FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

VCS Based platform for legislation: GitLaw

Manuel Braga da Costa dos Santos Monteiro

DISSERTAÇÃO DE MESTRADO

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Professor Gil Gonçalves

July 25, 2021

VCS Based platform for legislation: GitLaw

Manuel Braga da Costa dos Santos Monteiro

Mestrado Integrado em Engenharia Informática e Computação

July 25, 2021

Abstract

The significance and value of democratic regimes in the world are undoubted. However, despite the apparent consensual opinion about democracy, it is difficult to ignore the intrinsic weaknesses of this form of government. Several times, moments where the general population is more anxious about those that govern us are justifiable by distrust in the system with reasons that extend from a psychological level (corruption, general dissatisfaction, or lack of power over the system) to a technical level (low development levels, unable to keep up with the changing legislation or, even, lack of understanding over the legislation). The human errors that weaken our democracy are difficult to change; however, the potential for improvement on the technical side and digital platforms are significant.

E-participation is a recent paradigm that is rising to change the perception of democracy to better fit the needs of today's society, especially amongst younger generations. In this context, version control systems can have a big impact on every democratic regime. This concept is widely used by software engineers and it is possible to associate a lot of similarities between these two opposite worlds: changes are proposed, changes are reviewed, changes are either accepted or discarded, past changes can be viewed and documented. In a current system where you cannot quickly identify what is new, altered or who even conceptualized it in the first place, this concept can provide a tool to understand the dynamics of a specific governmental law.

Therefore, the objective of this work is to build a platform based around a version control system dedicated to the legislation of any country considering, however, that in an early stage it would be for municipalities as they tend to have a less restricting bureaucratic process, making it easier to be implemented in a short-term. The ultimate goal is for the platform to have the entire legislation of a country and for all the subsequent alterations, to said legislation, to be pushed through using the platform. This would in turn make it easier for the citizens to view the current legislation, the history of a certain law - what was removed, altered, added – and if so propose alterations to the current legislation.

This tool could be a meeting point between society and political decision-makers. Each individual would be able to consult and know the recent legislative proposals through the platform bringing an easier way, not only for the lawmakers themselves to work as a team on a proposal but also for the general public to view in real-time what proposals are going through and being

voted on.

Keywords: Version control system, legislation, legislative process, E-participation, Version control system abstraction, VCS, Documentation, Repository

Resumo

É indiscutível a relevância dos regimes democráticos no mundo ocidental. Contudo, apesar do consenso sobre as vantagens da democracia, é difícil ignorar as fragilidades intrínsecas a essa forma de governo. A apreensão das pessoas com aqueles que nos governam pode justificar-se com a desconfiança no sistema por motivos diversos que vão desde questões psicológicas (corrupção, insatisfação geral ou sentimento de falta de poder sobre o sistema) até às questões técnicas (baixo nível de desenvolvimento do país, incapacidade de acompanhar as mudanças legislativas ou, ainda, incapacidade em compreender a legislação). Os erros humanos que enfraquecem a democracia são difíceis de mudar; no entanto, é significativo o potencial de melhoria que a componente técnica e as plataformas digitais podem trazer.

A e-Participação é um paradigma recente que pode contribuir para mudar a perceção sobre a democracia, adequando melhor esta forma de governo às necessidades da sociedade atual, especialmente nas gerações mais jovens. Nesse contexto, os sistemas de controle de versão podem ter um grande impacto nos regimes democráticos. O conceito de sistemas de controle de versão é amplamente utilizado por engenheiros informáticos e a sua aplicação no campo legislativo permite articular estas duas realidades diferentes: mudanças são propostas, mudanças são revistas, mudanças são aceites ou rejeitadas, mudanças anteriores podem ser visualizadas e documentadas. No sistema legislativo atual – no qual não é possível identificar rapidamente o que é novo ou alterado ou quem o propôs em primeiro lugar –, os sistemas de controle de versão podem fornecer uma ferramenta para entender a dinâmica de atualização de uma lei específica.

O objetivo deste trabalho é construir uma plataforma baseada num sistema de controle de versão dedicado à legislação de um país, considerando, num patamar inicial, os municípios, pois estes tendem a ter um processo burocrático menos restritivo, que facilitará a sua implementação no curto prazo. O objetivo final é que a plataforma integre toda a legislação de um país e que todas as alterações subsequentes à referida legislação sejam efetuadas através da plataforma. A sua utilização permitirá aos cidadãos não só um acesso fácil à legislação, como conhecer o histórico de uma determinada lei – o que foi retirado, alterado, adicionado – e, potenciando a sua participação na tomada de decisões –, propor alterações. Cada cidadão poderá, ainda, acompanhar, em tempo real, o processo de seleção das propostas e a sua votação.

Esta ferramenta pode constituir-se como um ponto de encontro entre a sociedade e os decisores políticos – facilitando e alargando a possibilidade de interação e participação dos cidadãos no processo de tomada de decisão, por um lado, permitindo aos legisladores uma perceção mais

real das necessidades das pessoas que representam, por outro.

Keywords: Version control system, legislation, legislative process, E-participation, Version control system abstraction, VCS, Documentation, Repository

Acknowledgements

I am deeply grateful to everyone that helped with the validation process without whom it would have not been possible to accomplish the results reached with this dissertation.

I would also like to thank all my friends without whom it would have been incredibly hard to complete this dissertation. So to Urso, Moreira, Lara, David, Henrique, Sofia, Cheio and many others that were always there when times were the hardest, thank you.

To my closest family who has helped me over my education course, my sisters, my mother, my uncle Miguel without whom I would have probably never grown fond of this area of study, to my grandparents, and Mochi thank you for the help and support.

"Education is what remains after one has forgotten everything he learned in school.
It is a miracle that curiosity survives formal education."

Albert Einstein

Contents

List of Figures

List of Tables

Abbreviations

VCS	Version Control System
CVCS	Centralized Version Control System
DVCS	Distributed Version Control System
LP	Legislative Process
PLP	Portuguese Legislative Process
PDM	political decision-makers
SD	Software Development
VM	Virtual Machine
CTL	Command line tool
WWW	*World Wide Web*

Chapter 1

Introduction

1.1 Context

VCS has been around for many years now, they are an integral part of any software developer's career, however, abstractions have been created for this concept to allow many other professional areas to enjoy the benefits of VCS.

In the software development world, they allow professionals of the area to more efficiently manage projects, this is made possible by using tools like GitHub, GitLab, BitBucket, amongst others. Which provides some features like the ability to have different branches of code which, in turn, allows developers to implement features without cross-contaminating any other developer's code, the ability to view past code alterations. The ability to allow code to be added or not to the main branch (usually called master, which is the code that's for production).

Just by the mere observation of what VCS allows us to do it is possible to see the benefits and how it can easily improve many areas in society, LP is just one of those areas.

In this dissertation the focus is going to be on the PLP, the main reason is that it would be impractical to focus on every single different LP in the world, and in fact, the goal is to generalize the LP process for any country. However, it is important to have a point of comparison to know if what is built during the process of this dissertation is a net improvement over the current process.

Because of this, it is important to understand the PLP. At the very beginning of the LP, there's is an idea that has to be put into written words, this work is, in general, done by political decision-makers, this is the proposal drafting stage. During this stage, a group of people work on the same paper and try out different ideas, but nothing is definite until the very end and changes are made throughout the entire document at a fast pace.

When the proposal is deemed ready by those that worked on it, it is brought to the Parliament to be discussed, it must be distributed with every congressperson some time in advance. It

is then discussed and can be either approved or rejected. An approval always requires more than 50 percent approval (with some exceptions) by the congresspersons that are present in the parliament during the voting process.

If a proposal is approved it is then published to Replubic's Diary (Diário da repbública) as it is here that the general population can view the current legislation of the country.

However, a specific legislation can be vetoed by the president after it is published, which forces the proposal to go back to the parliament and be discussed yet again.

It is also important to realize that even though the constitution foresees the participation of the general population in the LP in the case of the national parliament it is only allowed for citizens to participate in two ways: voting for representatives and public petitions.

Citizens can only participate directly (propose legislation alterations) in municipalities so the solution proposal for this dissertation will have different solutions for both of these use cases, mostly this will be translated into user role limitations depending on what the LP, nacional level or municipalities, is directed towards.

1.2 Motivation

The LP has not seen any major change since the implementation of democracy (not only in Portugal but almost all countries around the world) this is true even though everything around it has seen major disruptions with technology becoming cheaper and more user-friendly. With the push from the European Union to push more E-participation projects through, this dissertation fits in with the global needs by adding a layer of technology to a process still very much stuck in the past.

This push by the European Union is made possible by public investment and initiatives that promote the creation of platforms, tools, ideas that promote E-participation.

In sum, this means a lot of researchers, companies, and individuals have been working towards this objective. However, this dissertation pinpoints a problem that has been lacking in research or objective proposals for improvement.

1.3 Objectives

The main goal of this dissertation is to develop a platform capable of implementing a VCS in the context of the LP but more than that, it is also meant to be an all-engulfing place for everything that has to do with the LP, depending on what's possible to do with the time expected for the regular completion of this dissertation. But there are plans to implement more tools than just the VCS like for example a forum, a voting system (both input of Parlament votes and view of

said votes), and, additionally, to add the complete current legislation to the platform, this would require to implement a web scrapping algorithm over the current platform that exists.

It is required firstly to understand the current process and define a new one better suited for the proposed solution, the next chapter will be dedicated to this as it is really important to have full knowledge on the current system, all the information is present in the Portuguese constitution, however, meetings were made with some professionals that work on a daily basis with this type of platforms to better comprehend the details of the PLP.

1.4 Dissertation Structure

In this section, we'll be taking a general look into how this dissertation is structured and give a brief explanation of what each chapter will talk about, this will give the reader a good overlook into what this dissertation will focus on beforehand.

Chapter 2, **Literature Review**. The first part of any dissertation is to get an overall comprehension of the previous work done in the area of study. This chapter will study how VCSs have evolved over the years and give us a better look into this tool that is, for most developers, a "black box". The current platforms dedicated to democracy will also be studied so that it is possible to evaluate the current state of E-participation in Portugal and how all of them are lacking in features.

Chapter 3, **Problem Characterization**, will both Characterize the problem at hand and propose a possible solution for this issue. We'll also elaborate a possible work plan for this dissertation which will both be on the development process and the validation process

Chapter 4, **Platform Characterization**. A chapter is dedicated to the platform characterization, this is an important step before actually starting the development process as it will define the tasks needed to complete for the development of the platform in simple tasks which will help in making the process agile. We'll also define user roles, which will help in setting security access depending on which user type is logged in at the time.

Chapter 5, **Platform Development**. This is the main chapter of the dissertation as it takes an in-depth look into the platform.

Starting with the DevOps, where we'll evaluate the proposed architecture for this platform and how the docker is structured along with all the services used. Along with this, all the packages needed for the implementation of this platform are studied in what they do and how they work. Finally, we'll take a deep look into some of the most important features like the search function and the VCS implemented.

Chapter 6, **Validation Process**. As with any platform, the validation process plays an important part in making sure the implemented solution is viable and to study how it can be improved,

for this effect, two validation processes were made. In this chapter, we'll see how the validation process was structured and what conclusion was taken from each of them.

Chapter 7, **Conclusion and Future Work**. Finally, a Chapter is dedicated to the conclusions of this dissertation, compiling all that was learned in the validation process and general conclusions taken from the development process. Also, a subsection is dedicated to how the platform can be improved in the future, what can be done to make sure this is a viable solution for the problem studied in the first Chapter.

Chapter 2

Literature Review

In this chapter, the various components relevant to the problem addressed in the first chapter are explored. These components are:

- VCS Overview
- The history of VCS
- Characteristics of a VCS
- How it works
- Existing VCS Applications
- Existing E-Participation Applications

2.1 VCS Overview

VCS has become an integral part of every software engineer's life, it is not enough to just know how to code you have to be well versed in the ins and outs of a VCS tool (like GIT or SVN).

Even though VCS exists since 1972 with its first implementation being SCCS, GIT (created by Linus Torvalds, the creator of Linux, in 2005) is the one that is most used today, as we can see from the Figure below. This happens even though they all accomplish fundamentally the same task. The versioning of files, which, in summary, means that the history of files is kept with any change that's made, but how they all implement this system is widely different, not just different features but different performance as well.

[1]

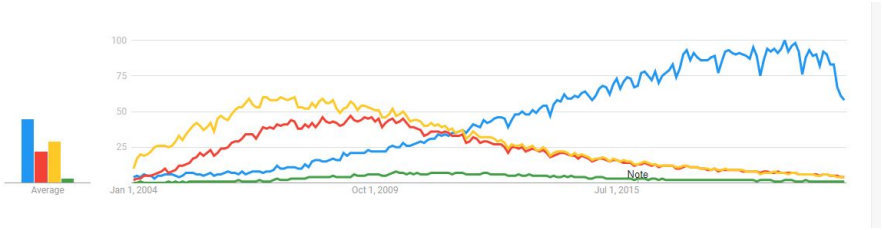

Figure 2.1: Popularity of git (blue), svn (red), apache subversion (yellow), mercurial (green) since 2004 - google trends

VCSs are especially important and relevant for software engineers because of the continuous changes made to code during development and after. This means at any point of the development process there are a big number of versions of the same project going around. VCSs resolve this issue. Not only that but without VCSs software engineers would have to keep multiple copies of their projects around and would have to manually add, remove or alter each line of code when another developer finished their work. This is time-consuming and is prone to errors.

2.2 The History of VCS

There are three generations to VCS throughout its history, with every generation we saw a significant improvement over the system in several different ways (security, availability, general features,...). However there are only two main differences which are centralized vs decentralized VCSs, this is because we could say the second generation is merely a transition generation as we'll study in the upcoming chapters.

We'll be just discussing one VCS per generation as every generation's VCS is similar.

2.2.1 First Generation - SCCS

The first VCS documented and widely used was SCCS, which was written in 1972 by Marc Rochkind. SCCS was limited as it allowed only one user to work on a file at a time and had no network features implemented however it set the stones for generations of VCS to come and completely changed the way we think about project management. We can think of it as the first VCS because it was the first to implement the concept of the history of files well, even though other concepts had already been in use, this was the first to take it to a standard where we can confidently say the VCS concept was in use.[2]

In Rochkind's paper, which was the inspiration for the "How it Works" chapter of this literature review, we can see exactly how it works in simple terms, the project was only put to open source in 2005, IBM was the official owner of the project up until then.

It lacked a lot of the features we currently have in systems like GIT, but it was an important first step into being able to do file versioning. This lack of features meant that it was easily adopted by any software developer as there existed only a handful of commands available and lightweight which, for the hardware available at the time, was an important concern to have in mind. SCCS was locking, file-oriented, and centralized (these concepts will be studied later on). [3]

All this simplicity came at a cost, the lack of network meant every developer had to work out of the same machine to access the repository.

2.2.2 Second Generation - SVN

During the 1990s we saw the appearance of the second generation of VCSs with CVS, it included network support, which was starting to be widely used during the late 1980s, and not just at a corporal level. But CVS had a tremendous amount of problems, some developers that were part of this project started a new one in the early 2000s, this was going to become SVN. It was officially released in 2004, even though it was in beta since 2002[4].

It improved by innovating certain aspects of CVS without ever losing the CVS's essence. While CVS only tracked modifications by file, SVN tracked each commit, so developers were able to view a commit as a block making it easy to see what was pushed in that instance. CVS was also filled with bugs and operations like renaming files could destroy the work put into that single commit, SVN was polished to a point where it was a stable VCS and one that was king until the appearance of a DVCS.[5]

2.2.3 Third Generation - GIT

In the third generation, the one that's currently widely used, we saw the appearance of a decentralized version control system (DVCS).

GIT, although not the first, is the most relevant today, so that's the one this section is going to be focused on.

GIT fixed all the problems that came with SVN and even added other features that today's developers can't live without. The main difference from the second generation's VCS was the implementation of a decentralized model[6], this concept will be studied in the next section, but in general, it means that every developer has a complete copy of the repository, this is good and revolutionary for many reasons but also has its drawbacks like having to download a big project

is more expensive (if you take into account that each repository contains the complete history of every single change made to every file it becomes easy to see why this might be an issue), although, with network and storage improvements, this is becoming a nonissue.[5]

2.3 Characteristics of a VCS

As per what's said in the previous subchapters, VCSs can have a lot of differences between them, these different characteristics define the general aspects of the VCS and give the user a broad look into the VCS. It is hence important to know what each definition of these different aspects means.

2.3.1 Centralized Model vs Decentralized Model

The model of the VCS is related to its architecture, meaning how it was distributed. In the earlier versions, VCSs were limited by the network capacity and technology in general, nowadays it is no longer a problem so VCSs have, for the most part, embraced decentralized models. there can be one of two models:

- Centralized
- Decentralized

2.3.1.1 Centralized

A CVCS, as the name implies, has just one main controller of the whole repository and, in turn, just one main copy of the repository, this requires developers to be connected to the internet at all times. The first VCSs ran on this concept, it is significantly less complicated than its counterpart. However, because everything runs off of just one entity this makes it a potential security hazard as there is one single point of failure.[6]

2.3.1.2 Decentralized

A DVCS takes a different approach to this issue[7], by allowing multiple copies of the repository to be made, every developer has a copy of the repository in their local machines which allows them to work without internet access, however for working collaboratively it will always be a necessity. There is no longer a need for a single entity controlling the entire repository, this means developers are no longer restricted and no longer have to "stay in line" to be able to work on a project. Because of this, it is designed to have the best of both worlds giving the user huge amounts of flexibility.[8]

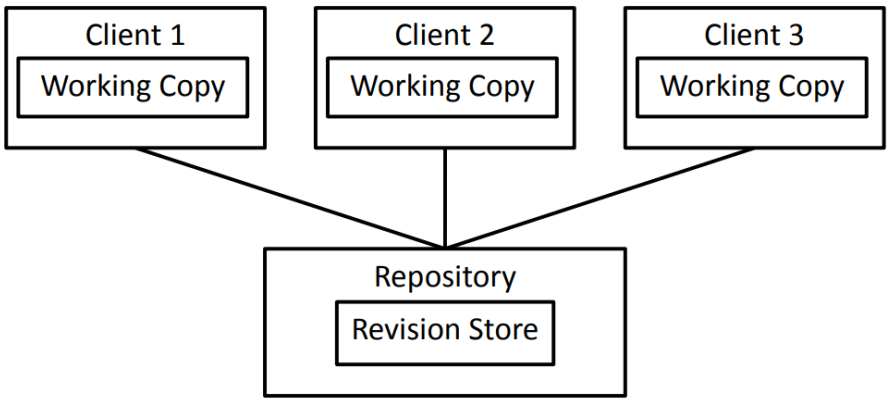

Figure 2.2: Structure of a DVCS

This also means that no one that is working on a specific repository is dependent on the original server and if the original server has a fatal failure (disk corruption, loss of power, ...) no work is lost as every change to the repository is stored both locally and in the main server. This makes it much more structurally sound.[9]

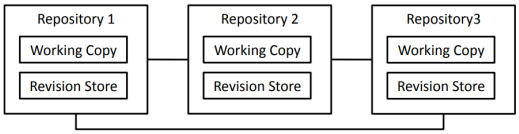

Figure 2.3: Structure of a DVCS

2.3.2 Merge Model vs Locking Model

Conflict resolution is an integral part of the VCS concept, VCSs try to solve this either by restricting access to the user or by allowing full freedom but making the user deal with conflicts later on. There are two different ways of going about this:

- Locking
- Merge

2.3.2.1 Locking

This is the most basic way of conflict resolution, it takes the more restrictive approach however which is why it is not used anymore and we've moved into a merge approach.

In summary, locking works by only allowing one user at a time to read and write to a file. As this was used for centralized VCSs then it becomes clear that by using this approach no conflicts would appear as there could only be one developer working on a specific file at a time, this would, in turn, mean conflicts can't exist as the file version a developer is looking at (if the developer has the lock on a file) is always the most updated version of that file. [2]

However, the problem that arises from this approach is quite clear. It is unproductive as if a developer is working on a file no one else can even read it. Making it inefficient in most, if not all, cases.

2.3.2.2 Merge

merge takes a different approach to this problem, instead of locking a user to a file, it allows multiple users to view and edit the same file. This is made possible by making sure that, when you try to commit to a file that has changed since the developer has started to work on it, you resolve any conflict that appears before that operation is successful.

This has obvious advantages because it allows for a more seamless work experience in a team. Merge before commit was popular during the second generation of VCSs they are however not as popular right now as they require the developer to solve conflicts right away which can impact productivity.[4]

2.4 How it works

For this dissertation the focus is going to be on a decentralized and merging VCS as it is the most flexible type of VCS and technologically advanced VCS. For that reason, in this chapter, the study case will be made around this type of VCS.

On its most basic form, a VCS stores every change made in what we can think of as a chain and each link contains all the information on the change the user made at that time, for example, in regular English we would say "The user as altered the line x of the file y by adding the string z in the t position" and this would correspond to one link.

But it does not end here. We can also have multiple chains, called branches, and merge them at any stage of the development process. When the VCS is first started and is empty all we have is an empty box (the main branch), however, we can an infinite amount of empty boxes from that main branch, and any change made to the new branches will affect the chains correspondent

to those branches, not the main one, they are completely independent from each other. This is incredibly useful as it means multiple people can work on the same project without interfering with one another. By merging different branches we are just combining the chains of the two branches, if there is a conflict (a link from each chain alters the value of the same line) then that issue must be resolved before merging.[2]

In a more detailed way, there are some different ways of implementing a VCS mainly Centralized vs Distributed, both of them have their pros. and cons. but distributed seems to have won the market.

2.4.1 GIT - a Technical analysis

For the development process, it is important to study the ins and outs of existing VCSs to understand exactly how they work and how it is possible to both adapt it to our needs and how we can improve upon existing solutions.

2.4.1.1 Objects

To keep track of everything that's going on when a user makes a commit to a specific branch GIT uses the main git object which has an SHA-1 key generated automatically when it is created, this object contains general data:[10]

- **commit** which contains general data like commit date and creator
- **blob** which contains all the file's data (raw source code, images, videos,...)
- **tree** Which contains other trees and blobs (acts like a directory)

By having access to a GIT repository it is possible to view each git object with the following script which will print out the contents of a specific git Object:

```
1 import zlib
2 compressed_contents = open(path_to_.git/objects/commitId/commitId, 'rb').read()
3 decompressed_contents = zlib.decompress(compressed_contents)
4 print(decompressed_contents)
```

This object is responsible for everything as it tracks every commit and all the changes made in each one of them. It is the most basic object and it is responsible for the ability to version files, the idea of branches is not present in this object as it is not relevant to them.

2.4.1.2 References

For branches, we need yet another tracker which will just keep track of the branches we have pulled from our git remote repo or already have in our local repo. In the .git/refs/ there are three folders:[10]

- **Heads** contains the commit that each branch is on locally.
- **Remotes** contain the commit that each branch was on the last time a pull action was made.
- **Tags** Contain a git object from a moment in time the user deemed important.

All of the files above (apart from the tags) are simple .txt files that just contain the hash for the commit in questions so it is easy to find out what they are just by opening up these files.

2.4.1.3 Packfiles

One of the most important steps in the process is compression, this is because in big projects if no compression was used then the repo would quickly grow to a size that would make it hard to work with, for the case of GitHub, for example, this problem is exponential as they host 100 million repos as of 2018.[11].

To save space git will run, periodically, this compression on multiple files (objects, refs,...) into one pak file. Objects that are still being used are left alone while objects the user has not used in a while are given permission to be compressed.

2.4.1.4 Conclusion

One interesting thing that might be derived from this is that Git does not need a database to run, it is a basic file system that does not require any overly complicated database system or technology that's beyond the reach of anyone. The database, in the case of Github for example, is just a means of storing users' data and which user has access to what repository, the VCS aspect of it is a separated entity.

2.5 Existing VCS Applications

Even though the focus of this dissertation is on the LP, it is important to study in what areas have VCSs been implemented, how was it implemented, and for what reason. By studying this we can draw similarities between both worlds and validate the hypotheses that VCS is a broad concept not limited to software engineering.

2.5.1 Plastic SCM

Plastic SCM is a distributed VCS made specifically for designers, it gives them the ability to do everything VCSs like GIT do, but improves upon the points where Git is just not enough for the requirements of designers. It is best seen as a merge between GIT and SVN as it allows for centralized work but can also allow for hosting in the cloud. [12]

However, the major differences from normal VCSs come from the fact that can allow for bigger than usual file sizes to be in the repository. This is important as binaries that designers use (.PSD, .IA, amongst others) are significantly larger than the usual files programmers are used to (.c, .java, amongst others, but all of them are, generally, basically, basic .txt files) this is then a specific requirement needed for designers, but it's not the only one available. It comes with a GUI that's easy and intuitive to use, this is also a must as their work does not require them to be as technically savvy as software developers, by giving them the option of having a graphic interface to go by it makes it much easier to understand what's going on, instead of the regular terminal software engineers are used to working with.

2.5.2 SimulDocs

Simuldocs aims to version any regular text document, more specifically word documents. This is aimed primarily at lawyers but has use cases much more general[13]. It works as a very basic VCS, abstracting most parts and thus making it incredibly easy to use. After the user creates a word doc, the document has to be uploaded to the Simul website, this will be the first commit to this repo.

A big team can be working on the same file as if two people make changes to the same place in the same file Simul will automatically branch out both of them, without the users having to implicitly create a new branch. Any conflicts are handled with easy to understand graphical elements, by underlying the parts that conflict and asking the user to pick which one will be kept in the merge.

2.6 Existing E-Participation Applications

It is also important to know exactly what E-Participation driven applications exist in the market currently as it will allow us to directly compare the hypothesis of this project and the state of this project at any moment of the development process with the existing applications. These applications are not based around VCS but try to engage the public in the legislative process either by allowing you to view documents, comment on them, or give you any relevant info that you might be looking for.

2.6.1 ConsultaLEX

When it comes to existing platforms that allow citizens to participate in the LP and to review said legislation, they are scarce however ConsultaLEX (www.consultalex.gov.pt) tries to accomplish this goal.

It does so by allowing users to comment and follow up with proposals, however, it does not allow them to create their proposals. And the search functions are somewhat restrictive as they don't allow for full-text-search in all the available proposals, only specific titles of said proposals. The website works and feels like a limited forum because of this.

This is an important first step as the Portuguese Constitution has an article dedicated to the participation of the general public in the legislative process (although it differs in range from the Parliament to the municipalities) . [14]

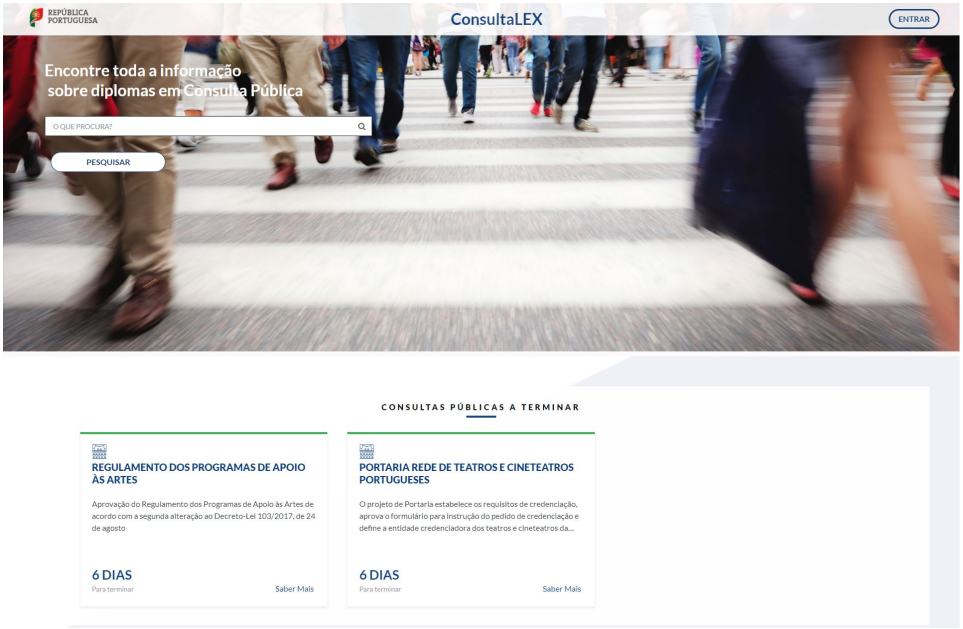

Figure 2.4: ConsultaLEX main webpage

2.6.2 Petição Pública

One tool that allows citizens to directly make petitions and to ask for change or answers on any topic they deem necessary is Petição Pública ("Public petition"). This Portuguese website has gained a lot of traction over the past few years as it is the primary website used for regular citizens to gain visibility and support over the causes they believe in.

Anyone can create a petition and anyone can support it, to do this only an ID number is necessary, making it really user-friendly.[**?**] This is important as the protection and right for people to present a petition either individually or collectively is protected under article 52 of the Portuguese Constitution.[15]

Chapter 3

Problem Characterization

In the previous chapter, several articles and papers were analyzed, these papers were mostly focused on how VCSs work and the history of this concept, however, they were all made with software developers in mind, which make them anything but user friendly, even with the rise of platforms which allow an easier view of the repository if you are not used to the terms and workflow of that repository it will be nearly impossible for a member of the general population without a background in software development to be able to understand or use said platforms.

When it comes to the LP as it stands today, it is still an over bureaucratic process, that is anything but transparent for the great majority of the Portuguese citizens. Even though it is hard to find empirical data to back this affirmation, talks were had with professionals that work daily with tools like the republic's diary and the feedback both on those platforms and the LP as a whole was negative.

The work that's done on a proposal is made with regular tools (word, google docs,...), when ready it is sent to the responsible entity over old fashioned means (email mostly) and when it comes to adding the proposal to the currently running legislation the document is added to the republic's diary and the vote is added to a different website, parlamento.pt.

The search of both legislation and votes of said legislation at the moment of either approval or rejection is, because of this, overly complicated, specially for those least used to using technology.

3.1 Proposed Solution

In this paper, we'll focus on what workflow can be easily understandable to an average user and how to implement it in a way that is easy to use but doesn't slack on any of the advantages of a

regular VCS. Security is also a big concern so we'll have to implement a mechanism to detect users and cross-check them with the governmental authorities.

Now that we have a good overlook over the legislative process, what is a VCS, and what is e-participation, it is important to figure out why a VCS is a good solution for the current LP faces and how they two could merge in one platform that would be a one bus stop for all the legislative needs of a particular governmental entity, whether this is a municipality or even at a national level, again, it is important to keep in mind that Portugal is taken as the use case study for this dissertation.

In later chapters, especially in the first validation process, we'll validate this idea with Gitlab, which although overly complicated for this problem it will give us a good overlook into if this solution is worth pursuing, but for now, let's look into what this merger of two worlds might look like.

The idea would go as follows. A citizen would login into the platform and create a new proposal, this action is equal to the branch creation in a normal VCS like GIT. Then the user would be able to work on their proposals (branches) and anytime a file gets edited by the user this would trigger a merge between the file state that exists in the user's browser and the one that exists within the server and trigger a merge conflict process if need be. This is needed because one important aspect of the proposed solution is to be able to have multiple users contributing to the same proposal, so one can change the file at the same time another user is changing it. Then, after the proposal is marked as ready (equal to opening a merge request in a VCS like GIT) the admin would be able to accept or reject the proposal, if it gets accepted the proposal is merged into the master branch with every file that got changed in the proposal replacing that same file in the master branch.

The potentialities of a VCS in this use case are almost endless; however, with the time restrictions of this dissertation's development time, it is important to focus on the main ones first (branch creation, branch editing, merge requests).

It would in sum, mean a more agile process for the PDM which would be the ones that would use the platform the most, it would mean that it would be easier for those in the judicial branch of the state to find legislation and find the history of said legislation and for all other citizens it would reflect a bigger trust in the system as it would be more transparent for all, this latest point although important would be hard to achieve due to the different abilities citizens have with computers and how at ease they feel with them, but an effort has to be made to make sure this would apply to the greatest amount of citizens as possible.

For context let's take a practical problem and analyze what steps a user would have to take in the current system and how it would work in the proposed solution's platform, we will access the Decree-Law number 3/2020

3.1.1 Current System

First of all the user would have to input the number of the decree in the input box and, when this is done, a big list of links will appear, similar to a normal search engine like Google.

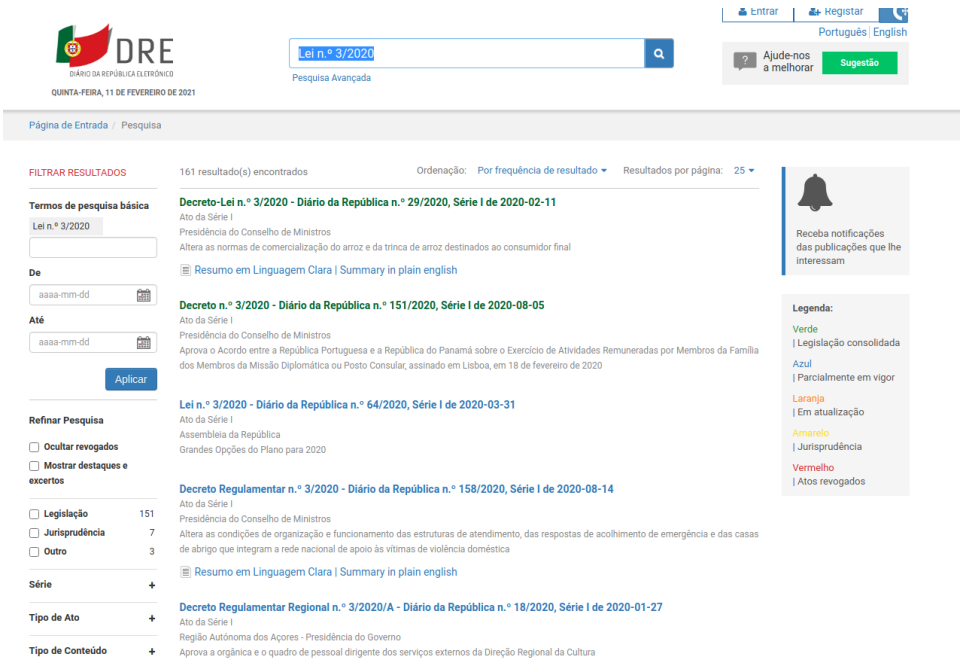

Figure 3.1: Diário da República - search for decree

After searching for the correct link, which might not be obvious at first, the user can enter the link and has two options for viewing the contents of it, either download the original pdf or view the the contents in plain text, this later solution might be confusing as it loses the format that's present in the pdf.

Finally, if the user wants to view past alterations of this same decree they will appear as loose links in the middle of the text. and this action will have to be done recursively if the user intends to analyze all the alterations of the decree in question.

3.1.2 Proposed solution

For the proposed solution we'll take the example of a regularly used VCS platform like GitLab which, even though the final platform will be different from this one, the core of the platform should be somewhat similar.

For the first step of our example scenario, the process would be similar. The user would input the number of the decree in the input box and, when this is done, A list with possible

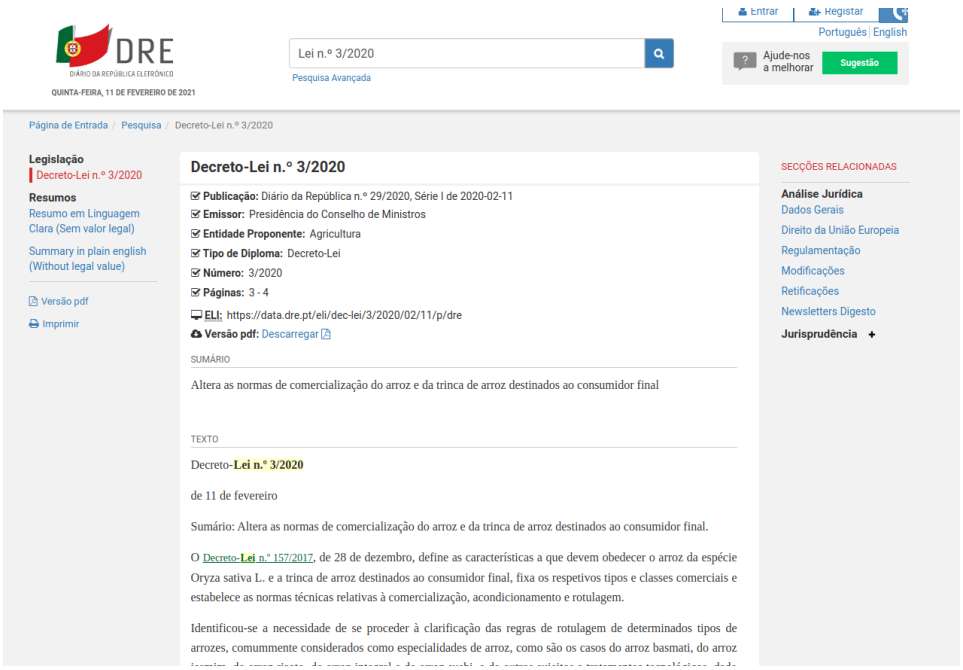

Figure 3.2: Diário da República - decree information page

corresponding decrees will appear, we will however try to make the whole user experience better by giving more visual info on each of the suggestions.

Then, when entering one of the options, the user will get the compiled markdown of the decree along with any other information relevant to the decree like date of approval along with who wrote it for and any other relevant data. The user will also be able to create a branch from this file and propose alterations to the file directly. A list of proposed alterations to this file will appear along with this page.

Finally, we have the final step of this scenario in which the user wants to see past alterations to the decree. In this case for each paragraph, the user will be able to click it and view every single past alteration seamlessly.

3.2 Methodology

The primary tasks for this dissertation are:

- Workflow study
- Evaluate the possibility of using an existing VCS

- Platform Development
- Platform Testing
- Platform Improvements
- Validation

3.2.1 Workflow Sudy

This will be the first part of the work plan and, notably, the most important part as it will set the stones for the rest of the development process. It is expected that a workflow can be conceptualized that better fits the needs of this specific application of a VCS. This will be validated by PDM and they will be involved in the process as much as possible as they are the primary target for an application of this kind.

3.2.2 VCS Evaluation

The next step is to decide if it is possible to use an existing VCS, like GIT for example, or if necessary to build one from scratch. This choice will be made by taking into account the workflow detailed in the task above. It will be studied on chapter 5.

3.2.3 Platform Development

In the platform development part, the focus is going to be solely on coding the platform according to the decisions made in the two points above and knowledge gained from the state of the art. During the first two months of this step, there is not going to be any validation as the goal is to have a working prototype ready to be presented to PDM to test and give their informed input, after that it is a matter of giving the finishing touches to the platform and finish writing the dissertation with everything that was made and learned from the development process.

3.2.4 Validation

Validation is something that will be always a part of the development process, even though there isn't any planned specific validation work for at least the first two months of development. There are planned three main validation timeframes:

- **Pre-development validation**: this step is to be done before actually starting the development process, at this stage the focus is going to be on building a mock scenario using existing VCS tools like GitHub and try to emulate the process that would be to take place

in the platform to be built. This is meant as proof of concept and to determine exactly perceived as being most important to PDM.

- **Prototype validation**: After the main prototype of the platform is ready, the second moment of validation is made, during this tests, the goal will be to see if the platform is up to the standards of PDM and to see if the process is viable or needs adjustments during the last few months of work.

Chapter 4

Platform Characterization

This is the first step in the development process itself, defining what the project will be more detailed in than the previous chapters. This is important because an effort was made to make sure this project follows the agile guidelines.

- Methodology
- User Roles
- User Stories

4.1 methodology

For this project, as per what was said before, AGILE methodologies were used, more specifically Scrum.

Scrum aims at breaking a big project into smaller and more manageable chunks or tasks. With a big focus of having a set of pre-determined tests scheduled throughout the development process which allow for the customer to be in constant contact with the development team, this aids in making sure the final product meets the product owner's requirements as best as possible as their input is present from the kick-off meeting to the day of the project handover.[16]

Sprints are sections of time in which the development team has a set of tasks attributed to them that they must resolve, this usually is one week long, hence the importance of having each task be small and concrete.

For this dissertation as the development team is made out of one developer, this methodology was applied more loosely, however, an effort was made to try and keep up with it especially when it came to the validation tests which is the main focus of Scrum.

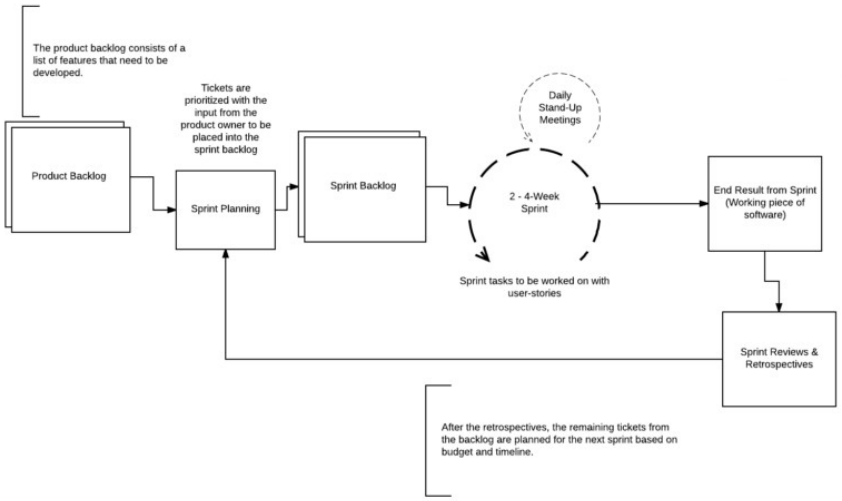

Figure 4.1: Scrum flow diagram

This subject is relevant for this chapter as it is here that the tasks planning will be made along with the definition of the user roles.

4.2 User Roles

The first step in project planning is to define which user types the software is going to have, this helps in defining permissions and features that each user type should have access to.

This is important as it groups single users into groups, as it would be not practical to implement features without knowing what user has permissions to access that same features.

Each user role is incremental meaning, the user role that's on op will have all the permission that the user role below has, this action is done recursively.

There are only three types of users, however, there are two main types, the admin and the normal user. This later one is subdivided into citizen and official, however, these two share the same abilities and restrictions, they are only meant to distinguish between elected officials and the normal citizen, right now they don't have any real distinction when it comes to usability.

Every user defaults to the citizen, the admin is the only one that can change this, and for a user to be an admin the change has to be deployed directly in the database for security reasons.

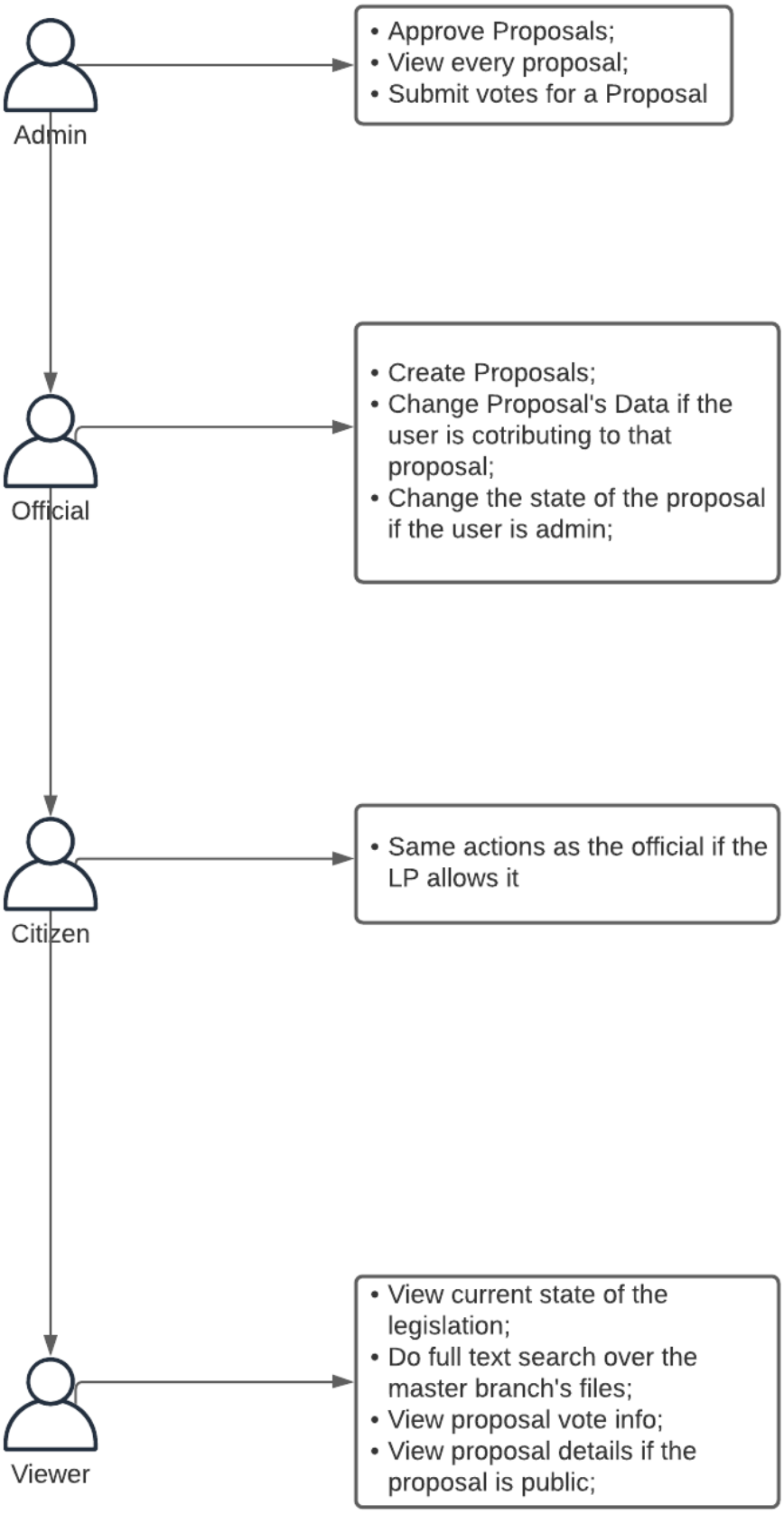

Figure 4.2: User roles

4.3 User Stories

One really important aspect of software development, specially when using g Scrum, is to define user stories these serve as a guideline for all the features a software should have and help developers to split the work into smaller, more manageable, chunks.

Each user story has information for what action that user story will do, the reason why it is needed and what user role that user story is meant for. In sum, a user story is an informal, general explanation of a software feature written from the perspective of the end user or customer.

This makes it so that the focus of the developer while developing a platform is on the end user, which in turn, guarantees the end result fits the end user's needs and not the perceived notions of what the developer thinks the user needs, this is a tactic used in app design, where personas are used to better understand what the end user will want from a specific project.

The sprints of the project development were made with these user stories in mind, different weights were given to each user story to make it easier to calculate the time needed for each one.[17]

Number	User Role	Action	Purpose
US 1	Viewer	View the current state of the legislation	So that I'm able to know the current legislation
US 2	Viewer	View Public Proposals	So that I can track their development process
US 3	Viewer	View Proposals Vote Info	So that I can know who voted for what
US 4	Viewer	View proposals that were voted on	So that I know what was voted on and what changes were voted on
US 5	Viewer	Search for text within the current legislation	So that I can see the files that talk about something I'm interest in
US 6	User	Login and register	So that I'm able to use all the functions of the platform
US 7	User	Create a proposal	So that I can make my suggestion about a topic
US 8	User	Change the current legislation files within my proposals	So that I can change the current legislation
US 8	User	Add or remove contributors to my proposal	So that I can co-work with other users
US 9	User	Create Files and folders for my proposal	So that the proposal better fits my needs
US 10	User	Delete Files or Folders	so that I can remove unwanted elements from my proposal
US 11	User	Change the current state of the proposal	So that I can submit it for a vote, delete it, or close it.
US 12	User	View the proposals I'm contributing to	So that I can easily find the ones I'm working on
US 13	Admin	View all proposals	So that I can keep track of every proposal
US 14	Admin	Filter Proposals	So that I can view a list of proposals with specific parameters
US 15	Admin	Input the vote results for a proposal	So that I can save the result of a voting process
US 16	Admin	Accept or decline a proposal	So that I can either merge a proposal with the master branch or reject it
US 17	Admin	Change any user's roles	So that I can make them a citizen or official

Table 4.1: User stories of the platform.

Chapter 5

Platform Development

The architecture of a platform like this is crucial to make sure a good balance between server cost, security, modularity, and security is achieved.

It is also important to choose some packages that make it possible to develop such a complicated and completed project like this, one of the most important acronyms a software engineer should keep in mind is KISS (keep it simple stupid), with this notion in mind some packages where used that allow for a faster development process along with many other advantages like the ability to use code that has been widely used and worked on by a large community.

With all of this in mind, an architecture was developed that made sure compromises are kept to a minimum and a good balance is reached.

1. Docker
    I Gitlaw - docker structure
    II frontend
    III backend
2. External microservices
3. Scrapping
    I process
4. Search function
    I Grep
    II Reverted indexation

5. VCS

    I Introduction - a database based VCS

    II VCS Flow

    III Comparison to git

6. Final Platform

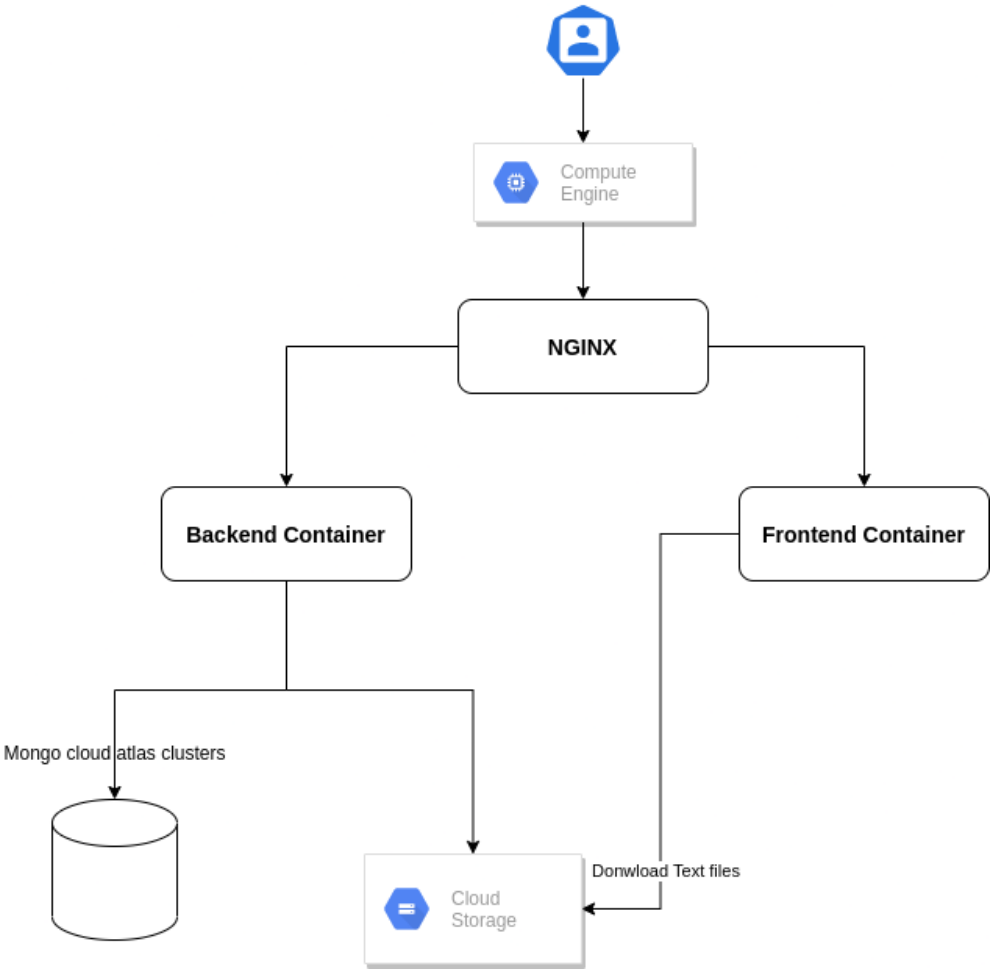

Figure 5.1: Platform architecture

5.1 Frameworks

5.1.1 Frontend

ReactJS was the chosen framework primarily due to it being the standard of the industry at the moment of making this dissertation as it is with ReactJs that most platforms are developed with. However, more than that, it was due to the fact that it is an incredibly powerful javascript framework that allows for everything between quick prototyping to full production ready websites, like facebook which is built with reactJs and it was facebook themselves that developed reactJS.

ReactJs achieves a much-needed degree of performance, especially with how complicated platforms keep getting, by not manipulating the real DOM, which stands for Document Object Model and it is the tree of elements that exist within a platform, and by, instead, having a representation of the real DOM in memory, this being called VDOM, which stands for virtual Document Object Model. This is useful as react can only update the part that suffered from a variable change, for example, instead of re-rendering the entire DOM, which is really resource intensive and is responsible for slowing down any application that does not employ a solution like this.

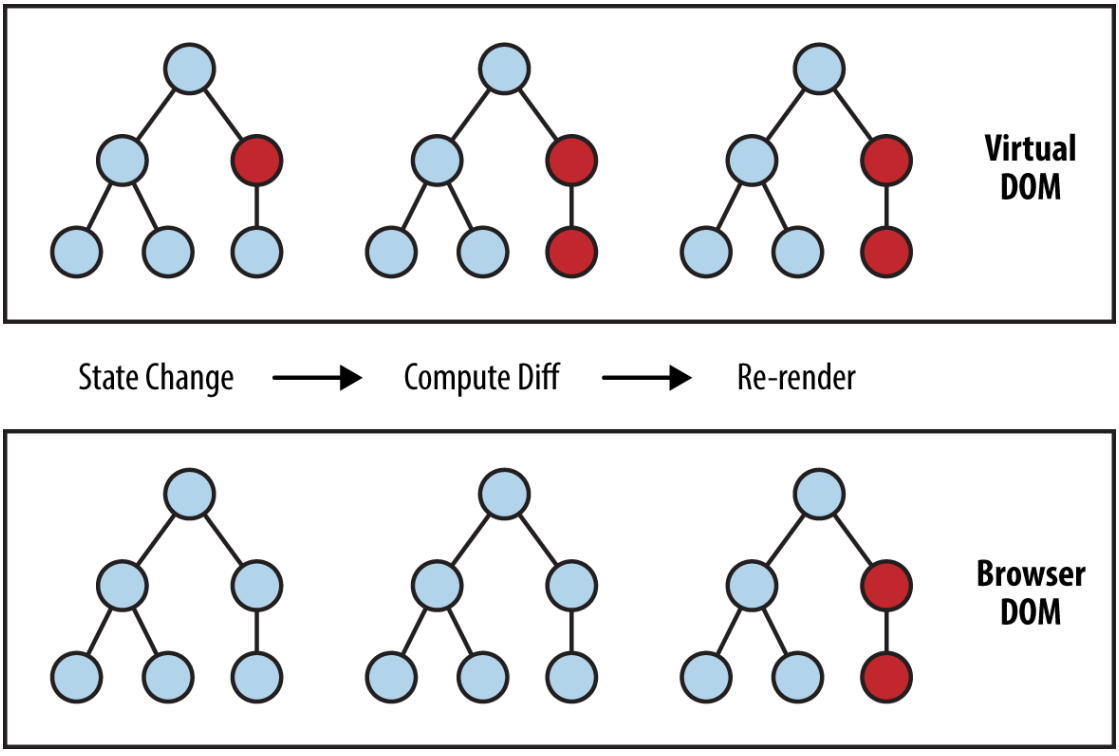

Figure 5.2: VDOM with DOM diagram

On top of this, it is also incredibly easy to work with as it employs a few core concepts that allow for easy separation between the View component from the model and controller component, especially when used in conjunction with redux, which we'll study later, and allows for a clear way to reuse components which save on development time and helps with the debugging process.

One other relevant aspect is the fact that with react native becoming more and more popular alongside reactJs it would be possible, although not the focus of this dissertation, to easily build a mobile application of this platform as there's a big overlap between the two technologies.

To provide a better user experience there are a few packages in use that have to be pointed out as they are integral to the platform, the number of external packages used was kept to a minimum to try and reduce the size of the final platform which can have a negative impact in the cost and performance of the platform. From all the packages used there are some that stand out either from their usefulness or because they resolve really specific issues encountered during the development process:

- **General component packages** - Some packages were used that emphasize giving the developer a set of components pre-styled that follow design guidelines like those employed by the companies that define the path in this area, like google. The main package for this is MaterialUI which provides almost all components that a developer would have to use in a solution like the one developed during the making of this dissertation, this makes it possible to standardize the design elements and makes the development process much faster.

- **Redux** - This package allows for easier state management in any application, this means it gives developers the ability to access a store containing any valuable data like the current logged in user's information, without it data would have to be passed throughout every single component manually, it acts as a singleton in java. Apart from this it also helps in structuring out the code in a better, cleaner way, by giving guidelines and a clear distinction between the logic part of the code and the UI part.

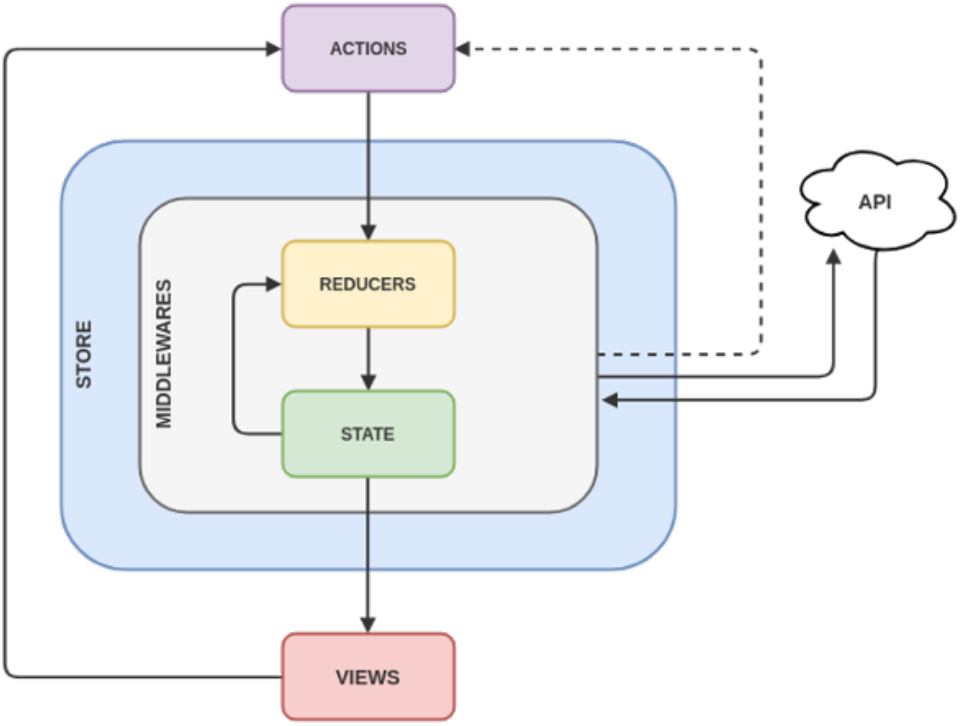

Figure 5.3: schematic of an application with Redux

- **CodeMirror** - during the development of the project one problem was found related to the optimization, or lack thereof, the textFields are not well optimized for big chunks of text (which almost every single piece of legislation is made out of), this translate into lag if a normal textfield is used. CodeMirror is a package that's used by platforms like GitHub for both the preview and edit of files. It manages the outlined problem by breaking the text into smaller chunks and only rendering the lines of text that are in view. It also brings some other features like highlighting, formatting, and text styling (useful for this platform as every text is markdown based), but the performance advantage is the main reason for using it.

5.1.1.1 Backend

For the backend the framework chosen was NodeJs, just like reactJs it is one of the most used frameworks as of making this dissertation. Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

From the development process perspective, this is incredibly powerful as it allows to use of just one programming language on both the frontend and backend, which, in turn, makes the process much easier.

Even though it is single-threaded, it employs an asynchronous, Non-blocking, solution. This makes it so that performance is not put in jeopardy even though only one thread is used as with any event that is called and takes some time to complete, for, example an HTTP request, the server won't be on hold while this action is taking place and does not have to put every other action on hold while the action is not resolved.

One other aspect to keep in mind is that even though the performance of nodejs is incredible when compared with other options available in the market it lacks when it comes to streaming files [18], for this reason this part is handled by an external service made for this type of action, which we'll study later on.

Some integral packages make the development possible (only the most important are contemplated in this paper):

- **express** - This nodejs framework contains a set of tools that make it easier to implement a backend server. The main focus for this platform was the routing feature, this makes it possible to create endpoints and to correctly route any API calls to the correct functions. It also has many other features like middleware which is needed to make the authentication system work, this is due to the fact that with every request the server has to validate the user that made the request, this can be made with a middleware that intercepts each request and does the validation, this saves on development time and makes it so that that check only has to be in one function instead of replicating it throughout each and every single created route.

- **@google-cloud/storage** - AS we'll see later on in the paper a big emphasis was put on the storage system, for this google cloud storage was used. This package is provided by Google and allows for an easy-to-use set of functions that allow for easy manipulation of the google cloud storage buckets of the platform. We'll study why google cloud storage was used later on and what advantages it brings.

- **passport** - For the authentication system the standard for today's platforms is JWT, which allows for the creation of signatures, which are the base of any authentication system and is what allows the backend server to validate a user, thus making sure the user has the needed credentials for accessing the data when a request is made. passport is a middleware that allows for easy integration of this standard. By setting the Authorization header of a request passport will intercept it and check if the user is valid or not.

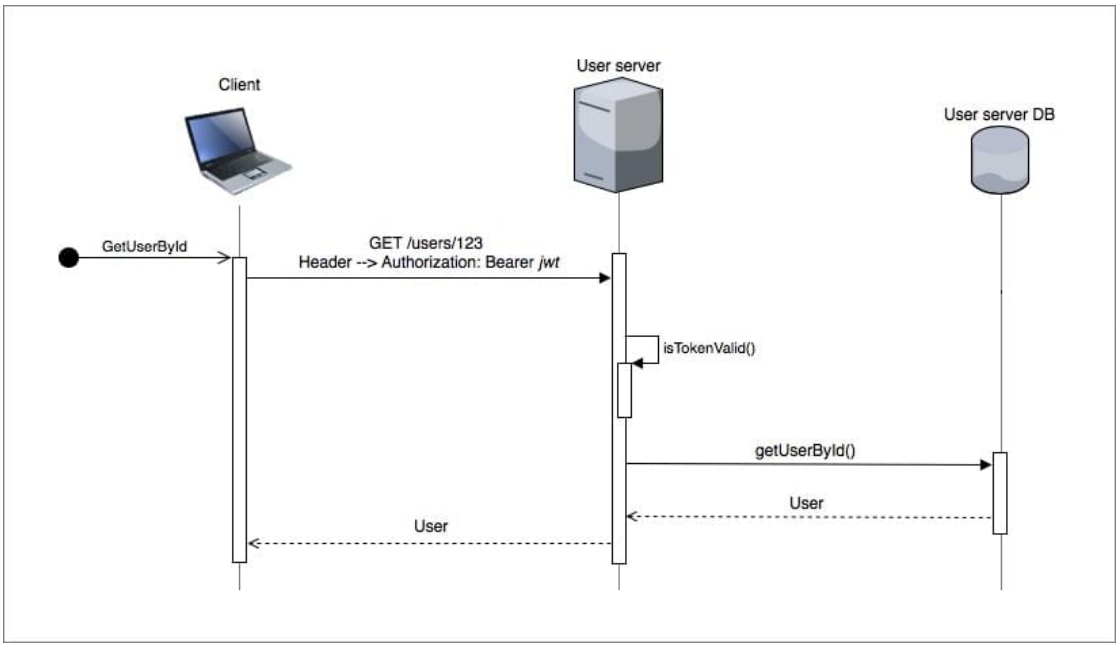

Figure 5.4: schematic of a user jwt token flow

5.2 Docker

Docker has become the standard when it comes to DevOps (even though Kubernetes have been gaining traction over the past few years), for this reason, this platform is all containerized making it easier to scale, test, and develop on.[19]

Application containerization is the act of packaging up all the code and everything that's needed to run it (for example, a react application will need the package.json, node_modules, ...) into a container that typically runs a lightweight and stripped-down version of an Operating System. Typically associated to a normal Virtual machine, even though this is not completely correct it helps to understand the power of a container without having to fully comprehend how it works.

Containerization brings a set net positive aspect to almost every project from which the most important are: [20]

- **Standardization** - by using containerization developers can make sure every single developer is using the same system thus reducing errors when it comes to the development process. In simplified terms, by using containerization a developer can "give" their machine to other developers without actually giving them their machine.

- **Isolation and security** - with containerization we ensure the application does not affect the machine it runs on in any meaningful way, as it acts as a sandbox, even though it isn't a Virtual Machine in this sense it acts like one, or, better yet, as all the advantages as a Virtual Machine.

- **Efficiency** - As containers aren't Virtual Machines, which need the full operating system to work, we get a much smaller package with just what is needed for the application to run and nothing else, therefore containers reduce the use of resources like RAM and CPU thus reducing energy expenditure and, ultimately, reducing cost.

- **Modularity** - By using containerization it becomes possible to segment the project into smaller chunks, microservices, which, in turn, makes it easier to fix problems that arise, as it is not needed to update the entire project, just that specific container.

- **Deploy** - With this technology it becomes possible to deploy an entire application (frontend, backend, database,...) in one server has docker will in sum, separate each part that has to be standalone into containers, which makes it possible to run a large number of services in just one server.

Containerization brings on a set of many other useful characteristics, but these are the ones that stand out, especially with this dissertation in mind.

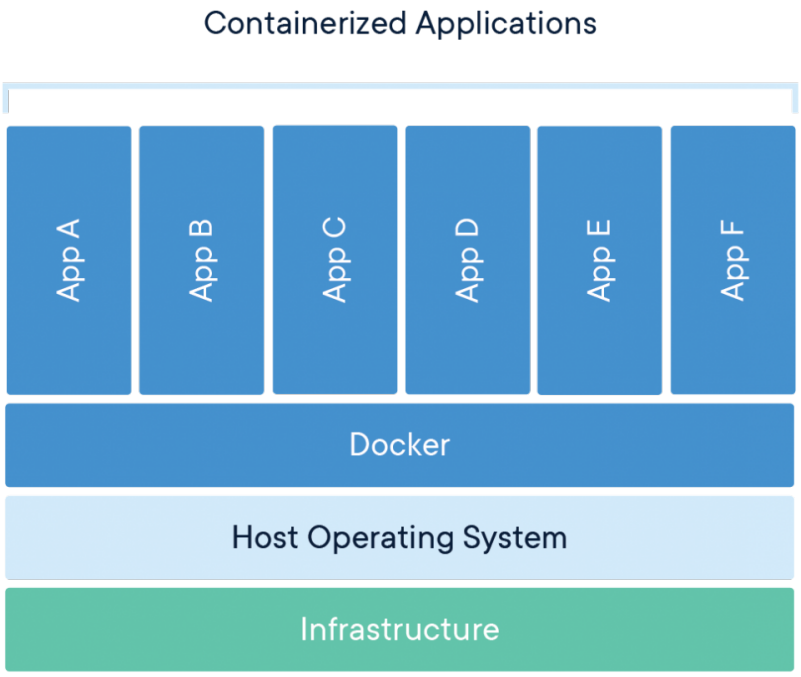

Figure 5.5: containerization of an application schematic

5.2.1 Gitlaw - docker structure

For this dissertation two main containers are used, one for the frontend and one for the backend. As for both storage and the database external services they aren't managed by containers.

5.2.1.1 Fontend

The front end is all made using ReactJS. the container runs a node image running the latest version. The docker file only imports the image and copies the folders needed from the basic react project created with the following command line:

```
1  $ npx create-react-app gitlaw
```

The Dockerfile is, for this reason, extremely simple:

```
# base image
FROM node:latest

# set the working directory
WORKDIR /app

# install and cache app dependencies
COPY package*.json ./

RUN npm install --quiet --force


COPY ./src ./src
COPY ./.env ./.env
COPY ./public ./public

# start app
CMD ["npm", "start"]
```

5.2.1.2 backend

The backend of the platform is also containerized and runs NodeJS the docker file is as follows:

```
# base image
FROM node: latest

# set the working directory
WORKDIR /app

# install and cache app dependencies
COPY package*.json ./

RUN npm install --quiet

COPY ./src ./src
COPY ./config ./config
COPY app.js ./

RUN npm install -g nodemon

RUN npm -g config set user root
```

```
19
20  EXPOSE 6200
21
22  # start app
23  CMD ["npm", "start"]
```

5.3 External microservices

This platform relies on a couple of external microservices for both the storage and database. By employing this tactic it is possible to reduce development time and to scale the project, if need be, in a much more agile way.

This project uses two external services:

- **Mongo cloud atlas** - This service allows for easy deployment of a mongo based database in the cloud. it makes it possible to scale the database depending on the use flow of the project, which makes it a reasonability inexpensive solution and makes it easier to share the same database throughout multiple servers, for example, if the load on the backend container is on the limit of the VM's capacity it would be possible to just start a new VM (scale horizontally) and easily share the same database.
- **Google cloud storage** - For the storage services it was incredibly important to make sure it was decentralized. Mostly for security reasons, this is a critical part of the platform as the legislation of any type of government institution is not something we can afford to lose for obvious reasons. google cloud storage is based around objects, meaning it does not run a regular storage system as we have on operating systems, this brings some challenges especially when it comes to text search as it is not possible to run operating system commands, however by keeping a copy of the master branch always updated in the backend container, but we'll see how this is done later on.

5.4 Scrapping

Scrapping is the process of extracting data from websites or web services, in an automated way. This allows for the quick aggregation of data seamlessly and parsing it into a usable structure by the service a developer is developing. The legality of this is highly debated, however, if the information is not behind any paywall or no restrictions are put into place, if the service a developer is scrapping is not put in jeopardy, it should not harm the service that's being scrapped. [21]

To be able to get the current legislation a scrapping script was written. Again, this was due to the lack of accessibility of information within the current system. dre.pt was scrapped and, because this dissertation is meant as a proof of concept, not everything was fetched (currently only around 4 months worth of legislation is available within the bucket, this equates to around 500 files worth of legislation). However, the code is ready and able to get everything (as long as it is within dre.pt) if need be.

For web scrapping to be possible one truth must hold throughout every goal element within the service, which the repeatability and standardization of elements within the web service, this means that every item that is being scrapped should have the same structure, or, at the very least, a finite amount of options so that every deviation is possible to control. dre.pt checks the points in this paragraph.

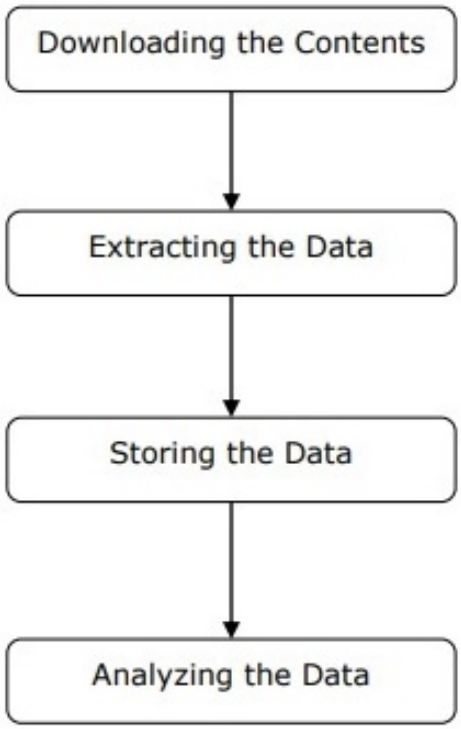

Figure 5.6: schematic of a web scrapping flow

5.4.1 process

The first step of the process is to find out how dre.pt is structured to develop a scraping strategy. For this specific website, the logic and structure are pretty simple. It is divided into:

- **Search page** - In here a user is shown a list of legislation approved by date, however, the information is limited to title, small description, and a link to the route that contains detailed information on the legislation. The URL format is https://dre.pt/web/guest/home/-/dre/calendar/normal/I?day=year-month-day
- **Information Page** - This page is responsible for displaying all the specific information of a piece of legislation like name, date, emitter, proponent, and the legislation itself. The URL of this page is irrelevant to be able to scrape this website

With the initial investigation completed it is important to evaluate how the script for this action works and all the tools needed to make sure the process is as easy as possible without compromising functionality.

Python was the chosen language since the best packages for this type of operation are written with it. the main one being BeautifulSoup which allows for the parsing of HTML into an easy to work object, this is important because the main goal of web scraping is to find the specific tags that contain the needed information and this package is made specifically with that in mind.

The first part for this specific scrapping script is to go through a range of dates which will allow the search URL to get populated and iterated over, the start and end variable correspond to the date from which we want to start getting the legislation from and the end date the last date to get the legislation from.

```
1 datelist = pd.date_range(start="2020-05-28", end="2021-05-28")
2 for item in datelist:
3     scrapeDate(item._date_repr)
```

Listing 5.1: Iterate through all wanted dates

As we can see for each date a function called scrapeDate is called, this function will make a get request to the search page using the date from that iteration and get the needed information for each item displayed.

```
1 def scrapeDate(date):
2     url = baseURL + date
3     print(url)
```

```
4
5       r = requests.get(url)
6
7       soup = BeautifulSoup(r.content, 'html5lib')
8
9       table = soup.find('ul', attrs={'class': 'list'})
10
11      for main in table.findAll('div', attrs={'class': 'diplomas'}):
12          for row in main.findAll('li', attrs={}):
13              scrapeItem(row.a['href'], date)
```

Listing 5.2: Fetch search page data

With the above function, a URL for the information page of each item fetched is parsed into a usable string it is then possible to scrape that URL to get the full information set of that specific legislation with the scrapeItem function.

```
1  def scrapeItem(url, date):
2      r = requests.get(url)
3
4      soup = BeautifulSoup(r.content, 'html5lib')
5
6      main = soup.find('li', attrs={'class': 'formatedTextoWithLinks'})
7      title = soup.find('h1', attrs={'class': 'diarios-diplomas-title'}).text
8      type = soup.find(
9          'li', attrs={'class': 'tipoDiploma.tipo'}).text.split(':')[1]
10     emitter = soup.find(
11         'li', attrs={'class': 'emissor.designacao'}).text.split(':')[1]
12     proponent = soup.find(
13         'li', attrs={'class': 'entidadeProponente'})
14
15     if proponent == None:
16         proponent = "No data"
17     print(type)
18     textHtml = main.find('div')
19
20     title.rstrip()
21
22     filename = title.replace(" ", "_").replace(
23         "  ", "").replace(".", "").replace("/", "_") + '.md'
24     folder = filename.split('_n_')[0]
25
26     if not os.path.exists(join('master', folder)):
```

```
27          os.makedirs(join('master', folder))
28
29      filepath = join('master', folder, filename)
30
31      markdown = md(str(textHtml))
32
33      with open(filepath, 'w', newline='') as f:
34          f.write(markdown)
35
36      sendToCloud(join('master', folder, filename), date, proponent, emitter)
```

Listing 5.3: Fetch information of legislation

By now all the Information is on our side, it is then a matter of saving it to the google cloud storage bucket in the correct folder, which is handled with the sendTocloud function.

5.5 Search function

One of the conclusions out of the multiple validation processes was that the search function was one of the most relevant features of this platform, due to this a lot of time and effort was put into making sure it would work as optimally as possible. the search function is a mix of full-text search over all the files in the master branch, the date from when the files were created and the root folder of the files to search over.

The full-text-search function is the only one that is not trivial. Two strategies were tried for this effect, a GREP call, and inverted indexation, both were analyzed, tried, and tested to ensure the best outcome possible. Lastly, with the results gotten from the tests done, inverted indexation was the chosen strategy as it was the one the gave better results and was the most efficient one.

5.5.1 GREP

GREP is a command function developed for UNIX systems that aims at searching multiple files for a specific pattern, whether it is a string or REGEX. It has a set of input options but, for the sake of this dissertation, they are not relevant so they are not going to be analyzed. GREP is a simple automaton. Brian Kernighan was the developer that implemented in the late 1970s. [22]

Let's take as example an input "Gitlaw: VCS Based Platform", let's say we want to find this string within a big directory with a set of files of varied size. For this effect one would run the following command:

```
1   $  grep -rnw '/path/to/somewhere/' -e 'Gitlaw: VCS Based Platform'
```

Listing 5.4: Grep file search

This command will return each occurrence of the input string with the path to the file where it exists, the line in which it exists, and the line of text where it exists.

Now let's analyze how the source code from a high-level perspective, instead of starting the search from the beginning of the input, GREP starts it from the last character of the input text and compares it with the character in the position equal to the length of the input text, let's say in the first iteration it isn't a match, then GREP moves to the next character of the file's text until a character that matches the last character of the input text is found then it works backward and moves the character of both the input text and the file's text to the last character position minus 1 and does this recursively until a full match is found. [23]

This solution will work, however, it has two major drawbacks. It is slow (depending on the size of the input text and the number of files within the repository) and it does not do full-text-search (meaning it will only show a match if a substring of file's text contains the same string as the input text). For a large-scale application like the one developed in this dissertation, this solution is not powerful enough.

5.5.2 Inverted Indexation

This solution is employed by all the major search engines like google (with a varying degree of complexity). The major amount of the work for this solution is done in the prepossessing of the files where indexation of each word is made and then the search query is made over the resulting data tree that is generated out of that process.

On a basic level, inverted indexation search for a directory of files works by iterating over every single file in the desired directory and indexing each word on the file (every word being every set of characters surrounded by spaces or any special character), this indexation process is based around storing the position of each word and the file where it is contained into an object. It is similar, in its essence, to a HashMap.[24]

However, some optimizations can be made in this prepossessing step. It is possible to black-list some words like pronouns, determiners, and every special character which doesn't provide relevant information in a full-text-search action, these are, for that reason, not added to the list. One other optimization made is to reduce every word to its most simple form, for example, words in the plural form are parsed into their singular form. All of these optimizations are made to reduce the size of the final indexation object.

This indexation has to only be done when the server is started, meaning there is a big overhead when starting the server however it makes the search query extremely fast, with the size of the directory being searched over having little impact on the execution time.

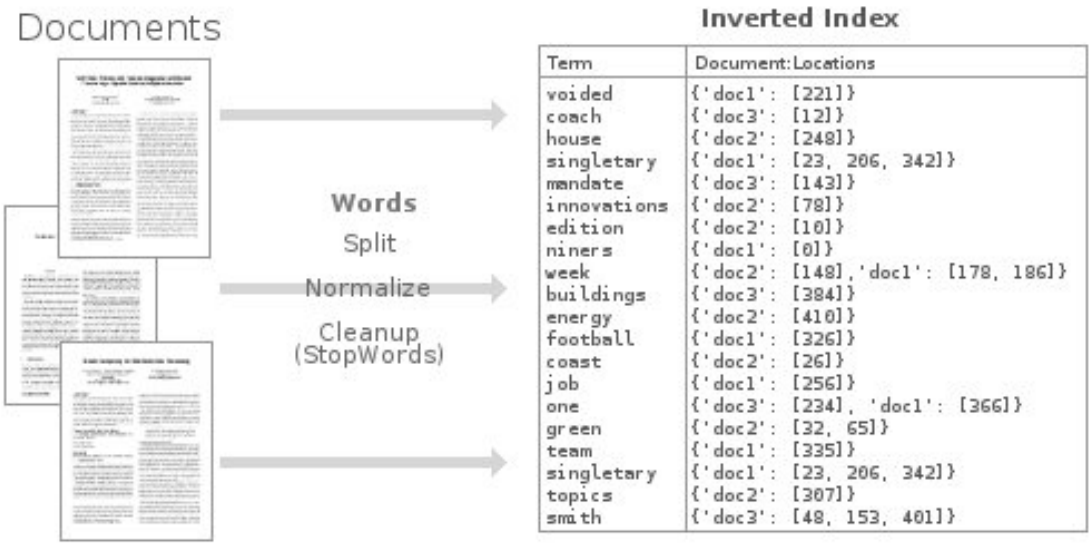

Figure 5.7: Inverted index flow schematic

Then, when it comes to the search action itself, the input text is divided into words, and a search is made for the word within the index table, then a score is given according to how many times it appears within each file and how close it is to other words of the input text. With this score, a hard limit is set to only display the files that are above a certain threshold. The next step is to cross-compare each resulting file to other search parameters.

5.6 VCS system

An integral part of this platform is the VCS system. Some pre evaluations were made to see if was possible to use systems that already exist and were studied in the State of the art.
The final decision was to built it from the ground up with google cloud storage in mind. In this chapter we will evaluate and explain the system and make a comparison to GIT specifically, even though both are completely different as the needs for this platform are really specific.

5.6.1 Introduction - a database based VCS

One big part of the reason why none of the existing VCS systems were used was not only due to the specifications of the project but also because modularity and ease of personalization is an important aspect as different government entities have different requirements.
In the first part of the thesis a study was made on the existing VCSs, specially GIT, for this a big part of the source code was read and studied.
Due to the fact that it has a lot of intrinsic parts, the conclusion was reached that to make any personalization to it was incredibly time consuming, so a different approach was taken.

With all of this in mind, instead of making a regular VCS, a VCS controlled by a database was made.
This means the history of a specific file is kept at all times in a database entry. The google cloud storage serves as a mere saving point for every file, even though some aspects of the VCS are kept within each bucket's object.

Anytime a proposal is created an entry for it is created in the database, then the id of the proposal's object is used as the main folder for it in the GCS's bucket, this folder has a placeholder file called info.json, this is needed due to the limitations of cloud storage services, as they are based around objects it is not possible to have empty folders.

The master branch is the only one that has a specific name.

After the creation of the proposal the creator will get admin privileges, which gives the user the possibility of:

- Add contributors to proposal
- Change state of proposal.

In order to display a proposal's dir list, a merge is made with the master branch any time a get request is made. this means if a change is made to a file in the proposal, that file is saved to the proposal's folder in the GCS's bucket, after, when a list Directory request is made for that proposal the function gets all the objects of the proposal and of the master branch, then a merge is made between both of them. While iterating the master's files array a check is made to see if the file in that iteration exists within the proposal's file array, if it does then it is replaced within the master's files array, in the end if a file exists in the proposal's files array that does not exist within the master's files array it is added to the later and then returned in the request.

Then we have to study how a file save is made. This is because there is a specific case that makes it so that a simple save is not enough. Let's take a practical example, contributor A starts editing file A, while this happens, contributor B edits the file A and saves it, contributor A then

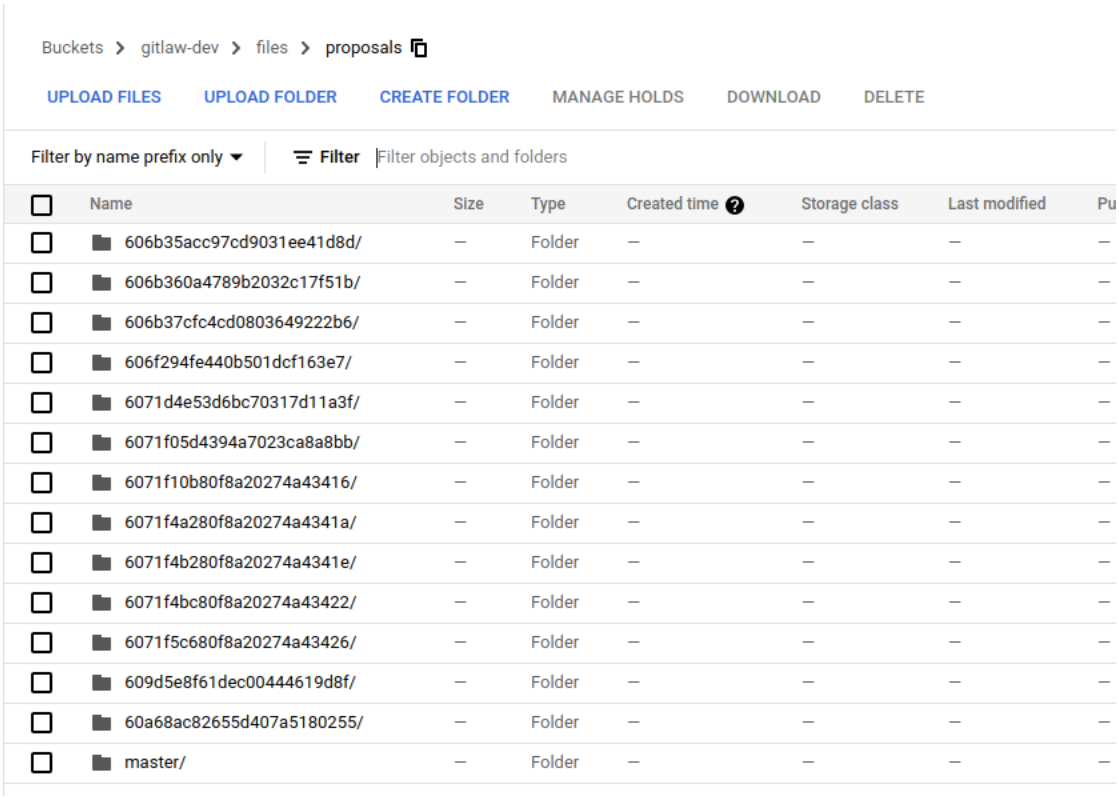

Figure 5.8: structure of branches on GCS

finishes and tries to save it. As now the version of file A has changed contributor A cannot save the file as this action would override the changes made by contributor B. So a process of merge conflict is started. the save request made by contributor A is denied and a list of the differences between the file A's version that's currently on the GCS's bucket and the contributor's A version is shown, allowing contributor A to pick between both of them for each change that's detected. A merge conflict process anytime the last update time that's in the metadata of the file is different from the one the current contributor's file that tht contributor is trying to save.

Finally we have the scenario where a proposal is approved and a merge to the master branch as to be made. For this action the operation made is just to move the files from the proposal's folder to the master's folder, overriding the files in the master's folder.

5.6.2 VCS Flow

The flow for this project is specific for a general use case. so it is really simple with just a depth of two. All the proposal branches have as a parent the master branch.

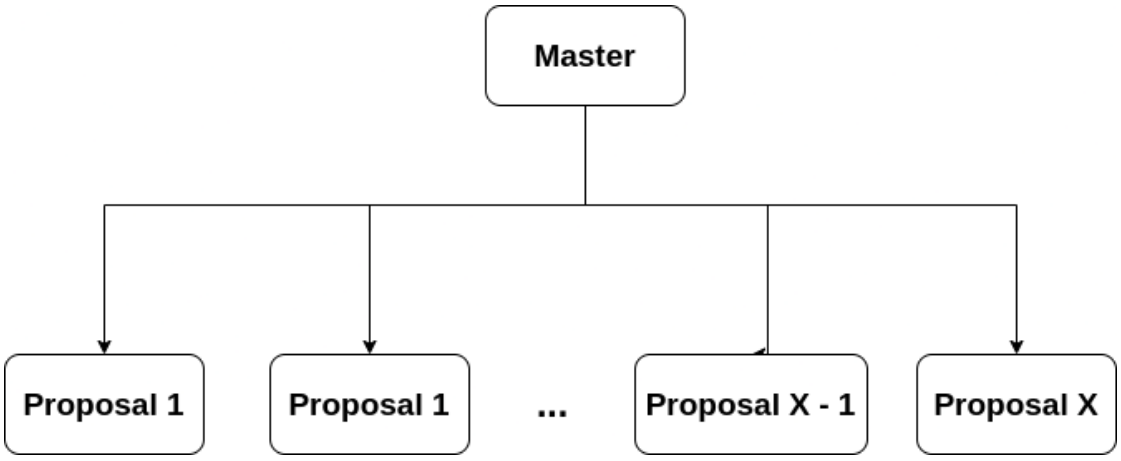

Figure 5.9: VCS flow diagram

Plans have been made to improve upon this in the future which, due to the current VCS structure would be reasonably easy to implement. However this will be studied in the conclusion chapter of this dissertation.

5.6.3 Comparison to git

This implementation, although sharing some similarities to GIT, it has some really clear differences. Which would be interesting to study and analyze, specially when it comes to the features of both.

All of the loss in features is justified for the fact that the platform is meant to be easy to use for non technical users, meaning some functionality was let go of for ease of use, and because developers have different needs, which are generally more complicated.

- **Branches** - The first big difference is the fact that a branch can only be created with a depth of one, meaning a user can only create a proposal from the master branch instead of being able to create nested branches.
- **Command line tool** - This project does not have any CLT available, again, this was due to the fact that it is meant for non-technical users and a CTL is something that most users would not feel comfortable with and would not be able to take advantage of.
- **Offline** - As it runs only in the browser it is not possible to work on a proposal offline with the platform.

5.7 Final Platform

With the architecture of the platform along with every aspect that's more technical studied in this chapter, this section will be dedicated to the user experience of the platform and how everything is connected and, thus, giving an overlook for how the platform works and what users might experience when using it.

In the following subsections, the platform is going to be displayed from the perspective of an official account, as this one displays most of what the platform offers.

5.7.1 Home Page

This page is responsible for giving the user the full directory of the master branch along with the vote information.

Each vote the appears corresponds with a merge from a proposal to the master branch and contains a summary about the proposal along with n expandable view that when in view displays the vote count for that particular proposal.

The endpoint for this page is *domain/*

5.7.2 Search Page

This page is responsible for giving the user the ability to search for files providing some tools that are more powerful than those given by just searching on the home page where you can only see the full directory.

The search function includes three parameters:

1. **Full-text-search** - will use the inverted indexation algorithm to find which documents contain the keywords and rank them from most to least relevant;
2. **Creation date** - range between two dates;
3. **root folder** - folder from which the search should start;

The endpoint for this page is *domain/search*

5.7.3 File Page

From the two pages above and from the proposal page (which acts similarly to the home page) we can click any file that appears and the user will be routed to the file page.

This page is responsible for almost all the actions possible to be made with files:

The endpoint for this page is *domain/file/:path/*

1. **Viewers** - Actions shown to users with and without admin privileges over the file in question (if it is a file in the master branch then this will appear to all users no matter the contribution status to a particular file);

    I **View File** - Users will be able to view the contents of the file with the markdown style pre-applied to the text;

    II **File History** - View file history;

    III **Forum** - A comment section for users to make questions or suggest alterations to the file in the forum like page;

2. **Contributers** - Actions are shown to users that are actively contributing to the file in question;

    I **Edit File** - Users will be able to edit the file in questions using codeMirror which will allow users to have a good experience and only need the browser to do the needed alterations;

    II **Resolve Merge Conflicts** - If, while saving a file, a merge conflict is detected then a merge conflict list will appear allowing the user to resolve any conflict that exists;

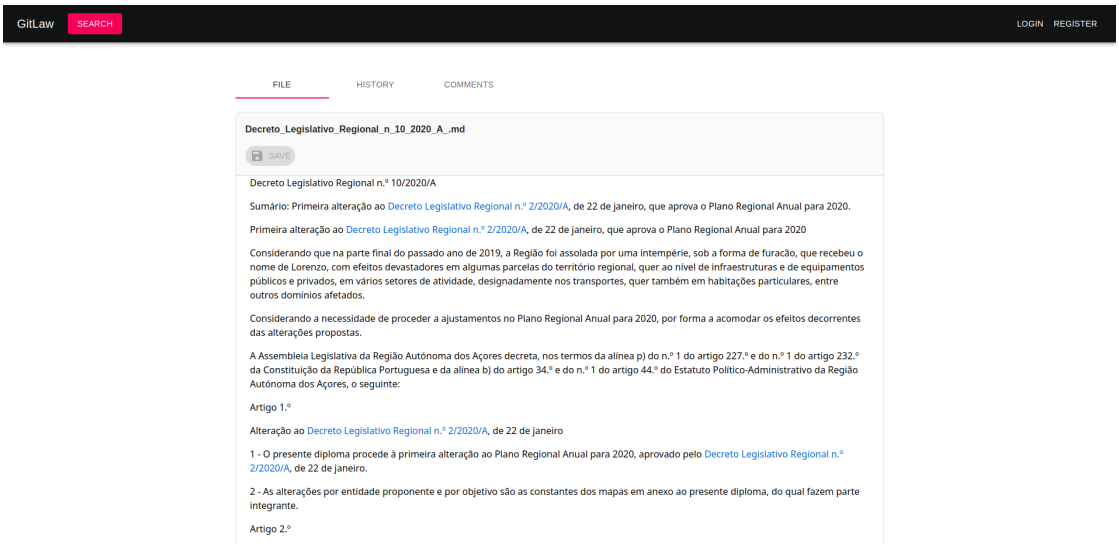

Figure 5.12: File Page

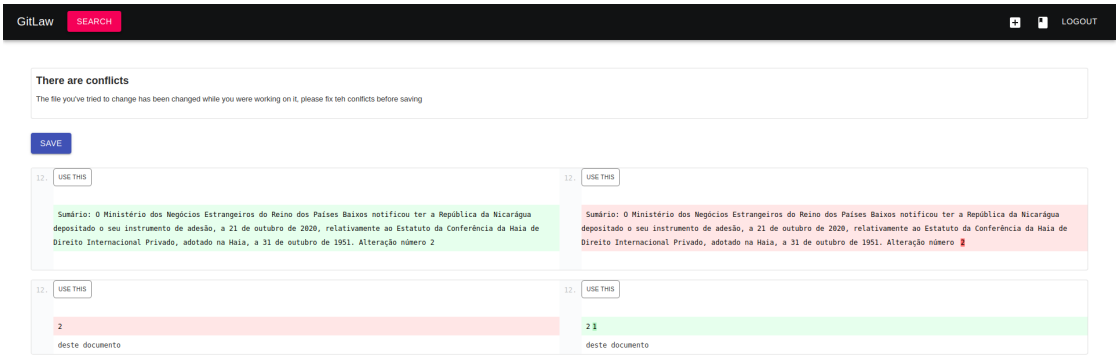

Figure 5.13: merge conflict Page

5.7.4 Personal Page

In this page the user is able to view all the information related to the user's account and view all of the proposals the user is contributing to, for this a simple search function was created that allows the user to easily find the needed proposals.

The endpoint for this page is *domain/archive/*

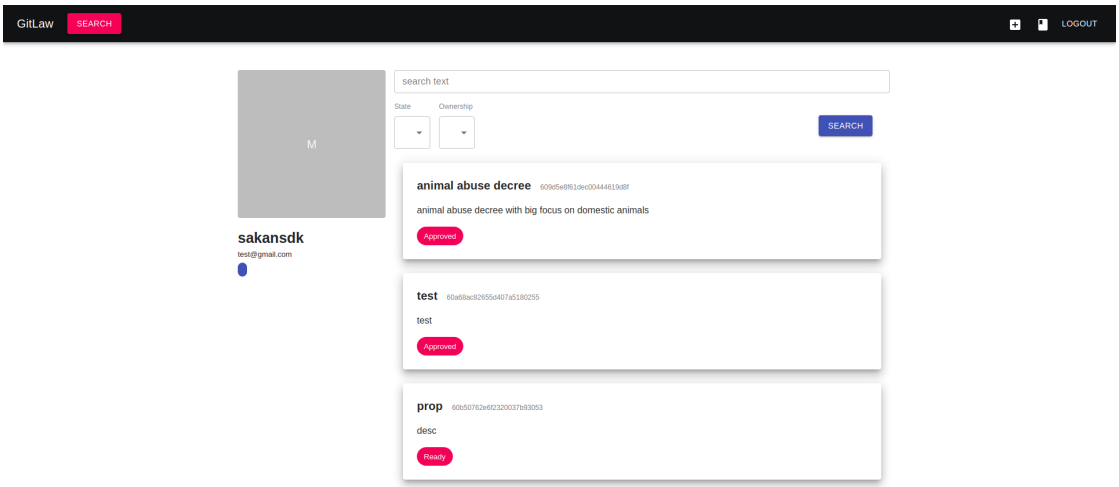

Figure 5.14: Personal Page

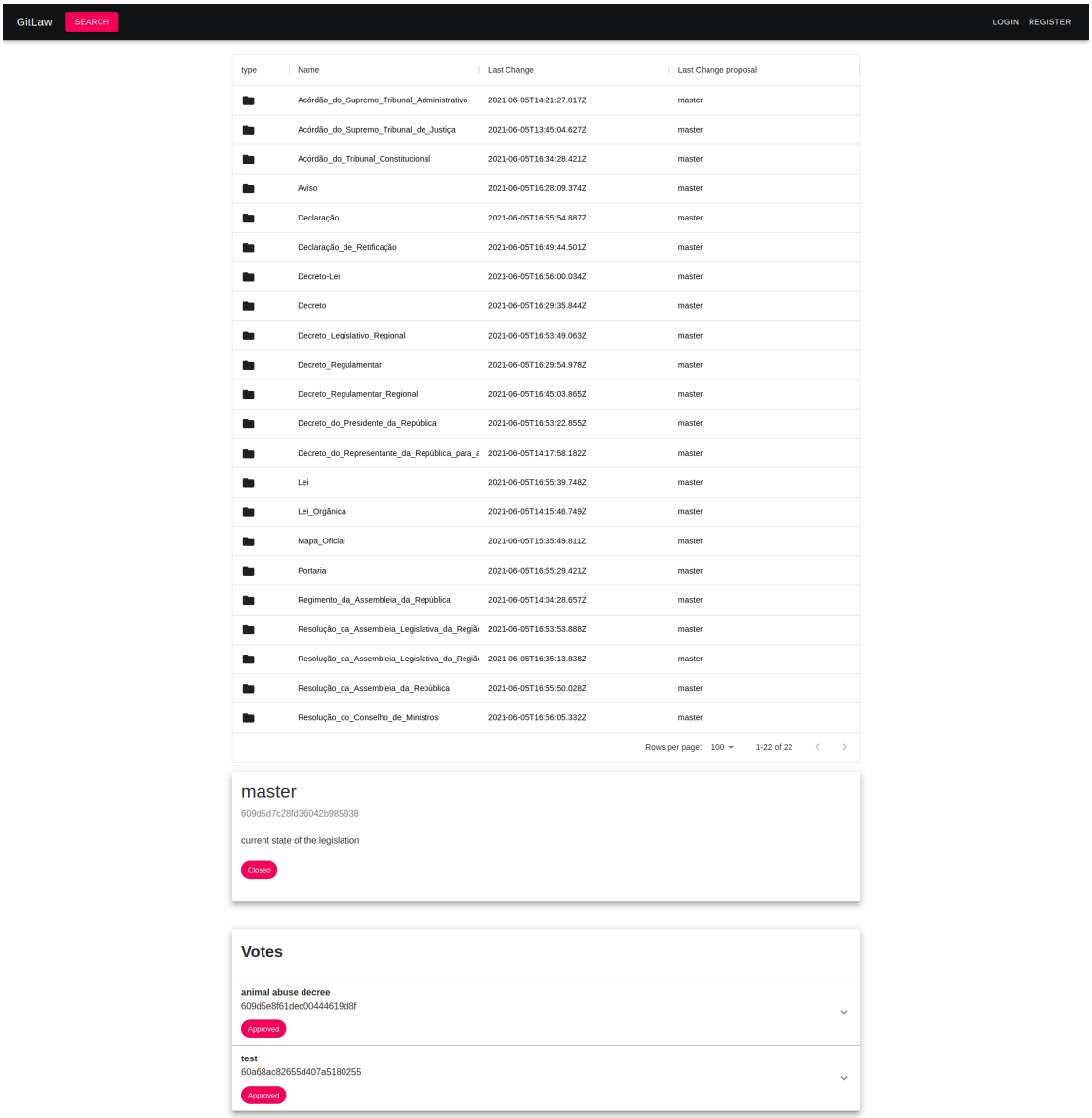

Figure 5.10: Home page

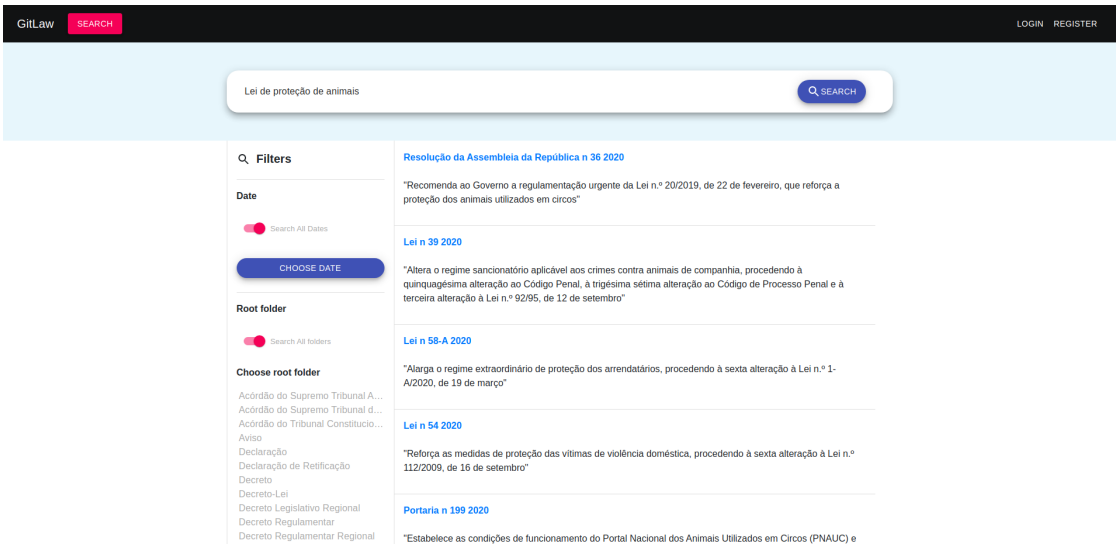

Figure 5.11: Search Page

Chapter 6

Validation process

For any project like this, it is extremely important to continuously validate the product so that it is possible to make sure the target users have a say in what the project will be and what direction to take in most aspects of it.

The main challenge arose from the lack of opportunities given the pandemic state of Portugal at the time, which made it hard to do validation tests especially for the first phase of validation.

For this dissertation three validation steps were planned in the beginning stages of the dissertation process:

1. Test Users
2. First validation
    I methodology
    II results
3. Second validation
    I methodology
    II results
4. Conclusion

6.1 Test Users

Test users are the main focus of the validation process, it is important to remember that in Scrum the input of the end-user is the central point of the development process. For this reason, a set

of users were chosen from a wide range of professional backgrounds. A big effort was made to try and make sure there was at least one test user for each user role defined earlier. With this in mind, the test users were divided as follows:

1. **Admin** - due to the fact that it would have been nearly impossible to get a hold of the person responsible for this action as this elected official is the number two of the country right after the president of Portugal, a replacement had to be chosen, this would have to be someone that has deep knowledge on the legislative process. For this one person from public administration was asked to participate in the process.

2. **Official** - for this set of test users two users were chosen that use the current implementation of the republic's diary, one a journalist and another Historian. Both of the professional lives of these two test users require almost daily use of the current platform.

3. **Citizen** - For the citizens' role it was important to have both test users that never use the republic's diary and some that use it from time to time. For this effect, seven test users were picked at random.

All of this test users played a part in the second validation part however only half participated in the first one, but both the admin and official roles' test users participated in both.

6.2 First validation

The first validation point was made before actually starting writing any code. The goal was just to validate the idea and to see what direction it should take. For this, a small group of potential users was picked and given a Github repository that closely matches what the idea of the project was at this point.

6.2.1 Methodology

The building of a Git project brought a set of problems due to the way the current legislation is made available online (which exacerbated the need for a better solution like the one proposed in this paper), as every legislation is in PDF it is not trivial to get a large set of files to do this. However, it was possible to find a website that provided a small subset of legislation in DOC format which makes it possible to parse into a TXT file and then into markdown, which is the end goal for every file in the platform as it makes it possible to get stylized text easily, this website is verbojuridico.net.

The first step with any validation test is to explain the reason behind the dissertation and give the users a brief explanation of how a VCS system works, with a bigger focus on Git.

Different degrees of help were given during the tests as most test users didn't have any previous knowledge about git or, in this case, GitLab, so some pointers were given to help make the process less tedious.

The idea was to give users a directory with some legislation and ask them to do some tasks:

- **create a branch from master** - Test users were asked to create a branch out of the master branch.
- **Edit legislation** - Test users were asked to edit a piece of legislation from their proposal branch.
- **Submit a pull request** - Test users were asked to submit a pull request to the master branch.
- **Search for text within the master branch** - Test users were asked to try and find a specific piece of legislation using Gitlab's search functionality.

These tests were carried out over a week and gave some really important insights into what users would value most and if they find the solution a net improvement over the current solution employed by the Portuguese government.

The test users were picked from a wide range of professional backgrounds with different degrees of involvement with the legislation process and the legislation itself. Keep in mind that the goal of this dissertation is to make legislation and the LP an approachable subject so it wouldn't be enough to validate it with just those that work with it daily.

6.2.2 Results

From the pool of 5 users, the feedback was overall positive. As all of them said it would be useful and a net improvement over the current system.

Test users were asked at the end of the tests what they valued the most, the most important features were as follow:

- **Search function** - Test users said the current search function in the current system is confusing and the search results were sub-optimal, GitLab made it easier to view the search entries.

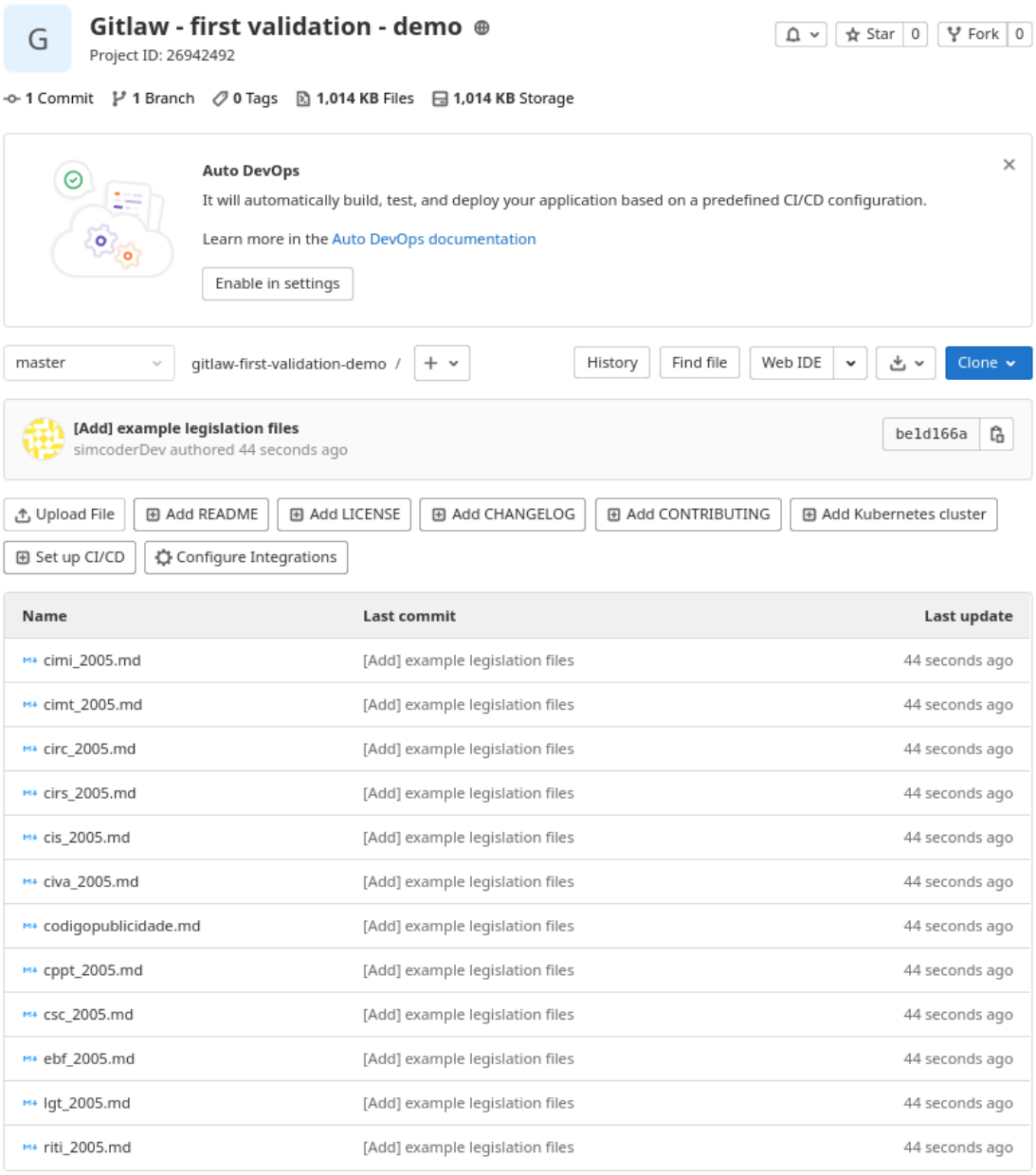

Figure 6.1: View of the gitlab repo

- **View voting with legislation change** - In the GitLab project with every pull request fake voting was placed in the description of the pull request, test users showed real interest in this action as the current solution does not show the votes with each legislation alteration.

- **Folder tree** - Test users said the way that files and folders are displayed in GitLab is better than the one in the current system.

The use of markdown was not pointed out by any of the users as a positive point, however, this might have to do with the fact that there is a learning curve associated with it, for that reason a choice was made to still make it the main type for every file.

6.3 Second validation

the second validation process was done after the platform had been developed, due to some problems it had to be postponed and the third validation process was eliminated as it would have been too close to delivery and without many improvements would have been done over the platform at this stage.

During this validation process, a lot of emphases was put into giving the test users full control of the platform and evaluate how their response was to the platform. The same users that were used in the first validation process were used for this one with 5 more test users.

6.3.1 Methodology

The methodology was the same as it was for the first validation step, the only difference being the platform used for the tests along with the number of files within the platform (this is due to the web scrapping of the dre.pt website, which provided all the legislation that was approved of since the origins of the website in question). Opposite to the first validation process, users were not given any help while using the platform, just a quick introduction to the idea of the dissertation and the reason for existing.

6.3.2 Results

Results were overall positive, with most of the flaws laying with the fact that the platform still had some server-breaking mistakes. Despite that, there were still some criticisms of the platform, it is important to review them and, if not possible to implement the majority of them until the delivery of this dissertation, add them to the future work recommendations. Of the criticisms that were received the most important ones are as follows:

- **Search function** - Test users said, once again, that this function is really important to them and that it would be helpful to have more search parameters like legislation topic, which would allow users to make sure the list of search results is customized to the search query in question.

- **User experience** - As it can be deduced from this paper, most time was spent on developing the backend, because of this, the frontend was neglected. test users made really strong critics of the way the frontend currently functions as it was not intuitive for most.
- **forum** - Test Users said that having a comment section for any file would be useful, transforming the platform into, on top of everything it already has, a forum, this feature was implemented in the middle of the validation process so half the test users managed to use it and give positive feedback on it.

Even though the VCS is the central aspect of this dissertation, most users saw the search function, the ability to view any voting process, and the comment section for each file as the features that they would use the most.

6.4 Conclusions

The validation process was very positive for the development of the platform and for the writing of this dissertation. it was a practical example of how powerful Scrum can be when used right.

For this dissertation, the validation process served as a validation of the initial concept and understood what needed to be improved after the prototype initial was developed.

Even though it was not possible to have a big test user base, this was primarily due to the pandemic state of the country as of making this dissertation, those that were asked to participate were from such different professional backgrounds that, for the context of this dissertation, it is possible to conclude that the results gotten from it are a good representation of what the real user base would be if/when the platform is released.

As for the results of the validation processes, the second one is the most important as it tested the real platform. The results were a net positive, 90 percent of all the test users said they would use this platform and track the legislative process more closely if this platform was in use. And from the 3 test users that had either the citizen or official role all said they would use the platform and all said this would drastically change how they interacted with the current system and would be a drastic improvement in the ease of use of this type of platforms.

Chapter 7

Conclusion and Future Work

In this chapter we will focus on the overall take away from the making of this dissertation, what was learned and what can be improved with future work.

- conclusion
- Future Work

7.1 Conclusion

This research aimed to provide a better solution to the current way most governments allow for the legislative process to be made with citizens in mind so that more trust is put into democracies, the end goal was to improve the democratization of the legislative process and improving the participation of citizens in the legislative process.

Based on a qualitative analysis it was possible to determine that, first and foremost, the premise for this dissertation was correct, being that, the current platform used for this effect (dre.pt) is outdated and not easy to use, lacking on some key features. The results show that the solution presented, although not perfect as it is in the proof of concept state, is a good replacement for the current one as it offers more and better features over the current one along with a better user experience.

It is important to also note that the decision to create a new VCS over using an existing one arose from the fact that, in the first stages of the dissertation, a deep study of the GIT's source code was done. Some problems were then identified, mostly that the tool is, or better yet, has become a really complex project and, due to this, any alteration that would have to be made on it would require significant work to be made. This is relevant because every governmental entity has really specific requisites for how their LP works and how a platform like the one develop

would have to integrate with it, meaning modularity and how easy it is to make changes to the source code is of the greatest importance. On top of the normal GIT objects don't contain all the information needed to get some of the features wanted in the proposed VCS, with each file alteration a large amount of metadata is stored to make sure the process is as transparent as possible and, again, if need be, there's always the possibility of adding more depending on the LP in question.

7.2 Future Work

As the platform developed during this dissertation is a proof of concept at the moment of writing this paper, there are still a lot of aspects to be improved, changed or added:

- **User Experience** - It is imperative that the user experience of the platform is worked on to make sure every user no matter the age, scholarship or level of ability with a computer. To keep in mind the focus for this dissertation was on the backend, primarily the search function and the VCS.
- **Security** - Due to the critical importance of this platform it is advised to do considerable work on this aspect. Currently there is just a basic authentication system and some checks on the backend for the endpoints that are user restricted, not much time was dedicated to this.
- **Modularity** - One aspect that was a goal from the start but was not possible to implement completely (although really important steps were taken towards it) is the modularity of the platform. As different governments and different entities within the governments have different necessities when it comes to the legislative process, it would be incredibly useful to give the entity that deploys this platform the ability to choose a different VCS flow if need be and do any other alteration that's necessary.
- **User customization** - It was suggested to improve upon the customization of each user experience in the validation tests and it is a valid criticism worthy of taking a look at. The idea would be to implement a text recognition engine to detect what each proposal is about (sports, medicine, school,...) and to allow each user to define what matters most to them and if a legislation passes that has attributed to it one of the matters of interest to the user, then the user would receive a notification for it.

Bibliography

[1] git, svn, apache subversion, mercurial - explore - google trends. https://trends.google.pt/trends/explore?date=all&q=git,svn,apache%20subversion,mercurial. (Accessed on 01/30/2021).

[2] M. J. Rochkind. The source code control system. *IEEE Transactions on Software Engineering*, SE-1(4):364–370, 1975.

[3] H. Koike and Hui-Chu Chu. Vrcs: integrating version control and module management using interactive three-dimensional graphics. In *Proceedings. 1997 IEEE Symposium on Visual Languages (Cat. No.97TB100180)*, pages 168–173, 1997.

[4] C Michael Pilato, Ben Collins-Sussman, and Brian W Fitzpatrick. *Version control with subversion: next generation open source version control*. " O'Reilly Media, Inc.", 2008.

[5] Stefan Otte. Version control systems. *Computer Systems and Telematics*, pages 11–13, 2009.

[6] B. de Alwis and J. Sillito. Why are software projects moving from centralized to decentralized version control systems? In *2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, pages 36–39, 2009.

[7] Carl Fredrik Malmsten. Evolution of version control systems-comparing centralized against distributed version control models. B.S. thesis, 2010.

[8] Nazatul Nurlisa Zolkipli, Amir Ngah, and Aziz Deraman. Version control system: A review. *Procedia Computer Science*, 135:408–415, 01 2018.

[9] Christian Rodriguez-Bustos and Jairo Aponte. How distributed version control systems impact open source software projects. pages 36–39, 06 2012.

[10] Git - book. https://git-scm.com/book/en/v2. (Accessed on 01/31/2021).

[11] Thank you for 100 million repositories - the github blog. https://github.blog/2018-11-08-100m-repos/. (Accessed on 02/05/2021).

[12] Plastic scm - features. https://www.plasticscm.com/features?gclid=CjwKCAiAgc-ABhA7EiwAjev-j-SB50U7aOWB1pey3rED1c2yTa4uVlXEY_CSdwVZi-l6vwI7QSP3lxoC6K8QAvD_BwE#graphical-user-interface. (Accessed on 01/29/2021).

[13] How it works | version control & collaboration for word | simul docs. https://www.simuldocs.com/how-it-works. (Accessed on 01/30/2021).

[14] Sónia Carvalho RODRIGUES. *A PRÁTICA LEGISLATIVA PORTUGUESA: QUESTÕES DE CIDADANIA*. PhD thesis, Universidade Nova de Lisboa.

[15] Constituição da república portuguesa. https://www.parlamento.pt/Legislacao/Paginas/ConstituicaoRepublicaPortuguesa.aspx. (Accessed on 01/30/2021).

[16] Ken Schwaber. *Agile project management with Scrum*. Microsoft press, 2004.

[17] Mike Cohn. *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

[18] Ioannis K Chaniotis, Kyriakos-Ioannis D Kyriakou, and Nikolaos D Tselikas. Is node. js a viable option for building modern web applications? a performance evaluation study. *Computing*, 97(10):1023–1044, 2015.

[19] Charles Anderson. Docker [software engineering]. *IEEE Software*, 32(3):102–c3, 2015.

[20] Babak Bashari Rad, Harrison John Bhatti, and Mohammad Ahmadi. An introduction to docker and analysis of its performance. *International Journal of Computer Science and Network Security (IJCSNS)*, 17(3):228, 2017.

[21] Richard Lawson. *Web scraping with Python*. Packt Publishing Ltd, 2015.

[22] Tony Abou-Assaleh and Wei Ai. Survey of global regular expression print (grep) tools. *Citeseer. Topics in Program Comprehension (CSCI 6306). Halifax, Nova Scotia, Canada*, 2004.

[23] fishthe treacherous optimization. https://ridiculousfish.com/blog/posts/old-age-and-treachery.html/. (Accessed on 02/05/2021).

[24] Ajit Kumar Mahapatra and Sitanath Biswas. Inverted indexes: Types and techniques. *International Journal of Computer Science Issues*, 8, 07 2011.