MINIPROJETO M2A

Trabalho Individual

Daniel Figueiredo
98498

URL's do Projeto

Gitlab: https://gitlab.com/pw-mctw/tdw-2022/tdw-mp2-dani-figueiredo
Site Deployed: https://dani-figueiredo-mp2.tdw-mctw.dev/

Tecnologias e Desenvolvimento Web
Dezembro 15, 2022

Índice

Introdução

No âmbito da unidade curricular de Tecnologias e Desenvolvimento Web, foi-nos proposto a realização de um projeto que consistia no desenvolvimento de uma aplicação de *front-end* em React para consumir e visualizar informação de uma qualquer fonte pública de dados.

Apresentação do projeto e da temática escolhida

A fonte pública de dados que decidi utilizar foi a PokeApi (https://pokeapi.co/), que tem informações acerca de todos os Pokémons existentes e as suas respectivas informações. Sendo que também tem outras informações tal como as *berries*, *encounters, evolution, locations* e mais algumas informações.

Apesar de ter tanta informação, para a minha aplicação, apenas decidi utilizar os Pokémons, bem como as informações de cada um e também na página de detalhes de cada Pókemon ter a lista das suas evoluções.

Design e estrutura da aplicação

Em termos de design da aplicação, tenho uma página inicial com um *screen* que mostra as informações do projeto e de que este se trata. Possui um botão que vai permitir "abrir" a Pokédex e assim mostrar a listagem dos Pokémons. Ao clicar num Pokémon abre uma página com as informações do Pokémon selecionado.

Também na página da Pokédex é possível fazer a utilização de filtros, tais como filtros de pesquisa, como pesquisa por nome e por ID e também pesquisa por tipo de Pokémon, também existem filtros de ordenação que permitem ordenar por ID, ordem alfabética e por altura ou peso.

Caso os Pokémons estejam a fazer o *load* da API é possível ver uma componente de *Loading* (Fig. 1) e caso haja algum erro é possível ver a componente de *Error* (Fig. 2), sendo que a aplicação e todas as páginas são relacionadas ao tema em questão.

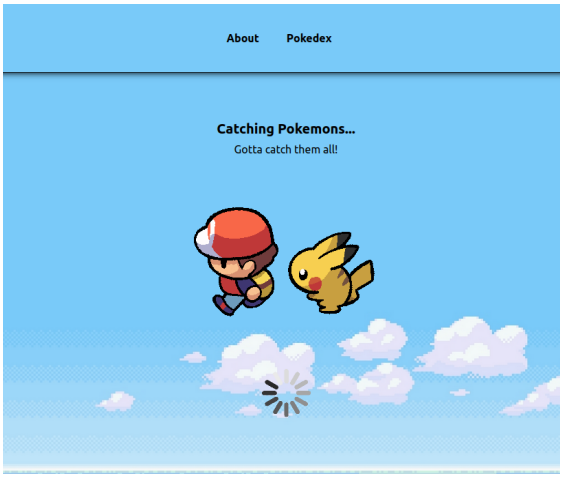

Fig. 1 - Loading Page

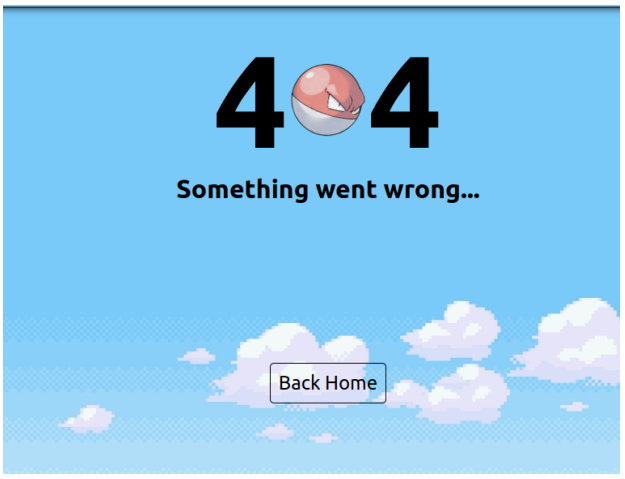

Fig. 2 - Error Page

Estratégia de implementação técnica

Organizacao do projeto

O projeto encontra-se organizado em diferentes pastas, como por exemplos as páginas do projeto (About, Pokedex, PokemonDetail) estão dentro de uma pasta chamada *pages,* sendo que as componentes usadas em cada uma destas páginas se encontram na pasta *components*. Os ficheiros necessários para a integração com o Redux encontram-se em outra pasta chamada *store.*

Decomposição de componentes complexos em múltiplos componentes

De modo a que algumas componentes não ficassem muito extensas e complexas, algumas componentes tiveram de sofrer um processo de *refactor*, como por exemplo da componente dos filtros e da componente da listagem de Pokémons, sendo divididas em outras componentes e colocadas em sub-pastas que se referem a estas componentes, *Filters e Pokemons,* respetivamente.

Aplicação apropriada de estilos por componente

Para fazer a estilização da minha aplicação, utilizei *styled components*, que permite criar estilos globais e que pude utilizar ao longo de várias componentes, bem como estilos individuais que estão definidos na própria componente.

Utilização de soluções apropriadas para gestão de ações assíncronas

Para fazer o tratamento da gestão de ações assíncronas foi utilizado o Redux-Thunk de forma a servir como *middleware* o que permitiu escreveu ações que fossem apenas executadas quando alguma condição fosse verificada, por exemplo.

Implementação de funcionalidades extra que tenham em vista melhorar a experiência do utilizador

De modo a melhorar a experiência do utilizador foi implementado um sistema de paginação de modo a que ao utilizador apenas fosse mostrado uma pequena lista de Pokémons e não uma lista exaustiva com centenas de Pokémons.

Desenvolvimento de uma pipeline de integração contínua de suporte à aplicação

Foi desenvolvida uma *pipeline* de integração contínua no Gitlab, através de conhecimentos previamente adquiridos nesta UC, sendo que a *pipeline* tem vários *stages*: *install*, *validation* (*prettier*), *build* e *deploy*, sendo que o *deploy* da aplicação está a ser feito para o *Netlify* e a aplicação está *deployed* no seguinte link: **https://dani-figueiredo-mp2.tdw-mctw.dev/** .

Utilização apropriada do git

Foi feita uma utilização apropriada do git, visto que foram criados *branches* para as respectivas *features* e *bugs* e em cada um destes *branches* foram feitos vários *commits*, sendo que posteriormente cada um desses *branches* foi *merged* com o *main branch.*

Principais decisões tomadas, desafios técnicos e respetivas soluções

Como a API do Pokémon, não tem um sistema de pesquisa de Pokémon, tive de fazer esta implementação e deste modo, para procurar pela lista toda de Pokémons tive de fazer um *fetch* de todos os Pokémons da API, com a ajuda do Redux e guardar numa lista estes Pokémons todos, assim tive de criar outra lista *filteredList* que apenas me ia mostrar os Pokémons que satisfaziam as condições necessárias e deste modo encontrei a solução para os filtros, tanto de pesquisa como de ordenação.

Obstáculos não ultrapassados

Como tinha de estar a fazer o *fetch* de uma quantidade tão grande de informação isso ia gerar um problema de eficiência que foi mais ou menos ultrapassado, visto que em vez de estar sempre a fazer o pedido, na primeira vez que esse pedido fosse feito essa lista de Pokémons era guardada em *local storage*, permitindo melhorar a eficiência.

Conclusões

Através da realização deste projeto, pude compreender ainda melhor de que forma a utilização do Redux pode ajudar na manutenção dos estados de uma aplicação mais complexa, de modo a não se perder pelo meio estados e de modo a não deixar o código uma confusão. Sendo assim, concluo que o Redux é uma ferramenta muito poderosa que permite não perder o fluxo dos estados da aplicação.