

CENTRO PAULA SOUZA
FACULDADE DE TECNOLOGIA DE FRANCA
“Dr. THOMAZ NOVELINO”

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CLEBER REZENDE SPIRLANDELI

TÍTULO DO TG

Subtítulo (se necessário)

FRANCA/SP

2017

CLEBER REZENDE SPIRLANDELI

TÍTULO DO TG

Trabalho de Graduação apresentado à Faculdade de Tecnologia de Franca - “Dr. Thomaz Novelino”, como parte dos requisitos obrigatórios para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Carlos Eduardo F. Roland

FRANCA/SP

2017

CLEBER REZENDE SPIRLANDELI

TÍTULO

Trabalho de Graduação apresentado à Faculdade de Tecnologia de Franca – “Dr. Thomaz Novelino”, como parte dos requisitos obrigatórios para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Trabalho avaliado e aprovado pela seguinte Banca Examinadora:

Orientador(a) : _____
Nome..... : Carlos Eduardo F. Roland
Instituição : Faculdade de Tecnologia de Franca – “Dr. Thomaz Novelino”

Examinador(a) 1 : _____
Nome..... : Examinador_1
Instituição : Instituição_1

Examinador(a) 2 : _____
Nome..... : Examinador_2
Instituição : Instituição_2

Franca, XX de dezembro de 2016.

AGRADECIMENTO (OPCIONAL)

Inicio meus agradecimentos para todos da minha família que me apoiaram desde o início do curso, agradeço também a todos os professores da Fatec Franca.

Por todos os ensinamentos, dedicação, motivação e a amizade que se formou ao longo de todos esses anos, o meu sincero e carinhoso muito obrigado!

DEDICATÓRIA (OPCIONAL – não tem este título)

Xxxxxxxxxxxxxx

EPIGRAFE (OPCIONAL - não tem este título)

O vídeo fornece uma maneira poderosa de ajudá-lo a provar seu argumento.

RESUMO

Palavras-chave:

ABSTRACT

Keywords:

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

ANP – Agência Nacional do Petróleo, Gás natural e Biocombustível

IHC – Interação Humano Computador

ÚNICA – União da Indústria da Cana-de-Açúcar

SUMÁRIO

1 INTRODUÇÃO	14
2 SISTEMAS DE INFORMAÇÃO	15
2.1 FUNDAMENTOS DE SISTEMAS DE INFORMAÇÃO	15
2.2 TIPOS DE SISTEMAS DE INFORMAÇÃO	16
3 GESTÃO DE PROCESSOS EMPRESARIAIS	31
3.1 COMPREENDENDO A ORGANIZAÇÃO E SEUS PROCESSOS	31
3.1.1 Processos gerenciais	33
3.2 ATIVIDADES E PROCESSOS LOGÍSTICOS	37
3.3 GESTÃO INTEGRADA DE PROCESSOS	42
4 GESTÃO EMPRESARIAL INTEGRADA	45
4.1 ESTRATÉGIAS DE GESTÃO DA INFORMAÇÃO	47
4.1.1 Deficiências de sistemas de informação no âmbito empresarial	52
4.2 VISÃO ESTRATÉGICA PARA O SUCESSO EMPRESARIAL	58
CONSIDERAÇÕES FINAIS	62
REFERÊNCIAS	64

1 INTRODUÇÃO

Introdução – introdução – introdução – introdução introdução – introdução –
introdução introdução – introdução – introdução introdução – introdução – introdução
introdução – introdução – introdução introdução – introdução – introdução introdução
– introdução – introdução introdução – introdução – introdução introdução –
introdução – introdução introdução – introdução – introdução introdução – introdução
– introdução introdução – introdução – introdução introdução – introdução –
introdução introdução – introdução – introdução introdução – introdução – introdução
introdução – introdução – introdução introdução – introdução – introdução introdução
– introdução – introdução introdução – introdução – introdução.

2. SISTEMAS DE INFORMAÇÃO

Fazer uma breve explicação do conteúdo do capítulo antes de inicia-lo.

2.1 ÁREA DE NEGOCIO

Inicio o tema falando sobre o transporte da população, voltado especialmente para a expansão de veículos para transporte pessoal, nos dias atuais, o sonho de se adquirir o transporte pessoal é cada vez maior e mais acessível, e com isso, a frequência a um posto de combustível passou a ser atividade essencial na vida das pessoas. Para grande parte da população, ir abastecer um veículo automotor se tornou cotidiano, levando em consideração, que é um público consumidor, e esse público consumidor é heterogêneo e encontrasse em todas as classes sociais, sexos, faixas etárias e toda a sociedade.

Os postos de combustíveis estão aumentando seu portfólio trazendo mais opções para o consumidor, como alguns tipos de gasolina (comum, aditivada, premium), álcool (comum e aditivado), diesel (S10 e S50), GNV (gás natural veicular) e aumentando também as opções nas lojas de conveniência, incluindo cafés da manhã, refeições e bebidas de várias marcas e modelos.

Existem algumas preocupações em relação ao meio ambiente, ligada aos combustíveis, as políticas federais dão preferência ao etanol, por ser uma opção nacional e que também é considerada a menos poluente atualmente.

2.2 COMÉRCIO DE COMBUSTÍVEIS

O mercado de combustíveis no Brasil é regulamentado pela lei federal 9.478/97:

Lei federal 9.478/97: Dispõe sobre a política energética nacional, as atividades relativas ao monopólio do petróleo, institui o Conselho Nacional de Política Energética e a Agência Nacional do Petróleo e dá outras providências.

Segundo informações da Agência Nacional do Petróleo, Gás Natural e Biocombustível (ANP), foi consumido 136,210 bilhões de litros de combustíveis no ano de 2013.

Em 2013 houve um aumento de 9,8% na venda de álcool hidratado, em relação a 2012, chegando a 10,816 bilhões de litros vendidos, segundo a União da Indústria da Cana-de-Açúcar (UNICA). Para a ANP, esse aumento nas vendas do etanol foi um reflexo referente à gasolina, que hoje esta passando da casa dos R\$3,00 o litro, em grande parte do Brasil.

2.3. IHC

Este capítulo trata do entendimento referente à Interação Humano-Computador (IHC), o termo IHC é muito usado para descrever sobre a interação do usuário com o sistema.

2.3.1. Conceitos

Os seres humanos tem a necessidade de uma boa e fácil interação com as interfaces do seu dia a dia, os projetos devem ser projetados com base em estudos de IHC. O planejamento das interfaces pode ser agravante no resultado de sucesso ou de fracasso.

Repare que nos smartphones ANDROID as telas, ícones, tamanhos, eventos, ações, cores e entre outros são padronizados, isso não é coincidência, essa padronização pode ser pesquisada por *Google Material Design*, isso facilita que quando o usuário comprar um novo aparelho celular ele rapidamente se adapte ao novo sistema e se existirem algumas possíveis modificações, o usuário entenderá de forma fácil e simples, evitando que precise aprender novamente como se usar o seu novo celular, gerando insatisfação.

A *Google Material Design* não é somente para smartphones, também há uma padronização para paginas web e entre outros. As padronizações não são obrigatórias, são apenas sugestões, mas o desenvolvedor decide se o seu projeto será construído nos padrões GOOGLE ou não.

2.4 JAVASCRIPT

Criado em 1995 por Brendan Eich como uma linguagem de programação padronizada pela ECMA Internacional (Associação Especializada em Padronização de Sistemas de Informação), foi lançada pela primeira vez como LiveScript na versão beta do navegador Netscape 2.0.

Em setembro de 1995 teve uma parceria com a empresa SUN MICROSYSTEMS e seu nome foi alterado para JavaScript. Após alguns anos, em dezembro 2009, a SUN MICROSYSTEMS foi adquirida pela empresa ORACLE CORPORATION que atualmente mantém a marca registrada da linguagem JavaScript.

Com o princípio de facilitar a integração com a parte visual das páginas web, conhecido como *Document Object Model* (DOM), foi desenvolvido para os navegadores web utilizarem em processos de *scripts*, que seria executado pelo computador do cliente, assim não seria utilizado o processador do servidor, tornando a aplicação mais rápida e com menor custo para a empresa que mantém o site web.

Atualmente é uma linguagem de programação de *cliente-side* mais utilizada pelos navegadores como CHROME, MOZILLA FIREFOX, OPERA e entre outros. Com as novas versões e/ou atualizações como o ECMAScript6, a linguagem se torna cada vez mais dinâmica melhorando suas funcionalidades como orientação a objetos, tipagem fraca e ganhando utilidade também para a linguagem de programação utilizada na parte de *back-end*, que é utilizada para processar os dados pelo servidor através de ambientes como NodeJS.

2.5 NODEJS

É uma plataforma construída utilizando como linguagem o JavaScript, baseado nos motores do Google Chrome V8, facilitando a criação de aplicações de forma mais rápida e escaláveis.

Node.js usa o modelo “Entrada e Saída” (I/O) – bloqueante e não bloqueante que torna leve e eficiente, ideal para aplicações com trocas de dados imensa e em tempo real. O projeto foi apresentado na JSConf 2009 Européia, por um programador chamado Ryan Dahl, e foi um projeto inovador e ousado, totalmente diferente das atuais plataformas da época, uma aplicação em JavaScript rodando no servidor. A referência a seguir é retirada do site NODEBR no segundo parágrafo dizendo:

“Aproveitando o poder e a simplicidade do Javascript, isso tornou tarefas difíceis de escrever aplicações assíncronas em tarefas fáceis. Desde quando foi aplaudido de pé no final do seu discurso, o projeto de Dahl tem recebido uma popularidade e uma aprovação sem precedentes.”

O Node.js foi criado pensando em como poderia diminuir as requisições ao servidor. Linguagens como JAVA e PHP, abrem uma conexão (uma nova Thread) que consome memória, em média 2MB. Em um sistema que tenha 8GB de RAM, isso põe o número máximo de conexões cerca de 4.000 usuários, se este número de usuários aumentar, terá que ser contratado um serviço maior de servidor, além disso, cada usuário pode fazer diversas requisições no servidor, consumindo um número maior de memória. Com isso toda a aplicação do servidor poderá ficar sobrecarregada, desde toda a arquitetura da aplicação web, incluindo velocidade de tráfego, velocidade da memória e velocidade do processador.

Além desses possíveis problemas apresentados, temos ainda as linguagens bloqueantes, ou seja, dependem de um processo para fazer o próximo processo, faz com que a aplicação demore mais tempo para acabar todos os processos, e temos o JavaScript que é não bloqueante. Na Figura 1, o gráfico representa o esforço que processador precisa para conseguir finalizar todas as tarefas solicitadas em seu menor tempo possível.

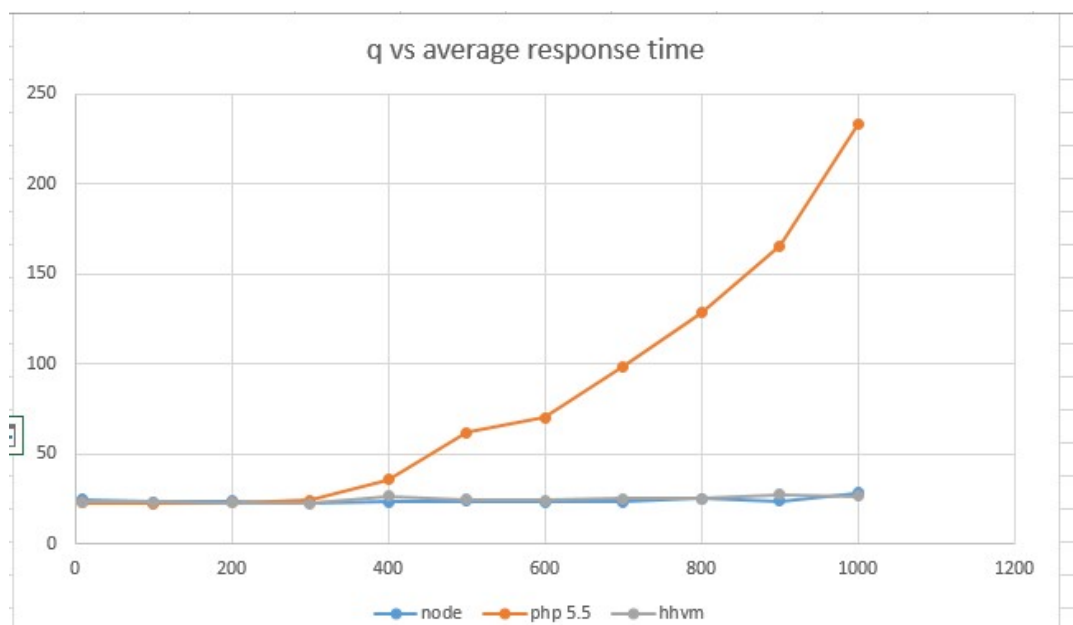


Figura 1

Fonte: <http://www.hostingadvice.com>

A linguagem PHP é usada em alguns exemplos, mas poderia ser representada por qualquer outra linguagem bloqueante como JAVA por exemplo.

Como é mostrado na Figura 1, a linguagem PHP que é Thread bloqueante, necessita de um esforço maior do processador para finalizar suas tarefas, enquanto o NodeJS que é Thread não bloqueante, exige menos do processador.

A Figura 2 exemplifica a abstração de como seriam as chamadas no processador solicitando novos Threads.

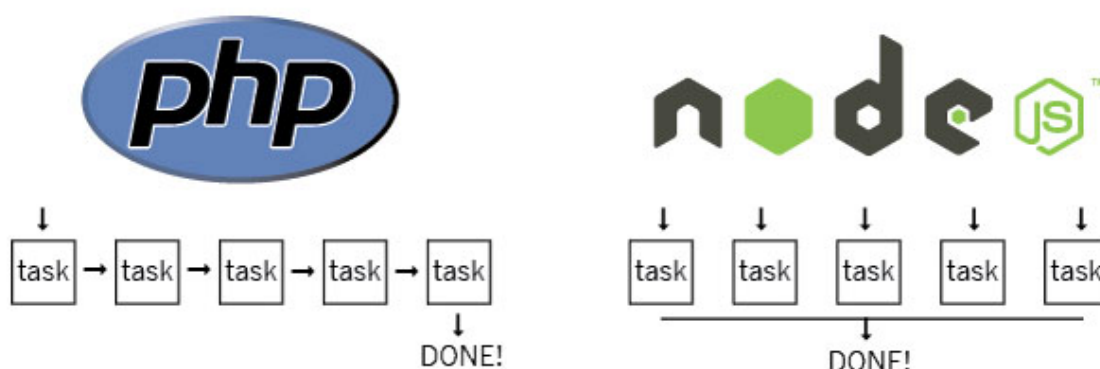


Figura 2

Fonte: <https://servercheck.in>

As linguagens bloqueantes, precisam da tarefa “task” anterior para fazer a tarefa seguinte, e para cada tarefa, é aberta uma nova Thread no

processador, enquanto as linguagens não bloqueantes entregam para o processador todas as tarefas que necessita e uma única Thread.

Na Figura 3, podemos entender o tempo em que o processador precisou para finalizar todas as tarefas, da esquerda para a direita, o tempo em que foi gasto desde que a tarefa chegou no servidor e a quantidade de memória ram que foi utilizada, observe os valores referentes ao PHP e ao NodeJS.

Referência 3: CombSort Strict CPU Test

O seguinte benchmark de [classificação CombSort](#) é um teste de CPU estrito.

Os resultados de referência:

	Tempo de CPU	Hora do sistema	RAM
PHP 5.6.4	102,69s	104.20s	2497508 KB
HHVM 3.5.0	12.56s	14,83s	362488 KB
Node.js v0.10.35	2,64s	2,64s	92240 KB

HHVM é sete vezes mais rápido do que PHP simples (por tempo de sistema), mas Node.js é mais do que cinco vezes mais rápido do que HHVM neste teste de número-crunching.

Em termos de uso RAM, HHVM é muito mais eficiente do que PHP, mas Node.js é ainda melhor.

Figura 3

Fonte: <http://www.hostingadvice.com>

Note que a linguagem NodeJS consegue ser extremamente mais rápida em comparação a uma linguagem bloqueante.

Finalizando, temos um gráfico na Figura 4 representando quantas solicitações o NodeJS consegue receber por segundo comparando com as demais linguagens concorrentes atualmente.

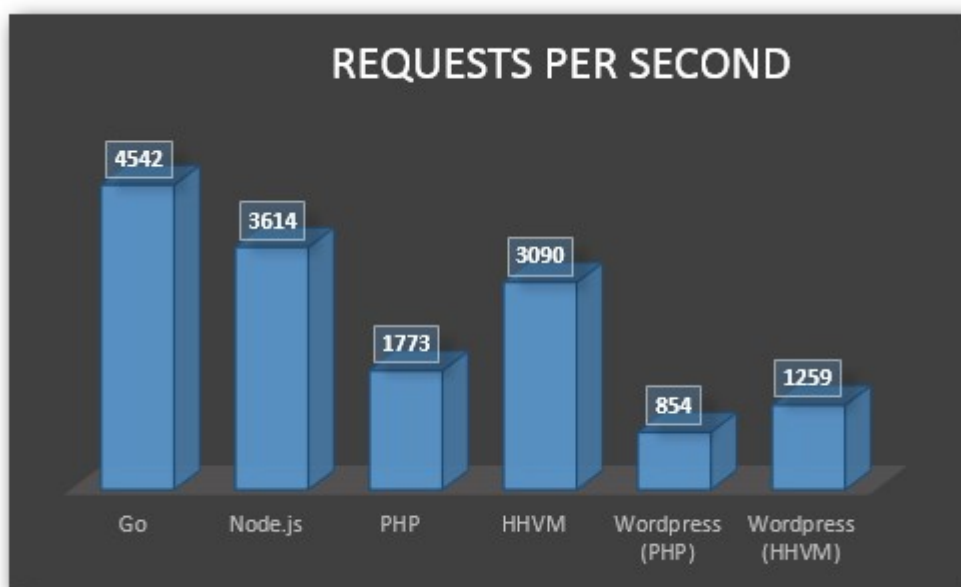


Figura 4.

Fonte: <http://www.hostingadvice.com>

2.6 ANGULARJS

Desenvolvido inicialmente por Misko Hevery e Adams Abrons em 2009, com o objetivo de facilitar a criação de aplicações web, anos após, Misko foi trabalhar na Google no projeto Google Feedback, e percebeu que os códigos eram complicados e contava com 17.000 linhas de código.

Em poucas semanas conseguiu reescrever o código do projeto inteiro reduzindo para 1.500 linhas, com isso, aos poucos o Google foi adotando o framework para os projetos internos, e atualmente, em 2017, estamos na versão AngularJS 4.

A linguagem AngularJS é baseada no JavaScript e funciona basicamente usando *ng- 'alguma coisa'*, por exemplo, ng-controller, ng-repeat, ng-click e entre outros.

Basicamente, existe um atributo chamado *ng-controller*, que contém todo o código, como funções de eventos, por exemplo, e esses códigos pertencem somente aquele bloco. Na Figura 5, temos um trecho de código contendo o *ng-controller*.

```
<div ng-controller="ListaComprasController">
  <table>
    <thead>
      <tr>
        <th>Produto</th>
        <th>Quantidade</th>
      </tr>
    </thead>
    <tbody>
      <tr ng-repeat="item in itens">
        <td><strong>{{ item.produto }}</strong></td>
        <td>{{ item.quantidade }}</td>
      </tr>
    </tbody>
  </table>
</div>
```

Figura 5

Fonte: <https://tableless.com.br>

Na Figura 5, é colocado um *ng-controller* chamado *ListarComprasController*, ele está dentro de uma `<div>`, isso representa que somente os elementos que estão dentro do bloco, filhos da `<div>`, poderão usar os códigos JavaScript referente aquele *ng-controller*. Os dados são recebidos em tempo real por uma variável chamada *\$scope*, essa variável é informada no *ng-controller*. A função do *\$scope* é pegar os dados que o usuário informa e transferir para o *ng-controller*, e as funções dentro do *ng-controller* também poderão usa-la.

Em seguida, temos o *ng-model*, ele é usado para pegar as informações que são digitadas pelo usuário e enviar essas informações para o *ng-controller* em tempo real. Seu melhor aproveitamento é usado dentro de um formulário. Na Figura 6, temos um trecho de código contendo o *ng-model*.

```
<form class="form-inline" name="formItem">  
  <input type="text" ng-model="item.produto">  
  <input type="number" ng-model="item.quantidade">  
  <button ng-click="adicionaItem()">adicionar item</button>  
5. </form>
```

Figura 6.

Fonte: <https://tableless.com.br>

Na Figura 6, o *ng-model* recebe as informações de “produto” e “quantidade”, e informa os dados em tempo real para alguma função do *ng-controller*.

Os dados são recebidos em tempo real por uma variável chamada *\$scope*, essa variável é informada no *ng-controller*, a função do *\$scope* é pegar os dados que o usuário informa e transferir para o *ng-controller*, e as funções dentro do *ng-controller* também poderão usa-la.

```
function ListaComprasController($scope) {  
    $scope.itens = [  
        {produto: 'Leite', quantidade: 2, comprado: false},  
        {produto: 'Cerveja', quantidade: 12, comprado: false}  
5.    ];  
  
    $scope.adicionaItem = function () {  
        $scope.itens.push({produto: $scope.item.produto,  
                            quantidade: $scope.item.quantidade,  
10.                            comprado: false});  
        $scope.item.produto = $scope.item.quantidade = '';  
    };  
}
```

Figura 7.

Fonte: <https://tableless.com.br>

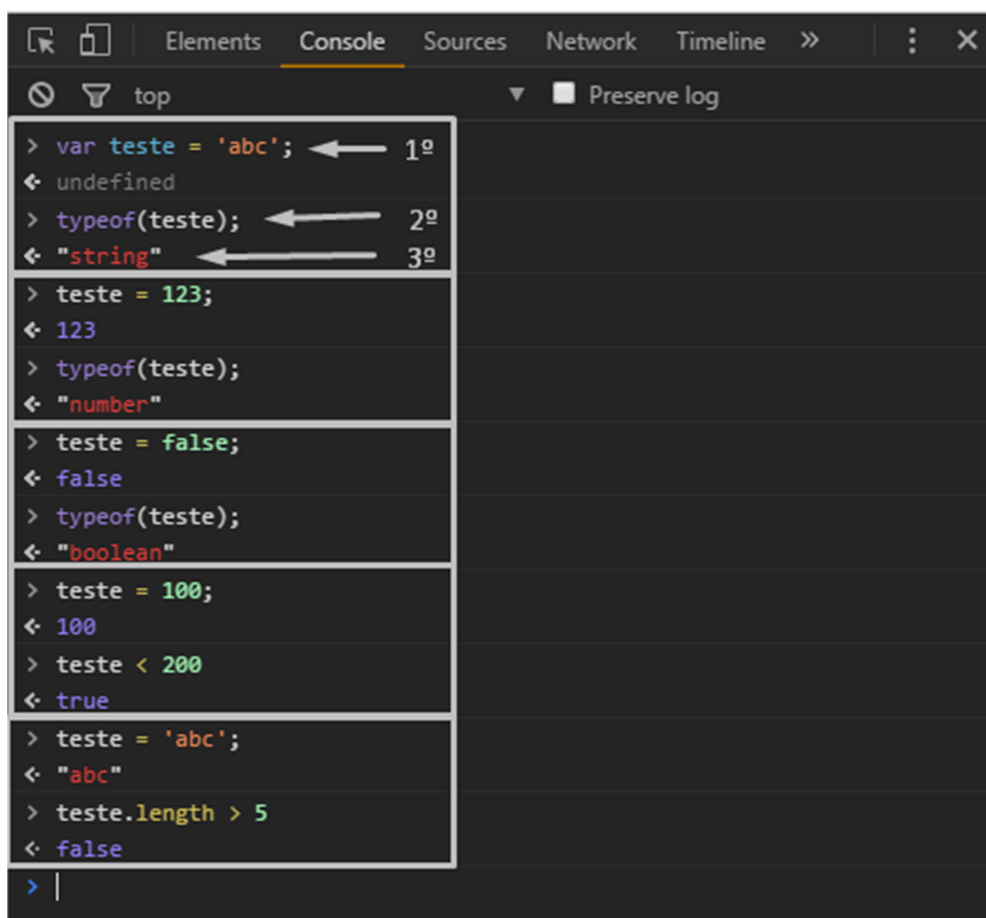
2.7 SEGURANÇA

No projeto proposto, também há uma verificação de segurança onde examina os valores recebidos do Front-End se são equivalentes aos dados esperados.

Por exemplo, a variável *SEXO*, é informado que ela recebera somente os valores “F” para feminino ou “M” para masculino, se for recebido qualquer outro valor, é retornado um erro informando ao usuário. Outros tipos de validação também são do tipo, campo obrigatório, tamanho mínimo e tamanho máximo, se o tipo da variável é string, number, boolean e entre outros.

Os valores são verificados pelo Front-End e também são verificados na API para garantir que os dados não sejam trabalhados de forma errada ou maliciosa.

Como JavaScript não é uma linguagem tipada, há algumas maneiras de saber seu tipo usando algumas funções, um exemplo básico é mostrado na Figura 8.



```
> var teste = 'abc'; ← 1ª
< undefined
> typeof(teste); ← 2ª
< "string" ← 3ª

> teste = 123;
< 123
> typeof(teste);
< "number"

> teste = false;
< false
> typeof(teste);
< "boolean"

> teste = 100;
< 100
> teste < 200
< true

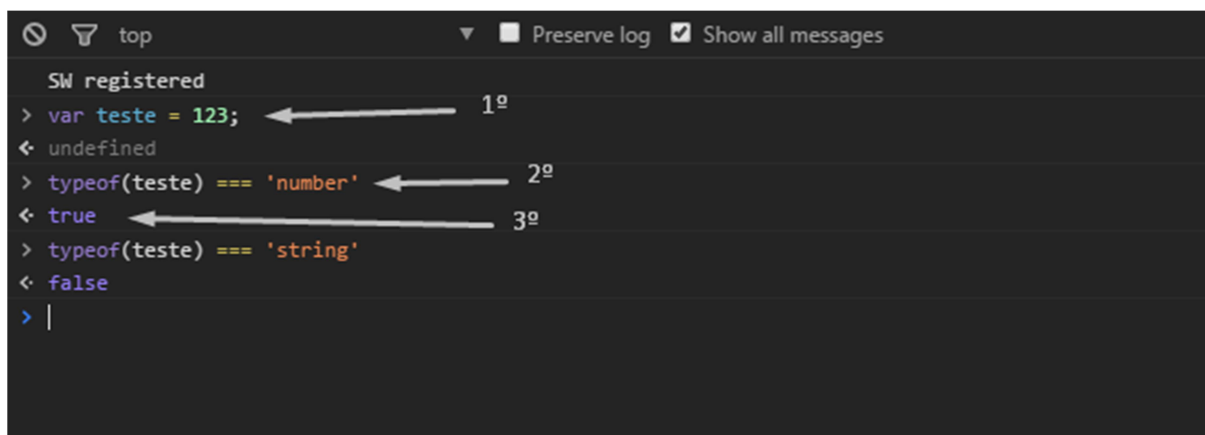
> teste = 'abc';
< "abc"
> teste.length > 5
< false

> |
```

Figura 8.

Fonte: O autor.

Como mostrado na Figura 8, primeiro criamos uma variável chamada “teste” e inserimos um valor dentro dela. Em seguida, usamos a função “typeof”, a função informa o tipo de dado que a variável contém. Outro exemplo é informado na Figura 9.



```
SW registered
> var teste = 123;
< undefined
> typeof(teste) === 'number'
< true
> typeof(teste) === 'string'
< false
> |
```

The screenshot shows a JavaScript console with a dark background. At the top, there's a status bar with 'SW registered', a 'top' link, and checkboxes for 'Preserve log' and 'Show all messages'. The console output shows a series of commands and their results. The first command is 'var teste = 123;', which returns 'undefined'. The second command is 'typeof(teste) === 'number'', which returns 'true'. The third command is 'typeof(teste) === 'string'', which returns 'false'. There are three annotations with arrows pointing to specific lines: '1º' points to the first command, '2º' points to the second command, and '3º' points to the third command.

Figura 9.

Fonte: O autor.

Como mostrado na Figura 9, podemos comparar o retorno da função “typeof” e comparar com a forma que deseja ser esperado, como string, number e ente outros, retornando “true” ou “false”.

Outra maneira muito utilizada é a criptografia, com ela podemos deixar senhas de uma maneira.....

Na figura 10, é mostrado um exemplo de criptografia MD5, um simples texto com as seguintes informações “123” se transforma em um conjunto de letras e números evitando que seja facilmente legível. Uma forma bastante utilizada é para embaralhar senhas de usuários. A criptografia MD5, utilizada no exemplo, não pode ser voltada a forma original. Se o banco de dados cair em mãos não autorizadas, as pessoas não conseguiram entender os dados por estarem todos criptografados.

No projeto apresentado, é utilizada a criptografia na senha do usuário, de forma que nem os desenvolvedores conseguiram entender.

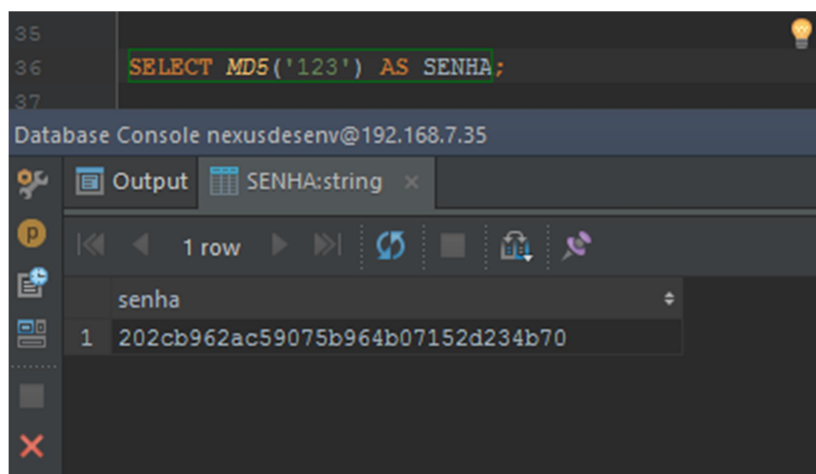


Figura 10.

Fonte: O autor.

2.7 POSTGRESQL

É um banco de dados relacional que vem ganhando bastante mercado

CONSIDERAÇÕES FINAIS

REFERÊNCIAS

Autor: Lauri Tadeu Corrêa Martins – Como montar um posto de combustível
Disponível em: <<https://www.sebrae.com.br/sites/PortalSebrae/ideias/como-montar-um-posto-de-combustivel,aae87a51b9105410VgnVCM1000003b74010aRCRD#naveCapituloTopo>> Acesso em: 19/11/2016.

PETROBRAS – Preços. Disponível em: <http://www.petrobras.com.br/pt/produtos-e-servicos/composicao-de-precos/>

Interação Humano-Computador:
<<https://static.colaboraread.com.br/contents/22317192-ab99-42dd-b68d-f5c896c8291d/assets/resources/978-85-87686-62-6-INTER-HUMANO-COMPUTADOR.pdf>> Acesso em: 03/12/2016

Google Material Design <<https://design.google.com/>> Acesso em: 03/12/2016

Acesso em 28/03/2017 - <https://www.significados.com.br/javascript/>

Acesso em 28/03/2017 - <https://pt.wikipedia.org/wiki/JavaScript>

Acesso em 28/03/2017 - <https://www.w3schools.com/js/>

Acesso em 28/03/2017 - <https://www.javascript.com/>

Acesso em 30/03/2017 - <http://nodebr.com/o-que-e-node-js/>

Acesso em 30/03/2017 - <http://www.hostingadvice.com/blog/comparing-node-js-vs-php-performance/>

Acesso em 30/03/2017 - <https://servercheck.in/blog/moving-functionality-nodejs-increased-server>

Acesso em 30/03/2017 - <https://tableless.com.br/criando-uma-aplicacao-simples-com-angularjs/>

Acesso em 30/03/2017 - <https://angularjs.org/>

APÊNDICES