The ALICE Service Work system is an app developed by the Glance Team that is responsible for providing an easy way for coordinators to manage and display such tasks and assignments. It is aimed to allow a straightforward development, with simple implementations and minimum coupling.

By implementing the ALICE Service Work system, the collaboration aims to reduce time spent on administrative tasks and allocate more resources to the experimental aspects of the job.

2 Foundation

Developing applications for a collaboration of about 2000 members makes crucial the adherence to best practices in order to ensure their effectiveness and success. Due to the characteristics of the endeavor, the Domain-Driven Design (DDD) [6] tied to the Hexagonal architecture [7] is the foundation of the ALICE Service Work system. With this approach we can achieve modular and scalable architecture that promotes flexible and maintainable codebases. Moreover, it is imperative to prioritize user-centered design principles throughout the development process to ensure the app's full success and widespread acceptance within the collaboration. In the subsequent subsections we will explain the principles of each methodology and explain how their integration enhances the reliability of the ALICE Service Work system.

2.1 Domain-Driven Design

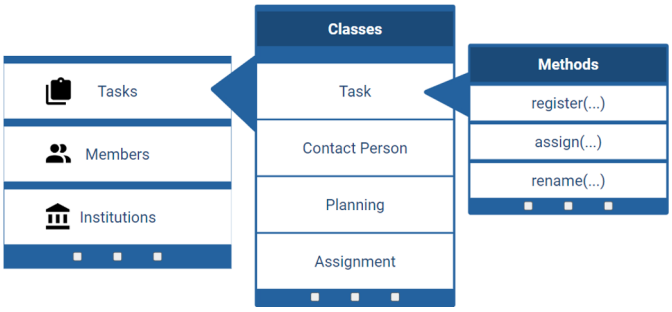

Figure 1: Bounded contexts in the Service Work system.

Domain-Driven Design (DDD) is a software development approach that emphasizes understanding the business domain and aligning the software design with it. It promotes the use of a shared language, known as “ubiquitous language”, to bridge the gap between technical and business domains. As part of this implementation for the Service Work system, we aim to divide its complex functionalities into smaller bounded contexts, with each having its own classes and methods, as described in Figure 1. It also demonstrates a consistent nomenclature for classes and methods, ensuring alignment with end users' and stakeholders' perceptions. DDD benefits collaborations by aligning the software design with unique business requirements and facilitating closer collaboration between the development team and domain experts. While DDD requires domain expertise, its benefits include improved software quality, reduced complexity and increased maintainability.

2.2 Hexagonal architecture

The Hexagonal architecture, also known as Ports and Adapters, is a software architectural pattern that isolates the core business logic from external dependencies, as shown in Figure 2. It divides the application into three layers: core application, ports and adapters. The core application layer contains the domain model and business logic, while the ports layer de nes interfaces for interacting with external systems. The adapters layer implements these interfaces, adapting the external systems to work with the core application. The Hexagonal architecture provides bene ts such as exibility, maintainability and testability by enabling easy replacement of external systems, promoting modular code and facilitating isolated testing of the core logic.

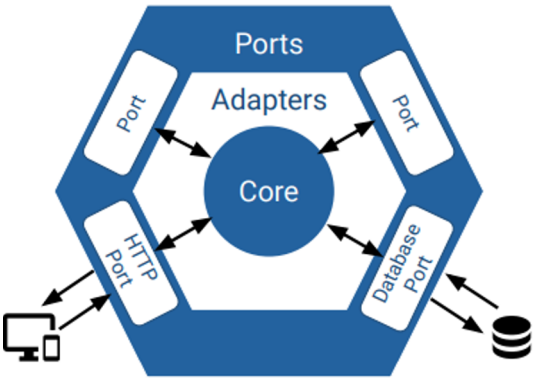

Figure 2: The hexagonal structure.

2.3 User-Centered Design

User-Centered Design (UCD) [8] is an approach that prioritizes the needs and preferences of end-users throughout the development process. It emphasizes understanding users goals, behaviors and expectations to create intuitive and user-friendly interfaces. UCD involves conducting user research, usability testing and iterative design to continuously improve the user experience. By incorporating UCD principles, collaborations can ensure the success and acceptance of the application among users. It promotes effective communication, reduces misunderstandings and enhances user satisfaction. User-Centered Design is essential for creating interfaces that align with user requirements and integrate smoothly with existing work ows.

3 Research methodology

The development of the ALICE Service Work system relies heavily on continuous communication with its end users. The aim is to gather valuable information on system usage and user needs, which is then used to drive the development process. The iterative nature of the development allows for the implementation of user-suggested solutions, making the app adaptable and responsive to user requirements.

End users are actively encouraged to provide feedback on their experience, usage patterns and any challenges they encountered. Different communication tools, such as Mattermost and

Outlook are constantly being used to keep in contact with end-users as well as direct reports through JIRA [9] task creations built in the app.

To assess the prioritization of incoming requests within the system, requests are categorized based on prede ned criteria, such as urgency, impact and complexity. Each request is assigned a priority level after a brief discussion between the scrum product owner and the stakeholders, enabling the system to allocate appropriate resources and address high-priority issues promptly.

4 Development and implementation

The development of the web application for the ALICE Service Work system followed a systematic and iterative approach. The project began with a thorough analysis of the experiment requirements, including the needs of collaborators, team leaders and administrators. This involved conducting interviews to gather insights into the desired features and functionalities.

Based on the gathered requirements, the development team proceeded with the design phase. User-centered design principles were employed to ensure the usability of the web app and intuitive navigation. Wireframes were created and validated through iterative feedback loops involving stakeholders.

After the design phase, the development team moved on to the implementation stage, with its schema represented in Figure 3. They utilized modern PHP web development frameworks and technologies to create a robust and scalable web application that serves as endpoints for HTTP protocol data manipulations that comes from a JavaScript frontend user interface. During the development process, they also prioritized security and privacy in line with CERN's requirements. This involved implementing measures like access controls and secure Single Sign-On (SSO) authentication provided by the IT department. The Slim framework [10] was

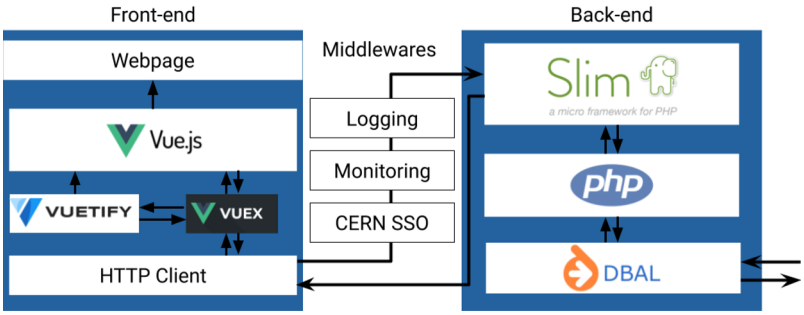

Figure 3: Service Work implementation.

used to build the web application's infrastructure with a minimalistic yet powerful foundation for the RESTful API, with possibility for handling routing and middlewares.

To interact with an Oracle database (also provided by the CERN IT infrastructure) the team incorporated the Doctrine DBAL library [11], a database abstraction layer that simpli es database access query building and result handling, enabling seamless integration with Oracle and ensuring e cient and secure data operations.

In the frontend, to enhance the user interface and provide a seamless and responsive experience, the development team incorporated the Vue.js framework [12], along with Vuetify.js [13] and Vuex libraries [14].

Vue.js allowed for the creation of dynamic and interactive user interfaces, enabling a smooth and reactive user experience. The team leveraged Vue.js components and directives to build reusable UI elements and implement frontend logic.

Vuetify.js, a Material Design component framework for Vue.js, was used to improve the visual aesthetics and ensure consistent styling throughout the web app. Vuetify's pre-built components and theming capabilities expedited the frontend development process.

Vuex, a state management pattern and library for Vue.js, facilitated centralized state management and provided a scalable approach to manage application data. With Vuex, the team efficiently handled shared data and implemented actions, mutations and getters to maintain application state.

By integrating Vue.js with Vuetify.js and Vuex, the ALICE Service Work system achieved a modern and responsive user interface, seamless data flow and enhanced frontend functionality.

4.1 Implementation challenges and solutions

During the implementation phase, several challenges were encountered. One significant challenge was the integration of the application with existing CERN internal solutions, such as the SSO for authentication. With the possibility of introducing middlewares with Slim, the solution could be made modular and shared through all the Glance team projects, reducing the amount of time needed. A second example could be the management of the dynamic relationships between ever-changing data elements, where modifying one data point could trigger chain reactions, requiring a reliable solution to handle these ties automatically. To address this challenge, we introduced an event-driven architecture. In this setup, specific key actions prompt predefined responses or workflows within the system, eliminating the need for a fixed, step-by-step sequence of operations. These workflows adjust dynamically to the context, ensuring adaptable and efficient handling of various scenarios and enhancing the system's overall flexibility and efficiency.

4.2 User feedback and satisfaction

Once the app was deployed, user feedback was collected to assess its effectiveness and user satisfaction. Interviews were conducted with coordinators and managers to gather their opinions and suggestions for improvement.

5 Findings and discussion

After the analysis of all the requirements, we came up with a list of key features and functionalities for the Service Work system, which are described below.

5.1 Key features and functionalities

5.1.1 Task management

The web app offers a robust task management system, see Figure 4, that empowers users to efficiently create, assign, and track service work tasks throughout the year. These tasks can be conveniently categorized based on the relevant projects and activities, allowing effective organization and streamlined workflow.

Tasks can be assigned to collaborators in a many-to-many relationship. Each assignment includes associated credits, representing a fraction or the total Full Time Equivalent (FTE) of

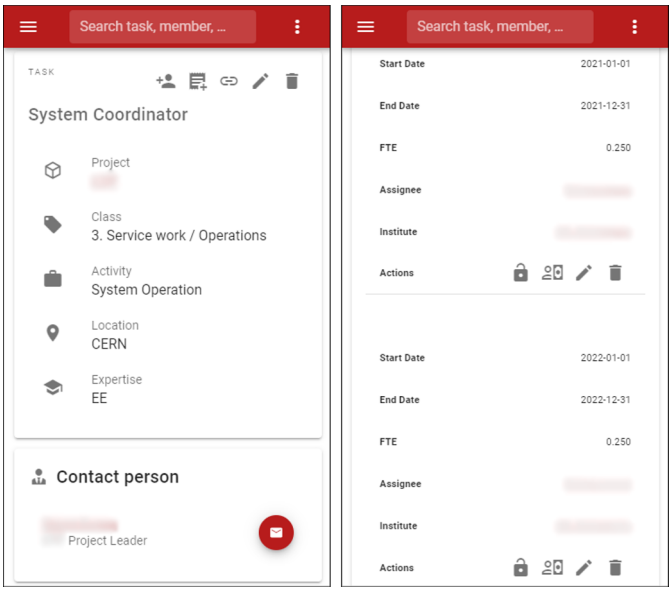

Figure 4: Task interface example.

the assignee, along with a designated period of validity. This critical information is utilized to calculate individual workloads, preventing members from becoming overburdened with an excessive amount of tasks.

5.1.2 Year planning

The app o ers a comprehensive year planning module that empowers coordinators to create and manage the work calendar ectively. With this module, coordinators can de ne the annual workload of the collaboration in advance, set FTE thresholds to determine if teams are underbooked or overbooked, and restrict certain actions, such as creating new assignments. Additionally, the year planning module provides coordinators with a view of the current situation and evolution of the entire collaboration, enabling them to make informed decisions and e ectively manage resources.

5.1.3 Credits accounting

The Credits accounting feature is a vital component of the system, responsible for accurately calculating work contributions. Each task within the system has its own set of assignments and plannings, where assignments are associated with speci c credit values. It ensures that the total sum of assignment credits during a designated planning period aligns with the planned credits for that period. These credits are attributed to the assignee's respective team, helping them progress towards their annual credit goal.

5.2 Usability and User Experience Analysis

The feedback received from users was largely positive. Team leaders appreciated the user-friendly interface, streamlined access to their team's tasks data and the ability to assign duties

to them. Overall, the user feedback indicated a high level of satisfaction with the app, highlighting its effectiveness in improving operational efficiency. However, some users suggested minor improvements to enhance performance issues and data gathering.

5.3 Identification of challenges

Despite its numerous benefits, the web application also faces few challenges that need to be addressed for optimal performance and user satisfaction and which are described below.

5.3.1 Increased data import options

One of the identified challenges is the need for increased data import options within the web application. Currently, there are instances where specific task assignment data originates from external sources, such as the ALICE collaboration website and human resources databases. However, the system does not fully cover the range of possible data entry points, resulting in the need for manual insertion into the database using direct SQL queries. This manual process introduces the risk of human errors and consumes valuable time that could be better utilized for development tasks. By providing support for a wider range of data import options, the application can reduce human errors associated with manual data entry and allow the team to allocate more time towards development tasks.

5.3.2 Better integration with external apps

The service works are just a part of all the activities done within the ALICE Collaboration, other kinds of tasks such as experimental shifts in the control room or on-call, for example, are credited via the ALICE Shift Accounting Management System (SAMS), also developed by the Glance Team. One of the main need of the collaboration is the possibility that information about both types of tasks be shared between these systems in a dashboard, enabling a broader visualization of the total work done by the teams. However, the challenge lies in the different timeline of the two systems' development. The SAMS system being considerably older than the Service Work system has resulted in integration difficulties due to legacy implementations and data inconsistency.

6 Conclusion

In conclusion, the ALICE Service Work system has successfully addressed the challenges associated with managing a large number of tasks within the ALICE collaboration at CERN. By adopting Domain-Driven Design (DDD) and the Hexagonal structure, the system exhibits a modular and scalable architecture that promotes flexibility and maintainability. The user-centered design principles implemented throughout the development process have resulted in an intuitive and user-friendly interface.

Through continuous communication with end-users, the development team gathered valuable feedback and implemented the solutions suggested by the users, resulting in an adaptable and responsive system. The key features and functionalities of the system, have proven to be effective in organizing and displaying service work activities.

Although the development process encountered challenges, the findings provide valuable insights for future improvements. Recommendations include implementing agnostic assignment import, enhancing the user interface and user experience, standardizing data and exploring integration with existing systems.

Overall, the ALICE Service Work system has significantly improved coordination and management within the collaboration, reducing administrative burdens and enhancing operational efficiency. By providing a user-friendly platform, the system allows for increased focus on experimental aspects. Future improvements based on the identified challenges and recommendations will further optimize the system and support the collaboration goals and needs.

Acknowledgments

The authors would like to thank CNPq, CAPES, FAPERJ and RENAFAE (Brazil), as well as CERN and the ALICE collaboration for providing financial support for this work. We also thank Mr. Joel Closier for supporting the preparation of the presentation and related paper.

References

[1] The ALICE Collaboration, JINST 3, S08002 (2008)
[2] Welcome to the ALICE Collaboration - General Information, URL https://alice-collaboration.web.cern.ch/general-information [accessed 21-Jul-2023]
[3] C. Maidantchik, F.F. Grael, K.K. Galvão, K. Pommès, J. Phys.: Conf. Ser. 119, 042020 (2008)
[4] The ATLAS Collaboration, JINST 3, S08003 (2008).
[5] The LHCb collaboration. JINST 3, S08005 (2008).
[6] E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software. (Addison-Wesley Professional, Boston, 2004)
[7] A. Cockburn, Hexagonal Architecture (2005), URL https://alistair.cockburn.us/hexagonal-architecture/ [accessed 21-Jul-2023]
[8] D. Norman, J. Nielsen. The Definition of User Experience (Nielsen Norman Group, 2010), URL https://www.nngroup.com/articles/definition-user-experience/ [accessed 21-Jul-2023]
[9] Atlassian. Jira Software, URL https://www.atlassian.com/software/jira [accessed 08-Aug-2023]
[10] The Slim Framework Team. Slim Framework, URL https://www.slimframework.com/ [accessed 21-Jul-2023]
[11] Doctrine Project. Database Abstraction Layer, URL https://www.doctrine-project.org/projects/dbal.html [accessed 21-Jul-2023]
[12] Vue.js - The Progressive JavaScript Framework, URL https://www.vuejs.org/ [accessed 21-Jul-2023]
[13] Vuetify - A Vue Component Framework, URL https://vuetifyjs.com/en/ [accessed 21-Jul-2023]
[14] What is Vuex?, URL https://vuex.vuejs.org/ [accessed 21-Jul-2023]