Título do Projeto: Rent A Car

Data de Início: 15/03/2017	**Data de Término:** 17/03/2017

Gerente de projeto: Eduardo Batista Beziaco

Cliente: BDR

Usuários: BDR

Proposito :
O propósito deste projeto é a participação de um processo seletivo técnico e prático o projeto e seus requisitos devem ser entregues em 48 horas.

Objetivos: O objetivo é que com a implementação do projeto o gerente do mesmo seja contratado ao final.

Requisitos do projeto (Estabelecidos pelo cliente):

1. Desenvolver um sistema Web em Java para cadastramento de veículos de uma locadora de automóveis.
2. O cadastro será apenas dos automóveis e os atributos são de livre escolha.
3. Deve ser possível cadastrar, visualizar, remover e editar estes veículos.
4. É necessário fazer a validação dos campos.
5. Deverá ter controle de paginação na tabela.
6. Utilizar as APIs atuais para a parte de UI (User Interface), exemplo: angular, react, backbone, vue e etc.
7. Não deve ser utilizado banco de dados.
8. O servidor de aplicação utilizado deve ser o Tomcat.
9. Todo o código deve ser documentado e comentado.
10. A entrega do sistema deverá conter:
    a. Um projeto do Eclipse com todas as fontes do sistema;
    b. Documentação técnica em PDF ou DOC;
    c. Arquivo README com informações sobre o sistema.
11. O projeto deverá ficar disponível no github e deverá ser enviada a URL do repositório para análise.

Atributos dos automóveis:
Para o cadastro de automóveis foram escolhidos os seguintes atributos:

1. Placa - Para fins de chaveamento a placa será única. (String)
2. Modelo - O modelo do veículo. (String)
3. Cor - A cor do mesmo. (String)
4. Ano de Fabricação - Para fins de seguro. (Integer)
5. Chassi - Para fins de seguro. (String)
6. Automático. - Para a preferência do cliente, sim ou não. (Boolean)

Implementação: Foi escolhido trabalhar apenas com Java(Servlets) e HTML puros.
Arquivos desenvolvidos:

1. Carro - Classe para conter os atributos já mencionados, com todos os Getters e Setters.
2. Controlador - Classe com uma lista de Carros com métodos para acesso a informação, possuindo uma variável estática de sua própria instância para facilitar a codificação já que não usamos Banco de Dados. Possui métodos de controle para acessar a lista e então manipula-la, como por exemplo, addCarro, editaCarro, getCarro , etc.
3. EditarCarro - Servlet para a edição das informaçõs em um form do HTML.
4. ExcluirCarro - Servlet para a confirmação de exclusão.
5. CadastrarCarro - Servlet génerico, sempre é chamado ao final de uma operação, mostrando as mensagens de sucesso ou falha, juntamente com um botão para um novo cadastro e outro para retornar à tela principal.
6. IndexServlet - Servlet principal nele é mostrado a lista de todos os carros já cadastrados, na tabela cada carro possui seu próprio botão de "Editar" e Excluir. Logo abaixo da tabela temos o botão para cadastro de novos veículos.
7. CadastroDeVeiculo - Não é uma classe, mas é um html puro para o cadastro de novos veículos.
8. estilo - Arquivo CSS para tornar as páginas mais visualmente agradáveis.

Limitações: Por falta de tempo não foi possível a utilização de alguma API para a criação do UI, porém as páginas foram personalizada com CSS. E também não foi implementado uma paginação na tabela. Pelo prazo de entrega ser de 48 horas a estrutura MVC não foi totalmente usada e algumas boas práticas foram deixadas de lado para agilizar a implementação.

Detalhes Técnicos Específicos:

1. O Servlet CadastrarCarro sempre irá receber um parametro escondido(hidden) chamado editou, que pode possuir os seguintes valores: "sim", "nao" e "ex", respectivamente dizem para a página agir como alteração, cadastro ou exclusão.
2. Entre algumas páginas também é passado o parâmetro escondido placaParam que leva a informação da placa adiante, para pesquisa, edição e ou exclusão do carro, de certa forma simulando uma chave em uma tabela de um Banco de Dados.
3. A classe Controlador possui uma referência para sua própria instância que é estática o que permite uma persistência de dados entre as telas do projeto de maneira rápida e fácil, mas de nenhuma forma isso é uma boa prática, nem para aplicações Web e nem para Desktop.
4. Por diversos motivos as portas padrões do Tomcat foram alteradas, Admin port para 8000, HTTP para 8085 e AJP para 8010. Foi usado a versão Tomcat 7.0 em um localhost normal.
5. Foi utilizado a IDE Eclipse Kepler para JAVA EE, o Tomcat foi configurado e usado diretamente dentro da IDE e a biblioteca java usada foi a versão jre1.8.
6. Conforme pedido o código foi bastante comentado por tanto existe bastante

informação nas linhas de comentário.

7. A página inicial e principal é a http://localhost:8085/RentAcar/index.
8. A validação de campos foi feita diretamente pelo html através da tag pattern, segue explicação para cada atributo;
    a. Modelo: ([A-z0-9\s]){2,} - Só aceita caracteres alfanuméricos e o campo precisa ter no mínimo de 2 caracteres.
    b. Ano de Fabricação: Não teve o pattern porém tem o minimo de 2000 e o máximo de 2017.
    c. Cor: ([A-z\s]){2,} - Aceita qualquer letra e o campo precisa ter mínimo de 2 caracteres.
    d. Placa Letras: (?=.*[A-Z]){3} - Aceita apenas letras maiúsculas e o campo precisa ter 3 caracteres, nem a mais nem a menos.
    e. Placa Dígitos: [0-9]{4} - Aceita apenas números e o campo precisa ter 4 caracteres, nem a mais nem a menos.
    f. Automático: Não teve pattern é um simples checkbox.
    g. Chassi: ([A-z0-9]){17} - Só aceita caracteres alfanuméricos e o campo precisa ter 17 caracteres, nem a mais nem a menos. Como os chassis de qualquer carro atual.