Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Microservise Patterns in Online Markdown Editor

Yevhen Chaban

Supervisor: Ing. Martin Komárek
Field of study: Software Engineering and Technology
May 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Chaban** Jméno: **Yevhen** Osobní číslo: **487025**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Mikroservisní vzory v markdown online editoru

Název bakalářské práce anglicky:

Microservise Patterns in Online Markdown Editor

Pokyny pro vypracování:

Study the microservice patterns [1], Docker and Kubernetes
Deploy existed "Git Based Markdown Editor Online" [2] application locally and to value stream delivery platform CodeNow [3].
Implement the most reasonable improvement, based on suggestions from application author thesis [2], and personal analysis.
Identify adequate microservice pattern for system improvement.
Design, implement a document selected microservice patterns.
Use iterative approach. Continuously build, test, deploy and document changes.

Seznam doporučené literatury:

[1] RICHARDSON, Chris. Microservices Patterns. Manning Publications, 2018. isbn 9781617294549.
[2] SAJDL, Vojtěch. GIT based markdown online editor. Prague 6, 2021. Available also from: https://dspace.cvut.cz/handle/10467/95359.
[3] CodeNOW. CodeNOW Documentation, 2022, https://docs.codenow.com/.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Martin Komárek kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **09.02.2023** Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

Ing. Martin Komárek podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.

Datum převzetí zadání Podpis studenta

Acknowledgements

I would like to thank my friends and my family, who support me all the way. I also would thank CTU for the given opportunities and the high quality of the study.

Declaration

In Prague, May 26, 2023

Abstract

The goal of this project is to get an overview and analyze the existing Git based markdown online editor application from CTU student Vojtech Sajdl, which supports the entire life cycle of software documentation. This application is described in his Master's thesis [1]. The best improvement of the application, focused on microservice patterns, will be suggested and implemented.

Keywords: Documentation management, Docusaurus, Markdown, Git, microservice architecture, microservices patterns

Supervisor: Ing. Martin Komárek

Abstrakt

Cílem projektu je udělat přehled a analýzu existující Git based markdown online editor aplikace, kterou vytvořil ČVUT student Vojtěch Sajdl. Aplikace je popsána v jeho diplomové práci [1]. Následovně se nabídne a naimplementuje nejlepší vylepšení, orientované na microservisní patterny.

Klíčová slova: Documentation management, Docusaurus, Markdown, Git, microservice architecture, microservices patterns

Překlad názvu: Mikroservisní vzory v markdown online editoru

Contents

Figures

Tables

Chapter 1

Introduction

In this chapter project motivation will be explained. Then, the chapter continues with the project structure.

1.1 Motivation

Documentation is an important part of almost every programmer's job. Developers consider documentation important, but they do not put the effort into writing and maintaining it. According to the feedback [2], there is not enough documentation on many projects.

One reason for this may be that developers do not have enough time to create documentation themselves. A partial solution to this problem may be to delegate the creation and maintenance of documentation.

However, delegating the creation of documentation to people who are not programmers can be problematic, because they do not have sufficient skills to work on the documentation project. In this case, software that allows users to work with documentation without technical skills could be very useful.

The Git based markdown online editor [3] is such an application. Based on this fact, improving an existing application makes sense.

1.2 Application functionality review

In Git based markdown online editor, we can divide the functionality into two groups: documentation life-cycle related and proofreading related. The use cases of each group will be shown in the next subsections.

Each use case in this subsection is related to the documentation life cycle. All use cases together implement CRUD operations for documentation.

- Create documentation.

  There are two options:

  - Create new.

  - Import existing.

- Delete documentation.
- Changes documentation (project) settings. It allows to change name, slug, or description.
- Adding/deleting pages and folders.
- Adding new members to the project. It allows to set up access rights for each member.

The second very important group of application functionality is proofreading. All use cases together also implement CRUD operations for proofreading requests. Also, every use-case includes the git administration part:

- Create proofreading request. A new branch for proofreading will be created.
- Make proofreading changes. Changes from the proofreader will be committed and pushed into the created branch.
- Reject the proofreading request. The request will be returned to the proofreader.
- Merge proofreading request. Changes will be merged into the source branch.

Chapter 2
Related theory

This section describes the technology stack used in this thesis project and introduces related concepts.

2.1 GitLab

GitLab is an open-source end-to-end software development platform that includes version control, issue tracking, code review, CI/CD [4], and more.

Here are the most important functionalities of GitLab:

- Version control

  Version control is an essential tool for application development. It allows us to keep track of code changes because it keeps a history of all additions, changes, and deletions to the file.

- Code review

  Gitlab allows teams to do code reviews with its built-in pool-request functionality. It allows control of the code changes, before merging two branches. Code review is an integral part of development and one of the fundamental factors in the growth of the developer level. It helps to find bugs and optimize code, to improve its quality, to check for security vulnerabilities, and more.

- Collaboration GitLab has built-in support for the branches. It allows a whole team of developers to work on one application in different branches and merge them when it is needed, for example, before the release.

- GitLab pages

  This feature allows users to deploy their static page directly from the repository by creating a simple configuration file. After repository is configured, the project will build and deploy after every commit to a specific branch or branches.

- GitLab authentication provider

  Gitlab can act as an authentication provider. It supports two different methods of authenticating.

One of them is authorization with Basic authentication. In this case, GitLab checks the user by his password and username, this method is often used to make calls to the GitLab API, because it is easier to implement.

The second one is token-based authorization with OAuth 2.0 protocol. In this case, GitLab returns a token to the user, which is then used to communicate between programs. There are several ways to implement this type of authorization. Section 2.1.1 will describe the type used in the application.

Based on these facts, we can conclude that GitLab is a powerful tool to control the full cycle of software development.

2.1.1 GitLab applications

Gitlab applications allow us to access two functions at once: use GitLab as an Authentication provider and perform actions on behalf of the user. To configure this functionality the user needs to register a new GitLab application and get the OAuth 2 Client ID and OAuth 2 Client Secret from it. The application will establish a connection with GitLab using the Client ID and Client secret and will receive a token from GitLab. This token is then used by third-party services to access the functionality of GitLab.

2.2 Docker

Docker is an open-source platform [5]. The platform allows developers to manage, deploy, create, and run containers. Docker is a comprehensive platform and includes many components such as DockerFiles, Docker images, Docker containers, Docker Hub, Docker Desktop, Docker daemon, Docker registry, and others. Below are listed the main components and their functionality.

2.2.1 DockerFile

DockerFile is a very valuable tool. It serves as a file with instructions on how to build a component. The developer creates this file in the component folder, then Docker reads it.

2.2.2 Docker images

A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries, and settings.

2.2.3 Docker container

A container is a standardized, executable component that combines application source code with the operating system (OS) libraries and dependencies required to run that code in any environment [6]. Docker images become containers at runtime when they are running on the Docker engine.

2.2.4 Docker Hub

Docker Hub is the world's largest repository of container images with an array of content sources including container community developers, open source projects and independent software vendors (ISV) building and distributing their code in containers [6]. Users get access to free public repositories for storing and sharing images, or they can choose a subscription plan for private repositories. With the Docker Hub, it is not necessary to create new images of the most-used services each time we need them. For example, Postgres has its own image exported. Thus, we can just pull it from the Hub.

2.3 Javascript

JavaScript is an essential component of the World Wide Web [7]. It is a widely used programming language. It is one of the most popular development languages for client-side web development. The JavaScript code usually works with an HTML document. JavaScript is a dynamic programming language that is supported by all modern browsers with a built-in JavaScript engine.

2.3.1 Node.js

Node.js is an asynchronous event-driven backend JavaScript runtime environment [8]. It is designed to build scalable network applications. Almost no function in Node.js directly performs I/O (Input/Output), thus the process never blocks, except for the cases when the I/O is performed using synchronous methods of the Node.js standard library. Because nothing blocks, scalable systems are very reasonable to develop in Node.js.

2.3.2 React

React is a JavaScript library for building user interfaces [9]. It is Declarative, the developer designs simple views for each state in the application, and React will efficiently update and render just the right components when data changes. Also, it is Component-Based, which means that developer builds encapsulated components that manage their own state, then composes them to make complex UIs.

2.4 Docusaurus

Docusaurus is an open-source project by Meta [10]. It can be used for building, deploying, and maintaining websites. Docusaurus enables a quick and easy start, thanks to its user-friendly and intuitive interface. Developers can use Markdown [11] or MDX [12] to create and format posts and documents.

One of the standout features of the Docusaurus is its ability to use React code in its posts and documents, which brings interactivity and flexibility into the documentation, and enables keeping up with new technologies.

The main goal of the docusaurus is to simplify the creation and development of the website, whether the developer or team of developers are working on a personal project or a complex enterprise website.

2.4.1 Docusaurus configuration file

The Docusaurus configuration file plays a critical role in defining the behavior and appearance of a Docusaurus website. This configuration file is typically placed in the root directory of the website and has a JavaScript file extension. Usually it exports configuration object with module.exports JavaScript functionality. Then, Docusaurus operates with the exported configuration file.

One of the key functions of the configuration file is to define the website's appearance, including its title, tagline, URL, etc. Also, there is a possibility to configure the behavior of the website, which includes the behavior of the website on broken links, adding and removing menu items and links.

2.5 Software architecture

Software architecture is a structure, that describes elements, their externally visible properties, and relations between them [13]. Using software architectures have a lot of advantages, but the most important are:

- Analyzing application before it is implemented.

  It is very important to look at the application and analyze it, before its implementation. It is much cheaper to find possible problems during architecture planning than to fix them in an already completed system. Also, in big systems, starting the development without a defined structure will cause communication problems between teams, as well as synchronization problems between different parts of the application. With this approach, the project is likely to be unsuccessful.

- Providing a basis for creating re-used solutions.

> The software architecture allows developers to save and reuse successful solutions that have justified themselves over time or by some other factors.

In software engineering, there is a possibility to expose the structure of a system while hiding most of the implementation details, by using software architecture.

2.6 Microservice architecture

Microservice architecture is the architecture that structures the application as a set of loosely coupled, collaborating services [14]. Each service is:

- Highly maintainable and testable

  There are several ways to define a set of microservices, but each of them focuses on creating a service that is responsible for only one area. This leads to improved maintainability and testability since changes affect only the area for which the service is responsible.

- Loosely coupled with other services

  Services communication goes only through strictly defined API, which means they do not depend on other services implementations.

- Independently deployable

  When microservice infrastructure is set up, each service can automatically be deployed, thus there is no problem in independently deploying since each microservice is automatically built and tested.

- Capable of being developed by a small team

  Because of the loose coupling and area of responsibility, it is able to develop part of the system independently. Different teams can work on the same project without slowing down each other by waiting for another team's changes. Also, the amount of necessary communication between the teams is reduced.

2.7 Program code parsing and modifying

Usually, for parsing program code, Abstract Syntax Trees (AST) are used. AST is a structure that contains nodes, each node represents a part of the code, which means that every variable, code block, and even comment is a specified node type.

An abstract syntax tree (AST) captures the essential structure of the input in a tree form, while omitting unnecessary syntactic details [15].

After the code is parsed into an AST, it becomes much easier to modify and analyze it with other computer programs. Developers do not need to manipulate strings of code lines and can simply modify each node separately while keeping the structure of the code intact. This approach can save time and reduce errors that may occur when modifying code manually.

In addition to simplifying code modification and analysis, parsing JavaScript code to an AST can also provide benefits in terms of code optimization and refactoring. AST can be used as an instrument for automatically eliminating redundant code or rewriting code to logically equivalent but faster operations.

Last but not least, the opportunity that AST brings is the possibility to implement code linters and static analysis tools based on AST.

Chapter 3

Deploy and Actual state of application

This section is focused on prerequisite knowledge for application understanding and improvement. It describes some of the application startup issues with the chosen solutions.

3.1 Prerequisites for understanding the application

For analyzing and continuing developing the application it was necessary to gain an understanding of such topics as Docker/Kubernetes, Microservice architecture, Git API, Docusaurus, Markdown, and the basics of React and NodeJs. In order to understand these topics, the course that fully covers Docker/Kubernetes problematic [16] has taken.

Microservices architecture basics were gained from the book "Microservices Patterns" [14], which was recommended by the supervisor. Thanks to this book I got an idea of what a microservice architecture is, what advantages it can bring to an application, its disadvantages as well as the difficulties of moving from a monolithic architecture to a microservice one.

One part of the assignment was to get through the documentation of the CodeNOW platform [17], and then use this platform to deploy the implemented solution.

CodeNOW is a comprehensive platform designed to simplify and optimize the DevOps cycle. It allows the developer to focus on the development of business logic, paying less attention to configuring and maintaining the infrastructure.

One of the key advantages of CodeNOW is its ability to accelerate time-to-market, giving companies a competitive advantage. By integrating various tools such as Kubernetes, Tekton, and Prometheus, CodeNOW supports continuous integration and continuous deployment (CI/CD) workflows.

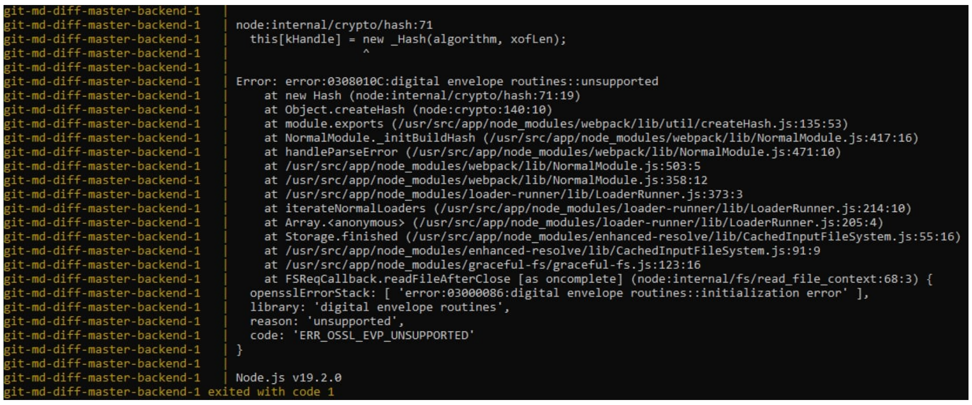
Other missing pieces of knowledge were gained from official documentation websites. I refer to them in the text and they are cited in the sources.

3.2 Startup application issues

The Git based markdown online editor [3] was not updated since April 2022. Therefore, it was expected to meet some problems during the first deployment. The first application launch attempt with Docker, following the readme file from the project author, was unsuccessful. All three components crashed on the same error (Figure. 3.1). This error is described on Node.js changelogs pages [18]. Two solutions to this problem were discovered. The first is to switch the Node.js version to the older one. The version must be lower than 17. The second solution is to run the project with a command-line option:

```
--openssl-legacy-provider
```

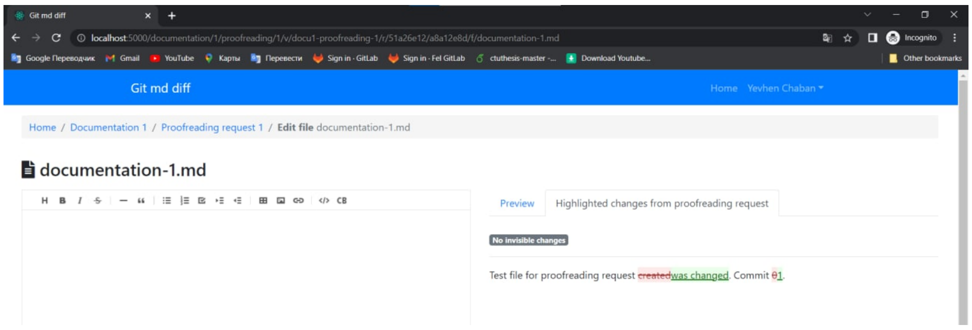
The second solution is recommended by Node.js creators, therefore it was chosen as the error fix.



```
git-md-diff-master-backend-1  |
git-md-diff-master-backend-1  | node:internal/crypto/hash:71
git-md-diff-master-backend-1  |   this[kHandle] = new _Hash(algorithm, xofLen);
git-md-diff-master-backend-1  |                   ^
git-md-diff-master-backend-1  |
git-md-diff-master-backend-1  | Error: error:0308010C:digital envelope routines::unsupported
git-md-diff-master-backend-1  |     at new Hash (node:internal/crypto/hash:71:19)
git-md-diff-master-backend-1  |     at Object.createHash (node:crypto:140:10)
git-md-diff-master-backend-1  |     at module.exports (/usr/src/app/node_modules/webpack/lib/util/createHash.js:135:53)
git-md-diff-master-backend-1  |     at NormalModule._initBuildHash (/usr/src/app/node_modules/webpack/lib/NormalModule.js:417:16)
git-md-diff-master-backend-1  |     at handleParseError (/usr/src/app/node_modules/webpack/lib/NormalModule.js:471:10)
git-md-diff-master-backend-1  |     at /usr/src/app/node_modules/webpack/lib/NormalModule.js:503:5
git-md-diff-master-backend-1  |     at /usr/src/app/node_modules/webpack/lib/NormalModule.js:358:12
git-md-diff-master-backend-1  |     at /usr/src/app/node_modules/loader-runner/lib/LoaderRunner.js:373:3
git-md-diff-master-backend-1  |     at iterateNormalLoaders (/usr/src/app/node_modules/loader-runner/lib/LoaderRunner.js:214:10)
git-md-diff-master-backend-1  |     at Array.<anonymous> (/usr/src/app/node_modules/loader-runner/lib/LoaderRunner.js:205:4)
git-md-diff-master-backend-1  |     at Storage.finished (/usr/src/app/node_modules/enhanced-resolve/lib/CachedInputFileSystem.js:55:16)
git-md-diff-master-backend-1  |     at /usr/src/app/node_modules/enhanced-resolve/lib/CachedInputFileSystem.js:91:9
git-md-diff-master-backend-1  |     at /usr/src/app/node_modules/graceful-fs/graceful-fs.js:123:16
git-md-diff-master-backend-1  |     at FSReqCallback.readFileAfterClose [as oncomplete] (node:internal/fs/read_file_context:68:3) {
git-md-diff-master-backend-1  |   opensslErrorStack: [ 'error:03000086:digital envelope routines::initialization error' ],
git-md-diff-master-backend-1  |   library: 'digital envelope routines',
git-md-diff-master-backend-1  |   reason: 'unsupported',
git-md-diff-master-backend-1  |   code: 'ERR_OSSL_EVP_UNSUPPORTED'
git-md-diff-master-backend-1  | }
git-md-diff-master-backend-1  |
git-md-diff-master-backend-1  | Node.js v19.2.0
git-md-diff-master-backend-1 exited with code 1
```

Figure 3.1: Error while deploying backend, frontend, and render components.

After deploying the application, its functionality was tested. Everything worked correctly, except proofreading. After the "proofreading request editing page" was opened, the part that is responsible for making proofreading changes was not loaded properly. Therefore, it was not possible to edit the document (Figure. 3.2).



Git md diff

localhost:5000/documentation/1/proofreading/1/v/docu1-proofreading-1/r/51a26e12/a8a12e8d/f/documentation-1.md

Google Переводчик · Gmail · YouTube · Карты · Перевести · Sign in · GitLab · Sign in · Fel GitLab · cthesis-master -... · Download Youtube... · Other bookmarks

Git md diff · Home · Yevhen Chaban

Home / Documentation 1 / Proofreading request 1 / **Edit file** documentation-1.md

documentation-1.md

Preview · Highlighted changes from proofreading request

No invisible changes

Test file for proofreading request ~~created~~was changed. Commit ~~0~~1.

Figure 3.2: Editing documentation part of the page was not loaded correctly.

To investigate this issue application was launched locally, component by component. While researching and debugging the code, I found that the wrong branch name has been saved to the database every time when the branch for proofreader changes has been created. Commit, which causes problems, was found in the author project repository.

After manually editing the branch name in the database, problems were solved. Therefore, the code was fixed and the application was ready for analysis.

Chapter 4
Analysis

In this chapter I review requirements for the development of the application with new functionality, which appeared during the work on the project and consultations with the supervisor. Also, in this section I give an overview and analysis of the application improvements, and the most reasonable improvement will be chosen and explained.

4.1 Requirements to the new functionality

Since a specific application improvement was to be chosen during this work, there were a few requirements:

- Increase the business value of the application.
- Increase application load-balancing ability.
- Make the creation and administration of documentation easier for non-technical people.

4.2 Application components and architecture

The architecture of the application (Figure. 4.1) consists of 3 components: the API (backend), the frontend, and the renderer. The backend is connected to the PostgreSQL [19] database. Thus, it is practically typical 3-layered architecture. Nevertheless, the code was written in such a way that it was easily separable into separate components for the microservice's patterns implementation in the future.

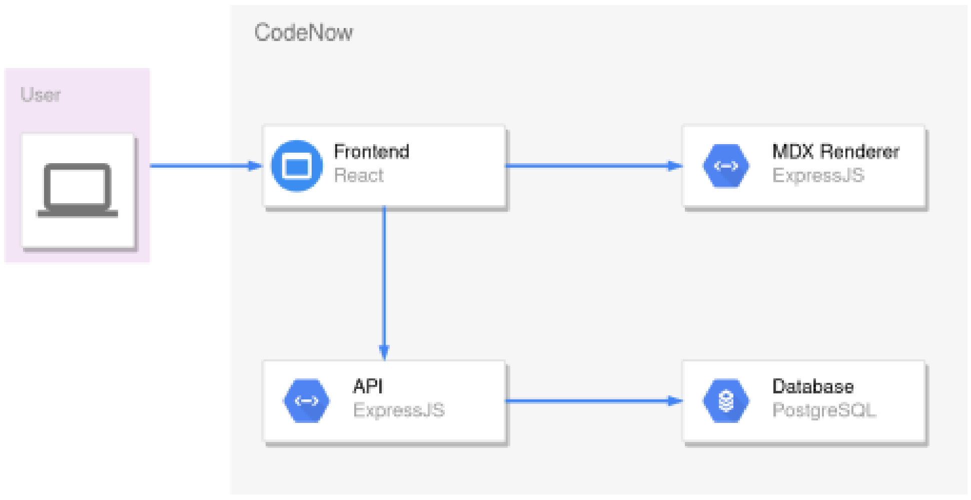


Figure 4.1: Application architecture [1].

4.3 Overview and analyses application improvements

The author of the original application suggests several possible improvements in his master thesis [1]:

- Diff algorithm.

  Diff algorithm is used to show the difference between two files, from different commits. The algorithm used in the application has two main problems: readability and validity. In some cases, the algorithm shows the difference between two files wrong. Also, the MDX toolkit, which is used in the application, is very sensitive to syntax errors, thus some of the syntax errors appear while rendering.

- Various UI and UX improvements.

  After user testing it was discovered that some small UI and UX changes are required.

- Better test coverage.

  Only part of the application is covered with basic tests.

- Database migration.

  The application does not have any database migration tool connected. It may cause problems in the future.

All mentioned improvements are relatively minor. Any of these improvements will not bring significant business value to the project and they also will not affect the usability of the application. So other ideas were needed.

Several improvements came up in the process of communication between the supervisor and the author of this thesis:

- Buildable and deployable docusaurus documentation.

  Implement a feature that allows building and deploying documentation from the website. This feature will be useful for non-technical people, because it reduces the amount of coding needed to create or edit documentation parts. The current version of the application is integrated with GitLab, which has the ability to set up automatic deployment using the .gitlab-ci.yml file [20].

- Implement chosen microservices pattern.

  Another way to improve the application is the implementation of the chosen microservices pattern. Depending on the chosen pattern it will bring certain advantages for the application. The chapter 4.4 will explain it in more detail.

- Implementation of the editing page for the docusaurus configuration file (docusaurus.config.js).

  Docusaurus can be partly configured by editing the docusaurus.config.js file, but it can be tricky for non-technical people. However, the config file has a JSON structure and can be relatively easily parsed to AST, modified on the frontend, and saved back.

4.4 Chosen improvements

Implementation of the CQRS pattern and configuration editing page was chosen as the most reasonable improvement of the application.

4.4.1 CQRS pattern

CQRS stands for Command and Query Responsibility Segregation. It is a pattern that separates, reads, and updates operations for data stores. Implementing CQRS in the application can maximize its performance, scalability, load-balancing, and security [21].

The key advantage of this pattern is the load-balancing possibility, which is important for this project. If data stores and components will be split into two separate parts with queries and commands, then we can increase the number of instances only for one part. For example, two instances of commands services and one for the queries, etc.

Of course, this pattern has several disadvantages as bringing complexity to the application, also it requires more database technologies to be involved and generally increasing points of failure in the application.

This thesis will focus on implementing part of this pattern, which is to divide the API backend into two parts. That means that the final version will include database stores separating and synchronization mechanisms that will synchronize data from different stores. I will describe possible ways to implement it in section 5.1.5. This decision was made after consultation with the thesis supervisor, who confirmed that I can implement only a part of the pattern in my thesis in the described above way.

The reasons for implementing only a part of the pattern are the relatively small amount of time and the large amount of work that it requires, which is outside the scope of the thesis project. Also, even partly implemented pattern will already affect the application performance and load-balancing possibility.

Basically, the application will be split by REST endpoints to the two different components, each component will be responsible only for the commands or queries respectively.

The implementation details of the pattern will be described in the Implementation chapter 5 with schemes of the new architecture.

4.4.2 Docusaurus configuration editing page

As it was mentioned in previous chapters, the docusaurus configuration file is an important part of the customization of the docusaurus website.

However, users, who do not have enough skills or do not want to learn how to operate with git, javascript, and configuration files, can not use the advantages of the docusaurus configuration file. Thus, developing something that will create the abstract level between the config file and users could be very helpful.

A part of this thesis will describe the second chosen improvement, which is extending an existing application functionality with the new Tab for the editing configuration file via convenient frontend components. Information from the frontend will be transferred to the backend component and then processed. Changes will be saved back to the configuration file and pushed to the specified GitLab branch.

Chapter 5
Implementation

The chapter is divided into two main parts: the CQRS template implementation and the docusaurus editing page implementation. Each part contains backend and frontend implementation details. React was a completely new framework to me, as well as Node.js. These frameworks and languages were chosen by the author of the original application, thus, changing the stack was unrealistic, and I gained missing pieces of knowledge. Therefore, everything was implemented in one stack: React and Node.js.

The deployment on the CodeNOW platform is described at the end of the chapter. It proved to be a challenging task due to a lack of experience with Docker containers, Docker files, and deployment platforms such as CodeNOW. In addition, there was an unforeseen error when connecting to the database, which is also described in the chapter.

5.1 CQRS

As was mentioned earlier, only a part of the pattern will be implemented.

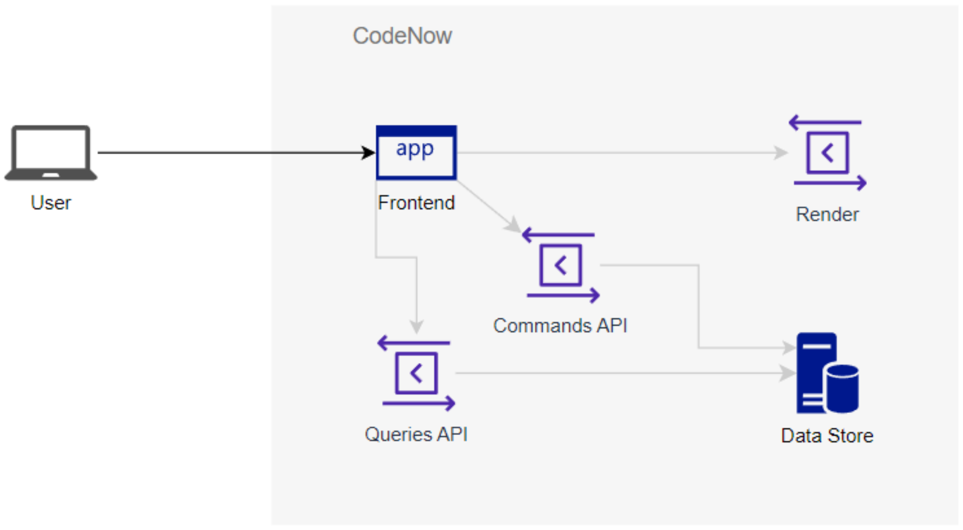


Figure 5.1: New architecture state.

5.1.1 Design changes

Pattern implementation will affect application architecture. Before modifying, the application had almost typical 3-layers architecture: backend, frontend, database and, in addition, it had render backend component (Figure. 4.1). Once the pattern is implemented, due to having two backend components instead of one, the architecture will change. The figure shows the new architecture(Figure. 5.1).

5.1.2 Backend configuration

All redundant code of each component was deleted, and each component has its own configuration file, code, and zone of responsibility. This leads to the separated, loosely-coupled backend components, that can be configured to use different data stores.

The components use the same structure of the configuration file (Code snippet 5.2).

```
{
    "app": {
        "port": 3000,
        "jwtSecret": "tokenSecret",
        "refreshSecret": "refreshSecret"
    },
    "gitlab": {
        "appid": "---",
        "secret": "---",
        "callback": "http://localhost:3000/auth/gitlab/callback",
        "authRedirect": "http://localhost:5000",
        "baseUrl": "https://gitlab.com"
    },
    "mail": {
        "host": "smtp.ethereal.email",
        "port": 587,
        "user": {
            "email": "---",
            "pass": "---",
            "name": "Git md diff mailer"
        }
    }
}
```

Figure 5.2: Backend component configuration file

The file contains three objects, which are responsible for setting up the application, the email client, and GitLab. In the GitLab section, there are five fields:

- "appid" and "secret"

This is data from the GitLab application, a topic covered in this paragraph 2.1.1

- callback

  This field contains the URL of the callback endpoint, which will redirect user to the frontend after authorization through GitLab.

- authRedirect

  The URL of the frontend component, for redirecting.

- baseUrl

  Base URL of the used GitLab domain.

The application has two ways to set database connection properties (Code snippet 5.3). The first is to add another object to the configuration file, which will store the connection data. The second way is to connect with data taken from environment variables. The second way is more suitable for deploying an application using the CodeNOW platform, as the platform sets the connection data to environment variables. As a result, developers do not have direct visibility of these variables, which precludes the possibility of writing them to the configuration file.

```
// staticConfig is a component configuration file variable

const config = {
    db: {
        host: process.env.DATABASE_HOST || staticConfig.db.host,
        port: process.env.DATABASE_PORT || staticConfig.db.port,
        database: process.env.DATABASE_DATABASE_NAME ||
            staticConfig.db.database,
        username: process.env.DATABASE_USERNAME ||
            staticConfig.db.username,
        password: process.env.DATABASE_PASSWORD ||
            staticConfig.db.password,
        ssl: staticConfig.db.ssl,
    },
}
```

Figure 5.3: Backend configuration variable.

Also, each component has its own DockerFile, which is the same for both components. Instructions in the DockerFile are common, they describe how to build a component and on which port it should be exposed 5.4).

```
FROM node:18

# Create app directory
WORKDIR /usr/src/app

# Copy contents of repository
COPY . .

# Install app dependencies
RUN npm ci --only=production

EXPOSE 80
ENV PORT=80

CMD [ "node", "-r", "dotenv/config", ".",
    "dotenv_config_path=./CodeNOW/config/.env",
    "dotenv_config_debug=true"]
```

Figure 5.4: DockerFile.

5.1.3 Backend implementation details

One of the requirements for splitting the backend was to keep access to GitLab for each component. In the current state, the application is using GitLab as an OAuth 2.0 authentication identity provider [22]. It means that on our side we need only configure the GitLab application on GitLab account and configure the Node.Js backend to connect through this application.

In the application settings, we can set the allowed URLs for the application. Each URL should be placed on the new line. The URL of the newly created component authentication callback was added to the list of the callback URLs. Then OAuth 2 Client ID and OAuth 2 Client Secret from the GitLab application were used to setup the GitLab authentication strategy using the Passport.js library (Code snippet 5.5).

```
// Gitlab strategy setup
passport.use(new GitLabStrategy({
    clientID: config.gitlab.appid,
    clientSecret: config.gitlab.secret,
    callbackURL: config.gitlab.callback,
    baseURL: config.gitlab.baseUrl,
}, ((accessToken, refreshToken, profile, done) => {
Auth.findOrCreateUser(profile, accessToken, refreshToken,
    'gitlab').then((usr) => done(null, usr.id));
})));
```

Figure 5.5: Gitlab strategy setup code snippet.

After, we can verify the received tokens by calling authenticate function (Code snippet 5.6).

```
// The Gitlab OAuth callback endpoint, redirects back to frontend
app.get('/auth/gitlab/callback', limiter,
(req, res, next) => passport.authenticate('gitlab', (err, user,
    info) => {
    ....
})(req, res, next));
```

Figure 5.6: Passport authenticate function.

5.1.4 Frontend

Because of the V. Sajdl code writing approach, it was relatively easy to extend frontend to call different backend components. Specifically, he uses the parameter from the configuration file in all requests instead of a hard-coded value. This allows to change the backend component address in the configuration file, instead of editing the address in each Request one by one. Nevertheless, all the requests that sent PUT, POST, or DELETE requests have been changed, therefore they use the URL of the appropriate component from the extended configuration file. No other changes on the frontend were needed.

5.1.5 Problems

In the current solution, the data store was not split into two different stores because it requires mechanisms, that will synchronize data stores and it cannot be implemented in the scope of this thesis, because there is no capacity for it.

Nevertheless, I understand how to implement such a mechanism, and further I will briefly describe how I would do it. One of the typical solutions that can be used in the future is based on event sourcing.

In this approach, when the command data store has changed, changes are recorded as an event, and the event is then published to the message broker or event store. Then events are consumed by particular data stores, that needs to be updated. This approach is supporting the asynchronous synchronization of data between different stores.

The final implementation will include these two aspects: splitting the backend, splitting the data stores, and combining it into one complete solution (Figure 5.7). With this architecture, scaling application data stores or backend components will be much easier.

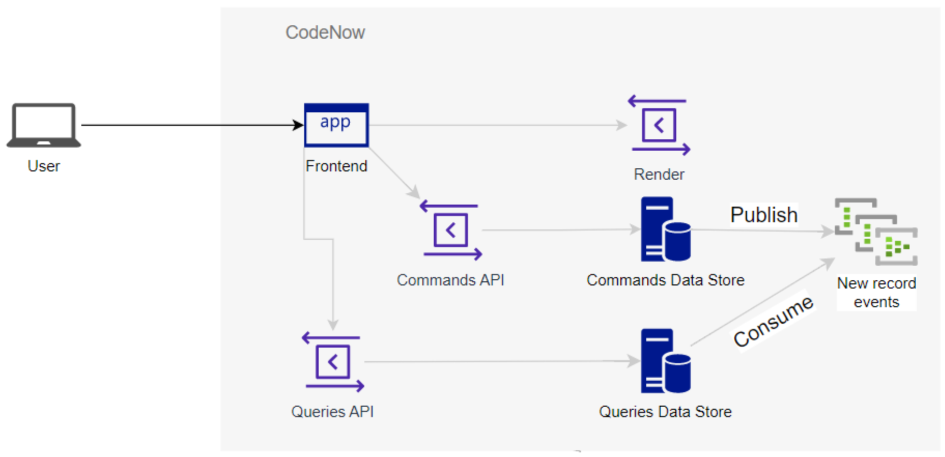


Figure 5.7: Future architecture state.

5.2 Docusarus configuration

The implementation part of the docusaurus configuration editing page was related to working with Abstract Syntax Trees (AST) and code parsing and modifying.

5.2.1 Backend

After research, I found that the best way to operate with the program code is AST parsing approach, so this approach was chosen.

Backend implementation consists of two key endpoints: get the configuration AST node from the configuration file (GET endpoint), get the modified configuration AST node from the frontend, and update the configuration file with it (PUT endpoint). There are several libraries that can help with parsing code to the AST, such as esprima [23] and recast [24]. In the first version of the implementation, esprima in combination with escodegen [25] was chosen, because it is more popular than recast, therefore it is easier to find materials about this library.

The implementation did not take the possibility of saving comments into account in the first stage. When this requirement came up, I ran into a problem. Despite everything, the library was erasing comments. A separate project was created to test this functionality, however, even with simplified versions of the files, keeping comments did not work. Nevertheless, I suppose that saving comments would be quite a useful feature since important notes are often written in comments. Consequently, I made an attempt to change the library and use recast library instead.

With the new library, it was possible to parse code to AST, save comments, modify nodes, and add or remove nodes. The recast library was written in a very similar way, so relatively small changes in the tree-pass algorithm, which updates the configuration file, were needed.

The GET endpoint is pretty simple, it parses the code into AST, then sends it in JSON format to the frontend. While the PUT endpoint is more complex and uses a specific algorithm to modify the AST. After the file is modified, it is saved to the GitLab repository.

With the recast library, we can parse JavaScript code to AST and back (Code snippet 5.8).

```
// Get JavaScript from the repository
const fileFromGitlab = await
    this.getDocusaurusConfigBlob(docuId, branch);

// Parse the JavaScript code into an AST, preserving comments
const ast = recast.parse(fileFromGitlab);

...

// Parse AST to the JavaScript file
const updatedConfigurationFile = recast.print(ast, {
  quote: 'single',
  tabWidth: 2,
  trailingComma: true,
  useTabs: false,
}).code;
```

Figure 5.8: Parsing JavaScript code to AST and back.

In order to change the configuration, we first need to find the node in the tree that is responsible for the configuration object, then find the nodes responsible for the individual values. We can do this by iterating through all the nodes of the tree and searching by key for the objects we need. The main algorithm itself can be divided into two parts: processing key-value items and processing a list with menu items. These parts are described below:

- Key-value fields

  Key-value fields are fields whose value is a string. For example title, tagline, url, baseUrl, etc. In AST they are simple nodes, which can be edited by editing specified fields.

  All changes to the key-value fields are done by modifying the 'value' and 'raw' fields of the AST node, that is responsible for the specific key (Code snippet 5.9).

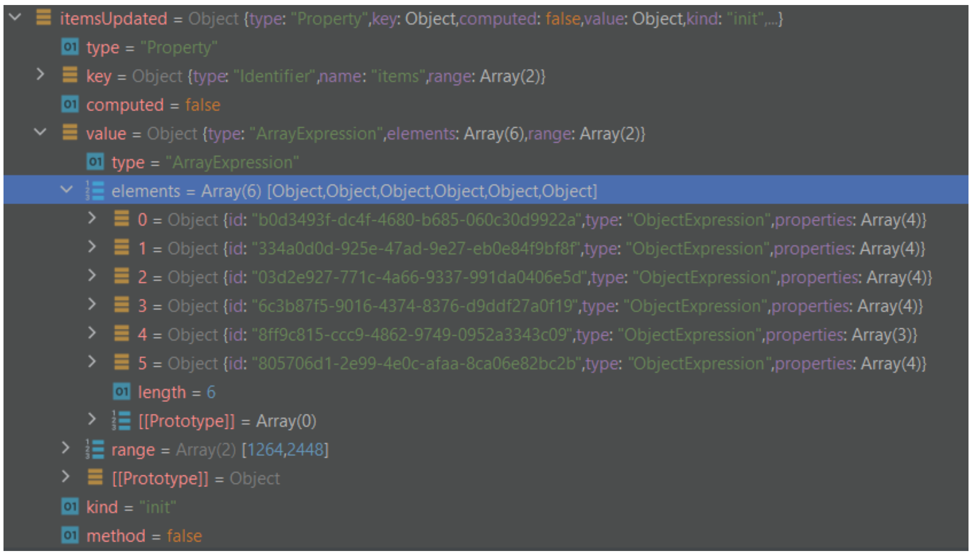
```
function setValueToAst(key, config) {
    // Find the property
    const astObject = config.expression.right.properties.find(
        (property) => property.key.name === key,
    );
    // Update the property value
    astObject.value.value = data[key].value;
    astObject.value.raw = '"' + data[key].raw + '"';
}
```

Figure 5.9: Setting value to the key-value fields.

- Menu items list node

  Menu items are part of the configuration file that contains the list of items from the docusaurus documentation menu.

  The list of menu items in AST is a node, that contains a list with all the menu items. Each item is also a node, which in turn contains specific item fields in properties (Figure. 5.10).



```
itemsUpdated = Object {type: "Property",key: Object,computed: false,value: Object,kind: "init",...}
  type = "Property"
  key = Object {type: "Identifier",name: "items",range: Array(2)}
  computed = false
  value = Object {type: "ArrayExpression",elements: Array(6),range: Array(2)}
    type = "ArrayExpression"
    elements = Array(6) [Object,Object,Object,Object,Object,Object]
      0 = Object {id: "b0d3493f-dc4f-4680-b685-060c30d9922a",type: "ObjectExpression",properties: Array(4)}
      1 = Object {id: "334a0d0d-925e-47ad-9e27-eb0e84f9bf8f",type: "ObjectExpression",properties: Array(4)}
      2 = Object {id: "03d2e927-771c-4a66-9337-991da0406e5d",type: "ObjectExpression",properties: Array(4)}
      3 = Object {id: "6c3b87f5-9016-4374-8376-d9ddf27a0f19",type: "ObjectExpression",properties: Array(4)}
      4 = Object {id: "8ff9c815-ccc9-4862-9749-0952a3343c09",type: "ObjectExpression",properties: Array(3)}
      5 = Object {id: "805706d1-2e99-4e0c-afaa-8ca06e82bc2b",type: "ObjectExpression",properties: Array(4)}
      length = 6
      [[Prototype]] = Array(0)
    range = Array(2) [1264,2448]
    [[Prototype]] = Object
  kind = "init"
  method = false
```

Figure 5.10: The list node of the menu items.

After receiving the configuration object from the frontend, we extract all items and then recreate the list node with the new information by pushing items one by one.

To push a new item to the list we need to create a simple JSON object (Code snippet 5.11).

When recast will transform AST to the JavaScript code these JSON objects will be regenerated to the code snippets.

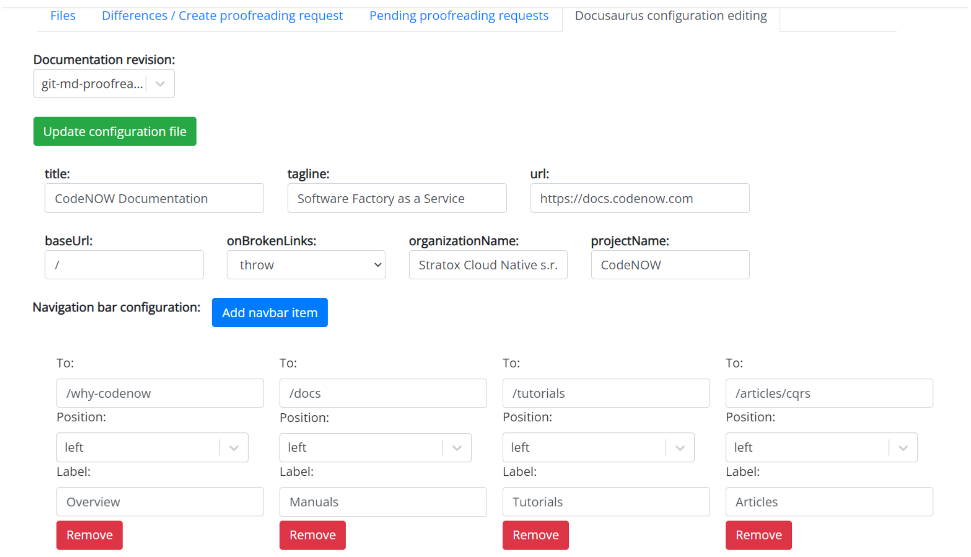
```
// Where the element is one of the items received from the
    frontend.

const elementToPush = {
    type: element.type,
    properties: element.properties,
  }
```

Figure 5.11: Setting value to the key-value fields.

5.2.2 Frontend

On the frontend I created a new Tab that consists of several sections, each responsible for a part of the configuration file (Figure. 5.12).


Files
Differences / Create proofreading request
Pending proofreading requests
Docusaurus configuration editing
Update configuration file
Add navbar item
Remove

Documentation revision: git-md-proofrea...

title: CodeNOW Documentation
tagline: Software Factory as a Service
url: https://docs.codenow.com
baseUrl: /
onBrokenLinks: throw
organizationName: Stratox Cloud Native s.r.
projectName: CodeNOW

Navigation bar configuration:

To: /why-codenow	To: /docs	To: /tutorials	To: /articles/cqrs
Position: left	Position: left	Position: left	Position: left
Label: Overview	Label: Manuals	Label: Tutorials	Label: Articles

Figure 5.12: Frontend tab.

At the top of the page, there is a branch selector. The first section contains simple fields, which consist of key-value pairs. The second section is more complex because it handles menu item administration. Users can add a new item, which can be of three different types: a link to a specified URL, a link to a folder in the Docusaurus website, or a link to a specified page. The dialog window for all three types is almost the same, with only the input labels and input count difference(Figure 5.13).

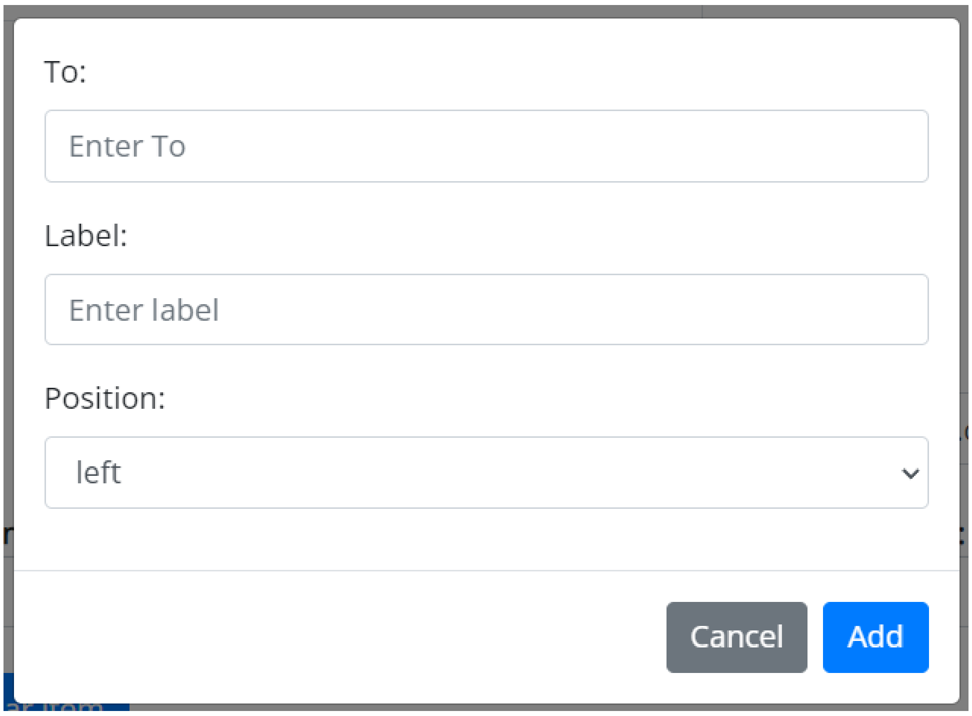


Figure 5.13: Dialog window.

Once an item is added, there are two operations available: items can be deleted, and item fields can be edited. Separate items are located in the Drag and Drop component, which allows users to manage the order of items.

5.2.3 Problems

The current solution has one minor problem: comments, that are located in the menu items section, are deleted after modifying the file with the application.

The comments in the menu items section require more significant changes in the code, because for all other elements we can simply change their values while maintaining the original structure of the node. It means that comments remain in their places. However, this is not the case with the menu items; we recreate this array from scratch. Because of the fact that the items can be added, removed, edited, and, most importantly, could change their order, it is necessary to send each item's comments to the frontend and back, which was not originally foreseen.

Chapter 6

CodeNow platform deployment

One of the supervisor's requirements was to deploy the application to the CodeNOW platform 3.1. The CodeNOW platform has preconfigured standard technology stacks. It was useful for creating the frontend component with the standard stack: npm, React, and JavaScript. However, I had to choose a custom stack for the backend, because the platform does not currently support the stack: npm and Node.js. The custom stack components are built using the DockerFile.

After all the components were created, they were merged into an application package, and the configuration for the application package was created.

The advantage of this approach is that in order to change some configuration properties user only needs to edit the application package configuration, without any additional changes in the code. After the application package was ready, it was deployed (Figure. 6.1). All container orchestration happens on the CodeNOW side, therefore, the developer does not have to set it up. Nevertheless, it is possible to set it up additionally if necessary.

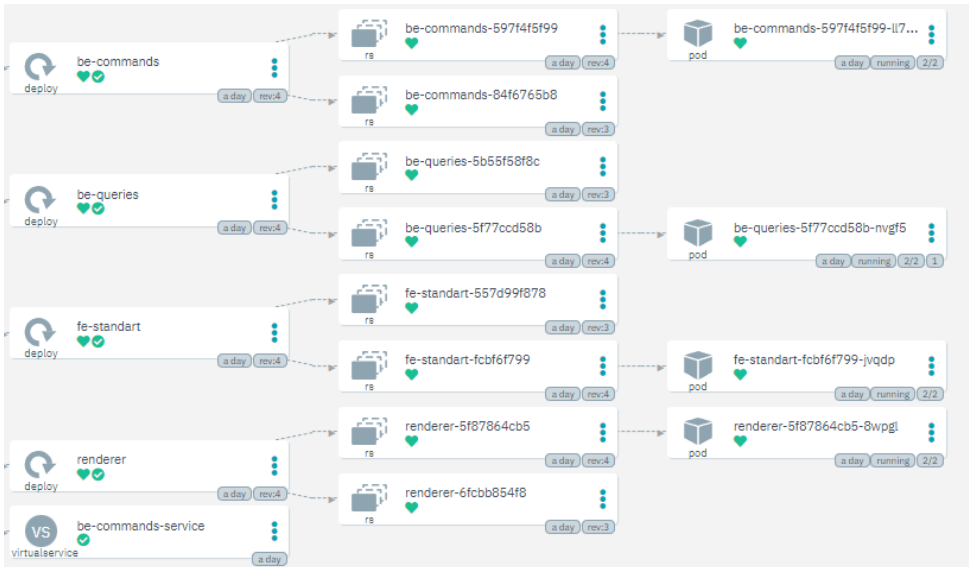


Figure 6.1: Part of the deployment.

6.0.1 CodeNOW database connection issues

During the deployment of the application to the CodeNOW platform, there were certain difficulties with the connection to the database. One of the recommended resources to read was the documentation of the CodeNOW platform. The documentation contains a guide for creating a database and connecting to it using Node.js. However, the connection was not successful, despite following the guide in the documentation.

I spent 10-15 hours trying to solve this problem. I describe all the actions I took to solve the problem below. When connecting following the guide, the database did not accept the connection, throwing an error shown on the screenshot (Code snippet 6.2).

```
ConnectionError [SequelizeConnectionError]: pg_hba.conf rejects
    connection for host "10.20.4.197", user "shareddb-owner",
    database "shareddb", SSL off
```

Figure 6.2: Connection error from the log.

Since the error does not report any specific information other than the data used to connect, I began to analyze what the problem could be. Connection data is set to the environment variables by the CodeNOW platform itself, it is unlikely that this is the problem.

Since the application had no problem connecting to the local database, but would not connect to the CodeNOW database, I decided to pay for database hosting to exclude a problem with the application itself. After paying for the database on ElephantSql hosting [26], the application was connected to the database successfully. This meant that the problem was with something else. I assumed that the problem is with SSL, and after corresponding with a CodeNOW employee my assumptions were confirmed. He said that the database only accepts secure SSL connections.

The guide says nothing about SSL, therefore I had to analyze this problem. I realized that in order to connect to the CodeNOW database I needed to get a certificate that would allow me to make a secure connection. However, after consulting with the supervisor, I found out that my predecessors had connected to the same database without any certificates configured. Then I started looking for the applications that are connected to the CodeNOW database to compare them with mine and find out what are the differences. The Spring application, that is connected to the CodeNOW database, was found. However, after studying the repository with the application, I did not find any information related to SSL certificates. The only difference from my application was that the Spring app used a preconfigured technology stack.

Using this information, I concluded that the CodeNOW platform configures

SSL certificates for the pre-configured stack somewhere in the background and the configuration of certificates for the custom stack must be done by the developer himself. However, this is not mentioned anywhere in the documentation of the platform, therefore it remains my guess.

During consultations with the supervisor, I also found out that the guide I used to connect to the database had never been tested. I concluded it is possible that the development team at CodeNOW was unaware of this problem. The problem has been described and reported to CodeNOW technical support.

As a hotfix, I created another database on the CodeNOW platform that accepts connections without SSL certificates. In the current state, the application is running on this database.

Chapter 7
Testing

Testing is a significant part of application development. It plays a crucial role in ensuring the quality and reliability of software products. By identifying and fixing problems early in the development process, testing helps prevent late delivery and software defects that can damage brand reputation and lead to disappointment and loss of customers.

During development, the implementation was tested manually and in other ways, which will be discussed below.

7.1 User testing

Three people were chosen to be the testers of the application. All of the participants are FEL SIT students, two of them are developers, and one is a UI designer. Each participant received a test scenario, a link to the app, and a list of questions to be answered after the test. The test scenario for all testers is the same, except for these extra steps for User 2:

- Login to the application with the GitLab account.
- Look to the GitLab repository and find the path of the docusaurus.config.js file.
- Open documentation "Test documentation" and go to the settings.
- Set this path in the settings of the documentation on the website.

The following is a common test scenario for all three users:

- Open documentation with the name "Test documentation".
- Open tab with docusaurus configuration editing.
- Select 'main' branch.
- Change 'title' to the "Application title".
- Change 'tagLine' to the "Application tagline".

- Change ’url’ to the "Application URL".
- Change ’onBrokenLinks’ to "ignore".
- Change ’projectName’ to the "Application project name".
- Change ’organizationName’ to the "Application organization name".
- Delete all menu items.
- Add an item of each type to the menu items with the following data:

  Item1 - Link to document: Type:’Doc’; DocId:’page-one’; Label:’Item 3’; Position:’left’;

  Item2 - Link to folder: To:’/docs’; Label:’Item 2’; Position:’left’;

  Item3 - Link to URL: href:’https://codenow.com’; Label:’Item 1’; Position:’left’;
- Change the item’s order to Ascending by the Item number from the label. You can do it by dragging and dropping items.
- Update configuration file
- Go to the repository and check that the docusaurus.config.js file contains changes made.

Each tester received the following list of questions to evaluate the functionality of the application:

- What are the pros?
- What are the cons?
- Have you noticed any bugs?
- Do you have any tips for improvement?

Below I give feedback from each of the testers:

- User 1

  Description of the tester: Java developer, with 2 years of commercial experience.

  Props: User-friendly design, easy to change the order of the items.

  Cons: Missing description of each item label.

  Bugs: Did not notice.

  Improvement: Item labels description, dark theme for the frontend.

- User 2

  Description of the tester: Java developer, with 4 years of commercial experience.

  Props: Nice design and colors, fast working application.

  Cons: It's not obvious, that changing the order of items is done by drag and drop.

  Bugs: Did not notice.

  Improvement: Move "Update configuration file" to the bottom of the page.

- User 3

  Description of the tester: UI designer, with 4 years of commercial experience.

  Props: Fast work of the app, mainly good interaction with the user (for example, drag and drop elements to change order), clean and minimalistic design, user-friendly representation.

  Cons: There is no confirmation window before deleting a navbar item; It is not obvious that navbar items are navbar items: "Navigation bar configuration" header does not look like a header; the "Update" button is on top of the page, which is not the usual location for the submit button (I would like to see it after the form and navbar configuration);

  Bugs: Did not notice.

  Improvement: Fix the UI/UX problems described in Cons.

After testing, it became clear that the app has some small unfinished details, such as the save button at the top of the page, the lack of a confirmation dialog when deleting an item, and the wrong label name. The testers praised the good design and performance of the application. No bugs were found.

7.2 Smartlook

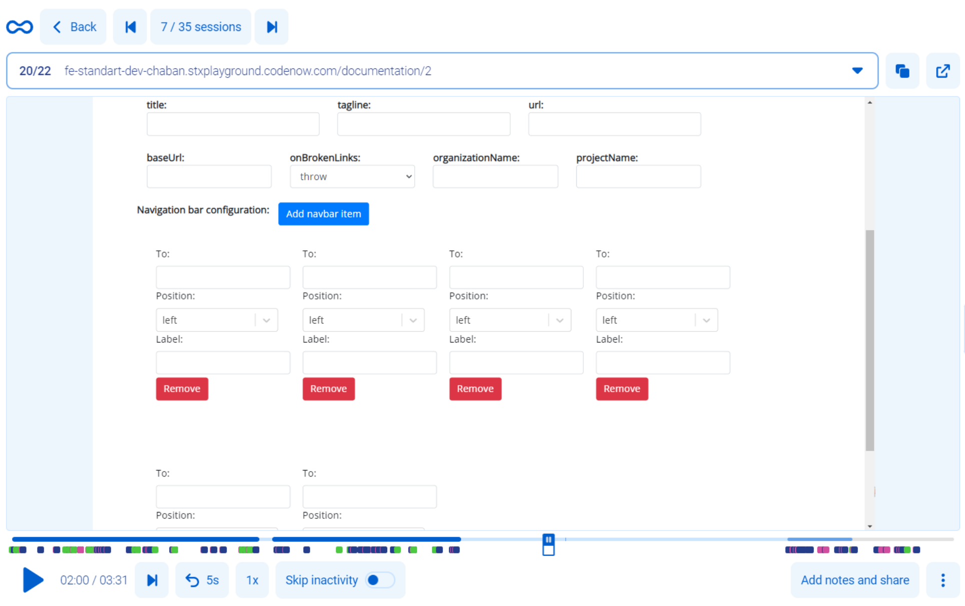
The Smartlook [27] is also connected to the app. Smartlook allows us to record the user's actions on the website, then it is possible to recreate the error or bug by following user actions.

Integrating Smartlook into the application is simple. To do this, a Smartlook account is required. In the user account, there is an API key, which should be passed to the Smartlook client initialization function (Code snippet 7.1).

After it, all user sessions will be recorded and stored. There is an example of the recorded user session.

```
// Initialize smartlook
smartlookClient.init(apikey);
```

Figure 7.1: Smartlook initialization.



Back | 7 / 35 sessions

20/22 fe-standart-dev-chaban.stxplayground.codenow.com/documentation/2

title: | tagline: | url:

baseUrl: | onBrokenLinks: throw | organizationName: | projectName:

Navigation bar configuration: Add navbar item

To: | To: | To: | To:
Position: left | Position: left | Position: left | Position: left
Label: | Label: | Label: | Label:
Remove | Remove | Remove | Remove

To: | To:
Position: | Position:

02:00 / 03:31 | 5s | 1x | Skip inactivity | Add notes and share

Figure 7.2: Example of the recorded session.

Chapter 8

Conclusion

This thesis focused on identifying and implementing improvements to an existing application. I reviewed the literature that pertains to this topic. Namely, the basic knowledge about microservice architecture and patterns was obtained, a thesis describing the application was read, and the documentation of the CodeNOW platform was studied. With the help of the course on Udemy [16], I gained knowledge about the Docker and the orchestration of containers, which came in handy when deploying the application. Next, the application was launched locally and on the CodeNOW platform using the obtained knowledge. I fixed the bug in the application and also solved the node version problem.

An analysis of the existing functionality was performed, and further improvements that could increase the business value were suggested.

I analyzed all the proposed improvements with the help of the supervisor, after which the two most reasonable were selected: partly implemented CQRS pattern and Docusaurus configuration editing page. I designed selected solutions and implemented them. The implementation proceeded according to the iterative approach, which consists of performing the work with a continuous analysis of the results. This was possible thanks to regular consultations with the supervisor.

During the implementation, I encountered problems with the implementation and the application deployment. The implementation problems were described and solved, while the problems with the CodeNOW platform were documented and handed over to the technical department.

The implementation was built, tested, and deployed to the CodeNOW platform, so the final version of the implementation did not have any significant problems or bugs. This was confirmed by the user testing of the application. Testers praised the design and performance of the application.

Overall, the implemented improvements have effectively addressed the identified requirements, resulting in an improved application with increased business value, improved load-balancing capability, and increased usability for non-technical users.

8.0.1 Further improvements

After implementation and testing parts, I formulated the following ideas to improve the application in the future:

- UI changes (Menu items label name, confirmation dialog, and position of the 'Update' button) 7.1.
- Data stores split 5.1.5.
- Saving comments from the navbar menu items section, after editing the configuration file 5.2.3.

Bibliography

1. SAJDL, Vojtěch. *GIT based markdown online editor*. Prague 6, 2021. Available also from: `https://dspace.cvut.cz/handle/10467/95359`. master thesis. CTU.
2. STETTINA, C. J.; HEIJSTEK, W. *Necessary and neglected?: an empirical study of internal documentation in agile software development teams*. Available also from: `https://dl.acm.org/doi/10.1145/2038476.2038509`.
3. *GIT based markdown online editor*. Available also from: `https://github.com/Pryx/git-md-diff`.
4. *What is CI/CD?* Available also from: `https://www.redhat.com/en/topics/devops/what-is-ci-cd`.
5. *Use containers to Build, Share and Run your applications*. Available also from: `https://www.docker.com/resources/what-container`.
6. *What is Docker?* Available also from: `https://www.ibm.com/topics/docker`.
7. *JavaScript Tutorial*. Available also from: `https://www.w3schools.com/js`.
8. *Node.Js*. Available also from: `https://nodejs.org/en`.
9. *React*. Available also from: `https://reactjs.org`.
10. *Docusaurus*. Available also from: `https://docusaurus.io`.
11. *Markdown Guide*. Available also from: `https://www.markdownguide.org`.
12. *MDX Docs*. Available also from: `https://mdxjs.com/docs`.
13. BASS, Len; CLEMENETS, Paul; KAZMAN, Rick. *Software Architecture in Practice: Second Edition*. Addison-Wesley Professional, 2003. ISBN 9780321154958.
14. RICHARDSON, Chris. *Microservices Patterns*. Manning Publications, 2018. ISBN 9781617294549.

15. JONES, Joel. *Abstract Syntax Tree Implementation Idioms*. 2003. Available also from: `https://hillside.net/plop/plop2003/Papers/Jones-ImplementingASTs.pdf`. Department of Computer Science at University of Alabama.

16. *Kubernetes for the Absolute Beginners - Hands-on*. [N.d.]. Available also from: `https://www.udemy.com/course/learn-kubernetes`.

17. *CodeNow docs*. Available also from: `https://docs.codenow.com`.

18. *Node.Js changelogs*. Available also from: `https://github.com/nodejs/node/blob/main/doc/changelogs/CHANGELOG_V17.md%5C#17.0.0`.

19. *PostgreSQL: The World's Most Advanced Open Source Relational Database*. Available also from: `https://www.postgresql.org`.

20. *GitLab pages*. Available also from: `https://docs.gitlab.com/ee/user/project/pages`.

21. *CQRS pattern*. Available also from: `https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs`.

22. *Configure GitLab as an OAuth 2.0 authentication identity provider*. Available also from: `https://docs.gitlab.com/ee/integration/oauth_provider.html`.

23. *Esprima*. Available also from: `https://esprima.org`.

24. *Recast*. Available also from: `https://www.npmjs.com/package/recast`.

25. *Escodegen*. Available also from: `https://www.npmjs.com/package/escodegen`.

26. *ElephantSql*. Available also from: `https://www.elephantsql.com`.

27. *Getting started with Smartlook*. Available also from: `https://help.smartlook.com/docs`.