UNIVERSIDADE PAULISTA - UNIP EaD

Projeto Integrado Multidisciplinar II

Curso Superior de Tecnologia em

Análise e Desenvolvimento de Sistemas

CASSIMIRO LIMA DE ARAUJO – RA 2525460

GEOVANNA FÉLIX MONTEIRO – RA 2521699

GUSTAVO RODRIGUES ROLIM – RA 2522555

LUAN LOPES BEZERRA – RA 2503460

SOFIA DE CASTRO ONGARO – RA 2528018

VITÓRIA FREITAS – RA 2523279

PLATAFORMA DE GESTÃO DE DADOS PESSOAIS

ALPHAVILLE

2025

CASSIMIRO LIMA DE ARAUJO – RA 2525460

GEOVANNA FÉLIX MONTEIRO – RA 2521699

GUSTAVO RODRIGUES ROLIM – RA 2522555

LUAN LOPES BEZERRA – RA 2503460

SOFIA DE CASTRO ONGARO – RA 2528018

VITÓRIA FREITAS – RA 2523279

PLATAFORMA DE GESTÃO DE DADOS PESSOAIS

Projeto Integrado Multidisciplinar em Análise e Desenvolvimento de Sistemas

Projeto Integrado Multidisciplinar para obtenção do título de tecnólogo em Análise e Desenvolvimento de Sistemas, apresentado à Universidade Paulista – UNIP EaD.

Orientador (a): Me. Tarcísio Peres

ALPHAVILLE

2025

RESUMO

O presente trabalho tem como objetivo criar um projeto para uma plataforma de gestão de dados pessoais. Para concluir as metas do projeto, que é o cumprimento da Lei Geral de Proteção de Dados, a segurança dos dados, uma boa infraestrutura computacional e um código bem estruturado. Objetivamos ter transparência, responsabilidade e o fortalecimento dos direitos dos titulares de dados na nossa plataforma, tudo isso para garantir a segurança dos dados dos usuários. Ademais, a equipe planejou o investimento necessário em cibersegurança, utilizando firewall de aplicações (WAF) e criptografia de dados em trânsito e em repouso e um controle de acesso granular. Outrossim projetamos, uma infraestrutura computacional muito bem arquitetada, para isso utilizamos uma arquitetura de servidor web, um sistema de armazenamento mySQL, uma infraestrutura de Nuvem Pública (*Cloud Computing*) e balanceamento de carga, que usaremos *Elastic Load Balancing,* que é um serviço distribuído pelo AWS.

Palavras-chave: plataforma de gestão de dados pessoais; LGPD; segurança dos dados; cibersegurança; infraestrutura computacional.

ABSTRACT

This work aims to create a project for a personal data management platform. To achieve the project goals, which are compliance with the General Data Protection Law, data security, a good computing infrastructure and well-structured code. We aim to have transparency, accountability and the strengthening of data subjects' rights on our platform, all to ensure the security of user data. In addition, the team planned the necessary investment in cybersecurity, using application firewall (WAF) and data encryption in transit and at rest and granular access control. We also designed a very well-architected computing infrastructure, for this we use a web server architecture, a mySQL storage system, a Public Cloud infrastructure (Cloud Computing) and load balancing, which we will use Elastic Load Balancing, which is a distributed service by AWS.

Keywords: personal data management platform; LGPD; data security; cybersecurity; computing infrastructures;

SUMÁRIO

1 INTRODUÇÃO REVER PARÁGRAFOS

Em um desenvolvimento de uma plataforma de gestão de dados pessoais é necessário seguir diversos objetivos, pois quando se trata desta plataforma o programador estará lidando com dados pessoais (conteúdos sensíveis), por isso necessita-se de cuidado ao desenvolvê-la. O trabalho presente, abordará o planejamento da plataforma de gestão de dados pessoais, desenvolvida pela equipe. Segmentada pela Lei Geral de Proteção de Dados. As tecnologias utilizadas nos segmentos de cibersegurança, e na infraestrutura computacional. Logo após, será exposto o código executado na linguagem Python da plataforma. Além disso, a Lei Geral de Proteção de Dados é uma etapa crucial, para desenvolver um site seguro e íntegro. Esta lei possui garantias aos titulares, uma forma de segui-la é a permissão do acesso aos dados para que se possa deletá-los e transferi-los, caso necessário. Projetamos esta plataforma para seguir a LGPD, assim teremos responsabilidade em relação a utilização de dados pessoais, que é um conteúdo sensível. Ademais, para a criptografia, foi escolhido a utilização de criptografia de dados em trânsito, e em repouso, fazendo com que os dados não possam ser lidos por pessoas não autorizadas, reforçando a segurança. Elas trafegam entre o navegador do usuário e o servidor, assim impedindo interceptações e protegem dados armazenados. No projeto foi adicionado, uma autenticação forte para reduzir o risco de invasões e firewall de aplicações (WAF), que atua na proteção da camada de aplicação, onde os usuários interagem. Teremos um monitoramento de intrusões, para isso teremos: IDS, IPS. E por fim testes de penetração que são simulações de invasões feitas por profissionais da área, para encontrar defeitos que poderiam passar despercebidos. A infraestrutura computacional da plataforma terá um servidor web, um sistema que armazena, processa e entrega conteúdos de sites por meio da internet, fazendo requisições HTTP ao servidor correspondente. Vamos usar MySQL como sistema de armazenamento, que se baseia em arquitetura modular. Aplicamos o balanceamento de carga que faz a distribuição do tráfego de rede, oferecendo suporte a aplicação, para isso escolhemos Elastic Load Balancing (ELB) da AWS, ele vai distribuir o tráfego para diversos destinos, fazendo de maneira confiável e rápida. O código Python foi pensado e projetado para proteger os dados e nos certificamos de que os direitos dos usuários sejam atendidos, eles podem acessar, editar e excluir seus dados. Que foram armazenados utilizando JSON, fizemos teste unitários, para isso usamos o unittest que pode ser importado de uma biblioteca Python. Desse modo vamos assegurar que a plataforma de gestão de dados pessoais tenha uma estrutura que garanta a segurança dos dados dos usuários, assim se tornando um software confiável para o uso deles, isso algo indispensável nos dias atuais e

para empresas que não seguem esses objetivos de segurança podem ter consequências severas, como multas e perda dinheiro e investimentos.

2 ÉTICA, LEI GERAL E PROTEÇÃO DE DADOS

A promulgação da Lei Geral de Proteção de Dados Pessoais (LGPD), Lei nº 13.709/2018 e com vigência em agosto de 2020, contribui com princípios essenciais, como transparência, proteção de forma igualitária, responsabilidade, segurança, fortalecimento dos direitos dos titulares de dados e respeito à privacidade. Além disso, a lei possui diversas garantias aos cidadãos, que podem revogar um consentimento, solicitar que dados sejam deletados, transferir dados para outro fornecedor de serviços e entre outras atribuições. Para melhor entendimento e exemplificação, de como a Lei Geral de Proteção de dados, abaixo se encontra um infográfico criado pela Serpro.

Fonte: SERPO

2.1 GESTÃO DE DADOS PESSOAIS

A plataforma criada pela equipe, funciona como um sistema de cadastro de pessoas, que, conforme analisado nos termos citados, faz-se de suma importância que ela esteja alinhada com a lei geral de proteção de dados. O que se faz presente em diversas etapas do cadastro do usuário.

2.2 CADASTRO

Quando o usuário tomar a decisão de criar uma conta no sistema, alguns dados essenciais serão pedidos, como: nome completo, e-mail, Registro Geral (RG), Cadastro de Pessoa Física (CPF), nacionalidade e data de nascimento. Essas informações são importantes para que seja possível verificar a identidade, evitar fraudes, garantir apenas um cadastro por pessoa, e para certificar que o usuário é maior de idade. Dados como RG e CPF foram incluídos no projeto a fim de simular uma verificação de identidade mais robusta, com as devidas medidas de proteção tendo em vista que são dados sensíveis, e com o consentimento do usuário. Ademais, para a construção deste sistema seguro, os dados são armazenados localmente em arquivos JSON, que reduzem os riscos de exposições on-line.

Na etapa do cadastro, o usuário deve criar uma senha forte, o que reforça a segurança dos dados, sendo a senha composta por uma letra maiúscula, uma minúscula e um número. Logo após a criação da senha, deve-se confirmá-la. Em cumprimento à base legal do consentimento, presente no Art. 9º - “O tratamento de dados pessoais somente pode ocorrer após o consentimento livre, expresso e informado do titular.” desse modo, ao longo do sistema pedimos permissão explícita, como por exemplo: "Você autoriza o uso dos seus dados conforme nossa política de privacidade?" Do mesmo modo deixamos de forma clara as políticas de privacidade e as opções de alteração ou exclusão dos dados.

3 SEGURANÇA DA INFORMAÇÃO EM UMA PLATAFORMA DE GESTÃO DE DADOS PESSOAIS

Sabendo da importância da cibersegurança, foram escolhidos métodos fundamentais para garantir a segurança da plataforma. A proteção de dados pessoais é uma prioridade, e para garantir que as informações dos usuários estejam seguras. Com essas medidas, buscamos construir uma plataforma confiável, segura e preparada para os desafios do ambiente digital atual. Com o aumento no volume de dados digitais e o fortalecimento de legislações voltadas à privacidade, como a LGPD, plataformas que armazenam e processam dados pessoais devem adotar medidas eficazes de segurança. A proteção dessas informações é um compromisso com a integridade, confidencialidade e disponibilidade dos dados dos usuários. Como enfatiza Andress (2016, p. 23), “a segurança da informação não é mais uma opção, e sim uma necessidade em qualquer sistema que lide com dados sensíveis” Nesse contexto, alguns mecanismos se destacam como pilares essenciais para garantir um ambiente seguro.

3.1 CRIPTOGRAFIA DE DADOS EM TRÂNSITO E EM REPOUSO

A criptografia é uma das ferramentas mais fundamentais na segurança da informação. Sua principal função é transformar os dados em códigos que não podem ser lidos por terceiros não autorizados. Quando aplicada aos dados em trânsito, como os que trafegam entre o navegador do usuário e o servidor, ela impede que interceptações durante a comunicação revelem informações sensíveis. Já a criptografia em repouso protege dados armazenados, seja em bancos de dados, arquivos ou dispositivos, garantindo que mesmo em caso de acesso físico indevido ou vazamento, as informações continuam inacessíveis sem as chaves de decriptação. Essa proteção é crucial para a plataforma que lida com cadastros, documentos pessoais e registros financeiros.

3.2 AUTENTICAÇÃO FORTE

A autenticação forte é outro componente indispensável para o controle de acesso seguro. Vai além do simples uso de login e senha, incorporando múltiplos fatores de verificação, como tokens, biometria ou códigos enviados por aplicativos autenticadores. Esse tipo de autenticação reduz significativamente o risco de invasão, mesmo que um dos elementos de acesso tenha sido comprometido. Em ambientes que lidam com dados sensíveis, esse nível de proteção é essencial para garantir que apenas usuários devidamente autorizados tenham acesso às informações.

3.3 CONTROLE DE ACESSO GRANULAR

A implementação de um controle de acesso granular permite que a plataforma defina com precisão quem pode acessar o que está dentro do sistema. Essa abordagem se baseia no princípio do menor privilégio, ou seja, cada usuário recebe apenas as permissões necessárias para desempenhar sua função. Além disso, o acesso pode ser restrito por tempo, local ou tipo de dispositivo. Isso evita, por exemplo, que um colaborador da área administrativa tenha acesso indevido a dados financeiros ou médicos. Esse controle refinado contribui para a segurança geral da plataforma e facilita a rastreabilidade de ações em caso de incidentes.

3.4 FIREWALL DE APLICAÇÕES (WAF)

O firewall de aplicações web, conhecido como WAF (*Web Application Firewall*), atua na proteção da camada de aplicação da plataforma, justamente onde os usuários interagem com os sistemas. Ele analisa o tráfego de entrada e saída, identificando e bloqueando padrões maliciosos, como ataques de injeção de SQL, *cross* site *scripting* (XSS) e acesso forçado a URLs. Além disso, o WAF pode ser configurado para responder automaticamente a ameaças, isolando sessões suspeitas e gerando alertas para a equipe de segurança. Em sistemas que

estão expostos à internet, essa barreira é essencial para evitar que vulnerabilidades sejam exploradas por atacantes.

3.5 MONITORAMENTO DE INTRUSÕES

O monitoramento contínuo do ambiente digital é indispensável para detectar comportamentos anômalos e tentativas de invasão. Sistemas de Detecção de Intrusão (IDS) e de Prevenção de Intrusão (IPS) funcionam como "sensores" que identificam atividades suspeitas e disparam alertas em tempo real. Essas soluções ajudam a responder rapidamente a ameaças, evitando que uma invasão se torne um incidente grave. O monitoramento também permite registrar eventos relevantes para auditorias futuras e reforçar a análise de vulnerabilidade. Em uma plataforma que lida com dados pessoais, a capacidade de identificar e agir diante de riscos iminentes é um fator determinante para a confiança do usuário.

3.6 TESTES DE PENETRAÇÃO (PENTESTS)

Os testes de penetração, ou *pentests*, são como simulações de invasões feitas por especialistas que tentam encontrar falhas de segurança antes que criminosos reais o façam. Estes testes ajudam a descobrir pontos fracos do sistema que passariam despercebidos no dia a dia. Com base nos resultados, os desenvolvedores e a equipe de segurança podem corrigir essas falhas e reforçar a proteção da plataforma. É uma forma preventiva e muito eficaz de manter os dados sempre seguros.

4 ATIVIDADE DE EXTENSÃO UNIVERSITÁRIA

(créditos: autoria própria)

5 INFRAESTRUTURA COMPUTACIONAL E ARQUITETURA DE SERVIDORES

Para arquitetura de servidores usamos um servidor web, que é um sistema composto por hardware e software que armazena, processa e entrega conteúdos de sites aos usuários por meio da internet. Ele é responsável por receber solicitações de navegadores (como Chrome ou Firefox) e responder com os arquivos necessários para exibir páginas web, como documentos HTML, imagens, vídeos e scripts. O funcionamento de um servidor web segue uma arquitetura cliente-servidor, primeiro tem a solicitação: O usuário digita uma URL no navegador, que envia uma requisição HTTP ao servidor correspondente. Segundo o processamento: O servidor web recebe a requisição, processa-a e determina qual conteúdo deve ser enviado de volta. Terceiro a resposta: O servidor envia os arquivos solicitados (como páginas HTML, imagens, etc.) de volta ao navegador, que os renderiza para o usuário.

Componentes de um Servidor Web são o hardware: É o equipamento físico, um computador ou servidor dedicado que armazena os arquivos do site e mantém uma conexão constante com a internet para atender às solicitações dos usuários e o software: inclui o servidor HTTP, que interpreta e responde às requisições feitas pelos navegadores. Exemplos populares de software de servidor web são o Apache HTTP Server: amplamente utilizado e de código aberto, Nginx: Conhecido por sua eficiência em lidar com múltiplas conexões simultâneas e o Microsoft IIS (Internet Information Services): Solução da Microsoft para servidores baseados em Windows. Tipos de Conteúdo Servidos, o estático: arquivos que não mudam, como imagens, folhas de estilo CSS e páginas HTML fixas. E o Dinâmico: gerado em tempo real, geralmente por scripts ou aplicações que interagem com bancos de dados, como páginas PHP ou ASP.NET. Para segurança e protocolos poderíamos usar dois métodos, o HTTP ou HTTPS, vamos falar da diferença de cada um. O HTTPS utiliza criptografia TLS/SSL para fazer a proteção dos dados transmitidos entre o servidor e o cliente. Porta comum do HTTP é a porta 80, do HTTPS é a 443.

Por isso chegamos à conclusão que o HTTPS seria uma opção melhor, por considerarmos ele superior ao HTTP. Além de servir páginas web, um servidor web pode gerenciar e-mails, transferir arquivos e hospedar bancos de dados servindo como *backend* para aplicações web dinâmicas. Um servidor web atua como um intermediário, ele vai entregar o conteúdo dos sites aos usuários de forma eficiente e segura. Por tanto acreditamos ser fundamental para o funcionamento do nosso projeto, que está projetado para estar disponível na internet.

5.1 SISTEMA DE ARMAZENAMENTO

Para o sistema de armazenamento usamos o MySQL que é baseado em uma arquitetura modular que permite o uso de diferentes mecanismos de armazenamento (*storage engines*), cada um com características específicas para atender a diversas necessidades de aplicações. O mecanismo de armazenamento padrão desde a versão 5.5 é o InnoDB, substituindo o MyISAM, que era o padrão anteriormente.

5.2 ARQUITETURA DO SISTEMA DE ARMAZENAMENTO MYSQL

O MySQL utiliza uma arquitetura de mecanismos de armazenamento plugáveis, permitindo que diferentes *engines* sejam integradas ao servidor de banco de dados. Essa arquitetura oferece flexibilidade para escolher o mecanismo mais adequado para cada tipo de aplicação, como processamento de transações, data *warehousing* ou alta disponibilidade. Os mecanismos de armazenamento são responsáveis por realizar as operações físicas nos dados, enquanto o servidor MySQL fornece uma interface consistente para os desenvolvedores e administradores de banco de dados.

5.3 INNDB: O MECANISMO DE ARMAZENAMENTO PADRÃO

O InnoDB é o mecanismo de armazenamento padrão do MySQL desde a versão 5.5, sendo reconhecido por sua robustez e flexibilidade. Ele oferece suporte completo a transações compatíveis com ACID, garantindo um alto nível de integridade dos dados. Além disso, suporta bloqueio em nível de linha, o que reduz significativamente a contenção e melhora a concorrência. O InnoDB também oferece suporte a chaves estrangeiras, assegurando a integridade referencial entre tabelas. Antes do InnoDB, o MyISAM era o mecanismo de armazenamento padrão do MySQL. Embora o MyISAM seja mais simples e oferece desempenho superior em operações de leitura, ele não suporta transações, chaves estrangeiras

ou bloqueio em nível de linha, o que limita sua aplicabilidade em sistemas que requerem alta integridade e concorrência. Além disso, o MyISAM é mais suscetível à corrupção de dados em caso de falhas do sistema.

5.4 O DESEMPENHO E QUALIDADE (REVER PARÁGRAFOS)

O InnoDB é projetado para oferecer alto desempenho e confiabilidade. O uso do buffer pool permite que dados frequentemente acessados sejam lidos diretamente da memória, reduzindo significativamente o tempo de resposta das consultas. Além disso, o suporte a bloqueio em nível de linha melhora a concorrência, permitindo que múltiplas transações sejam processadas simultaneamente com menor contenção. O InnoDB garante a integridade dos dados por meio do suporte completo a transações ACID, chaves estrangeiras e mecanismos de recuperação em caso de falhas. Essas características tornam o InnoDB adequado para aplicações críticas que exigem alta confiabilidade e consistência dos dados. O sistema de armazenamento do MySQL, com o mecanismo InnoDB em evidência, porque ele oferece desempenho, confiabilidade e flexibilidade. Sua arquitetura modular permite adaptar o banco de dados às necessidades específicas de diferentes aplicações, tornando o MySQL uma escolha sólida para uma ampla gama de cenários, desde aplicações web até sistemas corporativos de missão crítica. Com o uso dessas tecnologias citadas vamos possuir uma maior qualidade e desempenho, logo vamos melhorar a experiência do usuário em relação ao nosso software.

5.5 REDE

Aplicamos na plataforma uma infraestrutura de Nuvem Pública (*Cloud Computing*). Entre suas vantagens está a escalabilidade e flexibilidade, que permite que empresas ajustem recursos conforme a demanda, sem necessidade de grandes investimentos iniciais em infraestrutura, o acesso remoto, onde os usuários vão possuir possibilidade de acessar dados e ferramentas se tiverem conexão à internet, não importando o lugar onde estejam. Atualizações

automáticas, nos quais os provedores de nuvem frequentemente atualizam sistemas e aplicam patches de segurança automaticamente.

5.6 PLANO DE REDUNDÂNCIA

O plano de redundância que empregamos foi ter de ter mais de um servidor web e mais de um banco de dados funcionando ao mesmo tempo com pelo menos dois de cada, com sincronização contínua entre eles. Os componentes mais importantes devem estar em locais diferentes (zonas diferentes), assim evitando que um problema em um único lugar para todo o sistema. Os servidores Web devem ser configurados de forma que não dependam de informações salvas neles (infraestrutura "*stateless*"). E os dados da aplicação e as sessões dos usuários devem ser guardados em um banco central, como o Redis, para que seja possível trocar servidores sem afetar os usuários.

5.7 PLANO DE BACKUP

Para plano de backup teríamos mais frequência dos backups, como backups diários com apenas as mudanças e backups completos uma vez por semana. Guardar eles por no mínimo 30 dias e fazer cópias mensais (*snapshots*) armazenar por 6 meses a 1 ano. Algumas cópias seriam guardadas na mesma nuvem onde está o sistema e outra cópia em um local diferente, como outra região ou outro provedor (Azure, Google Cloud, S3 Glacier). O que deve ser incluído no backup: Banco de dados MySQL (cópia completa e imagens do volume), Arquivos de configuração dos servidores, Registros do sistema e da aplicação (logs), Arquivos dos usuários e envios (uploads). Além disso tudo fazer testes e verificações, testando se os backups podem ser restaurados pelo menos uma vez por mês e verificar automaticamente se os backups estão corretos logo após serem feitos.

5.8 MONITORAMENTO E SEGURANÇA

Monitoramento com alertas usando ferramentas como *Prometheus*, *Grafana*, *CloudWatch* e *Datadog*. Acompanhar o uso de CPU, memória, espaço em disco, número de conexões, atrasos e erros no sistema. Na parte da segurança o que temos em vista seria proteger os dados guardados usando criptografia (por exemplo, MySQL com TDE ou discos criptografados). Assim farão a proteção dos dados no momento da transmissão com HTTPS e TLS. E o uso de firewalls e tendo regras para segurança fixas e ter o controle de quem pode e quem não pode ter acesso ao sistema como por exemplo gestão de identidade e acesso. Com consequência disso tudo teremos menos chances de falhas no sistema, e se tiver falhas vamos minimizar os dados.

6 BALANCEAMENTO DE CARGA E PLATAFORMA EM NUVEM

Para nossa plataforma de gestão de dados, escolhemos usar o balanceamento de carga. Utilizaremos o serviço de *Elastic Load Balancing* (ELB) oferecido pelo AWS, pois apresenta diversos benefícios, vamos explicar quais são eles e o que seria um balanceamento de carga.

6.1 BALANCEAMENTO DE CARGA

Possui como objetivo fazer a distribuição do tráfego de rede para oferecer suporte a uma aplicação. Como aplicações atuais processam milhões de usuários ao mesmo tempo e retornam diversos tipos de dados, isso precisa ser feito de maneira rápida e confiável, para isso serve o balanceamento de carga. Ele fica entre os usuários e os servidores, atuando como um facilitador invisível.

6.2 BENEFÍCIOS

Os benefícios seriam: aumentando a tolerância a falhas, detectam problemas do servidor e redireciona o tráfego do cliente para um servidor que esteja disponível. Eles evitam gargalos de tráfego dos servidores, você pode adicionar ou remover servidores, caso precisar. Além de auxiliar na segurança do aplicativo, monitorando o tráfego e bloqueando conteúdo quando necessário, podem redirecionar o tráfego de ataque para servidores backend, assim diminuindo o impacto e roteando com firewalls de rede.

6.3 AUXÍLIO DO AWS NO BALANCEAMENTO DE CARGA

Elastic Load Balancing (ELB) é um serviço de balanceamento de carga, que vai distribuir o tráfego de aplicações para diversos destinos e dispositivos virtuais na AWS. Com ele você dimensiona aplicações sem precisar de *gateways* de API ou uma configuração muito

complexa. Utiliza o ELB para configurar 4 tipos de balanceadores de carga de software: *Application Load Balancer* para rotear solicitações em HTTP, *Network Load Balancer* para rotear solicitações em endereços IP, para dispositivos virtuais de terceiros se utiliza o balanceador de carga e para aplicações na rede clássica *Amazon* EC2 seria *o Classic Load Balancer*.

7 CÓDIGO PYTHON

```
#Importamos a biblioteca json para salvar dados usando json
import json
#Importamos a biblioteca re para procurar, validar ou substituir texto.
import re
#Importamos a biblioteca unicodedata para trabalhar com texto com
caracteres especiais
import unicodedata
#Para verificarmos a idade do usuário
from datetime import datetime
anoAtual = datetime.now().year
arquivos = "usuariosDados.json"
# Classe para fazer as verificações: rg, cpf, senha, estado civil e a
idade
class Verificacoes:
    # Função para verificar o RG
    def verificarRg(self, rg):
        #aqui vamos verificar o comprimento do rg, a quantidade de
digitos de um rg é de 8 a 11, aqui contei junto com os "." e "-".
        if len(rg) >=11  and len(rg) <= 14:
            return True
        else:
            return False
    # Função para verificar o Cpf
    def verificarCpf(self, cpf):
        cpfMinusculo = cpf.lower() #  Se tiver letra todas vão estar em
minusculo para facilitar a verificação
        possuiLetra = re.search(r'[a-z]', cpfMinusculo) # Utilizamos a
biblioteca re, para verificar se possui letra no input
        if not possuiLetra and len(cpf) == 14:
            return True
        else:
            return False
    # Função para verificar a senha
    def verificarSenha(self, senha):
        # Aqui verificamos se tem número
        temNumero = re.search(r'\d', senha)
        # Aqui verificamos se tem letra maiúscula
        temLetraMaiuscula = re.search(r'[A-Z]', senha)
        # Aqui verificamos se tem letra minúscula
        temLetraMinuscula = re.search(r'[a-z]', senha)
        # Nesse if se as três variáveis forem verdadeiras, retornará
```

```
como verdade, se não como falsa.
        if temNumero and temLetraMinuscula and temLetraMaiuscula:
            return True
        else:
            return False
    # Função para verificar o estado civil
    def verificadorDeEstadoCivil(self, estadoCivil):
        # Deixamos tudo minúsculo
        estadoCivilMinusculo = estadoCivil.lower()
        # Essas duas próximas variáveis servem para deixar o texto sem
acentos.
        string = unicodedata.normalize('NFD', estadoCivilMinusculo)
        stringSemAcento = re.sub(r'[\u0300-\u036f]', '', string)
        # Agora faremos a verificação:
        if stringSemAcento == "solteiro" or stringSemAcento == "casado"
or stringSemAcento == "divorciado" or stringSemAcento == "viuvo" or
stringSemAcento == "separado judicialmente":
            return True
        else:
            return False
    # Função para verificar se a pessoa é de maior e se colocou um ano
válido
    def verificarIdade(self, anoDeNascimento):
        verificar = anoAtual - anoDeNascimento
        if verificar >= 18:
            return True
        else:
            return False
# testes unitários:
import unittest
class TesteDeFuncoes(unittest.TestCase):
    def setUp(self):
        self.teste = Verificacoes()
    def testeRg(self):
        self.assertTrue(self.teste.verificarRg("22.222.222-2"))
        self.assertTrue(self.teste.verificarRg("1.234.567-ES"))
        self.assertFalse(self.teste.verificarRg("333"))
    def testeCpf(self):
        self.assertTrue(self.teste.verificarCpf("423.956.779-09"))
        self.assertFalse(self.teste.verificarCpf("blaafsaf"))
        self.assertFalse(self.teste.verificarCpf("333"))
    def testeSenha(self):
        self.assertTrue(self.teste.verificarSenha("12Ab"))
```

```
        self.assertFalse(self.teste.verificarSenha("blaafsaf"))
        self.assertFalse(self.teste.verificarSenha("1243"))
    def testeEstadoCivil(self):

self.assertTrue(self.teste.verificadorDeEstadoCivil("solteiro"))

self.assertFalse(self.teste.verificadorDeEstadoCivil("blaafsaf"))
        self.assertTrue(self.teste.verificadorDeEstadoCivil("CASADO"))
    def testeIdade(self):
        self.assertFalse(self.teste.verificarIdade(anoAtual - 17))
        self.assertTrue(self.teste.verificarIdade(anoAtual - 20))
        self.assertFalse(self.teste.verificarIdade(anoAtual + 1))
#Para fazer os teste digite no terminal: python -m unittest
GestaoDeDadosPessoais.py -v
# Funções para pegar, buscar, salvar, atualizar, editar e excluir os
dados
#Pegar
class AlteracoesNosDados:
    def pegar_dados(self):
        #utilizamos try e expept caso ocorra erros
        try:
            with open(arquivos, "r", encoding="utf-8") as f:
                # json.load pega arquivos em json e converte para
python
                dados = json.load(f)
                # Para fazer essa verificação se o arquivo não esta
sendo como uma lista, utilizamos a função isinstance, que já é do
python.
                if not isinstance(dados, list):
                    return []
                return dados
            # usamos expect para identificar o erro caso ele ocorra.
        except FileNotFoundError:
            # Caso não tenha um arquivo, vamos criar um
            with open(arquivos, "w", encoding="utf-8") as f:
                #Aqui cria uma lista vazia
                json.dump([], f)
            return []
        except Exception as e:
            print(f"Erro ao ler arquivo: {e}")
            return []
    # Função para verificar se o usuário existe
    def buscar_dados(self, dados, email, nome):
```

```
        dados = self.pegar_dados()
        for usuario in dados:
            if usuario.get("email") == email and usuario.get("nome") ==
nome:
                return usuario
        return None
    # Salvar
    def salvar_dados(self, usuario):
        dados = self.pegar_dados()
        # coloca o usuário na variável de dados.
        dados.append(usuario)
        # salva na variável arquivos
        with open(arquivos, "w", encoding="utf-8") as f:
            # json.dump pega arquivos em python e converte em json,
usamos ensure_ascii=False para os dados serem salvos como estão,
indent=4 formata o json com indentação de 4 espaços
            json.dump(dados, f, ensure_ascii=False, indent=4)
    # Atualizar
    def atualizar_dados(self, antigoEmail, novoEmail,
novaNacionalidade, novoEstadoCivil, nomeDoUsu):
        #pegamos  os dados
        dados = self.pegar_dados()
        #Agora vamos atualizar os dados
        for i, user in enumerate(dados):
            if user.get("email") == antigoEmail and user.get("nome") ==
nomeDoUsu:
                dados[i].update({
                "email": novoEmail,
                "nacionalidade": novaNacionalidade,
                "estado civil": novoEstadoCivil
                })
                print("Dados editados com sucesso.")
                break
        #Vamos salvar eles no arquivo json
        with open(arquivos, "w", encoding="utf-8") as f:
            json.dump(dados, f, ensure_ascii=False, indent=4)
    # Editar
    def editar_dados(self, email, nome):
        #Pegamos os dados que vão ser editados
        antigoEmail = email
        nomeDoUsu = nome
        novoEmail = input("Novo e-mail: ")
        novaNacionalidade = input("Nova nacionalidade: ")
```

```
        novoEstadoCivil = input("Novo estado civil (solteiro, casado,
divorciado, viúvo ou separado judicialmente): ")
        verificador = Verificacoes()
        # Verificação do novo estado civil
        if not verificador.verificadorDeEstadoCivil(novoEstadoCivil):
            print("Estado civil incorreto.")
            return
        #Vamos atualizar os dados, usando a função atualizar
        self.atualizar_dados(antigoEmail, novoEmail, novaNacionalidade,
novoEstadoCivil, nomeDoUsu)
    def excluir_dados(self, email, nome):
        dados = self.pegar_dados()
        for i, user in enumerate(dados):
            if user.get("email") == email and user.get("nome") == nome:
                del dados[i]
                print("Dados excluídos com sucesso.")
                break
        with open(arquivos, "w", encoding="utf-8") as f:
            json.dump(dados, f, ensure_ascii=False, indent=4)
# Função da entrada de dados
def entrar():
    print("Login: ")
    nome = input("Nome de usuário: ")
    email = input("Email: ")
    senha = input("Senha: ")
    alteracoes = AlteracoesNosDados()
    dados = alteracoes.pegar_dados()
     #  para evitar erros, aqui verificamos se os dados são uma lista,
se forem vai causar erros no json.
    if not isinstance(dados, list):
        print("Erro: Dados corrompidos. Por favor, recadastre-se.")
        return
    usuario = alteracoes.buscar_dados(dados, email, nome)
    # mensagem caso o usuário não seja encontrado
    if usuario is None:
        print("Usuário não encontrado.")
        return
    # para evitar erros, aqui verificamos se o usuário esta salvo como
um dicionário
    if not isinstance(usuario, dict):
        print("Erro: Dados do usuário corrompidos.")
        return
    if nome == usuario["nome"] and senha == usuario["senha"]:
```

```
        print("Dados do usuário:")
        print(f"Nome: {usuario['nome']}")
        print(f"Email: {usuario['email']}")
        print(f"Rg: {usuario['rg']}")
        print(f"Cpf: {usuario['cpf']}")
        print(f"Nacionalidade: {usuario['nacionalidade']}")
        print(f"Estado civil: {usuario['estado civil']}")
        res = input("Você deseja editar os seus dados (s/n): ").lower()
        if res == "s"  or res == "S":
            alteracoes.editar_dados(email, nome)
        resExcluir = input("Você deseja excluir os seus dados (s/n):
").lower()
        if resExcluir == "s" or resExcluir == "S":
            alteracoes.excluir_dados(email, nome)
    else:
        print("Dados não encontrados.")
# Função do cadastro de dados
def cadastrar():
    print("Cadastro: ")
    print("-------------------------------------")
    nome = input("Nome de usuário: ")
    email = input("Email: ")
    senha = input("Senha (precisa ter uma letra maiúscula, uma letra
minúscula e um número pelo menos): ")
    confirmaSenha = input("Confirmar senha: ")
    rg = input("Rg (exemplo: 22.222.222-2): ")
    cpf = input("Cpf: (exemplo: 222.222.222-22): ")
    nacionalidade = input("Nacionalidade: ")
    estadoCivil = input("Estado civil (solteiro, casado, divorciado,
viúvo ou separado judicialmente): ")
    anoDeNascimento = int(input("Seu ano de nascimento: "))
    alteracoes = AlteracoesNosDados()
    verificador = Verificacoes()
    dados = alteracoes.pegar_dados()
    for usuario in dados:
        if usuario.get("nome") == nome:
            print("Nome de usuário já existe.")
            return
    # Verificação de senha, cpf e rg
    if not verificador.verificarSenha(senha):
        print("senha não é forte o suficiente")
        return
    if not verificador.verificarCpf(cpf) :
```

```
        print("CPF escrito errado")
        return
    if not verificador.verificarRg(rg) :
        print("RG escrito errado")
        return
    if not verificador.verificarIdade(anoDeNascimento):
        print("Verifique o ano de nascimento, pode estar incorreto ou
ser menor de idade.")
        return
    # Verificação se as senhas são iguais
    if senha != confirmaSenha:
        print("Senhas diferentes.")
        return
    if not verificador.verificadorDeEstadoCivil(estadoCivil):
        print("Estado civil incorreto.")
        return
    usuario = {
        "nome": nome,
        "email": email,
        "senha": senha,
        "rg": rg,
        "cpf": cpf,
        "nacionalidade": nacionalidade,
        "estado civil": estadoCivil,
        "Data de nascimento": anoDeNascimento
    }
    alteracoes.salvar_dados(usuario)
    print("Cadastrado.")
print("Plataforma de Gestão de Dados Pessoais")
print("--------------------------------------")
while True:
    resposta = input("Digite 1, se deseja entrar em sua conta. Digite
2, se deseja se cadastrar. Digite 3, se deseja sair: ")
    if resposta == "1":
        entrar()
    elif resposta == "2":
        iniciarCadastro = input("Você autoriza o uso dos seus dados
conforme nossa política de privacidade (s/n): ")
        if iniciarCadastro == "s"  or iniciarCadastro == "S":
            cadastrar()
        else:
            break
    elif resposta == "3":
```

```
        break
```

8 CONCLUSÃO

Com base nesse trabalho, tendo em vista os termos supracitados e elencados, pode-se concluir, que para o desenvolvimento de uma plataforma de gestão de dados pessoais é necessário o cumprimento da lei geral de proteção de dados para ter confiabilidade em relação aos dados pessoais, para o auxílio disso precisamos trabalhar com a cibersegurança. Para assegurar que a plataforma seja bem desenvolvida é preciso uma excelente infraestrutura computacional e um código bem estruturado. Assim, os objetivos que a equipe obtinha em relação ao desenvolvimento da plataforma foram atingidos, pois com as tecnologias como criptografia, firewalls, testes de penetração vamos garantir a segurança dos dados. Utilizando servidor web, MySQL, Elastic Load Balancing (ELB) A infraestrutura computacional está muito bem articulada. Assim podemos resolver o problema de falta de segurança que está muito presente nos dias atuais. Considerações finais que temos é a importância do estudo de cibersegurança, LGPD, infraestrutura computacional e linguagens de programação como Python, dessa maneira entendemos a relevância delas para os projetos de software que possuem um alto nível de complexidade e vão tratar de assuntos de sensíveis, como é o caso de uma plataforma de gestão de dados pessoais.

9 REFERÊNCIAS

BENETON, José. Cibersegurança. Disponível em: https://ava.ead.unip.br/bbcswebdav/pid-4569238-dt-content-rid-15856343_1/institution/Conteudos_AVA/ASSOCIADAS_UNIP/D56G%20-%20Ciberseguran%C3%A7a/Livro-Texto_Unid_I.pdf. Acesso: 17/05/2025.

Gov.br. Lei geral de proteção de dados (LGPD). Disponível em:

https://www.gov.br/mds/pt-br/acesso-a-informacao/privacidade-e-protecao-de-dados/lgpd. Acesso: 18/05/2025.

Serpro. O que muda com a LGPD. Disponível em : https://www.serpro.gov.br/lgpd/menu/a-lgpd/o-que-muda-com-a-lgpd. Acesso: 12/05/2025.

Gov.br. Autoridade nacional de proteção de dados. Disponível em:

https://www.gov.br/anpd/pt-br. Acesso: 15/05/2025.

Bioni, Bruno. Proteção de dados contexto, narrativas e elementos fundantes. Disponível em:

https://observatoriolgpd.com/wp-content/uploads/2021/08/1629122407livro-LGPD-Bruno-Bioni-completo-internet-v2.pdf. Acesso: 15/05/2025.

aws. O que é Balanceamento de carga. Disponível em:

https://aws.amazon.com/pt/what-is/load-balancing/. Acesso: 30/05/2025 às 16:00.

W. Eduardo. Web Server: O que é e Como Funciona? Disponível em:

https://www.hostinger.com/br/tutoriais/web-server. Acesso 23/05/2025.

Redação DPOnet. 3 Vantagens de Contar Com um Software de Gestão de Dados. 30 de março de 2023. Disponível em: https://blog.dponet.com.br/vantagens-de-contar-com-um-software-de-gestao-de-dados/. Acesso 23/05/2025.

Equipe Editorial da Psico-smart. Ferramentas de produtividade baseadas em nuvem: vantagens e desvantagens. 28 de agosto de 2024. Disponível em: https://psico-smart.com/pt/blogs/blog-ferramentas-de-produtividade-baseadas-em-nuvem-vantagens-e-desvantagens-141732. Acesso 23/05/2025.

SANTODIGITAL. Tudo o que você precisa saber sobre a política de backup. 28 de agosto de 2024. Disponível em: https://santodigital.com.br/politica-de-backup/. Acesso 23/05/2025.

ControleNet. Web Server: Saiba o que é e como funciona um servidor web. Disponível em: https://www.controle.net/faq/web-server-o-que-e-como-funciona-um-servidor-web. Acesso 23/05/2025.