

DESIGN OF THE INFORMATION SYSTEM FOR MONITORING THE QUALITY OF THE UNIVERSITY

Michal Drobný

Abstract:

This article further analyzes the requirements for evaluating the internal quality system at Pan-European University (PEU), which are defined by PEU internal directives based on the conditions described in Act no. 269/2018 ensuring the quality of higher education and amending Act no. 343/2015 Coll. on public procurement and the amendment of specific laws as amended and based on this analysis, design and implement the information system that will serve to manage the internal quality system at PEU. The following chapters describe the entire process of creating this information system, from analysis, through implementation and testing to deployment in production and use at PEU, as well as how the system's future development should be managed.

Keywords:

Information system, a system of quality, evaluation of quality.

ACM Computing Classification System:

Software and its engineering. Software creation and management.

■ Introduction

The beginning of the 2022/2023 academic year brought the task for universities in Slovakia to adapt their existing study programs (hereinafter referred to as "programs") as well as habilitation and inauguration procedures (hereinafter referred to as "procedures") to the new Higher Education Act and the Quality Act. At the same time, it is necessary to prepare for a new method of accreditation of programs and procedures by universities.

Since this obligation arose for universities, an initiative was created at Pan-European University in Bratislava (hereinafter referred to as "PEU") to manage this process at PEU and its faculties more sophisticatedly compared to the circulation and approval of paper forms.

After processing the requirements for the management of this process, it was decided to create a custom information system that would reflect the needs associated with this process and, at the same time, be wholly owned by PEU.

The quality system defines how the university will ensure systematic and structured care for the quality of education, related creative activities, and maintenance and improvement. However, a recent change in legislation is changing the current way of managing and evaluating the quality of higher education. This change is directly related to the accreditation of study programs and procedures.

Law no. 269/2018 on ensuring the quality of higher education and amending Act no. 343/2015 Coll. on public procurement and on the amendment of specific laws as amended (hereinafter referred to as "Act No. 269/2018 Coll.") brought with it the creation of the Slovak Accreditation Agency for Higher Education (hereinafter called the "Agency").

This Agency oversees assessing the internal quality systems of individual universities and, based on this assessment, accrediting individual programs and procedures. It also complements the legislative framework, as determined by Act no. 269/2018 Coll.

The quality system at PEU is governed by the following internal directives and decrees, created upon legislation need based on Act No. 269/2018 Coll. Their directives are:

- Directive no. 1/2022 - Statute of the board for internal quality system evaluation
- Directive no. 2/2022 - Internal quality assurance system
- Directive no. 3/2022 - Evaluation of educational activity by students and graduates
- Directive no. 4/2022 - Rules for creating and implementing study program adjustments
- Quality indicators and their calculation

A new body, the Quality Council, was created to evaluate the internal quality system of PEU. The competence of the council is determined by Act no. 269/2018 Coll., the Agency's standards, the PEU statute, and its internal regulations, which are listed above. "The quality council has at least seven members". [1, p. 4] Only a person who is a recognized professional authority, a graduate, or a student who has achieved good academic results during their previous studies at PEU can be appointed as a member of the Quality Council.

For harmonizing programs and procedures, these programs are evaluated by a recognized external authority (hereinafter referred to as "an evaluator"). Their evaluation is part of the quality evaluation, and the documents created by the evaluators are an integral part of the documents, defined as self-assessment reports. This report is the basis for the responsible Agency in the accreditation process of individual programs and procedures. On its base, the university and individual faculties are granted accreditation for individual programs and procedures.

The following chapters describe the process of introducing process of creating the information system that serves quality management at the PEU.

1 System Design

The new information system for quality approval (hereinafter referred to as "the information system") was designed based on legislative requirements already mentioned in this article's introduction chapter. The proposal also included requirements from the future users of the information system.

The design of the information system should reflect these requirements and be implemented into one functional unit so that this system not only meets expectations but is also expandable and sustainable in the future.

1.1 Timetable

At the latest, PEU and its faculties had to harmonize their programs and procedures with legislative requirements by August 31st, 2022. Since the deadline that PEU had to meet was precisely defined, the schedule for implementing the information system was also subject to this deadline.

The university amended its internal directives, which created a prerequisite for fulfilling the legislative requirements as of April 1st, 2022. Subsequently, the Quality council was established to supervise the quality approval process. Individual faculties have started preparing the necessary documentation for unique programs and procedures.

In April 2022, a functional specification was created describing the process of approving programs and procedures by the Quality Council and the body of evaluators, as it should have been implemented in the information system.

This specification also describes individual functional elements of the information system users will work with. Great importance was placed on the security of the information system and its auditability. The implementation of the information system began at the beginning of May 2022. The first version was made available to users and testers at the end of May 2022.

During June 2022, the change requirements resulting from testing were implemented into the system, and bugs were fixed so that the system was ready for production launch on June 23rd, 2022 when the first official meeting of the Quality Council at PEU took place.

The following picture (Fig.1) describes the implementation timetable in main milestones.

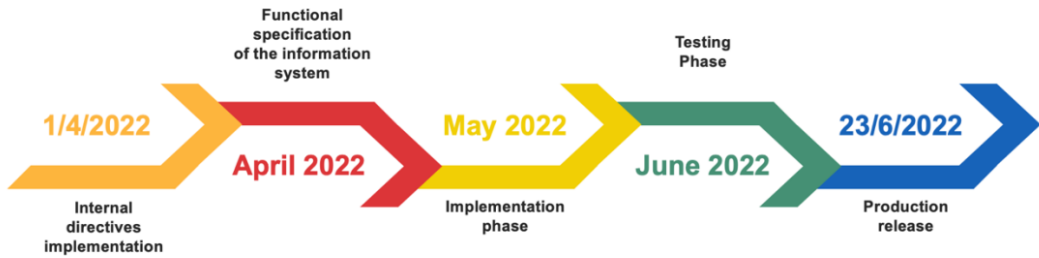


Fig.1. Implementation timetable.

1.2 Choice of Technology

Developing any information system involves making critical decisions about the technology stack used throughout the project. A well-thought-out decision process ensures that the chosen technologies align with the project's goals, meet the requirements, and contribute to the application's long-term success. In this section, we outline the decision-making process that led us to choose **PHP** with the **Laravel** framework and **MySQL** as the database for the information system.

Understanding the project requirements

Before evaluating different technologies, it is crucial to understand the project requirements clearly. Our information system needed to be scalable, secure, and easy to maintain while offering excellent performance. Additionally, we had a relatively short development timeframe.

Comparing various technology options

We compared several popular programming languages, frameworks, and databases to understand their suitability for our project. The contenders included Python with Django, Ruby with Rails, JavaScript with Node.js and Express, and PHP with Laravel. We also considered databases such as MySQL, PostgreSQL, and MongoDB.

Evaluating the PHP and Laravel combination

PHP is a widely used, open-source scripting language designed for web development. It offers several advantages, such as:

- **Mature ecosystem** (long history with extensive libraries and resources)
- **Affordability** (thanks to open-source code and wide availability of qualified developers)

- **Compatibility** (with different web servers and operating systems which facilitates deployment and integration with other technologies)

Laravel is a robust PHP framework that simplifies development by providing a clean and elegant syntax. It offers several benefits, such as:

- **Rapid development** (built-in tools and libraries such as Artisan, Eloquent ORM, and the Blade templating engine help speed up the development process)
- **Scalability** (horizontal scaling through queuing and buffering mechanisms that ensure our application can handle growing traffic demands)
- **Security** (robust security features such as protection against SQL injection and cross-site scripting, which ensure the security of our application)

Choosing MySQL as the database

“MySQL is a popular, open-source relational database management system (RDBMS)” [2, 2023]. It is known for its reliability, performance, and ease of use. We chose MySQL for the following reasons:

- **Compatibility** - MySQL is highly compatible with PHP and Laravel, making integrating with our chosen technology stack easy.
- **Cost-effective** - MySQL is an open-source solution with many cost-effective hosting options.
- **Performance** - MySQL offers excellent performance with indexing and caching mechanisms, ensuring quick data retrieval and storage for our application.

After thoroughly evaluating different technologies, we concluded that PHP with the Laravel framework and MySQL as the database was the optimal choice for the information system. This combination offers a cost-effective, scalable, and secure solution that aligns with our project requirements and ensures a smooth development process.

1.3 Data Model

Having decided to use a MySQL database for the information system, the clear choice was implementing the relational data model. This model allows us to efficiently organize and manage structured data in the database using tables consisting of rows and columns. The relational model also allows us to represent relationships between data entities using primary and foreign keys. This enables efficient querying and manipulation of data using a declarative language, such as Structured Query Language, represented via ORM integrated with Laravel Framework.

With this approach, we can achieve higher consistency, integrity, and normalization of data, which is essential for various applications, such as transactional systems, reporting, and data warehouses. As a result, implementing the relational data model in the information system will enable better data management and optimization.

Since our data model is pervasive, for this article, we will only summarize the most crucial data entities:

- **Academic titles** store information about academic titles, such as professors or assistant professors.
- **Activity log records** user activities, such as login/logout, page views, and other actions taken on the system.
- **Audit logs record** audit logs, such as changes to user roles, permissions, or access.

- **Document templates** store document templates indented from the documents necessary for the accreditation process.
- **Documents** store documents uploaded to the system.
- **Document versions** store document versions of uploaded documents.
- **Form of studies** stores information about forms of study, such as external or daily.
- **Hip approvals** record approvals for procedures accredited in institutions.
- **Hips** store information about procedures accredited in institutions.
- **Institutions** store information about educational institutions such as universities and faculties.
- **Level of studies** stores information about levels of study, such as undergraduate or graduate.
- **Field of studies** stores information about fields of study.
- **Self-assessment report approvals** record approvals for self-assessment reports.
- **Self-assessment reports** store self-assessment reports.
- **Study program approvals** record approvals for study programs.
- **Study programs** store information about study programs.
- **Fields of study** store fields of study accredited in instructions.

Overall, this data model is designed to support the management of the information system and all required functionalities specified in the functional specification.

2 Implementation

In the implementation phase of the development, we focused on turning the design and plans into a functioning system. We wrote the necessary code, created and integrated databases, and implemented all the required features and functionality.

We also conducted extensive testing and debugging to ensure the information system worked as expected. Once the information system was ready, we deployed it to production and made it available to the users. In the following part of the article process of implementing REST API and user interface will be explained in more detail.

2.1 REST API

REST API is considered a best practice for API communication in Laravel because it follows principles that make it more efficient and effective for exchanging data between systems. These principles include a stateless, client-server architecture, standardized HTTP methods for accessing resources, and the use of URLs to identify resources as stands in “What is REST API” [3, 2023]. This design provides a more organized and predictable way to communicate between systems and allows for greater flexibility in the types of requests and responses that can be used.

In the application, we used a REST API because it allowed us to easily create endpoints for different types of resources and utilize standardized HTTP methods for accessing and manipulating data. This allowed for more consistent and predictable interaction with our backend system, making developing and maintaining our application easier. Additionally, the use of REST API allowed us to easily integrate with other systems that also utilize this design, making it easier to expand and grow our application in the future.

The naming convention used in our application was modified, and we did not use the standard naming convention for REST APIs. The main reason for modifying the naming convention was code readability for future development. Our backend also modifies the data processing to make the REST API usable with the front end, to provide all data for blade templates, and to be used when changing the frontend framework in the future.

2.2 User Interface

The user interface was divided into two parts: user and administrative. Each part was designed independently to meet the requirements specified in the functional specification. The user interface was designed with a focus on ease of use and providing a positive user experience, while the administrative interface was designed with a focus on functionality and ease of management. This approach allowed us to tailor each interface to the needs of its respective user group, ultimately leading to a more effective and efficient system. Additionally, it facilitated easier maintenance and updates in the future, as changes to one interface would not impact the other.

User interface

For the user interface, we have decided to use Laravel Blade. Implementing an interface with Laravel Blade involves creating and structuring views that are returned as HTTP responses to user requests. Laravel Blade is a templating engine that provides a simple yet powerful way to create reusable HTML and PHP code. Blade templates use a special syntax that enables us to define layout files, include partial views, and pass data to views.

One of the advantages of using the request-response way of implementing UI instead of asynchronous JavaScript calls is that it enables us to easily create reusable UI components with no strict binding to any browser-side rendering. These components can be defined as Blade templates and included in other views, making it easy to create a consistent look and feel across our application. Additionally, Blade templates can be extended to create master layout files that define the overall structure of an application's UI. As our user interface is mainly structured as an Admin Template, usage of Master Layout was the right way how to structure our Frontend application.

Overall, implementing a user interface with Laravel Blade provided us with a powerful and flexible way to create reusable UI components, define master layout files, and generate clean and semantic HTML. These benefits, along with Laravel's robust routing and controller features, helped us implement a flexible user interface with a high focus on development speed.

Admin interface

We have decided to use Laravel Nova for generating the administration interface due to the pervasive data model that our information system is using. This decision allowed us to generate CRUD for all data entities implemented in the application and enable their fulfillment through the administration interface. Thanks to CRUD, it is possible to modify and delete these entities according to the needs and functional specifications of the application.

Laravel Nova provides an easy-to-use and customizable interface for managing data entities. It allows us to create, read, update, and delete records in our database. Additionally, it will enable us to manage relationships between entities and provides a search functionality that makes it easier to find specific records.

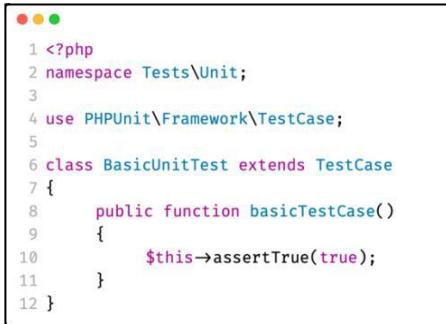
Using Laravel Nova for the administration interface provides several benefits, including reducing development time and increasing productivity. With Laravel Nova, we did not have to spend time creating custom administration interfaces for each entity. This allowed us to focus on other vital parts of the application, such as the user interface.

In summary, using Laravel Nova for the administration interface has been a wise decision for the information system. It allowed us to generate CRUD for all data entities, easily manage relationships, and access valuable features like metrics, custom filters, and custom actions. Additionally, it has helped us reduce development time and increase productivity, allowing us to focus on other essential parts of the application.

3 Testing

Our framework of choice, Laravel, provides several possibilities for automated testing, including Unit tests and Feature tests. Unit tests, when PHP Unit or PEST is used, allow for testing of individual methods and classes, while Feature tests allow for testing of functionality across multiple methods or classes.

One of the primary reasons we chose PEST as our testing framework and not PHP Unit tests is its simplicity and ease of use. PEST provides a more readable syntax and allows for faster writing of tests, as can be seen in (Fig.2) vs. (Fig.3).

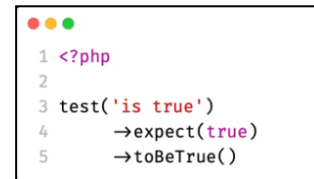


```

1 <?php
2 namespace Tests\Unit;
3
4 use PHPUnit\Framework\TestCase;
5
6 class BasicUnitTest extends TestCase
7 {
8     public function basicTestCase()
9     {
10         $this->assertTrue(true);
11     }
12 }

```

Fig.2. Basic PHP Unit Test.



```

1 <?php
2
3 test('is true')
4     ->expect(true)
5     ->toBeTrue()

```

Fig.3. Basic PEST Test.

It also integrates well with Laravel and supports Laravel's testing helpers, such as `actingAs()`, `assertDatabaseHas()`, and `assertSee()`. PEST also provides a test suite feature, which allows for the grouping of related tests and easier management of tests as the application grows.

Another benefit of using PEST is that it provides more helpful error messages, making it easier to identify and fix issues in the code. PEST also allows for parallel testing, which can significantly speed up testing times on larger projects.

PEST supports test-driven development (TDD) by providing a built-in `--watch` flag that automatically runs tests as changes are made to the code. This feature encourages developers to write tests before writing code, making it easier to identify issues early in the development process; even TDD was not used while developing the information system, this development pattern can help us, if used, with implementing new features without worrying about introduction of complex bugs into the information system.

The user interface underwent primary testing through manual methods. Users who were involved in the project were given the application for testing after the implementation of all functionalities. Complex test scenarios were excluded from the testing process.

The purpose of this approach was to identify any potentially poorly designed processes. By focusing on manual testing, users could better examine the environment and its functionality in a more natural setting.

As a result, this testing method allowed for a more comprehensive understanding of the user interface. By working within the environment and interacting with its features, users were able to uncover any flaws or issues that might not have been apparent during more complex testing scenarios.

4 Discussion

During the design and development of the information system, it was discovered that it is essential to include all materials processed in the system in the electronic form to streamline

the entire approval process. The implementation of electronic documents brought many benefits, particularly in the speed and efficiency of information processing.

Each document that must undergo the approval process retains a detailed history of what happened to it and who commented on it. This digital trail ensures transparency and traceability of operations, leading to improved communication and collaboration among involved parties.

The digitization process brought significant improvements to the speed of the approval process for programs and procedures and time savings for the PEU. This acceleration allowed for the streamlining of work for employees, the Quality Council, and Approvers, leading to better resource utilization and an increase in the quality of services provided by the information system itself.

■ Conclusion

The implementation of the application has brought significant benefits to PEU, including the alleviation of paper bureaucracy. This digital transformation of the quality monitoring process has streamlined processes and improved efficiency while also providing a more user-friendly experience for approvers, the Quality Council, and all personnel involved in the approval and accreditation process.

The application will continue to be expanded, adding new functionalities to enhance its capabilities. Future developments include the implementation of AI to optimize processes and creating documents for programs and procedures directly within the application. This innovation will allow for even more significant time savings and resource optimization.

■ Acknowledgement

Numerous individuals contributed to the successful completion of this project. We are grateful to everyone who participated in this endeavor. We want to extend our appreciation to **RNDr. Ján Lacko, PhD.**, and **doc. Ing. Zuzana Vincúrová, PhD.**, whose ideas significantly contributed to making this project a reality. Their assistance in developing detailed specifications for the application and designing the process as it exists today has been invaluable.

■ References

- [1] Pan-European University, Directive no.1/2022, 2022
- [2] Craig S. Mullins, July 27th , 2015, accessed April 29th, 2023. <https://www.techtarget.com/search-datamanagement/feature/MySQL-open-source-RDBMS-overview>
- [3] What is a REST API? 2023, accessed April 29th, 2023, <https://www.ibm.com/topics/rest-apis>

■ Authors



Michal Drobný

Pan-European University/Faculty of Informatics, Bratislava, Slovakia
xdrobny@paneurouni.com