

Pokemon-API Prototype

Juan David Cotacio Sánchez, Valery Dayana Triana Garcia

Resumen—Your abstract.

I. INTRODUCCIÓN

En un mundo donde la nostalgia se fusiona con la innovación, nuestro proyecto Pokémon-API emerge como un homenaje moderno a una de las franquicias más queridas en la historia de los videojuegos: Pokémon. Utilizando tecnologías como CSS, HTML, JavaScript, jQuery ,Artyom y Boostrap hemos creado una experiencia interactiva que transporta a los usuarios a través del universo Pokémon de una manera única y emocionante.

I-A. Como empezar el prototipo

Para la creación de este prototipo empece en como podia implementar estas tecnologías, para la inclusión de esta herramientas empece por crear un HTML y CSS a la vez además de importar las librerias que vamos a usar.



```
Project File Tree Reveal ▾
  ▾ API
    ▾ css
      main.css
    ▾ js
      artyom.js
      artyom.window.js
      artyom.window.min.js
      main.js
      test.json
      index.html
      Guitarrero
      App.js
```

```
<!DOCTYPE html>
1 <html lang="en">
2   <head>
3     <meta charset="UTF-8">
4       <meta name="viewport" content="width=device-width, initial-scale=1.0">
5       <title>Pokedex</title>
6     <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
7       integrity="sha384-Olq5wL5Iv6QDfNGzIuOyQ6E7hWt1I7qy7sC8xZn0GZPh6qYpGfnKT" crossorigin="anonymous">
8     <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js"
9       integrity="sha384-JZR6Spejh4U024JrcRvHlEJErKjs7ACoUHkfrhvsgFO3JwqXaUgA=" crossorigin="anonymous">
10    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6a3Gqgvzv3j吏" crossorigin="anonymous">
11    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScFdeoEjEqKJqsU5MfKt07hJfKm0=" crossorigin="anonymous">
12    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
```

Figura 1. Importación de librerias

I-B. Como se incluyo el CSS, HTML y Bootstrap

Primero empece a crear una estructura en mi HTML para darle el espacio, cree un contenedor general que iba almacenar mis botones y imagenes cabe recalcar que estos botones y barra busqueda son traidos de Boostrap, entonces ya traen un diseño por defecto pero aun así se necesita CSS con su respectiva class y id, para poder modificar su posición, tamaño o color.

En mi css quise añadir las respectivas imagenes de cada una de las cosas que estaba añadiendo en mi HTML, para darle posición y tamaños, en nuestra hoja de estilos lo mas destacable es el manejo de error 404, unos de los problemas que se me presento el de que el error no apareciera apenas se cargara la pagina entonces use la función "display: none;" el cual va ocultar en la pantalla mientras carga la pagina pero aun así se necesitaba manejar logica en JavaScript para poder manejar el error.

I-C. Acercamiento con la API

Para usar los conceptos de API, es necesario realizar con algo ya creado e implementarlo en nuestro proyecto, así fue como se llego Poke-API, esta contiene datos destacables de la famosa seria Pokémon, algunas de estos datos son: nombre, habilidades, imágenes o el tipo de este mismo.

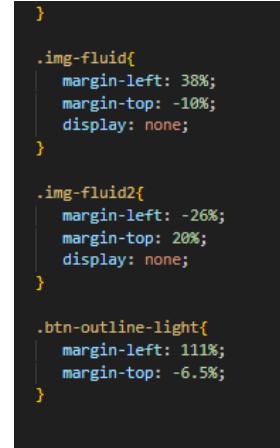


Figura 2. Uso del "display: none;"

Asi nuestra Pokedex se iba generar de forma dinamica y sencilla pues no se necesitara añadirlo de forma rustica, ademas de nuevos conceptos como son los Archivos JSON.

Resource for ditto



```
abilities: []
  • @: [] 2 items
    • ability: [] 3 keys
      • name: "leech-life"
        • url: "https://pokeapi.co/api/v2/ability/7/"
        • is_hidden: false
      slot: 1
    • !: [] 0 keys
    • ability: [] 3 keys
      • name: "reposter"
        • url: "https://pokeapi.co/api/v2/ability/150/"
        • is_hidden: true
      slot: 3
    source: 101
  • cries: [] 2 items
    latest: "https://www.githubusercontent.com/PokeAPI/cries/main/cries/pokemon/latest/132.ogg"
    legacy: "https://www.githubusercontent.com/PokeAPI/cries/main/cries/pokemon/legacy/132.ogg"
  • forms: [] 1 item
    • @: [] 2 keys
      • name: "normal"
        • url: "https://pokeapi.co/api/v2/pokemon-form/132/"
  • game_indices: [] 20 items
    height: 3
  • held_items: [] 2 items
View raw JSON (34.006 kB, 974 lines)
```

Figura 3. Collider en JavaScript

I-D. Implementación Java Script

Para la implementación del JS en este prototipo fue principalmente el uso de JQuery, Artyom y Ajax, para iniciar con nuestra parte logica fue importante empezar con JQuery para poder añadir eventos a nuestro boton con el click pues esta libreria nos iba sintetizar codigo y ademas al momento de combinarla con AJAX nos facilita la comunicación con nuestra API para asi poder realizar la búsqueda.

Una de las complejidades que usamos en nuestra parte logica fue traer los datos que se encontraban en diccionario, pero despues de investigación se puede dar cuenta que se puede traer `pokemon` pues es una de las funcionalidades con HTML.

Para solucionar el problema con los diccionarios usamos esa misma función del HTMl solo que ahora añadimos JSON.Stringify para poder traer del API una ubicación exacta de nuestro diccionario que contenía un Array ademas de añadir condicionales para la mejor gestión de información.

Para implementar la búsqueda de voz con la librería Artyom fue una de las cosas que pense que eran complejas pero no lo eran, para la implementación de esta misma se intalo la libreria se inicializo y se empezo con la parte logica, buscamos que las palabras que se detectaban en consola fueran remplazada en mi barra de búsqueda, para esto creamos una función combinado con un condicional, donde le añadimos "trim" "toLowerCase", son herramientas que nos ayudan a ignorar la mayuscula y espacios, pues sin esto el manejo del error 404 siempre iba a aparecer.

```
$document.ready(function () {
  $("#my-button").on("click", function () {
    $.ajax({
      url: "https://pokeapi.co/api/v2/pokemon/" + $("#txt-buscar").val(),
      type: "GET",
      contentType: "application/json",
      success: function (data) {
        $("#img_error404").hide();
        $("#imgtxt_error404").hide();
        console.log(data.sprites.other.home.front_default)
        $("#Imagen_pokemon").html("<img src=$data.sprites.other.home.front_default>");

        console.log(data.sprites.other.home.shiny)
        $("#Imagen_pokemon_shiny").html("<img src=$data.sprites.other.home.shiny>");

        console.log(data.base_experience)
        $("#type_pokemon").html('<p>Base Experience:$data.base_experience</p>');

        console.log(data.weight)
        $("#weight_pokemon").html('<p>Pokemon weight:$data.weightkg </p>');

        console.log(data.id)
        $("#id_pokemon").html('<p>ID:$data.id</p>');

        console.log(data.height)
        $("#height_pokemon").html('<p>Pokemon height:$data.heightcm</p>');

        console.log(data.name)
        $("#name_pokemon").html('<p>Name:$data.name</p>');

        console.log(data.types[0].type.name)
        $("#types_pokemon").html('<p>Types:$JSON.stringify(data.types[0].type.name)</p>');

        if (data.types.length > 1) {
          console.log(data.types[1].type.name)
          $("#types2_pokemon").html('<p>$JSON.stringify(data.types[1].type.name)</p>');
        }
      }
    });
  });
});
```

Figura 4. Implementación JQuery y AJAX

```
artyom.redirectRecognizedTextOutput(function(recognized, isFinal) {
  if(isFinal) {
    console.log("Texto final reconocido: " + recognized);
    // Actualiza el valor del campo de búsqueda
    $("#txt-buscar").val(recognized.trim().toLowerCase());
  } else {
    console.log(recognized);
  }
});
```

Figura 5. Implementación Artyom.js

I-E. Manejo del error 404

Para el manejo del error fue unas de las cosas que tuvieron un grado de dificultad cuando se realizaban búsquedas después de aparecer el error, para poder solucionar este problema se uso JavaScript para que se esconda el error siempre que se encontrara y después crear una función que haga que este error aparezca y ocultara los datos del Pokemon en cuestión.

I-F. Vista General del Prototipo

En la vista general de mi prototipo se logró realizar un pokewiki el cual sirve de consulta de los pokemones añadiéndole búsquedas de voz para dar más opciones a este prototipo ademas de los manejos de errores.

```
error: function (xhr, status, error, TypeError) {
  $("#img_error404").show();
  $("#imgtxt_error404").show();

  $("#Imagen_pokemon").html('');
  $("#Imagen_pokemon_shiny").html('');
  $("#type_pokemon").html('');
  $("#weight_pokemon").html('');
  $("#id_pokemon").html('');
  $("#height_pokemon").html('');
  $("#name_pokemon").html('');
  $("#types_pokemon").html('');
  $("#types2_pokemon").html('');
  $("#name_pokemon").html('');
  $("#pokemon_abilities").html('');
  $("#pokemon_abilities2").html('');
  $("#pokemon_abilities3").html('');
  $("#statsHP_pokemon").html('');
  $("#statsAttack_pokemon").html('');
  $("#statsDefense_pokemon").html('');
  $("#statsSpecial-Attack_pokemon").html('');
  $("#statsSpecial-defense_pokemon").html('');
  $("#statsSpeed_pokemon").html('');
```

Figura 6. Esconder Error 404

```
success: function (data) {
  $("#img_error404").hide();
  $("#imgtxt_error404").hide();
```

Figura 7. Mostrar Error 404

I-G. Instructivo

Para que el juego se pueda disfrutar en términos general se debe ajustar siempre valores, para que el contenido se adapte a la pantalla y no se vea desproporcionado se puede utilizar la combinación de teclas `ctrl + .` en su defecto `ctrl + -`, estos valores son adaptables al gusto del usuario pero para resoluciones 1920 x 1080, el 67 porciento queda perfectamente anclado.

Para la búsqueda de voz es necesario tener un micrófono funcional ademas de otorgar el permiso para que este pueda usar Artyom y por último usar navegadores compatibles con Artyom como lo son Chrome o Edge.

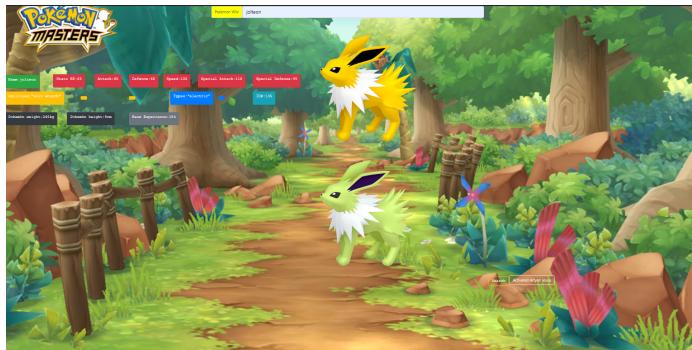


Figura 8. Vista Final Del Prototipo

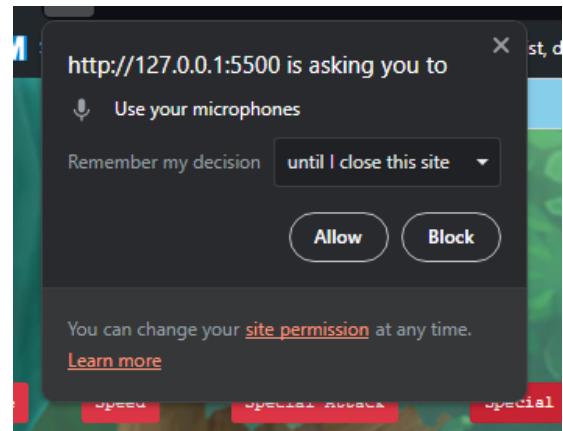


Figura 11. Volumen recomendado

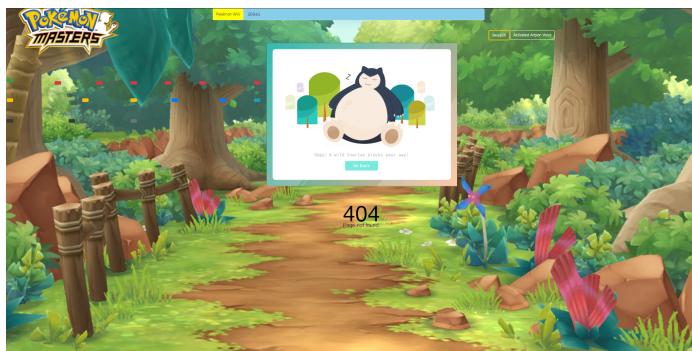


Figura 9. Vista Final Del Prototipo

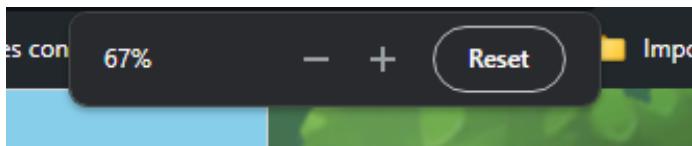


Figura 10. Zoom para resoluciones 1920 x 1080