FACULDADE DE TECNOLOGIA DE AMERICANA – MINISTRO RALPH BIASI

Curso Superior de Tecnologia em Segurança da Informação

Luís Antonio Queiroz Dias

UM AMBIENTE SEGURO PARA APLICAÇÃO COM FRAMEWORK WEB DJANGO

Americana, SP

2020

FACULDADE DE TECNOLOGIA DE AMERICANA – MINISTRO RALPH BIASI

Curso Superior de Tecnologia em Segurança da Informação

Luís Antonio Queiroz Dias

UM AMBIENTE SEGURO PARA APLICAÇÃO COM FRAMEWORK WEB DJANGO

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Segurança da Informação, sob a orientação do Prof. Marcus Vinícius Lahr Giraldi.
Área de concentração: Segurança da Informação.

Americana, SP

2020

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

D532u DIAS, Luís Antonio Queiroz

Um ambiente seguro para aplicação com Framework web Django. / Luís Antonio Queiroz Dias. – Americana, 2020.

54f.

Monografia (Curso Superior de Tecnologia em Segurança da Informação) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Esp. Marcus Vinícius Lahr Giraldi

1 Segurança em sistemas de informação 2. Rede de computadorees I. GIRALDI, Marcus Vinícius Lahr II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.518.5

Fatec
Americana
Ministro Ralph Biasi

FACULDADE DE TECNOLOGIA DE AMERICANA – MINISTRO RALPH BIASI

Luís Antonio Queiroz Dias

UM AMBIENTE SEGURO PARA APLICAÇÃO COM FRAMEWORK WEB DJANGO

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Segurança da Informação pelo Centro Paula Souza – FATEC Faculdade de Tecnologia de Americana – Ministro Ralph Biasi.

Área de concentração: Segurança da Informação.

Americana, 30 de junho de 2020.

Banca Examinadora:

_____

Prof. Marcus Vinícius Lahr Giraldi. (Presidente)

Especialista

Faculdade de Tecnologia de Americana – Ministro Ralph Biasi

_____

Prof. Me. Eduardo Antonio Vicentini (Membro)

Mestre

Faculdade de Tecnologia de Americana – Ministro Ralph Biasi

_____

Prof. Me. Wagner Siqueira Cavalcante (Membro)

Mestre

Faculdade de Tecnologia de Americana – Ministro Ralph Biasi

DEDICATÓRIA

A Deus e especialmente a minha família, minha esposa e minha filha que estão sempre ao meu lado em todas as situações, e com este trabalho não foi diferente, me apoiaram em todo o tempo dedicado a este curso até seu trabalho de conclusão. E dedico este trabalho também a todos aqueles que se esforçam e correm atrás de seus sonhos e objetivos.

AGRADECIMENTOS

Agradeço a todos da Faculdade de Tecnologia de Americana, desde seus funcionários até os professores que sempre nos deram suporte para um aprendizado constante, e na resolução de qualquer situação acadêmica.
Sou grato de forma especial aos Professores Prof. Marcus Vinícius Lahr Giraldi e Prof. Me. Carlos Eduardo Beluzo do IFSP Campinas, que sem eles a elaboração deste trabalho não seria possível. Ao Marcus pela mentoria, orientação e companheirismo, me dando suporte em todos os momentos. Ao Carlos por ter me dado a oportunidade de trabalhar com dados de um projeto tão importante como o SAMI, e me ajudar na construção da ideia do trabalho.

RESUMO

Este trabalho de forma geral apresenta conceitos fundamentais de *framework*, *framework web*, Sistema de Gerenciamento de Banco de Dados *PostgreSQL*, framework *DJANGO*, segurança da informação, *firewall* e certificado SSL e TLS. É estudado o ambiente de um servidor *web* que hospeda uma plataforma *web*, e tem como objetivo principal propor configurações seguras ao ambiente como um todo, com base nos princípios da segurança da informação. Aborda boas práticas de desenvolvimento seguro para *software* na área da saúde, e um estudo de caso com aplicação prática das medidas propostas. Por fim conclui-se o trabalho destacando a importância de se ter um ambiente *web* seguro.

Palavras Chave: *DJANGO*; Segurança da Informação; Servidor *Web*; Cerificado Digital; NGS1; NGS2.

ABSTRACT

This work in general presents fundamental concepts of *framework*, *web framework*, *PostgreSQL* Database Management System, *DJANGO framework*, information security, *firewall* and SSL and TLS certificate. The environment of a *web* server that hosts a *web* platform is studied, and its main objective is to propose secure configurations to the environment, based on the principles of information security. It addresses good practices for safe *software* development in healthcare, and a case study with practical application of the proposed measures. Finally, the work is concluded highlighting the importance of having a secure *web* environment.

Keywords: *DJANGO*; Information Security; *Web* Server; Digital Certificate; NGS1; NGS2.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ANS	*Agência Nacional de Saúde Suplementar*
API	*Application Programming Interface*
CFM	*Conselho Federal de Medicina*
CPF	*Cadastro de Pessoas Físicas*
CRSF	*Cross Site Request Forgery*
HTML	*Hyper Text Markup Language*
HTTPS	*Hyper Text Transfer Protocol Secure*
IP	*Internet Protocol*
LTS	*Long Term Support*
MAC	*Media Access Control*
MB	*Megabyte*
MTV	*Model Template View*
MVC	*Model View Controller*
NAT	*Network Address Translation*
NGS1	*Nível de Garantia de Segurança 1*
NGS2	*Nível de Garantia de Segurança 2*
OWASP	*Open Web Application Security Project*
PEP	*Prontuário Eletrônico do Paciente*
RES	*Registro Eletrônico de Saúde*
S-RES	*Sistema de Registro Eletrônico de Saúde*
SBIS	*Sociedade Brasileira de Informática em Saúde*
SGBD	*Sistema Gerenciador de Banco de Dados*

SQL	*Structured Query Language*
SSH	*Secure Shell*
SSL	*Secure Socket Layer*
TLS	*Transport Layer Security*
XML	*Extensible Markup Language*
XSS	*Cross Site Scripting*

1 INTRODUÇÃO

A segurança é um fator essencial na computação como um todo, e ter um ambiente com uma infraestrutura segura fará o diferencial para rodar aplicações da forma mais protegida possível.

Este trabalho tem como objetivo demonstrar e propor uma forma segura para um servidor *web* e seu ambiente, ou seja, desde o Sistema Operacional até as aplicações responsáveis pela disponibilidade da plataforma *web* nele instaladas.

O ambiente em questão se trata de um servidor *web* que hospeda uma plataforma *web* de *Data Science* aplicada à saúde, que está sendo desenvolvida no Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, no campus de Campinas, pela equipe do Professor Mestre Carlos Eduardo Beluzo. A plataforma SAMI tem como objetivo utilizar técnicas de *Data Science* para auxiliar no estudo da redução neonatal, mortalidade em bebês, de forma a gerar indicadores mais precisos. A plataforma será abordada de forma superficial nesse trabalho a nível comparativo, assim como o *framework* que constitui sua infraestrutura lógica, no caso o *DJANGO*, pois ele é peça chave para funcionamento da aplicação e está diretamente relacionado ao servidor *web*.

O *DJANGO* é um *framework web* desenvolvido em *Python* (*DJANGO*, 2019), e além dele será utilizado o sistema gerenciador de banco de dados *PostgreSQL*, que será responsável por armazenar os dados coletados pela plataforma *web*.

O capítulo inicial traz os conceitos de *framework* e *web framework*, de forma objetiva, com intuito de fixar o que é, sua utilidade e importância para o desenvolvimento de uma aplicação *web*, até a introdução conceitual do *DJANGO*.

No capítulo seguinte é apresentado como está disposto o ambiente, tal como o Sistema Operacional utilizado, o banco de dados, como é feito o acesso ao servidor *web*, e como estão configuradas as aplicações descritas.

No capítulo seguinte, foram propostas as medidas de segurança para o ambiente como um todo, como *firewall* para filtrar e controlar o tráfego do servidor *web*, o uso de uma certificação SSL e TLS para criptografia da comunicação, algumas diretrizes de segurança para o banco de dados *PostgreSQL*, e as funcionalidades de segurança do *DJANGO*.

Por último foram apresentadas medidas para um desenvolvimento seguro de *software* na área da saúde, uma que vez que a plataforma irá trabalhar dados sensíveis desse segmento. Seguindo, é mostrada a implementação das medidas de segurança propostas em um ambiente de teste, sugestões de segurança para comunicação entre cliente e servidor, e para o *framework*, e, ao final, apresentado sua conclusão.

O problema tratado nesse trabalho é justamente o fato de não terem sido aplicadas devidas configurações de segurança, não apenas no Sistema Operacional, mas também nas aplicações que se fazem necessárias para o servidor *web* e a plataforma *web*.

O método utilizado foi a pesquisa bibliográfica, realizada em livros e *sites* relacionados à segurança de aplicações *web*, *DJANGO*, *PostgreSQL*, *Linux*.

2 CONCEITOS DE *FRAMEWORK*

Quando o assunto é aplicações *web*, não há como não pensar em *framework web*, pois se faz necessário para a comunicação e integração dessas aplicações de forma simples e eficaz. Sendo assim, um *framework web* permite que aplicações diferentes interajam entre si independentemente da plataforma em que foram desenvolvidas, ou se trata-se de uma aplicação nova ou mais antiga.

Neste capítulo será explicado de forma simples e objetiva o que é um *framework web*, para que serve, sua utilidade para um desenvolvedor, e o *DJANGO*.

2.1 O QUE É UM *FRAMEWORK*

Segundo Willemann e Ibarra (2007), um *framework* trata-se de uma estrutura de suporte definida, na qual o projeto de *software* pode ser estruturado para que seja realizado o seu desenvolvimento. Ainda o *framework* pode conter programas de suporte, bibliotecas de códigos, linguagens de *script* e outros *softwares* que possam auxiliar em seu desenvolvimento.

Alvim (2008) afirma que:

> "Um *framework* é um conjunto de classes que colaboram entre si de modo a prover um reuso abrangente, de grandes blocos de comportamento. Por ser reutilizável e customizável de forma refinada através de diferentes técnicas OO, um *framework* permite um ganho impossível, ou muito difícil, de ser obtido via "chamadas de sub-rotinas", reuso típico dos ambientes de desenvolvimento de "terceira geração"".

Willemann e Ibarra (2007), ainda destacam o propósito pelo qual *frameworks* foram criados, tendo em vista facilitar o desenvolvimento do *software* de uma maneira mais abrangente, de forma que os envolvidos no projeto ganhem mais tempo para o detalhamento das regras de negócio, do que em detalhes relacionados ao desenvolvimento propriamente dito.

Dessa forma Willemann e Ibarra (2007), abordam algumas características que um *framework* precisa:

- ser reutilizável.
- facilitar o desenvolvimento de sistemas.

- possuir boa documentação.
- ser completo para o que propõe.
- ser eficiente.

2.2 UTILIDADE DE UM *FRAMEWORK*

Fica clara a utilidade de um *framework* ao pensar o quão benéfica é sua utilização em um projeto de desenvolvimento.

Santos e Carvalho (2015), afirma que:

> "[...] um projeto que utiliza *frameworks* pode trazer ganhos de qualidade e produtividade. A obtenção de boa estrutura para implementação, os recursos diversos que proporcionam a facilidade para construir um projeto de *software* e o cumprimento pleno da documentação, ajuda a diminuir custos estabelecidos e aumenta a produtividade dos desenvolvedores em relação à construção do projeto de *software*".

Santos e Carvalho (2015), ainda ressaltam a reutilização de componentes, objetos, funções e aplicações, proporcionando a equipe de desenvolvimento a possibilidade de reaproveitar o conteúdo de outros projetos já desenvolvidos anteriormente, o que pode evitar retrabalho e um gasto de tempo desnecessário com o planejamento do projeto.

Pensando no desenvolvimento propriamente dito, Santos e Carvalho (2015) destacam para os *frameworks* que:

> "[...] seu principal objetivo é simplificar trabalhos, tanto para usuários quanto desenvolvedores. E uma das principais características dos *frameworks* que utilizam o paradigma de orientação a objetos, é trabalhar com reutilização de códigos".

2.3 *FRAMEWORK WEB*

Framework web (CONFIGR, 2019), tem como seu principal objetivo proporcionar um desenvolvimento ágil e seguro das aplicações *web*, diminuindo assim tempo gasto e o nível de complexidade do projeto. Com avanço nos padrões das aplicações *web* como um todo, desde aumento na demanda quanto no avanço da tecnologia por elas exigidas, os *frameworks web* tem se tornado ferramentas indispensáveis de desenvolvimento.

2.3.1 O *FRAMEWORK DJANGO*

DJANGO é um *framework web* desenvolvido na linguagem *Python* (HOLOVATY; KAPLAN-MOSS, 2009). Dessa forma como todo *framework* proporciona mais facilidades para o desenvolvimento das aplicações *web*.

O *DJANGO* faz uso da arquitetura *Model View Controller* (MVC) (HOLOVATY; KAPLAN-MOSS, 2009). Trata-se de um padrão de arquitetura de *software*, onde a aplicação é separada em três camadas sendo, *Model*, *View* e *Controller* (Modelo, Visão e Controlador) (TABLELESS, 2015).

No MVC, o *Model* é a camada que representa dos dados da aplicação (KAMIDA, 2017), sendo o principal responsável pela leitura, escrita e validação dos dados (TABLELESS, 2015). A *View* é a camada responsável pela interação *web* como o usuário através do navegador, onde são exibidos e coletados os dados por meio de XML ou HTML (TABLELESS, 2015). O *Controller* é a camada que faz o controle de todas as requisições, é o responsável por determinar quais informações serão extraídas do banco de dados pelo *Model* e exibidas aos usuários pela *View* (KAMIDA, 2017).

Ainda no *DJANGO* foi implementada outra arquitetura denominada *Model Template View* (MTV), sendo *Model* também a camada que faz o acesso aos dados, o *Template* sendo a camada de apresentação, no caso a forma com que os dados são apresentados na página *web*, e a *View* é a camada que relaciona o *Template* com o *Model* (KAMIDA, 2017).

Fica dessa forma caracterizado que o *framework web* é peça chave para a aplicação *web*, consequentemente na interação cliente servidor.

3 DESCRIÇÃO DO AMBIENTE ATUAL

Neste capítulo será abordado o ambiente do servidor *web* no qual está hospedada a plataforma e todas as ferramentas que são necessárias para o seu funcionamento.

Quando se faz uma configuração de infraestrutura, colocar as aplicações em pleno funcionamento na maioria das vezes será a primeira preocupação, no entanto colocar essas aplicações para funcionar, sem que atendam às necessidades de segurança, pode trazer consequências indesejadas (DIGITALOCEAN, 2018).

No caso do ambiente que será estudado nem todas as aplicações atendem essas necessidades de segurança.

3.1 SISTEMA OPERACIONAL

Está sendo utilizado um Sistema Operacional *open source Linux*, especificamente a distribuição do Ubuntu Server 18.04.03 LTS, sendo essa uma distribuição que possui uma instalação mais enxuta e que permite um gerenciamento mais simples em relação as demais (UBUNTU, [s.d]).

A instalação e configuração do Ubuntu foi feita seguindo as definições padrões, e não houve nenhuma customização pós instalação que pudesse proporcionar mais segurança e confiabilidade ao ambiente, sendo que não existe nenhum *firewall* configurado nesta instalação do Sistema Operacional.

3.2 APLICAÇÕES PRESENTES NO AMBIENTE

As aplicações que se encontram instaladas e configuradas, consistem no SGBD *PostgreSQL* e no *framework web DJANGO*, que são necessários para o funcionamento da aplicação *web*. Ambas as ferramentas se encontram com as configurações padrão, sem nenhuma customização ou configuração adicional que pudesse trazer mais segurança a aplicação ao ambiente como um todo.

3.2.1 SISTEMA GERENCIADOR DE BANCO DE DADOS

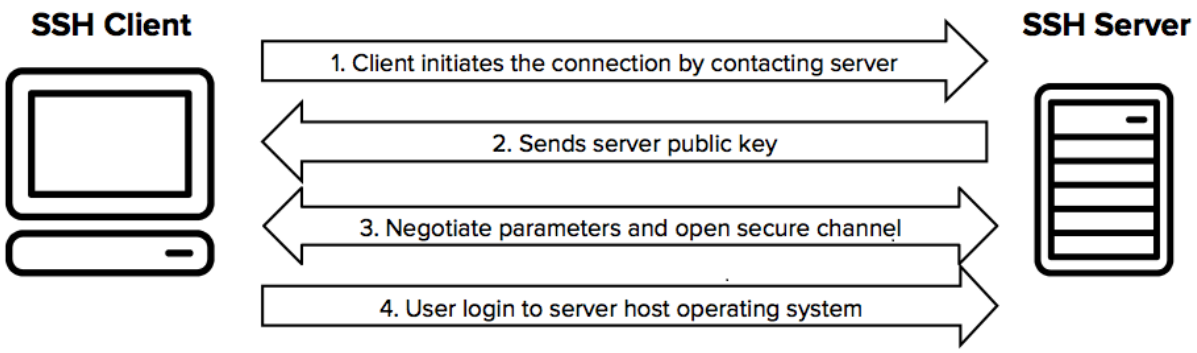
Em relação ao *PostgreSQL* (SOURCEFORGE, [s.d]), se trata de um sistema de gerenciamento de banco de dados objeto relacional, que foi desenvolvido baseado no sistema de gerenciamento de banco de dados de código aberto PostgreSQL Versão 4.2, desenvolvido pelo Departamento de Ciência da Computação da Universidade da Califórnia, sendo objeto relacional (SOURCEFORGE, [s.d]), significa: "[...] que permite aos desenvolvedores integrar ao banco de dados seus próprios tipos de dado e métodos personalizados".

O volume de dados a ser processado na base de dados não será muito grande, uma vez que não se trata de um *Big Data*, podendo variar entre 100MB e 500MB. As requisições que serão feitas a base de dados serão intermitentes, o que significa que não ocorrerá a todo momento, tendo assim apenas alguns conjuntos de dados que serão grandes, entretanto não ultrapassando o valor já destacado.

3.3 ACESSO REMOTO AO SERVIDOR

O acesso ao servidor é feito via SSH (*Secure Shell*), que se trata de um protocolo criptografado, utilizado para estabelecer uma comunicação entre as máquinas cliente e servidor (SSH.COM, [s.d]), conforme representado na Figura 1. Ao estabelecer a comunicação pode-se realizar o acesso ao servidor, tendo assim o controle para realizar as tarefas necessárias.

Figura 1 – Comunicação via SSH.



Fonte: ssh.com (2019).

A autenticação utilizada no acesso é realizada, tanto por *login* e senha do usuário administrador, no caso o *root*, quanto por chaves SSH, em que o usuário gera o par de chaves pública e privada, sendo sua chave pública inserida no servidor, e, ao se conectar no servidor, será exigida a identificação através da chave privada.

O acesso pelo par de chaves SSH mostrou ser uma forma simples e segura para autenticação no servidor, e a única configuração de segurança que foi aplicada durante a configuração do ambiente do servidor *web*. Isso traz mais segurança (DIGITALOCEAN, 2018), pois:

> "[...] a autenticação por chave SSH o permite desativar a autenticação baseada em senha. Chaves SSH geralmente possuem muito mais bits de dados do que uma senha, o que significa que há significativamente mais combinações possíveis que um atacante teria que executar".

Tendo em vista que nem todas as aplicações contidas no servidor *web* estão seguras, faz-se necessário que configurações de segurança sejam aplicadas para garantir a integridade e confidencialidade dos dados e do ambiente como um todo.

4 MEDIDAS DE SEGURANÇA PROPOSTA

A segurança da informação é essencial como um todo, principalmente para um ambiente que armazena dados e tem uma interação cliente servidor através de um serviço *web*.

Araújo (2015) define segurança da informação como: “[...] sendo uma proteção dos itens de informação em combate a ameaças que comprometem a sua integridade, confidencialidade e disponibilidade”.

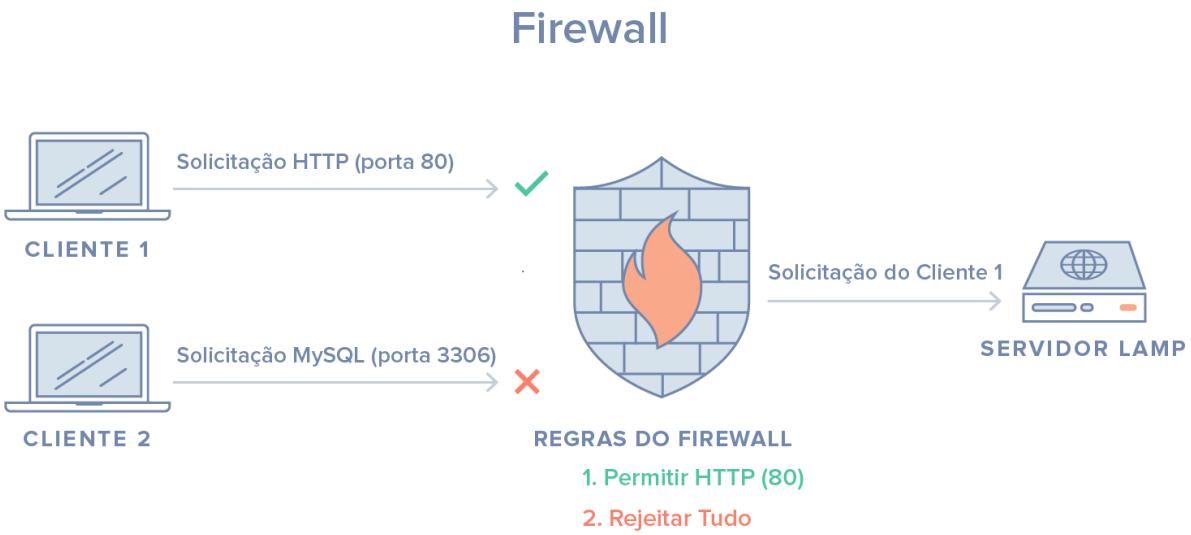
Segundo Moraes (2015), confidencialidade, integridade e disponibilidade são na verdade serviços de segurança, sendo que a confidencialidade garantiria o acesso à informação apenas aos usuários autorizados, a integridade irá garantir que a informação não será alterada em nenhuma circunstância adversa, e a disponibilidade irá garantir que a informação esteja sempre disponível quando solicitada.

Da forma que se apresenta todo ambiente, nota-se que está totalmente vulnerável e suscetível a ataques. Neste capítulo serão apresentadas várias medidas a serem aplicadas de acordo com a aplicação que não se encontra protegida.

4.1 *FIREWALL*

Um *firewall* pode tratar-se de uma peça de *software* ou *hardware* que faz o controle de acesso as portas e serviços que, de certa forma, é quem define como são feitas as restrições e os bloqueios dessas portas e serviços, com exceção daqueles que devem ser públicos (DIGITALOCEAN, 2018), conforme representado na Figura 2.

Figura 2 – *Firewall*.



Fonte: DigitalOcean (2018).

De acordo com Moraes (2015), o *firewall*: “[...] ou “parede de fogo”, é um sistema que atua como ponto único de defesa entre a rede privada e a rede pública. Ele pode ainda controlar o tráfego entre as sub-redes de uma rede privada”. Ainda destaca que todo o tráfego de entrada e saída deve passar de forma obrigatória por esse sistema, uma vez que ele pode restringir, negar e registrar tudo que passa por ele, e ainda afirma que um *firewall* não é um programa mas um conjunto de recursos de *hardware* e *software*, com objetivo de garantir a segurança da rede.

Segundo Moraes (2015), suas principais funções são:

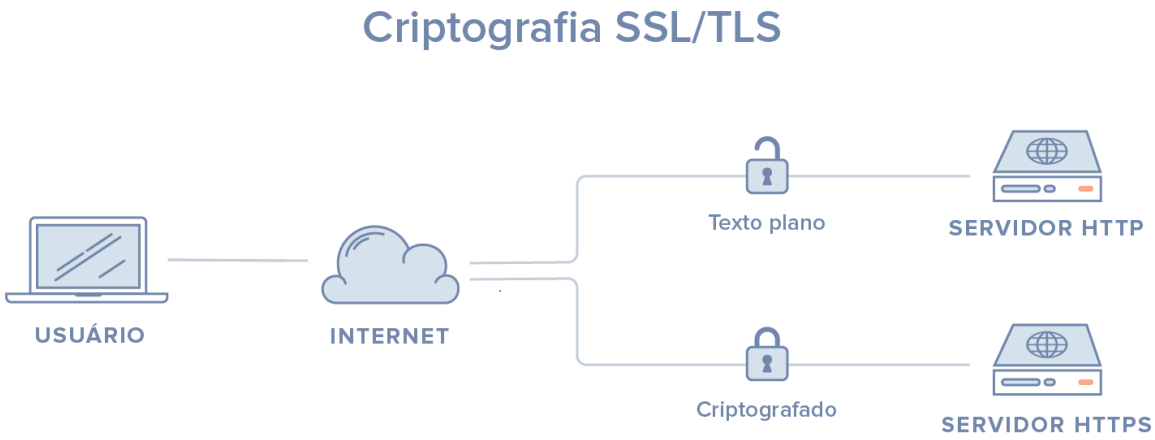
- Estabelecimento de um perímetro de segurança.
- Separar as redes e controlar os acessos.
- Ser um elemento central de controle e aplicação de políticas de segurança.
- Proteger sistemas vulneráveis na rede.
- Aumentar a privacidade.
- Registrar e gerar estatísticas do uso da rede e acessos indevidos.

Sem dúvidas, a implementação de um *firewall* no servidor *web* irá trazer mais segurança e controle de tudo que está entrando ou saindo do ambiente.

4.2 CERTIFICADOS SSL/TLS

O SSL (*Secure Socket Layer*) e TLS (*Transport Layer Security*), são protocolos de criptografia que tem a função de prover segurança durante a troca de dados entre as aplicações e servidores. Dessa forma, depois de feita a autenticação podem ser utilizados na criptografia da comunicação, conforme Figura 3 (DIGITALOCEAN, 2018).

Figura 3 – Certificados SSL/TLS.



Fonte: DigitalOcean (2018).

A Implementação de um certificado irá garantir mais segurança, pois estabelece uma autoridade de certificação, e ainda definindo-se um certificado para o servidor, permitirá que os elementos presentes na infraestrutura façam a validação da identidade dos outros elementos e criptografe seu tráfego, validação essa que é feita pela assinatura de uma determinada autoridade certificadora confiável configurada no servidor. Isso pode prevenir contra ataques de *man-in-the-middle*, em que o atacante faz-se passar por um servidor dentro de sua infraestrutura para fazer a interceptação do tráfego (DIGITALOCEAN, 2018).

4.3 SGBD *POSTGRESQL*

O *PostgreSQL* sem dúvidas é um poderoso Sistema Gerenciador de Banco de Dados (SGBD), que pode ser executado nos principais sistema operacionais, dos quais incluem *Linux*, Unix e Windows, e oferece confiabilidade, integridade dos dados e correção (IBM, 2009).

Dentre as ações a serem tomadas em relação ao banco de dados, criar os usuários e definir os privilégios de acesso é fundamental. Validar quem terá acesso ao banco de dados seja com um acesso direto ou através da aplicação, para assim definir um perfil e aplicar o mesmo, sendo que este pode ter suas permissões alteradas. O administrador do banco de dados deve ter conhecimento de todos que estão acessando o banco de dados.

Outra ação que pode ser tomada se refere à criação de senhas, sendo que cada uma deve ser forte, e na medida do possível que seja criptografada.

Ainda deve ser levada em consideração a forma que será feita a autenticação no banco de dados, levando em conta as informações que serão necessárias para autenticação, como porta e IP (IBM, 2009). Pensando dessa forma, além dos dados, a sessão também poderia ser criptografada, fazendo uso até mesmo de certificação SSL.

4.4 *FRAMEWORK WEB DJANGO*

O *DJANGO* possui algumas funcionalidades de segurança que serão abordadas a título informativo e como dica de implementação de segurança, uma vez que o trabalho não visa o desenvolvimento de uma aplicação *web*, mas sim um ambiente seguro onde estará hospedada.

O *DJANGO* possui proteção contra *Cross Site Scripting* (XSS) que, de acordo com *DJANGO* (2019):

> “Ataques XSS permitem a um usuário injetar *scripts* do lado cliente dentro do browser de outros usuários. Isso geralmente é feito armazenando *scripts* maliciosos no banco de dados onde ele será obtido e exibido para outros usuários, ou fazendo usuários clicarem em um *link* que faz com que o Java Script do invasor seja executado pelo *browser* do usuário”.

Ainda, os ataques XSS podem ter origem de qualquer fonte de dados não confiáveis, podendo ser *cookies* de *sites* maliciosos ou de serviços *web* (*DJANGO*, 2019).

O *DJANGO* conta com proteção contra *Cross Site Request Forgery* (CSRF) que, de acordo com *DJANGO* (2019) são: “Ataques de solicitações forjadas entre sites, na sigla em inglês, CSRF, permitem que um usuário malicioso execute ações usando as credenciais de outro usuário sem o seu consentimento ou conhecimento”.

Ainda assim vale ressaltar que, por mais que o *framework* possua proteção contra a maior parte dos tipos de ataques CSRF, é necessário que esteja sendo usado e configurado de forma adequada (*DJANGO*, 2019).

As consultas feitas pelo *framework* ao banco de dados possuem proteção conta SQL *Injection* que, de acordo com *DJANGO* (2019): “[...] é um tipo de ataque onde o usuário malicioso consegue executar código SQL arbitrário em um banco de dados. Isso pode resultar em registros sendo deletados ou vazamento de dados”.

Os códigos das consultas SQL que são feitos pelo *DJANGO*, são feitos de forma separada do parâmetro da consulta, pois esses parâmetros podem ter sido fornecidos pelo usuário, dessa forma não sendo seguro (*DJANGO*, 2019).

O *DJANGO* conta com proteção contra *Clickjacking* que, conforme *DJANGO* (2019):

> “*Clickjacking*, ou roubo de *click*, é um tipo de ataque onde um site malicioso embrulha outro site dentro de um *frame*. Esse ataque pode resultar em um usuário desavisado sendo levado a fazer ações não intencionadas no site alvo”.

A proteção propriamente dita está dentro do *form* do *middleware*, que no caso o navegador que tenha esse suporte, pode vir a prevenir que o *site* em questão possa ser aberto dentro de outro *frame* (*DJANGO*, 2019).

Outra opção para manter o *site* seguro, é a implementação utilizando o HTTPS (*Hyper Text Transfer Protocol Secure*), que é um protocolo de comunicação de Internet que utiliza o protocolo SSL. Sem esse tipo de proteção, existe a possibilidade de usuários mal-intencionados interceptarem informações entre o cliente e o servidor, e até mesmo alterar esses dados (*DJANGO*, 2019).

De certa forma, mesmo que as aplicações descritas venham propor uma forma de proteção segura em suas configurações, uma implementação do sistema deve ser feita de forma adequada para que venha aproveitar da melhor forma possível todas as configurações de segurança do ambiente como um todo.

5 APLICAÇÃO DAS MEDIDAS PROPOSTAS

Neste capítulo será abordada a aplicação prática das medidas de segurança proposta para o ambiente do servidor *web*, levando em consideração todas as ferramentas que compõem o funcionamento da aplicação, assim como abordagem de boas práticas de segurança para seu desenvolvimento.

Sobre o ambiente de teste, trata-se de um ambiente virtual e a criação do mesmo foi baseada na estrutura apresentada anteriormente no capítulo 3, com o intuito de chegar o mais próximo possível do ambiente original e obter os resultados mais condizentes com a realidade da estrutura que hospeda a aplicação.

Essa aplicação prática tem como foco a demonstração de configurações e práticas de segurança que possam vir a ser implementadas sem maiores dificuldades.

5.1 PRÁTICAS DE SEGURANÇA PARA DESENVOLVIMENTO

Por se tratar de uma aplicação que fará uso de uma base diretamente relacionada a área da saúde com dados sensíveis, como mortalidade neonatal e possível visualização de dados através de métodos de aprendizagem de máquina (MOTA, CAIO AUGUSTO DE SOUZA et al., 2019), é recomendado seguir algumas práticas de desenvolvimento seguro específicas para esse tipo de aplicação, justamente para que não haja exposição desses dados.

Atualmente no Brasil existe uma agência reguladora de saúde, a Agência Nacional de Saúde Suplementar (ANS), que está vinculada ao ministério da saúde, e tem como objetivo regulamentar a comercialização de planos de saúde por operadoras e seguradoras, e também aos serviços de saúde do setor público (MUNIZ; HESSEL, 2019).

A ANS, também preza pela segurança e privacidade dos dados, conforme a subseção IV da Resolução Normativa RN Nº 305, de 9 de outubro de 2012, que trata justamente de Segurança e Privacidade, visando não apenas a proteção dos dados do paciente, mas também o sigilo do profissional envolvido. Sendo assim:

> “Subseção IV

> Segurança e Privacidade
>
> Art. 14. O componente de segurança e privacidade estabelece os requisitos de proteção dos dados de atenção à saúde.
>
> § 1º O componente de segurança e privacidade visa assegurar o direito individual ao sigilo, à privacidade e à confidencialidade dos dados de atenção à saúde.
>
> § 2º O componente de segurança e privacidade baseia-se no sigilo profissional e segue a legislação vigente no País.” (ANS, 2012)

Dessa forma, em 2002 a Sociedade Brasileira de Informática em Saúde (SBIS) e o Conselho Federal de Medicina (CFM), motivados pelo fato de que as informações dos pacientes não estarem sendo armazenadas de uma forma segura, criaram um processo de certificação para os Sistemas de Registro Eletrônico de Saúde (S-RES), que seria: “[...] qualquer sistema de informação que capture, armazene, apresente, transmita ou imprima informação identificada em saúde”, que está ligado ao Prontuário Eletrônico do Paciente (PEP), e ao Registro Eletrônico De Saúde (RES), que é um tipo de registro eletrônico do paciente (SBIS, [s.d]).

Ainda de acordo com a Sociedade Brasileira de Informática em Saúde (SBIS), a certificação SBIS-CFM trata-se de um processo de auditoria que avalia se o *software* atende 100% dos requisitos proposto no manual (SBIS, [s.d]), tendo como objetivo:

- Aumentar a segurança da informação armazenada em sistemas de PEP/RES.
- Criar os regulamentos e normativas para o suporte legal para eliminação do papel (prontuário eletrônico).
- Melhorar a qualidade dos sistemas de informação em saúde no Brasil.

Visando atender a esses requisitos, foram criados dois Níveis de Garantia de Segurança (NGS) que, acordo com a CERTISIGN (2019), define-se como:

- NGS1: define uma série de requisitos obrigatórios de segurança, como controle de versão do *software*, controle de acesso e autenticação, disponibilidade, comunicação remota, auditoria e documentação.
- NGS2: exige a utilização de certificados digitais para os processos de assinatura e autenticação.

Os principais níveis de segurança serão abordados na próxima seção como requisito de segurança para implementação no desenvolvimento da aplicação.

5.1.1 REQUISITOS PARA DESENVOLVIMENTO DE *SOFTWARE* NA ÁREA DA SAÚDE

Nesta seção serão apresentados os requisitos de boas práticas de segurança em desenvolvimento para *software* da área da saúde, conforme as normas NGS1 e NGS2 da SBIS-CFM em sua última versão de 2019.

As normas NGS1 e NGS2 foram desenvolvidas para *softwares* na área da saúde que informatizam os prontuários dos pacientes. Elas estabelecem requisitos de segurança que devem ser adotados no desenvolvimento do S-RES, e abordam questões que vão desde o controle de versionamento, até a forma com que é estabelecida a comunicação entre cliente e servidor (MUNIZ; HESSEL, 2019).

Não serão abordados todos os itens das normas, mas apenas os que são mandatórios e estão relacionados com a elaboração deste trabalho, sendo que os demais itens poderão ser aplicados futuramente em uma possível implementação.

Os requisitos serão apresentados em forma de tabela descritiva, que contém quatro colunas com os seguintes campos:

- “ID” que corresponde ao número do item da norma;
- “Título” sendo o título do item;
- “Obrigatório” que corresponde à obrigatoriedade do item em relação a implementação da norma, que no caso está representado pela letra M de mandatório;
- “Requisito” correspondente à prática sugerida.

Quadro 1 mostra os requisitos de acordo com a norma NGS1 para os itens relacionados à identificação e autenticação de pessoas, controle de sessão de usuário, autorização e controle de acesso de pessoas, disponibilidade do RES, comunicação entre os componentes do S-RES, segurança de dados e privacidade.

Quadro 1 Requisitos NGS1 para implementação na aplicação.

ID	TÍTULO	OBRIGATÓRIO	REQUISITO

NGS1.02.02	Método de autenticação de pessoa	M	Neste requisito, o recomendado para a autenticação é o uso de um login de usuário com o nome do usuário e uma senha forte, e se possível com a utilização de um certificado digital com senha. Essas credenciais devem ser validadas no lado do servidor a fim de identificar a pessoa que está realizando o acesso.
NGS1.02.03	Proteção dos parâmetros de autenticação de usuário	M	Os usuários e suas respectivas senhas devem ser armazenados no banco de dados, sendo que a senha deve ser protegida por criptografia, e o acesso a elas devem se dar apenas aos usuários de banco dados utilizados pela aplicação e os administradores de sistema e banco de dados.
NGS1.02.04	Segurança de senhas	M	A senha deve conter no mínimo 8 caracteres devendo ter números, letras maiúsculas e minúsculas, e caracteres especiais como @ e #. O usuário deve ser capaz de trocar a senha quando necessário durante o acesso a aplicação, no caso em que a solicitação de troca não for feita pelo próprio usuário o processo deve ser bloqueado impedindo a visualização da
ID	**TÍTULO**	**OBRIGATÓRIO**	**REQUISITO**
			senha atual.

			A expiração de senha deve ser de ao menos 3 meses, e deve-se exigir que a nova senha seja diferente das anteriores, e o usuário deve ser sempre comunicado desse procedimento.
NGS1.02.05	Controle de tentativas de login	M	Para o controle de tentativas de acesso, é recomendado que não exceda o número de 10 tentativas, após isso o usuário dever ser bloqueado e seu acesso se dará apenas após o debloqueio feito pelo administrador do sistema. O usuário deve ser comunicado desse procedimento.
NGS1.02.06	Identidade única da pessoa e responsabilização	M	A identificação da pessoa deve ser feita de forma individual vinculada ao seu CPF. Os usuários deverão sempre ter esse documento validado em todos os processos realizados pela aplicação, para garantir que o mesmo não esteja associado a mais de um usuário cadastrado. Recomenda-se que para fins de responsabilização nenhum usuário seja removido do banco de dados, para possíveis auditorias e em casos que o usuário possa ter feito uso indevido da aplicação.
NGS1.02.08	Informações na autenticação	M	Após a autenticação bem sucedida, o sistema deverá
ID	**TÍTULO**	**OBRIGATÓRIO**	**REQUISITO**

			exibir a data e hora da última autenticação, tanto para as feitas com sucesso, quanto para as que não foram completadas.
NGS1.03.01	Bloqueio ou encerramento por inatividade	M	Deve-se criar um período de *timeout* para a sessão ativa em caso de inatividade, onde o usuário deverá ser bloqueado automaticamente e as telas que estavam sendo utilizadas sejam encerradas, e assim para visualizá-las novamente deverá ser aberto uma nova sessão. A nova sessão deverá ser exigida do usuário e o desbloqueio deverá ser feito apenas por ele.
NGS1.03.02	Segurança contra roubo de sessão de usuário	M	A sessão que estabelece a comunicação entre o cliente e o servidor deve ser criptografada, e devem ser implementados mecanismos que impeçam o roubo de sessão e sua reutilização, e esses mecanismos não devem ser visíveis aos usuários para que não venham a ser desabilitados.
NGS1.04.04	Papéis relacionados à TI	M	Fazer a gestão de toda a parte relacionada ao ambiente onde a aplicação está hospedada, ou seja, a administração total do sistema e suas funcionalidades como controle de acesso, perfis e grupos de usuários, e logs do
ID	**TÍTULO**	**OBRIGATÓRIO**	**REQUISITO**

			sistema. Definição e aplicação de uma política de backup e restauração da base de dados via SGBD.
NGS1.04.05	Configuração de controle de acesso	M	Definir uma política de acesso levando em consideração todos que irão acessar a aplicação e a base de dados, seus respectivos papéis, funções e o tipo das operações que serão feitas. Dessa forma poderão ser realizadas as devidas configurações de permissões e restrições de acesso.
NGS1.04.06	Usuário mínimo ativo e restrição de autoconcessão de direitos	M	Dever haver ao menos um super usuário com perfil de administrador e um usuário como gestor de acesso, sendo que este último não deve ter permissão para alterar suas próprias permissões.
NGS1.05.01	Cópia de segurança	M	Deve ser realizado um *backup full* da base de dados periodicamente conforme política já definida, armazenar essas cópias de forma segura e garantir a integridade dos dados exportados, para que em sua restauração todos os dados sejam recuperados de forma automática e sem erros. É recomendado sempre fazer testes nos *backups* para validar se tudo ocorrerá sem algum problema. Somente o administrador deverá ter
ID	**TÍTULO**	**OBRIGATÓRIO**	**REQUISITO**

			acesso para fazer o *backup* e a restauração.
NGS1.05.02	Integridade na restauração da cópia de segurança	M	Como já recomendado, deve ser verificado a integridade dos dados, e em caso de algum problema durante o processo de restauração, deve ser feito o *rollback* e nenhuma informação deve ser restaurada.
NGS1.05.03	Alerta de limiar de ocupação	M	A aplicação deve permitir um gerenciamento de espaço que identifique a ocupação, e gere um alerta ao administrador de TI quando o limite for atingido.
NGS1.06.01	Segurança da comunicação com componente de interação com o usuário	M	O componente que irá interagir com a aplicação, como por exemplo um navegador de internet, deve oferecer uma conexão segura com criptografia na comunicação, essa funcionalidade deve ser validada pela aplicação.
NGS1.06.02	Controle de acesso do cliente ao servidor	M	A aplicação deve identificar a origem da solicitação de autenticação para autorizá-la ou não. Deve gerar uma listagem dessas informações para armazená-las e consultá-las, para que assim possam restringir o acesso apenas as origens autorizadas. Tal processo poderia ser feito por uma tabela de endereços IP e o MAC da origem vinculados aos usuários.
ID	**TÍTULO**	**OBRIGATÓRIO**	**REQUISITO**

NGS1.06.03	Processamento de dados no lado servidor	M	É recomendado que todo o processamento de dados, assim como sua validação, seja feito apenas do lado do servidor, e que seja apresentado para o usuário apenas o resultado desse processamento.
NGS1.06.04	Segurança da comunicação entre os componentes	M	Assim como tratado no componente do usuário com a aplicação, a comunicação entre a aplicação e o banco de dados deve ser criptografada e uma autenticação criada apenas entre as duas pontas, ou seja, entre o servidor e cliente.
NGS1.06.05	Controle de acesso entre componentes	M	O acesso deve ser restrito e realizado apenas entre a aplicação e o banco de dados.
NGS1.07.04	Verificação de integridade dos dados	M	É recomendado que na aplicação seja implementado um tratamento que verifique a integridade dos dados, de forma que possa prevenir que qualquer ação possa causar inconsistência de dados, seja essa ação tomada por parte do usuário ou algum erro de sistema.
NGS1.07.05	Utilização do SGBD	M	É de extrema importância que todos os dados sejam armazenados apenas em um Sistema de Gerenciamento de Banco de Dados (SGBD). Em caso de anexos independente do formato, devem ser armazenados no
ID	**TÍTULO**	**OBRIGATÓRIO**	**REQUISITO**

			servidor em estrutura de diretório devidamente criptografado e acessível apenas pela aplicação, e tanto o diretório quanto o arquivo não devem ter nomes que possam identificar seu conteúdo.
NGS1.07.06	Impedir acesso direto ao SGBD	M	O acesso ao banco de dados pelo usuário, deve-se dar apenas pela aplicação com as devidas restrições definidas previamente, nunca diretamente pelo SGBD, sendo este último acessado pelo administrador.
NGS1.07.10	Validação de dados de entrada	M	O tratamento de validação dos dados de entrada deve ser feito antes de seu processamento, para evitar possíveis ataques, como de *SQL Injection*, evitando dessa forma um vazamento de dados sensíveis.
NGS1.07.11	Segregação dos dados por organização	M	É de vital importância que caso a aplicação trabalhe com uma base de dados privada, não pública, esta base não deve ficar disponível nem ser compartilhada com outra organização ou instituição, a não ser que haja um acordo devidamente formalizado entre as partes, algo fora desse contexto poderá caracterizar vazamento de dados. Vale ressaltar que o administrador sempre terá
ID	**TÍTULO**	**OBRIGATÓRIO**	**REQUISITO**

			acesso à informação, e deve fazer valer de sua ética professional.
NGS1.12.01	Concordância com os termos de uso	M	O termo de uso deve ser elaborado levando em consideração todas as políticas de acesso e segurança adotadas para o uso da aplicação e os dados por ela utilizados. Deve ser exibido para o usuário no primeiro acesso, ser claro em seus termos, e alertar o usuário sobre a confidencialidade dos dados, e das consequências caso seja feito uso indevido dessas informações. O usuário só poderá prosseguir com a utilização da aplicação após concordar com o termo.
NGS1.12.05	Propósito de uso	M	O usuário deverá preencher um documento onde irá descrever o propósito de uso das informações.

Fonte: Autoria Própria – adaptado da norma NGS1

Os requisitos de acordo com a norma NGS2 para os itens relacionados ao certificado digital. Os certificados digitais devem ser válidos e emitidos por uma entidade certificadora, ou a pessoa pode estar credenciada na Infraestrutura de Chaves Públicas Brasileira ICP-Brasil, que: “[...] é uma cadeia hierárquica de confiança que viabiliza a emissão de certificados digitais para identificação virtual do cidadão”, que é mantida pelo Instituto Nacional de Tecnologia da Informação ITI (ITI, 2017).

Quadro 2 Requisitos NGS2 para implementação na aplicação.

ID	TÍTULO	OBRIGATÓRIO	REQUISITO
NGS2.01.03	Validação do certificado digital antes do uso	M	Antes de sua utilização o certificado digital deve ser validado pela aplicação, levando em consideração a cadeia de certificação, criptografia, validade e revogação. Essa validação deve ser feita no lado do servidor fazendo uso dos certificados raiz já configurados nele, dessa forma apenas certificados de confiança serão permitidos.
NGS2.01.04	Configuração de certificados raiz do S-RES	M	A aplicação só deverá permitir a configuração de certificados de confiança devidamente registrados por uma entidade certificadora.
NGS2.01.05	Tipos de usuários para autenticação com certificação digital	M	Caso o usuário faça uso de uma certificação da cadeia da ICP-Brasil, a autenticação deve ser feita com ela.
NGS2.01.06	Compatibilidade com mídias para certificação digital	M	A aplicação deve ser compatível ao menos com as mídias que são homologadas pela ICP-Brasil.

Fonte: Autoria Própria – adaptado da norma NGS2

A implantação dessas normas tem como finalidade, não apenas certificar e ajustar o sistema em um conjunto de regras específico, mas também o tornar mais

seguro e confiável para aqueles que farão uso de suas funcionalidades. Como dito anteriormente que a aplicação irá trabalhar com dados sensíveis, ter esses parâmetros de segurança e confiabilidade trará mais visibilidade para o projeto, o que certamente será muito benéfico.

5.2 REGRAS DE SEGURANÇA *FIREWALL*

Conforme já destacada sua importância em um ambiente computacional, a intenção com a implementação de um *firewall*, é poder proporcionar uma camada de segurança a mais no ambiente, criando regras de acesso que podem liberar ou restringir alguns serviços dentro servidor.

Como está sendo utilizado um Sistema Operacional *Linux*, para o experimento foi utilizado o *iptables* na criação das regras de acesso, que é um *firewall* nativo do *Linux* e está presente na maioria das distribuições (4FASTERS, 2018). Dentre suas principais funções estão a criação e controle das regras de acesso e de NAT, o que proporciona uma melhor filtragem dos pacotes, e dessa forma, com um gerenciamento adequado, pode-se ter um controle das entradas e saída de uma rede.

5.2.1 REGRAS APLICADAS NO *FIREWALL*

Foram criadas regras pensando nos serviços que acessam o ambiente, que no caso seriam da aplicação *web*, do acesso feito pelo administrador do sistema via SSH, e do acesso feito pelos usuários através da internet.

Ao criar as regras, deve-se pensar, tanto no fluxo de entrada, quanto no de saída de pacotes, pois o acesso é feito nas duas vias. Essas regras foram salvas em um arquivo chamado ‘*rules.v4*’, que está no diretório ‘*/etc/iptables/rules.v4*’, para que o *iptables* possa sempre iniciá-las junto como o Sistema Operacional, uma vez que ele checa este arquivo automaticamente para verificar se há alguma regra especificada. A seguir, demonstram-se as regras que foram criadas para os serviços descritos.

As regras de acesso *web* liberam o fluxo nas portas 8000, 80 e 443, que representam os serviços do *DJANGO* e do *Nginx*, este último fazendo uso do HTTP e HTTPS.

- #iptables -A INPUT -p tcp --dport 8000 -j ACCEPT
- #iptables -A OUTPUT -p tcp --dport 8000 -j ACCEPT
- #iptables -A INPUT -p tcp --dport 80 -j ACCEPT
- #iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
- #iptables -A INPUT -p tcp --dport 443 -j ACCEPT
- #iptables -A OUTPUT -p tcp --dport 443 -j ACCEPT

Já as regras para acesso via SSH libera o fluxo na porta 8511 que foi alterada para esse experimento e por segurança, pois os ataques inicialmente sempre visam as portas padrão dos serviços.

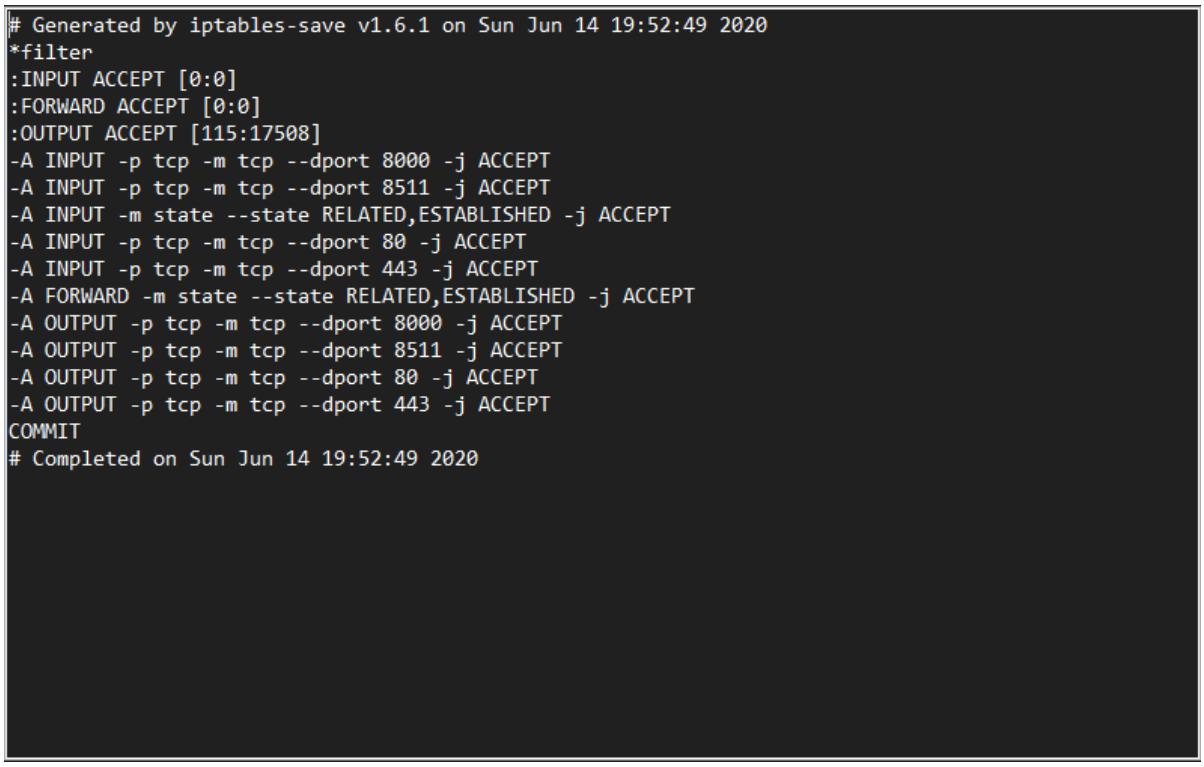
- #iptables -A INPUT -p tcp --dport 8511 -j ACCEPT
- #iptables -A OUTPUT -p tcp --dport 8511 -j ACCEPT

As regras para acesso vindo da Internet recebem apenas os pacotes que são destinados ao servidor, descartando os desnecessários .

- #iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
- #iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

A figura 4 mostra com as regras ficaram no arquivo ‘*rules.v4*’.

Figura 4 – Arquivo '*rules.v4*'.



```
# Generated by iptables-save v1.6.1 on Sun Jun 14 19:52:49 2020
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [115:17508]
-A INPUT -p tcp -m tcp --dport 8000 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 8511 -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 443 -j ACCEPT
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 8000 -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 8511 -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 443 -j ACCEPT
COMMIT
# Completed on Sun Jun 14 19:52:49 2020
```

Fonte: Autoria Própria.

Com a implementação dessas regras pode-se ter maior controle dos fluxos de acessos, e evitar possíveis tentativas de ataques.

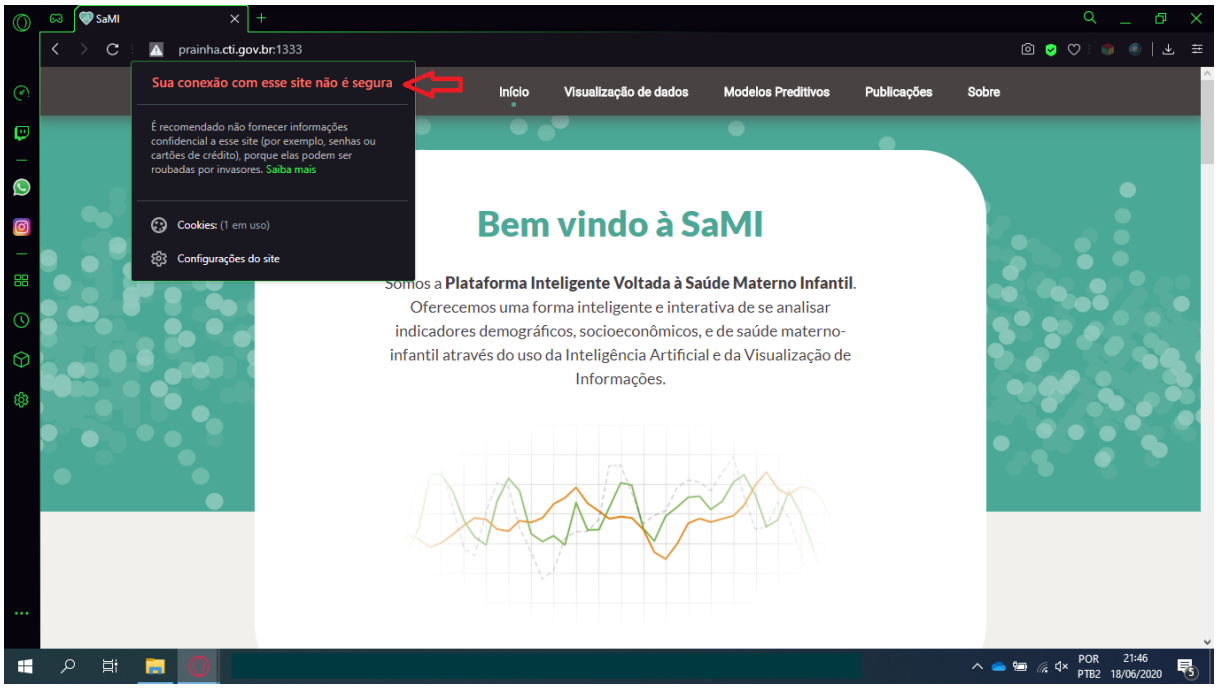
5.3 SEGURANÇA COM CERTIFICADO DIGITAL

O uso de certificado digital em páginas *web* é uma das principais formas de fazer uma conexão segura entre a aplicação e o servidor, pois estabelece uma conexão criptografada em que os dados podem trafegar de forma segura. A autoridade desse certificado é validada pelo protocolo HTTPS, que faz uso do protocolo HTTP com o protocolo de segurança SSL.

A aplicação que foi analisada, a plataforma SAMI, não faz uso de certificação digital, o que a torna completamente vulnerável e suscetível a ataques que podem interceptar a comunicação por captura de pacotes.

A figura 5 mostra como está a disposição da plataforma atual sem a utilização do HTTPS.

Figura 5 – Plataforma SAMI sem HTTPS.



Fonte: Autoria Própria.

Ao acessar o *site*, pode-se notar que no canto esquerdo do navegador é sinalizado um aviso de que a conexão não é segura, e ao clicar no botão, o aviso é exibido, tornando explicita essa informação.

Para garantir não apenas a segurança dos dados, mas também a confiabilidade da plataforma, deve ser implementado o uso do certificado digital de forma efetiva, sendo que este deve ser válido e emitido por uma entidade certificadora credenciada, ou ainda pela ICP-Brasil conforme recomendado na norma NSG2.

O Tratamento do certificado deve ser feito a nível de código, e a cadeia confiável deve ser devidamente configurada no servidor para assim estabelecer a comunicação assinada e criptografada entre cliente e servidor.

Com a implementação de um certificado digital, as vulnerabilidades a serem exploradas na plataforma diminuem, tornando assim o ambiente mais seguro e confiável.

5.4 SEGURANÇA EM BANCO DE DADOS

O Sistema Gerenciador de Banco de Dados utilizado, como descrito no capítulo 3, é o *PostgreSQL* e a aplicação atualmente não está alimentando uma base de dados real, pois está em fase de desenvolvimento. Dito isto, no experimento foram feitos testes de acesso externo ao banco de dados, como deve ser feito pelo administrador do banco de dados.

Serão feitos dois tipos de acesso ao banco de dados, pelo administrador de banco de dados através de uma interface que gerencie o banco de dados, como por exemplo *pgAdmin*, que é uma ferramenta nativa do *PostgreSQL*. A outra forma seria o acesso feito pela própria aplicação fazendo uso das requisições de consulta, inserção, entre outras.

Deve-se levar em conta que todas as operações relacionadas ao banco de dados devem ser de conhecimento do administrador de banco de dados ou de sistema, uma vez que farão todo o gerenciamento como criação de usuários, permissões, *backup*, auditoria etc.

Dessa forma, para que a aplicação faça o acesso à base de dados, deve ser criado um usuário de banco de dados com permissões limitadas de acordo com o tipo de acesso que será feito, como por exemplo, com permissões de consulta e atualização. Esse usuário do banco de dados irá corresponder ao usuário que está acessando a aplicação via *web*, e poderá ser criado pelo administrador de banco dados ou pela aplicação a nível de código, respeitando as especificações de permissões definidas pelo administrador.

5.4.1 ACESSO EXTERNO AO BANCO DE DADOS

Ao instalar o banco de dados, a primeira medida tomada foi alterar a senha do *superuser* (superusuário) do *PostgreSQL* com uma senha extremamente forte, fazendo uso de elementos em sua composição que dificultem sua descoberta, como a utilização de uma quantidade mínima de caracteres a ser definida pelo administrador, o uso de letras maiúsculas e minúsculas, de números, e caracteres especiais como ‘@’, ‘#’, entre outros. Essa medida faz-se necessário pois trata-se de um usuário com permissão total no banco de dados. Em seguida foi feita a alteração

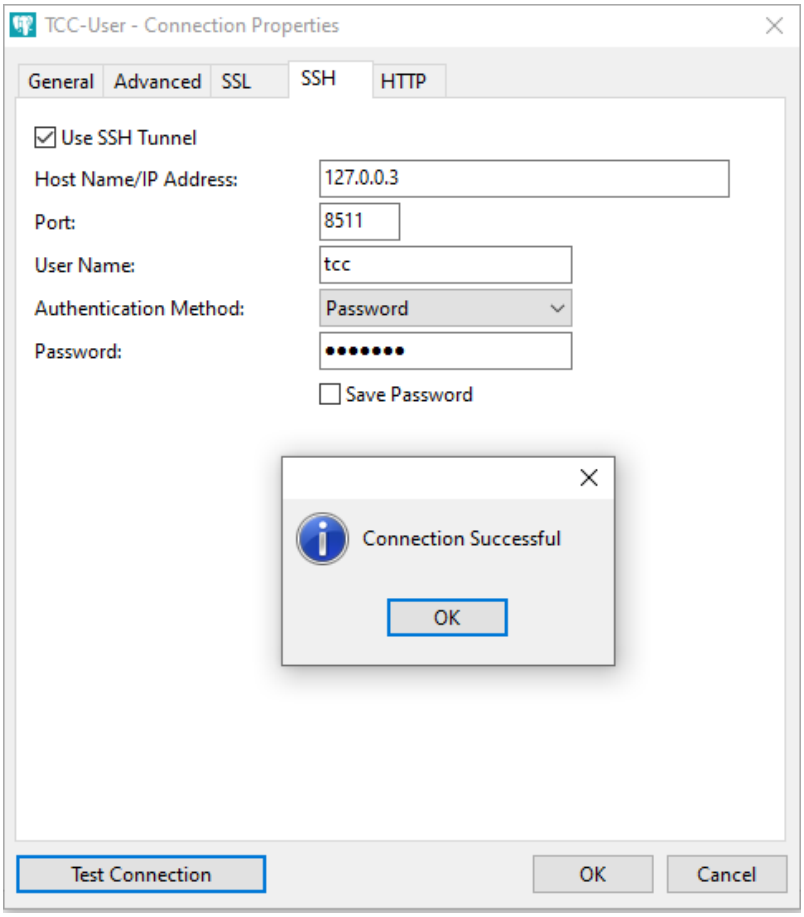
da porta de acesso padrão utilizada pelo SGBD. Essas alterações são necessárias, pois são vulnerabilidades a serem exploradas por um atacante.

Sendo assim, para realizar o acesso foi criado um usuário apenas com permissão de consulta e com uma senha forte. Também foram configurados os acessos via SSH com utilização de chaves pública e privada para cliente e servidor, e configuração de chaves SSL para cliente e servidor no *PostgreSQL*.

Após feitas as configurações, foram explorados dois tipos de acesso, sendo um utilizando autenticação via túnel SSH, e a outro como autenticação SSL. Em ambos os casos deve-se inserir as configurações de acesso ao banco de dados na interface que fará a conexão, como usuário, porta e senha de acesso.

A figura 6 mostra o acesso via túnel SSH fazendo uso do usuário, senha e porta atribuída na configuração deste protocolo.

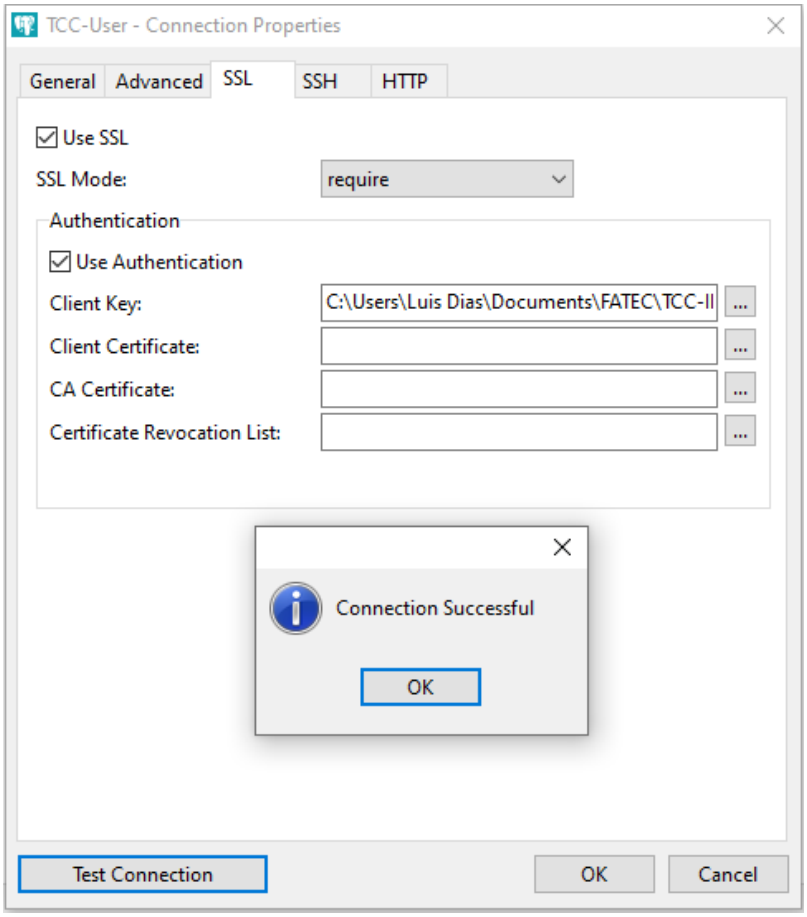
Figura 6 – Acesso ao Banco de Dados via túnel SSH.



Fonte: Autoria Própria.

Já a figura 7 mostra o acesso através do SSL fazendo o uso da chave configurada no cliente e no servidor.

Figura 7 – Acesso ao Banco de Dados por SSL.



Fonte: Autoria Própria.

Ficam demonstradas neste experimento, duas formas seguras de acessar a um banco de dados fazendo uso de uma conexão com criptografia. Essas formas de acesso seguras também devem ser implementadas no acesso ao banco de dados pela aplicação através de um certificado SSL válido.

5.5 TRATAMENTO DE SEGURANÇA NO *DJANGO*

Levando em consideração as funcionalidades de segurança existentes para o *DJANGO* descritas no capítulo 4, esta seção abordará algumas sugestões sobre questões práticas relacionadas a prevenção de ataques já citados em relação ao *framework*.

A OWASP, que é: "[...] uma comunidade aberta dedicada a permitir que as organizações desenvolvam, adquiram e mantenham aplicações e APIs confiáveis." (OWASP, 2017), trata diversas vulnerabilidades relacionadas a aplicações *web*, aos vetores de segurança, e às falhas e os impactos causados.

Para prevenção contra XSS, além da proteção nativa oferecida pelo *DJANGO* a nível de código, a OWASP sugere um tratamento de requisições não confiáveis HTTP, tendo em vista que o *script* será inserido no HTML (OWASP, 2017).

Para CSRF, como já ressaltado no capítulo 4, o *DJANGO* já possui proteção a nível de código implementada no *framework*, porém, ao utilizar uma navegação com HTTPS, essa proteção é potencializada (*DJANGO*, 2019).

Na questão do SQL *Injection*, o *framework* também já faz o tratamento nas requisições feitas ao banco de dados pela aplicação, porém, de acordo com a OWASP, o uso de APIs que tratem a injeção de dados e a validação dos dados que estão entrando na ponta do servidor é um ponto de atenção muito importante a ser tratado (OWASP, 2017).

Mesmo já tendo sido citadas algumas prevenções a nível de código, é necessário manter o código fonte fora do diretório raiz do servidor *web* e estabelecer um limite de *timeout* nas conexões de usuários como já sugerido na norma NGS1 (*DJANGO*, 2019).

Fica aqui algumas sugestões a serem aplicadas junto ao *framework* no processo de desenvolvimento, e que tornará a plataforma mais segurança e confiável, não apenas do ponto de vista computacional, mas também daqueles que farão uso da plataforma.

6 CONSIDERAÇÕES FINAIS

Este trabalho mostrou conceitos de *framework*, como o seu significado, sua importância e utilidade no desenvolvimento de uma aplicação *web*. Foi apresentado especificamente o *DJANGO*, sendo esse um *framework web* para aplicações desenvolvidas em *Python*, o qual foi utilizado na estrutura da plataforma *web* que está hospedada no ambiente estudado.

Foi descrito o ambiente atual do servidor *web* e seu funcionamento, assim como o Sistema Operacional utilizado, e as aplicações nele instalados que são fundamentais para o funcionamento da plataforma *web*, sendo elas o SGBD *PostgreSQL* e o *DJANGO*, e a forma como é feito o acesso ao servidor.

Foram discutidas as questões de segurança relacionadas ao ambiente do servidor *web* e as aplicações que estão instaladas. Dessa maneira, foram propostas algumas medidas de segurança para proporcionar mais proteção ao ambiente como um todo, desde seu acesso até os dados que serão armazenados, dentre os quais estão a configuração de um *firewall*, a utilização de certificação SSL e TLS, diretivas de segurança do *PostgreSQL*, e as funcionalidades de segurança já presentes no *DJANGO*.

Por fim, foram feitas recomendações de desenvolvimento de *software* seguro para área da saúde, baseado nas normas de certificação NGS1 e NGS2 da Sociedade Brasileira de Informática em Saúde (SBIS). Foram feitos os testes das configurações de segurança propostas no trabalho, assim como a apresentação de seu resultado e análise de vulnerabilidade.

É importante destacar que a segurança empregada a um ambiente que irá hospedar um serviço é fundamental e imprescindível, não apenas pelas vulnerabilidades que possam vir a ser exploradas em possíveis ataques, mas também pela integridade e preservação dos dados, e pela credibilidade que será demonstrada ao se ter uma aplicação segura que provem de um local seguro.

Conclui-se dessa forma que, explorar os recursos de segurança, será totalmente benéfico para toda a infraestrutura abordada, e para o andamento do projeto, que trabalha com dados de extrema importância.

7 REFERÊNCIAS BIBLIOGRÁFICAS

4FASTERS, *O guia do iniciante em iptables*, out. 2018. Disponível em: https://4fasters.com.br/2018/10/18/o-guia-completo-do-iniciante-em-iptables/ . Acesso em: 18 jun. 2020.

ALVIM, Paulo. ***Tirando o máximo do java EE 6 open source com jCompany developer suite***. 3. Ed. Belo Horizonte: Powerlogic Publishing, 2008.

AGÊNCIA NACIONAL DE SAÚDE SUPLEMENTAR (ANS), (s.d.). Disponível em: http://www.ans.gov.br/aans/quem-somos . Acesso em: 07 jun. 2020.

AGÊNCIA NACIONAL DE SAÚDE SUPLEMENTAR (ANS), *Resolução Normativa - RN nº 305*, out. 2012. Disponível em: http://www.ans.gov.br/component/legislacao/?view=legislacao&task=TextoLei&f%20ormat=raw&id=MjI2OA== . Acesso em: 07 jun. 2020.

ARAÚJO, Vitor Melo. ***Segurança da informação: uma abordagem holística com foco na implantação de um SGSI***. UNIFACS - Universidade Salvador Laureate, Salvador, 2015. Disponível em: https://books.google.com.br/books?id=KgByDwAAQBAJ&printsec=frontcover&dq=seguran%C3%A7a+da+informa%C3%A7%C3%A3o&hl=pt-BR&sa=X&ved=0ahUKEwjKsZiCy4HmAhUP01kKHTE6B0UQ6AEIRjAE#v=onepage&q=seguran%C3%A7a%20da%20informa%C3%A7%C3%A3o&f=false . Acesso em: 23 nov. 2019.

CERTISIGN, *Existe obrigatoriedade para o Certificado Digital no Prontuário Eletrônico?* set. 2019. Disponível em: https://blog.certisign.com.br/existe-obrigatoriedade-para-o-certificado-digital-no-prontuario-eletronico/ . Acesso em: 10 jun. 2020.

CONFIGR, *O que são frameworks para desenvolvimento WEB. (Conheça os melhores para PHP, Python e Ruby)*, jul. 2019. Disponível em: https://configr.com/blog/frameworks-para-desenvolvimento-web/ . Acesso em: 28 nov. 2019.

DIGITALOCEAN, ***7 Medidas de segurança para proteger seus servidores***, 2018. Disponível em: https://www.digitalocean.com/community/tutorials/7-medidas-de-seguranca-para-proteger-seus-servidores-pt . Acesso em: 23 nov. 2019.

DIGITALOCEAN, ***Como configurar chaves SSH no Ubuntu***, 2018. Disponível em: https://www.digitalocean.com/community/tutorials/como-configurar-chaves-ssh-no-ubuntu-18-04-pt . Acesso em: 23 nov. 2019.

DJANGO, **Documentação do *Django*: s*egurança no Django***, 2019. Disponível em: https://docs.djangoproject.com/pt-br/2.2/topics/security/ . Acesso em: 23 nov. 2019.

HOLOVATY, Adrian; KAPLAN-MOSS, Jacob. ***The definitive guide to django: web development done right***. New York, Springer-Verlag New York, Inc., 2008.

IBARRA, Gustavo Bestetti; WILLEMANN, David Pedro. ***Framework java de apoio ao desenvolvimento de aplicações web com banco de dados, utilizando struts, tiles e hibernate***. Universidade Federal de Santa Catarina, Florianópolis, 2007. Disponível em: https://projetos.inf.ufsc.br/arquivos_projetos/projeto_647/TCC%20-%20Framework%20Web%20Java%20-%20Gustavo%20&%20David%20-%202007-1.pdf . Acesso em: 10 nov. 2019.

INSTITUTO NACIONAL DE TECNOLOGIA DA INFORMAÇÃO (ITI), *ICP-Brasil*, jun. 2017. Disponível em: https://www.iti.gov.br/icp-brasil . Acesso em: 11 jun. 2020.

KAMIDA, Thiago Akira. ***Aplicação web para auxiliar alunos da universidade federal do paraná na escolha de orientador para trabalho de conclusão de curso***. Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2017. Disponível em: http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/7391/1/PG_COCIC_2017_1_10.pdf . Acesso em: 17 nov. 2019.

MORAES, Alexandre Fernandes. ***Firewalls: segurança no controle de acesso***. 1. Ed. São Paulo Érica., 2015.

MOTA, Caio Augusto de Souza; BELUZO, Carlos Eduardo; TRABUCO, Lavinia Pedrosa; SOUZA, Adriano; ALVES, Luciana; CARVALHO, Tiago. ***Desenvolvimento de uma Plataforma Web para Ciência de Dados Aplicada à Saúde Materno Infantil***. nov. 2019. Disponível em: http://ocs.ifsp.edu.br/index.php/conict/xconict/paper/view/5759/1402 . Acesso em: 07 jun. 2020.

MUNIZ, Jéssica Oliveira; HESSEL, Tiago. ***Segurança de Software para a Área de Saúde: Uma avaliação dos requisitos de segurança aplicada em software de Registro Eletrônico em Saúde***. Faculdade de Tecnologia de Americana, Americana, 2019.

OWASP, ***OWASP Top 10 – 2017: The Tem Most Critical Web Application Security Risks***, 2017. Disponível em: https://wiki.owasp.org/images/0/06/OWASP_Top_10-2017-pt_pt.pdf . Acesso em: 18 jun. 2020.

SANTOS, Antonio Henrique dos; CARVALHO, Nelson Ribeiro. *Frameworks e seus benefícios no desenvolvimento de software*. **Revista Pensar Tecnologia**, v.4, n.1, jan. 2015 Disponível em: http://revistapensar.com.br/tecnologia/pasta_upload/artigos/a95.pdf. . Acesso em: 10 nov. 2019.

SOCIEDADE BRASILEIRA DE INFORMÁTICA EM SAÚDE (SBIS), *Certificação de S-RES*. Disponível em: http://www.sbis.org.br/certificacao-sbis . Acesso em: 05 jun. 2020.

SOCIEDADE BRASILEIRA DE INFORMÁTICA EM SAÚDE (SBIS), *Manual de Certificação para Registro Eletrônico em Saúde, Resolução CFM nº 1821/2007*, mar. 2019. Disponível em: http://www.sbis.org.br/certificacao/Manual_Certificacao_SBIS-CFM_2019_v4-3.pdf . Acesso em: 05 jun. 2020.

SOURCEFORGE, Documentação do *PostgreSQL* 8.2.0, 2006. Disponível em: http://pgdocptbr.sourceforge.net/pg82/intro-whatis.html . Acesso em: 23 nov. 2019.

SOURCEFORGE, *Segurança Total em um Banco de Dados PostgreSQL*, 2009. Disponível em: https://www.ibm.com/developerworks/br/opensource/library/os-postgresecurity/index.html . Acesso em: 23 nov. 2019.

SSH, 2019. Disponível em: https://www.ssh.com/ssh/?utm_source=s&utm_medium=nav&utm_campaign=head Acesso em: 23 nov. 2019.

TABLELESS, *O que é MVC?* 2015. Disponível em: https://tableless.com.br/mvc-afinal-e-o-que/ . Acesso em: 17 nov. 2019.

UBUNTU SERVER, 2019. Disponível em: https://ubuntu.com/server . Acesso em: 23 nov. 2019.

8 ANEXO

TERMO DE AUTORIZAÇÃO DE USO

Paulínia, 18 de Junho de 2020.

Eu **Carlos Eduardo Beluzo** professor EBTT do quadro ativo do IFSP Câmpus de Campinas, Pesquisador vice-líder do Grupo de Pesquisa **PICAp – Pesquisa e Inovação em Computação Aplicada** – devidamente cadastrado no Diretório de Pesquisas do CNPQ, autorizo **Luís Antonio Queiroz Dias,** o aluno do curso superior em Segurança da Informação da Faculdade de Tecnologia de Americana, a utilizar o código fonte do projeto SaMI, disponível em https://github.com/plataformasami/www para atividades de avaliação e análise de seu trabalho de conclusão de curso sob o título de "**Um Ambiente Seguro para Aplicação com Framework Web Django**". O aluno compromete-se a não repassar e não divulgar o material aqui cedido.

Carlos Eduardo Beluzo
beluzo@ifsp.edu.br