

Augusto Amaral Araújo

Proposta de arquitetura para uso de certificados digitais em aplicações Laravel

Formiga - MG

2019

Augusto Amaral Araújo

Proposta de arquitetura para uso de certificados digitais em aplicações Laravel

Monografia do trabalho de conclusão de curso
apresentado ao Instituto Federal Minas Ge-
rais - Campus Formiga, como requisito parcial
para a obtenção do título de Bacharel em Ci-
ência da Computação.

Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais

Campus Formiga

Ciência da Computação

Orientador: Dr. Manoel Pereira Junior

Formiga - MG

2019

Augusto Amaral Araújo

Proposta de arquitetura para uso de certificados digitais em aplicações Laravel/ Augusto Amaral Araújo. – Formiga - MG, 2019.
60 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Manoel Pereira Junior

Monografia para trabalho de conclusão de curso (graduação) – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, Ciência da Computação, Formiga, 2019.

1. Palavra-chave1. 2. Palavra-chave2. 3. Palavra-chave3. I. Dr. Manoel Pereira Junior. II. Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais. III. Ciência da Computação. IV. Proposta de arquitetura para uso de certificados digitais em aplicações Laravel

Augusto Amaral Araújo

Proposta de arquitetura para uso de certificados digitais em aplicações Laravel

Monografia do trabalho de conclusão de curso apresentado ao Instituto Federal Minas Gerais - Campus Formiga, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Trabalho aprovado em 06 de junho de 2018.

BANCA EXAMINADORA

Dr. Manoel Pereira Junior
Orientador

Fulano
Convidado 1

Sicrano
Convidado 2

Formiga - MG
2019

DEDICATORIA: TODO

Agradecimientos

AGRADECIMENTOS: TODO

Resumo

// TODO

Palavras-chave:

Abstract

// TODO

Keywords:

Lista de ilustrações

| | |
|---|----|
| Figura 1 – Total de incidentes reportados ao CERT.br de 1999 a 2017. | 15 |
| Figura 2 – Processo de cifragem | 18 |
| Figura 3 – Criptografia Simétrica | 19 |
| Figura 4 – Criptografia Assimétrica | 20 |
| Figura 5 – Funcionamento básico de uma ICP tradicional | 21 |
| Figura 6 – Exemplo de Certificado A3 | 23 |
| Figura 7 – Uso de linguagens de programação no lado servidor de aplicações <i>web</i> . | 28 |
| Figura 8 – Fluxo aplicado no desenvolvimento do trabalho | 32 |
| Figura 9 – Tipos de Fluxo levantados | 33 |
| Figura 10 – Modelagem do fluxo das funcionalidades de autenticação e assinatura digital | 35 |
| Figura 11 – Organização dos arquivos do projeto | 36 |
| Figura 12 – Exemplo de janela de seleção do certificado no Firefox (informações ocultadas) | 37 |
| Figura 13 – Exemplo uso da funcionalidade Registrar com o Certificado do pacote . | 39 |
| Figura 14 – Exemplo de hierarquia de Autoridades Certificadoras | 49 |
| Figura 15 – Exemplo de formulario com opção para registrar utilizando o certificado | 57 |
| Figura 16 – Exemplo de formulario com opção para efetuar login utilizando o certificado | 58 |

Lista de tabelas

| | |
|---|----|
| Tabela 1 – Tecnologias <i>web</i> utilizadas na aplicação teste | 31 |
| Tabela 2 – Funcionalidades do protótipo do pacote desenvolvido | 41 |

Lista de códigos

| | | |
|------|---|----|
| 4.1 | Script utilizado para checar se um certificado é valido | 38 |
| 4.2 | Busca utilizada para encontrar um certificado cadastrado através do certificado exportado | 39 |
| 4.3 | Script que assina um arquivo e retorna a assinatura | 40 |
| A.1 | Exemplo de configuração de virtual host com SSL | 47 |
| A.2 | Exemplo de configuração para requisitar o certificado do cliente | 48 |
| A.3 | SSLCACertificateFile padrão utilizado nos testes | 49 |
| A.4 | Exemplo virtual host da aplicação | 51 |
| A.5 | Exemplo virtual host utilizado pelo pacote | 52 |
| A.6 | Exemplo de uso da função clientHasValidCert | 53 |
| A.7 | Exemplo de uso da função getClientCertificate | 53 |
| A.8 | Exemplo de uso da função certificateAlreadyRegistered | 54 |
| A.9 | Exemplo de uso da função getUserCertificates | 55 |
| A.10 | Exemplo de uso da função signFile | 55 |
| A.11 | Exemplo de uso da função checkFileSign | 56 |

Lista de abreviaturas e siglas

| | |
|-------|--|
| WEB | <i>World Wide Web</i> |
| HTTP | <i>Hypertext Transfer Protocol</i> |
| HTTPS | <i>Hyper Text Transfer Protocol Secure</i> |
| TLS | <i>Transport Layer Security</i> |
| RFC | <i>Request for Comments</i> |
| PHP | <i>Hypertext Preprocessor</i> |
| AC | Autoridade Certificadora |
| AR | Autoridade de Registro |
| ICP | Infraestrutura de chaves públicas |
| CSS | <i>Cascading Style Sheets</i> |
| HTML | <i>HyperText Markup Language</i> |
| PEM | <i>Internet Privacy Enhanced Mail</i> |
| CSR | <i>Certificate Signing Request</i> |
| SSL | <i>Secure Sockets Layer</i> |

Sumário

| | | |
|----------|--|-----------|
| 1 | INTRODUÇÃO | 15 |
| 1.1 | Justificativa | 16 |
| 1.2 | Objetivos | 16 |
| 1.3 | Estrutura do trabalho | 17 |
| 2 | FUNDAMENTAÇÃO TEÓRICA | 18 |
| 2.1 | Criptografia | 18 |
| 2.2 | Infraestruturas de Chaves Públicas | 20 |
| 2.3 | Certificado digital | 22 |
| 2.3.1 | Certificados A1 | 23 |
| 2.3.2 | Certificados A3 | 23 |
| 2.4 | Certificado X.509 | 24 |
| 2.5 | Assinatura digital | 26 |
| 2.6 | Linguagem PHP | 27 |
| 2.7 | Framework Laravel | 27 |
| 3 | MATERIAIS E MÉTODOS | 29 |
| 3.1 | PHP | 29 |
| 3.2 | PHP OpenSSL | 29 |
| 3.3 | Framework Laravel | 30 |
| 3.4 | Aplicação teste | 30 |
| 3.5 | Hardware | 31 |
| 3.6 | Metodologia | 32 |
| 4 | PROJETO E DESENVOLVIMENTO | 34 |
| 4.1 | Desenvolvimento da prova de conceito | 34 |
| 4.2 | Modelagem | 34 |
| 4.3 | Estrutura e organização | 36 |
| 4.4 | Obter e validar o certificado do cliente | 37 |
| 4.5 | Funcionamento | 38 |
| 4.6 | Autenticação | 39 |
| 4.7 | Assinatura Digital | 40 |
| 4.8 | Funcionalidades | 40 |
| 5 | RESULTADOS | 42 |

| | |
|--------------------|-----------|
| REFERÊNCIAS | 43 |
|--------------------|-----------|

| | |
|------------------|-----------|
| APÊNDICES | 45 |
|------------------|-----------|

| | |
|--|-----------|
| APÊNDICE A – DOCUMENTAÇÃO BÁSICA DO PACOTE DE- SENVOLVIDO | 46 |
|--|-----------|

| | | |
|------------|--|-----------|
| A.1 | Introdução | 46 |
| A.2 | Requisitos | 46 |
| A.3 | Instalação | 46 |
| A.4 | Configuração | 47 |
| A.4.1 | Apache 2 - Habilitando HTTPS | 47 |
| A.4.2 | Apache 2 - Requisitar certificados de clientes | 48 |
| A.4.3 | Apache 2 - Configurar subdomínio | 51 |
| A.5 | Funcionalidades | 53 |
| A.5.1 | Validar certificado | 53 |
| A.5.2 | Obter certificado | 53 |
| A.5.3 | Verificar se o certificado já foi cadastrado | 54 |
| A.5.4 | Obter certificados de determinado usuário | 54 |
| A.5.5 | Gerar assinatura digital | 55 |
| A.5.6 | Validar uma assinatura digital | 56 |
| A.6 | Funcionalidades do Controlador | 56 |
| A.6.1 | Certificado em JSON | 56 |
| A.6.2 | Registrar usuário utilizando o certificado | 57 |
| A.6.3 | Efetuar login utilizando o certificado | 57 |

| | |
|---------------|-----------|
| ANEXOS | 59 |
|---------------|-----------|

| | |
|---|-----------|
| ANEXO A – VARIÁVEIS DE AMBIENTE RELACIONADAS AO CERTIFICADO DO CLIENTE EXPORTADO DISPO- NÍVEIS NO APACHE | 60 |
|---|-----------|

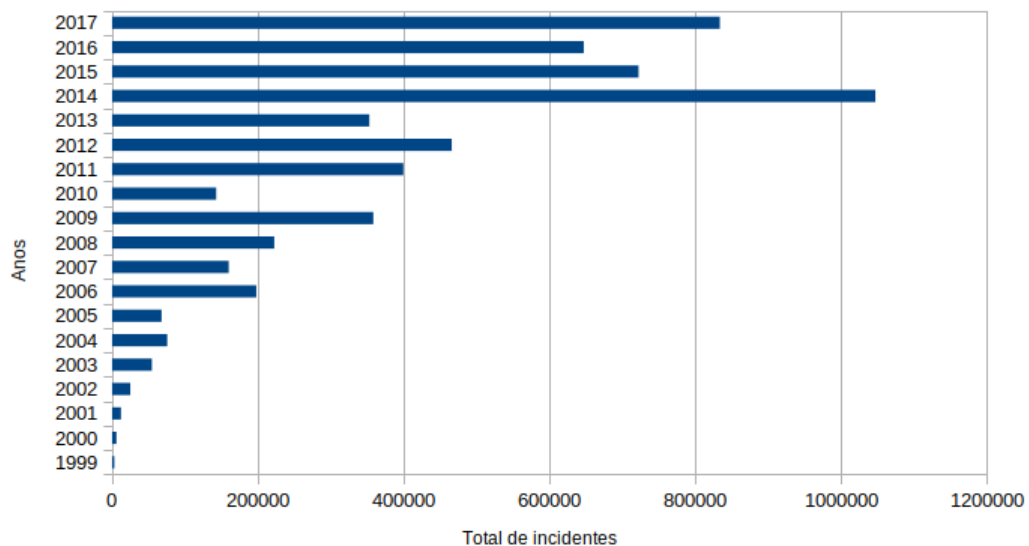
1 Introdução

O avanço tecnológico das últimas décadas vem promovendo mudanças significativas em diversos processos antes consolidados devido a facilidade no acesso da informação. Podemos citar por exemplo os comércios eletrônicos e o internet *banking*

Devido a comodidade e eficiência desses serviços online a tendência é que cada vez mais os clientes optem por essas alternativas e que elas se expandam a novas áreas do mercado. Essa tendência pode ser observada com pesquisas como a da [Febraban \(2018\)](#) que afirma que o internet *banking* já é o segundo serviço mais utilizado pelos clientes, totalizando cerca de 23%, e o *mobile banking* cresceu 70% em 2017 em relação a 2016.

Junto com a progressiva união dos processos com a tecnologia, também se observa o aumento na demanda para garantir a segurança da informação. A [Figura 1](#) exibe o total de incidentes de segurança entre 1999 a 2017 reportados ao o Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil. A integridade e autenticidade das informações neste tipo de serviço são de extrema importância visto a sensibilidade das informações que são trafegadas, como por exemplo informações pessoais ou bancarias.

Figura 1 – Total de incidentes reportados ao CERT.br de 1999 a 2017.



Fonte: ([CERT.BR, 2017](#))

A certificação digital é uma das tecnologias que vem sendo aplicadas para garantir a segurança da informação. Um certificado digital consiste basicamente em um arquivo eletrônico que serve como identidade de uma pessoa física ou jurídica. O certificado geralmente é utilizado na autenticação e assinatura digital que garante a um documento

digital a autenticidade, integridade e irretratabilidade.

No Brasil a lei brasileira determina que qualquer documento digital tem validade legal se for certificado pela ICP-Brasil. A ICP-Brasil é a infraestrutura de chaves públicas brasileira criada pela Medida Provisória 2200-2 de 2001, e oficializada pelo Decreto 3996 de 2001 e pela Lei 11419 de 2006. Uma ICP consiste basicamente em um conjunto de funções, cargos, políticas, *software*, *hardware* e procedimentos utilizados para gerenciar, armazenar e revogar certificados digitais. O mercado de certificação digital está crescendo no Brasil, de acordo com a ITI (2019) o número de certificados emitidos cresceu 23% em 2018 comparado a 2017.

Neste trabalho foi desenvolvido um protótipo de um pacote que integra funcionalidades relacionadas a certificação digital com aplicações web desenvolvidas com o *framework* Laravel. O pacote oferece suporte a autenticação e assinatura digital utilizando o certificado do tipo A1.

1.1 Justificativa

A certificação digital é uma tecnologia que esta sendo amplamente utilizada no mercado, no ano de 2018 foram emitidos 4.416.398 certificados, crescimento de 23,10% em comparação a 2017 (ITI, 2019). São mais de 26 milhões de certificados emitidos e cerca de 8 milhões ativos no padrão da ICP-Brasil.

O material disponível para desenvolvedores sobre a aplicação de certificação digital em aplicações web em geral é escasso. Não foi identificada nenhuma ferramenta para o *framework* Laravel que facilite a integração da certificação digital com aplicações Laravel, seja em autenticação ou assinatura digital.

1.2 Objetivos

Este trabalho tem como objetivo investigar o uso de certificados digitais em sistemas web em geral para desenvolver uma proposta de um *software* para a integração da certificação digital em aplicações web que utilizem o *framework* Laravel. Para alcançar o objetivo geral, faz-se necessário estabelecer alguns objetivos específicos, tais como:

- Levantar o estado-da-arte do uso de certificados digitais na web.
- Identificar os módulos de software necessários para melhor uso dos certificados digitais na web.
- Levantar os requisitos dos softwares identificados.

- Proposição de uma arquitetura para melhor uso dos certificados digitais nas aplicações web.

1.3 Estrutura do trabalho

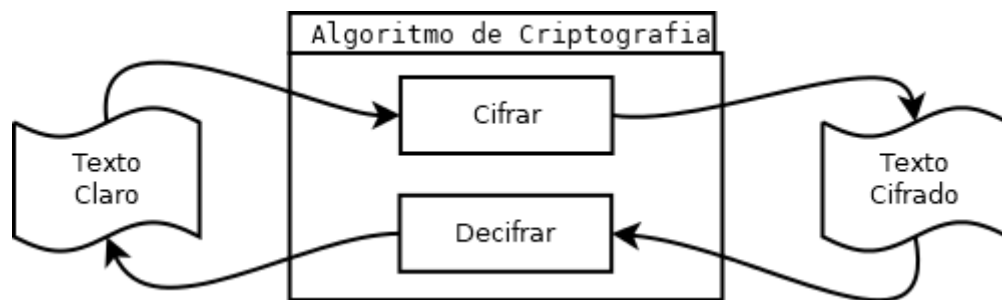
2 Fundamentação Teórica

Neste capítulo será abordado o embasamento teórico necessário para auxiliar o desenvolvimento do trabalho, especificamente nas áreas de criptografia e segurança, detalhando o processo de certificação digital, tipos de certificados e no *software o framework* Laravel e a linguagem de programação PHP.

2.1 Criptografia

A criptografia pode ser entendida como um conjunto de métodos e técnicas para cifrar ou codificar informações legíveis por meio de um algoritmo, convertendo um texto original em um texto ilegível, sendo possível mediante o processo inverso recuperar as informações originais (SINGH, 1999). Veja o processo na [Figura 2](#).

Figura 2 – Processo de cifragem



Fonte: Próprio autor

Basicamente, para se criptografar uma mensagem é utilizado códigos ou cifras. Um código manipula o significado da mensagem, normalmente pela substituição de frases e palavras. A cifra é aplicada na representação da mensagem, por meio da transposição e/ou substituição das letras da mensagem original.

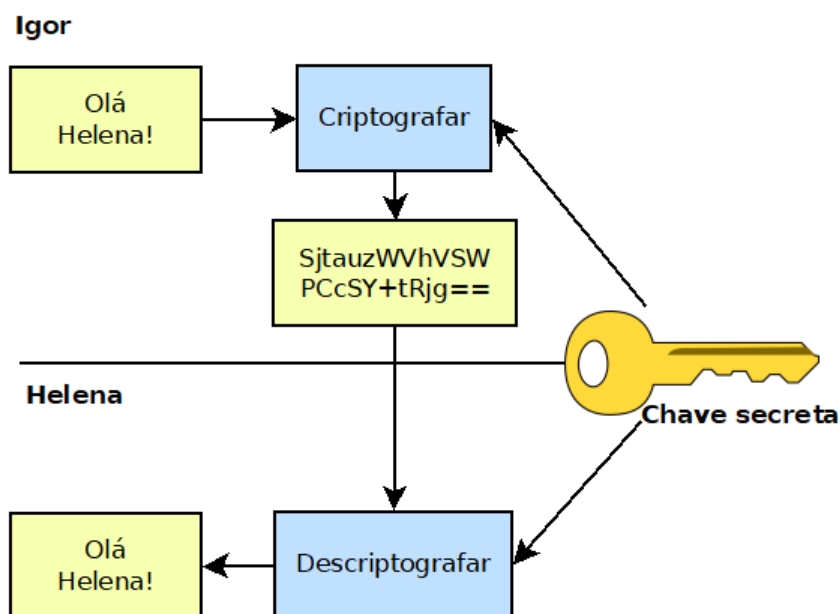
Criptografar uma mensagem consiste no ato de transformá-la em uma mensagem ilegível. A intenção é garantir a privacidade mesmo que a mensagem seja interceptada. O processo inverso é denominado de descryptografar, ou seja, transformar uma mensagem criptografada em sua forma original.

Normalmente os algoritmos de criptografia utilizam uma chave como parâmetro, sendo geralmente uma pequena sequência de caracteres ou resposta a algoritmos matemáticos. A chave deve ser idealmente secreta, conhecida apenas pelos comunicantes.

A criptografia é dividida em dois tipos: simétrica e assimétrica. Essa divisão é justificada pela diferença em como as chaves são utilizadas em cada tipo.

A criptografia simétrica refere-se aos métodos criptográficos onde o remetente e o destinatário compartilham da mesma chave. A chave é utilizada para cifrar e decifrar a informação, conforme exemplifica a [Figura 3](#).

Figura 3 – Criptografia Simétrica



Fonte: Próprio autor

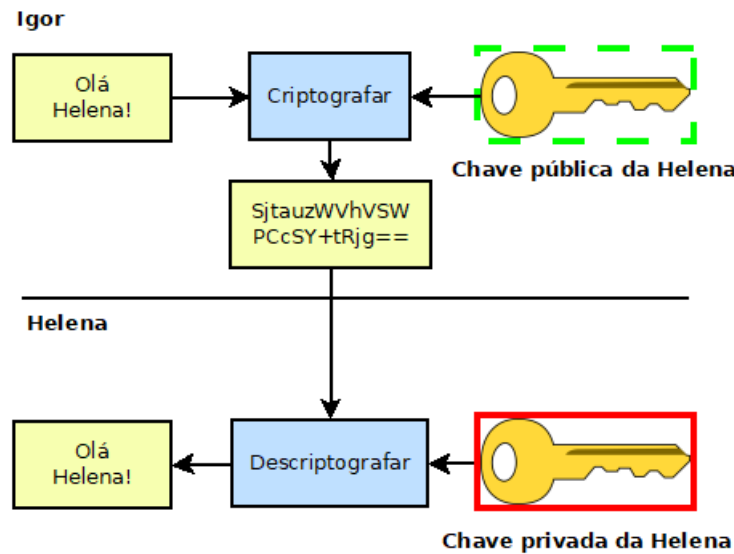
Os algoritmos de criptografia simétrica são mais simples e apresentam melhor desempenho se comparados aos de criptografia assimétrica. A grande desvantagem desses algoritmos é a necessidade de que todas as partes conheçam a mesma chave, gerando uma complexidade necessária no gerenciamento das chaves para usá-las com segurança.

A criptografia assimétrica foi proposta em 1976 por [Diffie e Hellman](#), também conhecida como criptografia de chaves públicas, introduziu o conceito da assinatura digital em documentos eletrônicos. Essa técnica utiliza um sistema de duas chaves distintas. Uma chave do par é pública e pode ser divulgada, e a outra é privada, que deve ser conhecida somente pelo seu proprietário. Se criptografar a mensagem com a chave privada ela somente será descriptografada pela chave pública e vice-versa. A [Figura 4](#) exemplifica o processo.

Nos sistemas de criptografia assimétrica a chave pública pode ser distribuída livremente, enquanto a privada deve se manter em segredo. Esse fator elimina a necessidade de um canal seguro para a troca inicial das chaves, tornando o gerenciamento das chaves mais simples e prático.

Os algoritmos de criptografia assimétrica geralmente são mais complexos e apresentam um pior desempenho em comparação aos algoritmos simétricos.

Figura 4 – Criptografia Assimétrica



Fonte: Próprio autor

2.2 Infraestruturas de Chaves Públicas

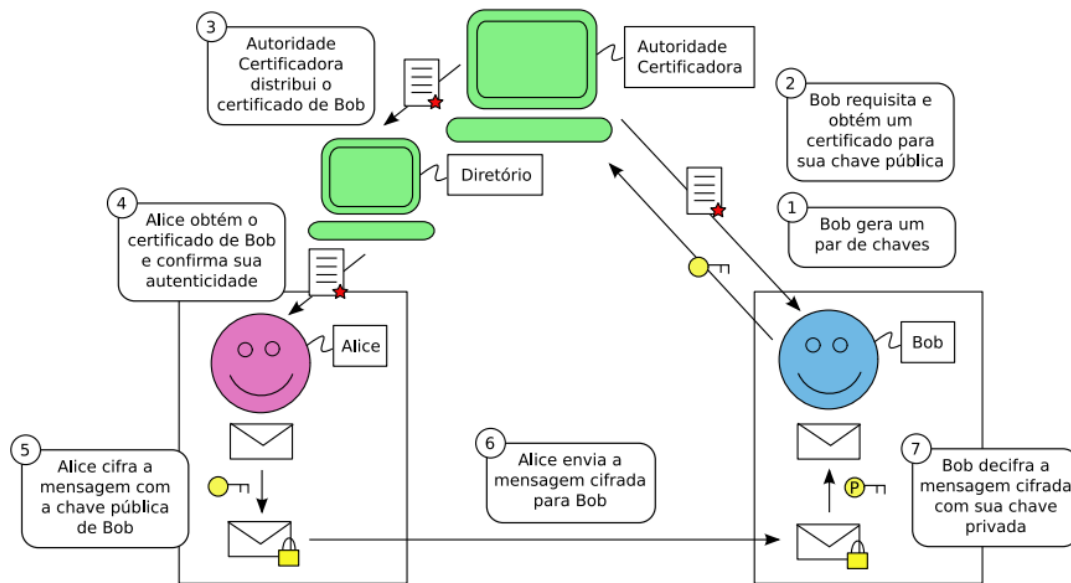
As “Infraestruturas de Chaves Publicas (ICPs, em inglês *PKI - Public Key Infrastructures*) são sistemas de recursos, políticas, e serviços que suportam a utilização de criptografia de tecla pública para autenticar as partes envolvidas na transação” (IBM, 2019). As ICPs já são amplamente utilizadas, por exemplo, no protocolo TLS (DIERKS; RESCORLA, 2008) na navegação de páginas na internet, e recentemente vem sendo utilizadas nos certificados digitais com destaque a Infraestrutura de Chaves Publicas Brasileira (ICP-BRASIL, 2019). O funcionamento básico de uma infraestrutura de chaves públicas está a representado na Figura 5

Não há um único padrão que defina os componentes de uma infraestrutura de chave pública, mas uma infraestrutura de chave pública geralmente inclui as seguintes entidades:

- **Autoridade Certificadora Raiz:** É a primeira autoridade da cadeia de certificação e esta encarregada de fiscalizar e auditar as outras entidades.
- **Autoridade Certificadora:** Autoridade responsável principalmente por emitir, distribuir, renovar, revogar e gerenciar certificados digitais.
- **Autoridade de Registro :** É a interface entre o usuário e uma AC.

Uma ICP vincula chaves públicas com as respectivas identidades de entidades (como pessoas físicas e jurídicas). Geralmente esta ligação é estabelecida por uma AC que realiza o registro e a emissão do certificado.

Figura 5 – Funcionamento básico de uma ICP tradicional



Fonte: (ARANHA, 2007)

A principal função da AC é assinar e publicar digitalmente a chave pública vinculada a determinado usuário. Esta operação utiliza a própria chave privada da AC, para que confiança da chave do usuário dependa da confiança da chave da AC. Se a AC for um terceiro sistema separado do usuário e do sistema ela é chamada de AR (SAMUELLE, 2011).

Exemplos de uso de ICPs (YANG, 2010):

- **Validação da identidade de um servidor web:** Valide a identidade do servidor da Web, verifique a assinatura digital da CA no certificado.
- **Criptografia de documentos:** O remetente de uma mensagem digital usa o certificado do destinatário para criptografar a mensagem para proteger a confidencialidade da mensagem. Apenas o destinatário que pode usar sua chave privada para descryptografar a mensagem.
- **Identificação digital:** O certificado do usuário é armazenado em um cartão inteligente para ser usado para verificar as identidades do titular do cartão.
- **Assinatura digital:** o remetente de uma mensagem digital usa sua chave privada para gerar uma assinatura digital anexada à mensagem. O destinatário usa o certificado do remetente para verificar a assinatura digital para garantir que a mensagem foi enviada pelo remetente reivindicado.

2.3 Certificado digital

Um Certificado digital, também conhecido como certificado de chave pública (*public key certificate*) ou certificado de identidade, é um documento eletrônico que prova a propriedade de uma chave pública. Este documento contém informações sobre a chave, o seu proprietário e da entidade que verificou o conteúdo do certificado.

Em uma ICP o certificado é normalmente assinado pela AC que emitiu o certificado, mas existem outros modelos como o de Teia de Confiança (*Web of trust*), onde o certificado é assinado pela própria entidade e assinado por outros que confiam naquela entidade. As assinaturas presente no certificado são atestamentos feitos por uma ou mais entidades que dizem confiar nos dados contidos naquele certificado.

Um certificado normalmente possui:

- As assinaturas da(s) entidade(s) que afirmam a autenticidade da relação entre o certificado e a chave publica contida nele.
- Período de validade.
- A chave pública referente a chave privada de posse da entidade especificada no certificado.
- Informações sobre a entidade que emitiu o certificado (Email, CNPJ, PIS, etc) .
- Localização do "centro de revogação", utilizado para verificar a validade do certificado.

O padrão mais comum para certificados digitais é o X.509, definido na [seção 2.4](#). Este padrão é ainda mais restrito por perfis definidos para certos casos de uso, como a ICP (X.509), conforme definido no RFC 5280 ([COOPER, 2008](#)).

Os certificados digitais são utilizados no protocolo TLS ([DIERKS; RESCORLA, 2008](#)), onde o proprietário é geralmente vinculado a um computador ou outro dispositivo. O TLS faz parte do HTTPS, o protocolo para navegação segura na web.

No mercado o certificado digital é utilizado como uma identidade virtual associando a identidade de uma pessoa física ou jurídica a uma chave pública. Dentre os serviços são feitos utilizando a certificação digital destaca-se: realizar transações bancárias, ter acesso e transmissão de informações a programas de governo como Receita Federal do Brasil, assinar documentos, assinar contratos e autenticação de serviços em geral.

Em 2001 com a criação da Infraestrutura de Chaves Públicas Brasileira – ICP Brasil, documentos digitais passaram a ter validade jurídica em todo Brasil.

No Brasil, dois tipos de certificados são os mais comuns, o Certificado A1 e A3.

2.3.1 Certificados A1

O Certificado A1 é um arquivo gerado por *software* que possui uma chave de 1024 *bits* e pode ser acessado por *login* e senha, geralmente fica instalado em um computador da empresa e apresenta menor custo.

O certificado é instalado no sistema e é requisitado diretamente, tudo isso em um processo automatizado sem a necessidade que usuários participem do processo.

Devido a automatização do processo a senha geralmente só é utilizada caso seja necessária a remoção do certificado de um computador para outro, possibilitando assim maior sigilo pois as senhas não precisam ser conhecidas por todos os usuários.

Com o certificado A1 é possível utilizar a certificação para emissões de nota fiscal eletrônica, nota fiscal de serviço eletrônica e nota fiscal de consumidor eletrônica simultaneamente.

O certificado só é valido por doze meses (um ano a partir da instalação) sendo necessário a sua renovação ou troca ao fim deste período.

Devido o certificado ser um arquivo eletrônico é imprescindível uma cópia *backup* do *software* por segurança, pois em caso de erro ou problema com a máquina/servidor onde está instalado, o certificado poderá ser perdido.

2.3.2 Certificados A3

O Certificado A3 é baseado em *hardware*, seja em *token* (USB) ou cartão com leitor específico em conformidade com a legislação da ICP-Brasil. Possui uma chave de 2048 *bits*, sendo mais seguro que outros certificados (como o A1, por exemplo) e tendo como principal vantagem a mobilidade, por ser gerado em um *token* ou cartão, o certificado pode ser levado e instalado em qualquer computador.

Figura 6 – Exemplo de Certificado A3



Fonte: (KHARITONOV, 2009)

Diferente do certificado A1 o A3 pode ter validade com duração de até três anos.

Para utilizar o certificado do tipo A3 é necessário usar a senha em cada uso, como consequência todos os usuários deverão saber a senha para que possam utilizá-lo.

Como o certificado é gerado em um *token* ou cartão, corre o risco de roubo, extravio ou dano, que podem invalidar seu uso.

Sua principal desvantagem é o motivo porque muitos sistemas só aceitem o A1 é que o certificado A3 só pode ser utilizado em um dispositivo por vez.

2.4 Certificado X.509

O Certificado X.509 é um certificado que usa o padrão X.509. Este padrão é utilizado por vários protocolos da Internet, inclusive no TLS/SSL ([DIERKS, 1999](#)), que é base do HTTPS.

O ITU-T X.509 (formalmente CCITT X.509) foi lançado inicialmente em 3 de julho de 1998 como parte das recomendações do padrão X.500 ([HOUSLEY, 1999](#), Cap 3.1). O formato do certificado no padrão de 1988 é chamado de formato da versão 1 (v1). Em 1993 o padrão X.500 foi revisitado e mais dois campos adicionais foram adicionados, resultando na versão 2 (v2) do formato. Esses dois campos adicionais podem ser usados para suporte de controle de acesso ao diretório.

Os RFCs do *Internet Privacy Enhanced Mail* publicados em 1993, incluem especificações para uma infraestrutura de chave pública baseada nos certificados X.509 v1 ([KENTR, 1998](#)). A experiência adquirida nas tentativas de implantar a RFC 1422 deixou claro que os formatos dos certificados v1 e v2 são deficientes em vários aspectos. Eram necessários mais campos para transportar informações que o design do PEM e a experiência de implementação se mostraram necessárias. Em resposta a esses novos requisitos, o ISO, IECM, ITU-T e o ANSI X9 desenvolveram o formato de certificado X.509 versão 3 (v3). O formato v3 estende o formato v2 incluindo a inclusão de campos de extensão adicionais. Tipos de campos de extensão específicos podem ser especificados em padrões ou podem ser definidos e registrados por qualquer organização ou comunidade. Em junho de 1996, a padronização do formato básico v3 foi concluída.

A estrutura de um certificado digital X.509 v3 ([HOUSLEY, 1999](#), Cap 4) é a seguinte:

- Certificado
 - Número da versão
 - Número de série
 - ID do Algoritmo de Assinatura
 - Nome do Emissor

- Período de validade
 - * Não antes
 - * Não após
- Nome do tópico
- Informações sobre a chave pública do assunto
 - * Algoritmo de chave pública
 - * Chave pública do assunto
- Identificador único do emissor (opcional)
- Identificador exclusivo do assunto (opcional)
- Extensões (opcional)
 - * ...
- Algoritmo de Assinatura de Certificado
- Assinatura de Certificado

As extensões definidas para os certificados X.509 v3 fornecem métodos para associar atributos adicionais com usuários ou chaves públicas e para gerenciamento de relacionamentos entre autoridades de certificação. O formato do certificado X.509 v3 também permite que as comunidades definam extensões privadas para transportar informações exclusivas para essas comunidades. Cada extensão em um certificado é designada como crítica ou não crítica. O sistema de uso de certificados deve rejeitar o certificado se encontrar uma extensão crítica que não reconhece ou uma extensão crítica que contém informações que não podem ser processadas. Uma extensão não crítica pode ser ignorada se não for reconhecida, mas deve ser processada se for reconhecida (COOPER, 2008, Cap 4.2).

No sistema X.509, para que uma organização tenha um certificado assinado primeiro ela deve gerar um par de chaves, mantendo em segredo a chave privada e a usando para assinar o CSR, que deve conter o *Common Name* e a chave pública, podendo ser acompanhado por outras credenciais ou provas de identificação exigidas pela AC. A autoridade de certificação emite um certificado que vincula uma chave pública a um *Common Name* específico.

Os navegadores como Firefox, Internet Explorer, Chrome, Opera e Safari já possuem um conjunto predeterminado de certificados raiz pré instalados, facilitando o funcionamento dos principais certificados SSL. Uma organização pode distribuir os certificados raiz confiáveis a todos os funcionários para que eles possam usar o sistema ICP da empresa.

Uma etapa importante da validação dos certificados são os caminhos de certificação citada na RFC 5280(COOPER, 2008, Cap 3.2): Se o usuário de um serviço de segurança

solicita o reconhecimento de uma chave pública geralmente é preciso obter e validar um certificado contendo a chave pública requisitada. Em geral, uma cadeia de múltiplos certificados pode ser necessária, incluindo o certificado do proprietário da chave pública (a entidade final) assinado por uma AC e zero ou mais certificados adicionais das ACs assinados por outras ACs. Essas cadeias, chamadas de caminhos de certificação, são necessárias porque um usuário de chave pública é inicializado apenas com um número limitado de chaves públicas de ACs.

2.5 Assinatura digital

Uma assinatura digital é um processo matemático para verificar a autenticidade de mensagens ou documentos digitais.

As assinaturas digitais são usadas na maioria dos protocolos criptográficos e são comumente usadas para transações financeiras, distribuição de *software*, *software* de gerenciamento de contratos e em outros casos em que é importante detectar falsificações ou adulterações.

Existem vários métodos para assinar documentos digitalmente, e esses métodos estão em constante transformação, mas basicamente um esquema de assinatura digital consiste tipicamente em 3 algoritmos:

1. Um algoritmo de geração de chaves que seleciona uma chave privada de maneira uniforme e aleatória a partir de um conjunto de possíveis chaves privadas. O algoritmo gera a chave privada e uma chave pública correspondente.
2. Um algoritmo de assinatura que, dada uma mensagem e uma chave privada, produz uma assinatura.
3. Um algoritmo de verificação de assinatura que, dada a mensagem, chave pública e assinatura, aceita ou rejeita a reivindicação de autenticidade da mensagem.

As duas propriedades principais necessárias consistem em primeiro, a autenticidade de uma assinatura gerada a partir de uma mensagem fixa e chave privada fixa pode ser verificada usando a chave pública correspondente. Em segundo lugar, deve ser computacionalmente inviável gerar uma assinatura válida para uma parte sem conhecer a chave privada dessa parte.

Geralmente no processo da assinatura digital é gerado um *hash* da mensagem através de uma função criptográfica de *hash*, que deve apresentar necessariamente as seguintes características:

- Deve ser impossível encontrar a mensagem original a partir do *hash* da mensagem.

- O *hash* deve parecer aleatório, mesmo que o algoritmo seja conhecido. Uma função de *hash* é dita forte se a mudança de qualquer bit na mensagem original resulta em um novo *hash* totalmente diferente.
- Deve ser impossível encontrar duas mensagens diferentes que levam a um mesmo *hash*.

Após gerar o *hash* da mensagem, o *hash* é criptografado através de um sistema de chave pública, para garantir a autenticação e a irretratabilidade. O autor da mensagem deve usar sua chave privada para assinar a mensagem e armazenar o *hash* criptografado junto à mensagem original.

Para verificar a autenticidade do documento, deve ser gerado um novo resumo a partir da mensagem que está armazenada, e este novo resumo deve ser comparado com a assinatura digital. Para isso, é necessário descriptografar a assinatura obtendo o *hash* original. Se ele for igual ao *hash* recém gerado, a mensagem está íntegra. Além da assinatura existe o selo cronológico que atesta a referência de tempo à assinatura.

2.6 Linguagem PHP

O PHP é uma linguagem multiparadigma de código aberto de uso geral utilizada principalmente para o desenvolvimento do lado servidor de aplicações web . Criada originalmente por Rasmus Lerdorf em 1994 e agora produzida pelo PHP Group.

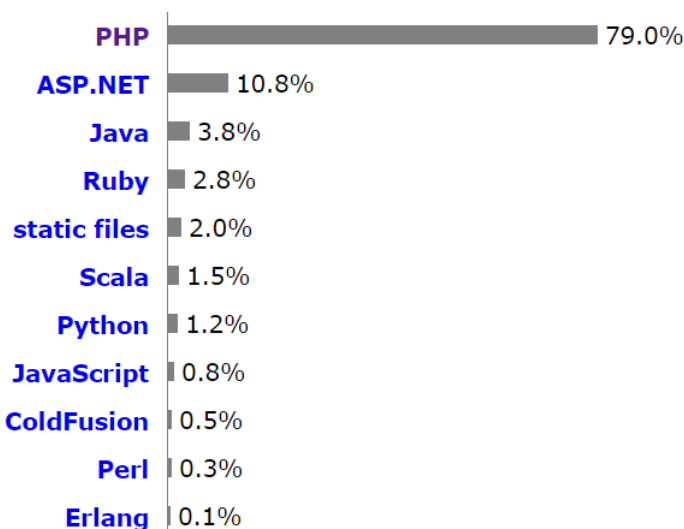
O PHP pode ser executado através de uma interface de linha de comando ou ser inserido dentro do HTML. O código geralmente é interpretado e executado por um modulo do servidor web que gera e retorna o resultado. O resultado pode ser qualquer tipo de dado, como um html, json, imagem ou um arquivo binário.

A [Figura 7](#) representa a utilização das linguagens, segundo a [W3Techs \(2019\)](#), no lado servidor das aplicações *web*. A linguagem PHP é a mais popular, utilizada em cerca de 79% dos servidores.

2.7 Framework Laravel

Segundo [Willemann e Ibarra \(2005, p.41\)](#):

Um framework ou arcabouço é uma estrutura de suporte definida em que um outro projeto de software pode ser organizado e desenvolvido, quando se analisa o conceito no âmbito do desenvolvimento de software. Um framework pode incluir programas de suporte, bibliotecas de código, linguagens de script e outros softwares para ajudar a desenvolver e juntar diferentes componentes de um projeto de software.

Figura 7 – Uso de linguagens de programação no lado servidor de aplicações *web*

Fonte: W3Techs (2019)

Laravel¹ é um *Framework* PHP grátis e *open-source* criado por Taylor Otwell em 2011 focado no desenvolvimento de aplicações web que seguem o padrão *model-view-controller* (MVC) de arquitetura de software. Seu principal objetivo é auxiliar no desenvolvimento de aplicações mais seguras, rápidas e com código limpo e simples, utilizando o padrão PSR-2² como guia de escrita de código.

Taylor Otwell criou o Laravel como uma alternativa à estrutura de outro framework de PHP CodeIgniter³, que não oferecia certos recursos, como suporte a autenticação e autorização do usuário. O beta do Laravel foi disponibilizado em 9 de junho de 2011, e sua versão 1 lançada no mesmo mês.

Desde o Laravel 3 o *framework* utiliza um sistema de empacotamento modular, com recursos para a fácil adição de pacotes as aplicações, e a partir da versão 4, o Composer é utilizado como um gerenciador de dependências do *framework*. Composer⁴ é uma ferramenta de gerenciamento de dependências para a linguagem de programação PHP, que permite declarar as bibliotecas que determinado projeto utiliza e as instala/atualiza.

O *framework* Laravel possui uma interface de linha de comando chamada *Artisan*⁵, que foi introduzida no Laravel 3. O *Artisan* fornece vários comandos úteis que podem ajudar no gerenciamento e criação de aplicações Laravel. Geralmente utilizado no gerenciamento de migrações e propagação de banco de dados, publicação de pacotes e geração de código.

¹ Disponível em <<https://laravel.com/>>. Acesso em janeiro de 2019

² Disponível em <<https://www.php-fig.org/psr/psr-2/>>. Acesso em março de 2019

³ Disponível em <<https://codeigniter.com/>>. Acesso em março de 2019

⁴ Disponível em <<https://getcomposer.org/>>. Acesso em março de 2019

⁵ Disponível em <<https://laravel.com/docs/5.8/artisan>>. Acesso em março de 2019

3 Materiais e Métodos

Neste capítulo são apresentados os materiais e a metodologia utilizados para o desenvolvimento do protótipo do pacote de certificação e da aplicação teste, tais como as bibliotecas, configurações e ferramentas utilizadas para a execução deste projeto.

3.1 PHP

O paradigma escolhido no desenvolvimento foi a de orientação a objetos. O gerenciador de dependências foi configurado para requisitar no mínimo a versão *stable* 7.1.3. Dentre os motivos para a escolha da linguagem destaca-se:

- O projeto é open-source e possui muitos coladores, a linguagem está sempre recebendo correções e novas funcionalidades.
- Linguagem desenvolvida especificamente para web com quase 25 anos de existência, robusta e com ampla documentação disponível.
- Extensão OpenSSL nativa que efetua a criptografia e descriptografia simétrica e assimétrica, PBKDF2, PKCS7, PKCS12, X509 e outras operações de criptografia.
- Popularidade, a [W3Techs \(2019\)](#) afirma que "o PHP é usado por 79,0% de todos os sites cuja linguagem de programação do servidor conhecemos".

3.2 PHP OpenSSL

Essa extensão vincula funções da biblioteca OpenSSL para criptografia e descriptografia simétrica e assimétrica, PBKDF2, PKCS7, PKCS12, X509 e outras operações de criptografia. Além disso, fornece implementação de fluxos TLS ([PHPGROUP, 2019](#), Introdução).

Utilizado amplamente pela maioria dos servidores de internet, o OpenSSL¹ é uma implementação de código aberto dos protocolos SSL e TLS. A biblioteca, escrita na linguagem C, implementa funções criptográficas básicas e fornece varias funções utilitárias.

A extensão fornece diversas funções relacionadas a certificados digitais que foram usadas nas funcionalidades do pacote protótipo desenvolvido.

¹ Disponível em <https://www.openssl.org/>. Acesso em maio de 2019

3.3 Framework Laravel

O gerenciador de dependências foi configurado para requisitar a versão 5.8 do *framework*. O Laravel foi escolhido devido suas funcionalidades e popularidade. Dentre as ferramentas e funcionalidades presentes no *framework* as seguintes foram utilizadas:

- **Artisan²**: Artisan é interface de linha de comando (CLI) incluída no Laravel. O Artisan possui comandos que ajudam na construção da aplicação e foi utilizado durante todo processo de desenvolvimento.
- **Blade³**: O blade foi utilizado como a *template engine* do projeto teste. Diferente de outras *template engines* o Blade não restringe o uso de PHP puro nas *views*.
- **Eloquent ORM⁴**: O Eloquent é um *Object Relational Mapper* incluído no laravel que fornece uma implementação do *ActiveRecord* para trabalhar com os dados. Cada tabela do banco de dados possui uma classe "Modelo" correspondente, a classe fornece métodos para interagir com o banco de dados como inserir, consultar, filtrar e outras funcionalidades.
- **Migrations⁵**: As *Migrations* são o controle de versão do banco de dados, permite um melhor controle das modificações realizadas no banco de dados.
- **Autenticação⁶**: O Laravel simplifica a implementação da autenticação nos projetos, oferecendo toda a estrutura de autenticação pré-construída. O pacote desenvolvido utiliza essa a estrutura para algumas funcionalidades, como por exemplo a autenticação pelo certificado digital.
- **Pacotes⁷**: Os pacotes são a principal forma de adicionar uma funcionalidade ao Laravel, é possível criar um pacote independente que pode ser acoplado como um módulo em qualquer aplicação Laravel. Neste projeto todas as funções relacionadas a certificação digital foram pensadas para ser um pacote Laravel independente.

3.4 Aplicação teste

Junto com o desenvolvimento do pacote também foi desenvolvido uma aplicação Laravel de teste simples, de gerenciamentos de usuários, com o objetivo de validar as funcionalidades do protótipo desenvolvido. A [Tabela 1](#) apresenta as tecnologias utilizadas na aplicação teste.

² Disponível em <<https://laravel.com/docs/5.8/artisan>>. Acesso em agosto de 2019

³ Disponível em <<https://laravel.com/docs/5.8/blade>>. Acesso em agosto de 2019

⁴ Disponível em <<https://laravel.com/docs/5.8/eloquent>>. Acesso em agosto de 2019

⁵ Disponível em <<https://laravel.com/docs/5.8/migrations>>. Acesso em agosto de 2019

⁶ Disponível em <<https://laravel.com/docs/5.8/authentication>>. Acesso em agosto de 2019

⁷ Disponível em <<https://laravel.com/docs/5.8/packages>>. Acesso em agosto de 2019

Tabela 1 – Tecnologias *web* utilizadas na aplicação teste

| Nome | Versão | Descrição |
|-----------------------|---------|---|
| php | ^7.1.3 | Linguagem script server-side. |
| laravel/framework | 5.8. | Framework PHP para desenvolvimento de aplicações web |
| laravel/tinker | ^1.0 | Console interativo Laravel que permite a interação com a aplicação pela linha de comando |
| laravel-mix | ^4.0.7 | Fornece uma API para definir as etapas de compilação do Webpack de uma aplicação Laravel usando varios pre-processadores Javascript e CSS |
| cross-env | ^5.1 | Executa scripts que configuram e usam variáveis de ambiente nas plataformas. O cross-env faz com que você possa ter um único comando sem se preocupar em definir ou usar a variável de ambiente corretamente para a plataforma. |
| lodash | ^4.17.5 | Biblioteca Javascript que prove funções comuns de programação usando o paradigma de programação funcional |
| popper.js | ^1.12 | Mecanismo de posicionamento, utilizado para posicionar um elemento próximo a outro elemento "referencia" |
| resolve-url-loader | ^2.3.1 | Um webpack loader que rescreve instruções url () baseadas no arquivo original. |
| sass | ^1.15.2 | Distribuição do Dart Sass compilado em JavaScript puro sem dependências externas. Ele fornece o comando "sass" que pode ser executado em uma API Node.js |
| sass-loader | ^7.1.0 | Carrega arquivos SASS/SCSS e os compila para CSS |
| vue | ^2.5.17 | Framework JavaScript focado no desenvolvimento de interfaces de usuário e aplicativos de página única. |
| vue-template-compiler | ^2.6.10 | Este pacote pode ser usado para pré-compilar modelos do Vue 2.0 em funções de renderização para evitar sobrecarga de compilação em tempo de execução |

Fonte: Próprio autor

3.5 Hardware

Tanto o protótipo do pacote e aplicação teste foram desenvolvidos em um computador *notebook* ASUS, modelo X555L, com processador Intel(R) Core(TM) i7-5500U CPU @ *clock* de 2.40GHz, memória RAM de 10 GB e sistema operacional Elementary OS 5.0 Juno.

Nos testes também foi utilizado um computador *desktop*, com processador Intel(R) Core(TM) i5-3450 CPU *clock* de 3.10GHz, memória RAM de 12 GB e sistema operacional

Windows 10 Pro de 64 bits.

3.6 Metodologia

O trabalho foi dividido sequencialmente em quatro etapas: pesquisa, prova de conceito, desenvolvimento do protótipo e testes. Essas etapas resultariam em duas macro entregas, sendo uma entrega na etapa de prova de conceito, onde seria validado os conceitos estabelecidos na pesquisa e a outra na etapa de testes, que seria a primeira versão do protótipo desenvolvido. A [Figura 8](#) ilustra o fluxo aplicado no desenvolvimento do trabalho.

Figura 8 – Fluxo aplicado no desenvolvimento do trabalho



Fonte: Próprio autor

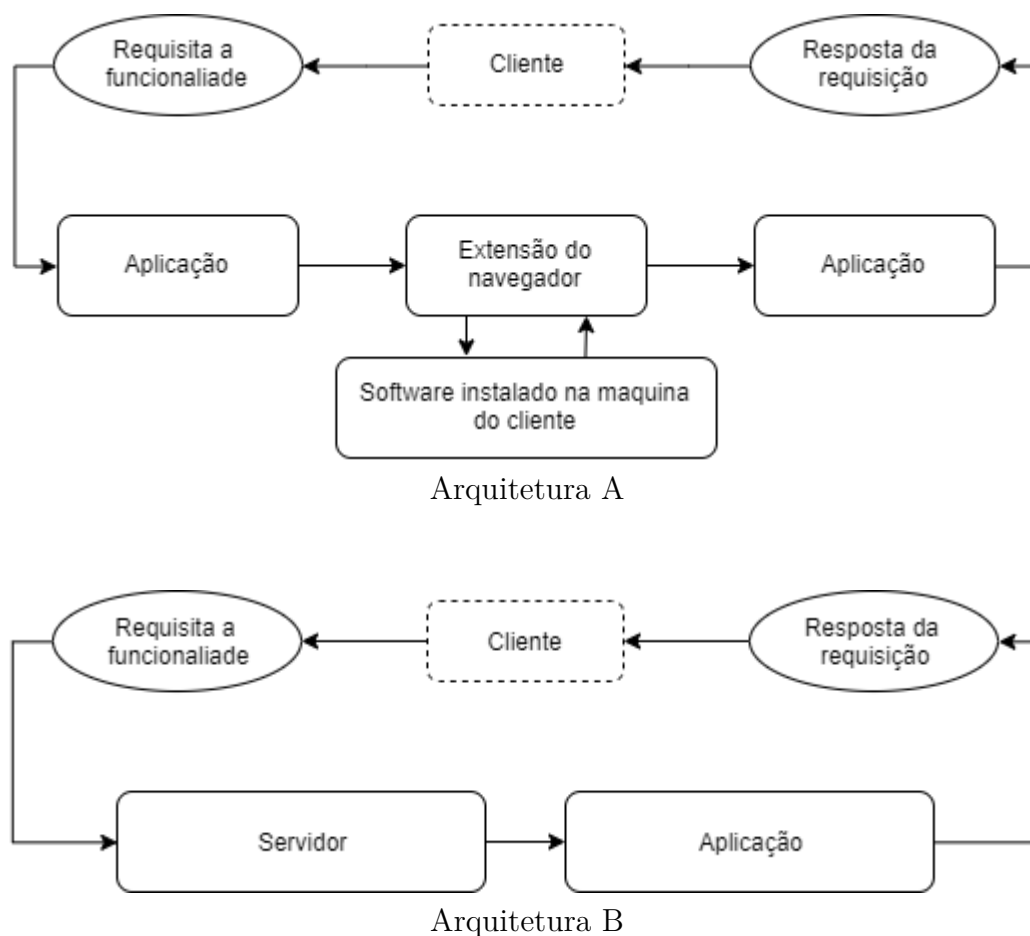
A primeira etapa realizada neste trabalho teve como objetivo levantar as ferramentas utilizadas pelas aplicações web que suportam a certificação digital presentes no mercado. Foram identificadas quatro ferramentas. Para cada uma dessas ferramentas foram analisadas as seguintes características: comunicação entre o cliente e o servidor, funcionalidades, suporte a diferentes tipos de certificados e usabilidade. Esta análise levou em conta o código e a documentação disponível de cada uma das ferramentas.

Como resultado desta pesquisa foram definidos duas arquiteturas base: Arquitetura A e Arquitetura B. As arquiteturas são representadas na [Figura 9](#). Essas arquiteturas resumem duas formas diferentes de integrar a certificação digital. Essa separação se deve principalmente ao suporte a diferente tipos de certificados e a distribuição do *software*.

Na Arquitetura A a aplicação se comunica com uma extensão no navegador do cliente, e essa extensão estabelece um canal de comunicação com um software instalado no cliente. Este software é responsável por gerenciar os certificados e realizar as funcionalidades que envolvem os certificados. Ele é necessário pois o certificado A3 é baseado em hardware, como explica a [subseção 2.3.2](#), desta forma as operações que o envolvem são executadas diretamente na máquina onde ele está inserido e somente enviadas para a aplicação.

Na Arquitetura B não existe software instalado no cliente, ela utiliza o próprio suporte da certificação digital presente nos navegadores e servidores. A falta desse software impossibilita o uso dos certificados baseados em hardware como o A3. Na autenticação o servidor é responsável por requisitar e exportar o certificado do cliente, desta forma a aplicação consegue ter acesso ao certificado e autenticar o usuário. Para realizar outras

Figura 9 – Tipos de Fluxo levantados



Fonte: Próprio autor

funcionalidades o cliente deve submeter o certificado e a senha, a aplicação pode optar por armazenar esse certificado ou solicita-lo toda vez que precisar.

Dentre essas duas alternativas de implementação a Arquitetura B foi a escolhida como a base do protótipo desenvolvido, essa escolha levou em conta o tempo disponível para a realização do trabalho e a facilidade na distribuição.

A etapa Prova de conceito teve como objetivo validar a pesquisa realizada, para isso foi desenvolvido uma aplicação simples com base nos métodos levantados.

Após o desenvolvimento da prova de conceito foi realizado uma análise sobre o que poderia ser melhorado e qual seria a forma de desenvolver as funcionalidades de certificação digital em um pacote Laravel. Após essa análise foi começado o desenvolvimento do protótipo.

A ultima etapa foi a de Testes, nesta etapa o pacote foi utilizado em uma aplicação simples onde foi testado cada uma das suas funcionalidades.

4 Projeto e Desenvolvimento

O desenvolvimento do projeto foi realizado após a conclusão da pesquisa inicial sobre as ferramentas que suportam a certificação digital em aplicações web. A primeira etapa foi validar a pesquisa, para isso foi desenvolvido uma prova de conceito. Após análises e testes realizados na prova de conceito o protótipo do pacote foi modelado e desenvolvido. Por fim foi desenvolvido uma aplicação teste que utilizava o protótipo do pacote com o objetivo de validar as funcionalidades desenvolvidas.

4.1 Desenvolvimento da prova de conceito

Para validar a pesquisa foi desenvolvido uma prova de conceito. No desenvolvimento foi utilizado somente o PHP com sua extensão do OpenSSL. Foi desenvolvido uma ferramenta simples com o objetivo de exportar o certificado do cliente e realizar algumas funcionalidades básicas com o certificado exportado. Nos testes foi utilizado o navegador Firefox, e o servidor da aplicação foi o Apache.

Como resultado a prova de conceito respondeu de forma satisfatória aos testes realizados. Foi observado algumas melhorias na comunicação e usabilidade que poderiam ser aplicadas no desenvolvimento do pacote, essas características foram consideradas na modelagem do protótipo.

4.2 Modelagem

Antes do desenvolvimento do prototipo do pacote foi realizado a modelagem da tabela responsável por armazenar os certificados digitais e o fluxo das duas principais funcionalidades: autenticação e a assinatura digital.

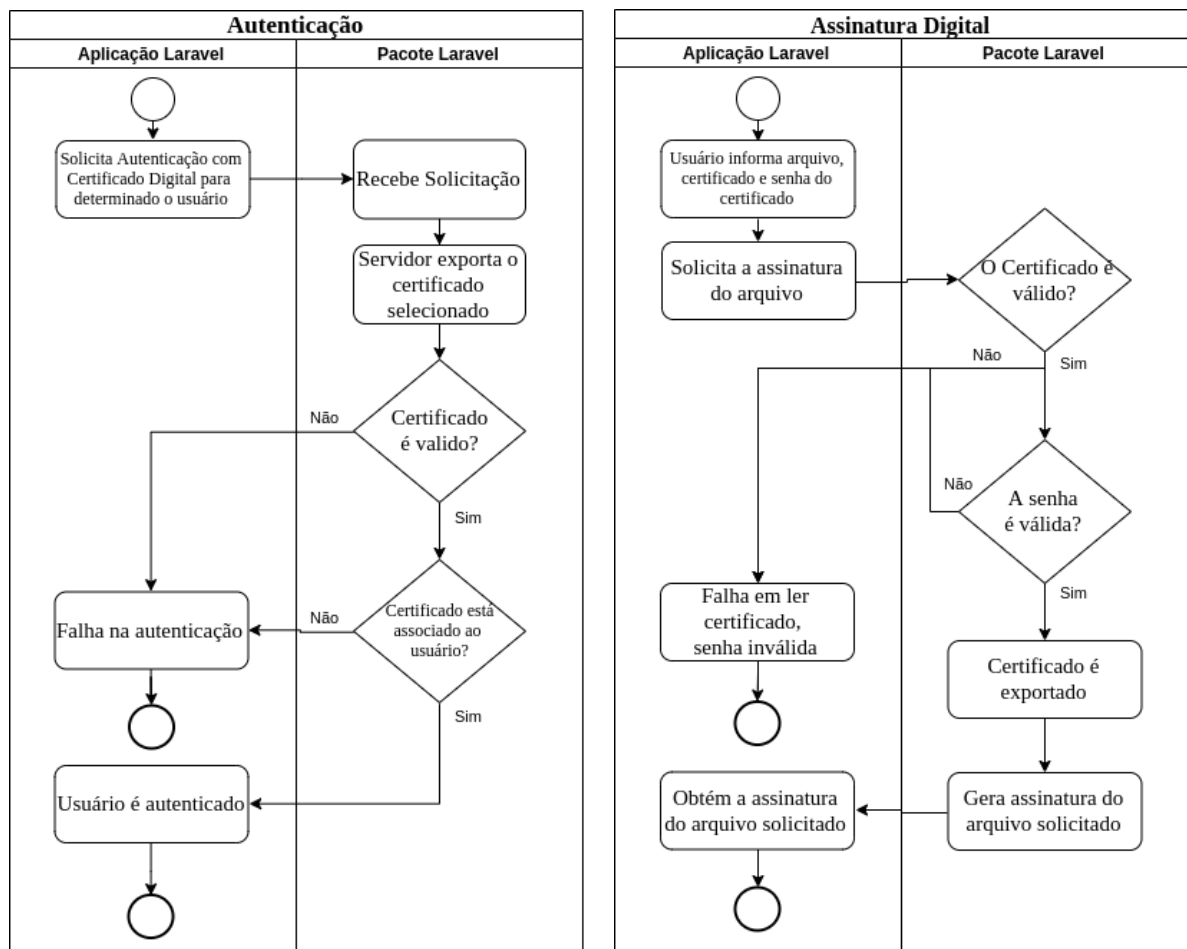
Com base na análise realizada na prova de conceito foi decidido que seria melhor dividir a aplicação e o pacote e em domínios diferentes, essa divisão é necessária pois o certificado do cliente é solicitado pelo servidor na camada de rede antes mesmo do código da aplicação. Ao separar a aplicação e o pacote em domínios diferentes é possível requisitar o certificado digital somente quando o usuário solicitar, desta forma o fluxo do usuário não é interrompido sem a necessidade.

A tabela modelada para armazenar os certificados não contem todas as propriedades acessíveis do certificado, foi armazenado somente o necessario para utilizar as funcionalidades implementadas. A tabela dos certificados contem as seguintes propriedades:

- **id**: Identificador único do certificado gerado pela aplicação.
- **user_id**: Chave estrangeira utilizada para relacionar o certificado a um usuário.
- **serial**: Número de série do certificado.
- **client_issuer_dn**: Informações sobre a distribuição e emissão do certificado.
- **name**: Nome e informações do cliente. No padrão E-CPF contem nome e CPF, já no E-CPNJ nome CPNJ.
- **email**: Email do proprietário do certificado.
- **pkey**: Chave pública do certificado.

A relação do certificado com o usuário é de "muitos para um", ou seja, é possível que um único usuário tenha mais de um certificado. Esta relação é estabelecida com o modelo de usuário padrão disponível no módulo de autenticação Laravel.

Figura 10 – Modelagem do fluxo das funcionalidades de autenticação e assinatura digital



(a) Fluxo Autenticação

(b) Fluxo Assinatura Digital

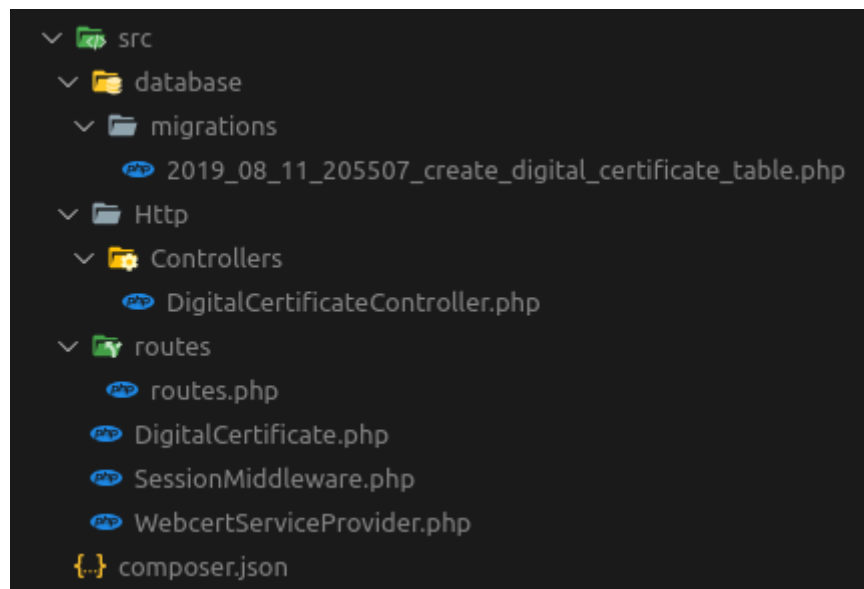
No fluxo definido para a autenticação, veja o fluxo na [Figura 10a](#), quando o usuário requisitar a autenticação ele será redirecionado para o sub domínio do pacote em uma rota específica relacionada a autenticação, onde o certificado selecionado pelo cliente será exportado pelo servidor. Será verificado se o certificado é válido e se ele está relacionado a algum usuário. Se uma dessas verificações falhar é retornado um erro para a aplicação, caso contrário o usuário que possui relação com aquele certificado é autenticado.

No fluxo definido para a assinatura digital, veja o fluxo na [Figura 10b](#), o usuário deve informar o certificado e sua senha junto com o arquivo que deseja assinar. Após informar esses dados o usuário é redirecionado para o subdomínio do pacote onde é checado se o certificado e a senha são válidos. Se uma dessas verificações falhar é retornado um erro para aplicação, caso contrário é gerado a assinatura do arquivo informado pelo usuário.

4.3 Estrutura e organização

Nesta seção será explicado a estrutura e organização do protótipo do pacote desenvolvido, a [Figura 11](#) mostra a organização dos arquivos do projeto. O projeto foi dividido nos seguintes arquivos e diretórios:

Figura 11 – Organização dos arquivos do projeto



Fonte: Próprio autor

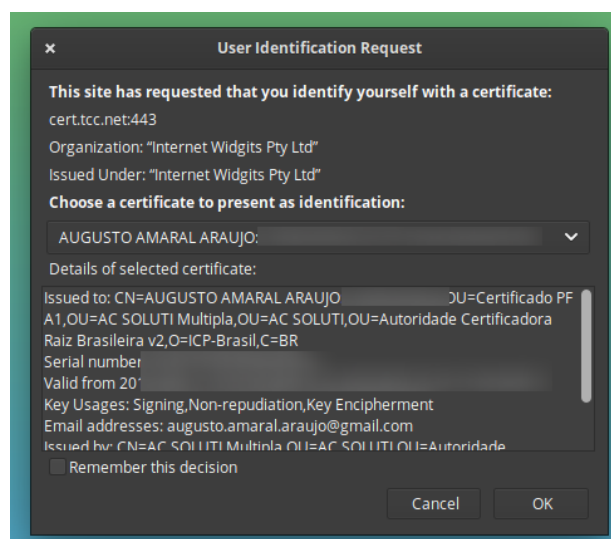
- **"composer.json"**: Arquivo que contém informações do pacote, como dependências e meta dados.
- **"src/"**: Pasta raiz do projeto onde todos os arquivos do projeto, excluindo apenas o "composer.json", são armazenados.

- **"src/database/migrations/"**: Pasta onde se encontram as *migrations* do projeto.
- **"src/Http/Controllers/"**: Pasta onde se encontram os *controllers* do projeto.
- **"routes/"**: Pasta onde se encontra as rotas do pacote, essas rotas são mapeadas para metodos do *controller*.
- **"DigitalCertificate.php"**: Classe **DigitalCertificate** que herda da classe **Model** do Laravel. Nessa classe estão presentes métodos de validação do certificado e outras funcionalidades.
- **"SessionMiddleware.php"**: Middleware das rotas do pacote.
- **"WebCertServiceProvider.php"**: Provedor de serviço do pacote, este arquivo contém o método para a inicialização do pacote.

4.4 Obter e validar o certificado do cliente

Para obter o certificado para as funcionalidades que não necessitam da chave privada do certificado, é utilizado o próprio suporte dos navegadores e servidores. O certificado do cliente é requisitado pelo servidor abrindo uma janela que solicita que o usuário selecione um certificado dentre os certificados cadastrados no navegador, veja um exemplo desta janela na [Figura 12](#) . Após selecionar o certificado ele é exportado do servidor para o PHP através da variável super global **\$_SERVER**. Todas as variaveis relacionadas ao certificado do cliente exportado do servidor Apache estão presentes no Anexo [A](#).

Figura 12 – Exemplo de janela de seleção do certificado no Firefox (informações ocultadas)



Fonte: Próprio autor

Antes de se utilizar o certificado exportado é preciso checar se ele é válido. Foi desenvolvido um método que checa: se a validação do servidor foi bem sucedida, se as variáveis obrigatórias estão preenchidas e se o certificado não está vencido. O [Código 4.1](#) é o script desenvolvido responsável por esta verificação.

Código 4.1 – Script utilizado para checar se um certificado é válido

```
if (!isset($_SERVER['SSL_CLIENT_M_SERIAL']))
    || !isset($_SERVER['SSL_CLIENT_V_END'])
    || !isset($_SERVER['SSL_CLIENT_VERIFY'])
    || $_SERVER['SSL_CLIENT_VERIFY'] !== 'SUCCESS'
    || !isset($_SERVER['SSL_CLIENT_I_DN'])
) {
    return false;
}

if ($_SERVER['SSL_CLIENT_V_REMAIN'] <= 0) {
    return false;
}

return true;
```

Se o certificado for válido ele é convertido em um vetor que contém: O *serial* do certificado, informações sobre a distribuição e emissão do certificado, nome, email e informações do proprietário. De posse desse vetor que representa o certificado é possível utilizar funcionalidades como cadastro e autenticação.

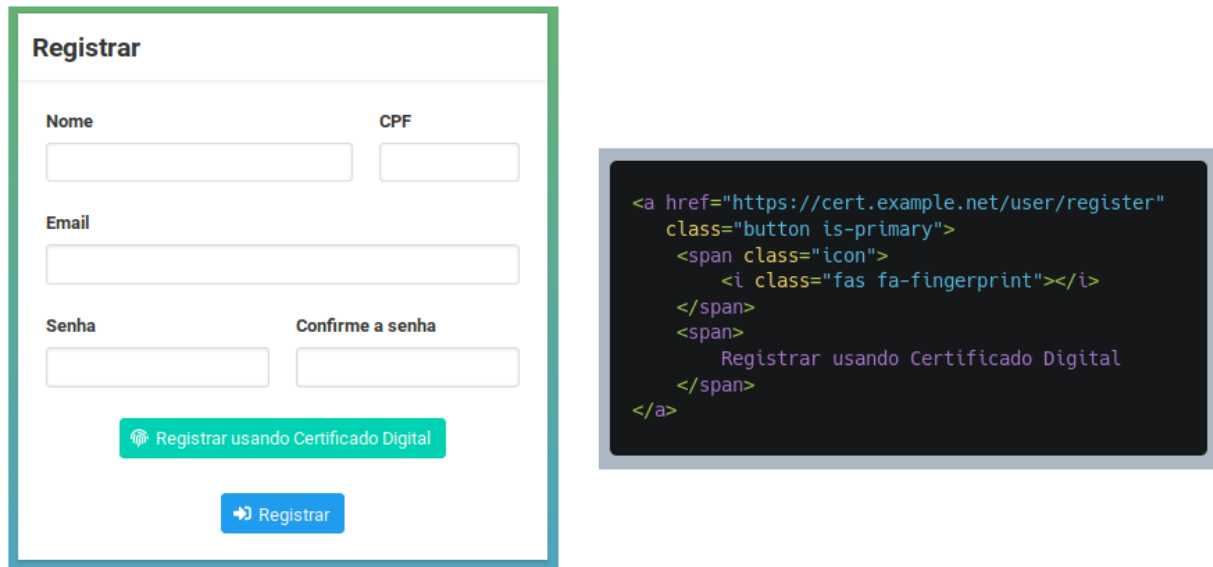
Se a funcionalidade necessitar da chave privada do certificado é utilizado outro método para obter o certificado, pois o método utilizado nas outras funcionalidades não possibilita o acesso a chave privada. Nesse caso foi utilizado as funções da extensão do OpenSSL do PHP, que recebem como parâmetro o arquivo eletrônico do certificado digital e sua senha e retornam o certificado do cliente junto com sua chave privada.

4.5 Funcionamento

Para as funcionalidades que exigem o certificado exportado do cliente, o usuário do pacote deve criar um link a uma rota específica que sera mapeada para o *controller* do pacote. Por exemplo, na [Figura 13](#) foi criado um botão que redireciona para rota de cadastro utilizando o certificado digital, quando o usuário clicar no botão ele será redirecionado e o seu cadastro sera efetuado utilizando o certificado exportado pelo servidor.

O certificado exportado pelo servidor não é suficiente para as funcionalidades que

Figura 13 – Exemplo uso da funcionalidade Registrar com o Certificado do pacote



```
<a href="https://cert.example.net/user/register"
class="button is-primary">
  <span class="icon">
    <i class="fas fa-fingerprint"></i>
  </span>
  <span>
    Registrar usando Certificado Digital
  </span>
</a>
```

Fonte: Próprio autor

necessitam de acesso a chave privada, como por exemplo a assinatura digital. Nessas funcionalidades o cliente deve preencher um formulário com o arquivo do certificado junto com a sua senha, e esse formulário deve ser submetido através de um método POST a rota do pacote responsável pela funcionalidade.

4.6 Autenticação

No desenvolvimento da funcionalidade de autenticação foi criado uma rota no pacote que ao ser acessada solicita o certificado do usuário, como explica a [seção 4.4](#).

O certificado exportado é utilizado como parâmetro de um metodo que checa se o certificado ja foi cadastrado no sistema. A busca utiliza o email e nome do proprietario do certificado e suas informações de emissão. O *serial* não é utilizado para identificar o certificado pois se o certificado for renovado o *serial* é trocado, sendo assim mesmo que o certificado seja um certificado renovado seu proprietario é identificado corretamente pela busca. Veja o [Código 4.2](#).

Código 4.2 – Busca utilizada para encontrar um certificado cadastrado através do certificado exportado

```
$cert = DigitalCertificate::where('serial', $certificate['serial'])->
  where('client_issuer_dn', $certificate['issuer_dn'])->
  where('name', $certificate['name'])->
  where('email', $certificate['email'])->
```



```
first();
```

Se o certificado exportado já estiver cadastrado no sistema é efetuado o login do usuário que está relacionado com este certificado através da função de autenticação do próprio *framework* Laravel, que recebe o identificador único do usuário como parâmetro.

4.7 Assinatura Digital

No desenvolvimento da assinatura digital foi criado uma rota que mapeia para um método no *controller* que recebe o arquivo do certificado digital e sua senha junto com o arquivo que deve ser assinado.

Para ter acesso a chave privada do arquivo do certificado digital foi utilizado as funções da extensão PHP OpenSSL que extraem as chaves do arquivo, como a `openssl_pkcs12_read`. Após obter a chave privada é utilizado a função `openssl_sign` que recebe como parâmetro o arquivo e a chave privada e retorna a assinatura do arquivo. Veja esse processo no [Código 4.3](#)

Código 4.3 – Script que assina um arquivo e retorna a assinatura

```
openssl_pkcs12_read($cert, $certs, $password);

$fileData = file_get_contents($file);
openssl_sign($fileData, $signature, $certs["pkey"]);
return $signature;
```

4.8 Funcionalidades

A [Tabela 2](#) lista todas as funcionalidades desenvolvidas no protótipo, a primeira coluna corresponde ao nome do método, a segunda a os parâmetros do método e a terceira a uma descrição resumida da funcionalidade. Mais detalhes e exemplos de uso de cada funcionalidade estão presentes na documentação da ferramenta que se encontra no [Apêndice A](#)

Tabela 2 – Funcionalidades do protótipo do pacote desenvolvido

| Nome | Parâmetros | Descrição |
|----------------------|--------------------------------|---|
| dump | Não possui | Retorna o certificado exportado em formato JSON |
| userRegister | Não possui | Registra um novo usuario no sistema utilizando as informações obtidas no certificado exportado, se o certificado for invalido é retornado um erro |
| userLogin | Não possui | Efetua o login do usuario relacionado ao certificado exportado, se o certificado for invalido ou não houver nenhum usuario relacionado é retornado um erro. |
| clientIsValidCert | Não possui | Verifica se o certificado exportado é valido, retornando verdadeiro ou falso |
| serverToArray | Não possui | Retorna um array contendo os dados do certificado exportado |
| getClientCertificate | Não possui | Retorna o certificado exportado |
| alreadyRegistered | Não possui | Checa se o certificado exportado ja esta associado a algum usuario, retornando o resultado da checagem |
| getUserCertificates | \$id | Retorna os certificados de determinado usuário |
| signFile | \$certFile, \$file, \$password | Recebe um certificado e sua senha e assina o arquivo informado, é retornado a assinatura gerada. Se ocorrer algum erro na operação esse erro é retornado |
| checkFileSign | \$file, \$sign, \$pkey | Recebe um arquivo e sua assinatura e verifica se determinada chave publica é o emissor da assinatura, é retornado verdadeiro ou falso. |

Fonte: Próprio autor

5 Resultados

Como resultado deste trabalho, obteve-se um protótipo de um pacote para integração da certificação digital em aplicações *web* Laravel. Este pacote disponibiliza funcionalidades para a autenticação e assinatura digital utilizando o certificado do tipo A1.

// TODO DETALHAR SOBRE OS TESTES (QUAL CERTIFICADO FOI USADO), PRINTS DA APLICAÇÃO TESTE, RESULTADOS DOS TESTES -

Referências

ARANHA, D. *Serviço de nomes e roteamento para redes de anonimização de tráfego*. 68 p. Tese (Doutorado), 04 2007. Citado na página 21.

CERT.BR. *Estatísticas dos Incidentes Reportados ao CERT.br*. 2017. Disponível em: <<https://www.cert.br/stats/incidentes/>>. Citado na página 15.

COOPER, D. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. 2008. IETF Tools. Disponível em: <<https://tools.ietf.org/html/rfc5280>>. Acesso em: 20 jan 2019. Citado 2 vezes nas páginas 22 e 25.

DIERKS, T. The tls protocol version 1.0. 1999. Disponível em: <<https://www.ietf.org/rfc/rfc2246.txt>>. Acesso em: 21 jan 2019. Citado na página 24.

DIERKS, T.; RESCORLA, E. The transport layer security (tls) protocol rfc 5246. 2008. Citado 2 vezes nas páginas 20 e 22.

DIFFIE, W.; HELLMAN, M. E. New directions in cryptography. 1976. Citado na página 19.

FEBRABAN. *Febraban - Pesquisa 26ª edição*. 2018. Portal Febraban. Disponível em: <<https://portal.febraban.org.br/noticia/3184/pt-br/>>. Acesso em: 20 fev 2019. Citado na página 15.

HOUSLEY, R. Internet x.509 public key infrastructure certificate and crl profile. 1999. Disponível em: <<https://www.ietf.org/rfc/rfc2459.txt>>. Acesso em: 25 jan 2019. Citado na página 24.

IBM. *IBM - Infraestrutura da Chave Pública (PKI)*. 2019. IBM IBM Knowledge Center. Disponível em: <https://www.ibm.com/support/knowledgecenter/pt-br/SSFKSJ_8.0.0/com.ibm.mq.sec.doc/q009900_.htm>. Acesso em: 19 mai 2019. Citado na página 20.

ICP-BRASIL. *ICP-Brasil*. 2019. Instituto Nacional de Tecnologia da Informação. Disponível em: <<https://www.it.gov.br/icp-brasil>>. Acesso em: 10 jan 2019. Citado na página 20.

ITI. *ICP-Brasil bate recorde de emissão de certificados digitais em 2018*. 2019. Instituto Nacional de Tecnologia da informação. Disponível em: <<https://www.it.gov.br/legislacao/17-noticias/indice-de-noticias/2479-icp-brasil-bate-recorde-de-emissao-de-certificados-digitais-em-2018>>. Acesso em: 22 fev 2019. Citado na página 16.

KENTR, S. *Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management*. 1998. IETF Tools. Disponível em: <<https://tools.ietf.org/html/rfc1422>>. Acesso em: 24 jan 2019. Citado na página 24.

KHARITONOV. *EToken_PRO_USB*. 2009. Wikimedia. Disponível em: <https://commons.wikimedia.org/wiki/File:EToken_PRO_USB.jpg>. Acesso em: 12 fev 2019. Citado na página 23.

- PHPGROUP. *OpenSSL*. 2019. Php manual. Disponível em: <https://www.php.net/manual/pt_BR/book.openssl.php>. Acesso em: 20 mai 2019. Citado na página 29.
- SAMUELLE, T. J. Mike meyers' comptia security+ certification passport. In: _____. [S.l.]: McGraw-Hill Education, 2011. p. 137. Citado na página 21.
- SINGH, S. The code book: The evolution of secrecy from mary, queen of scots to quantum cryptography. 1999. Citado na página 18.
- W3TECHS. *Usage statistics of PHP for websites*. 2019. Web technology Surveys. Disponível em: <[veysvw3techs.com/technologies/details/pl-php/all/all](https://w3techs.com/technologies/details/pl-php/all/all)>. Acesso em: 16 out 2019. Citado 3 vezes nas páginas 27, 28 e 29.
- WILLEMAN, D. P.; IBARRA, G. B. Framework java de apoio ao desenvolvimento de aplicações web com banco de dados, utilizando struts, tiles e hibernate. 2005. Citado na página 27.
- YANG, H. Pki tutorials - herong's tutorial examples. In: _____. [S.l.: s.n.], 2010. cap. Introduction of PKI (Public Key Infrastructure). Citado na página 21.

Apêndices

APÊNDICE A – Documentação básica do pacote desenvolvido

A.1 Introdução

O pacote LARACERT foi desenvolvido visando facilitar a integração da certificação digital em aplicações Laravel. No estado atual de desenvolvimento o pacote realiza a autenticação e gera a assinatura digital de documentos eletrônicos.

Esta documentação detalha o uso do pacote e possui instruções de instalação e configuração, recomendações, funcionalidades e exemplos de uso do pacote.

A.2 Requisitos

O pacote foi desenvolvido na versão 5.8 do Laravel. O servidor utilizado nos testes foi o Apache 2, portanto a documentação irá detalhar o processo de configuração para esse servidor em específico.

O servidor deve estar configurado com o HTTPS, ou seja você precisa de um certificado SSL para o servidor. Isso deve ser implementado antes de requisitar os certificados dos clientes.

A negociação do certificado ocorre na camada de rede, antes mesmo do código da aplicação. Por isso é necessário que o servidor possua a funcionalidade de requisitar o certificado do cliente e exportá-lo para a aplicação.

O pacote deve ser configurado em um subdomínio da aplicação principal, o qual será responsável por requisitar o certificado quando solicitado. A divisão da aplicação e do pacote é necessária para que a experiência do usuário não seja prejudicada, requisitando o certificado somente quando necessário.

Na aplicação deve ser possível que os usuários gerenciem os certificados associados a suas contas, com opções para remover e adicionar certificados. Um usuário pode possuir mais de um certificado, como por exemplo, um pessoal e outro do trabalho. Se o certificado for comprometido o usuário deve possuir uma maneira de desassociá-lo a sua conta.

A.3 Instalação

TODO

A.4 Configuração

A.4.1 Apache 2 - Habilitando HTTPS

Como dito nos requisitos, é necessário configurar o HTTPS. O código A.1 exemplifica uma configuração de *virtual host* com SSL habilitado.

Código A.1 – Exemplo de configuração de virtual host com SSL

```
// Virtual Host SSL ON

<VirtualHost *:443>
    ServerName myapp.com

    ErrorLog "logs/app_log"
    CustomLog "logs/app_clog" common

    LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon

    DocumentRoot /home/aaraujo/dev/html/app/public

    <Directory "/home/aaraujo/dev/html/app">
        AllowOverride all
        Header set Access-Control-Allow-Origin "*"
    </Directory>

    SSLEngine On

    SSLCertificateFile etc/ssl/app_cert.cert
    SSLCertificateKeyFile etc/ssl/app_key.key

</VirtualHost>
```

Também é necessário habilitar o módulo SSL do servidor web. É possível habilitá-lo utilizando o comando:

```
$ a2enmod ssl
```

O **a2enmod** é um script que habilita um módulo específico da configuração do apache 2. O comando cria um *symlinks* com o `/etc/apache2/mods-enabled` e também é possível utilizar o **a2dismod** para desabilitar um módulo.

Outra alternativa é modificar o arquivo de configuração `httpd.conf` do apache, acrescentando a linha:

```
LoadModule ssl_module modules/mod_ssl.so
```

Após essas alterações deve ser possível requisitar as páginas por HTTPS, lembre-se de reiniciar o servidor após alterar as configurações.

A.4.2 Apache 2 - Requisitar certificados de clientes

Um servidor web não requisita o certificado de um cliente por padrão, essa funcionalidade precisa ser habilitada. A seguir um exemplo de como deve ficar a configuração do seu *virtual host* para que essa funcionalidade seja habilitada:

Código A.2 – Exemplo de configuração para requisitar o certificado do cliente

```
...
</Directory>

SSLCACertificateFile /etc/ssl/cadeia_certificados.crt

SSLVerifyClient optional
SSLVerifyDepth 3
SSLOptions +StdEnvVars +ExportCertData
</VirtualHost>
```

A diretiva `SSLVerifyClient` determina o nível de verificação utilizado para a autenticação do usuário. A seguir os níveis disponíveis de verificação:

- `none` : Nenhum certificado é necessário.
- `optional` : O cliente pode optar em escolher ou não um certificado. Essa é a melhor opção se você quer uma aplicação que permita o login de diferentes formas, utilizando ou não o certificado.
- `require` : O certificado é obrigatório, portanto o cliente deve possuir um certificado válido.
- `optional_no_ca` : O cliente pode optar escolher ou não um certificado e ele não necessariamente é verificável. Essa opção não deve ser utilizada em produção pois é contra a própria ideia da autenticação, mas pode ser utilizada em testes.

A diretiva `SSLVerifyDepth` define o nível de profundidade que deve ser verificado antes de decidir se um cliente possui ou não um certificado válido. O certificado do cliente pode ser assinado por uma AC e seu servidor web confiará nas ACs especificadas no `SSLCACertificateFile`. A AC em si pode ser assinada por outra autoridade, a Figura 14 demonstra essa hierarquia.

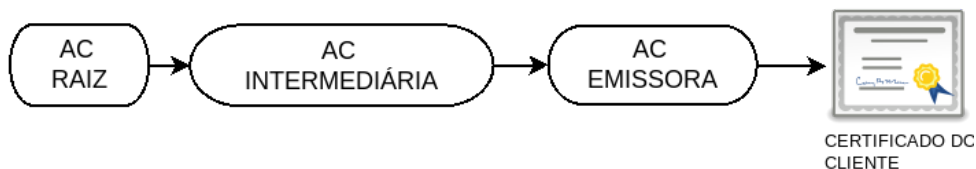


Figura 14 – Exemplo de hierarquia de Autoridades Certificadoras

A diretiva `SSLOptions` é utilizada para controlar diversas opções em tempo de execução, é possível utilizar mais de uma configuração utilizando o operador `+` para habilitá-las ou o `-` para desabilitá-las. Quando a opção `+StdEnvVars` está habilitada o servidor cria as variáveis de ambiente SSL, possibilitando que sua aplicação verifique e leia dados do certificado do cliente. Para exportar completamente o certificado do cliente você pode acrescentar a opção `+ExportCertData`.

A diretiva `SSLCACertificateFile` define o arquivo que contém as Autoridades Certificadoras que o servidor web confia. No Brasil geralmente você vai acrescentar no arquivo a AC da ICP Brasil e a da Raiz Brasileira juntamente com o certificado do próprio site. Exemplo do padrão de arquivo utilizado nas aplicações:

Código A.3 – `SSLCACertificateFile` padrão utilizado nos testes

```

CA do HTTPS do site
=====
(...)

= CA ICP Brasil =
(NF-e/e-CPF/e-CNPJ v5)Autoridade Certificadora Raiz Brasileira v5
=====
-----BEGIN CERTIFICATE-----
MIIGoTCCBImgAwIBAgIBATANBgkqhkiG9w0BAQ0FADCBzlELMAkGA1UEBhMCQlIx
EzARBgNVBAoMCKlDUC1CcmFzaWwxPTA7BgNVBAsMNELuc3RpdHV0byBOYWNPb25h
bCBkZSB1ZWNub2xvZ2lhIGRhIEluZm9ybWVjYW8gLSBjVEkxNDAYBgNVBAMMK0F1
dG9yaWRhZGUgQ2VydGhmaWNhZG9yYSBSYWl6IEJyYXNpbGVpcmEgdjUwHhcNMTYw
MzAyMTMwMTM4WWhcNMjkwMzAyMjM0OTM4WjCBzlELMAkGA1UEBhMCQlIx
EzARBgNVBAoMCKlDUC1CcmFzaWwxPTA7BgNVBAsMNELuc3RpdHV0byBOYWNPb25h
bCBkZSB1ZWNub2xvZ2lhIGRhIEluZm9ybWVjYW8gLSBjVEkxNDAYBgNVBAMMK0F1
dG9yaWRhZGUgQ2VydGhmaWNhZG9yYSBSYWl6IEJyYXNpbGVpcmEgdjUwggLiMA0G
CSqGSIb3DQEBAQUAA4ICDwAwggIKAoICAQD3LXgabUWsF+gUXw/6YODeF2XkqE
yfk3VehdsIx+3/ERgdjCS/ouxYR0Epi2hdoMUVJDNf3XQfjAWXJyCoTneHYA12Mc
MdvoqtLB2ileQlJiis0fttYTJayee9BAIdIrCor1Lc0vozXCpDtq5nTwhjLocaZtcu
Fsdrkl+n
  
```

```

bfYx15m7vjTkTMS6j8ffjmFzbNPDlJuV3Vy7AzapPVJrMl6UHPXCHMYMz10KxR/4
7S5XGgmLYkYt8bNCHA3fg07y+Gtvgu+SNhMPwWKIghYw+9vErOnavRhOimYo4M2
AwNpNK0OKLI7Im5V094jFp4Ty+mlmfQH00k8nkSUEN+1TGGkhv16c2hukbx9iCfb
mk7im2hGKjQA8eH64VPYoS2qdKbPbd3xDDHN2croYKpy2U2oQTVBSf9hC3o6fKo3
zp0U3dNiW7ZgWKS9UwP31Q0gwgB1orZgLuF+LIppHYwxcTG/AovNwa4sTPukMiX2
L+p7uIHExTZJJU4YoDacQh/mfbPIz3261He4YFmQ35sfw3eKHQSOLyiVfev/n0l/
r308PijEd+d+Hz5RmqIzS8jYXZleJxym4mEjE1fKpeP56Ea52LlIJ8ZqsJ3xzHWu
3WkAVz4hMqrX6BPMGW2LxOuEUQyIaCBg1lI6QLiPMHvo2/J7gu4YfqRcH6i27W3H
yzamEQIDAQABo4H1MIHyME4GA1UdIARHMEUwQwYFYEWBAQAwwOjA4BggrBgEFBQcC
ARYsaHR0cDovL2FjcmFpei5pY3BicmFzaWwuZ292LmJyL0RQQ2FjcmFpei5wZGZYw
PwYDVR0fBDgwNjA0oDKgMIYuaHR0cDovL2FjcmFpei5pY3BicmFzaWwuZ292LmJy
L0xuDUMfjcmFpenY1LmNybDAfBgNVHSMEGDAWgBRpQL512cTvOcTRERhbUvo+LZA
XjAdBgNVHQ4EFgQUaai+ddnE72znE0XkYW7laPi2QF4wDwYDVR0TAQH/BAUwAwEB
/zAOBgNVHQ8BAf8EBAMCAQYwDQYJKoZIhvcNAQENBQADggIBABRt2/JiWapef7o/
plhR4PxymlMIp/Jez5F0BZ1XafmYpl5g6pRokFrIRMFXYLhgo51I05InyCc9Td
6UXjlsOASTc/LRavyjB/8NcQjIRYDh6xf7OdP05mFcT/0+6bYRtNgsnUbr10pfsK
/UzyUvQWbumGS57hCZrAZOyd9MzukiF/azAa6JfoZk2nDkEudKOY8tRyTpMmDzN5
fufPSC3v7tSJUqTqo5z7roN/FmckRzGAYyz5XulbOc5/UsAT/tk+KP/clbbqd/hh
evmmdJclLr9qWZZcOgzUFU2YsgProtVu0ffNXGr6KK9fu44pOHajmMsTXK3X7r/P
wh19kFRow5F3RQMUZC6Re0YLfXh+ypnUSCzA+uL4JPtHIGyvkBWiulkustpOKUSV
wBPzvA2sQUOvqdbAR7C8jcHYFJMUK2HZFji7pxcWWab/NKsFcJ3sluDjmhizpQax
bYTfAVXu3q8yd0su/BHHbBpteyHvYyyz0Eb9LUysR2cMtWvfpU6vnoPgYvOGO1Cz
iyGEsgKULkCH4o2Vgl1gQuKWO4V68rFW8a/jvq28sbY+y/Ao0I5ohpnBcQOAawiF
bz6yJtObajYMuZtDDP8oY656EuuJXBjhuKAJPI/7WDtgfV8ffOh/iQGQATVMtgDN
0gv8bn5NdUX8UMNX1sHhU3H1UpoW
-----END CERTIFICATE-----

```

(e-CPF/e-CNPJ v2) Raiz Brasileira V2

-----BEGIN CERTIFICATE-----

```

MIIGoTCCBImgAwIBAgIBATANBgkqhkiG9w0BAQ0FADCB1zELMAkGA1UEBhMCQlIx
EzARBgNVBAoTCKlDUC1CcmFzaWwxPTA7BgNVBAsTNEluc3RpdHV0byBOYWNPb25h
bCBkZSB1ZWNub2xvZ2lhIGRhIEluZm9ybWJjYXZlLSB1Zm9ybWJjYXZlLSB1Zm9ybWJjYXZl
dG9yaWRhZGUGQ2VydGlmYW5hZG9yYXZlLSB1Zm9ybWJjYXZlLSB1Zm9ybWJjYXZl
NjIxMTkwNDU3WheNMjMwNjIxMTkwNDU3WjCB1zELMAkGA1UEBhMCQlIxEzARBgNV
BAoTCKlDUC1CcmFzaWwxPTA7BgNVBAsTNEluc3RpdHV0byBOYWNPb25hbCBkZSB1
ZW5hZG9yYXZlLSB1Zm9ybWJjYXZlLSB1Zm9ybWJjYXZlLSB1Zm9ybWJjYXZl
ZGUGQ2VydGlmYW5hZG9yYXZlLSB1Zm9ybWJjYXZlLSB1Zm9ybWJjYXZlLSB1Zm9ybWJjYXZl
DQEBAAQUAA4ICDwAwggIKAoICAQC6RqQO3edA8rWgffKVV0X8bYTzHgHJhQOtmKvS
8l4Fmcm7b2Jn/XdEuQMHPNlBAGLUcCxCg3lmq5lWroG8akm983QPYrfrWwdmlElk
nUasmkIYMPAkqFFB6quV8agrAnhptSknXpwuc8b+I6Xjps79bBtrAFTrAK1POkw8
5wqIW9pemgtW5LVUOB3yCpNkTsNBklMgKs/8dG7U2zM4YuT+jkxYHPePKk3/xZLZ
CVK9z3AAAnWmaM2qIh0UhmRZRDtTfgr20aah8fNTd0/IVXEfFWBDqhRnLNIJYKnIM
mpbeys8IUWG/tAUpBiuGkP7pTcMEBUfLz3bZf3Gmh3sVQOQzgHgHhATyjtAO8ly
UN9pvvAslh+QtdWudONltIwa6Wob+3JcxYJU6uBTB8TMEun33tcv1EgvRz8mYQSx
Epoza7WGSxMr0IadR+1p+/yEEmb4VuUOimx2xGsaesKgWhLRI4IYAXwIWN0VjhXZ
fn03tqRF9QOFzEf6i3lFuGZiM9MmSt4c6dR/5m0muTx9zQ8oCikPm91jq7mmRxxqE

```

```

14WkA2UGBEtSjYM0Qn8xjhEu5rNnlUB+l3pAAPkRbIM4WK0DM1umxMHFsKwNqQbw
pmkBNLbp+JRITz6mdQnsSsU74MlesDL/n2lZzzwwbw3OJ1fsWhto/+xPb3gyPnnF
tF2VfwIDAQABo4H1MIHyME4GA1UdIARHMEUwQwYFYEWBAQAwOjA4BggrBgEFBQcC
ARYsaHR0cDovL2FjcmFpei5pY3BicmFzaWwuZ292LmJyL0RQQ2FjcmFpei5wZGYw
PwYDVR0fBDgwNjA0oDKgMIYuaHR0cDovL2FjcmFpei5pY3BicmFzaWwuZ292LmJy
L0xDUmFjcmFpenYyLmNybDAfBgNVHSMEGDAWgBQMOSA6twEfy9cofUGgx/pKrTik
vjAdBgNVHQ4EFgQUDDkgOrcBH8vXKH1BoMf6Sq0yJL4wDwYDVR0TAQH/BAUwAwEB
/zAOBgNVHQ8BAf8EBAMCAQYwDQYJKoZIhvcNAQENBQADggIBAfmAFGkYbX0pQ3B9
dpth33eOGnbkqdbLdqQWDEyUEsaQ0YEDxa0G2S1EvLLJdgmAOWcAGDRtBgrmtRBZ
SLp1YPw/jh0YVXArnkuVrImrCncke2HEx5EmjkYTUTe2jCcK0w3wmisig4OzvYM1
rZs8vHiDKTVhNvgRcTMgVGNTQRHYE1qEO9dmEyS3xEbFithzJO4cExeWyCXoGx7P
34VQbTzq91CeG5fep2vb1nPSz3xQwLCM5VMSeoY5rDVbZ8fq1PvRwl3qDpdzmK4p
v+Q68wQ2UCzt3h7bhegdhAnu86aDM1tvR3lPSLX8uCYTq6qz9GER+0Vn8x0+bv4q
SyZEGp+xouA82uDkBTp4rPuooU2/XSx3KZDNEx3vBijYtxTzW8jJnqd+MRKKeGLE
0QW8BgJjBCsNid3kXFsygETUQuwq8/JAhzHVPuIKMgwUjdVybQvm/Y3kqPMFjXUX
d5sKufqQkplliDjNqWwOLQsVuzXxYejZZ3ftFuXoAS1rND+Og7P36g9KHj41hJ2M
gDQ/qZXow63EzZ7KFBySGZ7kNou5uaNCJQc+w+XVaE+gZhYms7ZzHJAaP0C5GIZC
cIf/by0PEf0e//eFMBUO4xcx7ieVzMnmpR6Xx21bB7UFaj3yRd+6gnkkeC6bgh9m
qaVtJ8z2KqLRX4Vv4EadqtKlTIUO
-----END CERTIFICATE-----

```

Após configurar todas essas diretivas e opções você deve reiniciar o servidor web. Depois de reiniciá-lo o navegador deve requisitar os certificados dos clientes ao acessar a aplicação.

É possível que o navegador não exiba a janela com os certificados. Isso ocorre geralmente quando o navegador não possui um certificado emitido pela lista de ACs que o servidor envia. Para resolver esse problema você deve configurar corretamente as diretivas `SSLCADNRequestFile` e `SSLCADNRequestPath`, que dizem respeito as ACs aceitas pelo servidor.

A.4.3 Apache 2 - Configurar subdomínio

Como dito nos requisitos na seção A.2, a negociação do certificado ocorre na camada de rede antes mesmo do código da aplicação. Uma solução para não interromper o fluxo do usuário, ou requisitar o certificado de um usuário que não necessariamente quer utilizá-lo, é criar um subdomínio e mapear as rotas do pacote para esse subdomínio. A aplicação então deve possuir dois arquivos de rotas, uma da aplicação e outra do pacote.

Os códigos A.4 e A.5 exemplificam essa divisão. O *virtual host* da aplicação é configurado normalmente, já o utilizado pelo pacote é configurado para requisitar o certificado do cliente.

```
<VirtualHost *:80>
  DocumentRoot "/app/public"
  ServerAdmin example.net
  <Directory "/app/">
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
    Header set Access-Control-Allow-Origin "*"
  </Directory>
</VirtualHost>
```

Código A.5 – Exemplo virtual host utilizado pelo pacote

```
<VirtualHost *:443>
  ServerAdmin cert.example.net
  DocumentRoot "/app/public"
  Header set Access-Control-Allow-Origin "*"
  <Directory "/app/">
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
    Header set Access-Control-Allow-Origin "*"
  </Directory>
  ServerName cert.example.net
  ErrorLog "logs/app.net_log"
  CustomLog "logs/app.net_clog" common
  SSLEngine on
  SSLOptions +StdEnvVars +ExportCertData
  SSLCertificateFile "/etc/ssl/example.cert"
  SSLCertificateKeyFile "/etc/ssl/example.key"
  SSLCACertificateFile "/etc/ssl/cadeia_de_certificados.crt"
  SSLVerifyClient require
  SSLVerifyDepth 10
</VirtualHost>
```

A.5 Funcionalidades

Esta seção descreve e exemplifica as funcionalidades presentes no Modelo **Digital-Certificate**.

A.5.1 Validar certificado

A função estática `clientHasValidCert` verifica se os campos do certificado exportado são válidos e se a checagem do servidor foi realizada com sucesso. A função retorna "verdadeiro" se o certificado for válido e "falso" se for inválido.

Código A.6 – Exemplo de uso da função `clientHasValidCert`

```
if (DigitalCertificate::clientHasValidCert())
{
    return DigitalCertificate::server_to_array();
}
else {
    return null;
}
```

A.5.2 Obter certificado

Se o certificado for válido, a função `getClientCertificate` retorna o certificado do cliente em um *array*, caso contrario é retornado `null`. O *Array* retornado pela função contém as seguintes propriedades:

- **ssl_serial**: Serial do certificado do cliente
- **ssl_client_issuer_dn**: Informação sobre a emissão e distribuição do certificado.
- **ssl_name**: Nome e informações do cliente. Exemplo: No padrão E-CPF é retornado `NOME: CPF`
- **ssl_email**: Email do proprietário do certificado.

Código A.7 – Exemplo de uso da função `getClientCertificate`

```
$cert = DigitalCertificate::getClientCertificate();

if ($cert != null &&
    DigitalCertificate::certificateAlreadyRegistered($cert))
```

```
{  
    return 'Already Registered';  
} else {  
    return 'Invalid or Not Registered';  
}
```

A.5.3 Verificar se o certificado já foi cadastrado

A função `certificateAlreadyRegistered` recebe como parâmetro um certificado e verifica se no banco já possui algum usuário associado a este certificado, retornando "verdadeiro" ou "falso".

Código A.8 – Exemplo de uso da função `certificateAlreadyRegistered`

```
if (!DigitalCertificate::  
    certificateAlreadyRegistered($certData))  
{  
    $user = new User();  
    $user->name = $certData['ssl_name'];  
    $user->email = $certData['ssl_email'];  
    $user->password = null;  
    $user->save();  
  
    $certData['user_id'] = $user->id;  
    DigitalCertificate::create($certData);  
    Auth::loginUsingId($user->id);  
  
    return redirect()->back();  
} else {  
    return redirect()->  
        back()->  
            withErrors('ERROR_CERT_ALREADY_REGISTERED')->  
                withInput();  
}
```

A.5.4 Obter certificados de determinado usuário

A função `getUserCertificates` recebe o "id" de um usuário e retorna um *Array* contendo todos os certificados associados.

Código A.9 – Exemplo de uso da função getUserCertificates

```
public function userCertificatesJSON()
{
    return json_encode(
        DigitalCertificate::
            getUserCertificates(Auth::id()));
}
```

A.5.5 Gerar assinatura digital

Para gerar a assinatura do arquivo o certificado exportado pelo servidor não é suficiente, pois não se tem acesso à chave privada do certificado. É necessário que o cliente envie para o servidor o arquivo do certificado e a senha para que seja possível assinar um arquivo digital.

A função `signFile` recebe três parâmetros: certificado, arquivo a ser assinado e senha. A função retorna a assinatura do arquivo informado.

No exemplo A.10 a função é utilizada para assinar um arquivo e fazer o download da assinatura gerada.

Código A.10 – Exemplo de uso da função signFile

```
public function signFile(Request $request)
{
    $sign = DigitalCertificate::signFile($request->file('cert'),
        $request->file('file'), $request->password);

    $headers = [
        'Content-type' => 'text/plain',
        'Content-Disposition' =>
            sprintf('attachment; filename="%s"',
                $request->file('file')->
                    getClientOriginalName().'signature'),
        'Content-Length' => mb_strlen($sign)
    ];

    return Response::make($sign, 200, $headers);
}
```


A.5.6 Validar uma assinatura digital

A função `checkFileSign` recebe três parâmetros: arquivo, assinatura e chave pública. Como resultado a função valida se o proprietário da chave pública gerou a assinatura do arquivo informado retornando "verdadeiro" ou "falso".

No exemplo A.11 a função é utilizada para checar se o arquivo foi assinado por algum usuário cadastrado na aplicação.

Código A.11 – Exemplo de uso da função `checkFileSign`

```
$result = DigitalCertificate::all()->map(function($item) use
($request) {
    $result = DigitalCertificate::checkFileSign(
        $request->file('file'),
        $request->file('signFile'),
        $item->cert);

    if ($result) {
        $user = User::All()->filter(function ($user) use ($item) {
            return $user->id == $item->user_id;
        })->first();

        return [
            'user' => $user->name,
            'check' => $result,
            'file' => $request->file('file')->getClientOriginalName()
        ];
    }
});
```

A.6 Funcionalidades do Controlador

Nesta sessão será descrito e exemplificado as funcionalidades presentes no Modelo `DigitalCertificateController`. A maioria dessas funcionalidades não devem ser acessadas diretamente pelo usuário e sim redirecionadas pela aplicação principal.

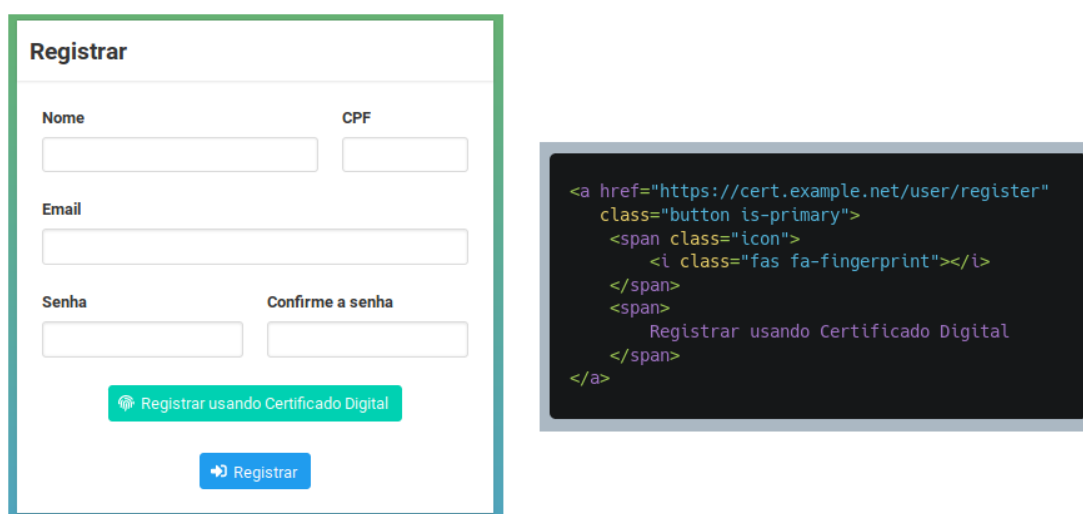
A.6.1 Certificado em JSON

Ao acessar a url `["SUBDOMINIO DO CERTIFICADO"]/json` é possível obter o certificado no formato JSON. As propriedades serão as mesmas utilizadas no modelo.

A.6.2 Registrar usuário utilizando o certificado

Para registrar o usuário utilizando o certificado é necessário redirecioná-lo para a url `["SUBDOMINIO DO CERTIFICADO"]/user/register`. Ao ser redirecionado, é verificado se o certificado é válido e se não foi utilizado por mais nenhum usuário. Se o certificado passar nessas verificações ele é cadastrado e associado ao usuário.

A figura 15 exemplifica a forma correta de se utilizar a funcionalidade. No exemplo foi criado um link "Cadastrar com o certificado" que redireciona ao subdomínio `cert.example.net` utilizado pelo pacote.



```
<a href="https://cert.example.net/user/register"
class="button is-primary">
  <span class="icon">
    <i class="fas fa-fingerprint"></i>
  </span>
  Registrar usando Certificado Digital
</a>
```

Figura 15 – Exemplo de formulário com opção para registrar utilizando o certificado

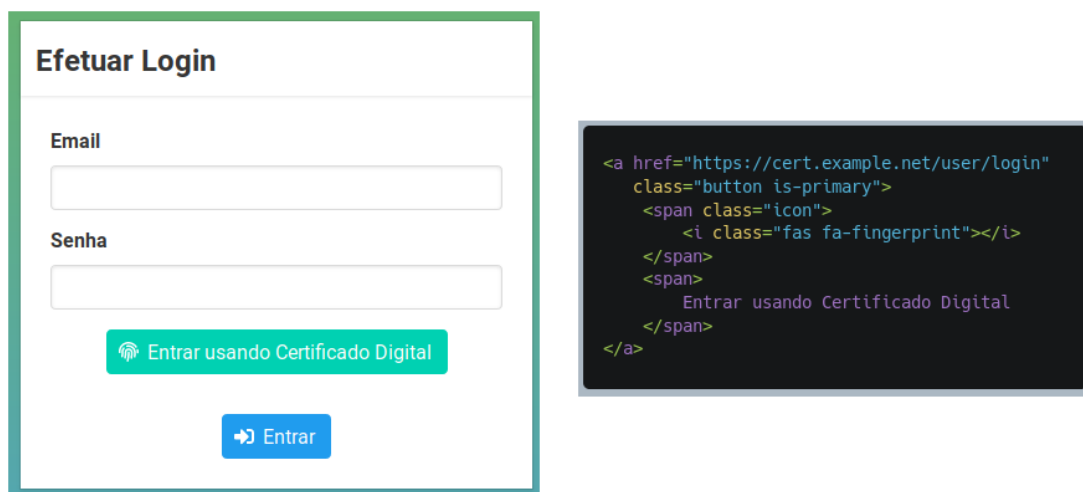
A.6.3 Efetuar login utilizando o certificado

Para efetuar o *login* é necessário redirecionar o usuário para a url:

`["SUBDOMINIO DO CERTIFICADO"]/user/login`

Ao ser redirecionado é verificado se o certificado é válido e se está registrado no sistema. Se o certificado estiver registrado é efetuado o login do usuário associado ao certificado.

A figura 16 exemplifica a forma correta de se utilizar a funcionalidade. No exemplo foi criado um link "Entrar usando Certificado Digital" que redireciona ao subdomínio `cert.example.net` utilizado pelo pacote.





The figure displays a login form and its corresponding HTML code. The form, titled "Efetuar Login", includes input fields for "Email" and "Senha" (Password). Below these fields, there is a green button with a fingerprint icon and the text "Entrar usando Certificado Digital". At the bottom of the form is a blue button with a right-pointing arrow and the text "Entrar". To the right of the form, a code block shows the HTML for the green button, using Bootstrap classes and a Font Awesome fingerprint icon.

Efetuar Login

Email

Senha

 Entrar usando Certificado Digital

 Entrar

```
<a href="https://cert.example.net/user/login"
class="button is-primary">
  <span class="icon">
    <i class="fas fa-fingerprint"></i>
  </span>
  <span>
    Entrar usando Certificado Digital
  </span>
</a>
```

Figura 16 – Exemplo de formulario com opção para efetuar login utilizando o certificado

Anexos

ANEXO A – Variaveis de ambiente relacionadas ao certificado do cliente exportado disponiveis no Apache

| Nome da variável | Tipo | Descrição |
|-------------------------|--------|--|
| SSL_CLIENT_M_VERSION | string | Versão do certificado |
| SSL_CLIENT_M_SERIAL | string | Serial do certificado |
| SSL_CLIENT_S_DN | string | Domain Name do proprietário do certificado |
| SSL_CLIENT_S_DN_x509 | string | Componente do DN do certificado |
| SSL_CLIENT_I_DN | string | Domain Name do emissor do certificado |
| SSL_CLIENT_I_DN_x509 | string | Componente do DN do emissor do certificado |
| SSL_CLIENT_V_START | string | Validade do certificado (tempo inicial) |
| SSL_CLIENT_V_END | string | Validade do certificado (tempo final) |
| SSL_CLIENT_V_REMAIN | string | Numero de dias restantes para o certificado expirar |
| SSL_CLIENT_A_SIG | string | Algoritmo usado na assinatura do certificado |
| SSL_CLIENT_A_KEY | string | Algoritmo usado na chave pública do certificado |
| SSL_CLIENT_CERT | string | Certificado codificado no formato PEM |
| SSL_CLIENT_CERT_CHAIN_n | string | Certificados da cadeia de certificados no formato PEM |
| SSL_CLIENT_VERIFY | string | Status de verificação do certificado NONE, SUCCESS, GENEROUS or FAILED |

Fonte: Apache Module mod_ssl