Instituto Infnet

JOÃO VITOR PEREIRA DE SOUZA

Desenvolvimento de Serviços Web e Testes com Java

Goiânia-GO
19/02/2023

JOÃO VITOR PEREIRA DE SOUZA

Desenvolvimento de Serviços Web e Testes com Java

Primeiro teste de performance da disciplina de Desenvolvimento de Serviços Web e Testes com Java .

Goiânia-GO
19/02/2023

Relatório

Verificação da escolha do contexto de negócio:

- Sistema de gerenciamento de pacientes em um Ambiente de saúde - Hospital
- Atributos: ID do paciente, Convênio vinculado ao paciente, seu nome e email respectivamente

```
public class PacienteEntity {
    3 usages
    private Integer idPaciente;
    3 usages
    private Integer convenioId;
    3 usages
    private String nome;
    3 usages
    private String email;
```

Pode ser instanciada a partir de um método de inserção na camada de Service, passando um construtor:

```
pacienteService.insertPaciente(new PacienteEntity(12, 3, "Santanas Ferreira", "joaquimferreira@gmail.com"));
```

Este método funciona da seguinte forma:

```
public PacienteEntity insertNewPaciente(PacienteEntity paciente) {
    List<PacienteEntity> pacientes = getListPacientes();
    paciente.setIdPaciente((pacientes.size() + 1));
    pacientes.add(paciente);
    System.out.println("Lista Atualizada de pacientes:");
    System.out.println("| ID | Nome | Idade |");
    for (PacienteEntity p : pacientes) {
        System.out.println("| " + p.getIdPaciente() + " | " + p.getNome() + " | " );
    }
    return paciente;
}
```

É utilizado um método que acessa o repositório de pacientes, com pacientes já instanciados por padrão, e logo após e adicionado o paciente recebido pelo método

de criação (insertNewPaciente), e após essa inserção ele irá mostrar no console o paciente criado e a lista atualizada de pacientes, veja abaixo:

```
Paciente criado: Santanas Ferreira
Lista Atualizada de pacientes:
| ID | Nome | Idade |
| 1 | Joaquim Ferreira |
| 2 | Maria Silva |
| 3 | Pedro Almeida |
| 4 | Ana Pereira |
| 5 | Lucas Souza |
| 6 | Santanas Ferreira |
```

- Competência 3:
  - ☑ Descreveu o contexto de negócio? Sim!
  - ☑ Possui 4 atributos relevantes ao contexto? Sim!
  - ☑ Validou a funcionalidade da classe? Sim!

- Validando o comportamento do método:

Foi criado um método de validação de email na classe do paciente, para que possa ser criado um paciente com email válido, utilizando REGEX (Expressão regular)

Exemplo da chamada ao método:

```
    PacienteService pacienteService = new PacienteService();
    PacienteEntity paciente = new PacienteEntity( idPaciente: 12,  conveniold: 3,  nome: "Santanas Ferreira",  email: "joaquimferreiragmail.com");
    if (paciente.validarEmail(paciente.getEmail())) {
        pacienteService.insertPaciente(paciente);
    } else {
        System.out.println("O e-mail fornecido não é válido. O paciente não será inserido.");
    }
}
```

Método:

```java
public boolean validarEmail(String email) {
    String regex = "^(.+)@(.+)$";
    return email.matches(regex);
}
```

Retorno do método para caso o email não seja válido:

```
/home/joaovsz/.jdks/openjdk-19.0.2/bin/java -javaagent:/snap/intellij-id
O e-mail fornecido não é válido. O paciente não será inserido.
```

E caso o paciente for inserido, ele retornará o nome do paciente que foi inserido e a lista de pacientes:

```
Paciente criado: Santanas Ferreira
Lista Atualizada de pacientes:
| ID | Nome | Idade |
| 1 | Joaquim Ferreira |
| 2 | Maria Silva |
| 3 | Pedro Almeida |
| 4 | Ana Pereira |
| 5 | Lucas Souza |
| 6 | Santanas Ferreira |
```

- Competência 4:
  - ☑ Analisou os valores do retorno? Sim!
  - ☑ Validou o comportamento do método? Sim!
  - ☑ Demonstrou a funcionalidade do método? Sim!

Explorando condições:

Inserção do paciente normal:

```java
PacienteEntity paciente1 = new PacienteEntity( idPaciente: 1,  conveniold: 3,  nome: "João Silva",  email: "joao.silva@email.com");
pacienteService.insertPaciente(paciente1);
```

Retorno:

```
Paciente criado: João Silva
Lista Atualizada de pacientes:
| ID | Nome | Idade |
| 1 | João Silva |
```

Inserção do paciente com id que já existe:

```
ID do paciente inválido ou duplicado. O paciente não será inserido.
```

Paciente com email inválido:

```
PacienteEntity paciente2 = new PacienteEntity( idPaciente: 2, conveniold: 3, nome: "Maria Souza", email: "maria.souza@invalido");
pacienteService.insertPaciente(paciente2);
```

Retorno:

```
/home/joaovsz/.jdks/openjdk-19.0.2/bin/java -javaagent:/snap/intellij
O e-mail fornecido não é válido. O paciente não será inserido.
```

Paciente com convênio negativo:

```
PacienteEntity paciente4 = new PacienteEntity( idPaciente: 3, conveniold: -1, nome: "Ana Lima", email: "ana.lima@gmail.com");
pacienteService.insertPaciente(paciente4);
```

Retorno:

```
/home/joaovsz/.jdks/openjdk-19.0.2/bin/java -javaagent:/snap/intel
ID do convênio inválido. O paciente não será inserido.
```