TDW - Módulo A - Miniprojeto React Avançado GitHub Users

André da Silva Rato (115030) - GITLAB & APP

15 de dezembro de 2022

1 Introdução

O presente relatório pretende clarificar e descrever os tópicos mais importantes da implementação do projeto desenvolvido no módulo 2A da unidade curricular **Tecnologias e Desenvolvimento Web**, no âmbito do **Mestrado em Comunicações e Tecnologias Web** da Universidade de Aveiro.

De modo a demonstrar o domínio dos conceitos lecionados durante as aulas da UC, decidi implementar uma *web app* de pesquisa de utilizadores do GitHub, onde é possível obter as informações gerais de cada utilizar, e aceder ao perfil do mesmo.

1.1 Next.js

A tecnologia escolhida para a implementação do projeto foi **Next.js**, uma *framework* desenvolvida e mantida pela equipa Vercel, e que é caracterizada principalmente pela sua semelhança ao React.js e pela possibilidade da renderização de aplicações pelo servidor (*server-side rendering*).

1.1.1 Routing

O **Next.js** possui um sistema de *routing* baseado em pastas, o que significa que qualquer ficheiro que seja adicionado ao diretório '`pages`' será considerado uma página.

A estrutura utilizada no projeto foi a seguinte:

```
(root)
    /pages
        /user
            [slug].tsx (página de informação do utilizador, tendo em conta o id)
        _app.tsx (layout principal da app)
        404.tsx (página 404)
        about.tsx
        index.tsx
```

2 Redux

O **Redux**, uma biblioteca de Javascript utilizada para a centralização de estados, foi a ferramenta escolhida para a gestão do estado global de toda a aplicação. Assim, foi possível criar o seguinte estado:

```
status:   indica o estado e que a aplicação se encontra (pode ser "default", "loading"
          ou "error")
users:    armazena a informação correspondente a cada um dos utilizadores
search:   armazena a informação correspondente a cada um dos termos pesquisados, sendo
          que cada termo possui a seguinte estrutura:
                [termo-de-pesquisa]: [
                    [número-total-de-páginas],
                    [20-utilizadores-da-página-1],
                    [20-utilizadores-da-página-2],
                    ... ]
```

3 API

A API escolhida foi a GitHub API, cuja documentação se encontra aqui. Foram apenas utilizados 2 *endpoints* no projeto, o `.../search/users?q=[termo-de-pesquisa]`, para obter os utilizadores correspondentes a uma pesquisa, e o `.../users/[username]`, para obter os dados de um certo utilizador.

4 *App styling*

Para estilização da *app* foi utilizado **SCSS**, devido à possibilidade de criação de *nested rules*. Recorri então ao *approach* de criar um ficheiro SCSS global, utilizado para definir *global styling rules* e importar os restantes ficheiros SCSS (`_colors.scss`, `_media.scss` e ficheiros de estilização dos vários componentes).

5 *App pages*

A *app* está dividida em 2 páginas principais, a *homepage* e a página de *about*. Existe ainda uma página dinâmica relacionada com o perfil do utilizador, e uma página de erro para páginas não existentes.

5.1 *Homepage*

Nesta página, é possível pesquisar-se utilizadores do GitHub pelo nome. Existe paginação controlada pelo Redux.js e por um estado local do componente. Caso seja necessário realizar-se um pedido à API para obtenção de novos dados, a aplicação fica em *loading*, sendo que é mostrado ao utilizador uma pequena animação de tal.

5.2 *User page*

Esta página é responsável por apresentar os detalhes gerais do utilizador passado no *url*, permitindo ainda navegar para o perfil do mesmo. Enquanto está a ser realizado um pedido, a aplicação fica com um *status* de *loading*, e a é mostrada uma pequena aniamação alusiva a tal.

5.3 404 *page*

Esta página é renderizada em três situações: caso a rota acedida não possua um ficheiro associado, caso o id passado à página do utilizador seja inválido, ou caso exista algum erro na pesquisa dos utilizadores. A mensagem disponibilizada na página é adequada ao erro lançado. O próprio Next.js renderiza a página 404 apenas se o *url* acedido não corresponder a uma página definida. Já em termos de id inválido a renderização é manual.

6 *Gitlab's pipeline*

Com o objetivo de gerar uma visualização real, num ambiente de produção, e de forma automática, foi implementada uma *pipeline* no projeto, de modo a que a cada *commit*, um novo *job* fosse iniciado e uma nova versão da app ficasse disponível. Assim, e com o intuito de possuir um ambiente de desenvolvimento e um ambiente de produção associados a *branches*, foram criados 2 *steps* independentes, um executado apenas quando o *branch* `dev` recebe atualizções (*commits*), e outro executado apenas quando o *branch* `main` recebe *commits* novos. Ambos os *steps* enviam uma *build* para a plataforma Vercel, que gera *urls* novos para cada *commit*, e utiliza o conteúdo do *branch* `main` como conteúdo de produção.

É de notar também que foi necessário definir algumas variáveis de ambiente no Gitlab para que a *pipeline* fosse executada sem quaisquer problemas:

- `VERCEL_ORG_ID`
- `VERCEL_PROJECT_ID`
- `VERCEL_TOKEN`

7 Responsividade

A responsidividade da *app* foi outro tema abordado ao longo do desenvolvimento da mesma, sendo que foi definido 1 *breakpoint* no ficheiro `/styles/_media.scss`, *breakpoint* este que foi utilizado para aplicar regras de estilização em dispositivos com um máximo de 900 píxeis de largura. Este *breakpoint* foi definido com auxílio de um *mixin*, uma diretiva que permite criar estilos para utilização em toda a *app*.

8 Problemas durante o desenvolvimento do projeto

Inicialmente, a ideia geral era implementar um índice de cartas de Pokémon. Foi tudo implementado, mas após rever tudo e avaliar o mesmo, decidi recriar o projeto e optar por algo mais simples, mas mais bem conseguido a nível de implementação e escolhas para a mesma. O código fonte deste projeto rejeitado encontra-se no Gitlab, sendo que o resultado final pode sr visualizado na Vercel.

No geral, o desenvolvimento do projeto foi fácil. No entanto, a maior adversidade que encontrei foi a configuração da *pipeline* do projeto, pois esta foi mais demorada que o previsto, sendo que o resultado final da mesma superou o esperado, sendo possível então simular o ambiente de dsenvolvimento a que estou habituado na empresa onde trabalho (um ambiente de desenvolvimento e outro de produção, ligados a 2 *branches* protegidos no Gitlab).

9 Conclusão

Com a realização deste trabalho foi possível, não só melhorar o meu conhecimento realtivo a React.js e a Next.js, mas também testá-lo na prática. Através da utilização de Next.js foi-me permitido entender melhor como funciona o *routing* do mesmo, entender as diferenças entre *server-side* e *client-side components*, e mudar o meu método de programação, tornando-o mais orientado para a melhoria da performance da *app*.

Para além disso, acabei por desenvolver um pouco mais a minha *skill* criativa que acaba por ser o meu ponto fraco. No entanto, acredito que tenho desenvolvido um projeto interessante, não só do ponto de vista tecnológico, mas também do ponto de vista visual.