The ATLAS Glance Qualification system's refactoring strategy

Ana Clara Loureiro Cruz, Carolina Niklaus Moreira da Rocha Rodrigues,
Gabriel Marçal Mendonça, Gabriela Lemos Lúcidi Pinhão, Leonardo Mira Marins, Pedro Henrique Goes Afonso,
Rafaella Lenzi Romano, and Rodrigo Coura Torres on behalf of the ATLAS Computing Activity

Introduction

Context

Considering the dynamic environment of the ATLAS Experiment at CERN, an efficient communication among its members is essential. So, in 2003, the Glance Project was launched to streamline data management and daily tasks for CERN's LHC experiments.

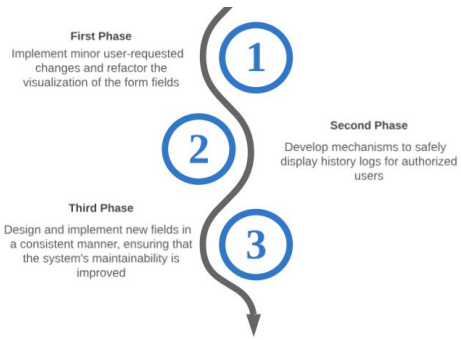
The Authorship Qualification system, created in 2014 using an in-house built framework called Fence [1], was designed to manage the ATLAS authorship qualification process for a person to become an author.

Although Fence initially fulfilled the system's needs with custom data management, it became outdated, complicating maintenance and expansion. Now, the system needs to migrate from Fence to a more modern and scalable architecture.

Deprecated system overview

Aspect	Method	Issues
Read use cases	Implemented using Fence	Outdated methods and rigid architecture.
Write use cases	Implemented using Doctrine ORM [2]	Inconsistent with reading architecture, and with the team's developments stack.
Overall structure	Mixed use of Fence and Doctrine ORM	Fragmented system, complicating maintenance and further development

Methodology



Development

The new structure

Given the necessity to migrate to a more modular and testable structure, aligning with modern software development practices, an hexagonal architecture based on Domain-Driven Design (DDD) was applied. The result of the new design is the creation of smaller and self-contained contexts, reducing technical complexity of the system.

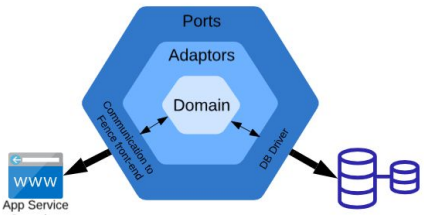


Figure 1: Hexagonal architecture structure.

To complete the migration with transparency to the end user, Fence's front-end was kept, and the controller was used to format the data of the new architecture in the same way as the old system. This allowed the back-end to be refactored without changing the front-end.

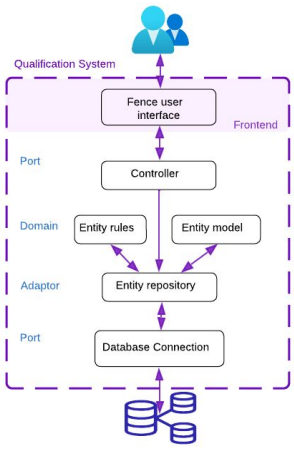


Figure 2: Refactored Qualification System structure.

The entity repositories being written in pure SQL skips the need for an abstraction layer on top of the database. Without this abstraction, data manipulation becomes more transparent for developers, making it easier to implement future changes.

Conclusion

Challenges encountered

Refactoring a system, such as the ATLAS Qualification System, requires a significant amount of development resources in each Scrum sprint [3]. To avoid exhausting all our resources on a single project, the team had to make pauses between the different phases. Managing these long project timelines became a real challenge. Also, migrating to different code architectures is not a simple process since it is necessary to ensure that all functionalities work the same after changes in the system.

Project Status

Results

- **Code quality:** the migration creates a more consistent architecture, matching the team's development stack.
- **Better responsiveness:** the migration allows developers to understand the system better and resolve bugs faster.
- **Automatic tests:** the refactoring process also made it possible to implement automatic tests, which were crucial in avoiding bugs.
- **Phases Strategy:** delivering the migration in versions allows risk mitigation, incremental improvements, and early feedback, ensuring testing and validation at each stage.

References

[1]Bruno Lange *et al* 2015 *J. Phys.: Conf. Ser.* 664 062026
[2]*Getting started with doctrine* (no date) *Doctrine Object Relational Mapper (ORM)*. Available at: https://www.doctrine-project.org/projects/doctrine-orm/en/3.2/tutorials/getting-started.html#what-is-doctrine (Accessed: 30 September 2024).
[3] LEMOS LÚCIDI PINHÃO, GABRIELA. "Enhancing product management in the ATLAS Management Glance team"