



TÉCNICO
LISBOA

Security Assessment Automated Reporting

Diogo Filipe Gomes Torres

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Alexandre Paulo Lourenço Francisco

Examination Committee

Chairperson: Prof. David Manuel Martins de Matos

Supervisor: Prof. Alexandre Paulo Lourenço Francisco

Member of the Committee: Prof. Sérgio Luís Proença Duarte Guerreiro

October 2020

Dedicated to the internet, half of my education.

Acknowledgments

I would like to thank everyone that helped me on this journey here.

To my advisor, Prof. Alexandre Francisco for all the help, guidance, motivation, feedback, and support he gave me along the way.

To the staff at MAINSEC, for providing this thesis and giving as much help as needed during the process.

To my parents, for all the support and care to get me through all my education and life, and to my brother, who introduced me to all that is technology.

To my friends, Vlad and Marta for both fun and reality checks, João for being the best roommate and to all my long time friends from Manta Rota.

To all my colleagues at Instituto Superior Técnico, with a special mention to Catarina, for all the restless nights and fun times we shared.

Last, but not least, I would like to thank Instituto Superior Técnico for the education I received, and the personal growth that came from it, which will be fundamental throughout my life.

Resumo

A criação de relatórios é uma forma de apresentar informação organizada com um propósito e público alvo. O resumo de um relatório pode ser partilhado oralmente, mas um relatório completo é, geralmente, transmitido na forma de um documento escrito [1].

No mundo empresarial, especialmente em TI(Tecnologias de Informação), o trabalho e desempenho tem de ser demonstrado através da criação de relatórios, mas esta tarefa requer um grande investimento de tempo e esforço.

O objectivo desta tese é capturar a repetição e previsibilidade na criação de relatórios e automatizar parcialmente esta tarefa para reduzir o tempo, trabalho e esforço investido pelos utilizadores. Para além de criar uma nova maneira de criar relatórios, esperamos disponibilizar uma plataforma onde os utilizadores podem gerir e distribuir os relatórios entre si.

Neste trabalho, respondemos ao desafio da produção de relatórios repartindo o relatório como um todo, em partes estáticas e partes dinâmicas que seriam posteriormente utilizadas como blocos de construção. Com base nisto, o utilizador irá despende menos tempo a escrever informação repetida e focará os seus esforços a escrever os dados relevantes caso-a-caso em cada relatório. A gestão de relatórios será feita providenciando os utilizadores com uma tabela com relatórios relevantes a si e permitindo a criação, remoção e modificação desses relatórios.

O produto desta tese foi um módulo de criação de relatórios composto por três principais componentes. Um servidor responsável pela manipulação dos utilizadores, clientes e relatórios produzidos guardados nas bases de dados ligadas à nossa aplicação e que também oferece uma API para um cliente fazer chamadas às funções implementadas no servidor. Uma aplicação cliente que acede à informação necessária utilizando a API criada no servidor e adicionalmente envia dados para serem guardados através da utilização da mesma API. Finalmente, uma interface mostra um sistema de login, um repositório de relatórios e, mais distintamente, um editor de relatórios.

Palavras-chave: Relatório, Projeto, Falha, Bases de dados, Serviço, Repositório, Editor, API, Utilizador

Abstract

A Report provides a way to present information in an organized format for a target audience and purpose. A summary of a report can be delivered orally, but complete reports are mostly transmitted in the form of written documents. [1]

Most businesses, especially in IT, must showcase their work and performance through the creation of reports. Yet this is a task that requires a large amount of time and effort.

The aim of this thesis is to capture the repetitiveness and predictability of report production, transforming it into a partially automated task. This leads to a decrease in user time, labor and errors, and a new way of creating reports. We intend to deliver a platform for the users to manage and distribute the reports amongst team members.

In this work, the challenge of report creation is tackled by reducing a report as a whole into both static and dynamic building blocks. This means the user will spend less time rewriting repetitive information and focus his efforts into writing the case-by-case data relevant to each report. The management of reports will be achieved by giving the users a table containing the reports relevant to them and enabling them to create, delete and modify those documents.

The result of this thesis was a reporting module composed of three main components.

First, a server responsible for manipulating: users, clients and the produced reports inside the connected databases. It also provides an API for the client application to make calls to functions implemented in the server.

Second, a client application that gets the necessary information from the databases using the server-side created API, sending data to be stored, using the same API.

Finally, an interface that presents a user login system, a report repository and, most importantly, a report editor.

Keywords: Report, Project, Issue, Knowledge Base, Service, Repository, Editor, API, User

Contents

- Acknowledgments v
- Resumo vii
- Abstract ix
- List of Tables xiii
- List of Figures xv
- Nomenclature xvii

- 1 Introduction 1**
- 1.1 Motivation 2
- 1.2 Contributions 3
- 1.3 Thesis Outline 4

- 2 Background 5**
- 2.1 Report Creation Tools 5
 - 2.1.1 SAP Crystal Reports 5
 - 2.1.2 TIBCO Jaspersoft Studio 6
 - 2.1.3 Overleaf 6
- 2.2 File Management 6
 - 2.2.1 Google Drive 6
 - 2.2.2 File Browser 7
- 2.3 Database Systems 7
 - 2.3.1 PostgreSQL 7
 - 2.3.2 MongoDB 7
- 2.4 Web API 8
 - 2.4.1 GraphQL vs REST 8
- 2.5 Web Application Framework 8
 - 2.5.1 Angular 8
 - 2.5.2 Vue.js 9
- 2.6 Discussion 9

- 3 Requirements 11**
- 3.1 Functional Requirements 11

3.1.1	Entities	11
3.1.2	Main Objects	12
3.1.3	Report Repository	13
3.1.4	Report Editor	18
3.2	Development Requirements	19
3.2.1	Front end	19
3.2.2	Web API	19
3.2.3	Report Service	20
3.2.4	Related Services	20
3.2.5	Technologies	21
4	SAAR2020	23
4.1	Approach	23
4.2	Implementation	24
4.2.1	Architecture	24
4.2.2	Server - Designing a GraphQL API	25
4.2.3	Main Object Types	26
4.2.4	Client - Using the GraphQL API	30
4.2.5	Client - Web Interface	31
5	Evaluation	37
5.1	Participants	37
5.2	Material	37
5.3	Tasks	38
5.4	Procedure	38
5.5	Results	40
5.5.1	Time taken to execute tasks	40
5.5.2	Satisfaction	41
6	Final Remarks	44
6.1	Future Work	45
	Bibliography	47
A	Report Document from SAAR202	49
B	Usability Tests Documents	87

List of Tables

- 3.1 Table caption shown in TOC. 13
- 5.1 List of usability testing tasks. 38
- 5.2 List of detailed usability testing tasks. 39

List of Figures

3.1	Use case for all Users in the Report Repository.	14
3.2	Use case for Project Manager in the Report Repository.	15
3.3	Use case for Auditor in Report Repository.	16
3.4	Use case for Reviewer Report Repository.	17
4.1	Architecture Diagram.	24
4.2	Login page presented present when starting to use SAAR2020.	31
4.3	Report Reporitory.	32
4.4	Create a new Project. Selecting a Reviewer from the existing Users in the database. . . .	33
4.5	Editing content in a section using a rich text editor.	34
4.6	Add an Issue to the Report. (a) - Issues sidebar. (b) - Issue automatically placed in the correct sections.	35
4.7	Editing an Issue and adding an image to the Issue.	36
5.1	Time each User took to execute the tasks in both solutions and Ratio.	40
5.2	Analysis of the time taken the execute the tasks.	41
5.3	Satisfaction Ratio Between Solutions.	42
5.4	Average score per question and the difference between those averages.	43

Nomenclature

Acronyms

API Application Programming Interface

CRUD Create Read Update Delete

IT Information Technology

ORM Object-Relational Mapping

REST Representational State Transfer

WYSIWYG What You See Is What You Get

Chapter 1

Introduction

Communication is a key characteristic of a successful relationship between a company and its costumers. This is especially true in software engineering, where the product usually takes the form of source code. However the end user is only interested in the final computer program and how to use it.

This is where a healthy flow of information makes the transaction worthwhile. Proper project documentation of the work aims to inform the client of what was done, how it was done and how valuable it is to the client. This documentation is also important to the producing company as a way of archiving its endeavors for future reference.

A report is a document created to deliver a specific set of information requirements to a certain group of people [2]. As such, it presents itself as a possible way to record the service provided and it is, in fact, a perpetual reality of any IT company. Reports are a way of reducing great amounts of data into the fundamental knowledge points, making it possible to rapidly acquire high level knowledge without needing to be aware of all the details of the day-by-day development process.

The type of information gained by creating reports lets both client and team reach a common ground of understanding about the created product so that both entities are able then to make critical and knowledgeable decisions for the steps to take place in the future.

Good quality reports, like any other type of work, are generated not by one person, but with team effort. For it to be possible, an infrastructure for team members to orderly interact has to be in place. This infrastructure comes in the form of management tools for users, clients, projects and a framework capable of storing all this data; the creation of this infrastructure is the core of our work.

1.1 Motivation

The subject of this thesis was introduced by MAINSEC and came to fruition based on challenges they face on a daily basis.

MAINSEC is an IT company with key expertise in information security and IT security consulting, management and professional services. The team at MAINSEC engages with other businesses to inspect and report the security state of their applications and infrastructure.

Right now, MAINSEC is presented with the challenge of optimizing the process of composing reports, due to the repetitive nature of this task. This type of work usually results in unnecessary strain for the writers and inevitably causes mistakes in production which lead to more wasted time and effort.

Currently, the writers at MAINSEC are using standard word processors like Google Docs or Overleaf (LaTeX editor) to generate reports based on security assessments of applications their clients use or offer. Each of these reports has been either created from scratch or built upon an existing template, but even when starting from a template the users still have to manually write in great quantities.

Every report has an ordered structure.

This structure is composed of static information present in every report, and dynamic elements that are added according to each unique assessment. But, some of the dynamic elements added to a document contain both static and dynamic data that writer has to manually input. Ideally, the editors could reduce the time spent producing the reports if they just focused on writing the dynamic parts.

After creating the reports there is a need to share files between team members and clients. This is accomplished with the use of basic file managers over online file hosting services and email correspondence. These methods are not centralized in one single application and lead to disorganization.

With regards to archiving and administration of files handled by the team, there is a problem with control, privacy and security, or the lack thereof, over the files stored in external services like Dropbox or Google Drive. As security is a great focus over at MAINSEC, the challenge rests in the fact that there is no centralized system where documents can be stored, shared and delegated to each working party. The projects themselves also lack an official and updated log of the many states that projects traverse.

This lack of structure brings about wasted time and a lack of control. It causes team members to not know what or when to do their respective tasks. Using external management systems creates a dependency on foreign entities, and hinders the access to information.

1.2 Contributions

Our contributions and, in particular, the contributions of this thesis are the following:

1. Provide a notion of what report creation is, why it is relevant and what are its current faults;
2. Study existing solutions and related work to target this problem;
3. Detail the specific requirements requested by MAINSEC, regarding the task at hand;
4. Propose and design a solution for this problem, comprised of:
 - A User dependant Login system that changes the way the application behaves relative to the permissions given to that User;
 - A Projects and Reports Repository for team members to manage attached team members and Clients, change meta information of these files (deadline, status,etc) and provide a way to easily find and open Reports for editing;
 - A Report Editor that encapsulates both the capabilities of a common text editor and the custom functionalities that enable the rapid creation of these specific Reports.
5. An evaluation of the created solution by way of a comparison with a similar system that is already in place.

We hope that these contributions help with the understanding and improvement of report creation tools and solutions proposed here may be useful for future work in this area.

1.3 Thesis Outline

This thesis is composed of multiple chapters, each will address at least one of the contributions mentioned above.

Firstly, in the introductory chapter (Chapter 1), we describe the problem we propose solve, as well as a brief introduction to the current paradigm when it comes to solving it. We also define our goals and the structure of the document.

This paradigm will then be elaborated in Chapter 2, where we study the different tools and approaches that are being used to solve the problem nowadays. For each of these approaches we describe their contributions and then discuss what we can learn from them, in order to design a better solution.

Chapter 3 lists all the requirements requested by MAINSEC, so that not only we formulate a solution for the problem in general, but also tailor the solution to meet the specific needs of the team members.

In Chapter 4 we will show our solution, which takes into account both our goals and what we have learned from the other works we studied in Chapter 2.

Then, in Chapter 5, we describe how we evaluated our solution, in terms of metrics that we can use to compare with similar approaches.

Lastly, in Chapter 6, we will conclude with what we have done throughout this thesis and prospects of future work in this area.

Chapter 2

Background

In the following sections we will present some tools and technologies used for report creation, editing and storing that served as inspiration for the development of our own software.

2.1 Report Creation Tools

The programs presented in this section are examples of possible solutions to the report creation goal of this thesis.

2.1.1 SAP Crystal Reports

SAP Crystal Reports is a system designed to take vast existing databases and present them as ready-to-consume information in the form of reports that people, both internally and externally, can use to keep informed and make better decisions. To reach this goal SAP Crystal Reports gives the user an editing software to create pixel-perfect reports that reflect great amounts of information as concise and easy to consume documents.

With this tool, a user can create a report template that can be executed one or more times based on the contents that will populate it. What this means is that a report can be generated to represent a one-time event or be recreated with a certain periodicity to keep the reader up-to-date on important information.

By creating reports using formulas, cross-tabs, sub-reports and conditional formatting, the user can get a clear picture of a hard to understand large data set as well as uncover conclusions that would otherwise be hidden.

This tool enables the user to connect to almost any source of data like large CSV files or NoSQL databases to then produce and distribute them as one of many popular formats like PDF, Word or HTML [3, 4].

2.1.2 TIBCO Jaspersoft Studio

Much like SAP Crystal Reports mentioned before, this editing software was made to design and run report templates using a plethora of visual components like charts, tables and maps.

After creating the layout of the report, the user is capable of exporting in many popular formats that fit any data need, like PDF and spreadsheets or raw CSV and XML documents.

Using Jaspersoft Studio, one can access different types of data sources, including big data, CSV, Hibernate, Jaspersoft Domain, JavaBeans, JDBC, JSON, NoSQL, TIBCO Spotfire® Information Links, XML, or your own custom data source. This software comes available as an Eclipse plug-in or a standalone application [5].

2.1.3 Overleaf

Overleaf is a cloud-based LaTeX editor that allows multiple users to write, edit and publish scientific documents [6, 7]. Overleaf provides the convenience of an easy-to-use LaTeX editor with real-time collaboration and the fully compiled output produced automatically in the background as you type.

Having to write in the form of source code requires a previous knowledge of the LaTeX language to produce documents, but creating a document with LaTeX frees the user from worry around design as this system provides the user with plenty of industry standard layouts and formatting [8].

Overleaf lets the user create templates of documents to be completed for each case, as well as publishing to mainstream supported file formats. However, it has no way of getting dynamic data from other sources and automatically fill in predefined sections.

2.2 File Management

This section focuses on the exploration of different management and distribution systems and how users interact with them.

2.2.1 Google Drive

Google Drive is a cloud-based multi-platform file storage and synchronization service [9].

The service is used by individuals and teams to store, manage and share files online. Users can choose who they share their files with and what permissions they have on those items. They can share access to folders or individual files.

As this tool is a part of the Google Office Suite, from Google Drive a user can open a document for collaborative editing in the other office applications, like Google Docs and Google Slides.

We did not research any more cloud storage services as they are all very similar, offering pretty much the same features as Google Drive, relevant to the needs of this thesis.

2.2.2 File Browser

The generic file browser is the most basic way of storing and managing files. It is included in every main Operating System and is used to store files offline and manage these files in a folder structure. If a user intends to share it with a team member, it must be done using external tools. At most, one can set read/write permissions per file or folder.

2.3 Database Systems

2.3.1 PostgreSQL

PostgreSQL is an open source object-relational database that aims to provide access to large data sets even in the heaviest workloads. With this database system developers can create applications that protect data integrity and are fault-tolerant, while being used by many concurrent users.

PostgreSQL is highly extensible, letting the developer define his own data types, custom functions and write code from different programming languages without the need to recompile the database.

The SQL standard is mostly respected by PostgreSQL, with a few exceptions where this compliance would hinder features or architectural structure. Most features and functions that form the SQL standard are supported even if sometimes with a different syntax [10].

2.3.2 MongoDB

MongoDB is a document oriented database with the scalability and flexibility that the user wants, whilst offering the querying and indexing needed.

By storing JSON-like data documents, MongoDB enables each document to have its own fields and data structure and be able to change these over time.

MongoDB includes useful features like Ad hoc searching, indexing and real time aggregation that provide the user with the capability for intricate access and analysis of his data [11].

2.4 Web API

Here resides the single point of contact between the client and the internal network of micro services (section 3.2.4). The Web API is responsible for routing the requests sent from the Front-end module to the inner micro-services that make the actual manipulation of reports, projects, clients and other data information.

2.4.1 GraphQL vs REST

GraphQL is a query language for APIs, as well as a run-time for fulfilling queries related to the user's data. Unlike other REST APIs, GraphQL provides clients with the power to request just what they need and nothing more through the implementation of a schema. Apart from being able to use a schema, the user is also able to change that schema over time and evolve the APIs or with the APIs with which it communicates.

Apps using GraphQL are fast and stable because they control the data they get, not the server. This means that by defining the requested data in a given query, the user only receives that data in question and not the unfiltered content provided by a server.

With GraphQL the user can write APIs that leverage existing data and code with GraphQL engines available in many languages. This way, the app only has to be concerned with a single API to access multiple and varied storage engines [12].

2.5 Web Application Framework

To create our web application we researched some options for development frameworks. We needed a toolkit that would help us develop the application quickly and that supported all our needs.

2.5.1 Angular

Angular is a platform made for development of web applications. Much like its former version, as AngularJS, Angular remains a popular option for its purpose. Its popularity comes from its capabilities regarding concepts like data binding, which makes dynamic page updates possible, directives that enable the users to create their own HTML tags and therefore a more personalized application. Furthermore, there is dependency injection to easily create reusable and testable code.

Angular is a fully re-written and redesigned version of AngularJS and now is the one that has become most widely used and actively maintained. Angular builds upon its predecessor by using TypeScript which is similar to JavaScript, so the user does not have to learn a whole new language, and extends the capabilities of AngularJS by being web, mobile, and native desktop among many other improvements [13, 14].

2.5.2 Vue.js

Vue.js is an open-source model–view–viewmodel progressive JavaScript framework for building user interfaces and single-page applications [15, 16]. Vue offers much of the same capabilities of Angular, but it does not require the use of Typescript and offers a much less opinionated perspective. This means that a user has a smaller learning curve, if he already knows how use vanilla Javascript, and has more flexibility on how to develop an application as Vue doesn't impose a Right Way to build an application [17].

2.6 Discussion

In this chapter, we presented relevant systems in the area of Report Creation and what other tools can be used to achieve a complete team framework for managing and storing reports.

Yet, no tool is without its imperfections, so we highlight the major faults and features of each type of tool to learn the most from them.

From the Report Creation applications we want to replicate the capability of using a template document, to be filled by the end user and to export as a PDF file, allowing any Client to see it or read it. From Crystal Reports and Jaspersoft Studio, we want to reproduce the capacity for automatically getting data from multiple sources into our document, but we want to use open-source software. Like Overleaf, we must implement an application capable of being used online by multiple users and to have the document automatically saved to the cloud, but we want to control the server where the documents are saved to and for the text editor to behave differently according to the user in session. What is missing from these tools is the custom functionality of adding Issues (Section 3.1.2), specifically, or even the capability to add other modular functionality as needed.

Google Drive, like any file browser, presents the files and folders in an ordered manner, but lacks specific information about said files that need to be present for the user to correctly manage them. Once again we need a system that acts in a particular manner depending on the user, beyond the read/write permissions provided by booth Google Drive and a file browser. The possibility to go from the file management software to the editing software is to be kept. The entities managed by repository are more complex than folders and files, we need a way to represent Clients, Users, Projects and Reports and there is no way to produce this functionality in either of the tools presented.

The database systems we introduce above satisfy our needs as they are already being used at MAINSEC and are an compulsory development requirement (Section 3.2.5).

To develop the web API, we focused on the advantages of GraphQL, because we must use different databases to store, manipulate and provide data to the client application.

Angular and Vue.js present a very similar offering and most developers end up choosing between these two by very particular reasons or simply because they already had past experience with one. Since there was already past experience with Angular, this is its main advantage.

Chapter 3

Requirements

Simply put, requirement specification is a way of keeping track, in writing, of the requisites of a product to be created for a client. This is probably the most important part of documentation created at the beginning of a project and, as such, it has to lay out information in a precise and explicit manner. This results in a clear understanding, by both parties, of the functions that have to be met by the end of the development process. This document will serve as the basis for future work, but it is only tasked with representing what the application has to do and not how it is to be done [18, 19].

In this chapter we show the guidelines to be followed and goals to be met during development, according to MAINSEC.

3.1 Functional Requirements

Hereinafter, we will describe which requirements must be met within each application interface and which requirements belong to their respective actor. So firstly, we will define the roles of Auditor, Reviewer and Project Manager.

3.1.1 Entities

1. The Auditor is the person that will interact directly with the existing application provided by the client and then write the report which contains all the issues found during the assessment. This is the sole entity responsible for creating new reports and directly change their contents and may have to do so repeatedly with each review of the document until it is approved by the Reviewer;
2. The Reviewer is the one who reads and makes suggestions for corrections to be made in the reports created by auditors;
3. The Project Manager is responsible for the management of the ongoing projects and guarantees that these are delivered properly. The management of projects includes creation of projects and reports, dictating the roles of the team members involved in each project, and giving the final validation of a report before delivery to the client.

3.1.2 Main Objects

The paragraphs below describe the objects that are stored and manipulated by the actors detailed in section 3.1.1.

Project

The Project is principal organizational component of this solution. The Project incorporates the information about the team members assign to it and which roles they play.

Each Project has one of three statuses: “Open”, “In Progress” or “Closed”.

Projects hold the Reports the Auditors write after an assessment of a Client application. Each Project has an attached Client.

Report

A Report is the document produced to describe the issues found by the Auditor during an application assessment.

In a management perspective, the Report keeps information about its deadline for review and delivery, and the different statuses it traverses through.

A Report starts its life in a “Open” status, changes to “In Progress” as the Auditor starts writing in it, then enters a cycle. This cycle begins when the Auditor finishes the first draft of the document and sets the Reports to “Review” and waits for the Reviewer actor to read and append comments as needed.

When the Reviewer finishes reading the produced content and writing his suggestions to the Auditor, he sets the Report to a “Reviewed” state.

The third phase of the review cycle is entered when the Auditor notices the newly written suggestions of the Reviewer and starts to edit the Report accordingly, setting the status to “In Progress”.

Th final status is “Closed”, which is set by Project Manager when he has read the Report and is confident to deliver it to the Client.

Issue

Issues are the main components that distinguish Reports. After the Auditor has finished the assessment on the application provided by the Client, the Auditor uses the Report to showcase every fault found.

The Auditor is tasked with matching the faults, found during the Client application assessment, with the corresponding Issue that best describes them. Having found the correct Issue, the Auditor must place the static contents of the Issue, retrieved from the Knowledge Base (3.2.4), in two sections of the document.

The two sections of the Report that list Issues are the “Summary of Assessment Results” and “Assessment Details”. In the “Summary of Assessment Results” section, Issues are listed in a table with their title, description and what is the subsection where this Issue is described in detail, in the “Assessment Details” section. In the “Assessment Details” section, the Issue will be described in full detail, bringing from the Knowledge Base (2), all the static information about this Issue.

The second part of every Issue is dynamically created by the Auditor, to explain how the Issue relates to the specific fault in the Client program. This dynamic data is written inside the subsection the Issue occupies in the “Assessment Details” section of the report. This dynamic data is composed of text and images that explain the state of the fault and what the Client could do to correct it.

3.1.3 Report Repository

The Report Repository offers a view of all projects and reports to be accessed and managed by the Auditor, Reviewer or Project Manager.

The interface presented to users must take the form of a table listing projects and reports. The columns delineate project and report features like name, status, deadlines, team members, etc.

There are capabilities exclusive to each user and a few that can be executed by all of them. In this table, we will detail the actions that each entity is capable of.

Capability	Project Manager	Auditor	Reviewer
Sort/filter projects and reports by feature	✓	✓	✓
Preview report	✓	✓	✓
Export report as PDF file	✓	✓	✓
Change report/project status	✓	✓	✓
Create a new project	✓	✓	✓
Set project meta information	✓		
Delete a project	✓		
Create a new report	✓	✓	
Set report deadlines	✓	✓	
Delete a report	✓	✓	
Edit report		✓	
Attach comments to found errors			✓

Table 3.1: Required capabilities per entity.

As shown in the table above, all parts involved can sort and filter through all active projects and reports for further actions. For example, in a table of projects, the user can use a search bar to find a project using a keyword. To change the order the projects are displayed, a user can choose, for example, to sort them by status.

Upon selecting a Report any user can preview how it looks when finished and save it as a PDF file.

Although all entities are able to change the status of a Report, there are different statuses available to different team members. This is demonstrated in the Use cases shown in Figures 3.1 to 3.4.

A new Project can be created by anyone, but only the assigned Project Manager can change and delete the Project from then on. The meta information edited by the Project Manager includes the Project name, team members, target Client and status.

Report initiation and deletion is exclusive to a Project Manager and its Auditor, and during its writing both these actors are able to change its deadlines. Only the Auditor has the right to write the Report.

Lastly, the Reviewer is the entity solely responsible for opening the Report to add comments for the Auditor to read afterwards.

In the next part of this section, we will show a use case for each entity using the Report Repository.

All Users Use Case

This Use Case shows the capabilities shared by all Users, when interacting with the Report Repository.

The repository shows the Projects and Reports in a table format. The order and contents of this table can be changed by sorting the Reports/Projects by feature, for example, closest deadline first. The contents of the table change if a User filters the items by searching for a keyword.

Having selected a Report, a User can preview its contents to see how it would look like as a finished product, and export the document to share it with a Client or team member.

Finally, anyone can start a new Project, assigning the people involved and setting its status.

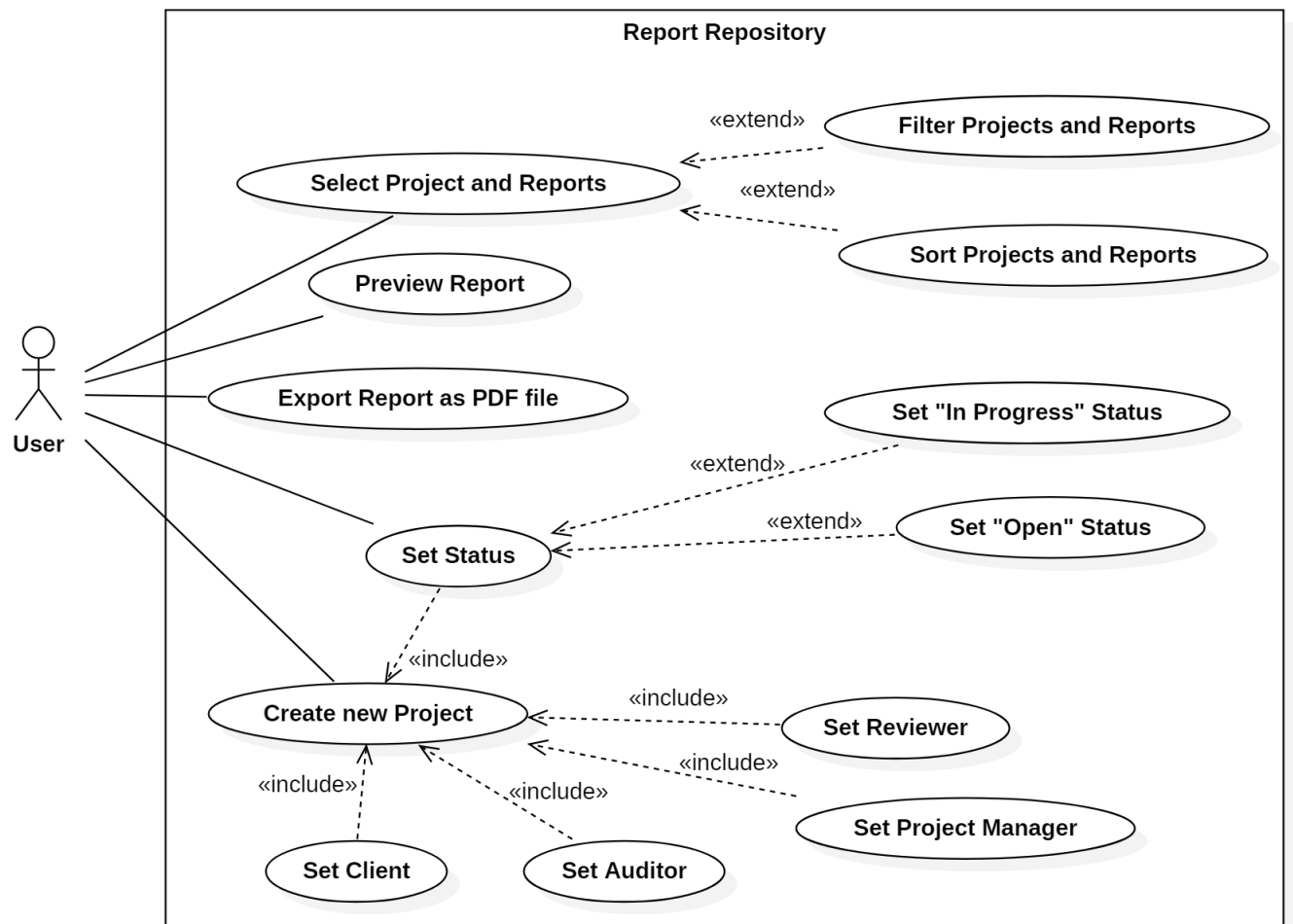


Figure 3.1: Use case for all Users in the Report Repository.

Project Manager Use Case

As the title entails, the Project Manager is in charge of project management. So, from the Report Repository, the Project Manager can delete and manipulate the properties of Projects. The Project Manager is able to change information about Projects, like the associated Client and integrating team members.

Reports can also be created and deleted by the Project Manager. After a final review, this User can change the status of a Report and set it up for delivery.

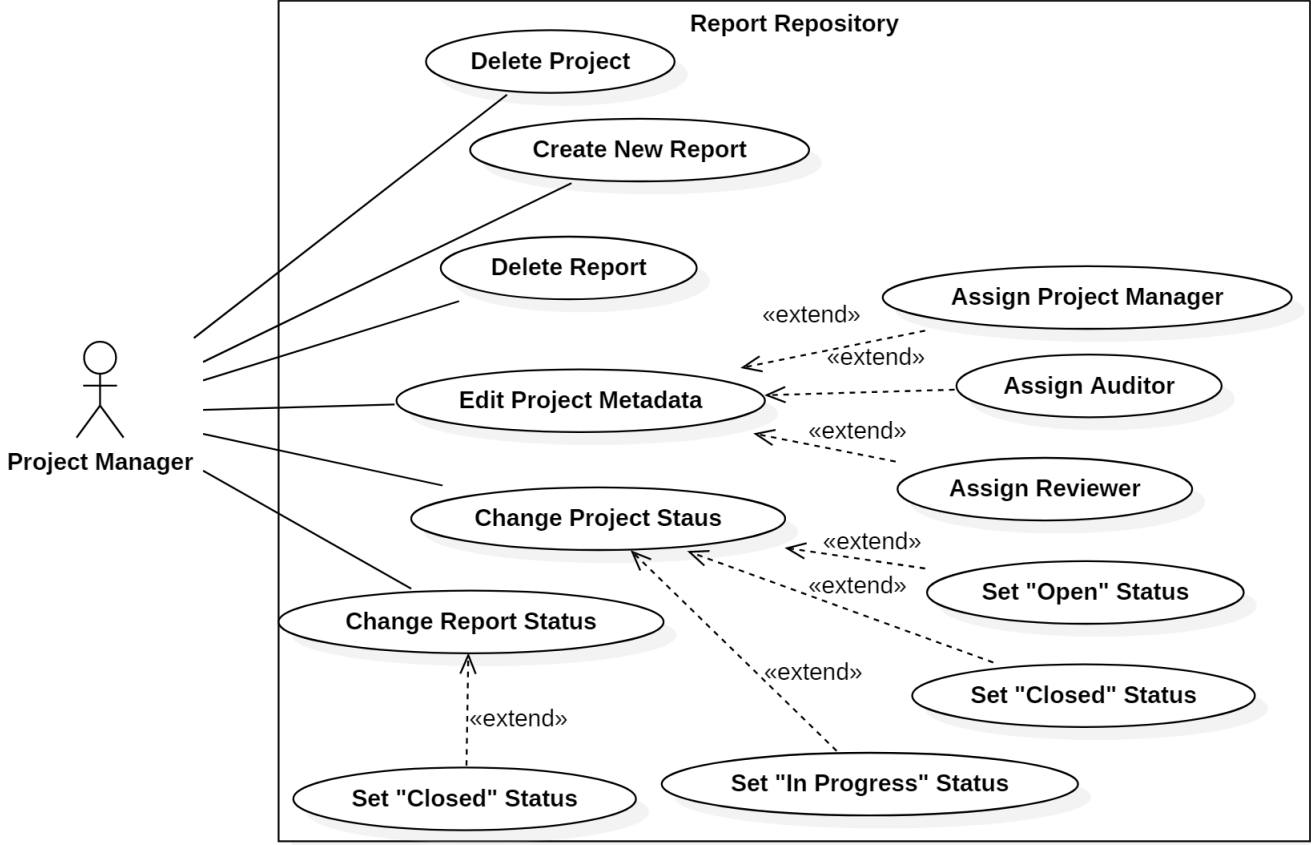


Figure 3.2: Use case for Project Manager in the Report Repository.

Auditor Use Case

In Report Repository, an Auditor can view the Projects and Reports he has been assigned to. Then he can create and edit metadata of Reports. The metadata includes, deadlines for review and delivery and the status of the Report. This User is the only one that can open up a Report to, then, edit its contents.

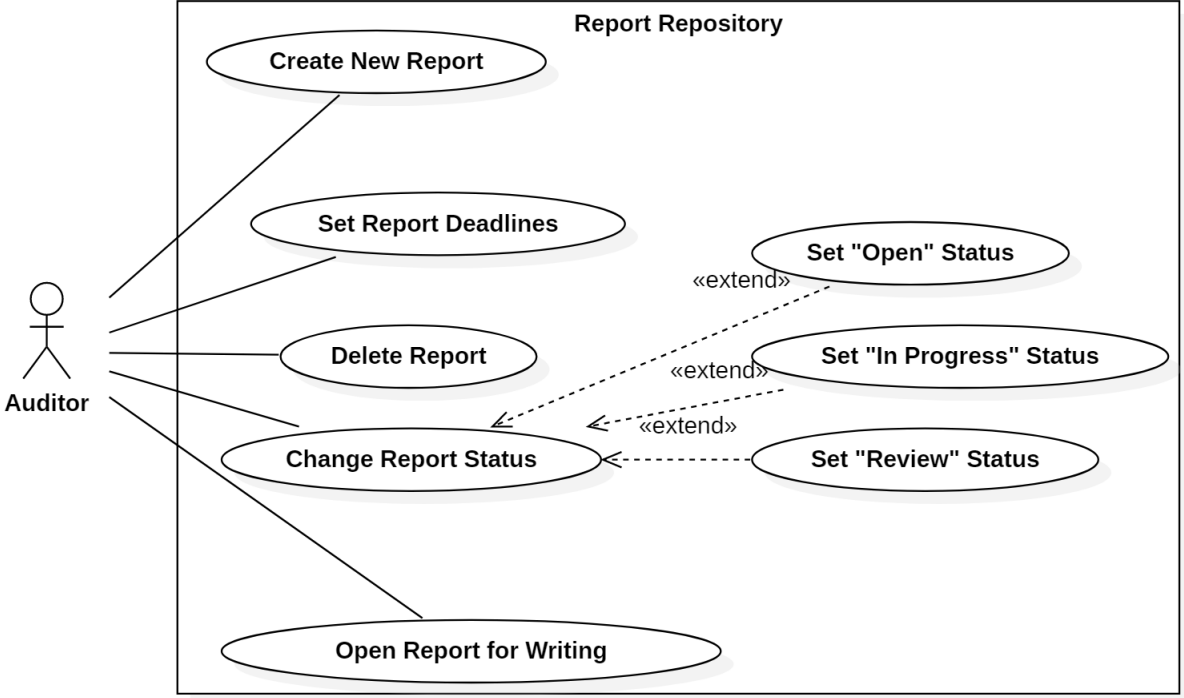


Figure 3.3: Use case for Auditor in Report Repository.

Reviewer Use Case

A Reviewer will have the job of detecting and providing comments on faults found in the report document. These suggestions attached to the errors are only viewable by company employees and do not change the content of the document in any way.

In the Report Repository, the Reviewer can open the reports assigned to him and after having left comments, or not, is able to change the status of the report so that the Auditor is notified to make changes on the document.

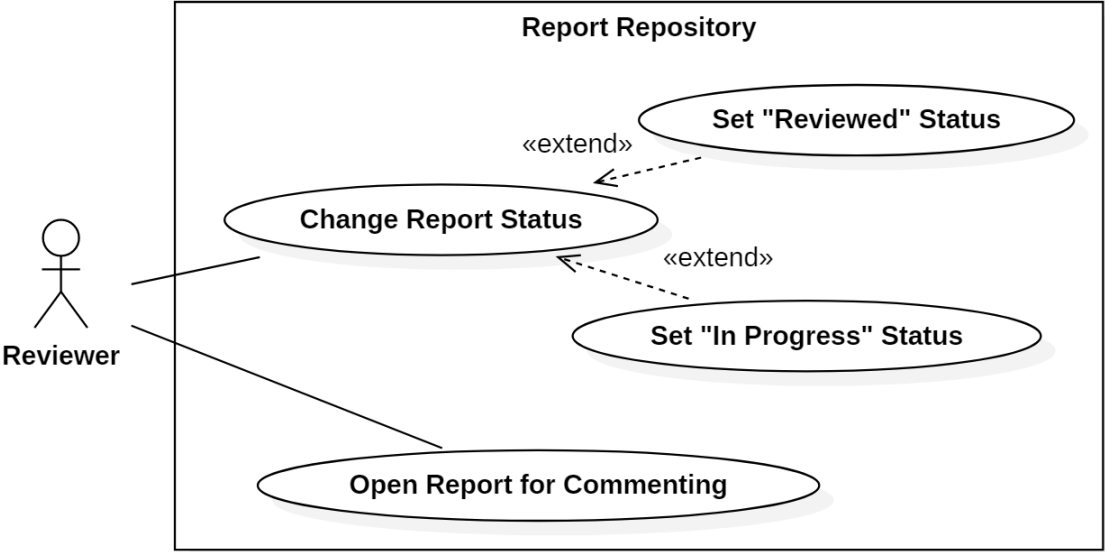


Figure 3.4: Use case for Reviewer Report Repository.

3.1.4 Report Editor

The Report Editor presents a single interface for two different users, but each user will get a different set of functionalities. These two users will be the Auditor and the Reviewer.

The layout must contain three main features: scrolling pages representing the opened document, just like any other word processor, where the Auditor sees and fills a report as a regular document in any other text editor; a sidebar with a search engine that pops up when an Auditor wants to insert a new Issue into the document, helping him, more easily, find the right Issue; and finally, a comment section for the Auditor to read and the Reviewer to write on.

The report must follow the existing ordered structure where the document is a composite of static and dynamic data. Some components are already partially constructed and present in every report and, then, the user only has to fill in the blanks. Other parts are uniquely added to each report, as is the case with the Issues.

An Issue is an object that can be inserted into the document, inside its respective section. Each of these Issues is made of three components: a static text description that is imported from the Knowledge Base (page 20); the technical details and current state to be manually filled in by the user; and the severity which determines where the *Issue* is placed in the document.

The comment section is placed alongside each respective page, i.e., a comment section only contains comments relevant to that page. The Reviewer add and modify annotations, while the Auditor only sees what the Reviewer wrote before.

With this basic knowledge of how the Report Editor operates, we will now enumerate the functionalities provided to both users.

Auditor

1. Add page content;
2. Edit page content;
3. Search Issue;
4. Insert Issue;
5. Assign a severity score to an Issue;
6. Edit technical details inside the Issue;
7. Read comments.
8. Periodic and automatic Report saving.
9. Manual saving of a Report.

Reviewer

1. Add comments;
2. Edit comments;
3. Periodic and automatic Report saving.
4. Manual saving of a Report.

3.2 Development Requirements

As the work produced during this project is to be integrated in an existing micro service platform, there are some development conditions that have to be respected for proper integration between new and existing software.

This section will specify how the new components will work, their functionalities and how they will interact with the existing system. Additionally, we will introduce how some existing micro services operate, as they are crucial to the new implementations.

3.2.1 Front end

The Front-end is where the user interface for the application will be developed. It is the visual representation of the functionalities of the product without ever demonstrating the complex inner workings that enable those functionalities.

This is how the user interacts with existing data and provides input for the internal mechanism. The main objective of this component is to create an interface similar to any regular word processor, so that the user is already familiar with the layout, adding other interface elements corresponding to a custom instrumentation like the sidebar search engine to select Issues.

3.2.2 Web API

The Web API is responsible for relaying client requests between the user interface and the internal network. This mechanism must have queuing capabilities to guarantee ordered delivery of messages, even in heavy and asynchronous workloads. A caching system is needed to maintain access to already authenticated user sessions.

3.2.3 Report Service

The Report Service will be a composition of three components, a back end server that directly connects to the required databases, the Web API (Section 3.2.2) definition and the a client application where the Front end (Section 3.2.1) is implemented.

The back end server has the responsibility of creating the necessary functions to manipulate data on multiple types of database systems. This is also where we will implement the new objects and respective operations needed for representing the Reports and necessary infrastructure around them.

A Web API must be defined as an abstraction layer for the client application, removing the need for the client to directly interact with data sources and having the advantage of only being required to use a single language to access those different databases. This Web API is what enable the connection between the server and client application layers.

Finally, the Report Service is where we will build the Front end interface to be utilized by the team members. This interface lets a team manage, create, edit and, lastly, export the result Reports to deliver to the Client.

3.2.4 Related Services

The services described below represent the infrastructure in which the the service produced in this thesis must integrate to.

1. **User Service** is a micro service responsible for every aspect of user management in the platform. Registration, authentication, authorization and removal of clients and their information are all functionalities associated with this service;
2. **Knowledge Base Service** will provide the database to be used as a source for searching Issues and later populate the created reports. Besides hosting data relating to Issues this service will hold other types of the static data required for the production of these reports such as Disclaimers, Terms of Use or other textual descriptions relevant to the client. Data creation, deletion and modification will be done exclusively by the Knowledge Base Service;
3. **Project Service**, similarly to the Knowledge Base Service, serves as a system for storing and maintaining data. In this case, it will manage projects themselves and all the information related to each project. Through this service a team members can create, delete, and modify projects. This includes attached reports, client information, assigned team members and scheduling.

3.2.5 Technologies

The current work was proposed by MAINSEC, and the end product has the objective of being integrated with an existing infrastructure. So, there are some technologies that must be used in order to comply with the other systems in place.

Our application has to be able to connect with the database systems, PostgreSQL and MongoDB, that hold information about Users and Clients working with MAINSEC, as well as the static data for Issues. Having these databases already in place, we are required to use them to store the newly created information from Projects and Reports.

Both Angular and GraphQL were not strictly marked down as the only options, but came highly suggested, because these technologies had been used by their team members. As such, they would be better equipped to help us with these tools during the development process.

Chapter 4

SAAR2020

4.1 Approach

Our approach started by understanding the existing services in place at MAINSEC. Looking at the services, we found what information they store and how to transfer that data between the services and our new application.

The User Service and Project Service use the PostgreSQL database system to store, respectively, user and projects information. The Knowledge Base Service is using MongoDB to store the static Issues and as referred in section 3.2.5, our Report Service must use this same database to store the reports as they are created.

To bridge all the required databases, existing and the ones formed during development, a better approach is a server that is capable of connecting to different databases, using distinct protocols and provide an interface to communicate with them all using only one protocol.

To satisfy the needs previously mentioned, we designed a GraphQL API using a NodeJS server that enabled the front-end to perform CRUD(create, read, update and remove) operations in our databases.

After making sure we were able to connect with the existing databases, we moved on to expanding the existing objects, Projects and Users, and created a way to archive Reports. All of this was possible using GraphQL.

Having the basic capabilities of our server working, we started building our interface in Angular. Based on what users at MAINSEC already used before and their current unmet needs, we started work on a Report Repository similar to most file storage services. This means, a table showing the Projects and/or Reports belonging to a given user, that showed in its columns some meta information (name, status, etc.) and the actions that the user could perform.

Regarding the Report Editor, we opted for a WYSIWYG (what you see is what you get) system where as the user interacts with the pages of the Report the interface reflects exactly what the exported document will look like.

The report mainly acts as a form to be filled by the user and that is done in one of three ways. The Auditor can double-click any page and submit, in a pop-up, the pertinent information to that page, as

most pages have exact amount and types of data to be filled. Some pages with sections that have free form content, present the user with a text editor similar to a regular word processor like Microsoft Word. Lastly, the Auditor adds or removes Issues and their respective content to the report by searching and selecting in a sidebar that presents itself when needed.

Finally, in the Report Editor interface, there is the possibility of reading or editing comments relevant to a given page. Each page where comments could be relevant has a comment section next to it. The Reviewer reads the report and proceeds to write comments on the necessary pages for the Auditor and later the Auditor uses this comments to correct and refine the document.

4.2 Implementation

To implement our solution in a safe environment and without compromising private client information, we recreated all the existing MAINSEC services and databases as mock-ups. This way we could use the existing functionalities and add to them without putting their service at risk.

4.2.1 Architecture

Our application follows a two layer structure, front-end and back-end, connected with an API that transfers the necessary data between them. The first layer is the Report Service, a back-end server where we designed the API responsible for making requests to get data from our databases and sending data to be archived in those same databases.

The front-end layer is composed of a client application that uses our GraphQL API to execute the tasks, sent from the user interface, on the back-end server. That same client application is where we developed the user interface using Angular.

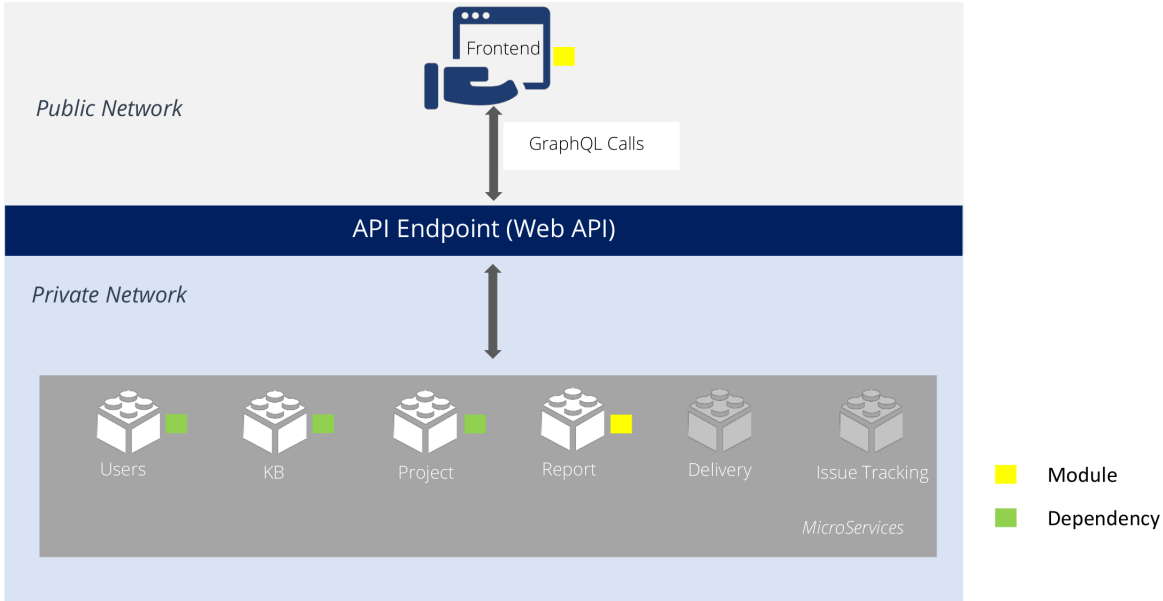


Figure 4.1: Architecture Diagram.

4.2.2 Server - Designing a GraphQL API

The back end of our application was implemented using Apollo Server [20], which is a GraphQL server that works with Node.js [21] HTTP server frameworks, in this case we are using Express [22]. Coupled with our Apollo Server, we used two ORM technologies, Sequelize [23] and Mongoose [24], to help us map the objects from the PostgreSQL and MongoDB databases.

With all the tools mentioned above, we developed our service using three distinct components to manage each object involved in our application. These components are the schema, the models and the resolvers, which we will explain below.

We begin with our schema files. These declare all object types and possible functions regarding these objects, using GraphQL schema. This means that for a given object, like a Project, we defined what are its features (name, status, date, password, etc.) and what actions the application can execute on these objects (create, read, update, delete). As shown in section 4.2.3.

Secondly, we create the models. In these files we define the type of each feature in an object, i.e., the name of an object would be saved as a string, and a list of Reports in a Project would be mapped to an array. The way we produce the model files depends on what ORM technology is being used to match each database. Models for objects stored in a PostgreSQL database are defined using Sequelize, and Mongoose is used for MongoDB database systems.

Lastly, resolver files contain the implementation of the functions that we can use to access and manipulate the data stored in our objects. For example, to change a review deadline of a Report, we have to input the Report id and the desired new deadline, then the service will return the updated Report object to be presented in the interface. As developing models differ depending on the target database, we have to use both Sequelize and Mongoose to implement how the CRUD functions are executed in their target database.

The complete service definition enables the client-side application to interact with the objects and their functions using only the generic GraphQL schema to make calls to the service. The service, then, internally executes the functions using their defined resolvers, without the user needing to know how each database works.

4.2.3 Main Object Types

In this section we will explain how we define the objects of our solution so that they work correctly with the different databases and store the required data.

In our implementation, we split Report into two objects. The first represents the report in the Report Repository, only containing meta information necessary for the Users to manage and select it for viewing or editing. The Report Document object defines how the document and its contents are stored in the MongoDB database system.

Project

```
type Project {
  id: ID
  name: String
  status: ProjectStatus
  reports: [Report]
  auditor: [Auditor]
  reviewer: [Reviewer]
  projectManager: [ProjectManager]
  client: Client
}
```

The Project object contains the project 'id' and 'name' for internal and external identification, the id being automatically created by PostgreSQL upon creation of a new project. The status of the project is an option of three statuses: 'Open', 'In Progress' and 'Closed'. Starting as 'Open' when the project is created, changes to 'In Progress' when reports start being written, and is 'Closed' when ready to be delivered to the client.

The Project also contains a list of its Reports, team members (Auditor, Reviewer and Project Manager) and a Client. These entities are linked to a Project using Association[25] from Sequelize. For example, Reports share an Association "belongsTo" with a single Project, while the Project shares an Association "hasMany" with its Reports.

Report

```
type Report {
  id: ID
  projectId: String
  reportDocumentId: String
  name: String
  status: ReportStatus
  reviewDeadline: String
}
```

```
    deliveryDeadline: String
}
```

Same as the Project object, the Report uses 'id' and 'name' for its identification and the 'id' is also generated by PostgreSQL. We chose to store the identification numbers for the parent Project, 'projectId', and corresponding Report Document, 'reportDocumentId', as these were used to get relevant information from every object without having to replicate the same data multiple times. For instance, as the Report Document is created it automatically fills who are its team members and Client, because we use the 'projectId' to get to those fields.

Both 'reviewDeadline' and 'deliveryDeadline' help the users organize their work schedule and the 'status' holds the 'Open', 'In Progress', 'Review', 'Reviewed', and 'Closed' states to achieve the same goal.

Report Document

```
type ReportDocument {
    id: ID
    reportId: String
    projectId: String
    cover: Cover
    executiveSummary: String
    introduction: Introduction
    assessmentInformation: AssessmentInformation
    summaryOfAssessmentResults: SummaryOfAssessmentResults
    assessmentDetails: String
    appendix: Appendix
    issues: [Issue]
    reviewerComments: ReviewerComments
}
```

In the case of the 'ReportDocument' object, the three identification features do the same job as the 'Report' object, but now the 'ReportDocument' 'id' itself is generated by MongoDB, as this is where we store the reports written by the Users.

The fields 'cover', 'executiveSummary', 'introduction', 'assessmentInformation', 'summaryOfAssessmentResults', 'assessmentDetails' and 'appendix' are complex objects composed of three types of data.

First, there is static data that has to be present in every document, regardless of the Project the Report belongs to.

Second dynamic data that is automatically filled by our client application using information from our server application. For example, when a new Report is created inside a Project, information about the team members and Client assigned to the Project is automatically inserted into the Report Document. If a Reviewer or Auditor is inserted or removed from a Project, that is reflected in the document.

Lastly, the dynamic data that is written by the Auditor responsible for the Report.

The 'Issues' field is an array of 'Issue' objects, that are described below.

The 'reviewerComments' field points to a 'ReviewerComments' object, that is described below.

Issue

```
type Issue {
    id: ID
    severity: Severity
    title: String
    description: String
    impact: String
    remediation: String
    cvssVector: String
    otherReferences: String
    technicalDetails: String
    currentStatus: String
    issueFigures: [IssueFigure]
    reviewerComment: String
}
```

The Issue has a unique 'id' from MongoDB, as MAINSEC already was storing them in this database. The static data imported from the servers is transmitted to our application though the fields: 'title', 'description', 'impact', 'remediation', 'cvssVector' and 'otherReferences'.

The 'severity' is selected by the Auditor when adding a new Issue to the document. The rest of the fields are dynamically submitted by the User, writing with a word processor to 'technicalDetails' and 'currentStatus' and adding relevant captioned images to the Issue through 'issueFigures'.

Each Issue keeps its comments, added by the Reviewer, in 'reviewerComments'.

Reviewer Comments

```
type ReviewerComments {
    cover: String
    executiveSummary: String
    documentManagement: String
    documentStructure: String
    assessmentScope: String
    constraints: String
    proceduresAfterTheAssessment: String
}
```

For the Reviewer comments we only allocated space to sections where dynamic data is entered. As each of these sections only take up a single page, all comments can be saved to a single string per section. As mentioned in the Issue object description above, there is no need to store comments for Issues in here, because each Issue archives its own comments.

Users and Clients

```
type User {
    name: String
    email: String
    password: String
}

type Auditor {
    name: String
    email: String
    password: String
    role: Auditor
}

type Client {
    name: String
    email: String
}
```

Each User begins as a basic User with only its login information ('name','email' and 'password'). As it is added to a Project as Auditor, Reviewer or Project Manager, the User object is extended with a 'role' field, affecting the way the application behaves.

A Client is just a User without a 'password' field as there is no intention for it to interact with our program.

4.2.4 Client - Using the GraphQL API

For the client application to get, manipulate and save data in our databases, we had to write functions to interact with the GraphQL API implemented in our server application. As it would be a taxing task to manually create client-side functions for every action possible in the API, we used the tool GraphQL Code Generator [26] to automatically generate the desired operations.

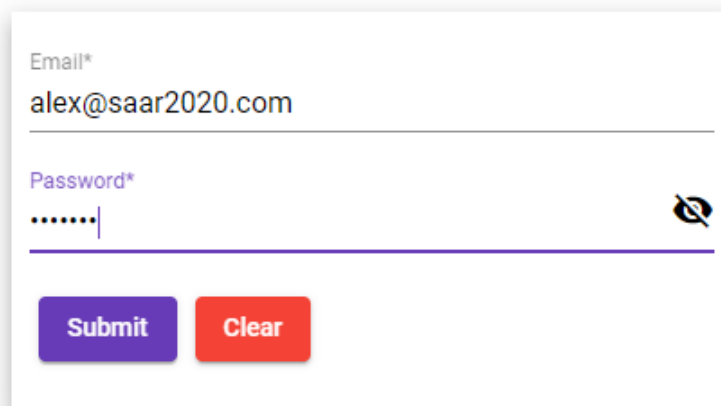
To effectively use GraphQL Code Generator, we created a folder where each file contains a mutation or query operation written in GraphQL schema. Then, using this folder, and giving the location to our GraphQL Server to GraphQL Code Generator, all the necessary code is generated into a target folder inside the application project and the Server functions become available across our Client application.

Automatic saving of a Report is possible, because the input of new information to the current document is done through the GraphQL API and each call only fills a portion of the document. This means, that filling a single page with content equals an automatic save, instead of saving an entire document at a time.

4.2.5 Client - Web Interface

In this section, we will demonstrate, through the help of figures, how the principal interface components and features were implemented and how the User interacts with them to accomplish the tasks proposed at the beginning of this thesis.

As a whole, the interface was developed using Angular. We chose Angular Material [27] as the design signature of our application. Angular Material is a component library, and we used this library to create the core components present throughout our application.



The image shows a login form with two input fields and two buttons. The first field is labeled 'Email*' and contains the text 'alex@saar2020.com'. The second field is labeled 'Password*' and contains seven dots, with a small eye icon to its right. Below the fields are two buttons: a purple 'Submit' button and a red 'Clear' button.

Figure 4.2: Login page presented present when starting to use SAAR2020.

Login

The first interaction the User has with the application is the login screen, where the system takes the User information to check if he can use the program, and to select the correct data to present to the User. The login information is submitted through a simple form.

Report Repository									Report Editor	↗
Projects	Reports	Clients	Users							
+ Create	Search									
Name	Status	Reports	Auditor	Reviewer	Project Manager	Client	Client Email	Actions		
Project B	OPEN	Relatorio 2 Relatorio 7	Alex,	Ana,	Pilar,	INESC	client@inesc.pt			
Project C	OPEN	Relatorio 3 Relatorio 8	Rafael,	Alex,	Ricardo,	IST	client@ist.pt			
Project A	INPROGRESS	Relatorio 1	Pilar,	Paulo,	Alex,	BNP	client@bnp.pt			

Items per page: 10 | 1 - 3 of 3 | < > >> <<

Figure 4.3: Report Repository.

Report Repository

As the User initiates a session he is presented with a list of Projects he is assigned to. These Projects can be filtered with the search bar on the top, and can be sorted by clicking one of its features. Searching and sorting capabilities are present in all other tabs as well.

From the Projects tab, any User can create a new Project. However, only a Project Manager can edit the meta information of a Project or delete it. This is demonstrated by the available buttons on the right. Alex only is the Project Manager in Project A.

When the "Create" Button is pressed, a dialog, Figure 4.4, will appear. Then, the User fills up the information accordingly. This is helped by the fact that, apart from the Project name, all fields only show information that respect the rules imposed by MAINSEC. Only a predefined status can be select, and the possible options in the Client, Auditor, Reviewer and Project Manager fields are entities that already exist in the connected databases. The same dialog is used to edit the Project information after it is created. To open a Report for viewing or editing, a User double-clicks the name of a given Report.

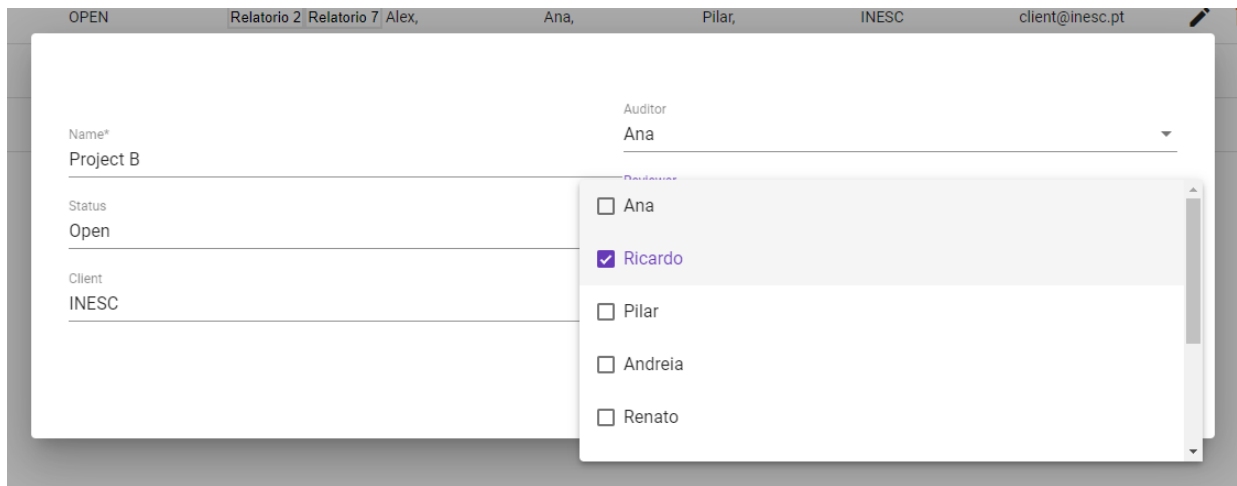


Figure 4.4: Create a new Project. Selecting a Reviewer from the existing Users in the database.

In the Reports tab, once again, a User only gets to see the Reports belonging to a Project he belongs to. Any User can create, edit or remove a Report inside a Project he is assigned to. Reports can also be opened in the Reports tab. Creating and editing a Report uses the same style of form in Figure 4.4.

The Clients and Users tabs show a list of all the Clients and Users to every User, as these function mainly as a contact book. Only the Project Managers can edit or delete elements of this list, but all Users can add Clients and Users as needed.

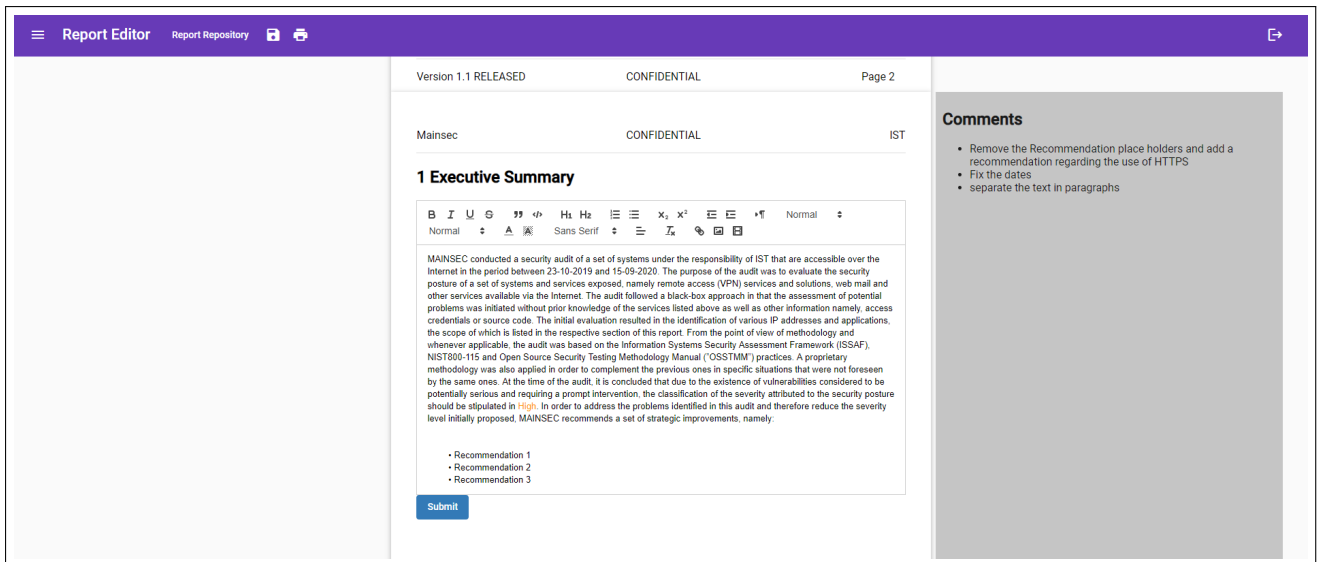


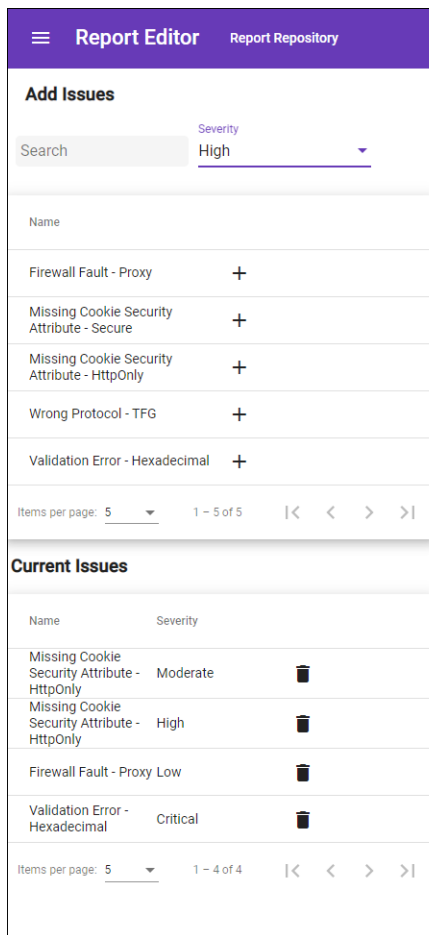
Figure 4.5: Editing content in a section using a rich text editor.

Report Editor

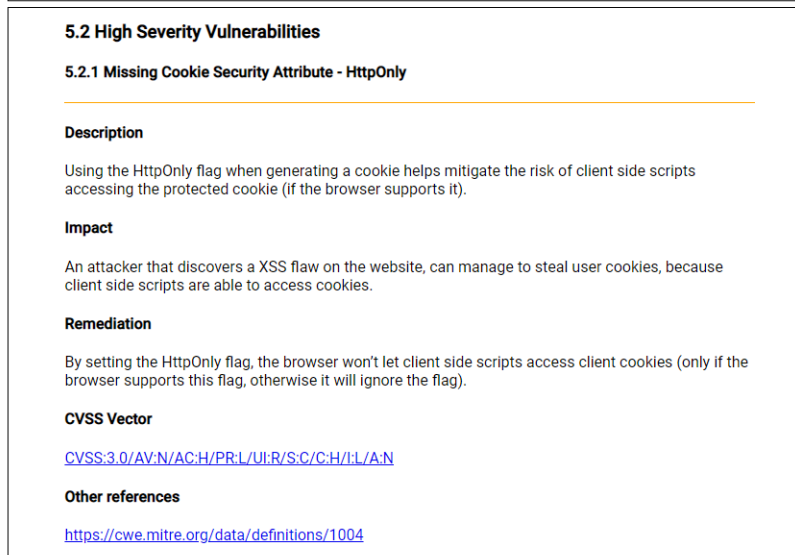
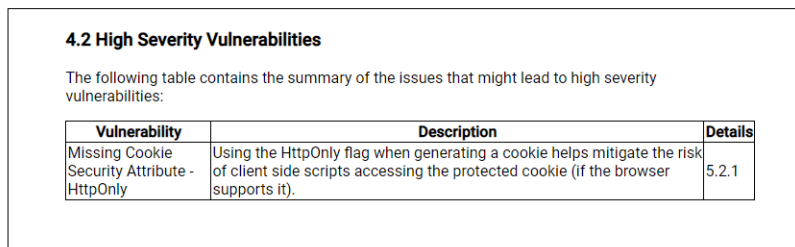
After the User selects a Report to open in the Report Repository, the interface changes to a WYSIWYG paradigm. A User can scroll to see how the Report will look like when exported as PDF, and double-click in a page to be edited.

As a target page is double-clicked, the correct tool to edit the present contents will present itself. Pages are edited either by filling form, much like in Figure 4.4, or as a rich text editor, shown in Figure 4.5. In our application, we chose Quill [28] as our text editor module, as it is easy to use and implement. Additionally, Quill can save the edited text, internally, as HTML. The HTML content produced by Quill has the advantage of being lightweight and fast to store in String format inside our database, and later on be read and reproduced to the interface just as easily.

On the right side of Figure 4.5, there is the comment section. This section shows a different behavior depending on the team member in session. The Auditor and Project Manager can only read the comments left by the Reviewer, while the Reviewer interacts with a text editor, similar to the one seen in Figure 4.5, to submit his comments on that page. This distinction is possible, because when the User opens a Report the application is aware of the User role in the respective Project.



(a)



(b)

Figure 4.6: Add an Issue to the Report. (a) - Issues sidebar. (b) - Issue automatically placed in the correct sections.

Add Issue to the Report

To manage the Issues in the Report, the Auditor must open the hidden sidebar by clicking the menu button on the top left corner. In the first table in the sidebar, shown in Figure 4.6 - (a), Issues from the Knowledge Base (section 3.2.4) can be searched and sorted. In the table below, the Auditor can remove Issues already in the Report by clicking in the trash can button of the target Issue.

To add an Issue to the document, one must first select a Severity from the selection element next to the search box. Then, after clicking the plus sign in the desired Issue, all the static data of that Issue will be automatically imported from the database to the Report.

Figure 4.6 - (b) shows the results of adding an Issue to the Report. First, a summary of the Issue is inserted in a table of the select severity section, including an indication to the full description in the next chapter. Second, a detailed definition of the Issue in the Assessment Details section.

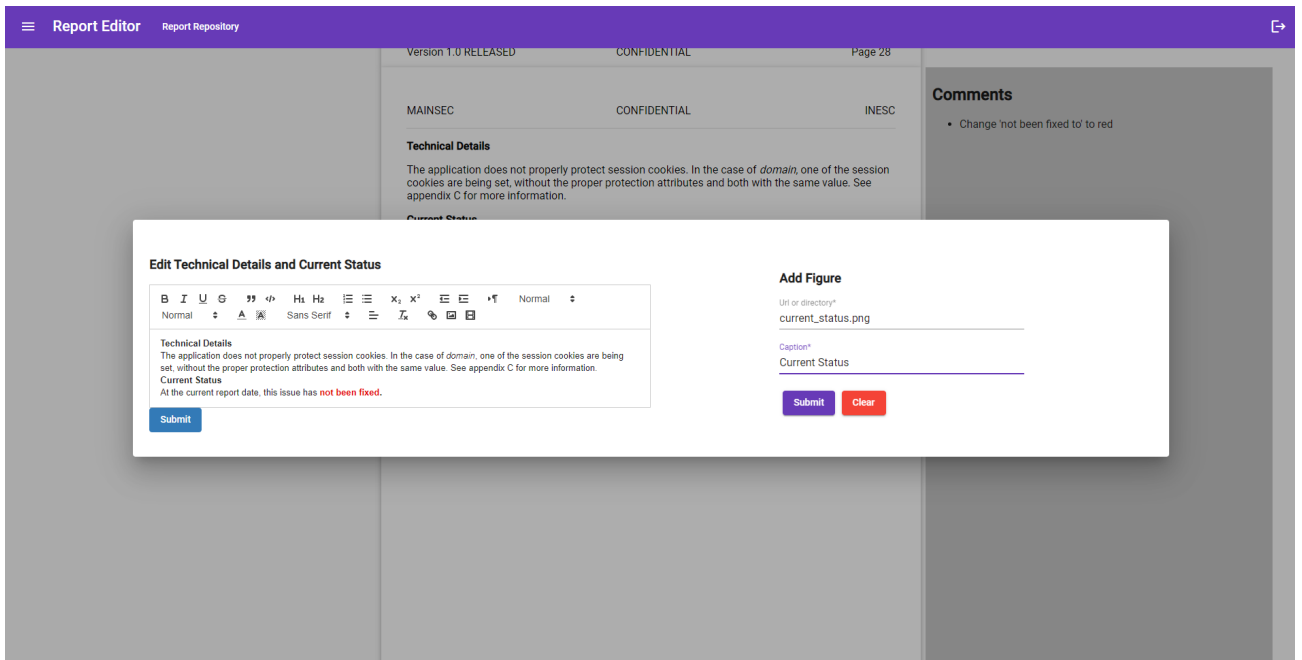


Figure 4.7: Editing an Issue and adding an image to the Issue.

Editing an Issue

After an Issue is inserted into the document, the Auditor has to complete it with the Technical Details and Current Status paragraphs. To do this, the writer double-clicks the recently created page with just the two titles of the paragraphs. In the dialog, Figure 4.7, the paragraphs can be written in the left text editor and images can be added through the form on the right.

Chapter 5

Evaluation

To test the usability of the system proposed in this document, we compared it to the workflow in place at MAINSEC. In this test, we have the volunteers execute a set of tasks in a Google Docs document and in our solution.

This chapter describes the study that took place to evaluate both solutions.

5.1 Participants

The tests were done by eight volunteers, six men and two women. All of our participants work in IT companies as developers and/or managers and everyone has experience in creating report documents. We chose Google Docs as the target of our comparison, because it is a very popular service. This was proven to us as all the participants had previous experience with this software, even if it was a small amount.

5.2 Material

We provided the testers with an instructions document and auxiliary document containing all the necessary data to insert an Issue in the Report.

The participants did the tasks in a desktop computer with two monitors. Each user had the solution being tested in one monitor, and the instructions and the auxiliary document in the other. We also used stopwatch to measure the time a tester took to execute all the tasks.

Task	Description
T1	Edit the document cover by changing two words and a date, in the correct format.
T2	Edit the headers by changing two words.
T3	Add three names to the Auditors and Reviewers table.
T4	Add a new row to the Document Management table and fill it with 3 words and a date, in the correct format.
T5	Add an Issue to the document using the provided Issue information and place it in two sections, each one with its own formatting.
T6	Edit the dynamic data of the added Issue by writing two paragraphs and adding a captioned image.
T7	Export the document as a PDF file.

Table 5.1: List of usability testing tasks.

5.3 Tasks

The task had a user open a report already filled with data, edit some of its contents and add new information to the document. They were designed to represent all the types of interactions a user would have with a full document, this includes editing text, editing tables, adding images, etc.

Each user performed the same seven tasks, detailed in Table 5.1, on both applications. Half of the users started with Google Docs and the other half started with SAAR2020, so that the results would not be affected by the user learning the tasks.

5.4 Procedure

The sessions had an approximate duration of 30 minutes and were done in a calm and quiet room with a desktop computer. Each session was initiated with an explanation of how the tests would take place and what they would be doing.

First, we explained the window layout in the monitors. The first monitor had the interface to be tested and the second one had the instructions for the tasks, as well as an auxiliary document containing the information for the Issue to be added to the Report.

As everybody had previous experience with Google Docs, and the instructions given provided all the necessary knowledge to use SAAR2020, no time was invested in learning the interfaces.

We briefly explained what the testers would be doing, without telling them any details on how to do it and informed them that there were two questionnaires at the end of the tasks.

Finally, we explained that their results would be kept anonymous. Any further questions were welcome to be posed during the test.

Then the testers started performing the asked tasks by following the instructions given and we started taking note of the time taken with our stopwatch.

Listed below, we explain in detail the differences on how the tasks are completed depending on the application used:


Task	Application	Details
T1	Google Docs	Write the new words in the correct place and detect the right date format to conform to it.
	SAAR2020	Replace the words in the correct fields of the pop-up form and select a date using a calendar interface. When the form is submitted, the words are placed in the rights spots and the date is formatted automatically.
T2	Google Docs	Write the new words in the correct place.
	SAAR2020	Do nothing, this information is updated when the user changes the cover.
T3	Google Docs	Go to the correct table and write the content given, in the right place.
	SAAR2020	Do nothing, this data is already filled as the Report has information about the team members working on it.
T4	Google Docs	Right-click to add a new row to the target table. Manually add all the contents, leaving room to mistakes in the remark and date entries.
	SAAR2020	Double-click to add a new row to the target table. Manually add the new version and editor, select a remark from a list of set options and add the date using the calendar interface. The data will have the correct format and the remarks can not be wrongly written as it was selected from a fixed list.
T5	Google Docs	Write the static information to two sections. In the first static information section, write a paragraph and add a table. Fill that table with a line and three columns with information from the auxiliary document. In the second static information section, write a title, three paragraphs and two hyperlinks. Add all of this content in the right severity sections and with the correct formatting and order.
	SAAR2020	Open the Issues sidebar on the left of the interface, set the severity of the Issue, search the desired Issue to be added and click the + button next to that Issue. The contents of that Issue will be added to both of the correct sections automatically, respecting the formatting rules.
T6	Google Docs	In the second section the Issue was inserted, the Assessment Details section, write the to dynamic paragraphs and add a captioned image.
	SAAR2020	In the second section the Issue was inserted, the Assessment Details section, double-click the Technical Details page and a pop-up will appear. Write the two dynamic paragraphs in the text processor one the left and add the URL and caption of the image to be added on the right. Then, submit the filled form. The content will be correctly placed and formatted in the document.
T7	Google Docs	In the toolbar, on the top, click the sequence File->Download->PDF Document (.pdf).
	SAAR2020	In the toolbar, on the top, click the  button.

Table 5.2: List of detailed usability testing tasks.

At the end of the task execution on both solutions, we asked the participants to rate each task on each solution using a Likert scale (1=not at all, 10=very) regarding how fast and how easy it was to do the task and how useful the interface was in helping the user perform the task.

5.5 Results

In this section, we present the results of the comparison between the two solutions. The results obtained from the two solutions are described by the metric and compared with each other.

5.5.1 Time taken to execute tasks

The time a tester took to execute the tasks was measured from the first click on the interface to the end of the last task, clicking the save button to export the file as PDF. This time was measured in minutes, and then converted to seconds to be analysed.

In Figure 5.1, on the left vertical axis, we show the time each user took to perform the tasks in each solution, in seconds. On the right vertical axis, we can see the ratio between the time taken in SAAR2020 and Google Drive.

We saw some difference in the duration of test completion amongst users. Having the fastest participant taking 206 seconds in SAAR2020 and the slowest taking 304 seconds. In Google Drive, the fastest time was 524 seconds and the slowest finished in 795 seconds. This happens as different participants have different levels of experience with the technologies used during the tests.

Even though the testers took more or less time performing tasks, we could verify a consistency in the ratio between solutions. With SAAR2020 times being, on average, 2.59 times faster than Google Drive. The User registering the smallest difference at 2.46 speed improvement and the biggest at 2.73 times.

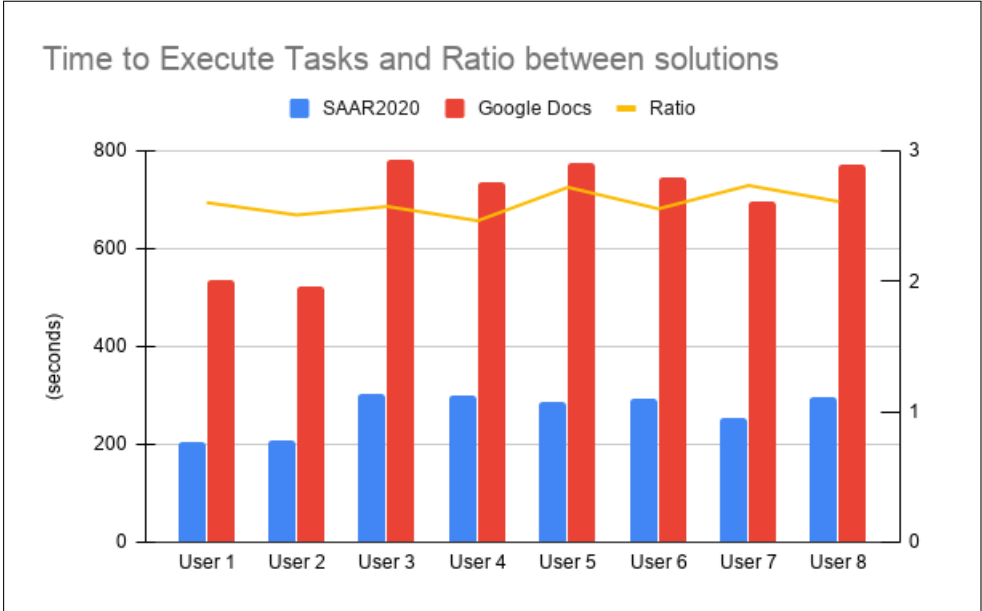


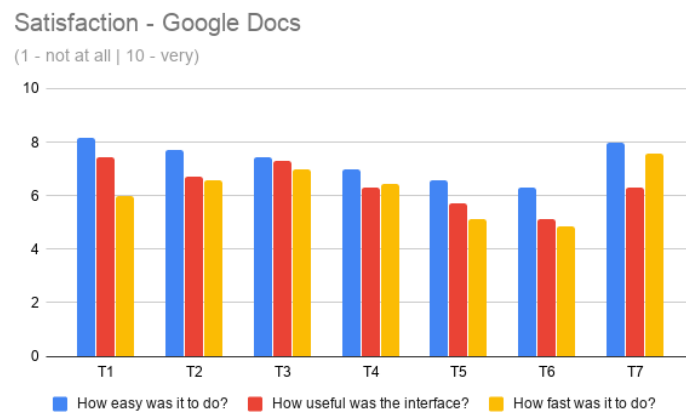
Figure 5.1: Time each User took to execute the tasks in both solutions and Ratio.

5.5.2 Satisfaction

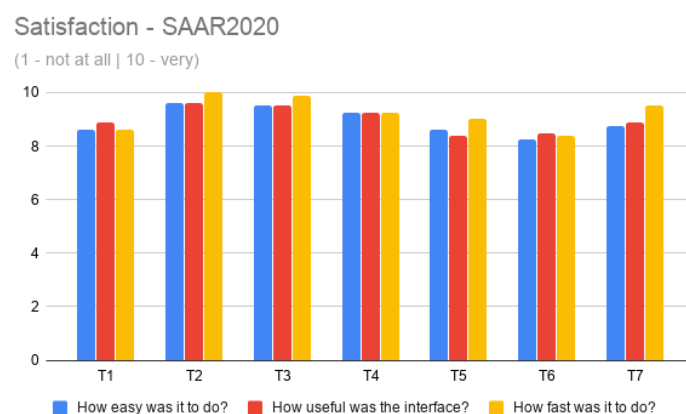
In Figure 5.2, we show the average score attributed in the three questions posed about each task. These scores were obtained by having the testers answer a questionnaire for each solution. These questionnaires were answered using a Likert scale. Every Likert scale has ten discrete, from 1 to 10 points. Score 1 represents, for example, "the task was not at all easy to do", and score 10 represents the reverse, "the task was very easy to do".

Looking at the scores for particular tasks, we can clearly see that the tasks that required the most effort were T5 and T6. These are the tasks that, in both solutions, require the most user interaction with the applications. So these tasks received the lowest scores.

Overall, we noticed an improved or constant score from Google Drive to SAAR2020. This happens, because at every step our application tries to reduce the input required from a user. This is especially noticeable in T2 and T3, where SAAR2020 got almost perfect scores, because both these steps are done automatically by the application itself in the background.



(a) Google Docs



(b) SAAR2020

Figure 5.2: Analysis of the time taken the execute the tasks.

We also took notice of the satisfaction ratio between the two solutions, in Figure 5.3. To do this, we gathered the previously measured satisfaction scores and divided the scores from SAAR200 by the scores from Google Docs. This graph shows how much the user felt the experience was improved per task.

The difficulty was the least affected, peaking at a ratio of 1.32 and being as low as 1.05. This happens because the user, ultimately, still has to do something, even if the task is smaller or simpler.

The usefulness improved in a greater manner, having top improvements of 1.47 and 1.65. These tasks were, respectively, T4, adding a version to the Report, and T6, editing an Issue inside the document.

Finally, the best results came from the enhancement in time taken to fulfill the tasks. Tasks T6 and T7, were, accordingly, 1.75 and 1.72 times faster in SAAR2020 than they were in Goggle Drive.

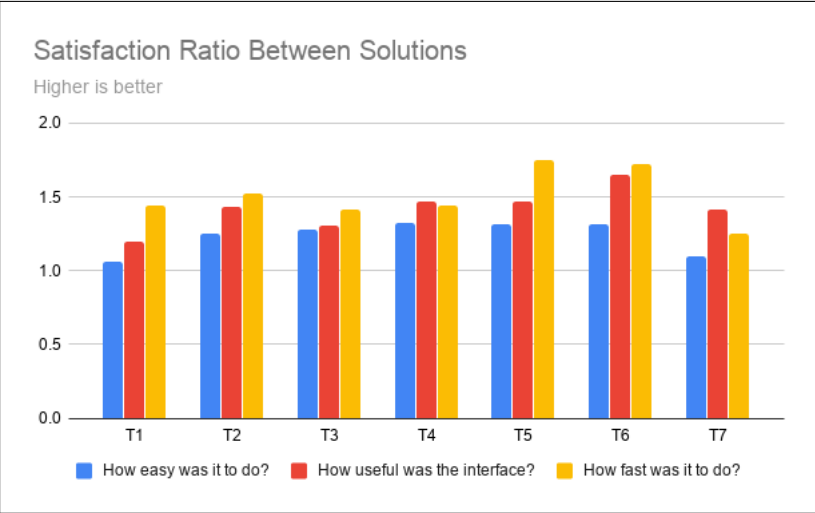


Figure 5.3: Satisfaction Ratio Between Solutions.

Below, in Figure 5.4, we averaged the scores of all the tasks show in Figure 5.3 to prove that the conclusions reached in the previous paragraph can also be applied when looking at the bigger picture. Then we calculated the difference between the solutions showing, for example, that testers gave on average 3 more points to the question "How fast as it to do?" when using SAAR2020.

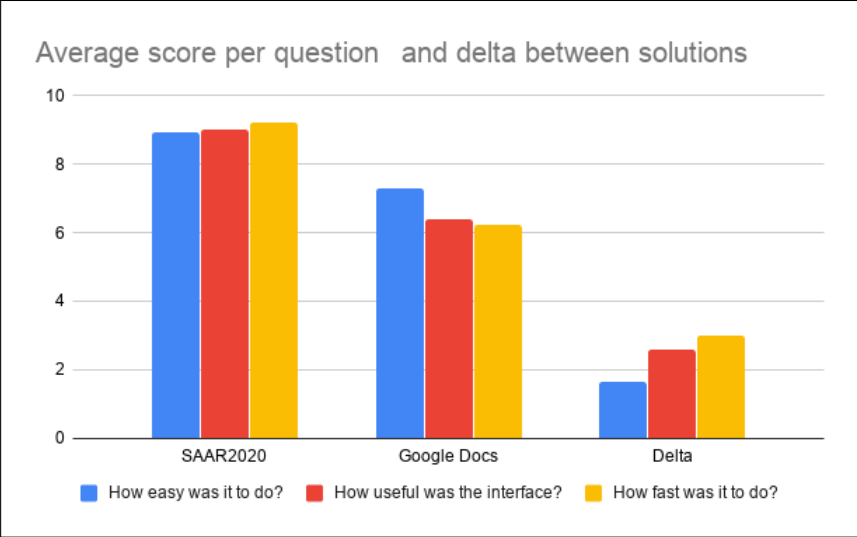


Figure 5.4: Average score per question and the difference between those averages.

Chapter 6

Final Remarks

Our work began by examining the difficulties experienced when creating and managing projects and reports That showed a need for an platform that could better organize the team members involved in this endeavour. We also found that there was as need and possibility for automation of the processes at hand.

We continued by reviewing the current state-of-the art in report writing, generic document management across users and what tools we could use for developing our proposed solution in a web environment. We discussed their strengths and shortcomings and used that knowledge to make both design and structural decisions that helped shape our solution.

We complemented the definition of our problem by thoroughly describing the requirements proposed to us by the team at MAINSEC.

This document introduced SAAR2020, a new service that aims to satisfy the core challenges presented. We explained what strategies and tools were used to implement the various components of our solution. There was a demonstration of what we built and how to use our application to reach the goal of better Report creation.

SAAR2020 was tested and evaluated by testers, simulating a real world interaction with the software and taking notice of how our solution compares to an existing option. With this tests, we proved that there was definitive improvement regarding usability and, most importantly, time investment from the users.

Then, we can conclude that there were, in fact, parts of creating a Report that could be automated and new management features that could help the team members in their daily workflow.

6.1 Future Work

During the development and evaluation of the prototype, we noted some possible suggestions that could be explored to further improve our solution.

Versioning. Although our client application is constantly saving the progress of the Auditor production, there is no way to go back after a submission, if a mistake is made. If the Auditor accidentally delete a whole section, he will have to rewrite it completely. If every version created a separate document, a User could go back to a previous stage.

Drag-and-drop Images. It is possible to add images to an Issue, but it has to be done through writing a URL or file directory. However, a drag-and-drop system is much more intuitive and fast, but this would mean having to store images in our databases, making our Report Document files much larger.

Preview Issue before adding it to the document. As the application stands, the Auditor adds an Issue by selecting it based on its title. This implies that the user knows exactly what Issues to add. By providing a preview of the contents of the Issue, the Auditor could double-check if that really is the correct Issue to add to the document.

Continue expanding automation. With more experience with creating Reports, more patterns could be uncovered and more automation introduced. For example, the “Executive Summary” is a dynamic text that has to be written by the Auditor, but parts of it must always be present and it respects a certain order. So, there is a possibility for pre-writing paragraphs that would only have to be completed by the editor.

Bibliography

- [1] Definition of REPORT. *www.merriam-webster.com*. URL <https://www.merriam-webster.com/dictionary/report>.
- [2] P. Madan. *Language proficiency in English*. Agarwal publication, 28/115, jyoti block, sanjay place, Agra-2, 2016. ISBN 9789385872280.
- [3] SAP SE. SAP Crystal Reports reporting and analytics solution, 2018. URL <https://www.crystalreports.com/>.
- [4] SAP SE. Analytics for the small to midsize business with SAP Crystal solutions SAP Solution Brief SAP Solutions for Small, Midsize Businesses. Technical report, 2018. URL <https://www.sapstore.com//medias/SAP-Crystal-Solutions-summary.pdf?context=bWFzdGVyfHBkZnN8MTA2NDc5M3xhcHBsaWNhdGlvbi9wZGZ8cGRmcy9oZWVvaGIyLzkwMTE2NDEwOTAwNzgucGRmfDRkZmNj>
- [5] TIBCO Software Inc. Jaspersoft® Studio — Jaspersoft Community, 2018. URL <https://community.jaspersoft.com/project/jaspersoft-studio>.
- [6] Write papers like a modern scientist (use Overleaf or Google Docs + Paperpile). *Simply Statistics blog*. URL <https://simplystatistics.org/2016/04/21/writing>.
- [7] Overleaf - About us. *Overleaf*. URL <https://www.overleaf.com/about>.
- [8] Introduction to LaTeX, . URL <https://www.latex-project.org/about/>.
- [9] Cloud Storage for Work and Home - Google Drive. URL <https://www.google.com/drive/>.
- [10] The PostgreSQL Global Development Group. PostgreSQL: About. URL <https://www.postgresql.org/about/>.
- [11] I. MongoDB. What Is MongoDB? — MongoDB, 2018. URL <https://www.mongodb.com/what-is-mongodb>.
- [12] Facebook Inc. Introduction to GraphQL — GraphQL, 2019. URL <https://graphql.org/learn/>.
- [13] Google. Angular - What is Angular?, 2019. URL <https://angular.io/docs>.
- [14] Ilya Bodrov-Krukowski. Angular Introduction: What It Is, and Why You Should Use It — SitePoint, 2018. URL <https://www.sitepoint.com/angular-introduction/>.

- [15] Guide: What is Vue.js? *Vue.js*. URL <https://vuejs.org/v2/guide/index.html#What-is-Vue-js>.
- [16] Introduction — Vue.js. *vuejs.org*, . URL <https://vuejs.org/v2/guide/>.
- [17] Comparison with Other Frameworks — Vue.js. URL <https://vuejs.org/v2/guide/comparison.html#Angular-Formerly-known-as-Angular-2>.
- [18] J. Donn Le Vie. Writing Software Requirements Specifications (SRS) — TechWhirl, 2010. URL <https://techwhirl.com/writing-software-requirements-specifications/>.
- [19] William M. Wilson. Writing Effective Natural Language Requirements Specifications. Technical report, 1999. URL <https://profinit.eu/wp-content/uploads/2016/03/WritingEffectiveSRS.pdf>.
- [20] apollo-server-express - npm. URL <https://www.npmjs.com/package/apollo-server-express>.
- [21] About — Node.js. URL <https://nodejs.org/en/about/>.
- [22] Express - Node.js web application framework. URL <https://expressjs.com/>.
- [23] Sequelize — Sequelize ORM. URL <https://sequelize.org/>.
- [24] Mongoose ODM v5.10.5. URL <https://mongoosejs.com/>.
- [25] Association — Sequelize. URL <https://sequelize.org/master/class/lib/associations/base.js~Association.html>.
- [26] GraphQL Code Generator — GraphQL Code Generator. URL <https://graphql-code-generator.com/>.
- [27] Angular Material UI component library. URL <https://material.angular.io/>.
- [28] Quill - Your powerful rich text editor. URL <https://quilljs.com/>.

Appendix A

Report Document from SAAR202

Example of a document produced by SAAR2020.

Company
LOGO

**External Penetration Test
Technical Report**

Prepared by Mainsec for IST
CONFIDENTIAL
1.1
RELEASED
September 13, 2020

Contents

1	Executive Summary	3
2	Introduction	4
	2.1 Document Information	4
	2.2 Document Structure	6
	2.3 Disclaimer	7
3	Assessment Information	8
	3.1 Assessment Scope	8
	3.2 Organizational and Technical Contacts	8
	3.3 Constraints	9
	3.4 Procedures After the Assessment	10
4	Summary of Assessment Results	11
	4.1 Critical Severity Vulnerabilities	20
	4.2 High Severity Vulnerabilities	21
	4.3 Moderate Severity Vulnerabilities	22
	4.4 Low Severity Vulnerabilities	23
	4.5 Minor Severity Vulnerabilities	24

5 Assessment Details	25
5.1 Critical Severity Vulnerabilities	26
5.1.1 Missing Cookie Security Attribute - Secure	26
5.2 High Severity Vulnerabilities	29
5.2.2 Missing Cookie Security Attribute - HttpOnly	29
5.3 Moderate Severity Vulnerabilities	33
5.4 Low Severity Vulnerabilities	34
5.5 Minor Severity Vulnerabilities	35
6 Appendix	36

1 Executive Summary

MAINSEC conducted a security audit of a set of systems under the responsibility of IST that are accessible over the Internet in the period between 23-10-2019 and 15-09-2020. The purpose of the audit was to evaluate the security posture of a set of systems and services exposed, namely remote access (VPN) services and solutions, web mail and other services available via the Internet. The audit followed a black-box approach in that the assessment of potential problems was initiated without prior knowledge of the services listed above as well as other information namely, access credentials or source code. The initial evaluation resulted in the identification of various IP addresses and applications, the scope of which is listed in the respective section of this report. From the point of view of methodology and whenever applicable, the audit was based on the Information Systems Security Assessment Framework (ISSAF), NIST800-115 and Open Source Security Testing Methodology Manual ("OSSTMM") practices. A proprietary methodology was also applied in order to complement the previous ones in specific situations that were not foreseen by the same ones. At the time of the audit, it is concluded that due to the existence of vulnerabilities considered to be potentially serious and requiring a prompt intervention, the classification of the severity attributed to the security posture should be stipulated in **High**. In order to address the problems identified in this audit and therefore reduce the severity level initially proposed, MAINSEC recommends a set of strategic improvements, namely:

- Recommendation 1
- Recommendation 2
- Recommendation 3

2 Introduction

2.1 Document Information

This document was issued by Mainsec on September 13, 2020. © Mainsec 2020

Responsibility Statement

Documents classified as "INTERNAL ONLY", "PRIVATE" or "CONFIDENTIAL" contain information that is not public domain and should not be available to the general public. Only the involved parties in the current project are authorized to hold this document. The document owner should not reveal, duplicate or provide any information contained in this document to third parties without explicit allowance from Mainsec.

Classification

CONFIDENTIAL

Document Owner

IST

Authors and Reviewers

	Name
Author(s)	Pilar
Approver(s)	Paulo
Reviewer(s)	Alex

Document Management

Version	Date	Editor	Remarks
1.0	2018-11-05	Some Audior	DRAFT
1.1	2020-09-13	Pilar	RELEASED

2.2 Document Structure

The remainder of this document is organized into three major sections:

- Section 3 provides the collected information to support the security assessment, including the assessment scope, testing accounts, roles, responsibilities and contacts, technical issues faced during the assessment and procedures to be considered after the assessment.
- Section 4 presents the summary of the assessment results, including the issue severity, title, brief description and its impact and a link to the technical details.
- Section 5 explains the technical details of the identified issues and the reproduction steps to replicate the vulnerabilities, provides examples to support evidence and presents a remediation approach.

This document also contains the following appendices:

- Appendix A contains the remaining evidences.
- Appendix B features a list of the tools used in the security assessment.

2.3 Disclaimer

Please consider the following information regarding security assessments:

Security assessments are usually conducted within a limited time frame. While all the effort is put through finding the maximum number of issues, it is impossible due to the nature of security assessments to guarantee that the assessed assets are free from all and further vulnerabilities.

In the case of a re-evaluation assessment, the focus of the assessment is verifying if the proposed remediation measures are properly implemented. However, as in many instances, assets are subjected to regular changes, new issues might be revealed during the re-evaluation assessment.

The proposed remediation measures are based on following good practices, however under certain conditions might negatively impact the affected assets, or be not applicable at all. A risk analysis should be performed to understand whether an implementation is beneficial and cost effective.

All the described issues in this report are verified by experienced and qualified security professionals, unless stated otherwise.

3 Assessment Information

3.1 Assessment Scope

Execution period	2020-08-12 - 2020-08-13
Asset Name(s)	External Assets
Asset(s) Description	Internet Presence
Asset Address(es)/URL(s)	List of IPS

3.2 Organizational and Technical Contacts

	Name	Contact
Security Tester (Auditor)	Pilar	pilar@saar2020.com
Security Tester (Reviewer)	Paulo	paulo@saar2020.com
Security Tester (Project Manager)	Alex	alex@saar2020.com
Customer	BNP	client@bnp.pt

3.3 Constraints

During the security assessment execution period, no major constraints were experienced.

3.4 Procedures After the Assessment

When preparing a security assessment, testing accounts are often provisioned with different rights and privileges and firewall rules are modified or temporarily disabled, etc.

! NOTE: If testing accounts were provisioned, they must be removed and their privileges revoked immediately and completely as such accounts are not required after the security assessment and might pose a security risk.

! NOTE: If firewall rules were temporarily modified or disabled, they must be restored to their initial configuration to avoid leaving the protected assets exposed after the security assessment.

During the security assessment, the behavior of the security testers might be intentionally intrusive, for instance, in an attempt to gain access to an asset, temporary files might be created on the target asset, database records can be modified whenever that does not affect the asset overall integrity or code can be inserted on the asset that allows remotely accessing it.

! NOTE: To ensure that the security assessment does not have further negative impact on the development, deployments or regular daily operations, assets must be cleaned or whenever possible, reverted to their original state.

4 Summary of Assessment Results

This section presents a summary of the assessment results.

To each issue, a severity level is assigned. The severity level is based on the impact and the exploitation likelihood. The assigned levels should be reviewed by the asset owner based on their judgment and knowledge of business context. Adding business context might change the initially assigned severity level.

The following severity levels are used to rate the identified issues during the security assessment:

Critical

An issue is rated critical severity if it leads to a vulnerability that when exploited results in a full loss of the data confidentiality, integrity and availability and if an attack can be easily conducted.

High

An issue is rated high severity if it leads to a vulnerability that when exploited results in a partial loss of the data confidentiality, integrity or availability and if an attack can be easily conducted.

Moderate

An issue is rated moderate severity if it leads to a vulnerability that when exploited results in a partial loss of the data confidentiality, integrity or availability, but it can only be exploited by skilled attackers and chained with other vulnerabilities.

Low

An issue is rated low severity if it leads to a vulnerability that when exploited results in a partial loss of the data confidentiality, integrity or availability, but it can only be exploited by very skilled attackers, when strictly chained with other vulnerabilities and only under certain circumstances.

Minor

A minor should not lead to vulnerabilities, however it might lead to a problem under certain conditions.

CVSS 3.0

The Common Vulnerability Scoring System (CVSS) provides a way to capture the principal characteristics of a vulnerability, and produce a numerical score reflecting its severity, as well as a textual representation of that score. The numerical score can then be translated into a qualitative representation (such as low, medium, high, and critical) to help organizations properly assess and prioritize their vulnerability management processes.

CVSS 3.0 Metrics

Attack Vector (AV)

This metric reflects the context by which vulnerability exploitation is possible. This metric value (and consequently the Base score) will be larger the more remote (logically, and physically) an attacker can be in order to exploit the vulnerable component. The assumption is that the number of potential attackers for a vulnerability that could be exploited from across the Internet is larger than the number of potential attackers that could exploit a vulnerability requiring physical access to a device, and therefore warrants a greater score.

Network (N)	A vulnerability exploitable with network access means the vulnerable component is bound to the network stack and the attacker's path is through OSI layer 3 (the network layer). Such a vulnerability is often termed "remotely exploitable" and can be thought of as an attack being exploitable one or more network hops away (e.g. across layer 3 boundaries from routers).
Adjacent (A)	A vulnerability exploitable with Adjacent network access means the vulnerable component is bound to the network stack, however the attack is limited to the same shared physical (e.g. Bluetooth, IEEE 802.11), or logical (e.g. local IP subnet) network, and cannot be performed across an OSI layer 3 boundary (e.g. a router).
Local (L)	A vulnerability exploitable with Local access means that the vulnerable component is not bound to the network stack, and the attacker's path is via read/write/execute capabilities. In some cases, the attacker may be logged in locally in order to exploit the vulnerability, otherwise, she may rely on User Interaction to execute a malicious file.
Physical (P)	A vulnerability exploitable with Physical access requires the attacker to physically touch or manipulate the vulnerable component. Physical interaction may be brief or persistent.

Attack Complexity (AC)

This metric describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. As described below, such conditions may require the collection of more information about the target, the presence of certain system configuration settings, or computational exceptions. Importantly, the assessment of this metric excludes any requirements for user interaction in order to exploit the vulnerability. This metric value is largest for the least complex attacks.

High (H)	A successful attack depends on conditions beyond the attacker's control. That is, a successful attack cannot be accomplished at will, but requires the attacker to invest in some measurable amount of effort in preparation or execution against the vulnerable component before a successful attack can be expected.
Low (L)	Specialized access conditions or extenuating circumstances do not exist. An attacker can expect repeatable success against the vulnerable component.

Privileges Required (PR)

This metric describes the level of privileges an attacker must possess before successfully exploiting the vulnerability. This metric is greatest if no privileges are required.

None (N)	The attacker is unauthorized prior to attack, and therefore does not require any access to settings or files to carry out an attack.
High (H)	The attacker is authorized with (i.e. requires) privileges that provide significant (e.g. administrative) control over the vulnerable component that could affect component-wide settings and files.
Low (L)	The attacker is authorized with (i.e. requires) privileges that provide basic user capabilities that could normally affect only settings and files owned by a user. Alternatively, an attacker with Low privileges may have the ability to cause an impact only to non-sensitive resources.

User Interaction (UI)

This metric captures the requirement for a user, other than the attacker, to participate in the successful compromise of the vulnerable component. This metric determines whether the vulnerability can be exploited solely at the will of the attacker, or whether a separate user (or user-initiated process) must participate in some manner. This metric value is greatest when no user interaction is required.

None (N)	The vulnerable system can be exploited without interaction from any user.
Required (R)	Successful exploitation of this vulnerability requires a user to take some action before the vulnerability can be exploited.

Scope (S)

Formally, Scope refers to the collection of privileges defined by a computing authority (e.g. an application, an operating system, or a sandbox environment) when granting access to computing resources (e.g. files, CPU, memory, etc). These privileges are assigned based on some method of identification and authorization. When the vulnerability of a software component governed by one authorization scope is able to affect resources governed by another authorization scope, a Scope change has occurred.

Changed (C)	An exploited vulnerability can affect resources beyond the authorization privileges intended by the vulnerable component. In this case the vulnerable component and the impacted component are different.
Unchanged (U)	An exploited vulnerability can only affect resources managed by the same authority. In this case the vulnerable component and the impacted component are the same.

Confidentiality (C)

This metric measures the impact to the confidentiality of the information resources managed by a software component due to a successfully exploited vulnerability. Confidentiality refers to limiting information access and disclosure to only authorized users, as well as preventing access by, or disclosure to, unauthorized ones. This metric value increases with the degree of loss to the impacted component.

High (H)	There is total loss of confidentiality, resulting in all resources within the impacted component being divulged to the attacker. Alternatively, access to only some restricted information is obtained, but the disclosed information presents a direct, serious impact.
Low (L)	There is some loss of confidentiality. Access to some restricted information is obtained, but the attacker does not have control over what information is obtained, or the amount or kind of loss is constrained. The information disclosure does not cause a direct, serious loss to the impacted component.
None (N)	There is no loss of confidentiality within the impacted component.

Integrity (I)

This metric measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of information. This metric value increases with the consequence to the impacted component.

High (H)	There is a total loss of integrity, or a complete loss of protection. Alternatively, only some files can be modified, but malicious modification would present a direct, serious consequence to the impacted component.
Low (L)	Modification of data is possible, but the attacker does not have control over the consequence of a modification, or the amount of modification is constrained. The data modification does not have a direct, serious impact on the impacted component.
None (N)	This metric measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. While the Confidentiality and Integrity impact metrics apply to the loss of confidentiality or integrity of data(e.g., information, files) used by the impacted compo

Availability (A)

This metric measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. While the Confidentiality and Integrity impact metrics apply to the loss of confidentiality or integrity of data (e.g., information, files) used by the impacted component, this metric refers to the loss of availability of the impacted component itself, such as a networked service (e.g., web, database, email). Since availability refers to the accessibility of information resources, attacks that consume network bandwidth, processor cycles, or disk space all impact the availability of an impacted component. This metric value increases with the consequence to the impacted component.

High (H)	There is total loss of availability, resulting in the attacker being able to fully deny access to resources in the impacted component; this loss is either sustained (while the attacker continues to deliver the attack) or persistent (the condition persists even after the attack has completed). Alternatively, the attacker has the ability to deny some availability, but the loss of availability presents a direct, serious consequence to the impacted component.
None (N)	There is no impact to availability within the impacted component.
Low (L)	There is reduced performance or interruptions in resource availability. Even if repeated exploitation of the vulnerability is possible, the attacker does not have the ability to completely deny service to legitimate users. The resources in the impacted component are either partially available all of the time, or fully available only some of the time, but overall there is no direct, serious consequence to the impacted component.

4.1 Critical Severity Vulnerabilities

The following table contains the summary of the issues that might lead to critical severity vulnerabilities:

Vulnerability	Description	Details
Missing Cookie Security Attribute - Secure	The secure flag is an option that can be set by the application server when sending a new cookie to the user within an HTTP Response. The purpose of the secure flag is to prevent cookies from being observed by unauthorized parties due to the transmission of the cookie in clear text.	5.1.1

4.2 High Severity Vulnerabilities

The following table contains the summary of the issues that might lead to high severity vulnerabilities:

Vulnerability	Description	Details
Missing Cookie Security Attribute - HttpOnly	Using the HttpOnly flag when generating a cookie helps mitigate the risk of client side scripts accessing the protected cookie (if the browser supports it).	5.2.1

4.3 Moderate Severity Vulnerabilities

No issues were identified and rated as moderate severity.

4.4 Low Severity Vulnerabilities

No issues were identified and rated as low severity.

4.5 Minor Severity Vulnerabilities

No issues were identified and rated as minor severity.

5 Assessment Details

This section explains the technical details of the found issues during the security assessment.

The details include the issue severity, a brief description, the impact that it might create as well as the proposed remediation measures. Lastly, the technical details should provide guidance to the reproduction of the issue in a safe manner as well support evidence for the existing issues.

5.1 Critical Severity Vulnerabilities

5.1.1 Missing Cookie Security Attribute - Secure

Description

The secure flag is an option that can be set by the application server when sending a new cookie to the user within an HTTP Response. The purpose of the secure flag is to prevent cookies from being observed by unauthorized parties due to the transmission of the cookie in clear text.

Impact

An attacker that intercepts the request can be able to view the cookie session id in clear text, and use it to impersonate that user on further communications with the application.

Remediation

By setting the secure flag, the browser will prevent the transmission of a cookie over an unencrypted channel.

CVSS Vector

[CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:L/A:N](#)

Other references

<https://cwe.mitre.org/data/definitions/614>

Technical Details

The application does not properly protect session cookies. In the case of domain, one of the session cookies are being set, without the proper protection attributes and both with the same value (See figure 1). See appendix C for more information.

Current Status

At the current report date, this issue has not been fixed (see Figure 2).

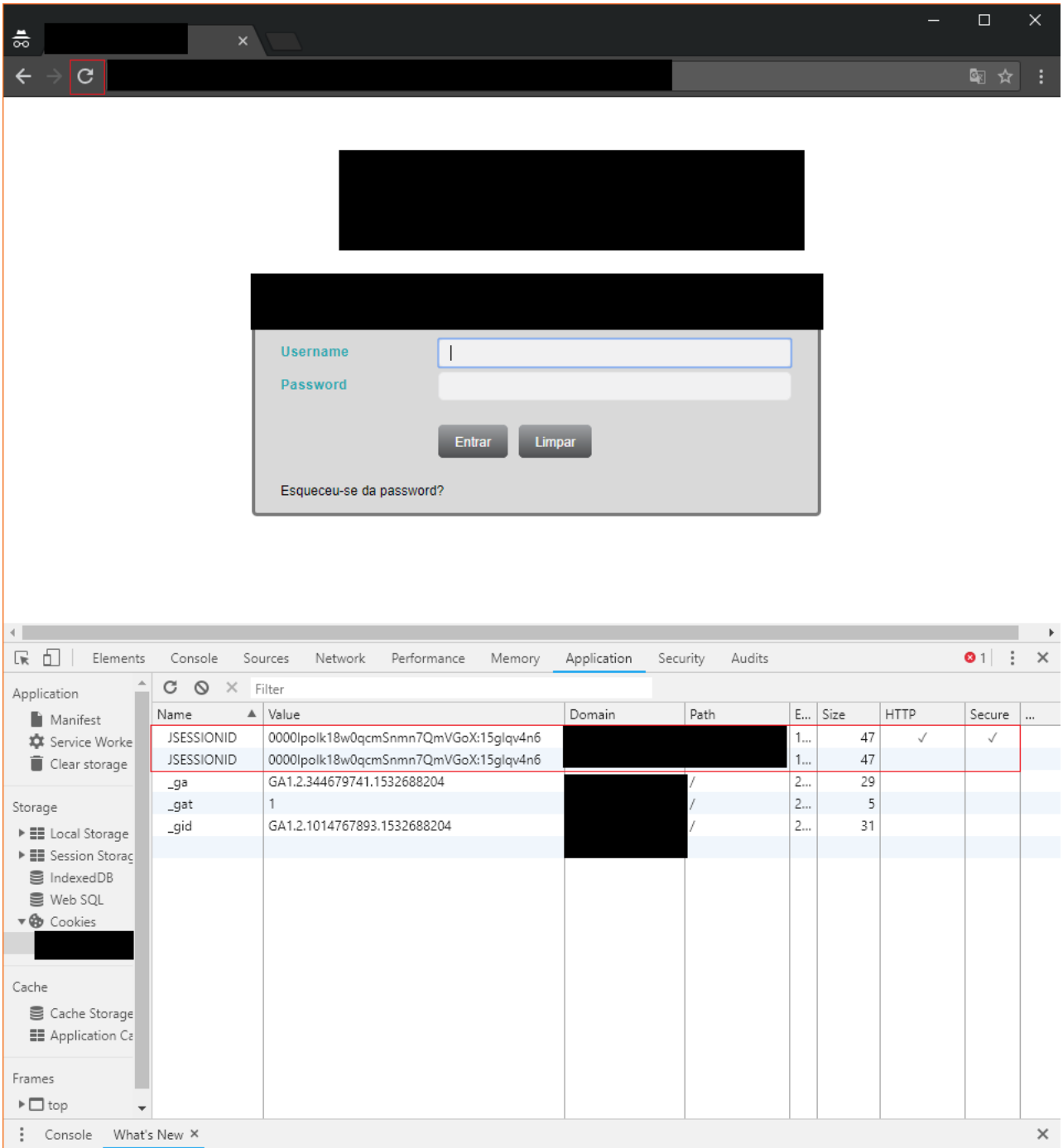


Figure 1 - Figure 1: Missing cookie secure and httponly attribute - HttpOnly.

5.2 High Severity Vulnerabilities

5.2.1 Missing Cookie Security Attribute - HttpOnly

Description

Using the HttpOnly flag when generating a cookie helps mitigate the risk of client side scripts accessing the protected cookie (if the browser supports it).

Impact

An attacker that discovers a XSS flaw on the website, can manage to steal user cookies, because client side scripts are able to access cookies.

Remediation

By setting the HttpOnly flag, the browser won't let client side scripts access client cookies (only if the browser supports this flag, otherwise it will ignore the flag).

CVSS Vector

[CVSS:3.0/AV:N/AC:H/PR:L/UI:R/S:C/C:H/I:L/A:N](https://nvd.nist.gov/vuln/detail/CVSS:3.0/AV:N/AC:H/PR:L/UI:R/S:C/C:H/I:L/A:N)

Other references

<https://cwe.mitre.org/data/definitions/1004>

Technical Details

The application does not properly protect session cookies. In the case of domain, one of the session cookies are being set, without the proper protection attributes and both with the same value (See figure 1). See appendix C for more information.

Current Status

At the current report date, this issue has not been fixed (see Figure 2).

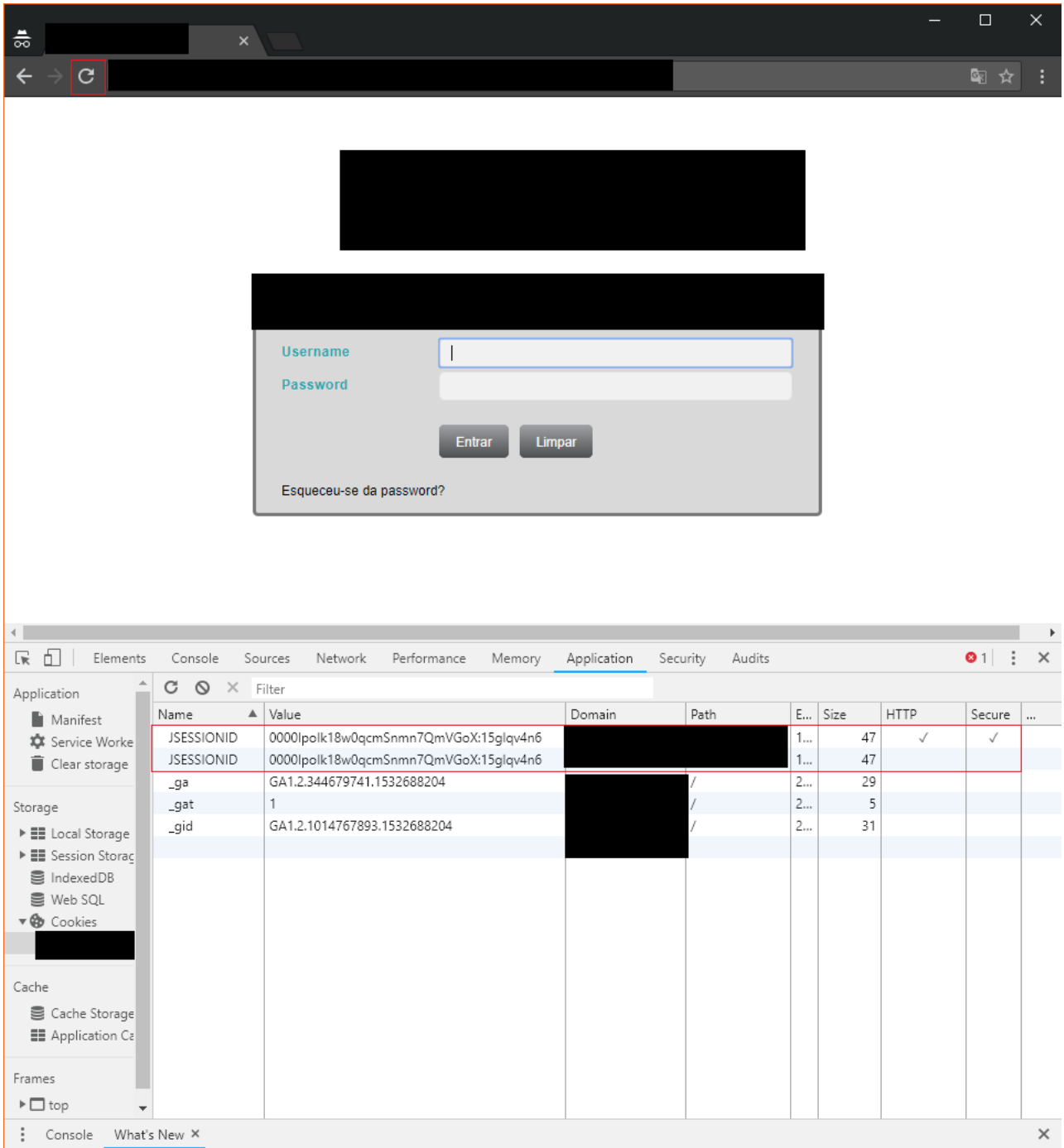


Figure 1 - Figure 1: Missing cookie secure and httponly attribute - HttpOnly.

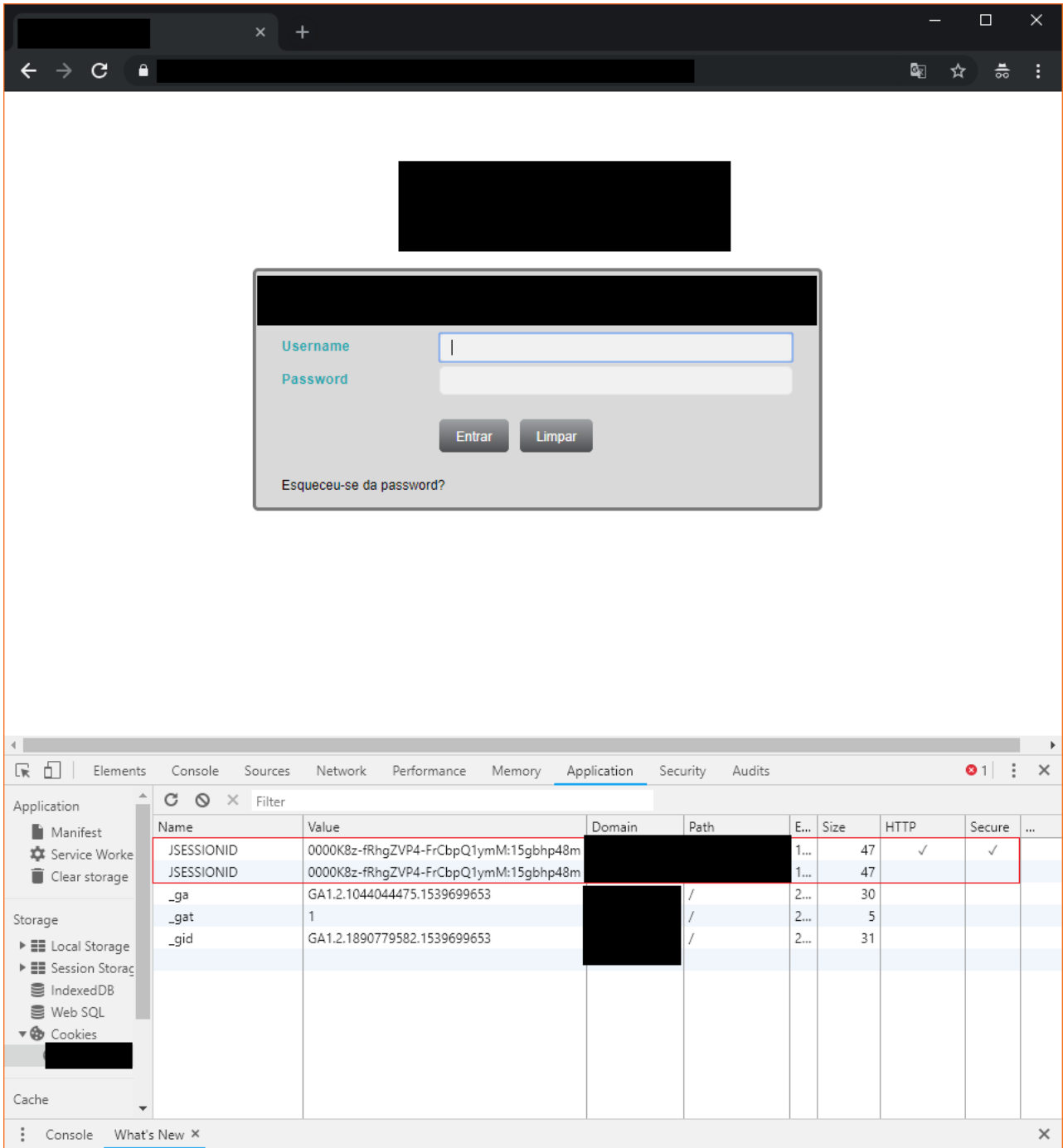


Figure 2 - Figure 2: Missing cookie secure and httponly attribute - HttpOnly.

5.3 Moderate Severity Vulnerabilities

No issues were identified and rated as moderate severity.

5.4 Low Severity Vulnerabilities

No issues were identified and rated as low severity.

5.5 Minor Severity Vulnerabilities

No issues were identified and rated as minor severity.

6 Appendix

Appendix B

Usability Tests Documents

The first document is the Instructions file given to the Usability Tests participants. The second is the auxiliary document containing the data for the Issue to be added to the Report Document.

Usability Test for SAAR2020 vs. Google Docs

My name is Diogo Torres, today I'm asking you to execute a set of tasks in Google Docs and in SAAR2020, the application developed for my thesis.

I will record the time taken to perform the tasks in each system and then ask you to take an anonymous questionnaire regarding these tasks.

To do the tests, execute a task at a time by following one of the columns, then do the next column. You can start with SAAR2020 or Google Docs, it is indifferent. An auxiliary document with content required for both interfaces will be provided.

In advance, thank you for your availability and help.

Here are the tasks, by order to be executed:

0a. Login

SAAR2020	Google Docs
- Enter the SAAR2020 application with credentials: <ul style="list-style-type: none">• Email: alex@saar2020.com• Password: auditor	- Open Google Drive URL.

0b. Open the Report

SAAR2020	Google Docs
- Double-click Report 1 from Project A	- Double-click Report 1 , this opens a new tab.

1. Edit Cover Page

SAAR2020	Google Docs
<ul style="list-style-type: none">- Double-click the Cover page of the document, the first page of the document.- Change the Company Logo entry to 'MAINSEC'.- Change the Target Company entry to 'IST'.- Change the Date to today's date, use the calendar interface by clicking the calendar button on the right.- Submit the form by clicking the Submit button.	<ul style="list-style-type: none">- Go to the Cover page, the first page of the document.- Change Company Logo to 'MAINSEC'- Change Target Company to 'IST'- Change the Date to today's date.

2. Edit Headers

SAAR2020	Google Docs
<ul style="list-style-type: none">- Do nothing, this task was accomplished in the previous step.	<ul style="list-style-type: none">- Click in any header and change the first entry to MAINSEC and the last to IST.

3. Add Auditors and Reviewers

SAAR2020	Google Docs
<ul style="list-style-type: none">- Do nothing, this information comes directly from our databases.	<ul style="list-style-type: none">- Go to section 2.1.- In the Authors and Reviewers table, set Auditor(s), Reviewer(s) and Approver(s) to 'Pilar', 'Paulo' and 'Alex'.

4. Add a Version

SAAR2020	Google Docs
<ul style="list-style-type: none">- Go to the second page of section 2.1.- Double-click the Document Management table.- Click the '+' button, on the last row.- Write the following fields:<ul style="list-style-type: none">● Version: 1.1● Date: today's date(use the calendar interface)● Editor: Alex● Remarks: Released- Submit the form by clicking the Submit button.	<ul style="list-style-type: none">- Go to the second page of section 2.1.- On the Document Management table, right-click the last row and select the Insert row below option- Write the following fields:<ul style="list-style-type: none">● Version: 1.1● Date: today's date● Editor: Alex● Remarks: RELEASED


5. Add an Issue

SAAR2020	Google Docs
<ul style="list-style-type: none">- Click the menu button (☰), on the top-left corner.- Select Critical from the drop-down menu to the right of the search bar- Inside the first table, click the '+' button to the right of Missing Cookie Security Attribute - HttpOnly.- The Issue is now present and correctly formatted in sections 4.1 and 5.1.	<ul style="list-style-type: none">- Go to section 4.1.- Delete the paragraph in the page- Copy the paragraph and table in the first page of the auxiliary document.- Replace [SEVERITY] with 'critical', in the paragraph- From page 2 of the auxiliary document, copy Vulnerability and Description to the respective columns- Write '5.1.1' in the Details column.- Go to section 5.1.- Delete the paragraph in the page.- Copy all the contents from section 5.2.1.- Paste the contents in section 5.1 and start editing the contents from there.- Change '5.2.1' to '5.1.1'- Change the Title, Description, Impact and Remediation to the content given in the auxiliary document.- Copy the text from CVSS Vector and Other References and paste them in the respective places.- Add the URLs as links to these texts, by right-clicking the text and selecting the Link option.

6. Edit the Issue

SAAR2020	Google Docs
<ul style="list-style-type: none"> - Go to the second page of section 5.1.1. 10- Double-click the page, a dialog will popup. - Add the content of Technical Details and Current Status from the auxiliary document (page 3) to the text editor on the left. - Submit the form by clicking the Submit button on the left. - On the two form fields on the right, add the corresponding URL and Caption given in the auxiliary document (page 4). - Submit the form by clicking the Submit button on the right. - Click anywhere out of the dialog, to see the results. 	<ul style="list-style-type: none"> - Go to the second page of section 5.1.1. - Replace the content of Technical Details and Current Status using the contents from the auxiliary document (page 3). - Go to the third page of section 5.1.1. - Delete the image and text copied from the other section. - Add the image from the auxiliary document (page 4). -Below the copied image write 'Figure 1- ', followed by the Caption in the auxiliary document (page 4).

7. Export the Report as a PDF file

SAAR2020	Google Docs
<ul style="list-style-type: none"> - Click the Save () button. 	<ul style="list-style-type: none"> - In the Toolbar above, click the sequence File->Download->PDF Document (.pdf)

This content belongs to the Issue to be added to the Report Document

The following table contains the summary of the issues that might lead to |SEVERITY| severity vulnerabilities:

Vulnerability	Description	Details

Vulnerability or Title

Missing Cookie Security Attribute - HttpOnly

Description

Using the HttpOnly flag when generating a cookie helps mitigate the risk of client side scripts accessing the protected cookie (if the browser supports it).

Impact

An attacker that discovers a XSS flaw on the website, can manage to steal user cookies, because client side scripts are able to access cookies.

Remediation

By setting the HttpOnly flag, the browser won't let client side scripts access client cookies (only if the browser supports this flag, otherwise it will ignore the flag).

CVSS Vector

Text :

CVSS:3.0/AV:N/AC:H/PR:L/UI:R/S:C/C:H/I:L/A:N

URL:

<https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:N/AC:H/PR:L/UI:R/S:C/C:H/I:L/A:N>

Other references

Text:

<https://cwe.mitre.org/data/definitions/1004>

URL:

<https://cwe.mitre.org/data/definitions/1004>

Technical details

The application does not properly protect session cookies. In the case of *domain*, one of the session cookies are being set, without the proper protection attributes and both with the same value (See figure 1). See appendix C for more information.

Current status

At the current report date, this issue has not been fixed.

URL:/assets/data/fig1.png

Caption: Missing cookie secure and httponly attribute - *HttpOnly*.

The screenshot shows a web browser window with a login form. The form has fields for "Username" and "Password", and buttons for "Entrar" and "Limpar". Below the form is a link that says "Esqueceu-se da password?".

The Chrome DevTools Application tab is open, showing a table of cookies. The table has columns for Name, Value, Domain, Path, Expires, Size, HTTP, Secure, and ...

Name	Value	Domain	Path	Expires	Size	HTTP	Secure	...
JSESSIONID	0000lpolk18w0qcm5nmn7QmVGoX:15glq4n6	[REDACTED]	[REDACTED]	1...	47	✓	✓	
JSESSIONID	0000lpolk18w0qcm5nmn7QmVGoX:15glq4n6	[REDACTED]	[REDACTED]	1...	47			
._ga	GA1.2.344679741.1532688204	[REDACTED]	/	2...	29			
._gat	1	[REDACTED]	/	2...	5			
._gid	GA1.2.1014767893.1532688204	[REDACTED]	/	2...	31			

