**Projeto POKEDEX**

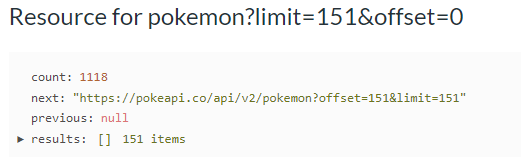
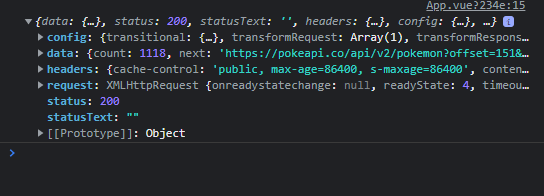
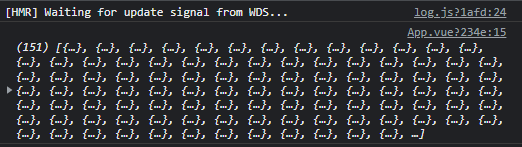
**Iniciando o projeto**

1. Neste projeto nós vamos recriar a pokedex do Pokemon no VueJs, o pokedex é um catálogo de todos os pokemons existentes na região.
2. Existe uma API na internet chamada pokeapi.co, que é gratuita e é ótima para testar aplicações FrontEnd.
3. Vamos começar criando o novo projeto VueJs
4. Na pasta onde o projeto será criado digite: vue create pokedexx
5. Após concluído a instalação vamos na pasta do projeto e vamos dar uma limpada no código.
6. O arquivo App.vue deve ficar da seguinte forma.
7. <template>
8. <div id="app">
10. </div>
11. </template>
12. <script>
13. export default {
14. name: 'App',
16. }
17. </script>
18. <style>
19. </style>
20. Durante a limpeza apagamos também o componente HelloWord

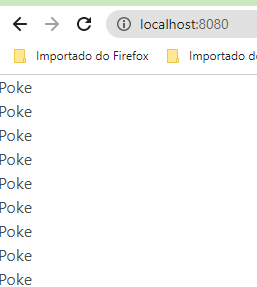
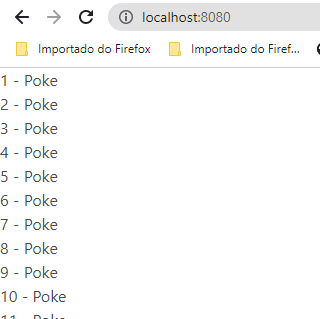
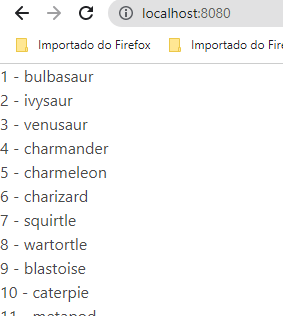
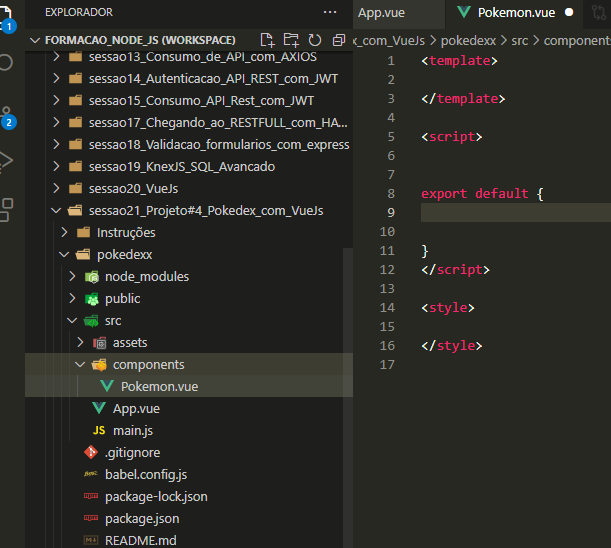
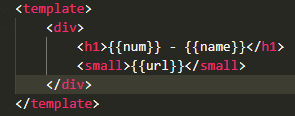
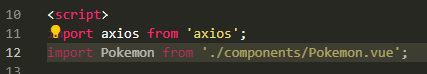
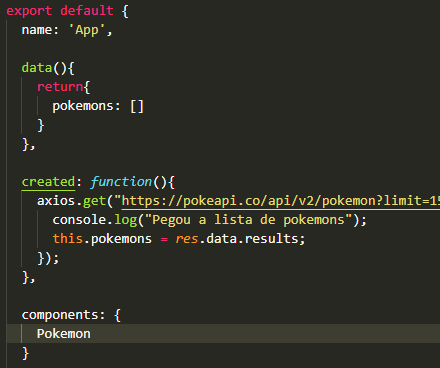
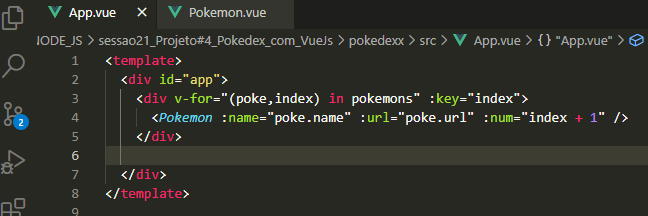
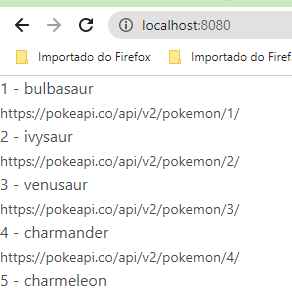
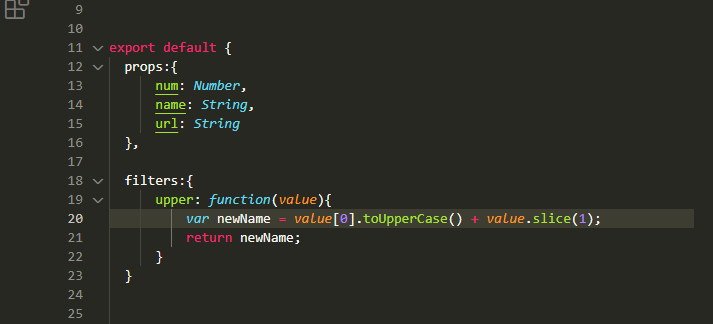
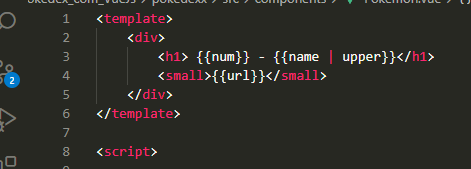
**Instalando as Bibliotecas**

1. **Biblioteca Bulma**
2. para o css do projeto
3. Parar o servidor dando cntr-C no terminal do vs-code, se estiver rodando, e na pasta do projeto digite: npm install bulma
4. **Biblioteca axios**
5. para permitir que sejam feitas requisições http ou seja requisições para outros servidores web. Adicionando o axios nós conseguimos fazer que a nossa aplicação converse com outras aplicações na web.
6. Na pasta do projeto digite: npm install axios .
7. Após a instalação das bibliotecas, rode novamente o servidor, na pasta do projeto digite:
8. Npm run serve
9. **Importando as bibliotecas**
10. No arquivo main.js importe a biblioteca bulma.
11. import "../node\_modules/bulma/css/bulma.css"
12. No arquivo App.vue importe a biblioteca axios, lembrando que esta biblioteca deve ser importância em todos os componentes que utilizarão este recurso.
13. import axios from 'axios';
14. nesse momento deve aparecer um erro no projeto, pois a variável criado axios foi importada porém ainda não está sendo utilizada, para resolver este problema momentaneamente basta comentar a linha //import axios from ‘axios’;

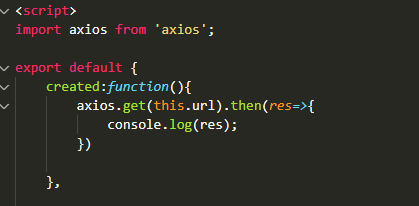
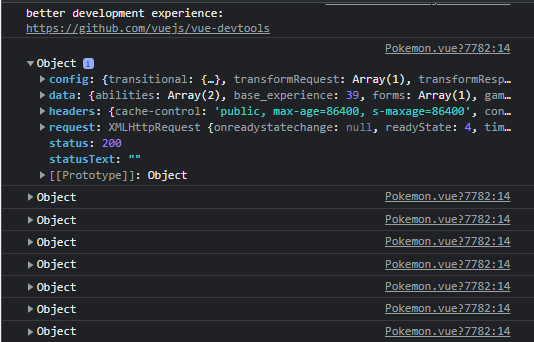
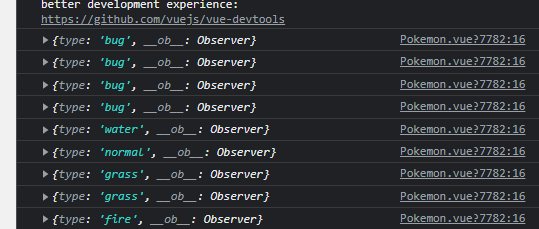
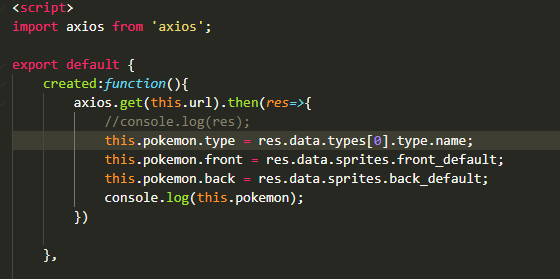
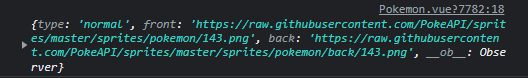
**Pegando os dados dos Pokemons da API pokeapi**

1. Para darmos continuidade ao projeto devemos pegar os dados dos pokemons, para isso temos que ir até a página da pokeApi para configurarmos a quantidade pokemons que iremos pegar por vez. Então na pagina <https://pokeapi.co/>
2. Digite no campo: pokemon?limit=151&offset=0 e clicke em submit.
3. Esse comando irá fazer uma requisição para a rota pokemon passando os atributos limit e offseet permitindo os pokemons até o número 151 começando do zero.
4. 
5. Essa consulta retornará
6. 
7. Vamos pegar essa lista de 151 pokemons e vamos enviar para nossa aplicação utilizando o axios para fazer uma requisição http para a url utilizada no item 3.
8. No componente App vamos descomentar a linha que importa o axios.
9. Para fazer uma requisição http nós devemos chamar esta requisição dentro de um método.
10. Como as informações dos pokemons são vitais para a aplicação, estas informações devem estar disponíveis na inicialização da aplicação, ou seja toda vez que a página for carregada com o componente, para isso vamos utilizar o método chamado create que irá fazer uma requisição com o axios.
11. Digite o código como abaixo
12. export default {
13. name: 'App',
14. created: *function*(){
16. }
18. }
19. Dentro do método create vamos utilizar o axios e os verbos http para fazermos a requisição dos pokemons. Neste caso vamos utilizar o verbo get pois queremos somente pegar os pokemons.
20. created: *function*(){
21. axios.get()
22. }
23. Dentro da chaves do get vamos inserir a url que será utilizada para a requisição, retornando um promisse com a resposta.
24. created: *function*(){
25. axios.get("https://pokeapi.co/api/v2/pokemon?limit=151&offset=0").then(*res* *=>* {
26. console.log(*res*);
27. });
28. }
29. Abrindo o console na página do projeto é possível ver todos os dados que são carregados com a resposta.
30. 
31. Porém vamos selecionar somente os pokemons melhorando a chamada da resposta no console.log
32. created: *function*(){
33. axios.get("https://pokeapi.co/api/v2/pokemon?limit=151&offset=0").then(*res* *=>* {
34. console.log(*res*.data.results);
35. });
36. }
37. 
38. Dentro do escopo de exports defaults{ }, vamos pegar todos os itens da resposta criando uma variável dentro da função dados, lembrando que esta função sempre retorna um objeto.
39. Esta variável será pokemons e irá receber um array.
40. data(){
41. return{
42. pokemons: []
43. }
44. }
45. No método create vamos passar o array que foi capturado pela requisição http para a variável pokemons que nós criamos.
46. created: *function*(){
47. axios.get("https://pokeapi.co/api/v2/pokemon?limit=151&offset=0").then(*res* *=>* {
48. console.log(*res*.data.results);
49. this.pokemons = *res*.data.results;
50. });
51. }
52. Como sabemos que a requisição está sendo feita corretamente, vamos alterar a mensagem do console.log para “Pegou a lista de pokemons”
53. created: *function*(){
54. axios.get("https://pokeapi.co/api/v2/pokemon?limit=151&offset=0").then(*res* *=>* {
55. console.log("Pegou a lista de pokemons");
56. this.pokemons = *res*.data.results;
57. });
58. }

**Criando o componente Pokemon**

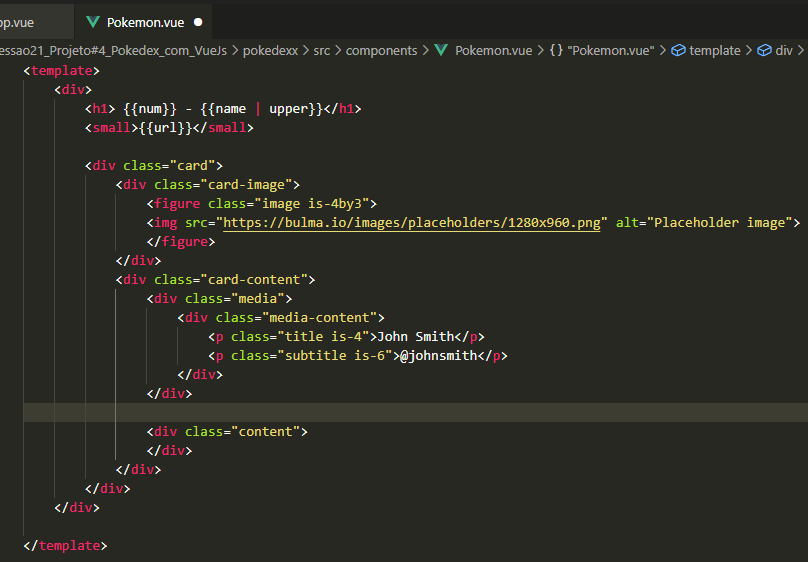
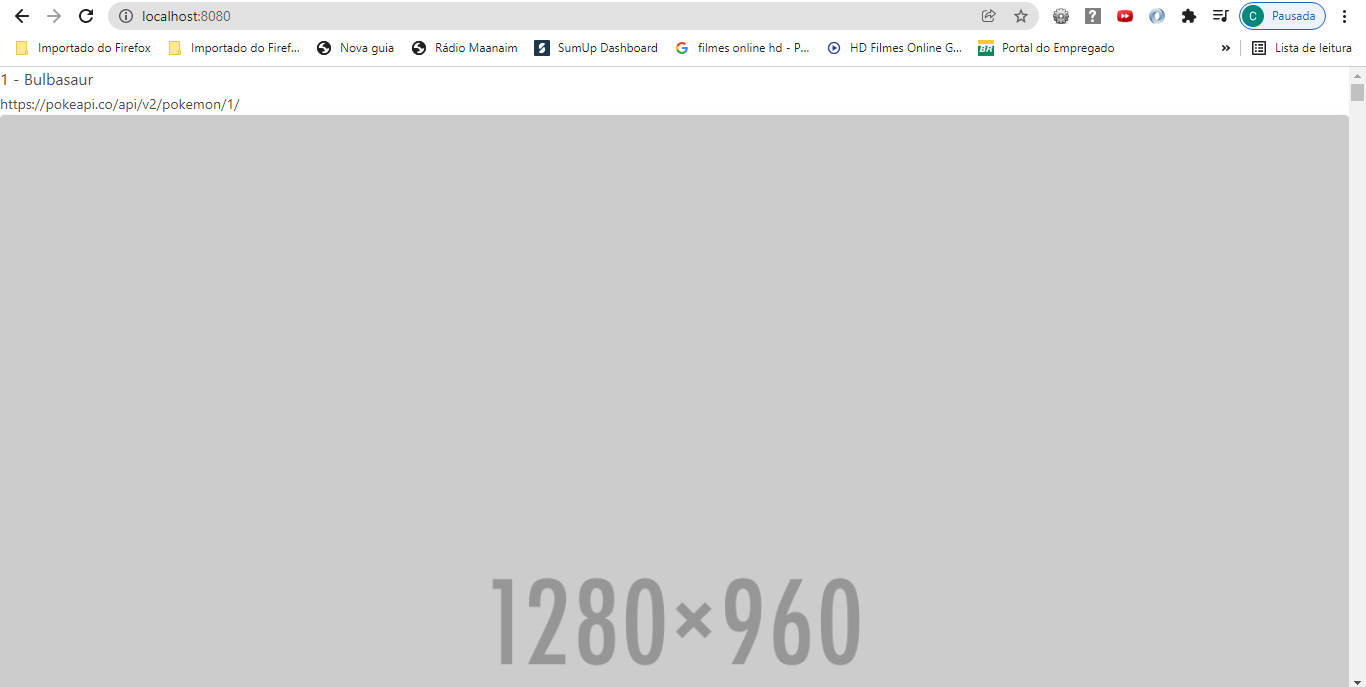
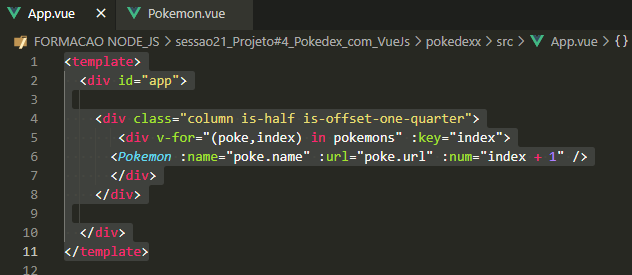
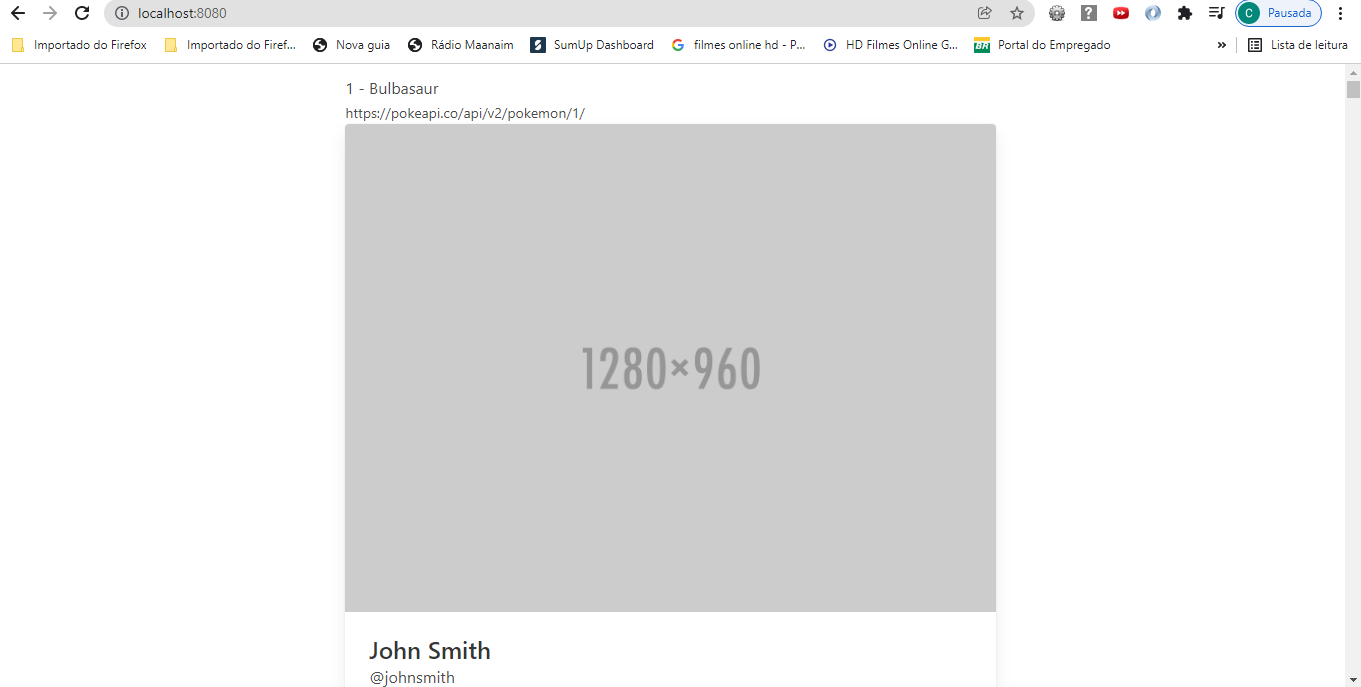
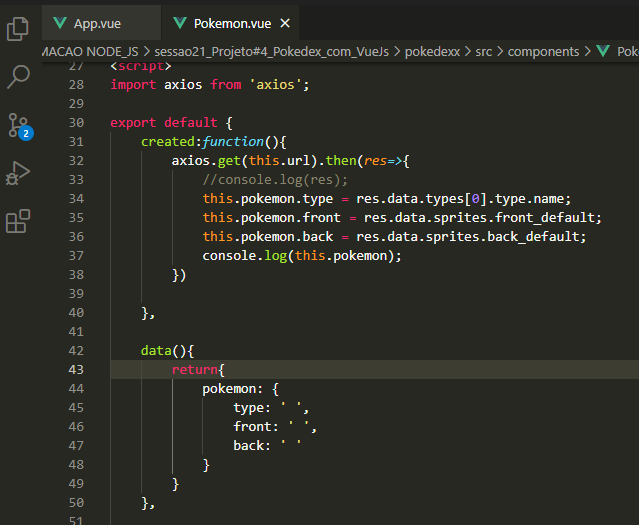
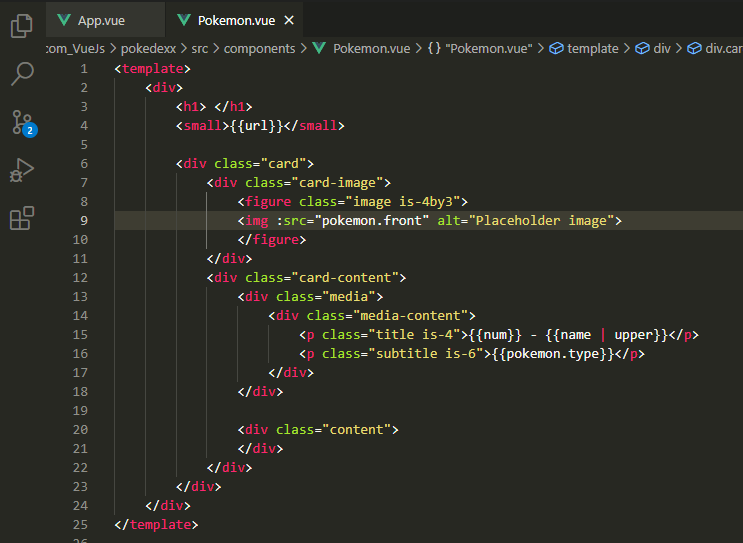
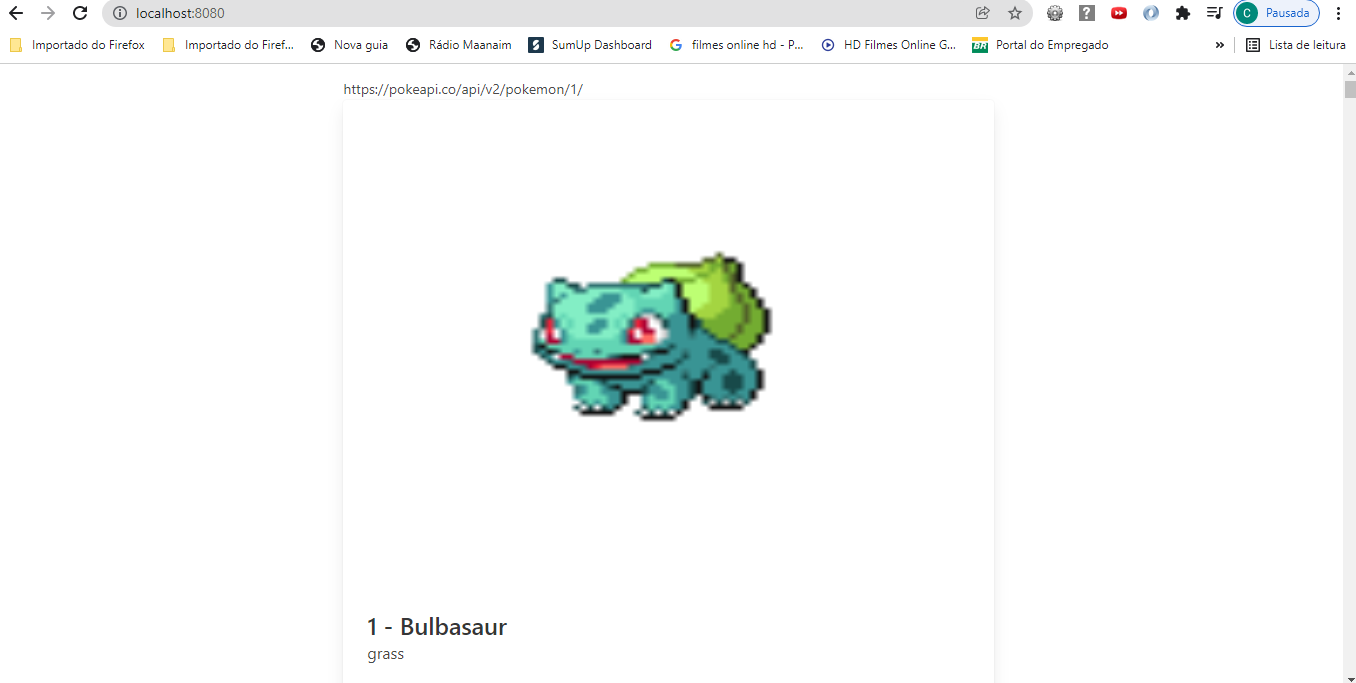
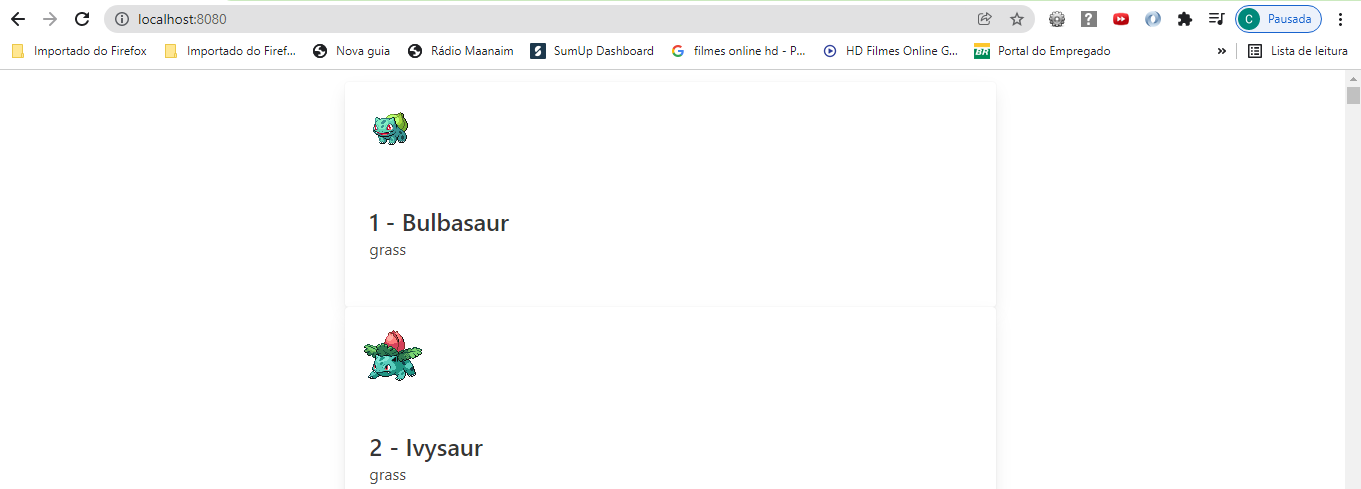
1. Como podemos perceber na api, o pokemon possui somente dois campos que são o nome e a url dele. Se fizermos uma requisição para a url do pokemon teremos as informações dele. 
2. No componente principal App, vamos fazer um v-for no array de componentes pokemon.
3. Para isso vamos criar uma div com v-for que irá receber um poke e um index no array de pokemons, utilizando o index para diferenciar um do outro. Para testar vamos criar um h1 – Poke.
4. <template>
5. <div id="app">
6. <div v-for="(poke,index) in pokemons" :key="index">
7. <h1>Poke</h1>
8. </div>
10. </div>
11. </template>
12. Na pagina do projeto o resultado será uma lista com um poke para cada pokemon
13. 
14. Para melhorar o teste vamos colocar no h1 uma interpolação mostrando o index + 1.
15. <div v-for="(poke,index) in pokemons" :key="index">
16. <h1>{{index + 1}} - Poke</h1>
17. </div>
18. O resultado na página
19. 
20. Para melhorar mais ainda, vamos incluir o nome do poke para exibir na página, fazendo uma interpolação utilizando name.
21. <div v-for="(poke,index) in pokemons" :key="index">
22. <h1>{{index + 1}} - {{poke.name}}</h1>
23. </div>
24. O resultado será:
25. 
26. Agora que testamos o carregamento da lista de pokemons, vamos criar um componente para o pokemon.
27. Dentro da pasta componentes do projeto vamos criar um arquivo chamado Pokemon.vue, setando a estrutura básica de um componente.
28. 
29. Esse componente por padrão irá receber algumas props que será o nome do componente, a url que será utilizada para fazer outras requisições e num que receberá o número do pokemon na pokeapi.
30. 
31. Dentro do template vamos inserir uma div com as informações que serão carregadas com o componente
32. 
33. Após isso vamos no app principal App.vue
34. E vamos importar o componente Pokemon.vue
35. 
36. Vamos registrar o componente Pokemon dentro do componente principal App.vue, para isso temos que adicionar o objeto components dentro da clausula export default.
37. 
38. Agora podemos utilizar o componente Pokemon dentro do v-for, passando as props através de um databynd.
39. Lembrando que o databynd nada mais é que as variáveis passadas como props pelo componente com dois pontos no inicio, que irá exibir o conteúdo passado pelo componente Pokemon
40. 
41. A pagina do projeto ficará desta forma
42. 
43. Vamos melhorar a exibição do nome, alterando a primeira letra para maiúscula. Faremos isso através de um filter.
44. Então dentro do componente Pokemon, vamos criar um objeto filter onde será criado os filtros. Dentro de filter vamos criar o filtro chamado upper.
45. Um filtro deve receber um valor e retornar outro.
46. Vamos criar uma variável que irá pegar a primeira letra do valor passado e transformar para maiúsculo utilizando a função Javascript toUpperCase(), retornando a primeira letra do nome em maiúscula. Porém para retirarmos a letra minúscula e adicionar a maiúscula, devemos utilizar a função slice() que irá excluir a letra com o índice fornecido.
47. 
48. Para isso funcionar temos que utilizar o filtro em frente a props que está sendo passada pelo componente Pokemon.
49. Então no h1 colocamos o filtro em frente a variável name, da seguinte forma.
50. 

**Buscado informações do Pokemon**

1. Para buscar mais informações no componente Pokemon, vamos importar a biblioteca axios dentro do componente e criar o método created para realizar requisições com o axios. Vamos
2. Fazer a requisição utilizando a url do Pokemon.
3. 
4. Se abrirmos o console iremos ver os objetos que carregam as informações de cada url
5. 
6. Vamos criar o método data() com uma variável chamada pokemon que irá receber um objeto.
7. A variável pokemon ira utilizar o tipo type e receberá o tipo do pokemon.
8. 
9. O resultado no console é
10. 
11. Além do primeiro tipo do pokemon, vamos carregar também dois sprites o da frente e o das costas.
12. 
13. No console podemos ver o retorno das informações
14. 

**Melhorando o Componente**

**Copiando Css do card do Bulma**

1. Vamos melhorar a aparência do componente utilizando os recursos do Bulma, vamos utilizar o card. Na página do Bulma basta procurar sobre o elemento card e copiar o código do card escolhido. No componente vamos colar o código css, apagamos o conteúdo da class contente, apagamos também div da imagem pequena ficando somente com a imagem grande.
2. O código ficará assim
3. 
4. A página ficará da seguinte forma:
5. 
6. Como resultado teremos uma imagem muito grande tomando toda a tela, para resolver isso vamos utilizar o recurso de colunas do bulma. Vamos reduzir o tamanho da coluna central para a metade e vamos centraliza-la .
7. Faremos isso no aplicativo principal App.vue. No template vamos criar uma div que irá englobar o v-for e terá uma classe column is-half isso fará com que o componente preencha somente a metade da página.
8. Para centralizar a coluna na página utilizamos ainda dentro da classe a expressão is-offset-one-quarter que irá adicionar um offset no inicio da página, de um quarto da largura da página.
9. O código no template do App.vue ficará assim
10. 
11. A página ficará assim:
12. 
13. Agora vamos preenche o conteúdo dos cards, mas antes temos que criar as propriedades type, front e back no método data() no componente Pokemon, pois essas variáveis estavam sendo criadas dinamicamente dentro da requisição do axios.
14. 
15. Após feito isso, vamos alterar o código do template no componente Pokemon. Vamos retirar o código que estava dentro do h1 e colocar dentro do parágrafo da classe “title is-4”. Na classe “subtitle is-6” vamos incluir o tipo do pokemon. Vamos ainda inserir a imagem do pokemon
16. Para isso vamos utilizar um dataBind no atributo src da tag img, alterando a url da imagem passando a variável pokemon.front que é onde está armazendo a imagem do pokemon, na pokeApi. Código ficará da seguinte forma:
17. 
18. A pagina ficara assim:
19. 
20. A imagem ficou desproporcional na tag figure no template do componente Pokemon, está sendo utilizado a classe ‘image is-4by3’ que redimensiona a figura para o tamanho da tela. Vamos retirar essa classe e apagar também a tag small onde está a url do pokemon.
21. O código ficará da seguinte forma:
22. 
23. A página ficará assim
24. 
25. Vamos melhorar o componente aplicado uma margem ao componente, para isso vamos criar um id na div principal do componente chamado pokemon.e aplicar uma margin-top nesta id, destro da área style do componente.
26. O código ficará assim:
27. 
28. 
29. A pagina ficara :
30. 