

**INSTITUTO FEDERAL DO MARANHÃO
SISTEMAS DE INFORMAÇÃO
PROGRAMAÇÃO EXTREMA**

Autores: José Matheus Aranha Maranhão e Gabriel Vinicius Pinto Portela.

Projeto TotemBUS

Atividade avaliativa apresentada à disciplina de
Programação Extrema
Prof. Mauro Lopes Carvalho Silva.

São Luís

2022

Sumário

1. Objetivo	3
2. História de Usuário	3
3. DOR	5
4. DOD	5
6. Tarefas dos desenvolvedores	6
7. Refatoração	6
8. Caso de Teste	8
9. Tela de Cadastro de Itinerários	9

1. Objetivo

Este documento tem como objetivo apresentar os itens utilizados para desenvolver a história: “**Cadastro de Itens de Itinerário**” do projeto de estudo de caso: “**Totem de Consulta das Linhas de Ônibus de São Luís**”, este documento aborda a história de usuário, o DOR, o DOD, as tarefas atribuídas aos desenvolvedores, um exemplo de refatoração e um caso teste.

2. História de Usuário

Cartão: 05	Projeto: Totem de Consulta das Linhas de Ônibus de São Luís	Estimativa: 03
Nome da História: Cadastro de Itens de Itinerário		Data: 19/12/2022
História: Como auxiliar de cadastro, preciso de uma interface que possibilite manter os dados dos itens de itinerários, para que seja possível visualizar todos os locais possíveis e que as informações geradas possam ser utilizadas no cadastro dos itinerários.		Critério(s) de Aceitação: <ul style="list-style-type: none">• Dado que o usuário logado é um Auxiliar de Cadastro/Quando informado a descrição do item, o tipo do item e ao clicar no botão [SALVAR]/Então deverá ser verificado se a descrição e o tipo do item não estão vazios, caso esteja o usuário deve ser informado e o item não deve ser cadastrado, pois não deverá ser permitido o cadastro de um item de itinerário vazio.• Dado que o usuário logado é um Auxiliar de Cadastro/Quando informado a descrição do item, o tipo do item e ao clicar no botão [SALVAR]/Então deverá ser verificado se o item cadastrado já existe, caso exista o usuário deve ser informado e o item não deve ser cadastrado, pois não deverá ser permitido o cadastro de itens repetidos;• Dado que o usuário logado é um Auxiliar de Cadastro/Quando não for selecionado um item de itinerário e ao clicar no botão [ALTERAR]/Então deverá ser informado ao usuário que nenhum item foi selecionado e nenhum dado pode ser alterado.• Dado que o usuário logado é um

	<p>Auxiliar de Cadastro/Quando selecionado um item de itinerário e ao clicar no botão [ALTERAR]/Então deverá ser verificado se a descrição do item e o tipo do item não estão vazios, caso um dos itens esteja vazio, o usuário deve ser informado e a alteração do item não deve ser realizada, pois não deverá ser possível itens de itinerários vazios.</p> <ul style="list-style-type: none"> • Dado que o usuário logado é um Auxiliar de Cadastro/Quando selecionado um item de itinerário e ao clicar no botão [ALTERAR]/Então deverá ser verificado se o item que deseja alterar já existe, caso exista o usuário deve ser informado e o item não deve ser alterado, pois não deverá ser permitido o cadastro de itens repetidos; • Dado que o usuário logado é um Auxiliar de Cadastro/Quando selecionado um item de itinerário e ao clicar no botão [EXCLUIR]/Então deverá ser apresentado ao usuário um caixa de seleção para a confirmação de exclusão do item, caso o usuário clique em sim, o item deve ser excluído, caso contrário o item deve permanecer na base. • Dado que o usuário logado é um Auxiliar de Cadastro/Quando não for selecionado um item de itinerário e ao clicar no botão [EXCLUIR]/Então deverá ser informado ao usuário que nenhum item foi selecionado e nenhum dado pode ser excluído. • Dado que o usuário logado é um Auxiliar de Cadastro/Quando selecionado o botão [LIMPAR]/Então os campos de cadastro deverão ser resetados, para que novos itens possam ser cadastrados. • Dado que o usuário logado é um
--	--

	<p>Auxiliar de Cadastro/Quando ao digitar no caixa de texto de argumento de pesquisa /Então deverá ser apresentado em uma tabela o item conforme o argumento digitado.</p> <ul style="list-style-type: none"> • Dado que o usuário logado é um Auxiliar de Cadastro/Quando selecionado o botão [LIMPAR PESQUISA]/Então o campo de argumento de pesquisa deve ser resetado, para que um novo argumento possa ser escrito ou para que todos os dados cadastrados sejam visualizados.
<p>Observações: Para nossa aplicação precisamos que existam tipos de itens de itinerário cadastrados, como por exemplo: Bairro e Ponto turístico.</p>	<p>Risco: Médio Esta funcionalidade será utilizada para cadastrar todos os locais disponíveis em São Luís, esses dados serão utilizados na montagem de itinerários.</p>

3. DOR

1. A história de usuário deve ser escrita no padrão INVEST
2. Os critérios de aceitação estão bem definidos
3. Os testes de aceitação devem ser escritos no padrão BDD
4. O design da tela deve estar pronto

4. DOD

1. Testes unitários devem ter sido criados
2. Todas as funcionalidades foram testadas
3. O código foi revisado por um colega do projeto
4. As funcionalidades atendem aos critérios de aceitação
5. Apresentado as partes interessadas

6. Tarefas dos desenvolvedores

Id	Tarefa	Tempo de Execução (Dias)	Responsável
1	Organizar o projeto de acordo com o padrão MVC	1	Gabriel
2	Criação da classe de conexão com o banco de dados	1	Gabriel
3	Criação do Model Item de Itinerário e tipo do item de itinerário	1	Gabriel
4	Desenvolver a funcionalidade de cadastrar um item de itinerário	2	Matheus
5	Desenvolver a funcionalidade de alterar um item de itinerário	2	Matheus
6	Desenvolver a funcionalidade de excluir um item de itinerário	2	Matheus
7	Desenvolver a funcionalidade de pesquisar um item de itinerário	1	Matheus
8	Desenvolver o Front-end da aplicação	3	Matheus
9	Testar todas as funcionalidades	2	Gabriel
10	Integrar todas as funcionalidades desenvolvidas	2	Gabriel

7. Refatoração

- Código não refatorado

```
private void btnCadastrarItemItinerarioActionPerformed(java.awt.event.ActionEvent evt)
{
    if (jtfId.getText().isEmpty()) {
        if (!jtfDescricao.getText().isEmpty() &&
!cbxTipoItemItinerario.getSelectedIndex().equals("Selecione")) {
            ModelItemItinerario = new ModelItemItinerario();
            ModelItemItinerario.setDescricaoItem(jtfDescricao.getText());

ModelItemItinerario.setIdTipoItem(idTipoItemItinerario.get(cbxTipoItemItinerario.getSelectedIndex() - 1));

            if
(controllerItemItinerario.cadastrarItemItinerario(ModelItemItinerario)) {
                JOptionPane.showMessageDialog(this, "Item cadastrado com sucesso",
"Atenção", JOptionPane.INFORMATION_MESSAGE);
                limparItensDeCadastro();
                carregarTabelaItensItinerario();
            } else {
```

```

        JOptionPane.showMessageDialog(this, "Erro ao cadastrar o item",
"Erro", JOptionPane.ERROR_MESSAGE);
    }
    } else {
        JOptionPane.showMessageDialog(this, "Erro ao cadastrar o item, pois um
item está selecionado", "Erro", JOptionPane.ERROR_MESSAGE);
    }

}

```

- **Código refatorado**

```

public boolean validarCampoIdParaCadastro() {
    if (jtfId.getText().isEmpty()) {
        return true;
    } else {
        JOptionPane.showMessageDialog(this, "Não foi possível cadastrar pois um
item está selecionado, favor limpar os campos", "Erro", JOptionPane.ERROR_MESSAGE);
        return false;
    }
}

public boolean validarCampoDescricao() {
    if (jtfDescricao.getText().isEmpty()) {
        JOptionPane.showMessageDialog(this, "A descrição não pode ser vazia",
"Erro", JOptionPane.ERROR_MESSAGE);
        return false;
    } else {
        return true;
    }
}

public boolean validarCampoTipo() {
    if (cbxTipoItemItinerario.getSelectedIndex().equals("Selecione")) {
        JOptionPane.showMessageDialog(this, "O campo tipo não pode ser vazio",
"Erro", JOptionPane.ERROR_MESSAGE);
        return false;
    } else {
        return true;
    }
}

```

```

private void btnCadastrarItemItinerarioActionPerformed(java.awt.event.ActionEvent evt)
{
    if (validarCampoIdParaCadastro() == true && validarCampoDescricao() == true &&
validarCampoTipo() == true) {
        ModelItemItinerario = new ModelItemItinerario();
        ModelItemItinerario.setDescricaoItem(jtfDescricao.getText());

ModelItemItinerario.setIdTipoItem(idTipoItemItinerario.get(cbxTipoItemItinerario.getSe
lectedIndex() - 1));

        if (controllerItemItinerario.cadastrarItemItinerario(ModelItemItinerario))
        {
            JOptionPane.showMessageDialog(this, "Item cadastrado com sucesso",
"Atenção", JOptionPane.INFORMATION_MESSAGE);
            limparItensDeCadastro();
            carregarTabelaItensItinerario();
        } else {
            JOptionPane.showMessageDialog(this, "Erro ao cadastrar o item",
"Erro", JOptionPane.ERROR_MESSAGE);
        }
    }
}
}

```

8. Caso de Teste

```

@Test
public void testCadastrarItemItinerarioRepetido() {

    System.out.println("cadastrarItemItinerarioController");
    ControllerItemItinerario controller = new ControllerItemItinerario();
    List<ModelItemItinerario> listaItensCadastrados = new ArrayList<>();
    listaItensCadastrados = controller.getListItensItineraioController();

    ModelItemItinerario itemItinerario = listaItensCadastrados.get(1);


    assertEquals(false,
controller.cadastrarItemItinerarioController(itemItinerario));


}

```


9. Tela de Cadastro de Itinerários

Cadastro | Itens Itinerario

 **Cadastro de Itens de Itinerario**

TotemBus 

Cadastro

ID:

Descrição:

Tipo:

Selecione

Salvar

Alterar

Excluir

Limpar

Itens Cadastrados

Pesquisa

Digite um argumento:

Limpar pesquisa

ID	TIPO	DESCRIÇÃO
1	ESCOLA	IFMA/Monte Castelo
3	ESCOLA	Escola Adventista/Cidade Operária
5	SHOPPING CENTER	Rio Anil Shopping
7	SHOPPING CENTER	Pátio Norte
8	SHOPPING CENTER	Shopping São Luis
9	SHOPPING CENTER	Shopping da Ilha
10	HOSPITAL	Hospital São Domingos
12	HOSPITAL	UDI