

3º ano da Licenciatura em Ciências da Computação

**Plataforma para Realização e Avaliação de Jogos**

**Terapêuticos**

Relatório de Desenvolvimento

Bruno Alves  
(a85481)

Daniel Ferreira  
(a85670)

Francisco Oliveira  
(a82066)

Ricardo Cruz  
(a86789)

23 de julho de 2021

# Agradecimentos

Gostaríamos de começar por agradecer a todas as pessoas que nos ajudaram na realização deste trabalho. Agradecer ao professor Pedro Rangel Henriques e também à nossa colega Rafaela Pinho pelo acompanhamento dado ao longo do semestre. Queríamos também agradecer ao Eng. António Aragão, técnico do Departamento de Informática, pela disponibilidade imediata em nos ajudar com todos os problemas técnicos que tivemos com o servidor e pela prontidão em atender aos nossos pedidos.

## **Resumo**

O Centro Neurosensorial de Braga (CNB) realiza tratamentos de memória e concentração em crianças e seniores para ajudar a melhorar as capacidades funcionais de nível cognitivo e comportamental, motor e emocional. Periodicamente, são realizados exames para avaliar a evolução dos pacientes e adequar o seu tratamento. Estes exames consistem em o paciente dizer, o mais rápido que conseguir, uma sequência de cores ou formas geométricas, dando o menor número de erros possíveis. Neste contexto, o projeto teve como intuito a criação de uma plataforma *WEB* onde estas provas pudessem ser executadas. O principal objetivo é automatizar esta tarefa, tornando mais eficiente e simples para os profissionais do centro, uma vez que, quando se realizam estes testes, têm de estar atentos a vários fatores em simultâneo. Após uma primeira fase de levantamento de requisitos, passou-se então à conceção da nossa solução. Primeiramente, implementamos uma base de dados que correspondesse às necessidades da aplicação. Posteriormente, desenvolvemos o código necessário para que o servidor pudesse responder a todos os pedidos feitos pelo utilizador, através de uma interface *WEB*. O principal desafio foi elaborar um algoritmo de verificação de resultados que se aproximasse o mais possível da realidade, visto que há um número vasto de situações possíveis de acontecer. Também houve a preocupação de tornar a aplicação o mais *user-friendly* possível, para facilitar a sua utilização.

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>5</b>
1.1	Contextualização . . . . .	5
1.2	Estrutura do Relatório . . . . .	5
<b>2</b>	<b>Análise e Especificação</b>	<b>7</b>
2.1	Descrição informal do problema . . . . .	7
2.2	Especificação de Requisitos . . . . .	7
<b>3</b>	<b>Concepção/desenho da Solução</b>	<b>9</b>
3.1	<i>Backend</i> . . . . .	9
3.1.1	Estrutura da solução . . . . .	9
3.1.2	Estruturas de Dados . . . . .	9
3.2	Algoritmos . . . . .	11
3.2.1	Avaliação da resposta dada pelo paciente . . . . .	11
3.2.2	Tratamento de texto . . . . .	13
3.3	<i>Frontend</i> . . . . .	14
3.3.1	Pacientes . . . . .	14
3.3.2	Exames . . . . .	15
3.3.3	Realizar Teste . . . . .	15
3.4	<i>API Speech-To-Text</i> . . . . .	17
3.5	Servidor . . . . .	17

<b>4 Codificação e Testes</b>	<b>18</b>
4.1 Alternativas, Decisões e Problemas de Implementação . . . . .	18
4.2 Testes realizados e resultados . . . . .	19
<b>5 Conclusão</b>	<b>20</b>
<b>A Tratamento de Texto</b>	<b>21</b>
<b>B Algoritmo de Avaliacao de Exames</b>	<b>23</b>
<b>C Diferentes Interfaces ao longo do desenvolvimento</b>	<b>25</b>

# **Lista de Figuras**

3.1	<i>Esquema da Base de Dados</i>	9
3.2	Pacientes	14
3.3	Exames	15
3.4	Realizar Teste	16
C.1	Interface 1.0	25
C.2	Interface 2.0	26
C.3	Interface 2.1	26
C.4	Interface 3.0	27
C.5	Interface Final	28
C.5	Interface Final	29

# Listings

3.1	Palavras corretas . . . . .	12
3.2	Palavras repetidas, omitidas, permutadas e erradas . . . . .	12
A.1	Tratamento de Dados . . . . .	21
B.1	Algoritmo de Avaliacao de Exames . . . . .	23

# Capítulo 1

## Introdução

### 1.1 Contextualização

Este trabalho foi realizado como projeto final do curso de Ciências da Computação, e o seu objetivo consistiu na criação de uma plataforma para a realização e avaliação de jogos terapêuticos. No meio de várias propostas, a escolha deste tema foi deliberada, unanimemente, por dois principais motivos. Um deles foi o fato de todos os elementos do grupo acharem interessante o desafio de criar uma aplicação *WEB*, uma vez que nos permitiria entender melhor todo o processo de desenvolvimento da aplicação, assim como a oportunidade de trabalhar com ferramentas e tecnologias novas, e conciliá-las com os conhecimentos adquiridos ao longo do curso. O outro motivo recaiu sobre a instituição que iria fazer uso da aplicação *WEB*. Acreditamos que o fato de o nosso trabalho vir a ser usado e aplicado num contexto real pelo CNB, que diariamente ajuda um grande conjunto de pessoas, serviria como motivação extra.

O Centro Neurosensorial de Braga (CNB) faz tratamentos de memória e concentração em crianças e seniores, para ajudar a melhorar as habilidades funcionais de nível cognitivo e comportamental, motor e emocional. Periodicamente são realizados testes para avaliar a evolução dos pacientes e adequar o tratamento. Estes testes, impressos em papel e reunidos num caderno, são realizados manualmente, e os resultados registados também manualmente em fichas de papel. Mas, de forma a simplificar o processo de executar estas provas, criou-se uma plataforma *WEB*, com o intuito de automatizar estas provas e, consequentemente, torná-las mais simples e eficientes.

### 1.2 Estrutura do Relatório

Este relatório encontra-se dividido em 5 capítulos:

- No primeiro capítulo, apresenta-se uma contextualização do trabalho e a estrutura deste relatório.
- No segundo capítulo, encontra-se uma análise ao trabalho proposto e especificados os requisitos da aplicação.

- No terceiro capítulo, são exposta as partes do *backend* e *frontend* da plataforma, explicando a lógica do funcionamento bem como as decisões tomadas.
- Na quarto capítulo, são discutidos decisões e problemas assim como alguns dos testes realizados.
- No quinto e último capítulo, apresenta-se uma revisão e análise final ao trabalho realizado, expondo o que foi feito durante o semestre e o que poderá ser feito no futuro para melhorar a plataforma.

# **Capítulo 2**

## **Análise e Especificação**

### **2.1 Descrição informal do problema**

Uma vez que a realização de exames orais, cujo propósito é testar a capacidade de concentração dos pacientes, implica que uma pessoa responsável esteja atenta a vários fatores em simultâneo (tempo de realização, deteção de erros, e.t.c.), o objetivo deste projeto foi elaborar uma plataforma que servisse de apoio à realização destes mesmo exames, e conseguisse, através do reconhecimento de voz, captar o que foi dito pelo paciente, cronometrando o tempo e, ainda, sugerindo um número de erros ocorridos.

### **2.2 Especificação de Requisitos**

A aplicação desenvolvida teve de satisfazer os seguintes requisitos:

- Inserção e remoção de um paciente;
- Consulta de informação detalhada de um paciente;
- Download da informação de um paciente em formato *CSV*;
- Inserção e remoção de uma prova;
- Consulta da informação de um exame;
- Realização de uma prova;
- Avaliação da resposta dada;
- Consulta dos resultados das provas;

- Integração do reconhecimento de voz para realização de exames *API Speech To Text* da *Microsoft*;
- *Restart* de uma prova;
- Submissão da realização de uma prova para a base de dados;
- Aplicação *User friendly*

# Capítulo 3

## Concepção/desenho da Solução

### 3.1 Backend

#### 3.1.1 Estrutura da solução

Devido à nossa escolha em utilizar *React* para tratar do *frontend*, e *Flask* para tratar do *backend*, o primeiro passo fundamental tomado foi definir as *routes* e os métodos *GET* e/ou *POST* que os nossos pedidos ao servidor viriam a ter.

#### 3.1.2 Estruturas de Dados

Depois de discutidos e definidos os requisitos e as *routes*, o passo seguinte foi a elaboração de uma base de dados que pudesse servir de suporte às funcionalidades previstas para a plataforma. A base de dados concebida foi então a seguinte:

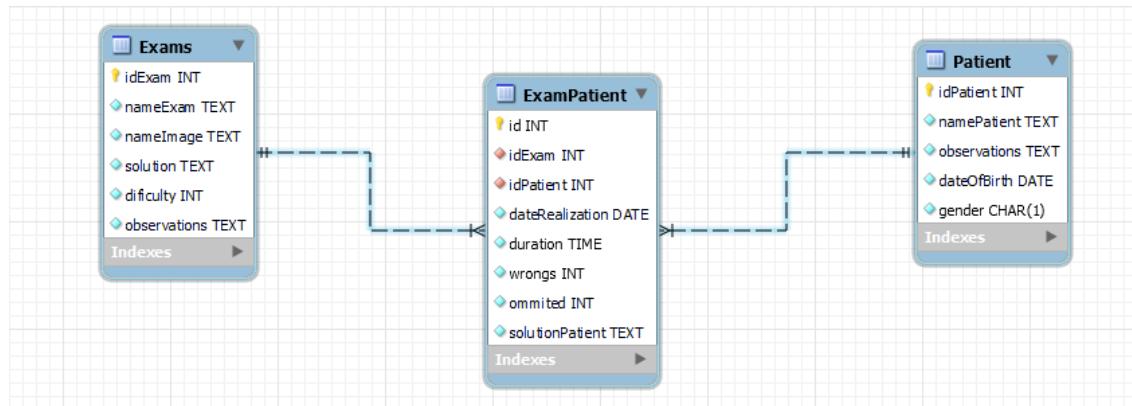


Figura 3.1: Esquema da Base de Dados

A tabela ***Exam*** representa os testes que os pacientes terão disponíveis para realizar. É composta pelas seguintes colunas:

- *idExam* que serve para distinguir cada exame;
- *nameExam* para colocar o nome que se ache adequado;
- *nameImage* que contém o *path* para a imagem do exame;
- *solution* com a solução do exame para posterior execução de um algoritmo de verificação;
- *difficulty* que representa o nível de dificuldade;
- *observations* onde podem ser colocadas observações que sejam relevantes.

A tabela ***Patient*** representa os pacientes da clínica. É composta pelas seguintes colunas:

- *idPatient* que serve para distinguir cada paciente;
- *namePatient* para colocar o nome do paciente;
- *observations* onde podem ser colocadas observações que sejam relevantes;
- *dateOfBirth* com a data de nascimento do paciente;
- *gender* com o género do paciente.

Uma vez que um exame pode ser realizado por vários pacientes e um paciente pode realizar vários exames, foi necessária a construção de uma tabela de ligação entre os pacientes e os exames, sendo essa a tabela ***examPatient***. Esta tabela irá conter toda a informação necessária e obtida numa realização de um exame. As suas colunas são as seguintes:

- *id* que serve para distinguir cada realização de exame;
- *idExam* chave estrangeira para poder associar o exame que foi feito;
- *idPatient* chave estrangeira para poder associar o paciente que fez o exame;
- *dateRealization* com a data de realização;
- *duration* com a duração da realização;
- *wrongs* com o número de erros feitos;
- *omitted* com o número de omissões feitas;
- *solutionPatient* com a resposta dada pelo paciente.

## 3.2 Algoritmos

### 3.2.1 Avaliação da resposta dada pelo paciente

Foi necessário desenvolver um algoritmo, para avaliação da resposta dada pelo paciente de forma a retornar as quantidades dos vários tipos de erros. Numa fase inicial, apenas se considerou 2 tipos de erros: **omissões** e **trocadas de cor**. No entanto, numa fase mais adiantada, optou-se por apresentar outros tipos de erros também, de modo a ajudar a perceber a resposta do paciente. Posto isto, os erros que foram considerados foram: **repetição de cor**, **permuta de cores** (trocar as posições de 2 cores), **troca de cores** (erro tradicional de dizer a cor errada), **omissões** e, por fim, **outros**, que representa qualquer outro tipo de erro que não os mencionados anteriormente.

Antes de se prosseguir com a explicação do algoritmo, ilustra-se abaixo as situações de erro possíveis:

#### Repetição

- Solução:  $x_{-1}, x_0, x_1, x_2$
- Resposta:  $x_{-1}, x_0, x_0, x_1, x_2$

#### Permuta

- Solução:  $x_{-1}, x_0, x_1, x_2$
- Resposta:  $x_{-1}, x_0, x_2, x_1$

#### Troca

- Solução:  $x_{-1}, x_0, x_1, x_2$
- Resposta:  $x_{-1}, x_0, x, x_2$

#### Omissão

- Solução:  $x_{-1}, x_0, x_1, x_2$
- Resposta:  $x_{-1}, x_0, x_2, x_3$

Quanto ao algoritmo em si, a estratégia seguida foi a de realizar uma comparação dos elementos da resposta dada e da solução, posição a posição. Pode-se então dividir o comportamento do algoritmo em dois cenários possíveis:

1. Num momento em que as palavras comparadas são iguais, avança-se na iteração de ambas as listas e coloca-se a palavra numa lista que representa a subsequência de elementos certos em que está:

---

```

1   if sol[i] == ans[j]:
2       currsubseq.append(ans[j])
3       i = i + 1
4       j = j + 1

```

---

Listing 3.1: Palavras corretas

2. Quando essa comparação retorna falso, tenta-se perceber que tipo de erro é que ocorreu. Nota para o facto de a ordem de verificação ser: repetição, permuta, troca, omissão e outro.

---

```

1      #REPETICAO
2      if i > 0 and (ans[j] == ans[j - 1] and ans[j + 1] == sol[i]):
3          j = j + 1
4          repeated = repeated + 1
5          flag=1
6
7      #PERMUTA
8      elif i < len(sol) - 1 and j < len(ans) - 1 and (sol[i + 1] == ans[j]
9      and sol[i] == ans[j + 1]):
10         i = i + 2
11         j = j + 2
12         wrongs = wrongs + 2
13         exchange = exchange + 1
14         flag=1
15
16      #TROCA
17      elif i < len(sol) - 1 and j < len(ans) - 1 and (sol[i + 1] == ans[j + 1]):
18          i = i + 1
19          j = j + 1
20          wrongs = wrongs + 1
21          flag=1
22
23      #OMISSAO
24      if(flag == 0):
25          for h in range(3+1):
26              if i < len(sol) - h and sol[i + h] == ans[j] and flag ==0:
27                  i = i + h
28                  ommited = ommited + h
29                  flag=1
30
31      if(flag == 0):
32          #app.logger.info("Entrou num caso nao coberto")
33          other = other + 1
34          i = i + 1
35          j = j + 1

```

---

Listing 3.2: Palavras repetidas, omitidas, permutadas e erradas

A criação deste algoritmo foi, talvez, a etapa mais complicada da realização deste trabalho, uma vez que vai avaliar algo que foi dito pelo paciente. Rápido se percebeu que existe um número praticamente ilimitado de situações diferentes e possíveis de acontecer. Dessa forma, o objetivo passou então por desenvolver algo que desse resultados aceitáveis e permitisse um posterior ajuste dos valores, antes destes serem colocados na base de dados.

### **3.2.2 Tratamento de texto**

Uma vez que se está a lidar com pessoas e a recorrer a uma *API Speech To Text*, há sempre a possibilidade de algo não funcionar corretamente. Por exemplo, o paciente dizer alguma palavra que não entre no vocabulário de palavras relacionadas com o teste, ou até a própria *API* reconhecer mal (apesar de pouco provável, pode acontecer). Posto isto, sentiu-se uma necessidade de se criar uma função que fizesse um tratamento ao texto dito pelo paciente, antes da execução do algoritmo falado anteriormente. Esta função (exposta no apêndice A) vai filtrar o que foi dito pelo paciente, para que as únicas palavras que "passem" sejam cores e/ou formas geométricas visto serem as únicas palavras relevantes para o exame.

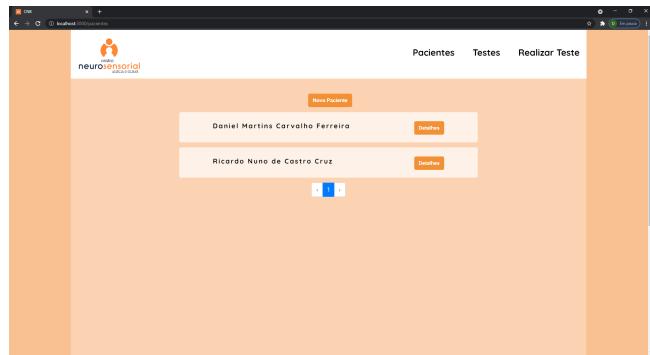
### 3.3 Frontend

Tendo em conta o utilizador final para a aplicação, foi necessário desenvolver uma interface gráfica *user-friendly* de maneira a que o utilizador consiga interagir de uma forma simples e prática com a aplicação. Esta interface tem como propósito, não só a interação com o utilizador, mas também permitir a comunicação através de *routes* com o *backend*.

De forma a simplificar o uso da aplicação, foi tomada a decisão de se dividir a aplicação *WEB* em apenas três páginas principais: Pacientes, Testes e Realizar Teste.

#### 3.3.1 Pacientes

Nesta página, é possível adicionar um paciente, através do botão "Novo Paciente", que redireciona para uma nova página em que é possível preencher toda a informação necessária para a criação de um paciente. Existe, também, a possibilidade de se encontrar todos os pacientes inscritos nesta plataforma. A cada paciente é atribuído um botão de "Detalhes", que tem como objetivo mostrar, detalhadamente, toda a informação inserida inicialmente na criação de cada paciente, assim como os últimos 3 exames efetuados. Por fim, com o botão "Transferir CSV", encontrado nos detalhes do paciente, é possível fazer o *download* de todos os exames que o utente realizou, assim como a classificação dos mesmos.



(a) Lista Pacientes

(b) Adicionar Paciente

(c) Detalhes do Paciente

Figura 3.2: pacientes

### 3.3.2 Exames

A esta página está associada uma listagem com os exames criados para serem utilizados para a avaliação dos pacientes. Cada exame tem um botão de "Detalhes" que permite expor, em detalhe, toda a informação desse mesmo exame. Tal como no cliente, é ainda possível encontrar um botão "Novo Exame" que permite a criação de um novo exame, de acordo com os requerimentos propostos. Um destes é selecionar uma imagem para o teste, à qual nós optamos por usar a plataforma *Firebase* para guardar todas as imagens e, assim, evitar usar armazenamento local.

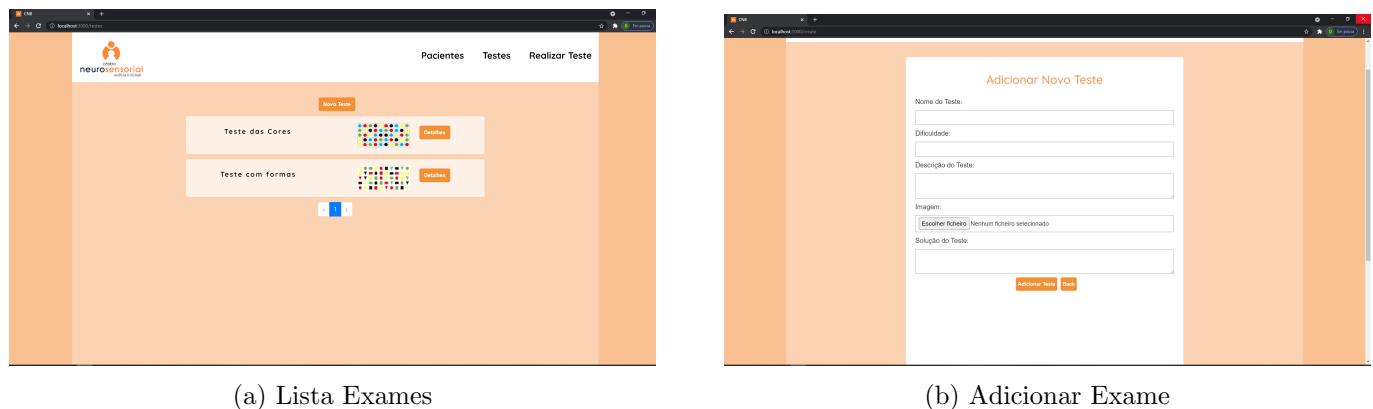


Figura 3.3: Exames

### 3.3.3 Realizar Teste

Nesta página, o(a) doutor(a) poderá escolher qual o teste que o seu paciente irá realizar, bem como o nome do utente. Na escolha do teste, irá aparecer uma *preview* com a imagem do teste escolhido, e uma caixa de texto denominada de "tentativa", que irá ter a resposta do utente ao exame efetuado.

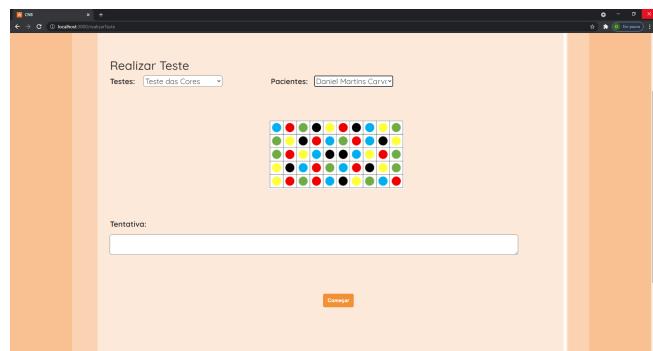
O botão "começar" tem como propósito começar o exame. Após carregar em "começar", é exibido um *pop-up*, de maneira a informar que pode começar a falar após o fecho desse *pop-up*. Numa primeira utilização, vai pedir para permitir a utilização do microfone e a escolha do microfone que pretende utilizar por parte do *browser*. Uma vez permitido, não irá mais aparecer esse aviso de permissão. Posto isto, o utente pode efetuar o exame.

Quando o paciente termina o exame, o(a) doutor(a) deverá carregar no stop e, se tudo correr conforme o planeado, irá aparecer a resposta do paciente na área de texto acima mencionada. Em seguida, deverá então carregar no botão de "avaliar", para o algoritmo supracitado verificar se a resposta do utente vai de acordo com a solução do exame (inserida na criação do teste).

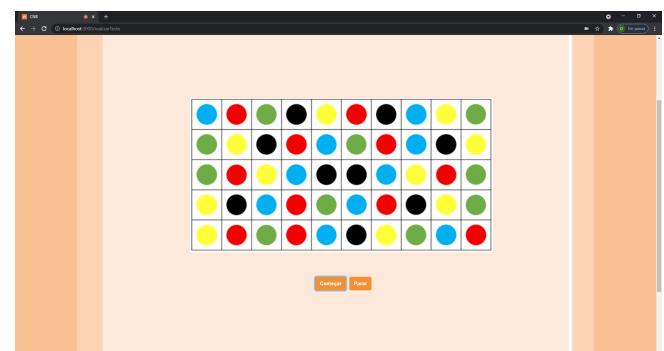
Após a avaliação por parte do algoritmo, aparecerá uma nova página com o resultado final, isto é, o número de palavras omitidas, trocadas, permutadas e erradas que o paciente teve com a sua resposta. Nessa mesma página, é também possível encontrar a imagem do teste, a resposta do utente, assim como um botão "mostrar tabela". Esta tabela, irá conter as maiores subsequências que o paciente teve até omitir, trocar, permutar ou errar uma palavra.

Tal como descrevemos, na eventualidade de o algoritmo não conseguir realizar uma avaliação perfeita em todos os casos possíveis a que a própria avaliação está sujeita, deve-se a alguns fatores que muitas vezes não são possíveis de controlar e/ou antecipar. Como tal, a melhor solução encontrada foi a de usar o algoritmo para dar uma sugestão de resultados. Se o(a) doutor(a) verificar que houve mais (ou menos) omissões, permutações, trocas ou erradas, poderá alterar o valor das mesmas.

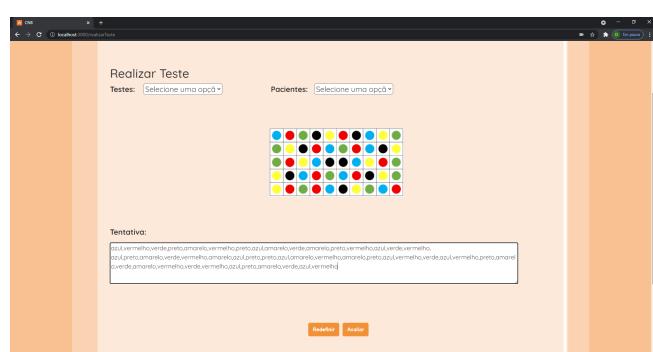
Após o(a) doutor(a) terminar de confirmar e corrigir os resultados deverá então carregar em "submiter", de modo a inserir a informação na base de dados e de maneira a que, *a posteriori*, apareça na página "Detalhes do paciente", o exame realizado com o dia em que ocorreu, a duração e os erros que teve.



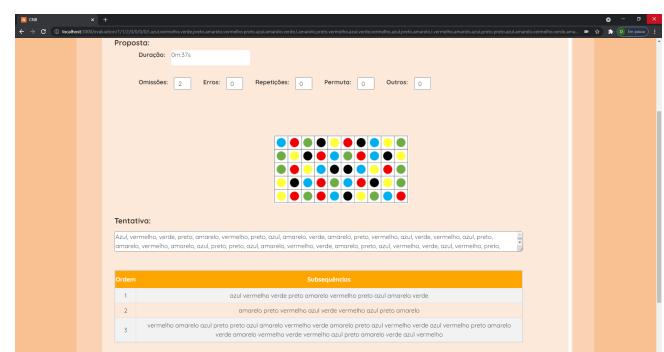
(a) Realizar Teste



(b) Realização do Teste



(c) Teste Realizado



(d) Avaliação do Teste

Figura 3.4: Realização e Avaliação do Teste

### 3.4 API Speech-To-Text

Um dos objetivos deste trabalho é facilitar a realização do teste do utente, isto é, o utente só precisa de falar para o microfone e o que ele falar é reconhecido e guardado, para posteriormente ser usado no algoritmo de avaliação. Para isto, foi necessário recorrer a uma **API Speech-To-Text**.

Após alguma investigação e operacionalização de testes à procura de qual *API* reconhecia melhor as palavras ditas, chegamos à conclusão que existiam duas *API's* muitíssimo boas e que se adequavam aos nossos requisitos, sendo elas a *API Speech-To-Text* da *Google* e da *Microsoft*.

Uma vez que a Universidade tem parceria com a *Microsoft*, com a opinião do Orientador, decidimos optar pela da *Microsoft*.

Inicialmente, esta *API* foi implementada no *backend*. Mas, rapidamente verificamos que não era a melhor opção pois, ao utilizar no *backend* iria utilizar o microfone do próprio portátil e, ao usar com o *Docker* (*container* como se fosse uma máquina de virtualização) percebemos que não funcionaria corretamente, pois esse *container* não tem acesso ao microfone local.

Para resolver esta situação, voltamos a reler a documentação da *API Speech-To-Text* e verificamos que constava lá que esta *API* tinha de ser implementada no *frontend* (*browser*), isto porque é o *browser* que vai pedir a permissão ao utilizador para utilizar o microfone.

Desta forma, foi implementada com sucesso esta tecnologia.

### 3.5 Servidor

De modo a termos a nossa aplicação WEB operacional e acessível ao ”público” através de um URL, foi necessário, por recomendação do Orientador, falar com o Sr. António Aragão e solicitar-lhe a ajuda necessária, no sentido de podermos colocar a nossa aplicação no servidor do Departamento de Engenharia Informática, na Universidade do Minho.

Após várias conversações com o Sr. António, foi preciso utilizar a tecnologia *Docker*, de modo a utilizar *containers* como se fossem máquinas virtuais. Esta tecnologia visa facilitar o uso desta aplicação em diferentes sistemas operativos. Ao utilizar um *container*, baseado em linux, não é preciso instalar as dependências na máquina local, pois irá instalar dentro desse *container*, e depois basta correr esse container.

Posto isto, foi possível meter a nossa aplicação no servidor. Mas, ao testar a aplicação no servidor, verificamos uma contrariedade: os *browsers* conhecidos, como o *Chrome*, *Firefox* e *Safari*, têm um protocolo de segurança apertado, e visto que o servidor operava sobre um domínio *http*, e como a *API Speech To Text* precisa da utilização do microfone, os *browsers* não permitiam que fosse pedido a autorização do microfone ao utilizador, uma vez que verificavam que o site não era seguro.

Depois de algumas pesquisas, verificamos que para ser possível o uso do microfone, era necessário que o nosso site tivesse um certificado *SSL*, isto é, que fosse *https*.

# Capítulo 4

## Codificação e Testes

### 4.1 Alternativas, Decisões e Problemas de Implementação

Uma das vertentes do trabalho que mais questões levantou ao longo do semestre foi a interface de utilizador e o seu visual. Desde a fase inicial do desenvolvimento da aplicação que se apresentou possíveis *designs* para a interface de utilizador. No entanto, com o passar do tempo, surgiram novos problemas que obrigavam a alterações na interface e, por algumas vezes, foi mesmo necessário haver uma reconstrução por completo da interface. Algumas imagens das diferentes interfaces podem ser encontradas no apêndice C.

Ainda, em relação à interface, o grupo deparou-se com mais um problema: ao utilizar valores fixos para definir os tamanhos dos diversos componentes da página, manifestavam-se situações em que, apesar de o resultado ser bom à primeira vista, se a aplicação *web* fosse aberta em diferentes máquinas com ecrãs de diferentes tamanhos e resoluções, o resultado obtido não seria o esperado. Mais, ainda, em alguns casos a aplicação ficava totalmente inutilizável. Este problema obrigou à utilização de valores percentuais em prol de valores fixos, de forma a que o aspeto resultante da aplicação *web* fosse proporcional ao tamanho do ecrã da plataforma em que a aplicação fosse aberta.

Tal como previamente mencionado, outro problema encontrado no decurso deste processo prendeu-se com as situações que o algoritmo de correção de exames não cobria e, como tal, não apresentava resultados fiáveis. Este problema conduziu, então, à tomada de decisão em se considerar que o algoritmo apenas daria sugestões de correção que pudessem ser posteriormente reavaliadas por um dos profissionais do CNB.

Para além destas situações, houve ainda a escolha da *API Speech-to-Text* e a implementação da aplicação *web* num servidor que, como já mencionado acima, levantaram problemas e questões. De facto, a implementação da aplicação num servidor obrigou a requisitar ajuda externa, sendo sempre realçada pelo orientador a importância deste passo, visto que é um cenário comum a aplicação começar a apresentar novos problemas assim que começa a funcionar num servidor.

## 4.2 Testes realizados e resultados

Ao longo do desenvolvimento deste projeto, foram desenvolvidos diversos testes. Alguns dos testes realizados consistiram apenas em navegar pela aplicação, com a finalidade de observar e avaliar a interface de utilizador. Ocorreu uma atenção redobrada, para garantir que a interface fosse o mais prática e simples possível, de modo a aumentar a produtividade da aplicação *WEB* quando a mesma fosse usada pelo CNB. Os testes realizados permitiram também verificar que todas as funcionalidades da aplicação *WEB* se encontravam a funcionar devidamente.

A realização de exames na aplicação foi, também, fortemente testada. Os primeiros testes realizados a esta vertente da aplicação consistiram em garantir que o reconhecimento de voz funcionasse corretamente para as mais diversas situações, uma vez que, de paciente para paciente, a velocidade com que o paciente fala, assim como o seu sotaque, pode variar bastante. Os testes foram feitos com um número de pessoas consideravelmente alto. A cada pessoa era pedido que falasse o mais naturalmente possível, de maneira a que os resultados obtidos fossem equivalentes com os resultados a obter quando a aplicação for utilizada pelo CNB no dia a dia. Posteriormente, os testes começaram a englobar também a avaliação de exames, de maneira a verificarmos o nível de precisão do algoritmo, e em que situações este falhava. Deste modo, foi possível otimizar o algoritmo consideravelmente.

## Capítulo 5

### Conclusão

Com a conclusão deste projeto, é possível afirmar, em retrospectiva ao trabalho realizado ao longo do semestre, que o grupo concretizou o desafio esperado: construir uma aplicação *WEB* a partir do zero. No decurso do processo de desenvolvimento, foi necessário mobilizar as diferentes bases que os membros tinham apreendido ao longo do curso e conjugá-las com uma aprendizagem praticamente diária de novos conceitos, tecnologias e ferramentas.

O envolvimento na concepção deste projeto contribuiu, fortemente, para que todos os elementos do grupo entendessem melhor como funciona o ciclo de desenvolvimento de uma aplicação *WEB*. Os múltiplos cons trangimentos e desafios que nos puseram à prova, como por exemplo, elaborar requisitos que fossem de encontro ao que o CNB precisasse, alterar ao centímetro a interface gráfica, pôr a aplicação *WEB* a funcionar num servidor, constituíram-se como vetores fundamentais para a consolidação e progressão no conhecimento.

Para além destes desafios mais técnicos, devido à dimensão do projeto, foi também necessário realizar uma divisão do trabalho pelos quatro membros que integram o grupo, com a premissa de comunicação frequente e assídua entre todos, de forma a articular e coordenar as tarefas assumidas e concretizadas por cada um, apoiando-se mutuamente.

Consideramos que a aplicação *WEB* desenvolvida possui uma interface gráfica com capacidade para permitir uma utilização fácil e prática, e, concomitantemente, permitir agilizar o processo de realização de exames, assim como a consulta de informações de pacientes e exames. Apesar das dificuldades encontradas na correção de exames, o grupo julga e atesta que os resultados apresentados pelo algoritmo de correção desenvolvido para aplicação *WEB* já são bastante expressivos. Contudo, subsistem variáveis incontroláveis, pois é para aplicar a pessoas e, como tal, existe a possibilidade de imponderáveis oriundas de personalidades distintas, que dificultam o procedimento no ato da aplicação deste instrumento.

Em síntese, todos os elementos do grupo estão agrados com o trabalho realizado, visto que o resultado final é uma aplicação *WEB* funcional, simples e prática, que cumpre e corresponde aos requisitos propostos, e se encontra a funcionar num servidor, podendo ser utilizada pelo Centro Neurosensorial de Braga a qualquer momento, de forma a economizar tempo e evoluir na utilização de ferramentas de índole tecnológica.

No futuro, se o CNB assim entender, será possível melhorar a aplicação *WEB* adicionando novas funcionalidades como por exemplo, comparação de testes, análise aprofundada do histórico do paciente recorrendo a gráficos, assim como otimizar o algoritmo de correção de testes.

# Apêndice A

## Tratamento de Texto

```
1 def removing_special_characters(text):
2     colours = ["azul", "amarelo", "vermelho", "verde", "preto", "roxo", "laranja", "branco"]
3     geometric_forms = ["tri ngulo", "ret ngulo", "quadrado", "c rculo"]
4
5     text = ''.join([str(elem) for elem in text]).lower()
6     if text[-1] != '.' or text[-1] != ' ':
7         text = text + '.'
8     text = text.split(',')
9     text = ','.join(text)
10    text = text.split('. ')
11    text = ', '.join(text)
12
13    aux = []
14    result = []
15    i = 0
16
17    while i < len(text):
18        #app.logger.info(i)
19        if text[i] != ' ':
20            aux.append(text[i])
21            i = i + 1
22        else:
23            if len(aux) > 0:
24                if aux[-1] == 'a':
25                    aux[-1] = 'o'
26                if aux[-1] == 's':
27                    aux.pop(-1)
28            textJoin = ', '.join(aux)
29
30            if textJoin in geometric_forms:
31                aux2 = []
32                i += 1
33                while i < len(text) and text[i] != ' ':
34                    aux2.append(text[i])
35                    i += 1
36
37                colour = ', '.join(aux2)
38
39                if colour in colours:
40                    res = textJoin + ' ' + colour
41
```

```
42         result.append(res)
43
44     else:
45         if textJoin == "azui":
46             textJoin = "azul"
47
48         if textJoin in colours:
49             result.append(textJoin)
50             i = i + 1
51     else:
52         i = i + 1
53     aux = []
54 return result
```

---

Listing A.1: Tratamento de Dados

## Apêndice B

# Algoritmo de Avaliacao de Exames

```
1 def algoritmo(sol, ans):
2     seqs = []
3     currsubseq = []
4     i = 0
5     j = 0
6     ommited = 0
7     wrongs = 0
8     repeated = 0
9     exchange = 0
10    other = 0
11
12
13
14    while i < len(sol) and j < len(ans):
15        flag=0
16        if sol[i] == ans[j]:
17            currsubseq.append(ans[j])
18            i = i + 1
19            j = j + 1
20
21    else:
22        if len(currsubseq) == 0:
23            #REPETICAO
24            if i > 0 and (ans[j] == ans[j - 1] and ans[j + 1] == sol[i]):
25                j = j + 1
26                repeated = repeated + 1
27                flag=1
28
29            #PERMUTA
30            elif i < len(sol) - 1 and j < len(ans) - 1 and (sol[i + 1] == ans[j] and sol
31 [i] == ans[j + 1]):
32                i = i + 2
33                j = j + 2
34                wrongs = wrongs + 2
35                exchange = exchange + 1
36                flag=1
37
38            #TROCA
39            elif i < len(sol) - 1 and j < len(ans) - 1 and (sol[i + 1] == ans[j + 1]) :
40                i = i + 1
41                j = j + 1
```

```

41             wrongs = wrongs + 1
42             flag=1
43
44             #OMISSAO
45             if(flag == 0):
46                 for h in range(3+1) :
47                     if i < len(sol) - h   and sol[i + h] == ans[j] and flag ==0:
48                         i = i + h
49                         ommited = ommited + h
50                         flag=1
51
52
53             if(flag == 0):
54                 #app.logger.info("Entrou num caso nao coberto")
55                 other = other + 1
56                 i = i + 1
57                 j = j + 1
58
59             else:
60                 currsubseq.insert(0, "!")
61                 seqs.append(currsubseq)
62                 currsubseq = []
63
64             if i < len(sol):
65                 ommited += len(sol) - i - 1
66                 currsubseq.insert(0, "!")
67                 seqs.append(currsubseq)
68
69             return (ommited, wrongs, repeated, exchange, other,seqs)

```

---

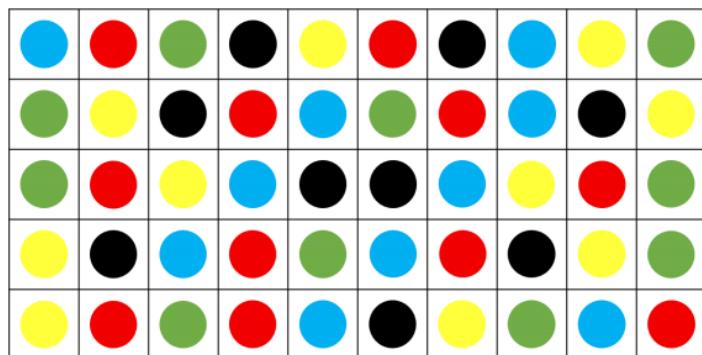
Listing B.1: Algoritmo de Avaliacao de Exames

## Apêndice C

# Diferentes Interfaces ao longo do desenvolvimento

Centro Neurosensorial Home Criar Teste

Primeiro teste



1

Descrição para o primeiro teste

Figura C.1: Interface 1.0

Figura C.2: Interface 2.0

Figura C.3: Interface 2.1

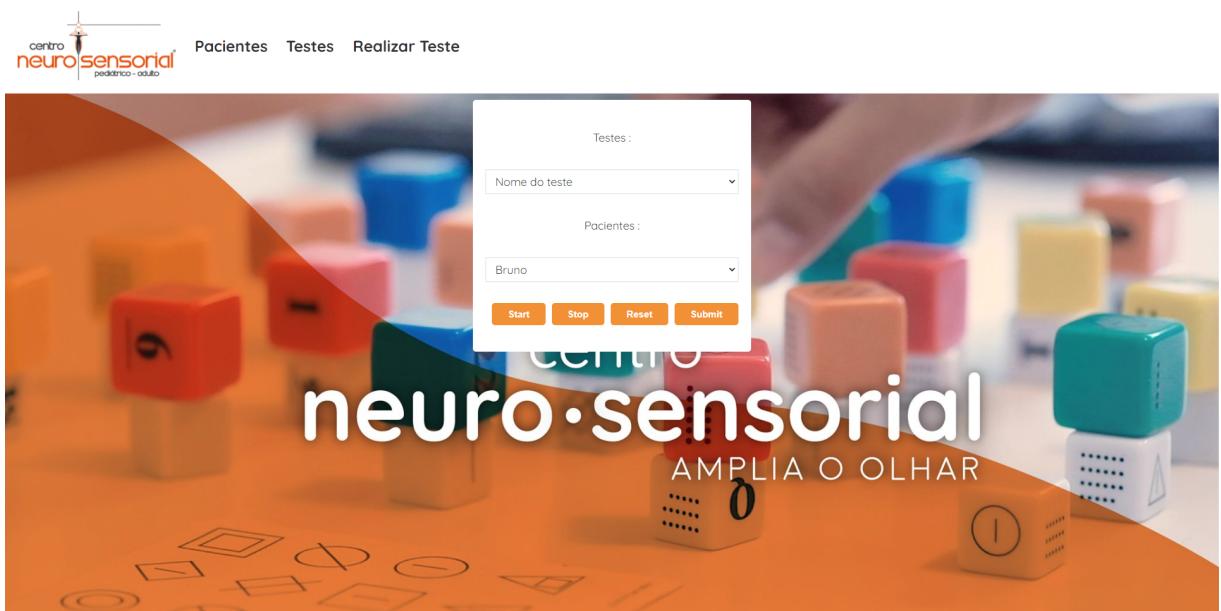


Figura C.4: Interface 3.0

The figure consists of two screenshots of a web application interface for 'centro neurosensorial'.

**Top Screenshot (Realizar Teste Page):**

- Header: 'Pacientes', 'Testes', 'Realizar Teste'.
- Left sidebar: 'centro neurosensorial' logo and text 'AUXÍLIO O OLHAR'.
- Form fields:
  - 'Testes': dropdown menu showing 'Teste das Cores'.
  - 'Pacientes': dropdown menu showing 'Daniel Martins Carvalho'.
  - 'Tentativa:' text input field.
- Middle section: A 4x12 grid of colored circles (blue, red, green, black, yellow) used for the color test.

**Bottom Screenshot (Pacientes Page):**

- Header: 'Pacientes', 'Testes', 'Realizar Teste'.
- Left sidebar: 'centro neurosensorial' logo and text 'AUXÍLIO O OLHAR'.
- Buttons:
  - 'Novo Paciente'
  - 'Daniel Martins Carvalho Ferreira' with 'Detalhes' button
  - 'Ricardo Nuno de Castro Cruz' with 'Detalhes' button
- Navigation:
  - Page number: '1'
  - Navigation icons: '<' and '>'

Figura C.5: Interface Final

**Adicionar Novo Paciente**

Nome do Paciente:

Data de Nascimento:

Género:

Masculino  
 Feminino  
 Outro

Observações:

**Ficha do Paciente**

Nome: Daniel Martins Carvalho Ferreira

Data de Nascimento: 1999-05-08

Género: M

Observações: Observações do paciente Daniel

Últimos 3 Testes:

Teste das Cores	Data de realização: 2021-06-23	Duração: 0m:1s	Erradas: 2	Omitidas: 4
Teste das Cores	Data de realização: 2021-06-23	Duração: 0m:1s	Erradas: 2	Omitidas: 1
Teste das Cores	Data de realização: 2021-06-23	Duração: 0m:3s	Erradas: 0	Omitidas: 2

**Pacientes** **Testes** **Realizar Teste**

**Voltar** **Transferir CSV** **Remover**

Figura C.5: Interface Final