



**IES AUGUSTO GONZALEZ DE
LINARES
DEPARTAMENTO DE INFORMATICA**

CLIENTE API


Pokedex

**PROGRAMACIÓN DE SERVICIOS Y
PROCESOS**

**GRADO SUPERIOR DE DESARROLLO DE
APLICACIONES MULTIPLATAFORMA**

2023/2024

Díez de Paulino, Albano

	Ciclo: Desarrollo de Aplicaciones Multiplataforma Modulo: Programación de Servicios y Procesos Título: Pokedex Tarea 2 RA4
--	--

Índice

1.	Elección de API.	2
1.1.	PokeAPI	2
1.2.	VoiceRSS	3
2.	Dependencias Cliente API (org.json).....	4
3.	Métodos Relevantes en la Aplicación.	5
4.	Demostración de funcionamiento.	7
5.	Investigación sobre la creación de servidores API REST	8
5.1.	¿Qué lenguaje, framework o librería se pueden usar?.....	8
5.2.	¿Qué elementos hardware y software necesitarías para desplegar la API REST?.....	9
6.	Bibliografía	10
7.	Tabla de Ilustraciones	11

1. Elección de API.

Partiendo de las premisas impuestas por el profesor Joaquín Franco, se me ocurrió la idea de realizar una copia de la Pokedex de la serie y juegos de Pokémon (Dispositivo que muestra la información de las criaturas de la serie de forma visual y auditiva).

El diseño que se quiere seguir en la aplicación de java es el de replicar el diseño de la Pokedex de la primera serie.

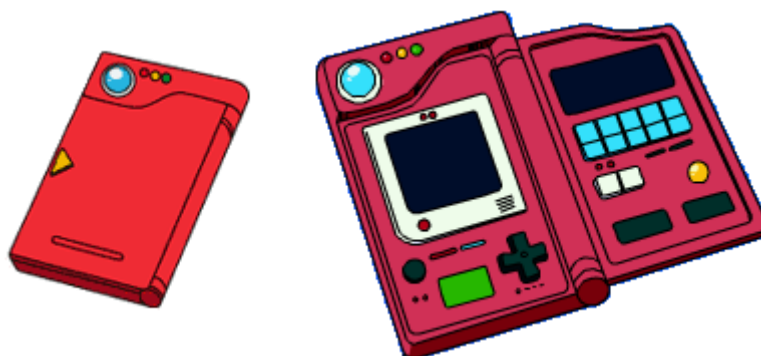


Ilustración 1 -Diseño Pokedex 1ª Serie

Para rescatar la información se usará una API pública disponible en internet y así cumplir el único requisito.

1.1. PokeAPI

La PokeAPI es una API RESTful de código abierto sin APIKEY que tiene múltiples endpoints, por ejemplo, se pueden rescatar todas las Berries (Bayas) a partir de este endpoint de tipo GET <https://pokeapi.co/api/v2/berry/{id or name}>

Pero para esta práctica solo uso el endpoint de tipo GET <https://pokeapi.co/api/v2/pokemon/{id}> para obtener toda la información de un Pokémon en concreto.

A continuación, se muestra una vista de tipo Árbol de una parte del JSON que devuelve el endpoint anterior.

```
▼ moves: [] 1 item
  ▼ 0: {} 2 keys
    ▼ move: {} 2 keys
      name: "transform"
      url: "https://pokeapi.co/api/v2/move/144/"
      ▶ version_group_details: [] 22 items
      name: "ditto"
      order: 214
      past_abilities: [] 0 items
      past_types: [] 0 items
    ▼ species: {} 2 keys
      name: "ditto"
      url: "https://pokeapi.co/api/v2/pokemon-species/132/"
    ▶ sprites: {} 10 keys
    ▶ stats: [] 6 items
  ▼ types: [] 1 item
    ▼ 0: {} 2 keys
      slot: 1
      ▼ type: {} 2 keys
        name: "normal"
        url: "https://pokeapi.co/api/v2/type/1/"
      weight: 40
```

Diagrama 1 – JSON Ejemplo

Si se desea obtener mas información de la API, los creadores dejan una documentación publica que dejo en la bibliografía de este documento.

1.2. VoiceRSS

Al inicio de la practica se tenia pensado usar una API de tipo TTS(Text to Speech) para mostrar la información de forma sonora, para ello se buscaron múltiples APIs pero la mas sencilla de usar y con capa gratuita es VoiceRSS.

Para hacer uso de ella es necesario registrarse en el sitio web y que te proporcionen una APIKEY, solo hay un endpoint de tipo GET disponible, pero que permite múltiples parámetros y que devuelve un archivo con extensión wav.

URL del Endpoint: [http://api.voicerss.org/?key=\[apikey\]&hl=en-us&src=Hello,world!](http://api.voicerss.org/?key=[apikey]&hl=en-us&src=Hello,world!)

Pero a mitad de desarrollo se decidió no usar esta API y no implementar esta funcionalidad porque la calidad del audio es muy pobre.

Si se desea obtener más información de la API, los creadores dejan una documentación publica que dejo en la bibliografía de este documento.

2. Dependencias Cliente API (org.json)

Para poder tratar los JSON de las respuestas de los endpoint en Java hay muchas dependencias, pero la que decidí usar es org.json

Es muy sencilla de usar ya que solo con dos clases se pueden traducir los JSON.

La primera clase es JSONObject, que transforma el JSON en un mapa (Diccionario) y la segunda clase JSONArray que transforma en un array de objetos.

A continuación, se muestra uno ejemplo de uso de las clases anteriores.

```
1 public static String[] getTypes() {  
2     String types[] = new String[json.getJSONArray("types").length()];  
3     for (int i = 0; i < json.getJSONArray("types").length(); i++) {  
4         types[i] = json.getJSONArray("types").getJSONObject(i).getString("name");  
5     }  
6     return types;  
7 }
```

Código 1 – Ejemplo de JSONObject y JSONArray

Código 2 – Ejemplo de JSONObject y JSONArray

3. Métodos Relevantes en la Aplicación.

La estructura de la aplicación sigue el modelo vista controlador por lo que lo más relevante es la clase que realiza las diferentes consultas a la PokeAPI.

Esta clase tiene un variable de tipo String que almacena el JSON y que es actualizado cada vez que el usuario avanza o retrocede en la UI. Por lo tanto, tiene un método que en mi realiza la consulta.


A partir de realizar la consulta la clase dispone de varios métodos para rescatar la información necesaria para la UI.

```
1  /**
2   * Metodo para hacer la consulta a la API de Pokemon
3   * @param id Id del pokemon a buscar
4   */
5  public static boolean consulta(int id) {
6      try {
7          HttpClient client = HttpClient.newHttpClient();
8          HttpRequest request = HttpRequest.newBuilder()
9              .uri(new URI("https://pokeapi.co/api/v2/pokemon/" + id))
10             .timeout(Duration.ofSeconds(10))
11             .GET()
12             .build();
13
14             HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());
15
16             json = new JSONObject(response.body());
17             return true;
18         } catch (URISyntaxException | IOException | InterruptedException ex) {
19             return false;
20         }
21     }
22 }
```

Código 3 – Método para realizar la consulta

```
1  /**
2   * Metodo para Obtener la Imagen del pokemon seleccionado
3   * @return Image Sprite del Pokemon
4   */
5  public static Image getSprite() {
6      return new Image(json.getJSONObject("sprites").getJSONObject("other").getJSONObject("home").getString("front_default"));
7  }
8
9  /**
10   * Metodo para Obtener el nombre del pokemon seleccionado
11   * @return String Nombre del Pokemon
12   */
13  public static String getName() {
14      return json.getString("name");
15  }
16
17  /**
18   * Metodo para Obtener las estadísticas del pokemon seleccionado
19   * @return Integer[] Array de 6 para las estadísticas
20   */
21  public static Integer[] getStats() {
22      Integer stats[] = new Integer[6];
23      for (int i = 0; i < 6; i++) {
24          stats[i] = json.getJSONArray("stats").getJSONObject(i).getInt("base_stat");
25      }
26      return stats;
27  }
28  }
29
30  /**
31   * Metodo para Obtener los tipos del pokemon seleccionado
32   * @return String[] Array de 2 para los tipos
33   */
34  public static String[] getTypes() {
35      String types[] = new String[json.getJSONArray("types").length()];
36      for (int i = 0; i < json.getJSONArray("types").length(); i++) {
37          types[i] = json.getJSONArray("types").getJSONObject(i).getJSONObject("type").getString("name");
38      }
39      return types;
40  }
```

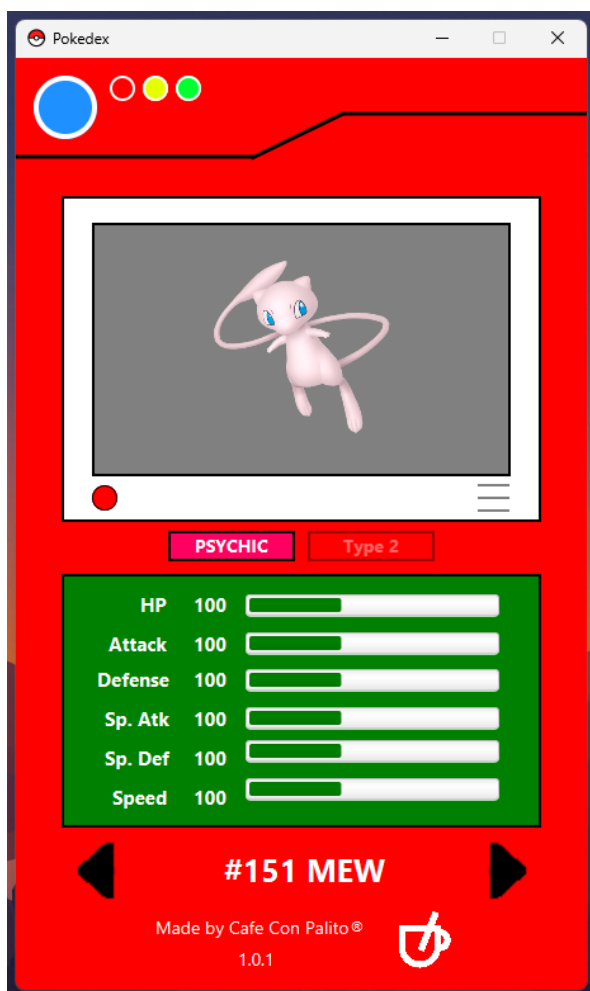
Código 4 – Métodos para Obtener información a partir del JSON

	Ciclo: Desarrollo de Aplicaciones Multiplataforma Modulo: Programación de Servicios y Procesos Título: Pokedex Tarea 2 RA4
--	--

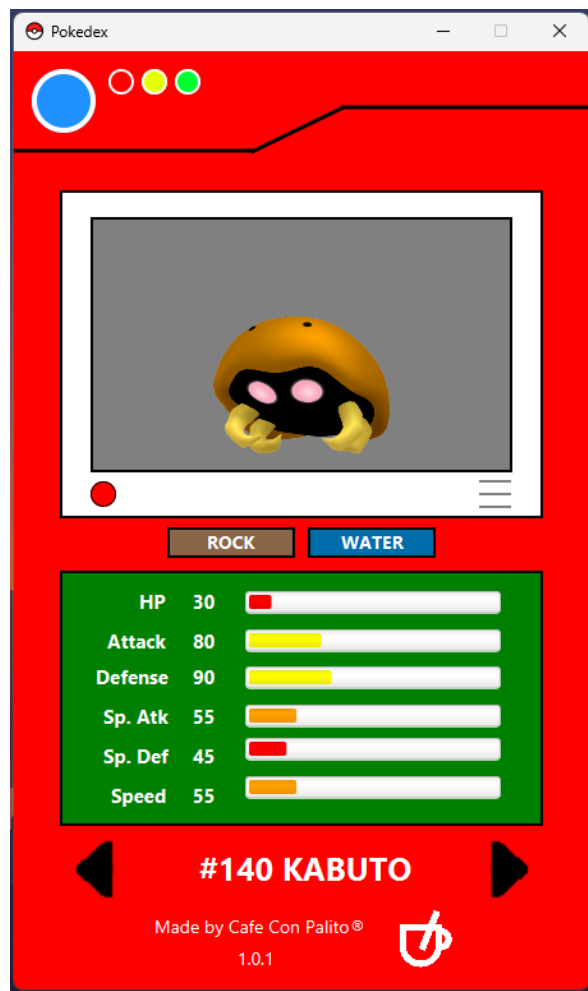
4. Demostración de funcionamiento.

La interfaz gráfica permite un uso muy sencillo de la aplicación ya que solo tiene un par de botones para navegar por la pokédex.

En ella se muestra una foto del Pokémon, unos textos con los tipos del Pokémon, ya sea uno o dos, y un cuadro con 6 barras mostrando las estadísticas.



Software 2 – Pokédex Pokemon Mew



Software 1 – Pokédex Pokemon Kabuto

5. Investigación sobre la creación de servidores API REST

Aunque la practica consistía en realizar consultas a una API REST también se puede programar un servidor, por lo que en este apartado se hace una breve explicación sobre servidores API.

5.1. ¿Qué lenguaje, framework o librería se pueden usar?

Con la mayoría de los lenguajes de programación se puede hacer una API REST, pero yo escogería uno de estos 3 que soportan conexiones concurrentes de forma nativa.

- **Lenguaje JAVA**
 - **Framework:** Spring
 - **Dependencias:** Spring Boot, Spring Data, Spring Web, Swagger
 - **Motivos para usar:** Es un lenguaje multiplataforma fuertemente tipado y fuertemente estructurado, por lo que, a un programador junior, el IDE le va a decir el más mínimo problema.
 - **Motivos para no usar:** Debes tener una buena base de programación para poder programar en este lenguaje y la configuración inicial es muy larga en el tiempo.
- **Lenguaje PYTHON**
 - **Framework:** FastAPI
 - **Dependencias:** SQLAlchemy, Alembic.
 - **Motivos para usar:** En el año 2024 es el lenguaje mas usado del mercado tecnológico, es muy simple de programar y la configuración inicial es muy rápida.
 - **Motivos para no usar:** Al no ser un lenguaje fuertemente tipado y estructura es muy fácil cometer errores de sintaxis.
- **Lenguaje C#**
 - **Framework:** .net
 - **Dependencias:** asp.net, Swagger.
 - **Motivos para usar:** Tiene una fácil configuración inicial, y es un lenguaje fuertemente tipado y estructurado, pero con funcionalidades modernas.
 - **Motivos para no usar:** Debes tener una buena base de programación para poder programar en este lenguaje.




5.2. ¿Qué elementos hardware y software necesitarías para desplegar la API REST?

En la actualidad para desplegar un servicio API REST se usa servicios en la nube como los de Amazon(AWS),Microsoft(Azure) o Google(Google Cloud).

Los requisitos de hardware pueden variar dependiendo el número de consultas simultáneas que se permitan al servicio o cantidad de datos que debe de procesar.

Los requisitos de software varían dependiendo el lenguaje usado pero lo común a todos los lenguajes es que la máquina que ejecuta el servicio debe de tener el compilador o el intérprete e instaladas las dependencias necesarias.

Para un despliegue más eficaz y rápido en producción por parte de los desarrolladores se usan herramientas de CI/CD como por ejemplo Jenkins.

	Ciclo: Desarrollo de Aplicaciones Multiplataforma Modulo: Programación de Servicios y Procesos Título: Pokedex Tarea 2 RA4
--	--

6. Bibliografía

Bayer, M., s.f. *SQLAlchemy*. [En línea]

Available at: <https://www.sqlalchemy.org/>

Hallett, P., 2024. *PokeAPI*. [En línea]

Available at: <https://pokeapi.co/>

Microsoft, 2024. *ASP.net*. [En línea]

Available at: <https://dotnet.microsoft.com/es-es/apps/aspnet>

Oracle, 2023. *docs.oracle.com - HTTPClient*. [En línea]

Available at:

<https://docs.oracle.com/en/java/javase/17/docs/api/java.net.http/java/net/http/HttpClient.html>

Ramírez, S., s.f. *FastAPI*. [En línea]


Available at: <https://fastapi.tiangolo.com/>

VMWare, 2024. *spring.io*. [En línea]

Available at: <https://spring.io/>

Voice RSS ©, 2024. *voicerss.org*. [En línea]

Available at: <https://www.voicerss.org/>

	Ciclo: Desarrollo de Aplicaciones Multiplataforma Modulo: Programación de Servicios y Procesos Título: Pokedex Tarea 2 RA4
--	--

7. Tabla de Ilustraciones

Código 1 – Ejemplo de JSONObject y JSONArray	4
Código 1 – Ejemplo de JSONObject y JSONArray	4
Código 2 – Método para realizar la consulta	5
Código 3 – Métodos para Obtener información a partir del JSON	6
Ilustración 1 -Diseño Pokedex 1ª Serie	2
Software 2 – Pokédex Pokemon Kabuto	7
Software 1 – Pokédex Pokemon Mew	7