FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	DevOps concepts - CI/CD, implementation of authorization & authentication, presented on a BI-DBS portal frontend
Student:	Volha Chukava
Supervisor:	Ing. Oldřich Malec
Study program:	Informatics
Branch / specialization:	Web and Software Engineering, specialization Web Engineering
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2023/2024

Instructions

The BI-DBS portal undergoes an evolutionary transfer to a microservice architecture with a Vue.js frontend. Therefore, the frontend needs new automated deployment, testing, and a clear permissions structure.

Fulfill the requirements:
- Describe the current and planned state of the BI-DBS portal.
- Analyze and describe DevOps principles, focus on CI/CD.
- Configure and describe an automated deployment for the BI-DBS portal.
- Analyze roles and permissions of the BI-DBS portal.
- Implement and describe the authorization and authentication services and provide suitable regression tests for a CI pipeline.

Electronically approved by Ing. Jaroslav Kuchař, Ph.D. on 26 January 2023 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

DevOps concepts - CI/CD, implementation of authorization & authentication, presented on a BI-DBS portal frontend

Volha Chukava

Department of Software Engineering
Supervisor: Ing. Oldřich Malec

April 25, 2023

Acknowledgements

I would like to thank ...

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on April 25, 2023 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Chukava, Volha. *DevOps concepts - CI/CD, implementation of authorization & authentication, presented on a BI-DBS portal frontend.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Abstrakt

Pro předmět BI-DBS na FIT ČVUT v Praze je vyvíjena nová aplikace. Požadavky pro tuto aplikaci jsou odvozeny z již existující aplikace. Tato bakalářská práce se zaměřuje na zlepšení procesů vývoje a údržby nového frontendu webové aplikace na základě analýzy stavu stávající aplikace a plánovaného stavu. Hlavními oblastmi zlepšení jsou přijetí metodiky DevOps a navržení nového zjednodušeného a přehledného systému řízení přístupů. Nakonec hlavním cílem této práce je implementace autentizace a autorizace, včetně omezení oprávnění, a automatizace procesů testování a nasazování.

Klíčová slova webová aplikace, frontend, CI, CD, OAuth, autentizace, autorizace

Abstract

A new application is being developed for teaching the BI-DBS subject at FIT CTU in Prague. The requirements for the application are derived from the existing application. This bachelor's thesis is focused on improving the development and maintenance processes of the new frontend for the web application based on analyses of the state of the existing application and the planned state. The main improvement points are adopting DevOps methodology and designing a new simplified and clear access management system. Conclusively, the main goal of the thesis is the implementation of authentication and authorization including restricting permissions, and automation of testing and deployment.

Keywords web application, frontend, DevOps, CI, CD, OAuth, authentication, authorizarion

Contents

List of Figures

List of Tables

Introduction

This thesis is a continuation of my development journey of the BI-DBS web application, which started in February 2022 as a part of the frontend development team. The project was very complex and challenging for maintenance and development. Not a long time after our team got into analyzing and developing the current application, a new solution to the maintenance problem was introduced to us by Ing. Andrii Plyskach. He came up with a new architectural design based on microservices architecture for the application, which he described in his master thesis[1]. The decision was made to follow that design, which meant creating a new project based on the requirements of the current application. The development process, as well as the project, was divided into client and server side parts.
For our team, it meant starting frontend development from the very beginning using the new technologies. Thus it gave us a lot of space for our ideas, but also a big responsibility. Therefore, since the creation of a new frontend project, my goal was to configure the project in a way to make it structured and secure, organize efficient development in a team as well as work on the implementation of the features. This goal remains the same for this thesis.
Firstly, I am going to analyze the current application state and the planned state, to see what we can expect from the changes. Secondly, for improving the efficiency of the software development process and application improvement and maintenance I will introduce the DevOps model and the instruction for its adoption. Besides, I will use that instruction for automating testing and development processes for the frontend. Finally, I am going to implement authorization and implementation based on OAuth 2.0 protocol[2] using the authorization microservice implemented in Andrii's thesis[1].

Chapter 1

Analysis of the application state

In this chapter, I will introduce the educational web application helping to teach database systems subject at the university. Furthermore, I will describe the current and planned state of the application from the perspective of software architectural patterns and the used set of core technologies with a focus on the frontend. The goal is to identify the existing problems of the current application, outline how they will be solved in a new portal, as well as to indicate what difficulties we can face developing the new application using a new stack of technologies and new architecture.

1.1 The BI-DBS portal

The BI-DBS portal is a web application used for teaching database systems subject in a bachelor's study program at the Czech Technical University at the Faculty of Information Technology. The portal is complex and has many useful functionalities. It allows managing and tracking all the student's studying progress during the semester, including semester tests, complex semester work, and exams. Besides, teachers have an overview of all their student's work in one application.
The application is developed by students and teachers in subjects BI-SP1, BI-SP2 subjects, bachelor and master theses. That is a unique fact about this project. Every year, new students begin working on application development. They are open to sharing their ideas for improving the application. Thus, we are designing and implementing a better and better product each year.

1.2 Current state of the application

The current BI-DBS portal was deployed for the first time in 2016[3]. Over time it gained new features and grew large. Currently, it has a total of around

120000-140000 lines of code[1]. Used technologies became less relevant and it became difficult to maintain it.

1.2.1 Architecture

The current application was built in a traditional way, using a monolithic architecture approach and following the Model-View-Presenter architectural pattern[4][5]. Figure 1.1 shows the visualization of the architecture. The application is presented as one monolithic unit, and it is composed of three components.

- *The model:* Communicates with the database and handles domain and business logic.
- *The view:* Provides visualization and directs user commands to the presenter, does not contain logic.
- *The presenter:* Manages interactions between the database and the view. Receives data from the model and formats it to display in the view.

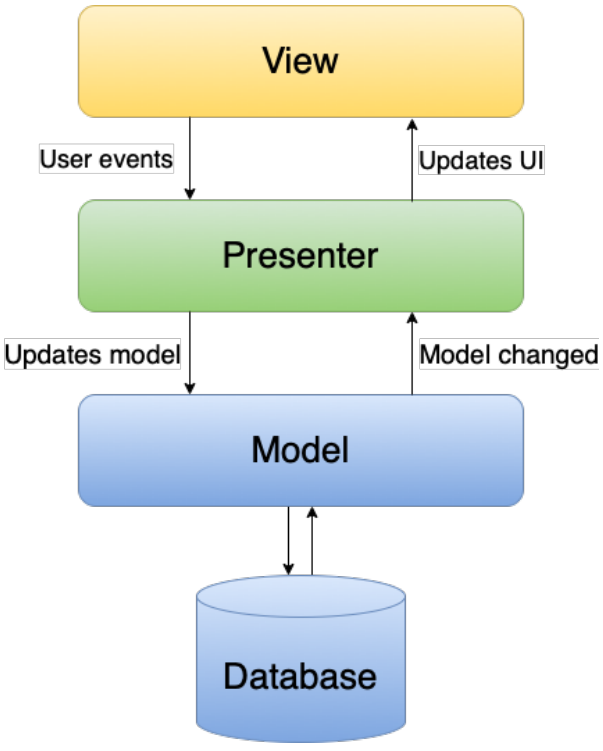

Figure 1.1: Monolithic architecture, MVP pattern

This architecture's main concept is having one code base that benefits in simplifying development, testing, debugging, and deployment.
However, we can have those benefits only until the application grows large. Then all those processes get slower, more complex, and become problematic. In addition, with a lack of flexibility and scalability, it becomes challenging to maintain the application and keep it secure.

The BI-DBS portal is being developed by students. Students generally do not have much experience developing large applications and dealing with complex dependencies. Besides, they have limited time to progress in learning and then designing and developing the portal. Therefore it takes a lot of time for students to learn before contributing to the project. Thus it is more problematic for students to benefit from learning and for maintainers to keep it functioning correctly.

1.2.2 Technologies

PHP. PHP is a general-purpose, open-source scripting language that can be integrated into HTML.[6][7] It differs from client-side scripting languages in that its HTML is generated on a server and then sent to a client. That feature allows rapidly building a web application with a thick server and thin client. This is one of the approaches to using PHP to build an application, and it is used in the current project.
Using this approach leads to creating dependencies between the user interface and the application logic, which makes any changes more effortful since a developer needs to adjust it on both sides.

Doctrine. "Doctrine ORM is an object-relational mapper for PHP 7.1+ that provides transparent persistence for PHP objects. It sits on top of a powerful database abstraction layer. One of its key features is the option to write database queries in a proprietary object-oriented SQL dialect called Doctrine Query Language."[8]
This framework did not cause problems during the development process and has no significant disadvantages for the correct operation of the BI-DBS.

Nette. Nette is an open-source framework for creating web applications in PHP. It helps with developing both the client and server sides of the application and also reduces security vulnerabilities. Moreover, it manages application states using sessions and routing.[9]
Frontend and backend dependencies are strengthened, indicating that they are a single unit. The fact that they are so strongly dependent is a drawback. Because of this, it is difficult to make changes to one side without having an impact on the other.

Latte. Nette framework uses a template system called Latte. It compiles templates down to the optimal PHP code.[10]

AdminLTE. AdminLTE is a fully responsive administration template. Based on Bootstrap 4.6 framework and also the JS/jQuery plugin.[11]

Vue 2. Vue.js is a javascript framework for building user interfaces and single-page applications.
Most of the frontend is implemented using Latte templates and AdminLTE bootstrap. However, in order to reduce dependencies between the frontend and the backend and also modernize it, a few components were refactored to the Vue.js version 2. The logic is defined using the Options API. It is a traditional object-oriented way, and up until Vue 2 it was the only way to create components in Vue.[12, 13]

Javascript. JavaScript is a high-level programming language used for defining the behavior of webpages. It is a dynamically-typed scripting language that lets you control multimedia, animate graphics, and generate dynamically changing content.[14]
In the current BI-DBS portal it is used for defining logic on the frontend. Dynamically-typed languages are easy for development, but this feature reduces the code's readability, requires more testing and are prone to run-time errors. Large applications like BI-DBS are likely to experience problems as a result of its drawbacks because it is better suited for smaller applications with simple logic.

Webpack. Primarily, Webpack is a static module bundler for modern JavaScript programs. When Webpack processes your application, it internally creates a dependency tree from one or more entry points and then merges every module your project requires into one or more bundles.[15]
Webpack is used in a current application for bundling a few modernized frontend components implemented in Vue.js and javascript.

1.3 Planned state of the application

The main reason for creating a new application instead of refactoring the current one is a change in the application's architecture. A new modernized architectural design of the BI-DBS portal was composed by Ing. Andrii Plyskach in his master thesis[1]. We are aiming to correct all the mistakes made in the current application. It is essential to ascertain that we have chosen the right stack of technologies according to the newly chosen architecture.

1.3.1 Architecture

Microservices architectural pattern[5] is based on the concept of a series of loosely-coupled services. They can be developed using different technologies and deployed independently. It is more complex architecture than a standard monolithic one. You can see the diagram illustrating microservices architecture in Figure 1.2.

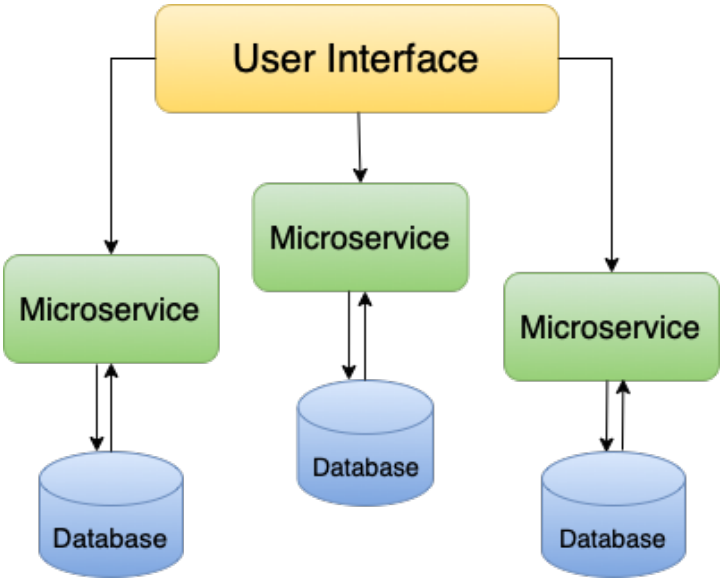

Figure 1.2: Microservice architecture

Advantages:

- *Code readability:* When services are not strongly dependent the code appears to be better structured and easier to understand, which is a significant benefit for the BI-DBS portal.
- *Independency in choosing a stack of technologies:* Microservices can be developed using different technologies which can be chosen according to each microservice functionality without affecting other microservices.
- *Faster deployments:* Since all microservices can be deployed independently, the deploying part is much smaller and the time for deploying one service is rather shorter.
- *Fault tolerance:* Because of loose-coupling, failing one of the microservices will not bring down the entire application.

Disadvantages:

- *Difficult debugging and testing:* Each service needs to be first tested separately and only then as one unit. Besides, it is more difficult to track down errors.
- *DevOps required:* To benefit from the fast deployment it should be configured and maintained. It requires knowledge of development operations.
- *Longer development time and limited reuse of code:* Microservices need to be managed separately, therefore it requires more time.

1.3.2 Technologies

PHP. Since version 5.0, PHP supports object-oriented functionality[16]. PHP is easy to learn, flexible, and supports all required functionalities for our application. It is used in a new project for a domain and business logic on the backend for API implementation.

Symfony. Symfony is a powerful backend framework for creating complex applications which consists of reusable components.[17] Thanks to Doctrine Symfony provides all of the tools required to use databases in the application. It is constantly growing and improving, besides it has a strong community. It is easy to learn and has well-written documentation.

Vue 3. When the decision to create a new BI-DBS portal had not yet been made, its frontend was getting modernized by rewriting components to Vue.js version 2. In the new project, it was chosen to carry on using the Vue.js framework but use a new version 3. This version comes with certain advantages for the application.[18]

New features:

- *Composition API:* Composition API is a set of APIs that allow us to create Vue components by importing functions rather than defining options. It mainly benefits our project in better code organization, thus making the project better structured and the code easier to read. Moreover, Composition API enables efficient logic reuse.[19]
- *Vite:* Vite is a frontend build tool from the creator of Vue.js - Evan You. It is a module bundler that bundles the entire project on startup, hot reloads, and compilation. The primary purpose for the change is for speed. The server starts instantly since it uses native browser support for JavaScript modules.[20]

- *Increased rendering performance.*
- *Smaller Vue core.*

Typescript Typescript is based on JavaScript, which is dynamically typed. TypeScript has an additional syntax that makes it statically typed. That has advantages in catching errors during development. It also gives a code more structure and makes it predictable. Typescript is more suitable for big applications than JavaScript. For our project, writing code that will be easy to read is crucial.[21]

Quasar Quasar is a web framework based on Vue.js. It provides us with ready-to-use components which are customizable and easy-extendable. Moreover, it makes the application less vulnerable to XSS attacks due to its escaping feature. When using Quasar, developers do not need deep knowledge of CSS and scripting languages to build a good-looking and responsive application. Besides, it is suitable for developing single-page applications(SPA).[22]

Pinia Pinia is a Vue.js storage library and state management framework. It is mainly designed for the development of front-end web applications, and it uses declarative syntax as well as its own state management API. It allows sharing a state across components and pages securely. Moreover, it has server-side rendering support. With Pinia plugins we can also persist the state across page reloading using local or session storage.[23]

1.4 Summary and implications

The BI-DBS development team aims to dispose of problems and modernize the current project in every single aspect of development. Starting from choosing the right stack of technologies and designing a suitable architecture to implementing more complex and valuable features. However, even correctly chosen technologies and architecture for reducing the problems of the current project do not save us from the potential new challenges brought by the changes.

1.4.1 Summary

In order to summarize all changes and provide a better visualization of them, I arranged them all in Table 1.1.
Evidently, the Nette framework is the core of the current project, which is responsible for managing the application in many ways. Although it can function well, it creates dependencies between the functionalities and makes the project less flexible, which is a significant disadvantage for large applications like the BI-DBS portal.

The planned state does not have dominating technologies that would cause this problem. Most of them are replaceable and flexible.

	Current state	**Planned state**
Architecture	Monolitic	Microservices
Backend language	PHP 7.2	PHP 8.0
Backend framework	Nette	Symfony
Frontend framework	Nette, Vue.js 2	Vue.js 3
Frontend templates and styling	Latte, AdminLTE	Quasar
Scripting language	Javascript	Typescript
Frontend templates and styling	Latte, AdminLTE	Quasar
Module bundler	Webpack	Vite
State management	Nette Sessions	Pinia
Routing	Nette Router	Vue.js Router

Table 1.1: Visualisation of changes.

1.4.2 Implications

From the analyses in sections 1.2 and 1.3, we can see that existing problems in the current project are eliminated by chosen architecture and technologies for the planned state. Let's examine the main possible negative effects of the changes and how to deal with them.

- Microservices architecture provides such advantages as agility and fast deployment. This architecture is more complex in comparison with a monolithic one. Therefore it comes along with establishing some of the DevOps principles for the project. Mainly configuring continuous integration and automated deployment. DevOps concepts and automation including CI and CD are described in the second chapter.

- In the current application, Nette is a core full-stack framework that is also responsible for managing the application's security. Besides, the monolithic architecture allows you to store all the data on the server side. The communication between the client side and server side is secure.
  In microservices applications, there is constant communication between the frontend and the backend and exchanging sensitive data. Therefore it is crucial to control every step of that communication with control of permissions and inputs validation on both the client and server sides. Thus the application should have a clearly defined access management system which I will introduce in the third chapter.

- Since all the services are developed, deployed and tested separately, there is a higher chance of failure in communication between them. Obviously,

It is not enough to test only the functionalities of singles services but to test their integration. Therefore it is essential to design a new integration testing system for the application. It is necessary to eliminate the possibility of the cascade failure of services.

Chapter 2

Analysis of the DevOps model

The DevOps and microservices are two important trends in application development. Considering that both of them are mainly focused on providing better agility, flexibility, and operational efficiency, we can assume that they would work better together.[24]
In this chapter, I will describe the main DevOps concepts and practices, analyze their possible advantages for the BI-DBS application and decide whether adopting the DevOps model would be beneficial.

2.1 What is DevOps?

The term **DevOps** is derived from the combination of software **dev**elopment and IT **op**eration**s**.
DevOps is a relatively new term. Around 2007 and 2008 concerns about the separate work of software creators and software operators were raised. The concept started to grow on online forums and meet-ups. The first conference named "DevOps" was held in 2009.[25]
DevOps is a software development methodology composed of a set of cultural philosophies, practices, and tools that improve an organization's ability to deliver applications, services, and improve products faster than traditional software development and infrastructure management processes. It represents a cultural shift that significantly affects a team that adopted that methodology and the software they make.[26]

2.2 DevOps concepts

DevOps concepts are a common set of rules which are the core of this methodology. It is not just a set of tools, but a cultural philosophy, a way of project lifecycle organization. Those rules are not strictly defined, they come from a DevOps culture and can be interpreted differently describing the same model.

In this section, I will analyze and combine the culture philosophy and most frequently mentioned rules[27, 28] in five concepts that represent the DevOps methodology.

2.2.1 Automation

DevOps approach is meant to benefit in fast development and improvement. Needless to mention that automation is one of the golden rules for increasing the speed of the application lifecycle. Everyone in a team should aim to automate as many phases of the process as it is possible. As a result, team members are satisfied with a decreased need for doing repetitive tasks. Thus they can focus on significant tasks and work on new features. Overall it helps minimize human errors and boosts team output.

Usage in the BI-DBS project This concept was the main reason for me to consider adopting DevOps model in the BI-DBS project. Due to microservices architecture the project needs to have new automated deployment and testing processes. These and other automation practices we might want to adopt are described in subsection 2.3.

2.2.2 Data-Based Decision Making

With the DevOps approach, decisions from choosing a technology stack for the application to adding features should be made based on collected data. The first part of making a decision should always be collecting as much relevant data as it is possible. Then, based on the collected data analysis of the team, a decision should be made. It helps to create software that solves real problems effectively. Decisions made without considering client feedback data, colleagues' opinions, and proper analysis would lead to creating badly-functional software full of useless features which does not fulfill the client's needs.

Usage in the BI-DBS project This concept is very suitable for our new growing project since in the current phase we create are creating a core which should be done properly based on analyses of collected data to avoid having useless features, too complex design and irrelevant technologies.

2.2.3 Responsibility Throughout the Lifecycle

DevOps methodology comes with a requirement for team members to fully understand the process of software development from the feature idea to implementation and deployment and take responsibility for it. End-to-end responsibility helps to reduce failures and resolve bugs quickly.

Usage in the BI-DBS project From my experience, students usually want to finish their part of the job as fast as it is possible. Therefore they sometimes tend to skip spending time to understand the idea of the task properly. Thus they get to implement it without thinking of the consequences their changes might cause. Moreover, they do not always get to test it properly. It is essential to integrate this concept more into development student teams to increase the quality of produced code.

2.2.4 Constant Improvement

Constant improvement is a special concept and practice of DevOps methodology. The main idea is a focus on improvements, updates, and experimenting. It tells each team member not to be afraid of failures but take them as an opportunity to learn. Whatever the outcome of an experiment, a person will have a deeper understanding of what works and what does not. Besides, this rule gives more responsibility to a person and allows them to consistently push code changes to minimize waste, do speed optimization and improve development efficiency.

Usage in the BI-DBS project This concept is friendly for students in a way that they can try new things without fear of failure if things do not work out. Adopting this concept will benefit the project in case it is used with the two previous concepts, otherwise, students might take the development less seriously and do experiments only for the purpose of faster finishing the task, but not improving.

2.2.5 Collaboration

This concept is a collaboration of different IT departments on the project. That means that the team's roles are not as strict and independent as in a traditional work and team organization. Developer and operation roles are getting closer to full-stack roles, leading to a better understanding of the software development lifecycle by the whole team.

Usage in the BI-DBS project This illustrating DevOps methodology concept is beneficial in two ways for the BI-DBS portal. Firstly, students will learn essential operation processes and understand the basics of automation. Secondly, the portal always needs at least one person to be available to manage application operations. Using this concept will increase the number of people who understand the processes and thus are able to manage operations in case of a need.

2.3 DevOps cycle and practices

DevOps concepts reflect in a set of practices during the DevOps delivery cycle. The cycle is visualized in Figure 2.1. These practices mainly represent the automation concept but also come together with other concepts.[29]

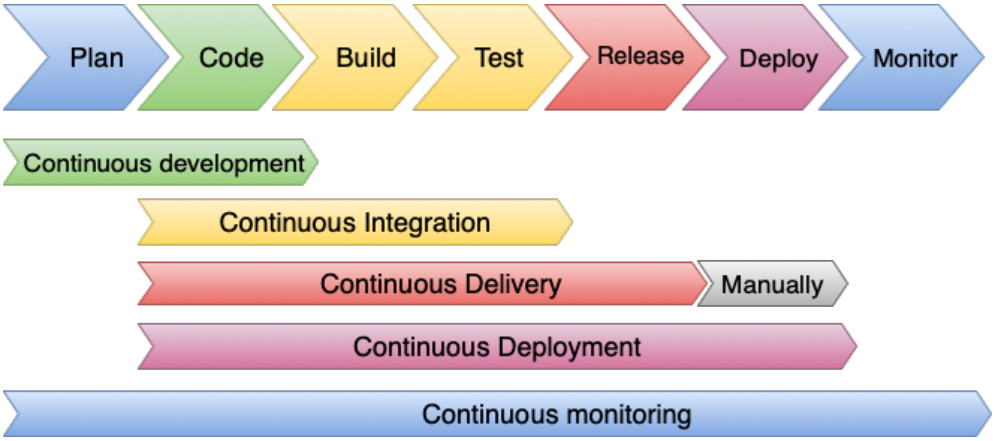

Figure 2.1: Devops cycle and practices

2.3.1 Continuous development

Continuous development is a practice composed of agile planning and coding. The goal of agile planning is to divide big problems into smaller logical problems, estimate the complexity of created tasks and plan the amount of tasks for some short time period(sprint), usually it is from one to four weeks period. This method allows getting some large significant tasks done in a shorter time, because after its division developers can work on the subtasks simultaneously and it is more effortless for testing.

2.3.2 Continuous integration (CI)

In order to avoid a problem with the integration of large parts of code, DevOps CI offers continuous pushing the code changes to the remote shared repository on the server. Every change pushed to the repository triggers a build and tests configured in a CI pipeline to make sure new changes do not affect already functioning features and also does not contain new errors.

2.3.3 Continuous delivery (CD)

Continuous delivery is an extension of continuous integration. After building and testing the code from the repository it automatically deploys releases to the testing environment and also prepares it to be deployed to the production.

It requires human intervention to deploy a release to production. This is a safer version of fast and frequent deployment, in a case when the pipeline does not contain strong testing tools and the application needs to be tested manually.

2.3.4 Continuous deployment (CDE)

Coupled with continuous delivery, the continuous deployment also deploys the release to the production. Using this practice no human intervention is required. Every change pushed to the main shared remote repository will be automatically deployed to production. The only obstacle to the deployment would be a failed build or test.

2.3.5 Continuous monitoring (CM)

Continuous monitoring is an automated method that allows to observe and discover compliance concerns and security vulnerabilities throughout the DevOps lifecycle. It also finalizes the cycle by providing feedback on monitoring and informing about existing or possible failures. It helps to resolve issues in real-time.

2.3.6 Infrastructure as Code (IaC)

Infrastructure as a code is a practice of managing infrastructure that enables automation in the DevOps lifecycle. It offers using scripts for configuring deployment environment and other infrastructure, including establishing a version control system.

2.3.7 Containerization

Containerization is the practice of packaging an application in one container. It provides better flexibility for deployment and needs fewer resources to run. Currently, Docker provides the most frequently used container toolset.

2.4 DevOps adoption

The idea of adopting the DevOps model came to me with a need to configure the new deployment of the new BI-DBS portal due to the transfer to microservices architecture. Before analyzing the DevOps concepts, I assumed that the DevOps model is just an automation idea. In fact, I was wrong and did not know it is a solid methodology bringing huge advantages to the project. From my own observations, it is a pretty common misunderstanding of the DevOps model, which leads to missing out important concepts.

"Even while automation helps speed up manual operations, cooperation and communication are the key objectives of DevOps. Automating your operations won't bring about the desired business benefits unless everyone involved in the software development, delivery, testing, and operating processes adopts excellent communication and collaborative practices."[30]
The analysis makes it clear that the DevOps model is suitable and valuable for the BI-DBS portal project management and development.

Adoption steps:

1. *Devops philosophy.* This thesis can be used to introduce the DevOps methodology to students. Before getting to development as well as learning the processes of development students should learn team organization management including DevOps concepts.

2. *DevOps Practices.* Analyze which practice we would like to adopt and how it will be beneficial and then complete the three next steps:

   a) Choosing relevant tools for a practice we would like to adopt

   b) Application of the practice using chosen tools

   c) Document the configuration of the practice for a team

   I will adopt the most important DevOps practices for the BI-DBS portal in the automation chapter using these steps.

Chapter 3

Analysis and design of the access management system

In this chapter I will describe the access management system designed for the new BI-DBS portal frontend, based on roles and permissions analysis. The goal is to simplify the access management system if it is possible and provide an overview of the permissions for the roles.

3.1 Roles

3.1.1 KOS roles

The BI-DBS portal receives information about an authorized user from the study information system(KOS) based on the course information. The course information contains the identifier of the semester and the type of study program. Generally, there are eight user roles that are defined by the KOS for subjects and courses.[31]

KOS roles for subjects and their general description:[32]

- *Guarantor.* Guarantor is a course administrator. Thus a person with this role typically has all permissions across all course management.
- *Examiner.* Examiners are responsible for managing and estimating students' exams. Therefore, this role would usually provide an access to exam materials and students' grades view and management.
- *Editor.* Editor is a person who can edit the information about the subject. This role would provide an access to subject information management.

- *Lecturer.* Lecturer role indicates that an individual with this role would need to be provided with access to manage course content including lectures, assignments and other study materials.

- *Instructor.* Instructor is the role for teachers of exercises parallels. This role implies that a user needs permission to manage exercise materials and also estimate students' tests, semester works and other assignments.

- *Laboratory instructor.* Laboratory instructor is an instructor who does teaching in the laboratories, which means that they may need to be provided with additional access to hardware and software management of the laboratory.

- *Teacher.* The teacher is a general role for a person, who does teaching in the course.

- *Student.* It is a base role for students, that usually grants basic permissions to a user like access to their personal information, study program information and its resources, and also allows managing their projects and submitting assignments.

Two of those roles are not used for the BI-DBS portal. The first of them is the laboratory instructor, as the BI-DBS subject study program does not include laboratory exercises. The second one is the editor, the BI-DBS portal does not provide functionalities for changing the information about the subject. Therefore, we have a total of six roles used for permissions control in the current application.
From these six roles in fact. Based on the feedback from the teachers and developers of the current BI-DBS portal and also my own research, we came to the conclusion that the access management system can be simplified by grouping the roles into three hierarchical layers.

Reasons for simplifying the access management:

1. Based on the permissions research of the current project, I can report that most of the roles for teachers have the same or almost the same permissions. And the differences are insignificant.

2. All the analyses and requirements of the BI-DBS portal designed by students including the theses are made for only three roles.

3. Due to a lack of information about the roles developers often do not have a good understanding of all roles meaning and thus they tend to forget to permit success for some roles or confuse them with others.

I am offering the simplification, which is based on grouping teaching roles that do not have significant permissions differences such as lecturer, examiner, instructor and teacher into one role. As I have already mentioned above, all the analyses are usually built on three roles where these 4 roles are taken as one role - teacher role, then there is a student and guarantor roles left which makes it a total of three.
After researching the functionalities and permissions structure of the current project and discussing the possible change with developers and managers, I came to the conclusion that it is absolutely safe to generalize the permissions for those four teacher roles. As the result, I got a new clear KOS roles structure, which is visualized in Figure 3.1.

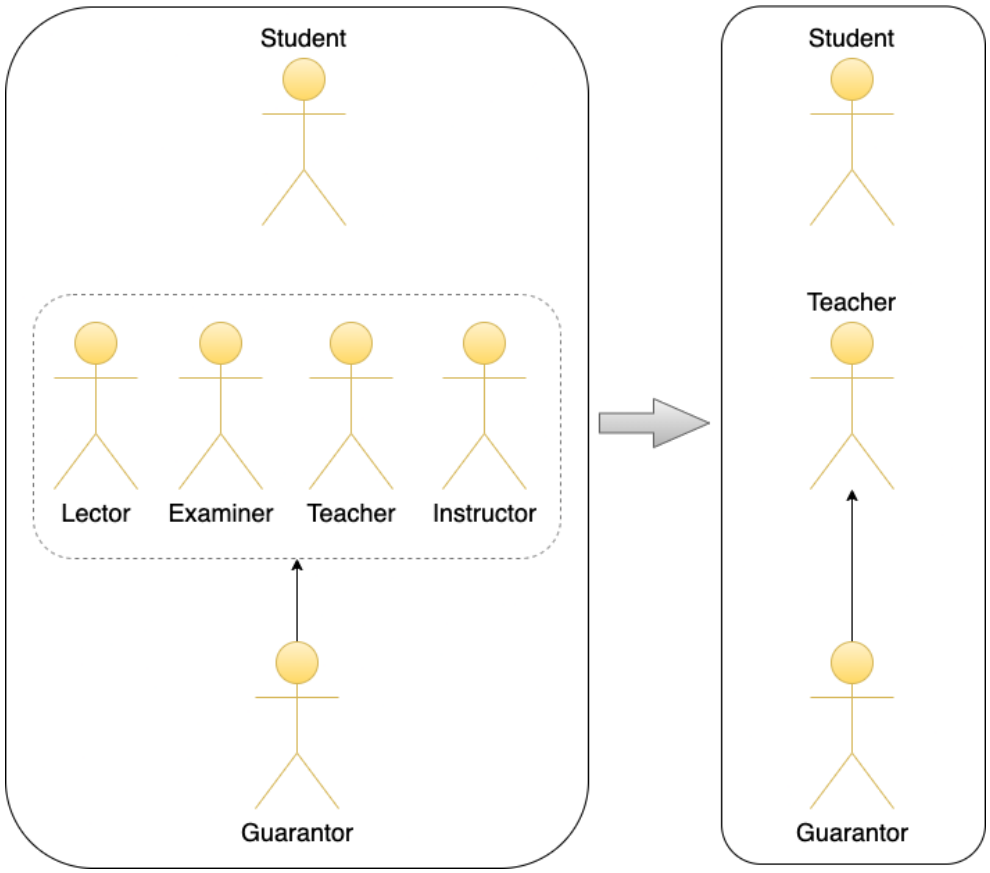

Figure 3.1: KOS roles

3.1.2 Other roles

However, the user roles defined by KOS are not fulfilling all the requirements for the application. There are some special cases that require additional roles

such as:

- *Impersonation as a student.* For the purpose of demonstrating the process of creating a semestral work and its management, teachers need to have a functionality that will allow them to authorize as a student to show the whole process from the students' side. This feature requires creating a test student, which is identical to a usual student but needs to differ from the usual KOS student role to exclude such students from counting the statistics.
- *Development and testing.* For the development and testing processes there is a need to have a test admin user, which will have an access to all functionalities.

For these requirements there were created two corresponding special roles:

- *Test student.* Identical to KOS student role from the perspective of permissions.
- *Root.* Role with access to all functionalities.

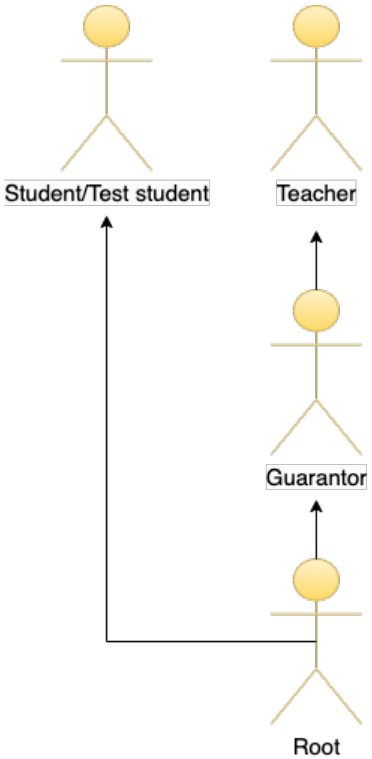

Figure 3.2: Other roles

3.2 Permissions

The BI-DBS portal functionalities are divided into several modules.

- *Administration.* The administration module provides functionalities for semester configurations.
- *Semester work.* The semester work module contains all the features for managing the semester work from its creation to evaluation.
- *Tests.* All components for the management of demo, semester and exam tests are placed in the tests module.
- *Connections.* The module which provides the configuration of the database connection is the connections module.
- *Students' score.* Users of the application can see the results of the student's performance during the semester in the student's score module.
- *Users.* Users module provides an overview of the users of the portal.
- *Data modeler.* Data modeler is a playground for drawing conceptual schemas.
- *Transformation modeler.* Transformation modeler is an extension of the data modeler, which also provides the generation of create a script based on the drawn schema.
- *Home.* The home page is composed of the overview of the semester.
- *Authorization.* The authorization module provides login and logout features.

These modules can be defined into three groups:

1. Modules with the same permissions for all roles: transformation modeler, data modeler, connections, authorization.
2. Modules available only for a certain role: administration, users.
3. Modules available for all the roles but with different permissions for their components: semester work, tests, students' score, home.

The first group does not need to be provided with an access management structure as all the modules from the group are available for any authorized user with any of the roles described in the previous section. Therefore the access validation to these modules and their components is simple and clear. The modules from the second group are available only for two roles: guarantor

and root.
Finally, the last group of modules has a more complex access management structure. These modules mostly have two types of components. The first type is components accessible for student, test student and root roles and the second type is accessible for teacher, guarantor and root roles. Therefore in the short description of the module's permissions by roles, I will use only two roles, student for the first type and teacher for the second type.

Semester work

- *Permissions for student.*
  - Semester work editor
  - Check and submission
  - Classification requirements
- *Permissions for teacher.*
  - Submitted semester works and submission status view
  - Semester works evaluation
  - Import and set deadlines
  - Classification requirements

Tests

- *Permissions for student.*
  - Taking demo tests
  - Taking assigned tests
- *Permissions for teacher.*
  - Create and edit assignments
  - Create and edit questions
  - Create test templates
  - Assign and start tests
  - Evaluate tests
  - Tests classification and statistics

Students' score

- *Permissions for student.*
  - View students' score with anonymized personal data
- *Permissions for teacher.*
  - View students' score

Home

- *Permissions for student.*
  - View of personal and course data
  - View of personal progress in a course
- *Permissions for teacher.*
  - View of the course statistics of students progress and activity

The detailed accesses to the components and functionalities of the BI-DBS portal are going to be presented by the use case diagrams by students, who will be implementing them. An example of such work is Radoslav Hašeks's master thesis[33] which focuses on analyses, designing and implementation of the tests module.

Chapter 4

Implementation

Chapter 5

Automation

Chapter 6

Testing

Chapter 7

Testing

Conclusion

First of the thesis goals was contributing to the project by improving the development efficiency and maintenance, which was achieved by introducing the DevOps methodology and also automating such processes as testing and development. The next goal was defined as adjusting the project to make it more structured and secure. Therefore, I have analyzed the current access management structure and came to the conclusion that it is unnecessarily too complicated. I have managed to logically simplify it and provide a design of a new access management system. Moreover, based on the clear permissions system I have strengthened the security by providing validation before every user's request. Finally, I planned to implement the authentication and authorization features. These functionalities were created using the OAuth 2.0 protocol and authorization microservice. Furthermore, I have reduced security vulnerabilities by providing a suitable way of storing sensitive data.
In my opinion, this thesis will definitely be useful for developers and managers of the BI-DBS portal. It can be used as a part of introduction materials to DevOps methodology for new students before getting to the application development as well as the documentation of the access management system for managers and developers.
From the analysis of the current and planned application state, I have outlined three challenges, which we are facing in the new application state due to the microservices architecture. I have provided the solutions for two of them. The deployment problem was resolved by automation of this process. Possible security vulnerabilities were decreased by creating an access management system. The testing challenge still remains unresolved. The application needs a strong set of integration tests. Ideally, they should be a part of CI/CD pipeline to reduce the probability of deploying errors. This is an idea for further improvement which can become a topic for a thesis or a set of tasks for the teams of BI-DBS developers.
Thanks to this thesis I have learned a lot about application security and efficient development. Moreover, from the development of the features I got

experience working with modern technologies such as Vue.js, Typescipt, Pinia and others.

Sources

1. PLYSKACH, Ing. Andrii. *Modernization and Migration of DBS portal*. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.
2. *OAuth 2.0 - OAuth* [online]. 2023. [visited on 2023-04-23]. Available from: `hhttps://oauth.net/2/`.
3. MALEC, Ing. Oldřich. *Project and infrastracture management of BI-DBS teaching portal*. Bachelor thesis. Czech Technical University in Prague, Faculty of Information Technology, 2017.
4. MIKE, Potel. *MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java* [online]. 1996. [visited on 2023-04-04]. Available from: `https://www.wildcrest.com/Potel/Portfolio/mvp.pdf`.
5. HARRIS, Chandler. *Microservices vs. monolithic architecture* [online]. 1996. [visited on 2023-04-08]. Available from: `https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith`.
6. *What is PHP?* [online]. 2023. [visited on 2023-04-05]. Available from: `https://www.php.net/manual/en/intro-whatis.php`.
7. *HTML: HyperText Markup Language* [online]. 2023. [visited on 2023-04-06]. Available from: `https://developer.mozilla.org/en-US/docs/Web/HTML`.
8. *Doctrine* [online]. 2022. [visited on 2023-04-05]. Available from: `https://github.com/doctrine/orm`.
9. *Nette* [online]. 2023. [visited on 2023-04-05]. Available from: `https://nette.org/en/`.
10. *Latte* [online]. 2023. [visited on 2023-04-07]. Available from: `https://latte.nette.org/en/`.

11. *AdminLTE* [online]. 2022. [visited on 2023-04-08]. Available from: https://github.com/ColorlibHQ/AdminLTE.

12. *The Progressive JavaScript Framework. Vue 2* [online]. 2023. [visited on 2023-04-07]. Available from: https://v2.vuejs.org/.

13. ZAKELŠEK, Haidi. *Options API vs. Composition API* [online]. 2022. [visited on 2023-04-08]. Available from: https://medium.com/codex/options-api-vs-composition-api-4a745fb8610.

14. *JavaScript* [online]. 2023. [visited on 2023-04-08]. Available from: https://en.wikipedia.org/wiki/JavaScript.

15. *Webpack concepts* [online]. 2023. [visited on 2023-04-06]. Available from: https://webpack.js.org/concepts/.

16. *PHP OOP - Object-Oriented Programming in PHP* [online]. 2022. [visited on 2023-04-08]. Available from: https://www.phptutorial.net/php-oop/.

17. *What is Symfony* [online]. 2023. [visited on 2023-04-06]. Available from: https://symfony.com/what-is-symfony.

18. *What's new in Vue 3 — a roundup* [online]. 2023. [visited on 2023-04-08]. Available from: https://medium.com/front-end-weekly/whats-new-with-vue3-5b6562d3898b.

19. *Composition API FAQ — Vue.js* [online]. 2023. [visited on 2023-04-08]. Available from: https://vuejs.org/guide/extras/composition-api-faq.html.

20. *Why Vite* [online]. 2023. [visited on 2023-04-08]. Available from: https://vitejs.dev/guide/why.html.

21. *TypeScript is JavaScript with syntax for types.* [online]. 2023. [visited on 2023-04-08]. Available from: https://www.typescriptlang.org.

22. *Why Quasar?* [online]. 2015. [visited on 2023-04-08]. Available from: https://quasar.dev/introduction-to-quasar.

23. *Pinia* [online]. 2023. [visited on 2023-04-09]. Available from: https://en.wikipedia.org/wiki/Pinia.

24. *Microservices and DevOps: Better together* [online]. 2023. [visited on 2023-04-15]. Available from: https://www.mulesoft.com/resources/api/microservices-devops-better-together.

25. *DEvOps* [online]. 2023. [visited on 2023-04-11]. Available from: https://en.wikipedia.org/wiki/DevOps.

26. *What Is DevOps?* [online]. 2023. [visited on 2023-04-11]. Available from: https://www.atlassian.com/devops.

27. *6 Principles of DevOps – DevOps Agile Skills Association (DASA)* [online]. 2023. [visited on 2023-04-12]. Available from: https://www.devopsagileskills.org/dasa-devops-principles.

28. *7 Principles of DevOps for Successful Development Teams* [online]. 2021. [visited on 2023-04-15]. Available from: https://blog.hubspot.com/website/devops-principles.

29. *DevOps: Principles, Practices, and DevOps Engineer Role* [online]. 2021. [visited on 2023-04-15]. Available from: https://www.altexsoft.com/blog/engineering/devops-principles-practices-and-devops-engineer-role/.

30. *DevOps: Principles, Practices, and DevOps Engineer Role* [online]. 2023. [visited on 2023-04-16]. Available from: https://appinventiv.com/blog/devops-adoption-and-implementation/.

31. *Course - KOSapi* [online]. 2015. [visited on 2023-04-25]. Available from: https://kosapi.fit.cvut.cz/projects/kosapi/wiki/Course.

32. *Usermap API* [online]. 2015. [visited on 2023-04-25]. Available from: https://rozvoj.fit.cvut.cz/Main/usermap-api.

33. HAŠEK, Bc. Radoslav. *dbs.fit.cvut.cz – tests*. Master thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Appendix A

Acronyms

BI-DBS - Database systems

BI-SP1 - Team software project 1

BI-SP2 - Team software project 2

MVP Model View Presenter

PHP Hypertext Preprocessor

HTML HyperText Markup Language

ORM Object Relational Mapping

SQL Structured Query Language

JS JavaScript

SPA Single-page application

DevOps Development and Operations

OAuth Open authorization

IT Informational technology

Appendix B

Contents of enclosed CD