**ESCOLA SENAI “PROFESSOR VICENTE AMATO”**

Curso Técnico de Informática

EDUARDO AUGUSTO GALVÃO

**DESCRIÇÃO NARRATIVA – PROJETO DE ESTACIONAMENTO**

**JANDIRA**

**2019**

EDUARDO AUGUSTO GALVÃO

**DESCRIÇÃO NARRATIVA – PROJETO DE ESTACIONAMENTO**

Projeto apresentado ao Curso Técnico de Informática, à Escola SENAI “Professor Vicente Amato”, à disciplina de Algoritmos como quesito para obtenção de nota.

Professora: Fernando Leonid

**JANDIRA**

**2019**

**DESCRIÇÃO NARRATIVA**

* **Adição de clientes ao estacionamento**

1. Validar a entrada de dados do painel principal. O campo com nome só pode receber letras e o campo da placa deverá receber 3 letras, seguidas de 1 hífen e 4 números. O campo de placa receberá uma máscara e o nome receberá um filtro constante ao teclar;
2. Ao clicar no botão “Adicionar”, checar se os dois campos estão preenchidos e se o campo da placa atende aos critérios exemplificados no campo. Caso não estejam cumprindo o especificado, exibir mensagem ao usuário;
3. Caso passe pela verificação, os dados do cliente e do veículo devem ser salvos no banco de dados em Web Storage, assim como a data, hora e milissegundos de entrada e de saída, porém as informações da saída receberão valor vazio (“ “), assim como um atributo valor valendo 0, que durante a retirada será atualizado;
4. A tabela principal deve ser atualizada, exibindo os clientes cadastrados, filtrando os que possuam data de saída nula;
5. Os campos de input devem ser limpos para receber uma futura nova adição.

* **Criação da Tabela Principal**

1. Assim que o primeiro cliente for adicionado, na página principal deverá haver uma tabela com os clientes ainda estacionados. Essa tabela deverá conter dados do cliente, do veículo, sua data e hora de entrada e um local para ações;
2. A tabela deverá obedecer ao fluxo de exibir somente o que existe no banco de dados e que possua data de saída nula;
3. A criação dos botões de ações, “Comprovante”, “Edição” e “Saída” deverá ser feita tendo um ID para cada, que terá a ação e o código do cliente no modelo “acao\_codigo”, facilitando a identificação do mesmo ao chamar funções;
4. A tabela deverá atualizar automaticamente após qualquer alteração no banco.

* **Adição de valores de cobrança e tolerância**

1. Ao clicar no botão “Preços”, deve abrir um modal com os campos de: valor da primeira hora, valor das demais horas e tolerância em minutos;
2. A entrada dos dados deve ser validada durante a escrita, impedindo qualquer caractere além de números. Também haverá validação de campos vazios ao adicionar os novos valores;
3. Caso passe pela verificação, os dados de valores deverão ser armazenados em um banco no *Web Storage* dedicado para os valores.
4. Os campos de input devem ser limpos para receber uma futura nova adição.

* **Exibição de relatório diário**

1. Quando o botão “Relatório Diário” do formulário principal for clicado, deve haver a exibição do relatório diário, contendo todas as informações dos clientes que já retiraram o veículo do estacionamento e o rendimento diário;
2. Quando clicado, haverá a chamada de uma função que filtrará os clientes que tem a data de saída diferente de vazio (“ ”) e exibirá seus dados em uma tabela;
3. Para o rendimento diário, haverá o uso do método *reduce*( ) para somar os valores de todos os clientes filtrados.

* **Exibição de comprovante**

1. Assim que um cliente for adicionado, deve estar disponível um comprovante com os dados da sua entrada ao clicar no botão “Comprovante” da tabela principal;
2. Ao realizar o clique, o javascript deverá capturar a ação e identificar o comando e código do cliente através do ID do botão gerado;
3. Identificada a ação do comprovante, deverá chamar uma função que exiba o modal do comprovante;
4. Deverá ser feita a leitura do banco do Web Storage, procurando um registro que tenha o mesmo código identificado anteriormente, e a partir disso exibir os dados do cliente e do veículo, assim como data e hora de entrada;
5. O modal deverá ter opções para imprimir ou fechar o comprovante.

* **Edição de informações de clientes**

1. Assim que um cliente for adicionado, deve estar disponível um botão “Editar” na tabela principal;
2. Ao realizar o clique, o javascript deverá capturar a ação e identificar o comando e código do cliente através do ID do botão gerado;
3. Identificada a ação do botão de edição, deverá chamar uma função que exiba o modal de edição;
4. Deverá ser feita a leitura do banco do Web Storage, procurando um registro que tenha o mesmo código identificado anteriormente, e a partir disso exibir os dados do cliente e do veículo para edição, se for desejada;
5. Caso o usuário clique em “Atualizar”, os campos de input deverão sofrer uma nova verificação, incluindo se os campos estão vazios, e se o nome e placa atendem aos critérios estabelecidos;
6. Atendendo as validações, os dados serão salvos em um novo modelo de registro e o banco do *Web Storage* sofrerá *splice( )* substituindo os dados e mantendo o código do cliente;
7. Após a atualização, a tabela principal deverá ser atualizada automaticamente.

* **Remoção de Clientes**

1. Assim que um cliente for adicionado, deve estar disponível um botão “Saída” na tabela principal;
2. Ao realizar o clique, o javascript deverá capturar a ação e identificar o comando e código do cliente através do ID do botão gerado;
3. Identificada a ação do botão de saída, deverá chamar uma função que valide a saída e calcule o valor final do cliente;
4. A primeira validação deverá existir verificando se já existe um banco de dados com os valores da primeira hora, demais horas e tolerância. Somente se já existir, será feito o cálculo do valor final de cobrança e o cliente ganhará uma data e hora de saída, caso contrário será mantido e haverá um aviso no sistema para preenchimento dos campos;
5. Caso já exista um banco com os valores, deverá ser feita a leitura do banco do Web Storage, procurando um registro que tenha o mesmo código identificado anteriormente, e a partir disso atualizar os dados desse cliente no banco de dados com uma data, hora e milissegundos da saída do mesmo;
6. Para o cálculo do valor de cobrança, deve haver uma leitura do banco de dados de valores e encontrar o maior código do banco. Encontrar o maior código do banco significa encontrar o mais recente, ou seja, os valores vigentes de cobrança;
7. Encontrados os valores vigentes, deve-se capturar o valor da primeira e demais horas, assim como a tolerância para realizar o cálculo;
8. O cálculo de valor de cobrança final será feito obtendo a diferença entre os milissegundos de entrada e saída e a partir disso checar quantas horas pagas o cliente permaneceu no estacionamento, considerando a tolerância em minutos definida;
9. Calculado o valor, o cliente terá uma atualização com os atributos atualizados e, conforme o critério de exibição da tabela principal, não será mais exibido. Porém, ao abrir o relatório diário, seu registro aparecerá, assim como o valor final pago, integrando o total no rendimento diário.