NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE INFORMÁTICA

SALVADOR GARCIA LUCAS CARDOSO DA SILVA

Licenciado em Engenharia Informática

FACILITAÇÃO DO USO DE ASP PARA A CRIAÇÃO DE HORÁRIOS

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa

Janeiro, 2025

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTAMENTO DE
INFORMÁTICA

FACILITAÇÃO DO USO DE ASP PARA A CRIAÇÃO DE HORÁRIOS

SALVADOR GARCIA LUCAS CARDOSO DA SILVA

Licenciado em Engenharia Informática

Orientador: Matthias Knorr
Professor Associado, Universidade NOVA de Lisboa

Júri

Presidente: João Manuel dos Santos Lourenço
Professor Associado, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Arguente: Vasco Miguel Gomes Nunes Manquinho
Professor Associado, Instituto Superior Técnico da Universidade de Lisboa

Orientador: Jörg Matthias Knorr
Professor Associado, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Janeiro, 2025

Facilitação do uso de ASP para a criação de horários

Este documento foi gerado com o processador (pdf/Xe/Lua)LaTeX e o modelo NOVAthesis (v7.1.18) [35].

Dedicada à minha família e amigos.

Agradecimentos

Por todas as orientações, esforços e acompanhamento que me proporcionou durante a fase de elaboração da presente dissertação, gostaria de agradecer ao professor Matthias Knorr. Adicionalmente, gostaria de agradecer à Faculdade de Ciências e Tecnologia, em especial ao departamento de informática e aos seus professores com quem tive o prazer de aprender.

Gostaria também de agradecer aos meus pais, por me terem proporcionado todas as condições que necessitei para a realização deste curso e por todos os conselhos que me deram. Agradecimento extensível ao meu irmão e avós.

Deixo ainda uma palavra aos meus amigos mais próximos por me terem acompanhado em todo o meu percurso académico e me terem ajudado sempre que precisei.

Resumo

Atualmente existe a necessidade, na Faculdade de Ciências e Tecnologia da Universidade NOVA de Lisboa, de possuir uma aplicação que permita o agendamento das apresentações dos documentos de preparação das dissertações, sem que se verifiquem sobreposições de horários, indisponibilidades de salas e/ou de algum/ns membro/s do júri designado/s para o efeito. A pertinência desta aplicação advém do facto de todos os alunos necessitarem de apresentar este documento num curto período de tempo e da possibilidade de um professor participar em várias apresentações.

Esta dissertação pretende, pois, dar resposta à conjugação de horários, sendo apresentado o desenvolvimento de uma aplicação, com recurso ao Answer Set Programming - Programação por Conjuntos de Resposta (ASP). A opção pelo ASP em detrimento de outras ferramentas, que são analisadas ao longo do documento, prende-se com o facto de apresentar inúmeras vantagens. Tratando-se de uma forma de programação declarativa, permite a resolução de diferentes problemas combinatórios, utilizando solvers eficientes na obtenção de soluções e requerendo como pressuposto a existência de conhecimento especializado do utilizador.

Para a construção destes horários, a composição dos júris e respetivas indisponibilidades, o horário e disponibilidade das salas, o horário e a duração das apresentações e a sobreposição de diferentes sessões nas quais participe o mesmo membro do júri, são requisitos que foram tidos em conta.

A aplicação desenvolvida é uma ferramenta que facilita ao utilizador responsável organizar os horários das apresentações, tanto a nível de recolha de informação, como na computação de um horário que satisfaça todas as restrições. Importa ainda sublinhar que cada utilizador poderá colocar individualmente as suas restrições e o organizador coloca as restrições restantes necessárias à computação do horário final, terminando com a aplicação a correr o programa ASP previamente desenvolvido.

Palavras-chave: ASP, Clingo, Horários, Aplicação

Abstract

There is currently a need at NOVA School of Science and Technology to have an application that allows presentations of dissertation preparation documents to be scheduled without overlapping schedules, unavailability of rooms and/or of any member(s) of the jury assigned for the purpose. The relevance of this application comes from the fact that all students need to present this document in a short period of time and that one teacher can take part in several presentations.

This dissertation, therefore, aims to respond to the combination of schedules by presenting the development of an application using ASP. The choice of ASP over other tools, which are analysed throughout the document, is due to the fact that it has numerous advantages. As it is a form of declarative programming, it allows different combinatorial problems to be solved, using efficient solvers to obtain solutions and requiring specialised knowledge on the part of the user.

To build these timetables, the composition of the juries and their respective unavailabilities, the time and availability of the rooms, the time and duration of the presentations and the overlapping of different sessions in which the same jury member participates are all requirements that have been taken into account.

The application developed is a tool that makes it easier for the responsible user to organise presentation schedules, both in terms of gathering information and computing a schedule that meets all the restrictions. It should also be noted that each user can individually place their own restrictions and the organiser places the remaining restrictions needed to compute the final timetable, ending with the application running the previously developed ASP program .

Keywords: ASP, Clingo, Timetables, Application

Índice

Índice de Figuras

Índice de Códigos Fonte

Glossário

API Abreviado de Application Programming Interface, é uma forma de dois ou mais programas conseguirem comunicar entre si [12]. *(pp. xxi, 3, 17, 18, 22, 33, 35, 36, 49, 54–56, 64)*

framework Estrutura de software reutilizável que fornece um conjunto de ferramentas para o desenvolvimento de aplicações. Fornece uma base sobre a qual os programadores podem modificar ou adicionar código [19]. *(pp. 34, 55, 67, 68)*

GitHub Plataforma web utilizada para controlo de versões no desenvolvimento de software. Permite que vários utilizadores trabalhem em projetos conjuntos [14]. *(pp. 14, 33, 50, 51, 55, 65)*

JDBC Abreviado de Java Database Connectivity, é uma API da linguagem Java utilizada para interagir com bases de dados relacionais [15]. *(p. 33)*

JPA Abreviado de Java Persistence API, é um conjunto de regras que facilita o mapeamento entre objetos e a base de dados em aplicações Java. Permite que classes Java sejam associadas a tabelas da base de dados relacionais, simplificando as operações CRUD, não havendo a necessidade de escrever diretamente comandos SQL. *(p. 36)*

npm Gestor de pacotes para o *Node.js*. Permite aos programadores instalar e compartilhar bibliotecas e módulos [38]. *(p. 68)*

plug-ins Programas de computador utilizados para adicionar funcionalidades especiais ou específicas a programas de maior dimensão [17]. *(p. 35)*

POST Método suportado pelo protocolo HTTP, utilizado para enviar dados a um servidor para o seu processamento [18]. *(pp. 33, 44, 56, 59)*

salt Valor adicional, gerado aleatoriamente, utilizado no processo de *hashing*. No contexto do *bcrypt*, o salt é combinado com a senha antes de aplicar a função de hashing [3]. *(p. 47)*

SVG Abreviado de Scalable Vector Graphic, é um formato de imagem vetorial baseado em XML, usado para descrever gráficos bidimensionais [47]. *(p. 68)*

SIGLAS

ASP Answer Set Programming - Programação por Conjuntos de Resposta *(pp. vii, ix, xix, 1–3, 5, 6, 9, 11–15, 17, 19, 22, 48–51, 54, 56, 58, 59, 61–64, 109)*

CRUD Create, Read, Update, Delete - Criação, Leitura, Atualização, Exclusão *(pp. xxi, 36, 38)*

DAO Data Access Object - Objeto de Acesso de Dados *(pp. 28, 38)*

DTO Data Tranfer Object - Objeto de Transferência de Dados *(pp. 37, 38)*

HTTP Hypertext Transfer Protocol - Protocolo de Transferência de Hipertexto *(pp. xxi, 35–38, 47, 54, 56, 68, 69)*

IDE Integrated Development Environment - Ambiente de Desenvolvimento Integrado *(pp. 12, 13)*

JSON JavaScript Object Notation - Notação de Objetos JavaScript *(pp. 36, 44, 56, 59, 64)*

JVM Java Virtual Machine - Máquina Virtual Java *(p. 34)*

JWT JSON Web Token *(pp. 36, 46, 47, 68)*

ORM Object-relational mapping - Mapeamento Objeto-Relacional *(p. 18)*

POM Project Object Model - Projeto Modelo de Objeto *(p. 35)*

REST Representational State Transfer - Transferência Representacional de Estado *(pp. 3, 22, 33, 36, 49, 54, 64, 68)*

SQL Structured Query Language - Linguagem de consulta estruturada *(pp. xxi, 27, 36)*

VSCode Visual Studio Code *(pp. 13, 14)*

1

Introdução

1.1 Contexto

O Answer Set Programming - Programação por Conjuntos de Resposta (ASP) é uma forma de programação declarativa que permite a solução de diferentes problemas combinatórios utilizando solvers eficientes na obtenção de soluções [34]. Os programas, por norma, são bastante compactos e rápidos de escrever.

Uma das principais vantagens é a capacidade que o ASP tem de tratar as incertezas e a informação incompleta, uma vez que permite a computação de múltiplas soluções.

O ASP tem sido bastante utilizado em diversas aplicações, marcando presença nas áreas da bioinformática e da robótica, entre outras [22]. Esta forte presença provém da facilidade em manter-se atualizada, uma vez que a sua programação é feita de forma declarativa. Mostra-se também como um boa solução para a resolução de problemas de configurações [23].

Apesar de ter inúmeras vantagens, apresenta, como requisito fundamental, a necessidade de conhecimento especializado do utilizador sobre o ASP.

O desafio de elaborar horários, designadamente para apresentações de documentos de preparação de dissertações num curto espaço de tempo, através da utilização de ASP e sem a necessidade de conhecimento especializado do utilizador, motivou a elaboração da presente dissertação.

1.2 Definição do problema

Considerando que todos os alunos para começarem a elaboração de uma dissertação de mestrado necessitam de efetuar uma apresentação do documento de preparação da dissertação, apresentações essas que ocorrem num curto espaço de tempo, é imprescindível um prévio agendamento de forma a acomodar cada uma das apresentações sem ocorrerem

sobreposições dos membros dos júris. Tendo em consideração o elevado número de apresentações a realizar, é de extrema utilidade a existência de uma ferramenta fácil e de simples utilização que dê resposta a esta necessidade premente.

Ao realizar-se o processo de elaboração dos horários, é necessário ter em consideração os seguintes requisitos:

- Prazo das apresentações;
- Salas disponíveis;
- Horário de utilização de cada sala;
- Composição do júri;
- Disponibilidade de cada membro do júri;
- Possibilidade de um professor ser membro de diferentes júris;
- Duração de cada apresentação.

Este processo de encontrar uma combinação possível é complexo e demorado. Uma das possibilidades de resolver o problema é o recurso ao ASP e ao solver clingo, no entanto, são tecnologias que requerem um conhecimento especializado por parte dos utilizadores.

1.3 Objetivo

O objetivo foi criar uma aplicação que sugere a melhor combinação de horários das várias apresentações de forma eficiente, eficaz e efetiva.

A aplicação que desenvolvemos facilita o trabalho da pessoa organizadora que, até agora, tinha a tarefa de recolher a informação de todos os intervenientes individualmente e compilar esta informação recolhida em conjunto com os recursos físicos disponíveis.

A nossa aplicação tem uma interface simples e intuitiva de modo a que os utilizadores consigam usufruir das suas funcionalidades sem terem conhecimentos de ASP. Esta interface permite aos membros dos júris introduzir diretamente as suas indisponibilidades horárias.

Uma vez incluídas as especificidades temporais de todos os membros dos júris na aplicação, esta gera automaticamente uma combinação de horários, onde serão visíveis as salas e os respetivos intervenientes.

Uma das funcionalidades da nossa aplicação é a possibilidade de efetuar o download do horário completo. Esta funcionalidade permite, não só a publicação deste documento na plataforma *Clip* para a visualização de todos os intervenientes, como também a sua impressão e afixação nas portas das salas onde vão decorrer as apresentações.

1.4 Estrutura do documento

Esta dissertação está organizada em nove capítulos, que cobrem os diferentes aspetos do trabalho realizado, desde o enquadramento teórico até à implementação e validação da solução desenvolvida.

- **Capítulo 2: Enquadramento** — Neste capítulo é apresentado o conceito de Answer Set Programming (ASP), a sua sintaxe e as suas principais características. São também introduzidos os solvers disponíveis para ASP, as funções aplicadas a conjuntos e as ferramentas de edição utilizadas no desenvolvimento do código ASP.
- **Capítulo 3: Estado da Arte** — Este capítulo discute as diferentes abordagens existentes para a criação de horários. Além disso, é explorada a interação do ASP com diversas linguagens de programação, analisando as possibilidades de integração entre diferentes tecnologias.
- **Capítulo 4: Aplicação** — Neste capítulo é descrita a aplicação desenvolvida, começando pela sua arquitetura geral, passando pela categorização dos tipos de utilizadores, e apresentando os user stories que guiaram o desenvolvimento das funcionalidades.
- **Capítulo 5: Base de Dados** — Este capítulo detalha a base de dados utilizada no sistema, destacando a escolha da tecnologia H2 e explicando o desenho das tabelas e a estrutura adotada.
- **Capítulo 6: Servidor** — Começamos por descrever as tecnologias utilizadas na implementação do servidor, a sua arquitetura, as funcionalidades e operações principais, bem como a integração com o microsserviço responsável pela computação dos horários. Também são discutidos os mecanismos de segurança aplicados à gestão dos utilizadores.
- **Capítulo 7: Microsserviço** — Neste capítulo é apresentado o microsserviço responsável pela computação dos horários utilizando ASP. São descritos o programa ASP utilizado e a API Representational State Transfer - Transferência Representacional de Estado (REST) que permite a comunicação entre o microsserviço e o servidor da aplicação.
- **Capítulo 8: Cliente** — Este capítulo foca-se na parte cliente da aplicação, descrevendo as tecnologias adotadas, a arquitetura do Frontend, a comunicação com o servidor, e

detalhando a navegação e as principais páginas e funcionalidades disponibilizadas aos utilizadores.

- **Capítulo 9: Testes** — São apresentados os testes realizados à aplicação, com o objetivo de garantir o correto funcionamento e a fiabilidade da solução proposta.
- **Capítulo 10: Conclusão** — O capítulo final apresenta, de forma resumida, a estrutura da aplicação e feedback da sua utilização. Adicionalmente propõe possíveis melhorias e direções para trabalho futuro.

2

Enquadramento

Neste capítulo vamos começar por explicar o que é o ASP, bem como os processos necessários para a escrita de um programa. Posteriormente iremos explicar o que é o Gringo, o Clasp e o Clingo. Por fim mencionaremos algumas das funções que podem ser aplicadas na escrita de programas em ASP.

2.1 O que é o ASP?

O ASP é uma forma de programação declarativa que permite a solução de problemas combinatórios, principalmente problemas do tipo NP-difícil. O seu uso foi identificado como um paradigma de programação em 1999 e desde então tem vindo a ganhar popularidade, derivado do facto de conseguir resolver problemas complexos [34].

Em ASP, os problemas são modelados como conjuntos de regras lógicas, que são depois processados por um solver, de modo a produzir conjuntos de respostas. Cada conjunto representa uma solução possível.

Uma vantagem da sua utilização é o facto dos programas serem compactos e rápidos de escrever. O programador pode concentrar-se em descrever o problema em vez de pensar no algoritmo de pesquisa que pretende utilizar [5].

Outra vantagem é a capacidade que o ASP tem de tratar as incertezas e a informação incompleta, uma vez que permite a computação de múltiplas soluções, possivelmente contraditórias.

2.2 Escrita de um programa ASP

A escrita de um programa em ASP pode ser dividida em 2 partes. Uma primeira onde o programador deve instanciar o problema e uma segunda onde é feita a codificação do mesmo. Por fim é necessário a execução do solver sobre o programa de forma a ser obtido o resultado.

Para este fim, iremos utilizar o código e a imagem do exemplo da torre de Hanoi, retirado do "Potassco user guide" [26], disponibilizado na página principal do potassco[1].

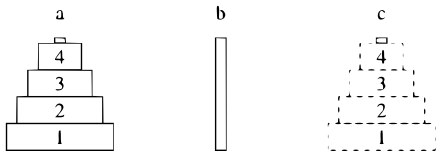

Figura 2.1: Torre de Hanoi: Situação inicial e final

Este exemplo consiste numa base com três pinos e em quatro discos de diferentes tamanhos. Inicialmente, os discos estão colocados, por ordem decrescente de diâmetro, no pino da esquerda. O objetivo é mover os discos para o pino da direita, como é possível observar na figura 2.1. Para isto, um disco não pode ser movido para um pino que contenha outro disco de diâmetro menor.

2.2.1 Instanciar o Problema

Inicialmente, quando um programador está a escrever um programa em ASP, deve instanciar o problema. Para isto, é necessário especificar os factos do problema. Um exemplo é o seguinte:

Código Fonte 2.1: Criação dos factos do programa "Torre de Hanoi"

```
1 peg(a;b;c).
2 disk(1..4).
3 init_on(1..4,a).
4 goal_on(1..4,c).
5 moves(15).
```

A descrição dos pinos é feita com o predicado *peg/1*, utilizando o *";"* de forma a indicar a existência dos pinos *a*, *b* e *c*.

Os discos são descritos usando o predicado *disk/1* e estão numerados de 1 a 4, sendo que, quanto maior é o número, menor é o valor do seu diâmetro.

A utilização da sintaxe *"1..4"* refere o intervalo de números inteiros de 1 até 4.

Os predicados *init_on/2* e *goal_on/2* traduzem, respetivamente, o posicionamento inicial e final de cada disco. O primeiro argumento de ambos indica o número do disco e o segundo argumento indica a letra do pino onde se encontra.

[1] https://potassco.org/

O predicado *moves/1* representa o número de movimentos com que o problema deve ser resolvido.

2.2.2 Codificação do Problema

Depois de indicar os factos do problema, é necessário codificar um gerador para gerar diversos conjuntos de resposta e codificar as restrições que irão atuar sobre estes conjuntos. Estas restrições vão ser utilizadas pelo solver na resolução do programa.

A definição das restrições do problema da torre de Hanoi pode ser ilustrada da seguinte forma:

Código Fonte 2.2: Criação das restrições do programa "Torre de Hanoi"

```
6  % Generate
7  1 { move(D,P,T) : disk(D), peg(P) } 1 :- moves(M), T = 1..M.
8  % Define
9  move(D,T) :- move(D,_,T).
10 on(D,P,0) :- init_on(D,P).
11 on(D,P,T) :- move(D,P,T).
12 on(D,P,T+1) :- on(D,P,T), not move(D,T+1), not moves(T).
13 blocked(D-1,P,T+1) :- on(D,P,T), disk(D), not moves(T).
14 blocked(D-1,P,T) :- blocked(D,P,T), disk(D).
15 % Test
16 :- move(D,P,T), blocked(D-1,P,T).
17 :- move(D,T), on(D,P,T-1), blocked(D,P,T).
18 :- goal_on(D,P), not on(D,P,M), moves(M).
19 :- not 1 { on(D,P,T) : peg(P) } 1, disk(D), moves(M), T = 1..M.
20 % Display
21 #show move/3.
```

É importante referir que todas as linhas que começam com % são comentários e que as variáveis *D*, *P*, *T* e *M* representam, respetivamente, os discos, os pinos, o número de movimentos já realizados e o número de movimentos com que o problema deve ser resolvido.

A parte % *Generate* cria todas as soluções candidatas para a resolução deste problema. Neste sentido, cria um predicado *move/3*, novo, que a cada movimento incrementa o valor de *T*. As soluções candidatas criadas são uma tentativa de acertar os movimentos a ser executados para a resolução do problema.

Na parte % *Define* são criados predicados auxiliares de modo a não considerar as soluções candidatas que não se confirmam como corretas. Na linha 9 é utilizado "_" para indicar que o valor de *P* é insignificante para o novo predicado *move/2*. Este novo predicado indica o movimento de um disco para um pino. O predicado *on/3* indica em que pino está um disco passados *T* movimentos. A linha 10 indica o pino onde se encontram os discos antes de se começarem a fazer movimentos. As linhas 11 e 12 indicam os estados em que

os discos se encontram depois de *T* movimentos. A linha 13 aponta que, quando existe o movimento de um disco e os outros continuam no mesmo pino, o número de movimentos é incrementado no predicado *on/3* de todos os discos, uma vez que o *T* representa o número de movimentos total e não o de cada disco. Por fim, é criado o predicado *blocked/3* para bloquear o movimento dos discos, com comprimento de diâmetro maior que o de cima, na jogada seguinte. É ainda importante referir que é feito o bloqueio quando *D-1=0*, uma vez que irá facilitar a parte de teste.

A parte % *Test* tem como função eliminar as soluções candidatas que não formam a solução correta. Para isto são indicadas as situações "ilegais" e o solver elimina as soluções que as contêm. A primeira restrição na linha 16 garante que um disco *D* não pode ser movido para um pino que contenha o disco seguinte com maior diâmetro e este esteja bloqueado, excluindo os movimentos em que um disco maior fica por cima de um mais pequeno e movimentos de mover um disco para o mesmo sítio. A linha 17 restringe os movimentos de todos os discos que estão bloqueados por ter um disco mais pequeno em cima. A restrição na linha 18 garante que todos os discos estão no lugar pretendido depois de serem executados o número de movimentos indicados na linha 5. A última restrição implica que todos os discos estão sempre em algum pino. Esta restrição não é necessária para o funcionamento do programa, no entanto, melhora significativamente o seu tempo de resolução.

Por fim, a parte % *Display* imprime o predicado *move/3* da solução, quando o programa é executado.

2.2.3 Solução do problema

A execução do programa escrito anteriormente é feita com o auxilio da ferramenta clingo. Esta ferramenta é evocada através do terminal com o seguinte comando:

```
1  clingo exemplo.lp
```

Ao executar este comando no terminal o solver vai imprimir:

```
1     clingo version 5.4.0
2     Reading from ...exemplo.lp
3     Solving...
4     Answer: 1
5     move(4,b,1) move(3,c,2) move(4,c,3) move(2,b,4) move(4,a,5)
6     move(3,b,6) move(4,b,7) move(1,c,8) move(4,c,9) move(3,a,10)
7     move(4,a,11) move(2,c,12) move(4,b,13) move(3,c,14) move(4,c,15)
8     SATISFIABLE
9
10    Models       : 1+
11    Calls        : 1
12    Time         : 0.012s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
13    CPU Time     : 0.016s
```

A primeira linha indica a versão do clingo. As duas linhas seguintes indicam o estado do clingo. Depois de o programa ser resolvido, é impressa a solução seguida do estado. Neste caso o estado é *"SATISFIABLE"*, no entanto existem outras possibilidades, como *"UNSATISFIABLE"* e *"UNKNOWN"*. A linha com *"Models : 1+"* impresso, indica que foi encontrada uma solução para o problema. A linha seguinte indica quantas vezes foi invocado o solver e as ultimas duas linhas mostram algumas estatísticas.

2.3 Gringo, Clasp e Clingo?

Neste ponto iremos explorar o solver Clingo e como ele combina o Gringo e o Clasp na resolução de programas ASP.

O Gringo recebe um programa com variáveis de primeira ordem e transforma-o num programa sem variáveis, uma vez que a maioria dos solvers trabalham com conjuntos de resposta sem variáveis. O resultado pode ser processado pelo solver Clasp. A sua principal função é lidar com a transformação e simplificação do código [25].

O Clasp é um solver utilizado na resolução de problemas escritos pro conjuntos de resposta. Este solver é conhecido pela sua eficiência em lidar com problemas complexos [25].

O Clingo combina o Gringo e o Clasp num mesmo sistema, oferecendo mais controlo e processamento de resolução de problemas [25]. Desenvolvido pela "University of Potsdam" na Alemanha, chamada Potassco (Potsdam Answer Set Solving Collection), o Clingo permite que os utilizadores escrevam programas em ASP e, de seguida, utiliza o Gringo para pré-processar as regras e chama o Clasp para a resolução do programa.

2.4 Funções aplicadas em conjuntos

Existem diversas funções que podem ser aplicadas aos conjuntos de forma a facilitar a programação em ASP [33].

2.4.1 Counting

A sintaxe *#count* pode ser utilizada para calcular o número de elementos de um conjunto, como por exemplo:

Código Fonte 2.3: Exemplo da sintaxe #count

```
1 N = #count{Y : aresta(X,Y)}
```

Neste exemplo, N representa o número de pontos Y que existem na aresta X.

2.4.2 Soma

A sintaxe *#sum* pode ser utilizada para calcular a soma de conjuntos de inteiros, como por exemplo:

Código Fonte 2.4: Exemplo da sintaxe #sum

```
number(1).
number(2).
number(3).
number(4).
res(P) :- P = #sum{X : number(X)}.
```

Neste exemplo, o predicado *res/1* fica com a soma de todos os números existentes.

2.4.3 Máximo e Mínimo

As sintaxes *#max* e *#min* podem ser utilizadas para representar, respetivamente, o maior e menor valor de um elemento de um conjunto, como por exemplo:

Código Fonte 2.5: Exemplo das sintaxes #max e #min

```
#max{X : p(X)}
#min{X : p(X)}
```

Neste exemplo, são calculados os valores máximo e mínimo presentes no predicado *p/1*.

2.4.4 Maximização e Minimização

A sintaxe *#maximize* pode ser utilizada para otimizar o primeiro modelo estável encontrado pelo clingo. Por exemplo:

Código Fonte 2.6: Exemplo da sintaxe #maximize

```
empregado("Carlos", 10).
empregado("Francisco", 20).
empregado("Miguel", 15).

1{maisBemPago(X):empregado(X,Y)}1.

#maximize{ Salario : empregado(Nome, Salario) , maisBemPago(Nome)}.
#show maisBemPago/1.
```

Neste exemplo, o clingo irá maximizar o valor do salário dos empregados todos, encontrando o que recebe mais. A solução seria impressa da seguinte forma:

```
    Clingo version 5.4.0
    Reading from ...[nome do programa].lp
    Solving...
    Answer: 1
    maisBemPago("Francisco")
    Optimization: -20
    OPTIMUM FOUND
```

```
8
9     Models       : 1
10      Optimum    : yes
11    Optimization : -20
12    Calls        : 1
13    Time         : 0.002s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
14    CPU Time     : 0.000s
```

É possível perceber que o clingo encontrou o empregado mais bem pago.

No mesmo exemplo, caso o objetivo seja encontrar o empregado que recebe menos, usaria-se a sintaxe *#minimize* em vez de *#maximize*. Também se deve alterar o predicado *maisBemPago/1* para *menosBemPago/1*, por uma questão de perceção. O resultado deste programa seria um pouco diferente:

```
1     clingo version 5.4.0
2     Reading from ...[nome programa].lp
3     Solving...
4     Answer: 1
5     menosBemPago("Francisco")
6     Optimization: 20
7     Answer: 2
8     menosBemPago("Carlos")
9     Optimization: 10
10    OPTIMUM FOUND
11
12    Models       : 2
13      Optimum    : yes
14    Optimization : 10
15    Calls        : 1
16    Time         : 0.002s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
17    CPU Time     : 0.000s
```

É possível ver que o clingo apresentou duas soluções, sendo a última a correta. Isto acontece porque o clingo começa com o primeiro valor encontrado e vai comparando os seguintes até encontrar um menor. Caso encontre um menor, apresenta-o.

2.5 Ferramentas de edição do ASP

Para facilitar o uso do ASP, existem várias ferramentas de edição de programas. Neste ponto iremos enumerar as mais conhecidas, identificando os seus pontos fortes e pontos fracos.

2.5.1 Editores de texto

Os editores de texto suportam qualquer tipo de linguagem de programação, sendo assim uma opção para a escrita de programas em ASP. A execução destes programas tem de ser feita através do terminal.

Por exemplo, o Notepad++ é um editor de texto muito conhecido, utilizado no sistema operativo windows, que suporta várias linguagens de programação, entre elas o ASP [28]. A sua utilização não necessita de uma ligação à internet.

Para escrever um programa em ASP utilizando o Notepad++, o utilizador começa por criar um novo ficheiro, sendo necessário alterar a linguagem predefinida para ASP.

Para executar o programa é necessário o utilizador ter uma instalação local do clingo no seu computador e "chamá-lo"através do terminal. Ao ter esta instalação local, o utilizador tem liberdade para especificar como quer que o clingo seja executado. Por exemplo, o utilizador pode escolher o número de soluções que vão ser impressas, caso o problema tenha solução. Outro ponto positivo de ter uma instalação local do clingo é a possibilidade do utilizador escrever e executar os seus programas sem ser necessária uma ligação à internet.

2.5.2 LoIDE

O LoIDE é um Integrated Development Environment - Ambiente de Desenvolvimento Integrado (IDE) online [48] onde é possível escrever programas em ASP e executá-los com diferentes solvers.

Este IDE é bastante intuitivo no que diz respeito à escrita e execução dos programas. Aberto o editor, o utilizador pode começar a escrever de imediato e sempre que necessitar de executar o programa, pode faze-lo utilizando o botão de grande dimensão e visibilidade no topo da página com a designação "Run!".

O LoIDE tem ainda um campo para escolhas livres, onde o utilizador pode especificar o que pretende do seu programa. Existe um manual de instruções para facilitar a perceção da sintaxe, no entanto, este manual é de difícil entendimento por parte de um utilizador sem os devidos conhecimentos sobre o tema.

Um dos pontos fortes deste IDE é o facto de permitir a partilha do ficheiro entre vários utilizadores através de um link, evitando o envio do ficheiro completo, no entanto esta partilha não possibilita que ambos os utilizadores escreverem no ficheiro em simultâneo.

Outro ponto forte é a possibilidade de manter vários ficheiros abertos e, caso se pretenda, executá-los em conjunto [36].

A utilização do LoIDE é possível em qualquer computador com uma ligação à internet. Esta funcionalidade pode ser vista como um ponto positivo e como um ponto negativo. O facto de ser possível utilizar este editor quando há uma ligação à internet, por um lado, faz com que não seja necessário o utilizador ter uma instalação local do clingo no computador

que está a utilizar, por outro lado, caso um utilizador não disponha de um acesso à internet não consegue executar os seus programas.

Este IDE não tem a versão mais atualizada do clingo, no entanto, uma vez que o código é open-source, é possível descarregá-lo, fazer esta atualização e executar o programa num servidor local.

2.5.3 Sealion

O Sealion é um plugin no Eclipse utilizado como IDE para a escrita de programas em ASP.

Foi concebido com o intuito de melhorar a produtividade dos programadores de ASP. Com o passar do tempo foram realizados diversos estudos, cujos resultados serviram para melhorar a qualidade e eficiência do desenvolvimento em ASP.

O seu principal objetivo é a disponibilização de um editor no qual seja fácil os utilizadores escreverem, editarem, executarem e fazerem o debug de programas em ASP.

Neste IDE existem configurações especiais para os programas Gringo, Clasp e DLV. No entanto também existe suporte para a utilização de outras de ferramentas externas.

O Sealion é o primeiro IDE que permite a um utilizador fazer debug do seu programa em tempo real. A estratégia implementada é a forma de debug por fases, ajudando os programadores a detetar facilmente a localização dos erros [41, 39].

2.5.4 iGROM

O iGROM é um IDE utilizado na escrita de programas em ASP que utiliza unicamente o solver DLV para os executar, o que significa que não tem suporte para clingo [29]. O iGROM é normalmente utilizado como um plugin no Eclipse.

Acresce que para a utilização deste IDE são necessárias competências aprofundadas no ASP, limitando deste modo a sua utilização generalizada por parte de utilizadores comuns.

2.5.5 Visual Studio Code (VSCode)

O Visual Studio Code (VSCode) é um editor de código, disponível em Windows, macOS e Linux, que suporta várias linguagens de programação [21]. Este editor dispõe de uma biblioteca de extensões que fornecem recursos e funcionalidades adicionais. Grande parte das extensões foram criadas para facilitar a escrita e a compilação de programas nas diversas linguagens existentes.

Para ASP existem duas extensões:

1. **Answer Set Programming syntax highlighter** — Esta extensão ajuda o utilizador a visualizar melhor a sintaxe enquanto escreve o programa [1].

2. **Answer Set Programming Language Support** — Para se utilizar esta extensão é necessário, primeiramente, instalar a extensão referida no ponto anterior. De seguida, para executar um programa, o utilizador pode escolher a opção que deseja. Uma das opções é imprimir todas as soluções no terminal, outra é imprimir apenas a primeira solução encontrada, por fim, é igualmente permitido ao utilizador escrever um ficheiro de configurações onde especifica as opções que pretende ver realizadas pelo solver. Importa, ainda, referir que esta extensão utiliza a versão mais recente do solver clingo para executar os programas e facilita a sua utilização. [24].

Além das vantagens que estas extensões apresentam, o facto de se utilizar o VSCode também apresenta vantagens. Este editor disponibiliza uma ligação rápida e simples com o GitHub, permite a escrita simultânea de vários programadores num mesmo ficheiro, dispõe de um acesso rápido a um terminal, entre outras.

2.6 Conclusão

Embora o solver *Clingo* e as ferramentas de edição do ASP proporcionem um ambiente capaz para a modelação e resolução de problemas complexos, o uso de ASP continua a ser um desafio para utilizadores comuns. A modelação em ASP exige um conhecimento avançado sobre lógica formal e uma compreensão detalhada da sintaxe e semântica do solver. Mesmo com as ferramentas disponíveis, a criação de um modelo eficiente e a interpretação correta dos resultados requerem experiência e familiaridade com a programação por conjuntos de resposta. Para utilizadores não especializados, a complexidade envolvida pode dificultar a adoção do ASP como solução para problemas de alocação de recursos e horários a apresentações.

3

Estado da Arte

Este capítulo irá abordar as alternativas atualmente existentes no que respeita à criação de horários, incluindo as ferramentas de edição mais frequentemente utilizadas pelos programadores de ASP, bem como referir as possibilidades de interação entre diversas linguagens de programação e o ASP. O capítulo irá terminar com a fundamentação das decisões que pretendemos tomar.

3.1 Criação de horários

Neste ponto iremos expor algumas ferramentas que podem ser adaptadas para gerir horários, enumerando algumas vantagens e desvantagens das mesmas.

3.1.1 Doodle

O Doodle é uma ferramenta de calendário online utilizada para coordenar horários de reuniões [13].

A utilização desta ferramenta é muito intuitiva. Um utilizador que queira organizar uma reunião com vários participantes começa por criar uma página Doodle onde define os horários que tem disponíveis para a sua realização. De seguida, partilha-a com todos os participantes para que estes insiram as suas disponibilidades. Quando todos os intervenientes tiverem respondido, o organizador analisa as respostas e com base nestas agenda a reunião.

Para se encontrar a solução da distribuição dos horários das apresentações dos documentos preparativos da dissertação de mestrado, utilizando o Doodle, seria necessário um grande esforço por parte da pessoa organizadora. O Doodle ajuda a agrupar as indisponibilidades de todos os intervenientes. No entanto, a parte de conjugar as disponibilidades de cada membro, com a possibilidade de um professor ser membro de diferentes júris e com as disponibilidades das salas, iria acrescentar um nível elevado de complexidade na organização.

A utilização desta ferramenta revela-se uma boa solução na criação de reuniões, porém não cumpre o objetivo de apresentar uma solução final sem intervenção exaustiva do organizador.

3.1.2 Microsoft Excel

O Microsoft Excel é um programa de software que se baseia em edição de folhas de cálculo, podendo ser considerado como uma ferramenta avançada de análise e visualização de dados [16].

O Microsoft Excel tem a vantagem de ser uma ferramenta com inúmeras funcionalidades, onde é possível fazer-se quase tudo, inclusive a criação e gestão de horários. No entanto, para a criação de horários destinados às apresentações dos documentos preparativos da dissertação de mestrado, não se afirma como uma solução adequada. A justificação para esta incapacidade deve-se ao facto do utilizador necessitar de dispor conhecimentos aprofundados das funcionalidades do Excel, não estando a ferramenta ao alcance de um utilizador comum. A exemplo do acabado de referir, dir-se-á que o organizador teria de criar tabelas com as disponibilidades de cada membro do júri, do local onde vão ser realizadas as provas e incrementar as restrições necessárias de modo a evitar sobreposições entre apresentações.

3.1.3 Aplicações

Existem múltiplas aplicações para a criação e gestão de horários.

A maioria destas aplicações destinam-se à gestão de horários individuais dos próprios utilizadores. No entanto, também existem aplicações, maioritariamente utilizadas por escolas e universidades, que geram horários a partir de restrições recebidas.

Estas aplicações têm vantagens, uma delas é o facto da maioria suportar a escrita das restrições de forma incremental, e outra é o facto de já terem um parâmetro destinado ao número das salas.

Existem também desvantagens nestas aplicações. Uma desvantagem é o facto das restrições terem de ser inseridas por uma única pessoa, traduzindo-se na centralização do trabalho que é manifestamente demorado e suscetível de erros. Adicionalmente são aplicações, por norma, pagas e que não contemplam as especificidades de gestão das várias necessidades e indisponibilidades inerentes ao processo de agendamento das apresentações. Assim sendo, a utilização de uma aplicação de gestão de horários escolares, como por exemplo a aplicação ascHorários[1], não se revela como sendo capaz de produzir os resultados desejados.

[1] https://www.asctimetables.com/

3.2 Interação entre ASP e diversas linguagens de programação

A utilização do ASP e do clingo é uma possibilidade para a resolução das incompatibilidades de horários. De forma a facilitar a sua utilização, torna-se necessário analisar diferentes possibilidades, nomeadamente ao nível de interação com uma linguagem de programação.

Nesta parte do capítulo vamos apresentar várias hipóteses para fazer esta interação e, consoante as suas vantagens e desvantagens, escolher a melhor para integrar a aplicação que pretendemos desenvolver.

3.2.1 Clingo4j

O Clingo4j é uma API da linguagem de programação Java, que auxilia os programadores a interagir com o clingo [42].

As suas principais funcionalidades são:

- Simplicidade no acesso ao clingo dentro de um programa Java;
- Leitura e escrita de programas em ASP;
- Possibilidade de acrescentar ao clingo componentes personalizadas em C++.

Embora esta API apresente diversas vantagens, existe um ponto fundamental que torna impossível a sua integração na aplicação a desenvolver. O Clingo4j não beneficia de atualizações desde Fevereiro de 2018.

3.2.2 Clingo-rs

O Clingo-rs é um software open-source utilizado para fazer a ligação entre a linguagem de programação Rust e o clingo [49].

Considerando que não desenvolvemos experiência com a linguagem Rust, optámos por dispensar a incorporação deste software na aplicação.

3.2.3 PyASP/Clyngor

O PyASP é uma biblioteca da linguagem de programação Python, permitindo a integração de programas ASP em Pyton [50, 51].

Esta biblioteca foi abandonada e reescrita, usando agora o nome Clyngor. Contudo, os programas escritos com a biblioteca PyASP conseguem ser facilmente convertidos para Clyngor.

As principais vantagens do Clyngor são:

- Existência de várias interfaces;
- Interfaces intuitivas;
- Suporte para uma otimização incremental;
- Possibilidade do utilizador instalar um módulo de Python/clingo [40].

Por defeito, é utilizado um compilador muito simples. No entanto, caso seja necessário ou solicitado pelo utilizador, é possível a utilização um compilador mais robusto [11].

O Clyngor é utilizado na bioinformática, na matemática, em aplicações web, entre outros [10].

3.2.4 Clorm

O Clorm é uma biblioteca do Python que disponibiliza uma interface Object-relational mapping - Mapeamento Objeto-Relacional (ORM) para o solver clingo, tendo como objetivo a facilitação da integração do clingo na escrita de aplicações em Python [8, 7].

As suas principais vantagens são:

- Intuição na relação do clingo com o Python;
- Implementação sobre a API oficial do clingo;
- Simplicidade na perceção da sintaxe;
- Facilidade na conversão de dados entre Python e clingo;
- Aceitação de pesquisas em bases de dados.

A documentação do Clorm é extensa, no entanto, é bastante acessível e de fácil compreensão [9].

3.3 Conclusão

Neste capítulo é possível perceber que as ofertas existentes no mercado para a criação de horários não servem para resolver o problema das apresentações dos documentos preparativos da dissertação de mestrado. Uma vez que não consideram, como deveriam, a conjugação dos seguintes fatores:

- Prazo das apresentações;
- Salas disponíveis;

- Horário de utilização de cada sala;
- Composição do júri;
- Disponibilidade de cada membro do júri;
- Possibilidade de um professor ser membro de diferentes júris;
- Duração de cada apresentação.

Tendo sido referido que a execução de programas em ASP não é facil para um utilizador sem conhecimentos especializados, apresentamos como solução a ultização de uma ferramenta que permita a interação entre uma linguagem de programação e o ASP.

O Clyngor e o Clorm são as opções que se apresentam como viáveis, face aos constrangimentos apresentados pelo Clingo4j (3.2.1) e pelo Clingo-rs (3.2.2). Adicionalmente, considera-se como decisiva e como uma mais valia inquestionável, a funcionalidade do Clorm em permitir pesquisas nas bases de dados.

4

Aplicação

A presente aplicação foi desenvolvida com o intuito de solucionar o problema complexo de agendamento das apresentações dos documentos de preparação das dissertações de mestrado. Este agendamento envolve a coordenação das composições dos júris, restrições horárias e recursos físicos, como as salas. O processo, além de ser difícil e demorado quando realizado manualmente, requer a consideração de múltiplas restrições, tais como os prazos das apresentações, disponibilidades individuais, composição do júri, sobreposição de horários e gestão de espaços.

O desenvolvimento desta aplicação proporciona uma significativa facilidade na recolha da informação necessária para este processo, podendo cada utilizador indicar as suas restrições nesta plataforma.

Para garantir que as necessidades de todos os utilizadores e as restrições são satisfeitas, a aplicação deve ser capaz de escalar um horário final que acomode todas as apresentações, assegurando a disponibilidade das salas e a não sobreposição de horários dos membros do júri.

Os utilizadores alvo desta aplicação incluem todos os indivíduos envolvidos no processo de agendamento das apresentações mencionadas anteriormente, sejam eles participantes ou organizadores.

A aplicação visa oferecer uma interface intuitiva e de fácil navegação, facilitanto a interação dos utilizadores com as funcionalidades disponíveis. Esta interface permite os utilizadores inserir as restrições de forma simples, assegurando que todas as necessidades são consideradas no processo de agendamento.

Neste capítulo, serão apresentados os detalhes sobre a estrutura e arquitetura da aplicação, que está dividida em quatro componentes principais: base de dados, servidor, cliente e microsserviço. Serão também discutidos os tipos de utilizadores e os seus diferentes

níveis de acesso, bem como os *user stories* que ilustram as funcionalidades disponíveis para cada grupo de utilizadores.

4.1 Arquitetura

A arquitetura da nossa aplicação segue o padrão de arquitetura em camadas e é composta por 4 componentes principais: um *Cliente*, um *Servidor*, um *Microsserviço* e uma *Base de dados*. O diagrama de componentes, presente na figura 4.1, ilustra a interação entre estes diferentes componentes.

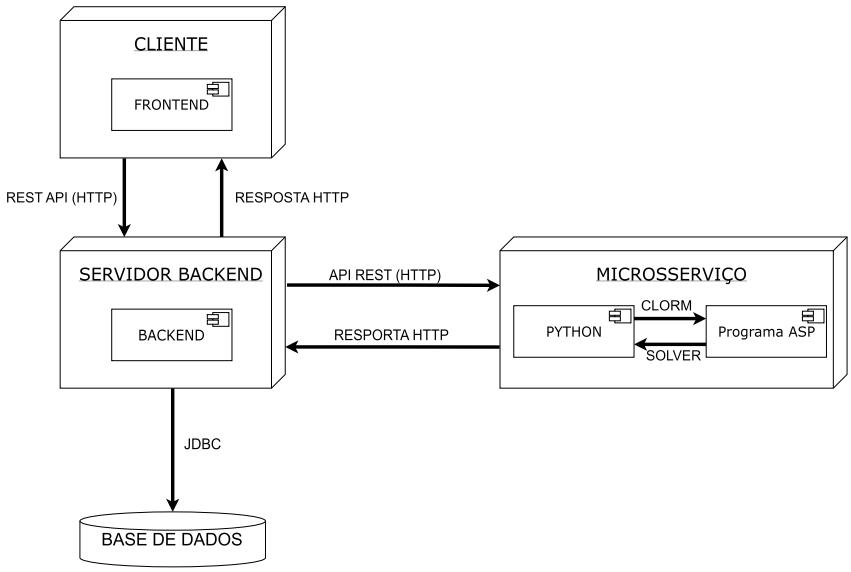

Figura 4.1: Diagrama de Componentes da aplicação

Cada componente será apresentado e detalhado nos próximos capítulos, de forma a facilitar a compreensão do funcionamento da aplicação.

A componente *Base de dados*, apresentada em mais detalhe no capítulo 5, é responsável por armazenar e gerir todos os dados da aplicação, garantindo a sua persistência e segurança. A base de dados escolhida foi a H2, uma solução compatível com o componente *Servidor*, leve e de fácil integração.

O componente *Servidor*, apresentado em mais detalhe no capítulo 6, atua como um meio intermediário entre a interface web, o microsserviço e a base de dados da aplicação. Este componente, desenvolvido em Kotlin e utilizando o Spring Boot, implementa uma API REST para controlar a comunicação entre os restantes componentes da aplicação.

O componente *Microsserviço*, apresentado em mais detalhe no capítulo 7, é composto por uma API REST desenvolvida em python e por um Programa ASP. A sua função é

gerar um horário final de forma independente, utilizando as informações recolhidas pela aplicação. O resultado é enviado para o *Servidor*, onde é realizada a atualização da base de dados.

O componente *Cliente*, apresentado em mais detalhe no capítulo 8, proporciona a interface web para a interação dos utilizadores com a aplicação. Esta interface web foi desenvolvida utilizando o React.

4.2 Tipos de Utilizadores

De forma a fortalecer a segurança da nossa aplicação, um utilizador registado tem uma *role* associada. A utilização de uma *role* permite categorizar os utilizadores de acordo com as suas permissões e restringir os seus acessos. Desta forma conseguimos assegurar que as operações são apenas executadas pelos utilizadores devidos.

É também importante referir que a operação de mudança de role tem verificações rigorosas para que não seja possível a alteração indevida de privilégios.

A *role* atribuída aos utilizadores pode ter os seguintes valores:

- *USER* — Utilizadores padrão;
- *ORGANIZER* — Utilizadores com responsabilidades ao nível de organização;
- *ADMIN* — Administradores da plataforma.

4.3 *User Stories*

Neste ponto descrevemos as operações de cada tipo de utilizador, consoante a sua *role* na aplicação. Um utilizador com uma *role* de maior privilégio herda as funcionalidades das *roles* inferiores.

4.3.1 Utilizador Padrão (*USER*)

Um utilizador padrão pode realizar as seguintes operações na aplicação:

- Enquanto *USER* eu quero poder fazer login na aplicação com as minhas credenciais (email e password);
- Enquanto *USER* eu quero poder consultar as informações de todas as apresentações;
- Enquanto *USER* eu quero poder consultar a informação relativa à minha conta de utilizador;
- Enquanto *USER* eu quero poder mudar a minha password;

- Enquanto *USER* eu quero poder apagar a minha conta na aplicação;
- Enquanto *USER* eu quero poder consultar as informações de todas as minhas apresentações;
- Enquanto *USER* eu quero poder consultar as informações as apresentações das quais sou orientador;
- Enquanto *USER* eu quero poder consultar as informações as apresentações das quais sou vogal;
- Enquanto *USER* eu quero poder consultar as informações as apresentações das quais sou participante extra;
- Enquanto *USER* eu quero poder adicionar um participante extra às apresentações nas quais sou o orientador ou vogal;
- Enquanto *USER* eu quero poder indicar as minhas indisponibilidades nos slots temporais que existem na aplicação;
- Enquanto *USER* eu quero poder indicar restrições temporais nas apresentações que sou orientador;
- Enquanto *USER* eu quero poder agrupar as minhas apresentações para sejam marcadas consecutivamente;
- Enquanto *USER* eu quero poder fazer logout da aplicação.

4.3.2 Utilizador Organizador (*ORGANIZER*)

Um utilizador com responsabilidades ao nível da organização, além de ter acesso a todas as operações de um utilizador padrão, pode ainda realizar as seguintes operações:

- Enquanto *ORGANIZER* eu quero poder fazer download de uma tabela, em pdf ou em csv, que contenha as informações de todas as apresentações;
- Enquanto *ORGANIZER* eu quero poder adicionar e remover salas na aplicação;
- Enquanto *ORGANIZER* eu quero poder adicionar e remover slots na aplicação;
- Enquanto *ORGANIZER* eu quero poder fazer upload de um ficheiro Excel com as informações relativas às apresentações e à composição dos júris;
- Enquanto *ORGANIZER* eu quero poder adicionar e remover apresentações individualmente;
- Enquanto *ORGANIZER* eu quero poder adicionar um participante extra às apresentações;

- Enquanto *ORGANIZER* eu quero poder consultar as apresentações de um determinado utilizador;
- Enquanto *ORGANIZER* eu quero poder consultar as indisponibilidades de cada utilizador;
- Enquanto *ORGANIZER* eu quero poder adicionar um novo utilizador na aplicação;
- Enquanto *ORGANIZER* eu quero poder consultar a informação de todos os utilizadores presentes na aplicação;
- Enquanto *ORGANIZER* eu quero poder remover um utilizador com a *role USER* da aplicação;
- Enquanto *ORGANIZER* eu quero poder iniciar a computação dos slots e das salas sobre as apresentações.

4.3.3 Utilizador Administrador (*ADMIN*)

Um utilizador administrador da plataforma, além de ter acesso a todas as operações de um utilizador padrão e de um utilizador organizador, pode ainda realizar as seguintes operações:

- Enquanto *ADMIN* eu quero poder remover um utilizador com a *role ORGANIZER* da aplicação;
- Enquanto *ADMIN* eu quero poder alterar as roles dos utilizadores *USER* e *ORGANIZER*;

5

Base de dados

Neste capítulo apresentamos a base de dados que utilizamos para armazenar e gerir as informações da aplicação. Começamos por indicar o uso da base de dados H2 e explicar a sua escolha, destacando as suas vantagens. De seguida, abordamos a sua estrutura, detalhando cada uma das tabelas, incluindo as suas propriedades e a importância de cada uma para o funcionamento da aplicação.

5.1 H2

H2 é uma base de dados relacional open-source escrita em Java que pode funcionar tanto em memória como em disco. É uma base de dados totalmente compatível com SQL e pode ser facilmente integrada no Spring Boot.

Esta base de dados pode ser utilizada em modo embutido (*embedded*), dentro da própria aplicação, ou em modo servidor, o que a torna bastante flexível nos diferentes cenários de uso [27].

A escolha da base de dados H2 para o desenvolvimento da nossa aplicação foi motivada por vários fatores:

- **Facilidade de configuração:** Não requer configurações externas, acelerando o processo de desenvolvimento;
- **Execução em memória ou em disco:** Oferece a flexibilidade de ser utilizada tanto em memória, durante o desenvolvimento, como em disco, enquanto a aplicação está em execução, permitindo a persistência dos dados;
- **Facilidade de integração com Spring Boot:** A integração com o Spring Boot é simples e rápida, acelerando o processo de desenvolvimento;
- **Experiência prévia:** A existência de experiência prévia da nossa parte também se revelou como um fator importante na escolha.

5.2 Estrutura da Base de Dados

A Figura 5.1 ilustra o diagrama de entidades e relações da base de dados da nossa aplicação. Este diagrama apresenta as entidades utilizadas na aplicação e as suas relações.

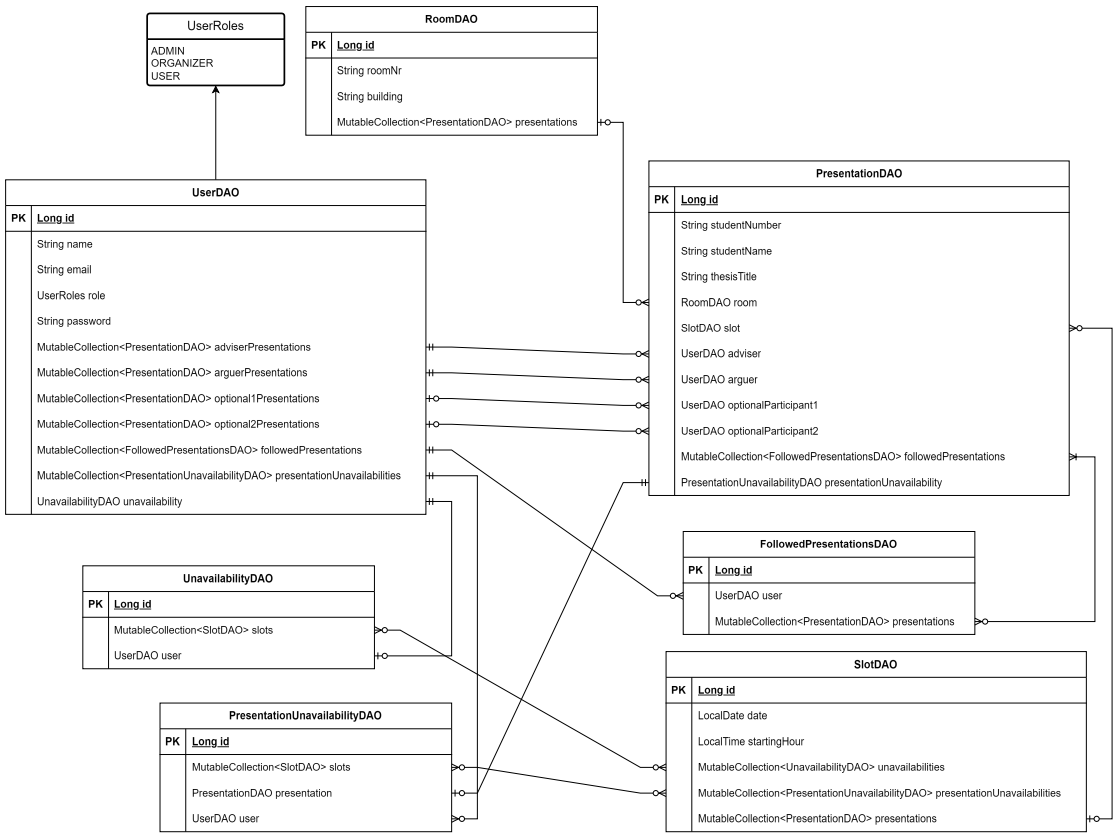

Figura 5.1: Diagrama de Entidades e Relações da Base de Dados

É importante informar que as tabelas na base de dados têm nomes diferentes dos presentes no diagrama. Esta diferença deve-se à convenção das nomenclaturas. As tabelas na base de dados são frequentemente nomeadas no plural para refletir o armazenamento de coleções de registos, enquanto as classes na aplicação utilizam o sufixo *Data Access Object - Objeto de Acesso de Dados (DAO)* para denotar a sua responsabilidade na gestão e acesso aos dados.

De seguida apresentamos as tabelas da base de dados e os seus atributos, explicando a sua necessiade e utilidade na aplicação.

5.2.1 Tabela *users*

A tabela *users* armazena as informações de todos os utilizadores registados na aplicação.

Os atributos de cada utilizador são:

- *id* — Identificador único de cada utilizador na tabela. Este valor é gerado automaticamente pelo sistema ao criar um novo registo, garantindo a sua unicidade;
- *name* — Nome do utilizador;
- *email* — Email do utilizador;
- *role* — Role de um utilizador na aplicação. Este campo pode assumir os valores *ADMIN*, *ORGANIZER* ou *USER*, definidos num *enum UserRoles*;
- *password* — Password encriptada do utilizador;
- *adviserPresentations* — Coleção de apresentações nas quais o utilizador é o orientador;
- *arguerPresentations* — Coleção de apresentações nas quais o utilizador é o vogal;
- *optional1Presentations* — Coleção de apresentações nas quais o utilizador é um participante opcional escolhido pelo orientador ou pelo vogal de cada apresentação;
- *optional2Presentations* — Coleção de apresentações nas quais o utilizador é um participante opcional escolhido por um organizador ou por um administrador da aplicação;
- *followedPresentations* — Coleção dos conjuntos de apresentações que o utilizador necessita que sejam agendadas de forma consecutiva;
- *presentationUnavailabilities* — Coleção de indisponibilidades das apresentações do utilizador;
- *unavailability* — Objeto do tipo *UnavailabilityDAO* que armazena a informação sobre as indisponibilidades do utilizador.

5.2.2 Tabela *presentations*

A tabela *presentations* armazena as informações das apresentações das dissertações criadas na aplicação. Inicialmente as apresentações são criadas sem horário e sem sala e esses atributos são apenas preenchidos depois da computação de todas as apresentações no microsserviço python.

Os atributos de cada apresentação são:

- *id* — Identificador único de cada apresentação na tabela. Este valor é gerado automaticamente pelo sistema ao criar um novo registo, garantindo a sua unicidade;
- *studentNumber* — Número do estudante que irá apresentar a sua dissertação;

- *studentName* — Nome do estudante que irá apresentar a sua dissertação;
- *thesisTitle* — Título da dissertação a ser apresentada;
- *room* — Sala em qual a apresentação irá decorrer;
- *slot* — Horário e dia no qual a apresentação irá decorrer;
- *adviser* — Orientador da dissertação a ser apresentada;
- *arguer* — Vogal da dissertação a ser apresentada;
- *optionalParticipant1* — Participante opcional da apresentação, escolhido pelo orientador ou pelo vogal da apresentação;
- *optionalParticipant2* — Participante opcional da apresentação, escolhido por um organizador ou por um administrador da aplicação;
- *followedPresentations* — Coleção dos conjuntos de apresentações seguidas desta apresentação;
- *presentationUnavailability* — Coleção das indisponibilidades horárias desta apresentação.

5.2.3 Tabela *rooms*

A tabela *rooms* armazena as informações das salas disponíveis para acomodar as apresentações.

Os atributos de cada sala são:

- *id* — Identificador único de cada sala na tabela. Este valor é gerado automaticamente pelo sistema ao criar um novo registo, garantindo a sua unicidade;
- *roomNr* — Número da sala;
- *building* — Edifício onde a sala está localizada;
- *presentations* — Coleção das apresentações a decorrer na sala.

5.2.4 Tabela *slots*

A tabela *slots* armazena as informações dos slots de tempo disponíveis para acomodar as apresentações.

Os atributos de cada slot são:

- *id* — Identificador único de cada slot na tabela. Este valor é gerado automaticamente pelo sistema ao criar um novo registo, garantindo a sua unicidade;
- *date* — Dia no qual o slot ocorre;
- *startingHour* — Horário de início do slot;
- *unavailabilities* — Coleção das indisponibilidades de utilizadores sobre o slot;
- *presentationUnavailabilities* — Coleção das indisponibilidades de apresentações sobre o slot;
- *presentations* — Coleção das apresentações a decorrer no slot.

5.2.5 Tabela *unavailabilities*

A tabela *unavailabilities* armazena os slots de tempo em que um utilizador não tem disponibilidade.

Os atributos de cada registo de indisponibilidades de um utilizador são:

- *id* — Identificador único das indisponibilidades de um utilizador na tabela. Este valor é gerado automaticamente pelo sistema ao criar um novo registo, garantindo a sua unicidade;
- *slots* — Coleção com os slots de tempo nos quais o utilizador não tem disponibilidade para participar;
- *user* — Utilizador "dono"das indisponibilidades.

5.2.6 Tabela *presentationUnavailabilities*

A tabela *presentationUnavailabilities* armazena os slots de tempo nos quais uma apresentação não pode decorrer. Esta tabela é maioritariamente útil para indicar as indisponibilidades de um elemento externo à faculdade, uma vez que as dos próprios utilizadores são guardadas na tabela *unavailabilities*.

No caso de uma dissertação ser desenvolvida em conjunto com uma empresa, será necessário considerar as indisponibilidades do orientador externo à faculdade. Nesta situação, em vez de ser criada uma conta de utilizador na aplicação, o orientador docente da faculdade indica as indisponibilidades do orientador externo como indisponibilidades da apresentação na qual este participa.

Os atributos de cada registo de indisponibilidades de uma apresentação são:

- *id* — Identificador único das indisponibilidades de uma apresentação na tabela. Este valor é gerado automaticamente pelo sistema ao criar um novo registo, garantindo a sua unicidade;
- *slots* — Coleção com os slots de tempo nos quais a apresentação não poderá ser marcada;
- *presentation* — Apresentação "dona"das indisponibilidades;
- *user* — Utilizador que criou o registo de indisponibilidades;

5.2.7 Tabela *followedPresentations*

Os atributos de cada grupo de apresentações seguidas são:

- *id* — Identificador único de cada grupo de apresentações seguidas na tabela. Este valor é gerado automaticamente pelo sistema ao criar um novo registo, garantindo a sua unicidade;
- *user* — Utilizador que realizou a operação de agregar as suas apresentações;
- *presentations* — Coleção de apresentações com necessidade de ser marcadas consecutivamente.

5.3 Conclusão

Neste capítulo, foi apresentada a estrutura da base de dados utilizada pela aplicação, com uma explicação detalhada sobre cada tabela e o papel que desempenha no armazenamento e gestão dos dados.

Com a base de dados devidamente estruturada e funcional, o próximo capítulo abordará a componente do Servidor (Backend), responsável por fazer a ponte entre o Cliente, a base de dados e o microsserviço responsável pela computação de horários. No Servidor, exploraremos as tecnologias utilizadas, a arquitetura adotada, bem como as operações sobre a base de dados são integradas e tratadas para suportar as funcionalidades da aplicação de forma segura e eficiente.

6

Servidor

O Servidor, que representa o Backend da nossa aplicação, consiste numa API REST desenvolvida com o propósito de atuar como um meio intermediário entre o Frontend e a base de dados. Outra das funções deste Backend é estabelecer uma ponte entre o microsserviço e a base de dados, garantindo uma comunicação consistente entre estes elementos. Consequentemente, o Backend centraliza a gestão das operações e coordena as interações entre os diversos componentes da aplicação.

A comunicação com o Frontend é alcançada através de endpoints RESTful que permitem a troca de dados entre o cliente e o servidor. A conecção à base de dados é conseguida utilizando o JDBC, garantindo assim a eficiência da comunicação com a mesma. Por outro lado, a comunicação com o microsserviço é feita através de um endpoint POST exposto pela sua API REST.

Nos pontos seguintes, serão discutidas as tecnologias utilizadas para o desenvolvimento, a arquitetura, as funcionalidades e operações que o Backend suporta, a integração do microsserviço e, por fim, como é garantida e implementada a segurança da aplicação.

O código referente a esta API está presente no GitHub através do link: `https://github.com/SalvadorSi/Aplicacao-Dissertacao-Salvador-Silva`.

A documentação desta API está presente no link: `https://documenter.getpostman.com/view/15105260/2sAXqzYeGs`.

6.1 Tecnologias

Neste ponto iremos apresentar as tecnologias utilizadas no desenvolvimento do Backend da nossa aplicação, escrevendo como cada uma contribuiu para o bom funcionamento e eficiência da aplicação.

Na escolha das tecnologias a utilizar, foi crucial que os seguintes requisitos fossem cumpridos:

- Escalabilidade — Capacidade de lidar com um número crescente de utilizadores e dados;
- Segurança — Implementação de mecanismos para proteger dados sensíveis;
- Facilidade de Integração — Compatibilidade e facilidade de comunicação com os outros componentes da aplicação;
- Manutenção e Extensibilidade — Facilidade na manutenção e adição de novas funcionalidades;
- Desempenho — Capacidade de responder a pedidos de forma eficiente minimizando latências.

É importante mencionar que, embora existem diversas tecnologias que cumprem estes requisitos, a escolha final foi motivada pela nossa experiência prévia com as ferramentas escolhidas.

6.1.1 Spring Boot

O Spring Boot é uma framework open-source que simplifica o processo de configuração e o desenvolvimento de microsserviços e aplicações Web. Um dos seus principais objetivos é minimizar a quantidade de configurações necessárias para a criação de um projeto, fornecendo um conjunto de ferramentas e dependências predefinidas. O Spring Boot oferece também a possibilidade de integrar novas dependências e tecnologias, como bases de dados e mecanismos de segurança [46, 52, 53].

No contexto da nossa aplicação, o Spring Boot revelou-se uma excelente escolha para o desenvolvimento do Backend. Esta framework permitiu a criação rápida e simples do projeto e também facilitou a incorporação das dependências extra necessárias durante o seu desenvolvimento. Uma das principais vantagens foi a facilidade de integração e comunicação com a base de dados, permitindo a implementação de repositórios de forma simples, ágil e eficiente.

6.1.2 Kotlin

Desenvolvida pela JetBrains, Kotlin é uma linguagem de programação multiplataforma, orientada a objetos e que compila para a Java Virtual Machine - Máquina Virtual Java (JVM). Esta linguagem de programação oferece várias melhorias relativamente ao Java tradicional, como seja uma sintaxe mais concisa, reduzindo a quantidade de código necessária para obter os mesmos objetivos, e ainda a redução de erros comuns, como o *NullPointerException*.

As melhorias do Kotlin levam a um aumento na produtividade dos programadores [32, 31].

Optámos por utilizar o Kotlin para o desenvolvimento do nosso Backend, devido às suas características que favorecem a produtividade, a legibilidade e a manutenção do código. A sua sintaxe simplificada e a redução do código contribuíram para um desenvolvimento mais rapido e menos propenso a erros. A interoperabilidade com o Java facilitou a integração com bibliotecas, tanto as já existentes no Spring, como as adicionais necessárias, durante o desenvolvimento.

6.1.3 Apache Maven

Hospedado pela Apache Software Foundation, Maven é uma ferramenta de automação de compilação. Utilizando um arquivo Project Object Model - Projeto Modelo de Objeto (POM), do tipo XML, é descrito o projeto a ser construído, definindo as dependências de módulos e componentes externos, bem como a ordem de compilação e os plug-ins necessários. Uma das principais vantagens do Maven é a sua capacidade de gerir automaticamente as dependências, evitando conflitos de versões e garantindo que o projeto utiliza as bibliotecas mais atualizadas e compatíveis [2].

A utilização desta ferramenta no desenvolvimento do nosso Backend foi incentivada principalmente pela gestão eficiente das dependências e pela automação do processo de compilação. A compatibilidade com o Spring Boot e com o Kotlin também contribuiu significativamente para a escolha da mesma.

6.1.4 Postman

O Postman é uma ferramenta utilizada no desenvolvimento, debug, teste e documentação de APIs. Permite aos programadores criar coleções de pedidos HTTP de forma intuitiva, facilitando a organização dos testes das APIs. Adicionalmente é possível verificar os tempos de resposta da API [43].

No âmbito do Backend da nossa aplicação, a ferramenta Postman demonstrou ser a escolha certa para testar os endpoints RESTful. As suas capacidades de executar os testes de forma simples e rápida, de validar as respostas e os códigos de erro, ajudou a garantir que os pedidos HTTP estavam a funcionar conforme o esperado.

6.1.5 Dependências

No desenvolvimento desta API utilizámos várias dependências que desempenham um papel fundamental na eficiência da aplicação. As dependências fornecem serviços e recursos que auxiliam na simplificação de diversos aspetos durante o desenvolvimento e ajudam à criação de um Backend robusto e escalável.

De seguida iremos explicar as principais dependências utilizadas neste projeto e o seu papel na arquitetura global:

- **Spring Boot Starter Data JPA**[1] — Fornece uma camada de abstração sobre o JPA, facilitando as operações na base de dados e garantindo a eficiência nas operações CRUD.
- **Spring Boot DevTools**[2] — Melhora o processo de desenvolvimento com reinicializações automáticas e recarregamentos ao vivo, agilizando o debug durante o desenvolvimento.
- **JSON Web Token (JWT)**[3,4,5] — Fornece uma autenticação segura e sem estado, gerando e verificando tokens JWT para gerir sessões dos utilizadores.
- **H2 Database**[6] — Base de dados relacional leve e embutida, projetada para utilização em aplicações Java. Oferece uma interface SQL padrão e uma fácil integração com o Spring.
- **Apache POI**[7] — Biblioteca usada para leitura e escrita de arquivos do Excel.
- **Fuel**[8] — Cliente HTTP usado para fazer chamadas a APIs REST, garantindo uma integração descomplicada de APIs externas.
- **Gson**[9] — Biblioteca utilizada para serializar e desserializar dados JavaScript Object Notation - Notação de Objetos JavaScript (JSON), garantindo uma troca de dados consistente entre as diferentes camadas da aplicação.
- **Spring Boot Starter Mail**[10] — Simplifica o envio de notificações por e-mail automatizadas.

6.2 Arquitetura

A arquitetura do Backend foi desenhada de forma modular, facilitando o desenvolvimento, a manutenção e a escalabilidade da aplicação. Esta divisão está contemplada no diagrama de pacotes presente na Figura 6.1.

[1] https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-jpa
[2] https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-devtools
[3] https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-api
[4] https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-impl
[5] https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-jackson
[6] https://mvnrepository.com/artifact/com.h2database/h2
[7] https://mvnrepository.com/artifact/org.apache.poi/poi
[8] https://mvnrepository.com/artifact/com.github.kittinunf.fuel/fuel
[9] https://mvnrepository.com/artifact/com.google.code.gson/gson
[10] https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-mail

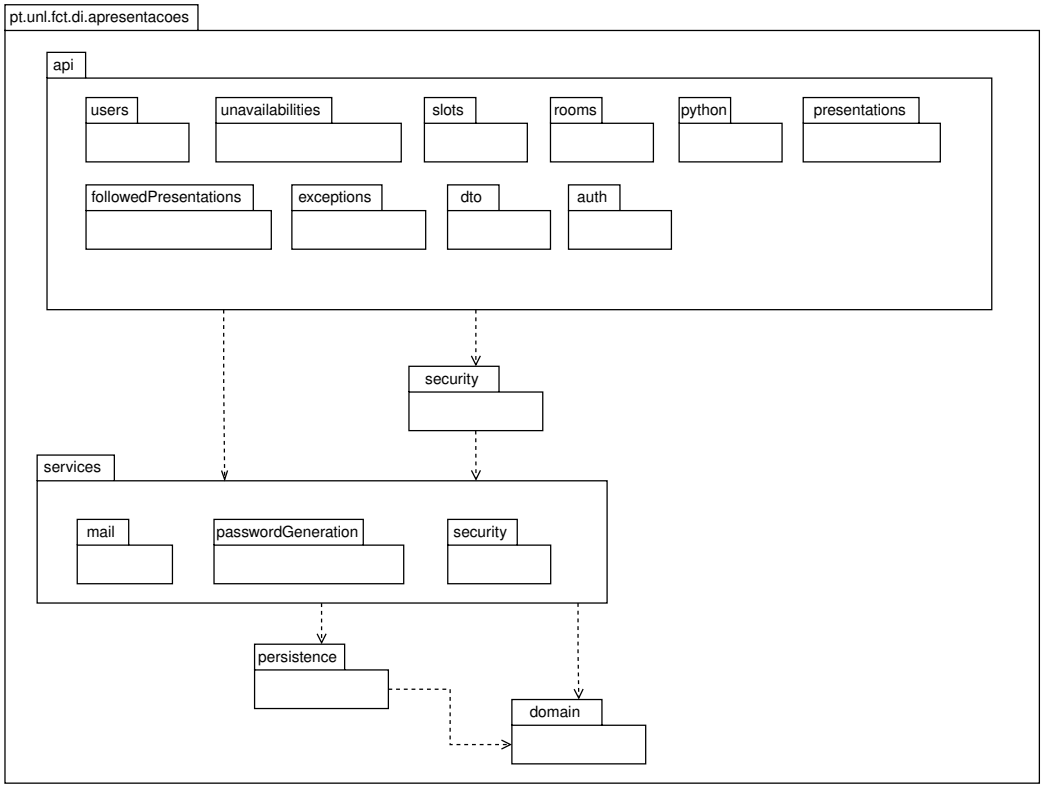

Figura 6.1: Diagrama de Pacotes do Backend

O módulo *API* é responsável por expor os endpoints RESTful, que interagem com o Frontend, e por passar os pedidos do cliente para os serviços apropriados. As vantagens indicadas na divisão do Backend em módulos também se aplicam neste caso, sendo o mesmo subdividido nos seguintes submódulos:

- **Auth** — Responsável pela gestão do acesso dos utilizadores;
- **Data Tranfer Object - Objeto de Transferência de Dados (DTO)** — Contém os DTOs utilizados na transferência de dados entre os restantes componentes da aplicação;
- **Exceptions** — Define as exceções lançadas pela aplicação e os códigos HTTP apropriados;
- **FollowedPresentations** — Responsável pela gestão das apresentações que necessitam ser marcadas consecutivamente;
- **Presentation** — Responsável pela gestão das apresentações;
- **Python** — Responsável pela comunicação com o microsserviço;
- **Rooms** — Responsável pela gestão das salas;
- **Slots** — Responsável pela gestão dos slots;

- **Unavailabilities** — Responsável pela gestão das indisponibilidades dos utilizadores e das próprias apresentações;

- **Users** — Responsável pela gestão dos utilizadores.

Com exceção dos submódulos DTO e Exceptions, os restantes são compostos por ficheiros de anotações, um ficheiro API implementado como uma interface e um ficheiro Controller. Os ficheiros de anotações são personalizados e permitem controlar as permissões e os acessos a cada uma das operações neste mesmo submódulo. O ficheiro API contém as assinaturas dos metodos, discrimina os vários endpoints, especificando os tipos de pedidos HTTP das diversas operações, e utiliza as anotações personalizadas criadas. O ficheiro Controller delega a execução das operações para os serviços apropriados e trata das exceções ocorridas.

O módulo *Services* engloba os serviços imprescindíveis para o bom funcionamento da aplicação. Estes serviços atuam como intermediários entre os controladores e os repositórios da aplicação, empregando as validações e implementando a lógica das operações delegadas pelos controladores. É ainda importante realçar que este módulo também está subdividido em submódulos:

- **Mail** — Utilizado para envio de emails personalizados aos utilizadores da aplicação;

- **PasswordGeneration** — Utilizado para gerar as passwords temporárias de acesso à aplicação;

- **Security** — Utilizado para garantir a segurança das operações na aplicação.

O módulo *Persistence* contém os repositórios das entidades presentes na aplicação, encarregues de realizar a comunicação com a base de dados. Cada repositório herda a interface *CrudRepository*, responsável por fornecer métodos padrão para as operações CRUD. Além destes métodos, cada repositório tem definidas consultas personalizadas para atender as necessidades específicas da aplicação.

O módulo *Domain* é composto por um ficheiro que centraliza as entidades que definem a estrutura da base de dados. As entidades estão representadas pelos DAOs que mapeiam as tabelas da base de dados para classes Kotlin.

O módulo *Security* é responsável pela configuração da segurança da aplicação. De forma a tornar a aplicação segura criámos diversas configurações e filtros para garantir que apenas os utilizadores autenticados e com as permissões adequadas têm acesso às funcionalidades desenvolvidas.

É importante assinalar que embora a interação entre *API* e *Security* não seja direta, as configurações e os filtros definidos em *Security* afetam o comportamento dos endpoints definidos em *API*. Estes filtros intercetam os pedidos e aplicam regras de segurança, como a verificação de autenticações e permissões, antes que o pedido seja processado pelos controladores do módulo *API*.

6.3 Funcionalidades e Operações

Este ponto apresenta uma visão geral das principais funcionalidades e operações implementadas no Backend da nossa aplicação.

Cada operação está associada a um controlador que, tal como referido no ponto anterior, é responsável por delegar a execução da operação para o respetivo serviço e tratar das exceções ocorridas durante essa execução. Cada controlador está vinculado a um único serviço, com exceção do controlador das indisponibilidades (*UnavailabilityController*), que interage com dois serviços: o *UnavailabilityService*, responsável pelas indisponibilidades dos utilizadores, e o *PresentationUnavailabilityService*, que lida com as indisponibilidades associadas às apresentações.

Para cada uma das entidades principais, iremos apresentar o diagrama de arquitetura correspondente, ilustrando as suas operações e interações. Esta visão proporciona uma compreensão mais clara da organização do Backend e das relações entre os diferentes componentes.

Para evitar que este ponto se torne demasiado extenso e complexo, as descrições detalhadas das operações foram transferidas para os apêndices A, B, C, D, E, F e G. Esta abordagem permite uma apresentação mais organizada e clara das funcionalidades implementadas.

De modo a assegurar a integridade e segurança dos dados, todas as operações passam por um processo de validação antes de acederem aos repositórios e serem efetivamente executadas. Estas validações asseguram que apenas utilizadores autenticados e com permissões adequadas conseguem realizar as ações permitidas.

Os possíveis erros e os respetivos códigos que podem ser retornados em cada operação estão igualmente documentados. Desta forma oferecemos uma visão completa sobre como o sistema responde a diferentes cenários, desde erros de autenticação, até conflitos de dados ou mesmo falhas inesperadas.

6.3.1 Utilizadores

A gestão dos utilizadores é uma funcionalidade central na nossa aplicação, permitindo a criação, atualização, consulta e remoção de informações relativas aos perfis dos utilizadores. Estas funcionalidades englobam diversas operações que podem ser realizadas por diferentes tipos de utilizadores. Deste modo foram implementadas várias validações para garantir que apenas utilizadores credenciados podem executar determinadas ações.

As interações entre componentes e serviços que suportam as operações referidas, estão representadas na figura 6.2.

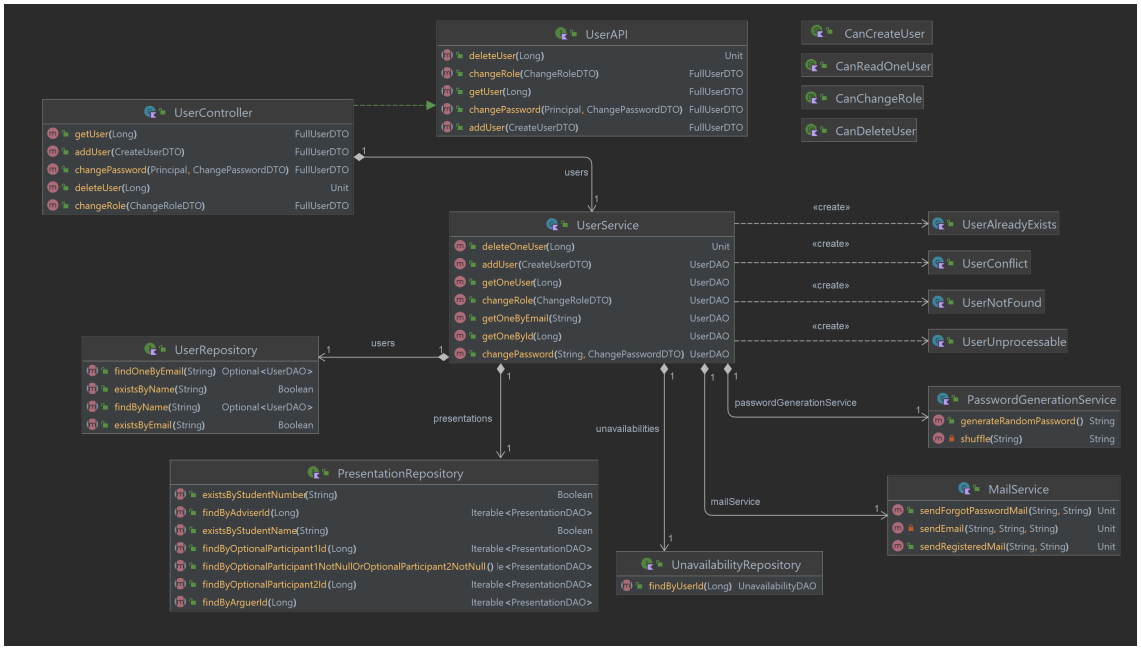

Figura 6.2: Diagrama de arquitetura da gestão dos utilizadores

As operações desenvolvidas com este fim estão indicadas no apêndice A.

6.3.2 Salas

A gestão das salas é uma funcionalidade essencial para o bom funcionamento da aplicação, permitindo o registo, consulta e remoção das mesmas. Estas operações são fundamentais para garantir a correta organização e alocação das apresentações aos espaços físicos disponíveis. Tal como na gestão dos utilizadores foram implementadas validações rigorosas para garantir que, apenas, os utilizadores autorizados podem realizar determinadas ações.

As interações entre componentes e serviços que suportam as operações referidas, estão representadas na figura 6.3.

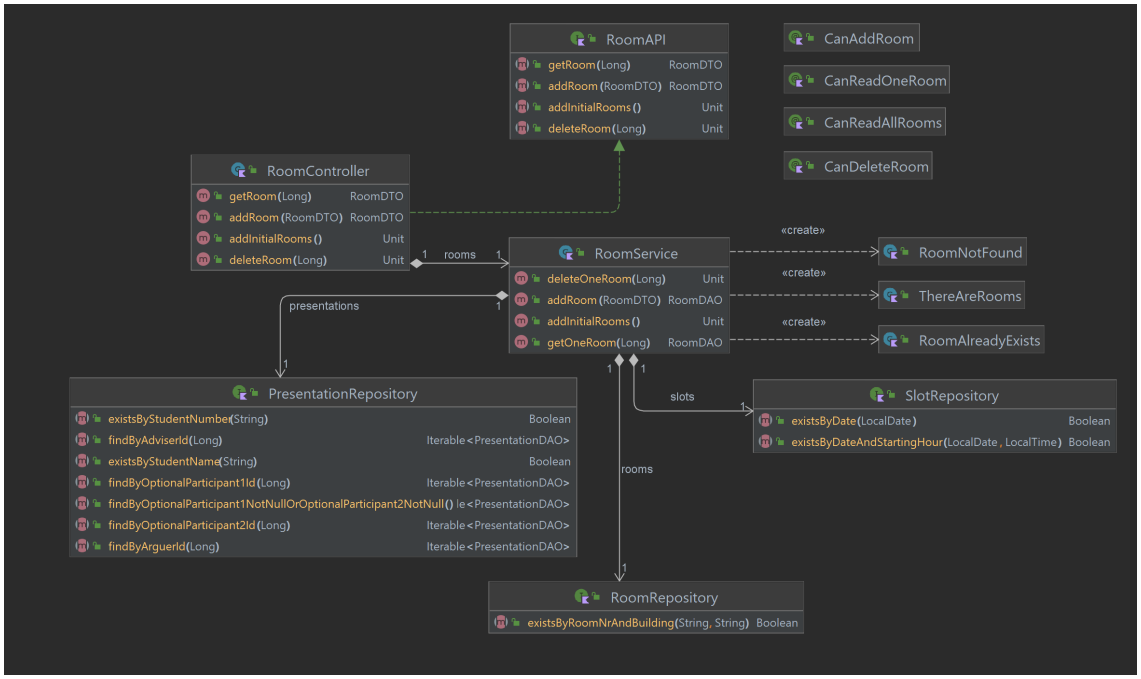

Figura 6.3: Diagrama de arquitetura da gestão das salas

As operações desenvolvidas com este fim estão indicadas no apêndice B.

6.3.3 Slots

A gestão dos slots é uma funcionalidade essencial para o bom funcionamento da aplicação, permitindo a criação, consulta e remoção de slots. Estas operações são fundamentais para garantir a correta organização temporária das apresentações. Tal como na gestão de utilizadores e salas, foram implementadas validações rigorosas para garantir que apenas os utilizadores autorizados podem realizar determinadas ações.

As interações entre componentes e serviços que suportam as operações referidas, estão representadas na figura 6.4.

As operações desenvolvidas com este fim estão indicadas no apêndice C.

6.3.4 Apresentações

A gestão das apresentações é uma funcionalidade fundamental na nossa aplicação, permitindo a criação, atualização, consulta e remoção de apresentações. Acresce a possibilidade de adicionar e remover participantes a cada apresentação.

Este conjunto de funcionalidades engloba diversas operações que podem ser realizadas por diferentes tipos de utilizadores. Deste modo foram implementadas várias validações para garantir que só os utilizadores autorizados podem executar determinadas ações.

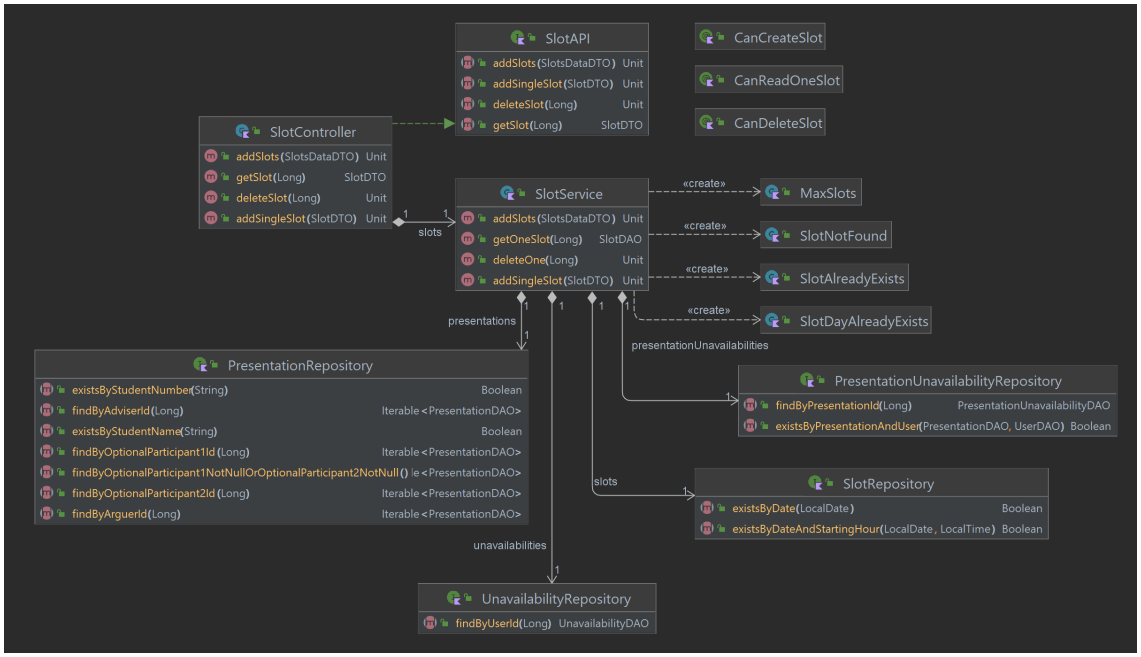

Figura 6.4: Diagrama de arquitetura da gestão dos slots

As interações entre componentes e serviços que suportam as operações referidas, estão representadas na figura 6.5.

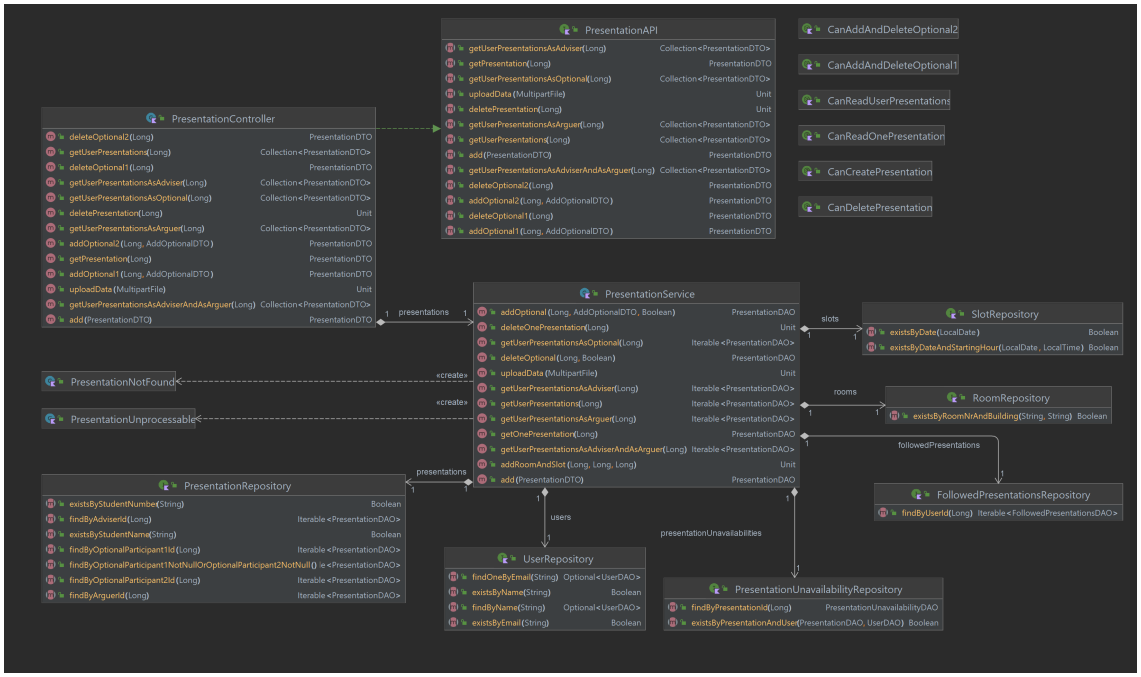

Figura 6.5: Diagrama de arquitetura da gestão das apresentações

As operações desenvolvidas com este fim estão indicadas no apêndice D.

6.3.5 Restrições dos Utilizadores

A gestão das restrições horárias dos utilizadores é uma funcionalidade essencial na nossa aplicação, permitindo a criação, atualização, consulta e remoção de restrições. Este conjunto de funcionalidades engloba diversas operações que podem ser realizadas por diferentes tipos de utilizadores. Deste modo foram implementadas várias validações para garantir que apenas utilizadores autorizados podem executar determinadas ações.

As interações entre componentes e serviços que suportam as operações referidas, estão representadas na figura 6.5.

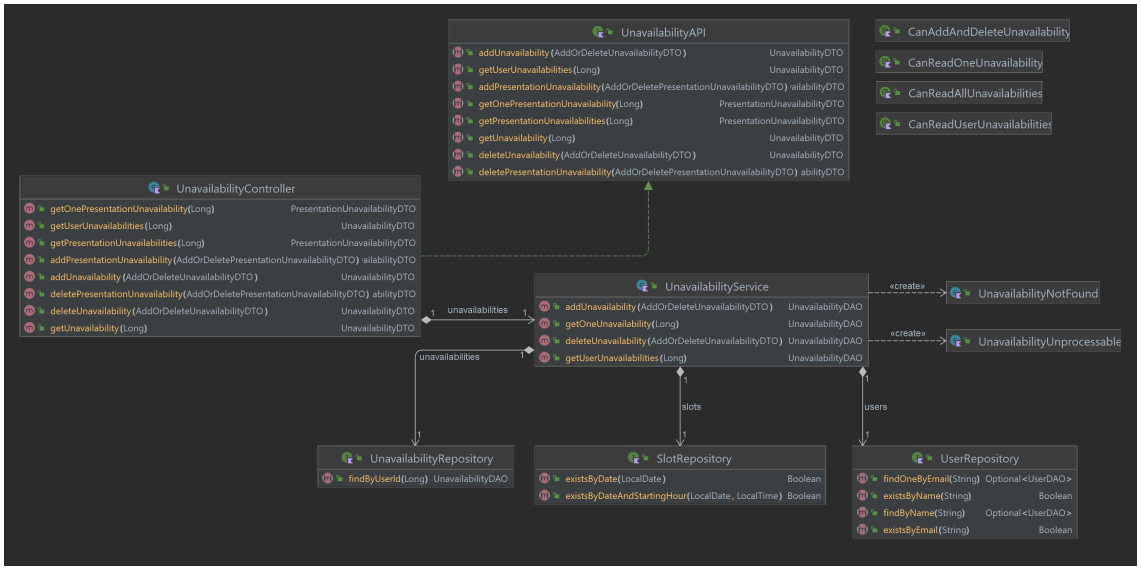

Figura 6.6: Diagrama de arquitetura da gestão das restrições dos utilizadores

As operações desenvolvidas com este fim estão indicadas no apêndice E.

6.3.6 Restrições das Apresentações

A gestão das restrições das apresentações é uma funcionalidade fundamental na nossa aplicação, permitindo a criação, atualização, consulta e remoção de restrições relativas a apresentações. Estas operações são tipicamente utilizadas para indicar as indisponibilidades de pessoas externas à faculdade e que, por isso, não estão registadas na aplicação.

Este conjunto de funcionalidades engloba diversas operações que podem ser realizadas por diferentes tipos de utilizadores. Deste modo foram implementadas várias validações para garantir que apenas utilizadores autorizados podem executar determinadas ações.

As interações entre componentes e serviços que suportam as operações referidas, estão representadas na figura 6.5.

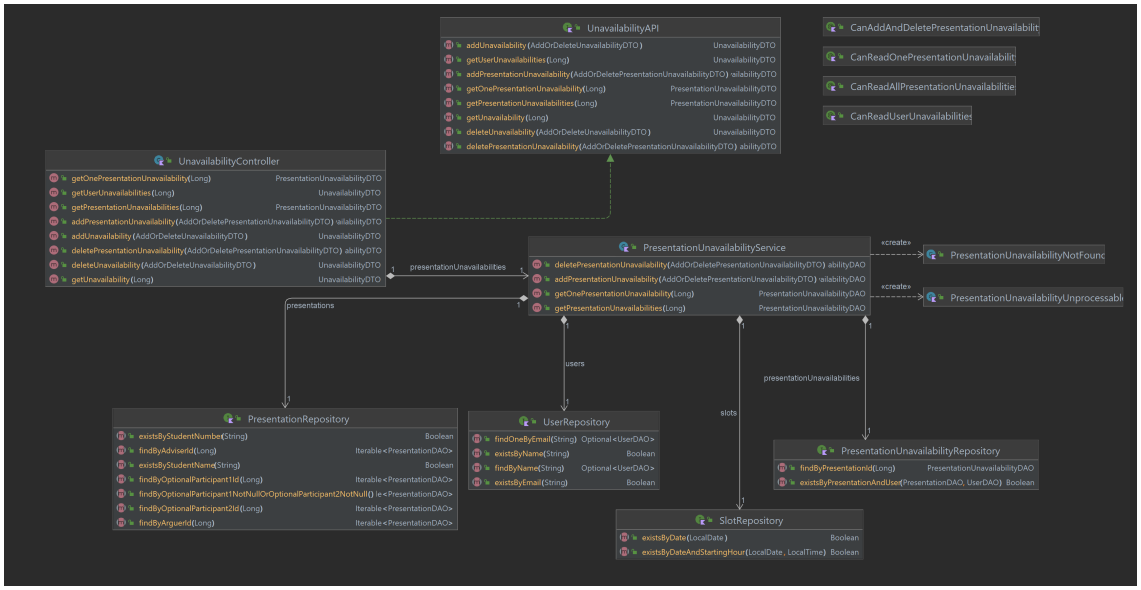

Figura 6.7: Diagrama de arquitetura da gestão das restrições das apresentações

As operações desenvolvidas com este fim estão indicadas no apêndice F.

6.3.7 Apresentações Seguidas

A gestão dos conjuntos de apresentações seguidas é uma funcionalidade essencial na nossa aplicação, permitindo a criação, atualização, consulta e remoção de apresentações seguidas. Este conjunto de funcionalidades engloba diversas operações que podem ser realizadas por diferentes tipos de utilizadores. Deste modo foram implementadas várias validações para garantir que apenas utilizadores autorizados podem executar determinadas ações.

As interações entre componentes e serviços que suportam as operações referidas, estão representadas na figura 6.5.

As operações desenvolvidas com este fim estão indicadas no apêndice G.

6.4 Integração do Microsserviço

A integração do Microsserviço no Backend é feita através de uma chamada a um endpoint POST. A chamada é utilizada para enviar as informações necessárias ao cálculo de alocação de salas e horários às apresentações. Os dados enviados estão encapsulados num objeto JSON que agrupa as seguintes informações:

- A composição do júri de cada apresentação, orientador e vogal;
- Os slots disponíveis;
- As salas disponíveis;

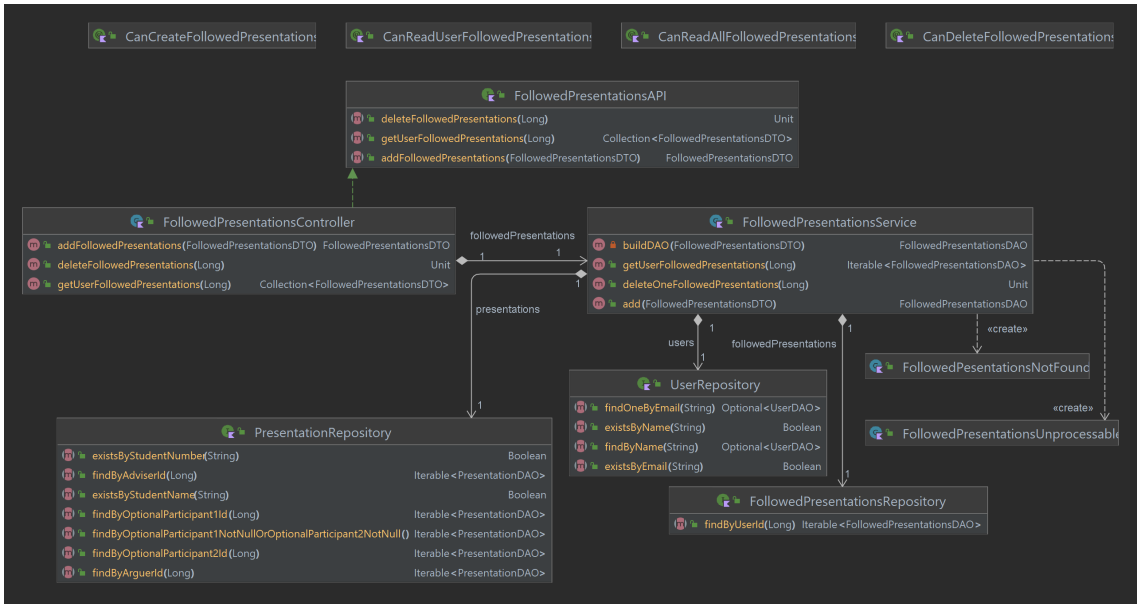

Figura 6.8: Diagrama de arquitetura da gestão das apresentações seguidas

- As indisponibilidades de cada utilizador;
- As indisponibilidades relativas a cada apresentação;
- A indicação dos utilizadores opcionais a cada apresentação;
- As apresentações que precisam de ser agendadas de forma consecutiva.

Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem criar utilizadores.

Após o envio dos dados, o backend retorna uma resposta confirmando o início da computação.

Quando o processamento no microsserviço termina é devolvida uma resposta que contém o ID da apresentação, o ID da sala e o ID do slot alocado para todas as apresentações. Ao receber a resposta, o Backend atualiza a informação na base de dados atribuindo as salas e a hora correta a cada apresentação.

6.5 Segurança e Autenticação dos Utilizadores

A segurança é uma preocupação central em qualquer sistema que lida com dados sensíveis e informações pessoais. Deste modo, foram adotadas diversas técnicas e tecnologias para garantir a proteção, integridade, disponibilidade e confidencialidade dos dados na nossa aplicação.

Inicialmente, indicaremos as principais tecnologias e mecanismos de segurança implementados no nosso Backend. De seguida, explicaremos o processo de login e como é feita a autenticação dos utilizadores na nossa aplicação. Por fim, abordaremos a funcionalidade de recuperação de password, descrevendo o seu procedimento.

No diagrama presente na figura 6.9, é possível observar os componentes utilizados na autenticação dos utilizadores e as suas interações. Este diagrama apresenta as operações de autenticação, geração dos tokens JWT, geração das passwords temporárias, serviço de emails, entre outros.

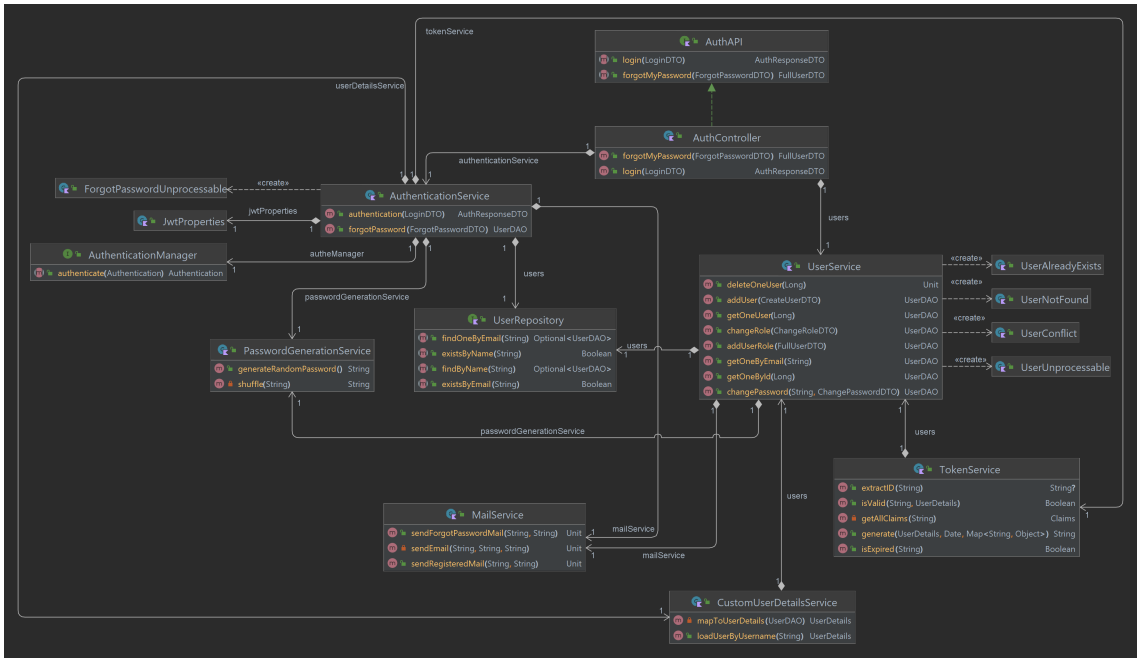

Figura 6.9: Diagrama de autenticação dos utilizadores na aplicação

6.5.1 Tecnologias e Mecanismos de Segurança Utilizados

A segurança da aplicação é acautelada por uma combinação de tecnologias e mecanismos que garantem a integridade, autenticidade e confidencialidade das interações entre o cliente e o servidor. As principais tecnologias e mecanismos de segurança utilizados são:

JWT O JWT é um padrão da internet (RFC 7519) que define uma forma compacta e segura de transmitir informações entre duas partes e é a tecnologia central de autenticação da nossa aplicação. Um JWT é composto por três partes: o *header* (cabeçalho), que indica o tipo do token e o algoritmo de assinatura utilizado; o *payload* (corpo), que contém as reivindicações (claims); e a *signature* (assinatura), que é gerada utilizando um segredo, garantindo a integridade do token [30].

Encriptação das Passwords As passwords dos utilizadores são encriptadas utilizando o algoritmo *bcrypt*. O *bcrypt* é um algoritmo de hashing de passwords projetado para ser resistente a ataques de força bruta. É utilizado um *salt* gerado aleatoriamente de forma a criar um hash único para cada password. O seu uso garante que passwords idênticas não tenham o mesmo hash [4, 6].

Filtros de Validação dos Tokens Para garantir que apenas utilizadores autenticados têm acesso aos recursos do nosso backend, implementámos diversos filtros de validação. Estes filtros verificam se o token JWT está presente, se está expirado e se a sua assinatura é válida. Se o token for inválido ou expirado, a requisição é recusada, protegendo a aplicação de acessos não autorizados ou indevidos.

6.5.2 Login

O processo de login requer que o utilizador forneça o seu email e a respetiva password. Estes dados são enviados para o sistema de autenticação a fim de serem validados. Caso a validação seja bem sucedida, é gerado um JWT contendo o identificador do utilizador, a data de emissão e a data de expiração do token e retornado ao utilizador. Este token é ainda assinado digitalmente, de forma a garantir a sua integridade e autenticidade.

O token JWT tem uma validade de 3.600.000 milissegundos, o que equivale a uma hora. Uma vez passado este período de tempo, o token expira e o utilizador deve voltar a autenticar-se na aplicação.

6.5.3 Autenticação

A autenticação na nossa aplicação é gerida utilizando JWT, garantindo a segurança das interações entre o cliente e o servidor. Inicialmente o utilizador faz login com as suas credenciais e recebe um token. Este token é incluído em todas interações subsequentes com o Backend, permitindo a sua autenticação e o seu acesso às funcionalidades protegidas.

Nas requisições subsequentes, o token JWT é enviado pelo utilizador no cabeçalho de autorização das requisições HTTP. O sistema valida o token, verificando a sua presença, confirmando que não está expirado e assegurando a sua integridade. Se o token for válido, as informações do utilizador são carregadas, permitindo o acesso aos recursos protegidos. Caso o token seja inválido ou estiver expirado, o acesso é negado e o utilizador é convidado a realizar novo login.

6.5.4 Recuperação da Password

O processo de recuperação de password requer que o utilizador forneça o seu email, a fim de ser verificada a sua existência na base de dados.

No caso do endereço de email não estar associado a nenhuma conta na aplicação, é retornada uma mensagem de erro.

No caso do endereço de email ser válido, o sistema gera uma nova password aleatória seguindo os requisitos de segurança estabelecidos (composta por 8 caracteres, contendo uma letra maiúscula, uma letra minúscula e um número), envia a nova password para o email do utilizador e guarda na base de dados a nova password encriptada (utilizando o algoritmo *bcrypt*).

Por fim, e após o login, o utilizador é incentivado a alterar a password.

6.6 Conclusão

Neste capítulo, foi apresentada a arquitetura do Servidor, as tecnologias utilizadas, as principais funcionalidades e operações implementadas, assim como a integração com o Microsserviço e os mecanismos de segurança adotados. O Servidor desempenha um papel central na aplicação, sendo responsável pela gestão das interações entre o Cliente, a base de dados e o Microsserviço, garantindo a correta execução das funcionalidades e a segurança dos dados.

No próximo capítulo, o foco será o Microsserviço, que tem um papel especializado na computação dos horários e na alocação de salas para as apresentações. Será explorado como o Microsserviço recebe, processa e devolve os dados ao Servidor, com destaque para o uso do ASP na resolução deste problema específico. Assim, a integração entre Servidor e Microsserviço será analisada em detalhe, elucidando como esses dois componentes trabalham em conjunto para oferecer uma solução robusta e eficiente.

7

Microsserviço

O microsserviço nasceu da necessidade de computação dos horários das apresentações utilizando um programa ASP. Desta forma, o microsserviço é composto por um programa ASP e por uma API REST. O programa ASP é responsável pela atribuição das salas e dos horários às apresentações de forma otimizada. A API REST é responsável pela comunicação com o Backend e pela evocação do programa ASP.

Neste capítulo iremos detalhar o funcionamento de cada componente deste microsserviço. Inicialmente explicaremos a composição e funcionamento do programa ASP e, por fim, o funcionamento da API REST, indicando a sua composição, execução e a integração do *Clorm*.

7.1 Programa ASP

O programa ASP é responsável por atribuir salas e horários às apresentações, utilizando as restrições de cada utilizador e de cada apresentação.

O processo de alocação foi dividido em duas fases. Uma primeira fase, responsável pela atribuição dos slots às apresentações, e uma segunda, responsável pela alocação das salas às apresentações.

Esta divisão mostrou-se importante na redução da complexidade das otimizações e do seu tempo de execução.

7.1.1 Atribuição de slots às apresentações

Esta primeira parte do programa ASP é responsável pela atribuição de um slot a cada apresentação. É importante salientar que o programa utiliza os factos inseridos pelo *Clorm*, referido no ponto 7.2.7.

Neste ponto utilizaremos excertos do código deste programa ASP para ajudar à sua perceção. No entanto, o código total deste programa está presente no GitHub através do link: `https://github.com/SalvadorSi/Aplicacao-Dissertacao-Salvador-Silva` e também é apresentado no anexo I.

Inicialmente é criado o predicado *sessionParticipants/2* para guardar a informação dos participantes de cada apresentação. Dado que os participantes opcionais de cada apresentação já foram adicionados pelo *Clorm*, este predicado é apenas criado com os orientadores e vogais das diversas apresentações.

```
1 sessionParticipants(Arguer,PresentationID):- jury(PresentationID,Arguer,_).
2 sessionParticipants(Adviser,PresentationID):- jury(PresentationID,_,Adviser).
```

De seguida, é gerado o predicado *Assign/2*, associando cada apresentação ao respetivo slot. Esta geração cria todas as combinações possíveis para os predicados *Assign*, desde que cada apresentação apenas tenha associado um slot e não contrarie os factos negados *Assign* inseridos pelo *Clorm*.

```
1 1 { assign(PresentationID,SlotID) : slot(SlotID) } 1 :-
      jury(PresentationID,_,_).
```

Uma vez feita esta geração, existem situações nas quais o predicado *Assign* indica acontecimentos "ilegais". De forma a abolir estas situações, criámos dois testes. O primeiro teste, descarta todas as opções nas quais existem duas apresentações da mesma pessoa no mesmo slot. O segundo teste, elimina todas as soluções nas quais um slot está associado a mais apresentações do que o número total de salas existentes.

```
1 :- 2{assign(PresentationID,SlotID):sessionParticipants(Participant,
     PresentationID)}, slot(SlotID), sessionParticipants(Participant,_).
2 :- N{assign(PresentationID,SlotID)},slot(SlotID),N=Rooms+1, numberRooms(Rooms
     ).
```

Depois do tratamento do predicado *Assign*, é gerado o predicado *Slots/4* que contém as informações finais deste programa.

```
1 slots(SlotID,PresentationID,Arguer,Adviser) :- assign(PresentationID,SlotID),
      jury(PresentationID,Arguer,Adviser).
```

De forma a facilitar a complexidade da otimização, são criadas novas instâncias do predicado *followedSlots* e são criados dois novos predicados: *Participates* e *Conn*. O predicado *Participates* é criado com a informação do slot em que cada utilizador participa. O predicado *Conn* é criado para indicar os utilizadores com apresentações seguidas.

```
1 followedSlots(SlotID1,SlotID2):- followedSlots(SlotID2,SlotID1).
2 participates(Participant,SlotID) :- assign(PresentationID,SlotID),
      sessionParticipants(Participant,PresentationID).
3 conn(Participant,SlotID1) :- participates(Participant,SlotID1), followedSlots
      (SlotID2,SlotID1), participates(Participant,SlotID2).
```

Por fim, é feita a otimização. A otimização consiste na diminuição do número de apresentações singulares. Para isso é utilizada a sintaxe *#minimize* e os predicados referidos anteriormente.

```
1 #minimize{1,Participant,SlotID:not conn(Participant,SlotID),participates(
      Participant,SlotID)}.
```

7.1.2 Atribuição das salas às apresentações

Esta segunda parte do programa ASP é responsável pela atribuição das salas às apresentações. O objetivo é garantir que cada apresentação ocorra numa sala e minimizar a troca de salas para participantes que têm apresentações consecutivas.

É importante salientar que o programa utiliza os factos inseridos pelo *Clorm*, referido no ponto 7.2.8.

Neste ponto utilizaremos excertos do código deste programa ASP para ajudar à sua perceção. No entanto, o código total deste programa está presente no GitHub através do link: `https://github.com/SalvadorSi/Aplicacao-Dissertacao-Salvador-Silva` e também é apresentado no anexo II.

Inicialmente, o predicado *Room/2* é criado para associar cada apresentação a uma sala, garantindo que cada apresentação tenha exatamente uma sala. Este predicado é definido de forma a assegurar que cada apresentação (*PresentationID*) está associada a uma única sala (*RoomID*), desde que o respetivo slot já tenha sido atribuído.

```
1 1{room(PresentationID,RoomId):room(RoomId)}1:- slots(_,PresentationID,_,_).
```

Para evitar conflitos, o programa introduz restrições que impedem que uma mesma sala seja atribuída a mais do que uma apresentação no mesmo slot. Além disso, uma restrição adicional garante que o número de apresentações atribuídas a salas num determinado slot não excede o número total de salas disponíveis, previamente definido no programa com o predicado *NumberRooms/1*.

```
:- 2{room(PresentationID,RoomId):slots(SlotID,PresentationID,_,_)},slots(
    SlotID,_,_,_),room(_,RoomId).
:- N{room(PresentationID,RoomId):slots(SlotID,PresentationID,_,_)},slots(
    SlotID,_,_,_),N=Rooms+1, numberRooms(Rooms).
```

De seguida, é gerado o predicado *RoomSlot/2*, associando cada sala (*RoomID*) ao respetivo slot (*SlotID*). Este predicado auxilia a aplicação de restrições adicionais para evitar configurações redundantes e melhorar a eficiência do processo de atribuição de salas.

```
roomSlot(RoomId,SlotID):- room(PresentationID,RoomId),slots(SlotID,
    PresentationID,_,_).
```

Foram também adicionadas restrições de simetria, que evitam soluções simétricas nas quais a atribuição de salas poderia ser considerada equivalente a outra solução já gerada. Este processo ajuda a reduzir o espaço de procura e a garantir que a atribuição de salas é feita de forma otimizada e sem redundâncias.

```
:- not followedSlots(SlotID,_), room(Presentation1,Room1),room(Presentation2,
    Room2),slots(SlotID,Presentation1,_,_), slots(SlotID,Presentation2,_,_),
    Presentation1>Presentation2,Room1<Room2.
:- not followedSlots(SlotID,_), roomSlot(Room1,SlotID), not roomSlot(Room2,
    SlotID), room(Room2),Room1>Room2.
```

O predicado *FollowedSlots* indica slots consecutivos, que é essencial para minimizar as trocas de sala entre participantes que têm apresentações seguidas. A regra seguinte usa este predicado para garantir que apresentações consecutivas para um participante têm preferencialmente a mesma sala, minimizando o esforço logístico e facilitando o acompanhamento das apresentações pelos intervenientes.

```
room(Presentation2,RoomId):- room(Presentation2,RoomId), followedSlots(Slot1,
    Slot2),slots(Slot1,Presentation2,Arguer,Adviser),slots(Slot2,
    Presentation2,Arguer,Adviser).
```

```
2 1{room(Presentation2,Room);room(Presentation3,Room)}1 :- room(Presentation1,
    Room), followedSlots(Slot1,Slot2), slots(Slot1,Presentation1,Arguer,
    Adviser), slots(Slot2,Presentation2,_,_),juryMember(Arguer,Presentation2)
    ,slots(Slot2,Presentation3,_,_),juryMember(Adviser,Presentation3),
    Presentation2>Presentation3.
```

O predicado *Room/2* é também utilizado para garantir que apresentações consecutivas com os mesmos participantes partilhem a mesma sala. Este predicado assegura que, se um membro do júri (*juryMember*) participa em duas apresentações consecutivas (*followedSlots*), a sala (*Room*) será a mesma para ambas as apresentações, sempre que possível. Além disso, restrições adicionais são usadas para garantir que outros membros do júri não têm de mudar de sala desnecessariamente.

```
1 room(Presentation2,Room) :- room(Presentation1,Room), followedSlots(Slot2,
    Slot1), slots(Slot1,Presentation1,_,_), juryMember(P1,Presentation1),
    slots(Slot2,Presentation2,_,_),juryMember(P1,Presentation2), juryMember(
    P2,Presentation1), not participates(P2,Slot2), juryMember(P3,
    Presentation2), not participates(P3,Slot1).
```

O predicado *JuryMember/2* é utilizado para identificar os membros do júri de cada apresentação, quer sejam vogais (*Arguer*) ou orientadores (*Adviser*). Estes predicados são gerados com base nas atribuições feitas pelos predicados *Slots/4*.

```
1 juryMember(Participant,PresentationID):- slots(SlotID,PresentationID,
    Participant,_).
2 juryMember(Participant,PresentationID):- slots(SlotID,PresentationID,_,
    Participant).
```

O predicado *Participates/2* é utilizado para identificar em que slots cada participante está envolvido, seja como vogal ou orientador, facilitando assim o processo de verificação de conflitos ou otimização das atribuições de salas.

```
1 participates(Participant,SlotID) :- slots(SlotID,_,Participant,_).
2 participates(Participant,SlotID) :- slots(SlotID,_,_,Participant).
```

A otimização é implementada através de duas minimizações diferentes, ambas utilizando a sintaxe *#minimize*. A primeira minimização visa reduzir o número de trocas de sala entre slots consecutivos, atribuindo a mesma sala para apresentações seguidas sempre que possível. A segunda minimização tenta minimizar as trocas de sala para participantes que têm apresentações consecutivas mas que são diferentes, ou seja, mesmo quando as apresentações não são seguidas diretamente, o objetivo é minimizar a necessidade de mudança de sala para o mesmo participante.

```
#minimize{1@1,RoomId,Slot1:roomSlot(RoomId,Slot1),roomSlot(RoomId,Slot2),
    followedSlots(Slot2,Slot1)}.
#minimize { 1@2,Participant,Presentation1:juryMember(Participant,
    Presentation1),juryMember(Participant,Presentation2),slots(Slot1,
    Presentation1,_,_),slots(Slot2,Presentation2,_,_), followedSlots(Slot1,
    Slot2),Presentation1!=Presentation2,room(Presentation1,Room1),room(
    Presentation2,Room2),Room1!=Room2 }.
```

Por fim, o predicado *FinalSlots/3* é criado para gerar a configuração final, onde cada apresentação está associada a um slot e a uma sala, facilitando a visualização dos resultados finais do agendamento.

```
finalSlots(RoomID,SlotID,PresentationID):- slots(SlotID,PresentationID,_,_),
    room(PresentationID,RoomID).
```

7.2 API REST

A API REST está encarregue de fazer a ligação entre o Backend e o programa ASP. Esta ligação é executada através de um pedido HTTP que é feito pelo Backend sempre que é necessária a computação das salas e horários das apresentações.

A programação desta API foi realizada recorrendo à linguagem de programação Python, uma vez que, como referido no ponto 3.2, dispõe da biblioteca *Clorm* que facilita a sua interação com programas ASP e com o solver clingo.

De forma a facilitar a perceção do funcionamento desta API, iremos dividir a análise nos seguintes pontos:

- **Criação da API** — Indicação da tecnologia utilizada para a criação do microsserviço;

- **Composição e Requisitos** — Indicação da composição da API e dos requesitos necessários para o bom funcionamento do microsserviço, os quais providenciados pelo Backend;
- **Criação das interfaces dos predicados** — Explicação das interfaces dos predicados, indicando as suas funções na computação do horário final;
- **Processamento dos Dados** — Extração e tratamento dos dados provenientes do Backend;
- **Ficheiro Auxiliar** — Explicação da necessidade de criação de um ficheiro auxiliar e a sua utilidade;
- **Geração dos Factos** — Geração dos factos necessários para a computação do horário final;
- **Atribuição de Slots às Apresentações** — Explicação de como é feita a computação da atribuição dos slots às apresentações;
- **Atribuição de Salas às Apresentações** — Explicação de como é feita a computação da atribuição das salas às apresentações;
- **Resposta** — Composição da resposta enviada para o Backend.

Neste ponto utilizaremos excertos do código para ajudar nas explicações relativas ao seu funcionamento. No entanto, o código total desta API está presente no GitHub através do link: `https://github.com/SalvadorSi/Aplicacao-Dissertacao-Salvador-Silva`.

7.2.1 Criação da API

A criação desta API foi realizada com o suporte da tecnologia Flask.

O Flask é um microframework web escrito em python que permite o desenvolvimento rápido e eficiente de aplicações web e APIs. É considerado um microframework porque não impõe a utilização de ferramentas ou bibliotecas específicas, mas permite a integração de bibliotecas e extensões conforme as necessidades dos projetos. Esta abordagem minimalista torna-o ideal para desenvolvimentos de APIs simples.

A escolha do Flask para o desenvolvimento deste microsserviço foi motivada por diversas razões:

- **Simplicidade:** O Flask é conhecido pela sua simplicidade e abordagem minimalista. Fornece o básico para a criação de APIs;
- **Flexibilidade:** Permite aos programadores escolherem as bibliotecas e ferramentas que pretendem usar, neste caso o *Clorm*;

- **Documentação:** Oferece uma documentação detalhada e bem estruturada, facilitando o desenvolvimento e a resolução de problemas.

7.2.2 Composição e Requisitos

Esta API é composta por apenas um pedido HTTP do tipo POST. Ao executar esta função, é necessário que o Backend (6) envie um JSON, no corpo (*body*) do pedido com todas as informações necessárias para a computação de um horário final. Estas informações são as seguintes:

- Constituição do júri de cada apresentação;
- Slots nos quais podem ser marcadas as apresentações;
- Salas disponiveis;
- Indisponibilidades de cada membro do júri;
- Restrições referentes às apresentações - indicadas pelos utilizadores - tipicamente utilizadas para indicar as indisponibilidades de pessoas externas à faculdade e que, por isso, não estão registadas na aplicação (ex: membros de empresas);
- Utilizadores opcionais adicionados a uma apresentação específica;
- As apresentações que necessitam ser marcadas de seguida - indicadas pelos orientadores de cada apresentação.

7.2.3 Criação das interfaces dos predicados

As informações provenientes do Backend necessitam de ser transformadas em predicados para poderem ser utilizadas na atribuição das salas e dos slots às apresentações. Neste sentido é necessário criar as suas interfaces.

Estes predicados podem ser divididas em dois tipos: os predicados de input, que serão inseridos no programa ASP com o auxilio do *Clorm*, e os predicados de output, que irão conter os valores finais das computações.

Existem oito predicados de input diferentes, como se pode observar no código fonte 7.1.

Código Fonte 7.1: Interfaces dos predicados de input dos programas ASP

```
1 class NumberRooms(Predicate):
2     number = ConstantField()
3
4 class Room(Predicate):
5     roomID = IntegerField
6
```

```
class Slot(Predicate):
    slotID = IntegerField

class FollowedSlots(Predicate):
    slotID1 = IntegerField
    slotID2 = IntegerField

class Jury(Predicate):
    presentationID = IntegerField
    adviserID = IntegerField
    arguerID = IntegerField

class SessionParticipants(Predicate):
    userID = IntegerField
    presentationID = IntegerField

class Participates(Predicate):
    userID = IntegerField
    slotID = IntegerField

class Assign(Predicate):
    presentationID = IntegerField
    slotID = IntegerField
```

O predicado *NumberRooms/1* funciona como uma constante, indicando o número de salas disponiveis para a computação do horário final.

O predicado *Room/1* referencia uma sala disponível e será apenas utilizado na computação das salas para as apresentações. O predicado *Slot/1* referencia um slot disponível e será apenas utilizado na computação dos slots para as apresentações. Cada sala e cada slot contêm o seu identificador (*ID*), sendo este importante para manter a consistência com a base de dados.

O predicado *FollowedSlots/2* guarda os identificadores de dois slots. Este predicado indica que os dois slots com os identificadores guardados são seguidos. Há a possibilidade de um slot existir em dois predicados *FollowedSlots* diferentes, um para indicar que slot está imediatamente antes e outro para indicar o slot que está imediatamente depois. Para dois slots serem considerados seguidos é necessário, por definição, que a diferença entre as suas horas de início seja superior à duração limite de uma apresentação, 45 minutos, mais a duração do intervalo, 15 minutos, ou seja 1 hora. Esta restrição divide os slots em conjuntos, um conjunto da manhã e um conjunto da tarde para cada dia em que é possível serem realizadas as apresentações. Esta divisão é particularmente importante para a atribuição das salas aos slots, quando estes já tiverem ligados às apresentações.

O predicado *Jury/3* especifica o grupo de juris para cada apresentação. A apresentação e os participantes do grupo de juris são guardados com o seu identificador de forma a manter a consistência com a base de dados.

O predicado *SessionParticipants/2* guarda os utilizadores opcionais adicionados para estarem presentes numa apresentação especifica. Para isto é guardado o identificador deste mesmo utilizador e da apresentação em que ele irá participar.

O predicado *Participates/2* é criado com o identificador de um utilizador e com o identificador de um slot. Este predicado indica os slots nos quais o utilizador com o identificador guardado não pode participar.

O predicado *Assign/2* é criado com o identificador de uma apresentação e com o identificador de um slot. À semelhança do predicado "Participates", este predicado indica quais os slots nos quais a apresentação com o identificador guardado não pode ser atribuída.

Em termos de predicados classificados como predicados de output, existem dois, referidos no código fonte 7.2. Esta particularidade deve-se ao facto do programa ASP estar dividido em duas partes, como indicado no ponto 7.1

Código Fonte 7.2: Interfaces dos predicados de output dos programas ASP

```
class Slots(Predicate):
    slotID = IntegerField
    presentationID = IntegerField
    adviserID = IntegerField
    arguerID = IntegerField

class FinalSlots(Predicate):
    roomID = IntegerField
    slotID = IntegerField
    presentationID = IntegerField
```

O predicado *Slots/4*, irá conter o resultado da atribuição das apresentações aos slots. Para além de guardar os identificadores do slot e da apresentação nele contida, também guarda os identificadores dos membros do juri. Esta característica é especialmente importante, uma vez que este predicado é utilizado como predicado de input na atribuição das salas às apresentações. Ao ser guardada toda esta informação sob o mesmo predicado, a complexidade das verificações e das otimizações efetuadas diminui.

O predicado *FinalSlots/3* irá conter o resultado final da computação das salas, slots e apresentações. Não há necessidade de guardar os membros do juri de cada apresentação como na primeira computação, dado que este resultado será enviado de volta para o Backend e esta informação está presente na base de dados.

7.2.4 Processamento dos Dados

Uma vez executado o pedido POST, existe uma função responsável pela verificação do conteúdo recebido. Esta função assegura o bom funcionamento do tratamento dos dados no resto da aplicação.

De seguida são extraidos os dados recebidos no JSON e adicionados a um dicionário. Esta transferência dos dados foi feita com o intuito a melhorar a legibilidade e a compreensão do código.

7.2.5 Ficheiro Auxiliar

Verificou-se a necessidade de escrever um ficheiro auxiliar, uma vez que, no *Clorm*, não é possível a inserção de restrições personalizadas. Desta forma, as restrições que indicam quais apresentações devem ser atribuídas a slots seguidos, são escritas neste ficheiro auxiliar e, mais tarde, carregadas como um programa ASP.

Existe uma função responsável pela escrita destas restrições no ficheiro auxiliar. Esta função, referida no código fonte 7.3, utiliza o dicionário referido no ponto anterior para obter os dados das apresentações que precisam de ser atribuídas a slots consecutivos. Uma vez obtida essa informação, é escrito em linguagem ASP a restrição no ficheiro auxiliar.

Código Fonte 7.3: Função auxiliar para a criação das restrições referentes às apresentações seguidas

```
1 def write_followed_presentations_program(data, program_name):
2     followedPresentations = data.get(FOLLOWED_PRESENTATIONS_KEY)
3 
4     file = open(program_name,"w")
5     for followed in followedPresentations:
6         ids = followed[FOLLOWED_PRESENTATIONS_IDS_KEY]
7         file.write(f"ok:- assign({ids[0]}, H1), assign({ids[1]}, H2),
    followedSlots(H1, H2).\n")
8         file.write(f"ok:- assign({ids[0]}, H1), assign({ids[1]}, H2),
    followedSlots(H2, H1).\n")
9         file.write(":- not ok.\n")
```

7.2.6 Geração dos Factos

Para facilitar a leitura e a perceção do código, foi criada uma função responsável pela criação dos factos. Esta função, apresentada no código fonte 7.4, recebe o dicionário referido no ponto 7.2.4 e gera duas listas de factos. Uma primeira lista *slot_assignment_facts*, na qual estarão contidos os factos necessários para a computação das apresentações e dos slots disponíveis, e uma segunda lista *room_assignment_standard_facts* que irá conter os factos padrão necessários para a computação das salas e das apresentações.

É importante destacar a particularidade dos predicados *Participates* e *Assign* serem negados na sua criação. Esta peculiaridade deve-se ao facto de ambos descreverem restrições nas quais o ator não poder ser associado ao slot com o identificador guardado. No predicado *Participates* é o utilizador com o identificador guardado que não pode ter a sua apresentação marcada no slot. No predicado *Assign* é a apresentação com o identificador recebido que não pode ser marcada no slot.

Para cada utilizador será criado o predicado *Participates* tantas vezes quantos os slots nos quais ele tem restrições.

Para cada apresentação será criado o predicado *Assign* tantas vezes quantos os slots nos quais a mesma não pode ser atribuída.

Código Fonte 7.4: Função responsável pela geração dos factos

```
1  def generate_facts(data, slot_assignment_facts,
       room_assignment_standard_facts):
2      # Access extracted data
3      rooms = data.get(ROOMS_KEY)
4      followed_slots = data.get(FOLLOWED_SLOTS_KEY, [])
5      all_presentations_juries = data.get(ALL_PRESENTATIONS_JURIES_KEY, [])
6      optionals = data.get(OPTIONALS_KEY, [])
7      user_unavailabilities = data.get(USERS_UNAVAILABILITIES_KEY)
8      presentation_restricitons = data.get(PRESENTATIONS_RESTRICTIONS_KEY)
9
10     # Create Rooms predicate
11     slot_assignment_facts.append(NumberRooms(number=str(len(rooms))))
12     room_assignment_standard_facts.append(NumberRooms(number=str(len(rooms)))
       )
13
14     for room in rooms:
15         room_assignment_standard_facts.append(Room(roomID=room.get(
       ROOM_ID_KEY)))
16
17     # Create Slot and FollowedSlots predicates
18     for set_of_slots in followed_slots:
```

```
19          for i, slot in enumerate(set_of_slots):
20              slot_id = slot.get(SLOT_ID_KEY)
21              slot_assignment_facts.append(Slot(slotID=slot_id))
22
23              if i < len(set_of_slots) - 1:
24                  next_slot_id = set_of_slots[i + 1].get(SLOT_ID_KEY)
25                  slot_assignment_facts.append(FollowedSlots(slotID1=slot_id,
    slotID2=next_slot_id))
26                  room_assignment_standard_facts.append(FollowedSlots(slotID1=
    slot_id, slotID2=next_slot_id))
27
28      # Create Jury predicate
29      for presentation_jury in all_presentations_juries:
30          slot_assignment_facts.append(Jury(presentationID=presentation_jury[
    PRESENTATION_ID_KEY], adviserID=presentation_jury[ADVISER_ID_KEY],
    arguerID=presentation_jury[ARGUER_ID_KEY]))
31
32      # Create OptionalParticipants predicate
33      for item in optionals:
34          slot_assignment_facts.append(SessionParticipants(userID=item[
    USER_ID_KEY], presentationID=item[PRESENTATION_ID_KEY]))
35
36      # Create Participates negative predicate
37      for item in user_unavailabilities:
38          for slot in item[SLOTS_ID_KEY]:
39              slot_assignment_facts.append(Participates(userID=item[USER_ID_KEY
    ], slotID=slot, sign=False))
40
41      # Create Assign negative predicate
42      for item in presentation_restricitons:
43          for slot in item[SLOTS_ID_KEY]:
44              slot_assignment_facts.append(Assign(presentationID=item[
    PRESENTATION_ID_KEY], slotID=slot, sign=False))
```

7.2.7 Atribuição de Slots às Apresentações

Efetuada a escrita do ficheiro auxiliar, inicia-se a primeira integração do *Clorm* na componente Python.

Inicialmente é criado um objeto do tipo *Control*. Este é o objeto onde serão carregados os predicados de input, referidos no ponto 7.2.3, os programas ASP e onde será executado o solver.

No ato da criação do objeto *Control*, como é possível conferir no código fonte 7.5, são indicados os predicados de input e de output do programa ASP.

Código Fonte 7.5: Criação do objeto de tipo Control

```
ctrl = Control(unifier=[Slot, NumberRooms, FollowedSlots, Jury,
    SessionParticipants, Slots, Participates, Assign])
```

De seguida, é carregado o programa auxiliar, o programa ASP responsável pela ligação das apresentações aos slots e adicionada a lista dos factos. Por fim, é executado o solver e guardado o predicado de output, como se verifica no código fonte 7.6.

Código Fonte 7.6: Ligação das apresentações aos slots

```
#Load the asp file that encodes the problem domain.
ctrl.load(FOLLOWED_PRESENTATIONS_PROGRAM)
ctrl.load(SLOTS_ASSIGNMENT_PROGRAM)

# Add the instance data and ground the ASP program
ctrl.add_facts(FactBase(slot_assignment_facts))
ctrl.ground([("base", [])])

result = None

def on_model(model):
    nonlocal result
    result = [(str(model)), model.cost]

with ctrl.solve(on_model=on_model, async_=True) as handle:
    handle.wait(TIME_LIMIT)
    handle.cancel()

slots_list = result[0].split(" ")
```

É ainda importante realçar a necessidade de acrescentar um limite de tempo na execução do solver. Esta medida foi implementada para que o programa não corra até acabar percorrer todas as soluções possíveis. Frequentemente, o solver encontra rapidamente uma solução ótima mas continua a verificar todas as soluções possíveis na expectativa de encontrar alguma melhor, o que, muitas vezes, não acontece. Após a realização de vários testes, 10 minutos revelou ser o valor que para os utilizadores é um período aceitável e no qual o programa obtém uma solução que, mesmo que nem sempre seja a solução ótima, é considerada aceitável.

7.2.8 Atribuição de Salas às Apresentações

A atribuição das salas às apresentações é feita consoante o número de conjuntos de slots que existem. Um conjunto de slots contém todos os slots referentes a uma manhã ou uma

tarde de um determinado dia.

Esta característica é fundamental para a redução do tempo de execução na atribuição das salas às apresentações. As otimizações realizadas no programa ASP procuram a atribuição da mesma sala a slots seguidos e às apresentações nas quais participam os mesmos docentes. Uma vez que os conjuntos de slots existem em representação dos períodos de apresentações seguidas, não se verifica a necessidade de ser atribuída a mesma sala à última apresentação de um conjunto e à primeira apresentação de outro.

Para cada conjunto de slots é, inicialmente, criada uma lista com o valor dos factos padrão referidos no ponto 7.2.6, uma vez que estes factos são necessários em todas as iterações. De seguida, são adicionados a esta mesma lista os predicados do tipo *Slots* que pertencem ao conjunto a ser computado. Estes predicados são o resultado de output da computação dos slots às apresentações.

Uma vez criados os factos necessários na iteração, é iniciada uma nova integração do *Clorm* na componente Python. Igualmente à atribuição dos slots às apresentações (ponto 7.2.7), é criado um objeto do tipo *Control*, carregado o programa ASP correspondente à atribuição das salas às apresentações, adicionada a lista de factos e executado o solver, como se verifica no código fonte 7.7. Por fim, o output do solver é guardado numa variável para ser utilizado mais à frente.

Código Fonte 7.7: Ligação das salas às apresentações

```
# Loop through each set in followed_slots
for sublist in followed_slots:
    #add standard facts for room assignment
    room_assignment_facts = list(room_assignment_standard_facts)

    for item in sublist:
        slot_id = item[SLOT_ID_KEY]

        # Find slots that match the slotID and add them into the facts list
        for slots_str in slots_list:
            if int(slots_str.split('(')[1].split(',')[0]) == slot_id:
                numbers = [int(num) for num in slots_str[6:-1].split(",")]
                room_assignment_facts.append(Slots(slotID=numbers[0],
    presentationID=numbers[1], adviserID=numbers[2], arguerID=numbers[3]))
    # -------------------------------
    #       Clorm program
    # -------------------------------
    ctrl = Control(unifier=[NumberRooms, Room, FollowedSlots, Slots,
    FinalSlots])
    ctrl.add_facts(FactBase(room_assignment_facts))
    ctrl.load(ROOMS_ASSIGNMENT_PROGRAM)
```

```
20      ctrl.ground([("base", [])])
21
22      with ctrl.solve(on_model=on_model, async_=True) as handle:
23          handle.wait(TIME_LIMIT)
24          handle.cancel()
25      final_rooms_list.extend(result[0].split(" "))
```

A estratégia de adicionar um tempo limite também foi utilizada na computação das salas. É importante referir que esta computação não costuma atingir o tempo limite, a medida foi adicionada por precaução, em caso de erro.

7.2.9 Resposta

A variável que contém o resultado final, sala e horário de cada apresentação, é adicionada a um objeto JSON e enviada para o Backend, onde será recebida e tratada de forma a manter a consistência na base de dados.

7.3 Conclusão

No presente capítulo, foi detalhado o funcionamento do Microsserviço, incluindo o uso de ASP, implicando a divisão do programa em duas fases e a implementação da API REST que faz a ponte com o Servidor. Através da utilização do ASP, o Microsserviço resolve de forma eficiente o problema de alocação de horários e salas, comunicando os resultados com o backend da aplicação.

No capítulo seguinte, será apresentado o Cliente da aplicação, onde exploraremos as tecnologias utilizadas no seu desenvolvimento, a arquitetura, e como este se comunica com o Servidor para garantir uma experiência fluida e interativa. Além disso, será abordada a navegação entre as várias páginas e as funcionalidades disponibilizadas aos utilizadores, completando assim a visão sobre a interação da aplicação com o utilizador final.

8

Cliente

O Cliente, que representa o Frontend da nossa aplicação, foi desenvolvido com o objetivo de proporcionar uma interface intuitiva e de fácil navegação para os utilizadores. A sua responsabilidade vai além da apresentação de dados ao utilizador, sendo também responsável pela interação com o servidor.

Neste capítulo, abordaremos as tecnologias que sustentam o desenvolvimento da interface, explicando as razões por detrás da sua escolha. De seguida apresentaremos a estrutura modular adotada, destacando as vantagens desta abordagem em termos de manutenibilidade e escalabilidade. Será também analisado o mecanismo de comunicação com o servidor Backend e o processo de navegação dentro da aplicação. Por fim será feita uma apresentação das diversas páginas da interface, detalhando as suas funcionalidades.

O código referente a esta interface web está presente no GitHub através do link: `https://github.com/SalvadorSi/Aplicacao-Dissertacao-Salvador-Silva`.

8.1 Tecnologias

Neste ponto iremos apresentar as tecnologias utilizadas no desenvolvimento do Frontend da nossa aplicação, destacando como cada uma delas desempenhou um papel fundamental na criação de uma interface web eficiente e intuitiva.

Na escolha das tecnologias a utilizar foi crucial que os seguintes requisitos fossem atendidos:

- Desempenho e Responsividade — É fundamental que a interface apresente um carregamento rápido e responda de forma fluida às interações do utilizador;
- Manutenibilidade — A utilização de um código que seja fácil de ler, entender e manter é vital para facilitar atualizações futuras;

- Escalabilidade — É necessário que o sistema suporte um crescimento potencial do projeto, permitindo a adição de novas funcionalidades;

- Interatividade — A experiência do utilizador deve ser rica e interativa.

É importante mencionar que, embora existam diversas tecnologias que atendem a estes requisitos, a escolha final foi incentivada pela nossa experiência prévia com as ferramentas escolhidas.

8.1.1 React

O React é uma biblioteca open-source do JavaScript, desenvolvida pelo Facebook, utilizada na construção de interfaces de utilizador em páginas web. Com uma abordagem baseada em componentes, o React permite o desenvolvimento modelar. Uma das suas princiapis vantagens é a renderização eficiente: apenas as partes da página que sofreram alterações são atualizadas, evitando renderizações desnecessárias de elementos que não foram modificados [44].

A escolha do React para o desenvolvimento do Frontend da nossa aplicação foi motivada pela sua capacidade de criar interfaces altamente responsivas e interativas. A sua abordagem modular alinha-se com os requisitos de manutenibilidade e escalabilidade indicados, uma vez que componentes isolados facilitam a leitura, compreensão e manutenção do código. A disponibilidade de uma vasta gama de bibliotecas e ferramentas desenvolvidas para serem facilmente integradas com o React revelou-se como uma vantagem significativa, aumentando a funcionalidade da aplicação.

8.1.2 TypeScript

O TypeScript, desenvolvido pela Microsoft, é uma linguagem de programação open-source que atua como um superconjunto do JavaScript, adicionando tipagem estática ao código. A tipagem do código permite a deteção de erros durante o processo de compilação, antes da execução da aplicação, oferecendo uma camada adicional de segurança e confiabilidade [45, 20].

Optar pelo TypeScript contribuiu significativamente para a manutenibilidade da nossa aplicação. A tipagem estática forneceu uma camada de segurança adicional, acelerando o desenvolvimento e facilitando o processo de debug. O TypeScript não só atende ao requisito de manutenibilidade, como também promoveu um código mais robusto e menos propenso a erros.

8.1.3 Next.js

O Next.js é uma framework de desenvolvimento React que permite a construção de aplicações web com renderização do lado do servidor, a geração de sites estáticos e dispõe de funcionalidades de roteamento automático. Criada pela Vercel, o Next.js simplifica o desenvolvimento de aplicações React, otimizando o seu desempenho e fornecendo uma estrutura sólida para o roteamento automático [37].

A utilização do Next.js na nossa aplicação foi motivada pela sua capacidade de fornecer um desempenho superior e uma experiência de utilizador otimizada. A renderização do lado do servidor permite que as páginas sejam rapidamente carregadas, melhorando a eficiência da interface. A estrutura de roteamento automático facilita a organização do projeto e a criação de novas páginas, atendendo ao requisito de escalabilidade.

8.2 Arquitetura

O Frontend da nossa aplicação foi desenvolvido com uma arquitetura modular e escalável, dividida em componentes que facilitam a navegação, a manutenção e o desenvolvimento do código.

A estrutura de pastas do projeto, presente no diagrama da figura 8.1, apresenta a organização modular adotada.

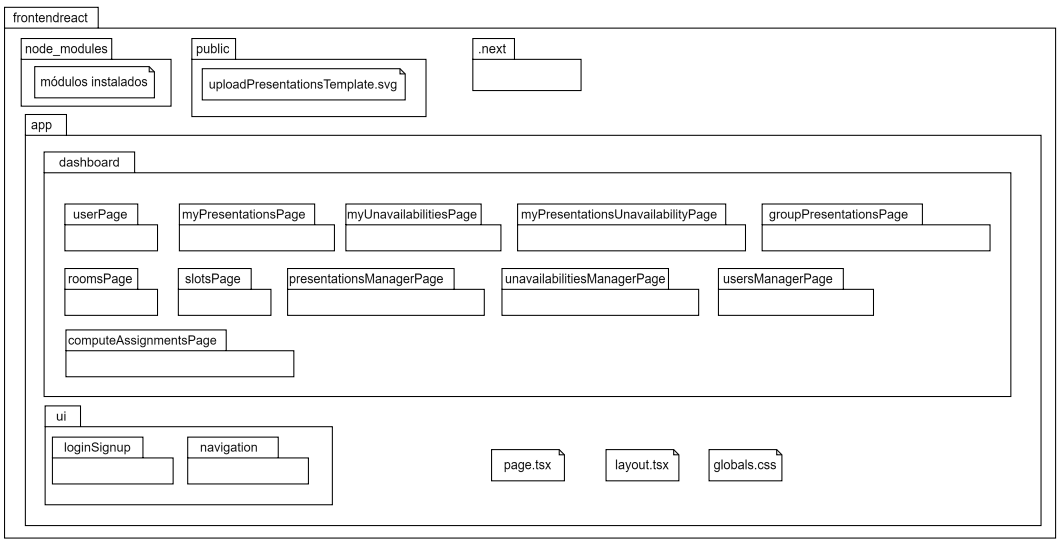

Figura 8.1: Diagrama da Arquitetura do Frontend

A pasta *node_modules* contém as dependências externas do projeto, instaladas através do gestor de pacotes npm. Esta pasta é fundamental para o funcionamento do projeto, uma vez que inclui todos os módulos necessários à execução do código da aplicação.

A pasta *public* foi criada com o intuito de guardar ficheiros estáticos acessíveis publicamente, como imagens e ficheiros SVG. Os recursos armazenados nesta pasta podem ser referenciados diretamente nas páginas da aplicação.

A pasta *.next* é gerada automaticamente pela framework Next.js durante o processo de *build* e desenvolvimento da aplicação. Esta pasta contém ficheiros compilados e otimizados essenciais para a execução da aplicação.

A pasta *app* contém a lógica e os componentes das páginas da interface web. Esta pasta é composta por duas pastas, *dashboard* e *ui*, e por três ficheiros:

- O ficheiro *page.tsx* define a página inicial da aplicação, onde é feito o login dos utilizadores;
- O ficheiro *layout.tsx* define o layout da aplicação;
- O ficheiro *globals.css* define a aparência global da aplicação.

A pasta *dashboard* agrupa as páginas acessíveis aos utilizadores autenticados. No ponto 8.5 individualizaremos cada página e entraremos em mais detalhe.

A pasta *ui* contém o formulário de login e a discriminação da navegação pelas páginas da aplicação, consoante o utilizador que está a aceder.

8.3 Comunicação com o Servidor

A comunicação entre o Frontend da aplicação e o servidor é realizada através da biblioteca Axios, que facilita o envio de pedidos HTTP para endpoints REST. Estes endpoints são fornecidos pelo Backend e seguem os métodos HTTP padrão: GET, POST, PUT, e DELETE, conforme as operações indicadas no ponto 6.3.

Durante o processo de login, o servidor gera e envia um token JWT. Este token é guardado em *SessionStorage*, juntamente com o ID do utilizador, extraído do próprio token. O uso de *SessionStorage* para o armazenamento temporário garante que o token é guardado apenas durante a sessão ativa do utilizador, reforçando a segurança da aplicação.

Sempre que um utilizador realiza ações que envolvem a manipulação ou consulta de dados no servidor (com exceção das operações login e recuperação de password), o token guardado em *SessionStorage* é incluído no cabeçalho do pedido HTTP. Este token autentica e valida as permissões do utilizador.

8.4 Navegação

A navegação na aplicação é facilitada por uma interface intuitiva, onde cada tipo de utilizador tem acesso a páginas com funcionalidades distintas, através da *dashboard*. A *dashboard*, posicionada na parte esquerda do ecrã, apresenta botões que correspondem aos *links* das páginas disponíveis para cada tipo de utilizador. Este design permite uma navegação rápida e intuitiva, garantindo que os utilizadores conseguem localizar facilmente as opções que necessitam.

Um utilizador do tipo *USER* tem acesso exclusivamente às funcionalidades que dizem respeito à sua conta, incluindo a gestão das suas informações pessoais, das suas apresentações, das suas restrições horárias e das restrições aplicáveis às suas apresentações.

Um utilizador do tipo *ORGANIZER* tem um acesso às páginas disponiveis para um *USER*, e tem ainda permissões adicionais. Estas permissões estão relacionadas com a gestão de todos os conteúdos da aplicação, incluindo salas, slots, apresentações, restrições horárias e utilizadores.

Um utilizador do tipo *ADMIN* possui acesso às páginas disponiveis para um *ORGANIZER*, no entanto, tem acesso a funcionalidades extra, tais como, a remoção de contas de utilizadores com a *role ORGANIZER* e capacidade de alterar as *roles* dos utilizadores.

O diagrama de navegação na figura 8.2 ilustra visualmente como as permissões e acessos são distribuídos entre os diferentes tipos de utilizadores, assegurando uma separação clara entre os vários níveis de acesso.

8.5 Páginas e Funcionalidades

Neste ponto, será feita uma descrição detalhada das páginas que compõem a interface do utilizador, destacando as suas funcionalidades principais e o papel que desempenham na experiência geral do utilizador. Cada página foi desenhada para garantir uma navegação eficiente, permitindo que os utilizadores acedam rapidamente às operações desejadas.

De seguida, apresentamos uma análise individual de cada página da aplicação, com uma explicação das suas funcionalidades.

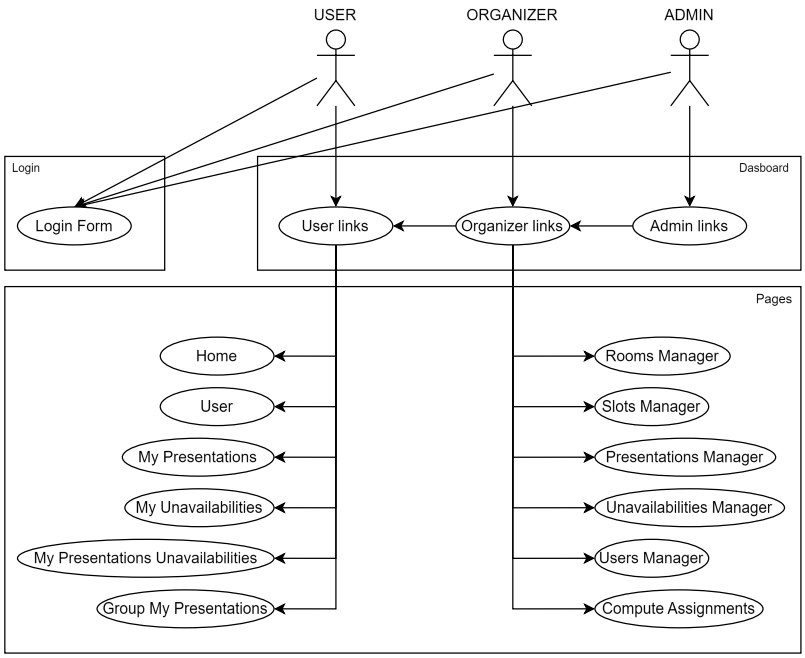

Figura 8.2: Diagrama de navegação da aplicação

8.5.1 Página de Login

A página de login é o ponto de entrada da aplicação. Nesta página, o utilizador é solicitado a inserir o seu endereço de email e a sua password. A interface é simples e intuitiva, com os campos devidamente identificados e um botão de submissão, como é possível confirmar na figura 8.3.

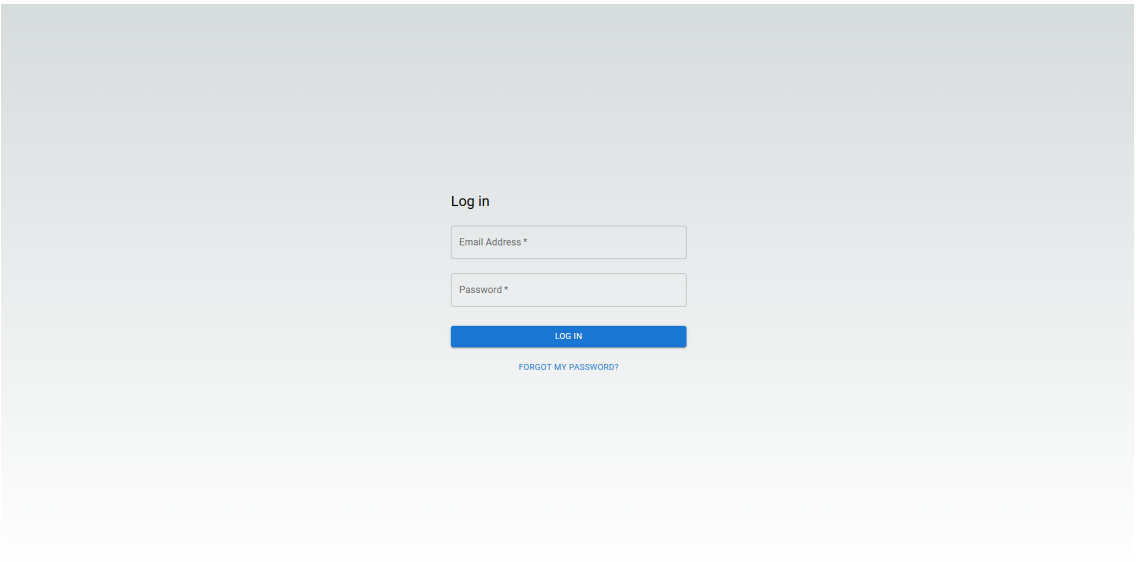

Figura 8.3: Página de Login

Para garantir que o email inserido tem o formato correto, o sistema realiza uma verificação em tempo real que valida o formato do email enquanto o utilizador o digita. Caso o formato seja inválido é exibida a mensagem de erro "*Invalid email format*", informando o utilizador.

Uma vez submetidas as credenciais, é feita a validação no servidor. Se o email ou a password estiverem incorretos, é apresentada a mensagem de erro "*Your email or password is incorrect*". Não é especificado o campo que contém o erro, aumentando assim a segurança da aplicação.

A página de login oferece também a funcionalidade "*Forgot my password*", que permite ao utilizador solicitar a recuperação da sua password. Ao clicar neste botão, o utilizador é direcionado para um campo onde deve introduzir o seu email. Após submeter o pedido, é apresentada a seguinte mensagem: "*If the email you provided is associated with an account in our application, you will receive a password reset email shortly. Please check your inbox.*". Esta abordagem garante que a privacidade do utilizador é preservada, não revelando se o email está ou não associado a uma conta no sistema.

8.5.2 Página *Home*

Na página *Home* todos os utilizadores têm acesso à lista completa de apresentações registadas na aplicação, no entanto, utilizadores com o *role* de *ORGANIZER* ou *ADMIN* têm acesso às funcionalidades de download, como ilustrado nas figuras 8.4 e 8.5.

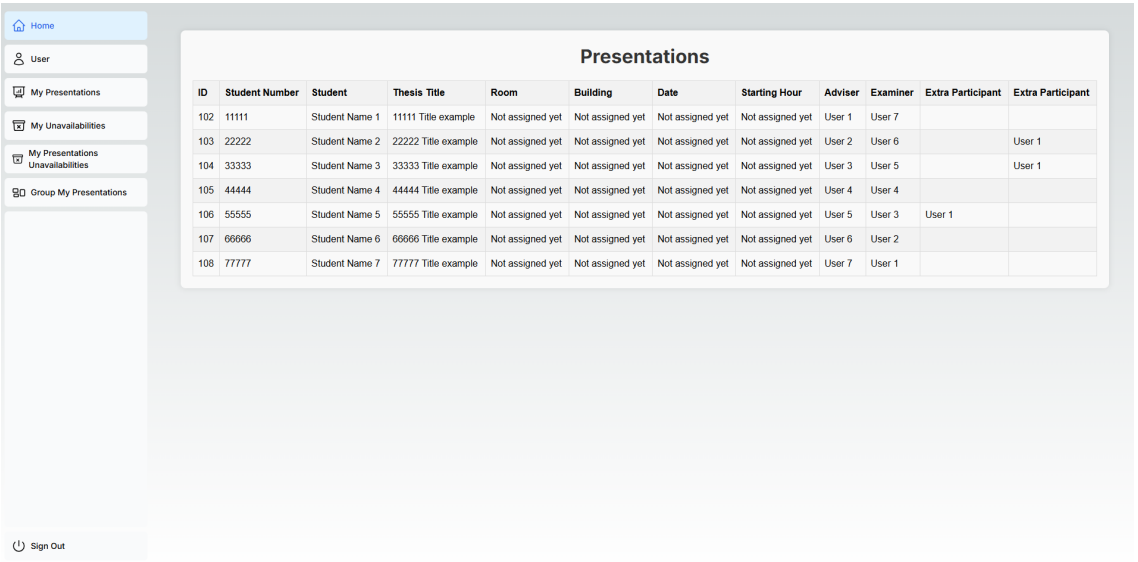

Figura 8.4: Exemplo de página *Home* para um *USER*

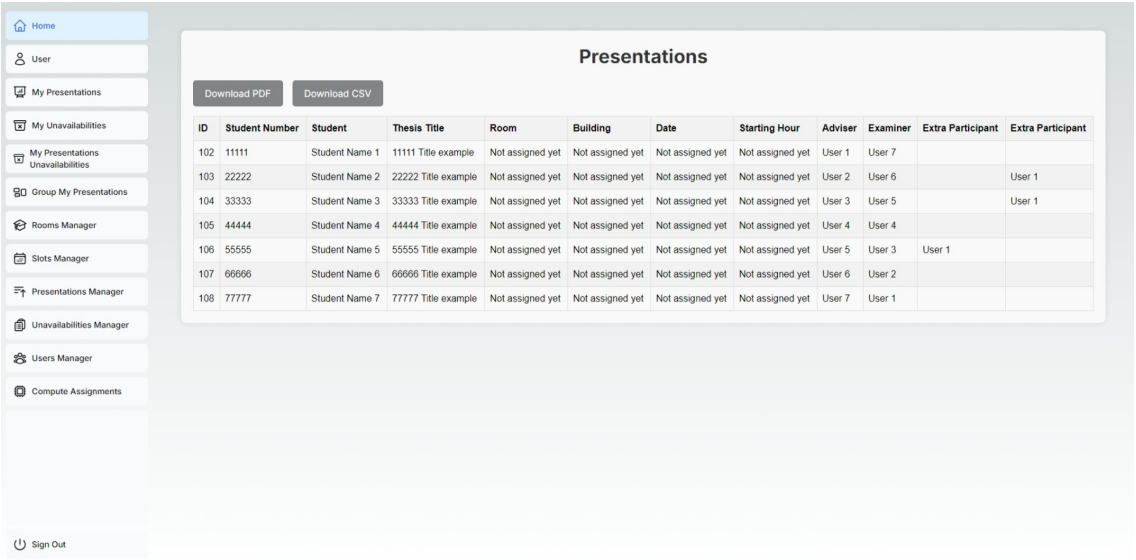

Figura 8.5: Exemplo de página *Home* para um *ORGANIZER* ou *ADMIN*

A opção *Download PDF* foi implementada utilizando as bibliotecas *jsPDF* e *autoTable*, que permitem gerar um ficheiro PDF com a lista de apresentações formatada em modo tabela[1].

A funcionalidade de *Download CSV* permite exportar a lista de apresentações para um ficheiro *CSV*.

Durante os processos de exportação, são ignoradas as colunas correspondentes aos *Extra Participants* (10ª e 11ª colunas da tabela), para simplificar o conteúdo do ficheiro.

8.5.3 Página *User*

A página *User* exibe as informações do utilizador autenticado, como o nome, email e a sua função na aplicação (*role*). Além disso, a página oferece duas opções: alterar a password e apagar a conta, como é possível confirmar na figura 8.6.

Ao selecionar a opção de alterar a password, é apresentado um formulário onde o utilizador deve inserir a sua password atual, a nova password e a confirmação da nova password. O sistema aplica algumas restrições para garantir a segurança deste processo.

- Se a password atual inserida estiver incorreta, a mensagem de erro "*Your current password is not correct.*" é exibida;

[1] https://www.npmjs.com/package/jspdf-autotable

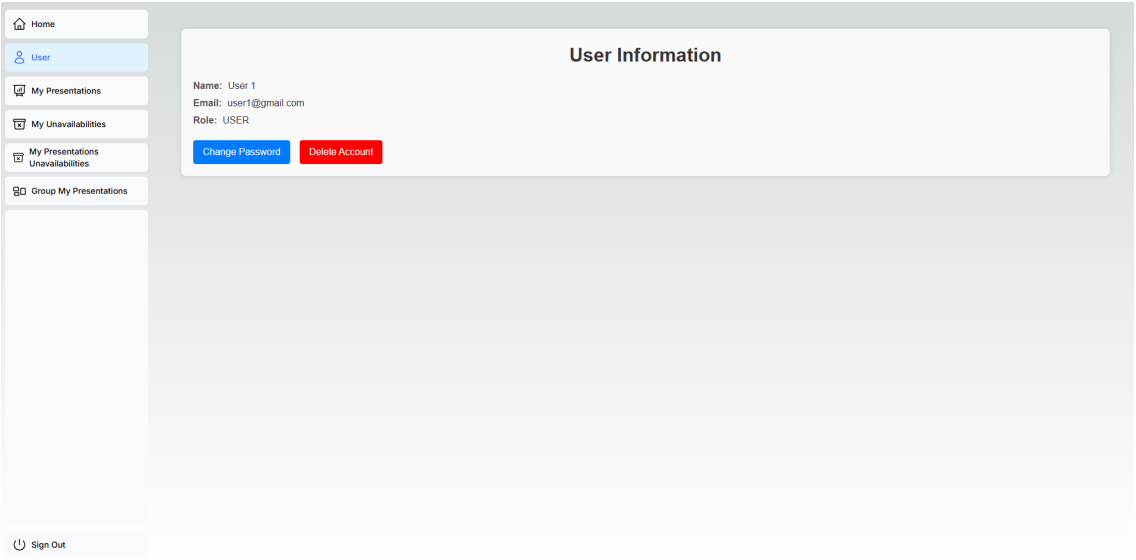

Figura 8.6: Exemplo da página *User*

- Caso a nova password e a confirmação não coincidam, o utilizador será informado através da mensagem "*New password and confirm password do not match.*", garantindo que os dados são inseridos corretamente antes da submissão;

- A nova password deve obedecer a requisitos mínimos de segurança, como ter pelo menos 8 caracteres, incluir uma letra maiúscula, uma letra minúscula e um número. Se estas regras não forem cumpridas, será apresentada a mensagem de erro "*Your new password must contain: At least 8 characters, At least one uppercase letter, At least one lowercase letter, At least one number.*".

Ao tentar apagar a sua conta, é apresentado ao utilizador uma mensagem de confirmação, que visa evitar ações acidentais. Esta mensagem solicita ao utilizador que confirme a sua intenção de prosseguir com a operação, reforçando a segurança deste processo.

8.5.4 Página *My Presentations*

A página *My Presentations* exibe uma tabela que lista todas as apresentações associadas ao utilizador, seja enquanto orientador, vogal ou participante adicional. Esta página permite uma visão clara e organizada de todas as apresentações relacionadas com o utilizador e ainda permite aplicar filtros personalizados, como é possível confirmar na figura 8.7.

As apresentações estão ordenadas consoante a sua importância, isto é, primeiro as apresentações nas quais o utilizador é orientador, depois aquelas em que é vogal e por fim como participante extra.

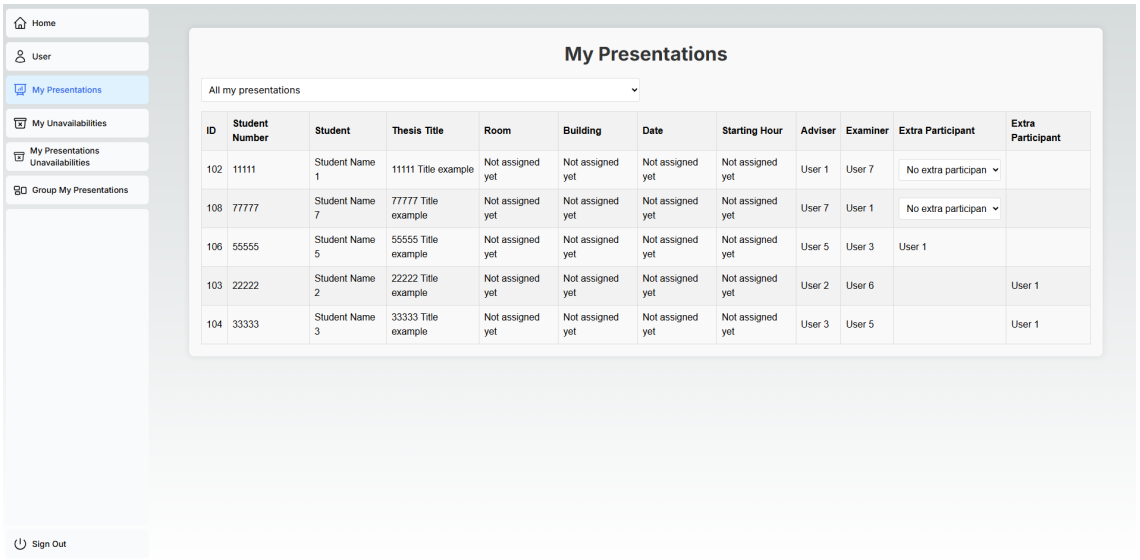

Figura 8.7: Exemplo da página *My Presentations*

Na parte superior da página, existe um *dropdown* que permite ao utilizador filtrar as apresentações exibidas na tabela. Os valores deste *dropdown* são:

- *All my presentations*: exibe todas as apresentações em que o utilizador está envolvido;
- *My presentations as adviser*: exibe apenas as apresentações em que o utilizador atua como orientador;
- *My presentations as examiner*: mostra as apresentações em que o utilizador atua como vogal;
- *My presentations as extra participant*: apresenta as apresentações em que o utilizador está envolvido como participante adicional.

Este sistema de filtros facilita a navegação e a gestão das apresentações, permitindo ao utilizador encontrar rapidamente as informações relevantes.

Para as apresentações em que o utilizador é orientador ou vogal, é possível adicionar um participante extra através de um *dropdown* que contém a lista de todos os utilizadores da aplicação. Caso o utilizador decida remover um participante extra ou não selecionar nenhum, existe ainda a opção "*No extra participant*".

8.5.5 Página *My Unavailabilities*

A página *My Unavailabilities* apresenta uma tabela com os horários e as datas dos *slots* disponíveis para a marcação das apresentações. Esta tabela permite que o utilizador informe as suas disponibilidades e indisponibilidades, de forma visual e intuitiva, para facilitar o agendamento das suas apresentações (ver figura 8.8).

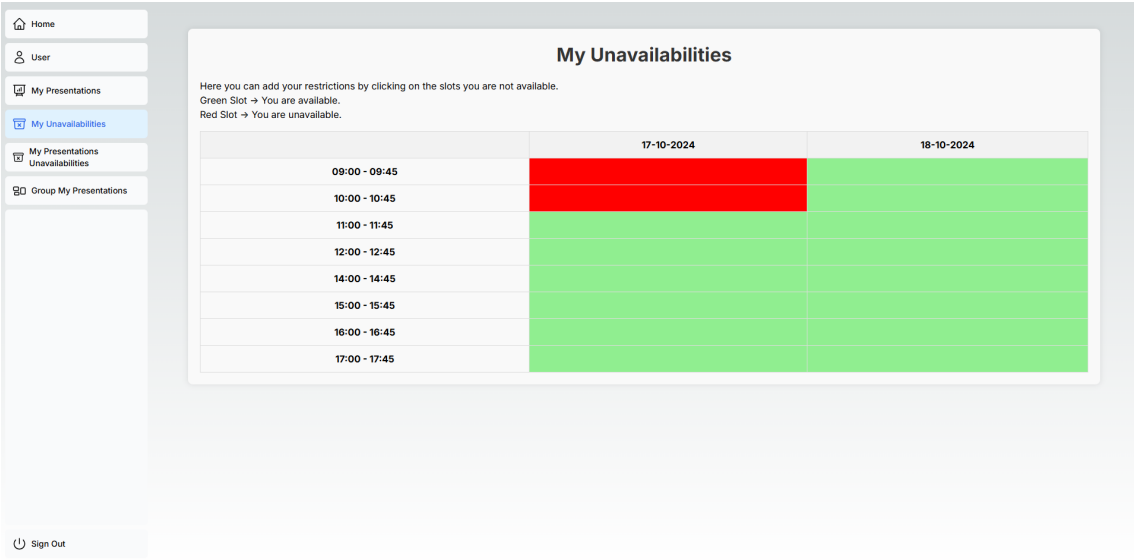

Figura 8.8: Exemplo da página *My Unavailabilities*

Por predefinição, os slots estão definidos como disponíveis para todos os utilizadores.

O utilizador pode clicar diretamente nos *slots* para indicar a sua disponibilidade ou indisponibilidade. Os *slots* marcados a vermelho indicam as indisponibilidades do utilizador, ou seja, os horários em que ele não estará disponível. Os *slots* marcados a verde indicam os horários em que o utilizador se encontra disponível para a marcação de apresentações.

Este sistema proporciona uma gestão visualmente clara das disponibilidades de cada utilizador.

8.5.6 Página *My Presentations Unavailabilities*

Na página *My Presentations Unavailabilities*, um utilizador pode definir as restrições de disponibilidade para as apresentações nas quais desempenha o papel de orientador. O utilizador pode selecionar a apresentação desejada a partir de um *dropdown* que lista as apresentações nas quais ele é orientador, como é possível ver na figura 8.9.

Tal como na página *My Unavailabilities*, é apresentada uma tabela com as datas e horários dos *slots* disponíveis para a marcação de apresentações. O utilizador pode clicar diretamente nos *slots* para indicar a disponibilidade (marcados a verde) ou a indisponibilidade (marcados a vermelho) dos participantes. Por predefinição, todos os *slots* são considerados disponíveis até que o orientador insira as restrições de disponibilidade.

Esta página é particularmente útil para inserir as disponibilidades de indivíduos externos à aplicação, como orientadores de dissertações realizadas em colaboração com

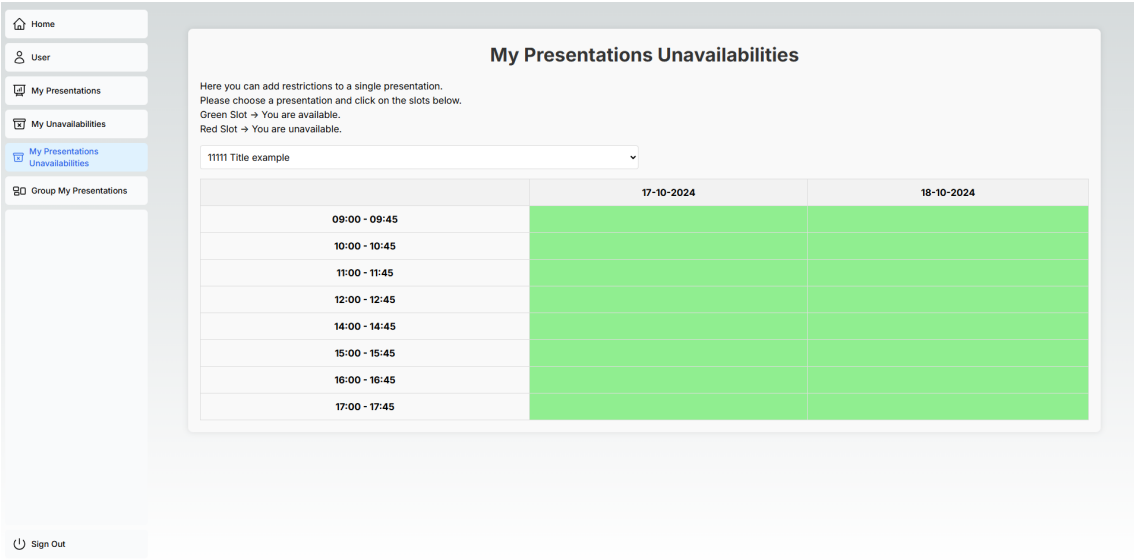

Figura 8.9: Exemplo da página *My Presentations Unavailabilities*

empresas, que não possuem uma conta na aplicação. Desta forma, é possível gerir as restrições de disponibilidade de todos os envolvidos na apresentação.

8.5.7 Página *Group My Presentations*

Na página *Group My Presentations*, os utilizadores têm a opção de selecionar duas apresentações para serem agendadas de forma consecutiva. As apresentações disponíveis para esta seleção são restritas àquelas em que o utilizador atua como orientador ou vogal. Esta página exibe também as apresentações que o utilizador já solicitou para serem agendadas consecutivamente, como ilustrado na figura 8.10.

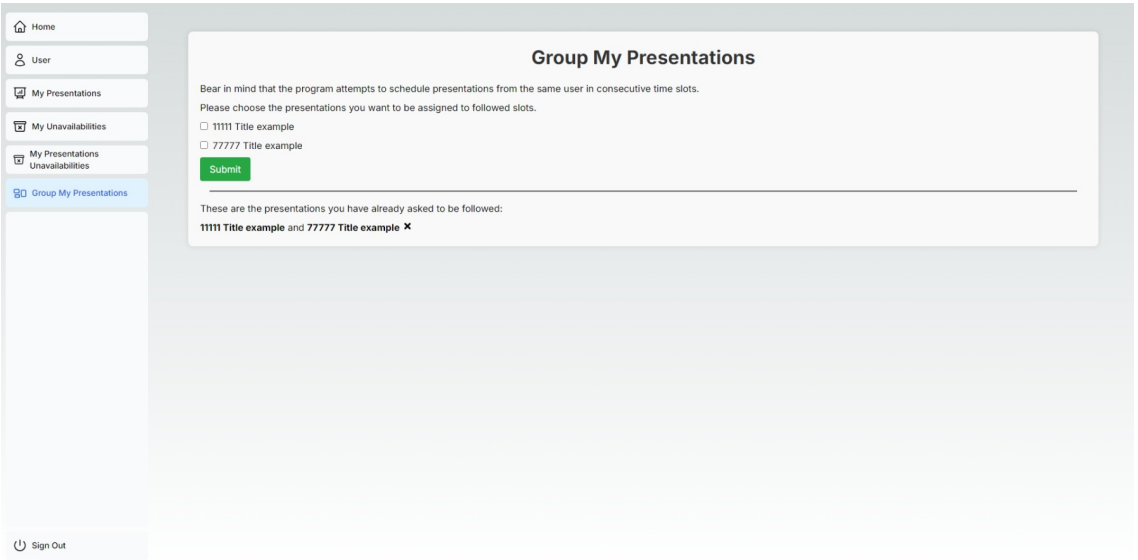

Figura 8.10: Exemplo da página *Group My Presentations*

Se o utilizador selecionar apenas uma ou mais de duas apresentações, é exibida a mensagem de erro "*You can only choose 2 presentations.*", assegurando o bom funcionamento desta funcionalidade.

Adicionalmente, ao tentar adicionar apresentações que já estejam programadas para serem agendadas de forma consecutiva, o sistema apresenta a mensagem de erro "*These presentations are already scheduled to be consecutive.*", prevenindo repetições de informação.

A página também permite que o utilizador cancele um conjunto de apresentações consecutivas previamente definidas. Quando esta ação é solicitada, o sistema apresenta uma mensagem de confirmação para evitar ações acidentais.

Este sistema de gestão de apresentações consecutivas não só facilita a organização das atividades do utilizador, como também garante um feedback imediato nas interações.

8.5.8 Página *Rooms Manager*

Na página *Rooms Manager*, o acesso é restrito a utilizadores com a *role* de *ORGANIZER* ou *ADMIN*. Esta página apresenta uma lista das salas registadas na aplicação e permite que o utilizador adicione ou remova salas, como é possível conferir na figura 8.11.

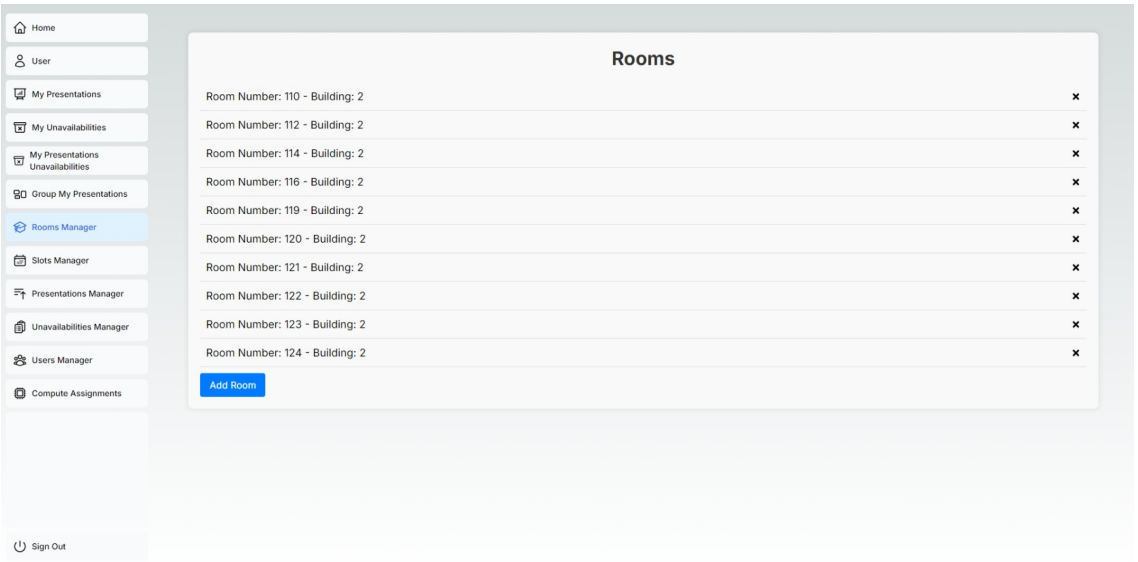

Figura 8.11: Exemplo da página *Rooms Manager*

Para adicionar uma sala, o utilizador deve inserir o número da sala e o edifício a que pertence. Caso já exista uma sala registada com esses mesmos dados, o sistema exibe a mensagem de erro "*The room you are attempting to add has already been added.*", prevenindo duplicações.

Ao remover uma sala, o sistema apresenta uma mensagem de confirmação, garantindo que o utilizador está ciente da ação que está prestes a realizar, evitando assim exclusões acidentais.

8.5.9 Página *Slots Manager*

Na página *Slots Manager*, o acesso é restrito a utilizadores com a *role* de *ORGANIZER* ou *ADMIN*. Nesta página é apresentada uma tabela com a data e hora dos *slots* já registados. O utilizador tem a possibilidade de adicionar ou remover *slots* conforme necessário, como ilustrado na figura 8.12.

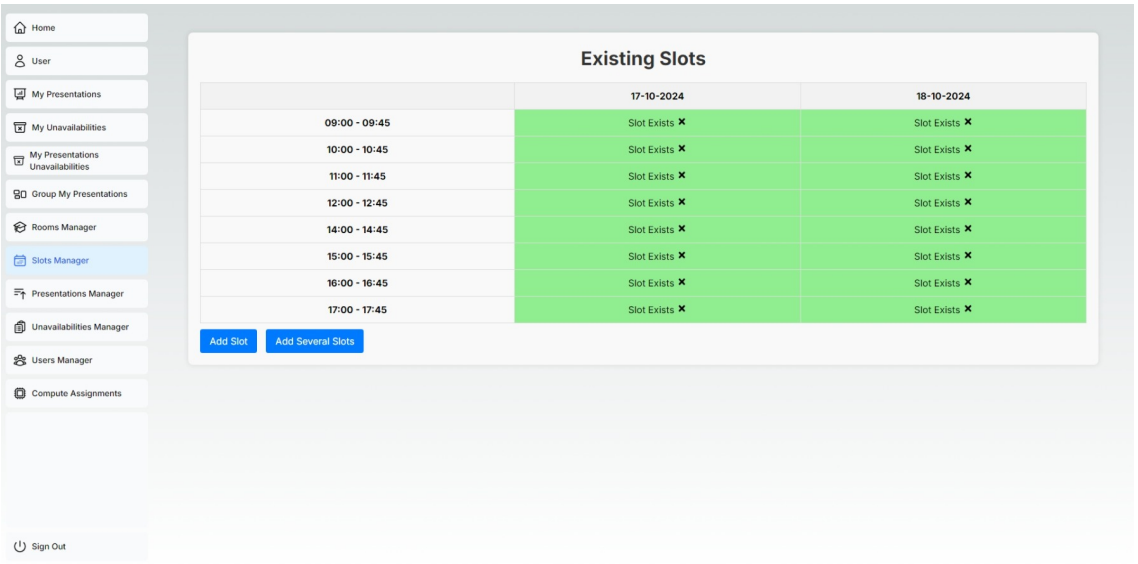

Figura 8.12: Exemplo da página *Slots Manager*

Para adicionar um *slot*, existem duas opções. Na primeira, o utilizador pode adicionar um único *slot*, inserindo a data e a hora inicial. Caso já exista um *slot* registado com a mesma data e hora, o sistema exibe a seguinte mensagem de erro: "*A slot with this date and time already exists. Please choose a different time.*".

Na segunda opção, o utilizador pode adicionar vários *slots* de uma vez. Para isso, é necessário fornecer a data inicial e o número de *slots* desejados. Neste caso, os *slots* são automaticamente criados com as horas predefinidas no sistema: o primeiro *slot* será às 9h, o segundo às 10h, o terceiro às 11h, e assim sucessivamente, até um limite de 8 *slots* diários (com horas predefinidas às 9h, 10h, 11h, 12h, 14h, 15h, 16h e 17h). Se o utilizador selecionar uma data em que já existam *slots* registados (independentemente da hora dos *slots* existentes), a operação será bloqueada, e será exibida a seguinte mensagem de erro: "*There are already slots scheduled for this date. Please select a different date.*".

Em ambos os casos, ao tentar inserir um *slot* com uma data no passado ou no dia presente, o sistema exibirá a mensagem de erro: "*Please insert a date in the future.*".

Ao remover um *slot*, o sistema apresenta uma mensagem de confirmação, garantindo que o utilizador deseja prosseguir com a ação, evitando remoções acidentais.

8.5.10 Página *Presentations Manager*

Na página *Presentations Manager*, o acesso é restrito a utilizadores com a *role* de *ORGANIZER* ou *ADMIN*. A funcionalidade principal desta página é a gestão das apresentações na aplicação, permitindo o carregamento de dados para a criação de várias apresentações e a criação e manipulação manual de apresentações individuais, como ilustra a figura 8.13.

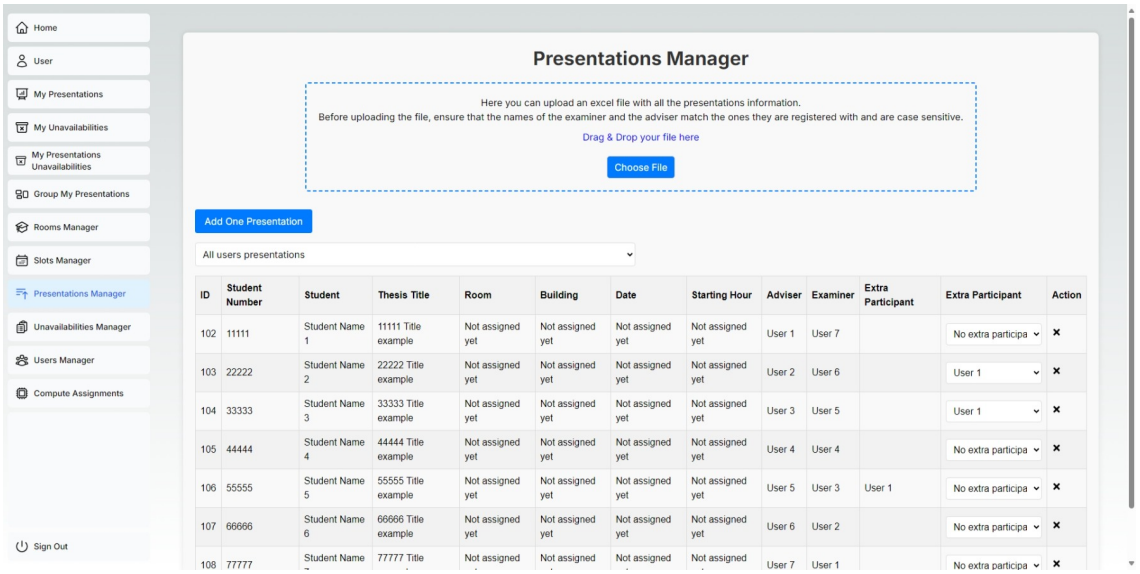

Figura 8.13: Exemplo da página *Presentations Manager*

Na parte superior da página, existe uma área dedicada ao carregamento de um ficheiro Excel contendo as informações necessárias para a criação de várias apresentações de forma automática. Caso o ficheiro não siga a estrutura predefinida, o sistema exibe a mensagem de erro "*Make sure your Excel file has a similar structure to this one:*", seguida de uma imagem exemplificativa da estrutura esperada para o bom funcionamento desta funcionalidade.

Além do carregamento de um ficheiro Excel, é possível adicionar uma apresentação manualmente. Para isso, o utilizador deve preencher os seguintes campos: número do estudante, nome do estudante, título da dissertação e selecionar o orientador e o vogal a partir de *dropdowns* com os utilizadores disponíveis na aplicação.

Abaixo da secção de criação de apresentações, é apresentada uma lista com todas as apresentações registadas na aplicação. O sistema permite a aplicação de filtros para

facilitar a visualização desta informação. Existe um *dropdown* ordenado alfabeticamente com os nomes de todos os utilizadores registados, onde é possível selecionar um utilizador específico e visualizar todas as apresentações em que este está envolvido. As apresentações são ordenadas da seguinte forma: primeiro como orientador, depois como vogal e, por fim, como participante extra. Existe ainda a opção de visualizar as apresentações de todos os utilizadores em simultâneo.

A página também permite a adição de um participante extra a cada apresentação. Para isso, existe um *dropdown* com os nomes de todos os utilizadores da aplicação ordenados alfabeticamente, onde o é possível selecionar o participante adicional ou escolher a opção "*No extra participant*"para não adicionar ninguém extra.

Por fim, é possível eliminar apresentações da aplicação. Ao tentar apagar uma apresentação, o sistema apresenta uma mensagem de confirmação, garantindo que a ação seja intencional e evitando a remoção acidental.

8.5.11 Página *Unavailabilities Manager*

Na página *Unavailabilities Manager*, o acesso é restrito a utilizadores com a *role* de *ORGANIZER* ou *ADMIN*. Esta página tem um caráter meramente informativo, permitindo a visualização das indisponibilidades dos utilizadores registados na aplicação, como ilustra a figura 8.14.

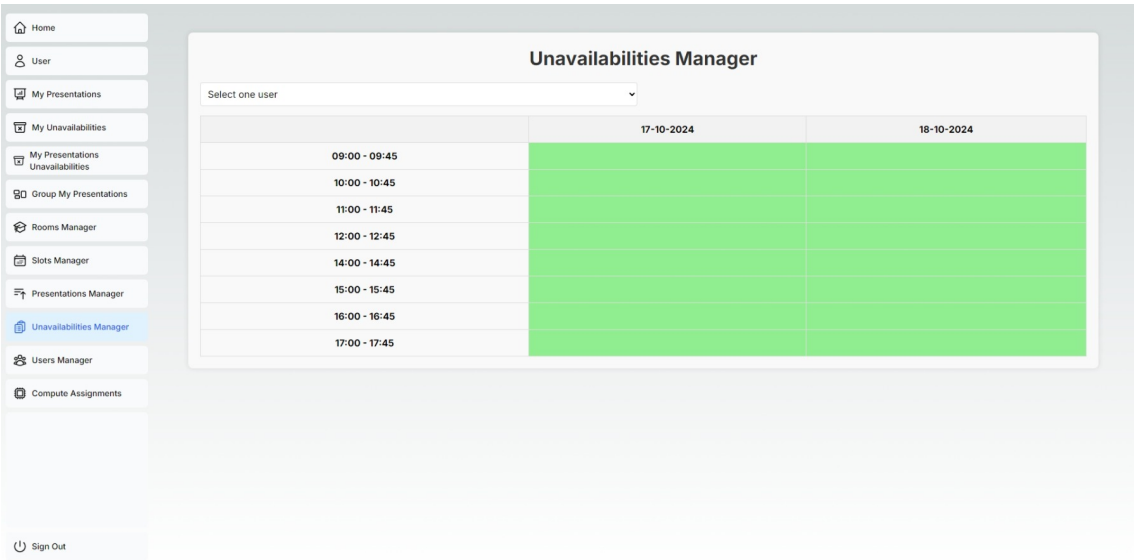

Figura 8.14: Exemplo da página *Unavailabilities Manager*

A interface apresenta um *dropdown* com os nomes de todos os utilizadores, organizados por ordem alfabética. Ao selecionar um utilizador, é exibida uma tabela que contém

todos os *slots* registados na aplicação, com suas respetivas datas e horários. Nesta tabela, as indisponibilidades do utilizador selecionado são destacadas, facilitando a consulta e gestão do agendamento de apresentações.

8.5.12 Página *Users Manager*

Na página *Users Manager*, o acesso é restrito a utilizadores com a *role* de *ORGANIZER* ou *ADMIN*. Esta página permite consultar todos os utilizadores registados na aplicação, conforme ilustrado nas figuras 8.15 e 8.16.

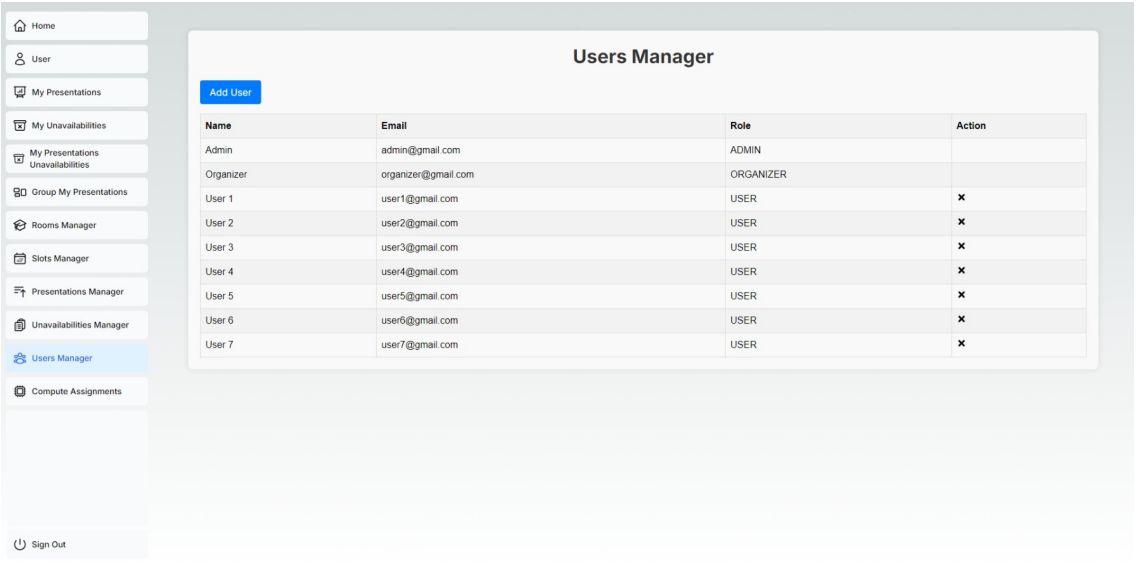

Figura 8.15: Exemplo da página *Users Manager* vista por um organizer

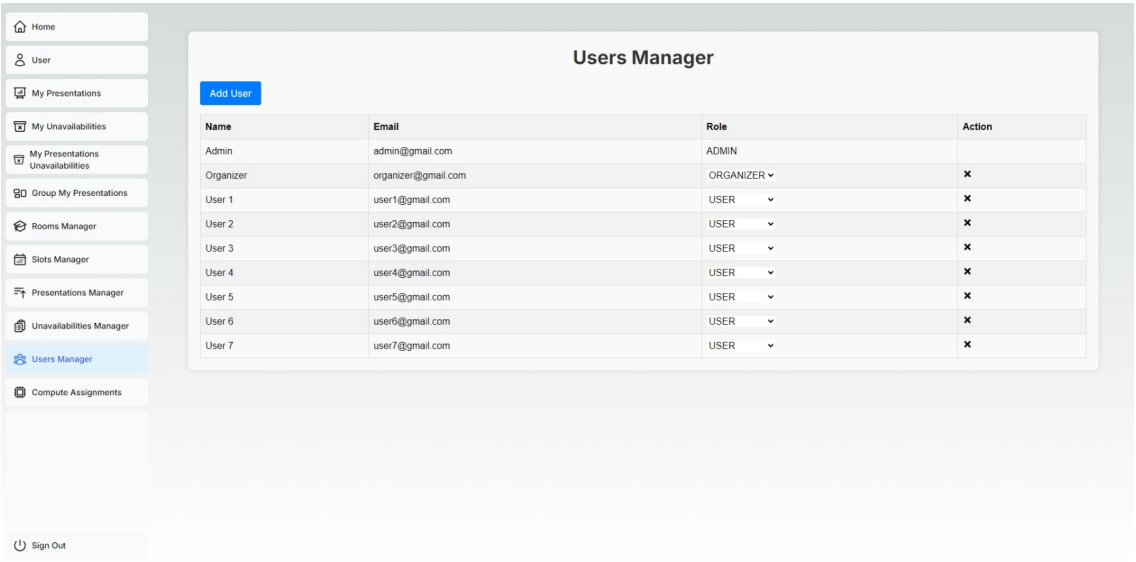

Figura 8.16: Exemplo da página *Users Manager* vista por um admin

Além de consultar os utilizadores, a página permite a adição de novos utilizadores à aplicação, solicitando o nome e o email do novo utilizador. Caso o nome não seja preenchido ou o email não seja válido, é exibida a mensagem de erro *"Please enter valid name and email."*. Após a adição de um novo utilizador, este recebe um email com uma senha temporária para aceder à aplicação.

Um *ORGANIZER* pode apagar as contas de utilizadores do tipo *USER*, enquanto um *ADMIN* tem permissão para apagar contas de utilizadores do tipo *USER* e *ORGANIZER*. Além disso, um *ADMIN* tem ainda a capacidade de alterar as *roles* dos utilizadores, como ilustrado na figura 8.16.

8.5.13 Página *Compute Assignments*

Na página *Compute Assignments*, o acesso é restrito a utilizadores com a *role* de *ORGANIZER* ou *ADMIN*. Esta página é utilizada exclusivamente para iniciar o processo de cálculo dos horários e da atribuição de salas para as apresentações, conforme ilustrado na figura 8.17.

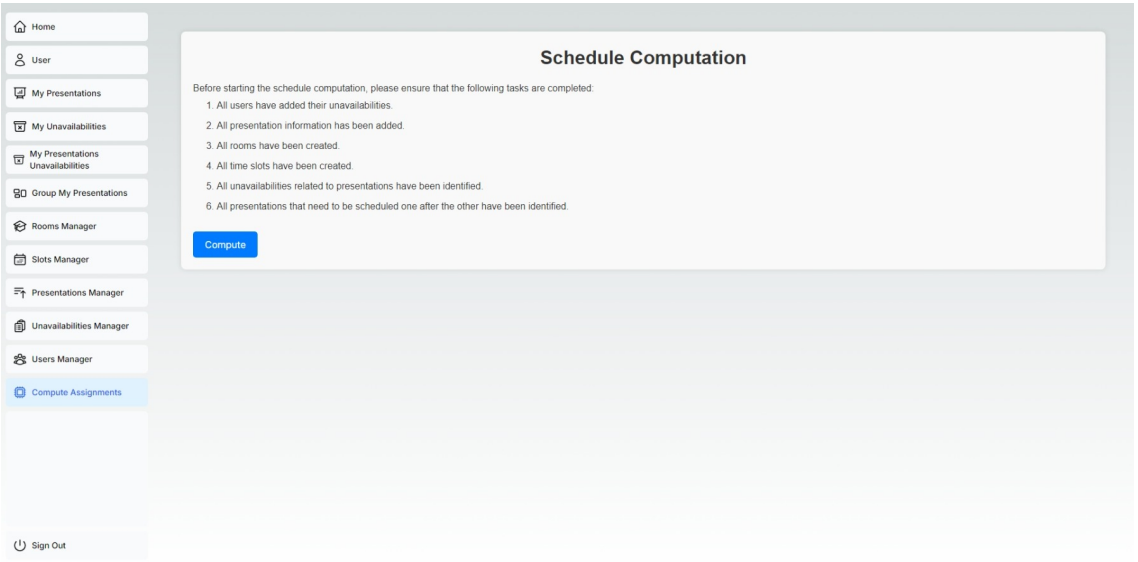

Figura 8.17: Exemplo da página *Compute Assignments*

Ao pressionar o botão *Compute*, o sistema inicia o processo de computação, exibindo a mensagem de feedback: *"The schedule creation has started. This operation should take from 10 to 20 minutes."*, indicando que a operação está em andamento e o tempo necessário para a sua conclusão.

8.6 Conclusão

Neste capítulo, foi descrito o Cliente da aplicação, abordando as tecnologias empregues no seu desenvolvimento, a arquitetura subjacente, bem como a comunicação eficiente entre

o Cliente e o Servidor. Explorámos também a navegação e as diversas funcionalidades oferecidas ao utilizador, que asseguram uma interação intuitiva e eficaz.

No entanto, para garantir que a aplicação responde de forma correta às necessidades dos utilizadores e que todas as funcionalidades operam como esperado, é essencial validar o seu comportamento. No capítulo seguinte, serão apresentados os testes realizados à aplicação, que visam assegurar o seu bom funcionamento e a adequação às especificações definidas.

9

Testes

Os testes realizados na aplicação foram essenciais para validar a sua robustez, usabilidade e funcionalidade. Estes testes foram divididos em duas fases principais: os testes iniciais, realizados com utilizadores externos e não especializados, e os testes de usabilidade, realizados posteriormente com indivíduos da área, com o objetivo de recolher feedback mais direcionado e técnico. Ambas as fases forneceram informações valiosas que permitiram confirmar a eficácia da aplicação e identificar potenciais melhorias.

9.1 Testes Iniciais

Os testes iniciais foram realizados por indivíduos externos ao projeto, incluindo amigos, colegas de curso e familiares, que não tinham qualquer familiaridade com a aplicação ou com a sua finalidade. O principal objetivo desta fase foi avaliar a usabilidade da interface e a acessibilidade das funcionalidades, garantindo que mesmo utilizadores sem qualquer conhecimento prévio conseguiam realizar as tarefas propostas de forma intuitiva e eficiente.

Durante estes testes, os utilizadores tiveram acesso às credenciais dos três tipos de perfis disponíveis na aplicação (*USER*, *ORGANIZER* e *ADMIN*) e foram-lhes atribuídas diversas tarefas e operações a serem executadas. Verificou-se que todos os participantes conseguiram completar as tarefas de forma rápida e eficiente, o que nos permite concluir que a interface é intuitiva e de fácil utilização, mesmo para utilizadores sem conhecimento prévio da aplicação ou da sua função.

Os testes que iremos apresentar, incidiriam na interação dos utilizadores com a aplicação, via Frontend. Estes testes foram divididos em quatro grandes categorias:

- **Login** — Validação das diversas tentativas de autenticação que o utilizador poderá realizar na aplicação;
- ***User*** — Verificação das funcionalidades a que um utilizador do tipo *User* tem acesso;

- ***Organizer*** — Validação das funcionalidades a que um organizador tem acesso, incluindo a comparação com as funções disponíveis para um utilizador do tipo *User*;
- ***Admin*** — Confirmação das operações a que um administrador tem acesso, com especial destaque para as diferenças em relação ao perfil de *Organizer*.

É relevante realçar que todos os casos de teste resultaram em sucesso, demonstrando a robustez da aplicação no cumprimento das suas funcionalidades principais e na usabilidade proporcionada aos diferentes perfis de utilizador.

9.1.1 Login

Grupo	Função a testar	Resultado esperado
Login	Não colocar user	Informação: "Your email or password is incorrect"
	Colocar user incorreto sem password	Informação: "Your email or password is incorrect"
	Colocar user incorreto com password correta	Informação: "Invalid email format"
	Colocar user incorreto com password incorreta	Informação: "Invalid email format"
	Colocar user correto sem password	Informação: "Your email or password is incorrect"
	Colocar user correto com password incorreta	Informação: "Your email or password is incorrect"
	Colocar user correto com password correta	É possível aceder
	Selecionar "Forget my password"	Apresenta caixa para escrever o email
	Preencher o email e selecionar "Send"	Informação: "If the email you provided is associated with an account in our application, you will receive a password reset email shortly. Please check your inbox.”.
	Selecionar Cancel	Volta à página de login
Signout	Selecionar a opção	Sai da aplicação

9.1.2 User

Grupo	Função a testar	Resultado esperado

Login	Acesso com utilizador tipo "Utilizador"	É possível aceder e tem os menus disponiveis: Home, User, My Presentations, My Unavailabilities, My Presentation Unavailabilities, Group my Presentations
Home	Selecionar a opção	Acede à lista de todas as apresentações
User	Selecionar a opção	Acede às informações do utilizador
	Selecionar "change password"	Acede a menu com possibilidade de alterar password
	Colocar password incorreta e nova password	Mensagem de erro: "your current password is incorrect"
	Colocar password correta e nova password sem regras de password	Mensagem de erro: "Your new password must contain: At least 8 characters, At least one uppercase letter, At least one lowercase letter, At least one number."
	Colocar password correta e nova password diferente da confirmação	Mensagem de erro: "New password and confirm password do not match"
	Colocar password correta e nova password	Password alterada
	Sair e voltar a entrar	A nova password funciona e a antiga não
	Selecionar "delete account"	Aparece mensagem de confirmação
	Selecionando "No"	Volta ao menu do utilizador
	Selecionando "Yes"	Volta ao login
	Tentar entrar com user e password	Mensagem de erro: "email ou password is incorrect"
My Presentations	Selecionar a opção	Acede à lista das apresentações do utilizador
	Selecionar as apresentações como Examiner	Apresenta apenas as apresentações como Examiner
	Selecionar as apresentações como Adviser	Apresenta apenas as apresentações como Adviser
	Selecionar as apresentações como extra participant	Apresenta apenas as apresentações como extra participant
My Unavailabilities	Selecionar a opção	Acede a lista das não disponibilidades

	Selecionar antes de definir Slots	Não apresenta opções
	Selecionar SLOT a verde	Passa a vermelho
	Selecionar SLOT a branco	Sem impacto
	Selecionar SLOT a vermelho	Passa a verde
My Presentations Unavailabilities	Selecionar a opção	Acede a lista de apresentações do utilizador para seleção
	Selecionar apresentação em causa	Possibilita escolha de indisponibilidade dentro dos slots existentes
Group My Presentations	Selecionar a opção	Acede a lista de apresentações do utilizador para seleção
	Selecionar 1 apresentação	Mensagem "You have to choose 2 presentations"
	Selecionar 3 ou mais apresentações	Mensagem "You have to choose 2 presentations"
	Selecionar 2 apresentações	As apresentações ficam agrupadas

9.1.3 Organizer

Grupo	Função a testar	Resultado esperado
Login	Acesso com utilizador tipo "Organizador"	É possível aceder e tem todos os menus disponiveis
Home	Selecionar a opção	Acede à lista de todas as apresentações
	Pressionar "Download PDF"	Faz download do conteúdo da lista de apresentações em PDF
	Pressionar "Download CSV"	Faz download do conteúdo da lista de apresentações em CSV
User	Selecionar a opção	Acede às informações do utilizador
	Selecionar "change password"	Acede a menu com possibilidade de alterar password
	Colocar password incorreta e nova password	Mensagem de erro: "your current password is incorrect"
	Colocar password correta e nova password sem regras de password	Mensagem de erro: "Your new password must contain: At least 8 characters, At least one uppercase letter, At least one lowercase letter, At least one number."

	Colocar password correta e nova password diferente da confirmação	Mensagem de erro: "New password and confirm password do not match"
	Colocar password correta e nova password	Password alterada
	Sair e voltar a entrar	A nova password funciona e a antiga não
	Selecionar "delete account"	Aparece mensagem de confirmação
	Selecionando "No"	Volta ao menu do utilizador
	Selecionando "Yes"	Volta ao login
	Tentar entrar com user e password	Mensagem de erro: "email ou password is incorrect"
My Presentations	Selecionar a opção	Acede à lista das apresentações do utilizador
	Selecionar as apresentações como Examiner	Apresenta apenas as apresentações como Examiner
	Selecionar as apresentações como Adviser	Apresenta apenas as apresentações como Adviser
	Selecionar as apresentações como extra participant	Apresenta apenas as apresentações como extra participant
My Unavailabilities	Selecionar a opção	Acede a lista das não disponibilidades
	Selecionar antes de definir Slots	Não apresenta opções
	Selecionar SLOT a verde	Passa a vermelho
	Selecionar SLOT a branco	Sem impacto
	Selecionar SLOT a vermelho	Passa a verde
My Presentations Unavailabilities	Selecionar a opção	Acede a lista de apresentações do utilizador para seleção
	Selecionar apresentação em causa	Possibilita escolha de indisponibilidade dentro dos slots existentes
Group My Presentations	Selecionar a opção	Acede a lista de apresentações do utilizador para seleção
	Selecionar 1 apresentação	Mensagem "You have to choose 2 presentations"
	Selecionar 3 ou mais apresentações	Mensagem "You have to choose 2 presentations"
	Selecionar 2 apresentações	As apresentações ficam agrupadas
Rooms Manager	Selecionar a opção	Acede à lista de Rooms

	Selecionar "Add Room"	Opção para selecionar "Room Number e "Building"
	Não selecionar nada e selecionar SUBMIT	Mensagem "Please enter both room number and building"
	Selecionar apenas "Room Number"e SUBMIT	Mensagem "Please enter both room number and building"
	Selecionar apenas "Building"e SUBMIT	Mensagem "Please enter both room number and building"
	Selecionar Cancel	Volta ao menu inicial dos ROOMS
	Selecionar "Room Number"e "Building"e SUBMIT	Fica criada Room com número e edifício escolhido
	Selecionar mesmo "Room Number"e outro "Building"e SUBMIT	Fica criada Room com número e edifício escolhido
	Selecionar outro "Room Number"e mesmo "Building"e SUBMIT	Fica criada Room com número e edifício escolhido
	Selecionar mesmo "Room Number"e mesmo "Building"e SUBMIT	Mensagem "The room you are attempting to add has already been added"
	Selecionar um room/building e pressionar X	Mensagem "Are you sure you want to delete?"
	Press YES	Room apagada
	Press NO	A room não é apagada
Slots Manager	Selecionar a opção	Acede ao menu de"SLOTS"
	selecionar "add Slot"	Apresenta hipótese de adicionar um SLOT
	Selecionar data e não selecionar hora e minuto	Slot fica criado às 00:00
	Selecionar data e hora e não selecionar minuto	Slot fica criado aos 00 minutos da hora indicada
	Selecionar data no passado	Mensagem "Please insert a date in the future"
	Selecionar data e hora e minuto	Slot fica criado no dia e hora indicado
	Selecionar mesma data e outra hora e minuto	Slot fica criado no dia e hora indicado

	Selecionar data e hora e minuto já existentes	Mensagem "There are already slots scheduled for this date. Please select a different date"
	Selecionar CANCEL	Volta à página inicial do menu "Slots"
	Selecionar "Add Several Slots"	Apresenta hipótese de adicionar um SLOT
	Não Selecionar data e usar número pré-definido	Mensagem "Please select date"
	Selecionar data já existente	Mensagem "A slot with this date and time already exists. Please choose a different time"
	Selecionar nova data	Slot fica criado
	Selecionar nova data com opção 8	Ficam criados 8 slots
Presentations Manager	Selecionar a opção	Acede ao menu de Upload
	Selecionar "Add one presentation"	Acede ao menu de criação
	Não preencher os campos todos	Mensagem "Please enter all information"
	Preencher os campos todos corretamente e selecionar submit	A apresentação fica criada
	Arrastar ficheiro incorreto	Mensagem: "Make sure our excel file has similar structure to this one: (*imagem da estrutura da tabela a ser inserida*)"
	Arrastar ficheiro correto	As várias apresentações ficam criadas
	Arrastar o mesmo ficheiro	Mensagem: "Make sure our excel file has similar structure to this one: (*imagem da estrutura da tabela a ser inserida*)"
	Selecionar as apresentações de um user	Aparecem apenas as apresentações desse user, independentemente do seu papel
Unavailabilities Manager	Selecionar a opção	Acede a lista das não disponibilidades

	Selecionar um utilizador	Apresenta as restrições que o utilizador definiu
Users Manager	Selecionar a opção	Acede ao menu de Users
	Selecionar "Add User"	Abre opção de adicionar utilizador
	Preencher Nome e SUBMIT	Mensagem "Please enter valid name and email"
	Preencher Mail e SUBMIT	Mensagem "Please enter valid name and email"
	Preencher mail já existente	Mensagem "Please enter valid name and email"
	Preencher Nome e Mail	Utilizador criado e mail enviado com informação de password
	Selecionar um room/building e pressionar X	Mensagem "Are you sure you want to delete?"
	Press YES	Room apagada
	Press NO	A room não é apagada
	Selecionar um room/building e pressionar X	Mensagem "Are you sure you want to delete?"
	Press YES	Room apagada
	Press NO	A room não é apagada
Compute assignments	Selecionar a opção	Acede ao menu de Schedule Computation
	Selecionar Compute	Após conclusão, fica atribuido o calendário

9.1.4 Admin

Grupo	Função a testar	Resultado esperado
Login	Acesso com utilizador tipo "Administrador"	É possível aceder e tem todos os menus disponíveis
Home	Selecionar a opção	Acede à lista de todas as apresentações
	Pressionar "Download PDF"	Faz download do conteúdo da lista de apresentações em PDF
	Pressionar "Download CSV"	Faz download do conteúdo da lista de apresentações em CSV
User	Selecionar a opção	Acede às informações do utilizador
	Selecionar "change password"	Acede a menu com possibilidade de alterar password

	Colocar password incorreta e nova password	Mensagem de erro: "your current password is incorrect"
	Colocar password correta e nova password sem regras de password	Mensagem de erro: "Your new password must contain: At least 8 characters, At least one uppercase letter, At least one lowercase letter, At least one number."
	Colocar password correta e nova password diferente da confirmação	Mensagem de erro: "New password and confirm password do not match"
	Colocar password correta e nova password	Password alterada
	Sair e voltar a entrar	A nova password funciona e a antiga não
	Selecionar "delete account"	Aparece mensagem de confirmação
	Selecionando "No"	Volta ao menu do utilizador
	Selecionando "Yes"	Volta ao login
	Tentar entrar com user e password	Mensagem de erro: "email ou password is incorrect"
My Presentations	Selecionar a opção	Acede à lista das apresentações do utilizador
	Selecionar as apresentações como Examiner	Apresenta apenas as apresentações como Examiner
	Selecionar as apresentações como Adviser	Apresenta apenas as apresentações como Adviser
	Selecionar as apresentações como extra participant	Apresenta apenas as apresentações como extra participant
My Unavailabilities	Selecionar a opção	Acede a lista das não disponibilidades
	Selecionar antes de definir Slots	Não apresenta opções
	Selecionar SLOT a verde	Passa a vermelho
	Selecionar SLOT a branco	Sem impacto
	Selecionar SLOT a vermelho	Passa a verde
My Presentations Unavailabilities	Selecionar a opção	Acede a lista de apresentações do utilizador para seleção
	Selecionar apresentação em causa	Possibilita escolha de indisponibilidade dentro dos slots existentes

Group My Presentations	Selecionar a opção	Acede a lista de apresentações do utilizador para seleção
	Selecionar 1 apresentação	Mensagem "You have to choose 2 presentations"
	Selecionar 3 ou mais apresentações	Mensagem "You have to choose 2 presentations"
	Selecionar 2 apresentações	As apresentações ficam agrupadas
Rooms Manager	Selecionar a opção	Acede à lista de Rooms
	Selecionar "Add Room"	Opção para selecionar "Room Number e "Building"
	Não selecionar nada e selecionar SUBMIT	Mensagem "Please enter both room number and building"
	Selecionar apenas "Room Number"e SUBMIT	Mensagem "Please enter both room number and building"
	Selecionar apenas "Building"e SUBMIT	Mensagem "Please enter both room number and building"
	Selecionar Cancel	Volta ao menu inicial dos ROOMS
	Selecionar "Room Number"e "Building"e SUBMIT	Fica criada Room com número e edifício escolhido
	Selecionar mesmo "Room Number"e outro "Building"e SUBMIT	Fica criada Room com número e edifício escolhido
	Selecionar outro "Room Number"e mesmo "Building"e SUBMIT	Fica criada Room com número e edifício escolhido
	Selecionar mesmo "Room Number"e mesmo "Building"e SUBMIT	Mensagem "The room you are attempting to add has already been added"
	Selecionar um room/building e pressionar X	Mensagem "Are you sure you want to delete?"
	Press YES	Room apagada
	Press NO	A room não é apagada
Slots Manager	Selecionar a opção	Acede ao menu de"SLOTS"
	selecionar "add Slot"	Apresenta hipótese de adicionar um SLOT
	Selecionar data e não selecionar hora e minuto	Slot fica criado às 00:00
	Selecionar data e hora e não selecionar minuto	Slot fica criado aos 00 minutos da hora indicada

	Selecionar data no passado	Mensagem "Please insert a date in the future"
	Selecionar data e hora e minuto	Slot fica criado no dia e hora indicado
	Selecionar mesma data e outra hora e minuto	Slot fica criado no dia e hora indicado
	Selecionar data e hora e minuto já existentes	Mensagem "There are already slots scheduled for this date. Please select a different date"
	Selecionar CANCEL	Volta à página inicial do menu "Slots"
	Selecionar "Add Several Slots"	Apresenta hipótese de adicionar um SLOT
	Não Selecionar data e usar número pré-definido	Mensagem "Please select date"
	Selecionar data já existente	Mensagem "A slot with this date and time already exists. Please choose a different time"
	Selecionar nova data	Slot fica criado
	Selecionar nova data com opção 8	Ficam criados 8 slots
Presentations Manager	Selecionar a opção	Acede ao menu de Upload
	Selecionar "Add one presentation"	Acede ao menu de criação
	Não preencher os campos todos	Mensagem "Please enter all information"
	Preencher os campos todos corretamente e selecionar submit	A apresentação fica criada
	Arrastar ficheiro incorreto	Mensagem: "Make sure our excel file has similar structure to this one: (*imagem da estrutura da tabela a ser inserida*)"
	Arrastar ficheiro correto	As várias apresentações ficam criadas

	Arrastar o mesmo ficheiro	Mensagem: "Make sure our excel file has similar structure to this one: (*imagem da estrutura da tabela a ser inserida*)"
	Selecionar as apresentações de um user	Aparecem apenas as apresentações desse user, independentemente do seu papel
Unavailabilities Manager	Selecionar a opção	Acede a lista das não disponibilidades
	Selecionar um utilizador	Apresenta as restrições que o utilizador definiu
Users Manager	Selecionar a opção	Acede ao menu de Users
	Alterar Role de user	O Role fica alterado
	Selecionar "Add User"	Abre opção de adicionar utilizador
	Preencher Nome e SUBMIT	Mensagem "Please enter valid name and email"
	Preencher Mail e SUBMIT	Mensagem "Please enter valid name and email"
	Preencher mail já existente	There is na error
	Preencher nome já existente	There is na error
	Preencher Nome e Mail	Utilizador criado e mail enviado com informação de password
	Selecionar um room/building e pressionar X	Mensagem "Are you sure you want to delete?"
	Press YES	Room apagada
	Press NO	A room não é apagada
Compute assignments	Selecionar a opção	Acede ao menu de Schedule Computation
	Selecionar Compute	Após conclusão, fica atribuido o calendário

9.2 Testes de Usabilidade

Após a conclusão dos testes iniciais e a correção dos *bugs* encontrados, foram realizados testes de usabilidade com utilizadores especializados na área. O objetivo destes testes foi avaliar a experiência de utilização da aplicação nos diversos cenários e obter feedback detalhado sobre a interface e funcionalidades oferecidas.

Aos participantes destes testes foi apresentado um documento com um *briefing* detalhado sobre o funcionamento dos testes, as credênciais de acesso à aplicação e as tarefas a realizar. Este documento está presente no anexo III.

Estes testes englobaram ainda um formulário onde os participantes avaliaram as tarefas e deixaram sugestões. Apresentamos os resultados obtidos divididos pelo tipo de utilizador. Por fim, há um ponto dedicado ao feedback geral da aplicação.

Esta estrutura permitiu uma análise detalhada do desempenho da aplicação e da perceção de usabilidade para cada perfil de utilizador, assim como uma visão consolidada das sugestões fornecidas pelos participantes.

9.2.1 Cenário *USER*

Neste ponto apresentamos os resultados das operações realizadas com as credenciais de um utilizador do tipo *USER*:

1. Adicionar uma restrição

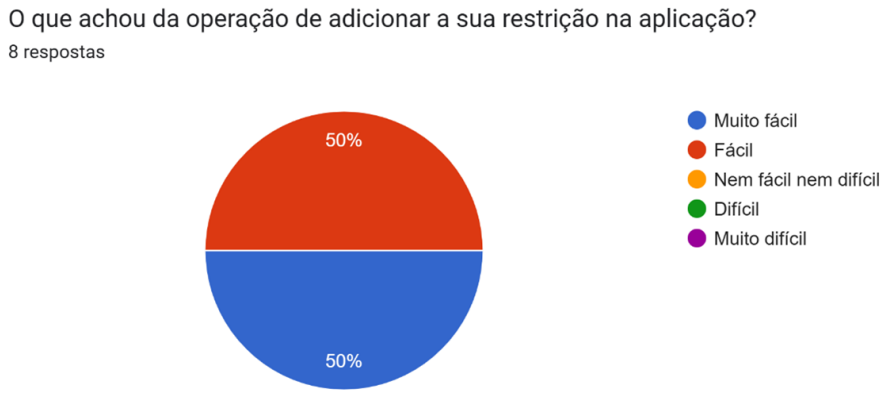

Figura 9.1: Gráfico do resultado de Adicionar uma restrição

Sugestões:

- Não se percebe à partida qual a diferença entre "My unavailabilities"e "My presentation unavailabilities".
- Na minha opinião este tipo de tarefa necessita de um botao de confirmação, para termos feedback relativamente à realização da operação. Quando alteramos a cor (disponibilidade) necessitamos de uma confirmação de que as alterações estão guardadas.

- Intervalos com horas completas em vez de terminar aos 45min (exemplo 14:00 - 15:00); daltónicos podem ter problemas com as cores - talvez acrescentar um X quando se coloca a vermelho.
- As slots funcionam como "toggle buttons"tornando a seleção fácil e rápida.
- Seria interessante poder seleccionar vários slots por arrasto. em vez de ter de seleccionar todos.

2. Consultar as apresentações nas quais é orientador

Figura 9.2: Gráfico do resultado de consultar apressentações enquanto orientador

Sugestões:

- Relativamente à visualização, sugeria pensar melhor na ordem das colunas. A ordenação das apresentações será cronológica, certo?

3. Agrupar apresentações

Sugestões:

- Só depois de colocar o primeiro par é que se percebe que depois podemos fazer blocos que vão sendo guardados.
- Não percebi logo que era num menu especifico.
- Acho que "followed"não é o termo a usar. Ainda mais, depois de agrupar para marcação consecutiva, aparece a frase "already asked to be followed", que deveria ser "to be consecutive"ou "to be grouped".
- Tenho algumas dúvidas em relação à utilidade desta funcionalidade...a ideia não é fazer o agrupamento por defeito? Não pode gerar restrições a mais?

Figura 9.3: Gráfico do resultado de agrupar apresentações

- Em vez da palavra "Following,"eu usava a palavra "grouping".
- No meu caso só tinha duas apresentações para agrupar e nesse caso foi muito fácil. Pelo interface assumo que será possível agrupar em múltiplos grupos, o que é interessante. Se sim, diria que a minha avaliação passa para "Muito fácil".

4. Adicionar restrições a uma apresentação específica

Figura 9.4: Gráfico do resultado de adicionar restrições a uma apresentação

Sugestões:

- E se o elemento externo não aparecer na lista dropdown? A parte de depois adicionar restrições a uma apresentação é simples.
- Podia dar para marcar um dia inteiro num clique.
- Dá-se o mesmo relativamente ao feedback.

- Não sei se fiz o que era devido, porque marquei a indisponibilidade como sendo minha.
- Suponho que a ideia era marcar esta restrição em "My Unavailabilities"em vez de em "My Presentations Unavailabilities". Se há necessidade de separar, tornar mais claro na descrição que a segunda é para outros casos e não usar "you are available"(parece ser igual à primeira).
- Como apenas há dois intervenientes nas provas não será necessário identificar as restrições de cada um... para o organizador talvez?
- Na primeira vista, não percebi uma grande diferença entre "My Unavailabilities"e "My Presentation Unavailabilities"Se calhar a segunda pode ser "Jury Unavailabilities"ou algo assim.
- Novamente, seria interessante uma selecção mais simples para múltiplos slots.

5. Alterar a password

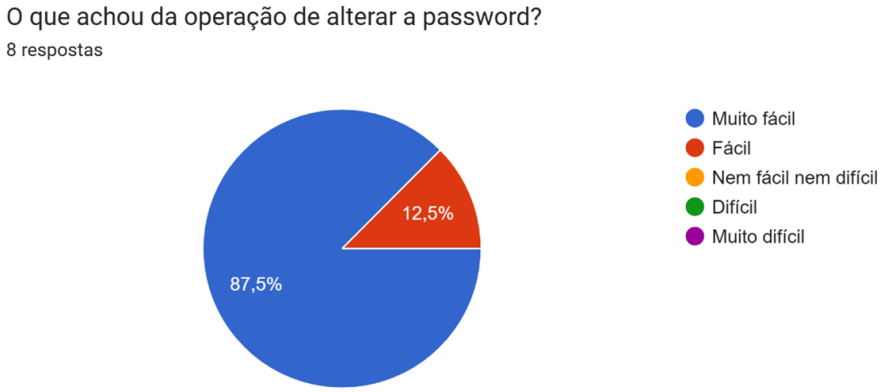

Figura 9.5: Gráfico do resultado de alterar a password

Sugestões:

- Depois de alterar a password, não era preciso aparecerem os campos de mudança de password.

9.2.2 Cenário *ORGANIZER*

Neste ponto apresentamos os resultados das operações realizadas com as credenciais de um utilizador do tipo *ORGANIZER*:

1. Fazer o download das apresentações para PDF

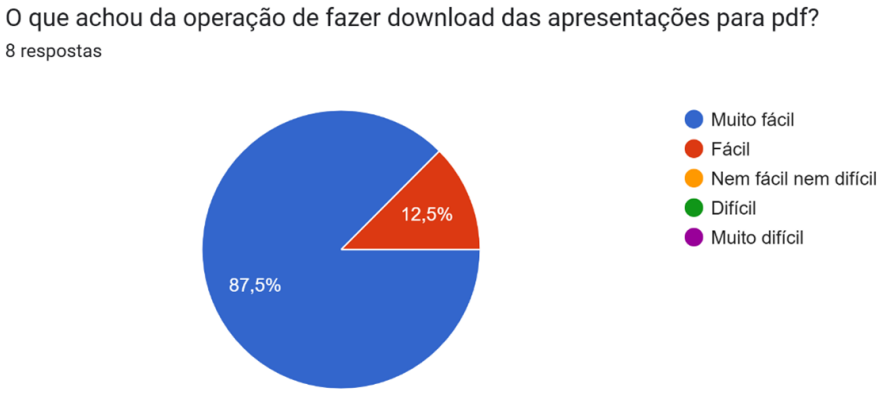

Figura 9.6: Gráfico do resultado de fazer downloar para PDF

Sugestões:

- Não aparecem as colunas dos participantes extra. Deveria ser uma opção?
- Da primeira vez, não reparei no botão.

2. Adicionar uma sala

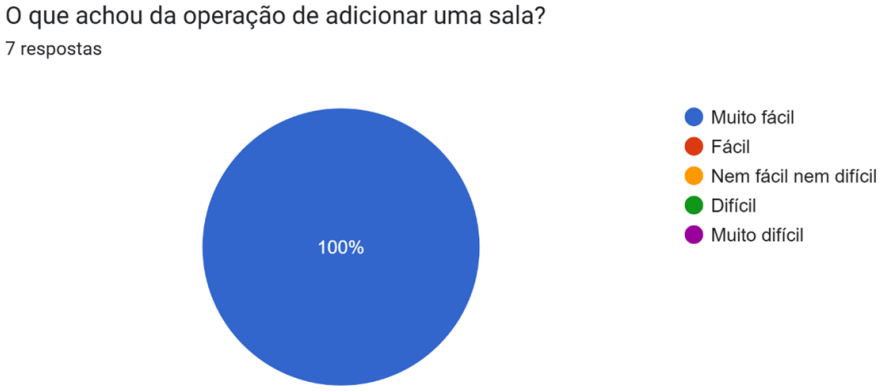

Figura 9.7: Gráfico do resultado de adicionar uma sala

Sugestões:

- No entanto, permite a inserção de salas que não existem. Sugiro usar listboxes, começando pela seleção do building. O botão cancel é necessário?

3. Adicionar 5 slots

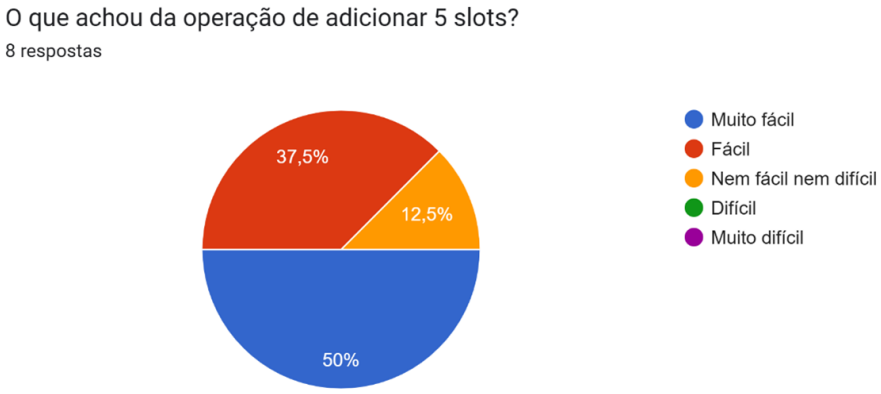

Figura 9.8: Gráfico do resultado de adicionar 5 slots

Sugestões:

- A início não reparei que dava para adicionar várias consecutivas, então adicionei uma. Depois tive de apagar essa para conseguir adicionar consecutivas. Talvez fosse mais intuitivo se desse para adicionar consecutivas mesmo que já lá estivesse alguma.
- Poderiamos definir a hora da slot inicialmente.
- No entanto, é restritivo, pois não dá para definir o range.
- If the task had been to add 5 non-consecutive slots it would have been rather lengthy. However, marking consecutive slots is very simple.

4. Adicionar um slot

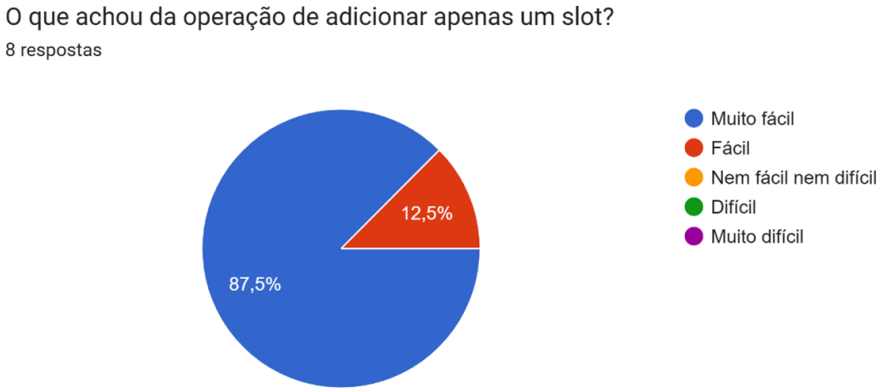

Figura 9.9: Gráfico do resultado de adicionar um slot

Sugestões:

- O campo da hora e minuto, apesar de se perceber, poderia ter uma indicação escrita.
- Aqui é possível adicionar slots a começar sem ser à hora certa. Se não sugeria usar a mesma técnica que é usada para indicar as indisponibilidades.
- Quando slot está visível graficamente, mas livre, podia haver a opção de adicioná-lo sem ter de recorrer ao calendário.

5. Adicionar uma apresentação

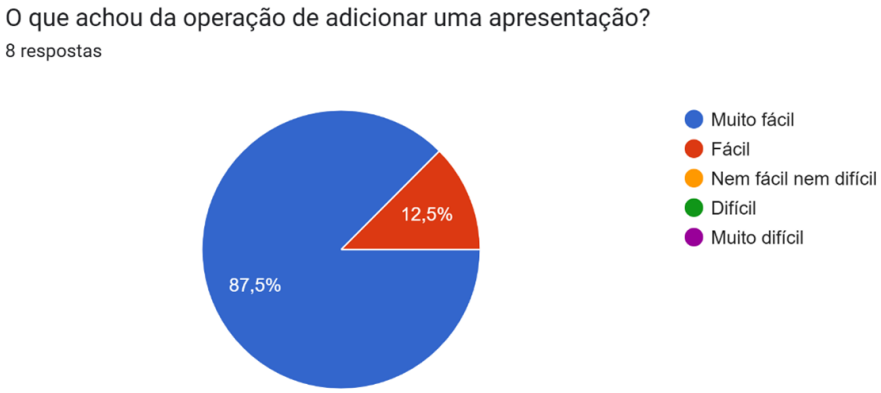

Figura 9.10: Gráfico do resultado de adicionar uma apresentação

Sugestões:

- Sem sugestões.

6. Adicionar um ficheiro excel com várias apresentações

Sugestões:

- Drag and drop é muito útil.

7. Apagar uma apresentação

Sugestões:

- Precisei de fazer zoom out para conseguir ver os campos todos, do título ao botão para apagar.
- Talvez ter para além do X a palavra "delete- idem para apagar utilizador.

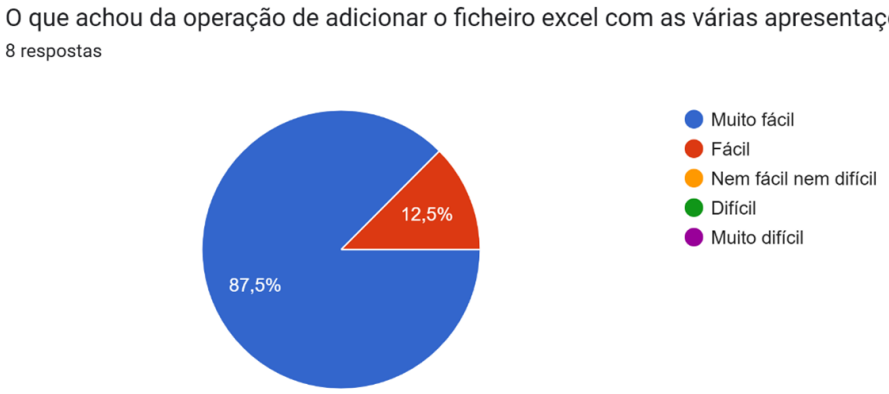

Figura 9.11: Gráfico do resultado de adicionar um ficheiro excel

Figura 9.12: Gráfico do resultado de apagar uma apresentação

- Para facilitar a localização da apresentação a apagar, seria interessante poder ordenar as mesmas por (ou mais) atributos.

8. Adicionar um utilizador

   Sugestões:

   - The text box area appears to be larger than it actually is. When I went to add the user's name and email I ended up clicking on the wrong spot without realizing.

9. Dar início à computação da alocação das salas e horários

   Sugestões:

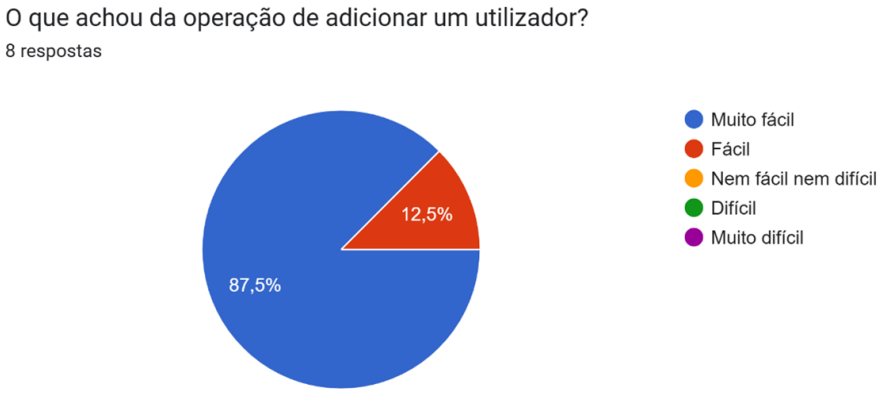

Figura 9.13: Gráfico do resultado de adicionar um utilizador

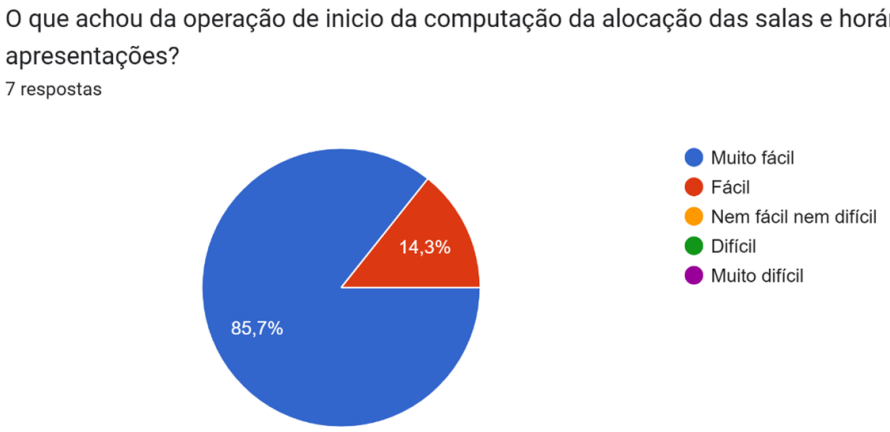

Figura 9.14: Gráfico do resultado de iniciar a computação

- Talvez pudesse haver logo a indicação que a operação demora alguns minutos (mesmo antes de iniciarmos o processo). Não é claro se podemos fazer outra coisa enquanto o Assignment corre.
- Não sei se tinha mesmo de esperar 10 minutos para ver alguma coisa.
- Penso que algumas verificações poderiam ser adicionadas neste ponto para informar o utilizador por exemplo quando alguma das componentes não tinha sido adicionada.
- Não reparei se algo foi computado... (fiz duas vezes).
- Para "garantir"que as restrições estão todas inseridas, a ideia é avisar as pessoas (email) quando o processo será efetuado? Depois de efetuado o processo, não seria melhor impedir a inserção de mais restrições?

9.2.3 Cenário *ADMIN*

Neste ponto apresentamos os resultados das operações realizadas com as credenciais de um utilizador do tipo *ADMIN*:

1. Mudar a *role* de um utilizador

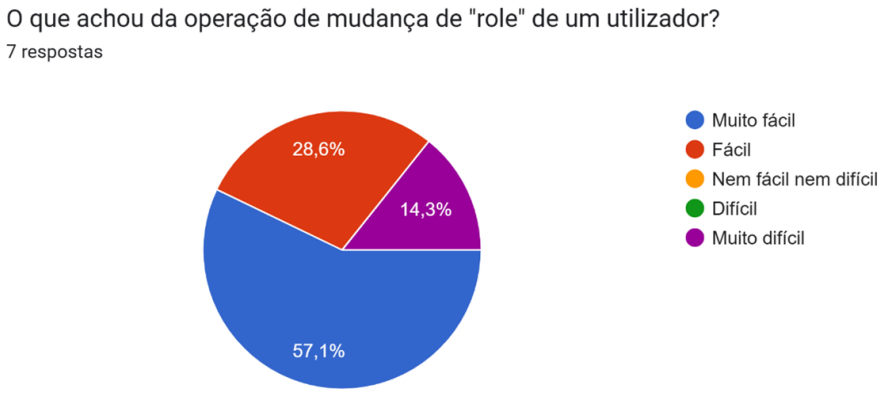

Figura 9.15: Gráfico do resultado de mudar a role a um utilizador

Sugestões:

- Deveria também aqui haver um botão de confirmação. Depois alterei para admin para experimentar e já não consegui desfazer esta alteração.
- Não consegui fazer. O User 1 é ADMIN. Parece-me que o seu papel não pode ser alterado e que não pode ser removido. Se fosse um dos outros teria sido muito fácil.
- O user 1 já era ADMIN e não podia ser mudado para ORGANIZER. Mudei o user 3 sem problemas. Reparei que se mudar alguém para ADMIN, nunca mais se pode mudar o seu role.
- Mas, ou eu vi mal, ou o user1 é admin e não é possível alterar a role :)
- The instructions ask me to change User1 to an organizer, but they are already an admin. So I assumed the instructions meant User2, and saw that it was super simple.
- A partir do momento que percebi que o role era editável (o que demorou 1 ou 2s), foi muito fácil. Talvez se possa utilizar um esquema de cores que torne mais claro os valores que podem ser alterados.

9.2.4 Feedbak Geral

Neste ponto apresentamos os resultados e o feedback geral da aplicação:

1. Clareza e simplicidade da aplicação

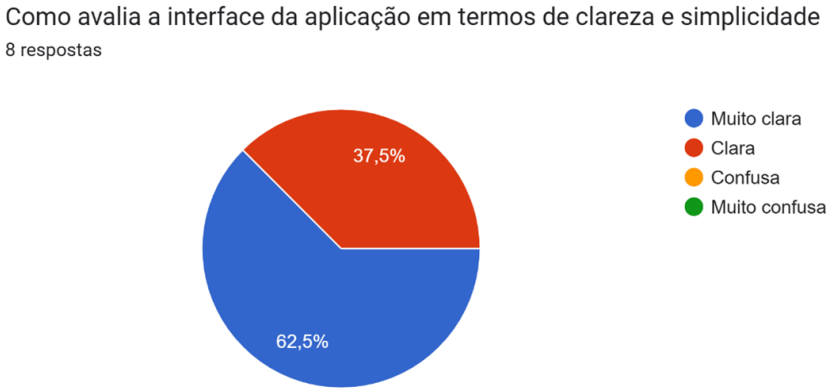

Figura 9.16: Gráfico do resultado da clareza e simplicidade da aplicação

2. Intuição da navegação

Figura 9.17: Gráfico do resultado da intuição da navegação

3. Encontrou algum erro ou dificuldade técnica? Se sim, por favor, descreva:

   a) Não.

   b) Nada.

   c) Penso que se deveriam verificar os termos usados em ingles para verificar se são os mais corretos. Achei que alguns não o estariam mas posso olhar para todos os textos mais tarde se ajudar.

   d) Um user com o role ADMIN não pode ser mudado de role.

   e) Esta pergunta é mesmo obrigatória? :)

f) Não cheguei a ver o resultado da computação dos assignments... estava a demorar :)

g) User1 was already an admin, as previously explained. Also there were issues with finding the text box for adding a user, as previously explained.

4. Tem algum outro comentário ou sugestão para melhorar a aplicação:

a) Acho que vai ser muito útil!

b) Está impecável. Só tenho pena de não ter conseguido fazer a última.

c) Nas colunas "Action"talvez passar a ter "X (delete)".

d) Sugeria criar uma separação entre as opções de menu que se referem às apresentações e as que se refere ao management (no caso do organizer e do administrador).

10

Conclusão

Nesta dissertação, foi desenvolvido com sucesso o objetivo principal: a criação de uma aplicação que automatiza a alocação de horários e salas para apresentações de documentos preparatórios das dissertações. O sistema é composto por diversos componentes — Servidor, Cliente, Base de dados e Microsserviço — que funcionam em conjunto para fornecer uma solução eficiente e otimizada para o problema. O Servidor é responsável por gerir as operações principais e a comunicação com o Microsserviço, enquanto o Cliente garante uma interface intuitiva para os utilizadores, facilitando o acesso e a interação. A Base de dados armazena e gere as informações de forma eficiente enquanto o Microsserviço utiliza o ASP para computar as soluções de alocação com base nas restrições fornecidas.

Ao longo deste trabalho foram exploradas várias soluções tecnológicas e de arquitetura para garantir a robustez da aplicação, a sua escalabilidade e característica intuitiva para todos os tipos de utilizadores. As funcionalidades implementadas e testadas por utilizadores com diferentes níveis de familiaridade com a aplicação evidenciam a eficácia do sistema e a simplicidade da interface, promovendo uma experiência de uso eficiente e sem complicações. O feedback obtido foi positivo, demonstrando que a aplicação não só atinge os objetivos propostos, como também facilita significativamente o processo de gestão das apresentações.

10.1 Trabalho Futuro

Embora os resultados alcançados com a aplicação tenham sido satisfatórios, há sempre espaço para melhorias e novas funcionalidades que possam tornar o sistema ainda mais eficiente e ágil. Uma direção importante para o trabalho futuro seria a criação de uma funcionalidade que permitisse verificar se as restrições impostas no sistema são consistentes e se é possível gerar um horário final sem conflitos, antes mesmo de iniciar o processo completo de computação. Esta funcionalidade teria o potencial de reduzir significativamente o tempo de espera, garantindo antecipadamente que o conjunto de restrições inseridas possa ou não produzir uma solução válida. Desta forma, o sistema tornaria-se mais responsivo

e dinâmico, permitindo ajustes antecipados nas condições impostas, sem a necessidade de esperar pelo processamento completo.

Além disso, dado o sucesso e a robustez do sistema desenvolvido, outro caminho relevante seria explorar a adaptação desta solução para outros cenários de planeamento e escalonamento. O sistema pode ser aplicado a diversos domínios, como a gestão de turnos de funcionários em empresas, a organização de eventos ou conferências com várias sessões simultâneas, ou mesmo em ambientes industriais para a alocação eficiente de recursos. Essas adaptações exigiriam ajustes nas restrições e critérios de otimização, mas a arquitetura desenvolvida nesta aplicação oferece uma base sólida para suportar estes diferentes contextos.

Estas expansões demonstram o potencial desta aplicação para se tornar uma ferramenta versátil e de grande utilidade em diversos cenários onde o escalonamento de recursos, tempo ou tarefas é necessário.

Bibliografia

[1] Abelcour. *Answer Set Programming syntax highlighter*. URL: https://marketplace.visualstudio.com/items?itemName=abelcour.asp-syntax-highlight&ssr=false#overview (acedido em 2023-01-31) (ver p. 14).

[2] *Apache Maven Project*. URL: https://maven.apache.org/ (acedido em 2024-09-12) (ver p. 35).

[3] D. Arias. *Adding Salt to Hashing: A Better Way to Store Passwords*. 2021. URL: https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/ (acedido em 2024-09-20) (ver p. xxii).

[4] D. Arias. *Hashing in Action: Understanding bcrypt*. 2021. URL: https://auth0.com/blog/hashing-in-action-understanding-bcrypt/ (acedido em 2024-09-20) (ver p. 47).

[5] M. Brain e M. D. Vos. «Answer Set Programming – a Domain in Need of Explanation A Position Paper». Em: ed. por T. Roth-Berghofer et al. 2008 (ver p. 5).

[6] *Class BCryptPasswordEncoder*. URL: https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/crypto/bcrypt/BCryptPasswordEncoder.html (acedido em 2024-09-14) (ver p. 47).

[7] *Clingo ORM (Clorm) — GitHub*. URL: https://github.com/potassco/clorm (acedido em 2023-02-04) (ver p. 18).

[8] *Clorm 1.4.1.post1*. 2022-10. URL: https://pypi.org/project/Clorm/ (acedido em 2023-02-04) (ver p. 18).

[9] *Clorm: An ORM API for Clingo*. URL: https://clorm.readthedocs.io/en/latest/ (acedido em 2023-02-04) (ver p. 18).

[10] *Clyngor — GitHub*. URL: https://github.com/aluriak/clyngor (acedido em 2023-02-03) (ver p. 18).

[11] *Clyngor 0.4.2*. 2022-03. URL: https://pypi.org/project/clyngor/ (acedido em 2023-02-03) (ver p. 18).

[12] W. contributors. *API — Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/API (acedido em 2023-02-01) (ver p. xxi).

[13] W. contributors. *Doodle(website) — Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/Doodle_(website) (acedido em 2023-02-06) (ver p. 15).

[14] W. contributors. *GitHub — Wikipedia, The Free Encyclopedia*. URL: https://pt.wikipedia.org/wiki/GitHub (acedido em 2023-02-01) (ver p. xxi).

[15] W. contributors. *Java Database Connectivity — Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/Java_Database_Connectivity (acedido em 2024-09-09) (ver p. xxi).

[16] W. contributors. *Microsoft Excel — Wikipedia, The Free Encyclopedia*. URL: https://pt.wikipedia.org/wiki/Microsoft_Excel (acedido em 2023-02-06) (ver p. 16).

[17] W. contributors. *Plug-in — Wikipedia, The Free Encyclopedia*. URL: https://pt.wikipedia.org/wiki/Plug-in (acedido em 2024-09-12) (ver p. xxi).

[18] W. contributors. *POST (HTTP) — Wikipedia, The Free Encyclopedia*. URL: https://pt.wikipedia.org/wiki/POST_(HTTP) (acedido em 2024-03-15) (ver p. xxi).

[19] W. contributors. *Software framework — Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/Software_framework (acedido em 2024-09-10) (ver p. xxi).

[20] W. contributors. *TypeScript — Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/TypeScript (acedido em 2024-09-24) (ver p. 66).

[21] W. contributors. *Visual Studio Code — Wikipedia, The Free Encyclopedia"*. URL: https://pt.wikipedia.org/wiki/Visual_Studio_Code (acedido em 2023-01-31) (ver p. 13).

[22] E. Erdem, M. Gelfond e N. Leone. «Applications of Answer Set Programming». Em: (2016). DOI: https://doi.org/10.1609/aimag.v37i3.2678 (ver p. 1).

[23] A. Falkner et al. «Industrial Applications of Answer Set Programming». Em: (2018). DOI: https://doi.org/10.1007/s13218-018-0548-6 (ver p. 1).

[24] F. Frankreiter. *Answer Set Programming Language Support*. URL: https://marketplace.visualstudio.com/items?itemName=ffrankreiter.answer-set-programming-language-support (acedido em 2023-01-31) (ver p. 14).

[25] M. Gebser et al. *A User's Guido to gringo, clasp, clingo and iclingo*. 2010 (ver p. 9).

[26] M. Gebser et al. *Potassco User Guide*. Second. 2019 (ver p. 6).

[27] *H2 Database Engine*. URL: https://www.h2database.com/html/main.html (acedido em 2023-02-10) (ver p. 27).

[28] D. Ho. *What is Notepad++*. URL: https://notepad-plus-plus.org/ (acedido em 2023-01-31) (ver p. 12).

[29] *iGROM: Overview*. URL: https://igrom.sourceforge.net/ (acedido em 2023-01-31) (ver p. 13).

[30] *JWT*. URL: https://jwt.io/ (acedido em 2024-09-19) (ver p. 46).

[31] *Kotlin*. URL: https://kotlinlang.org/ (acedido em 2024-09-11) (ver p. 35).

[32] *Kotlin Multiplatform*. URL: https://www.jetbrains.com/kotlin-multiplatform/ (acedido em 2024-09-12) (ver p. 35).

[33] V. Lifschitz. *Answer Set Programming*. 2019 (ver p. 9).

[34] V. Lifschitz. *What Is Answer Set Programming?* 2008 (ver pp. 1, 5).

[35] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (ver p. i).

[36] *Main Features — GitHub*. URL: https://github.com/DeMaCS-UNICAL/LoIDE/wiki/Main-Features (acedido em 2023-01-31) (ver p. 12).

[37] *Next.js*. URL: https://nextjs.org/ (acedido em 2024-09-24) (ver p. 67).

[38] *npm*. URL: https://www.npmjs.com/ (acedido em 2024-09-25) (ver p. xxi).

[39] J. Oetsch, J. Pührer e H. Tompits. «The SeaLion has Landed: An IDE for Answer-Set Programming—Preliminary Report». Em: (2011). DOI: https://doi.org/10.1007/978-3-642-41524-1_19 (ver p. 13).

[40] *Package clingo*. URL: https://potassco.org/clingo/python-api/5.6/clingo/ (acedido em 2023-02-03) (ver p. 18).

[41] Paula-andrabusoniu et al. «SeaLion: An eclipse-based IDE for answer-set programming with advanced debugging support». Em: (2013). DOI: http://dx.doi.org/10.1017/S1471068413000410 (ver p. 13).

[42] A. Petras. *Clingo4j — GitHub*. URL: https://github.com/clingo4j/clingo4j (acedido em 2023-02-01) (ver p. 17).

[43] *Postman*. URL: https://www.postman.com/ (acedido em 2024-09-12) (ver p. 35).

[44] *React*. URL: https://react.dev/ (acedido em 2024-09-24) (ver p. 66).

[45] *React*. URL: https://www.typescriptlang.org/ (acedido em 2024-09-24) (ver p. 66).

[46] *Spring Boot*. URL: https://spring.io/projects/spring-boot (acedido em 2024-09-11) (ver p. 34).

[47] *SVG*. URL: https://www.svgrepo.com/ (acedido em 2024-09-25) (ver p. xxii).

[48] *The official webpage of the LoIDE project, a web-based IDE for Logic Programming*. URL: https://demacs-unical.github.io/LoIDE/ (acedido em 2023-01-30) (ver p. 12).

[49] S. Thiele. *Clingo-rs — GitHub*. URL: https://github.com/potassco/clingo-rs (acedido em 2023-02-01) (ver p. 17).

[50] S. Thiele. *PyASP — GitHub*. URL: https://github.com/sthiele/pyasp (acedido em 2023-02-02) (ver p. 17).

[51] S. Thiele. *PyASP basics*. 2015-10. URL: https://sthiele.github.io/pyasp-basics/ (acedido em 2023-02-02) (ver p. 17).

[52] C. Walls. *Spring in Action*. Fourth. Manning Publications, 2015. ISBN: 978-1617292545 (ver p. 34).

[53] *What is Java Spring Boot?* URL: https://www.ibm.com/topics/java-spring-boot?mhsrc=ibmsearch_a&mhq=spring%20boot (acedido em 2024-09-11) (ver p. 34).

A

Operações de gestão dos Utilizadores

Este apêndice descreve detalhadamente as operações relacionadas com a gestão de utilizadores na aplicação. Estas operações são parte essencial para o bom funcionamento do sistema.

Cada operação é acompanhada por uma explicação das regras e restrições associadas, bem como os códigos de estado retornados em casos de sucesso ou erro. Este apêndice serve como referência técnica para entender como as interações de gestão de utilizadores são tratadas, assegurando a integridade e segurança das operações realizadas pelos utilizadores com as permissões necessárias.

A.1 Criar um utilizador

A operação "criar um utilizador"permite adicionar um novo utilizador na aplicação. Visto a aplicação ser interna à faculdade e de modo a impedir que sejam criados utilizadores indevidos, esta operação tem de ser realizada por um utilizador já registado na mesma. Esta operação recebe o nome e o email do utilizador a ser criado e irá efetuar algumas validações, tanto a nível do utilizador que está a realizar a operação, como dos dados recebidos.

É importante salientar que esta é uma operação transacional (*@Transactional*), sendo assim possível garantir a integridade dos dados. Caso haja algum erro durante o processo de criação de um novo utilizador, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem criar utilizadores.
- **Unicidade de Email e Nome** — Antes de criar um novo utilizador, o sistema verifica se já existe algum utilizador com o mesmo email ou nome na base de dados. Caso

se confirme, a criação do novo utilizador é impedida e enviado o devido código de estado.

Uma vez terminadas com sucesso as verificações, é iniciada a criação do novo utilizador no sistema.

Detalhes da Operação:

- **Geração de password** — O sistema gera uma password temporária e envia para o email fornecido, possibilitando o login do utilizador recentemente criado. Esta password segue os requisitos de segurança estabelecidos, incluindo um mínimo de 8 caracteres, contendo uma letra maiúscula, uma letra minúscula e um número. A password é guardada de forma encriptada na base de dados, utilizando o algoritmo *bcrypt*.
- **Role do utilizador** — Todos os utilizadores quando criados ficam com a *Role USER* associada.
- **Inicialização das listas das apresentações** — Ao criar um utilizador são inicializadas as listas de apresentações do utilizador, incluindo a lista de apresentações enquanto orientador, enquanto vogal e enquanto participante opcional. Estas listas serão posteriormente preenchidas na operação de criação das apresentações.
- **Inicialização da lista de apresentações seguidas** — Ao criar um utilizador é inicializada uma lista onde serão guardadas as necessidades do utilizador de ter apresentações seguidas. Esta lista será posteriormente preenchida pelo próprio utilizador em operações futuras.
- **Criação da entidade *UnavailabilityDAO*** — Uma entidade *UnavailabilityDAO* é criada e associada ao utilizador. Esta entidade será posteriormente preenchida com as indisponibilidades do utilizador.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando o utilizador é criado com sucesso, confirmando a operação e indicando que todas as entidades relacionadas foram incializadas corretamente.
- **HTTP 409 Conflict** — Retornado quando já existe um utilizador com o mesmo email ou nome.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a criação do utilizador.

A.2 Obter um utilizador

Esta operação permite consultar a informação relativa a um utilizador, recebendo como parâmetro o ID do utilizador a ser consultado.

Regras e Restrições:

- **Autorização** — Para consultar a informação relativa a um utilizador é necessário que o utilizador que executa a operação esteja autenticado e que seja, ou o próprio, ou possua um perfil do tipo *Organizer* ou *Admin*.
- **Existência do utilizador** — É necessário que o utilizador a consultar exista na base de dados.

Caso as verificações sejam cumpridas é enviada a informação do utilizador. Esta informação inclui o nome, o email e a role do utilizador, garantindo assim que dados sensíveis, como a password, não sejam expostos.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta da informação do utilizador pretendido é obtida com sucesso.
- **HTTP 404 Not Found** — Retornado quando não existe nenhum utilizador na base de dados com o ID recebido como parâmetro.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta do utilizador.

A.3 Obter todos os utilizadores

Esta operação permite consultar a informação de todos os utilizadores registados na aplicação.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem consultar todos os utilizadores registados na aplicação.

Esta operação envia uma lista com as informações de todos os utilizadores registados na aplicação. Tal como a operação "obter um utilizador", a informação inclui o nome, o email e a role dos utilizadores.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é obtida com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a operação de consulta dos utilizadores.

A.4 Obter os nomes de todos utilizadores

Esta operação permite consultar os nomes de todos os utilizadores registados na aplicação.

Regras e Restrições:

- **Autorização** — Todos os utilizadores da aplicação, desde que autenticados, podem consultar a lista de nomes dos utilizadores registados na aplicação.

Esta operação envia uma lista com os nomes de todos os utilizadores registados na aplicação.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é obtida com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta dos nomes dos utilizadores.

A.5 Apagar um utilizador

Esta operação permite remover um utilizador da aplicação, recebendo como parâmetro o ID do utilizador a ser removido.

É importante referir que, à semelhança da operação "criar um utilizador", esta é uma operação transacional (*@Transactional*), sendo assim possível garantir a integridade dos dados. Caso haja algum erro durante o processo de remoção de um utilizador, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Para remover um utilizador é necessário que o utilizador que executa a operação esteja autenticado. Além da autenticação, é possível remover um utilizador caso seja o próprio utilizador a querer apagar a sua conta, ou caso o utilizador possua um perfil do tipo *Admin* e, caso o perfil seja do tipo *Organizer*, é possível apagar um utilizador do tipo *User*.
- **Não ser do tipo *Admin*** — Caso não seja o próprio a apagar a sua conta de utilizador, é impossível remover um utilizador do tipo *Admin*.
- **Existência do utilizador** — É necessário que o utilizador a ser removido exista na base de dados.

Uma vez cumpridas as verificações, é iniciada a operação de remoção do utilizador do sistema.

Detalhes da Operação:

- **Remoção das apresentações** — Uma vez removido o utilizador, o sistema apaga todas as apresentações nas quais o utilizador era orientador e vogal. Consequentemente, todas as entidades *PresentationUnavailabilityDAO* dessas mesmas apresentações são apagadas tal como as entidades *FollowedPresentationsDAO*.
- **Remoção do seu nome** — Uma vez removido o utilizador, o sistema apaga o seu nome de todas as apresentações nas quais ele participava como participante opcional.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a remoção do utilizador é concluída com sucesso.
- **HTTP 404 Not Found** — Retornado quando não existe nenhum utilizador na base de dados com o ID recebido como parâmetro.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a remoção do utilizador.

A.6 Mudar a role de um utilizador

Esta operação permite mudar a role de um utilizador, recebendo como parâmetro o ID do utilizador a alterar e a sua nova role.

Esta é uma operação transacional (*@Transactional*), sendo assim possível garantir a integridade dos dados. Na eventualidade de existir algum erro durante o processo de mudança de role de um utilizador, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil *Admin* podem alterar roles a utilizadores do tipo *User* ou do tipo *Organizer*.
- **Existência do utilizador** — É necessário que o utilizador que irá ter a sua role alterada exista na base de dados.

Caso as verificações sejam cumpridas a role do utilizador é substituida pela nova role recebida como parâmetro.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a role do utilizador é alterada com sucesso.
- **HTTP 404 Not Found** — Retornado quando não existe nenhum utilizador na base de dados com o ID recebido como parâmetro.
- **HTTP 422 Unprocessable Entity** — Retornado quando o utilizador a ser alterado é do tipo *Admin*.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a mudança de role do utilizador.

A.7 Mudar a password de um utilizador

Esta operação permite alterar a password de um utilizador, recebendo como parâmetro a password antiga e a password nova para qual irá ser alterada.

Esta trata-se de uma operação transacional (*@Transactional*), razão pela qual é possível garantir a integridade dos dados. Caso haja algum erro durante o processo de mudança de password de um utilizador, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados podem alterar a sua própria password.

- **Existência do utilizador** — É necessário que o utilizador que executa esta operação exista na base de dados.
- **Correspondência de passwords** — É necessário que a password antiga digitada pelo utilizador corresponda à password registada na base de dados.

Após cumpridas as verificações a password do utilizador é alterada para a nova password recebida como parâmetro.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a alteração da password é concluída com sucesso.
- **HTTP 409 Conflict** — Retornado quando a password antiga recebida não tem correspondência com a password registada na base de dados.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a mudaça de password do utilizador.

B

Operações de gestão das Salas

Este apêndice descreve detalhadamente as operações relacionadas com a gestão das salas na aplicação. Estas operações são parte essencial para o bom funcionamento do sistema.

Cada operação é acompanhada por uma explicação das regras e restrições associadas, bem como os códigos de estado retornados em casos de sucesso ou erro. Este apêndice serve como referência técnica para entender como as interações de gestão das salas são tratadas, assegurando a integridade e segurança das operações realizadas pelos utilizadores com as permissões necessárias.

B.1 Criação das salas iniciais

Esta operação permite adicionar as salas predefinidas como iniciais. Esta operação foi criada com o intuito de agilizar o processo de criação de salas.

Esta é uma operação transacional (*@Transactional*), o que possibilita garantir a integridade dos dados. No caso de existir algum erro durante o processo de criação das salas, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem adicionar salas na aplicação.
- **Base de dados limpa** — É necessário que não exista nenhuma sala já criada na base de dados.

Depois de cumpridas as verificações, as salas começam a ser criadas na aplicação.

Detalhes da Operação:

- **Criação das salas** — As salas predefinidas criadas são as salas a que correspondem os números 110, 112, 114, 116, 119, 120, 121, 122, 123 e 124 do edifício 2.

- **Inicialização da lista de apresentações** — No ato de criação de cada sala é inicializada uma lista onde serão guardadas as apresentações a decorrer nessa mesma sala.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando todas as salas são criadas corretamente.
- **HTTP 409 Conflict** — Retornado no caso de já existir alguma sala criada na base de dados.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a criação das salas.

B.2 Criar uma sala

Esta operação permite adicionar uma sala à base de dados, recebendo como parâmetro o número da sala e o nome ou número do edifício.

Esta é, à semelhança da operação "criação das salas iniciais", uma operação transacional (*@Transactional*), sendo assim possível garantir a integridade dos dados. Caso haja algum erro durante o processo de criação da sala, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem adicionar salas na aplicação.
- **Unicidade na sala** — Antes de criar uma nova sala, o sistema verifica se já existe alguma sala criada na base de dados com o mesmo número e no mesmo edifício. Se assim for, a criação da sala é impedida e enviado o devido código de estado.

Após cumpridas as verificações, a sala é criada com os valores recebidos como parâmetro.

Detalhes da Operação:

- **Inicialização da lista de apresentações** — No ato de criação de uma sala é inicializada uma lista onde serão guardadas as apresentações a decorrer nessa mesma sala.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a sala é criada com os valores recebidos.
- **HTTP 409 Conflict** — Retornado no caso de já existir alguma sala criada com o mesmo número e no mesmo edifício.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a criação da sala.

B.3 Obter uma sala

Esta operação permite consultar a informação relativa a uma sala, recebendo como parâmetro o ID da sala a ser consultada.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem consultar a informação de uma sala.
- **Existência da sala** — É necessário que a sala a consultar exista na base de dados.

Uma vez cumpridas as verificações, é enviada a informação da sala. Esta informação inclui o número e o edifício.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta da informação da sala pretendida é obtida com sucesso.
- **HTTP 404 Not Found** — Retornado quando não existe nenhuma sala na base de dados com o ID recebido como parâmetro.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta da sala.

B.4 Obter todas as salas

Esta operação permite consultar a informação de todas as salas registadas na aplicação.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem consultar a informação das salas registadas na aplicação.

Esta operação envia uma lista com as informações de todas as salas registadas na aplicação. Tal como a operação "obter uma sala", a informação inclui o número e o edifício.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é obtida com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta das salas.

B.5 Apagar uma sala

Esta operação permite remover uma sala da aplicação, recebendo como parâmetro o ID da sala a ser removida.

Esta é também uma operação transacional (*@Transactional*), visto ser possível garantir a integridade dos dados. Caso haja algum erro durante o processo de remoção de uma sala, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem apagar uma sala.
- **Existência da sala** — É necessário que a sala a ser removida exista na base de dados.

A operação de remoção da sala do sistema é iniciada depois de cumpridas as verificações.

Detalhes da Operação:

- **Remoção da sala e do horário das apresentações** — Uma vez removida uma sala, todas as apresentações anteriormente atribuídas a esta ficarão sem sala e sem horário.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a remoção da sala pretendida seja concluída com sucesso.
- **HTTP 404 Not Found** — Retornado quando não existe nenhuma sala na base de dados com o ID recebido como parâmetro.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a remoção da sala.

C

Operações de gestão dos Slots

Este apêndice descreve detalhadamente as operações relacionadas com a gestão dos slots na aplicação. Estas operações são parte essencial para o bom funcionamento do sistema.

Cada operação é acompanhada por uma explicação das regras e restrições associadas, bem como os códigos de estado retornados em casos de sucesso ou erro. Este apêndice serve como referência técnica para entender como as interações de gestão dos slots são tratadas, assegurando a integridade e segurança das operações realizadas pelos utilizadores com as permissões necessárias.

C.1 Criar vários slots

Esta operação permite adicionar entre um a oito slots num mesmo dia. É recebido como parâmetro a data e o número de slots a serem adicionados nesse dia. Esta operação foi desenvolvida com o intuito de agilizar a criação de vários slots num mesmo dia, utilizando horários predefinidos.

Esta trata-se de uma operação transacional (*@Transactional*), pelo que é possível garantir a integridade dos dados. Na eventualidade de se verificar algum erro durante o processo de criação dos slots, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem adicionar slots na aplicação.
- **Unicidade na data dos slots** — É necessário que não exista nenhum slot criado na base de dados no dia pretendido.
- **Parametrização do número de slots** — É necessário que o número de slots esteja entre 1 e 8.

Uma vez cumpridas as restrições, é iniciada a operação de criação dos slots pretendidos.

Detalhes da Operação:

- **Criação dos slots** — Os slots predefinidos têm as horas de início predeterminadas com os seguintes valores: 9h00; 10h00; 11h00; 12h00; 14h00; 15h00; 16h00; 17h00. São criados o número de slots recebido em parâmetro, sendo feita a criação de forma incremental.
- **Inicialização das listas** — No ato de criação de cada slot são inicializadas as listas de indisponibilidades dos utilizadores, das apresentações, e as próprias apresentações. Estas listas vão ser preenchidas quando adicionadas restrições e apresentações na aplicação.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando todos os slots são criados com sucesso.
- **HTTP 409 Conflict** — Retornado se já existirem slots criados no dia pretendido e caso o número de slots seja menor que 1 ou maior que 8.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a criação dos slots.

C.2 Criar um slot

A operação "criar um slot"permite adicionar um slot à base de dados, recebendo como parâmetro a data e a hora de início do slot.

Esta é uma operação transacional (*@Transactional*), o que garante a integridade dos dados. Caso haja algum erro durante o processo de criação do slot, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem adicionar um slot.
- **Unicidade do slot** — É necessário que não exista nenhum slot com o mesmo dia e hora de início.

Cumpridas as restrições, é iniciada a operação de criação do slot.

Detalhes da Operação:

- **Inicialização das listas** — No ato de criação do slot são inicializadas as listas de indisponibilidades dos utilizadores, das apresentações, e as próprias apresentações. Estas listas vão ser preenchidas quando adicionadas restrições e apresentações na aplicação.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando o slot é criado com sucesso.
- **HTTP 409 Conflict** — Retornado se já existir um slot com o mesmo dia e hora de início.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a criação do slot.

C.3 Obter um slot

Esta operação permite consultar a informação relativa a um slot, recebendo como parâmetro o ID do slot a ser consultado.

Regras e Restrições:

- **Autorização** — Todos os utilizadores autenticados podem consultar a informação de um slot.

Caso as verificações sejam cumpridas é enviada a informação do slot. Esta informação inclui o dia e a hora de início do slot.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é obtida com sucesso.
- **HTTP 404 Not Found** — Retornado quando não existe nenhum slot na base de dados com o ID recebido como parâmetro.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta do slot.

C.4 Obter todos os slots

Esta operação permite consultar a informação relativa a todos os slots registados na aplicação.

Regras e Restrições:

- **Autorização** — Todos os utilizadores autenticados podem consultar a informação dos slots.

Caso as verificações sejam cumpridas é enviada a informação de todos os slots. Tal como na operação "obter um slot", esta informação inclui o dia e a hora de início dos slots.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é obtida com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta dos slots.

C.5 Apagar um slot

Esta operação permite remover um slot da aplicação, recebendo como parâmetro o ID do slot a remover.

Esta é uma operação transacional (*@Transactional*), sendo assim possível garantir a integridade dos dados. Caso haja algum erro durante o processo de remoção do slot, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem remover um slot.

Depois de cumpridas as restrições, é iniciada a operação de remoção do slot do sistema.

Detalhes da Operação:

- **Remoção do slot** — Uma vez removido o slot, todas as indisponibilidades de utilizadores e de apresentações sobre este slot também são removidas.
- **Remoção do horário e da sala das apresentações** — Uma vez removido o slot, todas as apresentações a ocorrer neste slot ficam sem sala e sem horário atribuído.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando o slot é removido com sucesso.
- **HTTP 404 Not Found** — Retornado quando não existe nenhum slot na base de dados com o ID recebido como parâmetro.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a remoção do slot.

D

Operações de gestão das Apresentações

Este apêndice descreve detalhadamente as operações relacionadas com a gestão das apresentações na aplicação. Estas operações são parte essencial para o bom funcionamento do sistema.

Cada operação é acompanhada por uma explicação das regras e restrições associadas, bem como os códigos de estado retornados em casos de sucesso ou erro. Este apêndice serve como referência técnica para entender como as interações de gestão das apresentações são tratadas, assegurando a integridade e segurança das operações realizadas pelos utilizadores com as permissões necessárias.

D.1 Criar uma apresentação

A operação "criar uma apresentação"permite adicionar uma apresentação na aplicação. São recebidos como parâmetro as seguintes informações:

- Número do estudante;
- Nome do estudante;
- Título da dissertação;
- Nome do orientador;
- Nome do vogal.

Esta é uma operação transacional (*@Transactional*), pelo que é possível garantir a integridade dos dados. Na eventualidade de se verificar algum erro durante o processo de criação da apresentação, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem adicionar apresentações na aplicação.
- **Existência do Orientador** — É necessário que exista o orientador cujo nome foi recebido como parâmetro.
- **Existência do Vogal** — É necessário que o vogal com o nome recebido em parâmetro exista da base de dados.
- **Orientador e Vogal diferentes** — Não é possível que um utilizador seja orientador e vogal da mesma apresentação.
- **Unicidade no aluno** — Um aluno pode ter apenas uma apresentação na aplicação.

Cumpridas as restrições, é iniciada a operação de criação da apresentação pretendida.

Detalhes da Operação:

- **Criação do objeto *PresentationUnavailabilityDAO*** — Quando é criada uma apresentação, é também criado o objeto *PresentationUnavailabilityDAO* com o orientador da apresentação. Este objeto irá ter as restrições horárias desta mesma apresentação.
- **Inicialização da sala e do slot a null** — Quando é criada uma apresentação, a sala e o slot onde esta decorrerá são inicializados a null. Este valores serão depois computados pelo Microsserviço.
- **Inicialização dos participantes opcionais** — Na criação da apresentação, os participantes opcionais são também inicializados a null, indicando que ainda não existe nenhum participante adicional na apresentação.
- **Inicialização da lista de Apresentações seguidas** — É também criada uma lista para depois serem alocados os grupos de apresentações consecutivas em que esta apresentação estará presente.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a apresentação é criada com sucesso na base de dados.
- **HTTP 422 Unprocessable Entity** — Retornado quando as restrições indicadas não são cumpridas.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.

- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a criação da apresentação.

D.2 Upload de várias apresentações

Esta operação permite adicionar várias apresentações na aplicação através de um ficheiro Excel.

Esta também se trata de uma operação transacional (*@Transactional*), sendo assim possível garantir a integridade dos dados. Na eventualidade de se verificar algum erro durante o processo de criação das apresentações, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem adicionar apresentações na aplicação.
- **Existência do Orientador** — É necessário que exista o orientador cujo nome foi recebido como parâmetro.
- **Existência do Vogal** — É necessário que o vogal com o nome recebido em parâmetro exista da base de dados.
- **Unicidade no aluno** — Um aluno pode ter apenas uma apresentação na aplicação.

Após verificação das restrições, é iniciada a criação das apresentações presentes no ficheiro Excel recebido.

Detalhes da Operação:

- **Criação das apresentações** — A informação é retirada do ficheiro Excel e as apresentações são criadas individualmente na aplicação. Para cada apresentação criada através desta operação, as inicializações referidas na operação "criar uma apresentação"também são realizadas.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando todas as apresentações são criadas com sucesso.
- **HTTP 422 Unprocessable Entity** — Retornado quando as restrições indicadas não são cumpridas.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.

- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a criação das salas.

D.3 Obter uma apresentação

Esta operação permite consultar a informação relativa a uma apresentação, recebendo o seu ID como parâmetro.

Regras e Restrições:

- **Autorização** — Todos os utilizadores autenticados na aplicação podem consultar a informação relativa a uma apresentação.
- **Existência da apresentação** — É necessário que a apresentação com o ID recebido exista na base de dados.

Caso as verificações sejam cumpridas, é devolvida a informação da apresentação.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é concluída com sucesso.
- **HTTP 404 Not Found** — Retornado quando a apresentação com o ID recebido não existe na aplicação.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta.

D.4 Obter todas as apresentações

Esta operação permite consultar a informação de todas as apresentações que existem na aplicação.

Regras e Restrições:

- **Autorização** — Todos os utilizadores autenticados na aplicação podem consultar a informação das apresentações.

Uma vez cumpridas as verificações, é devolvida a lista das apresentações que existem na aplicação.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é concluída com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta.

D.5 Apagar uma apresentação

Esta operação permite apagar uma apresentação da base de dados. Para esta operação ser executada é necessário a indicação do ID da apresentação a apagar.

Esta é uma operação transacional (*@Transactional*), pelo que é possível garantir a integridade dos dados. Na eventualidade de se verificar algum erro durante o processo de remoção de uma apresentação, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem remover uma apresentação na aplicação.
- **Existência da apresentação** — É necessário que exista a apresentação com o ID recebido.

A apresentação é removida da base de dados após as verificações serem confirmadas.

Detalhes da Operação:

- **Remoção do objeto *PresentationUnavailabilityDAO*** — Ao remover uma apresentação também é eliminado o objeto *PresentationUnavailabilityDAO* correspondente.
- **Remoção das entidades *FollowedPresentationsDAO*** — Todas as entidades *FollowedPresentationsDAO* que contém a apresentação removida são apagadas da aplicação.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a remoção é executada com sucesso.
- **HTTP 404 Not Found** — Retornado quando a apresentação com o ID recebido em parâmetro não existe.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.

- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a remoção da apresentação.

D.6 Obter as apresentações de um utilizador

Esta operação permite consultar as apresentações nas quais um utilizador participa. É recebido como parâmetro o ID do utilizador a procurar.

Regras e Restrições:

- **Autorização** — Para consultar as apresentações de um utilizador é necessário que o utilizador a executar a operação esteja autenticado e seja, ou o próprio, ou possua um perfil do tipo *Organizer* ou *Admin*.

Uma vez cumpridas as verificações, é devolvida a lista das apresentações nas quais o utilizador ou é orientador, ou é vogal, ou é um participante opcional.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é concluída com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta.

D.7 Obter as apresentações de um utilizador enquanto Orientador ou Vogal

Esta operação permite consultar as apresentações nas quais um utilizador é o orientador ou o vogal. Para a realização desta operação é necessário receber como parâmetro o ID do utilizador a procurar.

Regras e Restrições:

- **Autorização** — Para consultar as apresentações de um utilizador é necessário que o utilizador a executar a operação esteja autenticado e seja, ou o próprio, ou possua um perfil do tipo *Organizer* ou *Admin*.

Depois das verificações serem cumpridas, é devolvida a lista das apresentações pretendida.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é concluída com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta.

D.8 Obter as apresentações de um utilizador enquanto Orientador

Esta operação permite consultar as apresentações nas quais um utilizador é o orientador. Tal como na consulta anterior, o ID do utilizador é recebido como parâmetro.

Regras e Restrições:

- **Autorização** — Para consultar as apresentações de um utilizador é necessário que o utilizador a executar a operação esteja autenticado e seja, ou o próprio, ou possua um perfil do tipo *Organizer* ou *Admin*.

Após as verificações, é devolvida a lista pretendida.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é concluída com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta.

D.9 Obter as apresentações de um utilizador enquanto Vogal

Esta operação permite consultar as apresentações de um utilizador nas quais ele é o vogal. Nesta operação o ID do utilizador é recebido como parâmetro.

Regras e Restrições:

- **Autorização** — Para consultar as apresentações de um utilizador é necessário que o utilizador a executar a operação esteja autenticado e seja, ou o próprio, ou possua um perfil do tipo *Organizer* ou *Admin*.

Caso as verificações sejam cumpridas, é devolvida a lista das apresentações do utilizador pretendida.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é concluída com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta.

D.10 Obter as apresentações de um utilizador enquanto Participante opcional

Esta operação permite consultar as apresentações nas quais o utilizador com o ID recebido como parâmetro é um participante opcional.

Regras e Restrições:

- **Autorização** — Para consultar as apresentações de um utilizador é necessário que o utilizador a executar a operação esteja autenticado e seja, ou o próprio, ou possua um perfil do tipo *Organizer* ou *Admin*.

Cumpridas as verificações, é devolvida a lista pretendida.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é concluída com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta.

D.11 Adicionar o primeiro participante opcional a uma apresentação

Esta operação permite adicionar um participante opcional a uma apresentação. Para isso é necessário receber o ID da apresentação e o nome do utilizador a adicionar.

Esta trata-se de uma operação transacional (*@Transactional*), pelo que é possível garantir a integridade dos dados. Na eventualidade de se verificar algum erro durante o processo de adicionar um participante opcional à apresentação, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Para adicionar um primeiro participante opcional é preciso que o utilizador que realiza a operação esteja autenticado e que, ou tenha um perfil de *Organizer* ou *Admin*, ou seja o orientador ou vogal da apresentação.
- **Existência da apresentação** — É necessário que a apresentação com o ID recebido exista na aplicação.
- **Existência do utilizador a adicionar** — É necessário que exista o utilizador cujo nome foi recebido como parâmetro.
- **O utilizador a ser adicionado não seja o orientador ou o vogal** — Para adicionar um utilizador como participante opcional, é necessário que esse utilizador não seja nem o orientador nem o vogal da apresentação em questão.

Uma vez cumpridas as verificações, o utilizador é adicionado à apresentação.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando é adicionado com sucesso um utilizador.
- **HTTP 404 Not Found** — Retornado quando não existe a apresentação ou o utilizador a adicionar.
- **HTTP 422 Unprocessable Entity** — Retornado caso o utilizador a adicionar seja o orientador ou o vogal da apresentação.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a operação.

D.12 Adicionar o segundo participante opcional a uma apresentação

Esta operação permite adicionar um segundo participante opcional a uma apresentação. Para realizar esta operação é necessário receber o ID da apresentação e o nome do utilizador a adicionar como parâmetro.

Esta trata-se de uma operação transacional (*@Transactional*), pelo que é possível garantir a integridade dos dados. Na eventualidade de se verificar algum erro durante o processo de adicionar um participante opcional à apresentação, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem realizar esta operação.
- **Existência da apresentação** — É necessário que a apresentação com o ID recebido exista na aplicação.
- **Existência do utilizador a adicionar** — É necessário que exista o utilizador cujo nome foi recebido como parâmetro.
- **O utilizador a ser adicionado não seja o orientador ou o vogal** — Para adicionar um utilizador como participante opcional, é necessário que esse utilizador não seja nem o orientador nem o vogal da apresentação em questão.

Cumpridas as verificações, o utilizador é adicionado à apresentação.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando o utilizador é adicionado com sucesso.
- **HTTP 404 Not Found** — Retornado quando não existe a apresentação ou o utilizador a adicionar.
- **HTTP 422 Unprocessable Entity** — Retornado caso o utilizador a adicionar seja o orientador ou o vogal da apresentação.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a operação.

D.13 Apagar o primeiro participante opcional de uma apresentação

Esta operação permite remover o primeiro participante opcional de uma apresentação. O ID da apresentação na qual vai ser removido o utilizador é recebido como parâmetro.

Esta trata-se de uma operação transacional (*@Transactional*), pelo que é possível garantir a integridade dos dados. Na eventualidade de se verificar algum erro durante o processo de remover um participante opcional à apresentação, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Para remover o primeiro participante opcional é preciso que o utilizador que realiza a operação esteja autenticado e que, ou tenha um perfil de *Organizer* ou *Admin*, ou seja o orientador ou vogal da apresentação.
- **Existência da apresentação** — É necessário que a apresentação com o ID recebido exista na aplicação.

Verificadas as restrições, o utilizador é removido.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a remoção é executada com sucesso.
- **HTTP 404 Not Found** — Retornado quando a apresentação com o ID recebido não existe na aplicação.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a operação de remoção.

D.14 Apagar o segundo participante opcional de uma apresentação

Esta operação permite remover o segundo participante opcional de uma apresentação, cujo ID é recebido como parâmetro.

Esta trata-se de uma operação transacional (*@Transactional*), pelo que é possível garantir a integridade dos dados. Na eventualidade de se verificar algum erro durante o processo de remover um participante opcional à apresentação, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem realizar esta remoção.
- **Existência da apresentação** — É necessário que a apresentação com o ID recebido exista na aplicação.

Caso as restrições sejam cumpridas, o utilizador é removido da apresentação.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a remoção é executada com sucesso.
- **HTTP 404 Not Found** — Retornado quando a apresentação com o ID recebido não existe na aplicação.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a operação de remoção.

E

Operações de gestão das Restrições dos Utilizadores

Este apêndice descreve detalhadamente as operações relacionadas com a gestão das restrições dos utilizadores na aplicação. Estas operações são parte essencial para o bom funcionamento do sistema.

Cada operação é acompanhada por uma explicação das regras e restrições associadas, bem como os códigos de estado retornados em casos de sucesso ou erro. Este apêndice serve como referência técnica para entender como as interações de gestão das restrições dos utilizadores são tratadas, assegurando a integridade e segurança das operações realizadas pelos utilizadores com as permissões necessárias.

E.1 Criar uma restrição

A operação "criar uma restrição"permite a um utilizador adicionar um slot à sua lista de indisponibilidades. Para que esta operação seja realizada é necessário ser recebido como parâmetro o ID do utilizador e do slot.

Esta é uma operação transacional (*@Transactional*), pelo que é possível garantir a integridade dos dados. Na eventualidade de se verificar algum erro durante o processo de criação da restrição, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Para adicionar uma restrição é preciso que o utilizador a realizar a operação esteja autenticado e que seja, ou o próprio, ou tenha o perfil de *Organizer* ou *Admin*.
- **Existência do slot** — É necessário que o slot com o ID recebido exista na base de dados.

Uma vez verificadas as restrições, é adicionado o slot à lista de indisponibilidades do utilizador.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a criação da restrição é executada com sucesso.
- **HTTP 422 Unprocessable Entity** — Retornado quando o slot a adicionar à lista de restrições não existe.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a criação.

E.2 Obter uma restrição

Esta operação permite consultar uma restrição horária. Para executar esta operação é recebido como parâmetro o ID da restrição a consultar.

Regras e Restrições:

- **Autorização** — Para consultar a restrição, é necessário que o utilizador a realizar a operação esteja autenticado, e que ou seja o dono da restrição, ou tenha o perfil de *Organizer* ou *Admin*.
- **Existência da restrição** — É necessário que exista a restrição a consultar.

Confirmadas as verificações, é enviada a informação da restrição pretendida.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é efetuada com sucesso.
- **HTTP 404 Not Found** — Retornado quando a restrição não existe na base de dados.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta.

E.3 Obter todas as restrições

Esta operação permite consultar a informação de todas as restrições horárias presentes na aplicação.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem consultar todas as restrições presentes na aplicação.

Uma vez confirmadas as verificações, é enviada uma lista com todas as restrições registadas na aplicação.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é efetuada com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta.

E.4 Apagar uma restrição

A operação "apagar uma restrição"permite a um utilizador remover um slot da sua lista de indisponibilidades. Para que esta operação seja realizada é necessário que seja recebido como parâmetro o ID do utilizador e do slot.

Cumpre sublinhar que esta é uma operação transacional (*@Transactional*), pelo que é possível garantir a integridade dos dados. Na eventualidade de se verificar algum erro durante o processo de remoção da restrição, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Para remover uma restrição é preciso que o utilizador a realizar a operação esteja autenticado e que seja, ou o próprio, ou tenha o perfil de *Organizer* ou *Admin*.
- **Existência do slot** — É necessário que o slot com o ID recebido exista na base de dados.

Uma vez verificadas as restrições, o slot é removido da lista de indisponibilidades do utilizador.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a remoção da restrição é executada com sucesso.
- **HTTP 422 Unprocessable Entity** — Retornado quando o slot a remover da lista não existe.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a remoção.

E.5 Obter todas as restrições de um utilizador

Esta operação permite consultar todas as indisponibilidades horárias de um utilizador. O ID do utilizador é enviado como parâmetro.

Regras e Restrições:

- **Autorização** — Para poder consultar as restrições de um determinado utilizador é preciso que o utilizador que realiza a operação esteja autenticado e que seja, ou o próprio, ou tenha o perfil de *Organizer* ou *Admin*.

Caso as restrições sejam cumpridas, é enviada a informação relativa às restrições do utilizador.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é feita com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta das restrições.

F

Operações de gestão das Restrições das Apresentações

Este apêndice descreve detalhadamente as operações relacionadas com a gestão das restrições das apresentações na aplicação. Estas operações são parte essencial para o bom funcionamento do sistema.

Cada operação é acompanhada por uma explicação das regras e restrições associadas, bem como os códigos de estado retornados em casos de sucesso ou erro. Este apêndice serve como referência técnica para entender como as interações de gestão das restrições das apresentações são tratadas, assegurando a integridade e segurança das operações realizadas pelos utilizadores com as permissões necessárias.

F.1 Criar uma restrição

A operação "criar uma restrição"permite adicionar uma nova restrição a uma apresentação. Para a realização desta operação é necessário receber como parâmetro os IDs do slot e da apresentação.

Esta é uma operação transacional (*@Transactional*), pelo que é possível garantir a integridade dos dados. Na eventualidade de se verificar algum erro durante o processo de criação da restrição, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Para adicionar uma restrição a uma apresentação é necessário que o utilizador a efetuar a operação esteja autenticado e que, ou seja o orientador da apresentação, ou tenha um perfil de *Organizer* ou *Admin*.
- **Existência do slot** — O slot com o ID recebido tem de existir na aplicação.
- **Existência da apresentação** — A apresentação com o ID recebido tem de existir na aplicação.

Cumpridas as restrições, é adicionado o slot à lista de restrições da apresentação.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a criação da restrição é executada com sucesso.
- **HTTP 422 Unprocessable Entity** — Retornado quando não existe o slot ou a apresentação com os IDs recebidos.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a criação da restrição.

F.2 Obter uma restrição

Esta operação permite consultar a informação relativa a uma restrição horária. Para executar esta operação é recebido como parâmetro o ID da restrição a consultar.

Regras e Restrições:

- **Autorização** — Para consultar a restrição, é necessário que o utilizador a realizar a operação esteja autenticado, e que ou seja o orientador da apresentação, ou tenha o perfil de *Organizer* ou *Admin*.
- **Existência da restrição** — É necessário que a restrição com o ID recebido exista na base de dados da aplicação.

Caso as verificações sejam cumpridas, é devolvida a informação da restrição pretendida.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é efetuada com sucesso.
- **HTTP 404 Not Found** — Retornado quando a restrição não existe na base de dados.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta.

F.3 Obter todas as restrições

Esta operação permite consultar todas as restrições relativas a apresentações presentes na aplicação.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem adicionar apresentações na aplicação.

Uma vez cumpridas as verificações, é devolvida a lista de todas as restrições presentes na aplicação.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é efetuada com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta.

F.4 Apagar uma restrição

A operação "apagar uma restrição"permite remover uma restrição de uma apresentação. Para a realização desta operação é necessário receber como parâmetro os IDs do slot e da apresentação.

Esta é uma operação transacional (*@Transactional*), pelo que é possível garantir a integridade dos dados. Na eventualidade de se verificar algum erro durante o processo de remoção da restrição, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Para adicionar uma restrição a uma apresentação é necessário que o utilizador a efetuar a operação esteja autenticado e que, ou seja o orientador da apresentação, ou tenha um perfil de *Organizer* ou *Admin*.
- **Existência do slot** — O slot com o ID recebido tem de existir na aplicação.
- **Existência da apresentação** — A apresentação com o ID recebido tem de existir na aplicação.

Cumpridas as restrições, é removido o slot da lista de restrições da apresentação.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a remoção da restrição é executada com sucesso.
- **HTTP 422 Unprocessable Entity** — Retornado quando não existe o slot ou a apresentação com os IDs recebidos.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a remoção da restrição.

F.5 Obter as restrições de uma apresentação

Esta operação permite consultar as restrições de uma apresentação, cujo ID é recebido como parâmetro.

Regras e Restrições:

- **Autorização** — Para consultar as restrições de uma apresentação é necessário que o utilizador a realizar a operação esteja autenticado e que, ou seja o orientador ou vogal da apresentação, ou tenha um perfil de *Organizer* ou *Admin*.

Uma vez cumpridas as verificações, são devolvidas as restrições da apresentação pretendida.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é efetuada com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta.

G

Operações de gestão das Apresentações Seguidas

Este apêndice descreve detalhadamente as operações relacionadas com a gestão das apresentações seguidas na aplicação. Estas operações são parte essencial para o bom funcionamento do sistema.

Cada operação é acompanhada por uma explicação das regras e restrições associadas, bem como os códigos de estado retornados em casos de sucesso ou erro. Este apêndice serve como referência técnica para entender como as interações de gestão das apresentações seguidas são tratadas, assegurando a integridade e segurança das operações realizadas pelos utilizadores com as permissões necessárias.

G.1 Criar um objeto de armazenamento de apresentações seguidas

Esta operação permite a um utilizador indicar que necessita que duas apresentações sejam marcadas de forma consecutiva. Para a execução desta operação é preciso serem recebidos os IDs das apresentações e do utilizador.

Esta, sendo uma operação transacional (*@Transactional*), permite garantir a integridade dos dados. Na eventualidade de se verificar algum erro durante o processo de criação, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Para criar este objeto é necessário que o utilizador a realizar a operação esteja autenticado e que, ou seja o orientador ou vogal das apresentações, ou tenha um perfil de *Organizer* ou *Admin*.
- **Número limite de apresentações** — Apenas é possível adicionar duas apresentações a este objeto.

- **Unicidade do objeto** — É necessário que ainda não exista nenhum objeto com as apresentações a juntar.
- **Não existirem 3 objetos com a mesma apresentação** — Caso já existam 2 objetos nos quais exista uma das apresentações recebidas como parâmetro, a criação é impedida.

Cumpridas as restrições, é criado o objeto pretendido.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a operação de criação é realizada com sucesso.
- **HTTP 422 Unprocessable Entity** — Retornado quando as restrições indicadas não são cumpridas.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a criação.

G.2 Obter todos os objetos de armazenamento de apresentações seguidas

Esta operação permite a consulta de todos objetos que existem na aplicação.

Regras e Restrições:

- **Autorização** — Apenas utilizadores autenticados e com o perfil de *Organizer* ou *Admin* podem adicionar apresentações na aplicação.

Ao serem cumpridas as restrições, é enviada uma lista com todos os objetos presentes na aplicação.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é realizada com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta.

G.3 Obter todos os objetos de armazenamento de apresentações seguidas de um utilizador

Esta operação permite a consulta de todos os objetos de um utilizador, cujo ID é recebido como parâmetro.

Regras e Restrições:

- **Autorização** — Para consultar os objetos de um utilizador é necessário que o utilizador a executar a operação esteja autenticado e seja, ou o próprio, ou possua um perfil do tipo *Organizer* ou *Admin*.

Uma vez cumpridas as verificações, é enviada uma lista com os objetos do utilizador pretendido

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a consulta é realizada com sucesso.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a consulta.

G.4 Apagar um objeto de armazenamento de apresentações seguidas

Esta operação permite que um utilizador apague um objeto com duas apresentações seguidas. Para a execução desta operação é preciso receber o ID do objeto.

Esta é uma operação transacional (*@Transactional*), pelo que é possível garantir a integridade dos dados. Na eventualidade de se verificar algum erro durante o processo de remoção, toda a operação é revertida.

Regras e Restrições:

- **Autorização** — Para criar este objeto é necessário que o utilizador a realizar a operação esteja autenticado e que, ou seja ele o criador do objeto, ou tenha um perfil de *Organizer* ou *Admin*.

Cumpridas as restrições, é removido o objeto da base de dados.

Respostas e Códigos de Estado:

- **HTTP 200 OK** — Retornado quando a operação de remoção é realizada com sucesso.
- **HTTP 404 Not Found** — Retornado quando o objeto com o ID recebido não existe na base de dados.
- **HTTP 401 Unauthorized** — Retornado se um utilizador não autenticado ou com o token de autenticação expirado tentar realizar a operação.
- **HTTP 500 Internal Server Error** — Retornado em caso de erro inesperado durante a remoção.

I

Programa ASP para atribuição de slots

Código Fonte I.1: slotsAssignment.lp

```
% determine session participants
% create sessionParticipants for arguers and advisers
sessionParticipants(Arguer,PresentationID):- jury(PresentationID,Arguer,_).
sessionParticipants(Adviser,PresentationID):- jury(PresentationID,_,Adviser).

% precisely one slot per session
1 { assign(PresentationID,SlotID) : slot(SlotID) } 1 :- jury(PresentationID,_
    ,_).

% at most one session per slot per prof
:- 2{assign(PresentationID,SlotID):sessionParticipants(Participant,
    PresentationID)}, slot(SlotID), sessionParticipants(Participant,_).

% overall room limit
:- N{assign(PresentationID,SlotID)},slot(SlotID),N=Rooms+1, numberRooms(Rooms
    ).

% output configuration for occupied
slots(SlotID,PresentationID,Arguer,Adviser) :- assign(PresentationID,SlotID),
     jury(PresentationID,Arguer,Adviser).

% ----------Optimization---------

% accept any followedSlots order
followedSlots(SlotID1,SlotID2):- followedSlots(SlotID2,SlotID1).

%create participates predicates
participates(Participant,SlotID) :- assign(PresentationID,SlotID),
    sessionParticipants(Participant,PresentationID).

% find participant with followed presentations
```

```
27 conn(Participant,SlotID1) :- participates(Participant,SlotID1), followedSlots
       (SlotID2,SlotID1), participates(Participant,SlotID2).
28
29
30 % reduce number of single presentations
31 #minimize{1,Participant,SlotID:not conn(Participant,SlotID),participates(
       Participant,SlotID)}.
32
33 #show slots/4.
```

II

Programa ASP para atribuição de salas

Código Fonte II.1: roomsAssignment.lp

```
% Rooms assignment

1{room(PresentationID,RoomId):room(RoomId)}1:- slots(_,PresentationID,_,_).

% max one room per slot
:- 2{room(PresentationID,RoomId):slots(SlotID,PresentationID,_,_)},slots(
    SlotID,_,_,_),room(_,RoomId).

% max total rooms per slot
:- N{room(PresentationID,RoomId):slots(SlotID,PresentationID,_,_)},slots(
    SlotID,_,_,_),N=Rooms+1, numberRooms(Rooms).


%symmetry-breaking

roomSlot(RoomId,SlotID):- room(PresentationID,RoomId),slots(SlotID,
    PresentationID,_,_).

:- not followedSlots(SlotID,_), room(Presentation1,Room1),room(Presentation2,
    Room2),slots(SlotID,Presentation1,_,_), slots(SlotID,Presentation2,_,_),
    Presentation1>Presentation2,Room1<Room2.
:- not followedSlots(SlotID,_), roomSlot(Room1,SlotID), not roomSlot(Room2,
    SlotID), room(Room2),Room1>Room2.

room(Presentation2,RoomId):- room(Presentation2,RoomId), followedSlots(Slot1,
    Slot2),slots(Slot1,Presentation2,Arguer,Adviser),slots(Slot2,
    Presentation2,Arguer,Adviser).

```

```
21 1{room(Presentation2,Room);room(Presentation3,Room)}1 :- room(Presentation1,
       Room), followedSlots(Slot1,Slot2), slots(Slot1,Presentation1,Arguer,
       Adviser), slots(Slot2,Presentation2,_,_),juryMember(Arguer,Presentation2)
       ,slots(Slot2,Presentation3,_,_),juryMember(Adviser,Presentation3),
       Presentation2>Presentation3.
22
23 room(Presentation2,Room) :- room(Presentation1,Room), followedSlots(Slot2,
       Slot1), slots(Slot1,Presentation1,_,_), juryMember(P1,Presentation1),
       slots(Slot2,Presentation2,_,_),juryMember(P1,Presentation2), juryMember(
       P2,Presentation1), not participates(P2,Slot2), juryMember(P3,
       Presentation2), not participates(P3,Slot1).
24
25 juryMember(Participant,PresentationID):- slots(SlotID,PresentationID,
       Participant,_).
26 juryMember(Participant,PresentationID):- slots(SlotID,PresentationID,_,
       Participant).
27
28 participates(Participant,SlotID) :- slots(SlotID,_,Participant,_).
29 participates(Participant,SlotID) :- slots(SlotID,_,_,Participant).
30
31
32 % improved
33 #minimize{1@1,RoomId,Slot1:roomSlot(RoomId,Slot1),roomSlot(RoomId,Slot2),
       followedSlots(Slot2,Slot1)}.
34
35 % find room switches and minimize them
36
37 #minimize { 1@2,Participant,Presentation1:juryMember(Participant,
       Presentation1),juryMember(Participant,Presentation2),slots(Slot1,
       Presentation1,_,_),slots(Slot2,Presentation2,_,_), followedSlots(Slot1,
       Slot2),Presentation1!=Presentation2,room(Presentation1,Room1),room(
       Presentation2,Room2),Room1!=Room2 }.
38
39 % add room to slots
40 finalSlots(RoomID,SlotID,PresentationID):- slots(SlotID,PresentationID,_,_),
       room(PresentationID,RoomID).
41
42 #show finalSlots/3.
```

III

Documento apresentado nos Testes de Usabilidade

Briefing

Bem-vindo ao teste de usabilidade da nossa aplicação. O objetivo deste teste é avaliar a facilidade de uso e a intuição da interface da aplicação.

Durante o teste, será pedido para realizar algumas tarefas com diferentes tipos de utilizador: Utilizador comum (User), Organizador (Organizer) e Administrador (Admin). As credenciais necessárias para cada função serão fornecidas ao longo do teste.

Temos também um formulário que gostaríamos que respondesse enquanto realiza as operações.

Link para o formulário: https://forms.gle/Lm8M2u6stiQXj6cr8

Por motivos de força maior a aplicação ainda não está hospedada no domínio final. Os testes realizar-se-ão no link indicado abaixo, sendo necessário "Ir para 193.136.122.105 (não seguro)".

Link para aceder à aplicação: https://193.136.122.105/

Cenários USER

Imagine que é um professor e tem duas apresentações de documentos de preparação de dissertações para serem agendadas. As suas credenciais para aceder à aplicação são as seguintes:

Email: user2@gmail.com
Password: MinhaPassword1234

1. Imagine que no dia 27 de Novembro de 2024 tem um compromisso das 14h às 16h e, por isso, não pode ter apresentações marcadas nesse dia a essas horas. Adicione essas restrições na aplicação.
2. Imagine que pretende consultar as apresentações nas quais é o orientador do documento a ser apresentado.
3. Imagine que tem a necessidade de ter as apresentações cujos títulos das dissertações são "11111 Title example" e "77777 Title example" marcadas de forma consecutiva. Marque estas apresentações de forma consecutiva na aplicação.

4. Imagine que na sua apresentação “11111 Title example” participa um elemento externo à faculdade. Esse elemento tem um compromisso e não consegue estar presente na apresentação caso ela seja marcada no dia 28 de Novembro de 2024. Adicione estas restrições para a apresentação referida.
5. Imagine que pretende alterar a password da sua conta para “NovaPassword123”.

Cenários ORGANIZER

Imagine que é responsável por organizar todas as apresentações dos documentos de preparação das dissertações este ano. As suas credenciais para aceder à aplicação são as seguintes:

Email: organizer@gmail.com
Password: Mpassword123

1. Imagine que pretende fazer o download para um documento “PDF” de todas as apresentações registadas na aplicação.
2. Imagine que é necessário adicionar a sala número 1 do edifício 2 na aplicação.
3. Imagine que é necessário adicionar 5 slots com os horários 9:00, 10:00, 11:00, 12:00 e 14:00 no dia 29 de Novembro de 2024.
4. Imagine que é necessário adicionar um slot no dia 26 de Novembro às 11h30.
5. Imagine que precisa de adicionar a seguinte apresentação na aplicação:
    a. Título da dissertação: “Facilitação do uso de ASP para a criação de horários”
    b. Nome do estudante: “Salvador Silva”
    c. Número do estudante: 54597
    d. Nome do Orientador: “User 2”
    e. Nome do Vogal: ”User 3”
6. Imagine que pretende adicionar um ficheiro em formato excel com as apresentações. (O ficheiro será fornecido)
7. Imagine que pretende apagar a apresentação que criou com o nome “Facilitação do uso de ASP para a criação de horários”.
8. Imagine que pretende adicionar o utilizador:

    a. Nome do utilizador: “Salvador Silva”
    b. Email do utilizador: sgl.silva@campus.fct.unl.pt
9. Imagine que pretende iniciar a computação da alocação de salas e horários às apresentações registadas na aplicação.

Cenários ADMIN

Imagine que é um administrador da nossa aplicação. As suas credenciais para aceder à aplicação são as seguintes:

Email: admin@gmail.com
Password: PasswordAdm12

1. Imagine que pretende promover o utilizador “User 7” a organizador da aplicação.

Conclusão

O teste está concluído. Agradecemos os seus comentários no formulário. Muito obrigado por participar!