

Relatório do Marco 03

ConPorta - Testes e Integração dos casos de usos ProPorta e EdiPorta

01. ProPorta

Testes: ControladorProPortaTest.java

Controlador: ControladorProPorta.java

Funções testadas: buscarPessoas(), buscarPortarias(), buscarUndAdm(), salvar(Portaria portaria), PortariaDao.buscarPortaria(Long id);

Casos de Teste para o ControladorProPorta.java:

`casoTestBuscarPessoas()`

Verifica se os dados das Pessoas registrados na preparação do teste foram salvos corretamente.

Resultado Esperado: Os dados salvos na preparação do teste correspondem aos dados do recuperados do Banco de Dados.

Resultado Obtido:  `casoTestBuscarPessoas`

`casoTestBuscarUndAdm()`

Verifica se os dados das Unidades Administrativas registradas na preparação do teste foram salvos corretamente.


Resultado Esperado: Os dados salvos na preparação do teste correspondem aos dados do recuperados do Banco de Dados.

Resultado Obtido:  `casoTestBuscarUndAdm`

`casoTestBuscarPortarias()`

Verifica se os dados das Portarias registradas na preparação do teste foram salvos corretamente.

Resultado Esperado: Os dados salvos na preparação do teste correspondem aos dados do recuperados do Banco de Dados.

Resultado Obtido:  `casoTestBuscarPortarias`

`proPortaComDesignacaoComReferencia()`

Testa a função de salvar uma Portaria com Status de Proposta e com Designado e Referência.


Resultado Esperado: A portaria é gravada com seus dados correspondentes aos dados determinados no teste.

Resultado Obtido:  `proPortaComDesignacaoComReferencia`

`proPortaComDesignacaoSemReferencia()`

Testa a função de salvar uma Portaria com Status de Proposta e com Designado mas sem Referência.

Resultado Esperado: A portaria é gravada com seus dados correspondentes aos dados determinados no teste.

Resultado Obtido:  `proPortaComDesignacaoSemReferencia`

`proPortaSemDesignacaoComReferencia()`

Testa a função de salvar uma Portaria com Status de Proposta e sem Designado mas com Referência.

Resultado Esperado: A portaria é gravada com seus dados correspondentes aos dados determinados no teste.

Resultado Obtido:  `proPortaSemDesignacaoComReferencia`

`proPortaSemDesignacaoSemReferencia()`

Testa a função de salvar uma Portaria com Status de Proposta e sem Designado e sem Referência.


Resultado Esperado: A portaria é gravada com seus dados correspondentes aos dados determinados no teste.

Resultado Obtido:  `proPortaSemDesignacaoSemReferencia`

`proPortaRegraNegocioCamObri()`

Testa a regra de negócios camObri que define resumo, assunto, dtIniVig e expedidor como campos obrigatórios da portaria.


Resultados Esperados: Em caso de algum dos campos obrigatórios resultarem em não preenchidos retornará erro, caso contrário será possível gravar a portaria.

Resultado Obtido:  `proPortaRegraNegocioCamObri`

`proPortaTestTipoParametrosIncorretos()`

Testa se algum parâmetro opcional pode ser passado para o método controladorProPorta.salvar com seu tipo incorreto.

Resultados Esperados: Retornará mensagem de erro para cada parâmetro com tipo incorreto encontrado.

Resultado Obtido:  `proPortaTestTipoParametrosIncorretos`

`proPortaRegraNegocioAtribGerados()`

Testa a regra de negócio AtribGerados que define a geração dos atributos seqId e ultNumProp.

Resultados Esperados: Status da portaria é 'Proposta', Verifica seqId e ultNumProp

Resultado Obtido:  `proPortaRegraNegocioAtribGerados`

`proPortaTodosCamposPreenchidos()`

Testa a persistência de uma portaria salva com todos os campos preenchidos.

Resultado Esperado: Portaria salva com os dados enviados corretamente

Resultado Obtido:  `proPortaTodosCamposPreenchidos`

02. EdiPorta

Testes: ControladorEdiPortaTest.java

Controlador: ControladorEdiPorta.java


Funções testadas: buscarPessoas(), buscarPortarias(), buscarUndAdm(), salvar(Portaria portaria), buscarPortaria;

Casos de Teste para o ControladorEdiPorta.java:

`casoTestBuscarPessoas()`

Verifica se os dados das Pessoas registrados na preparação do teste foram salvos corretamente.


Resultado Esperado: Os dados salvos na preparação do teste correspondem aos dados do recuperados do Banco de Dados.

Resultado Obtido:  `casoTestBuscarPessoas`

`casoTestBuscarUndAdm()`

Verifica se os dados das Unidades Administrativas registradas na preparação do teste foram salvos corretamente.

Resultado Esperado: Os dados salvos na preparação do teste correspondem aos dados do recuperados do Banco de Dados.

Resultado Obtido:  `casoTestBuscarUndAdm`

`casoTestBuscarPortarias()`

Verifica se os dados das Portarias registradas na preparação do teste foram salvos corretamente.

Resultado Esperado: Os dados salvos na preparação do teste correspondem aos dados do recuperados do Banco de Dados.

Resultado Obtido:  `casoTestBuscarPortarias`

`buscarPortariaDesejadaTest()`

Testa a busca por uma portaria com status de Proposta pelo seu id, através do método `editarPortaria(Long idPortaria)`;

Resultado esperado: Retorna a portaria correspondente ao `idPortaria` mencionado.

Resultado Obtido:  `buscarPortariaDesejadaTest`

`buscarPortariaStatusInvalidoTest()`

Testa a busca por uma portaria com status diferente de Proposta pelo seu id, através do método `editarPortaria(Long idPortaria)`, como o método visa editar uma portaria, não deve obter sucesso na busca pela portaria.

Resultado Esperado: Retorna mensagem de erro devido ao status da Portaria não ser 'Proposta'.

Resultado Obtido:  `buscarPortariaStatusInvalidoTest`

`buscarPortariaInexistenteTest()`

Testa a busca por uma portaria inexistente pelo id da portaria, deve apresentar falha de Portaria não encontrada.


Resultado Esperado: Deve retornar a exceção de 'Portaria não encontrada'

Resultado Obtido:  `buscarPortariaInexistenteTest`

`editarPortariaSemReferenciaSemDesignadoTest()`

Testa a função de editar uma Portaria com Status de Proposta e sem Designado e sem Referência, atualizando os dados para a portaria que já estava 'proposta'.


Resultado Esperado: Os dados da portaria são atualizados de acordo com os novos dados fornecidos.

Resultado Obtido:  `editarPortariaSemReferenciaSemDesignadoTest`

`editarPortariaComReferenciaSemDesignadoTest()`

Testa a função de editar uma Portaria com Status de Proposta e sem Designado, mas com Referência, atualizando os dados para a portaria que já estava 'proposta'.


Resultado Esperado: Os dados da portaria são atualizados de acordo com os novos dados fornecidos.

Resultado Obtido:  `editarPortariaComReferenciaSemDesignadoTest`

`editarPortariaSemReferenciaComDesignadoTest()`

Testa a função de editar uma Portaria com Status de Proposta e com Designado, mas sem Referência, atualizando os dados para a portaria que já estava 'proposta'.

Resultado Esperado: Os dados da portaria são atualizados de acordo com os novos dados fornecidos.

Resultado Obtido:  `editarPortariaSemReferenciaComDesignadoTest`

`salvarPortariaComReferenciaComDesignadoTest()`

Testa a função de editar uma Portaria com Status de Proposta e com Designado e Referência, atualizando os dados para a portaria que já estava 'proposta'.

Resultado Esperado: Os dados da portaria são atualizados de acordo com os novos dados fornecidos.

Resultado Obtido:  `salvarPortariaComReferenciaComDesignadoTest`

`ediPortaRegraNegocioCamObri()`

Testa a regra de negócios camObri que define resumo, assunto, dtIniVig e expedidor como campos obrigatórios da portaria.

Resultado Esperado: Todos os campos obrigatórios preenchidos permitem salvar a portaria, caso algum esteja inválido o retorno é a mensagem de acordo com o campo.

Resultado Obtido:  `ediPortaRegraNegocioCamObri`