Anika Harju

DESIGN AND IMPLEMENTATION OF A WEB PORTAL IN A CONTAINERIZED ENVIRONMENT

South-Eastern Finland
University of Applied Sciences

XAMK

South-Eastern Finland University of Applied Sciences

Degree title	Bachelor of Engineering – Information Technology
Author(s)	Anika Harju
Project title	Design and Implementation of a Web Portal in a Containerized Environment
Year	2022
Pages	57 pages

ABSTRACT

A web portal is a must-have in today's business world. Constantly changing market forces require companies to become more adaptable and customer focused. A web portal is one way to connect a company and its audience through a single point of access, reducing customer and client communication. As a result, the web portal should provide a comprehensive view of service offerings via a dynamic user interface. Against this backdrop, this study examined the design, tooling, and best practices for building and deploying a web portal. The role of the Agile approach, DevOps, and containerization in web portal development was a significant focus of this study. Secondary sources for the study included peer-reviewed journals and academic books, while other electronic sources provided technical guidance for the web portal deployment. The theoretical foundation formed the basis for developing and deploying an active web portal application.

Keywords: web portal, container, user interface, front-end, back-end

Acknowledgements

First and foremost, I must thank my supervisor, Dr. Matti Juutilainen, for his dedication and guidance throughout this study. In addition, I would like to thank the managing director of Datalounges Oy, Kim Aaltonen, for his advice and support for this study. I am sincerely thankful to cloud architect Iiro Wallgren for sharing his technical expertise on the study subject. Last but not least, I express my heartfelt appreciation to my loving husband, Dr. Aaro Harju, for his support.

CONTENTS

1 INTRODUCTION

An online gateway is crucial in the business sector. The information must be available instantaneously, everywhere around the globe, for both clients and companies. In the online environment, a web portal has a specific function. (Sharma & Gupta 2005). The portal represents several web-based interfaces referred to as a dynamic one-stop homepage where visitors can tailor the material to their needs. A website that serves as a hub for information on the Internet is a web portal. (Mane & Pange 2015)

With User Interface (UI) technologies like React, Cascading Style Sheets (CSS), JavaScript (JS), and other responsive web design tools, a modern web portal design will enhance a company's brand. A web portal is the best digital marketing tool for promoting current and prospective software services to attract new customers. Clients can see and choose services using a self-service paradigm that is part of the website. The web portal also enables online shopping and presents targeted content, chat rooms, and service offerings in a dashboard-style layout. Using business intelligence reporting solutions' analytics, the online store also makes it possible to monitor customer activity. (Shivakumar 2016, 11)

1.1 Web Portal Services

Clients can view the company's services via the web portal. It gathers information from numerous sources and utilizes single sign-on (SSO) features. Shivakumar (2016, 4) listed the following as services offered by a web portal:

• Personalization provides content, a customized user experience, and functionality.

• With SSO, connections, and applications are simple to access.

• The ability to manage content and its formats for the administrator.

• Gathering information from several sources, including databases, third-party apps, Enterprise Resource Planning (ERP) systems, etc.

• Secured search tools that access all enterprise data sources.

• Language localization for the material.

• Functionality across several platforms, including tablets, smartphones, and desktop browsers, with multichannel access.

• Social media communication tools-blogs, wikis, forums, chats, and more.

1.2 Goals

The study aims to establish a web portal that offers a user-friendly, intuitive, and simple-to-navigate online platform based on business requirements. The study also looks at the design principles, best practices and deployment measures required for a web portal. The web portal would enable present and potential clients to access service metrics quickly and conveniently at their fingertips around the clock.

1.3 Research Questions

The study's focus is to develop a dynamic and accessible web portal centered around the following questions:

· What are the design principles of a web portal?

· What are the components and architecture of the web portal?

· What are the deployment methods for a web portal?

· What testing methods does a web portal employ?

The theoretical section of the study describes a service web portal, the design concepts, development methodologies, testing, security, and evaluation techniques. The practical section entails gathering requirements, choosing the best technology, building architecture and components, deployment, testing usability and development processes. The literature review details the description of a web portal, design principles, usability, security, testing, and evaluation in chapter 2.

2 LITERATURE REVIEW

2.1 The Web Portal

A web portal is a vast Internet resource system that provides information services. The terms portal website, portal site, and online gateway are all used to describe web portals. Search engine functions and indexing services were the initial focus of web portals, such as Yahoo, which served as a crucial gateway for new Internet users looking for information in the early years. (Che & Ip 2017) Nowadays, the Internet is a never-ending source of information of all types, and online portals are one tool business owners use to interact with clients on digital platforms (Paiano & Roggerone 2010, 65).

The main strength of a web portal is its capacity to provide the best integrations and advance corporate goals if the user experience is positive. The three key benefits of online portals for business models are content management, analytics, and search. (Shivakumar 2016, 3)

Since the popularized notion of online portals by search engine websites like Yahoo!, Excite, and Lycos, portals have become commonplace and an essential component of the web (Tatnall 2009, 3642). Web portals have become a fascinating topic in web engineering due to the rapid growth and the enormous increase in online services. Today's web portals reflect wide-ranging features, models, and structures. (Sampson & Manouselis 2005, 186, 206)

Portal technology has improved company capabilities while simultaneously addressing Internet opportunities and difficulties. Users can readily share information regardless of physical location (Ben-Natan et al. 2004, 4). A web portal uses a technological infrastructure that supports and secures the data and applications on the web. It's vital to note that web portals and web pages are different in that a portal focuses on users and clients, and a page focuses on the business/organization. (Bhojaraju 2007, 522)

The second-generation web portals have component-oriented architecture, and the capacity to build a portlet is essential (Costagliola et al. 2008, 215). A portlet is a web application that uses a portlet window to display chosen specific content. While each portlet is autonomous, it can connect via messages. Illustrated in Figure 1 are the various operating modes of portlets. (Polgar 2012, 49; Guruge 2003, 112)

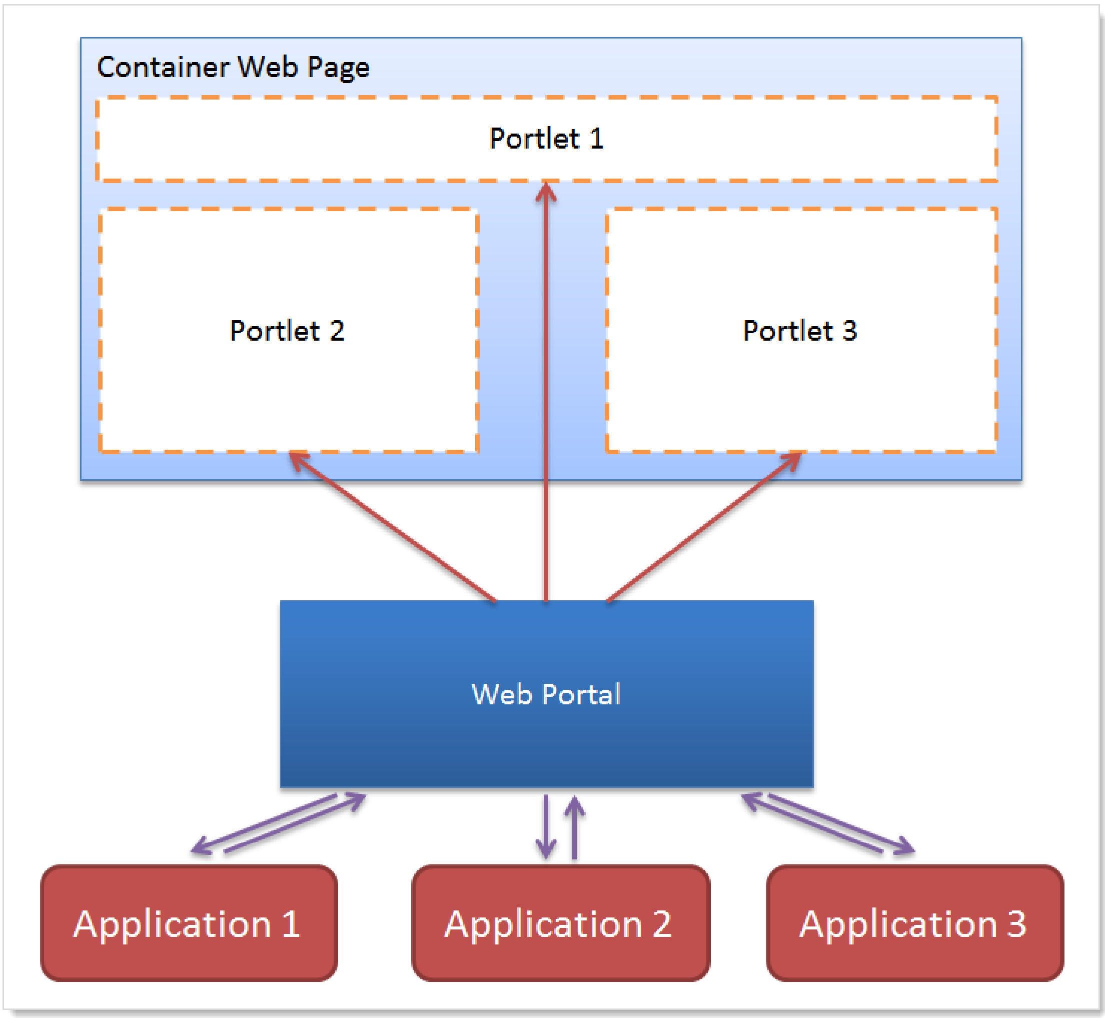


Figure 1. Internet and Intranet portals - How do they work? Source: Lozhkin 2011.

The placement of the portlet is essential because it affects both the overall usefulness and connectivity (Costagliola et al. 2008, 223, 226). Web portlets are smaller programs that condense online applications to package and make individual information units available. The portlets are installed, found, and made reachable. The system also saves and keeps track of each portlet session and user preferences. (Cheng et al. 2008, 163) If the business wishes to achieve effective connectivity between the portals, it is crucial to develop a controlled process for creating and deploying portlets (Costagliola et al. 2007, 516).

The specification outlines the window states and portlet modes. A portlet can be static content, or users must explore and interact with screens.

Additionally, portlets can carry out routine duties like user personalization or informational portlet advice. (Costagliola et al. 2007, 519)

Web portal technology eliminates organizational borders and functions as a central hub for data, apps, workflows, and services. The portal system, which can also serve as a central repository, is available after just one registration and login procedure, enabling users to find and connect several programs into a process chain. (Cheng et al. 2008, 163)

For a user or user group to use a web portal effectively, it must be customized and altered. The services of a web portal can frequently be tailored to a specific user or set of users through web portal customization. The user's interest should be taken into consideration when designing portal pages. As the design ideas improve the web portal and its functionality, the most commonly used portlets on the main page can be modified. (Costagliola et al. 2008, 223)

Building the portal system to serve as an intranet and content management system for the workforce of an organization is intelligent. Additionally, the web portal enhances communication between the company's clients online. (Cheng et al. 2008, 163) Web portals work as intended when the organization, presentation, and integration are developed and connected (Ben-Natan et al. 2004, xxv).

Nowadays, horizontal portals with shallow content and a sizable audience make up the majority of portals in use. Vertical portals, specialized portals, or vortals are information or knowledge portals that concentrate more on particular user groups and try to give users a single piece of information. (Bhojaraju 2007, 522)

2.2 Design Principles

The construction and upkeep of a web portal are expensive and require hired labor. As a result, the online portal needs to be developed with the user experience and results in mind. (Bringula & Basa 2011, 254) According to Bringula (2016, 816), the design of a web portal is crucial and should completely satisfy the user.

Creating a portal is a monumental task. It is vital to connect disparate subsystems into a uniform interface to satisfy disputed needs. People may get unsatisfied, lose interest, and switch platforms if the design does not offer a proper portal model. (Richardson et al. 2004, 175-176)

A design decision often involves an architectural or structural choice. It is vital to consider the user requirements and the available tools while creating an accessible design. (Richardson et al. 2004, 195) Site ontologies conceptually plan web portals with components like web pages, containers, and search rules (Zhou et al. 2005, 588).

The core elements of a portal design should include subject categories, personalization, visual design, and multilingualism (Large et al. 2004, 45, 52, 59-60). It is easier to develop a functional portal system when one is aware of the preferences and behavior of the target user groups under various usage scenarios (Agosti et al. 2010, 225-227). Precision, consensus, consistency, and aesthetics contribute to the performance and high usability of a portal, according to Bringula and Basa (2011, 255).

The importance of visual design web portal components such as color, font, graphics, motion, iconography, characterization, and layout are growing (Rau et al. 2007, 196; Large et al. 2004, 54-58). A compelling design encourages readers to concentrate on the content instead of being preoccupied with images (Rohira 2021). Users expect a company's online portal to be engaging, hence the design and aesthetic decisions selected must closely fit those needs (Shivakumar 2016, xxiv; Ruffini 2001).

Due to evolving trends and new technologies, portal designs are at a critical juncture. Portals make an effort to follow and adapt to modern trends. The gateways are gradually moving in the direction of becoming slender and light. User experience portals (UXP) are growing in popularity and adding more components to portal designs. (Shivakumar 2016, xxiv)

2.3 Usability

When evaluating a product or system, usability refers to how well the user interacts with it (Usability 2006). Usability seeks to increase user-friendliness so that users can gain tangible advantages and proposed objectives, such as the capacity to use the system more easily and effectively (Conte et al. 2007; Molina & Toval 2009). The International Organization for Standardization (ISO) defined usability as the degree to which a product or system can meet particular goals with validity, capability, and satisfaction (Go 2009, 195).

A key component of system development is usability. Systems with poor usability can be problematic for businesses and organizations, costing them money. (Tarafdar & Zhang 2005) The capacity to navigate a web portal is crucial to its usefulness. Users can find what they want with the help of straightforward navigation. (Pearson & Pearson 2008; Sindhuja & Dastidar 2009, 58)

Web portals require flexibility in addition to the usability (Gaedke et al. 2004, 517). Overwhelming content has drawbacks, particularly for visual search (Rau et al. 2007, 196). Web portals might be overloaded with pointless services if the pages are complete with all necessary information. Overloaded pages are not a good integration strategy. Users will become irritated if response times are slow and there are high error rates. (Zhou et al. 2005, 593)

Web portal service dissatisfaction is detrimental to the reputation, brand, and bottom line. Pages should only provide pertinent information about products and services offered by the company. (Zhou et al. 2005, 593; Rau et al. 2007, 196)

Functionality is close to usability. The enhanced functionality of a web portal depends on mobile responsiveness, registration, login, and customer feedback. Today, it's imperative that an online portal offers good engagement options, such as live chat, chatbots, email support, and social network chats, to ensure user happiness and comprehensive operation. (Rohira 2021)

Usability and functionality cannot be satisfied by design or aesthetics (Kumar 2016). Usability cannot compromise the security of the web portal because the portal needs to guard against both front- and back-end security breaches because more sophisticated security concerns, such as malware and malicious assaults, now threaten the integrity of companies. (Rohira 2021)

2.4 Web Portal Security

Vital information is stored online or transferred across websites, making it vulnerable to cyberattacks (Saini et al. 2014, 156). Cybersecurity is the defense of IT systems against various types of alterations or destruction. Web portals are infected with malware by hackers who want to make money or steal personal information. Security is a critical component of the architecture and foundation of the portal. (Nyirongo 2015; Rohira 2021; Chishti et al. 2008)

Web service security is influenced by an organization's security infrastructure, while enterprise application security impacts web service security. The focus areas of attention for portal designers and developers are configuring web components and integrating with enterprise back-end applications. (Richardson et al. 2004, 153) Reliable access control is the core of proactive defense strategies and procedures (Saini et al. 2014, 160; Nyirongo 2015).

2.5 Testing

The success of a web portal depends on continuous and iterative testing. Iterative testing identifies problems early and raises the standard of the finished product by utilizing test automation tools and CI techniques. Continuous and iterative portal testing yields the best results. (Shivakumar 2016, 248, 263)

Web testing is a technique used in software testing to check for potential defects in websites or web apps. Functionality testing, unit testing, usability testing, compatibility testing, interface testing, portlet testing, performance testing, and security testing are phases or components of web testing. (Web Application Testing… 2022) While unit testing confirms that every system component complies with specifications, functionality testing ensures that every web function is optimized to achieve intended results (Shivakumar 2016, 252). Usability testing provides information on the extent to which a product can be used by intended users to achieve specific goals with effectiveness, efficiency, and satisfaction in a specified context of use. It helps designers to identify what works well from the user perspective and what does not. (Ward & Hiller 2005, 156, 170) Usability testing guarantees that the applications are readily and conveniently available (Web Portal Testing n.d.).

The software undergoes compatibility testing to determine its ability to function on various hardware, operating systems, programs, network environments, or mobile devices (Hamilton 2022b; Shivakumar 2016, 253). Interface testing checks the accuracy of the communication between two separate software systems (Hamilton 2022c). The use of portlet preferences, boundary conditions, and fundamental portlet functionality are all ensured via portlet testing (Shivakumar 2016, 253). Performance testing looks at how resilient and responsive a system is to different workloads, whereas security testing provides extensive penetration testing by conducting assessments across several risk areas (Web Portal Testing n.d.; Ali-Shahid & Sulaiman 2016).

A web system developer must ensure that a web-based system is working following the specified requirements. Consequently, an application must undergo a rigorous inspection before being made available to end users. (Cavero-Baptista n.d.; Web Application Testing… 2022)

2.6 Web Portal Evaluation

A web portal's review process is participatory and continual. Various process phases, such as design, development, and deployment, involve all key actors. The review reveals the benefits and drawbacks of the portal services and offers suggestions for improvements. It is necessary to assess the functionality of all relevant web portal elements. The assessment procedure must be adaptable and analytical. (Sampson 2007, 383)

Granic et al. (2008) pointed out the significance of gathering both quantitative data and qualitative findings to receive an overall assessment and input for future growth. The evaluation process consists of three steps shown in Figure 2. All characteristics, including the evaluation objectives, are chosen at the preliminary stage. The second phase includes the completion of the evaluation activities. The collected data is compiled and synthesized with the goals during the analysis phase. During the review stages, developers, experts in evaluation, and users collaborate. (Sampson & Manouselis 2005)

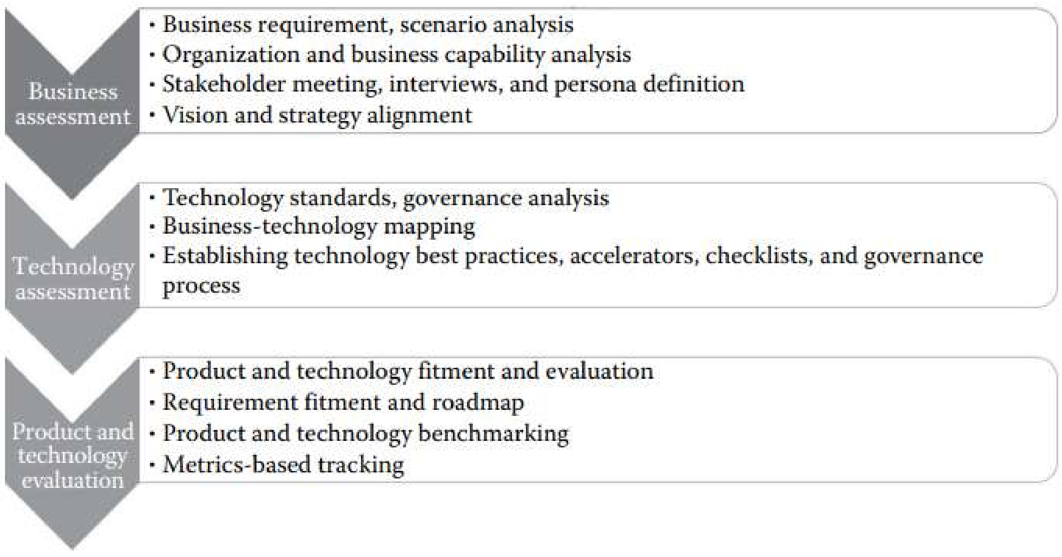

Business assessment
• Business requirement, scenario analysis
• Organization and business capability analysis
• Stakeholder meeting, interviews, and persona definition
• Vision and strategy alignment
Technology assessment
• Technology standards, governance analysis
• Business-technology mapping
• Establishing technology best practices, accelerators, checklists, and governance process
Product and technology evaluation
• Product and technology fitment and evaluation
• Requirement fitment and roadmap
• Product and technology benchmarking
• Metrics-based tracking

Figure 2. Portal assessment framework steps. Source: Shivakumar 2016, 42.

Careful analysis of the variations between intended and actual usage is necessary to provide the user with a high-quality software system experience (Besson et al. 2012, 192).

The following chapters will extend on the literature review by outlining collected data on the study topic. The third chapter discusses web portal development methodologies, tooling, and DevOps best practices.

3 DEVELOPMENT METHODOLOGIES

3.1 Agile vs Waterfall

It takes a lot of time and energy to develop software. Planning, designing, implementing, deploying, and monitoring are just a few of the several phases that go into the development process, which is labor-intensive and includes various components. It necessitates extensive analysis and several deadlines. Well-known software development approaches include Waterfall, Spiral, Agile, Scrum, and Lean. (Agile vs. Waterfall 2020)

The shortcomings of the Waterfall approach gave rise to the Agile methodology. The Waterfall technique was unable to meet the constant demands of technological advancement. (Simic 2020) The Waterfall approach is a conventional software development process with numerous phases that adheres to linear and sequential steps (Soni 2019).

The Agile method is concerned with production speed (Simic 2020). The approach is appropriate for compact, well-designed projects with precise specifications made explicit from the project's outset (Agile vs. Waterfall 2021; Agile vs. Waterfall 2020). The Agile methodology places priority on the working process than on in-depth documentation. It focuses on delivering modest, achievable project milestones through the development and testing lifecycle fluidly and swiftly in the short term, often two to four weeks. The development process is progressive, evolutionary, and iterative. The technique divides the production process into smaller parts and combines them to create the finished product. (Simic 2020; Hamilton 2022a)

The Waterfall model, on the other hand, adheres to precise processes and predetermined steps. The stringent process and complicated requirements make it impossible to implement mid-term adjustments without re-engineering everything. The Agile technique and agile processes are better suited for businesses on the market. (Agile vs. Waterfall 2020; Agile vs. Waterfall 2021)

The Agile lifecycle entails planning, designing, programming, testing, deploying, reviewing, and launching. Client contact and feedback are given a lot of weight in the Agile process. Every time a client offers feedback, a new step in the development process starts. (Agile vs. Waterfall 2020) Due to ongoing process evaluation, the Agile methodology enables clients to benefit immediately. Development teams focus on delivering work outcomes in more manageable chunks instead of the final release. (Hall n.d.) The Agile method focuses on functional and non-functional readiness, mid-project modifications, and values responding to input and adapting to new requirements. (Hamilton 2022a; Simic 2020).

During each sprint, the teams focus on designing, building, testing, adjusting, and deploying the software. The project is managed for each sprint until completed. After each sprint, the Agile methodology favors small-scale deployments. The process can accelerate in some areas of development, including code management and automation testing. (Simic 2020; Hamilton 2022a)

The Agile model has several advantages. It offers efficient daily management of challenging development projects and enhances teamwork and communication between teammates and clients (Simic 2020). As a result of the openness and feedback-based nature of the Agile model, clients are made aware at every level of the planning process. (Agile vs. Waterfall 2021; Agile vs. Waterfall 2020; Rafi et al. 2022, 1)

The disadvantage of the Agile approach is that if changes are not managed and communicated well, they could cause problems in other project areas (Agile vs. Waterfall 2021). Another barrier to the Agile approach is the necessity that teams be continuously productive (Hamilton 2022a).

There are significant contrasts between the Waterfall and Agile methods. The progress of an Agile project is reviewed daily, as opposed to the Waterfall paradigm's biweekly review. Agile teams function independently, while Waterfall teams follow the hierarchical structure. The Agile approach focuses on providing clients with business value, the Waterfall model is concerned with implementing the requirements and delivering a product once all the phases are complete. The Waterfall paradigm lacks a mechanism for feedback. (Agile vs. Waterfall 2020; Soni 2019; Seremet & Rakic 2021, 225)

3.2 What is DevOps?

Development and Operations (DevOps) is a word that is now widely used in the IT industry (Braunton 2018, 11). DevOps facilitates communication between developers and operations teams so they can function as one cohesive unit (Lwakatare et al. 2015; Humble & Molesky 2011, 6). Increasing efficiency, innovation, speed, reduction of errors, and value delivery to customers and businesses are the goals of DevOps (Adams 2021, 9; Seremet & Rakic 2021, 225; Miller et al. 2022, 25). An organization segmented workplace culture and ineffective interdepartmental coordination are to be changed using DevOps (Freeman 2019, 8; Miller et al. 2022, 25).

The DevOps mindset integrates operations, software development, system management, and quality monitoring to enable quicker and more frequent execution of changes in production at several levels, which might be cultural, practical, and tooling-related (Adams 2021, 7). Not only is DevOps a new way of working, but it's also a new way of thinking (Miller et al. 2022, 26).

Software development and IT operations can be automated and integrated using the DevOps conceptual framework and practice (Ogala 2022, 1). Improved scalability and monitoring of frequent end-to-end deployments using continuous integration (CI) and continuous delivery (CD) are made possible by DevOps. (Myrbakken & Colomo-Palacios 2017; Hamilton 2022a; Simic 2020; Krief 2019)

DevOps emphasizes teamwork, empowerment, and giving people priority over processes and tools. It is a combination of people, procedures, and products that values user needs. Value is the fundamental concept. Developers, testers, operation personnel, security experts, and infrastructure engineers are all part of the DevOps approach to the development process (Krief 2019; Freeman 2019, 7-8; Five Best Practices…2020). DevOps seeks to establish cross-functional, collaborative teams to take charge of the entire product lifecycle. The building blocks for teamwork are architecture, design, and development. (Adams 2021, 13; Braunton 2018, 11; Hall n.d.)

The DevOps application is a powerful technique for encouraging agile communication between the development team and IT operations, coordinating business activities with IT to support results like profitability (Pestchanker 2018; Chowdhury 2019). DevOps aims to build, test, and distribute software products in a timely and reliable manner. DevOps emphasizes the connection and loop between the stages of the development lifecycle seen in Figure 3. (Ogala 2022, 1)

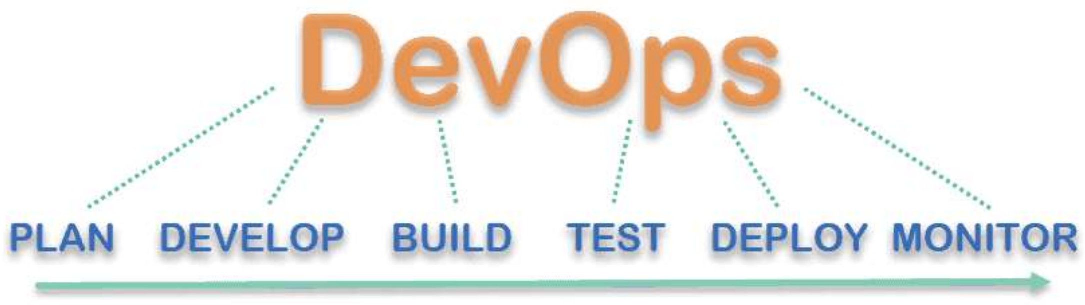


Figure 3. DevOps methodology. Source: Simic 2020.

DevOps values a culture that encourages constant improvement, changes, and adaptation (Soni 2019; Adams 2021, 13). DevOps culture focuses on people, procedures, and technologies to automate the application lifecycle while creating high-quality software to obtain competitive advantages (Soni

2019). Behavior, liability, trust, and ongoing productivity are all part of the DevOps culture shown in Figure 4 (Freeman 2019, 7-8; Adams 2021, 7). The DevOps culture places focus on client input and value and stakeholder interests (Adams 2021, 16-20).

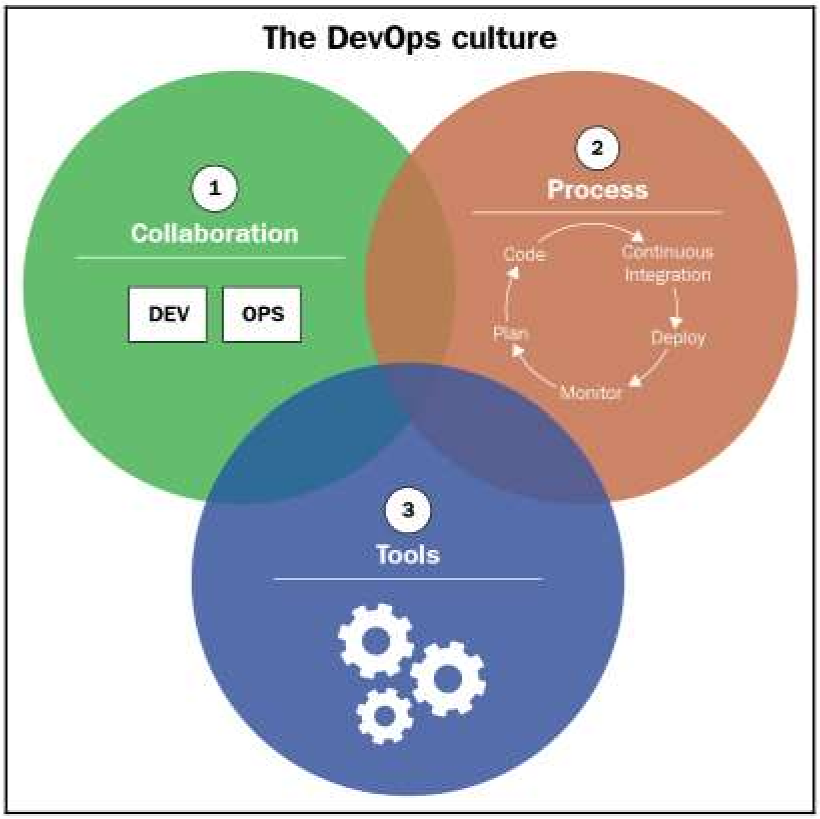


Figure 4. DevOps culture elements. Source: Krief 2019, 11.

DevOps techniques are linked to software, applications, and products and adhere to best practices to obtain better quality with effective communication and end-to-end responsibility (Soni 2019; Simic 2020; Ismail 2018). In contrast to conventional software development, DevOps involves continuously creating, testing, deploying, and maintaining software (Simic 2020). DevOps is focused on releasing code to manufacturing daily or every few hours and does not favor commonly used structures (Hamilton 2022a).

3.3 Technology Best Practices

A good DevOps best practice improves team dynamics, office atmosphere, and processes (Dhaduk 2022). Monitoring, infrastructure and deployment automation, performance management, and configuration management are all parts of the DevOps lifecycle (Dhaduk 2022; Ogala 2022, 1). To manage the complex web of dependencies found in modern apps, DevOps teams today use an application-centric approach rather than an infrastructure-centric one in the past (Five Best Practices...2020).

In DevOps, developers must adhere to and maintain quality standards using the best coding and architectural practices (Hamilton 2022a; Simic 2020). DevOps encourages frequent code revisions, which makes finding bugs in code straightforward (Chowdhury 2019). Developers and IT managers can quickly find and fix bugs with proper code version control. The DevOps teams can work together smoothly to correct errors if a failure occurs. (Chowdhury 2019; Ogala 2022, 5; Simic 2020) Version control makes it simpler to combine changes in the code within the files and store the codes in a single repository for recovery (Dhaduk 2022). To ensure that operator or organization notifications are received, automated control systems utilize DevOps (Ogala 2022, 2).

Developers may test and fix problems throughout the process with DevOps, not only in the final stage (Chowdhury 2019; Simic 2020). Manual testing, however, is extremely slow and prone to implementation problems (Krief 2019, 442). Automated testing is compulsory for a good DevOps framework. Automated testing provides meaningful feedback and determines the risk of the procedure. (Simic 2020; Chowdhury 2019)

A key component of DevOps best practices is continuous deployment. Instead of deploying the code together, continuous deployment deploys it in versions. (Chowdhury 2019; Best Practices…2019) Continuous deployment entails the automated release of updates that pose no damage to the current architecture with minimal human involvement (Simic 2020; Chowdhury 2019; Rafi et al. 2022, 2).

According to Dhaduk (2022), best practices also permit the adoption of microservices architecture, which divides complex programs into independent parts. Each component runs independently of the application, offers services, and uses application programming interfaces (APIs) for communication. (Dhaduk 2022)

Constant performance monitoring allows for application performance optimization seen in Figure 5 (Chowdhury 2019). Due to frequent and minor software releases, DevOps teams must continuously check performance (Simic 2020).

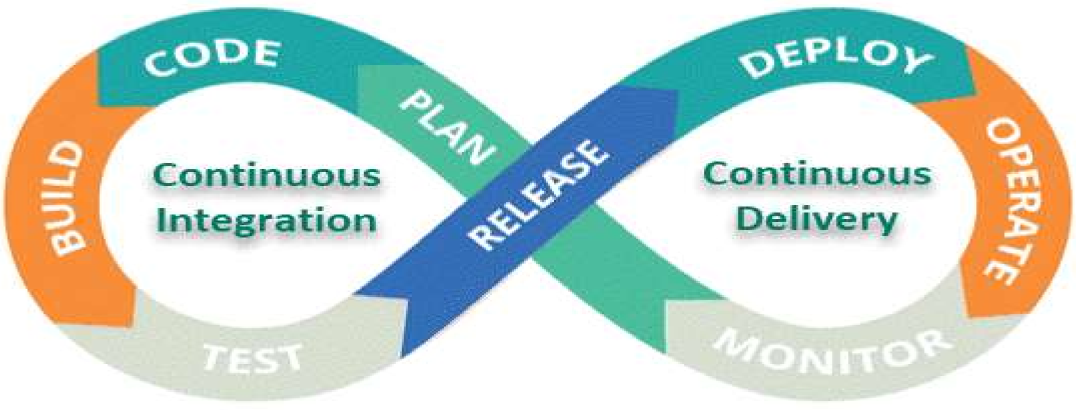


Figure 5. DevOps lifecycle by Continuous Integration and Continuous Delivery. Source: Ogala 2022, 4.

Before an application goes into production, developers and operations teams can use log and event management to track its activity. This process is a component of the overall pipeline delivery process. (Ogala 2022, 2) The goal is to prevent interruptions during the release procedure (Simic 2020).

3.4 Tooling to Support Developments

Developers can automatically merge unified code into a centralized repository as part of the CI process. Autonomous tools examine the accuracy of the code and make the necessary corrections illustrated in Figure 6. The end-users receive speedy, reliable delivery of the written code through CI. (Dhaduk 2022; Krief 2019, 13, 164)

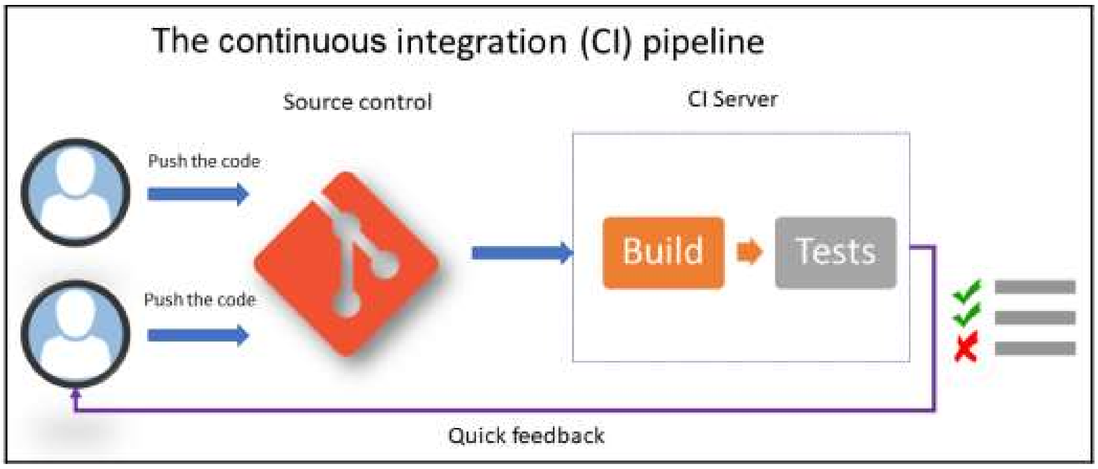


Figure 6. The pipeline of Continuous Integration. Source: Krief 2019, 14.

Upon completion of CI, the automatic deployment of the application occurs in a non-production environment. The CD process starts when CI creates a package for an application. The purpose of the CD is to test the complete application, including all its dependencies. CI and CD are commonly used interchangeably in integration settings shown in Figure 7. (Krief 2019, 15)

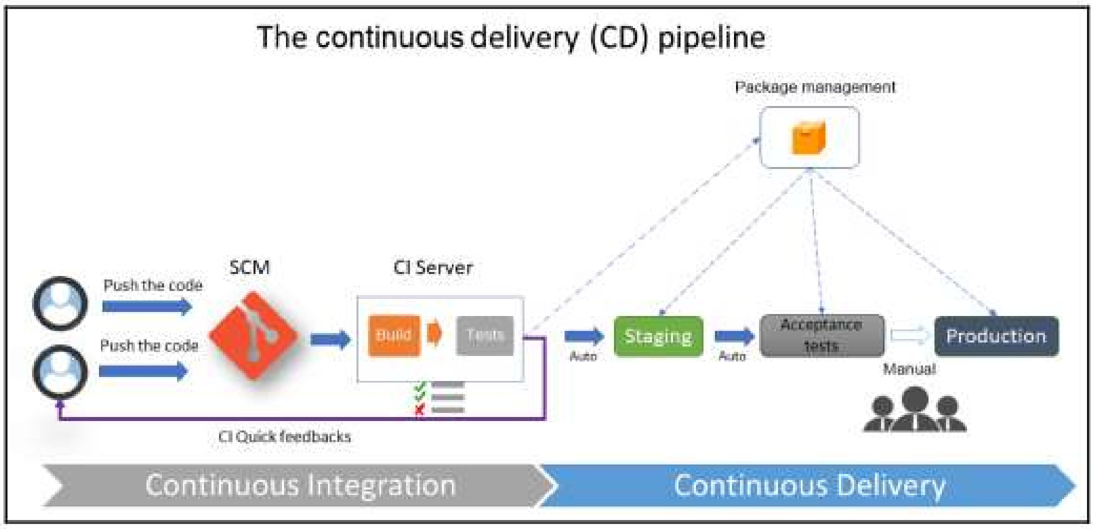


Figure 7. The pipeline of Continuous Delivery. Source: Krief 2019, 16.

The CI tool builds a pipeline for the development, staging, and production phases (Simic 2020). The CI tool runs tests against the scripts and applications after incorporating and reducing occurs (Dhaduk 2022; Chowdhury 2019).

The code is added to a shared repository multiple times every day. Each developer divides the job into smaller, compact portions of code and searches for any errors. (Simic 2020) The developer pushes the updated code to the staging environment for testing before proceeding to production (Braunton 2018, 12). The fourth chapter examines the business requirements for developing the web portal.

4 USE CASE: DATALOUNGES OY

4.1 Overview of Datalounges Oy

Datalounges Oy is a rapidly growing company founded in 2014. The infrastructure software company, based in Finland and Norway, uses business-to-consumer (B2C), business-to-business (B2B), business-to-government (B2G), and marketplace platforms as its business models. (Datalounges 2022b)

Datalounges Oy provides a wide range of IT infrastructure services, including Tuuli Web and Cloud-hosting, Tuuli Kubernetes, R4DAR, Kiubit DevSecOps, and Kiubit Platform, illustrated in Figure 8 (Datalounges 2022b).

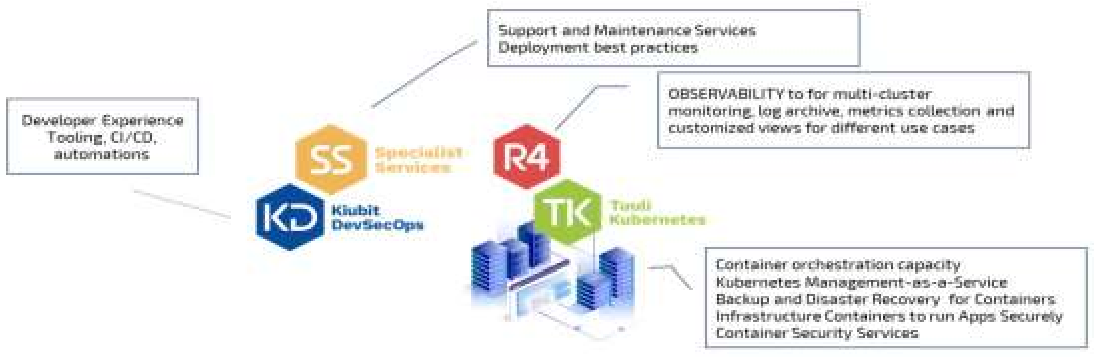

Support and Maintenance Services
Deployment best practices
Developer Experience
Tooling, CI/CD,
automations
OBSERVABILITY to for multi-cluster
monitoring, log archive, metrics collection and
customized views for different use cases
SS
Specialist
Services
R4
KD
Kiubit
DevSecOps
TK
Tuuli
Kubernetes
Container orchestration capacity
Kubernetes Management-as-a-Service
Backup and Disaster Recovery for Containers
Infrastructure Containers to run Apps Securely
Container Security Services

Figure 8. Datalounges software services. Source: Datalounges 2022a.

Datalounges Oy offers advanced deployment and maintenance services, Ceph, Kubernetes, Linux, Automation, OpenStack, and Container orchestration support, along with other specialized IT knowledge (Datalounges 2022b).

In this context, the web portal would eliminate back-and-forth communication between clients and Datalounges Oy to boost productivity and efficiency and provide customers with multichannel mobile support.

4.2 Business Requirements for the Web Portal

The needs of a web portal are defined based on decision-makers, employees, end users, preferences, intentions, beliefs, present conflicts, etc. It is essential to divide the requirements into functional and non-functional categories. While the non-functional requirements are concerned with accessibility, usability, and visuality, the functional requirements cover the major content concerns. (Arantes 2013, 140, 144; Besson et al. 2012, 191)

A vital requirement for a portal is the delivery of dynamic information, which involves effectively gathering the essential data and translating it into a specified display structure. A keyword-based search feature and navigation through the content pages without interruption are also crucial. The search tool allows end users to have quick access to essential information. The online portal should offer knowledge management options for sharing resources like files, papers, and collaboration tools like chat or blogs. (Shivakumar 2016, 199-200, 231)

A web portal framework should take the requirements and functionalities into account. The suggested framework seeks to establish the primary and secondary goals of the portal by gathering needs through interviews with various stakeholders, including end users, decision-makers, and employees. (Tsui et al. 2007, 1027)

A requirement is a feature or functionality the system must deliver. If the requirement is established but does not affect a stakeholder, it should be considered a design or implementation decision. The requirements are essential for documenting what stakeholders want a system to accomplish. Operational requirements prioritize continuous portal uptime to avoid user operations and maintenance tasks from conflicting. Interface requirements deal with the fundamental flow of content into and out of the portal through its portlets. The requirements must, above all else, be accurate, reasonable, and traceable to be considered valid. (Richardson et al. 2004, 177, 181)

The web portal business requirements document (BRD) outlined in this study are in keeping with the strategy, objectives, and aims of Datalounges Oy. The requirements encompassed the services provided by Datalounges Oy, and the needs of staff, clients, and business partners (see Appendix 1). The fifth chapter discusses essential architecture and components to build the web portal.

5 ARCHITECTURE AND COMPONENTS

5.1 The Web Portal Framework

The architecture for a web portal integrates several components, including content, information, application, and security architectures. Detailed explanations of strategies and scenarios are available in the content architecture. Information architecture elements include a paradigm for site classification, content, and portal navigation. (Shivakumar 2016, 74, 110-111)

The definition of the portal strategy, evaluation of the portal technology, and formulation of the portal roadmap are the three steps of the portal architecture. The content planning, user assessments, and needs make up the bulk of the site strategy. Different architecture views, portal principles, best practices, and technology evaluation are all included in the assessment of portal technology. Critical architectural technology components include portal performance and portal site usability. (Shivakumar 2016, 98, 110-111)

This study adopted a three-tier architectural framework to implement the functionality of the web portal demonstrated in Figure 9 and Table 1.

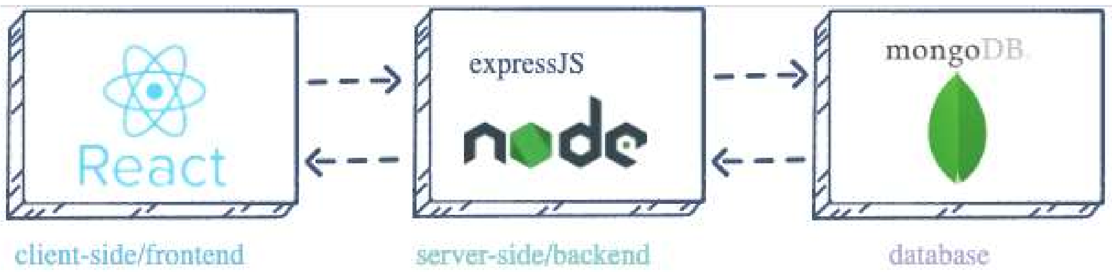
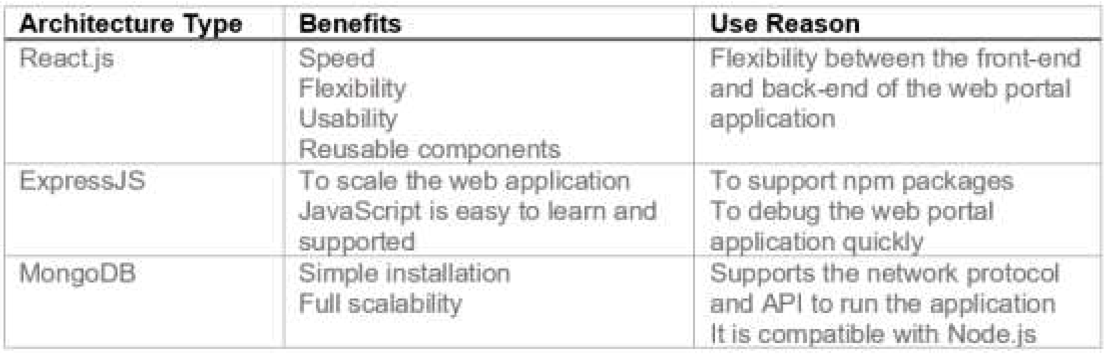


Figure 9. The web portal architecture. Source: MEAN Stack vs MERN Stack 2022.

The web portal used React.js at the front-end. React employs a component-centric methodology (Surve 2021). Functions in JS are React components. An individual Hypertext Markup Language (HTML) element, known as the root node, is the foundation for most React applications.



Architecture Type	Benefits	Use Reason
React.js	Speed Flexibility Usability Reusable components	Flexibility between the front-end and back-end of the web portal application
ExpressJS	To scale the web application JavaScript is easy to learn and supported	To support npm packages To debug the web portal application quickly
MongoDB	Simple installation Full scalability	Supports the network protocol and API to run the application It is compatible with Node.js

Table 1. Web portal architecture benefits

The only method to modify a React element is to continually render a new component because React elements are fixed and unchangeable (What is React? n.d.; Tutorial n.d.). React does not entirely replace HTML as a programming language, it makes it possible to create interfaces using similar syntax and to utilize JS to add dynamic capabilities (Barger 2021). Additionally, CSS coding is simple and unaffected by React (Eisenman 2016, 13). React offers web application components for buttons, text, labels, and grid designs. The wide-ranging ecosystem of the application contributes to the platform's enormous user base. (Surve 2021; Choi 2020)

In the middle layer Express and Node.js handled routing and Hypertext Transfer Protocol (HTTP) requests and responses of the web portal. Express.js is a straightforward and adaptable Node.js web application framework that offers a wide range of functionality for creating web and mobile applications. Express.js simplifies the creation of APIs and web apps, reduces coding time, and improves the effectiveness of web and mobile applications. The Expess.js web application is a layer added on top of Node.js that aids in managing servers and routes and is used to develop single page, multipage, and hybrid online applications. (Express 2017; ExpressJS Tutorial 2022; What is Express JS in Node JS? 2022)

In the back-end the MongoDB stored the data of the web portal. MongoDB is an open-source database management system that can handle and store document-oriented data. In contrast to relational databases, which use tables and rows, the architecture of MongoDB is made up of collections and documents, supporting a variety of data types. The features of MongoDB are wide-ranging and support ad-hoc queries, indexing, load balancing, aggregation, and server-side JS execution. With short declarative code, users of the MongoDB Query API can execute complicated analytical operations and even perform in-depth document queries. (Botelho 2020; MongoDB Tutorial 2022; Advantages of MongoDB n.d.)

5.2 Portal Elements

Content, design, personalization, and evaluation are the four essential components of a web portal. A web portal's organization, credibility, usefulness, and integration are all linked to the content. Focus areas for the design include architecture, usability, graphic design, and technical integrity. Navigational ease, the availability of content, and interface customization in response to user preferences and needs are all examples of personalization. Evaluation reveals the advantages and restrictions of the portal services. (Sampson 2007, 382-383)

The UI techniques for the web portal personalization framework are separate from the database storage and application logic by the framework's multi-level design. The three levels combine business logic, presentation, and data access. The data access layer communicates and saves data between the database and application. The business logic layer contains the parts needed to implement the application. The system's UI is the responsibility of the presentation layer. (Doulgeraki et al. 2007, 12-14)

To improve the usability and accessibility of a web portal, user profiling takes place in the presentation layer illustrated in Figure 10. Age, interests, and portal usage are all listed on a user's profile. The data is essential when modifying the portal to suit the tastes and demands of a user. (Doulgeraki et al. 2007, 15-16)

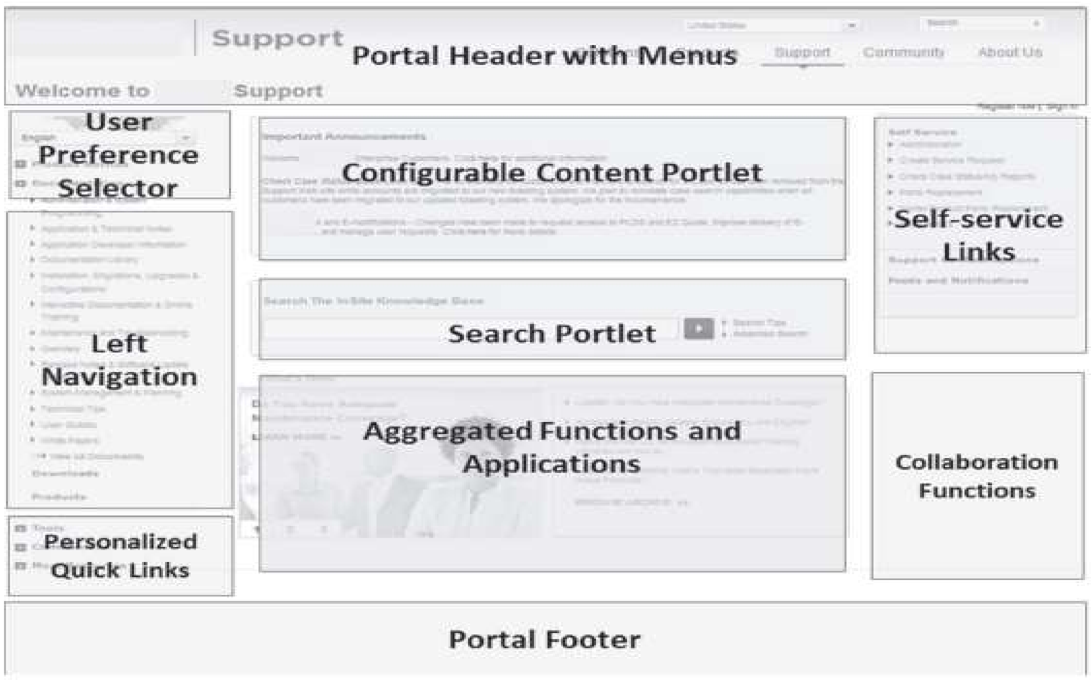


Figure 10. Web portal components. Source: Shivakumar 2016, 10.

A portal should also have content aggregation, search engines and tools, application integration, personalization, data capture, data mining, security and access components, and connectivity to other entities (Wojtkowski & Major 2005, 22).

The environment for a web portal should contain all the components required to support its operation (Fulmer 2007, 165). The portal environment includes a repository, gadgets, categorization tools, filters, and an index. Data organization requires a taxonomy, classification scheme, or folder structure. Software, web services, and content management systems integrate the applications. (Abrahams 2007, 163)

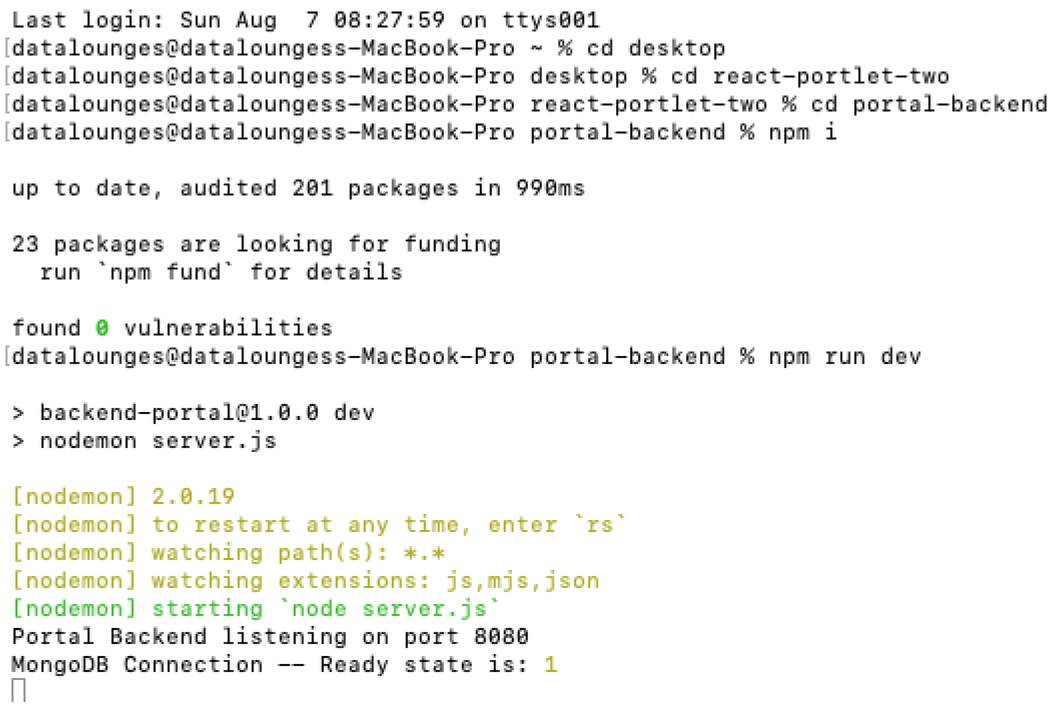
5.3 User Interface Architecture

The architecture for a web portal must take component-oriented philosophy into account. The internal architecture of a software system establishes the logic of the application specifications. The architectural design of UI, which displays the logical flow of the software, supports their look, feel, and usability aspects. (Qian et al. 2010, 299, 302, 312) Enhancing user interface clarity and appeal is a successful user attraction tactic (Agosti et al. 2010, 233).

Many programming languages have built-in elements and layout managers for creating UIs, including Java, Visual Basic, and others. It is challenging to arrange these components into a user interface. The mapping of components into engineering practice, the types required, and the correct arrangement within the domain and with goals must all be understood. (Qian et al. 2010, 302)

5.4 Web Portal Technologies

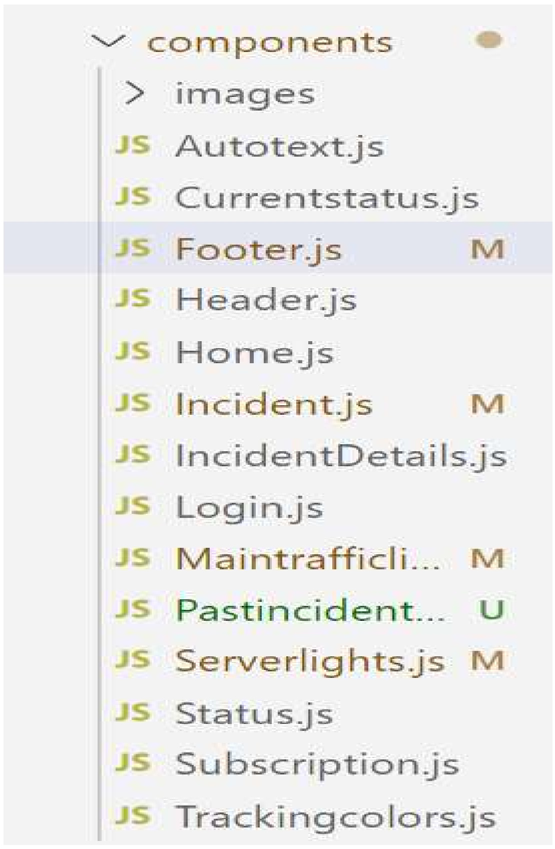
The UI of the web portal was created on the front-end using the React application and the open-source access management software Keycloak. With the integration of Express.js the functionality of the web portal was made possible in the back-end by an API, and MongoDB to store the data shown in Figure 11.



```
Last login: Sun Aug  7 08:27:59 on ttys001
[datalounges@dataloungess-MacBook-Pro ~ % cd desktop
[datalounges@dataloungess-MacBook-Pro desktop % cd react-portlet-two
[datalounges@dataloungess-MacBook-Pro react-portlet-two % cd portal-backend
[datalounges@dataloungess-MacBook-Pro portal-backend % npm i

up to date, audited 201 packages in 990ms

23 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
[datalounges@dataloungess-MacBook-Pro portal-backend % npm run dev

> backend-portal@1.0.0 dev
> nodemon server.js

[nodemon] 2.0.19
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Portal Backend listening on port 8080
MongoDB Connection -- Ready state is: 1
```

Figure 11. The back-end application of the web portal running on local host: 8080

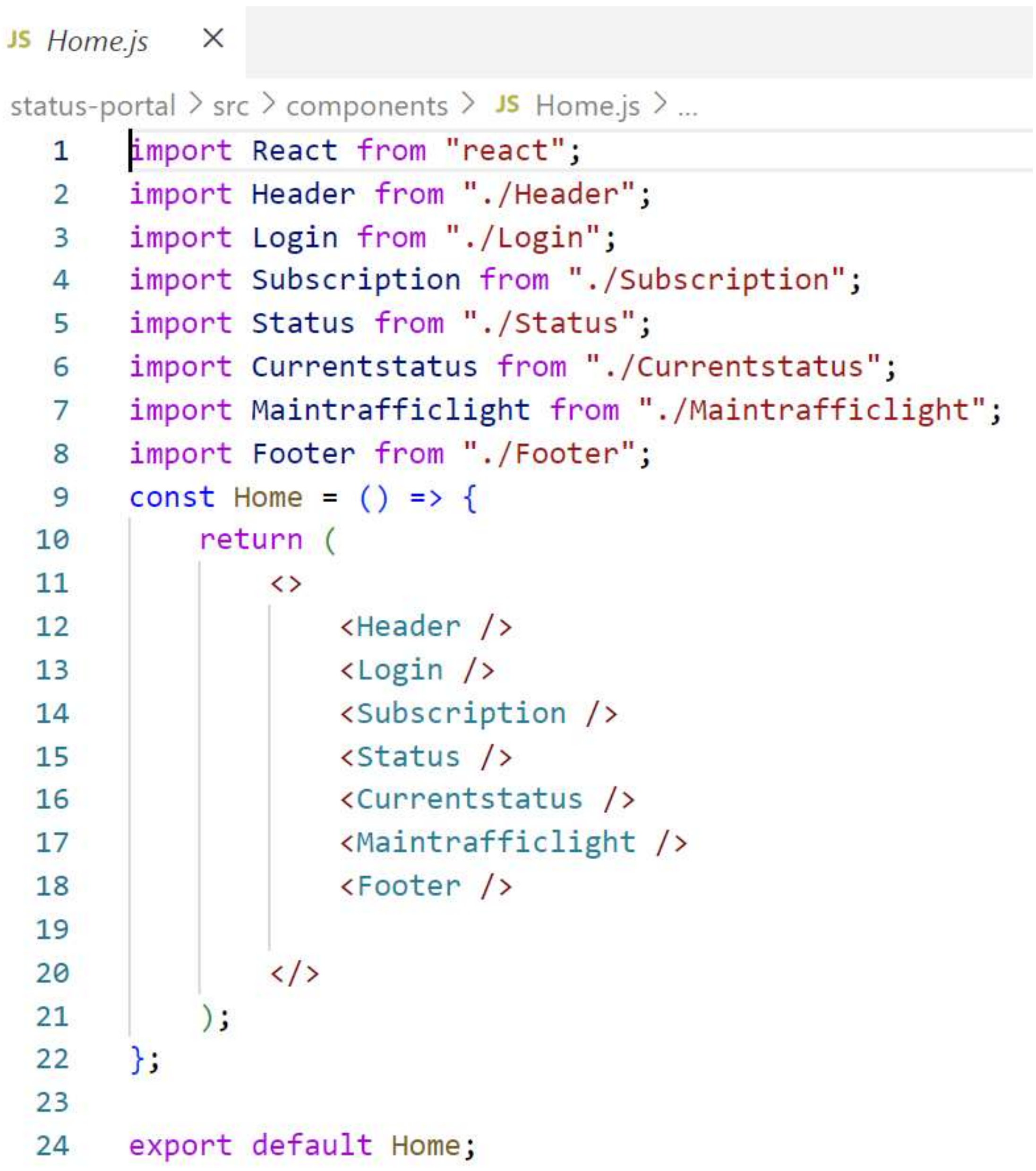
Several components were required using the React application illustrated in Figure12 to fulfil user requirements for the web portal mentioned in this study.



```
components
  > images
  JS Autotext.js
  JS Currentstatus.js
  JS Footer.js            M
  JS Header.js
  JS Home.js
  JS Incident.js          M
  JS IncidentDetails.js
  JS Login.js
  JS Maintrafficli...     M
  JS Pastincident...      U
  JS Serverlights.js      M
  JS Status.js
  JS Subscription.js
  JS Trackingcolors.js
```

Figure 12. Components of the web portal application

HTML and CSS were employed to develop the web portal (see Appendix 2) before dividing the UI into hierarchy components in React. The home component illustrated in Figure 13 imports the primary UI components of the web portal.


JS Home.js ×

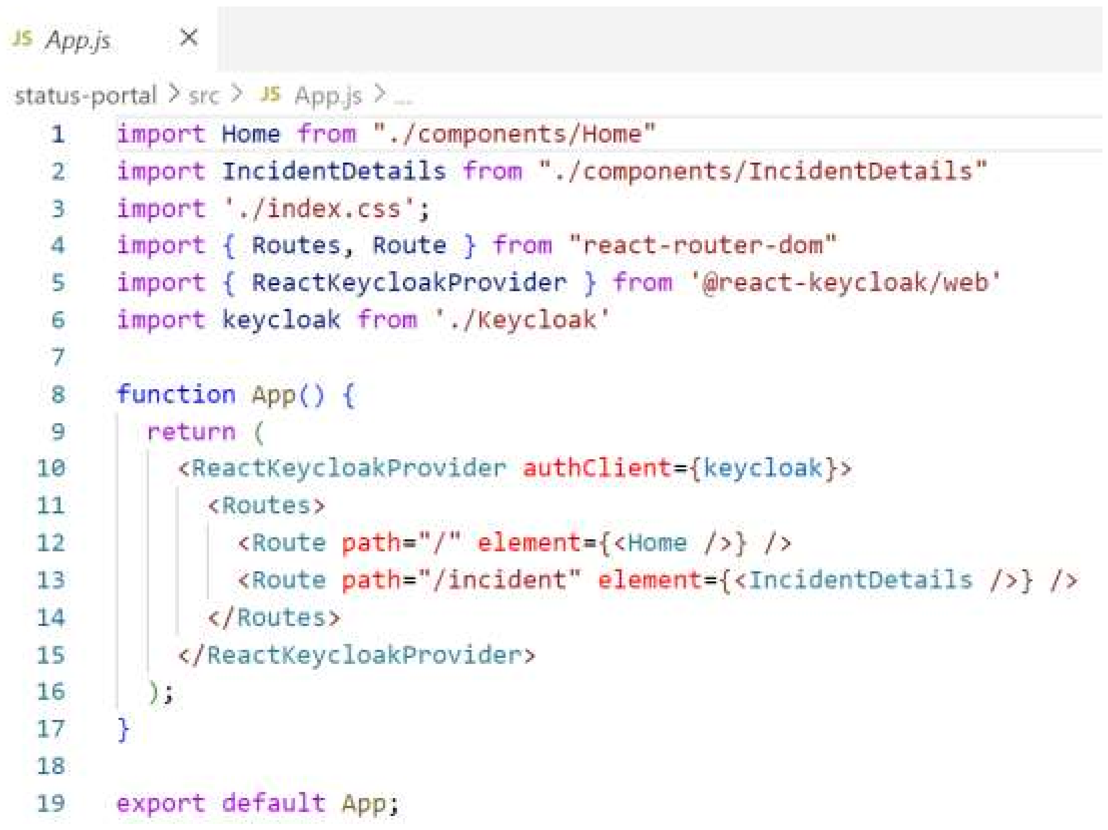
status-portal > src > components > JS Home.js > ...

```
import React from "react";
import Header from "./Header";
import Login from "./Login";
import Subscription from "./Subscription";
import Status from "./Status";
import Currentstatus from "./Currentstatus";
import Maintrafficlight from "./Maintrafficlight";
import Footer from "./Footer";
const Home = () => {
    return (
        <>
            <Header />
            <Login />
            <Subscription />
            <Status />
            <Currentstatus />
            <Maintrafficlight />
            <Footer />

        </>
    );
};

export default Home;
```

Figure 13. The home component of the web portal application

The header component displays the company logo and the login and subscribe to updates buttons. The login component prevents unauthorized access. Only users who subscribe through e-mail will receive notifications and updates on the status of the servers.

Announcements are published using the current status component. The past incident component allows users to view the history of occurrences. A separate incident detail component enabled the viewing of past events. The main traffic light component shows whether the servers are working or if there are any issues. The server light component reflects the metrics of the server. The tracking colors component monitors the servers to determine whether they are operational or experiencing degraded performance, outage, or under maintenance. The react-router-dom was imported and installed to enable views and navigation of the web portal seen in Figure 14.



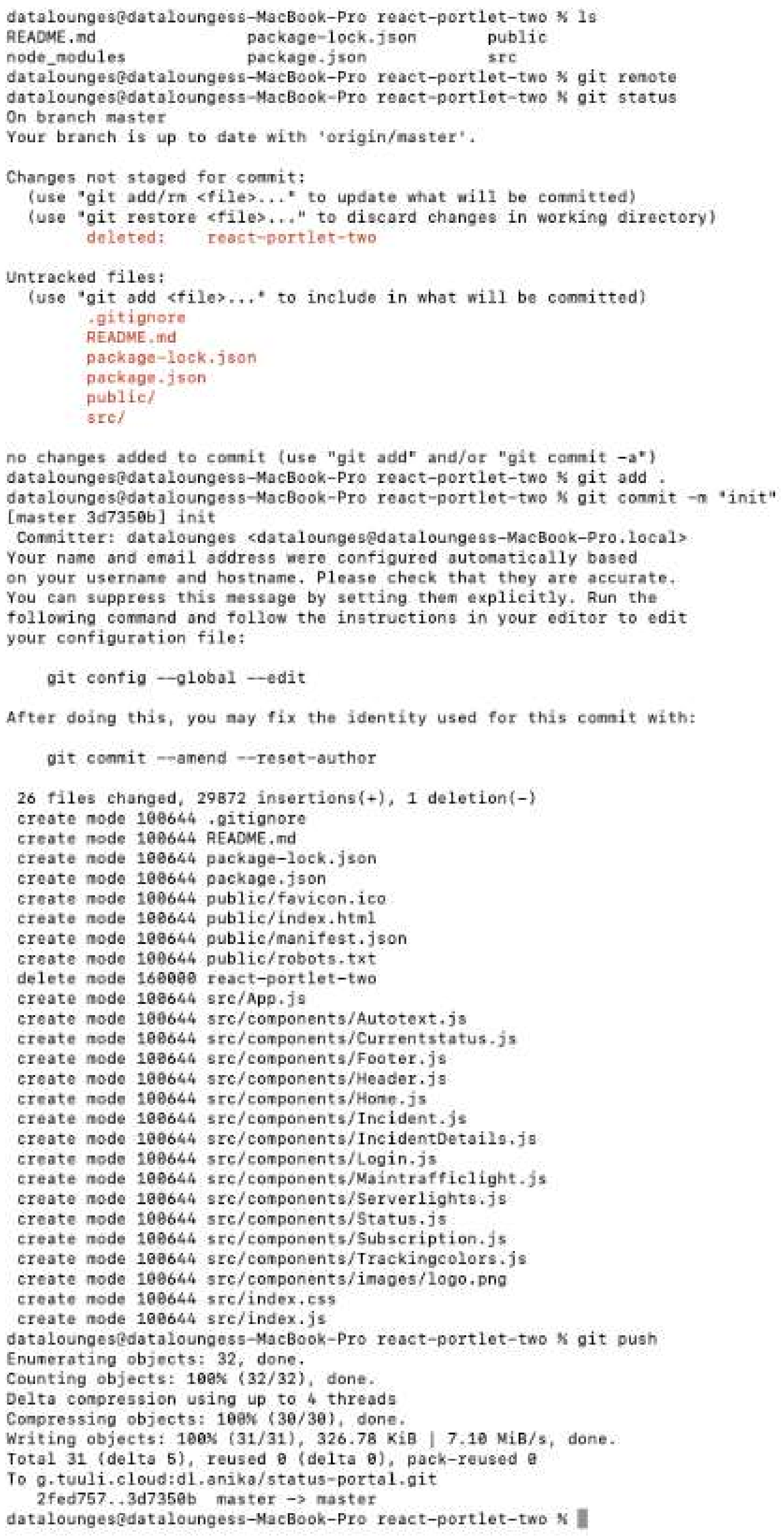
App.js

status-portal > src > App.js > ...

```
import Home from "./components/Home"
import IncidentDetails from "./components/IncidentDetails"
import './index.css';
import { Routes, Route } from "react-router-dom"
import { ReactKeycloakProvider } from '@react-keycloak/web'
import keycloak from './Keycloak'

function App() {
  return (
    <ReactKeycloakProvider authClient={keycloak}>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/incident" element={<IncidentDetails />} />
      </Routes>
    </ReactKeycloakProvider>
  );
}

export default App;
```

Figure 14. The App.js web portal component

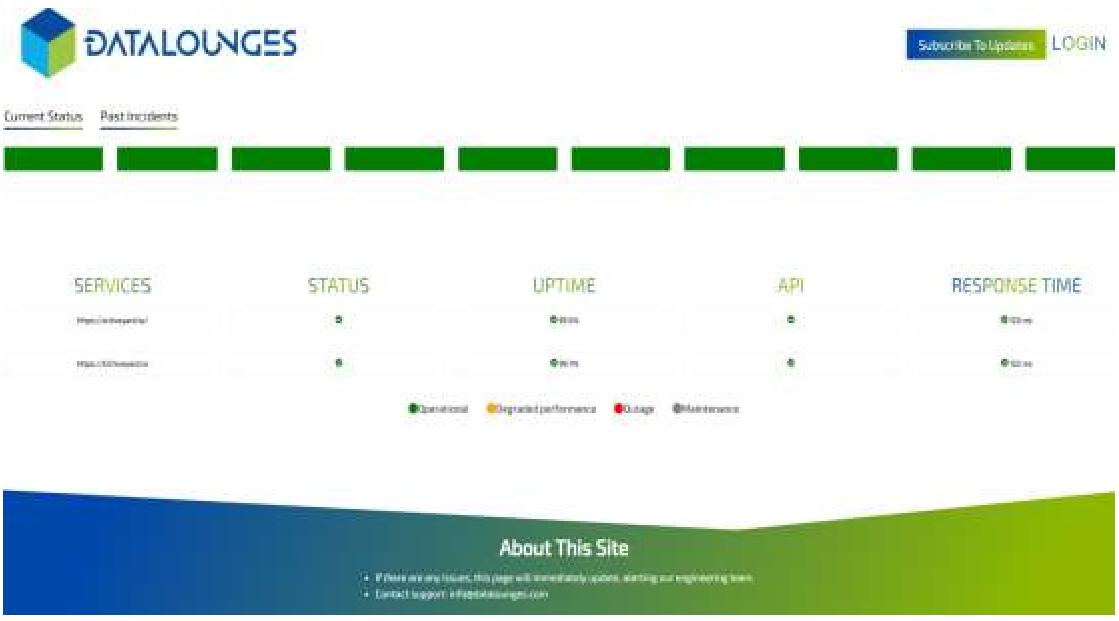
In implementing the DevOps practices for this study, a GitLab repository enabled version control for developers to add or modify the business requirements for the web portal seen in Figure 15. GitLab is a platform for joint software development and an open-source code repository for significant DevOps initiatives, providing a space for code management and tools for issue tracking and CI/CD. GitLab enables teams to produce more secure and compliant apps while reducing development costs, accelerating time to market, and improving cycle times from idea to production. (GitLab 2022; The One DevOps Platform 2022; GitLab n.d.) The GitLab repository supported the Agile approach and provided a platform for developers to track changes made to the web portal until all requirements were satisfied. The code for the web portal was committed regularly into the GitLab repository as part of the CI process. Through the automatic CD procedure, the code modifications were made available.



```
datalounges@dataloungess-MacBook-Pro react-portlet-two % ls
README.md               package-lock.json       public
node_modules            package.json            src
datalounges@dataloungess-MacBook-Pro react-portlet-two % git remote
datalounges@dataloungess-MacBook-Pro react-portlet-two % git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    react-portlet-two

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        README.md
        package-lock.json
        package.json
        public/
        src/

no changes added to commit (use "git add" and/or "git commit -a")
datalounges@dataloungess-MacBook-Pro react-portlet-two % git add .
datalounges@dataloungess-MacBook-Pro react-portlet-two % git commit -m "init"
[master 3d7350b] init
 Committer: datalounges <datalounges@dataloungess-MacBook-Pro.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 26 files changed, 29872 insertions(+), 1 deletion(-)
 create mode 100644 .gitignore
 create mode 100644 README.md
 create mode 100644 package-lock.json
 create mode 100644 package.json
 create mode 100644 public/favicon.ico
 create mode 100644 public/index.html
 create mode 100644 public/manifest.json
 create mode 100644 public/robots.txt
 delete mode 160000 react-portlet-two
 create mode 100644 src/App.js
 create mode 100644 src/components/Autotext.js
 create mode 100644 src/components/Currentstatus.js
 create mode 100644 src/components/Footer.js
 create mode 100644 src/components/Header.js
 create mode 100644 src/components/Home.js
 create mode 100644 src/components/Incident.js
 create mode 100644 src/components/IncidentDetails.js
 create mode 100644 src/components/Login.js
 create mode 100644 src/components/Maintrafficlight.js
 create mode 100644 src/components/Serverlights.js
 create mode 100644 src/components/Status.js
 create mode 100644 src/components/Subscription.js
 create mode 100644 src/components/Trackingcolors.js
 create mode 100644 src/components/images/logo.png
 create mode 100644 src/index.css
 create mode 100644 src/index.js
datalounges@dataloungess-MacBook-Pro react-portlet-two % git push
Enumerating objects: 32, done.
Counting objects: 100% (32/32), done.
Delta compression using up to 4 threads
Compressing objects: 100% (30/30), done.
Writing objects: 100% (31/31), 326.78 KiB | 7.10 MiB/s, done.
Total 31 (delta 5), reused 0 (delta 0), pack-reused 0
To g.tuuli.cloud:dl.anika/status-portal.git
   2fed757..3d7350b  master -> master
datalounges@dataloungess-MacBook-Pro react-portlet-two % ▌
```

Figure 15. The web portal components pushed to the GitLab repository

Clients can access the status page of Datalounges Oy services through the web portal illustrated in Figure 16.



SERVICES	STATUS	UPTIME	API	RESPONSE TIME
[illegible]	●	● [illegible]	●	● [illegible]
[illegible]	●	● [illegible]	●	● [illegible]

● Operational ● Degraded performance ● Outage ● Maintenance

About This Site

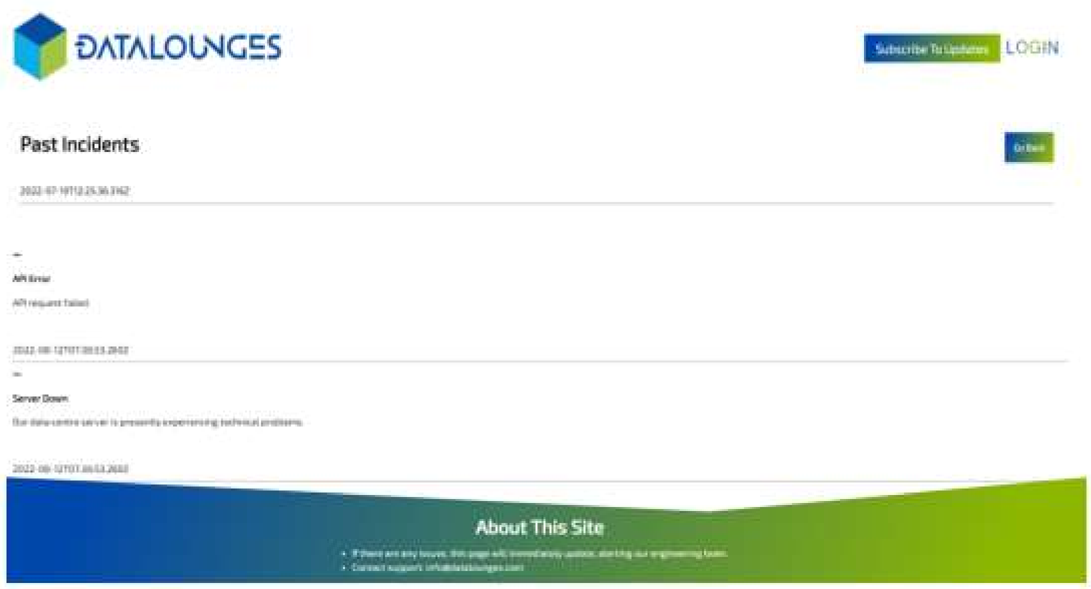
- If there are any issues, this page will immediately update, alerting our engineering team
- Contact support: info@datalounges.com

Figure 16. The status page of the web portal application

After clicking the login button, clients must enter their e-mail address and password to view the status page. Clients can choose to subscribe to alerts anytime an incident occurs.

The services of Datalounges Oy are conditionally monitored by a color-coded traffic light, with green denoting that all services are operational, red indicating a power outage, orange signifies performance degradation, and gray showing maintenance services. Clients can view the metrics showing the status, uptime, API, and response time of Datalounges Oy services.

Administrators with assigned roles can post unexpected incidents involving service interruptions using the current status link. The past incident link allows clients to view a history of service occurrences seen in Figure 17. The sixth chapter discusses the deployment process of the web portal.



Past Incidents

2022-07-19T12:25:36.316Z

API Error

API request failed

2022-08-12T07:00:53.260Z

Server Down

Our data centre server is presently experiencing technical problems.

2022-08-12T07:00:53.260Z

About This Site

- If there are any issues, this page will immediately update, alerting our engineering team.
- Contact support: info@datalounges.com

Figure 17. The incident page of the web portal application

6 DEPLOYMENT PROCESS

6.1 Deployment Planning

The success or failure of the web portal's release depends on a well-defined deployment process and its implementation. The planning, development, testing, staging, and production phases all play a part in the deployment process, which differs from project to project. Stages are chosen based on the unique requirements of the project. (Block et al. 2003, 429-430)

The implementation phases and timeline must be complied with to meet the requirements. A thorough planning procedure is also essential for a successful deployment. Careful planning ensures a seamless transition and timely troubleshooting if problems develop throughout the deployment process. The goals, scope, requirements, roles, and responsibilities outlined by developers should be the focus of the deployment plan. It is encouraged to implement a pilot program before starting the deployment. Before end users may access the system, deployment testing is also crucial. Finally, the deployment plan must include a schedule for employee training. (Deployment Planning Guide 2022)

Block et al. (2003, 429–430) emphasized the need for source control during the deployment procedure. The integrity of the source code and other resources should be examined and evaluated by the source control.

6.2 Container Orchestration and Management

A feasible method for creating edge computing infrastructures is containerization technology (Hwang et al. 2021; Pan & McElhannon 2017; Menouer 2021, 4268). With the help of container orchestration, it is easy to automate most of the work, enabling DevOps to manage operational complexity. The agile and quick-moving nature of DevOps teams makes it appropriate and suitable. (Parikh & Johri 2022)

Kubernetes, sometimes known as K8s has numerous capabilities to facilitate container deployment and dynamic resource management and also offers several deployment choices (Nguyen et al. 2022, 1) The multiple characteristics of Kubernetes enable the quick deployment of applications without the need for time-consuming configuration and installation procedures (Nguyen et al. 2022, 1; Felter et al. 2015; Xu et al. 2021; Bonomi et al. 2012).

Containerization makes it possible to distribute and upgrade software without experiencing any downtime, which helps achieve these objectives (Learn Kubernetes Basics 2022). Figure 18 shows the client-server architecture of Kubernetes.

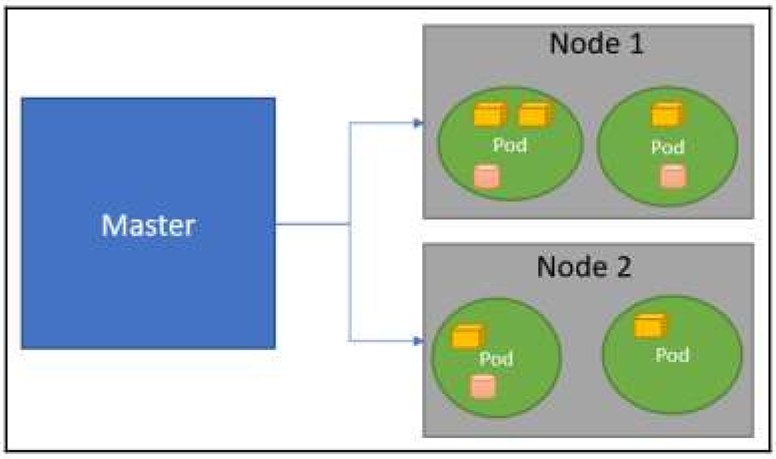
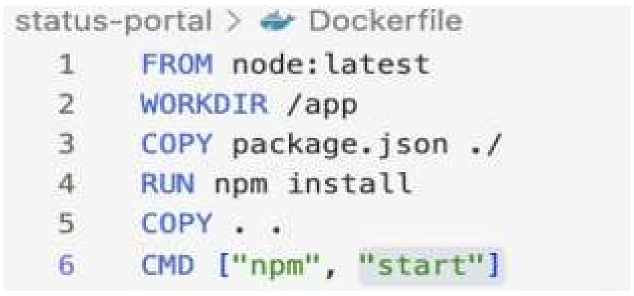


Figure 18. The architecture of Kubernetes. Source: Krief 2019, 248.

Master and worker nodes make up the Kubernetes cluster (Bentaleb et al. 2022, 1159). The master component of the cluster consists of nodes that stand in for the slave servers. To host containers and volumes, each node stores virtual component pods. (Krief 2019, 248) The master node serves as the foundation of the orchestration system, and the worker nodes run pods by providing specific services. Compared to virtual machines (VMs), containers are lighter, more portable, and use fewer resources.

The first step in the web portal deployment process was the installation of Docker software. The creation of a Dockerfile seen in Figure 19 was necessary to run each of the commands needed to build the image.

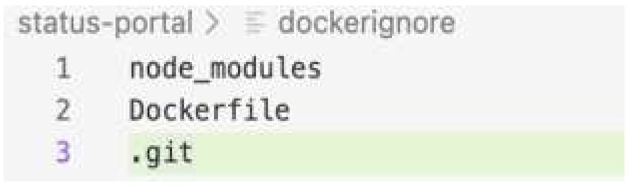


```
FROM node:latest
WORKDIR /app
COPY package.json ./
RUN npm install
COPY . .
CMD ["npm", "start"]
```

Figure 19. The Dockerfile created

The FROM command specified the use of the base image to launch a new build process. The WORKDIR command defined the type of app to be built and the working directory for the following commands. The COPY command duplicated the package.json file to the /app working directory. The RUN command indicated that the application required the installation of the node package manager (npm) to execute. The COPY . . command instructed for the copying of all files to the working directory of /app. The CMD command specified that npm start was required to view the web portal application.

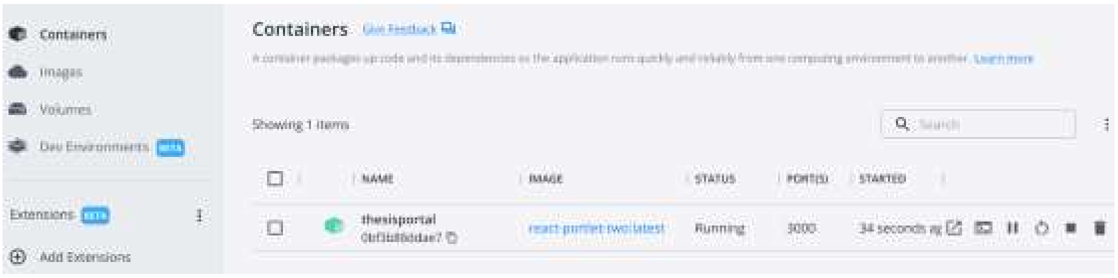
The Dockerignore file prevented node_modules, .git, and the Dockerfile from copying to the /app directory during the building process of the container illustrated in Figure 20.



```
node_modules
Dockerfile
.git
```

Figure 20. The Dockerignore created

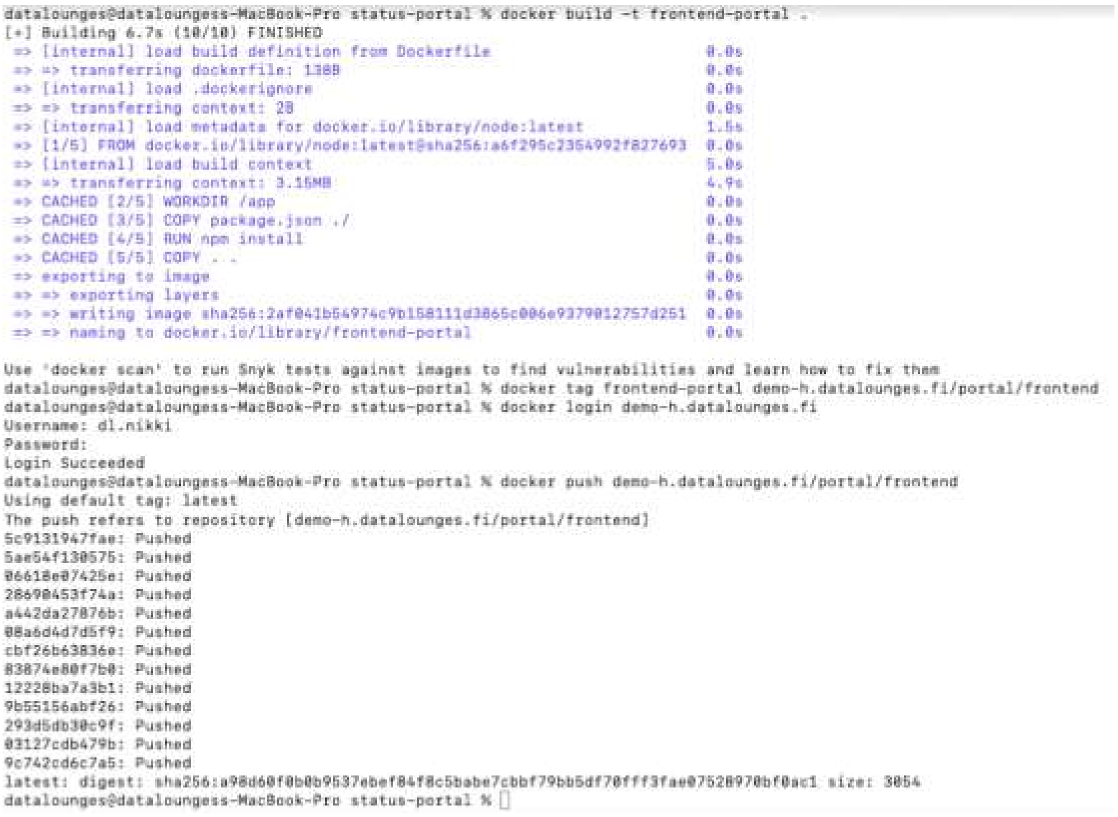
The command docker build -t frontend-portal . started the building process of the container. The command docker run --name thesisportal -d -p 3000:3000 frontend:latest specified the port to run the Docker container seen in Figure 21.



NAME	IMAGE	STATUS	PORT(S)	STARTED
thesisportal 0bf3b8bddae7	react-portlet-two:latest	Running	3000	34 seconds ag

Figure 21. The web portal container running on 3000

The command docker tag frontend-portal demo-h.datalounges.fi/portal/frontend tagged and pushed the image to the internal Harbor repository of Datalounges Oy shown in Figure 22. Authorization was required to complete the process.



```
datalounges@dataloungess-MacBook-Pro status-portal % docker build -t frontend-portal .
[+] Building 6.7s (10/10) FINISHED
 => [internal] load build definition from Dockerfile                          0.0s
 => => transferring dockerfile: 138B                                          0.0s
 => [internal] load .dockerignore                                             0.0s
 => => transferring context: 2B                                               0.0s
 => [internal] load metadata for docker.io/library/node:latest                1.5s
 => [1/5] FROM docker.io/library/node:latest@sha256:a6f295c2354992f827693     0.0s
 => [internal] load build context                                             5.0s
 => => transferring context: 3.15MB                                           4.9s
 => CACHED [2/5] WORKDIR /app                                                 0.0s
 => CACHED [3/5] COPY package.json ./                                         0.0s
 => CACHED [4/5] RUN npm install                                              0.0s
 => CACHED [5/5] COPY . .                                                     0.0s
 => exporting to image                                                        0.0s
 => => exporting layers                                                       0.0s
 => => writing image sha256:2af041b54974c9b158111d3865c006e93790127 57d251    0.0s
 => => naming to docker.io/library/frontend-portal                            0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
datalounges@dataloungess-MacBook-Pro status-portal % docker tag frontend-portal demo-h.datalounges.fi/portal/frontend
datalounges@dataloungess-MacBook-Pro status-portal % docker login demo-h.datalounges.fi
Username: dl.nikki
Password:
Login Succeeded
datalounges@dataloungess-MacBook-Pro status-portal % docker push demo-h.datalounges.fi/portal/frontend
Using default tag: latest
The push refers to repository [demo-h.datalounges.fi/portal/frontend]
5c9131947fae: Pushed
5ae54f130575: Pushed
06618e07425e: Pushed
28690453f74a: Pushed
a442da27876b: Pushed
08a6d4d7d5f9: Pushed
cbf26b63836e: Pushed
83874e80f7b0: Pushed
12228ba7a3b1: Pushed
9b55156abf26: Pushed
293d5db30c9f: Pushed
03127cdb479b: Pushed
9c742cd6c7a5: Pushed
latest: digest: sha256:a98d60f0b0b9537ebef84f8c5babe7cbbf79bb5df70fff3fae07528970bf0ac1 size: 3054
datalounges@dataloungess-MacBook-Pro status-portal % 
```

Figure 22. The image pushed to the Harbor repository of Datalounges Oy

The creation of a portal-service.yaml file was necessary listing the app, service type, name, protocol (Transmission Control Protocol), and targeted port seen in Figure 23.

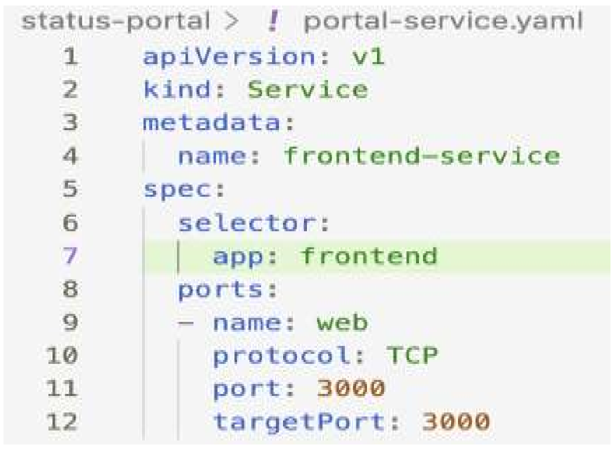

```yaml
status-portal > ! portal-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  selector:
    app: frontend
  ports:
  - name: web
    protocol: TCP
    port: 3000
    targetPort: 3000
```

Figure 23. The portal-service.yaml file created

The portal-deployment.yaml file specified the type of Kubernetes object, the number of required pods, the name and image of the application, and the particular port shown in Figure 24.

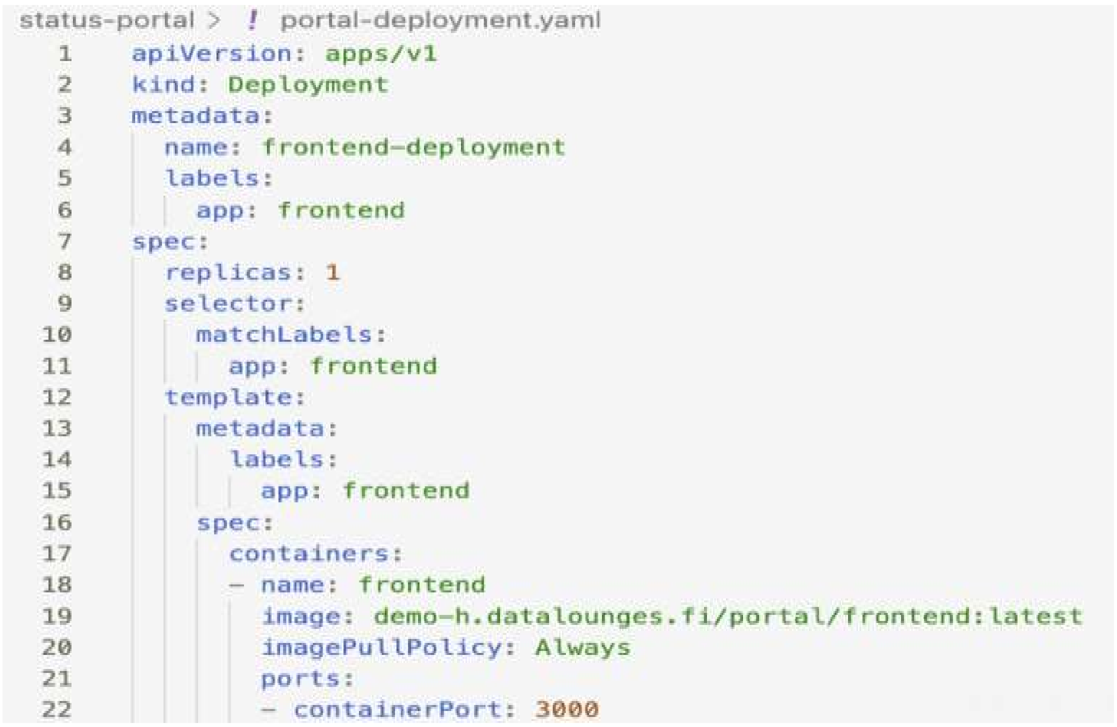

```yaml
status-portal > ! portal-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
  labels:
    app: frontend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: frontend
        image: demo-h.datalounges.fi/portal/frontend:latest
        imagePullPolicy: Always
        ports:
        - containerPort: 3000
```

Figure 24. The portal-deployment.yaml file created

The LENS/Kubernetes open-source platform enabled the service and deployment of the pod. The web portal application is active and operational, with the front-end listening on port 3000 and the back-end listening on port 8080 illustrated in Figure 25. The upcoming final chapter evaluates the results and summarizes the study's findings.

Figure 25. The active web portal application

7 RESULTS AND CONCLUSION

Secondary data provided the theoretical framework to build the web portal to achieve this study's objective. Before the development of the web portal, 26 peer-reviewed journals and 26 academic literature books served as a source of secondary data on the study subject to answer the research questions outlined in the introduction. Other electronic secondary sources provided knowledge for the practical implementation of the study.

A significant limitation of this study's secondary data acquisition was the scarcity of up-to-date information on the design and development of a modern web portal. Due to the limited availability of current secondary data, a comparative and critical analysis of web portal design and development was not probable. Overall the study's results revealed the steps required to build a functional, user-friendly web portal.

The study objectives provided the foundation to answer the first research question: What are the design principles of a web portal? The secondary research sources examined in this study demonstrated the importance of user experience and satisfaction as critical components of web portal design. Modern technologies, programming languages, and business requirements all play a role in creating an effective UI.

Based on the given business requirements, HTML, CSS, and React.js created the client-side of the web portal application in this study. The web portal design includes an expanded view of the system metrics of Datalounges Oy to meet the user requirements. To improve brand recognition and visual experience, the brand colors of Datalounges Oy were incorporated into the web design. Integrated into the web portal design are user preferences allowing clients to request e-mail alerts via the subscribe to updates button.

The web portal design also prioritized usability, allowing users to navigate and access information effectively.

Portal elements and infrastructure was the subject of the second research question: What are the components and architecture of the web portal? The designed web portal is responsive and built to work on all devices to meet adaptability and flexibility. The self-service capabilities of the web portal enable users to view Datalounges Oy incidents and system metrics without contacting the company directly. Open-source Keycloak software secures the web portal's authentication and login credentials.

This study used a three-tier architectural framework to develop the functionality of the web portal application on the front-end (client), back-end (application), and database (data). In the front-end of the web portal application, React.js components created dynamic user interactivity. The Express.js server-side framework runs inside a Node.js server in the application tier, where the front-end makes HTTP requests to the back-end of the web portal application based on the user request. In the database tier, MongoDB is used to fetch and post data in the web portal application via an API in Node.js and Express.js running on port 8080.

The third research question addressed the implementation stages: What are the deployment methods for a web portal? A containerized environment enabled the deployment of the web portal application. The deployment was quick and successful, requiring the installation of Docker software and building an image to spin the web portal container. It is justifiable to conclude that Docker and containers are critical tools to manage web deployment in a DevOps pipeline.

The fourth research question analyzed: What testing methods does a web portal employ? Before being launched to clients, the web portal application is subjected to continuous and iterative testing internally. Continuous functionality, usability, and security tests are ongoing.

Overall, the study's result is a functional and user-oriented web portal designed and deployed to meet Datalounges Oy's specific business requirements. The concept of the web portal application also integrated the need to meet additional business requirements in the future.

REFERENCES

Abrahams, B. 2007. Developing Semantic Portals. In Tatnall, Arthur (ed.) Encyclopedia of Portal Technologies and Applications. Hershey: Information Science Reference, 235-243.

Adams, M. J. 2021. DevOps Leadership: Steps for the Introduction and Implementation of DevOps. Norderstedt: BoD.

Advantages of MongoDB. n.d. MongoDB. Web site. Available at: https://www.mongodb.com/advantages-of-mongodb [Accessed 14 August 2022].

Agile vs Waterfall: Difference between Agile and Waterfall. 2020. IntelliPaat. Video. Available at: https://www.youtube.com/watch?v=aKf0Qkp14qQ [Accessed 13 May 2022].

Agile vs Waterfall: What`s the difference? 2021. Australian Institute of Project Management. Web site. Available at: https://www.aipm.com.au/blog/agile-vs-waterfall-whats-the-difference [Accessed 12 May 2022].

Agosti, M., Crivellari, F., Di Nunzio, G. M. & Gabrielli, S. 2010. Understanding user requirements and preferences for a digital library Web portal. *International Journal on Digital Libraries*, 11(4), 225-238. E-journal. Available at: https://link.springer.com/article/10.1007/s00799-011-0075-7 [Accessed 13 May 2022].

Ali-Shahid, M. M. & Sulaiman, S. 2016. A case study on reliability and usability testing of a Web portal. 9th Malaysian Software Engineering Conference. IEEE Xplore. WWW document. Available at: https://ieeexplore.ieee.org/document/7475191/figures#figures [Accessed 20 May 2022].

Arantes, F. L. 2013. Requirements Engineering of a Web Portal Using Organizational Semiotics Artifacts and Participatory Practices. *International Journal of Computer Science & Information Technology*, 5(2), 131-146. E-journal. Available at: https://www.academia.edu/21674242/REQUIREMENTS_ENGINEERING_OF_A_WEB_PORTAL_USING_ORGANIZATIONAL_SEMIOTICS_ARTIFACTS_AND_PARTICIPATORY_PRACTICES [Accessed 20 May 2022].

Barger, R. 2021. Why You Should Use React Components Instead of HTML? FreeCodeCamp. WWW document. Available at: https://www.freecodecamp.org/news/intro-to-react-components/ [Accessed 28 July 2022].

Ben-Natan, R., Gornitsky, R., Sasson, O. & Hanis, T. 2004. Mastering IBM WebSphere Portal. Indianapolis: Wiley Publishing.

Bentaleb, O., Belloum, A. S. Z., Sebaa, A. & El-Maouhab, A. 2022. Containerization technologies: taxonomies, applications and challenges. *The Journal of Supercomputing,* 78(1), 1144-1181. E-journal. Available at: https://www.researchgate.net/publication/352229587_Containerization_technologies_taxonomies_applications_and_challenges [Accessed 28 May 2022].

Besson, J., Lupeikiene, A. & Medvedev, V. 2012. Comparing Real and Intended System Usages: A Case for Web Portal. *Informatica*, 23(2), 191-201.

E-journal. Available at: https://www.researchgate.net/publication/289079064_Comparing_Real_and_Intended_System_Usages_A_Case_for_Web_Portal [Accessed 4 June 2022].

Best Practices for DevOps in the Cloud. 2019. eWeek, 1. WWW document. Available at: https://web.p.ebscohost.com/ehost/detail/detail?vid=3&sid=c3430621-711d-454f-9dc8-2cf77843b90e%40redis&bdata=JkF1dGhUeXBlPXNoaWlmc2l0ZT1laG9zdC1saXZl#AN=138198922&db=a9h [Accessed 10 July 2022].

Bhojaraju, G. 2007. KM Cyberary is a Gateway to Knowledge Resources. In Tatnall, A. (ed.) Encyclopedia of Portal Technologies and Applications. Hershey: Information Science Reference, 522-526.

Block, H., Castle, R. & Hritz, D. 2003. Creating Web Portals with BEA WebLogic. Berkeley, CA: Apress Media.

Bonomi, F., Milito, R., Zhu, J. & Addepalli, S. 2012. Fog Computing and Its Role in the Internet of Things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC ’12, Helsinki, Finland, 13–16. WWW document. Available at: https://conferences.sigcomm.org/sigcomm/2012/paper/mcc/p13.pdf [Accessed 8 July 2022].

Botelho, B. 2020. MongoDB. TechTarget. Web article. Available at: https://www.techtarget.com/searchdatamanagement/definition/MongoDB [Accessed 14 August 2022].

Braunton, A. 2018. Hands-On DevOps with Vagrant: Implement end-to-end DevOps and infrastructure management using Vagrant. Birmingham: Packt.

Bringula, R. P. 2016. Factors Affecting Web Portal Information Services Usability: A Canonical Correlation Analysis. *International Journal of Human-Computer Interaction,* 32(10), 814-826. E-journal. Available at: https://www.tandfonline.com/doi/abs/10.1080/10447318.2016.1199180 [Accessed 10 June 2022].

Bringula, R. P. & Basa, R. S. 2011. Factors Affecting Faculty Web Portal Usability. *Educational Technology & Society*, 14(4), 253-265. E-journal. Available at: https://www.researchgate.net/publication/259703194_Factors_Affecting_Faculty_Web_Portal_Usability [Accessed 15 June 2022].

Cavero-Baptista, L. n.d. Web Application Testing: The Basic of Web App Test Automation. Leapwork. E-article. Available at:https://www.leapwork.com/blog/web-application-testing-the-basics-of-web-app-test-automation [Accessed 4 June 2022].

Che, X. & Ip, B. 2017. Social Networks in China. Hull: Chandos Publishing.

Cheng, C. P., Law, K. H. & Bjornsson, H. 2008. A distributed portal-based platform for construction supply chain interoperability. In Zarli, A. & Scherer, R. (eds.) eWork and eBusiness in Architecture, Engineering and Construction. Boca Raton, FL: CRC Press, 161-170.

Chishti, M. A., Qureshi, S., Mir, A. H. & Haseeb, S. 2008. Deploying Open Source Web Portal as a Tool for Knowledge Sharing and Collaboration.

International Journal of Computing and ICT Research, 10-18. E-journal. Available at: https://www.academia.edu/20471856/DEPLOYING_AN_OPEN_SOURCE_WEB_PORTAL_AS_A_TOOL_FOR_KNOWLEDGE_SHARING_AND_COLLABORATION [Accessed 20 June 2022].

Choi, D. 2020. Full-Stack React, TypeScript, and Node: Build cloud-ready web applications using React 17 with Hooks and GraphQL. Birmingham: Packt Publishing.

Chowdhury, A. R. 2019. DevOps Best Practices: A Complete Guide. Stackify. E-article. Available at: https://stackify.com/devops-best-practices-a-complete-guide/ [Accessed 15 May 2022].

Conte, T., Massollar, J., Mendes, E., and Travassos, G. H. 2007. Usability Evaluation Based on Web Design Perspectives Paper presented at the First International Symposium on Empirical Software Engineering and Measurement. IEEE, 146-155. E-article. Available at: https://www.academia.edu/33707843/Usability_Evaluation_Based_on_Web_Design_Perspectives [Accessed 27 June 2022].

Costagliola, G., Ferrucci, F. & Fuccella, V. 2007. Java Portals and Java Portlet Specification and API. In Tatnall, A. (ed.) Encyclopedia of Portal Technologies and Applications. Hershey: Information Science Reference, 516-521.

Costagliola, G., Ferrucci, F., Fuccella, V. & Zurolo, L. 2008. Logging and Analysing User`s Interactions in Web Portals. In Filipe, J. & Cordeiro, J. (eds.) Web Information Systems and Technologies. Third International Conference, WEBIST 2007, Barcelona. Heidelberg: Springer, 213-229.

Datalounges. 2022a. Web site. Available at: https://mobile.twitter.com/datalounges [Accessed 29 April 2022].

Datalounges. 2022b. What we offer. Web site. Available at: https://www.datalounges.com/what-we-offer/ [Accessed 29 April 2022].

Deployment Planning Guide. 2022. ForgeRock. WWW document. Available at: https://backstage.forgerock.com/docs/am/5.5/deployment-planning-guide/

Dhaduk, H. 2022. 11 DevOps Best Practices to Follow for a Successful DevOps Journey. Simform. E-article. Available at: https://www.simform.com/blog/devops-best-practices/ [Accessed 15 May 2022].

Doulgeraki, C., Antona, M., Balafa, K. & Stephanidis, C. 2007. Accessible Personalized Portals. In Tatnall, Arthur (ed.) Encyclopedia of Portal Technologies and Applications. Hershey: Information Science Reference, 12-19.

Eisenman, B. 2016. Learning React Native: Building Native Mobile Apps with Javascript. Sebastopol, CA: O`Reilly Media Inc.

Express. 2017. Express. Web site. Available at: https://expressjs.com/ [Accessed 14 August 2022].

ExpressJS Tutorial. 2022. Tutorialspoint. Web site. Available at: https://www.tutorialspoint.com/expressjs/index.htm [Accessed 14 August 2022].

Felter, W., Ferreira, A., Rajamony, R. & Rubio, J. 2015. An updated performance comparison of virtual machines and linux containers. In Proceedings of the 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Philadelphia, USA, 171–172. WWW document. Available at: https://ieeexplore.ieee.org/document/7095802 [Accessed 7 July 2022].

Five Best Practices for Scaling DevOps Programs. 2020. eWeek, 1. WWW document. Available at: https://web.p.ebscohost.com/ehost/detail/detail?vid=8&sid=29a17437-7d58-4508-8423-0ea7785882a4%40redis&bdata=JkF1dGhUeXBlPXNoaWlmc2l0ZT1laG9zdC1saXZl#AN=145713804&db=a9h [Accessed 10 July 2022].

Freeman, E. 2019. DevOps for Dummies. Hoboken, NJ: John Wiley & Sons.

Fulmer, C. L. 2007. Comparing Portals and Web Pages. In Tatnall, A. (ed.) Encyclopedia of Portal Technologies and Applications. Hershey: Information Science Reference, 162-165.

Gaedke, M., Meinecke, J. & Nussbaumer, M. 2004. Supporting Secure Deployment of Portal Components. In Koch, N., Fraternali, P. & Wirsing, M. Web Engineering (eds.) 4th International Conference, ICWE 2004, Munich. Heidelberg: Springer, 516-520.

GitLab. n.d. Crunchbase. Web site. Available at: https://www.crunchbase.com/organization/gitlab-com [Accessed 14 August 2022].

GitLab. 2022. TechTarget. Web site. Available at: https://www.techtarget.com/whatis/definition/GitLab [Accessed 14 August 2022].

Go, K. 2009. What Properties Make Scenarios Useful in Design for Usability? In Kurosu, M. (ed.) Human Centered Design. First International Conference, HCD 2009. Berlin: Springer, 193-201.

Granic, A., Mitrovic, I. & Marangunic, N. 2008. Usability Evaluation of Web Portals. 30th International Conference on Information Technology Interfaces. IEEE Xplore. E-article. Available at: https://www.researchgate.net/publication/4359710_Usability_evaluation_of_web_portals [Accessed 5 June 2022].

Guruge, A. 2003. Corporate Portals Empowered With XML and Web Services. Burlington, MA: Digital Press of Elsevier Science.

Hall, T. n.d. DevOps Best Practices. Atlassian. E-article. Available at: https://www.atlassian.com/devops/what-is-devops/devops-best-practices#:~:text=DevOps%20best%20practices%20include%20agile,%2C%20observability%2C%20and%20continuous%20feedback [Accessed 13 May 2022].

Hamilton, T. 2022a. Agile Vs. DevOps: What`s the difference? Guru99. E-article. Available at: https://www.guru99.com/agile-vs-devops.html [Accessed 17 May 2022].

Hamilton, T. 2022b. What is Compatibility Testing? Forward & Backward Testing (Example). E-article. Available at: https://www.guru99.com/compatibility-testing.html [Accessed 4 June 2022].

Hamilton, T. 2022c. What is Interface Testing? Types & Example. E-article. Available at: https://www.guru99.com/interface-testing.html [Accessed 4 June 2022].

Humble, J. & Molesky, J. 2011. Why enterprises must adopt devops to enable continuous delivery. *Cutter IT Journal*, 24(8), 6. E-journal. Available at: https://www.researchgate.net/publication/298620122_Why_enterprises_must_adopt_devops_to_enable_continuous_delivery [Accessed 1 July 2022].

Hwang, J., Nkenyereye, L., Sung, N., Kim, J. & Song, J. 2021. IoT Service Slicing and Task Offloading for Edge Computing. *IEEE Internet Things Journal*, 8, 11526-11547. E-journal. Available at: https://arxiv.org/ftp/arxiv/papers/2008/2008.10210.pdf [Accessed 3 July 2022].

Ismail, K. 2018. Agile vs DevOps: What`s the Difference? CMS Wire. E-article. Available at: https://www.cmswire.com/information-management/agile-vs-devops-whats-the-difference/ [Accessed 17 May 2022].

Krief, M. 2019. Learning DevOps: The complete guide to accelerate collaboration with Jenkins, Kubernetes, Terraform and Azure DevOps. Birmingham: Packt Publishing.

Kumar, P. 2016. 4 crucial elements that dictate the success of a web portal. Chimes. E-article. Available at: https://www.itchimes.com/blog/4-crucial-elements-that-dictate-the-success-of-a-web-portal [Accessed 24 July 2022].

Large, A., Nesset, V., Beheshti, J. & Bowler, L. 2004. Criteria for Children`s Web Portals: A Comparison of Two Studies. *Canadian Journal of Information & Library Sciences*, 28(4), 45-72. E-journal. Available at: https://www.researchgate.net/publication/290671892_Criteria_for_children's_web_portals_A_comparison_of_two_studies [Accessed 2 July 2022].

Learn Kubernetes Basics. 2022. Kubernetes. Web site. Available at: https://kubernetes.io/docs/tutorials/kubernetes-basics/ [Accessed 21 May 2022].

Lozhkin, I. 2011. Internet portals, Intranet portals – how do they work? Arnica. E-article. Available at: https://www.arnicasoftware.com/blog/internet-portals-intranet-portals-how-do-they-work-/2897/index.aspx [Accessed 20 May 2022].

Lwakatare, L. E., Kuvaja, P. & Oivo, M. 2015. Dimensions of DevOps. In Lassenius, C., Dingsoyr, T. & Paasivaara, M. (eds) Agile Processes in Software Engineering and Extreme Programming. Cham: Springer, 212-217.

Mane, M. & Pange, B. 2015. Development of Library Portal-In Print and Non-Print Era. *International Journal of Advanced Library and Information Science*, 2(1), 121-125. E-journal. Available at: https://www.researchgate.net/publication/316479000_Development_of_Library_Portal-_In_Print_and_Non-Print_Era [Accessed 28 June 2022].

MEAN Stack vs MERN Stack. 2022. IntelliPaat. Web site. Available at: https://intellipaat.com/blog/mean-vs-mern-stack-development-difference/ [Accessed 10 August 2022].

Menouer, T. 2021. KCSS: Kubernetes container scheduling strategy. *The Journal of Supercomputing*, 77(5), 4267-4293. E-journal. Available at: https://www.researchgate.net/publication/345481618_KCSS_Kubernetes_container_scheduling_strategy [Accessed 29 June 2022].

Miller, A. W., Giachetti, R. E. & Van Bossuyt, D. L. 2022. Challenges of Adopting DevOps for the Combat Systems Development Environment. *Defense Acquisition Research Journal: A Publication of the Defense Acquisition University*, 29(1), 22-49. E-journal. Available at: https://web.p.ebscohost.com/ehost/pdfviewer/pdfviewer?vid=5&sid=67ce20f5-d066-4341-9942-96ff47e14f7d%40redis [Accessed 9 July 2022].

Molina, F. & Toval, A. 2009. Integrating usability requirements that can be evaluated in design time into Model Driven Engineering of Web Information Systems. *Advances in Engineering Software*, 40, 1306–1317. E-journal. Available at: https://www.researchgate.net/publication/222430027_Integrating_usability_requirements_that_can_be_evaluated_in_design_time_into_Model_Driven_Engineering_of_Web_Information_Systems [Accessed 3 July 2022].

MongoDB Tutorial. 2022. Tutorialspoint. Web site. Available at: https://www.tutorialspoint.com/mongodb/index.htm [Accessed 14 August 2022].

Myrbakken H. & Colomo-Palacios R. 2017. DevSecOps: A Multivocal Literature Review. In Mas, A., Mesquida, A., O'Connor, R., Rout, T. & Dorling A. (eds) Software Process Improvement and Capability Determination. SPICE 2017. Communications in Computer and Information Science, 770. E-article. Available at: https://www.semanticscholar.org/paper/DevSecOps%3A-A-Multivocal-Literature-Review-Myrbakken-Palacios/6487ea3275f532b50b55d6cd41813512fc12d7b9 [Accessed 5 June 2022].

Nguyen, Q-M., Phan, L-A. & Kim, T. 2022. Load-Balancing of Kubernetes-Based Edge Computing Infrastructure Using Resource Adaptive Proxy. *Sensors*, 22(8), 1-16. E-journal. Available at: https://web.p.ebscohost.com/ehost/pdfviewer/pdfviewer?vid=6&sid=217c326c-c210-4cac-85dd-95c8f367b107%40redis [Accessed 9 July 2022].

Nyirongo, A. 2015. Auditing Information Systems: Enhancing Performancee of the Enterprise. Bloomington, IN: Trafford Publishing.

Ogala, J. O. 2022. A Complete Guide to DevOps Best Practices. *International Journal of Computer Science and Information Security*, 20(2), 1-6. E-journal. Available at: https://www.academia.edu/75127013/A_Complete_Guide_to_DevOps_Best_Practices [Accessed 10 July 2022].

Paiano, R. & Roggerone, A. 2010. My E-Wall: Design of a Web Portal Fully Customizable and Ontology Based. *BRAIN. Broad Research in Artificial Intelligence and Neuroscience 1*, 65-73. E-journal. Available at: https://www.academia.edu/35791976/MY_E_WALL_DESIGN_OF_A_WEB_PORTAL_FULLY_CUSTOMIZABLE_AND_ONTOLOGY_BASED [Accessed 15 June 2022].

Pan, J. & McElhannon, J. 2017. Future Edge Cloud and Edge Computing for Internet of Things Applications. *IEEE Internet Things Journal*, 5, 439–449. E-journal. Available at: https://ieeexplore.ieee.org/document/8089336 [Accessed 17 June 2022].

Parikh, K. & Johri, A. 2022. Combining DataOps, MLOps and DevOps: Outperform Analytics and Software Development with Expert Practices on Process Optimization and Automation. E-book. BPB Publications.

Pearson, J. M. & Pearson, A. 2008. An exploratory study into determining the relative importance of key criteria in web usability: A multi-criteria approach. *The Journal of Computer Information Systems*, 48(4), 115–127. E-journal. Available at: https://www.tandfonline.com/doi/abs/10.1080/08874417.2008.11646041 [Accessed 30 May 2022].

Pestchanker, A. 2018. DevOps Master Courseware. E-book. Van Haren Publishing.

Polgar, J. 2012. Using WSRP 2.0 with JSR 168 and 286 Portlets. In Adamson, G. & Polgar, J. (eds) Enhancing Enterprise and Service-Oriented Architectures with Advanced Web Portal Technologies. Hershey: Information Science Reference, 37-49.

Qian, K., Fu, X., Tao, L., Xu, C. & Diaz-Herrera, J. 2010. Software Architecture and Design Illuminated. London: Jones and Bartlett Publishers.

Rafi, S., Akbar, M. A., AlSanad, A. A., AlSuwaidan, L., AL-AlShaikh, H. A. & AlSagri, H. S. 2022. Decision-Making Taxonomy of DevOps Success Factors Using Preference Ranking Organization Method of Enrichment Evaluation. *Mathematical Problems in Engineering*, 1-15. E-journal. Available at: https://www.hindawi.com/journals/mpe/2022/2600160/ [Accessed 2 June 2022].

Rau, P-L. P., Gao, Q. & Liu, J. 2007. The Effect of Rich Web Portal Design and Floating Animations on Visual Search. *International Journal of Human-Computer Interaction*, 22(3), 195-216. E-journal. Available at: www.researchgate.net/profile/Pei-Luen-Rau/publication/215520628_The_Effect_of_Rich_Web_Portal_Design_and_Floating_Animations_on_Visual_Search/links/56c57e9a08aeeeffa9e5f056/The-Effect-of-Rich-Web-Portal-Design-and-Floating-Animations-on-Visual-Search.pdf [Accessed 21 May 2022].

Richardson, W. C., Avondolio, D., Vitale, J., Len, P. & Smith, K. T. 2004. Professional Portal Development with Open Source Tools: Java Portlet API, Lucene, James, Slide. Indianapolis: Wiley Publishing.

Rohira, N. 2021. 8 Key Elements of a Web Portal. CRMjetty. E-article. Available at: https://www.crmjetty.com/blog/8-elements-web-portal/ [Accessed 24 July 2022].

Ruffini, M. F. 2001. Blueprint to develop a great web site. T H E (Technological Horizons in Education). E-article. Available at: https://www.thefreelibrary.com/Blueprint+to+Develop+a+Great+Web+Site.-a072960477 [Accessed 26 April 2022].

Saini, H., Mishra, S. & Prateek, P. 2014. Proactive Security Framework for Online Business Web Portals with Implementation Details. *International*

Journal of Information Secutiry Science, 3(2), 156-164. E-journal. Available at: https://dergipark.org.tr/tr/download/article-file/147963 [Accessed 3 July 2022].

Sampson, D. 2007. Evaluation of Web Portals. In Tatnall, Arthur (ed.) Encyclopedia of Portal Technologies and Applications. Hershey: Information Science Reference, 376-383.

Sampson, D. & Manouselis, N. 2005. A Flexible Evaluation Framework for Web Portals Based on Multi-Criteria Analysis. In Tatnall, A. (ed.) Web Portals: The New Gateways to Internet Information and Services. Hershey: Idea Group Publishing, 185-203.

Seremet, Z. & Rakic, K. 2021. Best Approach to Security in Azure DevOps. *Annals of DAAAM & Proceedings*, 10(2), 223-230. E-journal. Available at: https://www.daaam.info/Downloads/Pdfs/science_books_pdfs/2021/Sc_Book_2021-018.pdf [Accessed 18 June 2022].

Sharma, S. K. & Gupta, J. N. 2005. Designing e-commerce portal for an enterprise – A framework. In Tatnall, A. (ed.) Web portals: The new gateway to Internet information and services. Hershey: Idea Group Publishing, 99-118.

Shivakumar, S. K. 2016. A Complete Guide to Portals and User Experience Platforms. Boca Raton, FL: CRC Press.

Simic, S. 2020. DevOps vs Agile: Differences + Head to Head Comparison. PhoenixNAP. E-article. Available at: https://phoenixnap.com/blog/devops-vs-agile [Accessed 17 May 2022].

Sindhuja, P. N. & Dastidar, S. G. 2009. Impact of the factors influencing website usability on user satisfaction. *The IUP Journal of Management Research,* 8(12), 54–66. E-journal. Available at: https://www.semanticscholar.org/paper/Impact-of-the-Factors-Influencing-Website-Usability-SindhujapP.-Dastidar/ad84a8b9f1b74efcc440860b915891627d889cb4 [Accessed 29 May 2022].

Soni, M. 2019. Agile, DevOps and Cloud Computing with Microsoft Azure: Hands-On DevOps practices implementation using Azure DevOps. E-book. BPB Publications.

Surve, S. 2021. Why You Should Use React.js For Web Development. FreeComeCamp. E-article. Available at: https://www.freecodecamp.org/news/why-use-react-for-web-development/ [Accessed 28 July].

Tarafdar, M. & Zhang, J. 2005. Analyzing the influence of web site design parameters on web site usability. *Information Resource Management Journal*, 18(4), 62–80. E-journal. Available at: https://www.researchgate.net/publication/256294756_Analyzing_the_Influence_of_Web_Site_Design_Parameters_on_Web_Site_Usability [Accessed 19 June 2022].

Tatnall, A. 2009. Web Portal Research Issues. In Khosrow-Pour, M. Encyclopedia of Information Science and Technology. 2nd edn. Hersley: IGI Global, 3642-3647.

The One DevOps Platform. 2022. About GitLab. Web site. Available at: https://about.gitlab.com/?utm_medium=social&utm_source=linkedin&utm_content=bio [Accessed 14 August 2022].

Tsui, E., Yu, C. & Lau, A. 2007. A Two-Tier Approach to Elicit Enterprise Portal User Requirements. In Tatnall, A. (ed.) Encyclopedia of Portal Technologies and Applications. Hershey: Information Science Reference, 1026-1032.

Tutorial: Intro to React. n.d. React. Web site. Available at: https://reactjs.org/tutorial/tutorial.html [Accessed 3 July 2022].

Usability.gov. 2006. Usability Evaluations: Introduction. Human Computer Interaction. Web site. Available at: https://www.cs.auckland.ac.nz [Accessed 26 April 2022].

Ward, J. L. & Hiller, S. 2005. Usability Testing, Interface Design, and Portals. *Journal of Library Administration*, 43(1-2), 155-171. E-journal. Available at: https://www.tandfonline.com/doi/abs/10.1300/J111v43n01_10 [Accessed 16 June 2022].

Web Application Testing Complete Guide (How To Test A Website). 2022. Software Testing Help. WWW document. Available at: https://www.softwaretestinghelp.com/web-application-testing/ [Accessed 5 June 2022].

Web Portal Testing. n.d. Hurix Digital. Web site. Available at: https://www.hurix.com/lp/web-portal-testing/ [Accessed 5 June 2022].

What is Express JS in Node JS? 2022. Simplilearn. Web site. Available at: https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-express-js [Accessed 14 August 2022].

What is React? n.d. w3school. Web site. Available at: https://www.w3schools.com/whatis/whatis_react.asp [Accessed 28 July 2022].

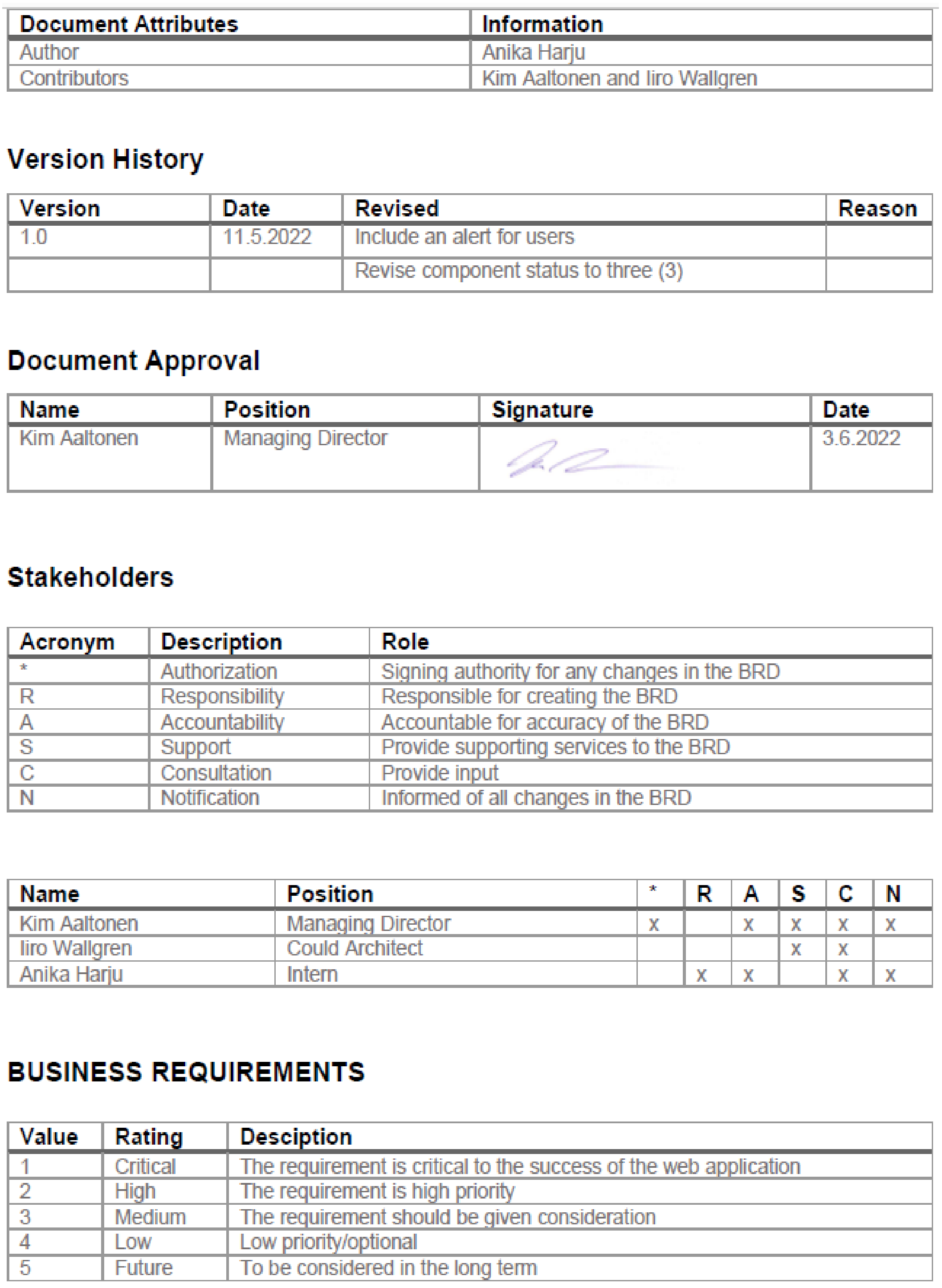
Wojtkowski, W. & Major, M. 2005. On portals: A parsimonious approach. In Tatnall, Arthur (ed.) Web Portals: The New Gateways to Internet Information and Services. Hershey: Idea Group Publishing, 21-29.

Xu, H., Huang, W., Zhou, Y., Yang, D., Li, M. & Han, Z. 2021. Edge Computing Resource Allocation for Unmanned Aerial Vehicle Assisted Mobile Network With Blockchain Applications. *IEEE Transactions on Wireless Communications,* 20, 3107–3121. E-journal. Available at: https://par.nsf.gov/servlets/purl/10212957 [Accessed 11 June 2022].

Zhou, J., Yu, Y., Zhang, L., Lin, C. & Yang, Y. 2005. Integrating Web Services into Ontology-Based Web Portal. In Zhang, Y., Tanaka, K., Xu Yu, J., Wang, S. & Li, M. (eds.). Web Technologies Research and Development - APWeb 2005: 7th Asia-Pasific Web Conference, Shanghai. Berlin: Springer, 585-596.

APPENDICES

1 Business Requirements Document (BRD)



Document Attributes	Information
Author	Anika Harju
Contributors	Kim Aaltonen and Iiro Wallgren

Version History

Version	Date	Revised	Reason
1.0	11.5.2022	Include an alert for users	
		Revise component status to three (3)	

Document Approval

Name	Position	Signature	Date
Kim Aaltonen	Managing Director	[illegible]	3.6.2022

Stakeholders

Acronym	Description	Role
*	Authorization	Signing authority for any changes in the BRD
R	Responsibility	Responsible for creating the BRD
A	Accountability	Accountable for accuracy of the BRD
S	Support	Provide supporting services to the BRD
C	Consultation	Provide input
N	Notification	Informed of all changes in the BRD

Name	Position	*	R	A	S	C	N
Kim Aaltonen	Managing Director	x		x	x	x	x
Iiro Wallgren	Could Architect				x	x	
Anika Harju	Intern		x	x		x	x

BUSINESS REQUIREMENTS

Value	Rating	Descption
1	Critical	The requirement is critical to the success of the web application
2	High	The requirement is high priority
3	Medium	The requirement should be given consideration
4	Low	Low priority/optional
5	Future	To be considered in the long term

User Requirements

Req ID	Req Type	Req Description	Priority	Comments
0001	Expanded views	Based on user, see an expanded view	Critical	ReactJs Front-end
0002	Login option	Login for registered users	High	Login with OpenID Connect
0003	Traffic light view	Service based visual operational view	Critical	Traffic lights should update without page refresh
0004	Push/pull functionality	Ability to push or pull notifications for service status	Critical	Back-end
0005	E-mail formalization	Formalize structure of the e-mail communication, information, incident and maintenance	Medium	Back-end
0006	Self-service registration	Register to the page as a self-service from Datalounges.com	Medium	Keycloak
0007	Acceptance of registration by an employee of Datalounges	An employee of Datalounges must accept the registration before a user gains access to the service or domain	Low	
0008	A notification e-mail to a superuser	Send an e-mail to the user support of Datalounges (datalounges.com)	Low	Back-end
0009	Accept terms and conditions package	Clients must accept terms of service etc. upon registration	Critical	Keycloak
0010	Confirmation e-mail for access	Clients to receive approved e-mail notification regarding access to the service	Low	Keycloak
0011	User information	First name, last name, business e-mail and the name of the organization	Medium	Keycloak
0012	Change of user information	Users can edit personal information except the e-mail	Medium	Keycloak
0013	Delete user as a self-service	A request must be made to delete user data	Medium	Keycloak

Functional Requirements

Req ID	Req Type	Req Description	Priority	Comments
0001	Kubernetes	Full Kubernetes cluster	Critical	
0002	Public IP	Available public IP	Critical	
0003	Color code formalizations	Incident: Red Maintenance: Yellow Info: Blue	High	
0004	Sub-messages per major event	Major event, title, the ability to write messages and updates related to incidents/other issues	High	
0005	A view of the announcements	Date, heading, type, text, history view of past incidents, and a link to more information	Critical	
0006	A view of the systems	Ensure the system is available: k2 and k3, AWS, R4DAR, DL Git, and DL Harbor	Critical	
0007	Interactions	Contact Datalounges, open support request, subscribe/unsubscripe	Future	

		to e-mails and info, and access to documentation		
0008	Admin view of the page	Ability to write announcements and sub announcements Select the type of communications info/maintenance/incident Option to timestamp the messages (date and time) Update/change announcements	Critical	

Reporting Requirements

Req ID	Req Type	Req Description	Priority	Comments
0001	Users in the system	Ability to export users from the system with their info		
0002	Announcements	Export the announcements		
0003	Report of logins	Successful, unsuccessful, active users, and date and time of users logged in		Keycloak

Usability Requirements

Req ID	Req Type	Req Description	Priority	Comments
0001	Table view	Table view of service traffic light	Critical	
0002	Date and time	Timestamp each notification	Critical	

Reliability Requirements

Req ID	Req Type	Req Description	Priority	Comments
0001	Auto failover	Multiple instances of the application (containers)	Critical	
0002	Retry for services	Upon single failure, retry due to network issues	Critical	
0003	Do not run in DL capacity	Use AWS workload to address problems and still communicate with clients		

Performance Requirements

Req ID	Req Type	Req Description	Priority	Comments
0001	Memory footprint	Deploy small containers to keep performance fast		
0002	Database	Separate database from application to improve performance and resilience		MongoDB

Supportability Requirements

Req ID	Req Type	Req Description	Priority	Comments
0001	Git	Ensure Git is accessible	Critical	
0002	Harbor	Ensure Harbor is accessible	Critical	

Testing Requirements

Req ID	Req Type	Req Description	Priority	Comments
0001	Separate testing and staging	Test environment: where to compile new versions (Datalounges) Staging: location of functionality check in AWS Production: separate in AWS		

Training Requirements

Req ID	Req Type	Req Description	Priority	Comments
0001	Team training	Responsibilities to check users and access requests Roles (announcements) What to do in case of an incident?		

Capacity Requirements

Req ID	Req Type	Req Description	Priority	Comments
0001	CPU	Minimum 2 vCPU	Critical	
0002	Memory	Minimum 4 GB	Critical	

Storage Requirements

Req ID	Req Type	Req Description	Priority	Comments
0001	Shared disk	Minimum 10 GB volume	Critical	

2 The HTML Code for the Web Portal

```html
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8" />
      <meta http-equiv="X-UA-Compatible" content="IE=edge" />
      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
      <link rel="stylesheet" href="portaltwo.css" />
      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.14.0/css/all.min.css"
         integrity="sha512-1PKOgIY59xJ8Co8+NE6FZ+LOAZKjy+KY8iq0G4B3CyeY6wYHN3yt9PW0XpSriVlkMXe40PTKnXrLnZ9+fkDaog=="
         crossorigin="anonymous" />
      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css" />
      <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.6.1/css/font-awesome.min.css" rel="stylesheet" />
      <link href=" https://fonts.googleapis.com/css?family=Exo 2   " rel="stylesheet" />
      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css" />
      <title>Document</title>
   </head>
   <body>
      <!--Header-->
      <div class="header">
         <img src="images/logo.PNG" alt="" class="logo" width="500" />
         <div class="btns">
            <a href="#subscribe" class="subscribe-btn">Subscribe To Updates</a>
            <a href="#login" class="login-btn">Login</a>
         </div>
      </div>
      <!--Login-->
      <div class="login-form hide" id="login">
         <form action="#">
            <a href="#" class="close-btn">x</a>
            <div class="data">
               <label>Email</label>
               <input type="text" placeholder="Enter your e-mail" required />
            </div>
            <div class="data">
               <label>Password</label>
               <input type="password" placeholder="************" required />
            </div>
            <!--Reset password-->
            <div class="reset-pass">
               <a href="#">Reset Password</a>
            </div>
            <!--Submit button-->
            <div class="btn">
               <div class="inner"></div>
               <button type="submit">Submit</button>
            </div>
         </form>
      </div>
```

```
<div class="subscribe-form hide" id="subscribe">
   <!--Subscribe form-->
   <form action="#">
      <a href="" class="close-btn">x</a>
      <div class="data">
         <input type="text" placeholder="Enter your e-mail" required />
      </div>
      <!--Subscribe submit-->
      <div class="btn">
         <div class="inner"></div>
         <button type="submit">Subscribe via E-mail</button>
      </div>
      <!--Subscribe sign up-->
      <div class="signup-link">
         Do you want notifications? <a href="#">Sign up now</a>
      </div>
   </form>
</div>
<br />
<!--Status-->
<div class="status">
   <a href="#current" class="current-status">Current Status</a>
   <a href="incidenttwo.html" class="past-incidents">Past Incidents</a>
</div>
<div class="current-form" id="current">
   <form action="#">
      <a href="#" class="close-btn"> x </a>
      <div class="announcement-data">
         <label>Issue</label>
         <input type="text" class="issue" placeholder="type here............." />
      </div>
      <div class="announcement-data">
         <label>Details</label>
         <textarea rows="2" cols="20" placeholder="type here..........."></textarea>
      </div>
      <div class="announcement-data">
         <label>Update</label>
         <textarea rows="2" cols="20" placeholder="type here..........."></textarea>
      </div>
      <div class="btn">
         <div class="inner"></div>
         <button type="submit">Submit</button>
      </div>
   </form>
</div>
<!--Main Traffic Light-->
<div class="all-systems">
   <div class="item1"></div>
   <div class="item2"></div>
   <div class="item3"></div>
   <div class="item2"></div>
   <div class="item2"></div>
   <div class="item3"></div>
   <div class="item3"></div>
   <div class="item2"></div>
   <div class="item3"></div>
   <div class="item3"></div>
</div>
<br />
```

```
    <!--Auto Text-->
    <div>
       <h1 class="sigh">About This Site</h1>
       <div>
          <ul class="auto-text">
             <li>
                <span>We place a great value on our uptime and performance.</span>
             </li>
             <li>
                <span>If there are any issues, this page will immediately update,
                alerting our engineering team.</span>
             </li>
             <li>
                <span>Please e-mail kim.aaltonen@datalounges.com if you need
                assistance.</span>
             </li>
          </ul>
       </div>
    </div>
    <!--Traffic Light-->
    <div class="service-checker1">
       <div class="service-heading">Services</div>
       <div class="service-heading">Status</div>
       <div class="service-heading">Response Time</div>
       <div class="service-heading">API</div>
       <div class="service-heading">Response</div>
       <!--Service 1-->
       <div class="monitor" id="service1">https://a.hiveyard.io/</div>
       <div class="monitor" id="checker1">
          <i class="fa fa-check-circle" style="color: green; font-size: 20px"></i>
       </div>
       <div class="monitor" id="checker2">
          <i class="fa fa-check-circle" style="color: green; font-size: 20px"></i>
       </div>
       <div class="monitor" id="checker3">
          <i class="fa fa-check-circle" style="color: green; font-size: 20px"></i>
       </div>
       <div class="monitor" id="checker4">
          <i class="fa fa-check-circle" style="color: green; font-size: 20px"></i>
       </div>
    </div>
    <br />
    <!--Service2-->
    <div class="service-checker2">
       <div class="monitor" id="service2">https://k3.hiveyard.io</div>
       <div class="monitor" id="checker5">
          <i class="fa fa-check-circle" style="color: green; font-size: 20px"></i>
       </div>
       <div class="monitor" id="checker6">
          <i class="fa fa-check-circle" style="color: green; font-size: 20px"></i>
       </div>
       <div class="monitor" id="checker7">
          <i class="fa fa-check-circle" style="color: green; font-size: 20px"></i>
       </div>
       <div class="monitor" id="checker8">
          <i class="fa fa-check-circle" style="color: green; font-size: 20px"></i>
       </div>
    </div>
    <br />
```

```
      <!--Add extra services here!!!!-->
      <!--Service3
         <div class="service-checker3">
           <div class="monitor" id="service3">https://www.google.fi/</div>
           <div class="monitor"></div>
           <div class="monitor"></div>
           <div class="monitor"></div>
           <div class="monitor"></div>
         </div>-->
      <!--Tracking colors-->
      <div class="system-colors">
         <i class="fa fa-circle" style="color: green"></i>
         Operational
         <i class="fa fa-circle" style="color: orange"></i>
         Degraded performance
         <i class="fa fa-circle" style="color: red"></i>
         Outage
         <i class="fa fa-circle" style="color: gray"></i>
         Maintenance
      </div>
      <!--Footer-->
      <div class="footer"></div>
   </body>
</html>
```