Poke Trade

Linguagem:

- ReactJs

Funcionamento:

Receber dados dos players que indique quantos pokémons de cada tipo ele quer enviar na troca

Calcular se a soma da base de experiência total dos pokémons escolhidos pelo player 1 está perto da soma da base de experiência total dos pokémons do player 2, assim, verificar se a troca é justa.

Paths:

Criar uma aplicação com três paths diferentes sendo:
{ / } = Home
{ /trade } = Trade
{ /result } = Result

- / : Home, contendo todas as regras e informando o usuário de como a troca funciona, incluindo ícones e interface interativa.

- /trade : Trade, página em que os usuário, tanto o player 1 como o player 2 vão fazer as escolhas de pokémon e a quantidade de cada um.

- /result : Result, página contendo os resultados da trade, nela existe todo o cálculo das bases de experiência e se a troca é justa ou não

Métodos:

- Conectar com API externa
- Receber dados da API externa e mostrar na página de 'Trade'.
- Receber mudanças nos inputs de quantidade de pokémons.
- Armazenar pokémons e suas quantidades.
- Validar a quantidade de pokémons.
- Setar objeto contendo todos os pokémons e quantidades escolhidos pelos players no localstorage.
- Pegar as informações do localstorage
- Calcular o resultado da troca

Componentes:

- Player:
    - Component utilizado como modelo para os dois players

- History
    - Componente que mostra a última troca e o seu resultado.

LocalStorage:

- Trade: objeto contendo pokémons e seus atributos e quantidade.
- Experiência total do player 1: Experiência total do player 1 calculada na página de trade
- Experiência total do player 2: Experiência total do player 2 calculada na página de trade
- Valid trade: boolean que seta se a troca é justa ou não.

Obs: De acordo com o tipo de API, o localstorage é um local de fácil acesso, dessa forma, facilita a entrada e retirada de valores que não possuem fator de segurança elevado, além de fazer o papel de armazenar os valores que podem ser acessados por componentes durante todo decorrer de uso, dessa forma, foi escolhido para deixar a API mais fácil de ser compreendida.

Design:

Para ajudar na escrita do código e na criação da interface, através do figma, todas as páginas devem seguir um padrão de cor, como ideia, utilizar o charmander e suas evoluções para mostrar progressão entre as páginas, desde as regras até o resultado.

Assim, é utilizado quatro imagens durante o codigo sendo elas:

- charmander.png
- charmaleon.png
- charizard.png
- history.png
- trade.png

Sendo que, trade.png e history.png vão ser ícones com funcionalidades, history mostra o resultado da última troca e trade é o botão que os players usam para realizar uma troca.

Todo o design das páginas vai ser desenvolvido para teste em aparelhos de computador, porém a responsividade deve fazer parte mesmo que reduzindo a ipads ou tablets.

Validação da Trade:

Para fazer a validação da trade foi escolhido o valor de 75 para a proximidade entre a soma das bases de experiência, devido a pokémons terem diferentes valores, sendo eles altos ou baixos, dependendo da quantidade de pokémons escolhidos pelo usuário o valor total pode ficar bem alto.

Conclusão:

Como prioridade deve-se ser levado em conta o máximo de clean code e mostrar como funciona a lógica por trás de todo o projeto e utilizar da melhor forma o framework..