

Challenge Front: *Pokémon Arena*

Este es el enunciado de la prueba técnica para el proceso de selección del equipo Front-Tech de [NTT Data](#).

La prueba consiste en crear un simulador de combate del histórico y conocido juego [Pokémon](#)

Descripción

Pokémon es un juego RPG creado por Nintendo en 1996, cuya misión en estos juegos es capturar y entrenar a estas criaturas llamadas Pokémon.

El objetivo de esta prueba es la creación de una aplicación Web para crear una arena de batalla simplificada donde podamos enfrentarnos con estas conocidas criaturas.

Esta batalla será por turnos y en cada turno el jugador al que le toca podrá elegir un ataque

A modo de ejemplo se muestra la visualización de una batalla en el propio juego



API

Para la obtención de cualquier dato que tenga relación con los pokémon, se utilizará la V2 de un API pública: <https://pokeapi.co>

Este API se puede sacar toda la información necesaria:

- Listado de pokémon y sus características: Vida, Ataques, Tipo, Daño, etc.
- Todos los tipos de ataque
- Tipo de pokémon y ratio de daño que hace a otros tipos
- Imágenes de los pokémon para la batalla

Además el API soporta multiidioma

Siguiendo las recomendaciones del propio equipo que desarrolla el API y con el objetivo de optimizar los recursos de la misma, se recomienda la utilización de un servidor que pueda cachear las peticiones a la misma. Para ello se puede utilizar esta librería y que el cliente del reto realice las peticiones frente a este servidor. La librería en cuestión es:

<https://github.com/PokeAPI/pokedex-promise-v2>

Parte 1: Juego Básico

Objetivo

Durante esta parte desarrollaremos la estructura básica del juego y la posibilidad de ejecutar batallas **1 VS COM** y decidir quién es el ganador.

En esta parte se tendrán que trabajar la parte de multilenguaje y theming.

Estructura

La aplicación consta de dos pantallas:

- La primera será **la pantalla de inicio** del juego, donde en esta primera versión se podrá configurar:
 - El **idioma**
 - El theming (**Dark o Light**) que queremos
 - El tipo de renderizado: **HTML o Canvas**. Si el usuario elige HTML, la pantalla de la arena se renderiza utilizando únicamente elementos HTML. Si el usuario elige Canvas, la parte de la arena de renderizar en Canvas 2d.
- La segunda será **la pantalla de la arena**, donde en esta primera versión se representará:
 - Un cuadro con la batalla de pokémon donde se representarán los **pokemon con sus imagenes y nombre, el estado de su vida actual, y el cuadro para elegir el ataque**(Podrás basarte en la imagen de la descripción de este enunciado)
 - Cuando la batalla haya finalizado, se mostrará al usuario en una modal quien ha ganado.

El diseño de las pantallas se deja a libre elección del candidato. No se evaluará el estilismo, aunque sí el correcto uso de HTML, CSS y JavaScript.

Ejecución

En esta primera versión, una vez el usuario ha seleccionado la configuración de la pantalla, empieza el juego.

La elección de los pokémon que se van a enfrentar será aleatoria, sin tener en cuenta sus tipos.

Cada pokémon tendrá 4 ataques que podrá utilizar. Estos ataques se elegirán también de manera aleatoria y **sólo podrán ser ataques que pertenezcan al pokémon** elegido

Una vez seleccionados los contrincantes, comenzará la batalla. Inicialmente el turno se elige de manera aleatoria y una vez que un jugador ha realizado el ataque, el turno pasa al siguiente jugador.

Durante el turno, el jugador solo podrá elegir un ataque y una vez elegido se realizará la animación que quitará vida al enemigo (siempre que ese ataque quite vida).

Tras el ataque, se reducirá la vida del contrario. Para ello habrá que tener en cuenta el **multiplicador de ataque** en función del tipo de pokemon (Esta información se encuentra en el API).

Por ejemplo un ataque eléctrico, causará más daño a un pokémon de tipo agua que a uno de tipo roca. Es importante detallar que es el tipo del ataque el que produce el multiplicador al tipo de pokemon.

Si a ese pokémon no le queda vida, **la batalla habrá terminado** e indicaremos al usuario el ganador y la posibilidad de volver a jugar.

Si al pokémon todavía le queda vida, se **pasa el turno al siguiente jugador**.

En esta primera versión jugaremos contra la computadora. En este caso cuando el turno es de la computadora, el ataque que lance se seleccionará de manera aleatoria (No se penaliza y por supuesto se valora si el candidato quiere implementar algo de IA)

Resumen de funcionalidades

- Pantalla de inicio
- Pantalla de la arena de batalla
 - Dos renderizaciones: HTML y Canvas 2d
- Dos Theming Dark y Light

- Multi-idioma
- Batalla:
 - Elección de los pokémon aleatoria
 - Elección de los 4 ataques del pokémon
 - Ataques por turno reduciendo la vida
 - Estado de finalización del juego y volver a jugar

Parte 2: Scoring y Partidas Guardadas

Objetivo

El objetivo de esta parte será añadir el scoring de las partidas a las que juguemos y poder compararnos con otros jugadores.

Ejecución

Para ello tendremos que crear algún sistema donde guardaremos la información para los usuarios y ***esta información deberá persistir en el tiempo.***

Para poder guardar el scoring, se le pedirá al usuario ***un alias en la pantalla de inicio.***

Si ese alias no existe en nuestra aplicación se le dará de alta, pero si existe recuperaremos la información de las partidas anteriores de ese usuario.

No hace falta implementar un login con usuario y contraseña o un sistema de autenticación, ya que es una aplicación de prueba. Si se desea incluir se valorará positivamente y si no se incluye NO se evaluará negativamente.

La información de las partidas anteriores y del ranking con otras personas, se mostrará en un panel en la pantalla de batalla.

Al finalizar cada batalla, se añadirá el scoring de la misma al usuario. La puntuación de la batalla será:

- 1000 ptos si el usuario ha ganado la batalla
- Se sumarán todos los puntos de daño que se le hayan hecho al pokémon contrincante

Resumen de funcionalidades

- Permitir persistir en la aplicación la información para el scoring
- Introducir el alias de usuario en la pantalla de inicio

- Añadir el scoring del usuario al final de cada batalla
- Mostrar el ranking del usuario con respecto a otros

Parte 3: Real-Time arena y Multijugador

Objetivo

El objetivo de esta parte es poder realizar batallas online 1vs1 con usuarios remotos (Que se encuentren en un ordenador diferente).

Estructura

Para poder jugar contra otro jugador, debemos darle al usuario la opción de elegir 1vsCom o 1vs1.

Si el usuario elige 1vs1, por simplificar, tendremos que permitir introducir el nombre del jugador con el que queremos jugar en la pantalla de inicio (Se podría crear un sistema de salas o partidas y poder seleccionarlas).

En la pantalla de la arena, en función de la opción elegida, deberemos poder enfrentarnos al jugador elegido y tendremos que dar un tiempo máximo para que el jugador en su turno pueda seleccionar un ataque. Pasado ese tiempo se cambiará al siguiente jugador.

Para el modo multijugador, el scoring de la batalla será diferente:

- 3000 ptos si el usuario ha ganado la batalla
- Se sumarán todos los puntos de daño que se le hayan hecho al pokémon contrincante

Resumen de funcionalidades

- Poder seleccionar el modo de juego que se quiera
- Poder incluir al jugador contrincante
- Jugar un 1vs1
- Limitar el tiempo del turno del jugador, indicándose en la partida

Tecnologías

La elección de tecnologías es libre, siempre y cuando esté basado en desarrollo **HTML**, **CSS** y **JS**.

Se puede utilizar frameworks y librerías JS/TS como Angular, React, Vue, etc.





También se puede utilizar Vanilla JS/TS.

Para la parte de estilización se puede usar tanto CSS como SCSS

Por tanto, No se podrá utilizar frameworks y librerías basadas en PHP, Java, C#, etc.

Para el despliegue se puede usar plataformas como Heroku o Netlify, o cualquier otra que se desee, el único requisito es que la ejecución sea accesible.

La aplicación debe soportar versiones iguales o superiores a:

			
Chrome 64	Edge 79	Firefox 78	Safari 12

Evaluación y Consideraciones

Para la evaluación de la prueba, además del cumplimiento de las funcionalidades, se le va a dar especial valoración a:

- Estructura del código.
- Diseño y arquitectura de la solución.
- Uso de principios de diseño de software y patrones.
- Clean code.
- Documentación y testing.
- Uso Estructura de Datos.
- Buenas prácticas del desarrollo front.

Por otro lado también serán valorados positivamente:

- Metodología de gestión para la prueba (Uso de Agile u otra metodología)
- Uso de TDD o BDD para la metodología de testing.

Formato de entrega

La entrega consta de dos partes:

- URL del código fuente, el cual debe estar en una plataforma GIT accesible (Gitlab o Github)
- URL del Código en ejecución, pudiendo utilizar plataformas gratuitas como Heroku o Netlify.

Plazo de entrega

La prueba se deberá entregar antes de que se cumplan dos semanas desde el envío de este documento.

Aún así, somos conscientes que debido a la ocupación laboral actual del candidato o a circunstancias externas, pueda necesitar una prórroga.

Si el candidato necesita una prórroga, deberá ponerse en contacto para establecer una nueva fecha máxima de entrega.

FAQS & Contacto

- *¿Es obligatorio la realización de la prueba?*

La prueba es opcional, aunque nos ayuda a evaluar mejor la situación de tu perfil dentro del equipo.

- *¿Tengo que completar toda la prueba?*

La prueba está dividida en diferentes partes que aumentan progresivamente la dificultad de la misma. Para poder evaluar la candidatura no será necesario la construcción de toda la prueba técnica, aunque sí muy valorable.

- *Tengo una duda o problema, ¿Quién me puede ayudar?*

Cualquier duda funcional, se podrá tomar un criterio propio sin problemas. El reto sienta las bases de la prueba, pero se pueden añadir extras que se consideren y si algo es ambiguo se pueden hacer suposiciones.