Objectives

For this assignment:

- You should use Understrap as a base template and will use the PokéAPI to fetch some data.
- We highly recommend the use of TypeScript for the frontend code.
- Understrap mix"underscores" with "bootstrap”.
- Customizing the base appearance of the template is completely optional.
- PHP version 8 must be used.
- You should avoid using ACF
- You should avoid using builder plugins like Elementor or WPBakery
- You should provide a repository to facilitate the code review process

You should:

1. Create a custom post type called “Pokémon” whose slug should be “pokemon”.
2. This post type must contain the following properties:
    a. Photo of the pokemon
    b. pokemon name
    c. pokemon description
    d. primary and secondary type of pokemon
    e. pokemon weight
    f. Pokedex number in older version of the game (you can find this info in the api)
    g. Pokedex number in the most recent version of the game (you can find this info in the api)
    h. (Optional) The attacks of said pokémon with its short description (in English).Said attacks must be stored as desired, considering efficiency and possible reuse.
3. Generate 3 pokemon manually with the data requested in point 2 using the PokéAPI.
4. Create a template for the custom post type "pokemon" and display:
    a. Photo of the pokemon
    b. pokemon name
    c. pokemon description
    d. Pokémon types (primary and secondary)
    e. Number of the pokedex in the most recent version and the name of the game
    f. Button in which, when clicked, an AJAX call is made to WordPress to show the number of the pokédex of said pokémon in the oldest version of the game with the name of said version.
    g. (optional) Table of movements of the pokémon with two columns:
        i. movement name
        ii. movement description
5. (Optional) Create a pokémon filter (TypeScript): Initially, this page will show a grid with the photos of the pokémon stored in the database. The user must be able to filter by type (the first 5 types returned by PokéAPI). When a filter is selected it should hide the photos of the pokemon whose first or second type does not match the selected filter. Limit to 6 pokemon per page.

6. (Optional) Create a custom url (for example http:/localhost/random) that shows a random pokémon stored in the database. Said URL must redirect to the permanent link of the returned pokémon.
7. (Optional) Create a custom url (for example http://localhost:generate) that when summoned will spawn a random pokemon by calling the PokéAPI. It can only be invoked by users who have post creation permissions or higher. This generated pokemon must be stored in WordPress as if it were a manually created post with the same data specified in point 2.
8. (Optional) Using the Wordpress REST API, generate an endpoint to list stored pokémon showing as ID the pokédex number in the most recent version of the game. Generate another endpoint to consult the data of the pokemon requested in point 2 in JSON format.
9. (Optional) Would it be possible to implement DAPI (or other similar APIs) in the developed solution? If so, what changes and abstractions would you propose in the different layers of the application to facilitate said integration? (the implementation is optional)
10. (Optional) The instance becomes more and more popular and starts receiving a lot of traffic generating heavy db usage. What would you do in this situation?

We will evaluate:

1. Readability of the code as well as the organization and its level of optimization.
2. Use of WordPress best practices (defined in the Coding Standards)
3. Solution scalability level
4. Testing
5. Knowledge of PHP and TypeScript
6. Setting up the development environment (how to reproduce the environment should be documented)
7. Documentation of the development process: how the solution was reached, difficulties encountered, interesting data in order to offer maintenance of the application...