Pokémon API

Yolk Studio

1 Overview

Your task is to build a Pokémon management system that handles basic CRUD operations for trainers and their Pokémon collections.

2 Core Requirements

2.1 API Endpoints

2.1.1 Trainer Management

- `POST /api/trainers` - Create a new trainer
- `GET /api/trainers` - Get all trainers
- `GET /api/trainers/{id}` - Get specific trainer with their Pokémons
- `PATCH/PUT /api/trainers/{id}` - Update trainer information
- `DELETE /api/trainers/{id}` - Delete a trainer

2.1.2 Pokémon Management

- `POST /api/trainers/{trainerId}/pokemon` - Add Pokémon to trainer
- `GET /api/pokemon` - Get all Pokémons

2.2 API Response Format

All API responses should follow this consistent structure:

```
{
  "success": true,
  "statusCode": 200,
  "message": "Pokemon found successfully",
  "data": [
    {
      "id": 1,
      "name": "Pikachu",
      "level": 25,
      "type": "Electric",
      "owner": "Ash Ketchum",
      ... // additional fields
    }
  ]
}
```

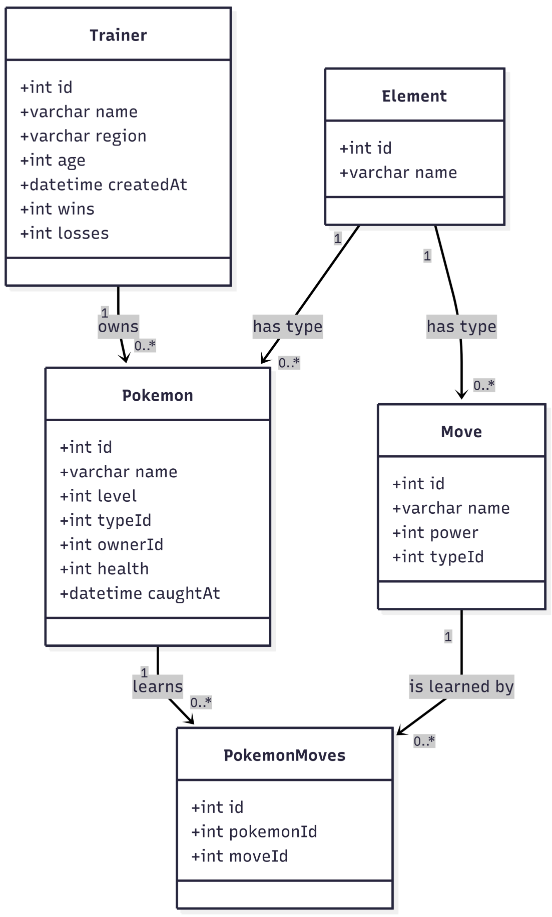
Listing 1: Standard API Response Format

3 Data Model & Database

3.1 Database Schema

The database structure follows the provided UML class diagram. Feel free to adjust the database model as needed - we are happy to discuss any modifications!

Note: For simplicity, assume that each Pokemon is unique (no case where two trainers have the same Pokemon).



3.2 Database Setup & Sample Data

We have included SQL scripts to help you get started:

- `create_pokemon_db.sql` - Creates the PostgreSQL database
- `pokemon_database_complete.sql` - Creates tables and populates with sample Pokémon data

Alternative Data Sources: You have flexibility in choosing your data source:

- **PokéAPI** (`https://pokeapi.co/`) - Comprehensive RESTful API with all Pokémon data
- **Custom Data** - Create your own dataset with preferred Pokémon and attributes
- **Hybrid Approach** - Combine provided scripts with additional data from external sources

4 Technology Stack

4.1 Recommended Technologies

- **.NET 9** - Primary framework
- **Entity Framework Core** with PostgreSQL - Data persistence
- **Swagger/OpenAPI** - API documentation
- **xUnit** - Testing framework (optional)

5 Bonus Features

Showcase your skills with these optional enhancements:

- **Advanced Search**
  - `GET /api/pokemon?name={searchTerm}` - Find Pokémon by closest name match
  - Handle typos and partial matches (e.g., "pikacu" → "Pikachu", "char" → "Charizard")
  - Support multiple search criteria (name, type, level range)
- **Filtering & Pagination**
  - Filter by type, level range, trainer region
  - Pagination with page size and page number
  - Sorting by name, level, catch date
- **Deployment & Containerization**
  - Provide Docker configuration (Dockerfile + docker-compose.yml)
  - Deploy to cloud provider (Azure, AWS, Google Cloud, or Railway)
  - Provide live demo URL if deployed

Feel free to showcase additional skills and creativity beyond these suggestions!

6 Submission Guidelines

1. Create your own repository
2. Complete the task according to the requirements above
3. Commit and push your changes regularly
4. Add a comprehensive README.md with setup instructions
5. Include any observations or design decisions you made
6. Provide us with a link to your repository

We wish you the best of luck and look forward to your solution!

— *Yolk Development Team*