RELATÓRIO

Trabalho Prático

Linguagens Script

Licenciatura em Engenharia Informática

O trabalho prático de Linguagens Script consiste na implementação do jogo da Sopa de Letras utilizando a framework React e a linguagem de programação Javascript.

Trabalho realizado por:

Tomás Gomes Silva - 2020143845

Tomás da Cunha Pinto - 2020144067

Índice

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.

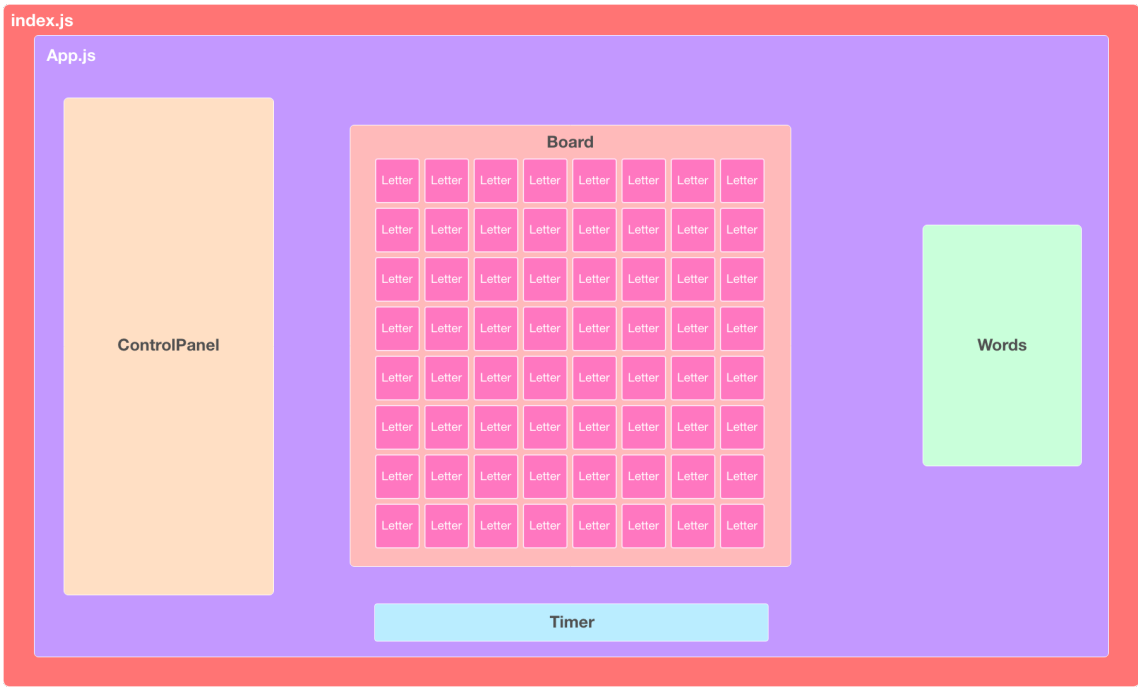
Equipa de Trabalho

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.



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.

```
return (
    <div className={"board " + gridClass}>
        {board.map((row, rowIndex) => {
            return row.map((letter, colIndex) => {
                return <Letter
                            key={rowIndex + "-" + colIndex}
                            letter={letter}
                            handleDragStart={props.handleDragStart}
                            handleDragEnter={props.handleDragEnter}
                            handleDragEnd={props.handleDragEnd}
                        />
            })
        })}
    </div>
)
```

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.

```
return (
  <div id="control-panel">

    <div id="scoreboard">
      <h1>Scoreboard</h1>
      <div className="score" id="score-1"><h1>1. Tomás Pinto (999)</h1></div>
      <div className="score" id="score-2"><h1>2. Tomás Silva (320)</h1></div>
      <div className="score" id="score-3"><h1>3. Professor (300)</h1></div>
    </div>

    <form className="form">
      <fieldset className="form-group">
        <label htmlFor="btLevel">Dificuldade</label>
        <select
          id="btLevel"
          defaultValue="0"
          onChange={props.onLevelChange}
          disabled={props.gameStarted}
        >
          <option value="0">Selecione uma dificuldade...</option>
          <option value="1">Simples (10x10) - 4 palavras</option>
          <option value="2">Intermédio (12x12) - 7 palavras</option>
          <option value="3">Avançado (15x15) - 10 palavras</option>
        </select>
      </fieldset>
    </form>
    <button
        type="button"
        id={props.gameStarted ? "stopBtn" : "startBtn"}
        disabled={props.selectedLevel === "0"}
        onClick={props.onGameStart}
    >{props.gameStarted ? "Parar o jogo" : "Começar o jogo!"}</button>

      <div className="extraWords">
      <input id = "inputWord" placeholder="Introduza uma palavra" disabled={props.ga
      <button
      type="button"
      id= "writeBtn"
      disabled={props.gameStarted || props.selectedLevel === "0"}
      onClick={props.onAddWord}
      >Adicionar Palavra</button>
      <h1>Palavras Extra</h1>
      <h2>{Math.abs(EXTRA_WORDS - props.extraWords.length) + " restante(s)"}</h2>
    </div>

    <div id="playerName">
      <h1>{props.playerName ? props.playerName : "Olá, estranho"}</h1>
```

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.

```
return (
    <div draggable="true"
        onDragStart={props.handleDragStart}
        onDragEnter={props.handleDragEnter}
        onDragEnd={props.handleDragEnd}
        className="piece letterWrap">
        <h1 draggable="false" className="letter">{props.letter}</h1>
    </div>
)
```

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

```
return (
    <div className="timer">
        <i className={"fa-solid fa-clock " + timerClass}></i>
        <span> Tempo Restante: </span>
        <span className={timerClass}>{timer}</span>
    </div>
)
```

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)**.

```
return (
    <div className="words">
        <ul className="words-list">
            {props.words.map(word => {
                if(word.found === false){
                    return (
                        <li key={word.key}
                            className="word"
                            color={COLOR_PALETTE[word.index]}
                            style={{color: COLOR_PALETTE[word.index]}}>
                            {word.word}
                        </li>
                    )
                } else {
                    return (
                        <li key={word.key}
                            className="word"
                            color={COLOR_PALETTE[word.index]}
                            style={{textDecoration: "line-through 3px " +
                                    COLOR_PALETTE[word.index],
                                    color: COLOR_PALETTE[word.index]}}>
                            <s>{word.word}</s>
                        </li>
                    )
                }
            })}
        </ul>
    </div>
)
```

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*

```
const handleDragEnd = (event) => {
  if(gameStarted){
    if(collectedLetters.length > 1){
      var completeWord = collectedLetters.map(letter => letter.innerText).join("");
      var wordFound = undefined;
      words.forEach(word => {
        if(word.word.toLowerCase() === completeWord.toLowerCase()){
          word.found = true;
          wordFound = word;
        }
      });
      if (wordFound){
        collectedLetters.forEach(element => {
          element.className = "piece letterWrap";
          element.style.backgroundColor = COLOR_PALETTE[wordFound.index];
        });
      } else {
        collectedLetters.forEach(element => {
          element.className = "piece letterWrap";
        });
      }
    } else {
      collectedLetters.forEach(element => {
        element.className = "piece letterWrap";
        element.style.cssText = "";
      });
    }
    setFoundWords(foundWords + (wordFound ? 1 : 0));
    totalCollectedLetters.push(...collectedLetters);
    setCollectedLetters([]);
  } else {
    event.preventDefault();
    return false;
  }
}
```

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

```
words.forEach((word, index) => {
  do {
    var currentWord = word.word;
    var wordLength = currentWord.length;
    var randomX = Math.floor(Math.random() * tabDim);
    var randomY = Math.floor(Math.random() * tabDim);
    var randomDirection = Math.floor(Math.random() * 8);
    //console.log("Try to place word: " + currentWord + " at position: " + randomX + "," + randomY + " in direction: " + randomDirection);
  } while (!placeWord(currentWord, wordLength, randomX, randomY, randomDirection));
  //console.info("Word placed: " + currentWord + " at position: " + randomX + "," + randomY + " in direction: " + randomDirection);
  word.x = randomX;
  word.y = randomY;
  word.direction = randomDirection;
});
```

```
case 4: // South to North
  if(xPosition - wordLength >= 0){
    for(i = 0; i < wordLength; i++){
      if(currentBoard[xPosition - i][yPosition] === "" || currentBoard[xPosition - i][yPosition] === word[i]){
        currentBoard[xPosition - i][yPosition] = word[i];
      } else {
        for(j = 0; j < i; j++){
          currentBoard[xPosition - j][yPosition] = "";
        }
        setBoard(currentBoard);
        return false;
      };
    }
    setBoard(currentBoard);
    return true;
  } else return false;
```