

Licenciatura em Engenharia Informática

# Índice

[Índice 2](#_Toc106153907)

[Resumo 3](#_Toc106153908)

[Equipa de Trabalho 3](#_Toc106153909)

[Divisão da Aplicação 4](#_Toc106153910)

[Componentes 5](#_Toc106153911)

[Board 5](#_Toc106153912)

[ControlPanel 6](#_Toc106153913)

[Letter 7](#_Toc106153914)

[Timer 7](#_Toc106153915)

[Words 8](#_Toc106153916)

[Limitações Conhecidas 9](#_Toc106153917)

[Desafios 10](#_Toc106153918)

# Resumo

O trabalho prático de Linguagens Script consiste na criação do jogo tradicional, Sopa de Letras, utilizando a linguagem de programação ***Javascript*** e a framework ***React***.

A aplicação é constituída por vários componentes com propriedades reativas que juntos formam a aplicação inteira como podemos ver na imagem abaixo.



# Equipa de Trabalho

O trabalho prático foi realizado pela seguinte equipa:

* Tomás Gomes Silva (2020143845)
* Tomás da Cunha Pinto (2020144067)

# Divisão da Aplicação

A imagem abaixo é um esboça que mostra de que maneira é que a nossa aplicação se encontra dividida.

Uma imagem com mesa

Descrição gerada automaticamente

# Componentes

Neste trabalho utilizámos componentes funcionais para representar várias partes da aplicação. Maior parte da lógica de jogo foi feita no ficheiro App.js e criámos várias variáveis de estado (utilizando o *useState* hook) que são passadas como propriedades (props) para os vários componentes.

## Board

O componente **Board** representa o tabuleiro onde as várias letras vão ficar durante o jogo. Para este componente são passadas as seguintes variáveis de estado:

* **selectedLevel:** dificuldade selecionada
* **board:** tabuleiro bidimensional com as letras
* **handleDragStart:** função que processa o evento de começar a arrastar uma peça
* **handleDragEnter:** função que processa o evento de passar o rato por cima de uma peça ao arrastar a primeira
* **handleDragEnd:** função que processa o evento de deixar de arrastar uma peça

Retornamos um div com os componentes **Letter** de que abordaremos mais à frente neste relatório.





## ControlPanel

O componente **ControlPanel**, é, como o nome indica, o painel de controlo onde o jogador consegue configurar várias opções de jogo e visualizar informação relevante acerca do mesmo.

Este componente contém informação relativa à tabela de pontuações, um seletor de dificuldade, uma secção para introduzir palavras extras, uma secção para definir o nome de jogador e um botão para começar/parar o jogo.

 

As propriedades (props) passadas para este componente são as seguintes:

* **gameStarted:** indica se o jogo está a decorrer ou não
* **onGameStart:** função que é chamada quando o botão de iniciar/parar o jogo é clicado
* **selectedLevel:** indica a dificuldade de jogo selecionada
* **onLevelChange:** função que é chamada quando a dificuldade é trocada
* **extraWords:** representa as palavras extra adicionadas
* **onAddWord:** função que é chamada quando o utilizador adiciona uma nova palavra extra
* **playerName:** variável que representa o nome do jogador
* **onChangeName:** função que é chamada quando o jogador muda o nome

## Letter

O componente **Letter** representa uma peça com uma letra que será colocada no tabuleiro ao lado de tantas outras.

Este componente apenas é chamada a partir do componente **Board** tantas vezes quanto forem necessárias para preencher o tabuleiro dependendo do nível selecionado.

As propriedades que este componente recebe são as funções que são chamadas dependendo da ocorrência de um certo evento e é passada também a letra que é para ser colocada nessa peça.





## Timer

O componente Timer é um componente simples que apenas apresenta o tempo de jogo restante até este terminar.

Este componente recebe as seguintes variáveis de estado:

* **timer:** variável que guarda o tempo de jogo restante
* **gameStarted:** variável que indica se o jogo se encontra em execução
* **selectedLevel:** variável que indica a dificuldade selecionada





## Words

Este componente contém as palavras que o jogador precisa de encontrar de forma a ganhar o jogo. O mesmo é apresentado na lateral direita da aplicação e as propriedades que lhe são passadas são o array com as palavras a encontrar (**words**) e a variável que indica se o jogo se encontra em execução (**gameStarted)**.





# Limitações Conhecidas

Neste trabalho, felizmente, existem muito poucas limitações. As limitações de que temos conhecimento são as que se seguem:

* Se o jogo acabar por tempo e o jogador estiver a arrastar uma peça, as peças selecionadas ficam marcadas no tabuleiro mesmo após o jogo terminar e estas só desaparecem quando é começado um novo jogo
* Ao selecionar as letras que compõem uma palavra, ao arrastar, existe uma possibilidade de algumas letras pelo meio não serem registadas
* Selecionar as palavras na diagonal é bastante complicado

# Desafios

Um dos principais desafios deste trabalho foi implementar um algoritmo que permitisse ao jogador selecionar as palavras que ia encontrando durante o desenrolar do jogo.

Para resolver este problema, utilizámos alguns *drag events*. Para o jogador selecionar a palavra basta clicar e arrastar do início ao fim da mesma. O que realmente acontece “por de trás dos panos” é o seguinte:

* Ao arrastar a primeira, a mesma é adicionada ao um array de letras encontradas (se a mesma ainda não lá estiver) – *evento onDragStart*
* Seguidamente, sempre que a letra que estamos a arrastar passe por cima de outra letra, adicionamos essa letra ao array de letras encontradas (se a mesma ainda não lá estiver) – *evento onDragEnter*
* Finalmente, quando o jogador largar o rato (deixar de arrastar), verificamos se as letras que se encontram no array de letras encontradas formam alguma palavra que ainda não tenha sido encontrada – *evento onDragEnd*

Uma imagem com texto

Descrição gerada automaticamente

Outro desafio encontrado foi: como posicionar as palavras na grelha. Isto foi algo que deu que pensar, mas no final de contas não era assim tão complicado de implementar.

O algoritmo para posicionar uma palavra na grelha funciona da seguinte forma:

* Gerar uma posição aleatória no tabuleiro e uma direção aleatória
* Dependendo da direção, verificamos se a posição onde a letra se encontra somada com o tamanho da palavra não ultrapassa o tamanho do tabuleiro
  + Se isso acontecer, voltamos ao início deste algoritmo
* Começamos a colocar letra a letra. O espeço onde vamos colocar a letra tem de estar vazio ou se existir lá uma letra, esta tem de ser igual à que queremos colocar
* Se conseguirmos colocar todas as letras, a palavra foi colocada com sucesso, mas caso tenha havido uma letra que não tenha conseguido ser colocada voltamos ao início deste algoritmo

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente