IES INCA GARCILASO (Montilla)

2º Curso DESARROLLO DE APLICACIONES WEB **Nombre:**

DESARROLLO WEB EN ENTORNO SERVIDOR **Apellidos:**

SOLUCIÓN Prueba 3 y 4

Tiempo estimado: 160 minutos

12 Enero 2024

Normas:

- Pon tu nombre y apellidos en la parte superior de este documento.
- Para cada ejercicio realiza las capturas de pantalla necesarias donde se muestre que lo has resuelto de forma satisfactoria.
- Copia las capturas dentro de este documento.
- Al finalizar:
  - **Subir a la plataforma Moodle este documento relleno en formato PDF con la resolución de esta prueba.**
  - **Sube a Github el código e indica la URL a continuación**

URL de repositorio de Github → _____

Genera en tu PC o portátil la siguiente estructura de carpetas:

examen-dwes-3-4
├─── **ej1**
├─── **ej2**
├─── **ej3**
├─── **ej4**
└─── .git

En cada carpeta trabajarás el código correspondiente a cada ejercicio.

AYUDA para GIT

No haremos control de versiones para cada ejercicio, sino que lo hacemos de una vez para todo.

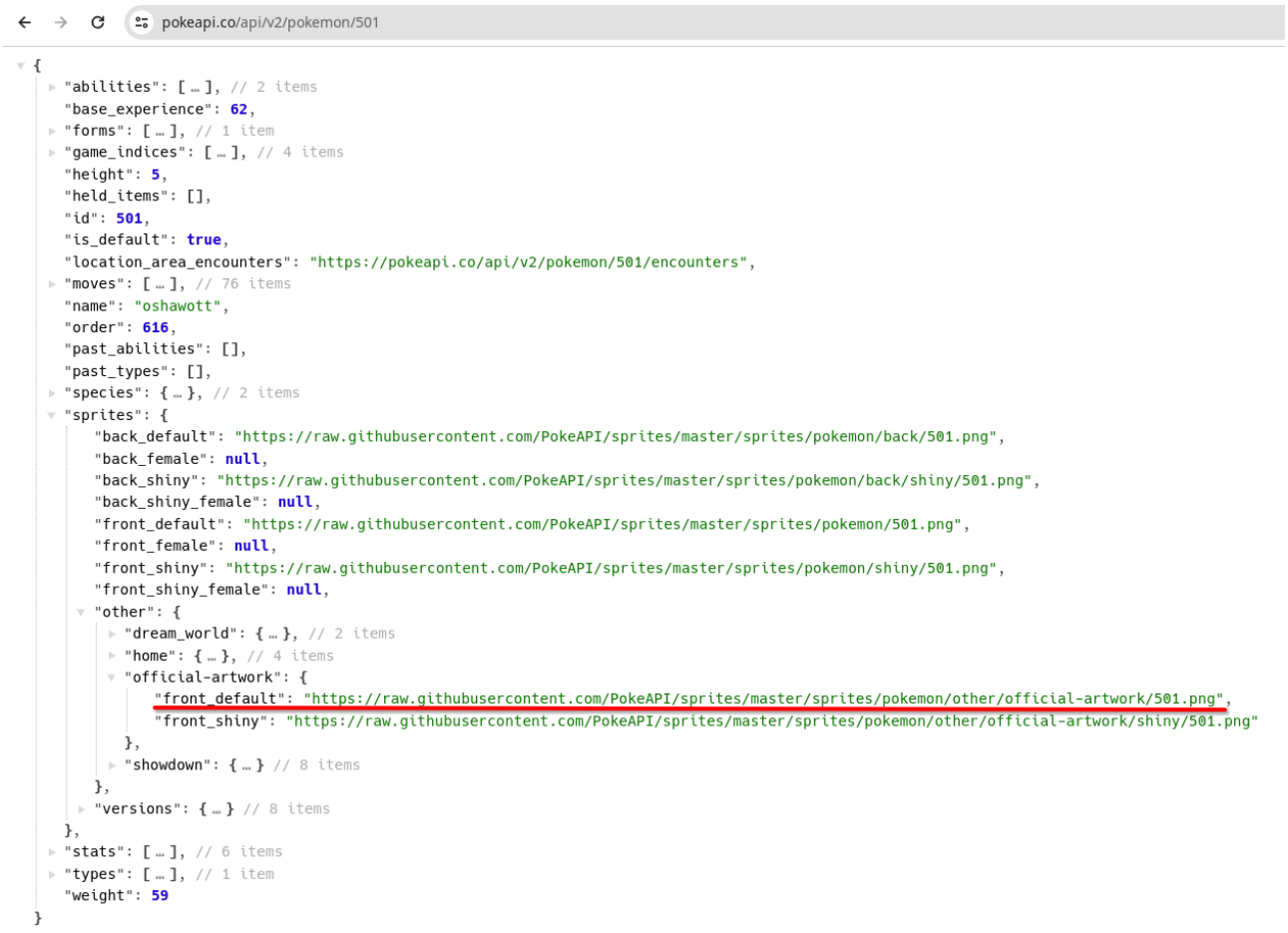
Para inicializar y guardar cambios:

```
 jose@slimbook  ~/Proyectos/examen-dwes-3-4  ls
ej1  ej2  ej3  ej4
 jose@slimbook  ~/Proyectos/examen-dwes-3-4  git init
Initialized empty Git repository in /home/jose/Proyectos/examen-dwes-3-4/.git/
 jose@slimbook  ~/Proyectos/examen-dwes-3-4  main  echo  "node_modules \n.next" > .gitignore
 jose@slimbook  ~/Proyectos/examen-dwes-3-4  main  git add .
 jose@slimbook  ~/Proyectos/examen-dwes-3-4  main +  git commit -m "Solución a Prueba DWES - Temas 3, 4"
```

El resto de pasos puedes obtenerlos desde Github al crear un repositorio nuevo.

1 TEMA 3. Ejercicio 1 (NODE+EXPRESS)

Trajabaremos con la API REST de Pokemon: (https://pokeapi.co/api/v2/pokemon/501), donde el último número indica el id del personaje.



Crea una aplicación que muestre un pequeño formulario con un input de tipo number o de tipo range y un botón de submit. El rango de números admitido está entre 1 y 1000.

El formulario enviará mediante POST dicho número al servidor, el cual obtendrá de la API REST la información del personaje solicitado y la mostrará por pantalla, tal como se ve en la captura más abajo. Se mostrará el nombre del personaje y la imagen disponible en la propiedad que aparece subrayada en la captura superior.

PISTA:

Para acceder a las propiedades de un objeto, podemos usar 2 formas:

- Primera forma: objeto.propiedad1.propiedad2
- Segunda forma: objeto[‘propiedad-1’][‘propiedad-2’]

La segunda forma la usamos cuando el nombre de las propiedades contienen caracteres como el guión medio.

Ejemplo de ejecución

localhost:3000

Personajes de Pokemon

Personaje 501

Consultar

Datos del personaje 501

Nombre: **oshawott**

Imágen:

2 TEMA 3. Ejercicio 2 (NODE+EXPRESS)

IMPORTANTE: Se comprobará que los identificadores de constantes, variables, rutas y funciones se refieren a libros y no a users, products u otros.

Crea una API REST que gestione información acerca de libros, con las propiedades que se muestran en el siguiente ejemplo:

```
{
id: 1,
titulo: "Harry Potter",
autor: "J. K. Rowling",
info: {
      editorial: "Oceano",
      paginas: 345,
      ediciones: [ 1997, 1998, 2000 ]
      }
}
```

IMPORTANTE: Las propiedades id, paginas y ediciones[] se guardarán en formato Number, aunque se envien a la API en formato string. En este caso deberás realizar la conversión necesaria.

Al iniciar la aplicación, estarán dados de alta 4 libros.

Los endpoints deben permitir:

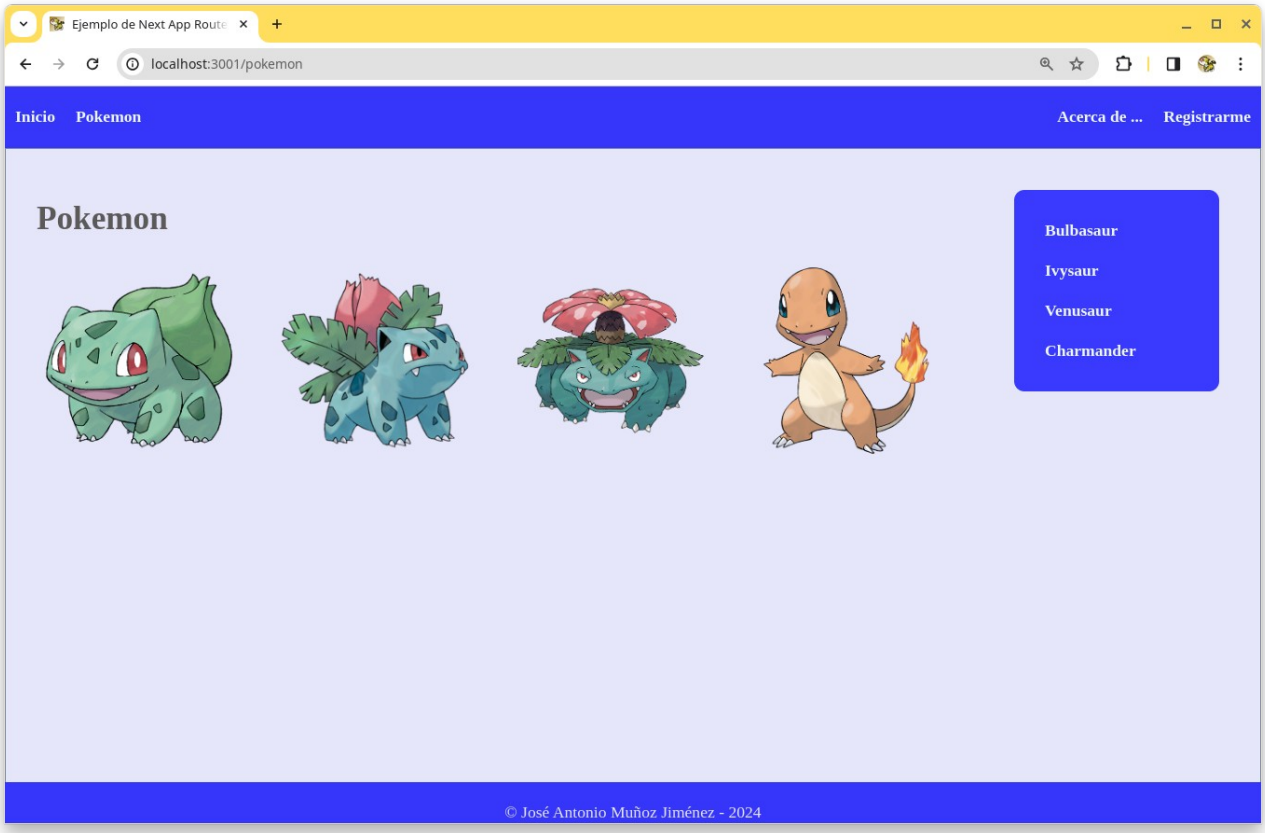
1. Listar todos los libros
2. Listar un libro por id
3. Insertar un libro (el id se generará de forma automática)
4. Modificar un libro por id
5. Eliminar un libro por id

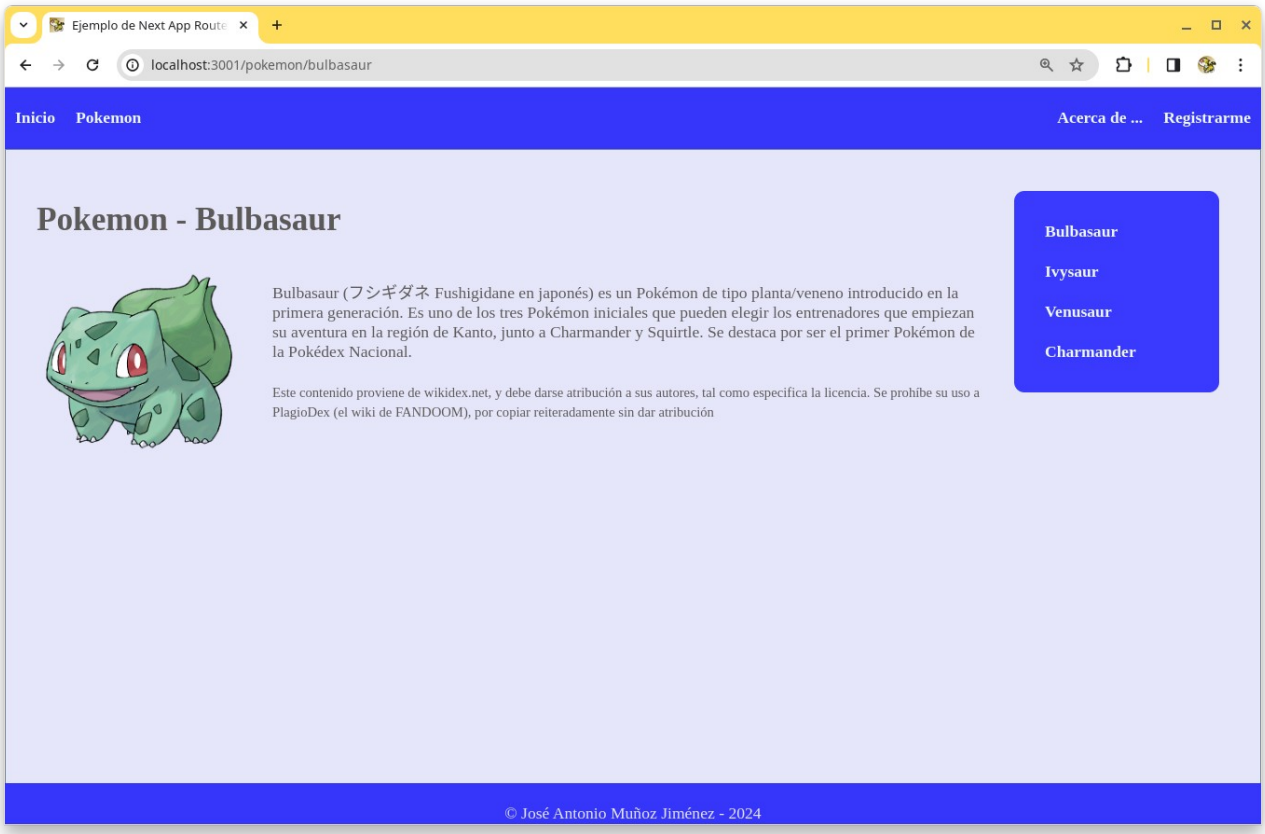
Instala el plugin Rest Client de VSCode, si no lo tienes.

Crea un archivo llamado **api.rest** con las 5 peticiones anteriores a la API. Haz una captura de cada una de ellas.

3 TEMA 4. Ejercicio 3 (NEXTJS, APP ROUTER)

Usando App Router, elabora la siguiente aplicación.





Las imágenes pueden obtenerse desde

- https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/official-artwork/1.png
- https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/official-artwork/2.png
- https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/official-artwork/3.png
- https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/official-artwork/4.png

La información de cada Pokemon puede obtenerse de
https://www.wikidex.net/wiki/Lista_de_Pok%C3%A9mon

Requisitos:

- En todas las páginas aparecerá la barra de navegación y el footer.
- En las páginas de Pokemon y subpáginas aparecerá el menú a la derecha.
- El menú anterior no aparecerá en las páginas Inicio, Acerca de… y Registrarme.
- Se valorará que el resultado final se aproxime lo máximo posible a las capturas mostradas.

IMPORTANTE: Deberás modificar el siguiente archivo para que quede así:

Puesto que las imágenes se descargaran desde `raw.githubusercontent.com` debemos configurar NextJS para que lo tenga en cuenta.

Archivo next.config.js

```
/** @type {import('next').NextConfig} */
const nextConfig = {
  images: {
    remotePatterns: [
      {
        protocol: 'https',
        hostname: 'raw.githubusercontent.com',
        pathname: '/PokeAPI/**'
      },
    ],
  }
}
module.exports = nextConfig
```

4 TEMA 4. Ejercicio 4 (NEXTJS, API, ROUTE HANDLER)

En este ejercicio se pide lo mismo que en el Ejercicio 2, pero esta vez lo haremos con NextJS.

IMPORTANTE: Se comprobará que los identificadores de constantes, variables, rutas y funciones se refieren a libros y no a users, products u otros.

Crea una API REST que gestione información acerca de libros, con las propiedades que se muestran en el siguiente ejemplo:

```
{
id: 1,
titulo: "Harry Potter",
autor: "J. K. Rowling",
info: {
      editorial: "Oceano",
      paginas: 345,
      ediciones: [ 1997, 1998, 2000 ]
      }
}
```

IMPORTANTE: Las propiedades id, paginas y ediciones[] se guardarán en formato Number, aunque se envien a la API en formato string. En este caso deberás realizar la conversión necesaria.

Al iniciar la aplicación, estarán dados de alta 4 libros.

Los endpoints deben permitir:

1. Listar todos los libros
2. Listar un libro por id
3. Insertar un libro (el id se generará de forma automática)
4. Modificar un libro por id
5. Eliminar un libro por id

Instala el plugin Rest Client de VSCode, si no lo tienes.

Crea un archivo llamado **api.rest** con las 5 peticiones anteriores a la API. Haz una captura de cada una de ellas.