Universidade Federal do Rio de Janeiro
Instituto Multidisciplinar
Computação III
Discentes: Douglas Castro
Renan Procópio

Relatório do Trabalho Final

Sumário

I. Como executar a aplicação no TomCat

Segue os passos para executar a aplicação Carona no TomCat:

- ➔ Neste trabalho foi utilizado o banco de dados mysql e a ferramenta de administração phpmyadmin do pacote de servidores de código aberto Xampp.
- ➔ Crie um banco utilizando o phpmyadmin, após tê-lo criado execute o script createDB.sql que está localizado no pacote “conexaoBanco” dentro da pasta src do projeto.
- ➔ Altere o usuário e senha do banco na classe ConnectionFactory.java que também está localizada no pacote “conexaoBanco” dentro da pasta src do projeto.
- ➔ Inicie o TomCat (o projeto foi concebido na versão v7.0 do TomCat)
- ➔ Acesse a url: http://localhost:8080/CaronaAgoraVai/Login.jsp
- ➔ Logue com o usuário padrão:
  - email: default@default
  - senha: 1234

II. CRUDs

A. Logradouro

Create	Read	Update	Delete
✓	✓		✓

B. Usuário

Create	Read	Update	Delete
✓	✓	Nome, Telefone	

C. Grupo

Create	Read	Update	Delete
✓	✓	Nome, descrição e limite minimo	

D. Avaliação

Create	Read	Update	Delete
✓			

E. Carona

Create	Read	Update	Delete
Só se usuário e veículo não estiverem atrelados a uma carona no mesmo dia e horário.	✓	Só se usuário e veículo não estiverem atrelados a uma carona no mesmo dia e horário.	

F. Veiculo

Create	Read	Update	Delete
✓	✓	Cor	

G. Solicitação

Create	Read	Update	Delete
✓	✓		

H. Parada

Create	Read	Update	Delete
Apenas motoristas podem criar paradas.	✓		✓

III. Requisitos

A. Requisitos Funcionais

1. O sistema deverá permitir usuários que fazem parte do grupo mudem seu nome, descrição e limite mínimo de avaliações ruins.
2. O sistema deverá permitir que o usuário que obtenha mais avaliações ruins que o limite do grupo fique inativo no mesmo.
3. O sistema deverá permitir que o último usuário ativo no grupo coloque-o como inativo.
4. O sistema deverá permitir que motoristas criem caronas.
5. O sistema deverá permitir que usuários participem de caronas.
6. O sistema não deverá permitir que usuários participem da mesma carona mais de uma vez (isto é ocupem mais de uma vaga).
7. O sistema não deverá permitir que usuários participem de mais de uma carona com mesma data e hora.
8. O sistema deverá permitir que usuários criem grupos, de acordo com a Regra de Negócio RN02.
9. O sistema deverá permitir que usuários de um grupo convidem outros usuários através de seu email.
10. O sistema deverá permitir que usuários participem de mais de um grupo.
11. O sistema deverá permitir que usuários altere seu nome e seu telefone.
12. O sistema não deverá permitir que usuários alterem regras de grupos.
13. O sistema deverá permitir que usuários adicionem veículos.
14. O sistema deverá permitir que usuários alterem a cor de seus veículos.
15. O sistema deverá permitir ao usuário avaliar anonimamente outro usuário de acordo com a regra de negócio RN01.
16. O sistema não deverá permitir que um usuário inativo no grupo tenha acesso a informações do grupo no qual ele está inativo.
17. O sistema não deverá deixar o usuário remover seus veículos.
18. O sistema deverá permitir a criação de Logradouros.
19. O sistema não deverá permitir que o logradouro seja alterado ou removido.
20. O sistema deverá deixar o motorista decidir em qual carro seu ofertará a carona.
21. O sistema deverá permitir que o motorista altere o veículo no qual ele oferta a carona de acordo com a Regra de Negócio RN03.

22. O sistema deverá permitir que o motorista altere a origem e o destino da carona de acordo com a Regra de Negócio RN04.
23. O sistema deverá permitir ao usuário se candidatar a passageiro de uma carona, de acordo com a Regra de Negócio RN05.
24. O sistema deverá permitir ao motorista associar o usuário a uma parada através do email do usuário.
25. O sistema deverá permitir ao motorista criar paradas, caso as mesmas não existam.
26. O sistema deverá permitir que um usuário cancele a carona a qual ele está ofertando.
27. O sistema deverá ao usuário participante da carona cancelar sua candidatura a carona.
28. O sistema deverá permitir que o motorista informe quando a carona terminou.

B. Requisitos Não-Funcionais

1. O logradouro deverá ser criado através de uma API disponível no link https://viacep.com.br/, onde o usuário informa o cep e o número somente.
2. O sistema deverá ser implementado na linguagem Java.
3. O sistema deverá ser implementado seguindo o padrão arquitetural da camada de domínio chamado Módulo Tabular.
4. O sistema deverá ser implementado seguindo o padrão arquitetural da camada de dados chamado Mapeador de Dados.

C. Regras de Negócio

1. Regra de Negócio RN01: A avaliação varia entre 1 à 5 estrelas.
2. Regra de Negócio RN02: O usuário que criou o grupo é automaticamente adicionado ao grupo.
3. Regra de Negócio RN03: O motorista poderá modificar o veículo desde que não altere o número de vagas ofertadas.
4. Regra de Negócio RN04: Origem e Destino só podem ser alterados desde que não hajam passageiros na carona
5. Regra de Negócio RN05: Deverá haver vaga no veículo.

IV. Diagrama de Classes

Vide arquivo anexo “*DCDominio*” e “*DCMappers*”.

Os arquivos ficaram com uma dimensão muito grande, devido a este fato não foram exportados como imagem, necessitando assim do programa Astah para abrí-los.

V. Relatório de Testes Unitários

Vide arquivo anexo “Relatório de Testes Unitários”

VI. Relatório de Testes Funcionais

Identificador: CTF01
Descrição: Caso de Teste para criar um Usuário no Sistema
Item de Teste: Criar um usuário
Setup: tabela de usuário criada com a lista de usuários do sistema.
Teste:

Fazer validações dos dados necessários para a criação do usuário.
UsuarioMapeador uMap.adicionar(Usuário u)
Verificar se o usuário foi inserido no banco.

Resultados Esperados:

Criar um Usuário.

Teardown : deletar a entrada criada.

Identificador: CTF02
Descrição: Caso de Teste para alterar um usuário no Sistema.
Item de Teste: Alterar um Usuário

Setup: tabela de usuário criada com a lista de usuários do sistema, com necessariamente uma entrada válida (um usuário criado)
Teste:

Fazer validações dos dados necessários para a alteração do usuário.
UsuarioMapeador uMap.atualizar(Usuário u)
Verificar se o usuário foi alterado no banco.

Resultados Esperados:

Dados do usuário serem alterados.

Teardown : deletar os usuários criados e alterados.

Identificador: CTF03
Descrição: Caso de Teste para criar uma avaliação no sistema.
Item de Teste: Avaliar um Participante da Carona
Setup: tabela de avaliação criada, tabela de usuários criada, tabela de carona criada; inserir dois usuários (1 para avaliar e outro para ser avaliado), inserir carona e adicionar estes dois usuários a esta carona.
Teste:

AvaliarMapeador aMap.adicionar(Avaliacao a)
Verificar se o usuário foi inserido no banco.

Resultados Esperados:

Avaliação concluida.

Teardown : deletar os usuários criados, a carona criada, e a avaliação criada.

Identificador: CTF04
Descrição: Caso de Teste para criar um veículo no sistema.
Item de Teste: Criar um veículo.
Setup: tabela de veículos criada, criar um usuário para teste.
Teste:

Fazer validações dos dados necessários para a criação do veículo.
VeiculoMapeador vMap.adicionar(Veiculo v)
Verificar se o veículo foi inserido no banco.

Resultados Esperados:

Veículo criado no banco.

Teardown : deletar o veículo criado.

Identificador: CTF05

Descrição: Caso de Teste para alterar um veículo no sistema.
Item de Teste: Alterar um veículo.
Setup: tabela de veículos criado, criação de um veículo.
Teste:
Fazer validações dos dados necessários para a alteração do veículo.
VeiculoMapeador vMap.alterar(Veiculo v)
Verificar se o veículo foi alterado no banco.
Resultados Esperados:
Veículo alterado no banco.
Teardown : deletar o veículo criado/alterado.

Identificador: CTF06
Descrição: Caso de Teste para criar um grupo no sistema.
Item de Teste: Criar um Grupo.
Setup: tabela do grupo criada, criar usuário para teste.
Teste:
Fazer validações dos dados necessários para a criação do grupo.
GrupoMapeador gMap.adicionar(Grupo g)
Verificar se o grupo foi inserido no banco.
Resultados Esperados:
Grupo criado no banco.
Teardown : deletar o usuário criado e o grupo criado.

Identificador: CTF07
Descrição: Caso de Teste para alterar um grupo no sistema.
Item de Teste: Alterar um Grupo.
Setup: tabela do grupo criada, criação de um grupo
Teste:
Fazer validações dos dados necessários para a atualização do grupo.
GrupoMapeador gMap.atualizar(Grupo g)
Verificar se o grupo foi alterado no banco.
Resultados Esperados:
Grupo alterado no banco.
Teardown : deletar o usuário criado e o grupo criado/alterado.

Identificador: CTF08
Descrição: Caso de Teste para criar um logradouro no sistema.
Item de Teste: Criar um Logradouro.
Setup: tabela logradouro criada.
Teste:

Fazer validações dos dados necessários para a atualização do grupo.
LogradouroMapeador logMap.criar(Logradouro l)
Verificar se o logradouro foi criado no banco.

Resultados Esperados:

Logradouro criado no banco.

Teardown : deletar o logradouro criado.

Identificador: CTF09
Descrição: Caso de Teste para criar uma parada no sistema.
Item de Teste: Criar uma Parada.
Setup: tabela parada criada, criar uma carona teste e um logradouro teste, criar usuário para teste.
Teste:

Fazer validações dos dados necessários para a criação da parada.
ParadaMapeador parMap.adicionar(idLogradouro, idCarona, idUsuario, nome)
Verificar se a parada foi criada no banco.

Resultados Esperados:

Parada criada no banco.

Teardown : deletar a parada criada.

Identificador: CTF10
Descrição: Caso de Teste para criar uma solicitação no sistema.
Item de Teste: Criar uma Solicitação.
Setup: tabela solictação criada, criar uma carona teste e criar usuário para teste.
Teste:

Fazer validações dos dados necessários para a criação da solicitação.
SolicitacaoMapeador solMap.adicionar(Solicitacao s)
Verificar se a solicitação para a carona foi criada no banco.

Resultados Esperados:

Solicitação criada no banco.

Teardown : deletar a solicitação criada.

Identificador: CTF11
Descrição: Caso de Teste para criar uma carona no sistema.
Item de Teste: Criar uma Carona.
Setup: tabela carona criada, criar um veículo teste, criar logradouros e paradas para teste.

Teste:

Fazer validações dos dados necessários para a criação da carona.
CaronaMapeador carMap.adicionar(Carona c)
Verificar se a carona foi criada no banco.

Resultados Esperados:

Carona criada no banco.
Paradas criada no banco e associada a carona criada
Logradouros criados no banco e associados as suas respectivas paradas criadas

Teardown : deletar a carona, logradouros e paradas criadas.

Identificador: CTF12
Descrição: Caso de Teste para alterar uma carona no sistema.
Item de Teste: Alterar uma Carona.
Setup: tabela carona criada, criar carona teste.
Teste:

Fazer validações dos dados necessários para a alteração da carona.
CaronaMapeador carMap.adicionar(Carona c)
Verificar se a carona foi alterada no banco.

Resultados Esperados:

Carona alterada no banco.

Teardown : deletar a carona criada.

Identificador: CTF13
Descrição: Caso de Teste para adicionar um usuário à uma carona no sistema.
Item de Teste: Adicionar um usuário à uma Carona.
Setup: tabela carona criada, criar carona teste, criar usuário de teste
Teste:

Fazer validações dos dados necessários para a adição do usuário a carona.
Criar um Logradouro
Criar uma Parada associada ao logradouro criado, a carona e ao usuário
CaronaMapeador carMap.adicionarUsuario(Usuario u, Carona c)
Verificar se a uma parada foi associada ao usuário, a carona e ao logradouro informado

Resultados Esperados:

Parada associada ao usuário e a carona criada.

Teardown : deletar a carona criada, logradouro criado, parada criada, e o usuário criado.