Rapport de Projet : API Pokémon avec Ruby on Rails

Introduction :

Dans ce rapport, je présente le processus de développement et de mise en œuvre d'une API RESTful conçue pour la gestion des informations sur les Pokémon. J'ai utilisé le framework Ruby on Rails pour ce projet. Mon but principal était de créer une API qui soit non seulement efficace et sécurisée, mais aussi simple et intuitive à utiliser, tout en adhérant aux principes fondamentaux des services REST.

Choix de Technologie :

J'ai opté pour Ruby on Rails qui peut vraiment aider à développer des applications robustes assez rapidement. Ce que j'aime bien avec Rails, c'est qu'il suit une approche de "Convention plutôt que Configuration". Ça veut dire qu'on n'a pas à passer beaucoup de temps à configurer des tas de trucs au début, et que le projet reste bien organisé, ce qui est super quand on débute et qu'on veut éviter de se perdre dans son propre code.

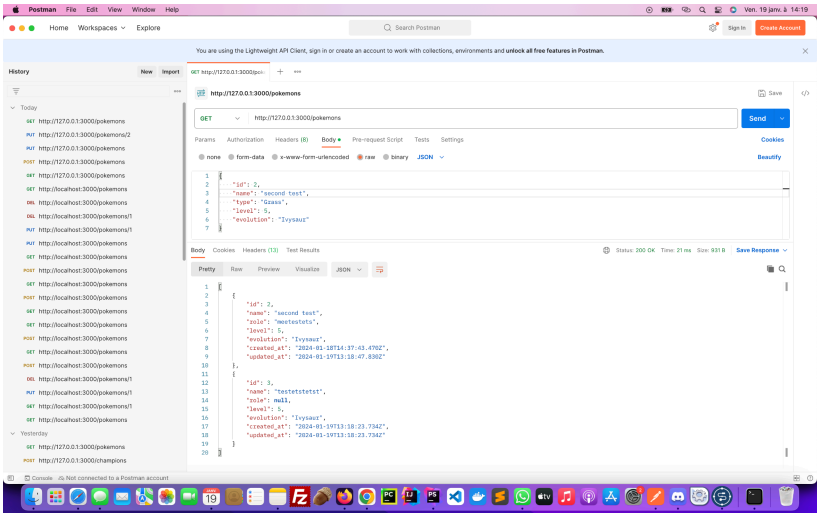
Conception de l'API :

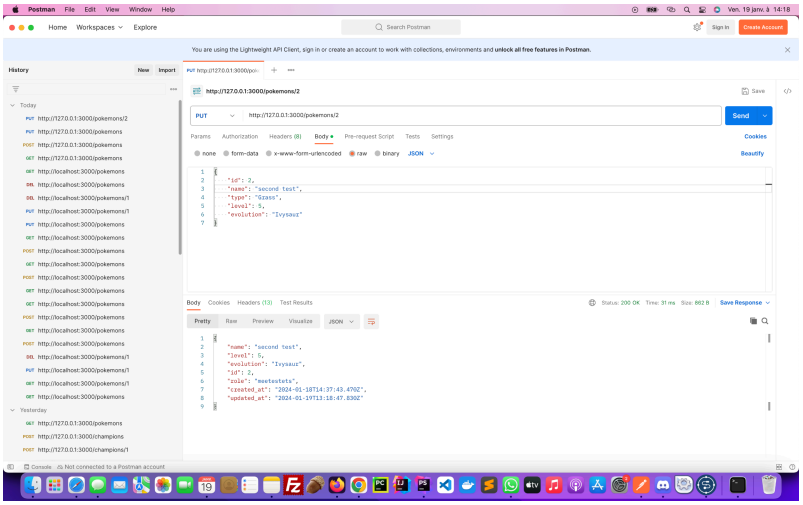
L'API a été conçue pour gérer les opérations CRUD (Create, Read, Update, Delete) pour les Pokémon. Chaque Pokémon est représenté avec des attributs tels que `name`, `type`, `level`, et `evolution`.

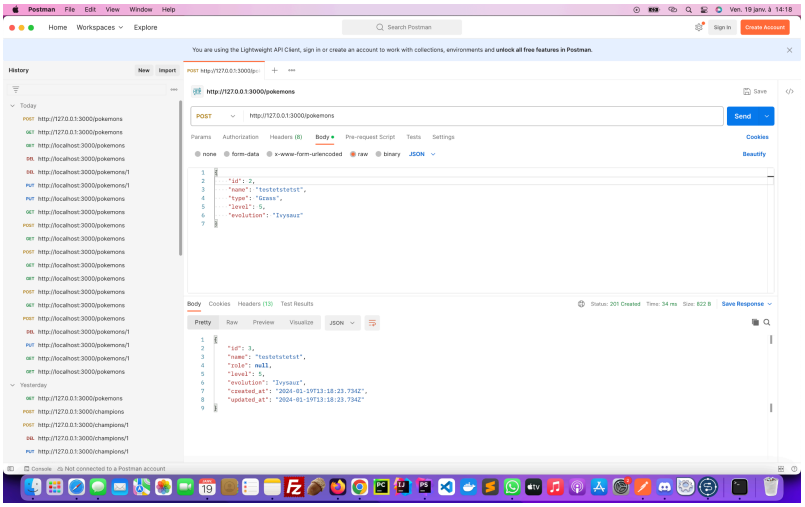
- **Modèles et Migrations** : Les modèles `Pokémon et Champion` ont été créés avec des attributs appropriés. Les migrations Rails ont été utilisées pour structurer la base de données de manière efficace.
- **Contrôleurs** : Le contrôleur `Pokemon Controller` a été conçu pour gérer les requêtes entrantes et envoyer des réponses appropriées. Des méthodes spécifiques ont été implémentées pour chaque action CRUD.
- **Routes** : Les routes ont été définies dans `config/routes.rb`, utilisant la fonctionnalité `resources` de Rails pour générer automatiquement des routes conformes aux standards REST.

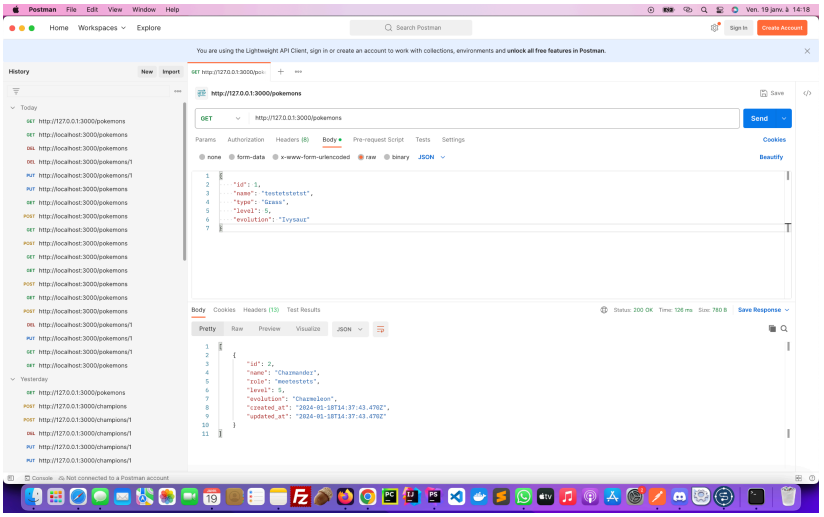
Mon approche des Tests :

Pour m'assurer que mon API était vraiment fiable et solide, j'ai pris le temps de tester que toutes les fonctionnalités clés de l'API marchaient bien, surtout pour les opérations CRUD - créer, lire, mettre à jour, et supprimer des données. En testant chacune de ces opérations, je pouvais être plus confiant que mon API faisait exactement ce qu'elle était censée faire.


Postman File Edit View Window Help
Home Workspaces Explore
Search Postman
Sign In
Create Account
You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.
History
New
Import
http://127.0.0.1:3000/pokemons
GET
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary JSON
Beautify
{
"id": 2,
"name": "second test",
"type": "Grass",
"level": 5,
"evolution": "Ivysaur"
}
Body Cookies Headers (13) Test Results
Status: 200 OK Time: 21 ms Size: 931 B
Save Response
Pretty Raw Preview Visualize JSON
[
{
"id": 2,
"name": "second test",
"role": "meetestets",
"level": 5,
"evolution": "Ivysaur",
"created_at": "2024-01-18T14:37:43.470Z",
"updated_at": "2024-01-19T13:18:47.830Z"
},
{
"id": 3,
"name": "testetstetst",
"role": null,
"level": 5,
"evolution": "Ivysaur",
"created_at": "2024-01-19T13:18:23.734Z",
"updated_at": "2024-01-19T13:18:23.734Z"
}
]
Today
GET http://127.0.0.1:3000/pokemons
PUT http://127.0.0.1:3000/pokemons/2
PUT http://127.0.0.1:3000/pokemons
POST http://127.0.0.1:3000/pokemons
GET http://127.0.0.1:3000/pokemons
GET http://localhost:3000/pokemons
DEL http://localhost:3000/pokemons
DEL http://localhost:3000/pokemons/1
PUT http://localhost:3000/pokemons/1
PUT http://localhost:3000/pokemons
GET http://localhost:3000/pokemons
POST http://localhost:3000/pokemons
GET http://localhost:3000/pokemons
POST http://localhost:3000/pokemons
GET http://localhost:3000/pokemons
GET http://localhost:3000/pokemons
POST http://localhost:3000/pokemons
GET http://localhost:3000/pokemons
POST http://localhost:3000/pokemons
DEL http://localhost:3000/pokemons/1
PUT http://localhost:3000/pokemons/1
GET http://localhost:3000/pokemons/1
GET http://localhost:3000/pokemons
Yesterday
GET http://127.0.0.1:3000/pokemons
POST http://127.0.0.1:3000/champions
Console Not connected to a Postman account


Postman File Edit View Window Help
Home Workspaces Explore
Search Postman
Sign In
Create Account
You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.
History
New
Import
http://127.0.0.1:3000/pokemons/2
PUT
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary JSON
Beautify
{
"id": 2,
"name": "second test",
"type": "Grass",
"level": 5,
"evolution": "Ivysaur"
}
Body Cookies Headers (13) Test Results
Status: 200 OK Time: 31 ms Size: 862 B
Save Response
Pretty Raw Preview Visualize JSON
{
"name": "second test",
"level": 5,
"evolution": "Ivysaur",
"id": 2,
"role": "meetestets",
"created_at": "2024-01-18T14:37:43.470Z",
"updated_at": "2024-01-19T13:18:47.830Z"
}
Today
PUT http://127.0.0.1:3000/pokemons/2
PUT http://127.0.0.1:3000/pokemons
POST http://127.0.0.1:3000/pokemons
GET http://127.0.0.1:3000/pokemons
GET http://localhost:3000/pokemons
DEL http://localhost:3000/pokemons
DEL http://localhost:3000/pokemons/1
PUT http://localhost:3000/pokemons/1
PUT http://localhost:3000/pokemons
GET http://localhost:3000/pokemons
POST http://localhost:3000/pokemons
GET http://localhost:3000/pokemons
POST http://localhost:3000/pokemons
GET http://localhost:3000/pokemons
GET http://localhost:3000/pokemons
POST http://localhost:3000/pokemons
GET http://localhost:3000/pokemons
POST http://localhost:3000/pokemons
DEL http://localhost:3000/pokemons/1
PUT http://localhost:3000/pokemons/1
GET http://localhost:3000/pokemons/1
GET http://localhost:3000/pokemons
Yesterday
GET http://127.0.0.1:3000/pokemons
POST http://127.0.0.1:3000/champions
POST http://127.0.0.1:3000/champions/1
Console Not connected to a Postman account





Conclusion :

Travailler sur ce projet a été une super occasion pour moi de mettre en pratique tout ce que j'ai appris sur le développement web avec Ruby on Rails. En fin de compte, j'ai pu créer une API qui, je pense, est à la fois claire et efficace pour gérer les infos sur les Pokémon et Champions.