FACULDADE DE CIÊNCIAS EXATAS E DA ENGENHARIA

Licenciatura em Engenharia Informática

Modificação do Sistema NeuroRehabLab Task Generator 2.0

Projeto/Estágio

Realizado por:

José Pita - 2080121@student.uma.pt

Orientador:

Dra. Ana Lúcia Faria

Co-Orientadores:

Profº Filipe Quintal

Profº Sergi Bermúdez i Badia

2023/2024

Índice

1 Introdução

Os problemas cognitivos podem apresentar uma variedade de dificuldades que afetam as funções mentais, que incluem memória, raciocínio, concentração, resolução de problemas e linguagem, prejudicando a vida cotidiana de uma pessoa. Esses problemas podem ser causados por uma série de fatores, englobando doenças neurológicas, lesões cerebrais traumáticas e transtornos de desenvolvimento.

Entre as doenças que afetam a cognição, a Doença de Alzheimer é a mais comum e devastadora, com 7.3 Milhões de cidadãos europeus sofrendo de uma das várias formas de Demência [1]. Esta doença neurodegenerativa é caracterizada pela perda progressiva da memória e outras funções cognitivas, estimando-se em Portugal a existência de cerca de 182 526 pessoas com Demência.[1].

O tratamento da doença pode incluir medicamentos para controlar os sintomas e atrasar a sua progressão, contudo atualmente não existe cura para a doença, apenas o alívio temporário de alguns sintomas [2].

Uma abordagem não farmacológica é o treino cognitivo, que envolve uma série de exercícios e atividades projetadas para melhorar as funções mentais, como memória, atenção, linguagem e funções executivas. Todavia, a sua implementação, como tarefas papel e lápis, é limitada devido às exigências em termos de tempo, custo e recursos humanos, carecendo de um psicólogo para conduzir os exercícios com base no perfil e desempenho do paciente [3]. Além disso, a locomoção do paciente para a instalação, exige disponibilidade e transporte [4].

As soluções baseadas nas Tecnologias de Informação e Comunicação (TIC), visam enfrentar estes desafios, oferecendo uma melhoria nos serviços clínicos, aumentando a intensidade e personalização das tarefas [5].

Uma das soluções é o *NeuroRehabLab Task Generator 1.0* [5], uma aplicação Web geradora de programas personalizados de treino cognitivo, com base em 11 tarefas de papel e lápis selecionadas em unidades clínicas públicas e privadas [6].

1.1 Descrição do problema

Embora o sistema *NeuroRehabLab Task Generator 1.0*, **Figura 1**, seja uma ferramenta gratuita e cumpra para a eficiência na geração de tarefas para um determinado paciente, levando em consideração as funções cognitivas, a sua execução apresenta desafios. As tarefas são exercícios e atividades projetadas para melhorar as funções cognitivas dos pacientes, como a memória, atenção, linguagem e funções executivas. Os exercícios incluem problemas matemáticos, sopa de palavras, labirintos, ordenação de ações, associação de imagens, entre outros, conforme ilustrado na **Figura 2**.

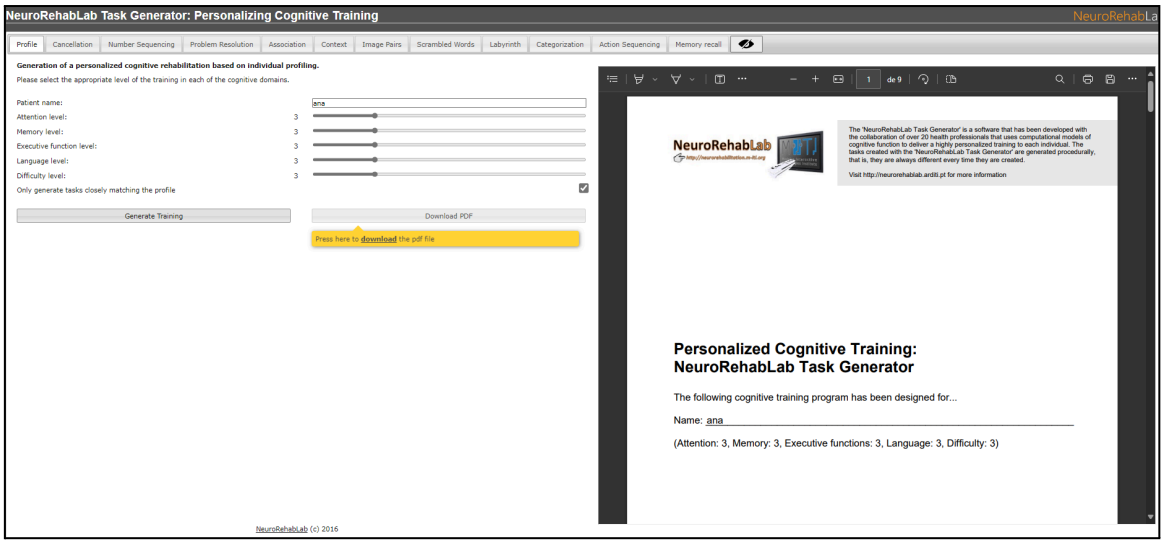

Figura 1 - NeuroRehabLab Task Generator 1.0 (tarefa Profile Task)

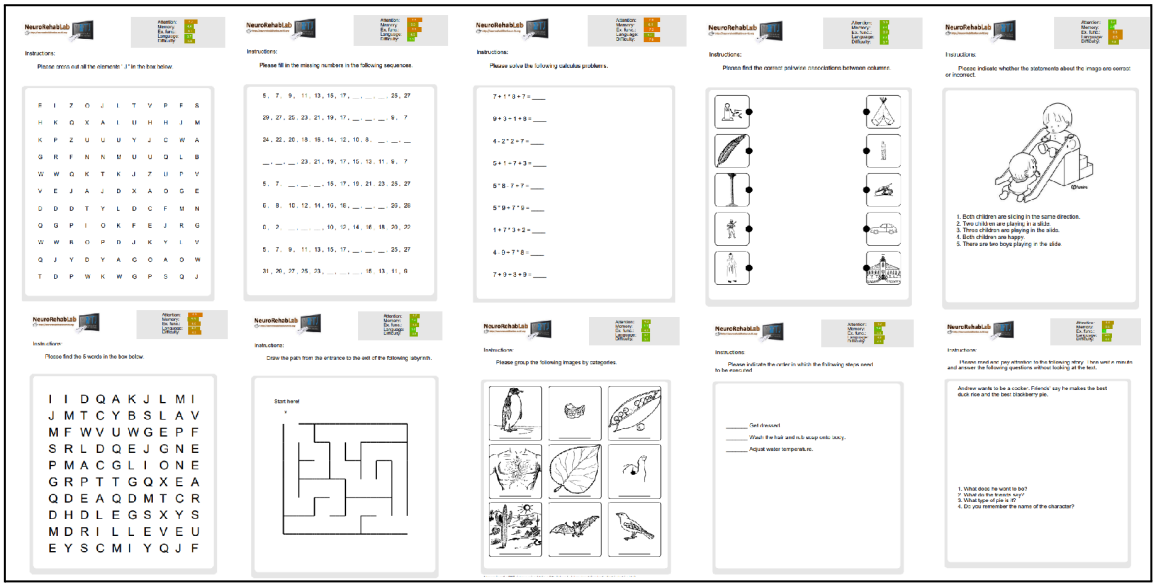

Figura 2 - Exercícios gerados pelo NeuroRehabLab Task Generator 1.0

No âmbito do *NeuroRehabLab Task Generator 1.0* as tarefas necessitam de ser impressas, requerendo de uma impressora o que algumas clínicas podem não ter disponível, na finalidade de obter-se a tarefa em papel para a sua prática. Além disso, necessitam de ser desempenhadas pelo paciente acompanhadas e examinadas por um profissional de saúde, requerendo a locomoção de um dos intervenientes, profissional de saúde ou paciente, para a sua operação. Este processo impõe, muitas das vezes, a disponibilidade do profissional de saúde para a condução, avaliação e armazenamento de dados dos treinos, a fim de observar a evolução do paciente, acarretando custos de tempo, transporte e recursos humanos.

2 Solução Proposta

Tendo em conta o problema descrito acima, foi desenvolvida uma versão do sistema *NeuroRehabLab Task Generator 1.0* para a execução das tarefas digitalmente, isto é, o sistema não necessita de imprimir tarefas para serem executadas por pacientes, a sua prática encontra-se totalmente no formato digital, através de uma aplicação multiplataforma, **Figura 3**, desenvolvida advindo de um site, inicalmente, elaborado.

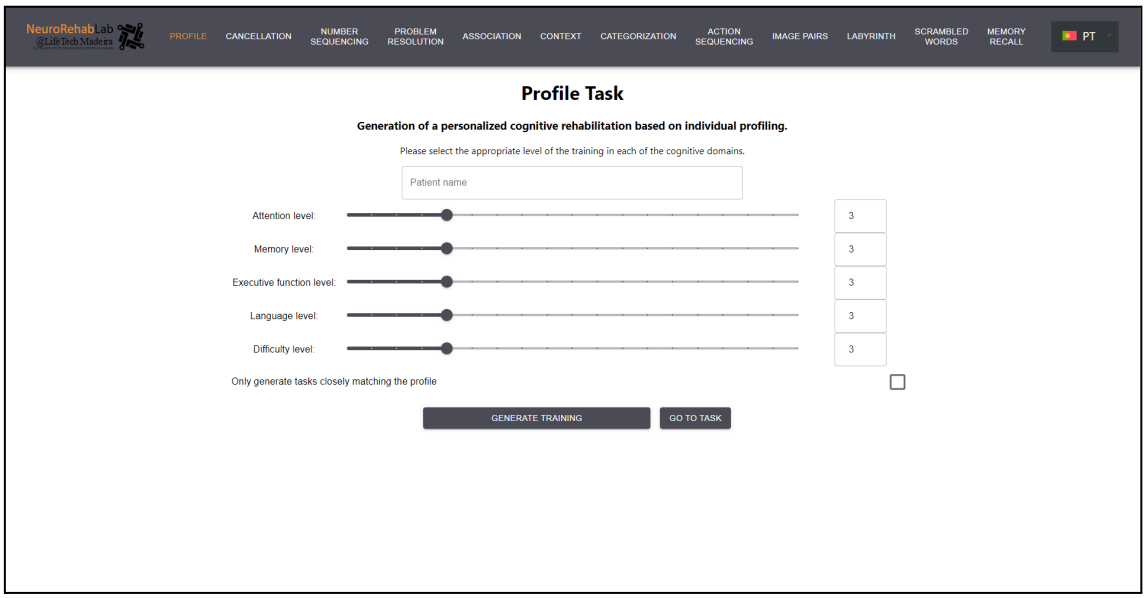

Figura 3 - NeuroRehabLab Task Generator 2.0 (tarefa Profile Task)

Esta versão mantém a lógica de produção e geração de tarefas da versão anterior, mas torna-as iterativas e acessíveis para tablets [7], solucionando a dificuldade tempo e recursos humanos para a sua elaboração e custo de produção.

No entanto, o produto ainda necessita de um profissional de saúde para a sua avaliação e registo da sua evolução dos pacientes. Visto que, o sistema atual guarda, apenas, objetos resultantes da submissão da tarefa, que incluem a resposta do paciente e os elementos para construção da tarefa. Para abordar estas necessidades, o sistema atual teria de ser modificado, de modo a aplicar funcionalidades que permitam a avaliação automática de cada tarefa e o registo contínuo da evolução do paciente. Essas modificações incluem:

- desenvolvimento de algoritmos que avaliem, automaticamente, o desempenho do paciente em cada tarefa
- elaboração de um sistema para armazenar as informações sobre o modelo do paciente e avaliação de cada treino, garantindo que todos os dados tornem-se disponíveis para análise e revisão posteriormente

- produção de um algoritmo para a pesquisa de treinos de um paciente específico
- elaboração de um sistema para guardar treinos de pacientes, assegurando a sua continuação posterior, caso não seja possível a sua conclusão
- criação de uma interface de utilizador intuitiva e fácil de usar para profissionais de saúde e pacientes

Com as afirmações acima em mente, o sistema a ser modificado tem de assegurar não apenas a redução da intervenção direta do profissional de saúde, mas também oferecer uma forma mais eficiente e precisa de acompanhar e avaliar o progresso dos pacientes.

3 Revisão da Literatura

Nesta revisão, apresentam-se dois estudos que exemplificam diferentes abordagens tecnológicas para a avaliação e reabilitação cognitiva, variando de ferramentas baseadas em papel e lápis a sistemas interativos complexos que utilizam tecnologias emergentes. Além disso, expõe-se as ferramentas cruciais para a modificação e desenvolvimento do sistema *NeuroRehabLab Task Generator 2.0*, tendo em conta a solução pretendida.

A lista de artigos é a seguinte:

||1|| **Ana Lúcia Faria and Sergi Bermúdez i Badia. 2015. Development and evaluation of a web-based cognitive task generator for personalized cognitive training: a proof of concept study with stroke patients**

||2|| **Elena de la Guía, María Dolores Lozano, and Victor R. Penichet. 2013. Cognitive rehabilitation based on collaborative and tangible computer games**

Os dois artigos compartilham um objetivo semelhante, desenvolver ferramentas que facilitem a reabilitação cognitiva de maneira personalizada e eficaz, contudo cada estudo aborda esta meta de maneira distinta.

Faria e Bermúdez i Badia (2015), desenvolveram uma ferramenta web gratuita que fabrica tarefas de reabilitação cognitiva personalizadas em formato PDF, aplicando tarefas de papel e lápis padronizadas e adaptadas às necessidades específicas de cada paciente. No entanto, o estudo de *De la Guía, Lozano e Penichet (2013)*, visou criar um sistema interativo utilizando tecnologias emergentes, como como Near Field Communication (NFC) e interação tangível, para melhorar o desempenho dos pacientes com Alzheimer em situações da vida real.

Os métodos e tecnologias produzidas nos dois estudos diferem, refletindo em diferentes abordagens. O estudo de *Faria e Bermúdez i Badia (2015)*, focou-se na parametrização de 11 tarefas de reabilitação cognitiva, ajustadas para atender diferentes domínios cognitivos, enquanto o de *De la Guía, Lozano e Penichet (2013)* à utilização da tecnologia NFC e dispositivos móveis para interações colaborativas.

Os resultados de ambos os estudos destacam a eficácia das abordagens personalizadas e tecnológicas para a reabilitação cognitiva, proporcionando uma base sólida para o desenvolvimento de sistemas eficazes e intuitivos.

Para o sistema a modificar, o *NeuroRehabLab Task Generator 2.0*, os conteúdos retidos destacam a importância de continuar a melhorar a interface e experiência do utilizador, em especial na redução da necessidade de intervenção médica direta e aprimoramento da usabilidade para profissionais de saúde e pacientes.

3.1 Ferramentas de Produção

Nesta seção, apresenta-se ferramentas/tecnologias que foram utilizadas para a fabricação da adaptação do sistema, utilizando-se Visual Studio Code como ambiente de desenvolvimento, empregando-se de extensões para a visualização de tabelas das bases de dados, assim como para a eficiência na escrita do código.

3.1.1 HTML: HyperText Markup Language

HTML ou HyperText Markup Language é o bloco de construção mais básico da web. Ele define o significado e a estrutura do conteúdo da web, através de uma série de elementos e tags. No desenvolvimento de sistemas web é utilizado para definir a estrutura e o layout das interfaces de utilizador, como textos, hiperlinks e arquivos de mídia.

3.1.2 CSS: Cascading Style Sheets

CSS ou Cascading Style Sheets é uma linguagem de estilo usada para descrever a exibição de um documento escrito em HTML ou em XML. O CSS descreve como elementos são mostrados no ecrã, aplicando estilos nos elementos do documento, como cores, fontes, fundo, etc.

3.1.3 Javascript

Javascript é uma linguagem que permite páginas da Web interativas, sendo, juntamente com HTML e CSS, uma das principais tecnologias da World Wide Web. É atualmente a principal linguagem para programação *client-side* em navegadores web, para a criação de interfaces de utilizador responsivas e interativas, e bastante utilizada do lado do servidor através de ambientes como Node.js [10].

3.1.4 React

React ou React.js é uma biblioteca de front-end de Javascript, criado pelo Facebook em 2011, com o foco em criar interfaces de utilizador em páginas web, destacando-se pela sua arquitetura declarativa e baseada em componentes.

React cria um DOM [11] virtual em memória, em vez de manipular diretamente o DOM do navegador, no qual realiza todas as suas operações, apenas em elementos que precisam de ser mudados, antes de mudar o DOM do navegador, garantindo um modelo de renderização eficiente.

3.1.5 MongoDB

MongoDB é um software de banco de dados NoSQL, ou seja, banco de dados não relacionais, que pode gerir informações orientadas a documentos para o armazenamento ou recuperação de dados. Este banco de dados é muito utilizado devido à sua flexibilidade, escalabilidade, desempenho em tempo real e armazenamento de grandes volumes de dados [12].

3.1.6 MySQL

MySQL é um sistema de gerenciamento de banco de dados relacional, que utiliza a linguagem SQL, *Structured Query Language*, para a criação, modificação e extração de dados. Este organiza os dados em uma ou mais tabelas de dados nas quais os dados podem estar relacionados entre si [13]. É um dos sistemas de banco de dados mais populares do mundo devido à sua facilidade de uso, confiabilidade e desempenho [14].

4 Implementação

Neste capítulo, será apresentado o processo de implementação das modificações necessárias no sistema *NeuroRehabLab Task Generator 2.0*. A secção abrange desde a definição dos requisitos até o desenvolvimento e evolução do produto, que inclui detalhes sobre o armazenamento de dados e avaliação das tarefas.

4.1 Requisitos

Na fase inicial, para a modificação do sistema *NeuroRehabLab Task Generator 2.0* foi categórico impor requisitos prioritários, sobre que tarefas seriam cruciais implementar no sistema atual, sendo estas atribuídas pelos orientadores e desenvolvedor, que operou na construção da versão anterior do produto.

Ao longo das reuniões, apresentações de funções implementadas e ao elaborar programas do sistema, foram encontrados outros requisitos, a fim da produção de um produto que concretizava o que era esperado.

4.1.1 Requisitos funcionais

O requisito funcional representa o que o software faz, em termos de tarefas e serviços [16]. A modificação do sistema deve operar os seguintes requisitos:

1. O sistema deve calcular a pontuação que o utilizador obteve após submeter a tarefa.
2. O sistema deve apresentar a pontuação de um utilizador numa determinada tarefa após a sua submissão.
3. O sistema deve apresentar na tarefa *Image Pairs* alguma atividade para o utilizador realizar, após o tempo de exibição das imagens terminar.
4. O sistema deve apresentar o modelo do paciente.
5. O sistema deve apresentar uma tabela, contendo a pontuação e a data de conclusão de cada treino realizadas por um determinado paciente.
6. O sistema deve terminar o treino de um paciente após 30 minutos da sua inicialização.
7. O sistema deve armazenar um novo treino na base de dados, caso o paciente tenha nenhum treino por completar.
8. O sistema deve armazenar o modelo de um novo paciente.
9. O sistema deve atualizar o modelo do paciente.
10. O sistema deve atualizar o treino armazenado na base de dados, quando o paciente proceder para a próxima tarefa.

4.2 Processo

Antes de proceder com a implementação de código, tornou-se imperativo o estudo e aprendizagem da linguagem de programação, na qual o sistema está construído, React. Para tal, utilizou-se a plataforma online *Codecademy* [15] proposta pelo colega que elaborou a versão anterior do sistema.

Após a compreensão da escrita e leitura da linguagem, sobre a qual grande parte do sistema operava, procedeu-se com a revisão e entendimento do código base do qual iria se construir e paralelamente análise dos dados que eram armazenados em JSON.

O desenvolvimento da modificação do sistema pré-existente, realizou-se de maneira incremental, resultante de diversas interações com os orientados. A cada etapa, o trabalho desenvolvido era apresentado, permitindo a obtenção de feedback, para ajustar e melhorar as funcionalidades implementadas. Após a apresentação, eram propostas funcionalidades e priorizadas conforme a importância e urgência. Este processo cíclico de apresentação, feedback e exposição e priorização de funcionalidades garantiu a implementação de um produto que atendesse às expectativas e necessidades impostas pelos orientadores.

As interações ocorreram presencialmente, o que ajudou na comunicação sobre os aspectos a desenvolver, envolvendo a discussão e apresentação da evolução do trabalho realizado semanalmente. Tendo a fabricação deste trabalho, principais funcionalidades, realizada na seguinte ordem: cálculo da pontuação de cada tarefa, armazenamento do modelo do paciente, armazenamento das informações para a geração de cada treino e armazenamento da avaliação do treino de um paciente.

Assim, o processo de desenvolvimento pode ser visualizado de forma simplificada na **Figura 4**, que resume, de forma abstrata, todas as fases.

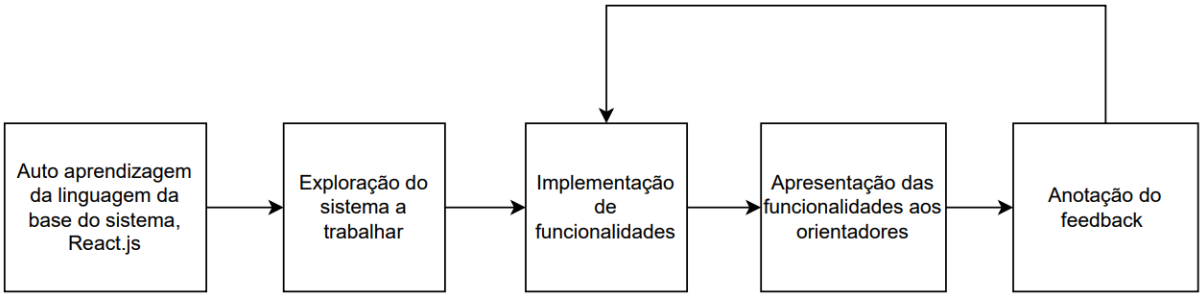

Figura 4 - Processo de desenvolvimento

4.3 Evolução do protótipo

Como enunciado na seção anterior, cada etapa do desenvolvimento foi cuidadosamente planeada e executada, começando pelo cálculo da pontuação de cada tarefa, seguido pelo armazenamento do modelo do paciente, e culminando no armazenamento das informações necessárias para a geração e avaliação de cada treino.

4.3.1 Cálculo da Pontuação de Cada Tarefa

O cálculo da pontuação de cada tarefa foi fabricado em função dos dados armazenados em JSON, que eram o objeto resultante da submissão dos dados da tarefa após a sua conclusão, constituído pelos atributos ***task***, que indica qual a tarefa associada ao objeto e ***data***, que contém as propriedades relevantes para a implementação do cálculo da pontuação, sendo elas as seguintes:

- ***task***, que inclui propriedades para a construção da tarefa pelo sistema, como *instructions*, *model* e outros atributos específicos de cada tarefa.
- ***submitted***, é um atributo com a resposta dada pelo utilizador, diversificado entre as várias tarefas do sistema, ou seja, esta propriedade é distinta entre cada tarefa, tendo a capacidade de ser uma lista, matriz ou objeto dependendo da tarefa.

A implementação do cálculo para cada tarefa ocorreu utilizando-se testes unitários [17], para detectar erros logo no início do ciclo de desenvolvimento e verificar se o cálculo da pontuação para cada tarefa estava funcionando conforme o esperado. A validação desses cálculos foi realizada em conjunto com os orientadores, assegurando a precisão e a adequação dos resultados.

A seguir, serão apresentados detalhadamente os métodos e algoritmos utilizados para o cálculo das pontuações para cada tipo de tarefa, assim como os objetivos de cada uma.

4.3.1.1 Cancellation Task

Nesta tarefa o utilizador tem como objetivo selecionar, de um conjunto de elementos, todos os elementos que correspondam ao elemento alvo requerido pela tarefa, assim além dos atributos indicados na propriedade ***task*** o objeto JSON da tarefa tinha:

- ***target***, o elemento alvo a ser selecionado na tarefa
- ***taskArray***, o array com os elementos que será mostrado para o utilizador selecionar
- ***vectorTarget***, o array que representa a localização do/s elemento/s alvo apresentados ao utilizar, ou seja, caso o elemento alvo encontra-se nessa posição do array era colocado a 1, de outra forma a 0
- ***number***, variável que determina se símbolos, letras ou números serão usados
- ***size***, variável que define o tamanho dos elementos exibidos

Sendo a resposta do utilizador, ***submitted***, um array de números inteiros, compreendendo as posições dos elementos selecionados na tarefa.

O cálculo da avaliação da tarefa é o número de elementos corretamente assinalados menos os elementos incorretamente assinalados. Este resultado deve ser convertido numa percentagem em relação ao número de elementos existentes para assinalar.

Para isso, verificou-se se as posições dos elementos assinalados pelo utilizador, no atributo ***submitted***, correspondiam à localização dos elementos alvo, no atributo ***vectorTarget***, identificando-se assim o número de elementos assinalados corretamente. Para obter o número de elementos incorretamente assinalados, examinou-se a quantidade de elementos alvo não selecionados, no atributo ***vectorTarget***, juntamente com o número de elementos incorretamente selecionados, no atributo ***submitted***.

Após adquirir o número de elementos corretamente e incorretamente assinalados, verificou-se se o número de elementos incorretamente assinalados era inferior ao número de elementos corretamente assinalados, caso se confirme, subtraiu-se o número de elementos incorretamente assinalados do número de elementos corretamente assinalados e dividiu-se o resultado pelo total de elementos alvo existentes, multiplicando-se por 100 para obter a pontuação em percentagem.

4.3.1.2 Sequencing Task

O objetivo desta tarefa é o preenchimento de valores ausentes em sequências, ordenadas ascendentemente ou descendentemente, que possuem um certo tamanho entre valores. Desta forma, salvo dos atributos indicados na propriedade ***task***, o objeto JSON da tarefa tinha o *sequence*, uma matriz de sequências, inserindo-se, posteriormente, as seguintes propriedades para a ajuda no cálculo:

- ***step***, o espaçamento entre cada valor das sequências
- ***orderSquences***, um array com a identificação da maneira como está ordenada cada sequência, ou seja, a posição elemento corresponde à ordenação na posição sequência no atributo *sequence*, caso o valor corresponda a 0 a ordenação é ascendente e 1 para descendente

Tornando-se a resposta dada pelo utilizador, ***submitted***, uma matriz semelhante ao atributo *sequence*, contudo as sequências não continham os valores apresentados aos utilizadores, apenas os valores escritos pelos menos na sequência.

A avaliação da tarefa baseia-se no número de sequências corretamente resolvidas. Este resultado deve ser convertido numa percentagem em relação ao número de sequências existentes para resolver. Com esse propósito, juntou-se ambas as matrizes ***sequence*** e ***submitted*** substituindo-se na matriz sequence os valores para o utilizador preencher, assinalados com o caracter '_', pelos valores introduzidos pelo utilizador. Após a combinação, tendo em conta a ordenação da sequência, dada pelo atributo ***orderSquences***, e o espaçamento entre cada valor, atributo ***step***, analisou-se cada elemento. Na situação de na sequência existir um elemento inadequado, esta era considerada incorreta. No entanto, se todos os elementos fossem adequados, esta estaria correta.

4.3.1.3 Problem Resolution Task

Esta tarefa requer ao utilizador resolver uma série de problemas, apresentando para cada um deles os seus resultados. Assim sendo, à exceção dos atributos indicados na propriedade ***task***, o objeto JSON da tarefa tinha:

- ***sequenceArray***, um array que armazena equações matemáticas na forma de uma matriz de elementos ou em formato textual

- ***explicit***, variável que determina se é um problema explícito ou implícito

Com apenas estes atributos, não era possível o cálculo da pontuação, logo inseriu-se o atributo ***results***, que acomoda os resultados dos problemas dados pelo sistema, tanto implícitos como explícitos. Para a adquirir os resultados dos problemas explícitos, apenas precisou-se utilizar a função *eval* de Javascript para devolver da string das equações, no atributo ***sequenceArray***, para os seus devidos resultados. Já para os resultados dos problemas implícitos, necessitou-se de uma análise profunda explorando-se todos os casos possíveis e transformá-los em equações, de modo a reutilizar o mesmo método para conseguir os resultados nos problemas explícitos.

A resposta do utilizador, ***submitted***, equivale a um array com as respostas do utilizador para cada um dos problemas, na mesma ordem em que eles são apresentados no sistema ou guardados no atributo ***sequenceArray***.

A avaliação da tarefa fundamentou-se na comparação dos elementos nas mesmas posições no array contendo as respostas corretas e do array com as respostas dadas pelo utilizador, determinando o número de problemas respondidos corretamente. Este resultado foi então convertido em uma percentagem em relação ao total de problemas a serem resolvidos.

4.3.1.4 Association Task e Image Pairs

A tarefa *Association* tem o propósito a associação de duas imagens que se encontrem relacionadas entre si. No entanto, a tarefa *Image Pairs* possui a mesma finalidade que a tarefa *Association*, mas com uma abordagem diferente. Inicialmente, ela exibe os pares de imagens a serem associados por um período de tempo, depois os oculta por um intervalo, e, por fim, solicita ao utilizador que associe os pares de imagens compatíveis aos apresentados inicialmente. Desta forma, por serem relativamente semelhantes na execução da atividade, possuem a mesma propriedade ***task***, que engloba:

- ***leftObjects***, uma matriz de objetos a serem exibidos no lado esquerdo, os quais possuem atributos como *id* e *image*

- ***rightObjects***, uma matriz semelhante à *leftObjects*, contudo contém os objetos a serem exibidos na lado direito

- ***pairing***, um array de objetos, cada um representando um par de objetos, com um objeto de cada das duas matrizes anteriores.

- ***note***, uma nota referenciando imagens provenientes do banco de dados IPNP, visível apenas quando clipart imagens são utilizadas

Correspondendo a respostas dada pelo utilizador, ***submitted***, a um array de objetos, cada um representando um par de objetos selecionados pelo utilizador.

Neste sentido, a avaliação da tarefa apoia-se no número de associações corretas dadas pelo utilizador. Para isso, foram analisadas as associações fornecidas pelo utilizador, no atributo ***submitted***, verificando-se se ambos os pares de objetos disponham do mesmo atributo ***id***, excluindo os dois primeiros caracteres que identificavam o lado no qual eram exibidos. Desta forma, determinou-se o número de associações corretas e com este resultado converteu-se numa percentagem em relação ao número de associações existentes para resolver.

4.3.1.5 Context Task

Esta tarefa implica a validação de descrições de uma imagem, ou seja, o utilizador indica se a descrição representa ou não a imagem apresentada. Para tal, a propriedade ***task***, contém:

- ***image***, a imagem que é exibida para interpretação
- ***sentences***, um conjunto de descrições a serem avaliadas como corretas ou incorretas
- ***correct***, do conjunto das descrições a serem avaliadas apenas as corretas
- ***incorrect***, do conjunto das descrições a serem avaliadas apenas as incorretas

A resposta do utilizador, ***submitted***, é um array de objetos com as propriedades ***sentence***, que contém a descrição, e ***selection***, que indica a avaliação da descrição feita pelo utilizador.

Para avaliar a tarefa, percorreu-se o array resultante da submissão (***submitted***), verificando se a descrição (***sentence***) estava contida no conjunto de descrições indicado pelo atributo ***selection***, ou seja, verificou-se se a descrição estava incluída no array ***correct*** ou ***incorrect***, conforme indicado pelo ***selection***. Assim, determinou-se o número de descrições corretamente identificadas, convertendo-se esse resultado em uma percentagem em relação ao total de descrições a serem resolvidas.

4.3.1.6 Categorization Task

Nesta tarefa, é apresentada uma série de imagens ao utilizador, na qual deve identificar a categoria correspondente para cada imagem. Assim, a propriedade ***task***, desta tarefa inclui, para além dos atributos **model** e **instructions**, os seguintes:

- ***taskArray***, um array de objetos, cada um contendo as propriedades ***category*** e ***dataUrl***
- ***note***, uma nota referenciando imagens provenientes do banco de dados IPNP, visível apenas quando clipart imagens são utilizadas
- ***allCategoriesList***, um array contendo todas as categorias
- ***categories***, uma variável que faz referência ao número de categorias
- ***elements***, uma variável que faz referência ao número de elementos por categoria

A resposta do utilizador, ***submitted***, corresponde a um array semelhante ao ***taskArray***, com os objetos nas mesmas posições. No entanto, o atributo ***category*** é preenchido com a categoria selecionada pelo utilizador.

A avaliação da tarefa efetuou-se percorrendo a resposta do utilizador (***submitted***) e comparando as categorias dos elementos com as categorias correspondentes no ***taskArray***, verificando se estão corretas, definindo o número de elementos corretamente categorizados. Este resultado converteu-se numa percentagem em relação ao número de elementos existentes para categorizar.

4.3.1.7 Action Sequencing Task

A tarefa tem como finalidade, a ordenação de um conjunto de ações para a realização de uma determinada ação atribuída pelo sistema, como por exemplo escovar os dentes. Para isso, a propriedade ***task*** tem os atributos:

- ***action***, a ação a ser representada, especificada apenas se for uma ação explícita
- ***sentences***, um array de objetos das etapas selecionadas para serem ordenadas
- ***sentencesOrdered***, um array com as ações ordenadas. Adicionado de forma a ser possível a sua avaliação.

A resposta do utilizador, ***submitted***, coincide com a um array de objetos, que incluem os pares de objetos selecionados pelo utilizador, ou seja, um indica a posição da ação, atributo ***left***, na ordenação e o outro a ação, atributo ***right***.

Desta forma, a avaliação da tarefa realizou-se percorrendo a resposta do utilizador (***submitted***), na qual para cada objeto era comparado a ação no array ***sentencesOrdered*** na posição indicada no atributo ***left***, com a ação atribuída pelo utilizador para essa posição, isto é, apresentada no atributo ***right***, com o objetiva determinar o número de elementos ordenados corretamente. Portanto, com este resultado, converteu-se para percentagem em relação ao número de elementos a ordenar total.

4.3.1.8 Maze Task

Nesta tarefa o utilizador tem de percorrer um labirinto, ou seja, começa num ponto inicial, A, e tem de descobrir o melhor caminho e menor caminho para chegar ao ponto final, B. Para tal, esta tarefa na propriedade ***task*** possui os atributos:

- ***maze***, ama matriz, fornecida por um algoritmo, representando o labirinto
- ***minMovesToFinish***, uma variável que indica o número mínimo de movimento para chegar de A a B. Adicionada, de modo a ser possível e rápida a avaliação da tarefa, obtida através da função, pré-existente, *solve*

A resposta do utilizador, ***submitted***, é o caminho percorrido pelo utilizador para chegar ao ponto final, juntamente com o número de movimentos realizados para lá chegar.

Desta maneira, tendo em conta o número de movimentos realizados pelo utilizador para chegar ao final (***UserMov***) e o número de movimentos mínimos (***MinMov)***, efetuou-se a avaliação da tarefa dividindo-se ***MinMov*** por ***UserMov***, resultando na avaliação do desempenho do utilizador na tarefa.

4.3.1.9 Soup of Letters Task

A tarefa tem como intenção a procura de palavras, num conjunto de letras aleatórias. Para tal intuito, a propriedade ***task*** apresenta os atributos:

- ***n_elements***, uma variável que indica o número de letras a serem exibidas na grade
- ***words***, um array contendo as palavras a serem localizadas
- ***matrix***, uma matriz 2D que representa a grade para exibição
- ***matrix_words***, um array 2D que renderiza a grade, exibindo apenas as palavras
- ***wordsWithPosition***, uma array de objetos, com cada objeto representando uma palavra e as posições das letras correspondentes

A resposta do utilizador, ***submitted***, engloba as palavras encontradas pelo utilizador (***highlightedWords***) e as letras selecionadas pelo mesmo (**highlilghtedLetteres**).

A avaliação da tarefa e a indicação do seu resultado levaram em conta o número de palavras encontradas pelo utilizador (***highlightedWords***) em relação ao número de palavras que deveriam ser localizadas (***words***), convertendo-se esse resultado em percentagem.

4.3.1.10 Memory Recall Task

Nesta tarefa é exibido um texto, durante um período de tempo, e seguidamente um conjunto de afirmações, às quais o utilizador indica se são verdadeiras ou falsas, com relação ao texto apresentado anteriormente. Para tal, a propriedade ***task***, além de incluir o ***model*** e ***instructions***, tem os atributos:

- ***image***, a imagem extraída de histórias de imagens
- ***text***, uma variável que contém a história
- ***sentences***, um array contendo as declarações extraídas da história, com a respectiva avaliação, verdadeiro ou falso

A resposta do utilizador, ***submitted***, é um array de objetos com da respectiva declaração de texto com a avaliação declarada pelo utilizador.

Deste modo, comparando as avaliações das declarações dadas pelo utilizador com o array ***sentences***, com as avaliações corretas, determinou-se o número de questões respondidas corretamente, obtendo-se assim a avaliação da tarefa. Esse resultado foi,

então, convertido numa percentagem em relação ao número de questões a serem respondidas.

4.3.2 Armazenamento dos Dados

Para o armazenamento de dados cruciais para a construção de um treino, ou seja, a realização sequencial das tarefas apresentadas anteriormente, e para a análise da evolução do paciente durante a sua execução, utilizou-se dois banco de dados distintos: um relacional (MySQL) e um não relacional (Mongodb).

O banco de dados MySQL foi utilizado para o armazenamento das informações relacionadas ao paciente, que incluem o seu modelo e avaliação dos treinos. A escolha do MySQL foi motivada, devido ao desempenho para lidar com grandes volumes de dados e consultas complexas, custo-benefício e a alta compatibilidade com diversas plataformas e linguagens de programação.

O banco de dados Mongodb deu-se o armazenamento das informações referentes à construção de um treino para um paciente específico. As razões para a sua utilização englobam o desempenho, proporcionar o armazenamento de dados de forma flexível, sem a rigidez dos esquemas de tabelas e por ser otimizado para grandes volumes de dados.

De seguida, expõe-se a evolução do armazenamento dos dados para cada banco de dados, MySQL e Mongodb, de que forma os dados foram utilizados, como foram obtidos e porquê de encontrarem-se armazenados.

4.3.2.1 Armazenamento do Modelo do Paciente e Avaliação do Treino

O armazenamento das informações do paciente, como modelo e avaliação de cada treino, como mencionado na secção anterior, foram guardadas no banco de dados MySQL. Neste sentido, determinou-se que a base de dados necessitaria de três tabelas:

- ***patients***, tabela que conteria as informações demográficas do paciente, contudo apenas abriga o identificador único de pesquisa do paciente, que serve para no sistema procurar pelo modelo e avaliações de um determinado paciente

- ***patientmodels***, tabela que abriga as informações do modelo de um paciente, isto é, valores de atenção, memória, linguagem, função executiva e dificuldade, que atualiza dependendo do desempenho do paciente nos treinos, ou seja, da sua avaliação do treino

- ***profilescores***, tabela que enquadrar a pontuação do utilizador obtida da conclusão de um treino em cada tarefa, a respectiva avaliação final (média das pontuações) e a data e hora da sua conclusão

A relação inicial proposta entre estas tabelas é a seguinte:

- Um paciente possui um único modelo

- Um paciente pode ter zero ou mais avaliações de treino

Essa estrutura, **Figura 5**, permitiu uma organização eficiente dos dados, facilitando a recuperação e atualização das informações do paciente conforme ele realizava e concluía os treinos.

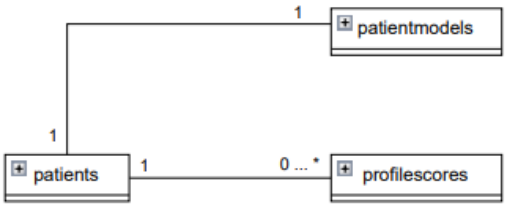

Figura 5 - Diagrama UML da relação entre as tabelas no banco de dados MySQL

No entanto, ao longo do processo de interação com os orientadores, foi estabelecida a necessidade de guardar toda a evolução dos pacientes, mais concretamente a evolução do seu modelo. Deste modo, alterou-se a relação entre as tabelas *patients* e *patient models* para uma relação um para muitos, para que os pacientes tivessem diversos modelos, ou seja, ser praticável a visualização da mudança/evolução do modelo do paciente, **Figura 6**.

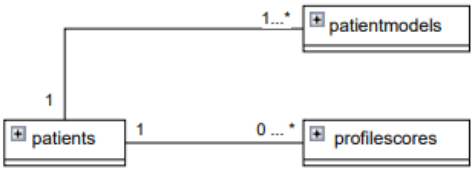

Figura 6 - Diagrama UML com alteração da relação entre as tabelas *patients* e *patientmodels*

Todavia, verificou-se uma insuficiência de informação relativamente ao treino de cada paciente, visto que seria vital saber para qual modelo tinha sido realizado o treino do paciente. Para tal, alterou-se a relação existente entre a tabela *profilescores* e moveu-se-a, colocando-a entre as tabelas *patientmodels* e *profilescores*, obtendo-se o diagrama final e utilizado da **Figura 7**.

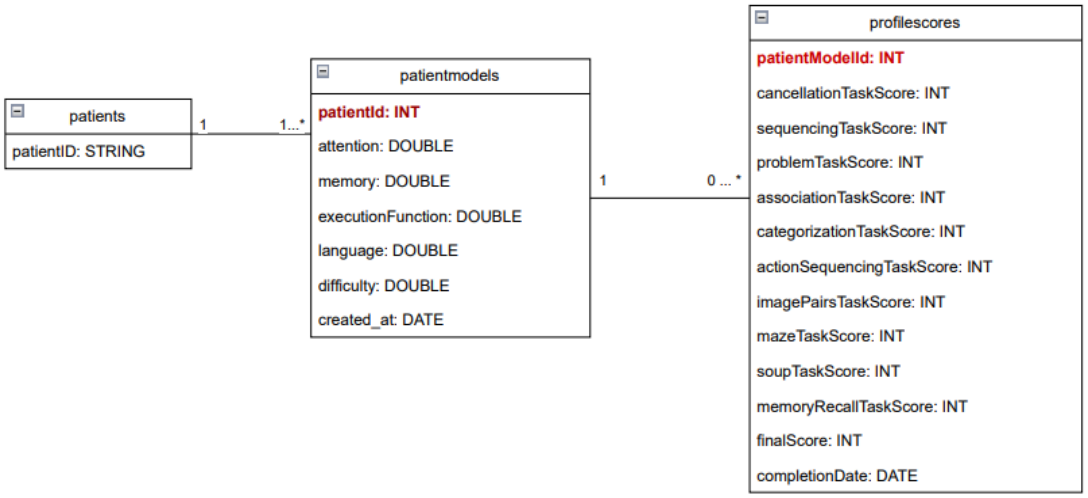

Figura 7 - Diagrama UML final das tabelas do banco de dados MySQL

Tendo-se as tabelas estabelecidas e respectivas relações, modificou-se a interface da tarefa *Profile Task*, que incluía a inserção de dados do paciente, como nome e modelo, para a criação de um treino. Desta maneira, utilizando elementos pré-existentes, alterou-se a sua apresentação, **Figura 8**, exibindo-se dados do paciente, modelo mais recente e avaliação dos treinos realizados, dispondo-se do nome do paciente introduzido, para a pesquisa e caso o mesmo não exista, a adição do paciente, e respectivo modelo, no banco de dados, que ocorre ao criar um treino.

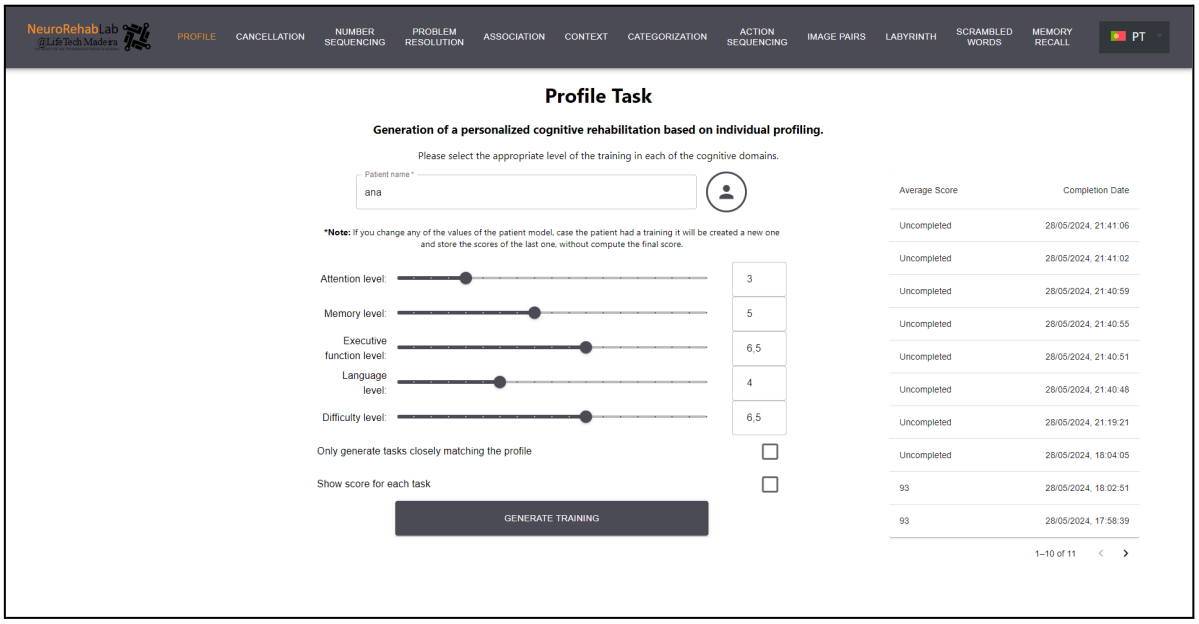

Figura 8 - NeuroRehabLab Task Generator 2.0 modificado (tarefa Profile Task)

4.3.2.2 Armazenamento das Informações do Treino

O armazenamento das informações do treino de um paciente, ou seja, os dados necessários para a geração das tarefas ou continuação da realização das tarefas, alojou-se no banco de dados Mongodb. Para tal, construiu-se duas coleções:

uma para guardar os dados do treino e uma para acomodar os elementos para a construção de cada tarefa, **Figura 9**.

Figura 9 - Diagrama UML das coleções no banco de dados Mongodb

A coleção *ProfileTask* é utilizada para a busca e armazenamento de treinos incompletos de um paciente, a fim da construção das tarefas restantes para serem concluídas pelo sistema, atualizando-se as tarefas restantes e os resultados obtidos, ao ser submetida a conclusão de cada tarefa. Nessa finalidade, possui os atributos:

- ***patientId***, variável com o identificador utilizado para procurar por um paciente específico
- ***timeEachCycle***, duração da realização do treino após o seu início
- ***tasks***, array com os identificadores das tarefas, a serem realizadas pelo paciente, na coleção *Task*
- ***results***, array com pontuação obtida pelo paciente em cada tarefa concluída durante o treino

A coleção *Task* guarda o nome da tarefa e os dados necessários para apresentação dessa tarefa no sistema. Para tal, compreende os atributos:

- **task**, variável que indica o nome da tarefa
- **data**, que engloba as informações da tarefa, ou seja, a propriedade *task* indicada na secção 4.3.1, que incorpora os atributos para a construção da tarefa pelo sistema

5 Avaliação

A avaliação do sistema foi realizada principalmente em relação às funcionalidades, que como mencionado anteriormente, foram ajustadas e melhoradas semanalmente.

5.1 Algoritmos de avaliação do treino

Dada a importância dos algoritmos para avaliar o desempenho do paciente nas diferentes tarefas, pois proporciona a avaliação de uma terapia/treino, é crucial garantir a corretude da sua implementação.

Deste modo, as funções de avaliação de cada tarefa foram sujeitas a testes unitários, permitindo validar as funcionalidades de avaliação, verificando se os valores esperados condizem com os resultados dados pelas funções construídas, como por exemplo o teste produzido para avaliar a tarefa *Problem Task*, para problemas explícitos e de tamanho 2, ilustrado na **Figura 10**, no qual através do objeto resultante da submissão dos dados da tarefa, contudo com apenas a propriedade *data* (ver secção 4.3.1), utilizou-se a avaliação apresentada na secção 4.3.1.3, para obter o desempenho e verificar se coincide com a pontuação esperada.

```
test("score of the task problem when size = 2 and it's explicit ", () => {
  const data = {
    task: {
      model: {
        attention: "7.0",
        memory: "6.1",
        exfunctions: "7.2",
        language: "4.7",
        difficulty: "6.0",
      },
      instructions: "Por favor, resolva os seguintes problemas.",
      sequenceArray: [
        ["6", " * ", "7"],
        ["3", " + ", "2"],
        ["2", " * ", "1"],
        ["8", " + ", "3"],
        ["4", " * ", "4"],
        ["2", " + ", "3"],
        ["5", " + ", "4"],
        ["5", " + ", "1"],
        ["5", " + ", "1"],
      ],
      explicit: "0",
      results: [42, 5, 2, 11, 16, 5, 9, 6, 6],
    },
    submitted: ["42", "5", "2", "11", "16", "5", "9", "7", "7"],
  };
  expect(evaluateProblemTask(data)).toBe(78);
});
```

Figura 10 - Teste unitário para a tarefa Problem Task

A utilização dos testes, assegurou que os algoritmos medissem corretamente o desempenho dos pacientes, proporcionando dados confiáveis para o acompanhamento e evolução da terapia.

5.2 Usabilidade

Todavia, para garantir que o sistema se encontre compreensível e fácil de usar, foi necessário avaliar a sua usabilidade. Para tal, utilizou-se testes de usabilidade, com o intuito de avaliar a facilidade de utilização do sistema e a experiência do utilizador.

Os testes foram efetuados em computador, com apenas um participante, que possui 20 anos, tem aptidão tecnológica e não usufruiu do sistema anterior ou do desenvolvido. Ao participante foi apresentado o consentimento informado, para que tenha total compreensão do que estão a concordar ao realizar o teste. Além disso, indicou-se os objetivos do teste, isto é, a identificação de problemas de usabilidade e aspectos a melhorar, e as tarefas a realizar:

1. Pesquisar pelo paciente ‘ana’
2. Criar um treino para o paciente ‘ana’
3. Retroceder ao formulário da tarefa *Profile Task*
4. Adicionar um novo paciente
5. Dizer a pontuação do paciente ‘ana’ para a data 29/04 na hora 11:44
6. Modificar um dos valores do modelo do paciente ‘ana’

As tarefas pretendem analisar o quão fácil para um utilizador, seja um paciente ou um psicólogo, a pesquisa e criação de dados referentes a um paciente, assim como a produção de um treino.

Desta maneira, aplicou-se os métodos think-aloud, no qual o participante narra todos os seus pensamentos durante a realização da tarefa, para obter dados qualitativos, e métricas de usabilidade: tempo para concluir a tarefa, número de erros cometidos na tarefa e nível de dificuldade ao realizar a tarefa (1 - muito fácil, 5 - muito difícil), obtendo-se dados quantitativos.

6 Resultados

Os resultados quantitativos obtidos podem ser observados na **Tabela 1**. Estes dados representam as métricas de usabilidade recolhidas da observação e questionamento ao participante, durante a realização de cada tarefa, mencionada na seção 5.

Tarefa	Tempo de conclusão	Nível de dificuldade	Número de erros	Comentários
1	14s	1	0	—
2	24s	2	0	Consistência na língua utilizada. Ou português ou inglês
3	5s	3	0	Voltar ao formulário utilizando a barra de navegação é difícil
4	75s	1	0	Separar o adição da pesquisa do paciente
5	53s	2	1	—
6	27s	1	0	—

Tabela 1 - Dados quantitativos recolhidos dos testes de usabilidade com o participante

Relativamente aos dados qualitativos recolhidos, ocorreram por meio da gravação da voz do participante permitindo a análise, posterior, das ações efetuadas e expressão de dificuldades sentidas durante a execução da tarefa.

7 Discussão

A modificação do sistema *NeuroRehabLab Task Generator 2.0*, no aspecto das funcionalidades para a obtenção da solução desejada, foi, de modo geral, um sucesso. As alterações implementadas, asseguraram a observação do progresso dos pacientes e a redução da intervenção do profissional de saúde.

No entanto, a eliminação da intervenção do profissional de saúde foi o método que não foi completamente alcançado. No sistema implementado, as funcionalidades de pesquisa ou adição de um paciente e modificação do modelo do paciente, requerem um profissional de saúde, na finalidade de registar e visualizar a evolução do paciente.

Além disso, analisando-se os resultados do testes de usabilidade verificou-se aspectos de interface a melhorar, referentes à pesquisa e adição do paciente no sistema.

Tendo em conta os dados quantitativos, apresentados na **Tabela 1**, verifica-se uma variação no tempo de conclusão das tarefas, nível de dificuldade indicada pelo participante e número de erros cometidos. As tarefas 1, 4 e 6 apresentam um nível de dificuldade baixo (nível 1) e sem erros, indicando que essas tarefas foram bem compreendidas e executadas, sugerindo uma boa usabilidade. No entanto, é de notar que a adição do paciente, tarefa 4, demorou um tempo consideravelmente alto e durante a realização da tarefa o participante questionar-se sobre a localização de tal funcionalidade, dá suporte, a posteriormente, ao redesenho da interface para separar estas duas funcionalidades, a fim de reduzir a complexidade percebida pelos utilizador, tornando-a mais intuitiva e fácil de usar.

Por outro lado, a tarefa 3, que envolvia retornar ao formulário utilizando a barra de navegação, foi considerada média (nível 3), pois durante a realização da tarefa foi encontrado um erro na barra de navegação, no qual o utilizador não conseguia retroceder ao formulário do *Profile Task*, utilizando o respectivo botão na barra de navegação.

A tarefa 2 também destacou uma inconsistência na língua utilizada, visto que alguns dos textos apresentados estão estáticos em inglês, o que pode causar confusão e dificultar a usabilidade de quem operar o sistema em português.

A tarefa 5, com nível de dificuldade de 2 e um erro cometido, sugeriu que embora a tarefa não seja difícil, a forma como a visualização da pontuação é apresentada pode confundir o utilizador. Visto que, como se observou com o participante durante os testes, enquanto o campo de introdução do “id” do paciente não for desfocado a informação não exibe, indicando a necessidade de aprimorar a interface para fornecer feedback imediato e claro.

Em suma, o sistema atingiu muitos de seus objetivos, contudo existem áreas de aprimoramento, especialmente na usabilidade e na interface de utilizador, para tornar o sistema mais eficiente e intuitivo para os utilizadores finais.

8 Conclusão

A modificação do sistema *NeuroRehabLab Task Generator 2.0* visou não apenas reduzir a intervenção direta do profissional de saúde, mas também oferecer uma maneira mais eficiente e fácil de acompanhar e avaliar o progresso dos pacientes. A implementação das novas funcionalidades foi, em grande parte, bem-sucedida, resultando em uma ferramenta que facilita a observação do progresso dos pacientes e reduz a intervenção direta do profissional de saúde.

Embora a eliminação total da intervenção do profissional de saúde não tenha sido alcançada, o sistema avançou consideravelmente em automatizar muitas das tarefas que anteriormente exigiam acompanhamento constante. Isso representa um passo importante para tornar o processo de reabilitação mais acessível e menos dependente de recursos humanos intensivos.

Além disso, os resultados dos testes de usabilidade indicaram áreas de aprimoramento, especialmente na interface de utilizador, que devem ser abordadas para tornar o sistema mais intuitivo e acessível.

Em resumo, a modificação do sistema *NeuroRehabLab Task Generator 2.0* atingiu muitos de seus objetivos principais, proporcionando uma solução digital eficiente para a reabilitação cognitiva. As melhorias sugeridas com base nos testes de usabilidade fornecerão um caminho claro para futuros desenvolvimentos, com o objetivo de tornar o sistema ainda mais intuitivo e autónomo, beneficiando tanto os profissionais de saúde quanto os pacientes.

Referências

[1] “Fact-Sheet - Associação Alzheimer Portugal”. [Online]. Disponível em: https://alzheimerportugal.org/fact-sheet/

[2] “How Is Alzheimer’s Disease Treated?”, National Institute on Aging. [Online]. Disponível em: https://www.nia.nih.gov/health/alzheimers-treatment/how-alzheimers-disease-treated

[3] B. A. Wilson, “Cognitive Rehabilitation: How it is and how it might be,” J. Int. Neuropsychol. Soc., vol. 3, no. 05, pp. 487–496, 1997.

[4] J. Solana, C. Cáceres, A. García-Molina, E. Opisso, T. Roig, J. M. Tormos, and E. J. Gómez, “Improving brain injury cognitive rehabilitation by personalized telerehabilitation services: Guttmann neuropersonal trainer,” IEEE Journal of Biomedical and Health Informatics, vol. 19, no. 1, p. 124–131, Jan 2015.

[5] A. L. Faria e S. B. i Badia, “Development and evaluation of a web-based cognitive task generator for personalized cognitive training: a proof of concept study with stroke patients”, em Proceedings of the 3rd 2015 Workshop on ICTs for improving Patients Rehabilitation Research Techniques, em REHAB ’15. New York, NY, USA: Association for Computing Machinery, out. 2015, pp. 1–4. doi: 10.1145/2838944.2838945.

[6] S. Badia e A. Faria, “Programa adaptável para Reabilitação Personalizada do Acidente Vascular Cerebral”.

[7] T. Marcos, “Task Generator 2.0: leveraging personalized paper and pencil cognitive training to a tablet approach”. 2024.

[8] E. de la Guia, M. D. Lozano, e V. M. R. Penichet, “Cognitive Rehabilitation Based on Collaborative and Tangible Computer Games”, apresentado na ICTs for improving Patients Rehabilitation Research Techniques, mai. 2013. [Online]. Disponível em: https://eudl.eu/doi/10.4108/icst.pervasivehealth.2013.252375

[9] G. H. Kwon, L. Kim, e S. Park, “Development of a Cognitive Assessment Tool and Training Systems for Elderly Cognitive Impairment”, apresentado na 3rd International Workshop on Pervasive Computing Paradigms for Mental Health, mai. 2013. [Online]. Disponível em: https://eudl.eu/doi/10.4108/icst.pervasivehealth.2013.252384

[10] “JavaScript | MDN”. [Online]. Disponível em: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[11] “Document Object Model (DOM) - Web APIs | MDN”. [Online]. Disponível em: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

[12] “What is MongoDB? Features and how it works – TechTarget Definition”, Data Management. [Online]. Disponível em: https://www.techtarget.com/searchdatamanagement/definition/MongoDB

[13] “MySQL”, Wikipedia. [Online]. Disponível em: https://en.wikipedia.org/w/index.php?title=MySQL&oldid=1223967813

[14] “O que é o MySQL?” [Online]. Disponível em: https://www.oracle.com/br/mysql/what-is-mysql/

[15] “Learn React”, Codecademy. [Online]. Disponível em: https://www.codecademy.com/learn/react-101

[16] “Requisito funcional”, Wikipédia, a enciclopédia livre. [Online]. Disponível em: https://pt.wikipedia.org/w/index.php?title=Requisito_funcional&oldid=66062733

[17] “O que são testes unitários?”, Blog: Recrutar Desenvolvedores e Conteúdo de Programação. [Online]. Disponível em: https://coodesh.com/blog/dicionario/o-que-sao-testes-unitarios/