



SENAI / SC  
Data: 12/11/2024

Professor Responsável:  
**Vitor Freitas Santos**

# **Protótipo de Gestão Hoteleira** **Funcionários**

**Time de Desenvolvimento:**

- Samara de Souza Bento
- Ana Julia Darela Bardini
- Jean Matei Rodrigues
- Jean Vitor Alano dos Reis
- Fabiano Barbosa Junior
- Taina Rigoni Dutra



## **1 – Introdução**

A Pousada Quinta do Ypuã, com seu ambiente acolhedor e sua crescente popularidade, tem buscado melhorar a eficiência de suas operações administrativas para proporcionar uma experiência cada vez mais agradável aos hóspedes. Para alcançar esse objetivo, foi iniciado o desenvolvimento de um sistema de gestão hoteleira sob medida, que irá centralizar e facilitar o controle de reservas, o gerenciamento de clientes e a organização interna. Essa solução visa automatizar processos críticos para a pousada, reduzindo o uso de planilhas manuais e o tempo gasto com atividades rotineiras.

O sistema será dividido em módulos, cada um focado em um aspecto específico da gestão da pousada. Um dos principais módulos é o de Funcionários, que visa simplificar o gerenciamento do time de colaboradores. Esse módulo permitirá aos gestores da pousada acompanhar a equipe e registros importantes dos funcionários, contribuindo para uma administração mais organizada e ágil. Ao otimizar a gestão de equipe, o módulo de funcionários fortalecerá a eficiência interna e apoiará a excelência do atendimento, alinhando-se ao objetivo da pousada de melhorar continuamente a experiência dos visitantes e de seus colaboradores.

## **2 – Visão Geral do Módulo**

O módulo de funcionários contribui diretamente para uma gestão mais eficiente da equipe, permitindo que os gestores da pousada tenham um controle organizado e simplificado das informações dos colaboradores. A possibilidade de acessar e atualizar dados rapidamente ajuda na organização de horários, escalas e na alocação de recursos humanos, o que resulta em uma operação mais fluida e eficiente. Ao manter o módulo sem funções adicionais, como cálculo de folha de pagamento e avaliações de desempenho detalhadas, o sistema se mantém fácil de usar e focado nas necessidades operacionais diretas da pousada.

O módulo de funcionários do sistema de gestão para a Pousada Quinta do Ypuã é uma ferramenta projetada para simplificar o gerenciamento de colaboradores. Ele fornece um conjunto básico de funcionalidades que abrange o registro, alteração, listagem e exclusão de funcionários, atendendo de forma eficaz às necessidades administrativas da pousada.

### **Funcionalidades do Módulo:**

Registro de Funcionários:



- Permite cadastrar novos funcionários, inserindo informações essenciais, como nome, cargo e salário.
- Facilita o armazenamento centralizado de dados, tornando-os acessíveis sempre que necessário para consultas ou atualizações.
- O cadastro garante que todas as informações relevantes estejam disponíveis para referência e gerenciamento.

#### Alteração de Funcionários:

- Os dados dos funcionários podem ser atualizados facilmente, permitindo a edição de informações como mudanças de cargo, ajustes ou atualização de cadastro.
- Essa funcionalidade é especialmente útil para acompanhar a evolução e mudanças na equipe ao longo do tempo, mantendo o registro atualizado e preciso.

#### Listagem de Funcionários:

- Permite visualizar uma lista completa de todos os funcionários cadastrados, exibindo informações básicas como nome, cargo e salário.
- Essa listagem é fundamental para uma visão geral da equipe, facilitando o planejamento de escalas e a consulta de dados de forma organizada.

#### Exclusão de Funcionários:

- A função de exclusão permite remover colaboradores que já não fazem parte da equipe da pousada, mantendo o sistema organizado e sem registros desatualizados.
- Inclui confirmações para evitar exclusões acidentais e assegurar que apenas os funcionários que realmente não pertencem mais ao quadro sejam removidos.

Embora o módulo de funcionários forneça um conjunto sólido de funcionalidades básicas, ele foi desenvolvido com o foco em um uso direto e simplificado. Dessa forma, algumas funções mais complexas não estão incluídas:

#### **O que o módulo não inclui:**

##### Cálculo de Folha de Pagamento:

- Esse módulo não realiza o cálculo de salário ou benefícios. A folha de pagamento deverá ser processada separadamente, utilizando outro sistema ou serviço.

##### Gestão de Benefícios e Incentivos:

- A administração de benefícios, como planos de saúde, vale-transporte e outros incentivos, não faz parte do escopo do módulo e deverá ser gerenciada externamente.

##### Avaliação de Desempenho Completa:

- O módulo não inclui relatórios de produtividade ou avaliações detalhadas de desempenho. As informações relacionadas à frequência e pontualidade são limitadas e devem ser complementadas com outras ferramentas se necessário.

### 3 – Protótipo Front-end

1.No Gerenciamento de Funcionários, na parte de cargos colocamos os disponíveis na empresa para cadastro ou alteração tanto de cargo, salário de uma forma sem pré definição.

```
<!-- Campos de Cargos -->
<select id="campo-cargo">
  <option value="">Selecione</option>
  <option value="Gerente">Gerente</option>
  <option value="Aux Administrativo">Aux Administrativo</option>
  <option value="Atendente">Atendente</option>
  <option value="Vendedor">Vendedor</option>
  <option value="Camareira">Camareira</option>
  <option value="Aux Limpeza">Aux Limpeza</option>
  <option value="Recepcionista">Recepcionista</option>
  <option value="Cozinheiro">Cozinheiro</option>
  <option value="Balconista">Balconista</option>
```

Colocamos no campo select, um campo de opções para poder aumentar o número de cargos possíveis e também para evitar erros ortográficos dentro do sistema de uma forma que o usuário/funcionário não precise cadastrar os cargos até então solicitados pela empresa.

2. Ainda no Gerenciamento de Funcionários, temos uma aba com salário a ser preenchido, podendo variar para mais ou para menos dependendo da negociação

```
<!-- Campo de salário-->
<div class="item-form">
  <label for="campo-salario">
    Salário
  </label>
  <input type="text" id="campo-salario" placeholder="Digite o Salário">
</div>
```

Colocamos o salário em uma função label, que consiste em servir basicamente de caixa para que o usuário coloque as informações referente ao salário.





### 3. Modelo visual do front Gerenciamento de Funcionários.

#### Gerenciamento de Funcionários

**Nome completo \***

  
**Data de nascimento \***  
**Documento \***  
**País:**  
**Estado:**  
**Cidade:**  
**Cargo: \***  
**Salário**  

**Salvar** **Limpar**

| ID | Nome completo          | Data de nascimento | Documento | País      | Estado         | Cidade | Cargo         | Salário |   |
|----|------------------------|--------------------|-----------|-----------|----------------|--------|---------------|---------|---|
| 1  | Rodrigo Vulcano Mendes | 08/08/1998         | 123456    | Argentina | Rio de Janeiro | Laguna | Recepcionista | 4580000 |   |

1 registros encontrados

O código JavaScript implementa a lógica de um módulo de gerenciamento de funcionários, utilizando localStorage para armazenar os dados localmente no navegador. Ele permite criar, editar, listar e excluir registros de funcionários. Abaixo estão os principais pontos do código:

#### 1. Inicialização e Armazenamento:

- Verifica se a lista de funcionários já existe no localStorage; caso contrário, inicializa um array vazio e o armazena.

#### 2. Carregamento do DOM:

- Executa a função listar() ao carregar o HTML para preencher a tabela com os dados armazenados.
- Adiciona eventos para os botões "Salvar" e "Cancelar".

### 3. Cadastro e Edição:

- O botão "Salvar" captura os valores dos campos de entrada, realiza validações obrigatórias e cria um objeto funcionario com os dados.
- Caso o ID já exista, ele edita o registro correspondente. Caso contrário, gera um novo ID e adiciona o funcionário à lista.
- Os dados são atualizados no localStorage, o formulário é resetado, e a tabela é atualizada.

### 4. Listagem de Funcionários:

- A função listar() exibe os dados de cada funcionário em uma linha da tabela HTML e adiciona botões de ação (editar e excluir) para cada registro.

### 5. Edição e Exclusão:

- Edição: Os botões de edição preenchem o formulário com os dados do funcionário selecionado, permitindo alterações.
- Exclusão: Os botões de exclusão removem o funcionário da lista e atualizam o localStorage e a tabela.,

### 6. Funções Auxiliares:

- getIndiceListaPorId(): Retorna o índice do funcionário no array com base no ID.
- getMaiorIdLista(): Obtém o maior ID atual na lista, facilitando a criação de novos IDs.
- resetarForm(): Limpa os campos do formulário para um novo cadastro.

Essas funcionalidades tornam o módulo de funcionários eficiente para gerenciamento local em uma interface de aplicação hoteleira.

```
// Cria objeto
let funcionario = {
  id: (id != "") ? id : getMaiorIdLista() + 1,
  nomeCompleto: nomeCompleto,
  dataNascimento: formatarDataParaBR(dataNascimento),
  documento: documento,
  pais: pais,
  estado: estado,
  cidade: cidade,
  cargo: cargo,
  salario: salario
};

// Altera ou insere uma posição no array principal
if (id != "") {
  let indice = getIndiceListaPorId(id);
  listaFuncionarios[indice] = funcionario;
} else {
  listaFuncionarios.push(funcionario);
}

// Armazena a lista atualizada no navegador
localStorage.setItem('listaFuncionarios', JSON.stringify(listaFuncionarios));

// Reseta o formulário e recarrega a tabela de listagem
this.blur();
resetarForm()

// Recarrega listagem
carregar("Salvo com sucesso!");
listar();
};
```

## 4 - Protótipo Back-end

### 1. Classe FuncionarioDAO

A classe FuncionarioDAO gerencia as operações de CRUD para a tabela funcionario no banco de dados MySQL. Utiliza consultas SQL para inserir, atualizar, deletar e buscar dados. Esta classe também usa o PessoaDAO para gerenciar os dados pessoais associados a cada funcionário.

### 2. Classe FuncionarioController

Essa classe gerencia as solicitações e as direciona para a camada de serviço. Utiliza a enumeração Funcionalidade para definir o tipo de operação.

```
package controller;

import enumeration.Funcionalidade;
import exception.FuncionarioException;
import model.Funcionario;
import test.FuncionarioTest;

import java.text.ParseException;

public class FuncionarioController { 3 usages

    // Atributos
    private FuncionarioTest funcionarioTest; 5 usages

    // Construtor que inicializa a instância de FuncionarioTest
    > public FuncionarioController(FuncionarioTest funcionarioTest) { this.funcionarioTest = funcionarioTest; }

    // Método para gerenciar e executar testes conforme a funcionalidade
    © public String testar(Funcionalidade funcionalidade) throws FuncionarioException, ParseException { 1 usage
        switch (funcionalidade){
            case LISTAR :
                return funcionarioTest.listar(); // Chama o teste de listagem
            case CADASTRAR:
                return funcionarioTest.cadastrar(); // Chama o teste de cadastro
            case ALTERAR:
                return funcionarioTest.atualizar(); // Chama o teste de atualização
            case EXCLUIR:
                return funcionarioTest.excluir(); // Chama o teste de exclusão
            default:
                return null; // Retorna null para funcionalidades desconhecidas
        }
    }
}
```

### 3. Classe FuncionarioService

Esta classe encapsula a lógica de negócio e validação de dados antes de interagir com a camada DAO. O método validarCampos verifica os campos obrigatórios e lança uma

```
public class FuncionarioService { 6 usages

    // Atributos
    // DAO de funcionários para acessar o banco de dados
    private FuncionarioDAO funcionarioDAO; 7 usages

    // Construtor
    // Construtor que inicializa o DAO de funcionários
    public FuncionarioService(FuncionarioDAO funcionarioDAO) { this.funcionarioDAO = funcionarioDAO; }

    // Método para listar todos os funcionários
    public String listar() throws FuncionarioException, ParseException { 1 usage
        ArrayList<Funcionario> funcionarios = funcionarioDAO.selecionar();
        String lista = "";
        if (funcionarios.size() > 0) {
            for (Funcionario funcionario : funcionarios) {
                lista += funcionario + "\n";
            }
        } else {
            lista = "Nenhuma pessoa encontrada.";
        }
        return lista;
    }
}
```

exceção com mensagens detalhadas se houver erros.

### 4. Classe FuncionarioTest

Implementa testes básicos para verificar o funcionamento das funcionalidades de listagem, cadastro, atualização e exclusão de funcionários.

```
public class FuncionarioTest { 6 usages

    // Serviço de funcionários usado nos testes
    private FuncionarioService funcionarioService; 5 usages

    // Construtor
    // Construtor que inicializa o serviço de funcionários
    public FuncionarioTest(FuncionarioService funcionarioService) { 1 usage
        this.funcionarioService = funcionarioService;
    }

    // Métodos de testes
    // Método para listar todos os funcionários
    public String listar() throws FuncionarioException, ParseException { 1 usage
        return funcionarioService.listar();
    }

    // Método para cadastrar um novo funcionário
    public String cadastrar() throws FuncionarioException { 1 usage
        return funcionarioService.cadastrar(
            "Sasá",
            "04/07/2004",
            "123",
            "Brasil",
            "SC",
            "Tubarão",
            "RH",
            1200.00
        );
    }
}
```



## 5. Classe FuncionarioException

Define exceções específicas para erros na manipulação de dados de funcionários. Permite mensagens de erro claras para o usuário.

```
1 package exception;
2
3 public class FuncionarioException extends Exception { 32 usages
4
5     // Método para exceções
6     public FuncionarioException(String mensagem) { 11 usages
7         super(mensagem);
8     }
9
10
11 }
12 |
```

## **5 – Conclusão**

O desenvolvimento do sistema de gestão hoteleira para a Pousada Quinta do Ypuã atingiu diversos objetivos planejados, contribuindo significativamente para a organização e eficiência da operação administrativa. Com a implementação do módulo de Funcionários, a pousada conseguiu centralizar o controle de colaboradores, incluindo o registro, listagem, atualização e exclusão de dados de forma simplificada. Esse módulo foi desenvolvido para focar nas necessidades operacionais diretas, evitando a inclusão de funcionalidades mais complexas, como o cálculo de folha de pagamento e gestão de benefícios, o que resultou em uma interface direta e de fácil uso para os administradores.

Entre os principais objetivos alcançados, destaca-se a otimização do tempo gasto nas atividades de cadastro e consulta de informações dos funcionários, o que facilitou a alocação de recursos humanos e o planejamento de escalas. Além disso, o projeto trouxe um aprimoramento na segurança e na precisão dos dados, com o uso de confirmações para exclusões e a possibilidade de atualização de informações. As dificuldades enfrentadas incluíram a integração entre as camadas de controle e serviço, especialmente para manter a consistência dos dados e garantir uma experiência fluida para os usuários. Os testes foram essenciais para assegurar o funcionamento correto e a robustez das operações de CRUD.

Para melhorias futuras, sugere-se ampliar as funcionalidades com a inclusão de módulos adicionais, como o de gestão de benefícios e relatórios de desempenho, que ajudariam a aumentar o controle administrativo e oferecer insights para decisões estratégicas. Além disso, a integração com sistemas de folha de pagamento externos poderia completar a gestão de recursos humanos de forma mais abrangente. Com esses aprimoramentos, o sistema pode evoluir para oferecer um suporte mais completo e automatizado às operações da pousada, melhorando a experiência tanto dos colaboradores quanto dos hóspedes.

## **6 – Referências**

Blog Egs Sistemas, 13/01/2023. Disponível em [<https://blog.egssistemas.com.br/posts/para-que-serve-o-cadastro-de-funcionario>] Acesso em 29/10/2024.

Arend, Felipe Gabriel. Geração de operações crud a partir de metadados(2011). Disponível em [[https://repositorio.ufsm.br/bitstream/handle/1/25212/TG328\\_Felipe%20Gabrie%20Arend.pdf?sequence=1](https://repositorio.ufsm.br/bitstream/handle/1/25212/TG328_Felipe%20Gabrie%20Arend.pdf?sequence=1)] Acesso 11/11/2024.