Documentación del Backend - Aplicación Pokédex

Índice

- Documentación del Backend - Aplicación Pokédex
  - Índice
  - Descripción General del Proyecto
  - Stack Tecnológico
  - Estructura del Proyecto
  - Sistema de Autenticación
    - Métodos de Autenticación
    - Características de Seguridad
  - Puntos Destacables del Backend
    - Sistema de Autenticación JWT
    - Integración con PokeAPI
    - Sistema de Favoritos en Tiempo Real
  - 📌 Guía de Pruebas en Thunder Client - Pokédex
  - 🌐 URL Base de la API
  - 📌 Test
    - ◇ **Ruta para testeo**
  - 📌 Base de datos
    - ◇ **Poblar Base de Datos**
  - 📌 Autenticación (Usuarios)
    - ◇ **Registro de Usuario**
    - ◇ **Inicio de Sesión**
  - 📌 Pokémon
    - ◇ **Obtener todos los Pokémon**
    - ◇ **Obtener un Pokémon por nombre**
  - 📌 Favoritos y Ranking en Tiempo Real
    - ◇ **Añadir un Pokémon a Favoritos**
    - ◇ **Eliminar un Pokémon de Favoritos**
    - ◇ **Obtener Favoritos del Usuario**
    - ◇ **Obtener Ranking de Pokémon más favoritos (en tiempo real)**
  - 🛠️ Autor y Créditos

Descripción General del Proyecto

Servicio backend desarrollado en Node.js para una aplicación Pokédex que proporciona información sobre Pokémon, autenticación de usuarios y gestión de favoritos. El sistema se integra con la PokeAPI y mantiene una base de datos local sincronizada.

Stack Tecnológico

- **Entorno de Ejecución**: Node.js
- **Framework**: Express.js
- **Base de Datos**: MongoDB con Mongoose

- **Autenticación**: JWT
- **Plataforma de Contenedores**: Docker
- **Gestor de Paquetes**: npm
- **Integración de API**: PokeAPI

Estructura del Proyecto

```
backend/
├── config/
│   ├── db.js            # Conexión a MongoDB
│   └── cors.js          # Configuración de CORS
├── controllers/
│   ├── authController.js  # Lógica de autenticación
│   ├── pokemonController.js # Gestión de Pokémon
│   └── userController.js  # Gestión de usuarios
├── middleware/
│   └── authMiddleware.js  # Verificación de JWT
├── models/
│   ├── Pokemon.js        # Esquema de Pokémon
│   └── User.js           # Esquema de usuarios
├── routes/
│   ├── pokemonRoutes.js  # Rutas de Pokémon
│   └── userRoutes.js     # Rutas de usuarios
├── server.js             # Punto de arranque
└── app.js               # Configuración de Express
```

Sistema de Autenticación

Métodos de Autenticación

1. **Autenticación JWT**
   - Tokens JWT para sesiones
   - Encriptación de contraseñas con bcrypt
   - Middleware de protección de rutas

Características de Seguridad

- Headers de autorización Bearer
- Protección CORS
- Encriptación de contraseñas
- Validación de tokens

Puntos Destacables del Backend

Sistema de Autenticación JWT

```
// authMiddleware.js
const protect = async (req, res, next) => {
```

```
  if (
    req.headers.authorization &&
    req.headers.authorization.startsWith("Bearer")
  ) {
    try {
      token = req.headers.authorization.split(" ")[1];
      const decoded = jwt.verify(token, process.env.JWT_SECRET);
      // ...
```

Integración con PokeAPI

```
// pokemonController.js
const fetchPokemonsFromAPI = async (req, res) => {
  try {
    const baseUrl = process.env.POKEAPI_URL;
    const response = await fetch(`${baseUrl}/pokemon?limit=151`);
    // ...
```

Sistema de Favoritos en Tiempo Real

```
// userController.js
const toggleFavorite = async (req, res) => {
  const { pokemonName } = req.params;
  const userId = req.user._id;
  // ...
```

📌 Guía de Pruebas en Thunder Client - Pokédex

🌐 URL Base de la API

```
http://localhost:4000/api
```

📌 Test

◇ Ruta para testeo

Endpoint:

```
GET http://localhost:4000/api/test
```

Descripción: Comprueba que la API funciona correctamente.

Respuesta:

```
{
  "message": "API funcionando correctamente",
  "timestamp": "2025-02-15T19:07:04.219Z"
}
```

📌 Base de datos

◇ Poblar Base de Datos

Endpoint:

```
GET http://localhost:4000/api/pokemons/fetch
```

Descripción: Realiza un fetch a la PokeApi y almacena el resultado en la base de datos de MongoDB

Respuesta:

```
{
  "message": "Base de datos actualizada sin duplicados"
}
```

📌 Autenticación (Usuarios)

◇ Registro de Usuario

Endpoint:

```
POST http://localhost:4000/api/users/register
```

Descripción: Registra un nuevo usuario.

Cuerpo de la petición:

```
{
  "username": "ejemplo",
  "email": "ejemplo@email.com",
  "password": "contraseña123"
}
```

Respuesta:

```
{
  "_id": "65a48f7b34abc123",
  "username": "ejemplo",
  "email": "ejemplo@email.com",
  "token": "jwt_token"
}
```

◇ Inicio de Sesión

Endpoint:

```
POST http://localhost:4000/api/users/login
```

Descripción: Inicia sesión y devuelve un token JWT.

Cuerpo de la petición:

```
{
  "email": "ejemplo@email.com",
  "password": "contraseña123"
}
```

Respuesta:

```
{
  "_id": "65a48f7b34abc123",
  "username": "ejemplo",
  "email": "ejemplo@email.com",
  "token": "jwt_token"
}
```

📌 Pokémon

◇ Obtener todos los Pokémon

Endpoint:

```
GET http://localhost:4000/api/pokemons
```

Descripción: Retorna todos los Pokémon almacenados en la base de datos.

Respuesta:

```
[
  {
    "pokemonId": 25,
    "name": "pikachu",
    "types": ["electric"],
    "img": "url_de_imagen",
    "stats": [{ "base_stat": 55, "stat": "attack" }]
  }
]
```

◇ Obtener un Pokémon por nombre

Endpoint:

```
GET http://localhost:4000/api/pokemons/{name}
```

Descripción: Retorna los datos de un Pokémon específico.

Ejemplo:

```
GET http://localhost:4000/api/pokemons/pikachu
```

Respuesta:

```
{
  "pokemonId": 25,
  "name": "pikachu",
  "types": ["electric"],
  "img": "url_de_imagen",
  "stats": [{ "base_stat": 35, "stat": "hp" }]
}
```

📌 Favoritos y Ranking en Tiempo Real

◇ Añadir un Pokémon a Favoritos

Endpoint:

```
POST http://localhost:4000/api/users/favorite/{name}
```

Descripción: Añade un Pokémon a la lista de favoritos del usuario autenticado.

> [!NOTE] **NECESITA UN TOKEN**

Encabezados:

```
Authorization: Bearer {jwt_token}
```

Ejemplo:

```
POST http://localhost:4000/api/users/favorite/charmander
```

Respuesta:

```
{
  "message": "charmander añadido a tus favoritos",
  "pokemon": {
    "_id": "67b0d510d23ff08c3cd6a286",
    "pokemonId": 4,
    "__v": 0,
    "img":"url_de_imagen",
    "name": "charmander",
    "stats": [
      {
        "stat": {
          "name": "hp"
        },
        "base_stat": 39,
        "_id": "67b0e7232af377fdfe812449"
      },...
    ],
    "types": [
      "fire"
    ]
  }
}
```

◇ Eliminar un Pokémon de Favoritos

Endpoint:

```
DELETE http://localhost:4000/api/users/favorite/{name}
```

Descripción: Elimina un Pokémon de la lista de favoritos del usuario autenticado.

Ejemplo:

```
DELETE http://localhost:4000/api/users/favorite/charmander
```

Respuesta:

```
{
    "message": "charmander eliminado de favoritos"
}
```

◇ Obtener Favoritos del Usuario

Endpoint:

```
GET http://localhost:4000/api/users/favorites
```

Descripción: Devuelve la lista de Pokémon favoritos del usuario autenticado.

> [!NOTE] **NECESITA UN TOKEN**

Encabezados:

```
Authorization: Bearer {jwt_token}
```

Ejemplo de respuesta:

```
[
  {
    "_id": "67b0d510d23ff08c3cd6a284",
    "pokemonId": 2,
    "__v": 0,
    "img": "url_de_imagen",
    "name": "ivysaur",
    "stats": [
      {
        "stat": {
          "name": "hp"
```

```
        },
        "base_stat": 60,
        "_id": "67b0e7232af377fdfe81243d"
      },...
    ],
    "types": [
      "grass",
      "poison"
    ]
  },...
]
```

◇ Obtener Ranking de Pokémon más favoritos (en tiempo real)

Endpoint:

```
GET http://localhost:4000/api/users/favorites/ranking
```

Descripción: Retorna los Pokémon más agregados a favoritos por todos los usuarios **en tiempo real**.

Ejemplo de respuesta:

```
[
  {
    "_id": "67b0d510d23ff08c3cd6a29b",
    "pokemonId": 25,
    "img":"url_de_imagen",
    "name": "pikachu",
    "favoriteCount": 1 // <--- Usuarios que han añadido este pokemon a fav.
  }
]
```

📌 **Cuando un usuario añade un Pokémon a favoritos, el ranking se actualiza automáticamente para todos los usuarios en tiempo real.** Esto se logra mediante:

1. **MongoDB Aggregation** → Cuenta cuántos usuarios han agregado cada Pokémon.
2. **Context API (`RankingContext`)** → Gestiona la actualización del ranking en React.
3. **Actualización Automática** → Cada vez que un usuario agrega o elimina un favorito, la lista se actualiza globalmente.

🛠️ Autor y Créditos

📌 **Desarrollado por:** Ana María García.

📌 **Datos obtenidos desde:** PokeAPI.