

**DAS** Departamento de Automação e Sistemas  
**CTC** Centro Tecnológico  
**UFSC** Universidade Federal de Santa Catarina

# Gestão de dossiê pertencente ao titular de um Certificado Digital com garantia de integridade em uma Autoridade de Registro

*Relatório submetido à Universidade Federal de Santa Catarina  
como requisito para a aprovação da disciplina:  
DAS 5511: Projeto de Fim de Curso*

*Vinicius Corrêa Menezes*

*Florianópolis, Agosto de 2018*



**Gestão de dossiê pertencente ao titular de um  
Certificado Digital com garantia de integridade em  
uma Autoridade de Registro**

*Vinicius Corrêa Menezes*

Esta monografia foi julgada no contexto da disciplina  
**DAS 5511: Projeto de Fim de Curso**  
e aprovada na sua forma final pelo  
**Curso de Engenharia de Controle e Automação**

*Prof. Joni da Silva Fraga*

---

Banca Examinadora:

Darlan Vivian  
Orientador na Empresa

Prof. Joni da Silva Fraga  
Orientador no Curso

Prof. Hector Bessa Silveira  
Responsável pela disciplina

Prof. Ricardo José Rabelo  
Responsável pela disciplina

Prof. Julio Elias Normey Rico  
Responsável pela disciplina

Dr. Leonardo Martins Rodrigues, Avaliador

Murilo Rodegheri Mendes do Santos, Debatedor

Guilherme Espíndola Winck, Debatedor

# Resumo

A utilização de documentos em papel, com aposição de assinaturas manuscritas ou impressões digitais, constitui prática que atravessa gerações. As civilizações aprenderam, por força da tradição, a conferir status de confiança aos documentos assim firmados. Os recentes avanços da tecnologia da informação e comunicação, contudo, produziram instrumentos adequados à modernização destes procedimentos. Novidades tecnológicas na gestão e segurança de documentos eletrônicos vêm lançando uma nova perspectiva sobre o tema, gerando debates, projetos e iniciativas variadas. No Brasil, a criação da Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil), através da Medida Provisória 2.200-2/2001, representou um marco histórico para a nação ao conferir eficácia probante aos documentos eletrônicos assinados digitalmente. Como resultado, a substituição de documentos físicos por equivalentes eletrônicos vem ganhando atenção crescente no país. A Certificação Digital é a tecnologia que adota mecanismos de segurança, através de algoritmos matemáticos, capazes de garantir autenticidade, confidencialidade, integridade e não-repúdio às informações eletrônicas. O processo de obtenção de um Certificado Digital compreende quatro momentos distintos: a solicitação, a identificação presencial, a validação e verificação dos documentos e a emissão. Primeiramente, o interessado em adquirir um Certificado Digital precisa gerar uma requisição e juntar alguns documentos, tais como cópia do seu RG, CPF e comprovante de residência. Esta requisição é normalmente feita pelo próprio interessado no Módulo Público e encaminhada, via internet, a uma Autoridade de Registro. Posteriormente, o interessado deve se dirigir pessoalmente à AR, levando consigo os documentos solicitados. Na AR, os Agentes de Registro conferem os dados e, se tudo estiver correto, enviam a requisição do certificado para a Autoridade Certificadora, responsável por sua emissão. O trabalho presente se dá neste contexto e visa apresentar o processo de implementação de um módulo no sistema de Autoridade de Registro para o gerenciamento do dossiê necessário para a emissão de Certificados Digitais no contexto da ICP-Brasil, utilizando *frameworks* para a implementação de um sistema cliente-servidor. O módulo foi desenvolvido utilizando Angular para a aplicação cliente e Laravel como servidor, realizando o intermédio com o banco de dados PostgreSQL. A implementação deste módulo visa agilizar o processo de emissão dos Certificados Digitais e facilitar a recuperação dos documentos para fins de auditorias realizadas na Autoridade de Registro.

**Palavras-chave:** ICP-Brasil, Certificação Digital, Segurança, Dossiê.



# Abstract

The use of paper documents, with the affixing of handwritten signatures or fingerprints, is a practice that goes through generations. Civilizations have learned, by virtue of tradition, to confer trust status to the documents thus signed. Recent advances in information and communication technology, however, have produced appropriate tools for the modernization of these procedures. Technological innovations in the management and security of electronic documents have been launching a new perspective on the subject, generating debates, projects and varied initiatives. In Brazil, the creation of the Brazilian Public Key Infrastructure (ICP-Brasil), through Provisional Measure 2.200-2/2001, represented a historic milestone for the nation by providing proofing effectiveness to digitally signed electronic documents. As a result, the replacement of physical documents by electronic equivalents has been gaining increasing attention in the country. Digital Certification is the technology that adopts security mechanisms, through mathematical algorithms, capable of guaranteeing authenticity, confidentiality, integrity and non-repudiation of electronic information. The process of obtaining a Digital Certificate comprises four distinct moments: the request, the presence identification, the validation and verification of the documents and the issue. Firstly, those interested in acquiring a Digital Certificate must generate a requisition and attach some documents, such as a copy of their RG, CPF and proof of residence. This request is usually made by the interested party in the Public Module and sent via the Internet to a Registration Authority. Subsequently, the interested party must go personally to the AR, taking with them the requested documents. In AR, the registry agents check the data and, if everything is correct, send the request for the certificate to the Certification Authority, responsible for issuing it. The present work takes place in this context and aims to present the process of implementing a module in the Registration Authority system for the management of the dossier necessary for the issuance of Digital Certificates in the context of ICP-Brasil, using frameworks for the implementation of a client-server system. The module was developed using Angular for the client application and Laravel as server, performing the intermediate with the PostgreSQL database. The implementation of this module aims to speed up the process of issuing Digital Certificates and facilitate the retrieval of documents for the purposes of audits performed at the Registration Authority.

**Keywords:** ICP-Brasil, Digital Certification, Security, Dossier.





# Lista de ilustrações

Figura 1 – Processo de cifragem. . . . .	21
Figura 2 – Sigilo utilizando criptografia assimétrica. . . . .	22
Figura 3 – Autenticidade utilizando criptografia assimétrica. . . . .	23
Figura 4 – Assinatura Digital utilizando algoritmos de chave pública. . . . .	25
Figura 5 – Conferência da Assinatura Digital. . . . .	25
Figura 6 – Conteúdo de um Certificado Digital segundo padrão X.509. . . . .	26
Figura 7 – Sistema de banco de dados. . . . .	32
Figura 8 – Ferramenta Astah . . . . .	39
Figura 9 – Balsamiq Mockups - Exemplo de protótipo. . . . .	40
Figura 10 – Diagrama de casos de uso - Gerenciamento do dossiê. . . . .	49
Figura 11 – Protótipo - Listagem de documentos. . . . .	49
Figura 12 – Protótipo - Cadastro de documento. . . . .	50
Figura 13 – Protótipo - Edição de documento. . . . .	50
Figura 14 – Protótipo - Exclusão de documento. . . . .	50
Figura 15 – Protótipo - Visualização de documento. . . . .	51
Figura 16 – Modelo Entidade Relacionamento - Documentos. . . . .	52
Figura 17 – Modelo Lógico - Documentos. . . . .	53
Figura 18 – Arquitetura da comunicação entre sistemas. . . . .	56
Figura 19 – Estrutura do backend em camadas. . . . .	57
Figura 20 – Arquitetura da comunicação entre sistemas detalhada. . . . .	60
Figura 21 – Arquitetura MVC. . . . .	61
Figura 22 – Estrutura do diretório contendo a lógica de uma requisição. . . . .	62
Figura 23 – Seletor do componente responsável pelo gerenciamento do dossiê. . . . .	62
Figura 24 – Operação para retorno de dados relacionados aos documentos de uma requisição. . . . .	63
Figura 25 – Postman - Teste de cadastro de documento. . . . .	66
Figura 26 – Postman - Teste de listagem de documentos. . . . .	66
Figura 27 – Postman - Teste de edição de documento. . . . .	67
Figura 28 – Postman - Teste de exclusão de documento. . . . .	67
Figura 29 – Execução de testes unitários. . . . .	68
Figura 30 – Tela - Gerenciamento de requisições. . . . .	69
Figura 31 – Tela - Requisição. . . . .	70
Figura 32 – Tela - Novo documento. . . . .	70
Figura 33 – Tela - Novo documento com campos preenchidos. . . . .	71
Figura 34 – Tela - Editar documento. . . . .	71
Figura 35 – Tela - Visualizar documento. . . . .	72

Figura 36 – Tela - Excluir documento. . . . .	72
Figura 37 – Tela - Confirmar aprovação - Assinatura Agente de Registro. . . . .	73
Figura 38 – Tela - Confirmar aprovação - Carimbo do Tempo. . . . .	73
Figura 39 – Tela - Sucesso após aprovar requisição. . . . .	74
Figura 40 – Tela - Status da requisição atualizado. . . . .	75
Figura 41 – Tela - Gestão de documentos ao realizar solicitação no MP. . . . .	76
Figura 42 – Tela - Gestão de documentos através do uso do protocolo no MP. . . . .	76

# Lista de tabelas

Tabela 1 – Requisições para cada operação presente no módulo. . . . .	59
---	----



# Lista de abreviaturas e siglas

AC - Autoridade Certificadora

ANSI - American National Standards Institute

API - Application Programming Interface

AR - Autoridade de Registro

CG - Comitê Gestor

CNH - Carteira Nacional de Habilitação

CNPJ - Cadastro Nacional da Pessoa Jurídica

Cotec - Comissão Técnica

CPF - Cadastro de Pessoas Físicas

CRUD - Create, Read, Update, Delete

CSS - Cascading Style Sheets

DDL - Data Definition Language

DETRAN - Departamento Estadual de Trânsito

DML - Data Manipulation Language

DOM - Document Object Model

HTML - Hypertext Markup Language

HTTP - Hypertext Transfer Protocol

ICP - Infraestrutura de Chaves Públicas

ITI - Instituto Nacional de Tecnologia da Informação

JPEG - Joint Photographic Experts Group

JSON - JavaScript Object Notation

Modelo ER - Modelo Entidade-Relacionamento

MP - Módulo Público

MVC - Model-View-Controller

MVC - Model-View-Controller

ORM - Object Relational Mapper

PDF - Portable Document Format

PHP - Hypertext Preprocessor

PNG - Portable Network Graphics

PSS - Prestadores de Serviço de Suporte

REST - Representational State Transfer

RG - Registro Geral

SAS - Sistema de Auditoria e Sincronismo

SCT - Sistema de Carimbo do Tempo

SGBD - Sistemas de Gerenciamento de Banco de Dados

SPA - Single Page Application

SQL - Structured Query Language

UML - Unified Modeling Language

URI - Universal Resource Identifier

URL - Uniform Resource Locator

W3C - World Wide Web Consortium

XML - Extensible Markup Language

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>17</b>
1.1	Processo de Obtenção de um Certificado Digital	18
1.2	Motivação	19
1.3	Objetivos	19
1.3.1	Objetivo Geral	19
1.3.2	Objetivos Específicos	19
1.4	Organização do Relatório	20
<b>2</b>	<b>FUNDAMENTAÇÃO TEÓRICA</b>	<b>21</b>
2.1	Criptografia	21
2.1.1	Algoritmos Criptográficos de Chave Pública	22
2.1.1.1	Confidencialidade	22
2.1.1.2	Autenticidade	23
2.2	Assinatura Digital	23
2.3	Certificado Digital	25
2.3.1	Burocracia	26
2.3.2	Segurança	27
2.4	Carimbo do Tempo	27
2.5	ICP-Brasil	28
2.6	Emissão de um Certificado Digital	28
2.6.1	Módulo Público	28
2.6.2	Autoridade de Registro	29
2.6.3	Autoridade Certificadora	29
2.7	Sistemas Distribuídos	29
2.8	Cliente-Servidor	30
2.9	Engenharia de Software	30
2.9.1	Engenharia de Requisitos	31
2.10	Banco de Dados	31
<b>3</b>	<b>MATERIAIS, TÉCNICAS E MÉTODOS</b>	<b>35</b>
3.1	Linguagens	35
3.1.1	HTML	35
3.1.2	Javascript	35
3.1.3	PHP	35
3.1.4	JSON	36
3.2	Banco de Dados	36

3.2.1	PostgreSQL . . . . .	36
<b>3.3</b>	<b>Arquitetura . . . . .</b>	<b>37</b>
3.3.1	REST . . . . .	37
<b>3.4</b>	<b>Bibliotecas e Frameworks . . . . .</b>	<b>37</b>
3.4.1	Angular . . . . .	37
3.4.2	PrimeNG . . . . .	38
3.4.3	Laravel . . . . .	38
<b>3.5</b>	<b>Ferramentas . . . . .</b>	<b>39</b>
3.5.1	Astah . . . . .	39
3.5.2	Balsamiq . . . . .	40
3.5.3	Guzzle . . . . .	41
3.5.4	Postman . . . . .	41
3.5.5	GIT . . . . .	41
<b>3.6</b>	<b>Metodologia . . . . .</b>	<b>41</b>
3.6.1	Scrum . . . . .	41
3.6.1.1	Vocabulário . . . . .	42
<b>4</b>	<b>PROJETO . . . . .</b>	<b>45</b>
<b>4.1</b>	<b>Engenharia de Requisitos . . . . .</b>	<b>45</b>
4.1.1	Documentação de Requisitos de Software . . . . .	45
4.1.1.1	Objetivo do Documento . . . . .	45
4.1.1.2	Escopo do Sistema . . . . .	45
4.1.1.3	Descrição Geral do Sistema . . . . .	46
4.1.2	Requisitos do Sistema . . . . .	46
4.1.3	Requisitos Funcionais (RF) . . . . .	46
4.1.4	Requisitos Não Funcionais (RN) . . . . .	48
<b>4.2</b>	<b>Diagrama de Casos de Uso . . . . .</b>	<b>48</b>
<b>4.3</b>	<b>Protótipos . . . . .</b>	<b>48</b>
<b>4.4</b>	<b>Modelagem do Banco de Dados . . . . .</b>	<b>51</b>
4.4.1	Modelo Entidade-Relacionamento (ER) . . . . .	51
4.4.2	Modelo lógico . . . . .	51
<b>5</b>	<b>IMPLEMENTAÇÃO . . . . .</b>	<b>55</b>
<b>5.1</b>	<b>Arquitetura geral . . . . .</b>	<b>55</b>
<b>5.2</b>	<b>Servidor . . . . .</b>	<b>56</b>
5.2.1	Módulo AC . . . . .	57
5.2.2	Módulo AR . . . . .	58
<b>5.3</b>	<b>Cliente . . . . .</b>	<b>59</b>
<b>5.4</b>	<b>Integridade das informações . . . . .</b>	<b>63</b>



<b>6</b>	<b>RESULTADOS</b>	<b>65</b>
<b>6.1</b>	<b>Testes com Postman</b>	<b>65</b>
<b>6.2</b>	<b>Testes Unitários</b>	<b>65</b>
<b>6.3</b>	<b>Funcionalidades do Sistema</b>	<b>68</b>
6.3.1	Gerenciamento do dossiê	69
6.3.2	Fluxo de aprovação	72
6.3.3	Assinatura Agente de Registro	73
6.3.4	Carimbo do Tempo	74
<b>6.4</b>	<b>Módulo Público</b>	<b>75</b>
<b>7</b>	<b>CONCLUSÕES</b>	<b>77</b>
<b>7.1</b>	<b>Perspectivas Futuras</b>	<b>77</b>
	<b>REFERÊNCIAS</b>	<b>79</b>



# 1 Introdução

A evolução da computação e tecnologia permitiu o avanço em diversas áreas e a criação de diversas aplicações que facilitam e auxiliam a vida das pessoas. Aplicações web têm se tornado cada vez mais importantes em nossas vidas. A conveniência do acesso à qualquer momento, as simples interfaces e o acesso massivo à internet tornam estas aplicações elementos estratégicos para todos os setores da economia.

Atualmente as empresas passam por um momento no qual uma grande diversidade de produtos é lançado no mercado. Historicamente, os bancos são pioneiros em implementar estas alternativas, principalmente para pessoas físicas. Entretanto, quando falamos em pessoa jurídica o universo se amplia e, juntamente com ele, os riscos, as questões de segurança e, principalmente, a forma de comprovação de operações. As vantagens oferecidas pelo universo digital são bem mais atrativas que o universo físico, obrigando empresas a atualizarem seus setores financeiros, contábeis e jurídicos para se certificarem das implicações diretas sobre o seu negócio, sob pena de perderem competitividade ou ficarem defasadas tecnologicamente.

Dessa forma, se acompanharmos a evolução tecnológica da maioria das empresas, verificamos que muitos processos já bastante comuns e rotineiros não possuem muito tempo de existência. As informações constantes em documentos eletrônicos vão de simples trocas de mensagens a dados sensíveis, como números de cartões de crédito, dados bancários e acordos comerciais. Provavelmente, nas primeiras vezes em que foram utilizados estes recursos, muitos ficaram receosos a respeito de sua confiabilidade e ainda preferiram entrar na fila dos bancos para efetuar tais movimentações e obter a comprovação física de suas transações.

Esta troca de informações entre instituições vem se expandindo e crescendo rapidamente, principalmente através das redes de computadores, como é o caso da Internet. A utilização de documentos eletrônicos apresenta vantagens sobre os documentos em papéis, tais como facilidade de administração, cópia, armazenamento e transmissão. O interesse no uso destes documentos eletrônicos foi intensificado quando passou a existir legislação imputando validade legal ao mesmo. Neste contexto, o certificado digital e a assinatura digital são fundamentais para garantir aspectos como autenticidade, integridade e não-repúdio nas transações [1].

Para que isto aconteça, a informação deve estar sempre íntegra, com garantia de autenticidade (original), confidencial (visível a quem tem permissão) e irretroatável (não pode ser negada). A certificação digital é a tecnologia que provê estes mecanismos. No cerne da certificação digital está o certificado digital, um documento eletrônico que contém

o nome, um número público exclusivo denominado chave pública e muitos outros dados que mostram quem somos para as pessoas e para os sistemas de informação. A chave pública serve para validar uma assinatura realizada em documentos eletrônicos.

A certificação digital tem trazido inúmeros benefícios para os cidadãos e para as instituições que a adotam. Com a certificação digital é possível utilizar a Internet como meio de comunicação alternativo para a disponibilização de diversos serviços com uma maior agilidade, facilidade de acesso e substancial redução de custos. A tecnologia da certificação digital foi desenvolvida graças aos avanços da criptografia nos últimos 30 anos.

A evolução tecnológica e o novo embasamento jurídico permitem o emprego de documentos eletrônicos com segurança, confiabilidade e validade, aliviando os sistemas administrativos e melhorando a eficiência destes processos.

## 1.1 Processo de Obtenção de um Certificado Digital

De forma genérica, para a emissão de um certificado digital há a necessidade de três módulos (2.6):

- Módulo Público (MP);
- Autoridade de Registro (AR);
- Autoridade Certificadora (AC).

O processo de obtenção de um certificado digital compreende quatro momentos distintos:

1. **A solicitação:** Primeiramente, o interessado em adquirir um Certificado Digital precisa gerar uma requisição. Esta requisição é normalmente feita pelo próprio interessado no Módulo Público e encaminhada, via internet, a uma Autoridade de Registro;
2. **A identificação presencial:** Após a solicitação, o interessado deverá providenciar os documentos necessários para emissão do tipo de certificado escolhido e agendar uma visita em uma Autoridade de Registro para realizar a identificação presencial;
3. **A validação e verificação dos documentos:** Após a apresentação dos documentos, o interessado deverá, então, aguardar o processo de validação e verificação por parte da Autoridade de Registro. Caso esteja em conformidade, pode-se prosseguir para a etapa de emissão do Certificado Digital;

4. **A emissão:** Nesta fase o sistema iniciará o processo de geração do par de chaves, onde o tamanho e algoritmo de assinatura utilizado dependerá do tipo de certificado escolhido.

## 1.2 Motivação

Atualmente o interessado deve levar toda a documentação até uma Autoridade de Registro, onde os agentes irão analisar e escanear estes documentos, validando ou não a requisição do Certificado Digital. Este processo acaba sendo trabalhoso muitas vezes pelas seguintes razões:

- Se o interessado esquecer algum documento necessário para a emissão do Certificado Digital, o mesmo deverá agendar outra visita portando todos os documentos necessários;
- As cópias dos documentos do titular são muitas vezes armazenados como documentos físicos, dificultando sua localização com rapidez.

Neste escopo, este trabalho tem como motivação a necessidade de desenvolvimento de um sistema para o gerenciamento da documentação eletrônica (dossiê) do titular do Certificado Digital, que foi capturada pelo Agente de Registro durante a validação ou que poderá ser fornecida pelo próprio titular através do Módulo Público.

## 1.3 Objetivos

### 1.3.1 Objetivo Geral

O objetivo geral deste trabalho é a criação de um módulo no sistema de Autoridade de Registro para o gerenciamento do dossiê necessário para a emissão de Certificados Digitais no contexto da ICP-Brasil. O sistema deve garantir a integridade das informações armazenadas e a validade jurídica dos documentos através do uso da Assinatura Digital (Seção 2.2) e do Carimbo do tempo (Seção 2.4). Assim, o sistema provê um armazenamento seguro, garantindo a integridade das informações e agiliza o processo tanto de emissão de certificados quanto de auditorias realizadas na Autoridade de Registro.

### 1.3.2 Objetivos Específicos

Em termos específicos, os objetivos do módulo que será desenvolvido são:

- Os documentos cuja cópia deva constar no dossiê (ex: documentos de identificação apresentados pelo titular) devem ser digitalizados e assinados digitalmente com o certificado ICP-Brasil;
- Todos os arquivos que compõem um dossiê devem ser organizados de forma a permitir sua recuperação conjunta, para fins de fiscalização e auditoria;
- O diretório ou sistema onde são armazenados esses arquivos deve ter proteção contra leitura e gravação, dando permissão de acesso somente aos Agentes de Registro ou responsáveis designados formalmente para trabalhar com os documentos.

## 1.4 Organização do Relatório

Este relatório está organizado da seguinte maneira:

O Capítulo 2 tratará resumidamente de alguns conceitos, técnicas e fundamentação teórica, básicos para compreensão das atividades desenvolvidas neste projeto.

O Capítulo 3 abordará as tecnologias, ferramentas e técnicas que foram utilizadas para o desenvolvimento do sistema.

O Capítulo 4 abordará mais detalhadamente o processo de levantamento dos requisitos para o desenvolvimento do sistema.

O Capítulo 5 abordará a implementação do sistema, utilizando os conceitos apresentados no Capítulo 2 e as ferramentas apresentadas no Capítulo 3, de modo a cumprir os requisitos levantados no Capítulo 4.

O Capítulo 6 mostrará os testes realizados e os resultados obtidos ao final deste projeto.

## 2 Fundamentação Teórica

Nesta seção são abordados alguns conceitos, técnicas e fundamentação teórica, básicos para compreensão das atividades desenvolvidas durante o projeto.

### 2.1 Criptografia

A palavra criptografia tem origem grega e significa a arte de escrever em códigos de forma a esconder a informação na forma de um texto incompreensível. A informação codificada é chamada de texto cifrado. O processo de codificação ou ocultação é chamado de cifragem, e o processo inverso, ou seja, obter a informação original a partir do texto cifrado, chama-se decifragem [2].

A cifragem (Figura 1) e a decifragem são realizadas por programas de computador chamados de cifradores e decifradores. Um programa cifrador ou decifrador, além de receber a informação a ser cifrada ou decifrada, recebe um número chave que é utilizado para definir como o programa irá se comportar. Os cifradores e decifradores se comportam de maneira diferente para cada valor da chave. Sem o conhecimento da chave correta não é possível decifrar um dado texto cifrado. Assim, para manter uma informação secreta, basta cifrar a informação e manter em sigilo a chave.

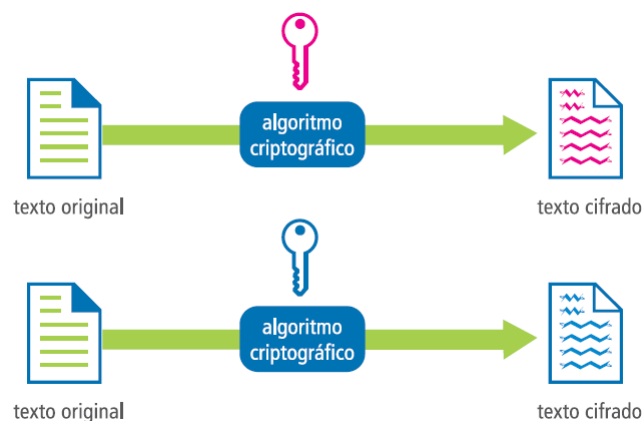


Figura 1 – Processo de cifragem.

Atualmente existem dois tipos de criptografia: a simétrica e a de chave pública. A criptografia simétrica realiza a cifragem e a decifragem de uma informação através de algoritmos que utilizam a mesma chave, garantindo sigilo na transmissão e armazenamento de dados. Como a mesma chave deve ser utilizada na cifragem e na decifragem, a chave deve ser compartilhada entre quem cifra e quem decifra os dados. O processo de compartilhar

uma chave é conhecido como troca de chaves. A troca de chaves deve ser feita de forma segura, uma vez que todos que conhecem a chave podem decifrar a informação cifrada ou mesmo reproduzir uma informação cifrada.

Os algoritmos de chave pública operam com duas chaves distintas: chave privada e chave pública. Essas chaves são geradas simultaneamente e são relacionadas entre si, o que possibilita que a operação executada por uma seja revertida pela outra. A chave privada deve ser mantida em sigilo e protegida por quem gerou as chaves. A chave pública é disponibilizada e tornada acessível a qualquer indivíduo que deseje se comunicar com o proprietário da chave privada correspondente [2].

### 2.1.1 Algoritmos Criptográficos de Chave Pública

Os algoritmos criptográficos de chave pública permitem garantir tanto a confidencialidade quanto a autenticidade das informações por eles protegidas.

#### 2.1.1.1 Confidencialidade

O emissor que deseja enviar uma informação sigilosa deve utilizar a chave pública do destinatário para cifrar a informação (Figura 2). Para isto é importante que o destinatário disponibilize sua chave pública, utilizando, por exemplo, diretórios públicos acessíveis pela Internet.

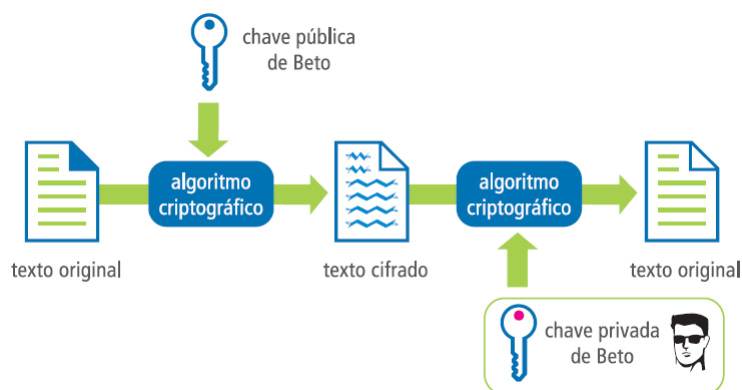


Figura 2 – Sigilo utilizando criptografia assimétrica.

O sigilo é garantido, já que somente o destinatário que possui a chave privada conseguirá desfazer a operação de cifragem, ou seja, decifrar e recuperar as informações originais. Por exemplo, para Alice compartilhar uma informação de forma secreta com Beto, ela deve cifrar a informação usando a chave pública de Beto. Somente Beto pode decifrar a informação, pois somente Beto possui a chave privada correspondente.



### 2.1.1.2 Autenticidade

No processo de autenticação, as chaves são aplicadas no sentido inverso ao da confidencialidade. O autor de um documento utiliza sua chave privada para cifrá-lo de modo a garantir a autoria em um documento ou a identificação em uma transação (Figura 3). Esse resultado só é obtido porque a chave privada é conhecida exclusivamente por seu proprietário.

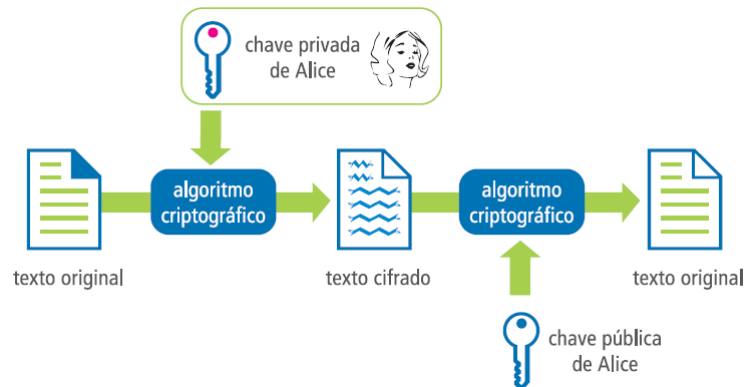


Figura 3 – Autenticidade utilizando criptografia assimétrica.

Assim, se Alice cifrar uma informação com sua chave privada e enviar para Beto, ele poderá decifrar esta informação pois tem acesso à chave pública de Alice. Além disto, qualquer pessoa poderá decifrar a informação, uma vez que todos conhecem a chave pública de Alice. Por outro lado, o fato de ser necessário o uso da chave privada de Alice para produzir o texto cifrado caracteriza uma operação que somente Alice tem condições de realizar [2].

## 2.2 Assinatura Digital

O mesmo método de autenticação dos algoritmos de criptografia de chave pública operando em conjunto com uma função resumo, também conhecido como função de hash, é chamada de Assinatura Digital.

O resumo criptográfico é o resultado retornado por uma função de hash. Este pode ser comparado a uma impressão digital, pois cada documento possui um valor único de resumo e até mesmo uma pequena alteração no documento, como a inserção de um espaço em branco, resulta em um resumo completamente diferente [2].

A Assinatura Digital é uma técnica capaz de agregar características de segurança semelhantes às obtidas ao assinarmos um documento utilizando papel e caneta, mas em meio eletrônico. Trata-se de uma forma de assinar arquivos digitais do tipo PDF, DOC, ou qualquer outro formato.

Quando buscamos formalizar um acordo ou demonstrar a autoria de determinada informação em meio físico, fazemos isso na forma de uma assinatura. Esta assinatura é, geralmente, uma marca manuscrita, com uma representação gráfica que faz alusão ao nome de seu autor. A assinatura carrega consigo características grafotécnicas dadas pela singularidade na forma de escrever de cada pessoa. Tais características, como a pressão utilizada na escrita e detalhes da caligrafia do indivíduo, são analisadas e comparadas com exemplares reconhecidamente provenientes do mesmo autor. Dá-se assim o processo de conferência ou validação de uma assinatura.

Ao receber um documento assinado, preocupamo-nos com sua autenticidade e autoria, ou seja, se ele foi realmente assinado pelas partes interessadas no documento, de forma legítima. Muitas vezes, recorre-se a um cartório para que seja conferida a assinatura de uma pessoa, processo conhecido como reconhecimento de firma. Outra preocupação frequente é relacionada com a integridade do conteúdo, onde são verificadas possíveis rasuras ou sinais que evidenciam uma tentativa de falsificação/modificação.

Em meio virtual, tal como ocorre no papel, é também possível obter garantias desejáveis a um documento assinado, como a autenticidade, autoria e integridade. Tudo isso graças a um conjunto de procedimentos matemáticos: os chamados algoritmos criptográficos de chave pública. Essas funções produzem um código de identificação único que associa um autor ao conteúdo produzido. Esse código é conhecido como Assinatura Digital e tem a propriedade de poder ser verificável por qualquer outra pessoa a quem interesse o documento, assim como ocorre com a assinatura de próprio punho.

Em uma assinatura convencional, manuscrita, a segurança do processo reside na capacidade de cada um de nós em produzir uma marca única, dadas as particularidades em nossa escrita. Já na Assinatura Digital, a segurança está atrelada à posse de um par de chaves criptográficas único para cada pessoa, conhecidas como chave privada de assinatura e chave pública de verificação. Essas duas chaves são geradas aleatoriamente por funções matemáticas e trabalham em conjunto. Tudo que uma assina, a outra, e somente a outra, é capaz de verificar. Assim, a chave privada é guardada sob a posse do usuário enquanto a chave pública é distribuída livremente na forma de um Certificado Digital.

Como ilustrado na Figura 4, a Assinatura Digital emprega uma primeira função matemática, conhecida como resumo criptográfico ou hash, para produzir uma representação reduzida e única do conteúdo a ser assinado. É sobre esse identificador que se aplica um segundo procedimento, a função criptográfica de assinatura propriamente dita, que faz uso da chave privada do usuário para produzir a Assinatura Digital. Por fim, é gerado um pacote assinado que contém o documento, a assinatura e o certificado do assinante, utilizado para verificar o processo. O certificado digital contém a chave pública do signatário, permitindo que qualquer interessado confira sua autenticidade.

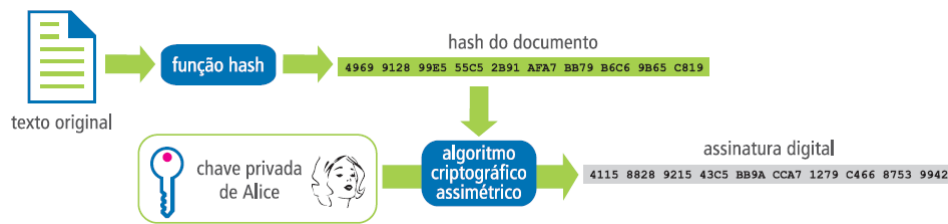


Figura 4 – Assinatura Digital utilizando algoritmos de chave pública.

Quem recebe o documento precisa utilizar a chave pública contida no Certificado Digital para reverter o processo de criptografia, obtendo novamente o hash calculado pelo assinante. Tirando um novo hash do documento recebido e comparando com o obtido no passo anterior, o verificador tem a certeza de que se trata do mesmo conteúdo assinado (Figura 5). Qualquer mínima alteração, de uma letra sequer, produzirá um resumo completamente diferente e invalidará a assinatura.

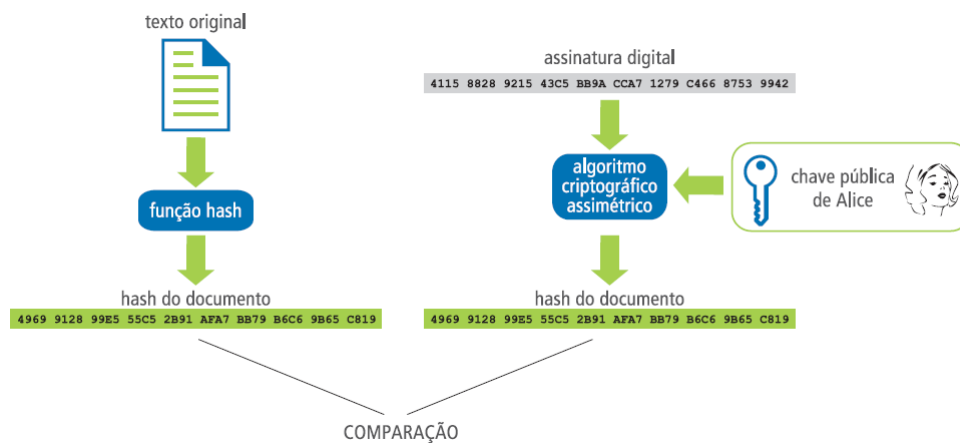


Figura 5 – Conferência da Assinatura Digital.

A vantagem da utilização de resumos criptográficos no processo de autenticação é o aumento de desempenho, pois os algoritmos de criptografia assimétrica são muito lentos. A submissão de resumos criptográficos ao processo de cifragem com a chave privada reduz o tempo de operação para gerar uma assinatura por serem os resumos, em geral, muito menores que o documento em si. Assim, consomem um tempo baixo e uniforme, independente do tamanho do documento a ser assinado.

## 2.3 Certificado Digital

O Certificado Digital é uma identidade eletrônica para pessoas ou empresas. Ele equivale à uma carteira de identidade do mundo virtual. Imagine uma versão eletrônica de todos os seus documentos, segura e com autenticidade garantida por criptografia complexa.

Com ele, é possível garantir de forma inequívoca a identidade de um indivíduo ou de uma instituição, sem uma apresentação presencial. Na prática, funciona como um CPF ou um CNPJ eletrônico. Essa ferramenta está disponível no Brasil desde 2001, após a criação da Infraestrutura de Chaves Públicas Brasileira – ICP Brasil.

### 2.3.1 Burocracia

Uma das principais funções do Certificado Digital é otimizar processos de assinatura de documentos, reduzindo custos com burocracia, impressão e cartórios, além de manter as mesmas características jurídicas dos documentos tradicionais.

Ao receber um documento de papel assinado, é comum analisar sua autenticidade e autoria, ou seja, se ele foi realmente assinado pelas partes interessadas no documento. Muitas vezes, recorre-se a um cartório para a conferência de uma assinatura de uma pessoa, processo conhecido como reconhecimento de firma. Outra preocupação frequente é relacionada com a integridade do conteúdo, onde são verificadas possíveis rasuras ou sinais que evidenciam uma tentativa de falsificação/modificação. Da mesma maneira, os documentos digitais, para terem validade jurídica, devem obedecer à uma série de normas e estarem protegidos por criptografia de alta complexidade. Desta forma, o Certificado Digital pode ser aplicado em uma série de atividades envolvendo contratos, laudos e outros documentos. Com ele, o usuário pode assinar um contrato sem sair de sua casa ou empresa, com a mesma legitimidade.

Tecnicamente, o Certificado Digital consiste em um arquivo eletrônico, emitido por uma Autoridade Certificadora. Ela funciona como se fosse o DETRAN (Departamento Estadual de Trânsito) para a emissão de carteiras de habilitação: é a entidade responsável por verificar a identidade do titular antes de realizar a certificação. A Figura 6 ilustra o conteúdo de um Certificado Digital.

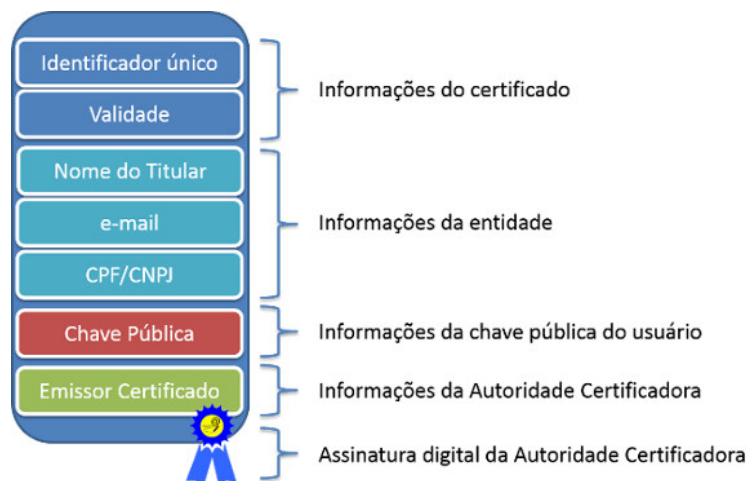


Figura 6 – Conteúdo de um Certificado Digital segundo padrão X.509.

Vantagens do Certificado Digital:

- É possível assinar documentos de qualquer lugar;
- Validade jurídica igual ao CPF ou CNPJ;
- Segurança: é impossível fraudar uma Assinatura Digital;
- Praticidade e economia, com a redução do volume de papel;
- Agilidade na assinatura dos documentos;
- Possibilidade de utilizar facilidades online de diversas empresas e órgãos públicos. Um dos órgãos federais que mais faz uso da Certificação Digital é a Secretaria da Receita Federal do Brasil como alternativa para dar agilidade e comodidade ao contribuinte, sem deixar de garantir o sigilo fiscal estipulado por lei.

Quem deseja obter o seu Certificado Digital deve procurar uma dessas instituições, que verificará seus documentos e criará sua identidade digital. Esta consiste em um arquivo contendo os dados referentes à pessoa ou empresa, protegidos por criptografia altamente complexa e com prazo de validade pré-determinado.

### 2.3.2 Segurança

O Certificado Digital é um arquivo que pode ser armazenado em dispositivos criptográficos portáteis, como um token (semelhante a um pendrive) ou smartcard. Também é possível guardar a chave em uma nuvem online, onde poderá ser acessada remotamente.

A principal funcionalidade do Certificado Digital é permitir a manifestação de intenções e consentimentos na forma eletrônica, processo conhecido como a Assinatura Digital de documentos. O emprego da assinatura digital permite o envio de e-mails assinados ou até mesmo secretos, a contratação de serviços, a delegação de poderes, entre tantas outras atividades do dia a dia que costumavam exigir assinatura manuscrita e reconhecimento de firma.

## 2.4 Carimbo do Tempo

É uma informação de data e hora segura aplicada a uma assinatura digital. Assim, o carimbo do tempo serve como evidência de que a assinatura de um documento eletrônico era válida naquela data e hora.

A presença de um carimbo de tempo prorroga a vida da assinatura de um documento eletrônico, uma vez que é possível verificá-la com base na data em que a assinatura foi

produzida. Uma assinatura sem um carimbo só permanece válida enquanto o certificado do signatário é válido. No entanto, com a aposição de um carimbo à assinatura, mesmo que o certificado do assinante deixe de ser válido após a assinatura do documento, seja por revogação ou expiração, a verificação da assinatura é feita com base na data e hora em que foi produzida. Com o carimbo, a assinatura mantém-se válida enquanto o carimbo é válido.

Na prática, um carimbo de tempo é um documento eletrônico assinado que contém o hash de sua respectiva assinatura e a data e hora de sua geração. Os carimbos de tempo são gerados por um sistema computacional confiável, o Sistema de Carimbo do Tempo (SCT). O relógio do SCT garante a data e hora correta, pois é sincronizado pelo Sistema de Auditoria e Sincronismo (SAS) da Autoridade Certificadora Raiz [3].

## 2.5 ICP-Brasil

A Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil) é um conjunto de normas, padrões, procedimentos e entidades que têm por objetivo manter, no Brasil, uma estrutura segura para a emissão de certificados digitais. A ICP-Brasil é governada por um Comitê Gestor (CG), formado por representantes do governo e da sociedade civil, e por uma Comissão Técnica (Cotec), que tem por objetivo auxiliar os membros do Comitê Gestor. Além dessas duas comissões, faz parte da ICP-Brasil o Instituto Nacional de Tecnologia da Informação (ITI), responsável por implementar a AC-Raiz e de credenciar e auditar todas as entidades integrantes da infraestrutura. Atualmente, podem ser credenciadas na ICP-Brasil autoridades certificadoras (AC), autoridades de registro (AR), autoridades de carimbo do tempo (ACT), prestadores de serviço de suporte (PSS) e auditores independentes [4].

## 2.6 Emissão de um Certificado Digital

De forma genérica, para a emissão de um Certificado Digital há a necessidade de três módulos. O primeiro é o Módulo Público (MP), o segundo módulo é a Autoridade de Registro (AR) e o terceiro é o módulo de Autoridade Certificadora (AC).

### 2.6.1 Módulo Público

O Módulo Público fornece uma interface para que o usuário final possa fazer a solicitação de um Certificado Digital. O MP é um portal que mantém o controle de acesso e os links para os sistemas que contém a inteligência de tramitação e fluxo de informações de operações dos processos, que podem ser:

- Interface para o cadastramento de novos usuários;

- Interface para solicitação de Certificado Digital;
- Interface para instalação de Certificado Digital;
- Possibilitar o acompanhamento através de interface do status do processo de emissão do Certificado Digital.

### 2.6.2 Autoridade de Registro

Uma Autoridade de Registro (AR) é uma entidade que verifica os dados do solicitante de um Certificado Digital. Além da conferência dos dados, a AR tem a obrigação de verificar pessoalmente a identidade do titular. Isso é feito por funcionários da AR especialmente treinados para este fim, conhecidos como Agentes de Registro. Os Agentes de Registro, após conferirem os dados diante do titular, encaminham à Autoridade Certificadora a requisição do certificado. Após a emissão do certificado pela AC, o certificado é entregue ao solicitante pelos próprios agentes de registro ou através de uma consulta a uma página web. De posse de certificado, o titular pode assinar documentos eletrônicos [3].

### 2.6.3 Autoridade Certificadora

Uma Autoridade Certificadora (AC) é uma entidade que emite Certificados Digitais. No Brasil, a Autoridade Certificadora precisa ser credenciada e periodicamente auditada pela Autoridade Certificadora Raiz (AC-Raiz). Um Certificado Digital só é emitido pela AC após a conferência dos dados do usuário, que é feita por Autoridades de Registro [3].

## 2.7 Sistemas Distribuídos

Segundo Coulouris [5] um sistema distribuído é definido como “aquele no qual os componentes de hardware ou software localizados em computadores interligados em rede, se comunicam e coordenam suas ações”.

A palavra-chave para compreender a dimensão dos sistemas distribuídos é compartilhamento. Os compartilhamentos disponíveis pela web e dispositivos móveis são tão naturais, que nem notamos a dimensão dessas estruturas. Segundo Coulouris, a heterogeneidade das estruturas encontram-se nos aspectos de: redes, hardware de computadores, sistemas operacionais, linguagens de programação, e a implementação dos diferentes desenvolvedores.

Mesmo com essas diversas heterogeneidades, um sistema distribuído se comporta de forma consistente e transparente para os usuários, durante as transições desses diferentes meios, a camada responsável por mascarar essa heterogeneidade denomina-se middleware,

que, segundo Coulouris, “fornece uma abstração, assim como, o mascaramento da heterogeneidade”. É nessa camada que as conversões de tipo de dados acontece, e que possibilita a comunicação em sistemas heterogêneos.

Dentre os modelos existentes, é discutido na Seção 2.8 o modelo cliente-servidor, que foi o escolhido para a implementação do projeto. Essa arquitetura contém processos clientes interagindo com processos servidores visando o compartilhamento de recursos. Em determinado ponto, servidores podem se tornar clientes devido à dependência de recursos [5].

## 2.8 Cliente-Servidor

A arquitetura cliente-servidor é um dos estilos mais encontrados para aplicações distribuídas baseadas em redes. Essa arquitetura trabalha com dois elementos, o cliente e servidor. Enquanto o papel do servidor é esperar solicitações de um cliente e retornar um feedback para o mesmo, o papel do cliente é enviar requisições através de solicitações a um servidor.

Com essas informações, podemos classificar o servidor denominando-o de backend, onde a parte lógica da requisição é processada, e o cliente como frontend, responsável pela interação humano-computador. Essa junção com a adição de um sistema operacional e as infraestruturas envolvidas caracterizam a arquitetura [6].

## 2.9 Engenharia de Software

Segundo Sommerville [7] a engenharia de software tem por objetivo apoiar o desenvolvimento de um software, incluindo técnicas, projeto e evolução dos sistemas. É uma área da engenharia que se preocupa com os aspectos da produção de um software.

A engenharia de software surgiu para corrigir problemas com relação ao desenvolvimento de projetos de software. A partir desse surgimento, modelos de processos e guias de desenvolvimento foram criados para otimização do processo de desenvolvimento do software [8].

O cientista de computação Friedrich Ludwig Bauer, em 1969, disse que a engenharia de software “é a criação e a utilização de sólidos princípios de engenharia a fim de obter software econômicos que sejam confiáveis e que trabalhem eficientemente em máquinas reais”. Já a IEEE (Instituto de Engenheiros Eletricistas e Eletrônicos) desenvolveu uma definição mais abrangente que é “aplicação de uma abordagem sistemática, disciplinada e quantificável, para o desenvolvimento, operação e manutenção do software, isto é, a aplicação da engenharia ao software” [9].



Para Pressman [8]: “a engenharia de software é uma tecnologia em camadas que deve estar fundamentada em um comprometimento organizacional com a qualidade, para promover um aperfeiçoamento contínuo dos processos. Onde a base da engenharia de software é a camada de processos, que mantém as camadas de tecnologia coesas e possibilita o desenvolvimento de software de forma racional e dentro do prazo, que é relacionado com o controle do gerenciamento de projetos de software e estabelece o contexto no qual são aplicados métodos técnicos. Assim, fornecendo as informações técnicas necessárias para o desenvolvimento de um software”.

### 2.9.1 Engenharia de Requisitos

A engenharia de requisitos é uma ação da engenharia de software que inicia durante a atividade de comunicação e continua na de modelagem. Ela deve ser adaptada às necessidades do processo, do projeto, do produto e das pessoas que estão realizando o trabalho. Segundo Sommerville [7] a engenharia de requisitos é o conjunto de descrições do que o sistema deve fazer, os serviços que serão oferecidos e as restrições de seu funcionamento. Esses requisitos refletem as necessidades dos clientes para que o sistema possa atender às funcionalidades específicas.

Por sua vez, Pressman [8] descreve que a engenharia de requisitos fornece mecanismo apropriado para entender aquilo que o cliente deseja, analisando as necessidades, avaliando a viabilidade, negociando uma solução razoável, especificando a solução sem ambiguidade, validando a especificação e gerenciando as necessidades à medida que são transformadas em um sistema operacional.

## 2.10 Banco de Dados

Silberschatz, Korth e Sudarshan [10] definem banco de dados como uma coleção lógica de dados relacionados e coerentes que possuem significado implícito e representam aspectos do mundo real. Em outras palavras, banco de dados é uma coleção de dados inter-relacionados representando informações sobre um domínio específico, sempre que possível agrupando as informações que se relacionam e tratam do mesmo assunto.

Por exemplo, a lista de nomes, números telefônicos e endereço de pessoas que conhecemos, esses dados podem ser escritos em uma agenda de telefones e armazenado em um computador. Essas informações são uma coleção de dados com um significado implícito, conseqüentemente, um banco de dados.

Por sua vez, Date [11] conceituou que sistema de banco de dados é um conjunto de componentes básicos: dados, hardware, software e usuários, conforme ilustrado na Figura 7.

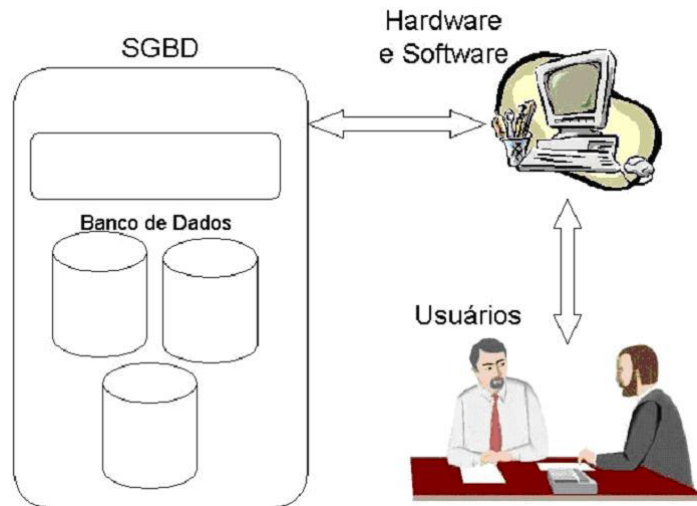


Figura 7 – Sistema de banco de dados.

Existem diferentes modelos de banco de dados para Silberschatz, Korth e Sudarshan [10], esses modelos são:

**Modelo relacional:** Usa uma coleção de tabelas para representar os dados e as relações entre eles, onde cada tabela possui diversas colunas, e cada coluna possui um nome específico. Modelo relacional é um exemplo de modelo baseado em registros, esse modelo é o mais usado nas maiorias dos sistemas de banco de dados atuais.

**Modelo de entidade/relacionamento (ER):** É baseado em uma percepção de um mundo real que consiste em uma coleção de objetos básicos.

**Modelo de dados baseado em objeto:** Pode ser visto como uma extensão ao modelo ER com noções de encapsulamento, métodos (funções) e identidade de objeto. O modelo de dados relacional de objeto combina recursos do modelo de dados orientado a objeto e do modelo de dados relacional.

**Modelo de dados semiestruturado:** Permite a especificação dos dados de forma que itens de dados individuais do mesmo tipo possam ter diferentes conjuntos de atributos.

Normalmente os bancos de dados são operados por SGDB (Sistema Gerenciador de banco de Dados). Segundo Elmasri e Navathe [12], SGDB é uma coleção de programas que permitem o usuário manipular as informações, criar, manter um banco de dados e interagir com o usuário. O SGDB é, portanto, um sistema de software que facilita os processos de definição, construção, manipulação e compartilhamento de banco de dados entre vários usuários e aplicação.

Segundo Silberschatz, Korth e Susarshan [10], um sistema de banco de dados fornece uma linguagem de definição de dados para especificar o esquema de banco de dados, DDL (Data Definition Language), e uma linguagem de manipulação de dados para expressar as consultas e atualizações de banco de dados, DML (Data Manipulation

---

Language). Essas duas linguagens formam partes de uma única linguagem de banco de dados, a linguagem SQL (Structured Query Language).

A SQL é uma linguagem de consulta estruturada padrão do banco de dados. A SQL foi desenvolvida pela IBM no início dos anos 70 e tinha por objetivo demonstrar a viabilidade da implementação do modelo relacional. SQL é uma linguagem universal para todos os bancos de dados. A linguagem SQL é um padrão ANSI (American National Standards Institute), para ser compatível com o padrão ANSI a linguagem deve possuir pelo menos, os comandos principais como (Select, Update, Delete, Insert). Assim, SQL tornou-se a linguagem mais utilizada para os SGBD.



## 3 Materiais, Técnicas e Métodos

Neste capítulo são abordadas todas as bibliotecas, metodologias, frameworks e tecnologias que auxiliaram no desenvolvimento do sistema.

### 3.1 Linguagens

#### 3.1.1 HTML

O HTML – Hypertext Markup Language – é uma linguagem de marcação de textos e a principal linguagem do conteúdo exibido na Internet. Esta linguagem descreve o documento exibido estruturalmente, bem como semanticamente.

O HTML é um padrão internacional definido e mantido pela W3C – World Wide Web Consortium. Para o compartilhamento de arquivos através da Internet, Tim Berners-Lee disponibilizou uma maneira organizada e que se diferenciou da maneira vigente na época. A comunicação era possível apenas com envio de texto puro e a partir deste avanço era possível acessar documentos com um software chamado de navegador, facilitando o acesso a arquivos e tornando-os mais ricos, isto é, contendo além de texto, imagens e métodos sofisticados de formatação [13].

#### 3.1.2 Javascript

JavaScript é uma linguagem de programação interpretada. Foi originalmente implementada como parte dos navegadores web para que scripts pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido.

É atualmente a principal linguagem para programação *client-side* em navegadores web. Foi concebida para ser uma linguagem script com orientação a objetos baseada em protótipos, tipagem fraca e dinâmica e funções de primeira classe. Possui suporte à programação funcional e apresenta recursos como fechamentos e funções de alta ordem comumente indisponíveis em linguagens populares como Java e C++. É a linguagem de programação mais utilizada do mundo [14].

#### 3.1.3 PHP

O PHP (Hypertext Preprocessor) foi concebido em 1994 como resultado do trabalho desenvolvido por Rasmus Lerdorf. Em 2002, já era utilizado por mais de nove milhões

de domínios em todo mundo. Em 2012, esse número estava próximo de um bilhão de domínios. Inicialmente, o PHP foi denominado Personal Home Page, mas posteriormente foi alterado para Hypertext Preprocessor, nome esse escolhido por meio de votação realizada na comunidade do PHP [15].

O PHP é uma linguagem de script embutido em HTML no servidor, projetada especificamente para o desenvolvimento Web. É possível embutir/inserir PHP dentro nas páginas HTML, pois o código PHP será executado/interpretado pelo servidor. O PHP é considerado um módulo oficial do servidor HTTP Apache, isso significa que o mecanismo de script do PHP pode ser construído/interpretado no próprio servidor Web, tornando a manipulação de dados mais rápida. Assim como o servidor Apache, o PHP é compatível com várias plataformas, o que significa que ele executa seu formato original nas mais diversas plataformas, UNIX, Windows, Mac e Linux [16].

Atualmente uma das grandes vantagens do PHP frente a seus concorrentes (Java Server Pages, ASP NET e Perl) é o alto desempenho, interface para os mais diferentes bancos de dados, bibliotecas para muitas tarefas comuns da web, baixo custo, portabilidade, facilidade de aprender e utilizar a linguagem, disponibilidade de código e suporte.

### 3.1.4 JSON

O JSON (JavaScript Object Notation) é um modelo para armazenamento e transmissão de informações no formato texto. Apesar de muito simples, tem sido bastante utilizado por aplicações Web devido à sua capacidade de estruturar informações de uma forma bem mais compacta do que a conseguida pelo modelo XML, tornando mais rápido o *parsing* dessas informações. Isto explica o fato de o JSON ter sido adotado por empresas como Google e Yahoo, cujas aplicações precisam transmitir grandes volumes de dados [17].

## 3.2 Banco de Dados

### 3.2.1 PostgreSQL

É um Sistema Gerenciador de Banco de Dados (SGBD) que tem a estrutura de um modelo de banco de dados relacional (Seção 2.10). Ele possui uma comunidade ativa de desenvolvedores, que melhoram cada vez mais a sua codificação, uma documentação rica e fornece ferramentas robustas para tratar as transações. Além disso, todos os grandes sistemas operacionais são suportados pelo PostgreSQL (GNU/Linux, Unix (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), e MS Windows). Também é um software de código aberto [18].

## 3.3 Arquitetura

### 3.3.1 REST

O REST é um sistema baseado no modelo cliente-servidor (Seção 2.8). Esta definição permite que vários clientes, implementados em diferentes linguagens e tecnologias, consigam realizar a comunicação com o servidor, simplificando seu desenvolvimento e tornando-o escalável [19].

Recurso é a unidade mínima do REST, está presente em todos os aspectos do protocolo e trata-se do conjunto de dados que trafegam por ele [20]. Para acessar estes recursos, são utilizados identificadores globais, chamados URI – Universal Resource Identifier, que utilizam o protocolo HTTP para realizar esta comunicação. Desta forma, cliente e servidor trocam recursos: o cliente envia um recurso que contém uma solicitação para um outro recurso, enquanto o servidor envia o recurso que foi solicitado.

Uma característica importante de um sistema REST é o não armazenamento de nenhum dado entre as requisições. Esta característica é conhecida como *stateless*, o que significa que cada uma das requisições do cliente para o servidor precisa conter toda a informação necessária para compreender a requisição sem depender de uma informação obtida previamente [19].

## 3.4 Bibliotecas e Frameworks

### 3.4.1 Angular

Angular é um *framework* utilizado para a construção de aplicativos do lado do cliente, utilizando as linguagens HTML (Seção 3.1.1), Javascript (Seção 3.1.2) e também Typescript [21].

O principal objetivo do *framework* Angular é facilitar o dinamismo da exibição das informações contidas em um documento HTML. Com ele, é possível, através de eventos, requisições e até mesmo rotinas programadas, alterar o conteúdo visual que está sendo exibido sem precisar recarregar a aplicação.

Para tal manipulação dos dados, é necessário que o conteúdo dessas rotinas e eventos fique na parte do cliente da aplicação. Sendo assim, o navegador interpreta toda a lógica contida no Angular. Através de tags de marcação é possível construir no HTML elementos desse *framework*, e, a partir dessas tags, são interpretados os valores que elas contêm. Caso uma rotina altere o valor de um objeto ou variável do Angular, no mesmo instante, o elemento visual marcado é alterado também.

A estrutura do *framework* Angular é considerada uma estrutura MVC (Model-

View-Controller) [22], contendo uma arquitetura que possui os seguintes componentes:

- Módulo: *Tag* de marcação responsável por modularizar a parte do HTML na qual o *framework* atua;
- Componente: É responsável pelo controle de uma view. Após a definição da lógica desse componente, ele é responsável pela exibição dos dados;
- Template: É uma parte do HTML que junto ao componente exibe informações na tela;
- Metadata: Dita ao Angular como processar uma classe;
- Data Binding: Responsável pela ligação da parte funcional e a parte visual da aplicação, mais especificamente, é o que liga as funções com o conteúdo da view (ligação entre templates e componentes);
- Injeção de dependência: É a forma de fornecer instâncias de classes para a aplicação, ou seja, o conteúdo de manipulação de dados.

Diferentemente da sua versão anterior, conhecida como AngularJS 1.x, o Angular é construído utilizando o Typescript, que é um super conjunto do Javascript em sua versão ES6 – ECMAScript 6. Na prática, o Typescript é uma adaptação do javascript para a produção de software em larga escala.

### 3.4.2 PrimeNG

A PrimeNG é uma biblioteca líder de componentes de interface de usuário para aplicações que utilizam o *framework* Angular (Seção 3.4.1), contendo mais de 80 componentes. A biblioteca PrimeNG teve enorme sucesso no mundo Angular devido ao seu desenvolvimento ativo em um curto espaço de tempo. É uma biblioteca em rápida evolução, que está alinhada com a última versão do Angular. Ao contrário dos concorrentes, a PrimeNG foi criada com aplicativos corporativos em mente [23].

### 3.4.3 Laravel

O Laravel é um *framework* PHP (Seção 3.1.3) de código aberto. Este *framework* fornece suporte a instalação de pacotes externos através do gerenciador de dependências *Composer* [24].

O ciclo de vida das aplicações construídas com Laravel inicia com todas as requisições sendo recebidas pelo arquivo `index.php`. Dentro deste arquivo, é gerada uma instância do componente HTTP Kernel que inicializa os demais componentes do Laravel.



O framework gera uma instância do HTTP Kernel, que é o componente responsável por processar uma imensa lista de inicializadores que terão a responsabilidade de capturar as variáveis de ambiente, executar todas as configurações, handlers, logs, etc. Tudo isso é feito antes mesmo da requisição ser processada.

São também responsabilidades do Kernel:

- Inicializar os serviços da aplicação;
- Receber requisições HTTP;
- Responder requisições HTTP.

O Laravel utiliza o conceito de rotas para distribuir as requisições para seus respectivos recursos. As rotas fazem o mapeamento de método e recurso que uma requisição está solicitando.

## 3.5 Ferramentas

### 3.5.1 Astah

Astah é uma ferramenta de modelagem UML (Unified Modeling Language). Esta ferramenta foi desenvolvida na plataforma Java, o que garante sua portabilidade para qualquer plataforma que possui JVM (Java Virtual Machine) [25]. Na Figura 8, vemos um exemplo do uso da ferramenta Astah.

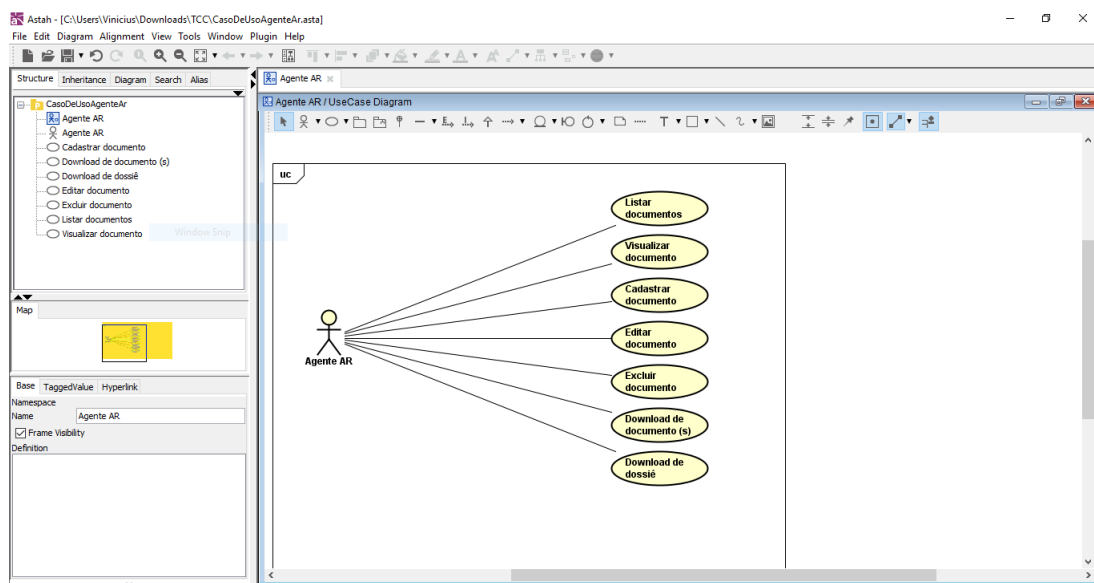


Figura 8 – Ferramenta Astah

Com uma interface simples, Astah possibilita qualquer pessoa a projetar um sistema complexo. A ferramenta é utilizada nos diagramas dinâmicos, voltada para modelagem de sistema.

### 3.5.2 Balsamiq

Balsamiq Mockups é uma ferramenta de design de interface de usuário para criação de *wireframes*. Pode ser usado para gerar esboços das ideias do produto, para facilitar a discussão e compreensão antes de escrever qualquer código [26].

Com o Balsamiq é possível gerar protótipos de baixa fidelidade, no qual pode ser visto um exemplo na Figura 9, de maneira rápida. Esses protótipos contêm os principais elementos da interface do sistema, facilitando assim a compreensão de como o sistema deve se comportar. Os protótipos construídos no Balsamiq são protótipos descartáveis, ou seja, não serão utilizados em outras fases do processo de desenvolvimento, exceto para consulta.

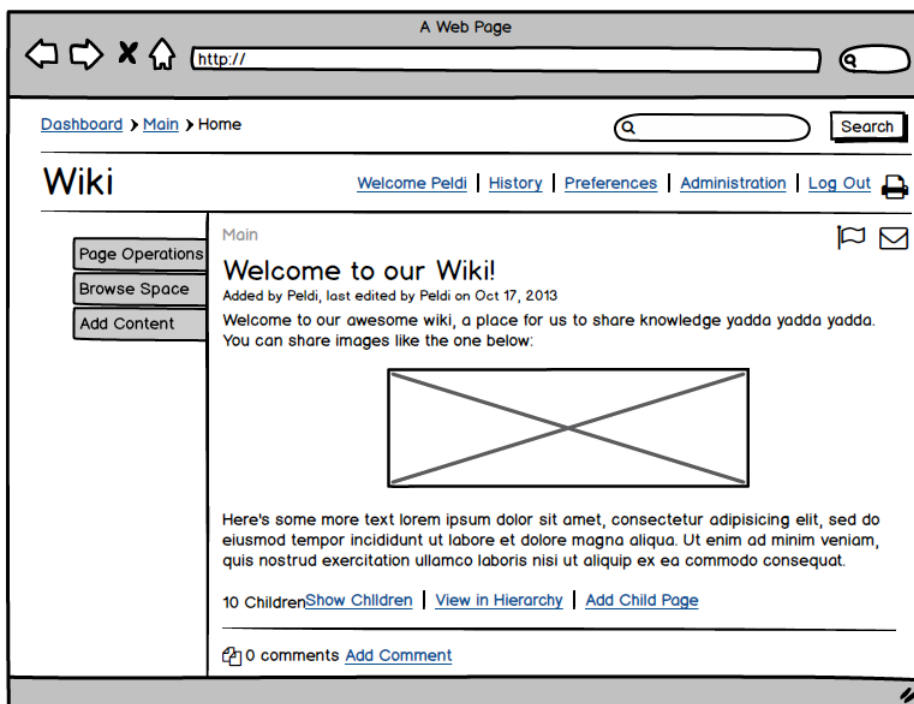


Figura 9 – Balsamiq Mockups - Exemplo de protótipo.

A ferramenta tem como forma de trabalho a técnica drag-and-drop, que consiste em arrastar um componente e soltá-lo onde se deseja. O Balsamiq oferece os principais componentes utilizados em interfaces, como botões, painéis, menus, tabelas, diferentes tipos de telas e outros.

### 3.5.3 Guzzle

O Guzzle é um cliente HTTP PHP que facilita o envio de solicitações HTTP e trivial para integração com serviços da web. Possui interface simples para construção de strings de consulta, solicitações de POST, streaming de grandes uploads, streaming de grandes downloads, uso de cookies HTTP, upload de dados JSON, etc.

Pode enviar solicitações síncronas e assíncronas usando a mesma interface. Usa interfaces do PSR-7 para solicitações, respostas e fluxos. Isso permite que você utilize outras bibliotecas compatíveis com o PSR-7 com o Guzzle. Abstrai o transporte HTTP subjacente, permitindo que o usuário grave código agnóstico de ambiente e transporte; ou seja, não há dependência forte em loops de eventos cURL, PHP streams, sockets ou não-bloqueantes [27].

### 3.5.4 Postman

O Postman é um software usado para emular requisições web. Assim, sem precisar que a requisição venha do sistema, é possível testar a qualidade do código produzido, e se está atendendo a necessidade em questão. As requisições do Postman são feitas usando a linguagem de marcação JSON, por onde são passadas as informações para o *backend* [28].

### 3.5.5 GIT

GIT é um sistema para controle de versões de arquivos, muito usado para versionamento de código. Com seu uso, diversas pessoas podem trabalhar simultaneamente no mesmo projeto, alterando e ou criando novos arquivos e permitindo que os mesmos possam existir sem o risco de suas alterações serem sobrescritas [29]. Para versionar o código deste projeto foi utilizado o GIT juntamente ao Gitlab para que o mesmo pudesse ser acessado facilmente de qualquer máquina.

## 3.6 Metodologia

### 3.6.1 Scrum

Para o desenvolvimento do sistema foi adotado o *framework* ágil de desenvolvimento de *software* Scrum. O Scrum foi concebido em 1993 por Jeff Sutherland a partir da necessidade de encontrar uma metodologia que abordasse o problema do desenvolvimento de software de uma forma não tradicional. Em 1995, Ken Schwaber refinou a metodologia baseado na sua própria experiência no desenvolvimento de sistemas e processos [30].

O Scrum é um conjunto de atividades e práticas de gestão que devem ser seguidas para garantir que no final do projeto seja entregue um produto como especificado pelo

cliente. Ela segue o paradigma iterativo e incremental onde a cada iteração pequenas partes do sistema são entregues. Isso é importante, pois garante que teremos partes funcionais do sistema e essas passarão por todas as atividades até a implantação. As principais características do Scrum, segundo Ferreira [30], são:

- É um processo ágil para gerenciar e controlar o desenvolvimento de projetos;
- É um wrapper para outras práticas de engenharia de software;
- É um processo que controla o caos resultante de necessidades e interesses conflitantes;
- É uma forma de aumentar a comunicação e maximizar a cooperação;
- É uma forma de detectar e remover qualquer impedimento que atrapalhe o desenvolvimento de um produto;
- É escalável desde projetos pequenos até grandes projetos em toda empresa.

#### 3.6.1.1 Vocabulário

**Time Scrum:** No Time Scrum não existe um líder para ditar as regras do desenvolvimento, o time é autogerido. Por isso, a opinião de todos os envolvidos é importante para o progresso do projeto.

**Scrum Master:** O Scrum Master não é o líder do time já que o scrum não possui um, mas sim a pessoa com bom conhecimento técnico em Scrum que organiza o projeto facilitando o andamento e garantindo que todos cumpram suas tarefas.

**Sprint:** É um ciclo de desenvolvimento ou em outras palavras um tempo determinado para entregar uma tarefa. No final de cada *Sprint* é entregue algo concreto.

**Product Owner:** É o responsável por traduzir as histórias do cliente em requisitos para que a equipe possa trabalhar no *Sprint*.

**Equipe de desenvolvimento:** São aqueles que participarão na codificação do Produto.

**Eventos:** São pequenas conversas com tempo fixo, objetiva e ágil.

**Reunião de planejamento:** Reuniões realizadas para definir os objetivos do *Sprint*. Para calcular a quantidade máxima de horas de cada reunião, multiplica-se cada semana do *Sprint* por 2; se um *Sprint* tem duração de 1 semana, por exemplo, a reunião terá no máximo 2 horas de duração.

**Reunião diária:** Reunião com no máximo 15 minutos para revisar o andamento do projeto.

**Revisão da Sprint:** Revisar os itens desenvolvidos e mostrar as novas funcionalidades.

**Retrospectiva da Sprint:** Serve para fazer um balanço geral da Sprint e verificar o que funcionou bem, as melhorias que tem que ser feitas, os erros e o que será feito para melhorar.

**Product backlog:** É uma lista com tudo que o cliente desejaria como funcionalidade. É criado pelo product owner e sofre mudanças a cada solicitação do cliente.

**Sprint backlog:** É uma lista com o que deve ser feito na próxima Sprint.



## 4 Projeto

O objetivo deste trabalho é a criação de um módulo no sistema de Autoridade de Registro para o gerenciamento da documentação necessária para emissão de certificados digitais. Neste capítulo é abordado mais detalhadamente o processo de levantamento de requisitos, protótipos e casos de uso do módulo/sistema. No escopo deste projeto, mesmo o objetivo sendo a criação de um módulo que fará parte de um sistema maior, denomina-se o módulo como um projeto de um sistema em menor escala.

### 4.1 Engenharia de Requisitos

Vimos quando falamos de Engenharia de Requisitos (Seção 2.9.1), que nessa etapa buscamos técnicas para colhermos requisitos do sistema a ser desenvolvido. Assim, definimos um conjunto de medidas a serem extraídas informações para auxiliar o desenvolvimento do sistema a ser criado, para atingir objetivos como: qualidade de software, produtividade no desenvolvimento, permitindo que o desenvolvedor atenda os prazos estabelecidos, qualidades e as funcionalidades do sistema que o cliente almeja.

#### 4.1.1 Documentação de Requisitos de Software

Esta etapa envolve as atividades de determinar os objetivos do sistema e as restrições associadas ao mesmo. Feita esta análise do sistema e delimitado o seu escopo, os seus requisitos devem ser analisados e especificados.

##### 4.1.1.1 Objetivo do Documento

A documentação dos requisitos tem como objetivo descrever e especificar os requisitos funcionais e não funcionais do sistema a ser desenvolvido, para atender todas as necessidades do cliente, especificando de uma maneira clara e objetiva para que o produto seja feito pelo desenvolvedor conforme o cliente deseja.

##### 4.1.1.2 Escopo do Sistema

O sistema tem como objetivos principais realizar operações de: listagem, visualização, cadastramento, edição, exclusão e *download*. O sistema poderá ser acessado somente por pessoas que possuam perfil de Agente de Registro.

#### 4.1.1.3 Descrição Geral do Sistema

**Descrição de solução:** Criar um módulo eficaz no sistema de Autoridade de Registro de modo a permitir o gerenciamento dos documentos pertencentes ao titular da requisição de um Certificado Digital.

**Benefícios:** Prover um armazenamento seguro, garantindo a integridade das informações e agilizar o processo tanto de emissão de Certificados Digitais quanto de auditorias realizadas na Autoridade de Registro.

#### 4.1.2 Requisitos do Sistema

Ao decorrer das primeiras *Sprints* (Seção 3.6.1) realizadas durante o projeto, fez-se o levantamento dos requisitos que o sistema deveria atender. Estes requisitos são divididos em dois tipos: Requisitos funcionais e requisitos não funcionais.

Em Engenharia de Software (2.9), um requisito funcional define uma função de um sistema de software ou seu componente. Uma função é descrita como um conjunto de entradas, seu comportamento e as saídas.

Um requisito não funcional de software é aquele que descreve não o que o sistema fará, mas como ele fará. Assim, por exemplo, têm-se requisitos de desempenho, requisitos da interface externa do sistema, restrições de projeto e atributos da qualidade. A avaliação dos requisitos não funcionais é feita, em parte, por meio de testes, enquanto que outra parte é avaliada de maneira subjetiva.

#### 4.1.3 Requisitos Funcionais (RF)

- RF01 - Listar documentação exibindo: tipo de documento, tipo de arquivo, tamanho e comentário;
- RF02 - Cadastrar documento com as seguintes informações: Id da requisição ao qual pertence, nome original do arquivo enviado, nome aleatório gerado, caminho indicando onde o documento foi salvo, extensão do arquivo indicando seu tipo, tamanho, tipo de documento, comentário, hash do arquivo, assinatura digital, campo indicando se o documento foi assinado por um agente AR ou por um carimbo do tempo e datas de cadastro e atualização;
- RF03 - O hash do documento deve ser calculado utilizando o algoritmo SHA-256;
- RF04 - O cadastro só deve aceitar arquivos nos formatos PDF, PNG e JPEG;
- RF05 - Os documentos devem ser salvos em pastas que possuam como nome o id da requisição ao qual estes pertencem;



- RF06 - Otimizar arquivos do tipo imagem (PNG e JPEG) caso possuam resoluções muito altas;
- RF07 - Editar um documento, permitindo a alteração do tipo de documento e comentário;
- RF08 - Excluir documento;
- RF09 - Permitir download individual ou múltiplo dos documentos;
- RF10 - Permitir download do dossiê;
- RF11 - O dossiê deve conter todos os documentos contidos na requisição e, caso a requisição já tenha sido aprovada, deve conter também os arquivos de assinatura para cada documento que faz parte da requisição;
- RF12 - Os arquivos de assinatura que fazem parte do dossiê devem ter o mesmo nome do documento original mas com a extensão .p7s;
- RF13 - Os downloads devem ser feitos em forma de arquivo .zip;
- RF14 - Deve verificar ao cadastrar um documento se este já não existe na requisição através da comparação dos hashes dos documentos;
- RF15 - Deve registrar em LOG com nível INFO todos os eventos de manipulação da documentação;
- RF16 - Deve registrar em LOG com nível ERROR todas as exceções que possam ocorrer;
- RF17 - Definir os tipos de documentação conforme normas ICP Brasil: documento de identificação (CNH, RG), endereço, título eleitor, PIS/PASEP, etc;
- RF18 - Não deve permitir o cadastramento, edição ou remoção de documentos após validação/aprovação da requisição;
- RF19 - Implementar o tratamento de erros no cadastramento, edição e remoção de documentos;
- RF20 - Ao aprovar a requisição, assinar cada documento pertencente a esta. Dependendo da política do certificado, deve ser assinado com o certificado do agente AR ou utilizando um carimbo do tempo;
- RF21 - Integrar com extensão do navegador para acessar o certificado e chave privada do agente AR;
- RF22 - Filtrar os certificados pelo CPF do agente AR autenticado no sistema;

- RF23 - Utilizar as funções da extensão do navegador para geração de assinatura baseada apenas no hash do documento.

#### 4.1.4 Requisitos Não Funcionais (RN)

- RN01 - O gerenciamento dos documentos deve ser intuitivo e de fácil uso;
- RN02 - O módulo do gerenciamento de documentos deve seguir o padrão do sistema;
- RN03 - O módulo deve ser responsivo para boa utilização em *tablets*;
- RN04 - O módulo deve ser desenvolvido seguindo o padrão do sistema, utilizando o *framework* Laravel (Seção 3.4.3) para o *backend* e o *framework* Angular (Seção 3.4.1) para o *frontend*;
- RN05 - O banco de dados utilizado deve seguir o padrão do sistema que é o PostgreSQL (Seção 3.2.1);
- RN06 - Durante a comunicação entre o cliente e o servidor deve haver uma indicação de processamento da operação;
- RN07 - A tabela e campos do banco de dados devem estar em Português;
- RN08 - Sempre que possível deve ser colocado no LOG as informações da requisição e em caso de erros, as possíveis causas.

## 4.2 Diagrama de Casos de Uso

Na elaboração do diagrama de casos de usos é exibido uma visão geral da funcionalidade do módulo e como os usuários irão utilizá-lo. No diagrama foram descritas as principais funcionalidades do módulo e como que o usuário irá se relacionar com o mesmo. A Figura 10 demonstra os casos de uso do sistema utilizando o *software* Astah (Seção 3.5.1).

## 4.3 Protótipos

A etapa de prototipação foi realizada com a intenção de fornecer um esboço das telas do módulo, para ter uma ideia do que seria. Foram criados protótipos simples de baixo nível técnico, que serviram apenas para se ter a interpretação e compreensão de como que seriam as telas do sistema e suas funcionalidades.

A Figura 11 demonstra como seria a visão dos documentos dentro de uma requisição por parte do Agente de Registro. As Figuras 12 à 15 demonstram os protótipos de telas para as operações de cadastro, edição, exclusão e visualização de documentos.

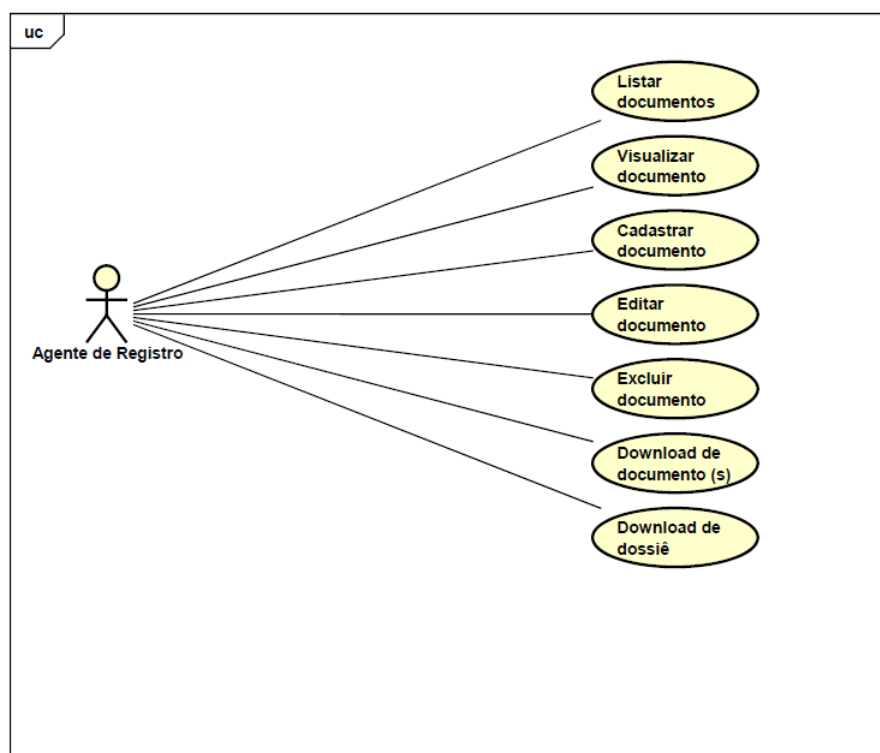


Figura 10 – Diagrama de casos de uso - Gerenciamento do dossiê.

Novo documento	Download	Download Dossie	Visualizar	Editar	Deletar
Tipo de documento		Tipo de arquivo	Tamanho	Comentário	
Documento de identificação		PNG	1024 KB	Baixa qualidade	
CPF		PDF	2048 KB		

Figura 11 – Protótipo - Listagem de documentos.

Novo documento

\* Tipo de documento

Comentário

+ Documento

RG.png 1024 KB

Figura 12 – Protótipo - Cadastro de documento.

Editar documento

\* Tipo de documento

Comentário

Figura 13 – Protótipo - Edição de documento.

Confirmação

Tem certeza que deseja  
excluir o documento?

Figura 14 – Protótipo - Exclusão de documento.

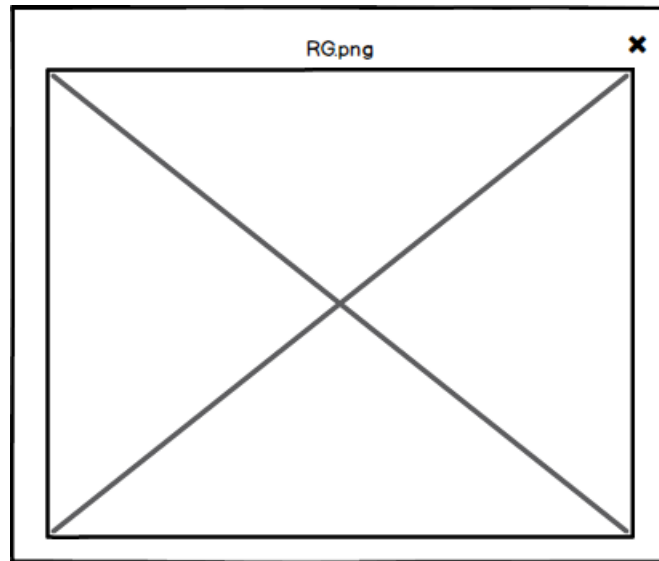


Figura 15 – Protótipo - Visualização de documento.

## 4.4 Modelagem do Banco de Dados

Nesta seção é apresentado como o banco de dados para o projeto foi desenvolvido e estruturado. Foram utilizados o modelo entidade-relacionamento e o modelo lógico (Seção 2.10).

### 4.4.1 Modelo Entidade-Relacionamento (ER)

O processo de criação do modelo de banco de dados foi realizado através do uso do *software* Astah (3.5.1). Primeiramente, foi desenvolvido o DER (Diagrama de entidade-relacionamento), que nada mais é que uma representação visual das entidades/tabelas do banco de dados e suas cardinalidades.

O modelo entidade-relacionamento é considerado a base da área moderna da tecnologia de banco de dados, ou seja, um projeto que usa uma base de dados precisa ter o modelo entidade-relacionamento bem definido [11].

A Figura 16 apresenta como o diagrama entidade-relacionamento foi desenvolvido para atender o proposto no projeto. A tabela de requisições já estava modelada no sistema da AR em desenvolvimento, sendo mostrada apenas para demonstrar a sua relação com a tabela de documentos, que faz parte do desenvolvimento do módulo proposto neste trabalho.

### 4.4.2 Modelo lógico

Com base no diagrama entidade-relacionamento foi desenvolvido o modelo lógico da tabela de documentos. O modelo lógico possibilita um detalhamento maior da entidade

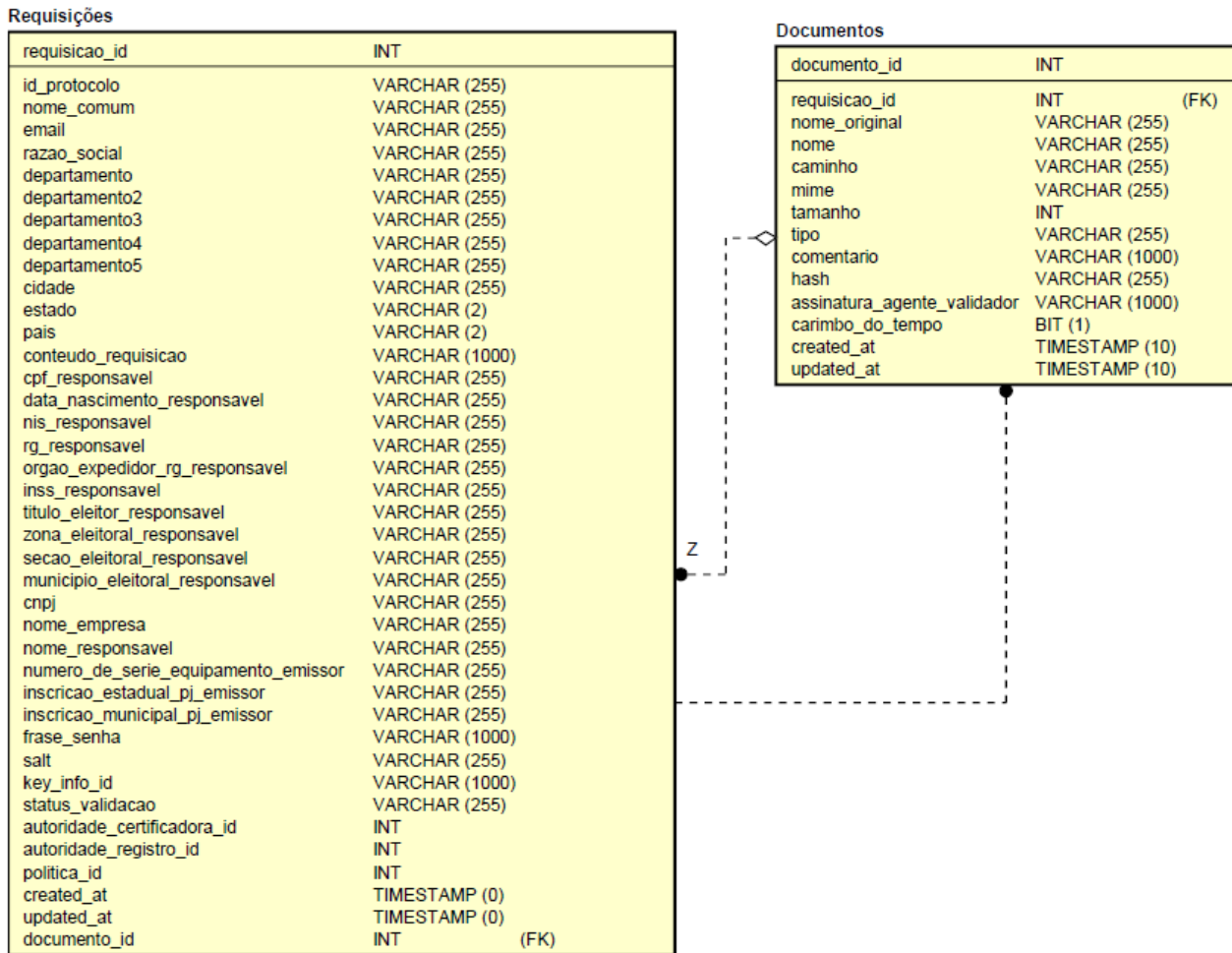


Figura 16 – Modelo Entidade Relacionamento - Documentos.

desenvolvida, sendo possível visualizar os tipos de dados, os atributos, chaves primárias e estrangeiras, etc. A Figura 17 mostra a tabela de documentos e sua estrutura lógica.

```
CREATE TABLE documentos
(
  id serial NOT NULL,
  nome_original character varying(255) NOT NULL,
  nome character varying(255) NOT NULL,
  caminho character varying(255) NOT NULL,
  mime character varying(255) NOT NULL,
  tamanho integer NOT NULL,
  tipo character varying(255) NOT NULL,
  comentario text,
  requisicao_id integer NOT NULL,
  hash character varying(255) NOT NULL,
  created_at timestamp(0) without time zone,
  updated_at timestamp(0) without time zone,
  assinatura_agente_validador text,
  carimbo_do_tempo boolean DEFAULT false,
  CONSTRAINT documentos_pkey PRIMARY KEY (id),
  CONSTRAINT documentos_requisicao_id_foreign FOREIGN KEY (requisicao_id)
    REFERENCES requisicoes (id) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE RESTRICT,
  CONSTRAINT documentos_tipo_check CHECK (tipo::text = ANY (ARRAY[
    'doc_identificacao'::character varying, 'comprovante_endereco'::character varying,
    'titulo_eleitor'::character varying, 'pis_pasep'::character varying,
    'cpf'::character varying, 'ato_constitutivo'::character varying,
    'ata_eleicao'::character varying, 'cnpj'::character varying,
    'outro'::character varying]:text[]))
)
WITH (
  OIDS=FALSE
);
ALTER TABLE documentos
  OWNER TO postgres;
```

Figura 17 – Modelo Lógico - Documentos.





## 5 Implementação

Neste capítulo é descrita a implementação do módulo proposto neste projeto. O desenvolvimento do sistema foi feito através das tecnologias e ferramentas apresentadas no Capítulo 3 e seguindo os requisitos apresentados no Capítulo 4. Primeiramente, é apresentada a arquitetura utilizada e em seguida é realizada uma melhor descrição do desenvolvimento das partes do servidor e do cliente.

### 5.1 Arquitetura geral

A arquitetura implementada atualmente se baseia no modelo cliente-servidor (Seção 2.8). A tecnologia cliente-servidor é uma arquitetura na qual o processamento da informação é dividido em módulos ou processos distintos. Um processo é responsável pela manutenção da informação (servidores) e outros responsáveis pela obtenção dos dados (os clientes). Os processos cliente enviam pedidos para o processo servidor, e este por sua vez processa e envia os resultados dos pedidos.

O documento DOC-ICP-03.01 [31] descreve as características mínimas de segurança para as Autoridades de Registro da ICP-Brasil. No item 6.2.3 temos a seguinte descrição em relação ao armazenamento de documentos:

- O armazenamento definitivo dos dossiês de titulares de certificado, em papel, digitalizados ou eletrônicos, deve ser feito:
  1. Em um dos pontos de centralização da AR, para aquelas que possuam mais de uma instalação técnica;
  2. Ou no ponto de centralização da AC à qual a AR está vinculada;
  3. Ou na AC emissora para os casos de certificados A CF-e-SAT (dossiê eletrônico).

Ciente do critério definido pela ICP-Brasil, chegou-se ao desenvolvimento da arquitetura da Figura 18.

O fluxo das operações realizadas no gerenciamento do dossiê do titular da requisição funciona da seguinte maneira:

1. O Agente de Registro, através da aplicação *frontend* da Autoridade de Registro, acessa a tela de uma requisição pertencente a algum titular;
2. Ao acessar a tela da requisição do certificado, uma solicitação é feita ao *backend* da AR utilizando código HTTP (GET);

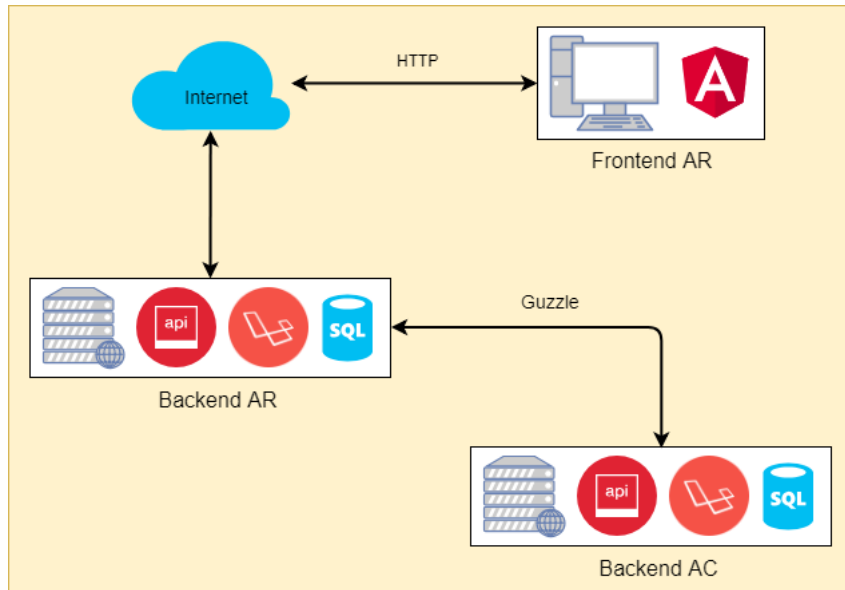


Figura 18 – Arquitetura da comunicação entre sistemas.

3. O *backend* da AR recebe a solicitação, faz algumas validações e reencaminha a solicitação para o *backend* da AC através do uso da ferramenta Guzzle (Seção 3.5.3);
4. O *backend* da AC recebe a solicitação do *backend* da AR, faz algumas validações e retorna os dados pedidos em forma de JSON (Seção 3.1.4);
5. O *backend* da AR recebe os dados retornados pelo *backend* da AC e retorna para o *frontend* da AR;
6. O *frontend* da AR recebe os dados solicitados e conclui o processo mostrando a documentação pertinente à requisição.

A arquitetura descrita foi baseada no item que define que o armazenamento definitivo dos dossiês dos titulares de certificado deve ser feito no ponto de centralização da AC à qual a AR está vinculada [31]. Por isso, é necessário a comunicação entre o *backend* da AR e o *backend* da AC, pois os documentos ficam salvos no sistema da AC, sendo então esta a responsável por retornar estes dados.

## 5.2 Servidor

De maneira geral, a aplicação *backend* é alocada diretamente em um servidor da Internet, deixando as rotas de acesso aos recursos disponíveis para uso. As rotas funcionam como porta de entrada para as funções de inclusão, atualização, remoção e leitura. Através delas, as aplicações direcionam as requisições entre as classes do sistema até que as informações necessárias sejam operadas no banco de dados.

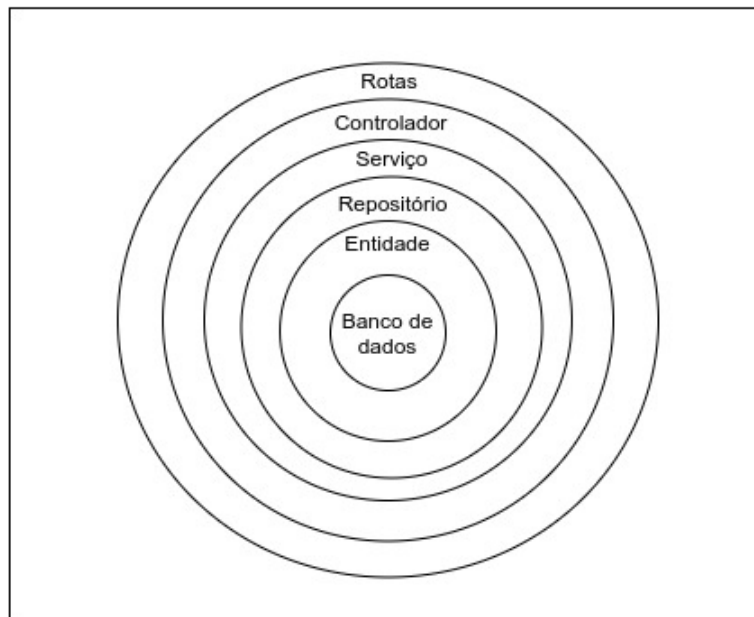


Figura 19 – Estrutura do backend em camadas.

### 5.2.1 Módulo AC

A estrutura do módulo desenvolvido no *backend* da AC funciona através de camadas. A Figura 19 mostra um esquema geral da estrutura do projeto. Para acessar o banco de dados, é obrigatório que a requisição seja repassada por todas as camadas, iniciando pelo arquivo de rotas. O arquivo de rotas faz o uso da classe `Route` do *framework* Laravel (Seção 3.4.3), que permite registrar as rotas e as deixar disponíveis para qualquer função do HTTP.

Utilizando os métodos fornecidos pelo `Route`, são passados dois parâmetros: `$uri` e `$callback`. O primeiro parâmetro é o ponto de acesso da requisição a partir da URI do servidor. O segundo é a função a ser executada com o valor recebido pela URI.

No caso deste projeto, o primeiro parâmetro é o recurso que deve ser acessado e o segundo é o caminho para a função que deve ser executada no controlador quando os parâmetros da rota forem atendidos. Por exemplo, para realizar uma consulta e retornar todos os documentos de uma requisição, o cliente realizará, em ambiente local, uma requisição para `http://localhost/api/requisicoes/{requisicao_id}/documentos`.

Quando o arquivo de rotas identificar o método HTTP presente na requisição, encaminhará os parâmetros para o seu controlador `DocumentosController`, apontando em seguida qual método do controlador o sistema deverá executar. A função deve vir precedida de um sinal de arroba (`@`). Então, tem-se: `Route::get("documentos", "DocumentosController@index")`.

O controlador tem seu funcionamento baseado no direcionamento das informações

para a camada de serviços. Esta classe recebe a requisição completa e condiciona a um objeto que pode ser operado dentro do *backend*. Observando o segundo parâmetro para `Route::get("documentos", "DocumentosController@index")` no exemplo citado, tem-se o nome da classe do controlador `DocumentosController` e o nome de uma função presente na classe, neste caso "index".

Dentro da função "index" é onde ocorre o apontamento mapeado para a classe de serviço. Neste caso, é realizada uma chamada para um método também existente na classe `DocumentosService`. Quando a função do controlador termina de processar o valor do retorno, este retorno é dado já convertido em JSON.

A camada de serviços é responsável por implementar as regras de negócio da aplicação. Nesta camada são realizados os tratamentos de exceção, validação de campos importantes vazios, e tratamento de mensagens de alto nível para o usuário, sendo possível também construir consultas mais complexas em que não seja possível utilizar os assistentes de mapeamento de objetos através da camada de repositórios.

A camada de repositório é a camada responsável por acessar os assistentes de mapeamento de objetos (ORM – Object Relational Mapper). O *framework* Laravel (Seção 3.4.3) fornece por padrão o seu ORM Eloquent. O mapeamento de objetos realiza a representação das tabelas do banco de dados em forma de orientação a objetos e é realizada de maneira automática quando utiliza-se o Laravel e o Eloquent.

Para que a classe de repositório saiba qual é a tabela que deve operar, é criada uma classe auxiliar chamada Entidade. Esta classe possui o papel de trabalhar com o Eloquent para facilitar a tarefa do desenvolvedor em criar funções básicas de inserção, edição, remoção e leitura de dados. As entidades possuem atributos básicos:

- `$table`: Informa qual a tabela no banco de dados que é espelhada na classe;
- `$primaryKey`: Informa a chave primária da tabela;
- `$fillable`: Informa quais os campos poderão receber valor ao realizar operações de manipulação de dados.

A Tabela 1 mostra as requisições necessárias para cada uma das operações disponíveis no módulo.

## 5.2.2 Módulo AR

Como descrito na Seção 5.1, as requisições HTTP enviadas do cliente (*frontend*) da AR são primeiramente recebidas pelo *backend* da AR e em seguida encaminhadas para o *backend* da AC, o qual é o responsável pelo retorno dos dados referentes à requisição do cliente.

Ação	Requisição	Método	Argumentos	Resultado Esperado
Listar documentos	requisicoes/{requisicao_id}/documentos	GET	requisicao_id	Retorno em JSON de todos os documentos pertencentes à requisição do id passado como argumento
Visualizar documento	documentos/{documento_id}	GET	requisicao_id, documento_id	Retorno do tipo blob do arquivo correspondente ao documento selecionado, podendo ser uma imagem ou PDF
Cadastrar	documentos	POST	arquivo_documento, requisicao_id, tipo, comentario	Retorno em JSON dos dados cadastrados
Editar	documentos/{documento_id}	PATCH	requisicao_id, documento_id, tipo, comentario	Retorno em JSON dos dados atualizados
Excluir	documentos/{documento_id}	DELETE	requisicao_id, documento_id	Retorno com código HTTP de sucesso: 200
Download documento (s)	documentos/download	POST	requisicao_id, documentos_ids[]	Retorno do tipo blob de um zip contendo os arquivos dos documentos selecionados
Download dossiê	documentos/dossie	POST	requisicao_id	Retorno do tipo blob de um zip contendo os arquivos dos documentos selecionados, podendo conter também os arquivos de assinatura de cada documento

Tabela 1 – Requisições para cada operação presente no módulo.

Para realizar este encaminhamento das informações entre os *backends* da AR e AC foi utilizado a ferramenta Guzzle (Seção 3.5.3). O funcionamento do *backend* da AR tem comportamento similar ao do *backend* da AC, a diferença é que o *backend* da AR faz o encaminhamento direto na camada do Controlador, não utilizando assim as camadas de Serviço e Repositório.

Quando o arquivo de rotas identifica o método HTTP presente na requisição, ele encaminha os parâmetros para o controlador `DocumentosController`, apontando em seguida qual método do controlador o sistema deve executar. Dentro do método específico para o tipo de requisição recebida são feitas algumas validações e posteriormente através do uso das funções fornecidas pelo Guzzle, é feito o reenvio dos dados para o *backend* da AC (Figura 20).

## 5.3 Cliente

O desenvolvimento da parte *frontend* do módulo proposto neste trabalho teve como intuito prover todas as funcionalidades levantadas e que podem ser vistas no diagrama de casos de uso na Seção 4.2. O sistema foi desenvolvido utilizando o *framework Angular* (Seção 3.4.1) com o auxílio da biblioteca PrimeNG (Seção 3.4.2).

O Angular adota o padrão semelhante ao MVC (Model-View-Controller) (Figura 21) para estruturar qualquer aplicação. Os dados para apresentar ao usuário corresponde ao modelo em um projeto Angular, que, em sua maior parte, é composto de dados puros

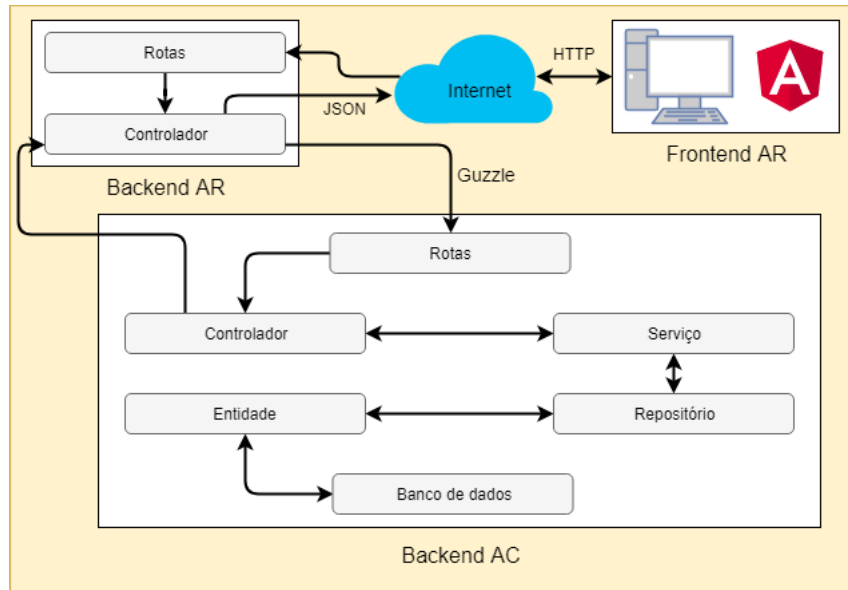


Figura 20 – Arquitetura da comunicação entre sistemas detalhada.

e é representado por meio de objetos JSON. A interface do usuário ou o HTML final renderizado, que o usuário vê e com o qual interage, e que exibe os dados ao usuário, corresponde à visão. Por fim, a lógica de negócios e o código que acessa os dados, decide que parte do modelo deve ser apresentada ao usuário, lida com validação e assim por diante, ou seja, é a lógica central da aplicação, corresponde ao controlador.

A abordagem MVC ou semelhante a ela é adequada por alguns motivos consistentes:

- Há uma clara separação de responsabilidades entre as diversas partes de sua aplicação, com estrutura e padrões bem definidos;
- O Angular é totalmente compatível com o MVC. O controlador jamais fará uma referência direta à visão, assim, o controlador fica independente da visão, além de permitir testar facilmente o controlador sem a necessidade de instanciar um DOM.

O Angular permite aplicar praticas-padrão de engenharia de *software* testadas e aprovadas, tradicionalmente utilizadas do lado do servidor, na programação do lado cliente para acelerar o desenvolvimento de *frontends*. Esse novo método de desenvolvimento possibilitou que aplicações web de pequeno e grande porte fossem desenvolvidas de modo que exigissem menos linhas de código, com isso um aumento de produtividade e o resultado final trazendo uma grande melhora no desempenho da aplicação.

A implementação da interface do módulo seguiu o que foi demonstrado nos protótipos realizados e que podem ser vistos na Seção 4.3. A estrutura do componente criado seguiu o padrão do sistema *frontend* já em desenvolvimento da AR que é o padrão do Angular.

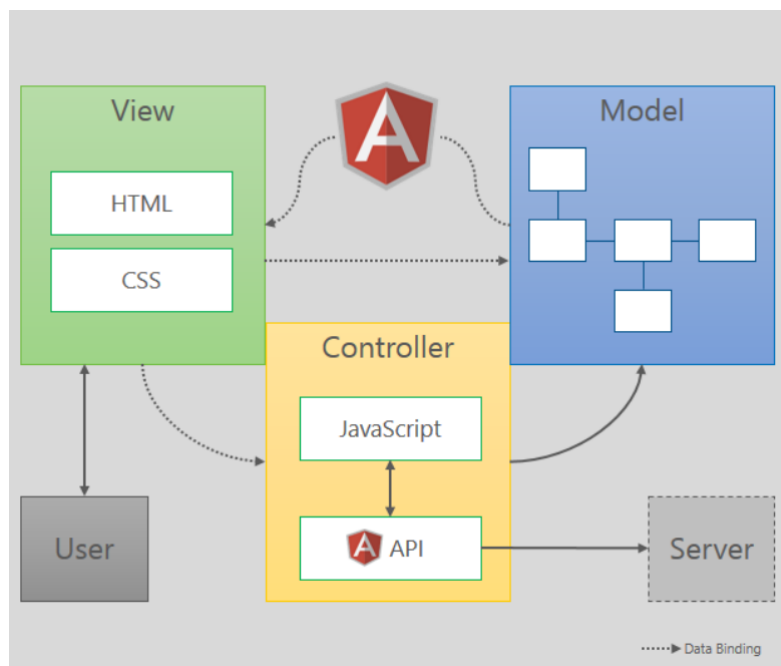


Figura 21 – Arquitetura MVC.

Uma aplicação cliente utilizando Angular tem seu ponto de acesso no arquivo `index.html`, dentro do diretório `src`. Este diretório contém armazenados os códigos-fonte que são exclusivos da aplicação, isto é, os arquivos feitos para este trabalho bem como os arquivos de configuração do *framework*. O diretório `src` fornece acesso ao diretório `app`, que contém dois dos principais arquivos da aplicação: `app.module.ts` e `app.component.ts`. Estes são os arquivos que possuem as configurações mínimas do projeto.

Dentro do mesmo diretório `src`, foram criados os diretórios para armazenar cada uma das páginas a serem exibidas dentro do sistema da AR, bem como os componentes auxiliares. Em geral, cada um dos diretórios contém os arquivos:

- `<classe>.component.ts`;
- `<classe>.component.html`;
- `<classe>.component.scss`;
- `<classe>.module.ts`.

A Figura 22 demonstra como ficou estruturado o diretório que contém a lógica responsável pela requisição de um Certificado Digital. Pode-se notar que o módulo do gerenciamento do dossiê foi incluído dentro deste diretório por fazer parte de uma requisição.

Dentro do diretório que representa o componente de gerenciamento de documentação (document) temos quatro arquivos:

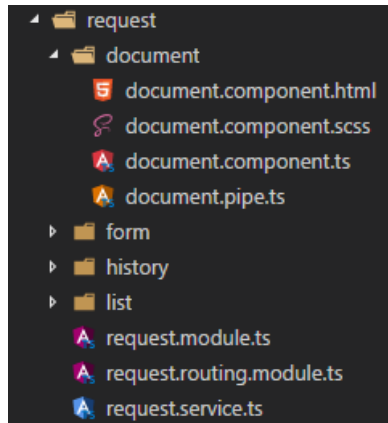


Figura 22 – Estrutura do diretório contendo a lógica de uma requisição.

```

<!-- Documentos -->
<p-card id="p-card-documents" class="ui-g" [hidden]="stepsItems[activeIndex].id !== 'documents'">
  <div class="ui-g-12 ui-g form-group">
    <app-document class="ui-g-12"></app-document>
  </div>
</p-card>

```

Figura 23 – Seletor do componente responsável pelo gerenciamento do dossiê.

- `document.component.html` e `document.component.scss`: Estes arquivos são responsáveis pela interface do módulo. Dentro do arquivo `.html` estão inseridas as tags fornecidas pela biblioteca PrimeNG. No arquivo `.scss` estão inseridas possíveis regras para ajustes na interface do componente;
- `document.component.ts`: Este arquivo é o responsável pela parte do Controlador na arquitetura MVC. É nele que ficam as lógicas das operações disponibilizadas pelo módulo ao usuário;
- `document.pipe.ts`: Este arquivo implementa a ideia de Pipe fornecida pelo Angular. Pode-se dizer que os Pipes são uma maneira elegante de realizarmos transformações no nosso *frontend*. Com ela se pode criar funções ou filtros, que podem ser utilizadas em qualquer parte do *template* do projeto. Um exemplo que foi utilizado no módulo foi transformar a informação do tamanho de um documento, que vem do servidor como bytes, e mostrar no *template* como kilobytes.

Com a criação do componente para tratar a parte da documentação, a sua integração com o resto do sistema se tornou fácil, pois bastou introduzir o seletor correspondente ao módulo (`app-document`) no componente de requisição. A Figura 23 mostra o seletor inserido dentro do arquivo `request-form.component.html`.

As operações que necessitam retornar ou enviar dados ao servidor são realizadas através do arquivo `request.service.ts` (Figura 22). Um exemplo de uma operação que envia



```
// Documentos
getDocuments(requestId: number): Observable<DocumentModel[]> {
  const params = {
    autoridade_certificadora_id: this.authSvc.acId.toString(),
    requestId: requestId.toString()
  }
  return this.http.get<DocumentModel[]>('documentos', {params: params})
}
```

Figura 24 – Operação para retorno de dados relacionados aos documentos de uma requisição.

uma requisição para o servidor de modo a receber os dados dos documentos pertencentes a requisição é mostrada na Figura 24. Pode ser observado na imagem que uma chamada HTTP do tipo GET é realizada para o endpoint “documentos” com os parâmetros necessários. Após o retorno dos dados do *backend*, o controlador do módulo de documentos realiza as lógicas necessárias antes dos dados serem renderizados no *template*.

## 5.4 Integridade das informações

A integridade das informações relacionadas aos documentos de uma requisição de Certificado Digital foi de grande importância durante a implementação do módulo escopo deste trabalho. Para garantir a integridade dessas informações, utilizou-se dos conceitos de Assinatura Digital (Seção 2.2) e Carimbo do Tempo (Seção 2.4).

Como foi visto na Seção 2.6, uma das etapas para a obtenção de um Certificado Digital é a verificação e validação da documentação pertencente ao titular por um Agente de Registro. Dependendo do resultado dessa etapa, o Agente de Registro, através do sistema *frontend* da AR pode aprovar ou reprovar a requisição do titular.

Em caso de aprovação da requisição, uma das seguintes operações ocorrem:

- Se a política do Certificado Digital requisitado necessitar da assinatura do Agente de Registro, o *frontend* do módulo desenvolvido através da integração com uma extensão presente no navegador acessa os Certificados Digitais presentes na máquina em operação. Estes certificados são filtrados de acordo com o CPF do Agente de Registro logado na sessão do sistema. Ao aprovar a requisição, o agente escolhe seu Certificado Digital fornecido pela extensão do navegador e a extensão gera uma Assinatura Digital utilizando este certificado para os dados da requisição e para os dados de cada documento pertencente a ela. Assim, os dados da requisição, da documentação e a assinatura gerada são enviados ao *backend* da AR que, posteriormente, são encaminhados para o *backend* da AC a qual a AR pertence. Ao completar a operação, cada documento terá no banco de dados a respectiva Assinatura Digital salva em seu registro;

- Se a política do Certificado Digital necessitar de uma Assinatura Digital, sendo que esta não precisa ser gerada pelo Certificado Digital de um Agente de Registro, o *frontend* só envia as informações da requisição e dos documentos pertencentes a ela ao *backend* da AR. Ao chegar no *backend* da AC vinculada, os documentos são assinados utilizando o Carimbo do Tempo, através de um método já fornecido e implementado no sistema atual.

Para realizar a verificação de integridade de certo documento, basta recuperar o arquivo original, pegar sua respectiva Assinatura Digital salva no banco de dados e seguir o conceito apresentado na Seção [2.2](#).

## 6 Resultados

Neste capítulo, são apresentadas as funcionalidades do módulo proposto neste projeto. Primeiramente, são apresentados os testes realizados utilizando a ferramenta Postman (Seção 3.5.4) e o conceito de testes unitários. Em seguida, é realizada uma melhor descrição das funcionalidades e telas desenvolvidas. Por fim, também é mostrada a implementação do módulo desenvolvido no sistema do Módulo Público.

### 6.1 Testes com Postman

Em uma das etapas de testes utilizou-se o Postman, uma ferramenta extremamente eficiente para realizar testes de API. É utilizada por empresas de grande porte como Microsoft, Cisco, Oracle etc.

Dentre as características da ferramenta, é possível:

- Fazer requisições simples ou complexos com agilidade;
- Salvar requisições;
- Utilizar, organizar e automatizar coleções.

Através do seu uso, foram criadas requisições para testar se o *backend* se comportava de acordo com o esperado. As Figuras 25 a 28 demonstram as requisições feitas na ferramenta para as operações CRUD na documentação. Como pode ser observado, algumas das operações retornam dados em JSON, enquanto outras possuem apenas um código HTTP indicando o estado da resposta.

### 6.2 Testes Unitários

Em outra etapa de testes, realizou-se a implementação de teste unitários utilizando o padrão de testes do *framework* Laravel. O Laravel utiliza como padrão para testes unitários o *framework* PHPUnit. O PHPUnit é um *framework* de testes unitários para a linguagem PHP que provê um ecossistema para a execução de testes de forma automatizada. “Unitário” se refere literalmente à unidade, pequenas partes. Logo, testes unitários são testes para pequenas partes de código. No objetivo geral, testando cada unidade vamos saber se toda nossa aplicação está funcionando corretamente, e se não está, qual parte (unidade) está falhando.

Assim, foram feitos testes para casos de sucesso e erro para as seguintes partes:

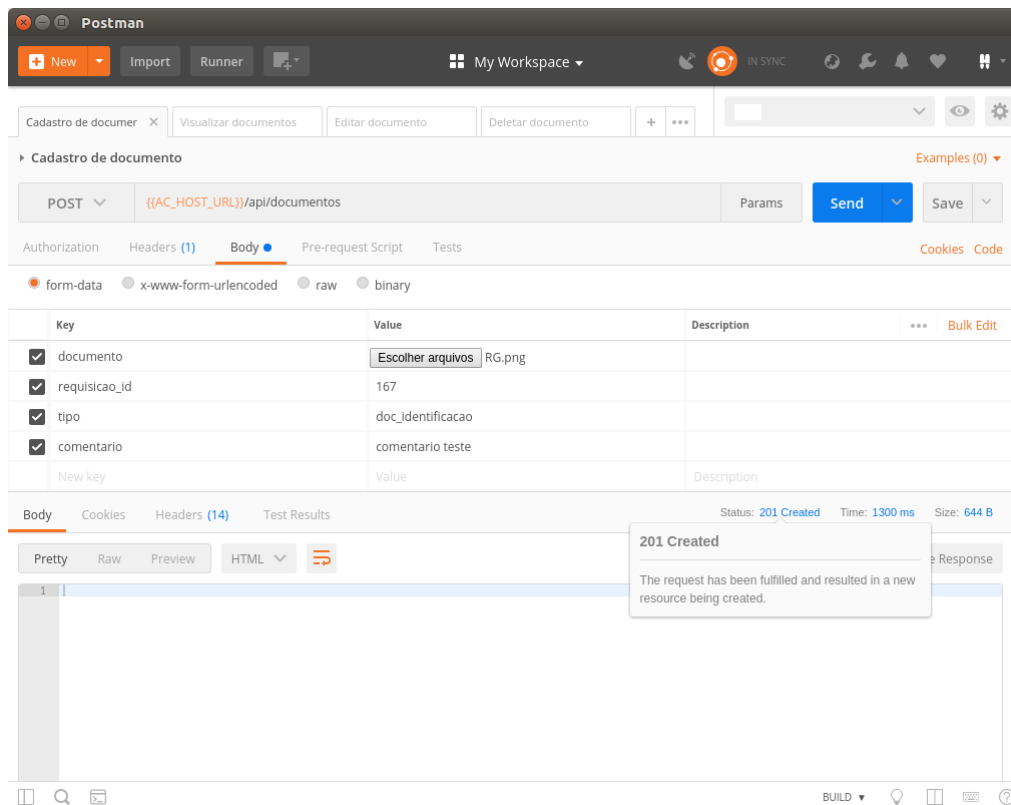


Figura 25 – Postman - Teste de cadastro de documento.

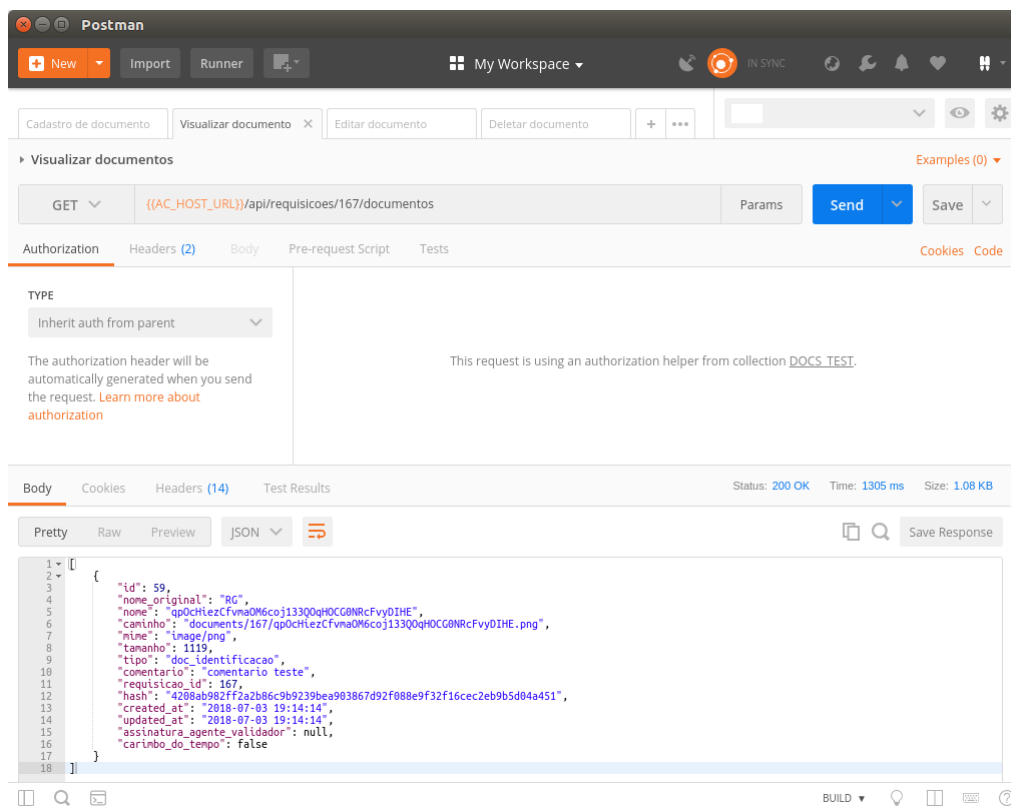


Figura 26 – Postman - Teste de listagem de documentos.

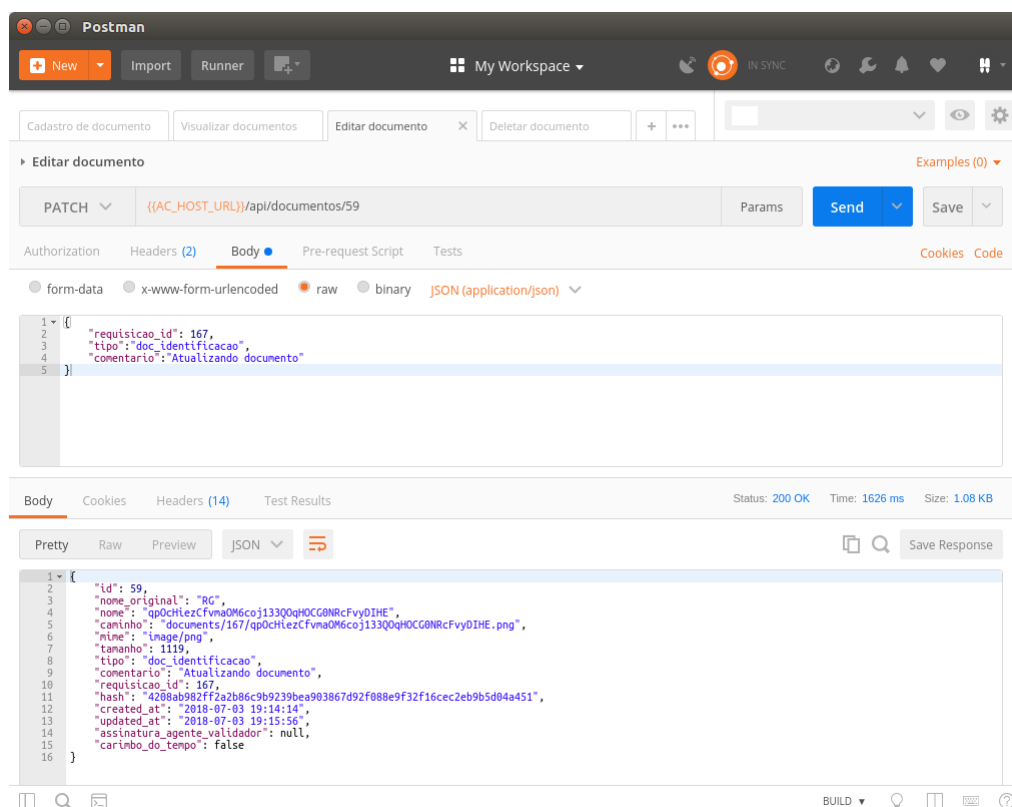


Figura 27 – Postman - Teste de edição de documento.

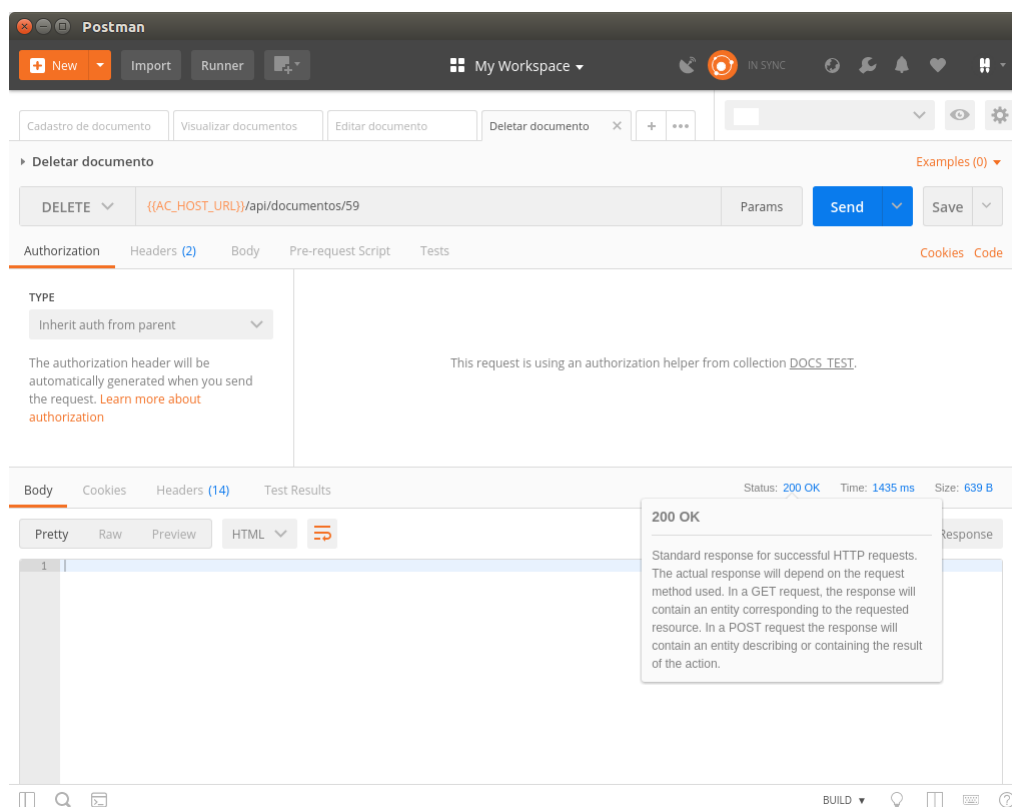
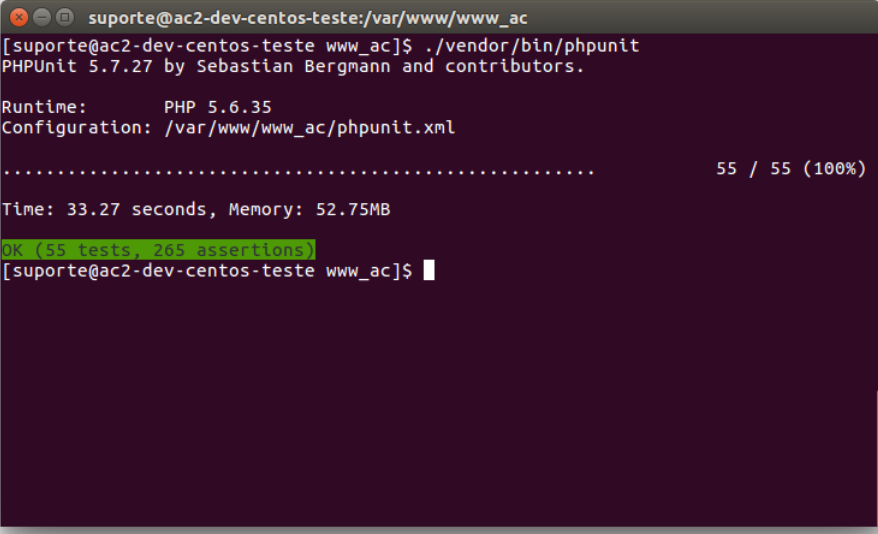


Figura 28 – Postman - Teste de exclusão de documento.



```
suporte@ac2-dev-centos-teste:/var/www/www_ac
[suporte@ac2-dev-centos-teste www_ac]$ ./vendor/bin/phpunit
PHPUnit 5.7.27 by Sebastian Bergmann and contributors.

Runtime:      PHP 5.6.35
Configuration: /var/www/www_ac/phpunit.xml

.....                                                55 / 55 (100%)

Time: 33.27 seconds, Memory: 52.75MB

OK (55 tests, 265 assertions)
[suporte@ac2-dev-centos-teste www_ac]$
```

Figura 29 – Execução de testes unitários.

- Serviço de gerenciamento (CRUD) da documentação do dossiê;
- Controle de aprovação da requisição.

Na parte de documentação, foram feitos testes para as operações CRUD (Create, Read, Update e Delete) diretamente na tabela do banco de dados, testes para a camada de repositório e, posteriormente, foram feitos testes utilizando as operações presentes na camada de serviço do *backend* da AC que provê as funcionalidades de manipulação da documentação em mais alto nível.

Já os testes para o controle de aprovação da requisição foram feitos testando diferentes tipos de políticas, validando quando um requisição não precisava de aprovação por parte de um Agente de Registro, quando a requisição precisava da assinatura do Agente de Registro para realizar a aprovação. Por fim, foram validadas requisições que não necessitavam da assinatura de um agente, mas precisavam de um Carimbo do Tempo. A Figura 29 demonstra a execução do PHPUnit que faz a verificação dos testes realizados. Pode-se perceber que os testes foram concluídos com sucesso.

### 6.3 Funcionalidades do Sistema

Nesta seção, são descritas as funcionalidades desenvolvidas juntamente com as telas criadas para o módulo/sistema.

The screenshot shows a web application interface titled 'Infraestrutura de Chaves Públicas' and 'AR PRINCIPAL - AC VINICIUS'. The main heading is 'Gerenciamento de Requisições e Certificados'. Below this is a search filter section with fields for 'Protocolo', 'Email', and 'CNPJ'. The 'Nome' field is filled with 'vinicius'. There is also a 'Status' dropdown menu set to 'Nenhum' and a 'Buscar' button. Below the search section is a table with the following data:

Disponível	Protocolo	Nome	Email	Status Req.
✓	AC8D7B3005451	Vinicius Teste	vinicius.menezes@...com.br	Aprovado
✓	7C396BC322418	Vinicius Dupla	vinicius.menezes@...com.br	Validado
✓	25FF7D5D39660	Vinicius	vinicius.menezes@...com.br	Aprovado
✓	9BCC4B97B915D	Vinicius	paulo3@...com.br	Aprovado
✓	5A655EC92CD7E	Vinicius Menezes	vinicius.menezes@...com.br	Reprovado
✓	6F1EAD22216DC	Vinicius Correa	vinicius.menezes@...com.br	Aprovado
✓	6C575D0D5D0A2	Vinicius Correa	vinicius.menezes@...com.br	Iniciado

Figura 30 – Tela - Gerenciamento de requisições.

### 6.3.1 Gerenciamento do dossiê

A Figura 30 ilustra a tela de gerenciamento de requisições quando acessada por um Agente de Registro. Esta é uma tela que já estava implementada no sistema da Autoridade de Registro e é através dela que o Agente de Registro tem acesso a todas as requisições de Certificados Digitais. Pode-se notar que foi realizado um filtro nas requisições de modo a mostrar somente as requisições cujo nome do titular é “vinicius”. Também pode ser visualizado o estado da requisição na última coluna da tabela presente na Figura 30.

Ao selecionar uma requisição, o Agente de Registro é direcionado para a tela ilustrada na Figura 31. As informações da requisição são divididas em diferentes “steps”, onde estamos interessados no “step” Documentos. É neste “step” que pode ser visualizado o módulo implementado que foi escopo deste trabalho. Pode ser observado que o gerenciamento da documentação é feita através de uma tabela contendo os documentos pertencentes à requisição e um menu acima da tabela que oferece as operações que atuam nos documentos.

As opções que ficam ativas inicialmente são as de Novo documento e Download Dossiê. Caso o Agente de Registro selecione algum documento na tabela, as outras opções do menu também são ativadas. Ao clicar na opção de Novo documento, um diálogo é

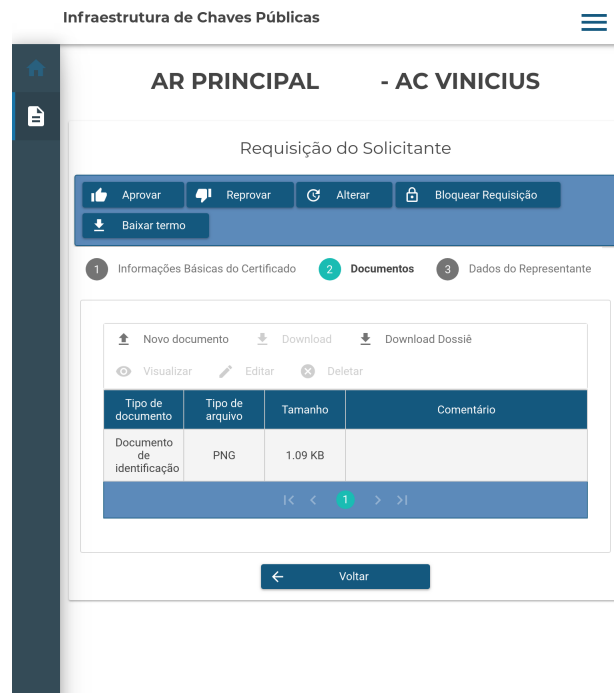


Figura 31 – Tela - Requisição.

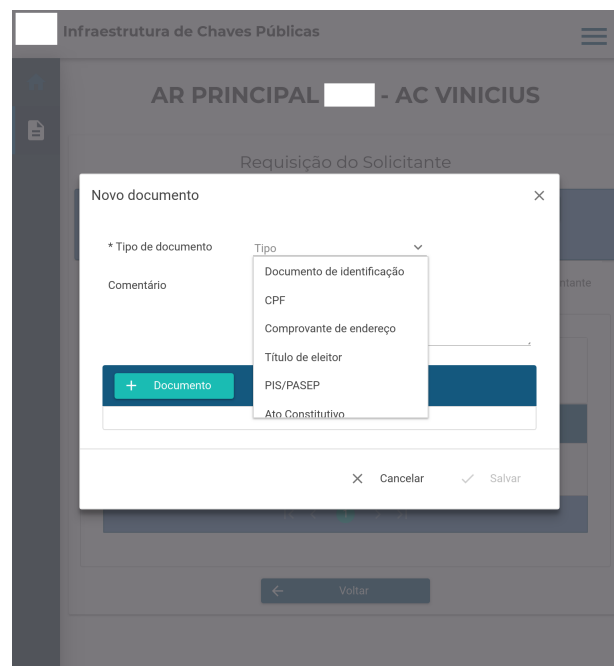


Figura 32 – Tela - Novo documento.

aberto na tela. Este diálogo pode ser visto na Figura 32.

Na tela de cadastro de documento é obrigatório selecionar o tipo do documento que se deseja cadastrar e selecionar o arquivo correspondente. A opção Salvar só fica disponível depois que os campos obrigatórios são preenchidos (Figura 33).

Ao clicar na opção de Editar, um diálogo similar ao de Novo documento é aberto,



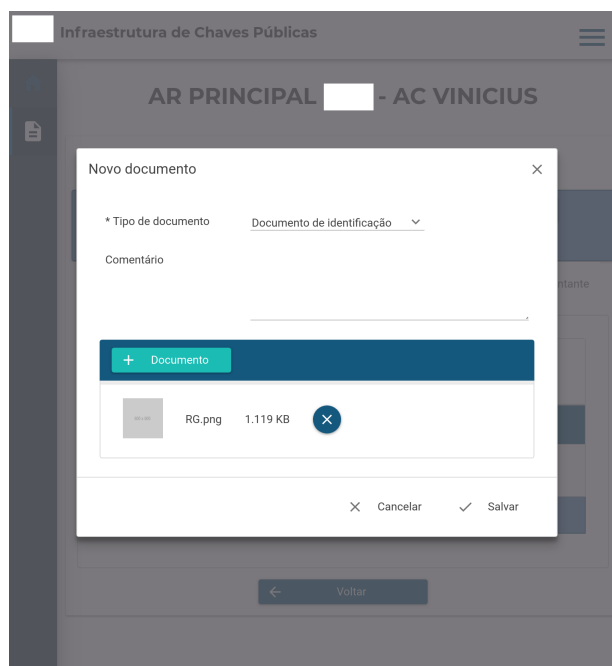


Figura 33 – Tela - Novo documento com campos preenchidos.

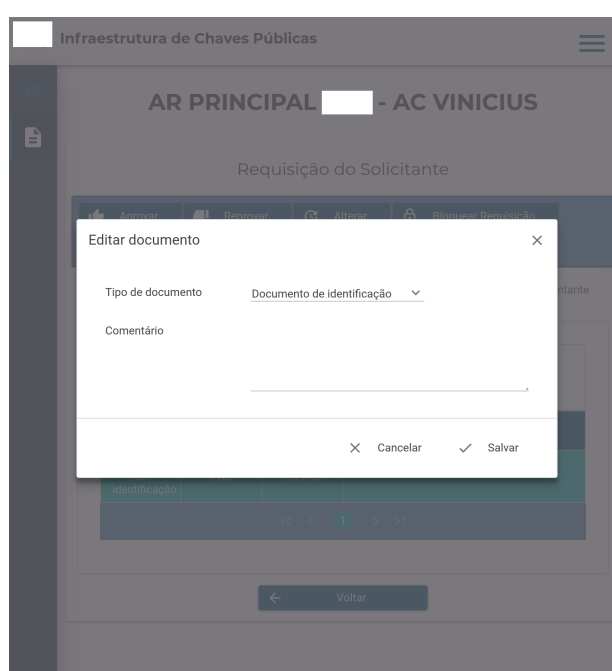


Figura 34 – Tela - Editar documento.

com a diferença de não ter a seleção de arquivo. O diálogo de edição pode ser visto na Figura 34.

Caso o Agente de Registro queira visualizar algum documento, através da opção visualizar presente no menu, um diálogo que ocupa a tela inteira aparece contendo a imagem ou PDF referente ao documento (Figura 35).

Ao tentar excluir um documento, um diálogo de confirmação é mostrado conforme

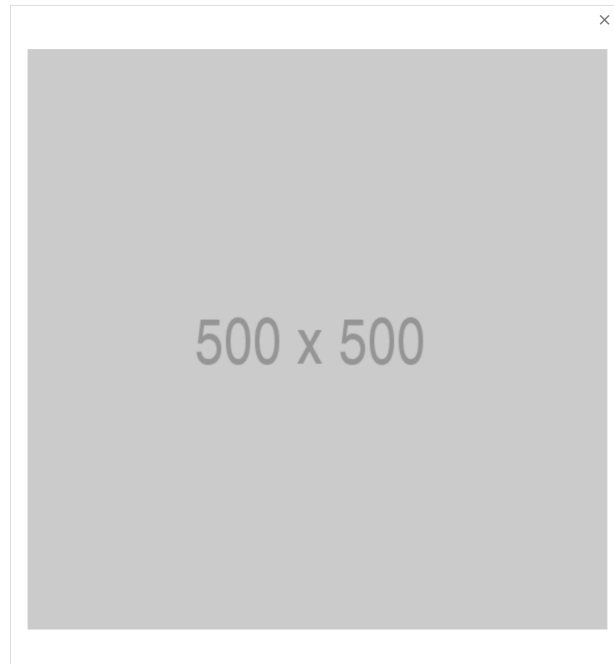


Figura 35 – Tela - Visualizar documento.

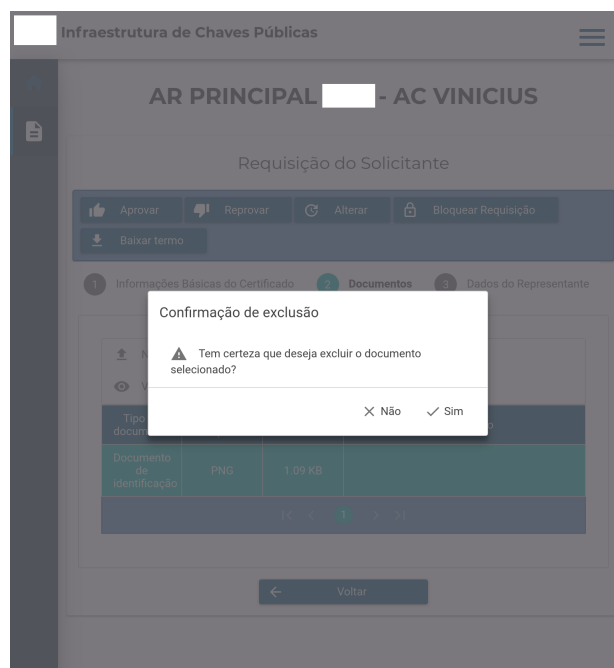


Figura 36 – Tela - Excluir documento.

a Figura 36.

### 6.3.2 Fluxo de aprovação

O Agente de Registro, após verificar e validar a documentação do titular da requisição, segue o fluxo para aprovação da mesma. Ao clicar em Aprovar, dependendo da política do Certificado Digital requerido, uma das duas telas ilustradas nas Figuras 37 e

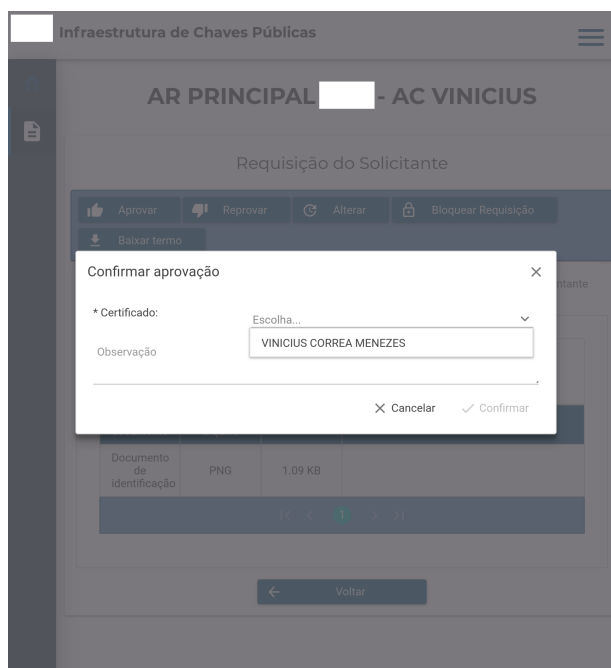


Figura 37 – Tela - Confirmar aprovação - Assinatura Agente de Registro.

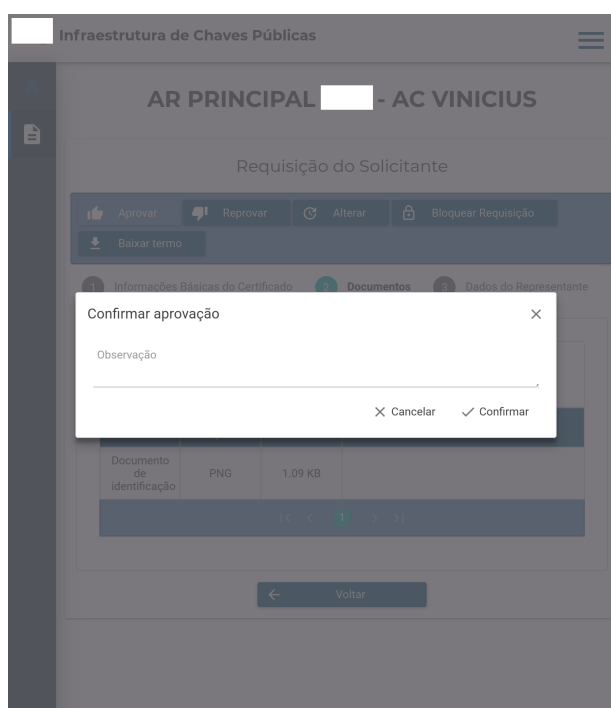


Figura 38 – Tela - Confirmar aprovação - Carimbo do Tempo.

38 são mostradas.

### 6.3.3 Assinatura Agente de Registro

No caso da política do Certificado Digital requisitar uma Assinatura Digital por parte do Agente de Registro, a tela da Figura 37 é mostrada. Nela, pode-se observar um

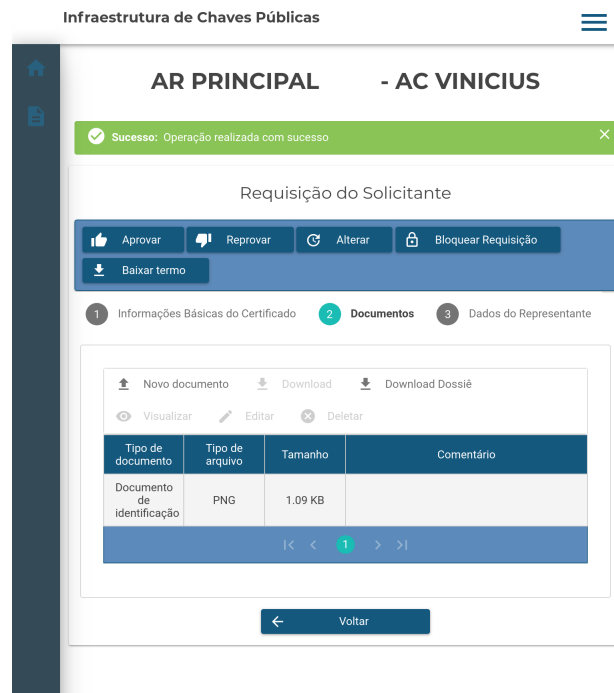


Figura 39 – Tela - Sucesso após aprovar requisição.

campo que mostra os Certificados Digitais disponíveis através da integração com uma extensão do navegador.

Autorizado o acesso aos Certificados Digitais e selecionado o certificado correspondente ao Agente de Registro logado na sessão, segue-se o fluxo de aprovação com o uso de Assinatura Digital do agente. As informações da requisição, dos documentos e a Assinatura Digital realizada em cima destes dados são enviadas então para o *backend* onde são validadas e caso tudo esteja correto, o estado da requisição é atualizado.

#### 6.3.4 Carimbo do Tempo

No caso da política do Certificado Digital requisitar uma Assinatura Digital, sendo que esta não precisa ser por parte do Agente de Registro, a tela da Figura 38 é mostrada. Nela, pode-se observar somente um campo para observações em relação à aprovação.

Neste caso são enviados os dados da requisição e dos documentos ao *backend*, sendo este o responsável por realizar a assinatura com Carimbo do Tempo nas informações recebidas. Se tudo estiver correto, o estado da requisição é atualizado.

A Figura 39 demonstra a mensagem de operação realizada com sucesso após a aprovação de uma requisição e a Figura 40 demonstra o estado atualizado da requisição após a aprovação.

Infraestrutura de Chaves Públicas

### AR PRINCIPAL - AC VINICIUS

Gerenciamento de Requisições e Certificados

Filtros

Nome: vinicius correa

Protocolo: \_\_\_\_\_

Email: \_\_\_\_\_

CNPJ: \_\_\_\_\_

Status: Nenhum

Última atualização entre: \_\_\_\_\_ e \_\_\_\_\_

Buscar

Aprovar/Rejeitar Histórico Revogar... Baixar termo...

Disponível	Protocolo	Nome	Email	Status Req.
✓	6F1EAD22216DC	Vinicius Correa	vinicius.menezes@...com.br	Aprovado

1 / 20

Figura 40 – Tela - Status da requisição atualizado.

## 6.4 Módulo Público

Através do desenvolvimento modular da parte de gestão de dossiê de uma requisição, foi possível sua fácil implementação no sistema do Módulo Público. Assim, um usuário, através do uso do MP, é capaz de enviar a documentação necessária para a emissão de um Certificado Digital, podendo enviar documentos ao realizar a solicitação ou depois através do uso do protocolo recebido. A Figura 41 ilustra a tela de solicitação de um Certificado Digital, mostrando o “step” da gestão de documentos e a Figura 42 ilustra a possibilidade de envio de documentos após receber o protocolo da solicitação realizada anteriormente.

Módulo Público

### Formulário de Solicitação de Certificados

Informações do Emissor

AC: **AC FINAL**

Selecione uma Política: **Política Teste**

1 Informações Básicas do Certificado 2 **Documentos** 3 Dados do Representante

Novo documento Visualizar Editar Deletar

Download

Tipo de documento	Tipo de arquivo	Tamanho	Comentário
Nenhuma documentação encontrada			

Legenda:

\*: Campos obrigatórios

Solicitar

Figura 41 – Tela - Gestão de documentos ao realizar solicitação no MP.

Módulo Público

### Consultar protocolo

\* Protocolo: **EAAB34E390512**

\* Senha emissão de certificado: **.....**

Consultar

### Dados do Certificado

Nome: Requisicao teste

CNPJ: 757.490.922-91

Email: vinicius.menezes@...com.br

Novo documento Visualizar Editar Deletar Download

Tipo de documento	Tipo de arquivo	Tamanho	Comentário
Nenhuma documentação encontrada			

Enviar documentos

Figura 42 – Tela - Gestão de documentos através do uso do protocolo no MP.

## 7 Conclusões

Como resultado deste trabalho, desenvolveu-se o módulo responsável pelo gerenciamento do dossiê de uma requisição de Certificado Digital. O escopo do trabalho foi o desenvolvimento de um módulo focado na sua implementação dentro do sistema da Autoridade de Registro. Como o seu desenvolvimento foi feito de forma modular, também foi possível a sua implementação dentro de um Módulo Público.

Deste modo, um usuário, através do uso do Módulo Público, é capaz de enviar todas as informações e documentos necessários para a emissão de um Certificado Digital, facilitando a validação por parte de um Agente de Registro e reduzindo o tempo na emissão do certificado. Um Agente de Registro também pode inserir os documentos durante a validação presencial. Dessa maneira, é garantida a segurança e rapidez na hora de acessar tais informações, eliminando a necessidade de procura física dos documentos.

O desenvolvimento do módulo foi composto por uma parte *frontend* contida no sistema da Autoridade de Registro, e duas partes *backend*, uma contida no sistema da Autoridade Certificadora e outra contida no sistema da Autoridade de Registro.

O uso dos *frameworks* Laravel e Angular se mostraram bastante produtivos, agilizando o desenvolvimento e evitando a duplicação de código. A metodologia Scrum utilizada ao decorrer do projeto se mostrou bastante eficiente, pois cada *Sprint* possuía tarefas bem objetivas possibilitando a entrega de pequenas versões do módulo em menos tempo.

Os testes realizados através do uso da ferramenta Postman e da biblioteca PHPUnit forneceram uma maneira de testar as funcionalidades que o módulo deveria oferecer antes do sistema ser colocado em produção, algo que é muito importante em desenvolvimento de sistemas atualmente.

No desenvolvimento do serviço de integração (REST), que possui características integradoras de Web Services, mas sem os custos da mesma, foi possível perceber que grandes empresas de diferentes segmentos estão usufruindo destas tecnologias para que suas soluções estejam cada vez mais integradas, e que o usuário (cliente) esteja sempre desfrutando de sua tecnologia em qualquer tipo de ambiente.

### 7.1 Perspectivas Futuras

Atualmente, os documentos ficam armazenados nos servidores da própria Autoridade de Registro cadastrada no sistema. No futuro, uma possibilidade é realizar o armazenamento dessa documentação em serviços fornecidos pela Amazon, como o S3 e o Glacier, seja como servidor principal ou como um servidor de *backup*. Segundo o site da

Amazon:

O Amazon S3 é um armazenamento de objetos criado para armazenar e recuperar qualquer quantidade de dados de qualquer local: sites e aplicativos móveis, aplicativos corporativos e dados de sensores. O serviço foi projetado para oferecer resiliência de 99,99% e armazena dados para milhões de aplicativos usados por líderes de mercado em todos os setores. O S3 oferece recursos abrangentes de segurança e conformidade que cumprem até os requisitos normativos mais rigorosos.

O Amazon Glacier é um serviço de armazenamento na nuvem seguro, durável e de custo extremamente baixo para arquivamento e *backup* de dados de longo prazo. Ele foi projetado para oferecer resiliência de 99,99% e oferece recursos abrangentes de segurança e conformidade que podem ajudar a cumprir até mesmo os requisitos normativos mais rigorosos.

A vantagem em utilizar estes serviços está em ter melhor segurança contra acidentes que possam ocorrer em servidores das Autoridades de Registro. Outro trabalho futuro seria a criação de um mecanismo automático de auditoria dos documentos, com geração de um relatório com informações de controle e auditoria.



## Referências

- 1 LIMA, M. F. de. *Assinatura Digital - Solução Delphi and Capicom*. [S.l.]: Visual Books, 2005. Citado na página 17.
- 2 DEMETRIUS, L. *Informática para concursos públicos*. [S.l.]: Clube de Autores, 2016. Citado 3 vezes nas páginas 21, 22 e 23.
- 3 SOFTPLAN. *Assinatura Digital de Documentos Eletrônicos no Brasil: Conceitos Básicos e Infraestrutura*. <[https://www.softplan.com.br/saj/downloads/cartilha\\_eletronica.pdf](https://www.softplan.com.br/saj/downloads/cartilha_eletronica.pdf)>. Citado 2 vezes nas páginas 28 e 29.
- 4 LAW, T. *O reconhecimento e a execução de sentenças arbitrais estrangeiras no Brasil*. [S.l.]: Livrus, 2016. Citado na página 28.
- 5 COULOURIS, G.; KINDBERG, T.; DOLLIMORE, J. *Sistemas Distribuídos: Conceitos e Projeto*. 4. ed. [S.l.]: Bookman, 2007. Citado 2 vezes nas páginas 29 e 30.
- 6 SALEMI, J. *Banco de Dados Cliente/Servidor*. Rio de Janeiro: IBPI Press, 1993. Citado na página 30.
- 7 SOMMERVILLE, I. *Engenharia de Software*. 7. ed. Rio de Janeiro: McGrawHill, 2011. Citado 2 vezes nas páginas 30 e 31.
- 8 PRESSMAN, R. S. *Engenharia de Software: uma Abordagem Profissional*. 9. ed. [S.l.]: Pearson Prentice Hall, 2011. Citado 2 vezes nas páginas 30 e 31.
- 9 MEDEIROS, H. *Princípios da Engenharia de Software*. 2017. <<http://www.devmedia.com.br/principios-da-engenharia-de-software/29630>>. Citado na página 30.
- 10 SILBERSCHATZ, A.; KORTH, H.; ABRAHAM, S. *Sistema de Banco de Dados*. 5. ed. [S.l.]: Elsevier, 2011. Citado 2 vezes nas páginas 31 e 32.
- 11 DATE, C. J. *Introdução a Sistemas de Bancos de Dados*. 8. ed. [S.l.]: Elsevier, 2004. Citado 2 vezes nas páginas 31 e 51.
- 12 NAVATHE, R. E. e S. B. *Sistema de Banco de Dados*. 7. ed. [S.l.]: Pearson, 2005. Citado na página 32.
- 13 MOZILLA. *HTML*. <<https://developer.mozilla.org/pt-BR/docs/Web/HTML>>. Citado na página 35.
- 14 FLANAGAN, D. *JavaScript. O Guia Definitivo*. [S.l.]: Bookman, 2012. Citado na página 35.
- 15 WELLING, L.; THOMSON, L. *PHP e MySQL Desenvolvimento Web*. [S.l.]: Elsevier, 2003. Citado na página 36.
- 16 MAYA, E. A. *Linguagem de Programação II (PHP)*. <[https://mafiadoc.com/linguagem-de-programacao-ii-php-alcides-maya-tecnologia\\_5a20ab8f1723dd87a652c6f7.html](https://mafiadoc.com/linguagem-de-programacao-ii-php-alcides-maya-tecnologia_5a20ab8f1723dd87a652c6f7.html)>. Citado na página 36.

- 17 GONÇALVES, E. C. *JSON Tutorial*. <<https://www.devmedia.com.br/json-tutorial/25275>>. Citado na página 36.
- 18 POSTGRES. *PostgreSQL Organization*. <<https://www.postgresql.org/about>>. Citado na página 36.
- 19 FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado) — University of California. Citado na página 37.
- 20 SAUDATE, A. *REST: Construa API's inteligentes de maneira simples*. [S.l.]: Casa do Código, 2014. Citado na página 37.
- 21 ANGULAR. *Angular Organization*. <<https://angular.io/guide/architecture>>. Citado na página 37.
- 22 FREEMAN, A. *Pro AngularJS - Learn to harness the power of modern web browsers from within your application's code*. [S.l.]: Apress, 2015. Citado na página 38.
- 23 JONNA, S.; VARAKSIN, O. *Angular UI Development with PrimeNG*. [S.l.]: Packt, 2017. Citado na página 38.
- 24 SILVA, W. A. L. *Laravel Tutorial*. <<https://www.devmedia.com.br/laravel-tutorial/33173>>. Citado na página 38.
- 25 NETO, M. F. *Tutorial da ferramenta de modelagem ASTAH*. <<https://goo.gl/mLF11o>>. Citado na página 39.
- 26 BALSAMIQ. *Mockups Application Overview*. <<https://docs.balsamiq.com/desktop/overview>>. Citado na página 40.
- 27 GUZZLE. *Guzzle Documentation*. <<http://docs.guzzlephp.org/en/stable>>. Citado na página 41.
- 28 TECHNOLOGIES, P. *Postman*. <<https://www.getpostman.com/products>>. Citado na página 41.
- 29 SILVERMAN, R. E. *Git - Guia Prático*. [S.l.]: Novatec, 2013. Citado na página 41.
- 30 FERREIRA, D. *Um Modelo Ágil para Gestão de Projectos de Software*. 2005. <<https://goo.gl/r1iHE9>>. Citado 2 vezes nas páginas 41 e 42.
- 31 ICP-BRASIL. *Características mínimas de segurança para as AR da ICP-Brasil*. 2005. <[http://www.iti.gov.br/images/repositorio/legislacao/documentos-principais/DOC-ICP-03.01\\_-\\_Versao\\_2.3\\_CHARACTERISTICAS\\_MINIMAS\\_DE\\_SEGURANCA\\_PARA\\_AS\\_AR\\_DA\\_ICP-BRASIL.pdf](http://www.iti.gov.br/images/repositorio/legislacao/documentos-principais/DOC-ICP-03.01_-_Versao_2.3_CHARACTERISTICAS_MINIMAS_DE_SEGURANCA_PARA_AS_AR_DA_ICP-BRASIL.pdf)>. Citado 2 vezes nas páginas 55 e 56.