

**Engenharia de Software**

**Ano letivo 2019/20**

**Architectural Design Document (ADD)**

**Detailed Design Document (DDD)**

**Álvaro Magalhães, No37000**

**Nikolaos Perris, No36261**

**Pedro Alves, No36651**

Índice

[1. Introdução 3](#_Toc29752767)

[1.1 Objetivos 3](#_Toc29752768)

[1.2 Âmbito da aplicação 3](#_Toc29752769)

[2. Descrição Geral 3](#_Toc29752770)

[2.1 Arquitetura da Aplicação 3](#_Toc29752771)

[3. Componentes do sistema 3](#_Toc29752772)

[3.1 Controladores 4](#_Toc29752773)

[3.2. Repositórios 4](#_Toc29752774)

[3.3. Serviços 4](#_Toc29752775)

[3.4 Modelos 4](#_Toc29752776)

[4. Testes desenvolvidos 4](#_Toc29752777)

[5. GitHub 5](#_Toc29752778)

[6. Codacy 5](#_Toc29752779)

# Introdução

Este documento descreve a abordagem desta aplicação sendo baseado no URD. 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.

## Objetivos

A aplicações terá como objetivos responder às necessidades dos utilizadores (explicadores ou alunos), que poderão ver disponibilidades de explicações e agendar, caso seja aluno, ou definir as disponibilidades, caso seja explicador.

## 1.2 Âmbito da aplicação

A aplicação foi desenvolvida a pensar num conjunto de universidades, porém o segundo serviço não foi desenvolvido com sucesso, de acordo com o tempo previsto, então só foi desenvolvido o 1º serviço, sendo assim possível realizar todas as atividades previstas mas apenas numa universidade, sem esta ser instanciada em vários serviços como desejado.

# Descrição Geral

A aplicação deve permitir ao aluno fazer pesquisas de explicadores por vários parâmetros , marcar explicações entre outras funcionalidades. A aplicação deve impedir o aluno de agendar uma explicação sobreposta a outra.

## 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 iria registar os pedidos do utilizador, vai buscar os dados desejados às várias instâncias do Web Service 1 e encaminha os resultados para o utilizador.

# Componentes do sistema

## 3.1 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: Appointment, Availability, College, COurse, Degree, Explainer, Language, Student.

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”.

Adicionalmente a este ainda 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.

## 3.2. 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.

## 3.3. 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.

## 3.4 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...).

# Testes desenvolvidos

Foram feitos testes aos controladores para testar o MVC da aplicação.

Foram feitos testes unitários para verificar as inserções.

# 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/pedroalvesk/EngenhariaSoftware>

# Codacy

O projeto foi colocado no Codacy para avaliação e foi cotado com rank A.