> Ficha Prática Nº11 (React – Logica das cardas do Jogo de Memória)

Durante as últimas aulas práticas, foram introduzidos vários conceitos para implementação do Jogo de Memória. Entre eles, criação de ***componentes*** React, ***props*** e ***Hooks***, como o **useState** e o **useEffect**, que permitem a criação de variáveis de estado e implementar a gestão de efeitos "secundários", no ciclo de vida do componente, respetivamente. Esta ficha, pretende dar continuidade e concluir o Jogo de Memória em React, aplicando todos esses conceitos, bem como criação de várias funções *javascript*, de forma a implementar a restante lógica do jogo.

> Preparação do ambiente

a. Efetue o download e descompacte o ficheiro **ficha11.zip** disponível no *inforestudante.*

NOTA: *Os alunos que desejarem, podem continuar a resolução da ficha 10 que implementaram na última aula. Devem, no entanto, eliminar a class "**flipped**" aplicada, no componente **Card.***

***Os alunos que optem pelo download do código fornecido nesta ficha**, devem ter em consideração, que apenas está disponível a versão para implementação com o nível básico. Recomenda-se a resolução das fichas anteriores para quem pretender o jogo na sua totalidade.*

b. Inicie o *Visual Studio Code* e abra **a pasta no *workspace***.

c. No terminal, digite os seguintes comandos:

- → **`npm install`** (apenas quem usa ficha11.zip fornecido)
- → **`npm start`** (para iniciar a aplicação)

d. Visualize a página no browser, endereço http://localhost:3000/,
que deverá ter o aspeto da figura 3.

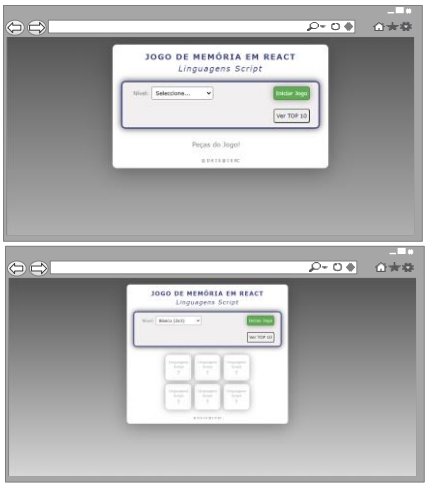
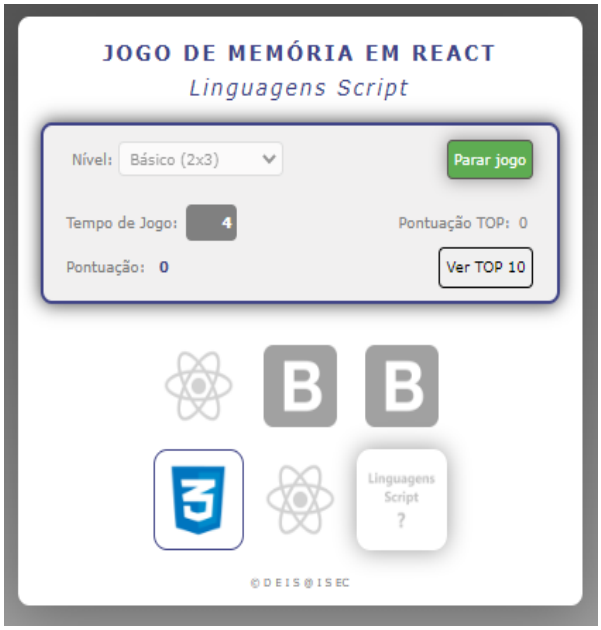
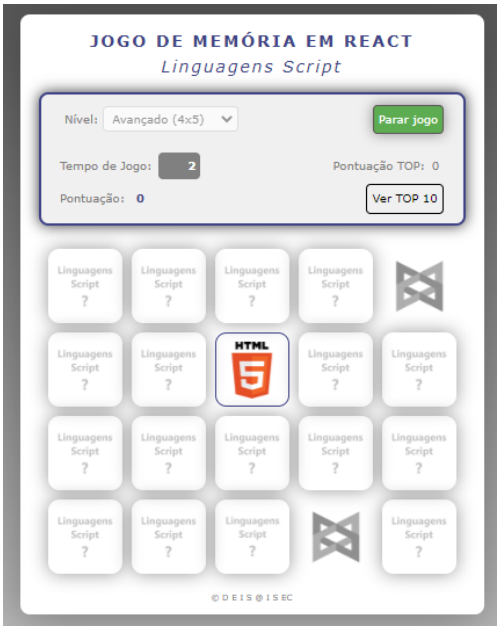


Figura 3 – Estado Inicial da Aplicação

Parte I – Rotação das Cartas e fim de Jogo

Para implementar esta parte de código implemente os seguintes passos.





1> De forma a implementar a lógica do jogo de memória, serão usados dois arrays, um com as cartas rodadas, que irá ter no máximo duas cartas, e um array com as cartas com o par encontrado.

a. No componente **GamePanel**, implemente os seguintes passos:

- → Especifique duas variáveis de estados:
  1. `flippedCards`, em que o estado inicial é um array vazio;
  2. `matchedCards`, em que o estado inicial é um array vazio;
- → Implemente a função callback **handleClickCard** que quando o jogo se econtra iniciado, atualiza a variável de estado, acrescentado a carta às flippedCards;
- → Copie a seguinte função que verifica se duas cartas viradas e se sim atualiza a variável de estado das cartas dos pares encontrados e a das cartas rodadas:

```
const processMatchingCards = () => {
    const [card1, card2] = flippedCards;
    const cardsAreEqual = card1.name === card2.name;
    if (cardsAreEqual) {
      setTimeout(() => {
        setMatchedCards((previousState) =>
[...previousState, ...flippedCards]);
        setFlippedCards([]);
      }, 500);
    } else {
      setTimeout(() => {
        setFlippedCards([]);
      }, 500);
    }
  };
```

- Implemente os seguintes useEffects:
  1. Sempre que um jogo inicia, eliminar todos os dados existentes nas variáveis de estado `flippedCards` e `matchedCards`
  2. Terminar o jogo quando a variável de estado *matchedCards* é atualizada. O fim de jogo pode ser verificado se o número de cartas nesse array é igual ao numero de cardas do painel;
  3. Processar a função `processMatchingCards` quando o jogo estiver em curso e o array das cartas virada tiver dois elementos;
  4. `matchedCards`, em que o estado inicial é um array vazio;
- Especifique todos os imports necessários de forma que possa usar o hook useEffects;
- Invoque o componente card da seguinte forma, não esquecendo de importar as funções `check*` que se encontram nos helpers:

```
<Card
   key={card.key}
   card={card}
   onClickCard={handleClickCard}
   flipped={checkIfIsFlipped(matchedCards, flippedCards,card.id)}
   matched={checkIfIsMatched(matchedCards, card.id)}
  />
```

b. No componente **Card**, efetue as seguintes alterações:

- Altere o código necessário de forma a obter correctamente as props;
- Especifique as variáveis `flippedClass, matchedClass` e `cardFrontClass`, o qual deverão ficar com o valor " flipped" ," inative" caso as propriedades correspondentes *matched* e *flipped* forem verdadeiras. Estas variáveis devem ser aplicadas na className do div.
- Especifique a variável `cardFrontClass`, o qual deverá ficar com o "grayscale" caso a propriedade *matched* for verdadeira. Esta deverá ser aplicada na className do img card-front;
- Adicione o seguinte código de forma a não serem aceite mais cliques do rato, se por acaso a carta estiver virada.

```
const handleClickCaptureCard = (event) => {
    if (flipped) {
      event.stopPropagation();
    }
  };
```

- Adicione os seguintes atributos ao div.

```
onClick={() => {
        onClickCard(card);
}}
      onClickCapture={handleClickCaptureCard}
```

c. No componente **App**, implemente os seguintes passos:

→ Invoque o componente **GamePanel** da seguinte forma:

```
<GamePanel cards={cards}
  selectedLevel={selectedLevel}
  gameStarted={gameStarted}
  onGameStart={handleGameStart}
/>
```

d. Ao iniciar novo jogo, certifique-se que realente o painel foi atualizado. Caso contrario, efetue as devidas alterações para geração de um novo painel sempre que inicia um jogo.

2> Especifique o código necessário para que a janela modal de fim de jogo seja apresentada com o valor d pontuação. Note que o componente já se encontra implementado.

Parte II – Pontuação de Jogo

Implemente o código para atualizar a pontuação do jogo. Deverá especificar a variável de estado **totalPoints** e a função que permite atualizar a pontuação é a seguinte:

```
// Pontuação: numero de cartas/2 * o tempo (quanto maior o nível mais ganha!)
// Desconta 5 pontos sempre q não faz par.
const updatePoints = (operacaoSoma = true) => {
  let pointsSum = totalPoints;

  if (operacaoSoma) {
    pointsSum += timer * (cards.length / 2);
  } else {
    pointsSum < 5 ? (pointsSum = 0) : (pointsSum -= 5);
  }
  setTotalPoints(pointsSum);
```

Parte III – Alteração do Timer para o componente Control Panel

3> O código para contador do tempo foi especificado no componente **App**. Efetue as alterações necessárias de forma a que o mesmo se encontre no componente **ControlPanel**.