

Mini-Projeto do Módulo Javascript: Relatório

Rodrigo Martins Graça [107634]

October 27, 2024

Recursos	URL's
Game	https://rodrigograc4.github.io/TDW/Mini_Projeto/
Git Repository	https://github.com/rodrigograc4/TDW/tree/main/Mini_Projeto

Contents

1	Introdução	3
1.1	O que é o Poke Apollo	3
2	Elementos Dinâmicos	3
2.1	Loading Page	3
2.2	Página Inicial	3
2.3	Escolha de Pokémons e Ataques	4
2.4	Sistema de Batalhas	4
2.5	Highscores	4
3	Estratégia Geral de Implementação	5
3.1	Configuração do Firebase	5
3.2	API de Dados dos Pokémons	5
3.3	Ciclos do Jogo	5
4	Principais Desafios Técnicos e Soluções	5
4.1	Integração com Firebase	5
4.2	Lógica da batalha dinâmica	6
4.3	Esconder os botões	6
4.4	Responsividade	6
5	Obstáculos Não Ultrapassados	7
5.1	Status Moves e Tipos	7
6	Conclusão	7

1 Introdução

1.1 O que é o Poke Apollo

Poke Apollo é um jogo de batalhas Pokémon desenvolvido como parte do módulo M1B da disciplina de Tecnologias de Desenvolvimento Web. Inicialmente, o objetivo é aplicar os conhecimentos de JavaScript, uso de APIs externas e estruturação de interfaces dinâmicas.

Neste jogo, o jogador vai escolher o seu nome, pode escolher um Pokémon e também os seus ataques antes de entrar em combate contra outros Pokémon gerados aleatoriamente.

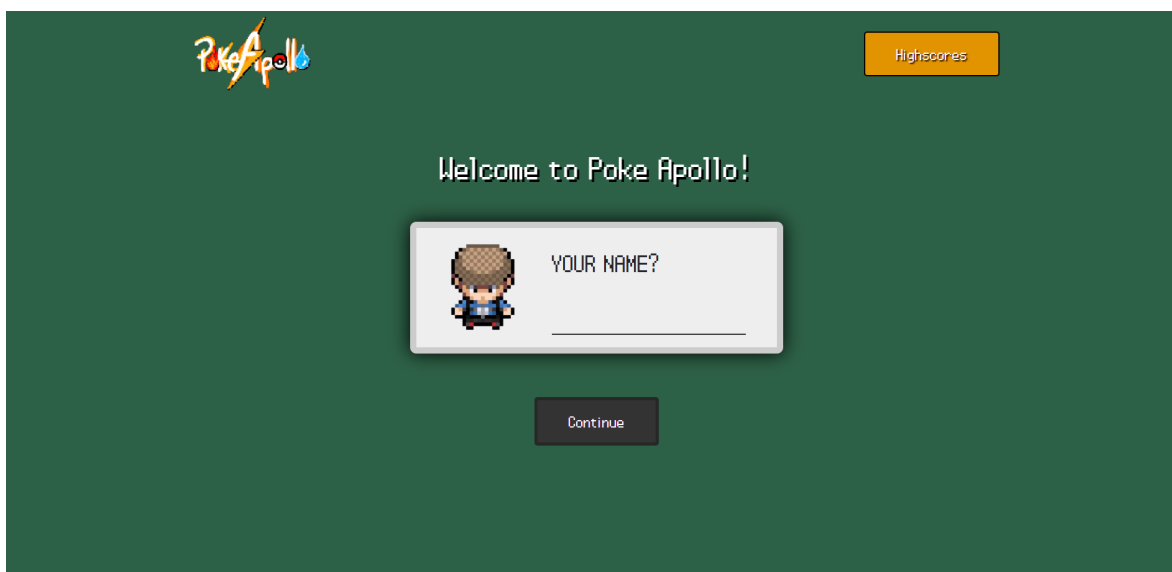
2 Elementos Dinâmicos

2.1 Loading Page

Ao iniciar o jogo, uma tela de carregamento animada com estrelas aparece brevemente. As estrelas são geradas e posicionadas aleatoriamente a cada vez que a página é carregada. Após 2 segundos, a tela de carregamento desaparece e o jogador é direcionado para o jogo. Este elemento foi implementado através de JavaScript e manipulação dinâmica do DOM para adicionar as estrelas ao fundo.

2.2 Página Inicial

Uma tela de boas-vindas pede o nome ao jogador, que será usado para identificar o progresso e os resultados de batalhas. Assim como para guardar o seu highscore no fim das batalhas.



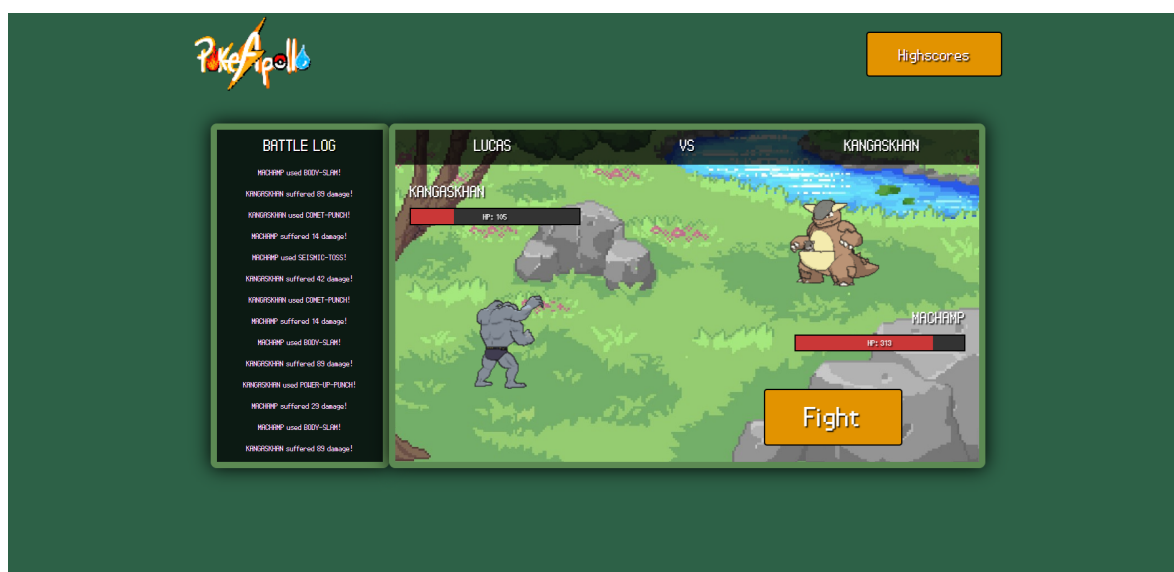
2.3 Escolha de Pokémons e Ataques

O jogador escolhe um Pokémon de uma lista gerada aleatoriamente via chamada à API, PokeAPI. Nesta escolha é também feita outra chamada para obter as imagens dos respetivos Pokémons. Esta é feita também a uma API cujos donos são a PokeAPI mas localizada num repositório github em vez de num url padrão.

Após a seleção do Pokémon, essa interface é escondida e aparece outra que permite ao jogador a escolha de 4 moves (ataques), opções que vêm da PokeAPI.

2.4 Sistema de Batalhas

A batalha entre o Pokémon do jogador e o inimigo é exibida em tempo real, com barras de vida e um battle log que descreve todos os moves usados e o dano causado. Para este sistema são mais uma vez feitas chamadas à PokeAPI para saber os stats base de cada Pokémon, e através de um cálculo efetuado em Javascript obtém-se a vida máxima de cada Pokémon assim como depois o dano que este vai causar ao Pokémon inimigo.



2.5 Highscores

A cada batalha que o jogador finaliza é executado um método post para uma Database localizada no Firebase Firestore. Isto faz com que, caso o jogador feche o jogo, pelo menos as vitórias que já apresentava são guardadas, não perdendo assim registo de nenhum Highscore.

O jogador pode também entrar na página de Highscores, que através de um método get, permite verificar as pontuações de todos os jogadores que já experimentaram o jogo.

3 Estratégia Geral de Implementação

3.1 Configuração do Firebase

Foi configurado o Firebase para armazenar as pontuações dos jogadores. O código que está nos ficheiros `highscores.js` e `script.js`, gere a autenticação, assim como a leitura/escrita dos dados na coleção "highscores" que está no projeto Poke Apollo que criei no Firestore.

3.2 API de Dados dos Pokémons

O jogo utiliza a PokeAPI para obter todas as informações dos Pokémons, tanto as imagens através de um repositório Github assim como as estatísticas de batalha. URLs configuradas no início dos ficheiros direcionam para as sprites frontais e traseiras, adequadas para exibir tanto o Pokémon do jogador quanto o do inimigo durante as batalhas.

3.3 Ciclos do Jogo

Após a tela inicial, o jogador escolhe um Pokémon e define os seus ataques. Cada escolha é processada por funções JavaScript que manipulam o DOM para criar cards dinâmicos e interativos.

As batalhas no jogo decorrem numa forma de ciclo, em que são efetuados turnos, para o jogador e o seu inimigo escolherem os seus ataques. é utilizada a PokeAPI para ver os stats de velocidade dos Pokemons e verificar qual deveria atacar primeiro. Assim como para o calculo do dano, que necessita de uma função com os stats de ataque do Pokémon que via usar o move e dos de defesa do inimigo.

Quando a batalha acaba, se o jogador ganhar uma vitória é adicionada à Database, e vai novamente começar um novo ciclo, em que aparece um inimigo random e toda uma batalha acontece novamente. No caso do jogador perder, o ciclo acaba e é apresentado um botão de continuar na mesma para que reencaminha para o início do jogo em vez de seguir para outra batalha.

4 Principais Desafios Técnicos e Soluções

4.1 Integração com Firebase

Apesar de já ter trabalhado com o Firebase anteriormente, e a criação da Database para este projeto ter sido bastante 'straight-forward', o mesmo não se verificou para a integração no Javascript.

Anteriormente apenas tinha integrado o Firebase com Python e os problemas vieram pois a forma original de conectar com o Javascript pelos visto já não era possível devido a certas dependências que ao importar o Firebase davam erro.

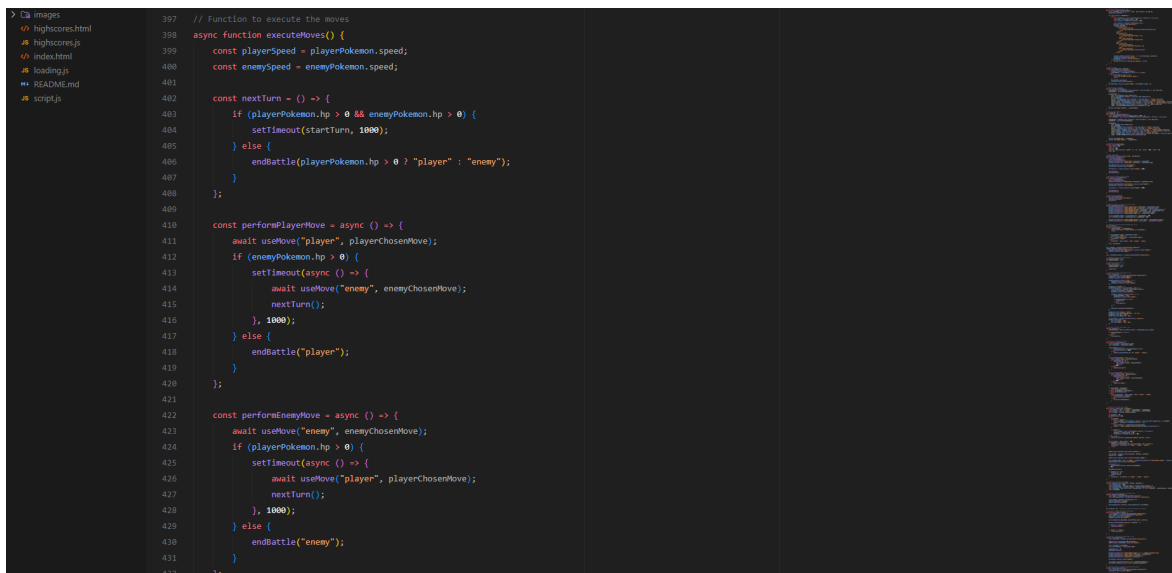
Deu um adicional trabalho de pesquisa mas no StackOverflow consegui encontrar uns 'imports' diretos pelo html feitos por outra fonte mas que funcionaram.

Após isto tive de arranjar forma de não estar constantemente a fazer os pedidos e alterar resultados de forma errada então coloquei umas condições antes do metodo de write/post, em que o score tem de ser maior que o anterior para o mesmo nome para se registar a pontuação nova.

4.2 Lógica da batalha dinâmica

A batalha foi algo que eu fui implementado as features aos poucos, para poder ir ver um resultado sendo concretizado ao mesmo tempo. Mas por conta disso, tive uma dificuldade adicional pois a cada feature tinha de alterar as funções e a forma como elas entravam no ciclo da batalha.

No fim a solução passou por ter numeras diferentes funções, como o startTurn e o executeMoves, que assim permitiram me alterar o turno, usando o Pokémon mais rápido respetivamente o seu move e também, ser possível visualizar esse dano antes do próximo ataque acontecer.



```
397 // function to execute the moves
398 async function executeMoves() {
399   const playerSpeed = playerPokemon.speed;
400   const enemySpeed = enemyPokemon.speed;
401
402   const nextTurn = () => {
403     if (playerPokemon.hp > 0 && enemyPokemon.hp > 0) {
404       setTimeout(startTurn, 1000);
405     } else {
406       endBattle(playerPokemon.hp > 0 ? "player" : "enemy");
407     }
408   };
409
410   const performPlayerMove = async () => {
411     await useMove("player", playerChosenMove);
412     if (enemyPokemon.hp > 0) {
413       setTimeout(async () => {
414         await useMove("enemy", enemyChosenMove);
415         nextTurn();
416       }, 1000);
417     } else {
418       endBattle("player");
419     }
420   };
421
422   const performEnemyMove = async () => {
423     await useMove("enemy", enemyChosenMove);
424     if (playerPokemon.hp > 0) {
425       setTimeout(async () => {
426         await useMove("player", playerChosenMove);
427         nextTurn();
428       }, 1000);
429     } else {
430       endBattle("enemy");
431     }
432   };
433 }
```

4.3 Esconder os botões

Devido a querer fazer o jogo acontecer numa uma página, com muitas interfaces e elementos diferentes, foi necessário estar sempre a esconder uns e mostrar os outros.

Para isso criei uma class em css chamada 'hidden', e com a ajuda das numeras funções como já falei anteriormente, foi possível passo a passo esconder os elementos desnecessários e mostrar os importantes adicionando e removendo a class 'hidden'.

4.4 Responsividade

Por conta de cada pokemon ter um tamanho diferente, mesmo com uma tentativa de responsividade dos botões ao colapsar as tabelas. Na interface da batalha especialmente estava a ser impossivel concretizar a responsividade de forma correta.

Como solução tive de definir as imagens com tamanhos específicos por height e width, não sendo o mais correto pois agora há pokemons pequenos que são apresentados maiores do que deviam mas o problema da responsividade que era mais importante foi resolvido na sua maioria.

Digo na sua maioria pois com um battle log e todos os botes de movimentos apresentados, quando a width chega a um tamanho bastante reduzido torna-se impossível apresentar a batalha como um todo e nesse caso é mostrado um elemento de texto a dizer que não é possível jogar com aquelas dimensões.

5 Obstáculos Não Ultrapassados

5.1 Status Moves e Tipos

Penso que o único obstáculo não ultrapassado, nunca foi algo pensado devido ao reduzido tempo para a concretização do projeto. Seria bastante possível realiza-lo com mais tempo.

No caso dos status moves refiro-me há existirem diversos ataques que não tem o objeto de dar dano ao inimigo mas sim terem outros efeitos. Podem ser muitos mas como exemplo posso referir que existem moves de aumentar stats de ataques, outros que colocam o inimigo a dormir ou paralisado, não podendo assim atacar-nos, etc.

Acerca dos tipos, cada Pokémon e cada move tem um tipo diferente, como Fogo, Elétrico, Água entre outros. Estes tipos têm também influencia no dano que um ataque tem no inimigo, dando alguns um Boost de 2x e outros cortando o dano pela metade (0.5x). Seria uma feature muito engraçada de implementar, mas mais uma vez reforço, era possível fazer, só não dentro do prazo estipulado para este Projeto.

6 Conclusão

Este Projeto incorporou técnicas avançadas de JavaScript e manipulação de APIs, oferecendo uma experiência que me proporcionou diversão ao fazer o jogo e também reforçar a minha aprendizagem.

A integração com o Firebase foi fundamental para implementar um sistema de pontuação competitiva, que incentiva os jogadores a tentarem o seu maior score no Poke Apollo. Ao superar desafios como a lógica complexa de batalhas e a integração de APIs, o projeto desenvolveu ainda mais as minhas skills de criar interfaces dinâmicas e interativas com armazenamento de dados em tempo real.