CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

Bruno Henrique Duwe

**PostgreControl - Software Web para Monitoramento de Bancos de Dados PostgreSQL.**

RIO DO SUL

2021

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

Bruno Henrique Duwe

**PostgreControl - Software Web para Monitoramento de Bancos de Dados PostgreSQL**

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí, como condição parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Orientador: Esp. Cleber Nardelli

RIO DO SUL

2021

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

Bruno Henrique Duwe

**POSTGRECONTROL - SOFTWARE WEB PARA MONITORAMENTO DE BANCOS DE DADOS POSTGRESQL**

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí- UNIDAVI, a ser apreciado pela Banca Examinadora, formada por:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor Orientador: Esp. Cleber Nardelli

Banca Examinadora:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. M.e Fernando Andrade Bastos

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. M.e Marciel De Liz Santos

Rio do Sul, 29 de novembro de 2021.

Aquele que tem uma razão para viver pode suportar quase tudo (Friedrich Nietzsche).

A minha família por todo apoio.

AGRADECIMENTOS

A minha família por todo apoio oferecido durante o período de formação acadêmica e realização deste trabalho de finalização.

Ao professor e orientador Cleber Nardelli, por todo tempo e auxílio dispendido para realização deste projeto.

RESUMO

O banco de dados é uma ferramenta de extrema importância que muitas vezes não recebe a devida atenção, e quando ocorrem problemas como lentidão ou erros eles devem ser detectados e eliminados o mais rápido possível já que um SGBD parado ou lento pode significar prejuízos. Para isto é necessário que haja um profissional com conhecimento técnico sobre o tema, para que sejam aplicadas as correções, este profissional nem sempre está disponível fazendo com que pessoas sem conhecimento tentem resolver o problema pelo método tentativa e erro e todo esse processo leva tempo, fazendo com que os usuários do SGBD tenham que lidar com instabilidade ou indisponibilidade dos dados. Para que pequenos problemas não se tornem maiores o monitoramento do banco de dados é necessário, detectando problemas no início, dando assim maior tempo de reação e correção. Com isto surge a pergunta: é possível disponibilizar acesso remoto e facilitado aos principais indicadores de monitoramento no SGBD PostgreSQL de forma proativa e simples? Pensando nestes pontos o presente trabalho pretende desenvolver um protótipo de software para monitoramento de banco de dados PostgreSQL chamado “PostgreControl”, que visa trazer informações importantes em um único local. Ele trará gráficos e informações tabulares sobre o estado do banco de dados em geral, sobre as tabelas e as conexões, além de métricas que facilitam a visualização por pessoas com menor conhecimento técnico, tornando a detecção de problemas mais rápida e intuitiva.

**Palavras-Chave:** monitoramento, banco de dados, tecnologia.

ABSTRACT

The database is an extremely important tool that often does not receive due attention, and when problems such as slowness or errors occur, they must be detected and eliminated as quickly as possible, since a stopped or slow DBMS can mean losses. For this it is necessary that there is a professional with technical knowledge on the subject so that corrections can be applied, this professional is not always available, causing people without knowledge to try to solve the problem by trial and error and this whole process takes time, causing DBMS users to deal with data instability or unavailability. So that small problems do not become bigger, database monitoring is necessary, detecting problems at the beginning, thus giving more reaction and correction time. With this, the question arises: is it possible to provide remote and facilitated access to the main monitoring indicators in the PostgreSQL DBMS in a proactive and simple way? With these points in mind, this work intends to develop a prototype software for monitoring PostgreSQL database called “PostgreControl”, which aims to bring important information in a single place. It will bring graphs and tabular information about the state of the database in general, about the tables and connections, as well as metrics that facilitate the visualization by people with less technical knowledge, making troubleshooting faster and more intuitive.

**Keywords:** monitoring, database, technologies.

LISTA DE FIGURAS

Figura 1- Instalador do agente Datadog 31

Figura 2- Arquivo de configuração do módulo PostgreSQL 32

Figura 3- Exemplo de *Dashboard* 32

Figura 4- *Dashboard* do pgAdmin 34

Figura 5- Diagrama de caso de uso completo 39

Figura 6- Funções do *front end* de conexão com o banco de dados 41

Figura 7- Função do *front end* para gerar gráfico pizza de linhas movimentadas 42

Figura 8- Função do *front end* para gerar tabela das conexões com o banco de dados 43

Figura 9- Lista de rotas do *back end* 44

Figura 10- Função cadastro 45

Figura 11- Funções para criptografia dos dados 46

Figura 12- Funções para execução de comandos SQL 46

Figura 13- Página principal 47

Figura 14- Página de cadastro 48

Figura 15- Monitoramento geral do banco de dados 49

Figura 16- Monitoramento geral das tabelas 50

Figura 17- Exemplo de gráfico de barras 50

Figura 18- Monitoramento das conexões 51

LISTA DE QUADROS

Quadro 1 - Principais tags HTML 20

Quadro 2 - Maneiras de aplicar o CSS 22

Quadro 3 - Diretrizes para criação de funções 24

Quadro 4 - Fundamentos do Node.js 25

Quadro 5 - Regras de negócio 36

Quadro 6 - Requisitos não funcionais 37

Quadro 7 - Requisitos funcionais 38

LISTA DE ABREVIATURAS E SIGLAS

API Application Programming Interface

BSD Berkeley Source Distribution

CSS Cascading Style Sheets

DBA Database Administrator

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

SGBD Sistema de Gerenciamento de Banco de Dados

SQL Structured Query Language

**SUMÁRIO**

1. INTRODUÇÃO 13

1.1 problema de pesquisa 14

1.2 objetivos 14

1.2.1 Geral 14

1.2.2 Específicos 14

1.3 JUSTIFICATIVA 15

2. REFERENCIAL TEÓRICO 16

2.1 APLICAÇÕES WEB 16

2.2 LINGUAGENS DE PROGRAMAÇÃO 18

2.3 HTML 19

2.4 CSS 21

2.5 JAVASCRIPT 23

2.6 NODEJS 25

2.7 BANCO DE DADOS 26

2.8 SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS 27

2.9 POSTGRESQL 28

2.10 LINGUAGEM SQL 29

3. Metodologia DA PESQUISA 30

3.1 ESTADO DA ARTE 30

3.1.1 Datadog 30

3.1.2 pgAdmin 33

4. POSTGRECONTROL - SOFTWARE WEB PARA MONITORAMENTO DE BANCOS DE DADOS POSTGRESQL 35

4.1 VISÃO GERAL DO PROTÓTIPO DE SOFTWARE 35

4.2 REGRAS DE NEGÓCIO 36

4.3 REQUISITOS 36

4.3.1 Requisitos não funcionais 37

4.3.2 Requisitos funcionais 38

4.4 DIAGRAMAS 38

4.5 IMPLEMENTAÇÃO 40

4.5.1 *Front end* 40

4.5.2 *Back end* 43

4.6 FUNCIONAMENTO 46

4.6.1 Visão geral 47

4.6.2 Cadastro do banco de dados 47

4.6.3 Monitoramento geral 48

4.6.4 Monitoramento de tabela 49

4.6.5 Monitoramento de conexão 51

5. CONSIDERAÇÕES FINAIS 52

5.1 RECOMENDAÇÕES DE TRABALHOS FUTUROS 53

1. introdução

Na grande maioria dos casos o uso de banco de dados é indispensável para o crescimento e estabilidade de uma aplicação de software, pois permite que o armazenamento e a manipulação dos dados sejam feitas por várias conexões, sendo pessoas ou maquinas, sem que estes precisem se preocupar em como é feito, visto que é o Sistema Gerenciador de Banco de Dados (SGBD) quem controla estas tarefas. Segundo Coronel e Morris (2018) o SGBD atua como um intermediário entre o usuário e o banco de dados, onde o banco de dados armazena as coleções de arquivos de tal forma que apenas o SGBD consiga ter acesso para ler, atualizar, inserir e deletar informações. Ter um SGBD entre os aplicativos e o banco de dados oferece algumas vantagens importantes. O SGBD permite que os dados no banco de dados sejam compartilhados entre vários aplicativos ou usuários. Além disso, o SGBD integra as diferentes visualizações dos usuários dos dados em um único repositório de dados abrangente. Mas mais importante é o fato de o SGBD abstrair dos usuários o controle de acesso aos dados, incluindo como eles são escritos, onde eles são escritos e em qual formato, trabalhando diretamente com o Sistema Operacional.

Ao realizar o uso do SGBD sem a devida atenção e manutenção proativa, o mesmo poderá ter problemas futuros no funcionamento e desempenho. Segundo Matthew e Stones (2005) o desempenho é frequentemente um problema presente em bancos de dados maiores. Não importa a rapidez com que o banco de dados é executado, quase sempre há alguém que perguntará se ele pode ter seu desempenho melhorado sem alterar especificações dos equipamentos físicos ou manter o desempenho reduzindo os recursos computacionais da máquina.

Mesmo que indiretamente, o desempenho e o funcionamento constante do banco de dados são assuntos recorrentes para os Desenvolvedores e de maneira direta aos Administradores de Banco de Dados (DBA). Com isso a necessidade do monitoramento do banco de dados, com métricas para verificar se existem problemas e onde eles estão presentes, para que decisões possam ser tomadas rapidamente é essencial. Há ainda um terceiro grupo de usuários, os Administradores de Tecnologia que nem sempre possuem o devido conhecimento na gestão do SGBD.

Pensando nos pontos acima citados, pretende-se no trabalho desenvolver um protótipo de ferramenta capaz de suprir as necessidades do administrador de banco de dados, trazendo as métricas de forma visual diretamente para os administradores. Com gráficos e relatórios a ferramenta trará um maior controle e permitirá que o administrador de banco de dados possa tomar as decisões de forma mais precisa, mas também servirá de apoio ao administrador de tecnologia, ao demonstrar o estado do SGBD de forma facilitada e sintetizada.

1.1 PROBLEMA DE PESQUISA

Como disponibilizar acesso remoto e facilitado aos principais indicadores de monitoramento no SGBD PostgreSQL de forma proativa?

1.2 Objetivos

Na presente seção serão expostos em forma de tópicos os objetivos que o trabalho visa alcançar.

1.2.1 Geral

* Desenvolver protótipo de software web para monitorar bancos de dados PostgreSQL.

1.2.2 Específicos

* Identificar os principais itens de monitoramento do SGBD;
* Realizar análise de requisitos do protótipo com base nos itens identificados;
* Criar as funcionalidades no protótipo;
* Testar recursos criados, aplicando revisões necessárias;
* Documentar recursos criados.

1.3 Justificativa

Com o uso diário do banco de dados podem ocorrer problemas como lentidão ou erros que precisam ser detectados e eliminados o mais rápido possível, já que um SGBD parado ou lento significa prejuízos. Para encontrar os processos e descobrir onde está o problema é necessário conhecimento técnico que nem sempre está disponível no momento, fazendo com que a tentativa e erro tenha que ser usada, todo esse processo leva tempo, onde os usuários do SGBD tenham que lidar com instabilidade ou indisponibilidade dos dados. Assim surge a necessidade de monitoramento do SGBD. Com isso é possível descobrir a fonte do problema muito mais rápido, permitindo assim uma tomada de decisão imediata, reduzindo os prejuízos.

Pensando nisso o “PostgreControl”, um protótipo de software para monitoramento de banco de dados PostgreSQL, visa trazer informações importantes em um único local, de forma facilitada e precisa. Esta ferramenta trará gráficos e relatórios importantes para o monitoramento e administração do SGBD no dia a dia, por meio de informações em tempo real visando facilitar o trabalho do DBA e Administradores de Tecnologia.

Apresentada a ideia, esta pesquisa se faz necessária para obtenção de uma ferramenta que consiga ser sucinta e traga as informações necessárias para o monitoramento diário do banco de dados PostgreSQL.

2. REFERENCIAL TEÓRICO

Quando se fala sobre inovação vem à mente internet, que é a rede mundial de computadores que já possui seu espaço, mas continua em expansão. Segundo Robbins (2018), a internet experimentou uma expansão inicial eufórica, um colapso impulsionado pela economia, um renascimento impulsionado pela inovação e uma evolução constante ao longo do caminho. Uma coisa é certa: a internet como meio de comunicação e comercial veio para ficar. Não só isso, ela encontrou seu caminho em dispositivos como smartphones, tablets, TVs e muito mais.

2.1 Aplicações web

O uso de aplicações web tem crescido muito nos últimos anos, estas aplicações têm as mais variadas funções. Segundo White Belt Mastery (2020), hoje, a internet está acessível em quase todas as partes do mundo. Nas últimas duas décadas, a internet e a web cresceram rapidamente. Se você voltar duas décadas atrás, os sites eram muito diferentes. Eles não eram nada atraentes e o mais importante, eram estáticos. Por estático, quero dizer, tudo em uma página da web era fixo. Mas hoje em dia, os sites são dinâmicos. Em um site estático, tudo é fixo até que alguém altere manualmente por trás. Esses sites são criados usando HTML e são a parte mais direta do desenvolvimento de sites. Todos os usuários que visitam um site estático têm a mesma visualização. Mas o conteúdo de um site dinâmico pode ser diferente para cada usuário. Por exemplo, a página inicial da Amazon é relativamente diferente para um usuário autenticado e um usuário não autenticado. Se você não estiver autenticado, não poderá ver as informações da sua conta, histórico de pedidos e outras coisas. Ele aparece apenas quando você faz login com suas credenciais. Um site dinâmico está vinculado a pelo menos um banco de dados onde todas as informações dinâmicas são armazenadas. Esse banco de dados não existe no caso de sites estáticos. A interação do usuário é outra parte essencial de um site dinâmico.

Para Ubelhor e Hur (2017), as aplicações web permitem o uso de seus sistemas e software para mais usuários e geralmente envolvem vários servidores. Esses servidores podem estar próximos, em todo o país ou até mesmo em outro país. Os usuários estarão se conectando e acessando as aplicações usando navegadores da web. As aplicações se comunicam com as páginas dinâmicas da web usando uma ou mais ferramentas adicionais.

Aplicações web normalmente utilizam o modelo cliente-servidor, onde cada camada tem sua função específica. De acordo com Ubelhor e Hur (2017), no lado do cliente, o conteúdo dinâmico é gerado no sistema do cliente. O servidor recupera páginas HTML com código incorporado e as envia ao cliente. O navegador então processa o código embutido na página. O código do lado do cliente pode ser usado para coisas como alteração de conteúdo, validação de entrada, identificação de condições de ambiente ou gatilho de eventos de ação. As operações realizadas no lado do cliente em um modelo cliente-servidor requerem acesso a informações ou funcionalidades disponíveis no cliente, mas não no servidor. O cliente é um recurso de usuário dedicado, portanto, o processamento no lado do cliente pode fornecer tempos de resposta rápidos quando projetado e codificado adequadamente. Concluir o processamento no lado do cliente também pode reduzir os riscos de segurança. As configurações do navegador do usuário determinam como o código do lado do cliente incorporado em HTML será tratado. A programação e os comandos do lado do servidor fornecem outra maneira de adicionar dinamicidade ao seu site. Os comandos ou programas do lado do servidor são executados quando chamados do cliente através de uma solicitação HTTP.

Sobre lado do servidor continua Ubelhor e Hur (2017). Alguns dos usos da programação e comandos do lado do servidor incluem:

* processar consultas de usuários, recuperando e retornando dados selecionados;
* adicionar, alterar ou excluir o conteúdo da página da web;
* acessar dados ou bancos de dados para recuperar, adicionar, alterar ou excluir dados;
* criar aplicativos para processar informações;
* desenvolver aplicativos com base na lógica de negócios, fornecendo segurança e controle de acesso;
* validar dados ou entradas;
* adaptar a saída para diferentes tipos de navegadores personalizando páginas da web para torná-las amigáveis ao usuário;
* melhorar o desempenho do aplicativo;
* traduzir ou transformar formatos de dados;
* interface com outros aplicativos ou sistemas.

Basicamente, o desenvolvimento do lado do servidor é usado para todas as tarefas para as quais as linguagens de programação tradicionais são usadas. Ao contrário dos comandos utilizados no lado do cliente, os comandos utilizados no lado do servidor não permitem que o usuário visualize o código-fonte. Os comandos do lado do cliente têm maior acesso a informações e funções no computador do usuário. Os comandos do lado do servidor exigem que o interpretador da linguagem esteja instalado no servidor e produza os mesmos resultados, independentemente do navegador do cliente ou sistema operacional usado. Os scripts do lado do cliente precisam ser escritos em uma linguagem compatível com os navegadores usados pela maioria dos usuários do site.

2.2 Linguagens de programação

Para criação de aplicações web utiliza-se de linguagens de programação, aplicando estas linguagens em documentos escritos utilizando palavras especificas, estas palavras juntas criam um código que quando é interpretado por um navegador, por exemplo, geram os resultados visuais e comportamentais. A programação é um grande desafio pois faz com que seja utilizado tecnologias distintas para obter os resultados esperados. Segundo Silveira e Almeida (2013, p. 1), “Não importa sua idade, profissão e objetivo: programar é mais do que divertido, é um constante desafio. Queremos tornar o programa mais rápido, mais legível, mais elegante e mais útil.”. Isso é refletido quando fazemos o caminho inverso, olhando para um site e pensando em todo trabalho feito para obter este resultado. De acordo com Robbins (2018), quando olho para um site, vejo a infinidade de decisões e áreas de especialização necessárias para criá-lo. Os sites são mais do que apenas códigos e imagens. Frequentemente, começam com um plano de negócios ou outra missão definida. Antes do lançamento, o conteúdo deve ser criado e organizado, a pesquisa é realizada, o design desde os objetivos mais amplos até os pequenos detalhes deve acontecer, o código é escrito e tudo deve ser coordenado com o que está acontecendo no servidor para torná-lo realidade.

Em projetos de programas de computador um ponto importante é a escolha da linguagem de programação que será utilizada, essa escolha normalmente é complexa, envolvendo vários fatores. Segundo Ubelhor e Hur (2017), a linguagem que você escolher depende de você e será o produto de uma decisão que considera muitos fatores, como seus conjuntos de habilidades atuais, conjuntos de habilidades de seus colegas de trabalho, conjuntos de habilidades de recursos disponíveis, ordens de executivos, aplicativos existentes e preferências pessoais. Onde cada uma das linguagens tem seus próprios recursos, vantagens e desvantagens exclusivos.

Quando o assunto é linguagens de programação para desenvolvimento web, naturalmente vem a mente uma área em crescimento com tecnologias mais recentes, quando se comparado com linguagem de programação em geral. Segundo Ubelhor e Hur (2017), embora o desenvolvimento web já exista há alguns anos, ainda está em sua infância em comparação com o desenvolvimento de aplicativos de negócios não baseados na web. O crescimento foi como um incêndio e, de forma alguma, o desenvolvimento web para aplicativos de negócios é uma tecnologia madura. Como acontece com qualquer outra tecnologia em crescimento, novas linguagens e ferramentas de desenvolvimento da web continuam a ser desenvolvidas e introduzidas. Há muito debate sobre quais são as melhores ferramentas e o que será popular no futuro.

Esta diferença pode ser explicada por Ubelhor e Hur (2017) onde muitas linguagens e ferramentas usadas para desenvolvimento web são bastante semelhantes. Frequentemente, uma combinação de linguagens é usada para criar um aplicativo web. O desenvolvimento web tende a ser muito mais aberto do que o desenvolvimento tradicional no que diz respeito a misturar as coisas. Tradicionalmente, por exemplo, o COBOL é usado em um *mainframe* e o *Visual Basic* é usado em um servidor Microsoft. O desenvolvimento da web, por outro lado, normalmente não bloqueia uma organização em uma plataforma específica ou em uma única linguagem.

2.3 HTML

 Quando se fala em desenvolvimento web o HTML deve ser mencionado, pois ele descreve a estrutura de uma página da web. Para Silveira e Almeida (2013, p. 9):

Está curioso com a sigla HTML? Ela significa *Hypertext Markup Language* (linguagem de marcação para hipertexto). Ou, em uma tradução bem aberta, uma linguagem que possui tags para marcar documentos do tipo hipertexto. Documento hipertexto é um que pode ter links para outros. O HTML sozinho nada mais é que uma forma de marcação (através das tags).

Já para Ubelhor e Hur (2017), HTML é a sigla para *Hypertext Markup Language*. *Hypertext* refere-se à capacidade de criar links para outras páginas da web. *Markup* significa que é usado para criar páginas formatadas com texto, imagens e outros recursos incorporados. O HTML foi criado por Tim Berners-Lee em 1990. Ele também inventou o termo “*World Wide Web*” e o primeiro navegador da Web, e acabou fundando o *World Wide Web Consortium*, também conhecido como W3C. Entre outras coisas, o W3C atua como um pastor zelando pelo desenvolvimento das tecnologias da web. Ele fornece diretrizes, padrões, recomendações e educação sobre muitos aspectos da tecnologia relacionada à web.

Continuam Ubelhor e Hur (2017), a codificação HTML é baseada em um conjunto de símbolos de marcação colocados em um documento HTML ou página da web. Esses símbolos de marcação identificam elementos estruturais ou tags que informam a um navegador da Web (e outros agentes de usuário) como renderizar e exibir uma página da Web.

E para White Belt Mastery (2020), a estrutura básica de uma página da web é criada usando HTML. Existem vários elementos HTML e eles são o bloco de construção dessas páginas. Os elementos HTML são usados na forma de tags. As tags são colchetes angulares com nomes HTML escritos dentro deles. Por exemplo, a tag HTML para imagem é <img />. A maioria dessas tags tem uma tag de fechamento como <p> e </p>. No entanto, algumas tags, como <img /> não exigem uma tag de fechamento.

O HTML possui tags com comportamentos específicos, conforme apresentado no Quadro 1:

**Quadro 1 – Principais tags HTML**

|  |  |
| --- | --- |
| **<!DOCTYPE>** | A declaração deste elemento define que este documento é um documento HTML5. A declaração não é uma tag HTML. É uma "informação" para o navegador sobre que tipo de documento esperar. |
| **<html>** | É o elemento raiz de uma página HTML, dentro dele estão todas as outras tags HTML. |
| **<head>** | Elemento é um contêiner para metadados (dados sobre dados). Os metadados não são exibidos. Os metadados normalmente definem o título do documento, conjunto de caracteres, estilos, scripts e outras metainformações. |
| **<title>** | Define o título do documento. O título deve ser somente texto e é mostrado na barra de título do navegador ou na guia da página. |
| **<body>** | Define o corpo do documento. O elemento contém todo o conteúdo de um documento HTML, como cabeçalhos, parágrafos, imagens, hiperlinks, tabelas, listas etc. |
| **<form>** | É usada para criar um formulário. |
| **<h1> até <h6>** | São usados para definir cabeçalhos HTML. Sendo <h1> o cabeçalho mais importante e <h6> o título menos importante. |
| **<p>** | Define um parágrafo. |
| **<br>** | Insere uma única quebra de linha. É uma tag vazia, o que significa que não possui tag final. |
| **<a>** | Define um hiperlink, que é usado para vincular uma página à outra. O atributo mais importante é o *href*, que indica o destino do link. |
| **<script>** | É usado para incorporar um script do lado do cliente (JavaScript). |
| **<style>** | É usado para definir informações de estilo (CSS) para um documento |
| **<img>** | É usado para incorporar uma imagem em uma página HTML. As imagens não são tecnicamente inseridas em uma página da web; as imagens estão vinculadas a páginas da web. A tag cria um espaço de retenção para a imagem referenciada. Possui dois atributos obrigatórios, sendo eles, o src que especifica o caminho para a imagem e o alt que especifica um texto alternativo para a imagem, se a imagem por algum motivo não puder ser exibida |
| **<nav>** | Define um conjunto de links de navegação. |
| **<div>** | Define uma divisão ou seção em um documento HTML. É usado como um contêiner para elementos HTML - que são estilizados com CSS ou manipulados com JavaScript. É facilmente estilizado usando o atributo *class* ou id e qualquer tipo de conteúdo pode ser colocado dentro da tag. |
| **<footer>** | Define um rodapé para um documento ou seção. Normalmente contém informação de autoria, informações sobre direitos autorais, informações de contato, mapa do site, navegação aos links principais e os documentos relacionados. |

Fonte: Elaborado a partir de W3Schools (2021)

Segundo Ubelhor e Hur (2017), HTML é uma linguagem muito flexível e não faz distinção entre maiúsculas e minúsculas. Isso significa que, mesmo que você codifique sua página da web com erros, o navegador ainda processará e exibirá a página corretamente para o usuário final.

2.4 CSS

Para White Belt Mastery (2020), *Cascading Style Sheets* ou comumente conhecido como CSS é a parte de apresentação de uma página da web. O HTML cria uma estrutura e CSS a converte em uma versão atraente e mais legível. Nenhum site está completo sem CSS hoje. Os usuários esperam que um site seja atraente, envolvente e, acima de tudo, devidamente legível.

Continua White Belt Mastery (2020), com o CSS você pode alterar a fonte, cor, tamanho, posições, *layouts* e muito mais. Existem várias maneiras de usar CSS em um arquivo HTML, cada uma com sua própria vantagem.

Já para Ubelhor e Hur (2017), o CSS é uma linguagem simples que define construções de estilo para formatar a aparência de uma página web. As regras de estilo definem um conjunto de instruções de formatação (um ou mais pares de propriedade-valor) que afetam os elementos HTML em uma página web. O CSS é uma linguagem de apresentação, em contraste com HTML, que é uma linguagem semântica. CSS é uma linguagem de apresentação porque configura, formata e estiliza os elementos HTML para dar à página da Web sua aparência visual. O HTML, em contraste, é uma linguagem semântica, o que significa que um elemento descreve seu significado associado ao conteúdo para um agente do usuário, como um navegador Web ou um leitor de tela. Em outras palavras, você não encontrará uma instrução de loop ou uma função em CSS. É a linguagem de complemento perfeito para HTML e foi projetada principalmente para funcionar com HTML.

Continua Ubelhor e Hur (2017), as regras e padrões de estilo CSS são definidos e mantidos pelo World Wide Web Consortium (W3C), que introduziu a primeira versão (ou nível) de CSS, CSS1, em 1996, seguido por CSS2 em 1998 e CSS3 em 2005. Cada nível mais recente de CSS é construído sobre o nível anterior. Na época em que isto foi escrito, CSS Nível 3 (CSS3) ainda era a versão mais atualizada usada no desenvolvimento da Web de hoje. Embora o CSS3 tenha sido lançado há mais de uma década, muitas novas propriedades e regras foram introduzidas gradualmente.

Visto que o CSS vem para aprimorar a parte visual do HTML pode-se tratar ele como uma linguagem complementar, ou seja, não se pode construir algo somente com CSS, ele deve ser aplicado sobre os elementos ou tags de outra linguagem.

O CSS trabalha atribuindo propriedades aos elementos da linguagem base. Para Ubelhor e Hur (2017), uma regra de estilo CSS contém duas partes: seletor e declaração. Um seletor pode ser qualquer tag HTML (como p, h1 ou *div*) ou um seletor definido pelo usuário, como uma classe ou id. A declaração é uma lista de pares de propriedade-valor que estão associados ao seletor. Uma propriedade é qualquer construção, como tamanho da fonte, cor ou posição com valores como 12px, vermelho ou relativo, respectivamente.

Continua Ubelhor e Hur (2017), um seletor pode ter mais de um par de valor de propriedade. Cada par deve ser separado por um ponto e vírgula (;) e deve começar em uma nova linha. Ao definir o estilo de um elemento (como uma tag <p>), a tag do elemento é listada sem os colchetes angulares (<>) e seguida por colchetes ({}). Dentro dessas chaves, qualquer propriedade disponível pode ser definida. Cada propriedade é seguida por dois pontos (:) e, em seguida, por seu valor associado. Cada valor é seguido por um ponto e vírgula (;). Você pode listar propriedades adicionais conforme necessário e a chave final é codificada após todas as propriedades listadas.

Finaliza Ubelhor e Hur (2017), CSS é uma linguagem flexível e segue regras semelhantes às do HTML. As regras de estilo podem ser escritas em uma única linha ou em várias linhas porque os espaços em branco são ignorados. Além disso, as regras de estilo podem ser aplicadas a mais de um elemento. Para aplicar as mesmas regras de estilo a vários elementos, basta separar cada seletor com uma vírgula.

Segundo W3Schools (2021), existem 3 maneiras de aplicar o CSS, externamente, internamente e em linha, conforme apresentado no Quadro 2.

**Quadro 2 – Maneiras de aplicar o CSS**

|  |  |
| --- | --- |
| **CSS externo** | Com uma folha de estilo externa, você pode alterar a aparência de um site inteiro alterando apenas um arquivo. Cada página HTML deve incluir uma referência ao arquivo de folha de estilo externo dentro do elemento <link>, na seção head. |
| **CSS interno** | Uma folha de estilo interna pode ser usada se uma única página HTML tiver um estilo exclusivo. O estilo interno é definido dentro do elemento <style>, na seção head. |
| **CSS em linha** | Um estilo embutido pode ser usado para aplicar um estilo único a um único elemento. Para usar estilos embutidos é necessário adicionar o atributo de estilo ao elemento relevante. O atributo de estilo pode conter qualquer propriedade CSS. |

Fonte: Elaborado a partir de W3Schools (2021)

2.5 javascript

Para White Belt Mastery (2020), JavaScript é uma linguagem de script que desempenha um papel vital no desenvolvimento de sites dinâmicos. É usado para interação com o usuário, gerenciamento de conteúdo, manipulação de bancos de dados e muito mais. JavaScript é uma vasta linguagem de programação que cobre vários conceitos e, o mais importante, muitos desses conceitos de maneiras diferentes.

Continua White Belt Mastery (2020), por definição, JavaScript é uma linguagem de programação multiparadigma de alto nível. É uma das principais tecnologias da *World Wide Web* (WWW), sendo as outras duas HTML e CSS. JavaScript é uma parte crucial e essencial do desenvolvimento de aplicativos da web. HTML cria uma estrutura básica e CSS a transforma em uma versão atraente. Um site com apenas HTML e CSS é um site estático que quase não tem uso. O foco principal de um site é a interação do usuário e se você precisa de um site dinâmico, precisa usar JavaScript com HTML e CSS.

White Belt Mastery (2020), finaliza dizendo que JavaScript é uma linguagem de script usada tanto no lado do cliente quanto no lado do servidor. No passado, o JavaScript só podia ser executado em um navegador, mas com a introdução do node.js, ele pode ser executado fora também. Estruturas da Web e bibliotecas como Angular, React, Vue são construídas usando JavaScript. Com o node.js, também é possível criar serviços no lado do servidor.

No JavaScript pode-se criar variáveis para receber valores que serão mantidos em memória e sempre que for necessário pode-se obter, alterar e até mesmo limpar o seu conteúdo. As variáveis possuem tipos, estes tipos definem o conteúdo que a variável carrega. Segundo White Belt Mastery (2020), em uma variável pode-se armazenar números de precisão simples, números de precisão dupla, textos, valores booleanos, objetos, datas etc. Esses são diferentes tipos de dados usados no mundo da programação. Um tipo de dado indica as características dos dados armazenados em uma variável. Nas outras linguagens de programação, como Java, é obrigatório especificar o tipo de dado de uma variável ao declará-la e, além disso, uma variável pode conter apenas esse tipo de dado. Mas não existem tais restrições no JavaScript. Os tipos de dados no JavaScript são divididos em duas categorias - primitivos e não primitivos. Os primitivos são *number*, *string*, *boolean* e *undefined* enquanto os não primitivos são *object*, *array* e *date*.

No JavaScript existe um método para reduzir código, este método consiste em escrever o código dentro de um agrupamento dando uma referência para ele, assim como definindo os parâmetros necessários para execução deste bloco e sempre que precisar executar este agrupamento de código específico basta chamá-lo pela referência passando os valores dos parâmetros necessários. Este método é chamado de função.

Segundo White Belt Mastery (2020), as funções são chamadas de blocos de construção principal de um programa. Eles fornecem capacidade de reutilização de código e ajudam a reduzir o tempo e o esforço. Pode-se chamar uma função a qualquer hora em qualquer lugar que quiser. Eliminando assim a necessidade de repetição. As funções são uma das partes mais importantes da programação. Quase todas as linguagens têm o conceito de funções.

Para Elliott (2014), existem diretrizes para criar melhores funções, conforme Quadro 3.

**Quadro 3 – Diretrizes para criação de funções**

|  |  |
| --- | --- |
| ***Don’t Repeat Yourself* (DRY)** | Bons programadores são preguiçosos e muito produtivos. Eles expressam muitas funcionalidades em pouco código. Depois de estabelecer um padrão para algo que é repetido no código, é hora de escrever uma função, objeto ou módulo que encapsula esse padrão para que possa ser facilmente reutilizado.Também pode-se colocar em quarentena essa funcionalidade em um único ponto do código, de modo que, se mais tarde você encontrar algo errado com o código ou o algoritmo, você só terá que consertar em um lugar.Escrever uma função reutilizável também força você a isolar o padrão do problema, o que ajuda a manter a funcionalidade relacionada agrupada. |
| ***Do One Thing* (DOT)** | Cada função deve fazer apenas uma coisa e fazê-la da melhor maneira possível. Seguir esse princípio tornará sua função mais reutilizável, mais legível e mais fácil de depurar. |
| ***Keep It Simple Stupid* (KISS)** | Os programadores costumam ser tentados a encontrar soluções inteligentes para os problemas. Isso é uma coisa boa, é claro, mas às vezes os programadores são muito espertos e as soluções são enigmáticas. Isso tende a acontecer quando uma única linha de código é usada para realizar mais do que uma única instrução. |
| ***Less Is More*** | Para ajudar na legibilidade e reduzir a tentação de fazer mais de uma coisa, as funções devem ser as mais curtas possíveis: código suficiente para fazer a única coisa para a qual foram feitas, e nada mais. Na maioria dos casos, as funções devem ter apenas um punhado de linhas. Se eles forem executados por muito mais tempo, considere dividir subtarefas e dados em funções e objetos separados. |

Fonte: Elaborado a partir de Elliott (2014)

As funções JavaScript podem ser utilizadas no HTML também e segundo Ubelhor e Hur (2017), uma função JavaScript é usada para agrupar e isolar o código que será executado por uma ocorrência de evento ou por uma chamada para a função, não simplesmente executado quando a página for carregada. Uma função pode ser chamada de qualquer lugar da página. Se a função for salva em um arquivo .js externo, ela também pode ser usada em outras páginas. As funções podem ser definidas na seção <head> ou <body> de um documento HTML. Se você deseja que uma função seja carregada antes de ser chamada, coloque-a na seção <head>.

2.6 Nodejs

Segundo Mardan (2018) o Node.js foi construído sobre o mecanismo Google Chrome V8 e seu *ECMAScript*, o que significa que a maior parte da sintaxe do Node.js é semelhante ao JavaScript *front end* (outra implementação do *ECMAScript*), incluindo objetos, funções e métodos. Ele é um executor de códigos JavaScript assíncrono orientado a eventos, sendo projetado para construir aplicativos de rede escalonáveis.

 Conforme Quadro 3, para Mardan (2018), os principais fundamentos do Node.js são:

**Quadro 4 – Fundamentos do Node.js**

|  |  |
| --- | --- |
| ***Loose typing*** | A tipificação automática funciona bem na maioria das vezes. É um ótimo recurso que economiza muito tempo e energia mental! Existem apenas alguns tipos de primitivos:• *String*• *Number* (inteiro e real)• *Boolean*• *Undefined*• *Null*Tudo o mais é um objeto. A classe é um objeto. A função é um objeto. *Array* é um objeto. *RegExp* é um objeto. Os objetos são passados ​​por referência, enquanto os primitivos são passados ​​por valores. |
| ***Buffer—Node.js super data type Object literal notation*** | *Buffer* é o tipo de dados. É uma adição Node.js a cinco primitivos (*boolean*, *string*, *number*, *undefined* e *null*) e objetos abrangentes (*arrays* e funções também são objetos) no JavaScript *front* *end*. Pense nos *buffers* como armazenamentos de dados extremamente eficientes. Na verdade, o Node.js tenta usar *buffers* sempre que pode, como ao ler de um sistema de arquivos e ao receber pacotes pela rede.O *buffer* é funcionalmente semelhante ao *ArrayBuffer* do JavaScript. Usa-se o nome de classe *Buffer* para trabalhar com objetos de *buffer*. |
| ***Functions Arrays*** | No Node.js (assim como no JavaScript), as funções são cidadãs de primeira classe e são tratadas como variáveis, porque são objetos! Sim, as funções podem até ter propriedades / atributos.*Arrays* também são objetos que possuem alguns métodos especiais herdados do *Array*. No entanto, os *arrays* JavaScript não são *arrays* reais; em vez disso, eles são objetos com chaves inteiras exclusivas (geralmente baseadas em 0). |
| ***Prototypal nature Conventions*** | Não há classes em JavaScript porque os objetos herdam diretamente de outros objetos, o que é chamado de herança prototípica. Existem alguns tipos de padrões de herança em JavaScript:• Clássico• Pseudoclássico• Funcional |

Fonte: Elaborado a partir de Mardan (2018)

Segundo Mardan (2018), cada script Node.js executado é, em essência, como um processo do sistema. Convenientemente, os desenvolvedores podem acessar informações úteis do processamento de código com o objeto do processo.

2.7 banco de dados

 Na atualidade o uso de bancos de dados está presente em quase todas as aplicações, pois é de extrema importância armazenar os dados de forma organizada e de acesso rápido. Para Coronel e Morris (2018), no mundo de hoje, os dados são onipresentes (abundantes, globais, em todos os lugares) e difundidos (inescapáveis, prevalentes, persistentes). Do nascimento à morte, gera-se e consume-se dados. A trilha de dados começa com a certidão de nascimento e continua até a certidão de óbito (e além!). Nesse meio tempo, cada indivíduo produz e consome enormes quantidades de dados. Os bancos de dados tornam os dados persistentes e compartilháveis de maneira segura.

 Continuam Coronel e Morris (2018), os dados não são apenas onipresentes e difundidos; também é essencial para as organizações sobreviverem e prosperarem. Imagine tentar operar um negócio sem saber quem são seus clientes, quais produtos você está vendendo, quem está trabalhando para você, quem lhe deve dinheiro e a quem você deve dinheiro. Todas as empresas precisam manter esse tipo de dados e muito mais. Tão importante quanto, eles devem ter esses dados disponíveis para as tomadas de decisões quando necessário. Pode-se argumentar que o objetivo final de todos os sistemas de informação de negócios é ajudar as empresas a usar a informação como um recurso organizacional. No centro de todos esses sistemas estão a coleta, armazenamento, agregação, manipulação, disseminação e gerenciamento de dados. Dependendo do tipo de sistema de informação e das características da empresa, esses dados podem variar de alguns *megabytes* a *terabytes* dentro do ambiente interno e externo da empresa.

Quando se trabalha com de bancos de dados é necessario entender a diferença entre dados e informações. Segundo Coronel e Morris (2018), os dados consistem em fatos brutos. A palavra bruto indica que os fatos ainda não foram processados para revelar seu significado. A informação é o resultado do processamento de dados brutos para revelar seu significado. O processamento de dados pode ser tão simples quanto organizar os dados para revelar padrões ou tão complexo quanto fazer previsões ou fazer inferências usando modelagem estatística. Para revelar o significado, a informação requer contexto. Por exemplo, uma leitura de temperatura média de 105 graus não significa muito, a menos que você também conheça seu contexto: essa leitura está em graus Fahrenheit ou Celsius? É a temperatura da máquina, do corpo ou do ar externo? As informações podem ser usadas como base para a tomada de decisões.

Com isso pode-se compreender a importância do banco de dados, pois sem ele armazenar os dados seria muito custoso e de difícil acesso, além disso, seria muito mais complexo transformar os dados em informações relevantes.

Para Coronel e Morris (2018), o gerenciamento eficiente de dados geralmente requer o uso de um banco de dados de computador. Um banco de dados é uma estrutura de computador integrada e compartilhada que armazena uma coleção dos seguintes:

* dados do usuário final, ou seja, fatos brutos de interesse do usuário final;
* metadados, ou dados sobre dados, por meio dos quais os dados do usuário final são integrados e gerenciados.

Continua Coronel e Morris (2018), os metadados descrevem as características dos dados e o conjunto de relacionamentos que vincula os dados encontrados no banco de dados. Por exemplo, o componente de metadados armazena informações como o nome de cada elemento de dados, o tipo de valores (numéricos, datas ou texto) armazenados em cada elemento de dados e se o elemento de dados pode ser deixado em branco. Os metadados fornecem informações que complementam e expandem o valor e o uso dos dados. Em suma, os metadados apresentam uma imagem mais completa dos dados no banco de dados. Dadas as características dos metadados, você pode ouvir que um banco de dados é descrito como uma “coleção de dados autoexplicativos”.

2.8 sistema de gerenciamento de banco de dados

O SGBD sigla referente à sistema de gerenciamento de banco de dados é um software responsável por intermediar a comunicação entre o usuário final e o banco de dados. Segundo Coronel e Morris (2018), um SGBD é uma coleção de programas que gerencia a estrutura do banco de dados e controla o acesso aos dados armazenados no banco de dados. Em certo sentido, um banco de dados se assemelha a um arquivo eletrônico muito bem-organizado no qual um software poderoso (o SGBD) ajuda a gerenciar o conteúdo do gabinete.

Continua Coronel e Morris (2018), o SGBD atua como intermediário entre o usuário e o banco de dados. A própria estrutura do banco de dados é armazenada como uma coleção de arquivos e a única maneira de acessar os dados nesses arquivos é por meio do SGBD. Ele recebe todas as solicitações de aplicativos e as traduz em operações complexas necessárias para atender a essas solicitações, ocultando grande parte da complexidade interna do banco de dados dos programas de aplicativos e usuários.

Finaliza Coronel e Morris (2018), ter um SGBD entre os aplicativos do usuário final e o banco de dados oferece algumas vantagens importantes. Primeiro, ele permite que os dados no banco de dados sejam compartilhados entre vários aplicativos ou usuários. Em segundo lugar, integra as muitas visualizações diferentes dos usuários dos dados em um único repositório de dados abrangente.

2.9 postgresql

O PostgreSQL é um software SGBD de bancos relacionais. Segundo Luzanov, Rogov e Levshin (2020), o PostgreSQL é o sistema de banco de dados de código aberto livre mais rico em recursos. Desenvolvido no ambiente acadêmico, ele reuniu uma ampla comunidade de desenvolvedores ao longo de sua longa história. Hoje em dia, o PostgreSQL oferece todas as funcionalidades exigidas pela maioria dos clientes e é usado ativamente em todo o mundo para criar sistemas críticos de negócios de alta carga. A confiabilidade é especialmente importante em aplicativos de nível corporativo que lidam com dados críticos para os negócios. Para isso, o PostgreSQL fornece suporte para servidores *hot standby*, recuperação *point-in-time*, diferentes tipos de replicação (síncrona, assíncrona e cascata).

Para Ferrari e Pirozzi (2020) o PostgreSQL é um conhecido banco de dados relacional de código aberto, e seu lema confirma o que o projeto pretende ser: o banco de dados de código aberto mais avançado do mundo. As principais qualidades que atraem massas de novos usuários a cada ano e mantêm os usuários atuais entusiasmados com seus projetos são sua estabilidade, escalabilidade e segurança sólidas, bem como os recursos que um sistema de gerenciamento de banco de dados de nível empresarial oferece.

Continua Ferrari e Pirozzi (2020), mas o PostgreSQL não é apenas um banco de dados; ele cresceu e se tornou um ecossistema completo de extensões, ferramentas e linguagens unidas por comunidades espalhadas por todo o mundo. PostgreSQL é um projeto de código aberto e totalmente desenvolvido no mundo do código aberto. Isso significa que não há uma única entidade responsável pelo projeto e o resultado é que o PostgreSQL não é um produto comercial. Em outras palavras, o PostgreSQL pertence a todos e qualquer um pode contribuir para isso. Graças a uma licença muito permissiva no estilo BSD (*Berkeley Source Distribution*), o PostgreSQL pode ser usado em qualquer projeto ou cenário, seja de código aberto ou fechado.

Por todos os pontos citados acima pode-se então concluir que o PostgreSQL é uma ferramenta completa para o controle de banco de dados relacional, sendo uma ferramenta de excelente qualidade, de código aberto e gratuita, sendo possível utilizá-la em qualquer projeto, independentemente do tamanho e da complexidade, o PostgreSQL vai atender e se adaptar as necessidades do projeto.

2.10 LINGUAGEM SQL

A linguagem SQL (*Structured Query Language*) é utilizada na comunicação com o SGBD.

Segundo Coronel e Morris (2018), uma linguagem de banco de dados permite que você crie estruturas de banco de dados e tabelas, execute tarefas básicas de gerenciamento de dados (adicionar, excluir e modificar) e realizar consultas complexas projetadas para transformar os dados brutos em informações úteis. Além disso, uma linguagem de banco de dados deve executar essas funções básicas com o mínimo de esforço do usuário, e sua estrutura de comando e sintaxe devem ser fáceis de aprender. Finalmente, deve ser portátil; ou seja, deve estar em conformidade com algum padrão básico, de modo que uma pessoa não precise reaprender o básico ao passar de um SGBD para outro. O SQL atende bem a esses requisitos ideais de linguagem de banco de dados.

Sobre as funções SQL continua Coronel e Morris (2018), categorizando-as em:

* Uma linguagem de manipulação de dados (DML). O SQL inclui comandos para inserir, atualizar, excluir e recuperar dados nas tabelas do banco de dados.
* Uma linguagem de definição de dados (DDL). O SQL inclui comandos para criar objetos de banco de dados, como tabelas, índices e visualizações, bem como comandos para definir direitos de acesso a esses objetos de banco de dados.
* Uma linguagem de controle de transação (TCL). Os comandos DML em SQL são executados no contexto de uma transação, que é uma unidade lógica de trabalho composta por uma ou mais instruções SQL, conforme definido pelas regras de negócios. O SQL fornece comandos para controlar o processamento dessas instruções em uma unidade indivisível de trabalho.
* Uma linguagem de controle de dados (DCL). Os comandos de controle de dados são usados ​​para controlar o acesso a objetos de dados, por exemplo, dar a um usuário permissão para visualizar apenas determinada tabela e dar a outro usuário permissão para alterar os dados desta mesma tabela.

Por todos os pontos destacados acima pode-se concluir que o SQL é uma linguagem completa para manipular dados e estrutura em um banco de dados.

3. Metodologia DA PESQUISA

O presente trabalho tem por método a pesquisa aplicada e descritiva, pois tem como objetivo desenvolver uma ferramenta para monitoramento de bancos de dados PostgreSQL. Esta ferramenta possui o objetivo de monitorar, conexões, tabelas e recursos utilizados pelo banco de dados, para solucionar o seguinte problema: Como disponibilizar acesso remoto e facilitado aos principais indicadores de monitoramento no SGBD PostgreSQL de forma pró-ativa e com isso ajudar na identificação de possíveis anomalias antecipando ações administrativas?

O trabalho terá como procedimentos a pesquisa documental e o estudo de caso, visto que foram analisados os recursos utilizados pelo PostgreSQL e os pontos críticos do uso do SGBD.

A pesquisa foi realizada da seguinte maneira: foram buscadas informações documentais através de livros e documentação de software, aplicando estas informações no desenvolvimento da ferramenta. Será utilizado para desenvolvimento do *front end* o HTML, CSS e Javascript, como facilitadores o Bootstrap, Jquery, Charts e Datatables. No *back end* será utilizado Node.js com Express, a interface com o SGBD será feita com o pg-promise e a criptografia dos dados de conexão será feito com o crypto-js.

3.1 ESTADO DA ARTE

 Atualmente existem algumas ferramentas de monitoramento para bancos de dados PostgreSQL e em sua grande maioria são pagas, porém bem completas, normalmente focadas no DBA. Para critério de análise, a seguir, demonstrarei a plataforma Datadog (paga) e o monitor embutido no pgAdmin (gratuito) com seus pontos positivos e negativos.

3.1.1 Datadog

O Datadog é uma plataforma de monitoramento que possui integração com inúmeros sistemas, aplicações e serviços, com o PostgreSQL sendo um deles. Ele conta somente com planos pagos possuindo uma versão de testes que dura 14 dias. Nesse plano o contratante adiciona somente os módulos que o interessam, pois cada modulo tem uma cobrança diferente.

Conforme figura 1, para realizar o monitoramento do banco de dados PostgreSQL é necessário que seja instalado e configurado um agente no computador. Este agente será o responsável por enviar as informações para a aplicação web.

 **Figura 1 – Instalador do agente Datadog.**



 Fonte: Acervo do autor

Depois de instalado o agente, deve-se realizar a configuração do banco de dados, que é feito via arquivo. É neste arquivo também que serão informadas as configurações de como será realizado o monitoramento, por exemplo, o tempo limite para que uma *query* seja considerada lenta. Na figura 2 pode-se ter clareza desta configuração.

 **Figura 2 – Arquivo de configuração do módulo PostgreSQL.**



 Fonte: Acervo do autor

Após efetuada as configurações é possível ter acesso as estáticas na página web, onde é possível configurar alertas, *dashboards* e métricas personalizados além dos pré-existentes, conforme figura 3.

**Figura 3 – Exemplo de *Dashboard*.**



Fonte: Acervo do autor

Após demonstrado como é realizado a configuração e uso do software pode-se demonstrar seus pontos positivos e negativos.

A plataforma é bem completa, contando com uma infinidade de métricas relacionadas a banco de dados, podendo ainda personalizar seus *dashboards*. É possível criar alertas, por exemplo, caso o uso de conexões do banco de dados passe de uma porcentagem pré-estabelecida será enviado um e-mail de alerta. A plataforma gera também um relatório com os comandos SQL que foram executados no período, com o tempo total de execução e número de vezes executadas.

Os pontos negativos ficam por conta da ferramenta e dos manuais não serem disponibilizados em português, sendo seus únicos idiomas suportados o inglês, francês e japonês, por conta disso há uma dificuldade na configuração além do passo-a-passo que é confuso. A plataforma perde a praticidade quando é necessário instalar o agente de conexão com o SGBD. Os planos são pagos em dólar o que torna a mensalidade muito mais cara e variável. O suporte é prestado nos idiomas citados acima, ou seja, se torna mais difícil obter informações e auxílios.

Como consideração final, creio que a plataforma é muito útil para empresas já consolidadas que queiram a obtenção de informações personalizadas em um único local, e para isso devem possuir um profissional com conhecimentos específicos em inglês e sobre banco de dados PostgreSQL que possa efetuar a configuração e personalização dos *dashboards* e alertas.

3.1.2 pgAdmin

O pgAdmin é uma ferramenta para administração e desenvolvimento de bancos de dados PostgreSQL, nele há um *dashboard* simples para auxiliar o desenvolvedor e o DBA. Neste *dashboard* se encontra basicamente informações sobre transações e linhas movimentadas. O pgAdmin é uma ferramenta gratuita e de código aberto disponível para várias plataformas.

Conforme figura 4, pode-se verificar que as informações disponibilizadas são simples e diretas, com um pequeno histórico, porém sem a possibilidade de alterar parâmetros de exibição.

**Figura 4 –*Dashboard* do pgAdmin.**



Fonte: Acervo do autor

Conforme as informações apresentadas pode-se considerar como pontos positivos da ferramenta a comodidade para o desenvolvedor ou DBA já ter um *dashboard* vinculado, podendo visualizar métricas básicas sem a necessidade de ferramentas externas. Como pontos negativos tem-se que são métricas muito básicas que não retornam todas as informações necessárias para o uso no dia a dia. Como seu dashboard é vinculado a uma ferramenta que permite execução de SQL isso se torna um ponto positivo e negativo ao mesmo tempo, por facilitar o uso para desenvolvedores e por não poder bloquear a função para pessoas leigas de uma forma fácil.

4. POSTGRECONTROL - SOFTWARE WEB PARA MONITORAMENTO DE BANCOS DE DADOS POSTGRESQL

Neste tópico serão abordadas as funcionalidades que o protótipo de software deve possuir e a forma como ele deverá funcionar. Para facilitar o entendimento, foram criadas seções com enfoque nas regras de negócio, requisitos funcionais, não funcionais, diagramas e uma apresentação detalhada sobre o desenvolvimento e uso do software.

4.1 visão geral do protótipo de software

O protótipo tem por objetivo principal facilitar o dia a dia do DBA e Administradores de Tecnologia trazendo métricas importantes para o monitoramento de bancos de dados PostgreSQL. As métricas fornecidas pelo “PostgreControl” junto com o conhecimento do DBA permitem que ações possam ser tomadas com um tempo de resposta menor. Por outro lado, elas também auxiliam o Administrador de Tecnologia, uma vez que são apresentados dados vitais resumidos e de fácil leitura, permitindo detectar o estado dos principais elementos do SGBD.

 O “PostgreControl” pode ser utilizado por todos que estão à procura de praticidade e que queiram métricas rápidas, sem a necessidade de uma configuração complexa, apenas informando os dados de conexão com o banco pode-se ter acesso a todas estas métricas de uma maneira simples.

 Ele é web e responsivo, o que permite maior portabilidade, podendo ser utilizado em celulares, tablets e computadores. Nesta primeira versão possui métricas em tempo real e conexão com apenas um banco de dados por vez, onde a configuração de conexão depois de feita será mantida para as demais sessões de uso.

 Para a comunicação foi adotado a arquitetura baseada em API, onde o Express oferece sistema de roteamento. Com este funcionamento há um desacoplamento entre *front end* e *back end*, permitindo com que haja maior liberdade em versionamento com novas funções ou alterações.

 A comunicação utilizada com o banco de dados procura ser rápida e direta, não mantendo conexões abertas sem uso. Os comandos SQL das consultas foram desenvolvidos para que utilizem o mínimo de recurso possível, utilizando de métricas e funções disponibilizadas pelo próprio SGBD. Estas métricas podem ser reiniciadas a qualquer momento pelo DBA diretamente pelo SGBD.

4.2 regras de negócio

As regras de negócio são responsáveis por elencar o modo de fazer, apontando o passo a passo necessário para execução de tarefas, governança do negócio e ideologias de decisão. Segundo Coronel e Morris (2018) uma regra de negócios é uma descrição breve, precisa e inequívoca de uma política, procedimento ou princípio dentro de uma organização específica.

 Conforme o quadro a seguir pode-se ver as principais regras de negócio presentes nesse projeto.

**Quadro 5 – Regras de negócio.**

|  |  |
| --- | --- |
| RN01 | O protótipo deverá permitir registrar apenas um servidor de banco de dados, podendo ser alterado conforme necessidade. |
| RN02 | Os dados devem ser obtidos em tempo real, com intervalo variável entre consultas. O intervalo é para que não sobrecarregue o banco de dados. Com a obtenção dos dados em tempo real permite que ações sejam mais eficazes. |
| RN03 | O autovacuum é utilizado pelo PostgreSQL para otimizar as buscas. Ele é iniciado automaticamente e é necessário quando a quantidade de linhas mortas na tabela for maior do que a configuração estipulada. |
| RN04 | Uma linha morta é gerada quando uma linha é alterada ou deletada, sendo assim marcada como inútil. |
| RN05 | Deadlock acontece quando há uma dependência entre duas consultas onde uma depende da outra causando assim um *loop*. |
| RN06 | Temp File acontece quando uma consulta ocupa mais memória do que foi configurado, sendo necessário utilizar o disco para armazenar o excedente. |
| RN07 | Saúde do Banco é uma métrica que mede como está o banco de dados, e nela é utilizado um cálculo para efetuar a classificação. Para encaixar como crítico é necessário que haja mais que um deadlock por hora ou haja mais que um temp file por hora. Para encaixar em alerta é necessário que haja menos que um deadlock por hora ou haja menos que um temp file por hora. Para encaixar em normal é necessário que não haja deadlock e temp file. |
| RN08 | Status é uma métrica utilizada para medir como estão as tabelas do banco de dados individualmente, e nela é utilizado um cálculo para efetuar a classificação. Para encaixar como crítico é necessário que a tabela ocupe um tamanho maior que 25 por cento do total ocupado pelo banco de dados ou que a tabela tenha mais que 25 por cento dos autovacuum executados no banco de dados. Para encaixar como alerta é necessário que a tabela ocupe um tamanho entre 10 e 25 por cento do total ocupado pelo banco de dados ou que a tabela tenha entre 10 e 25 por cento dos autovacuum executados no banco de dados. Para encaixar em normal é necessário que a tabela ocupe menos que 10 por cento do tamanho total do banco de dados e tenha menos que 10 por cento dos autovacuum executados no banco. |

Fonte: Acervo do Autor

4.3 requisitos

Os requisitos são as funcionalidades, exigências, utilidades e objetivos que o software precisa cumprir, de acordo com as necessidades empregadas pela empresa e usuários.

Os requisitos recebidos do cliente são escritos em linguagem natural. É responsabilidade do analista de sistema documentar os requisitos em linguagem técnica para que possam ser compreendidos e úteis pela equipe de desenvolvimento de software.

4.3.1 Requisitos não funcionais

Os requisitos não funcionais são aqueles que não estão relacionados explicitamente ao aspecto funcional do software, eles são características implícitas ou esperadas, que os usuários não presumem. Nesta categoria de requisitos estão presentes a segurança, armazenamento, configuração, performance, custo, flexibilidade, acessibilidade, entre outros. Segundo Sommerville (2016) os requisitos não funcionais, como o nome sugere, são requisitos que não estão diretamente relacionados com os serviços específicos entregues pelo sistema a seus usuários. Esses requisitos não funcionais geralmente especificam ou restringem as características do sistema como um todo.

Conforme o quadro a seguir pode-se ver os principais requisitos não funcionais presentes nesse projeto.

**Quadro 6 – Requisitos não funcionais.**

|  |  |
| --- | --- |
| RNF01 | As informações de conexão com o banco de dados devem ser criptografadas e gravadas em arquivo. |
| RNF02 | Deve possuir uma paleta de cores reduzida, procurando manter o foco nas informações disponibilizadas. |
| RNF03 | A linguagem empregada deve ser clara e descomplicada. |
| RNF04 | Deve focar em economizar recursos do banco de dados. |
| RNF05 | Quanto a configuração do banco de dados deve ser direto e descomplicado. |
| RNF06 | Deve permitir registrar e conectar apenas em bancos de dados PostgreSQL. |
| RNF07 | Deve manter um padrão visual harmônico, com interfaces semelhantes para facilitar o aprendizado |
| RNF08 | O usuário poderá filtrar dados presentes nas grades a fim de facilitar a obtenção de informações |
| RNF09 | O usuário poderá escolher o número de registro visíveis nas tabelas, entre 10, 25, 50 e 100 registros por página. |
| RNF10 | As cores das colunas de dados vitais ao monitoramento, devem obedecer a lista de cores primárias, verde, vermelho e amarelo |

Fonte: Acervo do Autor

4.3.2 Requisitos funcionais

Os requisitos funcionais são aqueles que estão relacionados diretamente ao aspecto funcional do software. Eles definem e descrevem detalhadamente toda e qualquer função e funcionalidade presente no software. Segundo Sommerville (2016) os requisitos funcionais de um sistema descrevem o que o sistema deve fazer. Esses requisitos dependem do tipo de software que está sendo desenvolvido, dos usuários esperados do software e da abordagem geral adotada pela organização ao escrever os requisitos.

Conforme o quadro a seguir pode-se ver os principais requisitos funcionais presentes nesse projeto.

**Quadro 7 – Requisitos funcionais.**

|  |  |
| --- | --- |
| RF01 | Possuir uma página inicial com informações do tempo de atualização dos dados nas páginas de consulta e possuir links para sites que possam auxiliar o usuário, sendo eles, documentação oficial do PostgreSQL, wiki do PostgreSQL e Stackoverflow. |
| RF02 | Possuir uma página para efetuar o cadastro do banco de dados, com o logo do protótipo, com os campos IP, porta, nome do banco, usuário, senha e SSL e com um botão para gravar. |
| RF03 | Na tela de cadastro do banco de dados abaixo do botão gravar retornar uma mensagem de validação das informações inseridas. |
| RF04 | Possuir uma página com informações gerais do banco de dados. Nela possuir o nome do banco, tamanho total em disco, total de conexões ativas e data do último reset do status. Possuir uma tabela com o percentual de commit, percentual de rollback, deadlocks, deadlocks por hora, temp files, temp files por hora e saúde do banco. Possuir um gráfico do tipo pizza com o percentual de linhas, inseridas, alteradas, deletadas e selecionadas. |
| RF05 | Possuir uma página com informações das tabelas do banco. Nela possuir uma tabela com o status, schema, nome da tabela, busca sequencial, busca indexada, linhas inseridas, linhas alteradas, linhas deletadas, linhas uteis, linhas mortas, total de linhas, contador de autovacuum, data do último autovacuum, dono e tamanho total. Possuir um gráfico com as 5 tabelas com maior quantidade de autovacuum, um gráfico com as 5 tabelas com a maior quantidade de linhas mortas e um gráfico com as 5 maiores tabelas. |
| RF06 | Possuir uma página com informações das conexões ativas com o banco. Nela possuir uma tabela com o PID, estado, usuário, nome de aplicação, data e hora de início, data e hora de alteração, PID lock e operação. |

Fonte: Acervo do Autor

4.4 diagramas

 O diagrama de caso de uso tem por finalidade tornar visível as interações feitas pelos atores com o sistema, mostrando o passo a passo das ações tomadas para realização das funções.

No “PostgreControl” há apenas um ator chamado de usuário, este por sua vez é responsável pela utilização total do protótipo. O usuário é a parte interessada pelo monitoramento do banco de dados inserido.

Conforme figura 5, pode-se visualizar os casos de uso disponiveis, sendo eles o cadastro do banco de dados que será alvo do monitoramento, o acesso ao painel de dados gerais, o acesso ao painel de informações sobre as tabelas e o acesso ao painel de conexões. Todas estas ações são independentes e podem ser acessadas a qualquer momento.

O caso de uso referente ao cadastro do banco de dados deve ser feito antes de acessar os paineis no primeiro acesso ao sistema, pois é destes dados que serão obtidas as metricas. Depois de informado somente é necessario acessar esta rotina caso o usuario queira alterar alguma informação.

**Figura 5 – Diagrama de caso de uso completo.**



Fonte: Acervo do autor

4.5 implementação

 A implementação foi realizada com o foco em tecnologias e ferramentas para web conforme mencionadas anteriormente. O grande foco foi no uso da linguagem Javascript, que foi utilizada tanto no *front end* com a linguagem base quanto no *back end* através do framework Node.js.

4.5.1 *Front end*

 No *front end* foi utilizado o HTML como base para exibição das informações, sendo responsável pela estrutura visual do projeto.

Sobre este conteúdo gerado pelo HTML foi aplicado o CSS estilizando as informações e com isto tornando-as mais visíveis e elegantes. Foi utilizado para isto a linguagem base responsável por pequenos detalhes e alterações especificas, a *framework Bootstrap* responsável por trazer padrões de estilo pré-configurados, sendo necessário somente a chamada destas classes, a biblioteca *Charts* responsável pelo desenho dos gráficos e a biblioteca *Datatables* com a função de melhorar a exibição das informações presentes nas tabelas.

Para o preenchimento das informações nos gráficos e tabelas e realizar a integração do *front end* com o *back end* foi utilizado o Javascript e com ele foram criadas as funções responsáveis por estas ações. Para inserir, testar e utilizar um banco de dados foram criadas as três funções citadas na figura 6, onde todas se comunicam com o *back end* através de rotas.

 **Figura 6 – Funções do *front end* de conexão com o banco de dados.**



 Fonte: Acervo do autor

 Para atualizar as informações presentes nos gráficos gerados pelo Charts foi também utilizado Javascript, porém com funções próprias desta biblioteca. Foi criado um *timeout* para que seja chamado a função base e assim manter as informações atualizadas. O código utilizado para gerar e atualizar um gráfico pizza com as informações dos tipos de movimentação das linhas do banco de dados é visualizado na figura 7.

**Figura 7 – Função do *front end* para gerar gráfico pizza de linhas movimentadas.**



Fonte: Acervo do autor

 Da mesma forma que os gráficos, as tabelas do Datatables são geradas e atualizadas por funções próprias. O *timeout* foi também utilizado para gerar a atualização automática das informações. Conforme figura 8, pode-se identificar a criação e atualização da tabela de conexões com o banco de dados.

**Figura 8 – Função do *front end* para gerar tabela das conexões com o banco de dados.**



Fonte: Acervo do autor

4.5.2 *Back end*

 No *back end* foi utilizado o Node.js, responsável por fornecer a base para a utilização do *framework* Express, onde foram construídas as rotas responsáveis pelas funções do protótipo. Conforme figura 9 podes-se verificar a lista de rotas disponíveis com suas dependências.

**Figura 9 – Lista de rotas do *back end*.**



Fonte: Acervo do autor

 Para a rota registrar que é responsável pelo cadastro do banco de dados foi utilizado os parâmetros passados pelo *front end* no formato *Json*. Estes parâmetros são tratados e utilizados para realizar um teste de conexão, caso a conexão seja feita com sucesso estes dados são criptografados e gravados em disco e caso ocorra algum erro na tentativa de conexão é retornado para o *front end* a mensagem retornada pelo SGBD. Na figura 10 pode-se verificar a função comentada acima.

**Figura 10 – Função cadastro.**



Fonte: Acervo do autor

 Para tornar ilegíveis os dados gravados em disco e com isso tornar o sistema mais seguro foi utilizado a biblioteca crypto-js. Para isto foram criadas duas funções, uma para encriptar os dados gravados em disco, ou seja, tornar os dados ilegíveis e outra para decriptar os dados recuperados do disco, ou seja, tornar os dados legíveis novamente em tempo de execução. Conforme figura 11 pode-se ver o funcionamento destas funções com o tipo de criptografia utilizado, o conteúdo a ser manipulado e a palavra-chave utilizada para criptografia.

**Figura 11 – Funções para criptografia dos dados.**

****

Fonte: Acervo do autor

 Toda a comunicação com o banco de dados foi feita através da biblioteca pg-promise que além de realizar esta ponte também cria uma *promise* tornando assim possível a execução de funções assíncronas. Todas as execuções dos comandos SQL foram feitas utilizando esta biblioteca através da função representada na figura 12.

**Figura 12 – Funções para execução de comandos SQL.**



Fonte: Acervo do autor

4.6 funcionamento

Neste capítulo será apresentada a estrutura do protótipo englobando as funções disponíveis, detalhando a utilização e suas principais características de uma forma visual.

Todas as rotinas desenvolvidas foram pensadas para retornar informações importantes sobre o banco de dados e métricas de uma forma visual, com isso direcionar a atenção do usuário para o que realmente necessita atenção. Pois ele permite também a utilização por um grupo de usuários que não possuem larga experiencia sobre banco de dados, mas necessitam monitorar suas principais estatísticas, como os Administradores de Tecnologia.

4.6.1 Visãogeral

 Conforme figura 13, ao acessar o protótipo visualiza-se a página principal, que possui uma frase de apresentação, alguns botões de encaminhamento para páginas sobre PostgreSQL, sendo elas a documentação oficial, a wiki oficial e a página sobre PostgreSQL do Stack Overflow e por último o tempo entre consultas das páginas de monitoramento.

**Figura 13 – Página principal.**



Fonte: Acervo do autor

4.6.2 Cadastro do banco de dados

O cadastro do banco de dados deve ser feito somente no primeiro acesso pois o protótipo grava em disco (no servidor de aplicação) as informações inseridas, sendo necessário novo cadastro somente se as informações de conexão forem alteradas ou se o monitoramento será feito sobre outro banco de dados. Ele permite conectar em qualquer banco de dados PostgreSQL local e remoto acima da versão 9.6, suportando também bancos que utilizem SSL na conexão, conforme figura 14.

**Figura 14 – Página de cadastro.**

****

Fonte: Acervo do autor

Após informar todos os dados de conexão e pressionar o botão gravar será efetuado o teste das informações inseridas e caso tenha sido possível efetuar a conexão será retornado uma mensagem de sucesso redirecionando o usuário automaticamente para a página principal, caso ocorra qualquer erro será retornado em forma de mensagem na tela e não será feito o redirecionamento.

4.6.3 Monitoramento geral

Conforme figura 15, o protótipo possui uma página para informações gerais e nela estão presentes o nome do banco de dados, o tamanho total, o total de conexões e a data da última redefinição do status. Nela possui também uma tabela com informações estatísticas referente a quantidade de *commit* e *rollback,* quantidade de *deadlock,* quantidade de arquivos temporários além da métrica saúde do banco que é classificada em normal representada pela cor verde, alerta pela cor amarela e crítico pela cor vermelha utilizando como parâmetros a quantidade de *deadlock* e a quantidade de arquivos temporários, conforme descrito na regra de negócio RN07, isso é um facilitador que permite rápida interpretação pelo usuário. A página possui também um gráfico do tipo pizza que tem como informação a porcentagem de linhas inseridas, atualizadas, deletadas e retornadas.

**Figura 15 – Monitoramento geral do banco de dados.**



Fonte: Acervo do autor

4.6.4 Monitoramento de tabela

O protótipo possui uma página para monitoramento das tabelas conforme figura 16. Nela estão presentes as informações gerais de nome, quantidade de linhas, quantidade de buscas sequenciais e por índice, quantidade e último autovacuum, tamanho total além da métrica status que pode ser classificada em normal representada pela cor verde, alerta pela cor amarela e crítico pela cor vermelha, sendo utilizado como parâmetros o tamanho da tabela em relação com o tamanho total do banco de dados e a quantidade de autovacuum da tabela em relação a quantidade total de autovacuum executados no banco, conforme descrito na regra e negócio RN08. Também é um facilitador, que permite rápida detecção de anomalias, mesmo por usuários que não possuem alto nível de conhecimento no SGBD.

**Figura 16 – Monitoramento geral das tabelas.**



Fonte: Acervo do autor

 Além do monitoramento geral das tabelas, a página possui três gráficos de barras, conforme o exemplo da figura 17. Estes gráficos são responsáveis por alertar de forma visual o usuário, sendo que eles retornam as cinco tabelas com maior número de autovacuum, as cinco tabelas com o maior número de linhas mortas e as cinco maiores tabelas do banco de dados.

 **Figura 17 – Exemplo de gráfico de barras.**



 Fonte: Acervo do autor

4.6.5 Monitoramento de conexão

 Conforme a figura 18 o protótipo possui uma página para monitoramento das conexões abertas com o banco de dados, e nela possui uma tabela com o PID da conexão, o estado, o usuário, o nome da aplicação, a data de início, a data de alteração, a operação sendo executada e a métrica PID bloqueando que traz na conexão bloqueada o PID da conexão que está bloqueando. Quando houver algum bloqueio o mesmo é exibido de forma facilitada, com fundo destacado na cor vermelha.

**Figura 18 – Monitoramento das conexões.**



Fonte: Acervo do autor

5. CONSIDERAÇÕES FINAIS

 O protótipo apresentado tem por finalidade auxiliar no dia a dia do administrador de banco de dados PostgreSQL, seja ele DBA ou Administrador de Tecnologia. Levando esta finalidade em consideração pode-se concluir que as funcionalidades desenvolvidas auxiliam na tomada de decisão e diminuem o tempo entre perceber o problema e tomar uma ação preventiva ou até mesmo corretiva.

 As métricas presentes conseguem gerar um fácil entendimento principalmente com os indicadores de criticidade que para os administradores com um conhecimento raso sobre banco de dados e sua manutenção, consigam visualizar e alocar recursos somente quando há algo fora do normal.

 As ferramentas utilizadas supriram todas as necessidades com isto todos os recursos pensados puderam ser desenvolvidos sem nenhum problema. As ferramentas utilizadas no *front end* em conjunto obtiveram um resultado satisfatório sendo possível dispor o design e as funcionalidade almejadas com ênfase no Javascript, que fez toda a conexão com o *back end* e carregamento dos gráficos e tabelas. As ferramentas utilizadas no *back end* tornaram simples a criação de todos os recursos, pois como foi utilizado um *framework* a dificuldade foi menor.

 O objetivo principal proposto no trabalho foi alcançado, pois com o desenvolvimento finalizado a ferramenta também conhecida como “PostgreControl” disponibiliza recursos importantes para um administrador que queira monitorar seu banco de dados de uma maneira fácil e rápida.

Os objetivos específicos permearam o desenvolvimento do trabalho, onde um a um foram sendo concluídos. Para isto foram levantados os principais itens de monitoramento do banco de dados PostgreSQL, com base nestes itens foram descritos os recursos, estes então foram desenvolvidos e por fim todos os recursos disponibilizados no protótipo foram finalizados e testados, sendo documentados neste trabalho.

Observa-se com base no estado da arte que o “PostgreControl”, figura entre a solução DataDog (focada para DBAs) e o PgAdmin (focada em desenvolvedores), mesmo que tanto DBA, como Desenvolvedor e Administradores de TIC, possam utilizar-se de ambas as ferramentas.

Posso então concluir que este trabalho foi desafiador, sendo realizado com todos os objetivos alcançados. Como o intuito do trabalho foi elencar os principais itens de monitoramento, a seguir são descritas recomendações futuras de funções interessantes para serem desenvolvidas tornando-o cada vez mais completo.

5.1 RECOMENDAÇÕES DE TRABALHOS FUTUROS

Para a continuidade do protótipo existem algumas implementações de novas funcionalidades que podem ser feitas, com a finalidade de tornar o projeto cada vez mais completo.

Ter a opção de gravar mais bancos de dados e criar assim uma lista onde o usuário pudesse alternar entre os bancos de forma mais rápida, sem a necessidade de cadastramento ao realizar esta troca.

Disponibilizar métricas com histórico, utilizando o próprio banco de dados para armazenar estas informações, em um schema novo criado automaticamente pela ferramenta. Com isto conseguiria trazer novos gráficos e métricas, além do fato de poder analisar métricas passadas.

Desenvolver um analisador de query, retornando o número de execuções e o tempo total dispendido. Com isto seria possível identificar se há alguma query lenta ou que é chamada muitas vezes.

Ter um executor de SQL embutido, tornando possível executar comandos SQL diretamente pela ferramenta, com isso facilitando a tomada de decisões descobertas na própria ferramenta.

**REFERÊNCIAS**

CORONEL, Carlos; MORRIS, Steven. ***Database Systems:*** *design, implementation, and management. 13*. ed. Massachusetts: Cengage, 2018.

DATADOG. ***Datadog***. Disponível em: < https://www.datadoghq.com >. Acesso em: 28 agosto 2021, 12:00:00.

ELLIOTT, Eric. ***Programming JavaScript Applications***. 1. ed. Sebastopol: O’Reilly Media, 2014.

FERRARI, Luca; PIROZZI, Enrico. ***Learn PostgreSQL:*** *build and manage high-performance database solutions using PostgreSQL 12 and 13*. 1. ed. Birmingham: Packt, 2020.

LUZANOV, Pavel; ROGOV, Egor; LEVSHIN, Igor. ***Postgres:*** *the first experience*. 6. ed. Moscow: Postgres Professional, 2020.

MARDAN, Azat. ***Practical Node.js:*** *Building Real-World Scalable Web Apps*. 2. ed. San Francisco: Apress, 2018.

MATTHEW, Neil; STONES, Richard. ***Beginning Databases with PostgreSQL:*** *from novice to professional*. 2. ed. New York: Apress, 2005.

ROBBINS, Jennifer Niederst. ***Learning Web Design:*** *a beginner's guide to HTML, CSS, JavaScript, and web graphics*. 5. ed. Sebastopol: O’Reilly Media, 2018.

SCHÖNIG, Hans-Jürgen. ***Mastering PostgreSQL 13:*** *build, administer, and maintain database applications efficiently with PostgreSQL 13*. 4. ed. Birmingham: Packt, 2020.

SILVEIRA, Paulo; ALMEIDA, Adriano. **Lógica de Programação:** crie seus primeiros programas usando JavaScript e HTML. 1. ed. São Paulo: Casa do Código, 2013.

SOMMERVILLE, Ian. **Software Enginnering**. 10. ed. Harlow: Pearson, 2016.

UBELHOR, Laura; HUR, Christian. ***Developing Business Applications for the Web:*** *with HTML, CSS, JSP, PHP, ASP.NET, and JavaScript*. 1. ed. Boise: Mc Press, 2017.

W3SCHOOLS. ***HTML Tutorial***. Disponível em: < https://www.w3schools.com/html/ >. Acesso em: 20 junho 2021, 11:00:00.

W3SCHOOLS. ***CSS Tutorial***. Disponível em: < https://www.w3schools.com/css/ >. Acesso em: 20 junho 2021, 12:00:00.

W3SCHOOLS. ***JavaScript Tutorial***. Disponível em: < https://www.w3schools.com/js/ >. Acesso em: 20 junho 2021, 14:00:00.

WHITE BELT MASTERY. ***Web Development for Beginners:*** *learn HTML/CSS/JavaScript step by step with this coding guide, programming guide for beginners, website development*. 1. ed. Seattle: KDP, 2020.