INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Desenvolvimento de um ERP com CI/CD, Autenticação e Auditoria do sistema

PORFÍRIO AFONSO FERNANDES
Outubro de 2023

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Development of an ERP with CI/CD application, Authentication and System Auditing

Porfírio Afonso Fernandes

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Computer Science, Specialisation Area of Software Engineering

Supervisor: Joaquim Filipe Peixoto dos Santos

ISEP, Porto, October 14, 2023

Declaration of Integrity

I declare that I have conducted this academic work with integrity.

I have not plagiarised or applied any form of misuse of information or falsification of results throughout the process that led to its preparation

Therefore, the work presented in this document is original and of my own authorship, and has not previously been used for any other purpose.

I also declare that I am fully aware of P.PORTO's Code of Ethical Conduct.

ISEP, Porto, October 14, 2023

Dedicatory

To my mentor Joaquim Filipe Peixoto dos Santos, that accepted this challenge and guided me so that this project could reach to a great result, even after a significant change in the project occured, by encouraging me to continue and finish this thesis.

To my family, that helped me support all the work that I invested in this project, and by helping me when I was struggling with the project.

To my friends who have lived with me over the last year, for being there by my side, and making sure that, in the middle of all the anxiety and fatigue I felt with dealing both work and university, they were there to help me enjoy life.

To my friends, for being around me during my free time, and making sure that I had all the strength needed to pursue the write of this dissertation.

Without the people around me, this would not be possible.

Abstract

Developing and maintaining software like ERPs can be challenging because of the complexity and the amount of data that these systems require maintaining. Many of the software programs can grow with weak structure, which lead to great effort to maintain, and with more probability to error.

This project proposes that a development cycle that incorporates DevOps can have major benefits, by not only removing some hassle the programmers and systems admins have with testing and deploying the system, but can also give a early feedback if the changes made into the application brings problems to the systems. The design of a CI/CD pipeline and audit logs, and the implementation in an ERP development helped get more feedback and cause of root problems, which lead to more confidence in the developers to make changes, and to escalate more quickly since the deployment is automatized.

Keywords: CI/CD, DevOps, Audit, ERP, Development

Resumo

Desenvolver "software" como os ERPs podem ser difícil de manter devido à complexidade e a quantidade de dados envolvida nestes sistemas. Isto leva a que muitos destes "softwares" cresçam com uma estrutura de código fraca, o que leva a um esforço adicional para manter, e com maior probabilidade para erros.

Este projeto propõe que a incorporação do conceito de DevOps no ciclo de desenvolvimento traz muitas vantagens, não só a remover algum trabalho dos programadores e dos administradores de sistemas ao ser mais fácil testar o sistema e fazer *deploy* do mesmo, mas também fornece uma forma de feedback mais rápida para eventuais erros. O "design" de uma pipeline CI/CD e logs para auditoria do sistema, e a respetiva implementação destes conceitos no desenvolvimento consegue dar mais feedback a problemas, o que leva a uma maior confiança dos programadores para fazer alterações, e conseguir escalar a solução mais rapidamente visto que a implantação é automatizada.

Contents

List of Figures

List of Tables

List of Source Code

Abbreviations

DevOps	Development and Operations.
DevSecOps	Development, Security and Operations.

List of Acronyms

AHP	Analytic Hierarchy Process.
CI/CD	Continuous Integration/Continuous Deployment.
CIA	Confidentiality, Integrity, and Availability.
DAST	Dynamic Application Security Testing.
DORA	DevOps Research and Assessment.
ERP	Enterprise Resources Planning.
FAST	Function Analysis System Technique.
FFE	Fuzzy Front-End.
IaC	Infrastructure as Code.
IAST	Interactive Application Security Testing.
MVP	Minimun Value Product.
NCD	New Concept Development.
NPD	New Product Development.
SAST	Static Application Security Testing.
VM	Virtual Machine.

Chapter 1

Introduction

This chapter provides an overview of the key concepts that led to the development of this dissertation, with a description of the main topic. First it is justified what is the motivation behind this dissertation, and the current problem that exist, which forms the basis of this dissertation. After that, is described the objectives that are intended from the development of this project, and what results should be expected from the development. In the following sections is explained what will be the approach for this thesis, that will guide the theory and the practical parts of it. It ends with a summary of the following chapters of this document.

1.1 Context

This dissertation was developed in the second year of Master's Degree in Informatics Engineering of Instituto Superior de Engenharia do Porto (ISEP), in the specialization of Software Engineering. It was a self-proposed project, supported by the supervisor, for implementing a Development and Operations (DevOps) process in the development of an application (in this case, an Enterprise Resources Planning (ERP) application), planned by three students in order to finish the Master's Degree, while demonstrating the skills acquired during the master's programme. The outcome is the analysis of the current DevOps state, implementation of it during the development of an application, and documentation of the advantages and disadvantages on using DevOps in a software development cycle.

The motivation for this project came from one of the subjects during the Master's Degree, demonstrating the importance of a good support process in the software development cycle, since it can affect how a code base and a product is maintained. Since then, the desire to explore the subject further was stronger, due to the fact that normally the focus of developers is programming software, and this project is a great opportunity to explore more about it.

Also, another reason was the increase in the market need of ensuring that the product is robust enough, since the shorted development cycles are causing more stress and effort to maintain a high level of efficiency and optimization, with fewer bugs. The products are growing to be more complex, with more integrations, dealing with scalability problems, and to minimize the downtime of the system, since many companies are now dependent of the software being completely functional, or otherwise they stop. And with more regulations and security concerns, the compliance also needs to be cared [1].

Software engineers need tools that can automate part of the task of code management, so they can focus only on the primary task of developing new features and fixes. The market, by many years, already offer many solutions that allow the developer to build, test and

deploy. However, these steps were normally manually done, and was prone to error and "good practices" of the developer. Can DevOps tools increase reliability, maintainability and number of deployments in a given application, while offering a better developer experience by automating part of their job [2]?

1.2 Problem

In an increasingly digital world it has become almost mandatory for all businesses to use IT systems to support their management. The digitalization made the market faster, which forced the rest of the companies to keep the pace, and join the IT adoption. New features are asked, the pressure rises to deliver faster, software must do more things and integrate with different sources, etc. It is crucial for an organization to be able to adjust everything about their work, in order to stay competitive and relevant [3]. Nowadays, it becomes usual to worry about how to create the infrastructure with the right hardware, to know all the software required to deploy an application, and after that how to maintain and have prepared the procedures to migrate or upgrade the systems when needed.

On the other side, software companies are now facing a tremendous search, since the market require systems that can support their processes, so they can have a chance to increase. So, the software cycle become important, as the release of a defective application to customers require detecting, reproduce, fix it, test the correction, and release the software again with the right corrections, while still trying to deliver new features that can correspond to their needs. This also applies to the security aspect of the applications, because systems become more interconnected with others software programs, that creates an exposed attack surface for hackers, and incorporate security as the root of the development is important, and not just as add-on that is checked before release, since the tools that can automate the exploration and application of vulnerabilities are rising [4].

Finally, developers have a problem with dealing with complex structure of code, and the infrastructure of the application deployed. The tools are becoming more abstract to the developer, which can lead to faster time to produce functionalities, but also more obscure problems related with the dependencies and tools used in the tech stack of the application. As such, it is necessary that the process of development and deployment can be as much automatized, consistent and observable, in order to maintain a fast and reliable way to deliver secure and robust software.

1.3 Objectives

In this dissertation it is expected the analysis of the DevOps methodology, and the tools that are available to enforce and automate the process. From the information gathered, implement, configure and enforce a DevOps process, while developing an ERP application.

Is intended to make an implementation of a Continuous Integration/Continuous Deployment (CI/CD) pipeline that uses this tool to enforce a rigorous process, while removing manual processes that could result in human error. The pipeline should enforce quality rules that obligate the developers to be more careful about the code written, and also automate the deployment of new code into the respective environments (staging and production), with the right management of the credentials and data that should be in place.

The logging and monitoring of the application and infrastructure is also expected to be made, since is one vital phase of the DevOps cycle. This requires what can be done to log and observable the application, and at the same time the system that is running the application, in order to keep track of the performance and possible failure of components while deployed.

This dissertation has the following objectives:

- Analysis and documentation of the DevOps processes and the different existing tools to support this process, and creation of a pipeline to automate the DevOps process;
- Analysis of the different logging methodologies and the different existing tools, and development of an application logging mechanism;
- Design and implementation of a CI/CD pipeline using a tool already existing in the market;
- Implementation of a monitoring system in order to, in conjunction with the logs of the system, keep track of the application performance and number of errors.
- Compilation of the results taken during development to analyse the importance for DevOps and audit tools use in development of an application.

With the concretization of the objectives proposed, it is expectable more comprehension about the DevOps cycle, and how it can reduce manual work, help reach faster times of deployment, reduce number of errors, and overall better experience for the developers.

The process of developing a practical application while pushing a DevOps process must ensure traceability of the bugs made, and how the developer can have early feedback in the process.

1.4 Approach for the project

This work will first conduct a study of the necessary knowledge related to the project, then a design will be performed to answer the problem, and then a description of the implementation with the results obtained.

To follow this segmentation, the study will be performed using scientific documents gathered from platforms like IEEE (IEEE Xplore), ACM Digital Library (About ACM DL) and Springer. Then, the system will be designed, describing the advantages and disadvantages of the approach, with the right explanation of the decision made. The next phase is the implementation, where the pipeline is configured, with the right optimizations, and made so that the process is validated. Finally, a set of tests will be made to validate if the design and implementation of the solution solves the problem.

1.5 Structure of the Document

The thesis follows a structure where each chapter has a specific purpose. The current chapter, as mentioned previously, provides the context and purpose of this dissertation, and the objectives that are proposed to be done and the solution being implemented.

Introduction: In this chapter it is contextualized the project, the purpose of it and the motivation behind it. It is described the objectives and expected results from it.

State of the Art: In this chapter it is made a contextualization of the current knowledge about the topics and concepts that are used in this project, to give the fundamentals needed to understand this document, like DevOps, Infrastructure as Code, etc. It is also described several tools that are available in the market and that can be used in the project.

Value Proposition: This part of the document is where is analysed the value that can be obtained from this project in terms of advantages in DevOps implementation during a software cycle. This includes the value proposition proposed, and the application of the methods that helps decide the solution that will go to be used.

Technologies Used: This chapter described all the tools used during the development of this project, with detail about the characteristics and the decision behind using them.

Design: This part of the document explains the solution implemented in this project, with their respective advantages and disadvantages.

Implementation: This chapter describes all the important details of the implementation of this project, from the infrastructure to the demonstration of the process.

Experimentation and Evaluation: In the chapter 7, is described which hypotheses will be tested in this project, and the results obtained after validating the values of the metrics with the objectives.

Conclusion: Finally, what goals where achieved, the limitations and future work, and the final appreciation is made in the chapter 8.

Chapter 2

State of the Art

In this chapter is present all the knowledge related to the main concepts of this project. This is necessary in order to collect, analyse and comprehend the necessary information in order to understand the problem, and create the background needed to design a solution.

The chapter 2 include both the scientific knowledge and the technologies and practical comparisons between them. It is described what is the DevOps and the concepts associated with it, the technologies involved in this project, and an ERP. The research for the technologies that this thesis focus the most is also described.

2.1 Context

The use of automated tools to compile, test and deploy already exist in the early 2000s, during the phase of divulgations of the Xtreme programming methodology. However, with the more adopted Agile work methodology, and the rise of competition in the market, cloud services and general computational power, only in the last years the CI/CD tools saw their boom.

In the same way, some studies were already made about this area, and what challenges and benefits the companies can face when embracing these tools. And the results, even with the effort necessary to implement, are enough to bring processes more efficient, that includes time and costs reduced, and overall increase in quality and productivity.

2.2 DevOps

Development and Operations (DevOps) is a collection of ideas, practices, processes and technologies that allow development and operation teams to work together to streamline product development. It helps the companies to give the next step into their teams processes development [5].

The main idea is to break the difference between the dev team and the operations/support team by combining and connect their tasks that create a specification that both parties understand and contribute to, not only to create better software, but also a good build and deployment of it [6].

As illustrated in the figure 2.1, the practices inherent to DevOps is:

- Test Automation
- Integrated Configuration Management

- Integrated Change Management
- Continuous Integration
- Continuous Delivery
- Continuous Deployment
- Application Monitoring
- Automated Dashboard

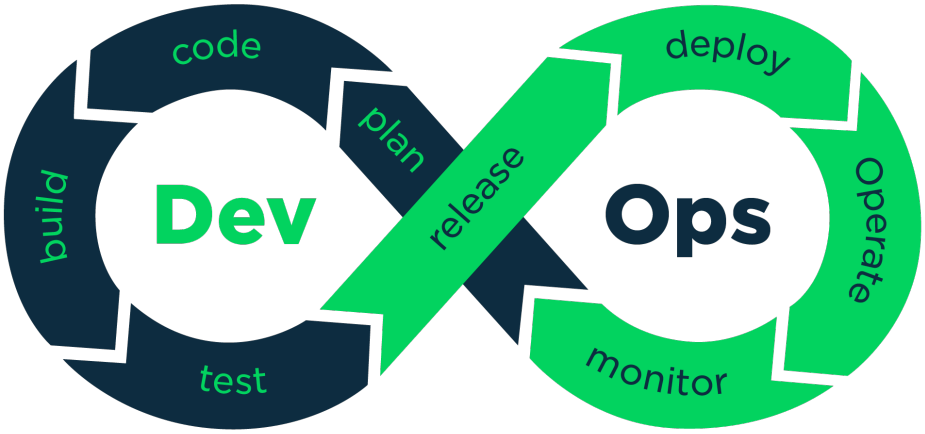

Figure 2.1: DevOps

Once code changes are integrated, they are automatically deployed to production, eliminating the need for manual deployment processes and reducing the potential for human error. This consequently shortens the time it takes to provide users with new features and bug fixes. As a result, teams can swiftly adapt to the evolving user requirements and business needs. These practices embody fundamental principles. For instance, it is expected that code changes are regularly integrated and tested into a central repository, often multiple times a day.

This enables developers to detect and resolve integration issues early in the development process, preventing them from escalating and becoming more challenging to fix. By integrating code changes frequently, it becomes possible to promptly identify and address integration problems, thereby reducing the likelihood of defects. Moreover, collaborating on a shared codebase becomes more convenient, minimizing the risks and time required to resolve issues and enhancing the quality of the codebase [7].

However, after the changes are implemented for users, it is crucial for teams to continuously gather feedback from stakeholders, users, and even other development and operations teams. This enables the team to comprehend the needs and concerns of different groups, facilitating more effective collaboration in delivering value. This feedback also provides valuable data for monitoring the system and identifying real-time issues, such as the frequency of errors and the alignment of the new changes with user expectations and desired features [8].

With these practices of connecting the workflow of managing the code, and the automation of the infrastructure, the company see an increase in efficiency and communication between their collaborators. Nevertheless, this implies a process and cultural shift inside a company in order for this to work [6].

The DevOps importance is still growing, with IDC predicts that the DevOps market is expected to grow from $2.9 billion to $8 billion by 2022. And the companies that already implemented DevOps has had a positive impact on their organization, helped them produce higher quality deliverables, and a reduction in time-to-market software and service in 2020. Some studies state that the use of DevOps philosophy deploy 973x times faster than the others companies. Also, the company have more time to innovate that wasting time testing and deploying new versions of the software [9].

One of the challenges around DevOps is how it is possible to track the performance of a software development. This becomes relevant, as more companies want to determine if they are getting positive return of investment in terms of effort by applying DevOps. As of today, a team in Google called DevOps Research and Assessment (DORA) identified 5 metrics to help get status about the developers [10]. They are:

- Deployment Frequency - How often an organization successfully releases to production
- Lead Time for Changes - The amount of time it takes a commit to get into production
- Change Failure Rate - The amount of time it takes a commit to get into production
- Time to Restore Service - How long it takes an organization to recover from a failure in production
- Reliability - Is meet or exceed the reliability targets of the company

This helps check how fast a team is working and if they are doing a good job. It also determines if they are focusing on making good quality work or just trying to finish quickly. The metrics help leaders know where to focus their efforts. For example, if a team sees that they are not delivering updates often enough, they can start using more automated testing to speed things up and fix mistakes quickly.

This can influence the team to make better choices that improve the system, make them more valuable, and change how people think. Other metrics that can also be used is how much of the code is tested, how many mistakes get missed, how many problems customers have, how long it takes for a computer program to be checked, or how long it takes to find a problem.

2.3 DevSecOps

In recent years, the paradigm of software development has witnessed a transformative shift toward greater agility and efficiency. DevOps, an approach that emphasizes collaboration between development and operations teams, has become pivotal in achieving rapid and continuous software delivery. However, as software development accelerates, security concerns must not be sidelined. From this concern Development, Security and Operations (DevSecOps) emerged, an extension of DevOps that integrates security practices seamlessly into the software development lifecycle, ensuring that security is not compromised in the pursuit of agility and innovation.

DevSecOps is the union of "Development," "Security," and "Operations," symbolizing the convergence of these domains to create a holistic approach to software development. DevSecOps is a way of making software that focuses on keeping it safe from hackers. It combines different parts of making software, like writing the code, making it secure, and keeping it working well. Unlike old ways of making software, DevSecOps tries to find and

fix security problems early on. It follows certain rules, like doing security checks right from the start, using computers to test for security problems, and making sure everyone on the team works together. The main goals of DevSecOps are to make software more secure, fix security problems faster, and make sure everyone on the team knows how to keep things safe [11].

By incorporating security practices throughout the entire DevOps lifecycle, starting from the planning phase where threat modelling and risk assessment take place, and continuing through coding, testing, deployment, and monitoring. Security checks and validations are integrated into each phase, ensuring that security is always a priority rather than an afterthought. A fundamental aspect of DevSecOps is the idea of "Security as Code." Similar to how infrastructure can be defined and managed as code, security policies, controls, and checks are also transformed into code. This approach enables security practices being automated, versioned, and reviewed alongside application code, thereby promoting consistency and accountability [12].

The DevSecOps process supports the use of automated security testing to identify vulnerabilities in both source code and running applications. This ensures that potential threats are detected early. With the increasing popularity of containerization and microservices architectures, DevSecOps also focuses on container security. Tools like vulnerability scanners and image signing mechanisms are used to ensure that containers are not compromised and do not contain known vulnerabilities. Tools like Infrastructure as Code frameworks and configuration management tools help automate security controls, allowing for consistent application of security measures across all environments [13].

DevSecOps requires a cultural transformation where security professionals collaborate closely with developers and operations teams. Encouraging shared responsibility for security fosters a cohesive approach to risk mitigation. Adopting it involves integrating a range of security tools and practices into existing DevOps pipelines. Ensuring seamless tool integration and managing the complexity can be a challenge, requiring careful planning and expertise. Developers and operations personnel need to acquire security knowledge and skills to effectively implement DevSecOps practices. Training programs and resources play a crucial role in upskilling teams.

2.4 CI/CD

Continuous Integration/Continuous Deployment (CI/CD) is a crucial component of DevOps. Its primary objective is to enhance a company's agility and ability to innovate by automating the deployment process, while also providing faster feedback from clients. CI/CD automates the entire software development life cycle, from code integration to deployment, using a set of automated tools and processes, such as version control systems, build tools, and deployment tools. The process begins with continuous integration, where code changes are merged into a central repository like Git. Automated build and test processes then validate the code changes to ensure they meet quality standards. If the changes pass these processes, they are automatically deployed to production, thereby completing the CI/CD cycle [14].

The proven benefits of implementing CI/CD have led to a growing number of companies adopting this process. However, it is important to note that CI/CD implementation requires proper evaluation and design, as well as a certain level of expertise [8]. The CI/CD urges the team to release from the practices acquired from Agile and Gitflow, and start using

a methodology where the developers should always merge their changes if the build can success. This helps reduce the friction associated with multiple branches, resulting in fewer changes remaining in isolated branches and being merged later. Additionally, the combination of automated tests, code coverage, and code review often results in development that is almost always ready for production.

It is important to note that "CD" can be a reference to Continuous Delivery or Continuous Deployment. Both methods are utilized, but the main difference is that Continuous Delivery requires a manual step prior to production deployment, while Continuous Deployment automates the deployment process.

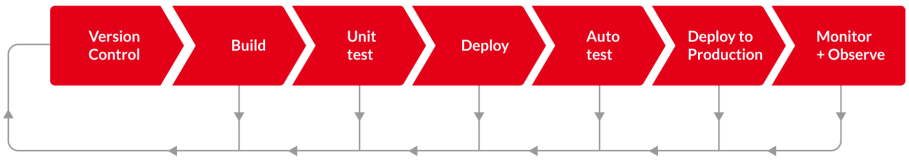

Figure 2.2: CI/CD

2.5 Tools CI/CD

In this section it is described some of the tools that exist in the market and allows the creation and execution of a pipeline for CI/CD. These tools offer hooks that can automatically be triggered when a new commit is made into a VCS repository like Git.

2.5.1 Jenkins

Jenkins, a widely used automation server, is a Java-based open-source platform, that offers developers the flexibility to carry out tasks like integration tasks, code building, testing, and software deployment. What sets Jenkins apart is its large collection of plugins that enhances its core functionalities and empowering users to achieve even greater levels of automation. Moreover, Jenkins is compatible with any machine equipped with a Java Virtual Machine. Its collaborative nature and extensive development history have propelled Jenkins to the forefront of the automation server industry, solidifying its position as a dominant player [15].

Jenkins has recently introduced a new feature that allows developers to create pipelines using a DSL (domain-specific language). This feature improves collaboration among teams by enabling them to define the pipeline within the project repository. However, some users have encountered difficulties with Jenkins as it may not be as user-friendly as other integration tools. This can be attributed to its outdated user interface and the presence of plugins that are no longer supported, deprecated, or even abandoned. Furthermore, the declarative pipeline does not fully support all the plugins available, and there is outdated documentation on how to effectively use Jenkins [16].

2.5.2 Gitlab

Gitlab is an open source version control system that provides users CI/CD tools that can enhance their experience. It allows managing, build, test, integrate and deploy source code in

a single tool, reducing the number of tools to learn. Also, because Gitlab can be configured out of the box, it is ideal for companies that have strict rules about accessing external resources[17].

GitLab's CI/CD capabilities help developers create and manage pipelines to automate the process of building, testing, and deploying software. These features can be configured to fit different software development workflows, and GitLab integrates with a variety of third-party tools to make it even easier. The GitLab pipeline is a series of stages that runs the jobs configured in the repository. Each operation is performed by one or more jobs that run in parallel or sequentially. You can define pipelines in GitLab using YAML syntax, and developers can manage them directly from the user interface.

GitLab offers a variety of pre-built templates for popular programming languages and frameworks, so you can quickly create and configure pipelines for your projects. In addition, GitLab provides built-in runners, which are agents that execute pipeline jobs, and the ability to configure custom runners based on specific requirements. GitLab is a tool that allows you to automatically deploy changes to production, or you can choose to do it manually [18].

GitLab is software that helps teams work more efficiently. It has many features like CI/CD, but also has features that allow teams to use other tools like code review, issue tracking, and project management. GitLab also supports integration with other tools to make it easier for teams to use them.

2.5.3 Travis CI

Travis CI is a continuous integration (CI) platform that allows software developers to automatically build, test, and deploy their projects. This helps teams ensure that their code is always in a releasable state, and bugs and issues are found and fixed early in the development process [19].

Travis CI integrates with various version control systems, including GitHub, GitLab, and Bitbucket, allowing developers to easily set up CI workflows. When a project is connected to Travis CI, the platform automates the build and test process every time changes are pushed to the version control repository. The process of building and testing Travis CI is defined by profiles, often called "Travis Profiles", which define the actions to be performed for a given project. The file can contain instructions for installing dependencies, compiling source code, running tests, and deploying the code to different environments such as development, test, and production.

Travis CI supports a wide range of programming languages and technologies, including popular frameworks such as Ruby on Rails, Node.js, and Django. It also integrates with many third-party tools and services, such as databases, cloud hosting platforms, and code coverage tools, to provide software teams with a comprehensive CI solution. One of the main advantages of using Travis CI is its scalability and reliability. The platform is designed to handle large and complex projects, with the ability to run multiple versions and tests in parallel on multiple machines. This helps teams save time and reduce the time it takes to complete a build or test cycle. Travis CI also provides powerful reporting and analytics capabilities to help teams monitor the health of their applications and track their progress over time. Teams can see real-time information on build, test and deployment status, as well as detailed performance and usage reports [20].

In general, Travis CI is a highly capable and adaptable CI platform that can greatly assist software teams in enhancing the efficiency and dependability of their software delivery pipelines. Whether you belong to a small development team focused on a single application or a large enterprise organization with numerous projects and development teams, Travis CI can effectively aid you in streamlining your software delivery process and guaranteeing that your code is consistently ready for release.

2.5.4 Circle CI

CircleCI is a continuous integration and continuous delivery (CI/CD) platform that helps software teams automatically build, test, and deploy applications. The platform enables developers to quickly and easily create end-to-end and repeatable build processes, simplifying collaboration and increasing the speed and reliability of software delivery[21].

With CircleCI, developers can define a set of instructions (or "workflow") that describe how to build, test, and deploy applications. It enables teams to automate tasks including compiling source code, running tests, building and packaging applications, and deploying code to different environments such as development, test, staging, and production. CircleCI supports multiple programming language and frameworks, including popular technologies such as Java, Ruby, Python, and Node.js. It also integrates with various third-party tools and services, such as GitHub, Bitbucket, and Slack, to streamline the software development process.

One of the key features of CircleCI is its ability to run builds and tests on multiple machines in parallel. This can significantly reduce the time it takes to complete a build or test cycle and help teams release new features and bug fixes faster. In addition, CircleCI provides powerful reporting and analytics capabilities to help teams monitor the health of their applications and track their progress over time. Teams can see real-time information about build, test and deployment status, as well as detailed performance and usage reports [22].

Overall, CircleCI is a powerful and flexible CI/CD platform that helps software teams improve the speed and reliability of their software delivery pipelines. Whether you are a small development team developing a single application or a large enterprise organization with multiple projects and development teams, CircleCI can help you optimize your software delivery process and improve the quality and speed of your software releases.

2.6 Virtualization of software

The operation of software requires appropriate hardware that supports the architecture of the application. This involves having adequate computational resources, which necessitates the use of physical machines with the right configurations and network connections. However, managing many physical machines can become complex and costly [23].

To address this issue, the concept of application virtualization was introduced. This process enables an application to run as if it were on a target machine, but it is actually executing in a virtualization layer between the application and the host operating system (OS). In a normal environment, the OS controls access to the machine's resources. There are two modes of operation: user mode, in which applications execute, and supervisor mode, in which the OS operates. To run virtualization systems, the guest OS runs in user mode, while the virtual machine monitor executing it operates in supervisor mode. When the guest OS needs to

execute a privileged operation, the virtualization must emulate the corresponding command, creating an overhead [24].

Although the concept of virtualization appeared in the 1980s, the decrease in hardware costs led to migration from mainframes, resulting in virtualization fading away. However, with the growing complexity of modern environments, increased network connections, and issues related to security and reliability, virtualization has become a crucial tool in maintaining and reproducing the machines required to run software. The use of virtual machines enhances security by compartmentalizing environments, reduces costs by maximizing use of the host hardware, and minimizes the risk of failures affecting other virtual machines in the event of a single virtual machine failure [25].

2.7 Containers

The advent of virtual machines (VMs) improved the efficiency of server management, but it still presented issues such as performance overhead and slow start times. To address these problems, the concept of containers emerged as a flexible and low-overhead solution. Containers differ from VMs in that they do not require a full virtual machine to run an application, but instead use an image file that represents the application and its configuration, and a container that runs an instance of the image. This makes it easier to move and run applications on any infrastructure, without the need for additional software or dependencies. A container is a lightweight and portable unit of software that packages all the necessary components, including code, libraries, and configuration files, into a single, self-contained unit. This makes it easy to move and run applications on any infrastructure, including laptops, servers, and cloud platforms, without the need for any additional software or dependencies [26].

This helps reduce the overhead of having multiple operating systems, while having the isolation between environments needed to run these containers. Also, it makes the escalation of applications more quickly and easily, since it only requires starting just another container from a particular image, as it provides a flexible and scalable infrastructure that can be adjusted as needed. Containers are isolated from one another, which means that they can run independently and do not interfere with each other. This makes them ideal for microservice-based architectures, where applications are broken down into smaller, independent services that can be developed, deployed, and maintained separately [27].

2.8 Container Tools

The following sections demonstrate some of the tools that exist that allow for the creation of an isolated container environment to run images that contain the application and dependencies necessaries

2.8.1 Docker

Docker is a platform that allows multiple applications to run each one in a different and isolated environment. This is possible since it utilizes the concept of containers to virtualize individual setups, not only allowing for flexibility but also to standardize the setup required to run an application. A container is a standalone executable package that includes everything

needed to run a piece of software, including the code, runtime, system tools, libraries, and settings.

By distributing and testing the application using Docker, the developers have more reliability to know that the application will run in every end user that will use the application. In the figure 2.3 it is possible to see how the Docker works in terms of runtime and registry of images in the cloud.

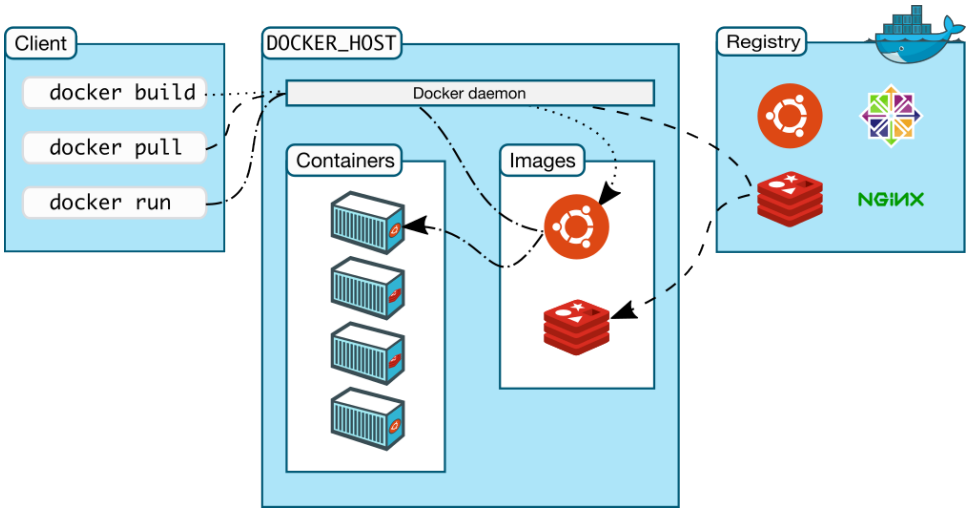

Figure 2.3: Containers

One of the key features of Docker is its use of images, which are snapshots of a container. Images are built from a set of instructions called a Dockerfile, which specifies the base image to use, the application code to include, and any additional dependencies or configuration [28].

It can also be used to orchestrate a service in the cloud, since the time it takes to start a new instance of an application is greatly reduced compared to manually launch the application into a new machine. And since it is lightweight, and it is possible to run multiple instances in the same host machine, the resources of a server can be used at his maximum [29].

The service offers also a way to configure how multiple services should run and communicate with each other, with the Docker Compose tool. With this tool, the possibility to configure all the services and network of an entire application leads to be ease to start an entire set of services in one command, in the right order.

2.8.2 Podman

Podman is an open-source tool that helps manage containers on Linux systems. Unlike Docker, it operates without a background process, making it simpler and more secure. It is compatible with Docker's commands and can run containers as a non-root user for added security. Podman can also work with Kubernetes and integrates with systemd for better control over container lifecycles. It follows industry standards, ensuring compatibility with other containerization technologies [30].

Podman extends its capabilities by integrating with Buildah, a container imaging tool. This separation of concerns between creating images and running containers increases its flexibility. Podman also offers network management, custom network creation, volume management for data persistence, and extensibility via plugins, enabling custom functionality and integration with various tools and services.

As a whole, Podman is well suited to environments where security, rootless containers, and Docker compatibility are a priority. It offers a versatile set of container management tools, making it an attractive choice for developers and system administrators working with containers on Linux systems [31].

2.8.3 Sysbox

Sysbox is an open source container execution and management utility that pushes the traditional boundaries of containerization technology. It has established itself as a standout solution in the container orchestration space, offering a sophisticated approach to address specific IT needs that require deeper integration with the host operating system.

A fundamental aspect that differentiates Sysbox from traditional container runtimes is its suitability for orchestrating system containers. These system containers have the unique ability to run system-level processes, including systemd and init, which are typically beyond the reach of standard containerization technologies. This makes Sysbox suitable for a variety of use cases where encapsulating system services and processes in containers is essential. The value of Sysbox is particularly evident in scenarios where system-level isolation, security, and process management are paramount. It offers a new approach by allowing these processes to run in containers, bringing the benefits of containerization, such as resource isolation and encapsulation, to the heart of the host system. This feature is of utmost importance in environments where running multiple system services requires strict control and isolation [32].

Sysbox further strengthens its status by offering seamless integration with the Systemd-Init system, the standard initialization and service management framework for most modern Linux distributions. The ability to run systemd in containers not only facilitates compatibility with existing service management practices, but also makes complex applications easier to manage and deploy [32].

In summary, Sysbox represents a paradigm shift in containerization technology, providing an advanced and versatile solution for orchestrating system containers. The ability to run system-level processes within containers not only extends the scope of containerization, but also strengthens the security, isolation, and manageability of system services and applications, making it an invaluable tool in today's landscapes. computer scientists.

2.9 Infrastructure as code

Infrastructure as Code (IaC) is a combination of the DevOps philosophy and version control system with a model that describes how the infrastructure as a system, like the computers and networks must be configured in order for the system works according to the specification. Is a component of the Continuous Delivery and "Integrated Configuration Management" concepts, and it helps to automatize the deployment by creating an environment to run the application that can be maintained by the developers, with easy rollback in case of failure

by using the version control system, and creating an agnostic way to recreate environments [33].

Normally is used declarative configuration files, which specify the desired state of the infrastructure, instead of, in an imperative way, declare what each node needs to have. These configuration files are used to automatically provision and manage the infrastructure, eliminating the need for manual processes and reducing the risk of errors. The configuration files can be used to manage a wide range of infrastructure, including servers, databases, networks, and cloud resources. The tools used for IaC can also be integrated with other tools and systems, such as CI/CD pipelines, to automate the entire infrastructure management process. This prevents the issues that occurs when trying to manage multiple servers, that is the possibility of inconsistency between them like configurations, dependencies, network connections, difficulty to create a new environment, and hard to maintain [34].

By using these tools, both developers and system administrators can work together to manage the requirements of a system, and how it can evolve during the development cycle of a product. This lead to a reduction in the cost and effort, since it removes manual component, increases speed by recreating a correct working environment more quickly, and the risk of error reduce the downtime of the product. Some advantages are the automation the process of configuring and deploying infrastructure, ensuring consistency and repeatability, version control of the infrastructure, easier to recover from failures and disasters, quickly scale the infrastructure, and better control over the infrastructure [35]. In the figure 2.4 it is possible to see the architecture of the IaC tools in general.

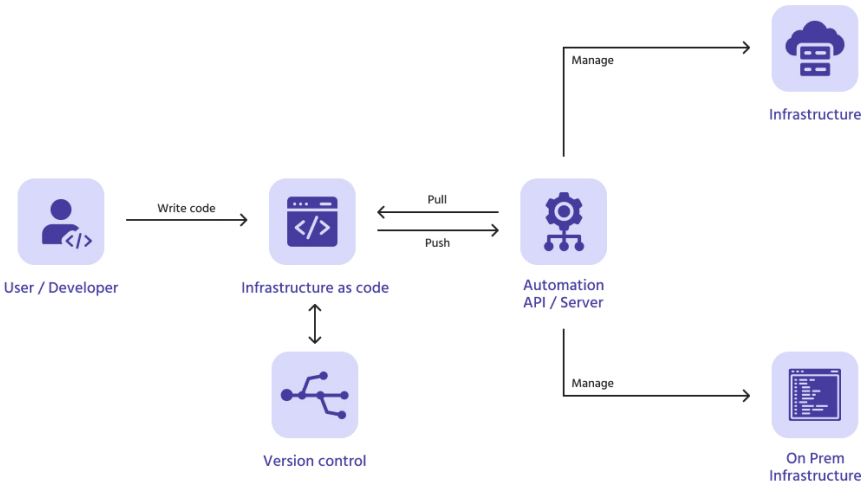

Figure 2.4: IaC

Implementing IaC can be challenging like the lack of technical expertise in using the tools, difficulty to integrate with existing systems, resistance to change by the development team, lack of standardization, security concerns about sensitive data management in the systems, resource constraints in terms of time and budget, and complexity to manage the infrastructure. The IaC started in the late 2000s with the launch of the CloudFormation, and since then the adoption of IaC has grown rapidly, since it tackles a problem that is managing a

cluster of servers and applications, and be able to scale it quickly. It's seen as a critical component of modern infrastructure management, and the use of others tools like containers and serverless computing can drive further adoption [36].

2.10 IaC Tools

In the following section is presented various tools that are available in the market, and what they have to offer to the developers in terms of state deployment management and migration of state when required.

2.10.1 Terraform

Terraform is a tool to track configurations of cloud and on-premise resources efficiently. This is possible by using human-readable configurations files that can be then versioned and reused, leading to a consistent workflow. Terraform supports a wide range of resource types, including virtual machines, databases, network configurations, load balancers, and more. It contains thousands of providers to manage types of resources and services [37].

To terraform have the workflow divided in three stages:

- Write, where is defined the resources needed for infrastructure, and the network to implement.
- Plan, where the platform creates an execution plan that will create, update or destroy based on the configuration and the infrastructure already present.
- Apply, where it performs the proposed operations in the correct order.

One of the key benefits of using Terraform is that it provides a unified and reusable way to describe your infrastructure, making it easier to manage, version, and track changes over time. It also supports state management, where it maintains the current state of the resources that has created, and uses that to determine what changes need to be made, modules where it is possible to reuse code, and workspaces, providing isolation between different parts of the infrastructure.

2.10.2 Ansible

Ansible is an open-source software platform for automation and configuration management. It provides a simple and efficient way to automate repetitive tasks, manage complex deployments, and orchestrate multi-tier infrastructure. Ansible uses a declarative language called YAML to describe the desired state of systems and applications, making it easy for even non-technical users to automate complex tasks. Ansible can automate a wide range of tasks, including the provisioning of servers, the configuration of software, and the deployment of applications. It supports multiple operating systems, including Linux, Unix, and Windows, and can automate tasks on remote systems without the need for an agent to be installed on each machine [38].

One of the key benefits of Ansible is its simplicity and ease of use. It uses a push-based approach to automation, which means that changes are pushed out to the target systems, rather than being pulled from a central repository. This makes it easy to automate a wide range of tasks, even in large and complex environments. Ansible also provides robust error

handling and reporting capabilities. It can detect and report errors in real-time, and provides detailed information about the state of systems and applications, making it easier to troubleshoot and resolve issues [39].

In addition to its core automation capabilities, Ansible also integrates with a variety of other tools and services, including cloud providers, databases, and source control systems. This makes it easy for teams to automate complex tasks that span multiple systems and applications, and to integrate Ansible into their existing workflows and processes.

2.10.3 Chef

Chef is a popular open-source tool used in DevOps and IT operations for automating the configuration, provisioning, and management of infrastructure. It uses a central repository called the Chef Server to store configuration data, policies, and cookbooks. Nodes, which can be physical servers, virtual machines, containers, or cloud instances, run the Chef client to fetch and apply configurations. Cookbooks are the main components in Chef automation and contain recipes, attributes, and resource definitions. Recipes define specific configuration tasks using resources and actions. Attributes allow customization of recipe behaviour, while roles simplify node configuration by grouping related recipes and attributes together [40].

Administrators and developers use the Chef Workstation to create and manage configurations related to Chef. Chef ensures that configurations can be applied multiple times without causing harm by only making necessary changes. The Chef Client on nodes communicates with the Chef Server to apply configuration instructions and maintain the desired state. Chef can integrate with cloud platforms and container orchestration tools to provision and manage resources. Using Chef allows organizations to automate infrastructure management, maintain consistency, and reduce errors. Its flexibility and automation capabilities make it valuable in modern IT environments.

2.11 Audit

The concept of audit plays a crucial role in ensuring the proper functioning and maintenance of a software application. It provides administrators and relevant business users with access to critical information regarding the reasoning behind a system's actions, especially in cases where these actions have significant implications for the company. However, many software systems lack efficient audit tools that are easy to use and provide the necessary information to all users. The implementation of audit functions in a software system offers several advantages, including increased control over the system, quick access to information during system malfunctions, and the ability to set up alerts and take action based on specific system activities [41].

There are various types of audit tools available, including system monitoring tools that monitor the performance and resource usage of the system and log management tools that aggregate and summarize log data from various sources. For instance, a study was conducted on tools like Moodle, a learning management system, and how it can help administrators understand student activity during coursework [42]. Despite the existence of logs, there are still several challenges associated with log management, such as the rapid growth in log size, difficulties in searching for logs, and cross-referencing logs across multiple components of a system.

Logging, when used as a fundamental aspect of the software development process, starting from the continuous integration and continuous delivery phase, allows the companies to improved customer support and reduced time and cost effort in fixing bugs during the production phase. Normally the logs contain information like performance evaluation, user actions, system actions, errors, crashes, and communication failures between components. Logs should be descriptive and provide contextual information that can assist in tracking a user session and actions across the logs [43].

2.12 Audit Tools

In this section is described some of the tools that are available in the market that allows to monitor and audit applications and systems, that can be used by developers and/or operations teams.

2.12.1 DataDogs

Datadog is a cloud-based monitoring and analytics platform that helps organizations gain visibility into the performance of their applications and infrastructure. It is designed to provide comprehensive observability across complex, dynamic, and distributed environments [44].

It offers features like agent bases monitoring, dashboards, alerts, Application Performance Monitoring (APM), security monitoring, etc. Furthermore, it is a versatile platform that allows the DevOps teams to acquire new information about the system behaviour and ensure the service level objective. Datadog's pricing is typically based on a subscription model, and the cost can vary depending on the specific features, usage, and scale of the monitoring and observability needs [44].

2.12.2 Grafana

Grafana is an open-source platform for monitoring, observability, and data visualization. It is commonly used to create interactive and customizable dashboards that display real-time data from various sources. Some main features include data visualization, a wide range of data sources, plugins that can extend the functionalities of the main software, alerting, templates and dashboards sharing. Grafana is widely used in DevOps, IT operations, and engineering teams to monitor infrastructure, applications, and services. It is known for its flexibility and ability to connect to a variety of data sources, making it a valuable tool for gaining insights into system performance and facilitating data-driven decision-making [45]. Grafana is mostly free and open-source, however if offers an enterprise version with a premium support.

2.12.3 Netwrix

Netwrix Corporation is a cybersecurity and information security software company that specializes in solutions for data security and governance. The company's products are designed to help organizations monitor and protect sensitive data, detect security threats, and ensure compliance with various regulations and standards.

Netwrix is used by a wide range of organizations, including enterprises, government agencies, healthcare providers, and financial institutions, to enhance their data security and compliance

efforts. The specific products and features offered by Netwrix may vary, so it's advisable to visit their official website for the latest information on their solutions and services [46].

2.13 Security

In today's interconnected and digital landscape, ensuring the security of software systems is of paramount importance. The rapid evolution of cyber threats demands a proactive and comprehensive approach to security throughout the software development lifecycle. This chapter explores the multifaceted domain of secure software development, delving into methodologies, practices, and tools that enable organizations to create resilient and secure software solutions.

The Secure Development Lifecycle (SDL) is a systematic approach that integrates security practices into the software development process. It typically comprises phases such as requirements analysis, design, coding, testing, deployment, and maintenance. Each phase incorporates security checks and validations to mitigate vulnerabilities. Understanding and defining security requirements is a crucial initial step in the SDL. Collaborative efforts involving security experts, developers, and stakeholders help identify potential threats, security controls, and compliance needs. The Confidentiality, Integrity, and Availability (CIA) triad forms the core of security objectives: ensuring confidentiality, maintaining data integrity, and guaranteeing system availability. Secure software development aims to uphold these principles to safeguard sensitive information and critical systems [47].

The principle of "Security by Design" emphasizes embedding security considerations into the very foundation of software architecture and design. It involves threat modelling, risk assessment, and the selection of appropriate security controls at the earliest stages of development [48]. One of the most common attack vectors is input manipulation. Secure coding practices involve thorough input validation and sanitization to prevent vulnerabilities like SQL injection, cross-site scripting (XSS), and command injection [49]. Implementing strong authentication mechanisms and fine-grained authorization controls is essential to prevent unauthorized access and data breaches. Regular code reviews and static analysis tools identify vulnerabilities early in the development process. This proactive approach assists in addressing security issues before they reach production [50].

Static Application Security Testing (SAST) is the analyses of source code, bytecode or binary code in order to detect patterns and structures that match bad coding practices, security issues or even known vulnerabilities. This is done without running the code, so it can be performed as early as possible in the development of an application. This allows the developers to improve overall code quality, by getting early feedback of the code that need to be changed and fixed, so it does not raise problems later on. Although this tools can miss important vulnerabilities because they don't know the context on which the code will be executed, it still brings value to use it, since it can also be incorporated nicely in a CI/CD pipeline [51].

Other type of analysis is the Dynamic Application Security Testing (DAST), where the application is put in a run state, to simulate real-world attacks. The tool is responsible to interact with the application, testing possible vulnerabilities based on inputs to the system. This brings another perspective of automatic analysis, since it simulates possible interactions with the system, sometimes even exhausting possibilities very fast in order to test the limits of the application, and how it handles such scenarios. It allows detecting runtime-specific

issues, misconfigurations and vulnerabilities related to the application's runtime environment [52].

Finally, other type of tool is the Interactive Application Security Testing (IAST), where is combined elements of SAST and DAST, such as the application is instrumented during runtime and analysed both runtime behaviour and source code flow. This offers precise identification of vulnerable code, with the cost of having more performance overhead, and may not run through all code paths [53].

They are other type of tools, like Software Composition Analysis, where the libraries used by the application are investigated for known vulnerabilities or licensing issues. Fuzz Testing, where is provided to the application malformed or unexpected inputs to an application to identify vulnerabilities arising from incorrect handling of data. Runtime Application Self-Protection (RASP) involves embedding security controls into the application itself to detect and prevent attacks during runtime [54].

2.14 Security Tools

In this section is demonstrated some of the tools that can be used to analyse the application in order to find known vulnerabilities.

2.14.1 OWASP ZAP

OWASP ZAP is an open-source security testing tool specifically designed for dynamic application security testing. As a part of the Open Web Application Security Project (OWASP), ZAP is a versatile and powerful tool that assists in identifying vulnerabilities in web applications through simulated attacks. Some of his functionalities are:

- **Proxy Mode**: Is allows intercepting proxy, allowing to view and modify the requests and responses between your browser and the target application. This can be used to explore and test various input vectors, headers, and parameters.
- **Active Scanning**: ZAP offers active scanning capabilities, where it automatically analyses the application by sending malicious payloads and probing for vulnerabilities such as cross-site scripting (XSS), SQL injection, and more.
- **Passive Scanning**: In passive scanning mode, ZAP monitors traffic and flags potential vulnerabilities without actively sending payloads. This helps identify security issues without disrupting the application's functionality.
- **Automated Spidering**: ZAP includes a spidering feature that automatically navigates through the application's links to discover and map out different components, pages, and endpoints.
- **Authentication and Session Management Testing**: ZAP supports testing of authentication mechanisms and session management. It allows you to capture and replay authentication sequences to assess their security.
- **Contextual Analysis**: ZAP provides contextual analysis, allowing you to define specific contexts for testing, such as specific areas of the application, to focus on certain vulnerabilities.

- **Extensibility and Automation**: ZAP is highly extensible and offers an API that enables users to automate scans, customize reports, and integrate the tool into their CI/CD pipelines.
- **Vulnerability Reporting**: ZAP generates detailed reports that highlight discovered vulnerabilities, their severity, and recommended remediation steps. This aids in communicating findings to development teams.

In summary, an open source tool that helps identify a wide range of vulnerabilities, having all automated, that can help developers understand how attacks occur, and given recommendations to address vulnerabilities, is a robust tool to use as a DAST tool and being incorporated in a CI/CD pipeline [55].

2.14.2 Burp Suite

Burp Suite is a DAST tool, developed by PortSwigger, that helps identify vulnerabilities in a web application by simulating various types of attacks [56]. This software comes with a range of features, such as:

- Proxy: The Proxy component acts as an intermediary between your web browser and the target web application. It allows you to intercept and manipulate HTTP requests and responses, which is crucial for understanding how an application works and for finding vulnerabilities. Security professionals use the proxy to analyse and modify requests and responses in real-time.
- Scanner: The Scanner automates the process of identifying security vulnerabilities in web applications. It performs various types of scans, such as vulnerability scans, crawling, and scanning for common issues like SQL injection, cross-site scripting (XSS), and more. The results are presented in a user-friendly format for analysis and further investigation.
- Intruder: The Intruder is a powerful tool for automated attacks against web applications. It allows users to customize and automate attacks by replacing specific parts of HTTP requests with payloads, making it useful for finding vulnerabilities related to input validation, parameter manipulation, and more.
- Repeater: The Repeater tool enables security professionals to send modified requests to the target web application repeatedly. This is useful for testing how the application responds to different inputs or to confirm the existence of vulnerabilities discovered during manual testing.
- Sequencer: The Sequencer tool is used for analysing the randomness and quality of tokens generated by the web application, which can be crucial for assessing session management and security features like anti-CSRF tokens and session identifiers.
- Decoder: The Decoder helps with various encoding and decoding tasks, which are essential when dealing with data manipulation, encryption, and obfuscation within web applications. It can decode different encoding schemes like base64 and URL encoding.
- Comparer: This tool allows users to compare two HTTP requests or responses to identify differences, which can be helpful when analysing the effects of different inputs on the application's behaviour.

- Extender: The Extender component allows users to enhance Burp Suite's functionality by writing custom extensions in various programming languages like Java or Python. This is useful for automating specific tasks or adding new features tailored to your testing needs.

2.15 ERP

An Enterprise Resources Planning (ERP) is a software that integrate several pieces of components, using a common database, and sharing the data between the various functional areas [57]. A representation of the various modules that can compose an ERP is demonstrated in the image 2.5

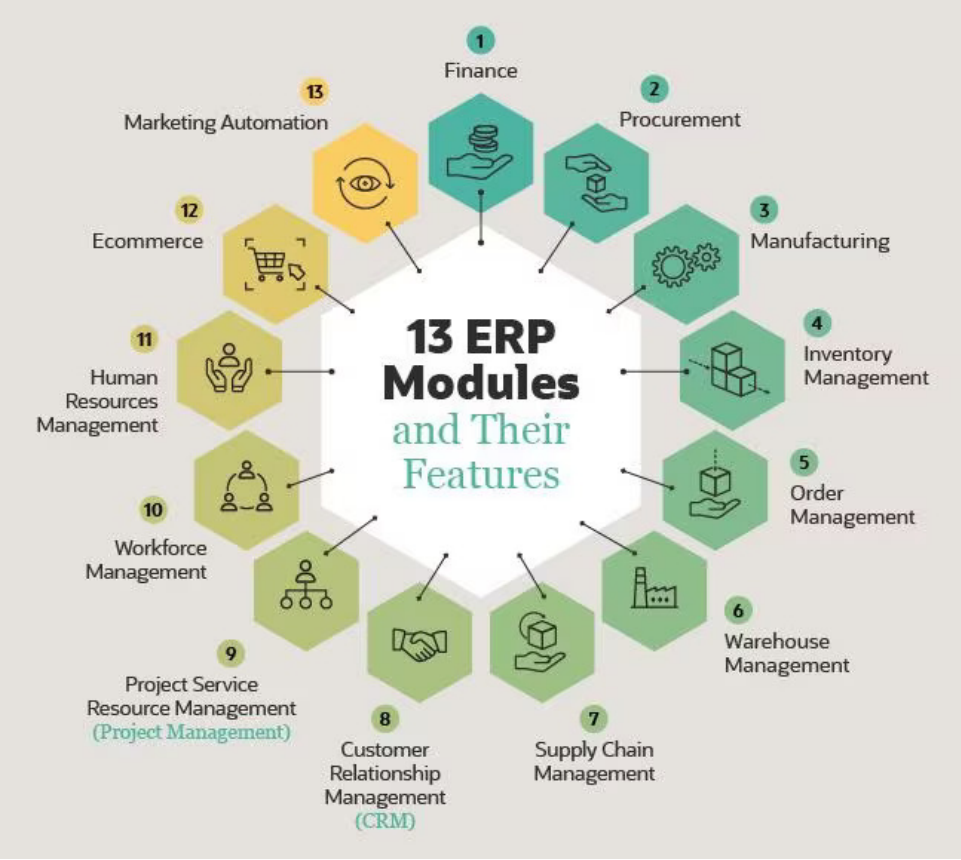

Figure 2.5: Various of the Modules integrated in ERPs

It's origin remotes to the Material Requirements Planning (MRP) and Manufacturing Resource Planning (MRP II), where both are closely related concepts in the history of ERP systems. From this base, ERP utilizes the knowledge acquired from MRP and MRP II to develop more sophisticated systems that are capable of managing all aspects of a business, from finance and accounting to human resources and supply chain management [58].

The ERPs popularity has grown over the last years, since it proven to increase the efficiency and speed of the processes that the companies had. The market that is available to use ERPs in their business, with more solutions available to choose, and the need to digitalize the companies to reduce paper use and getting more effectiveness in their processes are the leading reasons why companies invest in ERPs implementations [59].

However, there are reports that many of ERP's implementations get a significant customization, indicating that the all-in-one solution, that is the major point to implement an ERP, is not good enough to sell it to every company in the market, requiring an extra effort. Choosing the right system can be very challenging, since it not only require a system with all the functionalities that are needed, but also a vendor that fully understands the need of the market the client is in [60].

Despite the ERPs history, these systems can be applied to many type of sectors, like the education industry, where the ERPs are more used to manage the academic and research facilities, staff and student data, etc. In the healthcare industry, the ERPs are used for tracking the medical supplies, patient's records, and the overall facilities [61]. In today's world the increase for the Cloud based solutions increased, since it allows for less cost for the companies not only in hardware but also in maintenance, since all the process is managed by the company managing the ERP. This allows for more flexible processes, as the software is available everywhere with less effort to put. [59].

The goals that are most important from the ERPs are cost savings, better performance metrics and improved efficiencies in business transactions. Most of the actual trends are cloud subscriptions, and IA and IoT technologies incorporation in ERPs [57]. Most of the companies implement ERPs is to shift from legacy systems to a new one, that is more standardized and with more support.

The market already have ERPs software established, some of these examples are the Oracle Netsuite, Sage, SAP or FinancialForce, among others [62]. The Oracle Netsuite for example have a diversity of pre-built solutions so the product their sell is versatile enough to cover most of the market available. It also has plans to migrate the new clients from their existing software to help the process of streamlining with the change of software, and indirectly the processes of a company.

In the case of the Sage, although they have variants for small and accountants companies, their solution for medium companies with the Sage X3 ERP offers functionalities as controlling the shop floor, quality control, fixed assets, procurement, customer support, etc. SAP offers solution which includes analysis of the data from the customer to help create new business models to guide the client to a new business structure that can bring more value to the company.

Chapter 3

Value Proposition

This chapter describes the process that was used in order to assess the value that can be derived from this dissertation project. First is described the concept models for innovation and opportunity analysis, by following the New Concept Development (NCD). Afterwards, the opportunity that the project intends to address is identified and analysed, with the perceived value that the stakeholders derive, and with the description of the value proposition and the business model the project proposes to. Finally, the requirements are analysed and selected with support from Function Analysis System Technique (FAST) and Analytic Hierarchy Process (AHP) methods, with the conclusions taken from this analysis.

3.1 New Concept Development

The development of a product must require an understanding between the proposed solution and the value derived from the customer. But, in order to create a solution, first is required to create an idea of a product.

The concept of innovation results from the unsatisfaction with the current state of the art, and is connected to the introduction and implementation of new ideas or processes, with the objective of generating new ways to perform tasks, improving the processes associated with it. This process of innovation can be characterized with the model propose by Peter A.Koen, represented in the 3.1, where it's divided into three segments [63]. This model describes all the process from the creation to the commercialization of a product, and it is expected to be followed sequentially.

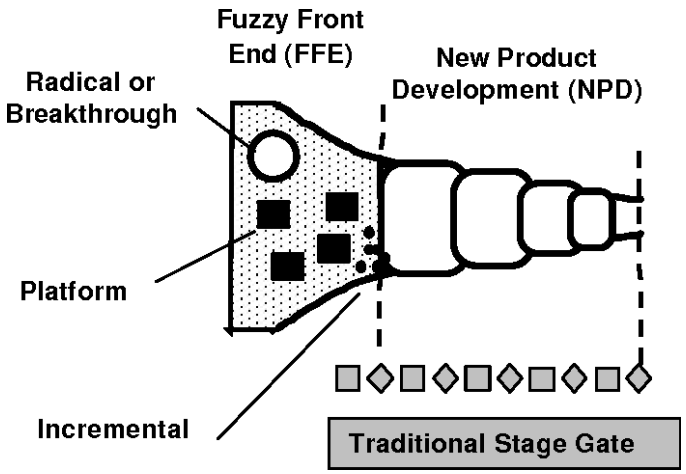

Figure 3.1: The Innovation Process

The first phase is the Fuzzy Front-End (FFE), that refers to the early stage of a product development, and is characterized by a high degree of uncertainty and ambiguity, as many decisions is still to be made for the product. This phase is critical, since it sets the rest of the project, and so the importance to gather information, generate and evaluate ideas, and making key decisions is very important. Afterwards is the New Product Development (NPD), where is performed the design, development and marketing of the product, in order to increase the market of a company, satisfying the need from the consumer. Finally, the commercialization process is when the company sells the product made, even if it's brand new or only a change to an existing one.

Since the first phase is very difficult to stipulate the decisions to be made, the NCD is a structured approach for the FFE that is based on the idea that the planning and management of this phase can help navigate the people through the uncertainty and ambiguity that is normally associated with this stage. The NCD can be described as a model that consists of five stages, like represented in the figure 3.2:

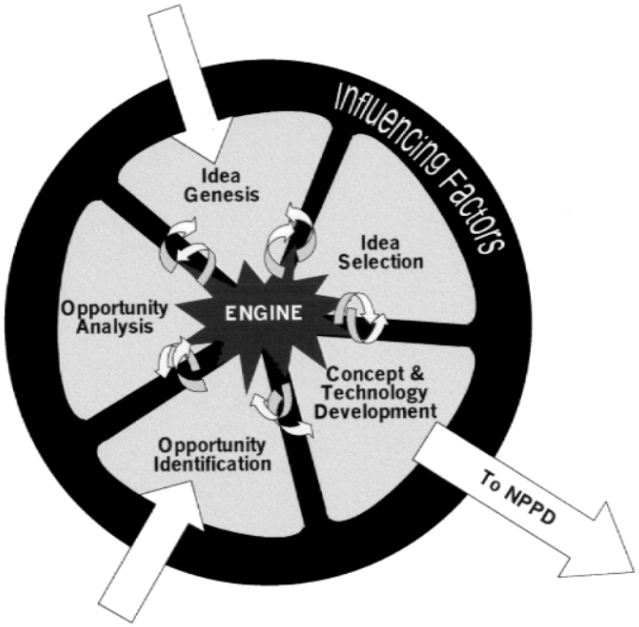

Figure 3.2: The NCD Model

- Idea generation: First phase of NCD, where ideas are generated. It can include market research, brainstorming sessions, and others activities to stimulate creative thinking.
- Idea selection: After generating the ideas, the team evaluate and select the most promising idea.
- Concept Development: The idea selected is refined, elaborating more detailed concept designs and prototypes.
- Business Analysis: The team generates an assessment of the potential market, target customers and financial viability.
- Implementation Planning: Is generated a plan to bring the product to market.
- Engine - Represents the organization strategy to impulse and work the five elements described before.

- Influencing Factors - Are external factors that are outside the organization environment, but can shift the innovation process.

By applying the New Concept Development, it's possible to analyse different characteristics of the problem, for both the external factors and the organization component, leading to the concept that will lead to the New Product Development phase of the innovation process. Thus, in the next sections, is presented the application of this model in order to generate of ideas and the respective selection.

3.2 Opportunity Identification

Development is always changing, with new technologies and tools to help developers make software faster, safer and with fewer errors. From this evolution, it is expected that the products turns out to be more robust. However, the products are also constantly changing, becoming more complex that ever, since the market needs shift when the previous needs are accomplished.

The maintenance of the products are becoming more challenging to be performed, and so the automation of the operations that support the development and deployment of the product are being more searched than ever. Take for example the Google Search Trend for the word "DevOps", shown in the image 3.3.

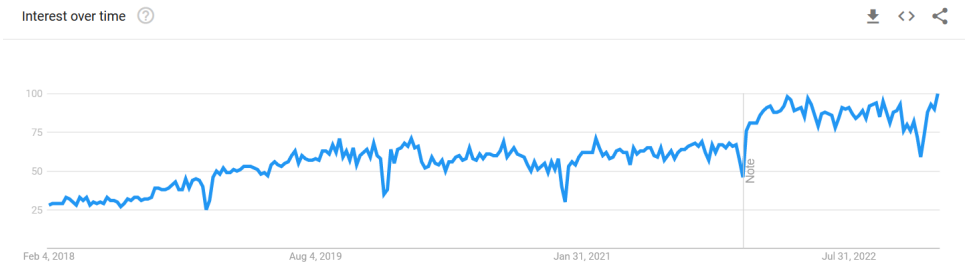

Figure 3.3: DevOps search on Google

So, the importance of implementing a good pipeline to build, test and deploy the tool, and audit the software to be faster to track bugs and usage of the system is a new priority settling in the software companies. And since applications are becoming generally more complex, with more integrations between different tools, and the need to scale up the application when is required to maintain a good service, is making companies turn to automation of tasks such a testing and deploying, in order to reduce the time and human resources needed. This allows the developers to have more time to make more functionalities, and the operations team to be able to deploy new changes and adjust the infrastructure as needed.

Incorporating DevOps in any software cycle can lead to the final product being more robust, thus requiring less support cost. This is specially important for complex applications like an ERP system, that contains several modules with information integrated, since the errors are detected during development, and the number of errors occurring on the client side are reduced [64].

3.3 Opportunity Analysis

The NCD model serves to demonstrate the potential and viability of an opportunity in the market. For that, a set of articles and reports are used to describe and explain why a given opportunity is worth the time and investment.

A Google Report from 2022 about the state of DevOps and his usage in enterprise solutions development states that the number of companies that are relying on the DevOps process to be more responsive and more resilient to product failure is increasing. About 69% of the respondents report that can deploy and take new changes into production between once per week and once per month, that can restore a service between one day and one week, and their average failure rate is around 16%-30%. This lead to teams with less burnout, less susceptible to errors, and less unplanned work [65].

In the same report it is stated that the usage of public cloud increased 36%, and that the use of public cloud help the reliability and time to recover of the product. This means that the investment in some cloud provider, or even private, can relieve the pain from the company and their clients, and also have more availability time.

Another report from Microsoft describes that the teams that implement DevOps have easier process to manage the infrastructure of the system in comparison to the others companies [66].

Although DevOps is a concept with already some established knowledge and business practical examples, it's still a challenge in the market to use this methodology at full, specially the cases where there is already a legacy product that requires several changes in place in order to fit in the DevOps. This shows that there is an opportunity to design and develop a DevOps process that can help in the implementation of a product like an ERP, and that can be effectively measured to prove the efficiency of the implementation when using this methodology.

3.4 Perceived Value

The perceived value is the perceived utility from a stakeholder, and the price associated with it. In terms of software development, the value generated for the customers is what can the system do and if it corresponds to all the needs required for a business.

To verify what value can be made from developing an ERP solution with the use of DevOps and Audit tools it is relevant to understand all the stakeholders involved, and what value can be generated from it. In the table 3.1 is demonstrated what is the value that the solution can bring to the stakeholders of this project.

3.5 Value Proposition

The Value Proposition Canvas is a framework that was developed by Alexander Osterwalder, and with the objective to describe the correlation between customers needs, and the benefits from using the product that is being discussed. It provides a framework for testing and refining the value proposition, enabling organizations and individuals to iteratively improve their offerings and better understand the needs and motivations of their customers. The canvas is divided into two sections, one representing the customer and the other representing the product or service [67].

Stakeholders	Benefits	Sacrifices
Developers	Can produce more product changes, and write tests and implantation configurations with less effort.	Loss of control, which can lead to understand less about how the system is implanted.
	Audit tools also help developers to better understand in which part of the system the changes they are making are causing wrong execution from the system, since this tools will warn the developer when the system detects something stop working.	The effort associated with implementing changes increase partially because of the effort required to also implement the log necessary in the system.
	Using some current technologies in the market lead to better support, taking the advantages of this tools like their performance, static analysis, features like garbage collection and such that prevents bugs, etc.	Adaptation from older and stable languages.
Customers	Software that can be proven more robust will lead to fewer errors, leading to more confidence about the system and what he can do. Products prepared for fast iterative implementation and deploy process lead to faster changes made to the product, and the possibility for automation the upgrades lead to easier process to maintain the clients. User experience is expected to be fast.	A product that change too quickly can lead to difficulty in using the system.

Table 3.1: Perceived Value by each Stakeholder

To apply the Value Proposition Canvas, the first step is to identify the customer segments being targeted, that includes defining the characteristics, needs, and motivations of the customers, as well as their pain points and the problems they are trying to solve. The second step is to define the customer problems and needs being addressed, covering the pain points and the jobs they are trying to get done. The third step is to define the unique value being offered to solve those problems, including the benefits and unique features of the product or service. The fourth step is to define the channels through which the value proposition is delivered, including the touchpoints and interactions with the customer.

The Value Proposition Canvas has become a widely used tool for product design, development, and marketing. Is a strategic tool used to help organizations and individuals create compelling value propositions for their products or services. One of the key conclusions that can be drawn from using the Value Proposition Canvas is a deeper understanding of the customer and their needs, enabling organizations to create more compelling value propositions that effectively address their customers' problems. Additionally, it can help organizations to identify and optimize their key differentiators, making it easier to stand out in a crowded market and differentiate themselves from their competitors.

In this dissertation the purpose is to create a good support development process that removes some manual review and pain from the developers when changing the product. So, one of the stakeholders are the developers that make part of the team developing the ERP, and

the value proposition is that DevOps will remove some hard work of the developers when included in the software development cycle. The other stakeholder are the customer that will be using the ERP system.

Considering the stakeholders mentioned above, it was designed the canvas presented in the figure 3.4, that describes in the details what the project have to offer.

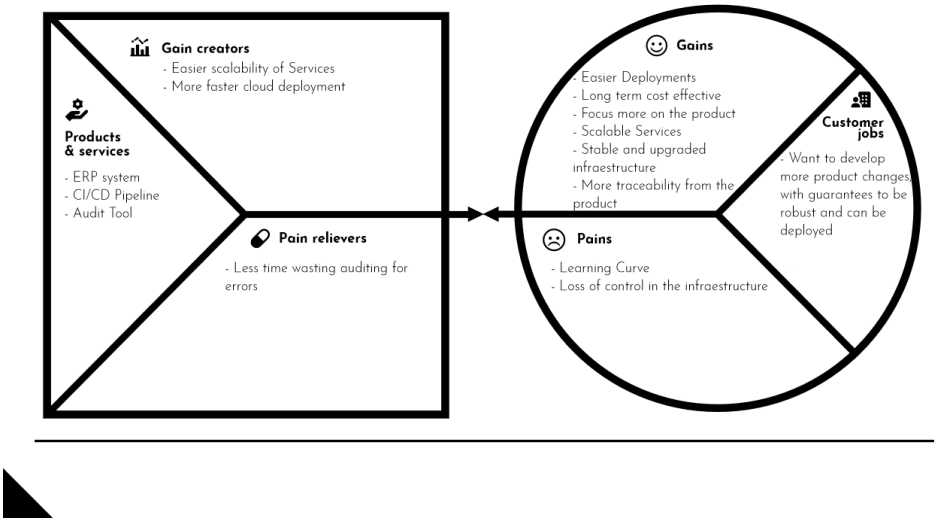

Figure 3.4: Value Proposition

3.6 Business Model CANVAS

The term business model refers to plan of a company to generate value and profit. The Business Model Canvas is an important management and Lean Startup tool used to develop and visualize the different elements of a business model. It was developed by Alexander Osterwalder and Yves Pigneur in their book "Business Model Generation", and has since become a widely used tool for entrepreneurs, startups, and established companies looking to develop and improve their business models [67].

It denotes the products or services the company want to develop and sell, in which markets, and the costs in order to implement it. Business models are important for both new and established businesses. They help new, developing companies attract investment, recruit talent, and motivate management and staff. The canvas is designed to be flexible and iterative, allowing organizations to quickly and easily test and refine their business models as they gain new insights and market feedback.

It is a visual representation of a company's asset, both physical and logical. To apply the Business Model Canvas, first is necessary to identify the characteristics of the customer segments being targeted, with the motivations and pain points of the customers, and identify the unique benefits and features of the product or service being offered to the customers. After that, is to consider the channels that will be used to deliver the product to the client, and the customer relationships, including the type of relationship being established with the

customer and the value of it. Next is required to identify the revenue streams and the pricing strategy being used, and the key resources, including the tangible and intangible assets that will be needed to deliver the value proposition. Finally, is defined the key activities, with the critical processes and operations of the business, the key partnerships, including the strategic relationships and collaborations required to deliver the value proposition, and the cost structure. By visualizing the business model in a structured and comprehensive way, organizations can quickly identify opportunities for improvement and areas for optimization, enabling them to make informed decisions about their strategy and direction.

In this project, although is an auto-proposal project, it is still possible to refer some principals business aspects from it. The business model canvas is represented in the figure 3.5, that describes the value behind the solution proposed.

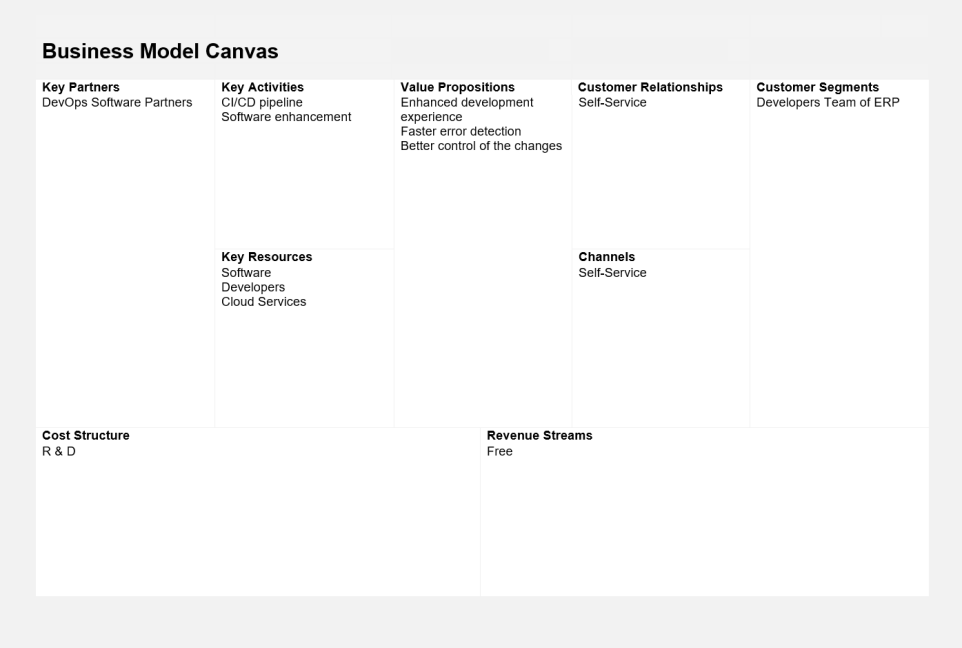

Figure 3.5: Business Model Canvas

3.7 Value Analysis Approaches

In order to develop an effective solution, it is necessary to select the right ideas, which is very challenging to choose properly. Since it is a crucial stage to conduct the development of a solution, a set of different techniques and methods can be used to guide this process of choice.

Therefore, a set of criteria must be established to determine what is most important, and once this is done, a list of potential approaches can be drawn up, and with the help of a methodical implementation, it will be possible to decide which solution should be used to implement the new system.

For this project it will be used two different methods to evaluate the ideas that can possible be used for this project:

- FAST design
- Analytic Hierarchy Process

The following sections will provide more detail on each method used to reach the decision about which idea should be selected.

3.8 FAST

The Function Analysis System Technique (FAST) technique is a graphical representation of the functional decomposition of a product or service, breaking it down into its constituent parts and functions. It starts with the identification of the main function of the product or service, which is represented by a box at the top of the diagram. This main function is then broken down into smaller, more specific functions, represented by boxes below the main function, and so on until all the functions of the product or service have been identified [68].

Is a tool used in value analysis to analyse and evaluate the functions of a product or service in order to identify areas for improvement and cost reduction. It supports the communication between team members by sharing common knowledge, defining and clarify the problem, identifying the basic function of the project.

This diagram helps thinking the problem objectively, identifying the scope of the project, and verify if a proposed solution achieves the needs of the project. It requires majoring questions "How?" and "Why?", where the "How" defines the bases from where it's reached the scope the study, while the "Why" defines the motives behind the scope of the study.

This representation provides an overview of the process under analysis, allowing users to track the critical path. The critical path represents the most important functions of the process, and this tool also shows supporting functions that, even though they are not part of the critical path, can adversely affect the normal course of operations and thus the full data migration process. There are tools available for almost every operation in this complex process that can make implementation easier.

By this principle, it was developed the diagram illustrated in the figure 3.6 that shows the reason behind the implementation, and how it will be done.

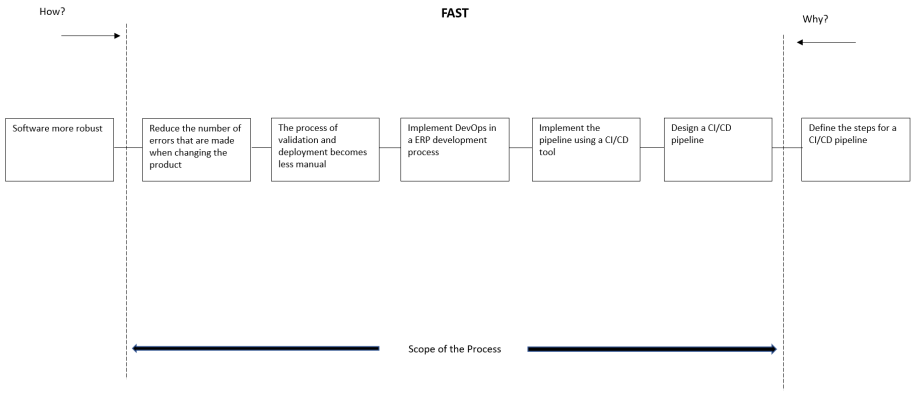

Figure 3.6: FAST Diagram

In the diagram it is described that, in order to implement a DevOps methodology in an ERP development process, it is required to specify and design what is needed to be done in the pipeline, and how will it be structured in order to reach the goals of the team. The main priority is to automatize the integration, and after that the deployment process. In the

automation process, the primary objective is to implement a continuous integration pipeline, and after that is completed, then add the deployment of the application into the pipeline. In this part is also necessary to analyse the application containerization.

3.9 Analytic Hierarchy Process

The Analytic Hierarchy Process (AHP) is a mathematical decision-making method that allows a decision maker to structure and examine complex decision problems. It was first introduced by Thomas L. Saaty in 1980, and it has since been used in a variety of fields, including business, engineering, public administration, and social sciences [69].

AHP involves the use of pairwise comparisons between decision criteria and alternatives, and the assignment of weights to the criteria and alternatives. These weights are used to calculate the overall importance of each alternative in relation to the decision problem.

The process of using AHP typically consists of several steps:

- Define the problem: The decision maker must clearly define the problem to be solved and determine the objectives of the decision.
- Identify the criteria: The decision maker must identify the criteria that will be used to evaluate the alternatives.
- Develop a hierarchy: The decision maker must arrange the criteria in a hierarchy, with the most important criterion at the top. This helps to clarify the relationships between the criteria and provides a framework for the analysis.
- Perform pairwise comparisons: The decision maker must perform pairwise comparisons between the criteria and between the alternatives, in order to determine their relative importance.
- Calculate weights: The pairwise comparisons are used to calculate the weights of the criteria and alternatives.
- Evaluate alternatives: The weights are used to evaluate the alternatives and determine which one is the best choice.

The application of the AHP allows to determine the alternative that the best choice for the decision problem. This conclusion is based on the relative importance of the criteria and alternatives, as determined by the decision maker through the pairwise comparisons [70].

With this, the first step is to build a hierarchical decision tree, where is present the problem, the criteria considered, and the alternatives available. The criteria considered for this project are:

- Security - the tool should be very secure, since the project have a special attention to the security aspect in the programming.
- Cloud Integration - the tool should present an easy process to install and/or deploy on the cloud
- Plugins - the tool should have a good plugin infrastructure.
- Reports - the tool should have good reports about metrics and other statistics about the pipeline and the execution of it.

- Price - The cost to use and maintain the tool is important.

The alternatives that are considered in this decision tree are the following:

- Jenkins
- Gitlab
- Travis CI
- Circle CI

After the definition of the decision tree, the next step is the creation of the pairwise comparison between the criteria, to determine which one are more important than others. For this step, the scale proposed by Saaty was used, and the result of this evaluation can be seen in the table 3.2.

	Security	Cloud Integration	Plugins	Reports	Price
Security	1	1	1	3	1
Cloud Integration	1	1	1/5	5	1
Plugins	1	5	1	5	3
Reports	1/3	1/5	1/5	1	1/5
Price	1	1	1/3	5	1
TOTAL	4 1/3	8 1/5	2 3/4	19	6 1/5

Table 3.2: Criteria pairwise comparison

The third step is the calculation of the relative priority of each criterion. This is possible by normalizing the comparison matrix and calculate the arithmetical average of each value. The table 3.3 demonstrates the values that originated the priority vector.

Security	0.2076
Cloud Integration	0.1701
Plugins	0.3907
Reports	0.0519
Price	0.1798

Table 3.3: Priority Vector

After this, is necessary to calculate the consistency ratio, to verify if the relative priority are consistent, thus proving that the matrix is consistent, and the values can be trusted. The value obtained in this case was 0.08, meaning the comparison matrix is effectively consistent.

The fifth step is the development of comparison matrix for each criteria, considering all the alternatives. In the appendix A it is demonstrated the calculations for each criteria.

Finally, in order to reach which tool should be considered to use in this project, is necessary to calculate the priority vectors of each criteria, and then multiply with the criteria relative priority. The result are shown in the table 3.4.

In this case, based on the results obtained, the most appropriate tool to implement a CI/CD pipeline, in comparison with the others tools, is Gitlab.

	Security	Cloud Integration	Plugins	Reports	Price	Vector	**Priority**
Jenkins	0.0979	0.1072	0.375	0.3042	0.4167	0.2076	0.2758
Gitlab	0.3226	0.1204	0.25	0.3875	0.4167	0.1701	0.2801
Travis CI	0.4060	0.4112	0.125	0.1292	0.0833	0.3907	0.2247
Circle CI	0.1735	0.3612	0.25	0.1792	0.0833	0.0519	0.2194
						0.1798	

Table 3.4: Alternatives composite priority

Chapter 4

Technologies Used

In this chapter is described all the technologies that were used during all the phases in the project as a support for the implementation.

4.1 Git

Git is a distributed version control system that enables developers to track changes in their source code, collaborate with others, and manage code history efficiently. It allows users to create branches for development, merge changes, and maintain a complete history of code revisions, making it a crucial tool for software development and collaboration [71].

4.2 Visual Studio

Visual Studio is an integrated development environment (IDE) developed by Microsoft. It provides a robust set of tools for software development across various programming languages and platforms. With features like code editing, debugging, profiling, testing, and collaboration tools, Visual Studio streamlines the entire software development lifecycle. It supports a wide range of languages, including C++, C, .NET, Python, and more, making it a versatile choice for building applications ranging from desktop software to web and mobile applications [72].

4.3 Visual Studio Code

Visual Studio Code is an open-source text editor created by Microsoft as an alternative to Visual Studio. You can use it only for the basic functionalities of writing files with syntax highlight for the specific language you are writing. But its great potential comes from the opportunity to use it for compilation, execution and debugging using plugins, as well as other functionalities that are available to everyone in its marketplace. It's one of the programmes most used by developers because it allows them to work with many languages in a single application, and due to its extensions, it can easily be a substitute for other more specialized programs that require the use of several programmes [73].

4.4 Postman

Postman is software that allows you to make HTTP requests to an application. More focused on testing WebAPIs, it allows you to execute several HTTP requests of different

types, associate tests with the requests in order to test the response code, body of the response received, execution speed, etc. You can share HTTP requests with other people, run Postman tests in CI/CD, generate documentation, work with different response formats, which makes it a complete tool for testing HTTP servers [74].

4.5 Putty

PuTTY is a widely used open-source terminal emulator and SSH client for Windows. It enables secure remote access to servers and devices using the SSH, Telnet, and other network protocols. PuTTY offers a command-line interface for managing and interacting with remote systems. Its simple and lightweight design makes it a popular choice for administrators, developers, and anyone needing to establish secure connections to remote servers or network devices [75].

4.6 WinSCP

WinSCP (Windows Secure Copy) is a popular open-source software application used for secure file transfer and remote file management between a local Windows computer and a remote server. It supports various file transfer protocols, including FTP, SFTP, SCP, and WebDAV, and provides a user-friendly graphical interface for ease of use. WinSCP is commonly used by system administrators, web developers, and anyone who needs to transfer files securely between their computer and a remote server [76].

Chapter 5

Design

In this chapter is explained what are the users of this project, followed by the requirements that are expected to be implemented. After that, the design of the pipeline is described, with the justification about the decisions made.

5.1 CI/CD

This dissertation is focused on the application of the DevOps concept during the software development cycle, and such it is necessary to analyse how a developers team can apply this in a new project, with the steps of the methodology that will be designed in order to implement.

During the development of the ERP, is expected to design and implement a CI/CD pipeline using a tool already established in the market in order to create the process needed to maintain a fluid process that ensure a good automation of steps used to validate the new changes that was made to the software. This helps the developers to focus more on the program, and be more committed to the changes be deployed more early to the clients, since it helps create a feeling of safe environment.

The following use cases was listed as the primary sources to implement the DevOps process during the development of the ERP.

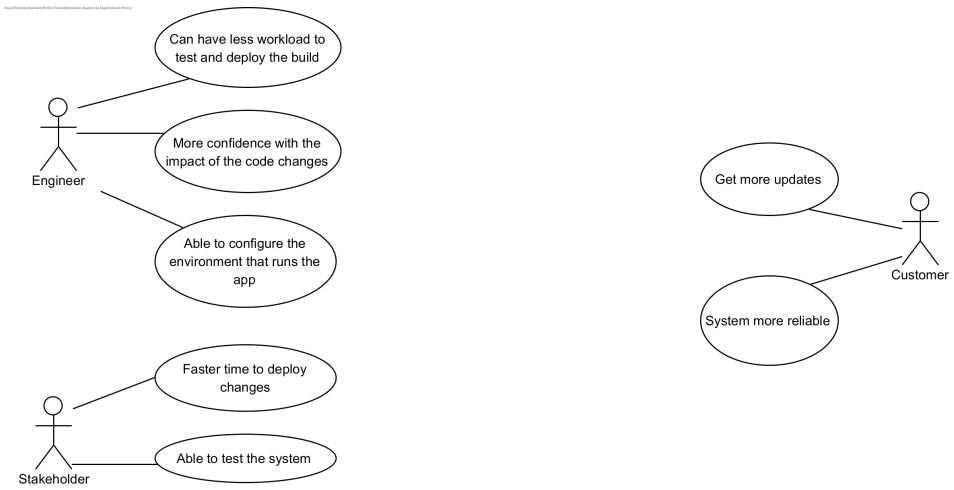

Figure 5.1: Use Case Diagram for DevOps

For the engineer it is important that the maximum time of his work is to develop new features, in order to grow the product more rapidly and to fill in the requirements of the clients. This can be very challenging, because making changes to the code already present in the product can be complicated if the product has not grown with the right structure, leading to a technical debt that suffocates new changes, and can even lead to more debt.

So, the DevOps methodology tries to facilitate the work of the engineer by removing the manual and gross time of the building, testing and deploying new changes, since part of this process, if not entirely, can be made automatically by the CI/CD process. With this, it helps to detect bugs more early, keep tracking of the changes made and their impact in the product, and can be used to implement others processes like code review.

In terms of Git branching, there are multiple ways to manage, depending on the size of the team, the number of commits made, and the overall team coordination. One of the examples is the Gitflow. Gitflow is a branching model and workflow for Git, designed to manage complex software development projects with structured branching strategies [77].

It defines two main branches: "Main" for stable production code and "Develop" for ongoing development. Developers create "Feature" branches for individual features or enhancements, which are later merged back into "Develop." When preparing for a release, a "Release" branch is created for testing and bug fixes before merging into both "Main" and "Develop." "Hotfix" branches are used to address critical issues in production and are merged into both "Main" and "Develop" as well. This approach promotes organization and stability in code development and release management, while creating very distinct branches, each one with their own purpose, separating the type of commits that can be made in each one.

While Gitflow offers structure and order to collaborative software development, it may introduce complexity, making it more suitable for larger projects with frequent releases and multiple contributors. Smaller teams or less complex projects might prefer simpler branching models like GitHub flow. The choice of workflow should align with the specific needs and dynamics of the project and team.

Others are simply have one main branch, and every time someone wants to commit just commit to main, like the Github Flow. GitHub Flow is a straightforward and lightweight Git branching strategy focused on enabling continuous delivery and frequent deployments [78].

It centres around two main branches: the "main" branch representing the production-ready code and feature branches created by developers for specific tasks or features. Developers work on these feature branches, implementing changes and improvements. When a feature is complete, they initiate a pull request (PR) to merge their branch into "main." The PR undergoes code review and testing, ensuring code quality and functionality. Once approved, the changes are merged into "main" and subsequently deployed to the production environment. GitHub Flow promotes a streamlined development cycle that emphasizes continuous integration, code review, and the swift delivery of new features and bug fixes, making it especially suitable for teams focused on agility and rapid development.

The simplicity of GitHub Flow makes it an accessible and efficient choice for many development teams, particularly those engaged in web development, cloud-native applications, and other fast-paced projects where frequent releases and collaboration are essential. Its ease of use and clear structure make it a popular choice for organizations that prioritize agility and responsiveness in their software development processes.

There are more approaches, like git rebase, where is maintaining a linear history, by developers rebasing their branches into the main, leaving only one single line of history, and others are even just directly make commits to a main branch, normally used only by smaller teams with minimal complexity.

In this project, the following branches will be used, taken into consideration the number of developers, and the cycle intended to have in this project:

- A development branch, where all the commits will have to be merged. This will be the most updated branch, as all the developments will get to this branch
- A developer, if find it better to have a separate branch for a specific functionality, it can create it, so that it can create a bigger change in the system, and only merge into the development branch when finished.
- A release branch, where the commits made into the development branch will be merged when the developers team decided that the features and hotfixes made is enough to create a new version of the software. In this case, a merge request will be performed, and a commit with a tag version will be made into the repository. The pipeline will make its final validations, and release the binaries and images correspondent to that version.

With this type of branching, it is possible to define what is the branch that represents the current software version, the version that have the latest code changes, and the process for releasing well-defined and automated.

This lead to the stakeholders gain more efficiency in their processes, because the engineers are working more on the product and less on maintenance and the pain to deploy new changes, leading to fewer costs to the company. Also, because changes can be deployed more times and faster, the business can deliver more features than the competition. To the customer, this DevOps methodology can result in a product more robust and with more changes and corrections made into the product, leading to more support.

With the CI/CD process, the pipeline must meet the requirements of the developers, operations teams and stakeholders of the company. The CI/CD must be the process that can:

- Automate the tasks that are normally associated indirectly with producing software, like building, run tests, check code quality, check code conformity with regulations, and deployment and configuration
- Make it faster as it would be done if it was a human make the similar steps
- Make it fail fast and deliver early feedback as possible, to reduce the time need to get the results from the integration of the changes into the codebase.
- Be a tool that can be a source of truth related to the build and testing, since it can exist cases where the developers may have local configurations that can prevent them from seeing the bugs that can occur in the production.

How can the pipeline be designed to cover these requirements? For that, the architecture proposed for this project is consisted in the following phases:

1. Pipeline is triggered by a commit made to a branch of the repository.

2. Pipeline makes the compilation of the project, to ensure the latest changes can produce a working executable.
3. The unit tests are run to ensure no change break the test suite.
4. At the same time the integration tests are run to ensure the services are able to communicate with each other.
5. After the tests are run, a static code analysis is made to analyse for bugs and code smells.
6. Next the stress tests are made to ensure the performance didn't reduce below the satisfactory value.
7. The deployment is tested onto a test environment to ensure the application and operating system can migrate the instances without issues.
8. Finally, the deployment can be made onto the production environment.

5.2 Audit Tool

The audit tool is a crucial tool in the maintenance of a working system, since it allows collecting data used to keep track of the system, collect metrics to study the overall working of the application during the time of execution, etc.

This helps not only the operations teams to giving proper support of the application for the clients, since they have more information available, but also for the developers, since the developers have more insights about how the code is performing in real scenarios, that can be affected by the volume of data or users, resulting in different code changes impact, that are not measurable or predictable in the development phase. And so, by having telemetry on the app, the tracking of the cause of problems is more easily findable, resulting in less time and costs in detecting and reproduce the scenarios, and also transmit more knowledge about what best practices should be enforced to reduce the risk of problems in the future [79].

In this project, in order to have the full cycle of DevOps, it is necessary to instrument the application and infrastructure with necessary tools to have the possibility to monitor when needed. So, the team designed a set of requirements in terms of logging, telemetry and auditing, that are considered to be essential when dealing with the monitoring:

- Collect logs from all the relevant services and the events of the operating system running the application;
- Aggregate and summarize the data into the relevant metrics and alerts;
- Platform to visualize all the data collected;
- Possibility to configure alerts for system downtime and others factors.

This requires that the architecture chosen make it this works is resilient to all type of system failures, without data loss and with rapid recovery. In the application was implemented an observability stack, e.g a tool that allows to log, collect and analyse the execution status of the application, traceability on indicators like latency and time spent to process a request, and metrics that illustrate the evolution of the system usage and performance.

In this project the OpenTelemetry was used to integrate the logging enhanced capabilities with Prometheus, that will serve as a time series database, and then use Grafana as a Dashboard tool for monitoring the system by using Prometheus as a data source for the data that will be displayed in the dashboards.

5.3 Security

Since this dissertation also focuses on DevSecOps and overall security, this plays an important aspect of the solution. This covers everything associated with the project, from the application that will be built, to the devops process and the tools used. This is to ensure that all the intellectual property is covered from unauthorized access by the use of vulnerabilities, and also to enforce good security practices among the developers and users.

For the application, it is important that the customers can feel safe when using it, i.e. that they can have trust in the data stored. For that, the application must already be designed to protect their users. After an analysis of the requirements for the application, it was decided to specify the following requirements:

- The users must authenticate in the system before start using it;
- The system must validate the level of permissions of the user before allowing the user to access certain functionalities;
- The system must provide more than one method of authentication;
- The system should implement good measures that prevent session hijack;
- The system should be validated against known vulnerabilities

In terms of authentication and authorization implementation, it will be used the Bearer authentication, with JSON Web Token (JWT), with the optional JSON Web Encryption (JWE) variant that encrypts the payload in order to prevent sniffing the payload of the content. This allows for stateless sessions, which reduce the overhead of the system in handling multiple sessions. Also, it allows the user to maintain the same session through multiple instances of the application.

For the source code and the CI/CD tool, it is important that is protected from thief, and accessing important customer's information. If this is not protected, then the application could be seriously compromised, not only by coping the product, but also interfering with the source code undetected, and possibly compromise customers machines. This is a serious question that should be also be taking care of.

For that, one of the important aspects is to have restricted access to the Gitlab instance. This can be enforced by setting up a whitelist of IPs that can access, or a VPN that can be used to then have access to the instance, instead of having it publicly available, since only a handful of people will use it.

Another measure that can be applied is the protection of the main branch of the repository, by only be possible to commit to it by a merge request. This creates another set of history, and also can be used to enforce code review of the code being pushed. The communication between the Gitlab and the possible environments should also be protected, by using HTTPS communications between the machines, and by only using elevated commands in an extreme scenario where other solution is not available.

The environments should also be accessible to their system and file system by using SSH that is not exposed to the internet, only exposing the application itself. The SSH connection should use certificates authentication, and the private key should be configured as a variable in the pipeline, since it allows to change the credentials without touching the pipeline, while also protecting from secrets exposed in the source code and version control repository.

Another way to enforce security is by using most of the time virtual machines and/or containers, that can run what they need, while also protecting the host environment from hijacking and gain privilege access to the infrastructure.

With this set of measures, it is much harder for intruders to infiltrate and start running unintended commands.

5.4 ERP

The ERP that will be developed in this dissertation, that is the base for this project, is focused on the small and medium companies businesses market. The system should help the owners of these places have a more agile process to manage and register their sales, with the importance that the data is correct. By making a software product oriented to this market, it is possible to test with an Minimun Value Product (MVP) to check if the application meets the requirements so that it can be used commercially in large scale.

With the before statement in consideration, it was developed the following use cases diagram, that represent the list of functional requirements that are taken into the scope of this project.

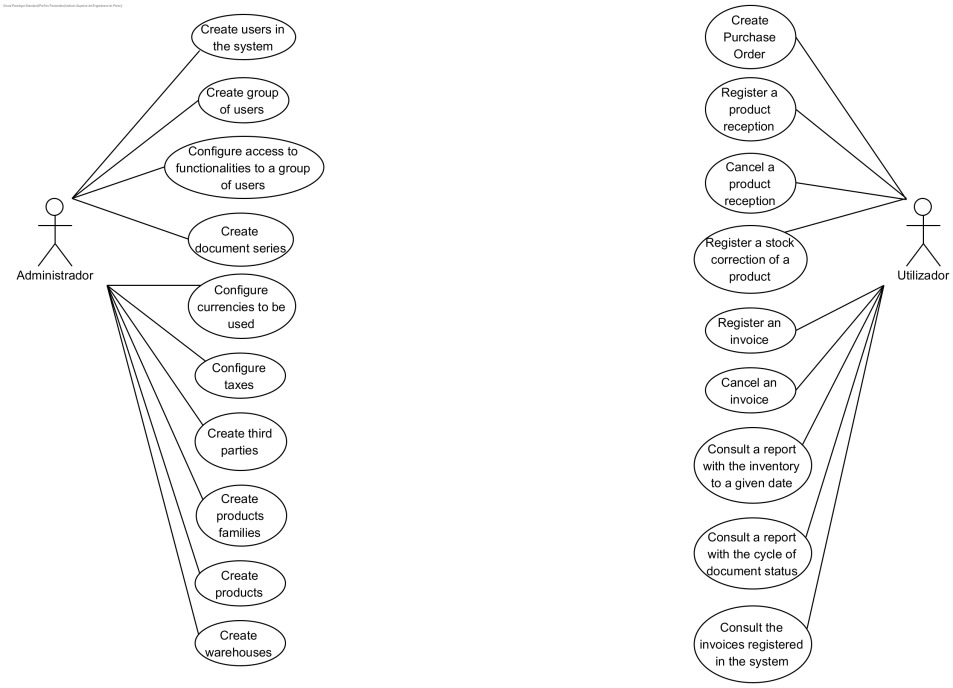

Figure 5.2: Use Case Diagram for ERP

The ERP will be composed by a base module that contains concepts of products, third parties, documents, series and currencies.

An administrator is supposed to make the configuration of the ERP that is appropriate for his business. For that it is possible to configure group of users and their accesses, and the

list of users in order to manage what persons can interact with the system. Also, the series of documents can be configured as their needs, so that the organization of documents in the system is as wanted. The currencies and taxes are an important configuration for documents that have associated monetary value. The products are what the business can buy or sell, and the warehouses is the facilities that can be logically created in order to distinguish the stocks and their locations. Finally, the configuration of third parties is important to maintain a list of suppliers and some customers.

Outside the scope of this dissertation, implemented by two others students, it will be created the stock modules, related to the management of the products and their inventory, and associated reception and documentation of products. Also, the sales module will be developed, with the invoice management and cash flow of the business, allowing the company to register the billing service made to their customers. To the normal user, is important to be able to make most of the movements that occur in the business, like the purchase orders and the product reception, invoice registration, and consult the history of these movements.

In terms of non-functional requirements, the following list was defined:

- The system must work both On-Premise and on Cloud environment;
- The system must work on Web browsers;
- The user must be able to work with the system in any operating system;
- The system should have multilanguage support;
- The system must be able to work with multiple users;
- The system should log the actions made in the system, in order to be possible to audit its status;
- The system must work with low latency;
- The system should use the lowest possible resources from the browser;
- The system must encrypt the data that is sensible or contain personal information;
- The system must comply with the GDPR;
- The system should be simple and easy to use;

Chapter 6

Implementation

In this chapter it will be discussed the technical decisions made, and described the process that was taken to conduct the development of the project.

6.1 Gitlab

So, as described in the chapter 3, the tool chosen was the Gitlab as it have strong tools to manage repositories, the implementation of CI/CD pipeline, and overall all the integrations and facilities made around DevSecOps specifically.

For this project, a self-hosted Gitlab was mounted, as it allow to have more control of the features enable, and also the liberty to use the majority of the resources without costs. They are still some features that are behind a paid licence, but with this setup it is possible to make adjustments to the setup made, like the security around accesses for example, and still have the code hosted on a private server.

To create this instance, a remote server running Ubuntu was used, as it is one of the most known distributions of Linux being used. Then, it was installed a Gitlab Community Edition version. There are plenty of ways to deploy a self-hosted instance of Gitlab, but it was used the Linux Package deployment, where the binaries are already compiled, and a combination of the chef tool with a ruby configuration allows the customization of the instance, with tools prepared to startup all the necessary services[80]. After that was made, the next steps was configured the users that will access this instance, the groups of users, repositories to be created and given the proper access.

After the installation, it was necessary to create the repository where all the work will be put, i.e. the code of the application, and all the necessary stuff to create the pipeline. The group decided to have two separated folders, one for the backend API project, and another for the frontend React project, allowing for better separation when dealing with building and deploying the code. This is visible in the figure 6.1.

Next, it was created the CI/CD pipeline, as designed in the 5 chapter. The Gitlab uses a .gitlab-ci.yml file to have all the necessary stages, jobs, steps and scripts necessary to run all the necessary configuration and execution when a commit is made. This file is at the root of the Git repository, and the user can edit it manually, or using the web browser with the built-in editor from Gitlab. By using this strategy, ensures that the Git is the single source of truth of everything, and also is very easy to migrate the repository into another installation of Gitlab, since the majority of the configurations is already versioned.

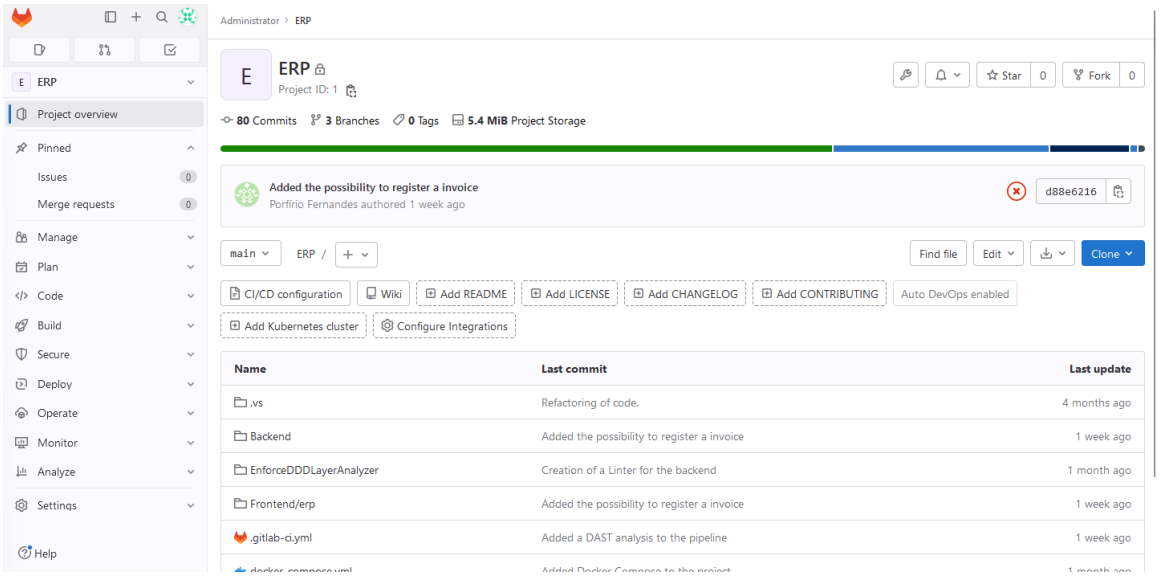

Figure 6.1: Repository in Gitlab

The Gitlab allow for some advance configurations in the pipeline, that can be based on the branch the commit was made, if the source code was changed, dependency between stages, etc. Also, it allows to be running in multiples environments, like Docker, Linux, Windows, etc., allowing for better control of the environment and dependencies to run a specific target or script, since it is normal to use external tools.

Gitlab, although is not only a CI/CD tool, offers many features, like pipeline visualization of the stages and jobs and their dependencies, simple parallel execution, management of artefacts, docker images and binaries, package registry, approval jobs, auto devops, secret management, cross-project pipeline, security scanning, etc.

The Gitlab ha1ve a concept of Runners, where the machines that are allocated to run the stages can be independent of the server that is running the Gitlab itself. By separating the applications needed to execute the pipeline from the Gitlab itself, is possible to spread the workload between one or multiple servers, allowing for faster time to execute a pipeline, by removing the waiting time of the pipelines. Then, depending on the type of job (whether it's supposed to run in docker, windows, shell, etc.), the Gitlab picks one of the available runners that is capable of executing the job, to run it and return if it encounters some errors. In this case, it was used a separated machine to have the Gitlab Runner configured. It was only necessary to install the package, and then running the command "gitlab-runner configure", that after a few steps, the instance is ready to start receiving job executions.

Since the Gitlab have a good community feedback, and is focused on DevSecOps, it already possesses several templates that can be imported to the pipeline. The templates can run a specific set of SAST, DAST, Container Scanning, Dependency Scanning, Secret Scanning, and much more, all already automatized based on the languages and projects that are presented inside the repository. For this project, it was imported the following templates, since they are very important to maintain a higher rate of confidence in the code produced by the developers.

So, with all configuring, it was time to set up the pipeline. In this case, and after a few adjustments during the implementation of the project, it was decided to have the following

stages.

- **Prebuild**: This stage is to ensure that all dependencies to build the application can be imported or build it first.
- **Build**: This stage is to ensure the latest changes made to the application does not prevent the application from building
- **Image**: Then, it is necessary to ensure that the build of a Docker image is also not broken, ensuring that all the artefacts to run the application can be put inside the image
- **Test**: After that, a series of tests are run (unit tests, integration tests, SAST tests, DAST tests, secret analysis, dependency scanning, etc.), to ensure that the application didn't break, or didn't bring new vulnerabilities into the code.
- **Analysis**: Then, the quality gates, and the scan for code smells is made into the application, to ensure the quality of the code produced didn't reduce below the expected.
- **Pages**: A simple stage to publish the results of the analysis into the repository
- **Deploy**: Deploy the new changes into the staging environment, by using IaC tools in order to only run the necessary steps to transit the infrastructure to a new state.
- **Release**: This stage only runs when a new version of the application is made. This is done by associated a tag with the commit. Then, the system will publish the binaries and image artefacts, and associated them with a new release entry in the repository. This allows that the versions that are published are easily tracked.

Inside each stage there's one or more jobs that are executed. The necessary dependencies are declared in each job, so besides the stage order, the jobs are ensured to have the required order to execute without errors. Then, some jobs have conditions to run, so it is possible to control whenever the jobs are not relevant to run, allowing to reduce the loading and costs of the runners, and improving the overall performance of the pipeline. One example of how the pipeline history is shown in Gitlab can be seen in the figure 6.2.

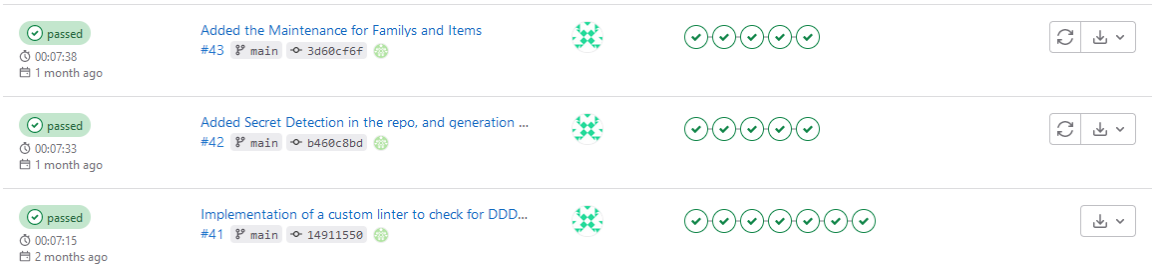

Figure 6.2: Pipeline History

After the pipeline, it was necessary to activate and configure the rest of the tools that Gitlab provides. One of them was the Gitlab Pages. A problem with Gitlab Pages is that it is designed to have of the pages have a prefix of the group in the domain. This requires, in order to use HTTPS, to use wildcards certificates, that can be more expensive than a regular fixed domain one. To activate, in the gitlab.rb file, it is necessary to activate the feature, configure the ports and location of the certificates, and restart the Gitlab instance.

Another feature that was required to activate was the Container Registry. To be able to use the Gitlab Container Registry as the repository for the images when pulling and pushing, it is necessary to expose the service at a specific port. To use it in the pipelines, it is possible to leave it dynamically, since the Gitlab expose internal variables that contain the URL, username and password, that can be then used in the scripts on the pipeline to make push or pull of images from the Container Registry.

Another feature was the Gitlab Kubernetes Agent. This agent is responsible to be a node on one Kubernetes cluster, and from that it can push configuration to the controller to change the number of replicas, or even the overall state of the cluster with modified images, all from the pipeline itself. After activating and configuring the proper ports, the repositories can then commit a file that contains the credentials and configuration needed for the agent to connect to the cluster and check the state of it.

One of the features that was changed was the prometheus module that is included in the instance. Since the group deployed a dedicated Prometheus and Grafana server, it was decided to pull also the metrics from the Gitlab into that instance. So, the prometheus were disabled, and only the metrics URL was exposed to pull the information from the Gitlab into the separated server that have the Prometheus instance.

6.2 Optimizations on the Pipeline

In this section will be described several steps that were performed, in order to adjust the pipeline to reduce the processing time of it, time of artefacts persistency, version management of the images and binaries, etc., effectively making the pipeline faster to run, and so less annoying to wait for the results of the CI/CD to gain feedback about the commit made.

First, the pipeline can be configured to have a specific order using the keyword *stages*, that, instead of putting the jobs and their scripts, it is possible to explicit give a macro order of the pipeline. Then, each job can be associated with a specific stage. This ensures an overall dependency and sequential validations of the jobs.

```
1  stages:
2      - prebuild
3      - build
4      - image
5      - test
6      - analysis
7      - pages_prep
8      - pages
9      - validate
10     - deploy
11
12 image_frontend:
13     tags: [shell]
14     stage: image
15     script:
16       - cd Frontend/erp
17       - docker build -t $CI_REGISTRY_IMAGE/frontend .
18       - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD
     $CI_REGISTRY
19       - docker push $CI_REGISTRY_IMAGE/frontend
```

Listing 6.1: Stages Configuration

One of the changes made was using the keyword *needs*, which makes the dependencies between jobs explicit. This helps in two situations: one is that the gitlab, when checking for the next jobs that can run, can pick jobs from stages ahead if the others cannot cause the pipeline to abort. The other is that the jobs can effectively configure more options when describing the dependence with another job, like the requirement to have the artefacts generated in the workplace when executing the job. This can also be done similarly with the keyword *dependencies*.

```
build_backend:
    tags: [docker]
    stage: build
    image: mcr.microsoft.com/dotnet/sdk:6.0
    script:
      - dotnet restore Backend/Piloto.sln
      - dotnet build --configuration Release Backend/Piloto.sln
    needs:
      - job: build_backend_custom_linter
        artifacts: true
    artifacts:
      paths:
        - '**/bin/Release/net6.0/*.dll'
        - '**/obj/project.assets.json'
      expire_in: 1 week
```

Listing 6.2: Needs usage in Gitlab configuration

Another optimization is make it explicit how much time the artefacts generated from the job should be persisted. This helps the developers to check if the jobs generated the right files, but after a while the Gitlab itself automatically release this files associated with the repository, making the size of the repo not grown exponentially.

```
build_backend:
    ...
    artifacts:
      paths:
        - '**/bin/Release/net6.0/*.dll'
        - '**/obj/project.assets.json'
      expire_in: 1 week
```

Listing 6.3: Expires usage in Gitlab configuration

One of the other optimization is the use of the keyword *rules* where it is possible to configure when certain jobs should be run. This can be when detecting file changes in a specific folder, what branch was the commit made, use of variables state within the pipeline

```
release_job:
    stage: release
    image: registry.gitlab.com/gitlab-org/release-cli:latest
    rules:
      - if: $CI_COMMIT_TAG                  # Run this job when a tag is
    created
    script:
      - echo "running release_job"
      - apk add curl
      - apk add jq
      - export BACKEND_CONTAINER_REGISTRY_ID=`curl --insecure -s --
    header "PRIVATE-TOKEN${RELEASE_TOKEN}" [REDACTED] | jq '.[] | select
    (.path == "root/erp/backend") | .id'`
```

```
11      - export FRONTEND_CONTAINER_REGISTRY_ID=‘curl --insecure -s --
    header "PRIVATE-TOKEN${RELEASE_TOKEN}" [REDACTED] | jq '.[] | select
    (.path == "root/erp/frontend") | .id’‘
12      - release-cli --insecure-https create --name "Release
    $CI_COMMIT_TAG" --description "$CI_COMMIT_TAG" --tag-name "
    $CI_COMMIT_TAG" --assets-link "{\"name\":\"frontend_binary\",\"url
    \":\"${PACKAGE_REGISTRY_URL}/frontend.tar\", \"link_type\":\"package
    \"}" --assets-link "{\"name\":\"backend_binary\",\"url\":\"${
    PACKAGE_REGISTRY_URL}/backend.tar\", \"link_type\":\"package\"}" --
    assets-link "{\"name\":\"backend_image\",\"url\":\"[REDACTED]\", \"
    link_type\":\"image\"}" --assets-link "{\"name\":\"frontend_image
    \",\"url\":\"[REDACTED]\", \"link_type\":\"image\"}"
```

Listing 6.4: Usage of rules keyword to determine if the pipeline should create a release entry in the repository

Similar optimization can be done with the keyword *when*, that allows to configure if a job should only run if everything before run with success, if others have failed, etc. This allows to handle specific conditions inside the pipeline, like for example publishing always the coverage reports even if the pipeline have failed, etc.

```
1  deploy_to_kubernetes:
2      stage: deploy
3      tags: [docker]
4      image: alpine:latest
5      script:
6          - apk add --no-cache curl
7          - curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.
       k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
8          - chmod +x kubectl
9          - mv kubectl /usr/local/bin/
10         - kubectl --kubeconfig .gitlab/agents/kubernetes/config.yaml
       apply -f db.yaml
11     environment:
12         name: kind-cluster
13     #when: manual
```

Listing 6.5: When usage in Gitlab configuration

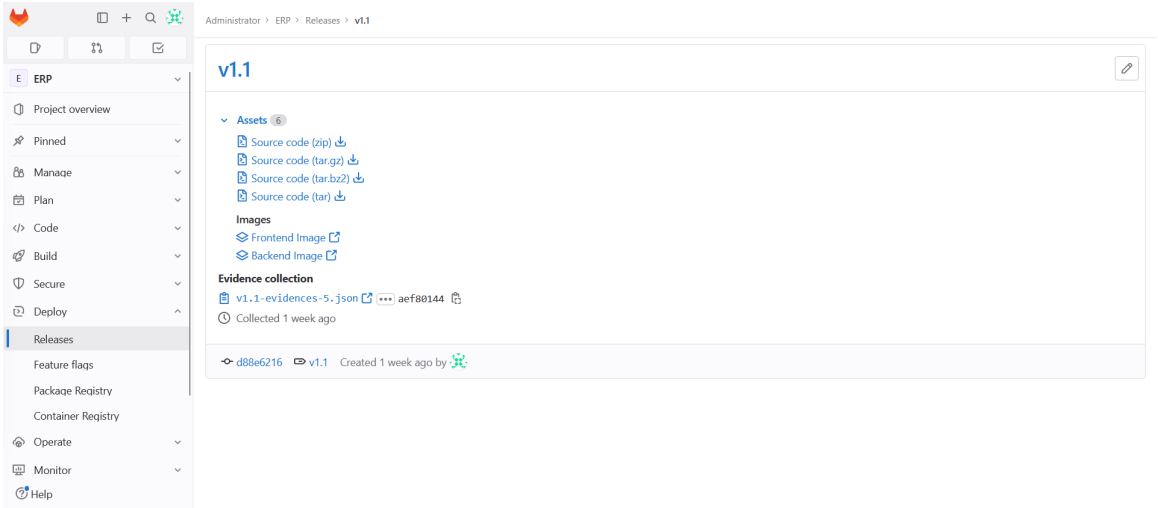

Figure 6.3: Release Screen

6.3 Linters

One of the concerns in today's markets is the incorporation of the security aspects related to software in the software development lifecycle, as the search for actively validate if the software produced is free of vulnerabilities is relevant to the most companies, since most of the tools are now connected to the internet and can have integrations with others software products, that can result in a larger surface to be attacked.

One tool used in this project was linters, since these tools integrate nicely in an IDE, it can be used to give extended warnings or errors about specific code styles and procedures that should be avoided, leading to a consistent code base in terms of styles, and also free of deprecated or not recommended approaches to implement some features. This can also be enforced when the project is build in the pipeline, ensuring that the code that is committed can only be considered to be merged if the changes respect the rules of that linters.

In terms of "Security By Design", it was also explored and implemented a way to create a custom linter that allows to enforce even more rules. For example, one of the strategies was to implement a linter that have a rule that doesn't allow specific imports to be made between layers of the project. This mean that the code is validated that the reference of others classes respect the layers that were designed in terms of project.

This was made for the backend part, where a Rosyln linter was implemented using the template provided in Visual Studio by Microsoft, and then generating a local nuget package[81]. The package was then imported in the backend project to enforce that the code respect the multiple layers, and how they can communicate between them.

First, it was creating a new project from a template provided by Visual Studio, shown in the figure 6.4. Then, the code made was to be registered every time the IDE or compiler detects the user of a using statement. In that case, it will run a specific method that will then enforce the rules to check what was the import. If the import does not respect what is expected, then it is issue a warning.

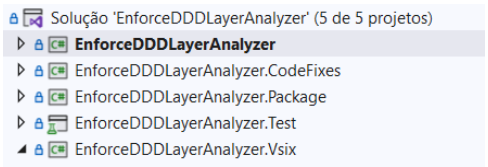

Figure 6.4: Linter Project

After that, it was generated the VSIX package, that can be installed in any Visual Studio installation, or a package, that can be imported as dependency for the project, as it can be seen in the figure 6.5. The developers can import locally, or as a nuget package from the Gitlab Package Registry (making it private to use), and then starting getting the same warnings and errors in the code as the pipeline. With this, it is possible to create new custom rules that can convenient enforce the necessary constraints to maintain the code.

For the frontend component, since it was used React, the group decided to use some linter already established that could enforce some best practices that help reduce the overall risk of bugs. After some research, it was decided to go with the Airbnb Javascript linter, that uses eslint to customize their rules. This rules can be quite aggressive, but it helps enforce robustness of the code by helping the developer produce better code.

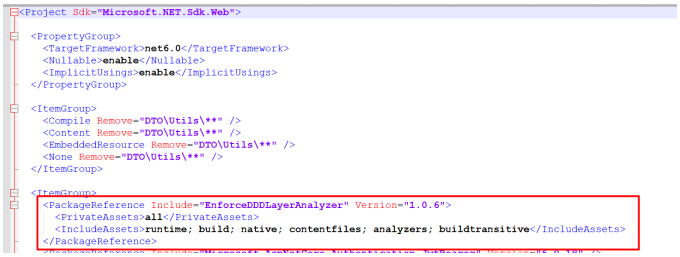

Figure 6.5: Linter Import

The configuration was very simple, since it is available as a npm package, as it can be seen in the 6.6 figure. After installing the package, the developer just requires the IDE that is using to have EsLint integration to start getting warnings visible in the code, with possibility to fix it. In this case, since it was used Visual Studio Code, the EsLint integration is one plugin install.

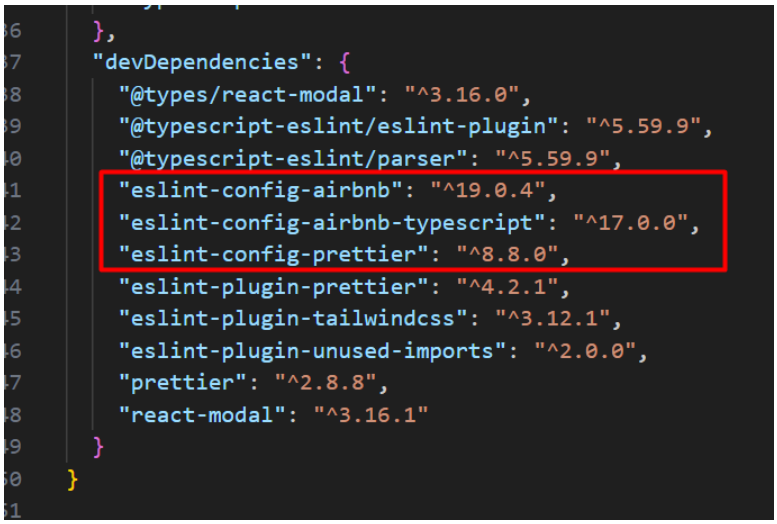

Figure 6.6: Eslinter NPM packages

As with the backend, the publishing steps enforce that no warnings are produced, so the developers, even if they are ignoring the linter while developing, before pushing they must correct the code so that the pipeline can execute correctly and their code can be merged.

6.4 Logging

Applications should not run without some sort of log or monitoring tool, since this leads to unpredictable and unknown errors, and even not knowing when a system starts to fail, since the only feedback that the developers have is the customers complaining about the malfunction of the system. To prevent that, and have the possibility to monitor what is happening in the system, it was considered the use of telemetry.

Observability is the concept of understanding what is happening in the system, without knowing how the system is implemented. So, this means that it should be possible to analyse the work, performance and other metrics of the application by considering the application as a black box system. In order to reach this level, the application must transmit to outside logs, metrics and traces. The instrumentation is considered satisfactory when the developer

doesn't need to have more information in order to debug an issue. This allows the developers to more easily respond to possible errors, even if they are in production environments, because they will have the right information that allows to them to track where the errors occurred, or where the system is getting malfunction. By investing the time at development phase to instrument the application, the costs to track the errors after are greatly reduced [79].

For this project it was considered the OpenTelemetry, since it is the open source, and is the library that is trying to standardize the format across several technologies and monitor tools. OpenTelemetry is specially useful when dealing with distributed application, since the trace context is propagated between the different services where the request passes by. This allows for a more understanding when searching for the errors' location across multiple services in the network. Besides the instrumentation, is necessary to export this data to some database or monitoring tool. The OpenTelemetry separate this component from the rest, allowing for better adaptability, since multiple exporters can be defined and swapped easily, without affecting the rest of the instrumentation process. Some examples of exporters that OpenTelemetry provides are Prometheus, Jaeger, etc.

In the case of the backend, a simple import of the package is required, and then a simple configuration in the startup of the project already allows keeping track of the requests that are made, and the time that the system take to process the request or even consult the database, as it can be seen in the figure 6.7. The developer can add manually more metrics and logs that he considered being relevant to be made when the application is working.

```
// Add OpenTelemetry
builder.Services.AddOpenTelemetry()
    .WithTracing(tracerProviderBuilder =>
    tracerProviderBuilder
        .AddSource("Backend")
            .ConfigureResource(resource => resource
                .AddService("Backend"))
            .AddAspNetCoreInstrumentation()
            .AddHttpClientInstrumentation()
            .AddSqlClientInstrumentation()
            .AddEntityFrameworkCoreInstrumentation()
            .AddJaegerExporter()
            .AddConsoleExporter()
    );
```

Figure 6.7: Include of OpenTelemetry in the Backend Project (.NET 6)

Then, the logs are exported to a Prometheus instance, which will then collect all the logs, traces and metrics and persist it, to then being able to being used in a monitor tool.

For the frontend, a similar approach was made, where the package was imported into the project, and then the necessary registration was made into the code. In this case, since it is a React with NextJS, both the server and client components are associated to be instrumentation.

For the database, in this case that the project is using MySQL, the Prometheus can also read directly from the MySQL the necessary database without further tools. So, it only requires the necessary configuration to access the database, with a user to connect, and Prometheus can consult the data to retrieve the metrics that can be then used to develop dashboards.

6.5 Sonarqube

It was decided to use Sonarqube for this part, since it was already knew by the group, and this tool have many features now that ever, like a sonarlint that can be integrated in IDEs like Visual Studio Code. Also, the code quality is a paid feature on Gitlab, that although it generates the JSON and HTML reports in the free version, to see in the Pipeline result is only possible if the user have a premium licence, or makes manual publish report to the Pages section of the repository [82].

Sonarqube is an open-source platform designed for continuous inspection of code quality and code analysis, with tools that identifies code quality problems, vulnerabilities in the code, code smells, and duplication code. It offers a static code analysis, with option to define custom rules, that can be integrated nicely in CI/CD pipelines, with dashboard to get the overall result of the quality of the project. It has the community and enterprise editions, and possibility to configure plugins to extend functionalities. This allows the enforcement not only of code styles, but also prevent even before the commit of changes of bad code or code with bad practices, since this will give warnings to the developer to rethink and reimplement a better solution while working on a code base.

For this project, it was used the Open-Source and Community Edition of the Sonarqube, since it requires only a server, with no extra costs in terms of licensing, and it was possible to make the adjustments needed. Also, since the author have already experienced with installing and configuring Sonarqube, it was made that decision. It was used a new server, installed the required dependencies, and configured the database.

After configuring the Sonarqube in the cloud as a self-hosted instance, the next step was to create the users that will have access to the instance. Next, the projects have being configured, in order to create the project keys for the right language that was going to be analysed, since the project is constituted in two languages, C# and JavaScript. It was exported the necessary configuration so that the developers can have integrated in their IDEs the necessary information to create better code, must like the Linters mentioned previously, and also to be configured in the pipeline jobs that will trigger the SonarQube scanner to check and send the analysis data to the instance, where the developer can then next for new bugs and code smells.

The variables necessary to run the tool (like the project key, the URL of the instance, etc.) is all in Gitlab variables, so that the information is not hardcoded in the repository. If some checks is not between the satisfying values, then it will also cause the pipeline to abort and not advance, and obligate the developer to revise the code in order to maintain the necessary quality levels. The projects can be then be seen as the figure 6.8 shows.

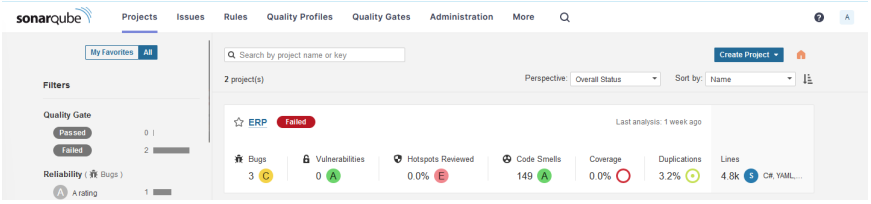

Figure 6.8: Sonarqube instance

6.6 Docker

Docker is a containerization tool that allows the developers to package and distribute applications together with their dependencies and system tools as an isolated and consistent environment. The images generated are lightweight, standalone units that contain everything needed to run the application. Docker makes it simple to build, deploy and manage running instances across different platforms and operating systems, by ensuring that they run consistently regardless of the underlying infrastructure.

In this project, it was decided to embrace the Docker tool, since its agnostic nature allow bettering understand all the configuration and libraries needed to run the application, and also have a consistent way to replicate a fully working environment to run the application, giving better facilities over installing standalone tools that require special attention to deal with dependencies and checks. Also, it helps reduce multiple environments difference, reducing the number of bugs occurring.

One of the strategies of deployment of the target application being developed is the use of Docker, since is environment agnostic, allowing for better accuracy, confidence and reliability when deploying the changes made into the product. For that, a Dockerfile was made for each component present in the application, where is specified the steps needed to build the image and then run it.

Although the docker could only be used after the build was made, by also making the build inside Docker, there's no need to worry about having the right tools in the server that is building in order to create a new version of the runtime. Also, since the Docker supports multi-stage builds of image, it is possible to use one image as a base to build, and then another with less resources and dependencies to run the final build. With this, anyone can rapidly create a new instance of the application.

It was created two Dockerfiles, one for the backend, and other for the frontend, each one with their scripts. After that, since the application to run requires at least three components (database, backend and frontend), a Docker Compose was created that ensure the images are build and configures how these containers can communicate between them. With this, the pulling of the latest changes, and the startup and shutdown of all containers is very simplified, reducing the risk of configuration drift, human error, etc [83].

```
1   version: '3'
2   services:
3   erp_backend:
4       image: REDACTED/backend
5       ports:
6         - "7141:80"
7       depends_on:
8       db:
9         condition: service_healthy
10      environment:
11        - ConnectionStrings__DefaultConnection=Server=db;Port=3306;
    Database=${MYSQL_ROOT_DATABASE};User=root;Password=${
    MYSQL_ROOT_PASSWORD}
12
13  db:
14      image: mysql:latest
15      environment:
16      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
17      healthcheck:
```

```
18          test: ["CMD", "mysqladmin" ,"ping", "-h", "localhost"]
19          timeout: 20s
20          retries: 10
21
22      erp_frontend:
23          image: REDACTED/frontend
24          ports:
25            - "3000:3000"
26
27      cadvisor:
28          image: gcr.io/cadvisor/cadvisor:v0.47.0
29          container_name: cadvisor
30          restart: unless-stopped
31          privileged: true
32          ports:
33            - "8080:8080"
34          volumes:
35            - /:/rootfs:ro
36            - /var/run:/var/run:ro
37            - /sys:/sys:ro
38            - /var/lib/docker/:/var/lib/docker:ro
39            - /dev/disk/:/dev/disk:ro
```

Listing 6.6: Docker Compose

Gitlab allows to have a private Docker image registry, that can be used to push new versions when a pipeline is running for a commit, or even upon a release of a new version, reducing the exposed information needed to a minimum, having great advantage in an enterprise environment, as it can be shown in the 6.9 figure [84]. Also, by being all in the same tool, is faster to track in which commit was originated the image with a specific version or changes.

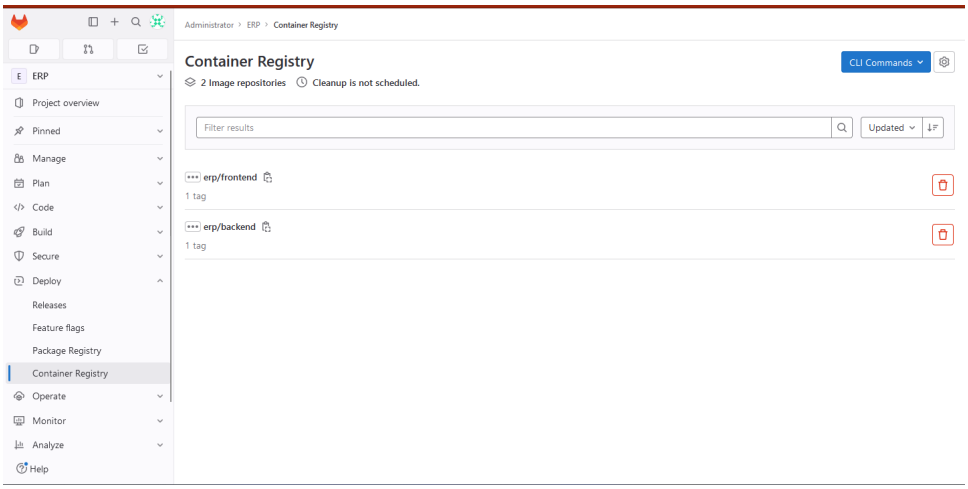

Figure 6.9: Gitlab Container Registry

6.7 Kubernetes

In the last section it was described how Docker and Docker Compose can help start in a server the containers necessary to run the application, making it nice to start up the application without too much of a problem.

However, one of the industrial needs nowadays is to be able to scale up the application as needed, since it can become used by many users, and also the possibility to have a second instance in case the firsts one starts to failing.

Kubernetes clusters consist of multiple nodes, including a control plane and worker nodes. It offers high availability, scalability, resource management, and self-healing capabilities for containerized applications. Kubernetes is an open-source platform that automates the deployment and management of containerized applications. It achieves load balancing through different service types, such as ClusterIP, NodePort, and LoadBalancer.

Another case that was considered was the use of Kubernetes to deploy the application, since it allows for better escalation in case of instances of the application where the number of users, and the number of actions made are significantly larger that the use of more than one machine can help relieve some bottlenecks when dealing with great spike of workload.

In the project it was created the necessary configurations files to create the pods of the backend, frontend and database. For that, is only require to describe what is the image to use, the name of the pod, number of replicas, environment variables, port mapping and the type of mapping. This was done to each of the component of the application, in order to separate the various configurations. Next, in order for the frontend to know where is located the backend in runtime (since it can have multiple instances), it was created a fourth configuration, so that it was created a special URL to redirect to the right pod. This is possible because the Kubernetes create a internal DNS entries, must like the Docker, but can be exposed to outside using an Ingress type of pod.

In this case, every time a URL with the host of the cluster, but with a path that starts with a specified string, then the Kubernetes will gracefully redirect the request to the right instance.

```
1   apiVersion: apps/v1
2   kind: Deployment
3   metadata:
4   name: erp-backend-deployment
5   spec:
6   replicas: 1
7   selector:
8       matchLabels:
9       app: erp-backend
10  template:
11      metadata:
12      labels:
13          app: erp-backend
14      spec:
15      containers:
16          - name: erp-backend
17          image: docker.io/erp-backend:latest
18          imagePullPolicy: Never
19          env:
20              - name: ConnectionStrings__DefaultConnection
21              value: "Server=db;Port=3306;Database=piloto;User=;
    Password="
22  ---
23  apiVersion: v1
24  kind: Service
25  metadata:
26  name: erp-backend-service
27  spec:
```

```
28      type: NodePort
29      selector:
30          app: erp-backend
31      ports:
32          - name: backend-port
33          port: 7141
34          targetPort: 80
```

Listing 6.7: Kubernetes Configuration for the Backend

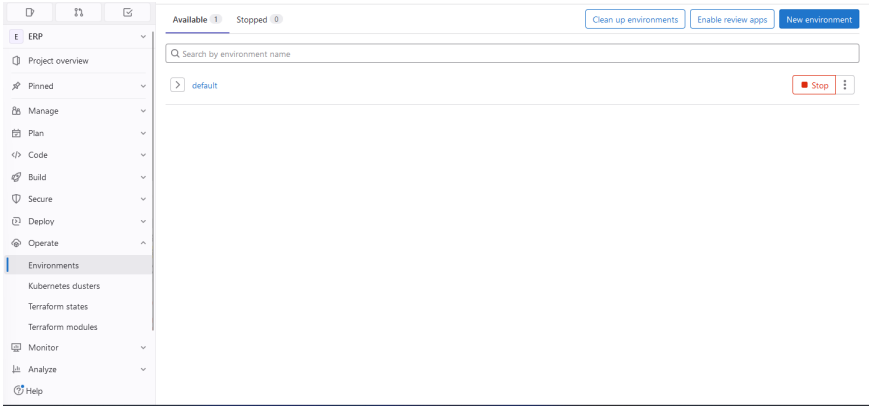

Figure 6.10: Gitlab Environments Management

6.8 Terraform

Terraform is an open-source infrastructure as code (IaC) tool that allows you to define, provision, and manage cloud and on-premises infrastructure using code. It enables automation, version control, and consistency in infrastructure deployment and management.

Although all the tools used before allows for automation of the process to start up the service, none of them allows to, consistently, create the infrastructure (that can be servers, Kubernetes, databases, file servers, etc.) necessary to run all the software that the ERP requires. Since normally making this process manually is tedious, and requires multiple steps, the use of Terraform was considered, since it allows automatizing even the creation of the servers, firewalls, etc.

Since the group was using Hetzner provider for the rent of servers, it was decided to experiment the creation of a terraform config file that can take use of the hetzner terraform provider to automate the creation of a server with all the necessary software to run the ERP, that can go from running natively in the server and with the runtime of the languages installed, with Docker only, or even spawn a Kubernetes cluster. All of that automated by the CI/CD pipeline, and repositories with terraform templates.

Then, every time is necessary to recreate the infrastructure of a client in the cloud, or even automatize the upgrade of the ERP version, it all can be made using only the repository, since the CI/CD plus the terraform script and the terraform state persisted in the backend of Gitlab (that have a special integration with Terraform) will take into account which part needs to be created again, as it can be seen in the figure 6.12.

This allows to reduce drastically the time it takes to create new servers, or make migrations between instances, as the process is all automated using a IaC tool.

```
terraform {
  required_providers {
    hcloud = {
      source = "hetznercloud/hcloud"
      version = "1.42.1"
    }
  }
}

provider "hcloud" {
  token = "                                                              "
}

resource "hcloud_server" "my_server" {
  name          = "teste1"
  image         = "ubuntu-22.04"
  server_type   = "cx11"  # Replace with desired server type
  ssh_keys      = ["deploy"]
  datacenter    = "fsn1-dc14"  # Replace with desired datacenter
  user_data     = "${data.template_file.instance.rendered}"
}


data "template_file" "instance" {
    template = "${file("${path.module}/code.tpl")}"
}

locals {
  server_exists = length(hcloud_server.my_server) > 0
}

resource "null_resource" "deploy" {
  triggers = {
    server_exists = local.server_exists
  }

  depends_on = [hcloud_server.my_server]

  provisioner "local-exec" {
    command = local.server_exists ? "echo 'Server already exists'" : "echo 'Server created and deployed'"
```

Figure 6.11: Terraform configuration for Hetzner Cloud

6.9 Atlantis

The use of infrastructure as Code helps to reduce work and have a single location of truth across many environments, since the operation teams are required to know exactly what are the steps necessary to create all the infrastructure needed. But, this still must require some previous validations, as a push of a bad configuration can result in a failure of the infrastructure. While this can manage manually, the process of test and validate becomes tedious.

For that, exist tool in the market that take advantages of the merge requests to detect if the IaC configuration suffer changed, and if that's the case then they will validate the config, plan the resources that will suffer the change, and associate the output to the marge request, giving the approvers information to be able to approve without need to manually check the changes.

One of those tools is Atlantis, that works by configuring webhooks associate with the repository, in this case Gitlab webhook when someone makes a merge request. Then, each time someone creates a merge request, the tool will automatically try to validate and plan the new terraform configuration, in order to leave associate with the merge request that the configuration is valid, and it will make some specific changes to the environment. Then, the users with authorization to approve the merge request can check this previous execution from Atlantis, and from that they can approve the merge, which lead the environment to have a new state, or reject it. This state change can then be made by Atlantis itself, or by the normal execution of the pipeline.

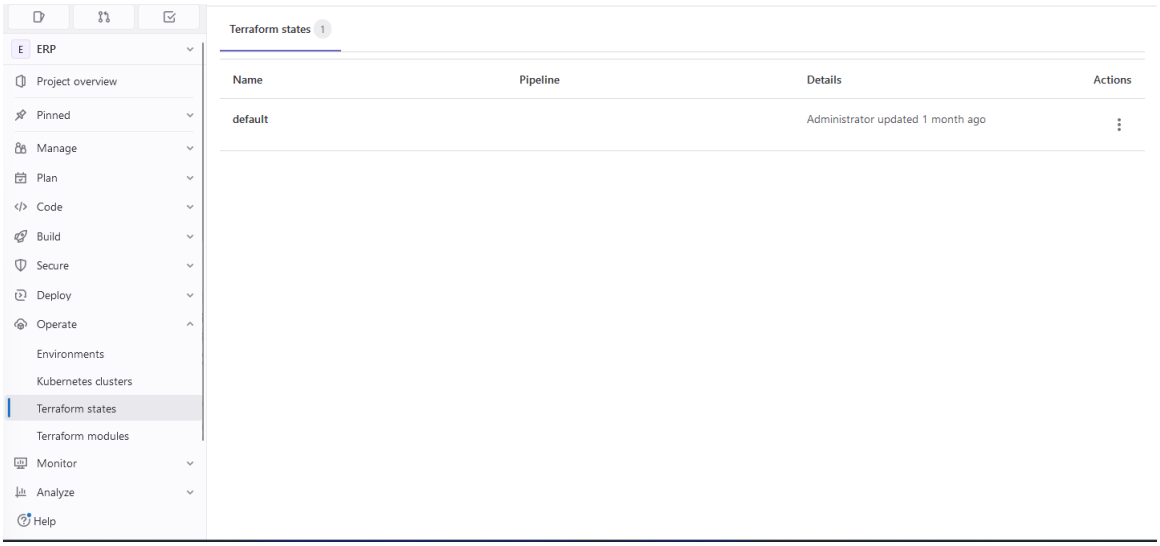

Figure 6.12: Gitlab with Terraform Integration to store state

6.10 SAST

One of the aspects of this project is the effort to maintain a high level of security in the application being developed, and how you can automatically guarantee that the system is secure. Static Code Analysers can already make some of this validations in terms of best practices, but security can require specific tools that are looking for vulnerabilities in the code. Well, it is possible to resort to some tools that can analyse the code and detect most of the vulnerabilities and bad practices being used by the developers, enforcing that they don't publish bad code into the clients.

Static Application Security Testing (SAST) is the type of tools that makes a static analyses of the code to detect such flaws. They can scan source code, bytecode or binaries, without running the code, meaning the effort to run is reduced and can be easily integrated into a CI/CD pipeline. This allows for early feedback for the developer, and it is possible to configure custom rules, depending on the type and target of the application. SAST have specific rulesets, deeper analysis and language expertise that is capable of detecting more types of vulnerabilities that a traditional static code analyser.

In this project, although Sonarqube already makes a great security analysis of the code, Gitlab offers templates to run SAST tools within the CI/CD. This is possible because Gitlab company maintain Docker images that incorporate several tools that, depending on the programming languages and projects in the repository Gitlab find, will execute the right tool to run the tests [85]. Tools like semgrep, SpotBugs, MobSF, and others, are used in the project.

```
include:
- template: Jobs/SAST.gitlab-ci.yml
- template: Security/Container-Scanning.gitlab-ci.yml
- template: Security/Secret-Detection.gitlab-ci.yml
```

The configuration is simply including a template and a few variables of configuration. Gitlab then already knows which images and tools needs to use, detects the type of projects and

solutions that exist in the repository, and automatically runs and generates reports about the findings. This is great, since the pain to use this tools becomes extremely reduced, since this doesn't require external tools, licences and servers. One example of the execution can be seen in the figure 6.13.

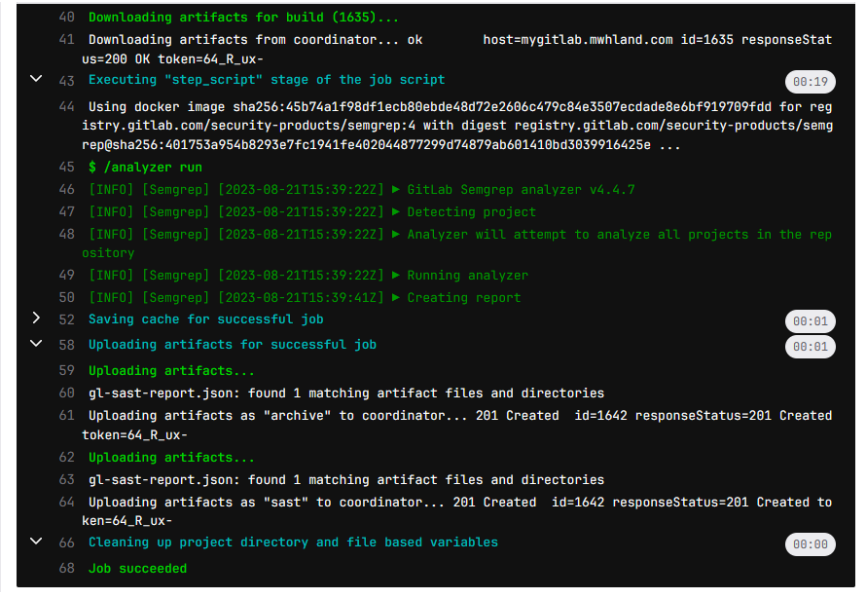

Figure 6.13: Execution of the SAST job

6.11 DAST

Some vulnerabilities require the context of a real code execution to detect variables values and code flow that can lead to such flaws. DAST tools emerged to resolve this problem, by scanning a running version of the application, and try to explore what inputs can cause a malfunction and unexpected behaviour.

Gitlab also offers tools already included with the product for the execution of DAST tools. Again, this demonstrates the simplification that Gitlab offers in terms of DevSecOps [86]. However, some of it features are behind a paywall (for example, the execution of the DAST API Analyser tool lead to licensing error). So, it was decided to explore another open source alternatives, that could be integrated in the pipeline. One of the tools is the OWASP ZAP.

OWASP ZAP, or the OWASP Zed Attack Proxy, is an open-source security testing tool that allows the developers to test the web applications that are being developed to find and fix security vulnerabilities. This tool searches for the most common flaws that are finding, by automate the tests that normally the hackers do.

This is possible by intercepting and manipulating the HTTP requests, crawling the HTML pages to find for possible inputs and forms that can be subject to attack the server with data that can lead to wrong function or open accesses to information not normally available.

In this case, the attack was made to the API for the backend, when it was used the option available in the OWASP ZAP to import an OpenAPI definition, in which .NET 6, with the integration with Swagger, can be easily generated, and after the import, it runs various attacks against the API. For this project in specific, since it is required to have a Bearer

Token in most of the endpoints, and since the OWASP ZAP can be limited in that regard, it was written an Authentication script, together with an HTTP Sender script, in order to authenticate first in the API using the specific route available, and then always pass the Bearer Token between all the requests to the API, so that the API can be properly tested.

```
owaspzap:
    tags: [docker]
    stage: dast
    image:
        name: ghcr.io/zaproxy/zaproxy:stable
    script:
        - zap-api-scan.py -t ${API_TEST_URL} -f openapi

dast_ui:
    tags: [docker]
    stage: dast
    image:
        name: "$SECURE_ANALYZERS_PREFIX/dast:
    $DAST_VERSION$DAST_IMAGE_SUFFIX"
    variables:
        GIT_STRATEGY: none
        DAST_BROWSER_SCAN: "true"
        DAST_ADVERTISE_SCAN: "true"
        DAST_USERNAME_FIELD: "id:username"
        DAST_PASSWORD_FIELD: "id:password"
        DAST_DISABLED_FOR_DEFAULT_BRANCH: "false"
        DAST_SUBMIT_FIELD: css:button[type='submit']
    allow_failure: true
    script:
        - export DAST_WEBSITE=${DAST_WEBSITE:-$(cat environment_url.txt
    )}
        - if [ -z "$DAST_WEBSITE$DAST_API_SPECIFICATION" ]; then echo "
    Either DAST_WEBSITE or DAST_API_SPECIFICATION must be set. See https
    ://docs.gitlab.com/ee/user/application_security/dast/#configuration
    for more details." && exit 1; fi
        - /analyze
    artifacts:
        reports:
        dast: gl-dast-report.json
```

Listing 6.8: DAST Configuration in the Gitlab pipeline

Since it will be run in the CI/CD pipeline, it was used a docker image of the OWASP zap, and generated the config with the automation framework available inside the OWASP ZAP GUI [87]. After generated the configs, the config and script files were then committed inside the repository. The URL is dynamic, since it will be run against another docker image, and so the API can be accessed using the container name, and the credentials are obfuscated using the Gitlab variables, that pass to the OWASP zap using environment variables for the image.

6.12 Prometheus and Grafana

In another section, it was described how the application can generate logs, metrics and traces and export this data to external data sources. This is useful, since the need to gather data such as amount of errors, time to process requests, etc. become easily manageable, leading to a better support experience. But, this alone does not lead to the value wanted for

the observability stack. And, although all the necessary steps to prevent bugs and errors was considered in this project, it is still a relevant step to incorporate the Operate and Monitor phase of the cycle. And so, all the application, and even the deployment in case of Docker and Kubernetes, export specific metrics that are being imported.

In the case of this project, this metrics is being collected to a Prometheus instance. Prometheus is an open-source monitoring and alerting system used to collect and store metrics from various sources like applications and infrastructure. It provides a querying language and alerting capabilities to help monitor the performance and health of systems, particularly in modern, dynamic, and cloud-native environments.

It was decided to install and mount a Prometheus instance on the cloud, with the right access administration and a database to store the data saved, in this case a Postgres one. The prometheus have a web portal that can display the metrics collected, as it is shown in the figure 6.14.

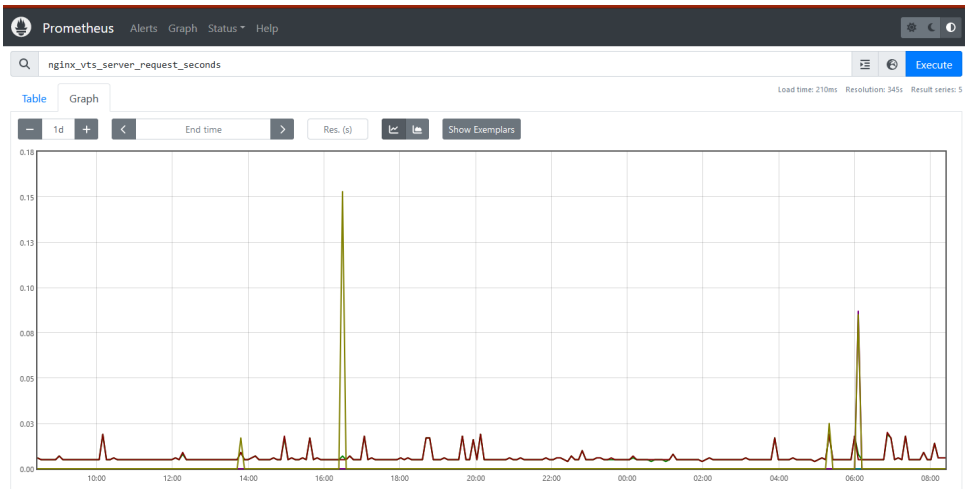

Figure 6.14: Prometheus portal

In order to collect the metrics from the application and deployment, it was configured the Prometheus to use file_sd dynamic configuration of sources. This allows the flexibility to register new sources, without the need to restart the Prometheus instance [88]. In this case, every time a new deployment is made, a new entry is added to a server that is running prometheus, and immediately become available to start tracking and store all the relevant metrics collected.

This is then used in Grafana Dashboards to have an aggregation of the most critical indicators, in order to keep track if the system is working and what is the level of workload, comparing to the available resources available. This allows for proactively create alerts when some metrics start to get values below the minimum considered as good. Not only is the target application being tracked, but also Gitlab and other components, which mean all the infrastructure is being monitored and controlled from Grafana, which leads to better control. One example of a dashboard can be seen in the figure 6.15.

6.13 ERP

In terms of the application that is our case study, initially it was proposed to be made by three students, but since the other members decided to not finish the thesis, this application

Figure 6.15: Grafana dashboard

was mostly made only by the author. The system was created with three components:

- A backend REST API made in .NET 6
- A frontend website using React and NextJS
- A database, in this case was used MySQL

The system have two main windows: the "back office", where the users can change and manipulate the configuration of the system, as it can be seen in the figure 6.16, and the "front office", where users can log in to perform tasks like Invoice Registration, Purchases Registration, etc. The users can insert a password or a PIN to authenticate themselves. The system is designed with big buttons, since this is intended as a POS system.

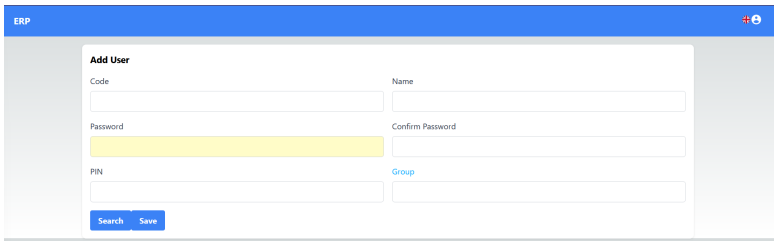

Figure 6.16: ERP Screen Example

In terms of backend, the system was designed with the structure:

- Repository: this layer is responsible to have the code necessary to persist and retrieve the data from the application.
- Model: this layer is responsible to have the code necessary to represent the multiples entities and value objects across the project
- Controller: this layer is responsible to have the methods that can be called by HTTP requests, with the respective inputs necessary.
- DTO: this layer is responsible to translate complex data received in requests, or sent as responses into objects. This can be objects with simpler attributes that the ones found in the model layer.

- Service: this layer is responsible to have the business logic, in terms of interactions with the system.
- Utils: this layer is responsible to have unique code related to functionalities like JSON parse, that is not used in one simple layer.

In terms of Frontend, the system was designed where each TSX file represents a pages or component that can be reused for multiple pages. The NextJS automatically map the folder structure to the URL where each page can be accessed, so no configuration is needed to map the URLs.

Chapter 7

Experimentation and Evaluation

In this chapter it is tested and measured the implementation made and what results were obtained, in order to identify if the solution that was created satisfy the requirements that were initially required.

For this, it is defined a set of hypotheses that addresses the objectives of this thesis, describing what is expected to find. Then, it will be demonstrated what measures were collected in order to validate if the hypotheses are proved to be correct or not.

Finally, a conclusion of the results obtained will be derived, with the right justification for it.

7.1 Problem Description

Development have drastically changed over the course of the years. Software usage is becoming more generalized among companies, and the number of features, integrations and architectures, together with the languages and tools that are provided to the developers have helped created more complex applications. But, complexity brings more risk for the maintenance and security aspects of a system. Besides, as some software have being in the development for several years, and teams are constantly rotating, the codebases can easily become out of control, and increasing the size of it only helps to destroy what is working fine [89].

Some solutions proposed are to increase the number of user tests that are made to the applications, while others is implementing methodologies like code review. However, this is all done manually, and so it causes an increase in the costs of paying developers to maintain software. Besides, error human is always involved, and as the application is more complex, it becomes impossible to test all the scenarios. Automated tools like test frameworks and deployment tools are available, and so it can generate consistent ways to maintain the software. But this is only good to use if the developer can use it as early in the development as possible, since it is when they have the time to fix all the problems before releasing the software [90].

So, how the developers can have confidence that their changes will only impact what is expected, without compromising the rest of the application? How can the task of ensuring the hundred or thousands of scenarios existing in the system being reproduced every time a change is made in order to validate the changes? How can the developer gain early feedback if something goes wrong when changing parts of the code?

7.2 Objectives

The purpose of this dissertation was to implement the DevSecOps methodology in a development cycle process, in order to validate what are the necessary steps that the team must pass in order to implement and have all the advantages from it. So, the principal objective is to explore a development environment and implementation of a lifecycle that includes a DevSecOps principle.

In order to achieve this goal, it is necessary to collect and compare the tools available that can provide the creation of CI/CD pipelines that can be used to automate the process of building, testing and deployment. Then, deploy the most appropriate tool to design and implement a development pipeline capable of running the necessary to steps validate the changes that a developer make when pushing new changes into the repository. After that, it is necessary, given the context of the application being developed, adjust the pipeline to meet the needs of the developers team.

Finally, the evaluation of the pipeline and their performance and effectiveness is necessary to conclude if the solution is suitable for the problem, and if it solves the problem initially proposed.

7.3 Hypotheses

Based on the objectives, the hypotheses were formulated according to test and conclude if the results verifies them. The following hypotheses were established:

- The use of a CI/CD tool covers all the process necessary to compile, test and deploy the application efficiently.
- The use of a CI/CD tool does not cover all the process necessary to compile, test and deploy the application efficiently.

In order to test these hypotheses, it will be collected metrics from the CI/CD tool that can provide sufficient feedback data that can be then analysed in order to derive conclusions that can then validate the hypotheses.

7.4 Identification of indicators and sources of information

In this section is described the indicators to evaluate the work, and the sources of information to be utilized within the context of this work.

7.4.1 Indicators

In order to reach the proposed objectives, it is necessary to define what is the data that will be collected in order to evaluate the project developed.

For this dissertation, in the context of DevSecOps, it was decided the following indicators, that can measure the efficiency of the process in the development cycle:

- Time to run the pipeline;
- Number of bugs detected in the repository;
- Number of vulnerabilities found in the application that was developed;

It is expected that, with these indicators, the data reflect the objectives proposed, and that it can represent sufficient data to derive conclusions about the implementation made.

7.4.2 Sources of Information

In order to retrieve the indicators, the information was extracted from the Gitlab instance that was used during the development of the application, particularly in the pipeline history, and the Sonarqube instance, that have the number of bugs detected. The Gitlab offers Dashboards and metrics that can be used by an admin to consult the data about anything that is related to a specific repository, and so, since it is self-hosted, the data can be extracted. As of the Sonarqube, the graphs that the tool provides, together with the number of bugs that can be tracked every time a commit is made, allows to produce the necessary representation of the number of bugs.

Also, the implementation itself was verified and the strengths and failures from the solution where revised to conclude if the solution meet the hypotheses or not.

7.5 Description of the evaluation methodology

The evaluation methodology defines how the hypotheses defined above will be validated, by justifying how the data will be used to derive conclusions that can be confronted with the expected results. In the following subsections, it is explained how the procedure was conducted.

7.5.1 Methodology - Indicators

During the implementation phase, it was collected the necessary data for each indicator, and stored in a separate file for later, when the implementation phase ends, aggregate the information in order to draw conclusions from it.

With the information from the indicators analysed, it is possible to verify if the process was indeed a motivator to develop a better application.

7.5.2 Methodology - Implementation

While performing the dissertation project, it was analysed if the process was better by using DevSecOps, according to the expected. Also, the implementation give more information and understanding about each objective that was proposed in the dissertation.

7.6 Evaluation Results

After the implementation phase was finished, it was necessary to gather the data collected and then analyse what objectives where fulfilled. For this, it was evaluated the implementation made, and with critical thinking, justified the results obtained.

7.6.1 Results - Indicators

The time to run the pipeline is a good metric to analyse, since the pipeline should be as fast as possible, in order for the developers to have immediate feedback, and also, by consuming

less time, more runs can be performed, and costs can be reduced, which mean that can be made more commits to the repository without compromising the performance.

In the figure 7.1 is shown the overall time to run the pipeline. Is important to take note that the pipeline was developed together with the application, and so it is possible to see the various phases of the pipeline development.

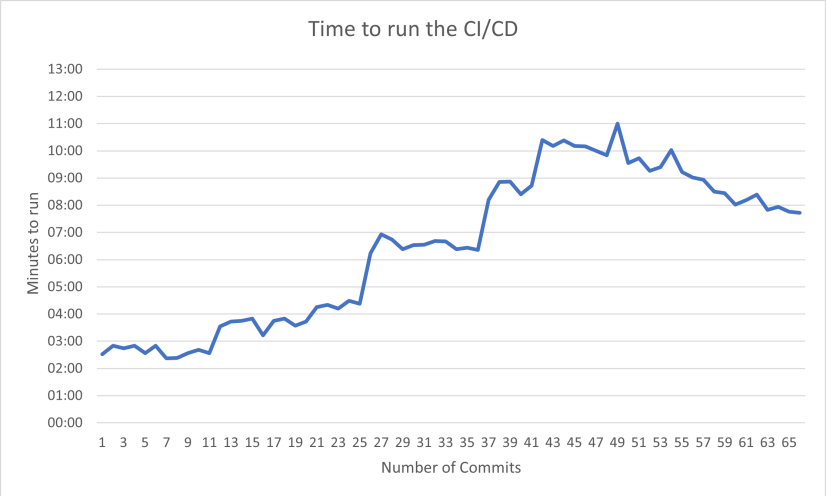

Figure 7.1: Time to run the CI/CD

First, the pipeline only made the build of the application (frontend and backend), having only two jobs to build each part of the application. The pipeline was simple at this time, meaning that the reasons to fail were very few. Then, it was added the step to test the application (both unit tests and also the tests that was included from templates in Gitlab like secret detection). This added already a sequence of the stages, creating the necessary order to make a complete run. Since the number of tests already added on this phase was with some magnitude, represented a larger time to run the pipeline at this phase.

After this, the static code analysis from Sonarqube agent was added. It was necessary to run twice, one for each part of the application, as they use different programming languages and tools, and the rules for each one is different. So, the two jobs were added at the same stage, and the report generated from this tool was published in two different projects in the Sonarqube instance.

Later on, the creation of Docker images was added, together with more tests from templates in Gitlab. In this case, the cache of Gitlab was useful, as it provided better time overtime, since the layers that must be run in each execution is reduced when the base image is already downloaded, together with the dependencies, meaning the build commands was the only part of the image creation that was necessary to run as full.

At a later stage, the pipeline was changed to include the deployment to a test environment, where the IaC tools were configured to recreate the server, if necessary, and push the latest image of the application into the server, and refresh the environment. This only run if the build and tests pass successfully, and the execution of this stage implied that the application should be up and running before considering the deployment a success. The Kubernetes was considered to use in the target server, but was later on moved to a Terraform script, which allowed to recreate all the infrastructure needed in one command.

Finally, the SAST and DAST tests where added, with improvements in the pipeline to become faster, by adding more cache flags, dependencies specification between jobs, etc. The SAST and DAST, although they are running relative fast because it was configured to not exhaust all the possibilities, it can exponentially grow each time a new endpoint in the backend or a page in the frontend is created, since a new set of tests will be necessary to run.

Although the time to run the pipeline was increasing every time a new stage was added, the job still runs fast enough to deliver early feedback to the developer making the commits. A stage that was not considered in this statistics, since it only runs few times is the release stage, where, when a commit is made with a tag that represents a new version of the application, will be run to add to the 'Releases' screen in the Gitlab, with the binaries and images created of the application, as well as a zip that contains the source code at that very moment.

Another indicator that was tracked was the number of bugs that were present in the application, since it can provide information to the developer about faulty code, before it is released and detected during production environment. This metric was gathered using the feedback returned from the Sonarqube analyses.

In the figure 7.2 it is possible to verify the number of bugs during the development. One aspect to take note is that the Sonarqube was only configured after some commits were already made, and after the deployment of Sonarqube, the developer also used the SonarLint integration to have the feedback directly into the IDE.

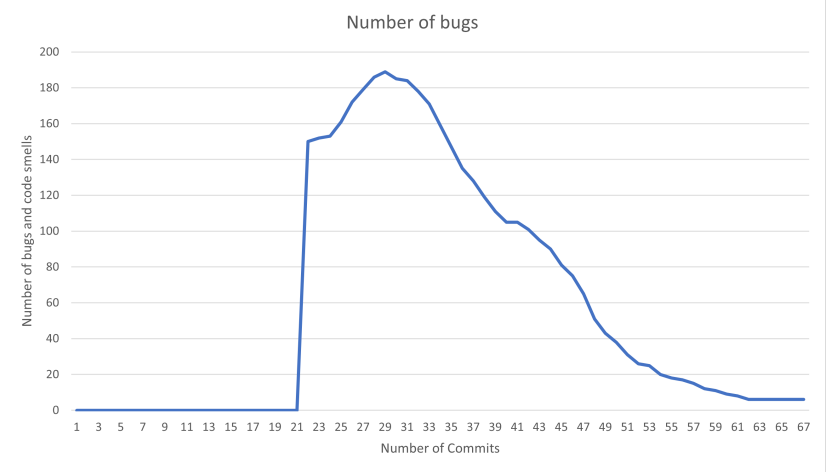

Figure 7.2: Number of bugs detected during the implementation phase

The initial number of bugs were not available since the addition of code analysis was only added in the middle of the implementation. But then, after the developer started to have information about code smells presented in the code already developed, the next changes also fixed the existing ones, while having the Sonarlint to have immediate feedback about what should be changed in the code while developing new features.

These two factors helped to reduce the number of bugs incrementally, since it was possible to add new features, while taking some time to reduce the number of bugs, and adding only few code smells and technical debt. So, the use of a static code analysis tool helped maintain good code quality in the project, while providing the developer the justification and the necessary steps to fix the bugs.

Finally, the number of vulnerabilities was considered as a metric to evaluate the hypotheses, since it was added SAST and DAST tests to determine if the application have into the account the possible security attacks that can be made. Also, since the dissertation as a focus in the security aspect of the application, it is important to also ensure that the application follows the best practices. In the figure 7.3 it is possible to see the evolution of the results obtained from both SAST and DAST tests.

Since this security tests was only added at a later phase of the project, the number of records is reduced. However, is it possible to see that the detection was effective, and by gaining the insight in the CI/CD pipeline, the developer had the information necessary to fix this issues, that is possible to see in the evolution of the next commits.

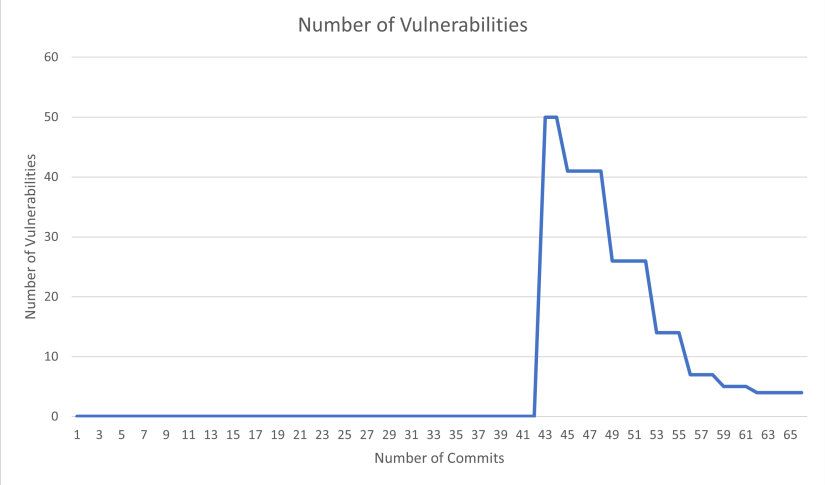

Figure 7.3: Number of vulnerabilities detected during the implementation phase

Although it is possible to further improve the pipeline and the application, the result is overall positive. The factors that contributed for this result was the pipeline that automated and enforced this type of checks, while the Gitlab pipeline tool offer ways to optimize the execution, leading to a lightweight process that contributes to a possible DevSecOps workflow.

7.6.2 Results - Implementation

During the development of the project and the use of the DevSecOps methodology and tools, it was possible to gather and adjust the process, in order to move forward and try to meet the objectives. Regarding this, below are identified the drawbacks along their explanations:

- **Pipeline CI/CD**: With Gitlab to manage the repository, and also to manage the execution of a multi-stage pipeline, it was possible to ensure that every change was considered valid if it allows the application to build, run the tests and deploy. The execution of an automate process that validates the application in several aspects ensures a safer application.
- **Linters**: By using Linters incorporated into the IDE and build tools of the projects, the developer is enforced to maintain a style and best practices that helps reduce the friction between multiple people, with different experience and way to program, separates their way of thinking and starting creating large gaps of code standards that affect readability.

- **Sonarqube**: The Sonarqube usage as a Static Code Analysis tool, most of the Code Smells and bugs are detected, and so the developer have the early feedback about how is changes affects the quality and maintainability of the system. Also, by integrating directly in the IDE, the immediate feedback was possible, so the developer can solve most of the problems without the need to commit the changes before.
- **SAST and DAST**: The incorporation of SAST and DAST tools, included in the Gitlab instance, allowed to test for possible vulnerabilities in the application. These tools were very practical, since the inclusion in the pipeline was very simple, and allowed for better comprehension about the vulnerabilities that can exist in a Web application.
- **OWASP ZAP**: By using the OWASP ZAP, the customization process to test the API was made possible, adding one type of test to execute that can leverage new vulnerabilities from the application.
- **Docker and Kubernetes**: These tools let create agnostic environments that were easily reproducible, which help avoid the misconfiguration that can happen during development and deployment, and stays undocumented until the problems arise.
- **Terraform**: With the terraform tool, the process to start and adjust the configuration of servers become simplified and predictable, since it becomes all scripted and automatic, leaving very less human error involved.
- **OpenTelemetry**: The use of the OpenTelemetry to, not only produce logs, but also to generate metrics and traces of the distributed system, it helps the developer to easily detect where the flaws are, and so the detection and fix of bugs is more rapid.
- **Prometheus and Grafana**: By using tools that collect and produces dashboards that allows to see summarizes, aggregates and alerts of the various metrics that are generated from the infrastructure, the developers and operations teams can work together, and obtain value from the analysis from the data collected.

With these tools, the DevSecOps, that covers the code, building, test, release, deployment, observability and security, was fully enforced, leading to a better developer and support experience for the application that was developed.

Chapter 8

Conclusion

In this chapter is shown what are the results from the objectives proposed in the chapter 1. It is also described what is necessary to evolve from future work, and what this project leaded in terms of results and experience.

8.1 Results

In this section is indicated what results the implementation of the DevSecOps during the development of the ERP instance leaded to.

8.1.1 DevOps

For the pipeline, by automatized a set of steps whenever someone pushes changes to the repository, or a new version is created, it helps gain confidence that the changes made to the product maintain a level of robustness that helps the overall support to the product, since fewer errors occur upon releasing the product.

By structuring the development process and the different phases of testing and releasing necessary to maintain good quality and fast deliveries, while deploying a pipeline that provides to the developer early feedback about all the changes (from him and others) made, the product become more secure. The results from the various metrics demonstrates that the developer can focus on what is important, and by investing some time in tooling and tests, the development cycle have great guaranties of quality and few errors. This leads to be easier to increase the size of the application without compromising the code into a spaghetti style where the next changes are very difficult to predict the impact.

8.1.2 ERP

For the application, the results where that most of the rules and validation were made using the "fail fast" approach, which mean the developer had instant feedback about common errors and being enforced to follow a strict style of programming. Although this can initially limit the creativity of the developer, it helps with maintenance by all the developers working on the same style, and by avoiding common errors to happen in the project. This increases the long run of the development, without compromising the technical problems that can occur from the software growing in size.

8.2 Goals Achieved

With this project, it was possible to analyse the latest trends about the DevOps and specifically the DevSecOps trend, and do a state of the current tools and methodologies that are available in the market. With that info, it was possible to apply in a practical context, and determine how to work with these tools and how it can be all integrated in a single workflow. From these, the advantages and disadvantages were determined to conclude that the DevSecOps is indeed a powerful methodology, with a great set of tools that can be utilized to support the process. It helps teams and leaders to enforce and maintain a high quality of the code and product produced, while being fast in delivering the changes to their customers.

However, the complexity of maintaining a tech stack that compresses more tools, the costs of maintain multiple services, and the overall restrictions in the process can lead to an overhead that for some small teams or applications it can be considered not worth it. Either way, is proven that, by reducing the human errors in the process and automatize all the chain line until the deployment is effectively.

8.3 Future Work and Limitations

For future work, the use of DevSecOps is ensured in this project, but it stills requires constant changes, since the application is at his beginning, and so many constraints are being discovered or decided. One of the tools used (Terraform) recently changed their ToS, which can lead to a shift in terms of the IaC tools being used in the future.

Other work that was considered to be implemented, but not finished, was a portal where a user, like a consultant, could register a new client, and the system would automatically pick the latest image released from the repository, customize the IaC configuration with the data introduced, and deploy a new instance corresponding for that customer. It would create all the servers and connections needed (database, monitoring tool, etc.), and display the URLs and credentials the user could use to connect to that instance. This work was initiated but never finished, but it was considered to implement in .NET, with Gitlab API to retrieve the latest release, create a new repository for the configuration of the new customer, and run the pipeline to trigger the IaC.

In terms of team organization, this project was initially started as a three person developing the same product, each one with their own focus for the dissertation. However, since the project was then only continued by one member, the application of DevSecOps would benefit if it was used by a large team to the end, since the impact of the stages involved could influence how the pipeline is designed. Although, that requirement is always a necessary step when designing a pipeline, since the number of commits and number of persons involved can influence the performance of the pipeline.

Also, the development of the ERP was impacted from this decision, since the initial concept was to three persons develop the application, while each one had a concrete focus for their thesis. With this, it was no longer possible to fully complete the application, and so this thesis only relied on the development made in the DevSecOps process.

This, however, didn't prevent this project to finish, since it was still possible to demonstrate the use of DevSecOps with the latest tools, and still take a look on how, in practice, can be applied when developing an application.

8.4 Final Appreciation

This dissertation allowed to apply some knowledge from some of the classes that were presented in the master's degree of Software Engineering, demonstrating that the degree offers some advanced topics about this field. The work that was done, although with some drawbacks, demonstrated that the DevSecOps methodology and tools can enhance the developer experience, leading to a better product. Also, it was used tools that allows full control of the application, which in other methodologies can be seen as a feature that is only developed at the very end, if the team have time. Compared to the DevSecOps, this leads to lose capability to understand and offer better support of the application.

Besides, the project allowed focusing on a subject that, although is can be considered as a support process in software development, it helps the awareness of the importance of a good code management, and how it can affect how a product can grow. By applying the knowledge necessary to build, test and deploy applications, in an automated, controlled and safer way, the developers can also understand how different parts of the system must be build to reach the client. The important is to understand when and how the tools can be applied in a development cycle.

Bibliography

[1] Scott Carey. *Complexity is Killing Software Developers*. Nov. 2021. url: https://www.infoworld.com/article/3639050/complexity-is-killing-software-developers.html.

[2] OutSystems. *What Is DevOps?* url: https://www.outsystems.com/glossary/what-is-devops/.

[3] Pixelplex. *7 Main Digital Transformation Challenges to Be Aware of in 2023 and Tips on How to Overcome Them*. url: https://pixelplex.io/blog/digital-transformation-challenges/.

[4] SuperOffice. *How digital transformation is driving the customer experience*. Mar. 2023. url: https://www.superoffice.com/blog/digital-transformation/.

[5] Hossein Ashtari. *CI/CD vs. devops: Understanding 8 key differences*. Accessed on 2023-02-03. Apr. 2022. url: https://www.spiceworks.com/tech/devops/articles/cicd-vs-devops/.

[6] Nasreen Azad. "Understanding DevOps critical success factors and organizational practices". In: *2022 IEEE/ACM International Workshop on Software-Intensive Business (IWSiB)*. 2022, pp. 83–90. doi: 10.1145/3524614.3528627.

[7] *Continuous integration*. Accessed on 2023-02-03. url: https://martinfowler.com/articles/continuousIntegration.html.

[8] Saumya Gupta et al. "Prevalence of GitOps, DevOps in Fast CI/CD Cycles". In: *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON)*. Vol. 1. 2022, pp. 589–596. doi: 10.1109/COM-IT-CON54601.2022.9850786.

[9] *11 key benefits of DevOps for your business: Epam anywhere business*. Accessed on 2023-02-03. url: https://anywhere.epam.com/business/devops-benefits-for-business.

[10] *Use four keys metrics like change failure rate to measure your devops performance | google cloud blog*. Accessed on 2023-01-29. url: https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance.

[11] Zaheeruddin Ahmed and Shoba. C. Francis. "Integrating Security with DevSecOps: Techniques and Challenges". In: *2019 International Conference on Digitization (ICD)*. 2019, pp. 178–182. doi: 10.1109/ICD47981.2019.9105789.

[12] Betsy Beyer et al. *Site Reliability Engineering: How Google Runs Production Systems*. 1st. O'Reilly Media, Inc., 2016. isbn: 149192912X.

[13] Gene Kim et al. *The DevOPS handbook: How to create world-class agility, reliability, and security in technology organizations*. Portland, OR: IT Revolution Press, 2016.

[14] Fiorella Zampetti et al. "CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study". In: *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2021, pp. 471–482. doi: 10.1109/ICSME52107.2021.00048.

[15] *Jenkins*. Accessed on 2023-01-29. url: https://www.jenkins.io/.

[16] Saurabh. *What is Jenkins?: Jenkins for continuous integration*. Accessed on 2023-01-30. Nov. 2022. url: https://www.edureka.co/blog/what-is-jenkins/.

[17] *GitLab - The DEVSECOPS platform*. Accessed on 2023-01-30. url: https://about.gitlab.com/.

[18] Codefresh. *GitLab CI: Feature Overview, Tutorial and Best Practices*. Accessed on 2023-01-22. July 2023. url: https://codefresh.io/learn/gitlab-ci/.

[19] *Travis-CI*. Accessed on 2023-01-30. Nov. 2022. url: https://www.travis-ci.com/.

[20] Yamini Priya. *CircleCI Vs Travis CI: Which is the Best CI/CD Tool?* Accessed on 2023-01-20. Jan. 2023. url: https://testsigma.com/blog/circleci-vs-travis-ci/.

[21] *CircleCI - Continuous integration and delivery*. Accessed on 2023-01-30. url: https://circleci.com/.

[22] Jordi Mon Companys. *What is circleci? features, review, Pricing & More: Harness*. Accessed on 2023-01-20. Aug. 2023. url: https://www.harness.io/blog/circleci.

[23] Matthew Portnoy. *Virtualization Essentials*. English. Paperback. Sybex, May 1, 2012, p. 304. isbn: 978-1118176719.

[24] *What is application virtualization?: Vmware glossary*. Accessed on 2023-02-02. Feb. 2023. url: https://www.vmware.com/topics/glossary/content/application-virtualization.html.

[25] Ali Yıldırım and H. Hakan Kilinc. "A Study on the Effect of Virtualization on Productivity". In: *2022 7th International Conference on Computer Science and Engineering (UBMK)*. 2022, pp. 94–97. doi: 10.1109/UBMK55850.2022.9919452.

[26] Tommi Mikkonen et al. "Cargo-Cult Containerization: A Critical View of Containers in Modern Software Development". In: *2022 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. 2022, pp. 93–98. doi: 10.1109/SOSE55356.2022.00017.

[27] Senecca Miller, Travis Siems, and Vidroha Debroy. "Kubernetes for Cloud Container Orchestration Versus Containers as a Service (CaaS): Practical Insights". In: *2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. 2021, pp. 407–408. doi: 10.1109/ISSREW53611.2021.00110.

[28] *Docker Overview*. Accessed on 2023-02-02. Feb. 2023. url: https://docs.docker.com/get-started/overview/.

[29] Mayank Modi. *Top Features of Docker That Stand You Out from the Crowd*. Accessed on 2023-06-22. url: https://www.knowledgehut.com/blog/devops/docker-features.

[30] Podman. *What is Podman?* Accessed on 2023-06-20. url: https://docs.podman.io/en/latest/.

[31] William Henry. *Podman and buildah for Docker users*. Accessed on 2023-06-20. Mar. 2023. url: https://developers.redhat.com/blog/2019/02/21/podman-and-buildah-for-docker-users.

[32] Nestybox. *Nestybox/sysbox: An open-source, next-generation "runc" that empowers rootless containers to run workloads such as Systemd, Docker, Kubernetes, just like VMS*. Accessed on 2023-08-01. url: https://github.com/nestybox/sysbox.

[33] Matej Artac et al. "DevOps: Introducing Infrastructure-as-Code". In: *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. 2017, pp. 497–498. doi: 10.1109/ICSE-C.2017.162.

[34] Matej Artac et al. "DevOps: Introducing Infrastructure-as-Code". In: *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. 2017, pp. 497–498. doi: 10.1109/ICSE-C.2017.162.

[35] Akond Rahman. "Characteristics of Defective Infrastructure as Code Scripts in DevOps". In: *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*. 2018, pp. 476–479.

[36] Akond Rahman. "Anti-Patterns in Infrastructure as Code". In: *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*. 2018, pp. 434–435. doi: 10.1109/ICST.2018.00057.

[37] *Terraform by HashiCorp*. Accessed on 2023-02-02. url: https://www.terraform.io/.

[38] Red Hat Ansible. *Ansible - How it works*. Accessed on 2023-02-02. url: https://www.ansible.com/overview/how-ansible-works.

[39] Rahul Awati and Meredith Courtemanche. *What is the ANSIBLE IT automation platform? – TechTarget definition*. Accessed on 2023-08-01. Mar. 2023. url: https://www.techtarget.com/searchitoperations/definition/Ansible.

[40] Chef. *Chef Software DevOps Automation Solutions*. Accessed on 2023-08-01. url: https://www.chef.io/.

[41] PeiBo Xie and Yanhong Wu. "Audit Computer Information Management and Software Development System in the Internet Era". In: *2021 International Wireless Communications and Mobile Computing (IWCMC)*. 2021, pp. 1993–1996. doi: 10.1109/IWCMC51323.2021.9498823.

[42] Arif Onan, Eda Gürlen, and Sevgi Turan. "Do moodle reports and logs meet the needs of educational supervision?" In: *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*. 2014, pp. 1–5. doi: 10.1109/CISTI.2014.6877087.

[43] Sepehr Amir-Mohammadian and Afsoon Yousefi Zowj. "Towards Concurrent Audit Logging in Microservices". In: *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. 2021, pp. 1357–1362. doi: 10.1109/COMPSAC51774.2021.00191.

[44] Datadog. *Datadog*. Accessed on 2023-06-30. Oct. 2016. url: https://www.datadoghq.com/product/.

[45] Grafana. *Grafana: The open observability plataform*. Accessed on 2023-07-13. url: https://grafana.com/.

[46] Netwrix. *Powerful Data Security Made Easy*. Accessed on 2023-06-30. url: https://www.netwrix.com/.

[47] Radek Fujdiak et al. "Managing the Secure Software Development". In: *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. 2019, pp. 1–4. doi: 10.1109/NTMS.2019.8763845.

[48] Anivesh Panjiyar and Debanjan Sadhya. "Defending against code injection attacks using Secure Design Pattern". In: *2022 29th Asia-Pacific Software Engineering Conference (APSEC)*. 2022, pp. 570–571. doi: 10.1109/APSEC57359.2022.00085.

[49] Goran Piskachev, Ranjith Krishnamurthy, and Eric Bodden. "SecuCheck: Engineering configurable taint analysis for software developers". In: *2021 IEEE 21st International Working Conference on Source Code Analysis and Manipulation (SCAM)*. 2021, pp. 24–29. doi: 10.1109/SCAM52516.2021.00012.

[50] Antonio Nehme et al. "Securing Microservices". In: *IT Professional* 21.1 (2019), pp. 42–49. doi: 10.1109/MITP.2018.2876987.

[51] Thomas Ryan Devine et al. "SREP+SAST: A Comparison of Tools for Reverse Engineering Machine Code to Detect Cybersecurity Vulnerabilities in Binary Executables". In: *2022 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2022, pp. 862–869. doi: 10.1109/CSCI58124.2022.00156.

[52] Manohar Marandi, A. Bertia, and Salaja Silas. "Implementing and Automating Security Scanning to a DevSecOps CI/CD Pipeline". In: *2023 World Conference on Communication & Computing (WCONF)*. 2023, pp. 1–6. doi: 10.1109/WCONF58270.2023.10235015.

[53] Hermawan Setiawan, Lytio Enggar Erlangga, and Ido Baskoro. "Vulnerability Analysis Using The Interactive Application Security Testing (IAST) Approach For Government X Website Applications". In: *2020 3rd International Conference on Information and Communications Technology (ICOIACT)*. 2020, pp. 471–475. doi: 10.1109/ICOIACT50329.2020.9332116.

[54] Sudip Sengupta. *SAST, DAST, IAST, RASP Explained in Short*. Nov. 2022. url: https://crashtest-security.com/sast-dast-iast-rasp/.

[55] Ivan Homola. *OWASP Zap: 8 Core Features (Pros & Cons)*. Accessed on 2023-07-11. Feb. 2023. url: https://www.codiga.io/blog/owasp-zap/.

[56] PortSwigger. *Burp Suite Professional Features*. Accessed on 2023-07-10. url: https://portswigger.net/burp/pro/features.

[57] Abdullah A. Al-Ghofaili and Majed A. Al-Mashari. "ERP system adoption traditional ERP systems vs. cloud-based ERP systems". In: *Fourth edition of the International Conference on the Innovative Computing Technology (INTECH 2014)*. 2014, pp. 135–139. doi: 10.1109/INTECH.2014.6927770.

[58] F. Robert Jacobs and F.C. 'Ted' Weston. "Enterprise resource planning (ERP)—A brief history". In: *Journal of Operations Management* 25.2 (Dec. 2006), pp. 357–363. doi: 10.1016/j.jom.2006.11.005. url: https://doi.org/10.1016/j.jom.2006.11.005.

[59] Ming Jia Lee, Whee Yen Wong, and Meei Hao Hoo. "Next era of enterprise resource planning system review on traditional on-premise ERP versus cloud-based ERP: Factors influence decision on migration to cloud-based ERP for Malaysian SMEs/SMIs". In: *2017 IEEE Conference on Systems, Process and Control (ICSPC)*. 2017, pp. 48–53. doi: 10.1109/SPC.2017.8313020.

[60] Neslihan Küçükateş Ömüral and Onur Demirörs. "Effort Estimation for ERP Projects — A Systematic Review". In: *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2017, pp. 96–103. doi: 10.1109/SEAA.2017.68.

[61] Ishan Aggarwal, A. Anirudh, and Raviteja Buddala. "Literature Review: ERP Implementation in Various Industries". In: *2021 Innovations in Power and Advanced Computing Technologies (i-PACT)*. 2021, pp. 1–6. doi: 10.1109/i-PACT52855.2021.9696962.

[62] *Best ERP systems in 2023: Compare reviews on 570+ | G2*. Accessed on 2023-02-02. url: https://www.g2.com/categories/erp-systems.

[63] Peter A. Koen et al. "1 Fuzzy Front End : Effective Methods , Tools , and Techniques". In: 2002.

[64] Barry W. Boehm. *Software Engineering Economics*. English. Hardcover. Prentice Hall, Nov. 1, 1981, p. 767. isbn: 978-0138221225.

[65] *2022 Accelerate State of DevOps Report*. Accessed on 2023-02-02. Sept. 2022. url: https://cloud.google.com/blog/products/devops-sre/dora-2022-accelerate-state-of-devops-report-now-out.

[66] *Enterprise devops report 2020–2021*. Accessed on 2023-02-02. Sept. 2020. url: https://azure.microsoft.com/en-us/resources/enterprise-devops-report-20202021/.

[67] Alexander Osterwalder and Yves Pigneur. *Business model generation*. en. Chichester, England: John Wiley & Sons, June 2010.

[68] James R Wixson. "2 Function Analysis and Decomposistion using Function Analysis Systems Technique". In: *INCOSE International Symposium*. Vol. 9. 1. Wiley Online Library. 1999, pp. 800–805.

[69] Thomas L. Saaty. *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation (Decision Making Series)*. English. Hardcover. McGraw-Hill, Jan. 1980, p. 287. isbn: 978-0070543713. url: https://lead.to/amazon/com/?op=bt&la=en&cu=usd&key=0070543712.

[70] Thomas L. Saaty. *Decision Making for Leaders: The Analytic Hierarchy Process for Decisions in a Complex World*. English. Kindle Edition. RWS Publications, Feb. 8, 2013, p. 342. url: https://lead.to/amazon/com/?op=bt&la=en&cu=usd&key=B00BDBNZDY.

[71] Git. *Git –distributed-is-the-new-centralized*. Accessed on 2023-01-15. url: https://git-scm.com/.

[72] Microsoft. *Visual Studio: IDE and Code Editor for Software Developers and Teams*. Accessed on 2023-05-16. url: https://visualstudio.microsoft.com/.

[73] Microsoft. *Visual Studio Code - Code Editing. Redefined*. Accessed on 2023-05-16. url: https://code.visualstudio.com/.

[74] Postman. *What is Postman?* Accessed on 2023-04-29. url: https://www.postman.com/product/what-is-postman/.

[75] PuTTY. *PuTTY: a free SSH and Telnet client*. Accessed on 2023-05-17. url: https://www.chiark.greenend.org.uk/~sgtatham/putty/.

[76] WinSCP. *WinSCP: Free SFTP, SCP, S3 and FTP client for Windows*. Accessed on 2023-07-02. url: https://winscp.net/eng/index.php.

[77] Atlassian. *Gitflow workflow: Atlassian Git Tutorial*. Accessed on 2023-05-29. url: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow.

[78] Github. *GitHub flow*. Accessed on 2023-05-29. url: https://docs.github.com/en/get-started/quickstart/github-flow.

[79] Gursimran Singh. *What is Observability?* Accessed on 2023-06-24. Mar. 2023. url: https://www.xenonstack.com/insights/what-is-observability.

[80] Gitlab. *Install GitLab with the Linux package*. Accessed on 2023-04-16. url: https://docs.gitlab.com/omnibus/installation/.

[81] BillWagner. *Tutorial: Write your first analyzer and code fix*. Accessed on 2023-06-15. url: https://learn.microsoft.com/en-us/dotnet/csharp/roslyn-sdk/tutorials/how-to-write-csharp-analyzer-code-fix.

[82] Gitlab. *Code Quality*. Accessed on 2023-07-01. url: https://docs.gitlab.com/ee/ci/testing/code_quality.html.

[83] Docker. *Docker Compose overview*. Accessed on 2023-05-24. Aug. 2023. url: https://docs.docker.com/compose/.

[84] Gitlab. *GitLab Container Registry*. Accessed on 2023-07-15. url: https://docs.gitlab.com/ee/user/packages/container_registry/.

[85] Gitlab. *Static Application Security Testing (SAST)*. Accessed on 2023-07-09. url: https://docs.gitlab.com/ee/user/application_security/sast/.

[86] Gitlab. *Dynamic Application Security Testing (DAST)*. Accessed on 2023-07-10. url: https://docs.gitlab.com/ee/user/application_security/dast/.

[87] ZAP. *ZAP Docker User Guide*. Accessed on 2023-07-18. url: https://www.zaproxy.org/docs/docker/about/.

[88] Prometheus. *Configuration: Prometheus*. Accessed on 2023-07-17. url: https://prometheus.io/docs/prometheus/latest/configuration/configuration/#file_sd_config.

[89] Dragos Dobrean. "Automatic Examining of Software Architectures on Mobile Applications Codebases". In: *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2019, pp. 595–599. doi: 10.1109/ICSME.2019.00094.

[90] Yuhoon Ki and Meongchul Song. "An Open Source-Based Approach to Software Development Infrastructures". In: *2009 IEEE/ACM International Conference on Automated Software Engineering*. 2009, pp. 525–529. doi: 10.1109/ASE.2009.73.

Appendix A

AHP Analysis

Pairwise Comparison

	Security	Cloud Integration	Plugins	Reports	Price
Security	1	1	1	3	1
Cloud Integration	1	1	1/5	5	1
Plugins	1	5	1	5	3
Reports	1/3	1/5	1/5	1	1/5
Price	1	1	1/3	5	1
SOMA	4 1/3	8 1/5	2 3/4	19	6 1/5

Normalized comparison matrix and estimated weights

	Security	Cloud Integration	Plugins	Reports	Price	Weights
Security	0,2308	0,1220	0,3659	0,1579	0,1613	0,2076
Cloud Integration	0,2308	0,1220	0,0732	0,2632	0,1613	0,1701
Plugins	0,2308	0,6098	0,3659	0,2632	0,4839	0,3907
Reports	0,0769	0,0244	0,0732	0,0526	0,0323	0,0519
Price	0,2308	0,1220	0,1220	0,2632	0,1613	0,1798
SOMA	1,00	1,00	1,00	1,00	1,00	

Consistency Test

STEP1		1	1	1	3	1	0,2076	1,10
		1	1	1/5	5	1	0,1701	0,89
		1	5	1	5	3	0,3907	2,25
		1/3	1/5	1/5	1	1/5	0,0519	0,27
		1	1	1/3	5	1	0,1798	0,95
STEP2		5,3179						
		5,2623						
		5,7526						
		5,1889						
		5,2665						
STEP3		5,3577						
STEP4			Consistency		IC	(lambdamax-n)/n-1	0,08941279	
STEP5	CR=IC/RI			0,08				

Security	Jenkins	Gitlab	Travis CI	Circle CI
Jenkins	1	1/5	1/5	1
Gitlab	5	1	1	1
Travis CI	5	1	1	3
Circle CI	1	1	1/3	1
Total	12,0000	3,2000	2,5333	6,0000

Cloud Integration	Jenkins	Gitlab	Travis CI	Circle CI
Jenkins	1	1	1/5	1/3
Gitlab	1	1	1/3	1/3
Travis CI	5	3	1	1
Circle CI	3	3	1	1
Total	0,4167	8,0000	2,5333	2,6667

Plugins	Jenkins	Gitlab	Travis CI	Circle CI
Jenkins	1	1	3	3
Gitlab	1	1	3	1/3
Travis CI	1/3	1/3	1	1
Circle CI	1/3	3	1	1
Total	2,6667	5,3333	8,0000	5,3333

Reports	Jenkins	Gitlab	Travis CI	Circle CI
Jenkins	1	1	3	1
Gitlab	1	1	3	3
Travis CI	1/3	1/3	1	1
Circle CI	1	1/3	1	1
Total	3,3333	2,6667	8,0000	6,0000

Price	Jenkins	Gitlab	Travis CI	Circle CI
Jenkins	1	1	5	5
Gitlab	1	1	5	5
Travis CI	1/5	1/5	1	1
Circle CI	1/5	1/5	1	1
Total	2,4000	2,4000	12,0000	12,0000

Security	Jenkins	Gitlab	Travis CI	Circle CI	Weights
Jenkins	0,08333	0,06250	0,07895	0,16667	0,09786
Gitlab	0,41667	0,31250	0,39474	0,16667	0,32264
Travis CI	0,41667	0,31250	0,39474	0,50000	0,40598
Circle CI	0,08333	0,31250	0,13158	0,16667	0,17352
Total	1,0000	1,0000	1,0000	1,0000	1,0000

Cloud Integration	Jenkins	Gitlab	Travis CI	Circle CI	Weights
Jenkins	0,10000	0,12500	0,07895	0,12500	0,10724
Gitlab	0,10000	0,12500	0,13158	0,12500	0,12039
Travis CI	0,50000	0,37500	0,39474	0,37500	0,41118
Circle CI	0,30000	0,37500	0,39474	0,37500	0,36118
Total	1,0000	1,0000	1,0000	1,0000	1,0000

Plugins	Jenkins	Gitlab	Travis CI	Circle CI	Weights
Jenkins	0,37500	0,18750	0,37500	0,56250	0,37500
Gitlab	0,37500	0,18750	0,37500	0,06250	0,25000
Travis CI	0,12500	0,06250	0,12500	0,18750	0,12500
Circle CI	0,12500	0,56250	0,12500	0,18750	0,25000
Total	1,0000	1,0000	1,0000	1,0000	1,0000

Reports	Jenkins	Gitlab	Travis CI	Circle CI	Weights
Jenkins	0,30000	0,37500	0,37500	0,16667	0,30417
Gitlab	0,30000	0,37500	0,37500	0,50000	0,38750
Travis CI	0,10000	0,12500	0,12500	0,16667	0,12917
Circle CI	0,30000	0,12500	0,12500	0,16667	0,17917
Total	1,0000	1,0000	1,0000	1,0000	1,0000

Price	Jenkins	Gitlab	Travis CI	Circle CI	Weights
Jenkins	0,41667	0,41667	0,41667	0,41667	0,41667
Gitlab	0,41667	0,41667	0,41667	0,41667	0,41667
Travis CI	0,08333	0,08333	0,08333	0,08333	0,08333
Circle CI	0,08333	0,08333	0,08333	0,08333	0,08333
Total	1,0000	1,0000	1,0000	1,0000	1,0000

Consistency Test

Security	STEP1						STEP2	STEP3	STEP4	STEP5
	1	1/5	1/5	1	0,09786	0,41711	4,26218	4,26509	0,08836	0,0789
	5	1	1	1	0,32264	1,39145	4,31266			
	5	1	1	3	0,40598	1,73849	4,28224			
	1	1	1/3	1	0,17352	0,72935	4,20326			
Cloud Integration	STEP1						STEP2	STEP3	STEP4	STEP5
	1	1/5	1/5	1	0,09786	0,43026	4,01227	4,03284	0,01095	0,0789
	5	1	1	1	0,32264	0,48509	4,02914			
	5	1	1	3	0,40598	1,66974	4,0608			
	1	1	1/3	1	0,17352	1,45526	4,02914			
Plugins	STEP1						STEP2	STEP3	STEP4	STEP5
	1	1	3	3	0,37500	1,75000	4,66667	4,66667	0,22222	0,19841
	1	1	3	1/3	0,25000	1,08333	4,33333			
	1/3	1/3	1	1	0,12500	0,58333	4,66667			
	1/3	3	1	1	0,25000	1,25000	5			
Reports	STEP1						STEP2	STEP3	STEP4	STEP5
	1	1	3	1	0,30417	1,25833	4,13699	4,15515	0,05172	0,04618
	1	1	3	3	0,38750	1,61667	4,17204			
	1/3	1/3	1	1	0,12917	0,53889	4,17204			
	1	1/3	1	1	0,17917	0,74167	4,13953			
Price	STEP1						STEP2	STEP3	STEP4	STEP5
	1	1	5	5	0,41667	1,66667	4	4	0	0
	1	1	5	5	0,41667	1,66667	4			
	1/5	1/5	1	1	0,08333	0,33333	4			
	1/5	1/5	1	1	0,08333	0,33333	4			

Matrixes with global priorities

	Security	Cloud Integration	Plugins	Reports	Price	Si'
Jenkins	0,0979	0,1072	0,3750	0,3042	0,4167	0,2076
Gitlab	0,3226	0,1204	0,2500	0,3875	0,4167	0,1701
Travis CI	0,4060	0,4112	0,1250	0,1292	0,0833	0,3907
Circle CI	0,1735	0,3612	0,2500	0,1792	0,0833	0,0519
						0,1798

Jenkins	0,2758
Gitlab	0,2801
Travis CI	0,2247
Circle CI	0,2194