



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona



DEVSECOPS: S-SDLC

ORIOL PORTELL PARERAS

Thesis supervisor: FEDERICO PÉREZ MARINA (PUNTO-FA, SL)

Tutor: RENÉ SERRAL GRACIÀ (Department of Computer Architecture)

Degree: Bachelor Degree in Informatics Engineering (Software Engineering)

Thesis report

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

28/06/2023

Resum

L'objectiu principal d'aquesta tesi és veure com s'incorpora la seguretat a DevOps en un entorn corporatiu. En concret, aquesta tesi busca veure com implementar S-SDLC. A més a més, la tesi mostra la implementació d'un CI/CD ben fet.

Durant el projecte s'han implementat noves eines que faciliten el desenvolupament segur i de qualitat al programador durant la fase de desenvolupament.

Resumen

El objetivo principal de esta tesis es examinar cómo se incorpora la seguridad a DevOps en un entorno corporativo. Específicamente, esta tesis busca ver cómo implementar S-SDLC (Ciclo de Vida de Desarrollo de Software Seguro). Además, la tesis muestra la implementación de un CI/CD (Integración Continua/Entrega Continua) bien realizado.

Durante el proyecto, se han implementado nuevas herramientas que facilitan el desarrollo seguro y de calidad para el programador durante la fase de desarrollo.

Abstract

The main objective of this thesis is to examine how security is incorporated into DevOps in a corporate environment. Specifically, this thesis aims to explore how to implement S-SDLC (Secure Software Development Life Cycle). Additionally, the thesis demonstrates the implementation of a well-executed CI/CD (Continuous Integration/Continuous Delivery).

During the project, new tools have been implemented to facilitate secure and high-quality development for the programmer during the development phase

Contents

1	Context and Justification	7
1.1	Introduction	7
1.2	Mango	7
1.3	DevSecOps	7
1.4	Devops	8
1.4.1	Agile	8
1.4.2	Devops and Agile	9
1.5	S-SDLC	9
1.6	Sec in DevSecOps	10
1.7	Problem	10
1.8	Actors Involved	11
2	Justification	12
2.1	Existing Solutions	12
2.2	Justification of the choice	12
3	Outreach	14
3.1	Objectives	14
3.2	Non-functional requirements	15
3.3	Risks	15
4	Methodology	16
4.1	Kanban	16
4.2	Version tracking and control	17
4.3	Tool to manage Agile projects	17
5	Time planning	18
6	Project estimation	19
6.1	Task definition	19
6.1.1	Project Management	19
6.1.2	Analyze and explain the Security of Mango	20
6.1.3	Automatic Static Analysis	21
6.1.4	Automatic Dynamic Analysis	23
6.1.5	Securing Kubernetes Deploys	25
6.1.6	Final documentation	26
6.2	Gantt chart	28
7	Risk management	30
7.1	Tasks and risks	30

8	Budget	33
8.1	Identification of costs	33
8.1.1	Human resources	33
8.1.2	Material resources	34
8.1.3	Indirect resources	35
8.1.4	Contingency Costs	36
8.1.5	Unforeseen costs	36
8.1.6	Final budget	36
8.2	Management control	37
9	Sustainability report	38
9.1	Economic dimensions	38
9.2	Environmental dimensions	38
9.3	Social dimensions	38
10	Integration of knowledge	40
11	Identification of laws and regulations	41
12	Technologies and concepts	42
12.1	Jenkins	42
12.2	Kubernetes	43
12.3	Sonarqube	44
12.4	Zap	45
12.5	Sonatype	46
12.6	Elasticsearch	46
12.7	Kibana	47
12.8	Grafana	48
12.9	NPM	48
12.10	Maven	49
12.11	Gitflow	50
12.12	Gitops	51
12.13	Version Control	52
12.14	Webhooks	52
12.15	Observability	53
12.16	Monitoring	53
13	Analyze and explain the security of Mango	54
13.1	Document the current security	54
13.1.1	CI/CD Implementation	54
13.1.2	Security implementation	59
13.1.3	ArgoCD	61
13.1.4	Monitoring	62
13.2	Explain possible improvements in the Security of Mango	64

14 Automatic Static Analysis	65
14.1 Set up Sonarqube Credentials in Jenkins	65
14.2 Create new methods in the Jenkins library	65
14.3 Import the SSL certificate in the necessary Docker	66
14.4 Investigate how Sonarqube quality gate works	67
14.5 Create a method in Jenkins library to implement the quality gate	68
14.6 Create a method in the Jenkins library to write messages in bitbucket PRs	70
14.7 Implement an API to obtain security vulnerabilities from Sonar .	70
14.8 Create confluence to allow execute Sonarqube9 locally	71
14.9 Problems related to Sonarqube	71
15 Automatic Dynamic Analysis	73
15.1 Test in local Zap	73
15.2 Create a new dockerfile to run zap and a script to obtain the result	73
15.3 Figure out how to get an actualized list of all the project that must be analyzed	74
15.4 Create a Jenkins job that is executed every night	74
15.5 Create a script to collect all the vulnerabilities information	75
16 Securing Kubernetes Deploys	77
16.1 New stage gitops pipeline	78
16.2 Kibana dashboard with vulnerabilities	78
17 Security Score Card	80
17.1 CronJob to Insert SSC information in ELK	80
17.2 Create dashboard in Kibana	82
18 Showcase Desgin System DevOps Configuration	84
18.1 Showcase Design System Configuration	85
18.2 Showcase Design System Kubernetes	85
19 Changes in the planification	88
20 Final budget	91
21 Achievement of technical competencies	93
21.1 Relationship of the project with the degree	93
22 Conclusions	94
23 Next steps	95
23.1 Prisma Cloud	95

List of Figures

1	Agile cycle [14]	9
2	Backend security considerations [15]	11
3	Kanban board [16]	16
4	Gantt diagram	29
5	Jenkinsfile of a specific project.	55
6	Common Jenkinsfile.	55
7	Frontend and backend pipeline CI CD	57
8	Frontend and backend pipeline CI CD	57
9	Frontend and backend pipeline CI CD	58
10	Node pod template	59
11	Jenkins credentials	65
12	Jenkins stage	66
13	Quality gate conditions	68
14	Overall code in sonarqube	69
15	New code in sonarqube	69
16	Dashboard Kibana Sonarqube	71
17	Dockerfile zap	74
18	Jenkins Job Zap	75
19	Zap podtempalte.	76
20	Zap Kibana dashboard.	77
21	Checkov Kibana dashboard.	79
22	Cronjob security scorecard.	81
23	Dockerfile security-score-card	82
24	Dockerfile security-score-card	83
25	Virtualservice showcase	86

1 Context and Justification

1.1 Introduction

The final degree thesis "DevSecOps: S-SDLC" belongs to the company Mango and the bachelor's in Computer Engineering is offered by the Facultat Informatica de Barcelona, with a specialization in Software Engineering.

This project pretends to analyze the whole DevSecOps system of Mango and to expand it by implementing new tools that improve the quality of the code and make it easier for Software Engineers to develop projects.

1.2 Mango

As I said above, I am going to do my thesis on Mango, a big retail Spanish Company focused on implementing new technologies such as RFID and improving its online business.

Particularly I am going to be part of the Software Engineering Department in the DevSecOps team. This team is formed by three persons apart from me. The whole department is formed by over forty software Engineers that program mainly using Spring Boot - Kotlin and Angular - Typescript.

1.3 DevSecOps

DevSecOps is the combination of the words Development, Security and Operations, but the first methodology used was just DevOps. Among what a lot of people think DevOps is, it is a methodology used to produce faster and higher-quality developments.

DevOps is a methodology used to speed the delivery of higher-quality software. It allows the software team to automate the majority of the steps of software development and IT operations teams.

It permits the integration of the efforts of development and IT operations teams into one unique team, two groups that traditionally worked separately from each other.

In practice, DevOps permit developers to work effectively with agile methodologies, since it speeds up the amount of time to deploy, so programmers can produce multiple releases every day, instead of producing only one big release every several months or even years. [12]

1.4 Devops

1.4.1 Agile

The Agile methodology is a way to manage a project by breaking it up into several phases and iterate over them. It involves constant collaboration between business part and developers.

When using Agile teams won't try to develop the whole project in a unique iteration. They will split the whole project into sprints and work in every sprint on different use cases and problems of the development of the project. After every sprint, the development team will present what they have been working on to the business part to check if they are meeting the requirements presented initially. If they are not fitting all the requirements or requirements have changed programmers will make a note and produce the necessary improvements.

In every sprint the developer team will iterate over the next steps:

- **Planning:** In the planning stage the developer team imagined the whole project and made the sprint backlog. Furthermore they calculated the average time of release product and determined the necessary technologies.
- **Building:** During this phase the team focuses on making the planned tasks during the Planning phase. It can involve developing use cases or provide solutions to specific problems.
- **Testing:** This step is the most vital. During this phase the developer team will ensure that all they have been producing work as it means to be. Furthermore this phase permit teams to check the quality of the code and find vulnerabilities.
- **Delivery:** After performing a series of testing, user training, enhancing system functionalities developers team release the software.
- **Feedback:** In this step the software team present to the client the product produced during the sprint. This phase permit a continuous flow of review and up-gradation of the software.

Agile is more open to changes than waterfall because if a concrete functionality of the application is not meeting the requirements, it will be noticed in the feedback of the sprint and the development team will be able to change it for the following sprint.

To sum up, Agile methodology has multiple advantages compared to Waterfall, it increases the alignment between stakeholders and developers, increase the quality of the product, since it is reviewed every sprint in small

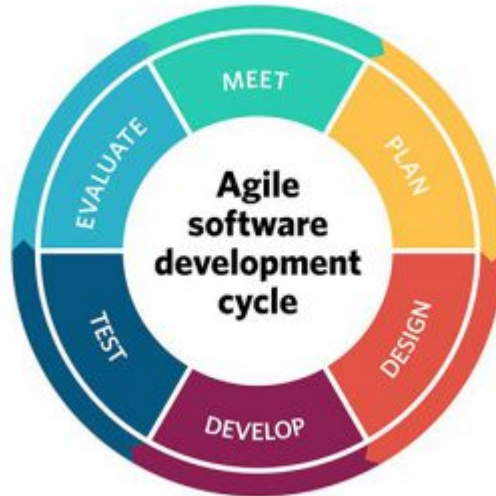


Figure 1: Agile cycle [14]

portions and permits companies to test new functionalities in the market without implementing them completely, if an initial test fit in the customers they will finish the development in future sprints. [3]

1.4.2 Devops and Agile

When in the early 2000, companies started to notice the multiple advantages of developing software projects using Agile a new need was born. Development teams noticed the necessity of a team that focuses on solving specific problems and performing specific tasks such as deploying the project and maintain them. DevOps permit to increase the velocity of developers teams when producing new releases and functionalities.

1.5 S-SDLC

The goal of the S-SDLC is to ensure that security is integrated into every aspect of the software development process, from design to deployment and beyond. By taking a proactive and comprehensive approach to security, organizations can reduce the risk of security breaches and other security-related incidents. S-SDLC includes several stages, including planning, design, development, testing, deployment, and maintenance. Each stage includes specific activities and controls that are designed to ensure that security is integrated into every aspect of the development process.

1.6 Sec in DevSecOps

DevSecOps, as its name suggests, involves the integration of security practices into the DevOps process. This means that security is no longer an afterthought, but is instead integrated into every stage of the development lifecycle. This includes designing and building secure systems, testing and verifying the security of these systems, and deploying them in a secure manner.

One of the key aspects of DevSecOps is the use of automatic security checks. This means that before code is merged or deployed to specific git branches, it undergoes a series of automated security checks. These checks can include everything from vulnerability scanning and penetration testing to code analysis and static code analysis. By performing these checks automatically, developers can identify and address security issues more quickly, reducing the risk of security vulnerabilities making their way into production systems.

1.7 Problem

When developing most of the teams use to leave for last Security. That is because usually programmers don't have much time to focus on this problem instead they have to focus on developing new functionalities. Furthermore not all security can be considered as part of the developer team, in a lot of cases operation team should take into consideration how to protect the system to avoid hackers accessing servers, or accessing the company network. Moreover, security vulnerabilities are not static, hackers always find new ways to access servers or new ways to take advantage, so every time that a release is performed vulnerabilities must be checked. Also, vulnerabilities must be checked routinely.

For example, if we want to check all the vulnerabilities of an API we should access a database that contains all the possible vulnerabilities and check one by one that we are accomplishing all the security recommendations. Also, we should check the vulnerabilities related to the language and framework that is used to program the API. Furthermore, when we deploy the API we must ensure that a hacker can't access the server where is deployed. Moreover, we must check all the URIs related to the API to ensure that all of them are correctly secured and no one can access to sensitive information.

As we can observe, preventing someone to hack our system and our applications can be really tedious if it has to be performed manually. That is why DevSecOps make sense, integrate technologies and tools that can perform this process automatically can speed up the process of developing new releases and also raise the quality of the product.

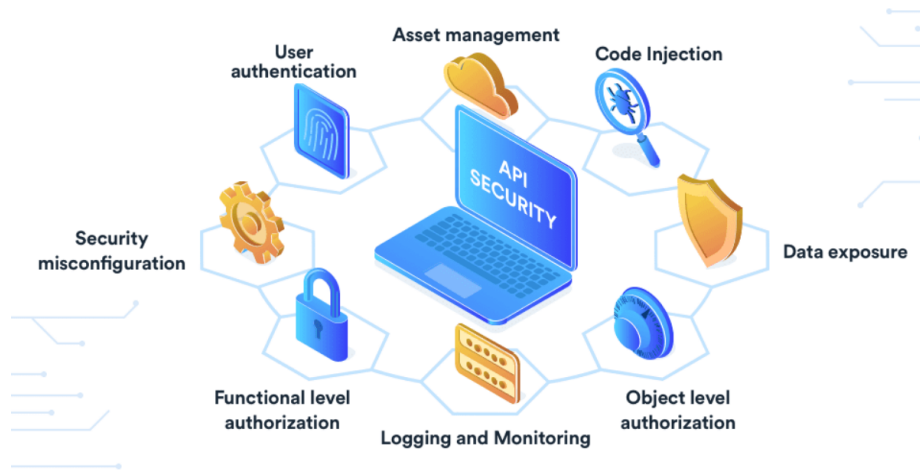


Figure 2: Backend security considerations [15]

1.8 Actors Involved

In this section i am going to talk about all the stakeholders, we have to take into consideration that this thesis is going to be done in Mango, a retail company focused on improve its online business and implement new systems such as RFID. Specifically, this thesis is going to be developed in the software engineering department. It is mainly focused on developing software for the company, i.e RFID, clock in application or tickets application.

- **Developers:** Software developers that develop new functionalities in the department. They are going to use the implemented tools to increase the quality of the code.
- **Operation team:** They are going to prepare the necessary machines to implement the new tools.
- **Security squad:** This team is going to participate in the selection of the tools and will help the team to understand vulnerabilities.
- **Business team:** Developments will be faster and better, so they will get a better product.
- **Thesis director:** Director of the thesis is going to be Rene Serral. He is going to supervise the thesis.

2 Justification

2.1 Existing Solutions

Nowadays exists multiple tools that allow you to automatically check the quality of the code and the different vulnerabilities depending on the technology you are using. The most common and important are:

- Sonarqube: [10] This technology is used for a lot of companies to analyse their applications. It has a free version that permits software teams to obtain information about vulnerabilities, code smells, coverage, bugs and more. Allow teams to perform static analysis.
- Coverity[6]: Coverity is a software testing tool that provides static code analysis to help identify defects and vulnerabilities in code. Coverity provides a range of metrics that can be used to evaluate the quality and security of software code, including: Defect density, Code coverage, Code complexity, Security vulnerabilities, Code maintainability.
- Zap[8]: OWASP ZAP (short for "Zed Attack Proxy") is a popular open source web application security testing tool. It is designed to help developers and security professionals identify vulnerabilities and security flaws in web applications. OWASP ZAP provides a range of features for security testing: Automated scanning, Manual testing, Extensibility.
- Acunetix [1]: Acunetix is a powerful web vulnerability scanner that helps identify security vulnerabilities in web applications. Accunetix include a lot of features: Detect vulnerabilities, prevent SQL injection, fast analysis, easy integration with multiple tools such as Jenkins Jira or AWS.
- Checkov [41]: Checkov is an open-source static analysis tool for infrastructure as code that helps developers and DevOps teams identify security and compliance issues in their cloud infrastructure code.

2.2 Justification of the choice

I am going to use multiple technologies of the presented above. I could create a tool that automatically generates analysis but this would take me a lot of time and some of the technologies presented are already really good.

For static analysis of the different applications I am going to implement Sonarqube 9, I could use Coverity but nowadays Sonar seems to be better because it's a really easy tool to integrate with Jenkins. Moreover, lately, Sonarqube has been working on improving its vulnerability database so now it's capable of detecting many vulnerabilities that Coverity isn't. Furthermore, Sonar includes more information in its reports and automatically generates graphs that permit tracking the evolution of the different projects. Also, Sonarqube takes into account more rules when analyzing projects looking for

bugs or code smells. Additionally, the Sonarqube community is bigger than the Coverity community.

For dynamic analysis I am going to implement Zap, probably Acunetix could offer the best features and more power to analyze and find vulnerabilities but is a commercial tool, so it requires a license to use. Since the company doesn't want to invest in these tools I will have to use Zap. It also offers support for APIs and a range of plugins and add-ons, allowing users to customize the tool to suit their specific needs. Moreover, Zap might not be as powerful as Acunteix but it also provides automated testing, reporting and extensibility features.

Last but not least, I am going to use Checkov to analyze Kubernetes YAML and Dockerfiles, not only automating the process of analyzing but also creating a tool that will allow backends and the DevSecOps team to automatically generate standards yamls that contain the smallest possible number of vulnerabilities.

3 Outreach

In this section, I am going to be talking about the main objectives of this thesis. Also, I'm going to explain the possible risks and non-functional requirements.

3.1 Objectives

- Improve the security of Mango
 - Improve the quality of the developments in Mango.
 - Ensure that it's really hard for a hacker to access vulnerability information
 - Protect data from hackers.
- Automate a static analysis of the code of the projects
 - Automate a static analysis of all the projects to improve the quality of the developments.
 - Print the vulnerabilities in some kind of graph to facilitate programmers detection of vulnerabilities.
- Automate a dynamic analysis of the code of the projects
 - Automate a dynamic analysis of all the projects to improve the quality of the developments.
 - Print the vulnerabilities in some kind of graph to facilitate programmers detection of vulnerabilities.
- Securing the server's deploy
 - Create a tool that allows backends to create Kubernetes yamls
 - Generate standard Kubernetes yamls using Checkov to reduce the number of vulnerabilities.

3.2 Non-functional requirements

Non functional requirements that all implemented tools should accomplish:

- Usability: All tools should be easy to use and understand for software developers.
- Availability: Tools should always be available to use and it should be easy to access them.
- Scalability: It should permit an increment of the number of projects to analyze.
- Security: Tools should only permit access to the information of the project to software developers of the department Software Engineering.

3.3 Risks

Possible risks and problems that could appear during the realization of this thesis.

- Inexperience with the different tools and technologies: I am going to use a lot of tools that I have never used such as Jenkins, Sonarqube, Kubernetes etc. My inexperience with them could be a problem to plan how much time I am going to spend on each part of the thesis.
- Fixed entry Date: If there is a setback it is going to be hard to improvise and might cause that I can't finish the whole proposed project.
- Dependence on others: I might have a task that depends on another programmer.

4 Methodology

The development of the project is going to be done using the methodology Agile Kanban. I'm going to use this specific version of Agile because it is the methodology used by the team DevSecOps and also because it would be really hard for me to plan how much time a specific task is going to take, as a consequence it would be tricky to plan sprints.

4.1 Kanban

Kanban methodology is a lean manufacturing approach to managing workflow. The methodology focuses on visualizing work, limiting work in progress, and continuously improving the process. In Kanban methodology, kanban cards are used to signal the need for work to be done and to track the progress of work through a process. The kanban board is a visual representation of the workflow and is divided into columns that represent different stages of the process, such as "To Do", "In Progress", and "Done".

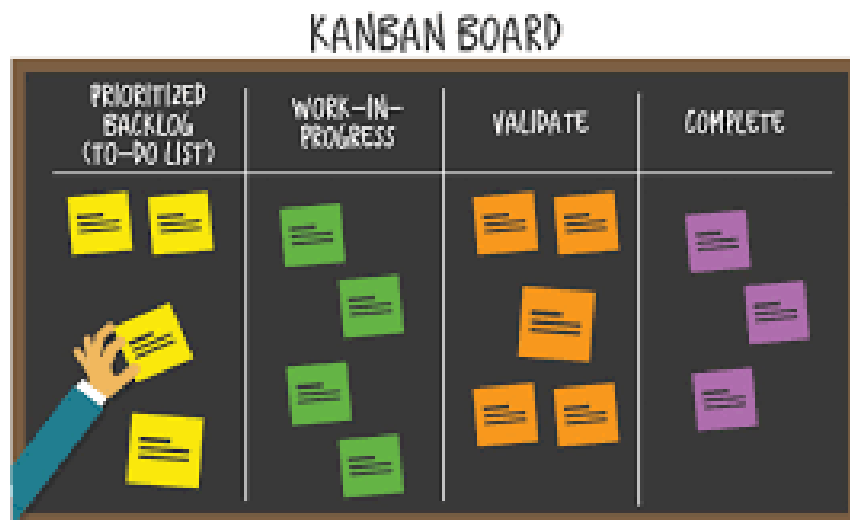


Figure 3: Kanban board [16]

Kanban methodology provides a flexible and adaptable approach to managing workflow that emphasizes visual management, limiting work in progress, and continuous improvement. It can be used in a variety of industries and applications, from software development to manufacturing and logistics.

4.2 Version tracking and control

To make possible the implementation of all the technologies and tools that automatically analyze code I am going to be working with:

- Git [9]: Is a popular version control system used in software development to manage changes to source code and other files. It allows multiple developers to work on the same project simultaneously, and provides a structured approach to tracking changes, collaborating, and merging code changes.
- Bitbucket: Is a web-based version control system that is primarily used for hosting and managing software development projects. One of the key benefits of Bitbucket is its integration with other development tools, such as Jira and Confluence
- Gitflow [30]: Is a popular branching model for software development using Git version control system. It provides a structured approach to managing code changes and releases, and is especially well-suited for larger development teams and complex projects.

4.3 Tool to manage Agile projects

- Jira [2]: Is a popular project management tool that is primarily used by software development teams to track and manage their work. Allows users to create and manage tasks, which can be organized into projects and workflows.

5 Time planning

The duration of my thesis going to be about 4 months approximately, from 14 February 2023 to 30 June 2023. I am going to be investing about 30 hours per week. This period contains 85 working days and 48 non-working days taking into consideration Saturdays, Sundays, Holy Week and other festivities.

From 19 February to 14 March the project management. During this time I am going to be, contextualizing my project, producing the time planning, defining my objectives making an analysis of risks and costs and doing the final sustainability report. Furthermore, I will be implementing necessary technologies to reach my objectives.

From 14 March to 20 June, I will be implementing, programming and configuring all the necessary technologies to reach my thesis goals. Also I am going to document all the work I do in this thesis memory.

From 20 June to 30 June i will have 10 days to finish my final document and prepare my thesis presentation. In total I will be investing in this thesis about 570 hours.

6 Project estimation

My project is going to be divided in 6 stages. The initial stage is the management of the project. After the initial stage, the following 4 stages are going to be about the implementation of the necessary technologies to achieve my thesis goal. The last one is to finish the documentation and prepare the thesis presentation.

To develop a software project it is very common to use Scrum, but since it would be really difficult to plan how much effort it is going to take to implement and develop each stage I am going to be using Kanban. Furthermore, the team were I am developing my thesis is working using this Agile methodology.

It is important to understand that I started my internship with Mango on 22-06-2022 as a software engineer backend. I will change my position to DevSecOps Engineer on 27-02-2023. Also, I will be working 30 hours per week in the company. Moreover, it is important to take into consideration that not all the hours that I am working in the company I am programming or implementing. There are dailys, plannings, etc.

As said above I can not exactly predict when I am going to finish each stage except the first and last one. But, I spoke with the senior DevSecOps Engineer of the team and made the following predictions:

Stage	Initial date	End Date
Project Management	20-02-2023	14-03-2023
Analyze and explain the Security of Mango	27-02-2023	10-03-2023
Automatic Static Analysis	13-03-2023	14-04-2023
Automatic Dynamic Analysis	17-04-2023	12-05-2023
Securing Kubernetes Deploys	15-05-2023	20-06-2023
Final documentation	21-06-2023	30-06-2023

Table 1. Different stages of the project and duration. Source: own compilation

6.1 Task definition

For all the tasks I will need at least, a Lenovo ThinkPad, a Lenovo screen, two mouses and two keyboards, one for the office and one for my own home and sony headphones.

It is important to take into consideration in the task description that when I am talking about the duration of the task it is the natural amount of hours that a task will take to be completed. That means that maybe there is an hour where I am reviewing my task with the senior but it is only one hour.

6.1.1 Project Management

- **PM1 - Context and outreach:** Explain context and outreach of the thesis

Duration: 20h
Dependencies: -
Human resources: Project manager
Material resources: Computer with internet, Overleaf, Grammarly

- **PM2 - Explain time planificaiton:** Definition of tasks and time planification
Duration: 20h
Dependencies: GP1
Human resources: Project manager
Material resources: Computer with internet, Overleaf, Grammarly, Ganttproject
- **PM3 - Economic and sustainability management:** Define the economic and sustainability plan
Duration: 20h
Dependencies: GP2
Human resources: Project manager
Material resources: Computer with internet, Overleaf, Grammarly
- **PM4 - Final document project management:** Review all the provided feedback from the previous stages and make the necessary changes to the final document.
Duration: 20h
Dependencies: GP1, GP2, GP3
Human resources: Project manager
Material resources: Computer with internet, Overleaf, Grammarly

6.1.2 Analyze and explain the Security of Mango

- **AS1 - Document the current security:** Explain the initial state of the Mango's security.
Duration: 30h
Dependencies: -
Human resources: security senior, DevSecOps Senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system
- **AS2 - Propose improvements to the security of Mango system:** Try to find possible solutions to the current problems in the security of Mango and propose them to the Security and DevSecOps team.

Duration: 30h

Dependencies: -

Human resources: security senior, DevSecOps senior, DevSecOps intern

Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system

6.1.3 Automatic Static Analysis

- **SO1 - Set up Sonarqube credentials in Jenkins:** Introduce in Jenkins the correct Sonarqube credentials to use it in the Jenkins library.
Duration: 4h
Dependencies: -
Human resources: DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Server with Sonarqube 9
- **SO2 - Create new methods in the Jenkins library:** Create methods in the Jenkins library that permit the execution of Sonarqube for each type of project in the corresponding pipeline.
Duration: 10h
Dependencies: -
Human resources: DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Server with Sonarqube 9, Visual studio code, Bitbukcet, Jira
- **SO3 - Import the SSL certificate in the necessary Docker:** Each pipeline is executed in a specific docker depending on the technology used to develop the project, it is compulsory to import the Sonarqube SSL certificate to execute from the specific docker the method implemented before.
Duration: 10h
Dependencies: -
Human resources: DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Server with Sonarqube 9, Visual studio code, Docker Desktop, Sonatype, Bitbukcet, Jira
- **SO4 - Test and make the necessary modifications:** Test that the corresponding methods work, and if they don't execute it local to figure out which is the error and make the necessary modifications.

Duration: 40h

Dependencies: -

Human resources: DevSecOps senior, DevSecOps intern

Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Server with Sonarqube 9, Visual studio code, Docker Desktop, Sonatype, Bitbukcet, Jira

- **SO5 - Investigate how Sonarqube quality gate works:** Investigate if it is possible to stop Jenkins pipelines with Sonarqube 9 and how to implement it.

Duration: 10h

Dependencies: -

Human resources: DevSecOps senior, DevSecOps intern

Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Server with Sonarqube 9, Bitbukcet, Jira

- **SO6 - Create a method in the Jenkins library to implement the quality gate :** Create a method to execute the quality gate and set the necessary parameters.

Duration: 10h

Dependencies: -

Human resources: DevSecOps senior, DevSecOps intern

Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Server with Sonarqube 9, Visual studio code, Docker Desktop, Sonatype, Bitbukcet, Jira

- **SO7 - Create a method in the Jenkins library to write messages in bitbucket PRs:** Implement a method that writes a comment in the Bitbucket PR, with the information of the quality gate(passed, failed, code coverage, vulnerabilities, etc)

Duration: 20h

Dependencies: -

Human resources: DevSecOps senior, DevSecOps intern

Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Server with Sonarqube 9, Visual studio code, Docker Desktop, Sonatype, Bitbukcet, Jira

- **SO8 - Implement an API to obtain security vulnerabilities from Sonar:** Create a Kubernetes cronjob that creates a pod with a script that collects from Sonarqube all the vulnerabilities information from all the projects and inserts that information in Kibana.

Duration: 20h

Dependencies: -

Human resources: DevSecOps senior, DevSecOps intern

Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Server with Sonarqube 9, IntelliJ, Bitbukcet, Jira

- **SO9 - Create confluence to allow execute Sonarqube9 locally :**

Create a confluence to allow developers to execute Sonarqube 9 locally.

Duration: 10h

Dependencies: -

Human resources: DevSecOps senior, DevSecOps intern

Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Confluence

6.1.4 Automatic Dynamic Analysis

- **ZAP1 - Test in local zap :** Test zap with the desktop version to discover what it can be done using this tool.

Duration: 10h

Dependencies: -

Human resources: DevSecOps senior, DevSecOps intern

Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Zap, Visual studio code, Docker Desktop, Sonatype, Bitbukcet, Jira

- **ZAP2 - Create a new dockerfile to run zap and a script to obtain the result :** Create a dockerfile and a script to be able to analyze all the webs of the department software engineering of Mango.

Duration: 40h

Dependencies: -

Human resources: DevSecOps senior, DevSecOps intern

Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Zap, Visual studio code, Docker Desktop, Sonatype, Bitbukcet, Jira

- **ZAP3 - Figure out how to get an actualized list of all the projects that must be analyzed :** Figure out how to get an actualized list of all the projects that must be analyzed.

Duration: 4h

Dependencies: -

Human resources: DevSecOps senior, DevSecOps intern

Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Zap, Visual studio

code, Bitbukcet, Jira

- **ZAP4 - Investigate how to analyze all the urls with zap and not only the introduced:** Discover how we can analyze all the URLs of a webpage instead of only analyzing the introduced.
Duration: 20h
Dependencies: -
Human resources: DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Zap, Visual studio code, Bitbukcet, Jira
- **ZAP5 - Investigate how to create contexts:** Discover how we can introduce in zap the credentials for webpages that need a log in.
Duration: 20h
Dependencies: -
Human resources: DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Zap, Visual studio code, Bitbukcet, Jira
- **ZAP6 - Create a Jenkins job that is executed every night :** Create a job in Jenkins that is executed every night and analyze all the projects of the department.
Duration: 10h
Dependencies: -
Human resources: DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Zap, Visual studio code, Bitbukcet, Jira
- **ZAP7 - Create a script to collect all the vulnerabilities information:** Implement a script that is capable of collecting all the vulnerabilities detected by the Zap, and introduce them in a Dashboard in Kibana.
Duration: 20h
Dependencies: -
Human resources: DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Zap, Visual studio code, Bitbukcet, Jira

6.1.5 Securing Kubernetes Deploys

- **SKD1 - Analyze vulnerabilities detected by checkov:** A member of the security team has already implemented checkov to analyze yamls, check which are the most common vulnerabilities.
Duration: 20h
Dependencies: -
Human resources: security senior, DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Zap, Visual studio code, Bitbukcet, Jira

- **SKD2 - Create template for deployment yaml:** Based on the previous analysis create a standard deployment yaml.
Duration: 30h
Dependencies: -
Human resources: DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Zap, Visual studio code, Bitbukcet, Jira

- **SKD3 - Think if it's necessary to create more templates:** Investigate if more templates can be created, such as configmaps, services or virutal services, and create them.
Duration: 30h
Dependencies: -
Human resources: DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Zap, Visual studio code, Bitbukcet, Jira

- **SKD4 - Generator of yaml's:** Create a program where you can introduce your variables and it generates a standard yaml.
Duration: 40h
Dependencies: -
Human resources: DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Zap, Visual studio code, Bitbukcet, Jira

6.1.6 Final documentation

- **FD1 - Final document and presentation:** Finish my final document and prepare my thesis presentation.

Duration: 30h

Dependencies: -

Human resources: DevSecOps senior, DevSecOps intern

Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Zap, Visual studio code, Bitbukcet, Jira

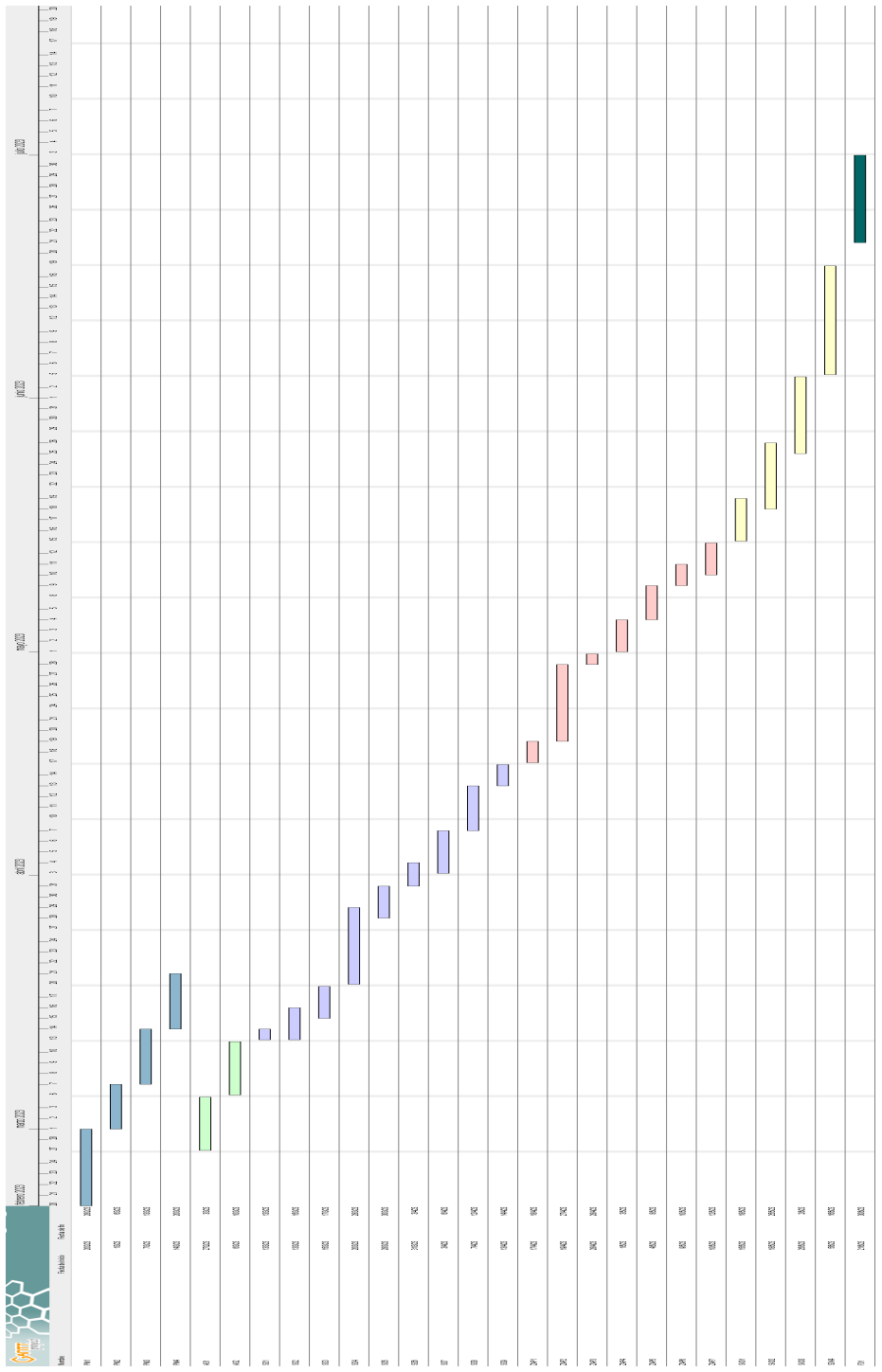
Task	Duration	Dependencies	Human Resources
PM1	20h	-	Project Manager
PM2	20h	PM1	Project Manager
PM3	20h	PM1, PM2	Project Manager
PM4	20h	PM1, PM2, PM3	Project Manager
AS1	30h	-	Security senior, DevSecOps Senior, DevSecOps intern
AS2	30h	-	Security senior, DevSecOps Senior, DevSecOps intern
SO1	4h	-	DevSecOps Senior, DevSecOps intern
SO2	10h	-	DevSecOps Senior, DevSecOps intern
SO3	10h	-	DevSecOps Senior, DevSecOps intern
SO4	40h	-	DevSecOps Senior, DevSecOps intern
SO5	10h	-	DevSecOps Senior, DevSecOps intern
SO6	10h	-	DevSecOps Senior, DevSecOps intern
SO7	20h	-	DevSecOps Senior, DevSecOps intern
SO8	20h	-	DevSecOps Senior, DevSecOps intern
SO9	10h	-	DevSecOps Senior, DevSecOps intern
ZAP1	10h	-	DevSecOps Senior, DevSecOps intern

ZAP2	40h	-	DevSecOps Senior, DevSecOps intern
ZAP3	4h	-	DevSecOps Senior, DevSecOps intern
ZAP4	20h	-	DevSecOps Senior, DevSecOps intern
ZAP5	20h	-	DevSecOps Senior, DevSecOps intern
ZAP6	10h	-	DevSecOps Senior, DevSecOps intern
ZAP7	20h	-	DevSecOps Senior, DevSecOps intern
SKD1	20h	-	DevSecOps Senior, DevSecOps intern
SKD2	30h	-	DevSecOps Senior, DevSecOps intern
SKD3	30h	-	DevSecOps Senior, DevSecOps intern
SKD4	40h	-	DevSecOps Senior, DevSecOps intern
FD1	30h	-	DevSecOps Senior, DevSecOps intern

Table 2. Table summarising tasks. Source: own compilation

6.2 Gantt chart

In the figure below we can see how tasks are spread out temporarily. As you can see in the Figure 4, the different stages are represented with different colors: blue for Project Management, green for Analyze and explain the Security of Mango, purple for Automatic Static Analysis, red for Automatic Dynamic Analysis, yellow for Securing Kubernetes Deploys, dark green for Final documentation.



7 Risk management

In the table 3 we can see the different risks and obstacles that we can find during the realization of the project. Also we can find an alternative plan.

Risk	Impact	Probability	Alternative plan
Inexperience with technologies	High	High	All the tasks that involve technologies that I have never used have been overestimated.
Fixed entry Date	Low	Medium	Since I am using Kanban if a task is not correctly scheduled because another took too much time it can be rescheduled without trouble.
Dependence on others	Low	High	Start another task that is not related to the current task.

Table 3. Risks and alternative plan. Source: own compilation

7.1 Tasks and risks

In this subsection I am going to be talking about which tasks can be affected by specific risks.

I am going to refer to the risk Inexperience with technologies as ITWT, to fixed entry date with FD and to Dependence on others as DO.

Task	Risk	Timeline delays	Alternative plan
PM1	-		-
PM2	-	-	-
PM3	-	-	-
PM4	-	-	-
AS1	DO	6h	If the security senior is occupied maybe it is going to take a while to obtain feedback from the analysis current system. I can start with another task if this happens.

AS2	DO	6h	If the security senior is occupied maybe it is going to take a while to obtain an answer of my proposals. I can start with another task if this happens.
SO1	-	-	-
SO2	ITWT	2h	My inexperience with Jenkins might be a problem. To solve the problem I can make pair programming with the senior.
SO3	ITWT	2h	My inexperience with Docker and SSL certificates might be a problem. To solve the problem I can make pair programming with the senior
SO4	ITWT, DO	8h	My inexperience with Jenkins and dockermight be a problem. To solve the problem I can make pair programming with the senior. The senior must ensure that what i have programmed is correctly implemented, so if he is occupied it might take a while to obtain a feedback. I can start with another task.
SO5	-	-	-
SO6	ITWT	2h	My inexperience with Jenkins and quality gates might be a problem. To solve the problem I can make pair programming with the senior
SO7	ITWT	4h	My inexperience with the API of Bitbucket and Jenkins might be a problem. To solve the problem I can make pair programming with the senior
SO8	-	-	-
SO9	FD	-	If something is blocking I can start with the next task.
ZAP1	-	-	-
ZAP2	ITWT	8h	My inexperience with Docker and Zap might be a problem. To solve the problem I can make pair programming with the senior
ZAP3	-	-	-
ZAP4	-	-	-

ZAP5	-	-	-
ZAP6	ITWT, DO	2h	My inexperience with Docker and Jenkins might be a problem. To solve the problem I can make pair programming with the senior. In this task the senior must ensure that all the previous steps and also this one is working properly, this might take a while so if the senior is not available i can start with another task.
ZAP7	FD	-	If something is blocking I can start with the next task.
SKD1	-	-	-
SKD2	ITWT	4h	My inexperience with Kubernetes might be a problem. To solve the problem I can make pair programming with the senior
SKD3	ITWT	6h	My inexperience with Kubernetes might be a problem. To solve the problem I can make pair programming with the senior
SKD4	DO	6h	In this task, the senior must ensure that all the previous steps have been performed correctly, so if the senior is occupied it is going to take i while. I can use this time to review the previous steps.
FD1	-	-	-

Table 4. Table explaining possible delays per task. Source: own compilation

8 Budget

8.1 Identification of costs

To correctly develop this thesis it is necessary some human and material resources. The material resources are the hardware and software that I am going to use to develop the project, and the human resources are the personal expenses. Furthermore, there are indirect resources, such as the price of the offices where I am going to develop the project. Also, there are unforeseen costs. To calculate all the costs I have rounded up all the results.

8.1.1 Human resources

Since i can not provide the exact salary of my coworkers I am going to be using GlassDoor[18] to estimate the salary of the poeple that is going to be part of the project. We can see all the information related to salary in the table 5.

Role	Salary/hour(net salary)	Salary/hour (gross salary)
Project Manager	20.85	27.105
DevSecOps senior	21.74	28.26
Security senior	22.4	29.12
DevSecOps intern	10	12.2

Table 5. Salaries per role. Source: own compilation

Taking into consideration the cost provided in table 5 we can calculate the cost of every task of the planification, due to each one having a different amount of hours assigned to each role. The computation can be seen in Table 6.

Task	Project Manager	DevSecOps senior	Security senior	DevSecOps intern	Total(euros)
PM1	542.1				542.1
PM2	542.1				542.1
PM3	542.1				542.1
PM4	542.1				542.1
AS1		141.3	145.6	366.4	653.3
AS2		141.3	145.6	366.4	653.3
SO1		28.26		48.8	77.06
SO2		28.26		122	150.26
SO3		28.26		122	150.26
SO4		141.3		488	629.3
SO5		28.26		122	150.26
SO6		28.26		122	150.26
SO7		56.52		244	300.52
SO8		56.52		244	300.52
SO9		28.26		122	150.26
ZAP1		28.26		122	150.26
ZAP2		141.3		488	629.3
ZAP3		28.26		48.8	77.06
ZAP4		84.78		244	328.78
ZAP5		84.78		244	328.78
ZAP6		28.26		122	150.26
ZAP7		84.78		244	328.78
SKD1		84.78		244	328.78
SKD2		113.04		366	479.04
SKD3		113.04		366	479.04
SKD4		141.3		488	629.3
FD1	542.1				542.1
TOTAL	2,710.5	1,640	291.2	5,344.4	9,986.1

Table 6. Amount of money human resources. Source: own compilation

8.1.2 Material resources

The project is going to last 5 months. I am going to be using some office material and some specific software. Also, to move to the office I am going to be using the bus. The amount of money required for the materials used in the project can be seen in detail in table 7.

Hardware materials have a finite life so in the case of this kind of materials I am going to calculate the amortization using the following formula:

$$\frac{Cost(euros) * duration\ project(months)}{Life\ span}$$

Material	Price(euros)	Life span	Amortization(euros)
Lenovo thinkpad	1000	4 years	104.16
Lenovo screen	600	4 years	62.5
Acer screen	350	4 years	36.45
Mouse home	40	4 years	4.16
Mouse office	40	4 years	4.16
Keyboard office	40	4 years	4.16
Keyboard home	80	4 years	8.32
Sony headphones	200	4 years	20.83
Total			647.24

Table 7. Amount of money materials. Source: own compilation

In addition to the materials listed in the table below, we have to add IntelliJ ultimate which have a cost of 72.5€ per month(362.5 euros per 5 months) and bus transport which has a cost of 1 euro per trip(40 euros 5 months).

Total: 647.24

8.1.3 Indirect resources

In the table 7 we can observe the cost related to the indirect resources. We have to take into consideration that the project is going to be developed in the offices of Mango in Palau Solita i Plegamans and at my home. In both places rent, electricity and wifi must be paid to allow de the development of the project. Also, when moving to the offices I am going to move using the bus.

Resource	Price(euros)	Months	Amortization(euros)
Rent office	60 per person	5	300
Rent apartment	350	5	1750
Wifi apartment	30	5	150
Wifi office	0.8 per person	5	4
Electricity office	14 per person [17]	5	70
Electricity apartment	40	5	200
Total			2,374

Table 8. Amount of money indirect resources. Source: own compilation

The rent of the office is a estimation based in the mean price of the offices in Palau Solita i Plegamans. Also, the electricity office is based on a study that indicates that most offices use 100kWh/m².

8.1.4 Contingency Costs

I will assign an extra 10% for contingency costs. In table 9 we can observe the computation for each type of cost.

Cost type	Cost(euros)	Contingency(%)	Final cost(euros)
Human resources	9,986.1	10%	10,984.71
Material resources	647.24	10%	711.7
Indirect resources	2,344	10%	2,578.4

Table 9. Amount of money indirect resources. Source: own compilation

8.1.5 Unforeseen costs

The unforeseen costs related to risks has been already taken into consideration, as said in the section risk management, by overestimating the amount of time that i will have to invest in each task.

The unforeseen costs related to material resources can be seen in table 10.

Material	Repair Price(euros)	Probability
Lenovo thinkpad	100	10%
Lenovo screen	50	10%
Acer screen	50	10%
Mouse home	10	10%
Mouse office	5	10%
Keyboard office	10	10%
Keyboard home	20	10%
Sony headphones	40	10%
Total	285	

Table 10. Amount of money materials. Source: own compilation

8.1.6 Final budget

To sum up, the final budget can be seen in table 11.

Type cost	Cost + contingencies
Human resources	10,984.71
Material resources	711.7
Indirect resources	2,578.4
Total	14,274.2
Unforeseen costs	285
Total with unforeseen costs	14,559.2

Table 11. Final budget. Source: own compilation

8.2 Management control

To have control over the deviation of the budget, at the moment that a task is finished I am going to recalculate the budget in function of the hours invested and the Unforeseen costs. I am going to use the following formulas:

- **Deviation of hours used per task**

$$(\text{Estimated hours} - \text{Real hours}) * \text{Estimated Cost}$$

- **Deviation of costs by the total hours used**

$$(\text{Estimated hours} - \text{Real hours}) * \text{Real cost}$$

- **Deviation of costs in human resources per task**

$$(\text{Estimated cost} - \text{Real cost}) * \text{Real Hours}$$

- **Deviation of materials costs**

$$\text{Estimated materials cost} - \text{Real materials cost}$$

- **Deviation of indirect costs**

$$\text{Estimated materials cost} - \text{Real materials cost}$$

- **Deviation of unforeseen costs**

$$\text{Estimated unforeseen cost} - \text{Real unforeseen cost}$$

- **Total deviation of hours**

$$\text{Estimated hours} - \text{Real hours}$$

- **Total deviation of costs**

$$\text{Total estimated costs} - \text{Total real costs}$$

9 Sustainability report

There are a lot of aspects to take into consideration when we talk about sustainability in the software engineering industry and the retail industry. The most common is environmental.

Another important aspect when talking about sustainability is the social aspect. In this ambit, we have to take into consideration the type of software we are using and the quality of the conditions of the people that work on the project. Even though it's hard to measure. The last aspect to take into consideration is the economic.

9.1 Economic dimensions

I have estimated the human resources and materials that i will need to develop this project. The cost estimate of this thesis includes only the necessary resources to correctly develop this project, as a consequence i can ensure that I am not going to be wasting resources.

This project will allow Mango developers to save a lot of time when developing new functionalities. That is because it is going to focus on automating the check of the quality and security of the multiple projects of Mango. As a consequence, programmers won't have to waste time anymore by manually checking if they are developing a secure code. Saving time is also saving money and resources, because software developers instead of investing time in checking the quality of the project, they can invest their time in developing better and new features.

9.2 Environmental dimensions

The only negative aspect that this project can have is the amount of electricity and resources, such as servers or computers, used to develop this thesis. Also, the servers will be running after the end of the thesis. But, for sure the automatic programs that I am going to use to analyze the projects are going to be faster than programers, so we will be saving a lot of energy that they would consume checking from their computers all the projects. Sadly, the only thing i can reuse is the software used to analyze the projects, it would be a waste of time and resources to remake the programs from zero.

9.3 Social dimensions

I think that throught this project I am going to learn a lot about how to develop software, and how to automate quality checks. Furthermore I am sure i will meet amazing people in Mango that will help me a lot to develop the project and will teach me a lot of useful staff. Also, I am sure i will share with these people good moments and will help me grow not only as a developer, but as person. This thesis is going to improve the quality of life of Software developers of Mango

since they won't have to check anymore manually the possible vulnerabilities and the quality of the code of the different projects.

10 Integration of knowledge

The main goal of this thesis is to integrate all the knowledge that I have learned during my Bachelor's Degree and apply them to the project. Furthermore, I want to point out that for me it has been also a great opportunity to incorporate many concepts that I have not learned during my Bachelor's or that might be related to other specializations that are not Software Engineering. Furthermore, learnt many concepts that probably are not taught in the University nowadays. In this section I am going to be explaining the subjects related to my project:

- **Web Applications and Services(ASW):** This subject helped me to clearly understand how to build a web project. Dividing the application into at least two parts, backend and frontend. Furthermore, provided information about how to create a great architecture for both parts and how to develop quality code. This subject helped me to understand how the different projects of the Mango department work.
- **Software Project Management(GPS):** This subject introduced the Agile methodology and explained the difference between using agile and using cascade. Furthermore, explained the different variations of Agile. This subject helped me to understand the different methodologies that the company uses.
- **Computer Networks(XC):** During this subject, I had the opportunity to learn the basic protocols in networking such as IP, TCP, and UDP. This was really helpful to understand some parts of the security of Mango as well as some parts of the CI/CD implementation.
- **Operating systems(SO):** During this subject I managed to learn basics in Main Functions of an Operating System, Internal Management of Services and Capabilities, Interaction with Interfaces and Components. It has been really helpful to work using different operating systems.
- **Business and Economic Environment(EEE):** This subject explained some basics of macroeconomics including how taxes work in Spain and some basic explanation about Business Economics. This subject was really helpful to make the economic part of this thesis.

11 Identification of laws and regulations

In this subsection I am going to explain the main laws and regulations that may affect in some way to my thesis and that I should take into consideration when implementing and documenting my thesis.

- **The General Data Protection Regulation (GDPR):** is a European Union (EU) regulation that sets the rules for the protection of personal data. It applies to organizations that process personal data of individuals within the EU, regardless of whether the organization is located within or outside the EU. The GDPR establishes fundamental principles, such as transparency, purpose limitation, data minimization, accuracy, storage limitation, and data security. It strengthens individuals' rights regarding their personal data, including the right to be informed, access, rectify, erase, data portability, object to processing, and not be subject to automated decision-making.
- **Intellectual Property Law:** it is a legal framework that regulates the rights and protections related to intellectual property. The LPI encompasses various forms of intellectual property, including copyright, neighboring rights, and industrial property rights. It defines the rights of creators and owners of intellectual works, such as literary, artistic, and scientific works, as well as software, databases, and trademarks. The law establishes the scope of these rights, including reproduction, distribution, public communication, and transformation of protected works. It also outlines the limitations and exceptions to these rights, such as fair use for educational or research purposes.
- **Organic Law on the Protection of Personal Data and Guarantee of Digital Rights:** it complements the GDPR and provides specific regulations and guidelines for the protection of personal data within the country. The LOPDGDD establishes the rights of individuals in relation to their personal data, including the right to access, rectify, erase, and object to the processing of their data. It also introduces provisions for the protection of digital rights, such as the right to digital privacy, freedom of expression, and the right to digital disconnection. The law sets out obligations for organizations handling personal data and establishes the Spanish Data Protection Agency as the supervisory authority.

12 Technologies and concepts

In this section, I am going to be explaining the technologies and concepts I will be using to reach my objectives and complete my tasks. It is really important to have a general understanding of the technologies explained in the following subsections to understand the thesis.

The technologies that I am going to explain in the following subsections are: Jenkins, Kubernetes, Sonarqube, Zap, Sonatype, Docker, Elasticsearch, Kibana, Grafana, NPM, Maven.

Concepts I will be explaining in the following subsections are: gitflow, gitops, version control, webhooks, observability, monitoring.

12.1 Jenkins

Jenkins[21] is an open-source automation server that is used to build, test, and deploy software applications. It is a popular tool used by software development teams to automate their build and deployment processes, and it is known for its flexibility, scalability, and ease of use.

Jenkins provides a web-based user interface that allows developers to create and manage pipelines, which are a series of steps that define how an application is built, tested, and deployed. These pipelines can be configured to automatically trigger builds based on changes to the source code repository, and they can be customized to include various types of testing, such as unit tests, integration tests, and end-to-end tests.

Jenkins supports a wide range of plugins that allow it to integrate with other tools and services, such as Git, Docker, Kubernetes, AWS, and many others. This makes it easy to create complex and sophisticated build and deployment workflows that are tailored to the needs of the organization.

In addition to its automation capabilities, Jenkins also provides extensive reporting and monitoring features, which allow developers to track the progress of builds and deployments, identify errors and issues, and quickly resolve them.

Overall, Jenkins is a powerful tool that helps software development teams streamline their build and deployment processes, reduce errors and downtime, and improve the overall quality of their applications.

12.2 Kubernetes

At a high level, Kubernetes [?] provides a platform for running containerized applications. It allows developers to deploy and manage containers across a distributed infrastructure, and provides a unified API for managing resources like pods, services, and volumes.

A Kubernetes cluster is a set of nodes (physical or virtual machines) that run containerized applications and are managed by a control plane. The control plane is responsible for managing the state of the cluster, including scheduling applications, scaling resources, and monitoring health.

- **Master nodes:** The master nodes are responsible for managing the state of the cluster. They run the control plane components, such as the API server, etcd, the scheduler, and the controller manager. The master nodes are usually highly available and are managed as a cluster themselves.
- **Worker nodes:** The worker nodes are responsible for running the containers that make up the application. They run the Kubernetes runtime, which manages the containers and ensures that they are healthy and available. It provides the computing resources, such as CPU and memory, needed to run the containers.
- **Networking:** Kubernetes provides a networking model that allows containers to communicate with each other and with the outside world.
- **Storage:** Kubernetes provides a variety of storage options, including local storage, network-attached storage (NAS), and cloud-based storage.
- **Control Plane:** The control plane is a set of components that manage the Kubernetes cluster. It includes components like the API server, etcd, the scheduler, and the controller manager.

Some of the key concepts and components of worker nodes:

- **Pods:** A pod is the smallest deployable unit in Kubernetes. It is a logical host for one or more containers, and it provides a shared network namespace and storage volumes.
- **Replication Controllers:** A replication controller ensures that a specified number of pod replicas are running at any given time. It can automatically scale up or down based on demand, and it can replace failed pods with new ones.
- **Services:** A service is an abstraction that provides a stable IP address and DNS name for a set of pods. It allows clients to access a group of pods with a single endpoint, and it provides load balancing and automatic failover. If we didn't have this the pod would have each time a different IP address and would be hard to manage connections with the pod and between pods.

- **Volumes:** A volume is a directory that is accessible to one or more containers in a pod. It can be used to store data that needs to persist across pod restarts, or to share data between containers.
- **ConfigMaps and Secrets:** ConfigMaps and Secrets are Kubernetes objects that can be used to store configuration data and sensitive information, respectively. They allow developers to manage application configuration and secrets separately from application code.

The API server is the central management point for Kubernetes, and it exposes a RESTful API that allows developers to manage resources like pods, services, and volumes. etcd is a distributed key-value store that is used to store cluster state information. The scheduler is responsible for assigning pods to nodes based on resource availability and constraints. The controller manager includes a set of controllers that watch for changes to cluster state and take actions to reconcile the state with the desired state.

To imagine the whole process, whenever we create a Deployment.yaml, it's assigned to a node, then to a replicaset and then that replicaset creates a pod with the specified objects and information.

12.3 Sonarqube

SonarQube[22] is a popular open-source tool for continuous code quality management. It provides a centralized platform for measuring and analyzing code quality across multiple languages and projects, and it integrates seamlessly with popular CI/CD tools like Jenkins and GitLab.

Key features of SonarQube:

- **Code Quality Analysis:** SonarQube provides a variety of code quality metrics, such as complexity, duplications, code coverage, and security vulnerabilities. It uses static code analysis techniques to identify issues in the code and provides actionable feedback to developers.
- **Integration with CI/CD Tools:** SonarQube can be integrated with popular CI/CD tools like Jenkins and GitLab, allowing developers to automate code quality analysis as part of their build and deployment processes.
- **Customizable Rules and Profiles:** SonarQube allows developers to define custom rules and quality profiles, tailoring the analysis to the specific needs of their project.
- **Dashboard and Reporting:** SonarQube provides a centralized dashboard and reporting interface, allowing developers and project managers to track code quality metrics across multiple projects and teams.
- **Language Support:** SonarQube supports a wide range of programming languages, including Java, C/C++, JavaScript, Python, and many more.

- Security Analysis: SonarQube provides a set of security rules to detect vulnerabilities like SQL injection, Cross-Site Scripting (XSS), and other types of security issues.
- Quality gates: Permits the definition of a standard that all the projects should pass. It allows stopping pipelines if the code analyzed doesn't accomplish the quality gate.

SonarQube is an essential tool for software development teams that value code quality and maintainability. By providing a centralized platform for code analysis and reporting, it helps teams identify and address issues in their codebase, leading to more maintainable and secure software.

12.4 Zap

ZAP[23], short for "Zed Attack Proxy," is an open-source web application security testing tool. It is designed to help developers and security professionals identify vulnerabilities and security issues in web applications during the development and testing stages. ZAP is developed and maintained by the Open Web Application Security Project (OWASP), a nonprofit organization focused on improving the security of software.

ZAP functions as a proxy server that sits between the user's web browser and the target application. It intercepts the communication between the browser and the application, allowing it to analyze and manipulate the traffic for security testing purposes. By acting as a "man-in-the-middle," ZAP can identify potential security vulnerabilities by examining requests and responses, performing various security tests, and providing detailed reports on the findings.

Zap gives the opportunity to automate the process of pentesting. It has a desktop version and a dockerfile version with a python file containing the basics analysis. Also it provides some key features like contexts.

In the context of ZAP (Zed Attack Proxy), a "context" refers to a specific configuration or environment within which security tests are performed. It allows you to define the scope and constraints for testing a particular web application or a set of related applications. Key things that you can define in Zap:

- Scope definition: By creating a context, you can specify the starting URL(s) for testing. ZAP will consider only the URLs within the defined context during the scanning and testing process. This allows you to isolate specific areas of a web application or focus on particular applications when testing a larger environment.
- Session management: ZAP maintains session information for each context, which helps it handle stateful applications. It manages cookies and session

tokens within the defined context, ensuring that requests and responses are correctly associated with the corresponding session.

Zap has many other features, but for this project we are going to use mainly the explained above.

12.5 Sonatype

Sonatype[24] is a company that specializes in providing software supply chain management solutions and tools. One of the primary offerings from Sonatype is the Nexus Platform, which is a comprehensive suite of tools designed to address the challenges associated with managing software artifacts, dependencies, and vulnerabilities. We are going to use mainly the nexus repository manager:

The primary function of the Nexus Repository Manager is to act as a centralized hub for storing and organizing software artifacts, making them easily accessible to development teams. Some key features about the Nexus Repository Manager:

- **Artifact Repository Management:** The Nexus Repository Manager supports multiple repository formats, including Maven, npm, Docker, and more. It provides a unified platform for hosting and managing software components in these formats. Developers can publish their own artifacts to the repository and retrieve dependencies required for their projects.
- **Deployment and Distribution:** The Nexus Repository Manager enables the distribution of artifacts to downstream environments or external users. It provides mechanisms for promoting artifacts through different stages of the software development lifecycle, facilitating seamless collaboration and deployment.
- **Integration with Build Tools:** The Repository Manager seamlessly integrates with popular build tools such as Apache Maven, Gradle, and others. This integration simplifies the build process by automatically resolving dependencies from the repository and ensuring consistent access to required components.

We are going to use that tool mainly to save and get docker images as well as store all the libraries created by the department, i.e. the design system (Atreyu) and the authentication library.

12.6 Elasticsearch

Elasticsearch is a highly scalable and distributed search and analytics engine. It is designed to handle large volumes of data and perform real-time search, analysis, and visualization of that data. Elasticsearch is part of the Elastic Stack, which also includes other components like Kibana for data visualization. The most important features in Elasticsearch are:

- **Distributed and Scalable:** Elasticsearch is designed to be distributed, meaning it can be spread across multiple nodes to handle large datasets and provide high availability and fault tolerance. It can scale horizontally by adding more nodes to the cluster, allowing for increased storage capacity and query throughput.
- **Near Real-Time Search:** Elasticsearch provides near real-time search capabilities, allowing you to index and search data with very low latency. This makes it suitable for applications that require fast and responsive searches.
- **Schemaless JSON Documents:** Elasticsearch stores and indexes data as JSON documents. It is schemaless, meaning you can index and search documents without explicitly defining a schema upfront. This flexibility allows for dynamic mapping and easy handling of evolving data structures.
- **Powerful Querying and Aggregation:** Elasticsearch provides a rich query language that allows you to perform complex searches, combining criteria like text matching, filtering, and aggregations. Aggregations enable data summarization and statistical analysis, allowing you to extract meaningful insights from your data.

Elasticsearch is widely used in various applications, including log analysis, monitoring, e-commerce search, content discovery, and enterprise search.

12.7 Kibana

Kibana [26] is an open-source data visualization and exploration platform that works alongside Elasticsearch. It is designed to provide a user-friendly interface for visualizing, analyzing, and interacting with data stored in Elasticsearch. Most important features for this project:

- **Data Visualization:** Kibana allows you to create rich visualizations of your data, such as line charts, bar charts, pie charts, maps, and more. You can customize the appearance and styling of the visualizations to suit your needs.
- **Dashboards:** Kibana enables you to build interactive dashboards by combining multiple visualizations into a single view. Dashboards provide a consolidated and holistic view of your data, allowing you to monitor key metrics and perform real-time analysis.
- **Integration with Elasticsearch:** Kibana seamlessly integrates with Elasticsearch, allowing you to leverage the power of Elasticsearch's search and analytics capabilities. You can directly access and visualize data stored in Elasticsearch indices without the need for complex data transformations or exports.

- **Aggregations:** Aggregations in Kibana are used to calculate and summarize metrics based on the data stored in Elasticsearch. They enable you to perform calculations, statistical operations, and other computations on your data. Aggregations operate on a set of documents and generate results that can be visualized or further analyzed.
- **Buckets:** Buckets in Kibana are containers that group documents based on specific criteria. They act as a way to partition data into subsets or categories for further analysis. Buckets can be created based on fields or certain conditions, and each bucket contains a subset of documents that satisfy the defined criteria.

By combining aggregations and buckets, you can gain a deeper understanding of your data, identify patterns, trends, and correlations, and generate meaningful insights for decision-making and analysis in Kibana.

12.8 Grafana

Grafana[27] is an open-source data visualization and monitoring tool used to create interactive and customizable dashboards. It allows users to connect to various data sources, collect and analyze data, and visualize it in real-time through a web-based interface. Grafana supports a wide range of data sources, including popular databases.

Most important features of Grafana for this project:

- **Data Visualization:** Grafana offers a rich set of visualization options, including graphs, charts, tables, and gauges. Users can create visually appealing dashboards by customizing the appearance, colors, and layout of the visual elements.
- **Data Source Integration:** Grafana can connect to numerous data sources, allowing users to fetch data from different systems and databases. It supports popular data sources like Graphite, Prometheus, Elasticsearch, InfluxDB, MySQL, PostgreSQL, and many more.
- **Alerting and Notifications:** Grafana allows users to set up alerts based on defined thresholds or conditions. When certain metrics breach the configured limits, Grafana can send out notifications via various channels, such as email, Slack, or other external services.

Grafana is widely used for monitoring and observability purposes, including infrastructure monitoring, application performance monitoring, cloud monitoring, network monitoring, and business intelligence.

12.9 NPM

NPM[28] stands for Node Package Manager. It is a package manager for JavaScript programming language and is the default package manager for

Node.js, a popular runtime environment for executing JavaScript code outside of a web browser.

NPM allows developers to discover, install, and manage reusable code packages, also known as "packages" or "modules," which contain JavaScript code and other related assets. These packages can range from small utility libraries to complete frameworks or applications.

Key aspects of NPM for this project:

- **Package Management:** NPM provides a command-line interface (CLI) that allows developers to interact with the NPM registry, a vast online repository of open-source JavaScript packages.
- **Dependency Management:** NPM helps manage dependencies between JavaScript packages. A package may require other packages to function correctly, and NPM automatically resolves and installs those dependencies.
- **Publishing and Sharing Packages:** NPM allows developers to publish their own packages to the NPM registry, making them available for others to discover and use. This fosters collaboration and code sharing within the JavaScript community.
- **Scopes and Organizations:** NPM supports scoped packages, allowing developers to group related packages under a specific scope. Scopes provide a way to namespace packages and differentiate them from others.

It empowers developers to leverage the vast array of open-source libraries available, saving time and effort in building robust JavaScript applications.

12.10 Maven

Maven[29] is a powerful build automation and project management tool primarily used for Java projects. It provides developers with a structured and standardized approach to building, packaging, and managing software projects. Maven simplifies the development process by automating repetitive tasks and managing project dependencies, making it widely adopted in the Java ecosystem. Important aspects of spring boot for the project:

- **Project Object Model (POM):** Maven uses a Project Object Model, which is an XML file called pom.xml, to define the project's configuration and dependencies. The POM specifies project metadata, such as the project's name, version, dependencies, build plugins, and more.
- **Dependency Management:** Maven handles project dependencies by automatically resolving and managing them. Developers specify the dependencies in the POM file, and Maven retrieves them from remote

repositories or local caches. It ensures that all required libraries and modules are available during the build process, simplifying dependency management.

- **Build Lifecycle:** Maven defines a standard build lifecycle consisting of phases and goals. The build lifecycle represents a sequence of steps that are executed in a predefined order. Examples of build lifecycle phases include compile, test, package, install, and deploy. Each phase can execute one or more goals, which are specific tasks such as compiling code, running tests, generating documentation, or creating a JAR file.

The most important for this project it's probably the Build Lifecycle since it simplifies a lot the scripting in the pipelines.

12.11 Gitflow

Gitflow[30] is a branching model and workflow that defines a specific structure and process for managing branches in a Git repository. It provides a clear set of guidelines and rules for branching, merging, and releasing code. When we use Gitflow we have at least 2 branches always alive:

- **Master Branch:** The master branch represents the mainline or stable codebase. It should always contain the production-ready code. Commits in the master branch typically correspond to releases.
- **Develop Branch:** serves as the integration branch for ongoing development. It acts as a base branch for creating feature branches and is used to collect and test new features.

In addition, we have supporting branches(feature branches, release branches, hotfix branches):

- **Feature Branches:** Feature branches are created from the develop branch and are used for implementing new features or changes. Each feature branch should be focused on a specific task or feature.
- **Release Branches:** Release branches are created from the develop branch when preparing for a new release. They provide a stable environment for final testing, bug fixes, and release-related activities.
- **Hotfix Branches:** Hotfix branches are used to address critical issues or bugs in the production code. They are created from the master branch, allowing for immediate fixes to be applied.
- **Bugfix Branches:** This branches are used to fix bugs detected in the release branches.

Imagine that we want to implement a new functionality and we want to use git-flow. The first thing we have to do is create a new feature branch where we

are going to develop that new functionality. After developing and testing we are going to open a PR from our branch to develop. Once our code is in develop we are going to create a Release Branch to make the final tests. If we detect a bug or error during the test in the release branch we can create a bugfix branch to solve it and merge it to our release branch. After all the test is done we have to open a pr from our release branch to master. Once the code is merged if a critical error is detected we can create a hotfix to quickly resolve it. After the merge between the hotfix and the branch master is done, we have to merge master to develop to avoid conflicts.

12.12 Gitops

GitOps[31] is a software development methodology and operational model that leverages Git as the single source of truth for both application code and infrastructure configuration. It brings the principles of version control and collaboration to the management of infrastructure and deployments. Furthermore, GitOps promotes the use of infrastructure-as-code principles, enabling infrastructure and configurations to be treated as code and version-controlled. Key concepts of Gitops:

- **Git as the Single Source of Truth:** In a GitOps workflow, the desired state of the entire system, including both infrastructure and application code, is stored in a Git repository. This repository acts as the single source of truth for the entire system's configuration.
- **Declarative Configuration:** The desired state of the system is defined declaratively in files stored in the Git repository. These files can include infrastructure-as-code templates (e.g., Kubernetes manifests, Terraform scripts) and application configuration files.
- **Continuous Deployment and Synchronization:** GitOps relies on continuous synchronization between the desired state defined in Git and the actual state of the system. This is typically achieved using an automated reconciliation process or an orchestration tool that continuously monitors the Git repository for changes and applies them to the target environment.
- **Infrastructure Provisioning and Deployment:** The infrastructure and application deployments are managed by an orchestration tool (e.g., Argo CD, FluxCD) that reads the desired state from the Git repository and applies it to the target environment. This includes provisioning infrastructure resources, deploying applications, and configuring any necessary components.

GitOps has gained popularity as a reliable and efficient approach for managing infrastructure and deployments, especially in cloud-native and containerized environments. By centralizing configuration and leveraging Git's version control capabilities, teams can achieve greater control, visibility, and collaboration in their software delivery processes.

12.13 Version Control

Version control[32], also known as revision control or source control, is a system that tracks and manages changes to files and code over time. It allows multiple developers to work collaboratively on a project, keeping track of modifications, and enables efficient management of different versions or revisions of the project's files. Version control systems provide a history of changes, facilitate collaboration, enable branching and merging of code, and help ensure the integrity and traceability of software development projects.

Key concepts of version Control:

- **Repository:** A version control system organizes files and code in a repository. The repository stores the complete history of changes made to the project, including additions, modifications, and deletions.
- **Branches:** Branching allows for the creation of independent lines of development. Branches are separate copies of the project's codebase that can be worked on simultaneously. They enable developers to work on different features or bug fixes independently without affecting the main codebase.
- **Merging:** Merging combines changes from one branch into another, typically integrating the changes made in a feature branch back into the main branch. Merging ensures that the latest changes from different branches are incorporated into the main codebase.
- **Conflict Resolution:** When changes made in different branches conflict with each other, version control systems provide tools to help resolve conflicts. Developers can compare conflicting versions, manually edit and merge code, and choose the desired outcome.

For this project we are going to use Git.

12.14 Webhooks

Webhooks[33] are a mechanism used for real-time communication between web applications. They allow one application to send automatic notifications or trigger actions in another application by sending HTTP requests to a predefined URL (callback URL) when a specific event occurs.

First of all an event has to occur in the source application, it can be from a user action or data update, that the source application wants to notify to other applications. The target application provides a URL where it wants to receive notifications from the source application. When the specified event occurs in the source application, it triggers the generation of a webhook payload. This payload contains relevant data about the event. The source application sends an HTTP POST request to the webhook URL of the target application, carrying the webhook payload as the request body. The target application receives the

incoming HTTP request at the specified webhook URL. It extracts and processes the payload, performing the necessary actions based on the received information.

12.15 Observability

The primary objective of observability[34] is to gain a holistic understanding of a system's behavior, performance, and dependencies. It aims to provide insights into the system's internal workings, identify root causes of issues, and enable effective troubleshooting and performance optimization.

Observability has a broader scope and aims to understand and analyze the system's internal state, behavior, and performance by collecting and analyzing various signals, including logs, metrics, and traces. It encompasses the ability to gain insights into the system's behavior, even for unknown or unexpected issues.

It focuses on capturing all relevant data and signals, allowing for deeper analysis and understanding of the system's behavior. Observability aims to answer questions and investigate issues that might not have been anticipated in advance.

12.16 Monitoring

The primary objective of monitoring is to ensure that a system is functioning within expected boundaries and to identify any deviations from normal behavior. Monitoring is often used for early detection of issues, triggering alerts, and ensuring predefined service level agreements (SLAs) are met. Monitoring typically follows a predefined set of metrics or thresholds that indicate the system's health or performance. It involves setting up monitoring tools, defining alert conditions, and collecting specific data points regularly. Monitoring focuses on collecting and analyzing predefined metrics and indicators that provide specific information about the health, performance, and availability of a system. It typically involves monitoring specific components, resources, or metrics of interest.

13 Analyze and explain the security of Mango

In this section I am going to explain the current state of the security of mango related to DevOps since my project is focusing on DevSecOps: S-SDLC. First of all, I am going to analyze all the current systems of DevOps and then explain how is currently considered security. After this explanation, I am going to point out the possible improvements to the current system.

13.1 Document the current security

In this subsection you are going to find a explanation of the current state of security in Mango related to DevOps. The next subsection I am going to point out the possible improvements.

13.1.1 CI/CD Implementation

To implement the CI/CD system in the software department we are using, Jenkins, Kubernetes, Sonatype, Docker and ArgoCD. Jenkins is mostly used to implement pipelines to automate multiple processes. We have multiple pipelines depending on the type of project. Furthermore, we are using a multi-branch system so depending on the version of the code we execute different steps for the same project.

A Jenkins pipeline is a set of stages that are executed one after another, each stage is responsible for one specific task i.e. (building the project, executing tests, etc). Jenkins pipelines can be implemented in many different ways, a really common one is having a Jenkinsfile in the project files that contains all the stages of the pipeline. But, that's a very tough way to work, since when you have to update a part of the file of all the backends projects you have to modify multiple files. To avoid that problem, the DevOps team decided to create a library that contains the specific stages for all the backends and all the frontends. So now a Jenkinsfile looks like figure 5 instead of figure 6.

As can be seen in figure 6, we are specifying the library from where we are going to get the groovy script that has to be executed for that specific project in this case is **mango-jenkins-shared-library**. After specifying the library we specify the pipeline that we want to execute, in this case **k8sNodePipeline**. Furthermore, the necessary variables for the correct execution of the pipeline are set:

- **dockerImageName:** Name that we are going to assign to the docker image.
- **namespace:** Namespace in Kubernetes where the application is running.
- **deploymentName:** Name of the deployment in Kubernetes.
- **containerName:** Name of the container.

```
@Library('mango-jenkins-shared-library')_
k8sNodePipeline {
    dockerImageName =
    namespace =
    deploymentName =
    containerName =
}
```

Figure 5: Jenkinsfile of a specific project.

```
Jenkins Declarative Pipeline:
Jenkinsfile (Declarative Pipeline)
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                echo 'Building..'
            }
        }
        stage('Test') {
            steps {
                echo 'Testing..'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying....'
            }
        }
    }
}
```

Figure 6: Common Jenkinsfile.

For frontend projects that use Node and Angular to develop we have a specific pipeline that perform the following stages, in the specific case of a pr:

- **Checkout:** In this stage the code is downloaded from a repository in bitbucket and the version is controlled.
- **Install Dependencies:** During this stage the necessary dependencies for the build of the project are installed. Mainly using npm install, also Jenkins has access to the Mango's Sonatype to download a package from there if necessary.
- **Unit test quality:** During this stage the tests of the code are executed.
- **Scan vulnerabilities:** In this stage the vulnerabilities of the code and dependencies are detected using a specific tool. Will go in deep when talking about how Security is integrated in all the Devops process.
- **Build:** During this stage the application is built.
- **Sonarqube:** During this stage the code is statically analyzed using Sonarqube 6. No quality gate implemented.
- **Post Actions:** Nothing.

Specific stages develop, release, master (apart from the specified below it also executes stages previously described: checkout, version control, install dependencies and build.):

- **Build Docker Image:** During this stage the docker image is built.
- **Publish Docker Image:** During this stage the Docker image built in the previous state is published to a repository in Sonatype.
- **Update GitOps deployment:** During this stage the repository of Gitops of the project is updated.
- **Deploy:** During this stage the kubernetes context is set, due to Mango has different Clusters for different applications and the image of the application on the deployment is changed using kubectl.

Aside of the pipeline that we use to build the front and back projects, we have other pipelines related to devops projects. For example, many of the tools and packages are developed using python so we have the following pipeline:

- **Checkout:** In this stage the code is downloaded from a repository in bitbucket and the version is controlled.
- **Install dependencies:** During this stage the necessary dependencies for the build of the project are installed. Mainly using npm install, also Jenkins has access to the Mango's Sonatype to download a package from there if necessary

Branch master

Full project name: OneAppGood/oneapp-front/master



Recent Changes

Stage View

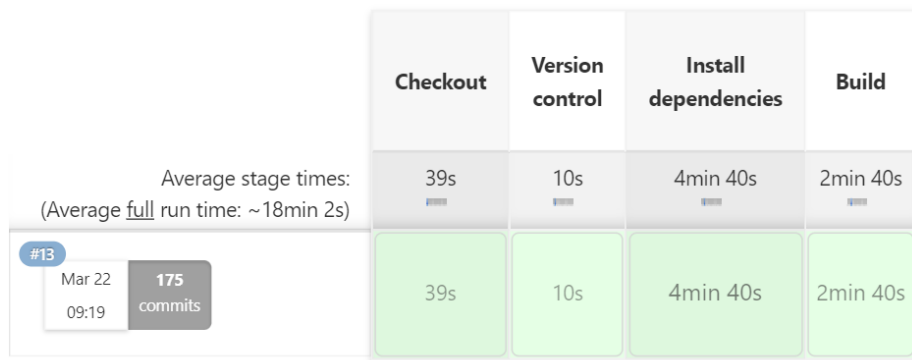


Figure 7: Frontend and backend pipeline CI CD

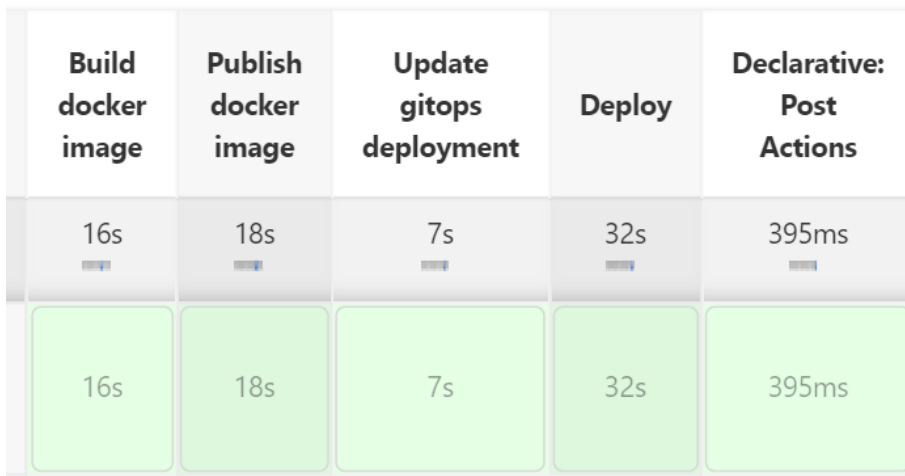


Figure 8: Frontend and backend pipeline CI CD

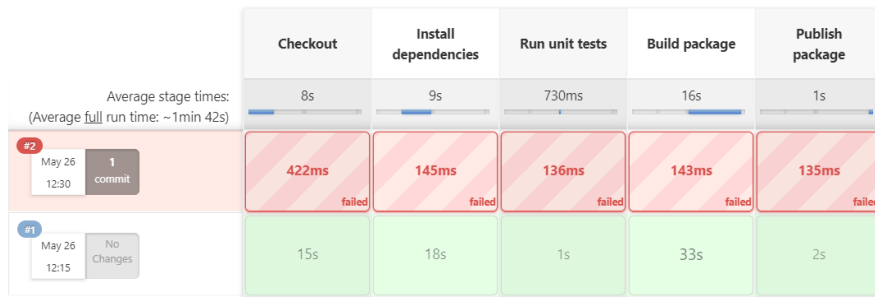


Figure 9: Frontend and backend pipeline CI CD

- **Unit tests:** During this stage the tests of the code are executed.
- **Build package:** During this stage the python package is build and prepared to be published in a private repository in Sonatype.
- **Publish package:** During this stage the package built previously is published to Sonatype.

There is another pipeline implemented related to devops projects, the one that is used to manage the repositories with all the kubernetes objects. There is a repository of gitops containing all the Kubernetes objects per project. The pipeline has the following stages:

- **Checkout:** Get the code from the gitops repository.
- **Check linter:** Check if the .yaml are in the correct format, if not the pipeline stops.
- **Security check:** Check and generate a report of the vulnerabilities detected by a certain tool (checkov). I am going to dig in this tool later.
- **Check image on sonatype:** Check if the docker image contained in the deployment exist.
- **Sync appset:** This stage is synchronizing argoCD. That's like a Kubectl apply of all the files contained.

We have to take into consideration how Jenkins is executing the different projects. There are many different ways of managing the slaves in Jenkins. A slave is basically a virtual machine or pod that is controlled by Jenkins to execute the different pipelines. One way to set slaves is having multiple virtual machines that are only used to execute pipelines but in Mango, we are using a different way, that is because executing pipelines in virtual machines has many side effects as taking too much time to execute the different pipelines and

```
Raw YAML for the Pod ?
apiVersion: v1
kind: Pod
metadata:
  name: k8s-node
spec:
  containers:
    - name: jnlp
      image: jenkins/inbound-agent
    - name: node
      image: [REDACTED]/k8s-node
      command: ['sleep', '999999']
      resources:
        requests:
          cpu: 2000m
          memory: 256Mi
    - name: tools
      image: [REDACTED]/k8s-tools
      command: ['sleep', '999999']
      resources:
        requests:
          cpu: 50m
          memory: 256Mi
  env:
    - name: DOCKER_HOST
      value: tcp://localhost:2375
    - name: dind-daemon
      image: [REDACTED]/dind-options
      resources:
        requests:
          cpu: 20m
          memory: 512Mi
  env:
```

Figure 10: Node pod template

harder scalability.

In Mango, we are using a plugin from Jenkins that allow us to execute the different pipelines in Kubernetes. We have a specific cluster in Kubernetes that is only used to execute tools related to DevOps and Jenkins pipelines. So that plugin allows us to specify a deployment.yaml from where create new pods depending on the type of pipeline.

In figure 10, we can observe an example of a pod template, were we get multiple docker images of the repository in Sonatype. Each image has been specifically created to run a type of pipeline, so we can have a better control of the vulnerabilities and dependencies of each docker image.

13.1.2 Security implementation

During the complete execution of the pipelines the security is implemented in many ways. First of all, all the passwords used by the different applications are in sealed-secrets in kubernetes.

A Sealed Secret in Kubernetes is an extension to the concept of a regular Secret that provides an additional layer of encryption and security. It is a Kubernetes custom resource that allows you to encrypt and store sensitive

data in a way that only authorized parties can decrypt and access.

The Sealed Secrets project is an open-source tool created by Bitnami that implements this concept. It provides a controller and a command-line tool to create Sealed Secrets and a controller that runs within your Kubernetes cluster to decrypt and create regular Secrets from the sealed data. To decrypt the information locally you need to have the key.

Furthermore, all the passwords of the different systems, wild cards, and public certificates are stored in a vault of secrets. So, if a programmer need to access Jenkins or Sonarqube can get the credentials from that vault, instead of having that information in their PC.

The vulnerabilities of the OS are taken into consideration, since all the backends are running using the same base docker image, as well as frontend images. All the vulnerabilities of this two images are analyzed and taken into consideration. Also, it is really easy to control it since we only have to control 2 different images. In these days the total number of vulnerabilities is 0. Furthermore, the vulnerabilities of the image built from the dockerfile of each project are controlled using the same tool and reported to programmers to allow them fix this specific vulnerabilities. The tool used is Grafeas Image Scanner.

Grafeas Image Scanner (Grafeas + trivy) or simply "Grype"[35] is an open-source vulnerability scanner specifically designed for container images. It is developed by Anchore, Inc. Grype helps to identify security vulnerabilities within container images by analyzing their software components and their associated metadata.

Here are some key features and functionalities of Grype:

- **Container Image Scanning:** Grype scans container images, such as Docker images or images used in Kubernetes deployments, to detect known vulnerabilities in the software packages and dependencies within those images.
- **Vulnerability Database:** Grype uses a comprehensive vulnerability database, which is regularly updated, to compare the versions of software packages in the scanned container images against known vulnerabilities. It provides accurate and up-to-date information about the security issues.
- **Multiple Image Formats:** Grype supports scanning different container image formats, including Docker images, OCI (Open Container Initiative) images, and ACI (App Container Image) images.
- **Easy Integration:** Grype can be easily integrated into container security workflows, CI/CD pipelines, or vulnerability management systems. It

provides a command-line interface (CLI) and a JSON-based output format that can be consumed by other tools or scripts.

In mango, we are also checking all the possible vulnerabilities generated by the dependencies of the projects. For this specific task, we are also using Gype. The analysis is performed in every pull request that is created in back and front projects. The vulnerabilities are reported to the developer by attaching a message in the pr in Bitbucket.

13.1.3 ArgoCD

To implement the gitops concept in the software department we are using ArgoCD[36].

Argo CD is an open-source continuous delivery (CD) tool specifically designed for Kubernetes-based applications. It enables automated deployment, configuration management, and synchronization of applications in Kubernetes clusters.

To implement it there are three pods in the Jenkins Cluster that are capable of applying, managing, and comparing all the objects of the Kubernetes cluster. That is a huge benefit since we don't need to add new pods in all the Clusters.

When the team started to implement ArgoCD, they decided to create a new repository per every single project, and in every repository, there are three folders at least: master, int, and dev. In every folder, there are the same Kubernetes objects when no one is making changes. It can be implemented in many other ways but that is the one they chose.

It is important to understand that ArgoCD can compare the state of the repository and the state of the Kubernetes objects and is going to consider the truth whatever is in the repository.

ArgoCD has many functionalities but the most important are:

- **Prune:** ArgoCD by default will never erase an object even if you have erased in the repository. Only when you active this functionality will erase it.
- **Dry Run:** This functionality allows to verify that the changes successfully work without applying them.
- **Refresh:** This functionality fetch the changes made in the git repository.
- **Sync:** Apply whatever is in the git repository.

To make it clear how ArgoCD works, let's explain how someone would make a change in a deployment.yaml using ArgoCD.

- **Pull Repository:** The developer needs to pull the repository where the deployment.yaml is.
- **New Branch:** The developer must create a new branch to push the changes made.
- **Make a change:** The developer would make the change first in the dev folder. After checking that everything is working he can add the modifications in int and master.
- **Push change:** Push a change to the repository and create a pull request to master and add as a reviewer someone from DevOps. Once the pr is approved he can merge the code to the master.
- **Refresh and Sync:** Once the code is in master he must go to ArgoCD and click the refresh and sync buttons.

That is a big improvement compared to how the workflow used to be. Everyone used to have all the Kubernetes context with admin access. That means that anyone could make a change in the objects. Furthermore, if someone did a change in the objects there was no way to go back, and you couldn't know who was doing the change. Now you can easily go back to the previous state, you can control who applies the objects and you can limit what certain teams can apply.

To sum up, ArgoCD is not only an improvement in Kubernetes management, but it is also an improvement in Security.

13.1.4 Monitoring

To monitor the resources in Kubernetes Mango is using mainly three technologies, Prometheus[37], Grafana and APM[38].

Prometheus is an open-source monitoring and alerting system designed for gathering and analyzing metrics from applications, systems, and services. It was created by SoundCloud and is now maintained by the Cloud Native Computing Foundation (CNCF).

Prometheus collects metrics from various sources, such as application code instrumentation, exporters, and service discovery mechanisms. It supports multiple data formats, including a simple plain text format and a more efficient binary format for high-volume data.

Prometheus provides a powerful query language called PromQL (Prometheus Query Language) to retrieve and analyze metrics data. It allows you to perform complex queries, aggregations, and transformations on the collected data. Additionally, Prometheus supports defining alerting rules to trigger notifications based on specific conditions or thresholds.

Prometheus provides built-in service discovery mechanisms to automatically discover and monitor targets, such as Kubernetes services, Consul services, or static configurations. This simplifies the process of adding and removing monitoring targets dynamically.

Prometheus follows a flexible data model based on time-series data. It stores metrics as time-stamped series of numerical values, along with labels that allow for multi-dimensional querying and filtering. This enables efficient storage and retrieval of metrics data.

In every node, some pods collect all the information from the node and insert it into Prometheus. Once the data is in Prometheus it can be used to create Dashboards in Grafana to monitor the amount of memory or CPU used by every pod.

APM stands for Application Performance Monitoring. It is a software practice and a set of tools designed to monitor and manage the performance and behavior of applications in production environments. APM provides insights into application performance, resource usage, and end-user experience to identify issues, optimize performance, and ensure efficient operation.

The tools provided by APM monitor and capture errors and exceptions that occur within the application. They collect stack traces, error messages, and contextual information to help identify and prioritize issues for resolution. Error tracking enables faster troubleshooting and proactive issue resolution.

It helps development and operations teams gain insights into the behavior and performance of their applications in production environments. It facilitates proactive monitoring, troubleshooting, and optimization of applications, ensuring that they meet performance expectations and provide a positive user experience.

The tools provided by APM collect and analyze various performance metrics and indicators from applications in real-time. This includes metrics like response time, throughput, error rates, CPU and memory usage, database queries, and more. By monitoring these metrics, APM helps identify bottlenecks, performance degradation, and resource issues.

13.2 Explain possible improvements in the Security of Mango

To explain the possible improvements in the security of Mango first we have to see what is the objective. For Mango, the objective is to completely work in S-SLCD.

The Secure Software Development Life Cycle (SSDLC), also known as Secure SDLC or Secure Development Life Cycle, is an approach to software development that emphasizes incorporating security considerations throughout the entire software development process. It aims to proactively identify and address security vulnerabilities and risks at each stage of the development life cycle.

The SSDLC typically consists of the following key phases:

- **Security Policy:** This step consist into the creation of a security Policy that all workers have to accomplish. Already created.
- **Application Secret Security:** Save passwords in a save mode. In the case of Mango, we are using a vault were we save all the passwords and vulnerable information.
- **Scan OS Vulnerabilities:** This step consist on analyzing the vulnerabilities of the operating systems that are used by the applications and programmers.
- **Scan Lib Dependency Vulnerabilities:**
- **IDE Security Tools:** Install and use a plugin in the IDE, related to security. This plugin will help developers to detect possible vulnerabilities.
- **SAST Analysis:** Static Application Security Testing. Introduce a tool in the environment of the developer to perform static analysis.
- **DAST Analysis:** Introduce a tool in the environment of the developer to perform dynamic analysis.
- **Security Dashboard:** Create a Security Dashboard to take control of the current situation and set goals and objectives.

Now we have set an objective we can see what we can do to improve the security of Mango. There are many steps that has been already acomplished as we can observe in the figure above. But there are 3 steps that remains to do, SAST Analysis, DAST Analysis and Security Dashboard. During my thesis I am going to implement this three last steps that will complete the Secure Software Development Life Cycle.

The image shows a screenshot of the Jenkins 'Add Credentials' form. At the top, there is a 'Kind' dropdown menu with 'Username with password' selected. Below this is a 'Scope' dropdown menu with 'Global (Jenkins, nodes, items, all child items, etc)' selected. The form contains several input fields: 'Username', 'Password', and 'ID', each with a help icon. There is also a checkbox labeled 'Treat username as secret' with a help icon.

Figure 11: Jenkins credentials

14 Automatic Static Analysis

During this section I am going to explain how I have implemented Sonarqube in the current system of Mango and how this implementation it's improving the overall code.

14.1 Set up Sonarqube Credentials in Jenkins

First thing I had to do is get familiarized in how Jenkins treat the credentials, since to access the Sonarqube server of Mango I will need to use specific credentials. To add new credentials we have to follow this steps:

- In the management section of Jenkins access the credentials subsection.
- Once you are in the credentials page you have to click add credentials.
- Now you can chose which type of credentials you want to add such as Username and Password, SSH Username with private key, Certificate, Secret text, etc.
- Once you have created the credentials in the management you can access them from any jenkinsfile executed in that server using: `withCredentials`, and specifying the id generated by Jenkins in that function.

14.2 Create new methods in the Jenkins library

This subtask mainly consists in create the mango-shared-jenkins-library, the library that contains all the pipelines and methods used by the pipelines, a

```

stage('Sonarqube'){
    when {
        branch 'PR-*'
        not {
            expression { env.CHANGE_BRANCH.startsWith('hotfix/') }
        }
    }
    steps {
        script {
            branch_pipe = "PR"
            container(PodContainer.maven) {
                Quality.updateMavenSonar this , pipelineParams.name
            }
        }
    }
}

```

Figure 12: Jenkins stage

new method that allows us to execute Sonarqube 9 no matter which project or technology is used.

To perform that specific task the solution was to create in each Jenkins pipeline a new stage. As can be seen in Figure 12, the stage is only executed during PR and never executed when the pr it's a hotfix. We decided to not execute the Sonarqube in a hotfix because we are going to implement the quality gate, and whenever you use a hotfix it's because you need to urgently fix a bug in production. Sonarqube script is going to be executed in a specific docker container from the pod where the pipeline is running. In this case, it is going to run in the maven container.

It is also important to take into consideration that the Sonarqube stage is executed after the build and test stages because Sonarqube needs a specific report generated by the test execution to run the analysis.

Inside the method `updateMavenSonar`, we are going to execute the following command:

```
./mvnw sonar:sonar -Dsonar.login=script.password -Dsonar.organization =
default -Dsonar.projectKey =gitRepository -Dsonar.qualitygate.wait=true
```

14.3 Import the SSL certificate in the necessary Docker

To be able to execute the sonar command in each pipeline we need to use the Sonarqube public SSL certificate since the Sonarqube server is using SSL. As we have seen in the previous task, the Sonarqube command is executed in a specific docker container depending on the technology we are using. Since every pipeline is going to be executed in a different image we have to import the certificate to multiple docker images. Furthermore, we have to take into consideration that every docker image used to execute pipelines has its own repository in bitbucket with the necessary files and its own Dockerfile.

To install an SSL certificate in Java we have to:

- **Obtain the SSL certificate:** In my case, this step consist into getting from the vault of secrets the correct certificate
- **Prepare the certificate file:** Convert it to PEM.
- **Install Certificate:** Use the keytool provided by java to install the certificate in the cacerts.

The task consisted on adding in each repository the corresponding docker image and adding a command in the docker file to copy the certificate as well as a command to insert the certificate in the cacerts.

14.4 Investigate how Sonarqube quality gate works

During this task I have been investigating what is a quality gate in Sonarqube and how to communicate to Jenkins the status of the quality gate.

In SonarQube, a quality gate is a set of predefined criteria or conditions that your project's code must meet to be considered of acceptable quality. It acts as a gatekeeper that checks the quality of your code and determines whether it can be considered "good" or "bad" based on the specified criteria.

When you analyze your code with SonarQube, it evaluates the code against the defined quality gate conditions. The analysis results are then compared to the thresholds set in the quality gate. If the analysis meets all the defined conditions, the quality gate status is "passed." If any of the conditions are not met, the quality gate status is "failed."

Quality gates in SonarQube are defined using a combination of static code analysis rules, metrics, and thresholds. They help enforce coding standards, promote best practices, and ensure that your code meets certain quality standards.

The communication of the quality gate to Jenkins is done using a webhook. A webhook is a mechanism that allows two applications or systems to communicate with each other in a real-time or near-real-time manner. It enables one application to send a notification or trigger an action in another application whenever a specific event or data change occurs. While I was doing this task realized that Sonarqube was not able to reach Jenkins since both servers where in two different firewall zones.

Furthermore, during the realization of this task I communicated to the developers the necessity of reaching an agreement on what was going to be the standard implemented in the quality gate.


















Conditions 			Add Condition	
Conditions on New Code				
Metric	Operator	Value	Edit	Delete
Coverage	is less than	70.0%		
Duplicated Lines (%)	is greater than	6.0%		
Maintainability Rating	is worse than	A		
Code Smells	is greater than	10		
Vulnerabilities	is greater than	0		
Reliability Rating	is worse than	A		
Security Hotspots Reviewed	is less than	100%		
Security Rating	is worse than	A		

Figure 13: Quality gate conditions

14.5 Create a method in Jenkins library to implement the quality gate

During this task my job was to create a new method in Jenkins to use the quality gate. After looking carefully to the current implementation of sonar, realized that the only necessary thing to do was to add a quality gate flag in the command executed by Sonarqube: **-Dsonar.qualitygate.wait=true** The `-Dsonar.qualitygate.wait=true` command in SonarQube is a parameter that can be used during the analysis of your code to make the analysis process wait for the quality gate evaluation to complete before proceeding.

By default, when you run the SonarQube analysis, the analysis itself is performed asynchronously, and the analysis task is submitted to the SonarQube server for processing. The analysis task is queued for execution, and the analysis process on the server starts immediately, while the quality gate evaluation runs in parallel.

When you include the `-Dsonar.qualitygate.wait=true` parameter during the analysis, it changes the behavior. The analysis process will wait for the quality gate evaluation to complete before continuing. This means that the analysis will not complete until the quality gate status is determined.

The purpose of using this parameter is to ensure that the analysis result reflects the final quality gate status. It allows you to have more accurate and consistent analysis reports, as you will know whether the code meets the defined quality gate criteria or not.

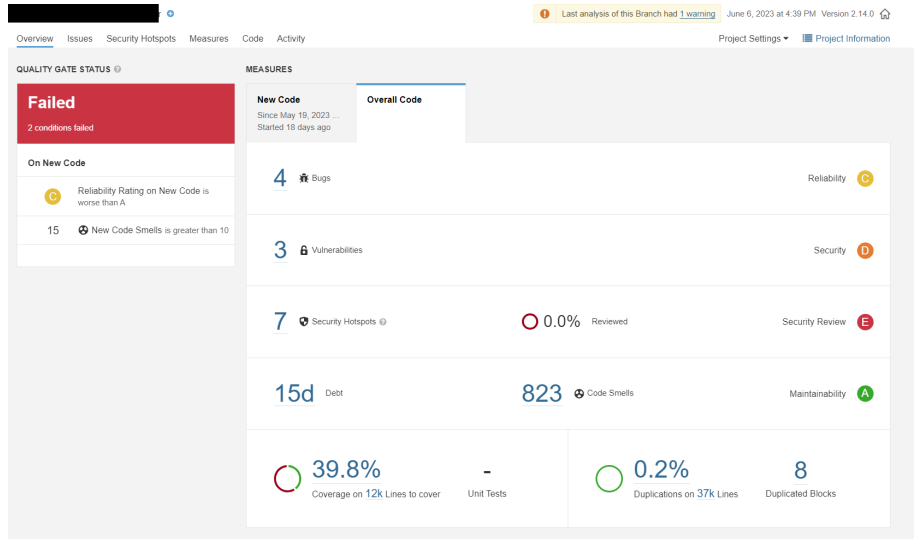


Figure 14: Overall code in sonarqube

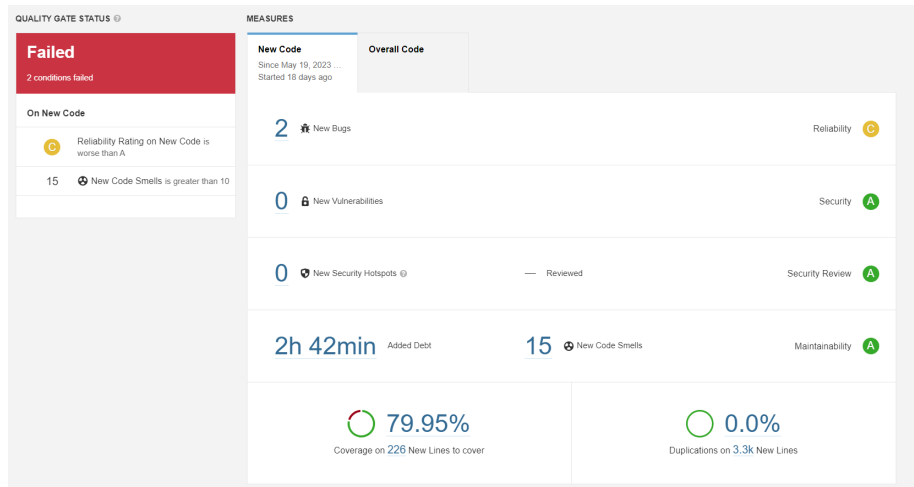


Figure 15: New code in sonarqube

14.6 Create a method in the Jenkins library to write messages in bitbucket PRs

During this task, my objective is to create a new method in the Jenkins library that attaches a new message in the PRs comments with all the quality gate information generated by Sonarqube. To perform that task I have created a script that gets the result from Sonarqube of the execution of the quality gate. To get the result I use the API provided by Sonarqube. Once I have this information I generate the URL of the pr based on some parameters provided by the pipeline environment. After that I make a request using the API from bitbucket to attach in the specific PR, all the information collected by Sonarqube.

14.7 Implement an API to obtain security vulnerabilities from Sonar

The backend was previously implemented when I started the project. But I had to make some modifications. The API is implemented in a Cronjob in a Kubernetes cluster.

A cron job[39] is a type of resource that allows you to schedule and automate the execution of tasks or jobs at specified time intervals or based on a cron-like schedule. It is designed to handle recurring, time-based operations within a Kubernetes cluster.

A cron job is defined by a schedule, which follows the same format as traditional cron expressions. It consists of specific time and date fields that define when the job should be executed. For example, a cron expression of `0 0 * * *` would schedule the job to run once every day at midnight.

The job template defines the task or workload that should be executed according to the cron schedule. It specifies the container image, command, arguments, and other details required to run the job. The job template is similar to a regular Kubernetes job but with the addition of the schedule.

The cronjob is basically deploying the backend and getting all the information from all the projects analyzed by sonarqube using its API. Once it has all the information it is inserted in an Index in Elasticsearch. Once the information is in the Index, Elasticsearch updates the information of a Dashboard in Kibana.

The backend has been developed using Kotlin, Maven, and Spring boot and during this task, I have updated all the necessary endpoints to the new ones from the new sonarqube server API. Furthermore, added some new information generated by the new Sonarqube9.

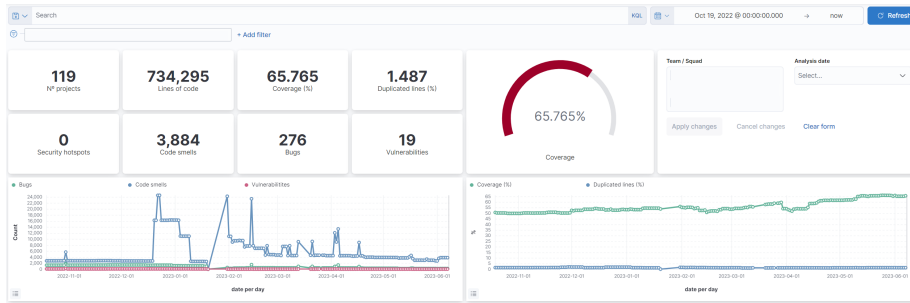


Figure 16: Dashboard Kibana Sonarqube

14.8 Create confluence to allow execute Sonarqube9 locally

The purpose of this task is to create a tutorial in confluence explaining how people can analyze locally, without the necessity of creating a new pr, the code with Sonarqube. To perform the task I have created the following confluence, explaining how to import the certificate and the command and necessary changes depending on the technology the developer is using.

14.9 Problems related to Sonarqube

During the implementation of Sonarqube, I faced many problems, in the previous task I avoided explaining most of them, as you may know developing and implementing new tools doesn't use to be as easy as planned, but I wanted to highlight a specific challenge that I faced during the implementation.

It is important to know that in Mango we are using the free tier of Sonarqube, so some functionalities as analyzing by branches are not available. Also, as previously mentioned, it is important to take into consideration that Sonarqube generates two reports, one for the new code and one for the overall code.

By default, the reference to consider if something is new code or is something that previously was in the project is the last analysis. That resulted in a big issue since we can face the following situation:

Two developers are working on the same code and created two different branches from development. Both have created a pull request to develop and are running the analysis in Sonarqube. The first developer executes the pipeline and the code is correctly compared, and the new code report is correctly generated since it is comparing to develop, but the second developer executes the pr and the new code is not correctly generated, that is because after the first programmer made the analysis the reference to get the new code

is the code on his pull request.

The solution to that problem was to investigate how Sonarqube manages the references, after some time I find out that Sonarqube gives you the following three options:

- Compare to Branch: Compare the code to a specific branch, can not be used in the free tier.
- Compare to a previous day: Compare to a previous day analysis.
- Compare to a previous analysis: Compare to a analysis generated before.

The solution was to use the third option. During the execution of the pipeline on master it executes in the post actions, a Sonarqube analysis, and sets it as a reference. That reference is only changed now whenever the build of master is executed. But, that is also generating other problems, when a new project comes up there is nothing on master so defining what is the reference code it's tricky.

I would also strongly recommend the creation of a command to set the reference code for new code from the local environment, which would help a lot with the derived problem explained before.

Another problem is that we are not executing hotfix the quality gate since we consider it an emergency tool, only used to make the necessary changes to fix the master. But, we are not controlling the quality of the code introduced by hotfix. Probably, the best way to fix this is to set a quality gate for the overall code.

An additional problem is related to false positives. There are many cases where Sonarqube is detecting code smells, bugs or duplicated code that are not. Sonarqube gives the option to disable specific code smells or erase the coverage. Even though I would strongly recommend creating in confluence a document explaining how to avoid false positives to make easier the onboarding process in the team.

I want to mention, that probably the whole success of Sonarqube in the Mango department it's completely related to the mindset of everyone related to the development process. It is really easy to figure out for a developer why quality code is really important, and why tech debt must be avoided, but it is not that easy for a business or a product owner with no background in development.

15 Automatic Dynamic Analysis

During this section I am going to explain how I have implemented in Mango a tool to perform automatic dynamic analysis.

15.1 Test in local Zap

The first thing to do when using new technology is to get familiarized with it. For that, I downloaded and installed the newest version of Zap[40] on my laptop and tested the basic functionalities. The basic analysis of zap consists in:

- **Scanning and Spidering:** ZAP can scan web applications to discover and analyze potential vulnerabilities. It has a built-in spidering functionality that navigates through the application, crawling and mapping its structure and functionalities.
- **Vulnerability Detection:** ZAP performs various security tests to identify common vulnerabilities, including but not limited to Cross-Site Scripting (XSS), SQL injection, Cross-Site Request Forgery (CSRF), security misconfigurations, and more. It analyzes input points, parameters, and responses to detect potential security weaknesses.
- **Active and Passive Scanning:** ZAP offers both active and passive scanning capabilities. Active scanning involves actively sending malicious payloads and checking the application's response for potential vulnerabilities. Passive scanning, on the other hand, observes and analyzes the application's traffic to identify security issues without actively interacting with the application.
- **API Testing:** ZAP can also be used for testing RESTful APIs and web services. It supports the analysis of API endpoints, request and response structures, authentication mechanisms, and potential security vulnerabilities specific to APIs.

All this information is reported in many ways, in the application there is a fancy screen where the user can read the report. Furthermore, Zap allows you to download the report in various formats, including JSON and XML.

15.2 Create a new dockerfile to run zap and a script to obtain the result

The purpose of this task is to create a dockerfile that allows to run an analysis in Zap. The technology has its own dockerfile with an specific python script to run a base analysis. This will help me a lot to automate the process.

```
FROM owasp/zap2docker-stable
RUN mkdir /zap/wrk
RUN pip install -i [REDACTED] mng-wrappers
RUN pip install elasticsearch
COPY zap.py /zap
```

Figure 17: Dockerfile zap

As can be seen in Figure 17 the base image is going to be created from the image provided by Owasp. The docker file is creating a new directory inside because it is necessary to run the analysis, it is the directory where the report is saved. Furthermore, I am installing Python and coping a script in Zap that in the future is going to generate and insert a report in an index in Elasticsearch.

15.3 Figure out how to get an actualized list of all the project that must be analyzed

Since we want to analyze all the URLs in the software department we want a method to automatically get all the URLs. We could have a file with all the URLs but it is not the best solution since it would have to be updated every time a new URL is added.

The final solution was to get the URLs from the kubernetes context. The command executed is the following: **kubectl get ingresses -n istio-system**.

15.4 Create a Jenkins job that is executed every night

When we were discussing the automation of the Zap, one of the options was to add it in the pipelines, but it was not possible since the Zap analysis takes too much time. So, we decided to implement it as a job in Jenkins.

A job is a fundamental unit of work and represents a task or a set of tasks that Jenkins performs. Jobs in Jenkins are used to automate various processes in the software development lifecycle, such as building, testing, and deploying applications.

As can be seen in Figure 18, the script executes the command explained above to get all the URLs. It is executed in the integration context since we want to analyze all the URLs in that cluster. After that, we are executing the analysis in the container implemented before and throw a script that scraps the report and inserts the information in an index in Elasticsearch.

It is important to explain that we implemented a new pod template in Jenkins and this specific script is being executed with that pod template. The pod template can be observed in figure 19. As can be seen, I am using

```

Script ?
1 def String[] projects
2 pipeline {
3   agent {
4     label "k8s-zap"
5   }
6   stages {
7     stage "Get projects to analyse" {
8       steps {
9         script {
10          container("tools"){
11            sh(script: "kubectl config use-context kubernetes-ist", returnStdout:true)
12            sh(script: "kubectl get ingresses -n istio-system", returnStdout: true)
13            projects = sh(script: "kubectl get ingress -n istio-system -o json | jq .items[].spec.rules[0].host", returnStdout: true).trim().split("")
14          }
15          sh(script: "kubectl config use-context kubernetes-ist", returnStdout:true)
16          def string[] projects_rfid = sh(script: "kubectl get ingress -n istio-system -o json | jq .items[].spec.rules[0].host", returnStdout: true).trim().split("")
17          projects = projects + projects_rfid
18        }
19      }
20    }
21  }
22  stage "Execute analysis" {
23    steps {
24      script {
25        container("zap"){
26          for(String project in projects){
27            if(project.trim()){
28              try {
29                sh(script: "/zap/zap-baseline.py -t https://${project} -j output.json", returnStdout: true)
30                catch(Exception e){
31                  sh(script: "python3 /zap/zap.py", returnStdout: true)
32                }
33              }
34            }
35          }
36        }
37      }
38    }
39  }
40 }

```

Figure 18: Jenkins Job Zap

multiple docker images, k8s-tools, zap, and jnlp. Also, it is providing the credentials to insert in Elasticsearch.

15.5 Create a script to collect all the vulnerabilities information

The purpose of this task is to create a script in Python that takes the report generated by Zap and scraps the necessary information as well as inserting it in Kibana. In addition, we want to create a Dashboard that can be used by anyone in the company to quickly find out where are we in terms of security and how we have evolved in the past times.

The script is inserting in Kibana more information that is not shown in Figure 20, that is because most of the information is about specific vulnerabilities what is confidential.

In figure 20 can be seen that the script is collecting the number of vulnerabilities by severity and the total number of vulnerabilities. Also, there is a dropdown in the dashboard that allows the user to select which vulnerabilities are shown in the dashboard by the general URL.

```
apiVersion: v1
kind: Pod
metadata:
  name: k8s-zap
spec:
  containers:
  - name: jnlp
    image: jenkins/inbound-agent
  - name: tools
    image: [REDACTED]:18079/k8s-tools
    command: ['sleep', '999999']
    resources:
      requests:
        cpu: 10m
        memory: 256Mi
  - name: zap
    image: [REDACTED]:18079/zap
    command: ['sleep', '999999']
    resources:
      requests:
        cpu: 10m
        memory: 256Mi
  env:
  - name: ELASTIC_USER
    valueFrom:
      secretKeyRef:
        name: elastic
        key: username
  - name: ELASTIC_PASSWORD
    valueFrom:
      secretKeyRef:
        name: elastic
        key: password
  - name: dind-daemon
    image: [REDACTED]:18079/dind-options
    resources:
      requests:
        cpu: 20m
        memory: 512Mi
  readinessProbe:
    tcpSocket:
      port: 2375
    initialDelaySeconds: 10
    periodSeconds: 5
  env:
  - name: DOCKER_OPTS
    value: [REDACTED]:18079'
  securityContext:
    privileged: true
  volumeMounts:
  - name: docker-graph-storage
    mountPath: /var/lib/docker
  volumes:
  - name: docker-graph-storage
    emptyDir: {}
```

Figure 19: Zap podtempalte.

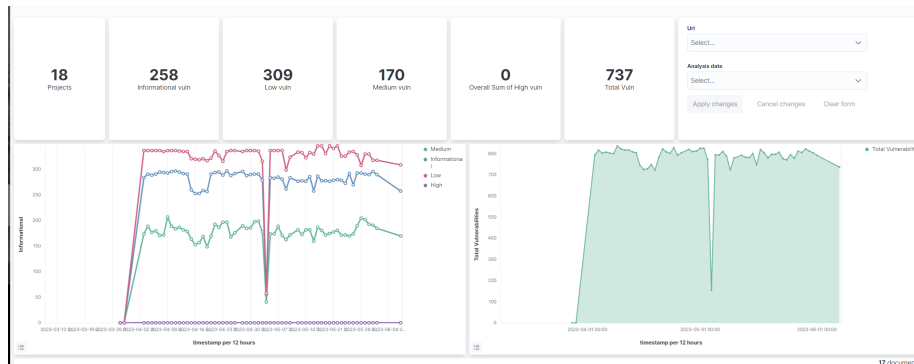


Figure 20: Zap Kibana dashboard.

16 Securing Kubernetes Deploys

Initially, the purpose of this section was to implement from the ground up a new security system in Kubernetes using Checkov. But, most of the epic was deleted after the creation of this section, since the team realized that many things related to the task didn't make sense or couldn't be done. The only part implemented is Checkov in the Gitops pipeline. Even though most of the tasks has been erased and won't be done I think it is important to take at least a quick look at how the technology works and how has been implemented, since securing infrastructure is a key part of the S-SLDC.

Checkov[41] is an open-source static analysis tool that helps you ensure the security and compliance of your infrastructure as code (IaC) templates. It is commonly used in cloud environments, such as AWS, Azure, and Google Cloud Platform, but it can also be applied to other IaC tools like Terraform and Kubernetes. In our case we are going to use it for Kubernetes yaml.

Checkov analyzes IaC files and scans them for potential security issues or misconfigurations. It uses a set of predefined rules or policies to evaluate your code against security best practices and compliance standards, such as CIS (Center for Internet Security) benchmarks. These policies cover a wide range of areas, including authentication, network security, encryption, logging, and more.

When you run Checkov on your IaC templates, it examines the code and provides a detailed report of any security issues it finds. Each issue is associated with a rule ID and includes information about the problem, the location in the code, and recommendations on how to fix it. This enables you to identify and rectify security vulnerabilities early in the development process.

Checkov follows the next steps to run an analysis:

- **Parsing:** Checkov starts by parsing the input files or directories containing infrastructure as code (IaC) templates. It supports various IaC formats, including Terraform (.tf), Kubernetes (.yaml or .yml),
- **Rule Evaluation:** Checkov applies a set of predefined rules or policies to the parsed IaC templates. These rules are based on industry best practices, security standards, and compliance benchmarks. Each rule defines a specific security check or a misconfiguration to look for within the code.
- **Issue Identification:** Checkov evaluates the IaC templates against the applied rules and identifies any security issues or misconfigurations. If a particular rule is violated, Checkov generates an issue report indicating the rule ID, severity level, location of the issue in the code, and additional information about the problem.
- **Results Reporting:** Checkov presents the analysis results in a readable format. The report includes details about each identified issue, such as the rule violated, a description of the problem, and recommendations for remediation. The severity level associated with each issue helps prioritize the necessary fixes.

Even though I didn't make the implementation, I am going to explain it since I feel it is interesting and key to document since the goal of this thesis is to explain how to implement a S-SDLC.

16.1 New stage gitops pipeline

In this subsection, I am going to explain how Checkov has been implemented in the CI/CD. I explained when talking about the pipelines in Jenkins that we had a specific one for the Gitops repositories. This task consisted in creating a new stage in that pipeline.

The new stage is executed in a specific container (gitops-scanner-security). This container has the tool preinstalled and performs a basic analysis of the objects cloned.

16.2 Kibana dashboard with vulnerabilities

The purpose of this task was to create a Dashboard in Kibana to have an overview of the state of the Kubernetes objects' security.

It is implemented through a Cronjob that is executed every night, the Cronjob executes a docker container specifically created through a docker file to execute a script that will execute and collect all the vulnerabilities of all the Gitops repositories. Once it has all the information it inserts the information

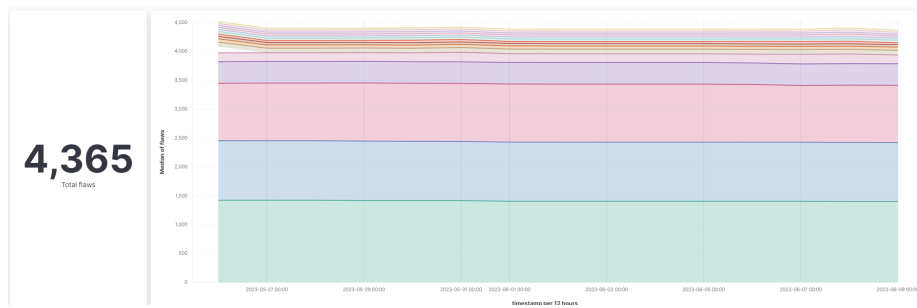


Figure 21: Checkov Kibana dashboard.

into an Elasticsearch Index.

As can be seen in the figure above, the script collects the total number of flaws and the flaws by project. The flaws per project are shown in the diagram each color represents a project.

17 Security Score Card

When I talked about how to improve the security of Mango, one was to create Security Dashboard. In this section, I am going to focus on creating a Security Dashboard using data provided by Security Score Card.

SecurityScorecard[42] is a cybersecurity company that offers a platform and services to assess and monitor the security posture and risk of organizations. The SecurityScorecard platform provides insights into an organization's security practices, vulnerabilities, and potential risks.

SecurityScorecard assigns a security rating to an organization based on various factors such as network security, patching cadence, endpoint security, and more. This rating helps businesses understand their security posture and compare it to industry benchmarks.

The platform continuously monitors an organization's digital footprint, including its network infrastructure, web applications, and third-party vendors. It collects data from various sources, such as public information, threat intelligence feeds, and proprietary sensors.

SecurityScorecard assesses the risk associated with an organization's digital assets. It identifies vulnerabilities, misconfigurations, and potential weaknesses that could be exploited by cyber attackers.

The platform also evaluates the security posture of an organization's third-party vendors or partners. It helps organizations identify and mitigate potential risks arising from their relationships with external entities.

We have to consider that security scorecard is used to evaluate the company for assurance companies. Based on the mark as well as vulnerabilities provided by SSC, the price of cybersecurity insurance is going to vary a lot.

17.1 CronJob to Insert SSC information in ELK

During this task, I am going to implement a brand new cronjob to get the data from Security Score Card and insert it in Kibana every night.

We can observe in figure 21 the final result of the security Score Card cronjob. I have configured a docker Container named security-score-card, this one is generated by a docker file that will be explained above. Furthermore, I specified the Elasticsearch credentials to insert the data generated by the script and the security-score-card-token to retrieve the data.

It can be seen in figure 23 the structure of the Dockerfile is used to generate the container that the cronjob is running. Explanation of the docker


```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: security-score-card
  namespace: tools
spec:
  schedule: "0 2 * * *"
  successfulJobsHistoryLimit: 1
  failedJobsHistoryLimit: 1
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: security-score-card
              image: 79/security-score-card:1.0.0
              imagePullPolicy: Always
              ports:
                - containerPort: 8080
              env:
                - name: ELASTIC_USER
                  valueFrom:
                    secretKeyRef:
                      name: elastic
                      key: username
                - name: ELASTIC_PASSWORD
                  valueFrom:
                    secretKeyRef:
                      name: elastic
                      key: password
                - name: SSC_TOKEN
                  valueFrom:
                    secretKeyRef:
                      name: security-score-card-credentials
                      key: token
          restartPolicy: OnFailure
```

Figure 22: Cronjob security scorecard.

```
FROM pappjam01.intranet.mango.es:18079/k8s-python:latest
WORKDIR /app
COPY requirements.txt .
COPY src/ .
RUN pip install -r requirements.txt
ENTRYPOINT ["python3", "/app/main.py"]
```

Figure 23: Dockerfile security-score-card

commands used:

- **FROM:** This command retrieves from a private sonatype repository the base image of the Dockerfile.
- **WORKDIR:** Is used to set the working directory for instructions that follow it in the Dockerfile.
- **COPY:** Used to copy a file or folder to the Docker image generated. In this case, we are copying requirements.txt and src/.
- **RUN:** Runs a specific command inside the Docker image. In this case, we are installing all the requirements specified by the document requirements.txt.
- **ENTRYPOINT:** This command is used to specify a command that must be run once the docker container is created.

The folder src copied to the container, has a script named main.py that is executed by the container. This script has been generated by me to retrieve all the necessary data from the security scorecard. This data consist in all the vulnerabilities detected as well as the score provided by SSC. Furthermore, the script is transforming the data and scrapping only the necessary information that lately is going to be inserted in Kibana by this script.

17.2 Create dashboard in Kibana

Once all the data is inserted into an index in Elasticsearch, the last thing we have to do is to create a Dashboard in Kibana. During this task, I generated the Dashboard that can be seen in Figure 24. To generate it I used aggregations and buckets. The main purpose of this Dashboard is to give a quick overview of the current situation of Mango Security, but also it pretends to be a way to easily discover vulnerabilities. That is why in the Dashboard we can observe the

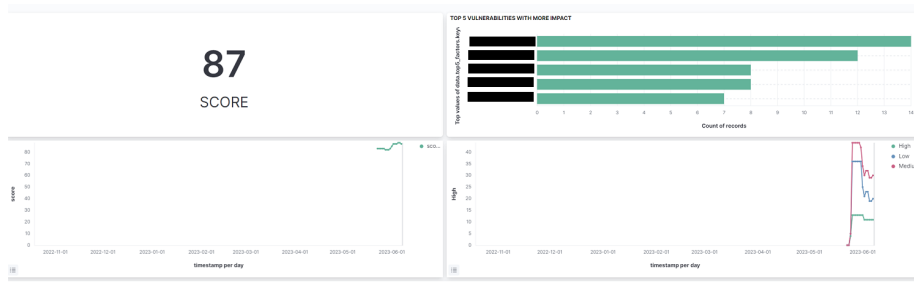


Figure 24: Dockerfile security-score-card

current grade, the evolution of the grade, the top 5 vulnerabilities that have more impact on the score, and the evolution of the total number of vulnerabilities by severity.

18 Showcase Design System DevOps Configuration

During the summer of 2022, Software Engineering decided to start the development and implementation of a Design System that will be used to implement new functionalities by all the frontends of the department. The Design system has been implemented from the perspective of the User Experience and frontends since they have a better overview and understanding of the components that are used and can be reused.

A design system[43] in software development is a collection of reusable components, guidelines, and standards that ensure consistency and cohesion in the design and development of software applications. It provides a centralized and unified approach to design, enabling teams to create user interfaces (UIs) that are visually consistent, intuitive, and efficient.

Benefits of using a Design System:

- **Consistency:** A design system establishes a consistent visual language, including typography, colors, spacing, and UI components. It ensures that all UI elements across different parts of an application or multiple applications within an organization maintain a cohesive look and feel.
- **Efficiency:** By providing pre-defined and reusable components, a design system enables developers to build UIs more efficiently. Instead of reinventing the wheel for each UI element, they can leverage existing components and design patterns, reducing development time and effort.
- **Maintenance:** With a design system, making design updates or implementing design changes becomes more manageable. Since components are centralized and reusable, making changes at the system level automatically propagates to all instances where those components are used, reducing the effort required for maintenance.
- **Accessibility:** Design systems often prioritize accessibility by incorporating inclusive design principles and ensuring that components are usable by individuals with disabilities. This helps to create more inclusive software applications that can be accessed by a diverse range of users.

It is important to consider that the Design System is implemented in a Monorepo that must contain at the end three different projects; A project with all the code of the components, another project containing the code of the storybook and a project containing the code for the showcase.

A monorepo refers to a version control system where multiple projects or software components are stored within a single repository. Instead of having

separate repositories for each project or component, all of them are managed together in a unified codebase.

Nowadays they have already implemented a web application that is a book store where developers can discover as well as obtain information about all the components implemented in the Design System. Now they want to implement a showcase, an application that is implemented from the ground up only using components from the library. The showcase can be used by all the programmers as an example of how to use the library.

18.1 Showcase Design System Configuration

During this task I have created from the ground up the project in Bitbucket as well as configuring Karma, all the dependencies needed, and Jenkins. Also, I want to point out that we figure out during the realization of this task that wasn't a good idea to develop the showcase inside the monorepo. At the very beginning, it seemed like a good idea to implement it inside, mainly because it allowed developers to test the new components developed in the Design System without the necessity to publish a new version. But, after struggling a lot with how dependencies were managed in the monorepo we decided to move the application to a new repository.

18.2 Showcase Design System Kubernetes

I created a different subsection to explain the configuration done in Kubernetes since it is a bit tricky. Since in the very beginning, the project was meant to be a mono repo we only have a single URL for all the projects, so we are in a strange situation where we are going to use a single URL for two different web applications.

The base configuration for any application in Kubernetes consist into having a deployment.yaml, a service.yaml and a virtual service. This Kubernetes objects has been explained before when talking about technologies that were going to be used. To do this task I have created all this objects in a gitops repository.

I want to highlight some specific modifications in terms of the virtualservice to allow that two different projects work using a unique URL. The showcase it's only meant to be accessed when you add in the url /showcase/*, so we have to specify in the virtual service that whenever someone tries to access /showcase/* it has to be redirected to the pod showcase instead of being redirected to the pod book store.

Another necessary modification is to add and rewrite some new commands in the default.conf of the showcase.

```
1  apiVersion: networking.istio.io/v1beta1
2  kind: VirtualService
3  metadata:
4    name: atreyu-showcase
5    namespace: atreyu
6  spec:
7    hosts:
8      - "atreyu.intranet.mango.es"
9    gateways:
10     - mng-gateway
11    http:
12     - match:
13       - uri:
14         | prefix: /showcase/login
15       - uri:
16         | prefix: /showcase/home
17       - uri:
18         | prefix: /showcase/agenda
19       - uri:
20         | prefix: /showcase/story
21       - uri:
22         | prefix: /showcase/start
23       - uri:
24         | prefix: /showcase/
25       - uri:
26         | prefix: /showcase
```

Figure 25: Virtualservice showcase

The `default.conf`[44] file contains configuration directives that specify how NGINX should handle incoming requests and serve web content.

The `default.conf` file defines the server-level configuration for NGINX. It includes directives such as server name, listening port, SSL/TLS settings, request handling rules, and more.

NGINX supports virtual hosts, which allow multiple websites or applications to be hosted on the same server. The `default.conf` file may contain multiple server blocks, each representing a virtual host with its own configuration settings.

The configuration in `default.conf` determines how NGINX should route incoming requests to the appropriate backend servers or services. This includes rules for proxying requests, load balancing, caching, and other advanced routing mechanisms.

First of all, we have to modify the location to `/showcase/` since all the requests done to the server must contain that prefix. Also, we have to change where the root of our server is, now it is going to be in a specific folder named `/showcase/`. Last but not least, we have to add the following command: **`rewrite /showcase/(.*) /1 break`**. This command will rewrite the URL used to get the main folders when accessing the server, so now when the browser tries to get the `main.html`, `main.js` will use that specific prefix and will be correctly redirected.

19 Changes in the planification

In this section I am going to explain which tasks has been erased and which tasks are new compared to the initial report provided to the thesis tutor. A new task has been created in the Sonarqube part:

- **SO10 - Sonarqube old Jenkins:** Implement Sonarqube in the old Jenkins used by some legacy applications..
Duration: 20h
Dependencies: -
Human resources: DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Server with Jenkins, Server with Sonarqube
9

In the Securing Kubernetes Deploy the task 4 (SKD4 - Generator of yaml's) has been eliminated.

Created new 2 new tasks to get the information of Security Score Card and put it nicely in a Kibana Dashboard.

- **SSC1 - CronJob to Insert SSC information in ELK:** Create a new CronJob in Kubernetes and develop a new script to collect the information from SSC and insert it to an index in Elasticsearch.
Duration: 10h
Dependencies: -
Human resources: DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Visual Studio Code, Elasticsearch, Kubernetes
- **SSC2 - Create dashboard in Kibana:** Create the dashboard in kibana with the data inserted in the index and make the necessary changes to the script that generates the information.
Duration: 10h
Dependencies: SSC1
Human resources: DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Visual Studio Code, Elasticsearch, Kubernetes

Created 2 new tasks to configure a new project in bitbucket, Jenkins and Kubernetes to develop a showcase for a Design System.

- SDSC1 - Showcase Design System Configuration:** Create a new repository in bitbucket with the correct version of angular and all the necessary dependencies.
Duration: 5h
Dependencies: -
Human resources: DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Bitbucket, Visual Studio Code, Kubernetes

- SDSC2 - Showcase Design System Kubernetes:** Create the necessary configuration of Kubernetes for this new project. Going to use a url already used for another part of the Design System.
Duration: 5h
Dependencies: -
Human resources: DevSecOps senior, DevSecOps intern
Material resources: Computer with internet, Overleaf, Grammarly, access to the Mango system, Bitbucket, Visual Studio Code, Kubernetes

Task	Duration	Dependencies	Human Resources
PM1	20h	-	Project Manager
PM2	20h	PM1	Project Manager
PM3	20h	PM1, PM2	Project Manager
PM4	20h	PM1, PM2, PM3	Project Manager
AS1	30h	-	Security senior, DevSecOps Senior, DevSecOps intern
AS2	30h	-	Security senior, DevSecOps Senior, DevSecOps intern
SO1	4h	-	DevSecOps Senior, DevSecOps intern
SO2	10h	-	DevSecOps Senior, DevSecOps intern
SO3	10h	-	DevSecOps Senior, DevSecOps intern
SO4	40h	-	DevSecOps Senior, DevSecOps intern
SO5	10h	-	DevSecOps Senior, DevSecOps intern
SO6	10h	-	DevSecOps Senior, DevSecOps intern

SO7	20h	-	DevSecOps Senior, DevSecOps intern
SO8	20h	-	DevSecOps Senior, DevSecOps intern
SO9	10h	-	DevSecOps Senior, DevSecOps intern
SO10	20h	-	DevSecOps Senior, DevSecOps intern
ZAP1	10h	-	DevSecOps Senior, DevSecOps intern
ZAP2	40h	-	DevSecOps Senior, DevSecOps intern
ZAP3	4h	-	DevSecOps Senior, DevSecOps intern
ZAP4	20h	-	DevSecOps Senior, DevSecOps intern
ZAP5	20h	-	DevSecOps Senior, DevSecOps intern
ZAP6	10h	-	DevSecOps Senior, DevSecOps intern
ZAP7	20h	-	DevSecOps Senior, DevSecOps intern
SKD1	20h	-	DevSecOps Senior, DevSecOps intern
SSC1	10h	-	DevSecOps Senior, DevSecOps intern
SSC2	10h	-	DevSecOps Senior, DevSecOps intern
SDSC1	5h	-	DevSecOps Senior, DevSecOps intern
SDSC2	5h	-	DevSecOps Senior, DevSecOps intern
FD1	30h	-	DevSecOps Senior, DevSecOps intern

Table 12. Table summarising tasks. Source: own compilation

20 Final budget

Changes in the budget due to changes in the management plan:

Task	Project Manager	DevSecOps senior	Security senior	DevSecOps intern	Total(euros)
PM1	542.1				542.1
PM2	542.1				542.1
PM3	542.1				542.1
PM4	542.1				542.1
AS1		141.3	145.6	366.4	653.3
AS2		141.3	145.6	366.4	653.3
SO1		28.26		48.8	77.06
SO2		28.26		122	150.26
SO3		28.26		122	150.26
SO4		141.3		488	629.3
SO5		28.26		122	150.26
SO6		28.26		122	150.26
SO7		56.52		244	300.52
SO8		56.52		244	300.52
SO9		28.26		122	150.26
SO10		28.26		122	272.26
ZAP1		28.26		122	150.26
ZAP2		141.3		488	629.3
ZAP3		28.26		48.8	77.06
ZAP4		84.78		244	328.78
ZAP5		84.78		244	328.78
ZAP6		28.26		122	150.26
ZAP7		84.78		244	328.78
SKD1		84.78		244	328.78
SSC1		28.26		122	150.26
SSC2		28.26		122	150.26
SDSC1		28.26		61	89.26
SDSC2		28.26		61	89.26
FD1	542.1				542.1
TOTAL	2,710.5	1,640	291.2	5,344.4	9,028.02

Table 13. Table with human resources expenses. Source: own compilation

The total amount has changed since I have added 50 hours to the project and eliminated 90.

To sum up, the final budget can be seen in table 13.

Type cost	Cost + contingencies
Human resources	9,028.02
Material resources	711.7
Indirect resources	2,578.4
Total	12,317.42
Unforeseen costs	285
Total with unforeseen costs	12,602.42

Table 14. Final budget. Source: own compilation

21 Achievement of technical competencies

- **CES1.1 To develop, maintain and evaluate complex and/or critical software systems and services.** It has been accomplished since I have been maintaining and evaluating with different tools critical corporate software.
- **CES1.2 To solve integration problems in function of the strategies, standards and available technologies.** I have solved and faced many problems using and thinking many strategies. Also implemented new technologies to solve specific problems.
- **CES1.3 To identify, evaluate and manage potential risks related to software building which could arise.** Done in the initial report(GEP) were I talked about risks.
- **CES1.8 To develop, maintain and evaluate control and real-time systems.** Accomplished, explained how to monitor a real time system.
- **CES1.9 To demonstrate the comprehension in management and government of software systems.** Explained in depth how to manage a software project and also how to develop using specific methodologies such as gitflow.
- **CES2.2 To design adequate solutions in one or more application domains, using software engineering methods which integrate ethical, social, legal and economical aspects.** During the initial report(GEP) took into consideration all these aspects.

21.1 Relationship of the project with the degree

- Usage of an Agile methodology to manage large software projects.
- Usage of git and a specific methodology, Gitflow, to successfully manage software projects.
- Automated processes related to software applications with Jenkins and Kubernetes.
- Development and deployment of infrastructure using Docker and Kubernetes.
- Used design patterns to develop new code using python.
- Understood how a real web application works. Also saw how the development process works and what is important to the developer to easily implement quality and safe code.

22 Conclusions

We can say that the project has accomplished the objective. Explain and implement in a real scenario S-SDLC.

The development of this thesis has taught me many new concepts as well as allowed me to meet new people. Through this thesis, I have discovered a new branch of Computer Science not much covered during my bachelor, DevSecOps. It has been a big challenge due to I did not have much experience with most of the technologies I have been working with (Jenkins, Docker, Kubernetes, Sonatype, Sonarqube, Zap). Furthermore, this thesis allowed me to understand how important is to have a friendly environment in a company, since this thesis and all the knowledge acquired wouldn't be possible without my teammates.

I am really happy with the experience. I have acquired much new knowledge that is valuable for my future. Also, I am sure that the Software Department is really happy with the technologies and systems implemented by me during the realization of the thesis.

Also, this thesis has discovered to me a career path, that I will continue in Nice(France) as a DevOps junior.

23 Next steps

There are many things that can be made to improve the system already implemented, even though the current system could be probably used as a reference on how to implement properly a DevSecOps system.

Improve the security of Kubernetes infrastructure using a technology like Prisma Cloud.

Change the tier of Sonarqube to the pay tier. This would increase the quality of the analysis and simplify the current implementation.

Lose dependency on Jenkins. Jenkins is probably not the best technology for the scripting part of DevOps. Losing dependency will allow us to change easily in the future to another technology like GitLab.

Implement a technology like RAM or Lighthouse to have metrics and traceability of the frontend part. It would be nice to know how much time takes for a web to charge and improve the time if necessary.

23.1 Prisma Cloud

Prisma Cloud offers a wide range of security capabilities that help organizations address various aspects of cloud security.

Prisma Cloud enables organizations to assess and manage the security posture of their cloud environments. It helps identify misconfigurations, vulnerabilities, and compliance violations across cloud accounts and services. By continuously monitoring the cloud infrastructure, CSPM helps maintain a strong security foundation.

Protects cloud workloads, such as virtual machines (VMs) and containers, against threats and vulnerabilities. It includes capabilities like vulnerability management, runtime protection, network segmentation, and threat intelligence to safeguard cloud-based applications and data.

It offers network security capabilities to protect cloud networks and applications. It includes features like network segmentation, firewalling, intrusion detection and prevention, and secure web gateways to enforce security policies and prevent unauthorized access.

References

- [1] Acunetix. (2021, June 22). Acunetix | Web Application Security Scanner. <https://www.acunetix.com/>
- [2] Atlassian. (n.d.-a). Jira | Software de seguimiento de proyectos e incidencias. <https://www.atlassian.com/es/software/jira>
- [3] Atlassian. (n.d.-b). What is Agile? <https://www.atlassian.com/agile>
- [4] B. (n.d.). checkov. <https://www.checkov.io/>
- [5] B. (2022, April 26). The Phases Of Agile Software Development Life Cycle Workflow And Project Management. Bitbytesoft.com. <https://bitbytesoft.com/phases-of-agile-software-development-life-cycle/>
- [6] Coverity Scan - Static Analysis. (n.d.). <https://scan.coverity.com/>
- [7] Lynn, R. (2020, March 9). Benefits of Agile. Planview. <https://www.planview.com/resources/guide/agile-methodologies-a-beginners-guide/benefits-agile/>
- [8] OWASP ZAP – Documentation. (n.d.). <https://www.zaproxy.org/docs/>
- [9] Perveez, S. H. (2023, January 30). What is Git: Features, Command and Workflow in Git. Simplilearn.com. <https://www.simplilearn.com/tutorials/git-tutorial/what-is-git>
- [10] SonarQube 9.9. (n.d.). <https://docs.sonarqube.org/latest/>
- [11] Team, K. (2022, July 21). ¿Qué es GitFlow? KeepCoding Tech School. <https://keepcoding.io/blog/que-es-gitflow/>
- [12] What is DevSecOps? | IBM. (n.d.). <https://www.ibm.com/topics/devsecops>
- [13] Kissflow, Inc. (2022, 24 agosto). What is Kanban Methodology | Introduction to Kanban Framework. <https://kissflow.com/project/agile/kanban-methodology/>
- [14] Johnivan, J. R. (2023, 29 enero). Agile Software Development Methodology Principles. Project-Management.com. <https://project-management.com/agile-software-development-methodologies/>
- [15] API Security | Wininovative. (s. f.). <https://winovative.com/services/enterprise-security-solution/api-security/>
- [16] Johnivan, J. R. (2023, 29 enero). Agile Software Development Methodology Principles. Project-Management.com. <https://project-management.com/agile-software-development-methodologies/>

- [17] Offices Electricity consumption | Power consumption in offices | ODYSSEEMURE. (s. f.). <https://www.odyssee-mure.eu/publications/efficiency-by-sector/services/offices-specific-energy-and-electricity-consumption.html>
- [18] Security | Glassdoor. (s. f.). <https://www.glassdoor.es/member/home/index.htm>
- [19] Buy IntelliJ IDEA Ultimate: Pricing and Licensing, Discounts - JetBrains Toolbox Subscription. (2021, 1 junio). JetBrains. <https://www.jetbrains.com/idea/buy/?section=commercial>
- [20] Jenkins User Documentation. (s.f.). Jenkins User Documentation. <https://www.jenkins.io/doc/>
- [21] Learn Kubernetes Basics. (s. f.). Kubernetes. <https://kubernetes.io/docs/tutorials/kubernetes-basics/>
- [22] SonarQube 10.0. (s. f.). <https://docs.sonarqube.org/latest/>
- [23] OWASP ZAP – Getting Started. (s. f.). <https://www.zaproxy.org/getting-started/>
- [24] What Is Software Composition Analysis (SCA Security) | Sonatype. (s. f.). <https://www.sonatype.com/launchpad/what-is-software-composition-analysis>
- [25] ¿Qué es Elasticsearch? (s. f.). Elastic. <https://www.elastic.co/es/what-is/elasticsearch>
- [26] ¿Qué es Kibana? (s. f.). Elastic. <https://www.elastic.co/es/what-is/kibana>
- [27] Introduction to Grafana | Grafana documentation. (s. f.). Grafana Labs. <https://grafana.com/docs/grafana/latest/introduction/>
- [28] What is npm. (s.f.). <https://www.w3schools.com/whatis/whatis-npm.asp>
- [29] Gaba, I. (2023). What is Maven: Here's What You Need to Know. Simplilearn.com. <https://www.simplilearn.com/tutorials/maven-tutorial/what-is-maven>
- [30] <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>
- [31] GitLab. (2023). What is GitOps? GitLab. <https://about.gitlab.com/topics/gitops/>
- [32] Atlassian. (s. f.). What is version control | Atlassian Git Tutorial. <https://www.atlassian.com/git/tutorials/what-is-version-control>
- [33] Guay, M. (2022). What are webhooks? zapier.com. <https://zapier.com/blog/what-are-webhooks/>

- [34] Bigelow, S. J., Nolle, T. (2022). What is observability? A beginner's guide. IT Operations. <https://www.techtarget.com/searchitoperations/definition/observability>
- [35] Wallen, J. (2022, 12 mayo). Scan Container Images for Vulnerabilities with Grype. The New Stack. <https://thenewstack.io/scan-container-images-for-vulnerabilities-with-grype/>
- [36] Codefresh. (s.f.). Understanding Argo CD: Kubernetes GitOps Made Simple. <https://codefresh.io/learn/argo-cd/>
- [37] Prometheus. (s. f.). Overview | Prometheus. <https://prometheus.io/docs/introduction/overview/>
- [38] Brush, K., Lockhart, E., Demaitre, E., Brunelli, M. (2022). What is APM? Application performance monitoring guide. Enterprise Desktop. <https://www.techtarget.com/searchenterprisedesktop/definition/Application-monitoring-app-monitoring>
- [39] CronJob. (2023, 7 marzo). Kubernetes. <https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/>
- [40] Dizdar, A. (2023b). OWASP ZAP: 8 Key Features and How to Get Started. Bright Security. <https://brightsec.com/blog/owasp-zap/>
- [41] Bridgecrew. (s.f.-b). What is Checkov? - checkov. <https://www.checkov.io/1.Welcome/What>
- [42] How SecurityScorecard calculates your scores. (2023, 5 enero). Help Center. <https://support.securityscorecard.com/hc/en-us/articles/8366223642651-How-SecurityScorecard-calculates-your-scores>
- [43] Design Systems: Step-by-Step Guide to Creating Your Own. (s.f.). <https://www.uxpin.com/create-design-system-guide/>
- [44] config | npm Docs. (s.f.). <https://docs.npmjs.com/cli/v9/using-npm/config/>