single Sign On system
Obtains business objects details
[RESR]
Makes business objects modification in country operational system
[REST]
Writes to
[JDBC]
Reads from
[JDBC]
Makes business objects modification in country operational system
[REST]
Obtains business objects details
[REST]
Authoriz
[oAuth2]
Use
[REST]
Authorize Users / Authenticate Users
[oAuth2/REST]
«container»
FMP RS Middleware
[Java, Spring Boot, MVC Web App]
Middleware application allowing to communicate with operational systems in particular countries
«container»
Database
[PostgreSQL]
Stores a normalized version of all business data
«external_system»
PT system
PT operational system
«external_system»
Gateway
Facade allowing to access Business function for third party applications (e.g Mobile App)
Obtains business objects details
[SOAP]
Makes business objects modification in country operational system
[SOAP]
«external_system»
Group of operational systems
Operationias systems for EE, HU, HR, LT, LV, BE, NL, LU, CZ, SK

Figure 1. Container diagram of architecture visualization.

Source: Sokół, 2023.

5. Database system selection

The selection of a database system is a critical component in the architecture of any information system, particularly for managing public utility buildings. For this project, the MySQL database system was chosen (Lis, 2005) as the foundation of the data infrastructure. The selection of MySQL is justified due to several compelling factors: its flexibility, high performance, open-source nature, and widespread familiarity among database designers and developers.

The database was meticulously designed to manage various types of information integral to the operations of cultural institutions. These include user profiles, details of activities and events, announcements, institutional data, and integration points with social media platforms like Facebook. Ensuring data durability was a paramount concern, meaning that information remains available and consistent across all devices, irrespective of where they are accessed. This durability guarantees that all users have access to the latest data, thus maintaining the integrity and reliability of the system.

As part of the database structure selection, A relational architecture was chosen for the database structure due to its proven efficiency in handling complex queries and ensuring data consistency. Relational databases, with their structured schema and support for SQL, are particularly adept at managing intricate relationships between different data entities.

The second critical aspect of the relational database choice is data consistency. Applying the Atomicity-Consistency-Isolation-Durability (ACID) principles in the MySQL database ensures that all transactional operations are safe, consistent, and durable. Atomicity guarantees that each transaction is treated as a single unit, which either fully completes or fully fails, ensuring no partial updates occur. Consistency ensures that the database remains in a valid state before and after the transaction. Isolation ensures that concurrent transactions do not interfere with each other, maintaining data integrity. Durability guarantees that once a transaction is committed, it remains so, even in the case of a system failure.

To effectively represent the data structures, detailed diagrams depicting the tables and their relationships were created. These diagrams illustrate the entities and their connections, providing a clear and organized view of the database schema. This work, as presented by Mazur H. and Mazur Z. (2020), ensures that all entities are properly defined and interlinked, facilitating efficient data retrieval and manipulation. These diagrams serve as a blueprint for the database, guiding the development and maintenance of the system.

6. Interaction with the Meta API

The Meta environment (Meta, 2023) was chosen as the social media platform for system integration. The system allows for the automatic creation of content on Facebook and Instagram, corresponding to posts created in the application. To achieve this, the Meta API was used.

The Meta API is based on the GraphOL standard (Lembo et al., 2022). This is a query language developed by Facebook that allows for precise specification of what data should be retrieved in each query. GraphOL enables dynamic determination of the structure of returned data, which reduces network traffic between the system and Meta services. By allowing developers to specify the exact data needed, GraphOL minimizes unnecessary data transfer and optimizes the performance of the application. This precision in data retrieval is crucial for maintaining the efficiency and responsiveness of the integrated system.

The first step of integration is registering a new application in the Meta developer panel (Developer, 2023). Along with the existing application in the developer panel, the Facebook Login procedure (Facebook, 2023) was used to generate tokens. Appropriate permission scopes were specified when generating the tokens. This data allows the initial goal to be fully achieved.

After generating the data required for verification, the post generation stage can begin. This process involves creating posts on the Facebook platform. The system automatically generates content based on predefined templates and user inputs, ensuring that the posts are consistent and aligned with the institution's branding and messaging guidelines. Similarly, updating content on Facebook is done by deleting the old post and creating a new one, according to the presented algorithm. This approach ensures that the latest information is always available to the audience, and outdated content is efficiently removed to prevent confusion.

7. Summary

In the area of communication, the system we introduced, which enables the publication of information related to the activities of cultural institutions, was highly rated by users. This functionality was effectively implemented, contributing to efficient communication between the cultural institution and its local community. Users appreciated the timely updates and the seamless integration with social media platforms like Facebook and Instagram, which significantly enhanced the visibility and reach of the institutions' activities. The system's ability to send email notifications further ensured that users remained informed about upcoming events and any changes, fostering a more engaged and informed community.

In the area of reservations, all necessary functionalities were properly introduced. The reservation system was designed to handle a variety of needs, including sign-ups for activities, ticket reservations, and participation in events. The ease of use and reliability of the reservation system received positive feedback from users. The system's intuitive interface allowed users to make reservations quickly and efficiently, reducing the administrative burden on the institutions. Moreover, the system's ability to manage high volumes of reservations without performance degradation demonstrated its robustness and scalability.

In the area of control, the creation of individual user profiles was enabled. This feature allowed users to freely browse their reservations and monitor payment statuses, providing a personalized and transparent experience. The user profile functionality also supported enhanced data management and security, as it enabled the system to track user interactions and preferences accurately. Users could update their profiles, view their reservation history, and check the status of their payments, which improved their overall experience and satisfaction. Additionally, the system's implementation of access controls ensured that user data was protected, adhering to best practices in data privacy and security.

The presented solution was positively evaluated by users. The feedback highlighted the system's effectiveness in meeting the needs of cultural institutions and their audiences. Users praised the system for its user-friendly design, reliability, and comprehensive features. The successful integration of communication, reservation, and control functionalities into a single cohesive system demonstrated the effectiveness of the architectural decisions made during development.

References

1. Bampakos, A., Deeleman, P. (2023). *Poznaj Angular.* Gliwice: Helion.
2. Brown, S. (2017). *Visualise, Document and Explore Your Software Architecture.* Realm Academy.
3. Developers (2023). Retrieved from: https://developers.facebook.com/docs/development/create-an-app/, 10.10.2023.
4. Facebook Login (2023). Retrieved from: https://developers.facebook.com/docs/facebook-login/, 10.10.2023.
5. Gąbka, M., Kiełczyński, K., Kruszyna, R., Puchała, P. (2023). *Aplikacja mobilna i webowa do wspomagania zarządzania domami kultury.* Wrocław: Politechnika Wrocławska.
6. Gough, J., Bryant, D., Auburn, M. (2020). *Architektura API. Projektowanie, używanie i rozwijanie systemów opartych na API.* Gliwice: Helion.
7. Jemerov, D., Isakova, S. (2018). *Kotlin w akcji.* Gliwice: Helion.
8. Kotlin (2023). Retrieved from: https://www.w3schools.com/kotlin, 20.10.2023.
9. Lembo, D., Santanelli, V., Savo, D.F., De Giacomo, G., (2022). GraphOL: A Graphical Language for Ontology Modelling Equivalent to OWL 2. *Informatics Conference on Modern Trends in Multiagent Systems.* Italy, doi:10.3390/fi14030078.
10. Lis, M. (2005). *PHP i MySQL dla każdego.* Gliwice: Helion.
11. Mazur, H. (2020). *Metodyka strukturalna projektowania relacyjnych baz danych.* Wrocław: Wydawnictwo Politechniki Wrocławskiej.
12. Meta (2023). Retrieved from: https://transparency.fb.com/pl-pl/research tools/meta-content-library/, 10.10.2023.
13. Sokół, T. (2023). Retrieved from: https://tomasokol.pl/wizualizacja-architektury-zgodnie-z-modelem-c4-podejscie-praktyczne/, 20.10.2023.
14. Stasiewicz, A. (2013). *Android. Podstawy tworzenia aplikacji.* Gliwice: Helion.
15. TypeScript (2022). Retrieved from: https://2022.stateofis.com/en-US/usage/, 10.10.2023.
16. Vanderkam, D. (2020). *TypeScript: Skuteczne programowanie.* Warszawa: Promise.
17. Walls, C. (2015). *Spring Boot in Action.* Gliwice: Helion.