Universidade Fernando Pessoa
www.ufp.pt

ENGENHARIA DE SOFTWARE

Relatório ADD da Aplicação de Gestão de Serviços de Explicações no âmbito de Universidades

Porto, 10 Janeiro de 2020

Fábio Fernandes Carvalho nº: 35553
Mariana Reto nº: 36320

Ficha de estado do documento

1. Document Title : Clínica Médica			
2. Issue	3. Revision	4. Date	5. Reason For Change
1	0	Novembro 2019	Rascunho
1	1	Janeiro 2020	Alterações de Conteúdo

Índice

1. Introdução

Este documento descreve a abordagem desta aplicação de Gestão de Explicações. É baseado no URD e irá utilizar os casos descritos abaixo. Este documento será a base do planeamento da fase de desenvolvimento e será atualizado usando os resultados das várias iterações de desenvolvimento, seguindo o modelo escolhido no DIP, para concluir o projeto.

1.1. Objetivos

A aplicação terá como objetivos responder às necessidades dos utilizadores, sendo estes, os alunos, ou explicadores se for necessário, que terão possibilidades tais como ver disponibilidades de horários de explicações de uma universidade e também fazer a gestão das suas explicações. Irá responder ao explicador que vai poder ver as explicações que irá dar num determinado dia.

1.2. Âmbito da Aplicação

A aplicação foi desenvolvida a pensar num conjunto de universidades, sendo assim possível realizar todas as atividades previstas em múltiplas universidades, sendo esta instanciada em vários serviços como desejado.

2. Descrição Geral

A aplicação deve permitir ao aluno fazer pesquisas, marcar explicações e ainda verificar as explicações marcadas, entre outras funcionalidades, inclusive as pedidas no enunciado para a fase 1 e fase 2. A aplicação deve impedir o aluno de marcar uma explicação que resulte em sobreposição.

É possível o explicador verificar as explicações que também tem marcadas para um dado dia e disponibilidade, também é possível este editar informação.

2.1. Arquitetura da Aplicação

Na arquitetura do projeto foi definido que ia ser utilizada uma arquitetura Broker, seguindo o padrão MVC. O Broker que regista os pedidos do utilizador, vai buscar os dados desejados às várias instâncias do Web Service 1 e encaminha os resultados para o utilizador.

2. Componentes do Sistema

2.1. Controladores

Os controladores são importantes para mapear a forma de como recebemos a informação, vão passar a informação ao seu respetivo serviço, para este tratar os dados.

Na fase 1 temos 8 controladores, para os 8 modelos utilizados: Aluno, Explicador, Cadeira, Explicacao, Curso, Faculdade, Idioma e Disponibilidade.

Todos eles têm mapeamentos de *endpoints* como por exemplo o findAll (“get” ou “read do CRUD”) do repositório, têm o save (“post” ou “create do CRUD”) do repositório e também um update do CRUD ou “PUT”, na classe Aluno foi criado um exemplo de DELETE também.

Adicionalmente a este aínda existem os *endpoints* pedidos pelo cliente no enunciado.

Em cada controlador foram inseridos **Loggers** para verificar quando se recebe um certo pedido HTTP com sucesso.

2.2. Repositórios

Temos também 8 repositórios, assim como nos controladores, para cada modelo.

Os repositórios vão desempenhar um papel muito importante na aplicação, são objetos de acesso à informação, têm, portanto, um mecanismo de armazenamento, recuperação, pesquisa e atualização de dados.

Estes, são chamados pelos serviços, para conseguirem tratar a informação de forma dinâmica, tendo acesso à base de dados, onde podem manipular a informação de forma mais eficiente através de *queries*.

2.3. Serviços

Seguindo a mesma lógica, temos 8 serviços, também um para cada um dos modelos existentes.

Os serviços têm também um papel importante a desempenhar na aplicação, chamados pelos controladores, usam os repositórios para manipular a informação que lhes é passada, aqui é localizado maior parte do trabalho fulcral a ser feito por cada *endpoint.*

Por fim devolvem a resposta ao controlador que os chamou.

3. Modelos

Como mencionado anteriormente, temos 8 classes modelo, estas vão ser as tabelas das bases de dados, vão possuir coleções e em cada uma destas foi adicionado um **Padrão de Design Builder**, cada uma das coleções ou variáveis que representam ligações, do diagrama de classes, a outros modelos tem as suas devidas anotações (exp: @ManyToMany, etc...).

4. Testes Desenvolvidos

Foram feitos testes unitários, testes de integração, testes nos controladores e também nos repositórios.

Verificando-se uma enorme cobertura de testes por todo o projeto.

Os testes são extremamente importantes para provar a consistência e qualidade do código da aplicação e mesmo para ajudar a entender mais facilmente a existência de erros.

Os testes unitários são feitos para testar maioritariamente classes modelo, como métodos de inserção por exemplo de Alunos ou Explicadores.

Os testes de integração são feitos para testar a existência de erros de interface.

Os testes dos controladores foram feitos para testar o MVC da aplicação.

Os testes dos repositórios foram feitos para testar a informação com o JPA.

5. GitHub

Todos os detalhes do desenvolvimento, como os commits feitos ao longo do tempo pela equipa de trabalho encontram-se no GitHub, no seguinte link:
https://github.com/fabiocarvalho1998/Projeto-ESOFT

6. Codacy

O projeto foi colocado no Codacy para avaliação e foi cotado com a nota B.