Relatório Linguagens Script – "Minesweeper"

Figura 1 – Interface do jogo Minesweeper e nível de jogo de intermédio (screenshot do jogo)

> *Resumo*

Este trabalho prático tem como objetivo o desenvolvimento de um jogo "Minesweeper" em **React JS**, cujo objetivo é desativar/encontrar todas as minas existentes num tabuleiro. Este jogo terá que ter 3 níveis: básico: 9x9 –10 minas | intermédio: 16x16 – 40 minas | avançado: 30x16 – 99 minas.

> *Equipa de trabalho*

O trabalho prático foi realizado pela seguinte equipa:

- Nuno Tomás Paiva

> *Components*

O jogo a implementar deverá ser acessível através do browser e deverá disponibilizar, pelo menos, as seguintes **funcionalidades**:

- Escolher o nível de jogo, que mudar o tamanho do tabuleiro e o número de minas;
- Deverá apresentar o número de minas a encontrar e o tempo em jogo;
- Identificação de fim de jogo, quando for selecionada uma mina ou todas as minas forem identificadas;
- E um botão de reset que permita jogar novamente.

> *Limitações conhecidas*

Acredito que este trabalho não tem nenhuma limitação.

> *Desafios*

Neste trabalho a principal limitação foi logo ao início, em colocar um array multidimensional num grid.

Outra limitação foi fazer com que se a célula selecionada não tivesse minas adjacente, procurar as células próximas células até que tenham minas.

> *Imagens dos diferentes níveis de jogo*

Básico:

Figura 2 – Campo do nível de jogo Básico

Intermédio:

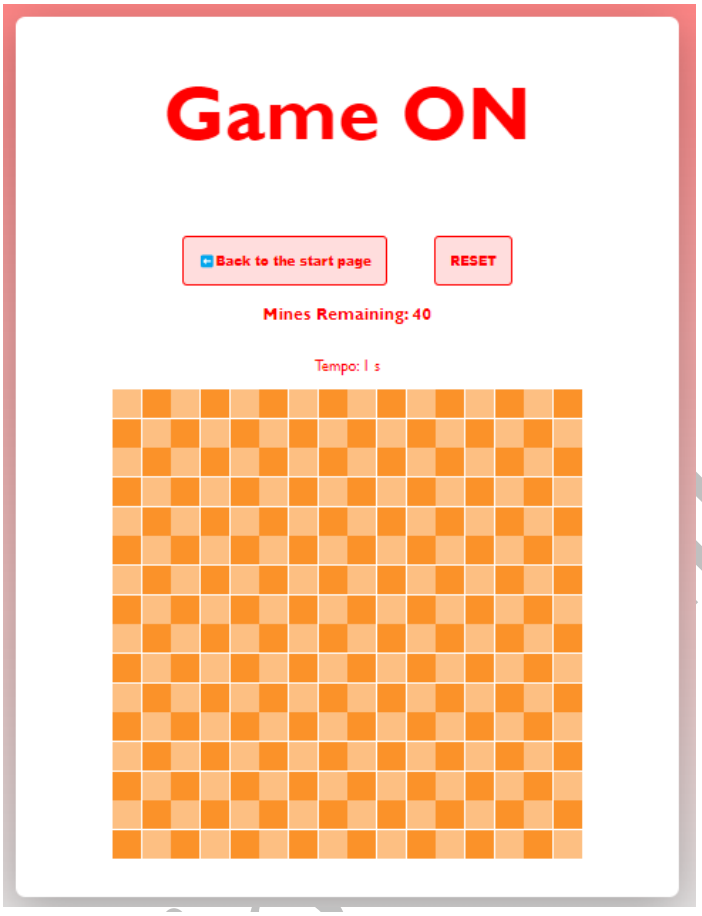


Figura 3 – Campo do nível de jogo Intermédio

Avançado:

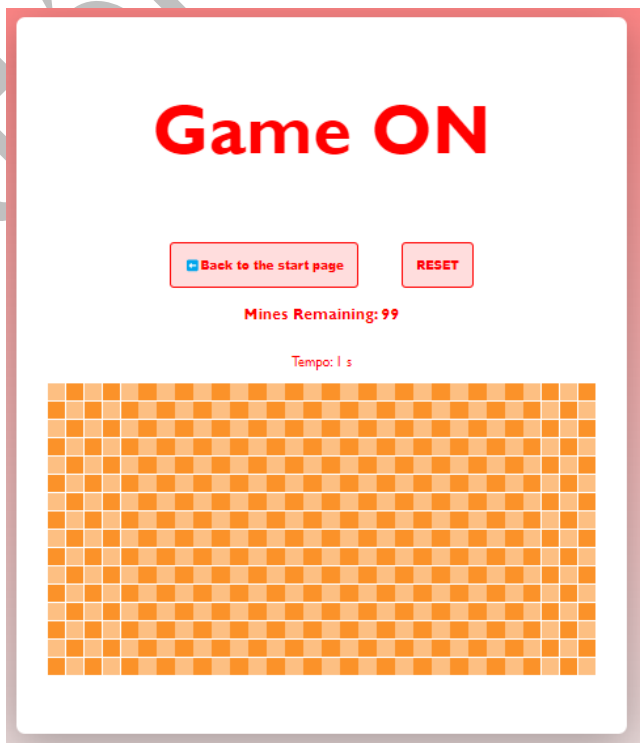
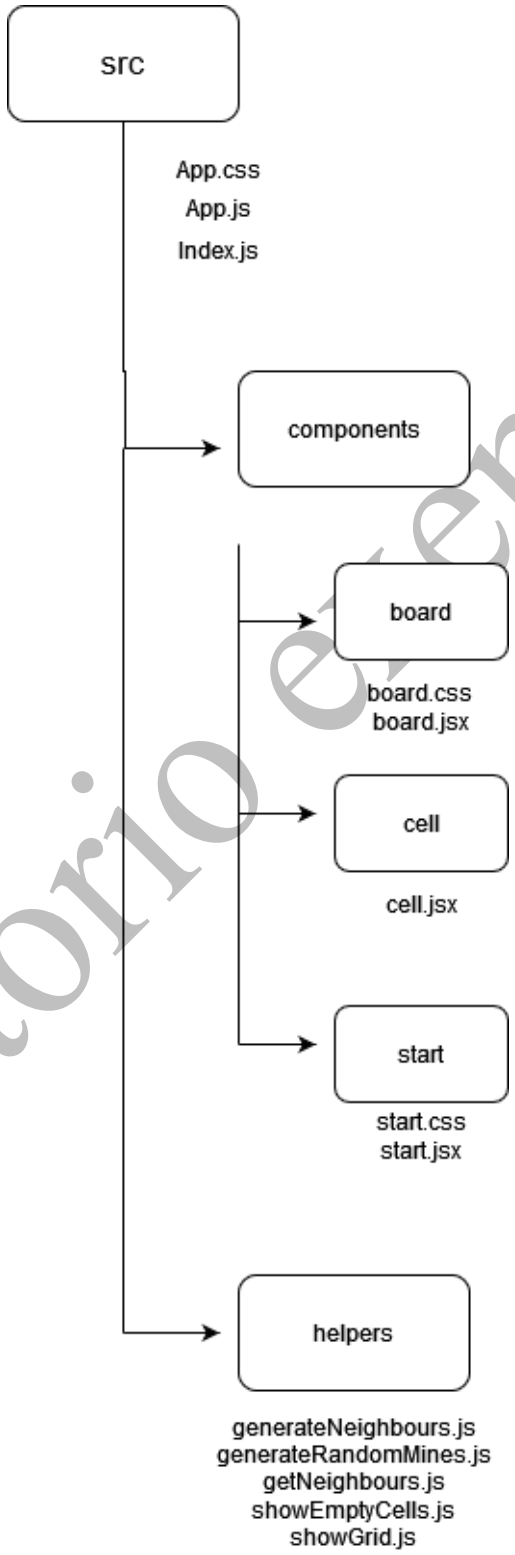


Figura 4 – Campo do nível de jogo Avançado

› *Diagrama de componentes*



> *Funcionalidades do componentes*

O **App.js** decide se o jogo já começou e nos envia para a start page ou para o campo.

O **App.css** dá style à pagina do campo.

No **board.jsx** é onde temos a grande parte da lógica do campo, tal como os eventos left (const onLeftClick) e right click (const onRightClick), a inicialização do campo, o temporizador, o reset do campo e sua devida atualização. Que será "enviada" para o **cell.jsx**, que decidirá que valores apresentar em cada cell.

O **board.css** dá style às cells.

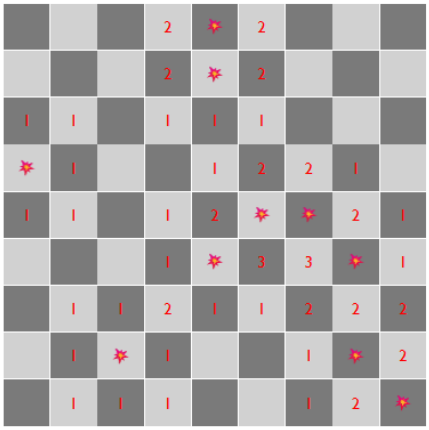
No **start.jsx** é onde é feita a lógica da escolha de nível e assim enviada a sua informação para o **board.jsx**.

Para dar style à página de start, temos o **start.css**.

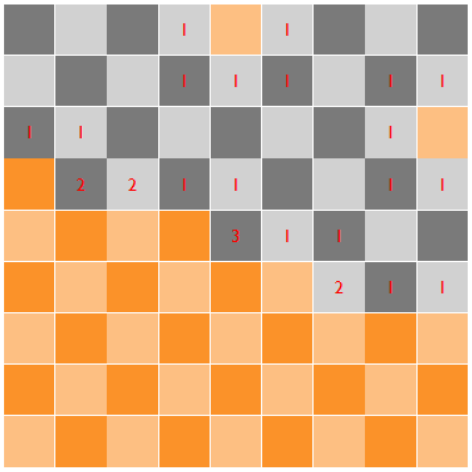
Para ajudar com estes processos temos as funções nos helpers.

A função **generateRandomMines.js** gera minas de forma aleatória ao longo do campo.

A função **generateNeighbours.js** gera o número de vizinhos em cada cell, com a ajuda da função **getNeighbours.js** que identifica as minas no perímetro da cell.



A função **showEmptyCells.js** é uma função recursiva que esvazia as células vizinhas vazias até encontrar as próximas células com indicação de minas adjacentes, esta função usa a função **getNeighbours.js** como ajuda.



A função **showGrid.js** é uma função que mostra o campo todo quando o jogador encontra uma mina, sendo esta função muito simples.

› *Conclusão*

Neste relatório, breve, abordei os principais aspetos relacionados com trabalho feito, expliquei as funcionalidades do mesmo e mostrei alguns problemas de que enfrentei. E concluindo, este trabalho deu-me um muito melhor entendimento sobre **React JS**.