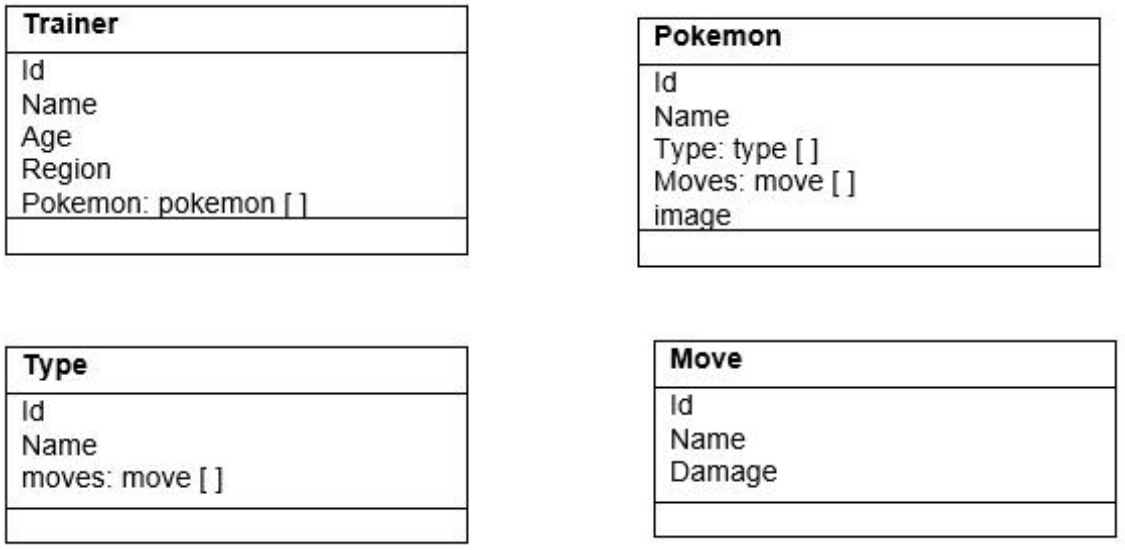
Table of Contents

Introduction

You will be creating a basic *Pokemon* app.
To do it you will need to build/consume a REST API.

Entities

These are the entities you are going to use to create the logic for our application:



The following URLs will be useful whether you are going to code the backend or the frontend, since it allows you to consume an external API.

- API Provider:
  https://pokeapi.co/
- Documentation:
  https://pokeapi.co/docs/v2
- Example of how to get a pokemon:
  https://pokeapi.co/api/v2/pokemon/ditto

Advices

You can choose between just doing frontend (in this case you will consume the pokemon API), just backend (you are going to build a REST API testable from a client like postman) or both. Doing both is a plus.

Not all challenges are required to be completed (however, complete as much as you can), choose the ones you feel more comfortable with.

Challenges

Backend

Challenge 1

- Create an API that supports all of the following endpoints. The response should use the API provider from above.

Note: You do not need to retrieve this information from a Database

Endpoint	Response
GET / pokemons/	Return an array of objects
GET / pokemons/:idPokemon	Return the specified object
GET / trainers/:idTrainer	Return the specified object

Challenge 2

- Persist the data into the database. All the requests previously mentioned should read or write the data from the chosen database.

Note: You should consider to save and get the information associated to the Entities we described in the beginning of this document.

Endpoint	Response
GET /pokemons/	Return an array from the DB
GET /pokemons/:idPokemon	Return a specific object from the DB
POST /pokemons/	Insert the object in the DB
PUT /pokemons/:idPokemon	Update the object in the DB
DELETE /pokemons/:idPokemon	Delete the object in the DB

Challenge 3

- Add support for the following features:

  Assign a new pokemon to a trainer:
  - **POST** /trainers/:idTrainer/pokemons/

  Get all the pokemons from a specific trainer:
  - **GET** /trainers/:idTrainer/pokemons

  Delete a pokemon based on its ID given a specific trainer:
  - **DELETE** /trainers/:idTrainer/pokemons/:idPokemon

Challenge 4

- Add a filter that allows you to get specific pokemons from a

  trainer: o **GET** /trainers/:idTrainer/pokemons?type=water

  With this filter you should return the pokemons from a specific trainer that match the condition, for example the pokemon type.

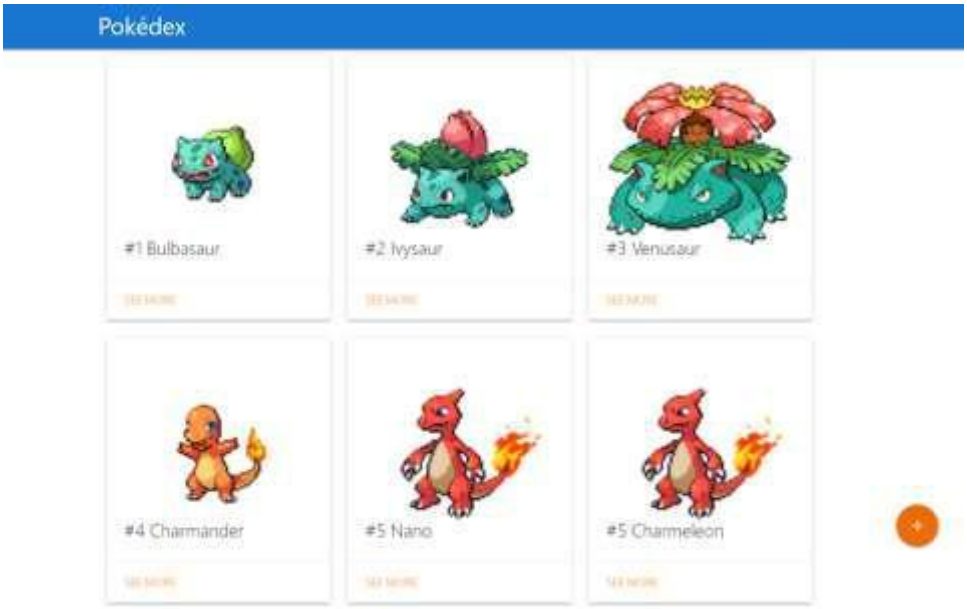
Challenge 5

- Add a pagination for:
  - **GET** /trainers/:idTrainer/pokemons?Page=2&RecordsPerPage=5

  With these parameters the endpoint should return the second page considering 5 items per page. Validate the maximum quantity of records per page that the user can send in the request, is a plus.

Frontend

The Mockups shown below are simple suggestions, you can use your own UI/UX design.



Challenge 1

Show a gallery of Pokémon, each item in the gallery should show:

- Pokémon name
- Pokémon image
- Pokémon number

The user interface must be responsive, which means that the appearance must work well on mobile and desktop devices.

API for pokemonlist: https://pokeapi.co/api/v2/pokemon?offset=0&limit=151

API for pokemonDetails providing a pokemonid: https://pokeapi.co/api/v2/pokemon/{pokemonId}/

Challenge 2

Add a "see more" button on each gallery item that will open a new page with more information about the selected Pokémon:

- Pokémon Name
- Pokémon Image

- Pokémon Number
- Pokémon Types
- Pokémon Moves

Use a suitable element to display the types and moves based on what is best based on the UX/UI.

Challenge 3

Add the ability to sort Pokémon by name, number, ascendent and descendent.

Challenge 4

Add the ability to add a new Pokémon using a form. All fields on the form should be validated and required. Use a multiselect dropdown to add the moves and types.

Challenge 5

Add the ability to edit a Pokémon information.

Challenge 6

Add the ability to remove a Pokémon. There must be a confirmation message before removing a Pokémon.

Challenge 7

Add the ability to filter Pokémon by name, moves and types.

- The user should be able to select the moves and types from a list, ex: the user can select Grass and Normal as types and Razor-Wind as movement then, the Pokémon gallery should display the Pokémon that are normal type and has razor-wind movement or the Pokémon that are of Grass type and has razor-wind movement.
- The user should be able to write a Pokémon name.