POKE STRATEGY

TFG

30 DE MAYO DE 2025

YESUA IBÁÑEZ CASTRILLO

Contenido

1.Introducción

El proyecto tiene como objetivo el desarrollo y el diseño de una aplicación web centrada en la temática de Pokémon. Dicha aplicación, desarrollada como un sistema completo de full-stack, permite a los usuarios registrarse, consultar información detallada sobre un Pokémon, construir equipos personalizados y analizar sus fortalezas y debilidades frente a otros tipos.

El sistema presenta funcionalidades típicas de una Pokédex con herramientas estratégicas que pueden resultar útiles tanto a jugadores casuales como a aquellos que se quieren incorporar a las batallas.

El proyecto nace con una doble finalidad, por una parte, aprender a utilizar frameworks que no se hayan visto en el curso escolar, y, por otro lado, ver resultados con de una fuente fiable y poder hacer las modificaciones correspondientes para poder trabajar con los datos que me son proporcionados.

Durante el desarrollo se han utilizado las siguientes tecnologías:

- Symfony
- Angular
- Docker
- MySQL
- Api-Plataform

Dichas tecnologías proporcionan al proyecto un entorno robusto, escalable y de fácil mantenimiento.

La motivación personal detrás del proyecto surge tanto del interés en el desarrollo web como del gusto por el universo Pokémon, lo que ha permitido una implicación creativa. Además, se ha buscado desde el principio que el resultado final pueda ser ampliado, mantenido y usado fácilmente por terceros, estableciendo las bases para una posible mejora futura hacia una herramienta más avanzada.

2. Estado del arte.

Actualmente existe una gran variedad de recursos digitales que permiten a los usuarios acceder a información sobre el universo Pokémon. Existen desde enciclopedias digitales y bases de datos hasta simuladores de combates y creación de equipos. Aunque muchos de estos recursos proporcionan funcionalidades como listados de Pokémon, estadísticas y movimientos, la mayoría carecen de elementos interactivos.

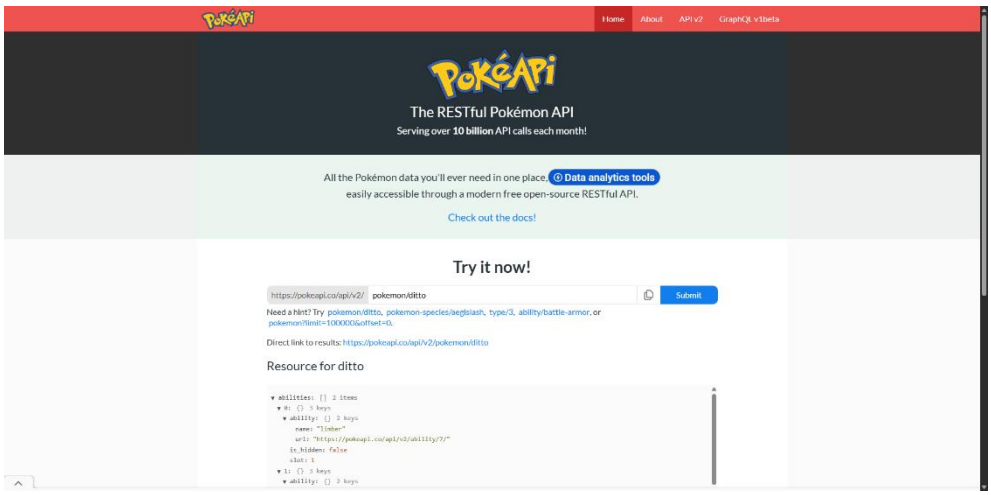
Entre los recursos digitales más destacados se encuentran:

-**PokéAPI**: es una API pública y gratuita que ofrece una interfaz para acceder a una extensa cantidad de datos estructurados en formato JSON sobre el mundo Pokémon.

Una de las funcionalidades más destacadas de PokéAPI es su buscador interactivo en la página de Inicio. A través de este buscador, los usuarios pueden consultar información detallada sobre Pokémon, como sus evoluciones o tipos, introduciendo la URL específica correspondiente a un **endpoint**.

Cuando se introduce la ruta en el buscador y se pulsa el botón “Submit”, la página muestra el contenido JSON devuelto por la API. Este JSON contiene toda la información estructurada del recurso solicitado.

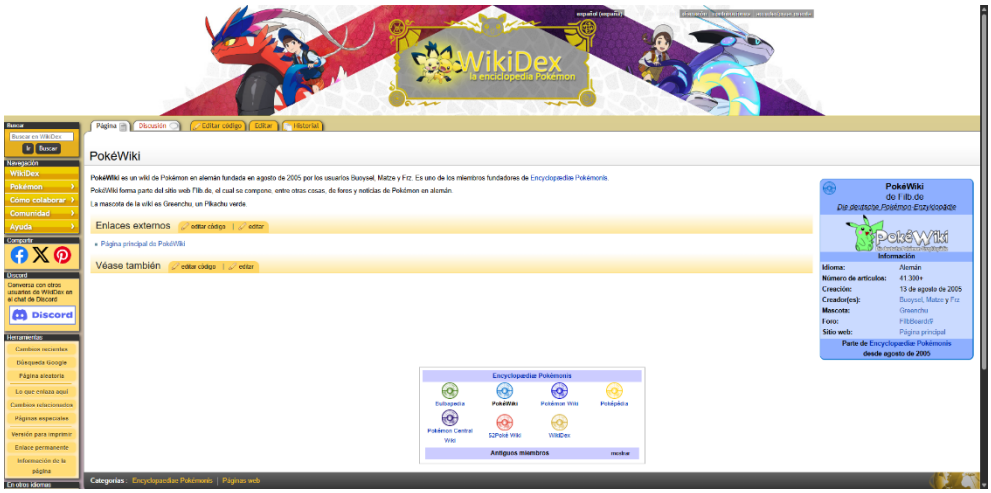
Enlace a PokéAPI: https://pokeapi.co/



-**WikiDex**: es una enciclopedia Pokémon que proporciona acceso a información sobre Pokémon, incluyendo datos sobre movimientos, estadísticas base y evoluciones, entre otros. También ofrece información relacionada con la franquicia: videojuegos, anime, manga y cartas coleccionables.

WikiDex es una wiki colaborativa y **cualquier usuario puede editar o ampliar el contenido**. Además, la plataforma ofrece un foro para debatir entre usuarios y comunicación en Discord.

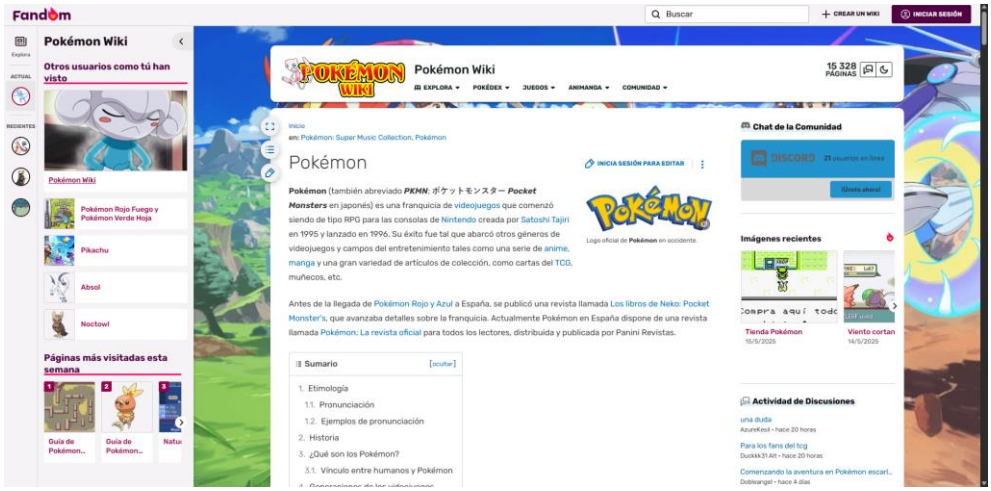
Enlace a WikiDex: https://www.wikidex.net/wiki/Pok%C3%A9Wiki



-**Pokémon Wiki**: es una enciclopedia online construida y mantenida por la comunidad, similar a Wikipedia, que ofrece contenido sobre Pokémon, incluyendo Pokédex[1], videojuegos, anime y manga.

La comunidad actualmente cuenta con más de 15.000 artículos y permite funcionalidades como el registro de usuarios, la participación en foros y la opción de crear tu propia wiki[2] temática.

Enlace a Pokémon Wiki: https://pokemon.fandom.com/es/wiki/Pok%C3%A9mon

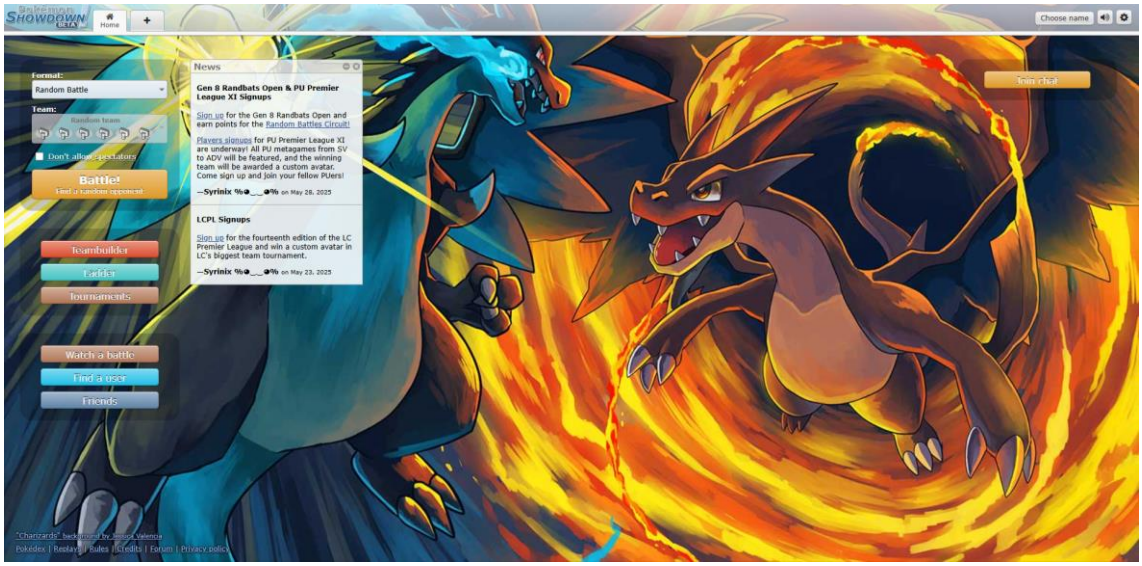


[1] **Pokédex**: enciclopedia digital del universo de Pokémon. Su función principal es **registrar y proporcionar información detallada sobre las diferentes especies de Pokémon**, incluyendo su nombre, tipo, descripción, hábitat, evoluciones y estadísticas básicas.

[2] **Wiki**: es un tipo de sitio web colaborativo que permite a usuarios crear, editar y organizar contenido directamente desde el navegador.

-**Pokémon Showdown**: es un simulador de combates Pokémon competitivo online. Permite a los usuarios enfrentarse en batallas usando equipos personalizados o aleatorios. Pokémon Showdown ofrece un chat para usuarios, creación de equipos, torneos y un sistema de combates en tiempo real, tanto como jugador como espectador.

Enlace a Pokémon Showdown: https://play.pokemonshowdown.com/



Análisis de los recursos existentes

-**PokéAPI**:

- El usuario no puede crear una cuenta.
- La efectividad entre tipos se obtiene de forma separada, no existen combinaciones automáticas para analizar fortalezas y debilidades de un equipo completo.
- Solo acepta peticiones GET. No tiene endpoints que permitan al usuario crear recursos como equipos.

-**WikiDex**:

- Gracias a la comunidad colaborativa, la información se mantiene al día con las actualizaciones continuas.
- Permite creación de usuario.
- No se pueden crear equipos.
- No existe un análisis dinámico de efectividad.
- No tiene API.

-**Pokémon Wiki**:

- Los usuarios pueden crear o editar información.
- Se pueden crear usuarios.
- Puedes crear tu propia wiki.
- No existe una carga dinámica de información, la información se introduce manualmente.

-**Pokémon Showdown**:

- Ofrece un simulador de combates en tiempo real.
- Permite crear equipos, pero no analiza dinámicamente la efectividad.
- No dispone de una Pokédex.
- Permite registro de usuario.
- Puedes competir sin necesidad de que el usuario inicie sesión. Por lo que iniciar sesión es opcional para competir, ya que el usuario puede hacerlo solo introduciendo un nombre de usuario.

Una vez hecho el análisis, se han detectado **carencias funcionales** en los recursos digitales actuales:

- Pocas herramientas que permita construir equipos de manera interactiva y las que existen, faltan de recursos. Aunque Pokémon Showdown permite construir equipos, no ofrece un análisis visual y dinámico de la efectividad del equipo mientras se construye, lo cual dificulta la toma de decisiones estratégicas.
- No se ofrecen funcionalidades para que los usuarios puedan guardar sus propios equipos y acceder a ellos desde su usuario. A pesar de que Pokémon Showdown sí que permite crear usuarios, una vez se borran las cookies se borra el equipo, ya que no cuenta con una BBDD que relacione y guarde el usuario con sus equipos guardados.
- Carencia de un análisis sobre las fortalezas y debilidades del equipo en función de los tipos de Pokémon.
- No existe una herramienta que combine información de Pokémon con la creación de equipos y análisis de efectividad entre tipos. Esto obliga a los usuarios a consultar múltiples recursos en vez de tener todo de manera centralizada.

Estas carencias evidencian la necesidad de una herramienta que permita tanto la exploración del universo Pokémon como la gestión de equipos. En base al análisis realizado, el presente proyecto propone una solución que combina una Pokédex con un sistema de análisis de equipos personalizados, integrando un sistema de autenticación y un almacenamiento de datos persistente.

3. Estudio de viabilidad. Método DAFO

Análisis DAFO

- Debilidades: tiempo limitado para la entrega, curva de aprendizaje en tecnologías nuevas como Symfony o Angular.
- Amenazas: aparición de errores complejos, dependencia de terceros (PokéAPI) y protocolos de seguridad.
- Fortalezas: motivación personal alta, diseño modular y escalable, uso de tecnologías actuales del mercado y base de datos propia.
- Oportunidades: posibilidad de evolución del proyecto, uso académico o incluso comercial, base sólida para portafolio profesional.

a. Estudio de mercado

El proyecto se sitúa en el apartado de herramientas web de Pokémon. Aunque existe Webs con esta misma temática dichas web no ofrecen ni la autentificación de usuario ni el almacenamiento persistente de datos.

1. Recursos Hardware (HW)

- Ordenador personal: Intel Core i7 (7ª generación), 16GB RAM, SSD NVMe.
- Conectividad a internet estable para pruebas y acceso a API externas.

2. Recursos Software (SW)

- Backend: Symfony 7.2.5, API Platform, PHP 8.2
- Frontend: Angular 17.3, TypeScript
- Base de datos: MySQL (gestionada con phpMyAdmin)
- Virtualización: Docker
- Testing/API: Postman,API Plataform
- Control de versiones: Git + GitHub

3. Recursos Humanos (RRHH)

- 1 desarrollador (con posibilidad de ampliación de personal para la dedicación de cada zona: frontend, backend)
- Revisión puntual del tutor académico

Planificación temporal o agenda de trabajo

1ª Semana: análisis y planificación.

2ª y 3ª Semana: Diseño de la base de datos y definición de entidades.

4ª y 5ª Semana: Implementación de API plataform con Symfony (Endpoints).

6ª y 7ª Semana: Desarrollo del Frontend con Angular y seguir con los Endpoints.

8ª Semana: Conexión frontend-backend. Página home de la web.

9ª Semana: Página de un Pokémon específico e implementación de funcionalidades de gestión de equipos.

10ª Semana: Login de usuario y correcciones de errores de funcionalidades.

11ª Semana: Diseño responsive y correcciones de errores.

12ª Semana: Documentación final.

4.Análisis de requisitos

a. Descripción de requisitos

i. Texto explicativo

Antes de comenzar con el desarrollo, se han identificado y definido los requisitos con el fin de asegurar que las funcionalidades fueran implementadas correctamente. Estos, se han organizado en: requisitos funcionales, requisitos no funcionales y diagramas de caso de uso.

-Requisitos funcionales

- Registro e inicio de sesión de usuarios.
- Consulta de información de Pokémon incluyendo id, nombre, imagen, tipos, datos del Pokémon (especie, altura, peso, generación, habilidades y habilidad oculta), evoluciones y efectividad.
- Construcción de equipos personalizados de hasta 6 Pokémon.
- Visualización de las fortalezas y debilidades del equipo en base a los Pokémon seleccionados.
- Guardado y recuperación de equipos por usuario.
- Visualización de equipos guardados del usuario.

-Requisitos no funcionales

- Interfaz responsiva que se adapte a distintos dispositivos.
- Carga eficiente de datos mediante llamadas API.
- Persistencia de datos utilizando base de datos MySQL.
- Sistema modular que permita ampliaciones futuras.

ii. Diagramas de caso de uso

-Caso de uso general: "Gestión de equipos Pokémon"

- Actores: usuario registrado
- Escenario: el usuario accede a la vista “Equipos”, selecciona diversos Pokémon, analiza la efectividad del equipo gracias al componente “Matchup Component”.

-Caso específico: "Ver la información de un Pokémon"

- Actores: usuario
- Escenario: el usuario accede a la página de Inicio, clica en alguna de las cards y accede al componente “Pokemon Detail Component”, que proporciona la información de los Pokémon.

-Caso específico: "Inicio de sesión"

- Actores: usuario registrado
- Escenario: el usuario introduce su correo y contraseña, se valida la sesión y se inicia sesión.

-Caso específico: "Creación de cuenta de usuario"

- Actores: usuario
- Escenario: el usuario introduce nombre, correo y contraseña. Se crea la cuenta de usuario y se inicia sesión automáticamente.

-Caso específico: "Guardar equipo"

- Actores: usuario registrado
- Escenario: el usuario selecciona un equipo de hasta 6 Pokémon, le da al botón “Guardar Equipo” de la vista “Equipos” y se guarda en la base de datos.

-Caso específico: "Ver equipos guardados"

- Actores: usuario registrado
- Escenario: el usuario le da al botón “Ver Equipos” de la vista “Equipos” y se muestran los equipos que haya creado.

-Caso específico: "Ver la efectividad de los equipos guardados"

- Actores: usuario registrado
- Escenario: el usuario accede a los equipos que ha creado clicando el botón “Ver Equipos” de la vista “Equipos”. Se muestran los equipos que haya creado y elige uno de ellos. El componente “Builder Component” se rellena con la información del equipo elegido y el componente “Matchup Component” muestra la efectividad del equipo.

5. Diseño

a. Diseño Entidad-Relación

Se ha elaborado un diagrama Entidad-Relación (E/R) para representar la estructura de las entidades y las relaciones que tienen entre ellas.

Las entidades principales son:

- Usuario: Gestiona el acceso y los equipos del usuario.
- Pokémon: Contiene los datos individuales de cada Pokémon.
- Tipo: Representa los tipos elementales (agua, fuego, planta, etc.).
- Pokemon_Tipo: Contiene la relación de Pokémon con sus tipos específicos
- Tipos_Eficaz: Representa los multiplicadores de un tipo contra otro (Eficaz/Poco Eficaz)
- Equipo: Contiene la asociación del equipo con su nombre y a que usuario le pertenece.
- Equipos_Pokemon: Gestiona los equipos con las ID de los Pokémons que le pertenecen.

Diseño Lógico Relacional

Se aplicado un proceso de normalización para realizar un modelo de relación de 1:N.
Las tablas se componen de los siguientes campos con sus características:

Equipo

Columna	Tipo	Nulo	Predeterminado	Comentarios
ID_Equipo *(Primaria)*	Int	No		
ID_Usuario	Int	No		
Numero_Equipo	Int	No		
Nombre_Equipo	varchar(100)	No		

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	ID_Equipo	4	A	No	
ID_Usuario	BTREE	No	No	ID_Usuario	2	A	No	

Equipos_Pokemon

Columna	Tipo	Nulo	Predeterminado	Comentarios
ID_Equipo *(Primaria)*	int	No		
ID_Pokemon *(Primaria)*	int	No		

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	ID_Equipo	4	A	No	
				ID_Pokemon	12	A	No	
ID_Pokemon	BTREE	No	No	ID_Pokemon	6	A	No	

Pokemon

Columna	Tipo	Nulo	Predeterminado	Comentarios
ID_Pokemon *(Primaria)*	Int	No		
ID_Api	Int	No		
Nombre	varchar(255)	No		
Imagen	varchar(255)	Sí	*NULL*	
Imagen 2D	varchar(255)	Sí	*NULL*	
Gif	varchar(255)	Sí	*NULL*	
Descripcion	text	Sí	*NULL*	
Evoluciona	Int	Sí	*NULL*	
Altura	Int	Sí	*NULL*	
Peso	Int	Sí	*NULL*	
Habilidades	text	Sí	*NULL*	
HabilidadOculta	varchar(255)	Sí	*NULL*	
Especie	varchar(255)	Sí	*NULL*	
Generacion	varchar(255)	Sí	*NULL*	

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	ID_Pokemon	151	A	No	

Pokemon_Tipo

Columna	Tipo	Nulo	Predeterminado	Comentarios
ID_Pokemon *(Primaria)*	int	No		
ID_Tipo *(Primaria)*	int	No		

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	ID_Pokemon	151	A	No	
				ID_Tipo	218	A	No	
ID_Tipo	BTREE	No	No	ID_Tipo	17	A	No	

Tipos

Columna	Tipo	Nulo	Predeterminado	Comentarios
ID_Tipos *(Primaria)*	Int	No		
Nombre	varchar(50)	No		
Icono	varchar(255)	Sí	*NULL*	

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	ID_Tipos	18	A	No	
Nombre	BTREE	Sí	No	Nombre	18	A	No	

Tipo_Eficaz

Columna	Tipo	Nulo	Predeterminado	Comentarios
ID_Tipo_Origen *(Primaria)*	int	No		
ID_Tipo_Destino *(Primaria)*	int	No		
Multiplicador	decimal(3,2)	No	1.00	

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo
PRIMARY	BTREE	Sí	No	ID_Tipo_Origen	18	A	No
				ID_Tipo_Destino	121	A	No
ID_Tipo_Destino	BTREE	No	No	ID_Tipo_Destino	18	A	No

Usuario

Columna	Tipo	Nulo	Predeterminado	Comentarios
ID_Usuario *(Primaria)*	Int	No		
Nombre	varchar(100)	No		
Correo	varchar(150)	No		
Pass	varchar(255)	No		

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	ID_Usuario	2	A	No	

a. Diseño Físico (MySQL)

Tabla Equipo

- ID_Equipo: int, clave primaria, no nulo
- ID_Usuario: int, no nulo
- Numero_Equipo: int, no nulo
- Nombre_Equipo: varchar(100), no nulo
- Índices: PRIMARY (ID_Equipo), índice por ID_Usuario

Tabla Equipos_Pokemon

- ID_Equipo: int, clave primaria compuesta, no nulo
- ID_Pokemon: int, clave primaria compuesta, no nulo
- Índices: PRIMARY (ID_Equipo, ID_Pokemon), índice por ID_Pokemon

Tabla Pokemon

- ID_Pokemon: int, clave primaria, no nulo
- ID_Api: int, no nulo
- Nombre: varchar (255), no nulo
- Imagen, Imagen 2D, Gif: varchar (255), opcional
- Descripcion: text, opcional
- Evoluciona: int, opcional (clave foránea a sí mismo)
- Altura, Peso: int, opcional
- Habilidades: text, opcional
- HabilidadOculta, Especie, Generacion: varchar(255), opcional
- Índices: PRIMARY (ID_Pokemon)

Tabla Pokemon_Tipo

- ID_Pokemon: int, clave primaria compuesta, no nulo
- ID_Tipo: int, clave primaria compuesta, no nulo
- Índices: PRIMARY (ID_Pokemon, ID_Tipo), índice por ID_Tipo

Tabla Tipos

- ID_Tipos: int, clave primaria, no nulo
- Nombre: varchar(50), no nulo
- Icono: varchar(255), opcional
- Índices: PRIMARY (ID_Tipos), índice único por Nombre

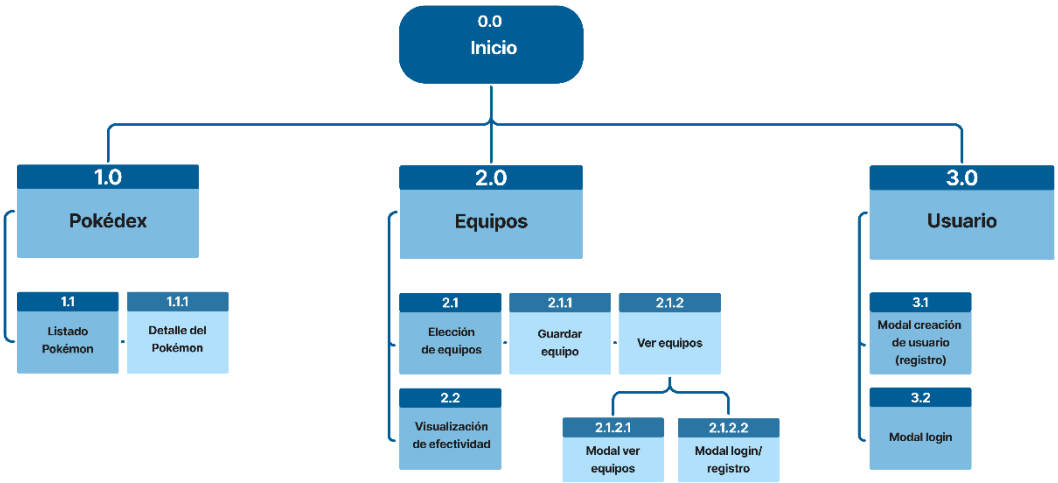
Tabla Tipo_Eficaz

- ID_Tipo_Origen: int, clave primaria compuesta, no nulo
- ID_Tipo_Destino: int, clave primaria compuesta, no nulo
- Multiplicador: decimal (3,2), no nulo, valor por defecto 1.00
- Índices: PRIMARY (ID_Tipo_Origen, ID_Tipo_Destino), índice por ID_Tipo_Destino

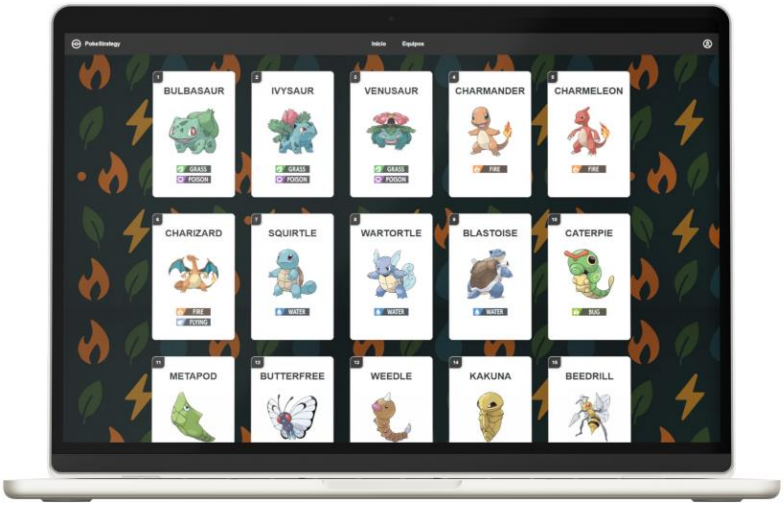
Tabla Usuario

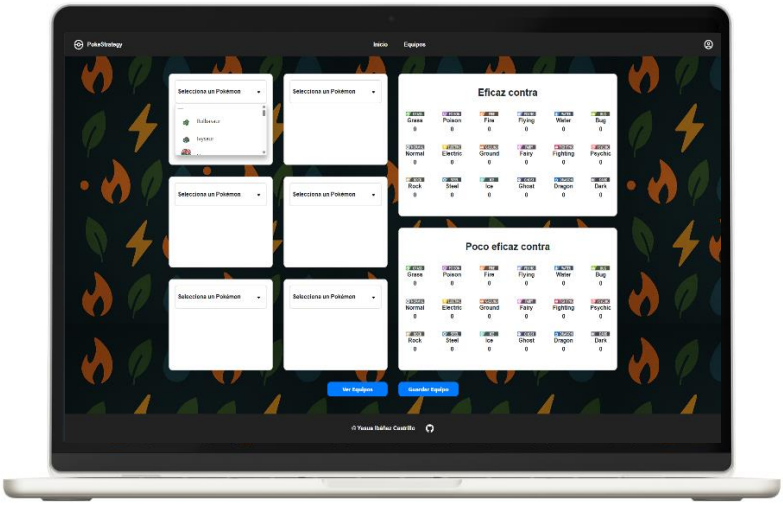
- ID_Usuario: int, clave primaria, no nulo
- Nombre: varchar(100), no nulo
- Correo: varchar(150), no nulo
- Pass: varchar(255), no nulo
- Índices: PRIMARY (ID_Usuario)

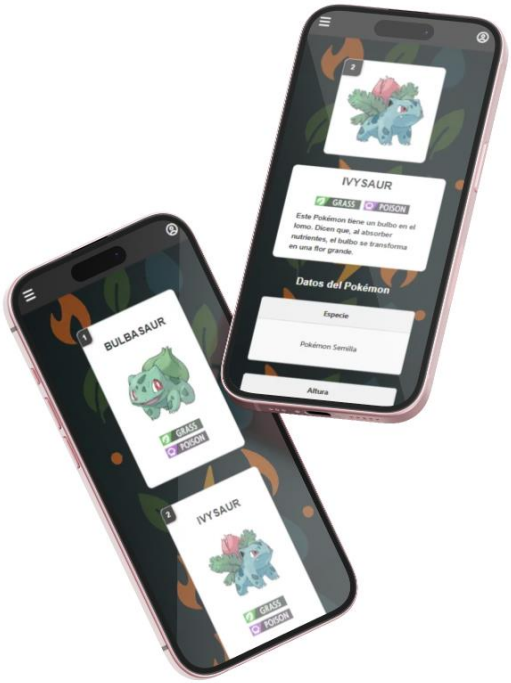
b. Mapa Web. Gráfico que muestra los enlaces entre páginas.

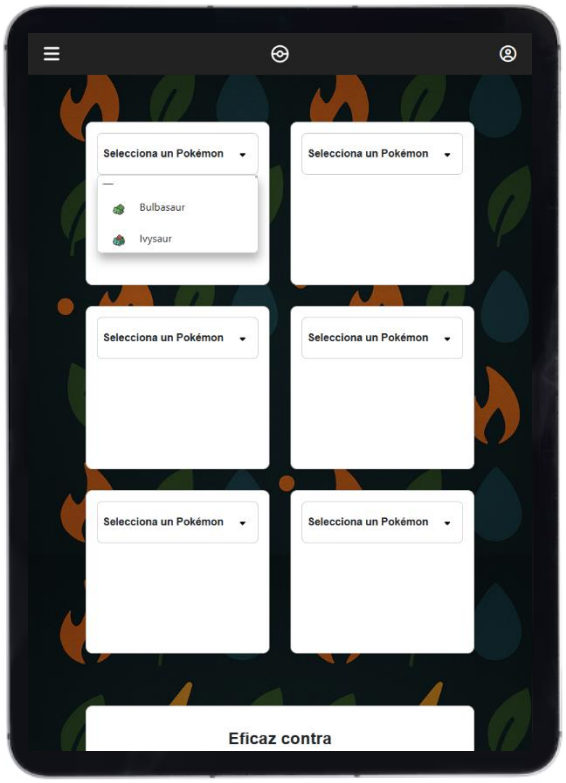


c. Mockups


PokeStrategy
Inicio
Equipos
BULBASAUR
IVYSAUR
VENUSAUR
CHARMANDER
CHARMELEON
CHARIZARD
SQUIRTLE
WARTORTLE
BLASTOISE
CATERPIE
METAPOD
BUTTERFREE
WEEDLE
KAKUNA
BEEDRILL


PokeStrategy
Inicio
Equipos
Selecciona un Pokémon
Eficaz contra
Poco eficaz contra
Ver Equipos
Guardar Equipo


BULBASAUR
GRASS
POISON
IVYSAUR
GRASS
POISON
IVYSAUR
GRASS
POISON
Este Pokémon tiene un bulbo en el lomo. Dicen que, al absorber nutrientes, el bulbo se transforma en una flor grande.
Datos del Pokémon
Especie
Pokémon Semilla
Altura


Selecciona un Pokémon
Bulbasaur
Ivysaur
Selecciona un Pokémon
Selecciona un Pokémon
Selecciona un Pokémon
Selecciona un Pokémon
Selecciona un Pokémon
Eficaz contra

d. Orientación a objetos

Se han definido clases en PHP (Symfony) que representan las entidades y sus relaciones:

- Usuario: contiene propiedades de autenticación y relación con equipos.
- Pokémon: incluye atributos como nombre, altura, tipo, habilidades y relaciones de evolución.
- Equipo: estructura de datos que agrupa hasta 6 Pokémon y pertenece a un usuario.
- Tipos: incluye los datos de cada elemento (Fuego, Agua, …) junto cos sus imágenes.
- Tipo_Eficaz, Equipos_Pokemon, Pokemon_Tipo: clases auxiliares que representan relaciones específicas.

Cada clase tiene getters/setters, anotaciones ORM y validaciones.

6. Codificación

Para la creación de la web, se optó por implementar una ***Single Page Application*** **(SPA)**. Una SPA es una aplicación web que carga una única página HTML y actualiza dinámicamente el contenido a medida que el usuario interactúa. Este tipo de aplicación permite una navegación rápida y fluida minimizando los tiempos de carga, ya que los usuarios pueden visualizar diferentes vistas sin necesidad de recargar la página.

Elegir una SPA es especialmente relevante en la actualidad, ya que la velocidad de carga y la eficiencia son factores clave para captar y mantener la atención de los usuarios. Con esta elección, se busca una mejor experiencia de usuario y reducir la tasa de abandono del sitio web.

a. Tecnologías elegidas y su justificación (lenguajes, frameworks, librerías, etc.)

Frontend

El desarrollo del *front-end* se ha llevado a cabo utilizando herramientas modernas y eficaces que optimizan el proceso de desarrollo y la experiencia de usuario. Para este proyecto, se han elegido el framework **Angular** y el lenguaje de programación **TypeScript**.

-Uso de Angular: Angular es un framework de código abierto desarrollado por Google, diseñado para crear aplicaciones web dinámicas y escalables, sobre todo se utiliza para el desarrollo de aplicaciones SPA. Este framework facilita la manipulación de datos en tiempo real y ofrece características como el enlace bidireccional de datos, la estructuración del código mediante módulos y un sistema de enrutamiento eficaz.

Debido a que este proyecto requiere una interfaz interactiva, este framework una opción que se ajusta a las necesidades. Angular ha sido elegido por su capacidad para crear SPA de forma eficiente y escalable que se ajusta perfectamente al proyecto PokeStrategy.

-Uso de TypeScript: TypeScript es un lenguaje de programación y es la versión mejorada de JavaScript, añadiendo tipado estático, lo que permite describir con mayor precisión los datos utilizados en el código. Gracias a ello, se consigue una mejor legibilidad y mantenimiento del código.

TypeScript ha sido elegido por su capacidad de prevenir errores durante la fase de compilación, antes de que el código sea ejecutado en el navegador. Esto mejora la fiabilidad y calidad del código, lo que proporciona un entorno de desarrollo más controlado y seguro.

Backend

El backend ha sido desarrollado utilizando el framework Symfony en su versión 7.2.5 junto con PHP 8.2, ofreciendo una arquitectura modular, clara y escalable.

Uso de Symfony: Symfony proporciona una estructura MVC (Modelo-Vista-Controlador) que facilita la separación de responsabilidades y la reutilización de código. También incorpora herramientas integradas como el sistema de rutas, servicios, eventos, validaciones y Doctrine ORM, lo que permite una integración directa con bases de datos relacionales. Symfony fue elegido por su en la empresa seleccionada de prácticas siendo asi una herramienta más tanto de aprendizaje como de uso en el dia a dia.

Uso de API Platform: Esta librería se integra perfectamente con Symfony y permite crear APIs automáticamente a partir de las entidades del dominio. Incluye soporte para serialización, deserialización, documentación automática con Swagger y validaciones automáticas. Se ha elegido API Platform porque reduce considerablemente el tiempo de desarrollo del backend.

Uso de PHP 8.2: La versión utilizada incorpora características modernas como tipos de retorno más estrictos, atributos, entre otras, que permiten escribir código más limpio, seguro y mantenible.

b. Entorno Servidor

El entorno servidor ha sido configurado utilizando contenedores Docker que incluyen los servicios necesarios para ejecutar correctamente el backend del proyecto. El servidor está compuesto por los siguientes elementos:

- **Symfony 7.2.5**: como framework principal del backend.
- **PHP 8.2**: ejecutado en un contenedor dedicado.
- **MySQL**: como sistema de gestión de base de datos.
- **phpMyAdmin**: para la gestión visual de la base de datos.
- **Nginx**: como servidor proxy inverso.

ii. Seguridad: evitar inyección en bases de datos

Se han aplicado diversas medidas para garantizar la seguridad del sistema y evitar vulnerabilidades como la inyección de SQL:

- **Uso de Doctrine ORM**: Doctrine permite trabajar con objetos en lugar de consultas SQL directas, evitando así la concatenación de cadenas en las consultas.
- **Parámetros preparados**: Symfony y Doctrine usan internamente sentencias preparadas para todas las consultas.
- **Validaciones de datos**: se aplican en las entidades y DTOs
- **Filtrado y saneamiento de entradas**: especialmente en los endpoints sensibles como login o creación de usuarios.

c. Entorno cliente.

i. Descripción general.

El entorno cliente se ha desarrollado con **Angular** utilizando **TypeScript** como lenguaje de programación. En concreto, la aplicación es una SPA y gracias a ello, se ha garantizado una experiencia de navegación fluida y sin recargar innecesarias entre las páginas. La SPA se comunica con el backend mediante **peticiones HTTP**, las cuales acceden a los distintos **endpoints** expuestos por el backend.

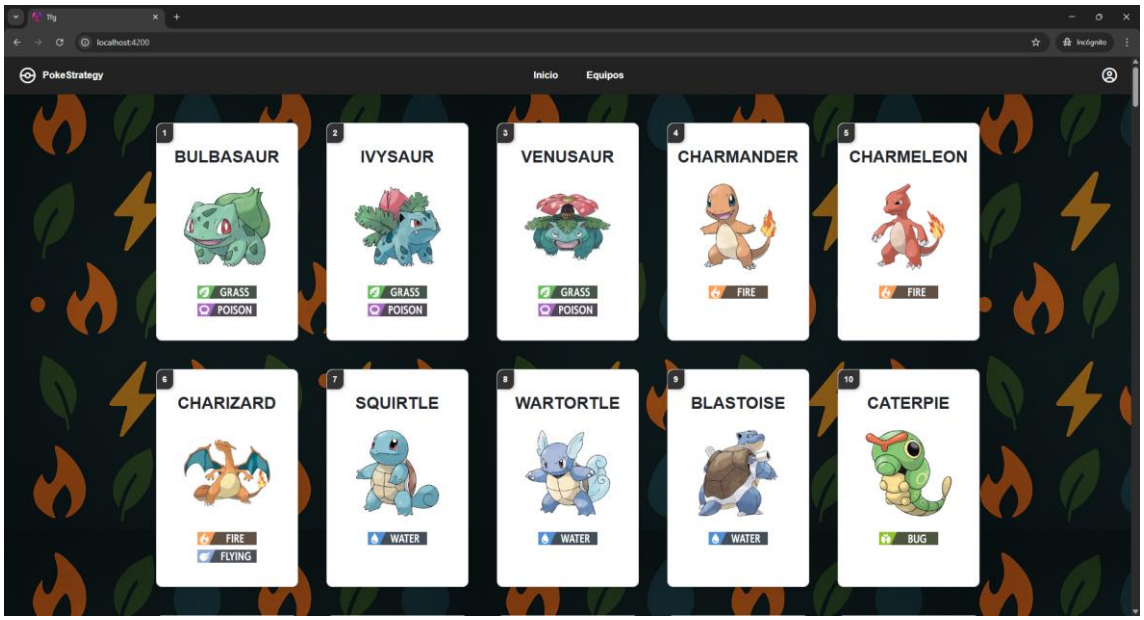
Respecto a la arquitectura de la aplicación se basa en **componentes**, lo cual permite la modularidad y la reutilización del código. Además, el proyecto está estructurado en diferentes **módulos**, organizando tanto los servicios como los componentes y las vistas.

En cuanto a los formularios, se han utilizado formularios reactivos para validar los datos introducidos, gestionar los errores y controlar el estado de los formularios de login y de registro.

Por último, la interfaz ha sido diseñada utilizando SCSS con estilos personalizados y se ha aplicado un diseño responsive asegurando que todos los componentes y vistas de adaptan correctamente a diferentes tamaños de pantalla.

ii. Asegurar la funcionalidad en los navegadores más usados (Firefox, Microsoft Edge, Opera, Safari).

La aplicación ha sido probada en los principales navegadores modernos para asegurar la funcionalidad independientemente del entorno de ejecución. Los navegadores testeados fueron Google Chrome, incluyendo modo incógnito, Firefox, Microsoft Edge y Opera.



Los aspectos que se comprobaron fueron:

- Navegación fluida.
- Correcto funcionamiento del enrutamiento.
- Renderizado correcto de los componentes.
- Verificación de estilos y comportamiento responsive.
- Gestión del almacenamiento local y recuperación de datos de usuario.
- Correcto funcionamiento de los formularios.
- Verificación de la creación de usuario y de guardado de equipos.

Los resultados fueron satisfactorios, ya que no se encontraron ni errores visuales ni lógicos que afectaran a la experiencia de usuario en ninguno de los navegadores evaluados.

d. Documentación interna de código (puede ir en un anexo).

Frontend

A continuación, se describen los pasos realizados para configurar el entorno del *front-end*, los componentes y vistas creadas junto con sus funcionalidades, cómo se ha gestionado el enrutamiento y aplicado los estilos.

i. Configuración del entorno de trabajo

En primero lugar se ha configurado el entorno de trabajo instalando las herramientas necesarias:

1. Node.js y npm: Angular requiere Node.js y su gestor de paquetes "npm". Node.js es un entorno que permite ejecutar código JavaScript en el servidor y "npm" es la herramienta oficial de Node.js para instalar, compartir y gestionar dependencias en proyectos basados en JavaScript.

2. Angular CLI (Command Line Interface): es la herramienta de línea de comandos de Angular y se ha instalado con el siguiente comando en la terminal: “npm install -g @angular/cli”

Una vez instalado Angular CLI, para la creación del proyecto se utilizó el siguiente comando: “ng new nombre-del-proyecto”. Con este comando, Angular CLI genera la estructura de carpetas y archivos necesarios para el proyecto, incluyendo el “package.json”, el cual contiene la lista de dependencias del proyecto necesarias.

Después de crear el proyecto y en la ubicación de la carpeta, se ejecutó el comando “npm install” para descargar todas las dependencias incluidas en el “package.json”.

Por último, para verificar que el proyecto se había creado de forma adecuada y funcionaba correctamente, se ejecutó el comando “npm start”, que inicia el servidor de desarrollo en aplicaciones Angular.

ii. Preparación del entorno de trabajo

El proyecto se ha estructurado y organizado siguiendo buenas prácticas de Angular:

-Se han creado componentes agrupados en módulos y carpetas según su funcionalidad, con el fin reutilizar, separar responsabilidades y permitir que sea un proyecto escalable.

-Se ha creado la carpeta "**components**" dentro de cada carpeta correspondiente, la cual tiene los distintos componentes reutilizables.

-Se ha creado la carpeta "**views**" dentro de cada carpeta correspondiente, que contiene las vistas principales de la aplicación.

-Se ha creado la carpeta "**services**" para los servicios.

iii. Creación de los componentes

El proyecto tiene una arquitectura basada en componentes reutilizables, generados con Angular CLI con el comando: “ng generate component components/nombre-del-componente”.

-**Módulo “Home”**: contiene los componentes encargados de visualizar la información principal de los Pokémon.

- **Componente PokemonListComponent**: se encarga de mostrar un listado de los Pokémon con la siguiente información de cada Pokémon: nombre, id, imagen y tipo. Los datos recibidos de los Pokémon se cargan dinámicamente desde la API mediante peticiones HTTP.
- **Componente PokemonDetailComponent**: muestra una ficha detallada de un Pokémon en concreto (nombre, id, imagen, tipo, descripción, datos del Pokémon (especie, altura, peso, generación, habilidades y habilidad oculta) y sus evoluciones). Los datos se obtienen a partir del id del Pokémon a través de la ruta “/pokemons/:id”, consumiendo la API.

-**Módulo “Teams”**: contiene los componentes relacionados con la creación y análisis de equipos de Pokémon.

- **Componente TeamBuilderComponent**: permite seleccionar Pokémon para formar un equipo de hasta 6 Pokémon. Los Pokémon se pueden seleccionar a través de una lista, donde se puede ver imagen y nombre del Pokémon. Del mismo modo que los dos componentes anteriores, los datos son traídos de la API.
- **Componente TeamMatchupComponent**: muestra la efectividad del equipo seleccionado en el componente “TeamBuilderComponent”, por lo que este componente está relacionado con el anterior. En “TeamMatchupComponent” se muestran dos tablas: una para las fortalezas del equipo y otra para las debilidades, que se calculan mediante peticiones a un endpoint personalizado “/api/tipos/{id}/efectividad. El componente presenta la información visualmente en forma de tabla de matchup y, del mismo modo que los componentes anteriores, trae información de la API, en este caso, los tipos.

-**Módulo "Shared"**: contiene componentes reutilizables presentes en todas las vistas de la aplicación.

- **Componente NavbarComponent**: es el menú de la web y está presente en todo momento. Incluye logotipo, nombre de la web, enlaces de navegación (Inicio y Equipos) y control de sesión de usuario (login y logout). Este tiene un diseño responsive, ya que se adapta a los diferentes tamaños de pantalla de diferentes dispositivos, convirtiéndose en un menú hamburguesa cuando es necesario. Por otra parte, también gestiona la apertura de los modales de autenticación.
- **Componente FooterComponent**: es el pie de página de la web, presente en todas las partes de la aplicación. Contiene el copyright, el nombre del creador de la web junto con un enlace al perfil de GitHub.

-**Módulo "Auth"**:

- **LoginComponent**:
    - Es un modal que permite a los usuarios autenticarse con su correo y contraseña. Si las credenciales son correctas, se guarda el nombre del usuario en el localStorage y se cierra el modal.
    - Se puede acceder a el de dos maneras: desde el menú o bien desde el botón "Ver Equipos" y "Guardar Equipos", ya que, si el usuario no ha iniciado sesión, el modal se abre automáticamente al clicar estos.
    - "LoginComponent" se comunica con el modal del componente "RegisterComponent" para permitir que, si el usuario no tiene cuenta, este se pueda crear una.
- **RegisterComponent**:
    - Es un modal que permite crear una nueva cuenta de usuario. Una vez creada la cuenta, se cierra el modal y se inicia sesión automáticamente.
    - Se accede a él mediante el modal del componente "LoginComponent".

iv. Creación de las vistas

Las vistas creadas son las siguientes:

-**Vista HomeComponent**: es la vista principal de la aplicación y el punto de entrada del usuario. Se accede a ella a través de la ruta "/" y es la página de inicio. Esta vista muestra el listado general de los Pokémon mediante el componente "PokemonListComponent" y permite al usuario poder acceder a los detalles del Pokémon clicando en su tarjeta, que lleva al componente "PokemonDetailComponent" mostrando el Pokémon seleccionado.

-**Vista TeamsComponent**: es la vista para crear y analizar equipos Pokémon y se accede a ella mediante la ruta "/teams". Esta vista integra los componentes "TeamBuilderComponent" y "TeamMatchupComponent" y, además, permite guardar equipos desde el botón "Guardar Equipo" y consultarlos o cargarlos de nuevo desde el botón "Ver Equipos".

v. Servicios

Se han creado los servicios necesarios organizados en las siguientes carpetas:

-**Auth**: para la autenticación

- **Login.service**: se encarga de gestionar el **inicio de sesión** de los usuarios. Este se comunica con el backend para autenticar al usuario mediante una petición POST con su correo y contraseña y si las credenciales son válidas, almacena los datos del usuario en el localStorage para mantener la sesión iniciada. “Login.service” es utilizado en el componente “LoginComponent”.
- **Usuarios.service**: sirve para la creación de nuevos usuarios. Realiza una petición POST para crear un nuevo usuario en la base de datos y es utilizado en el componente “RegisterComponent”.

-**Core**: para la lógica compartida:

- **Api.service**: es uno de los servicios principales del proyecto y es el que encapsula todas las peticiones HTTP a la API.
- **Modal-login.service**: controla la visibilidad del modal de login. Sus funciones principales son abrir o cerrar el modal de autenticación, actualizar el nombre de usuario tras el login y cerrar sesión.

vi. Implementación de estilos CSS

Para asegurar una buena organización del código y una gestión clara de los estilos, estos se han gestionado de dos maneras:

-**Estilos por componente**: cada componente tiene su propio archivo de estilos ubicado dentro de su misma carpeta, lo que facilita la mantenibilidad y la reutilización.

-**Estilos globales**: se ha creado un archivo "styles.css" en la carpeta src/, el cual contiene estilos generales que se aplican a toda la web.

vii. Diseño responsive

Se ha implementado un diseño responsive utilizando CSS Media Queries, asegurando que la aplicación se visualice correctamente independientemente del tamaño de la pantalla.

Para conseguir el diseño adaptable, se han aplicado las siguientes técnicas:

-**Puntos de quiebre (breakpoints)**: para ajustar la interfaz a distintas resoluciones de pantalla mediante media queries.

-**Flexbox**: utilización de “display: flex” junto con “flex-wrap” para adaptarse de forma automática al espacio disponible.

-Menú responsive (menú hamburguesa): en móviles y tablets, el menú de navegación se transforma en menú hamburguesa. Al hacer clic sobre el botón hamburguesa se abre, y se cierra automáticamente al seleccionar una de las páginas.

Backend

A continuación, se describen los pasos realizados para configurar el entorno del backend, las entidades y servicios creados, los controladores definidos, la configuración de seguridad y la estructura general del proyecto Symfony.

i. Configuración del entorno de trabajo

Se configuró el entorno de desarrollo backend de la siguiente manera:

1. **Instalación de Symfony CLI**: se utilizó el comando symfony new nombre-del-proyecto --webapp para crear el esqueleto del proyecto.
2. **Composer**: gestor de dependencias de PHP.
3. **Docker**: se configuró el entorno de ejecución en contenedores incluyendo servicios para PHP, MySQL y phpMyAdmin.
4. **.env.local**: se configuraron las credenciales de la base de datos y otras variables de entorno.

ii. Preparación del entorno de trabajo

El proyecto se ha organizado de la siguiente manera:

- **DTOs**: utilizados para estructurar la información devuelta por ciertos endpoints, en la carpeta src/Dto.
- **Providers**: lógica para personalizar las respuestas de endpoints específicos, organizados en src/Provider.
- **Processors**: para manejar peticiones POST y PUT, situados en src/Processor.

iii. Creación de las DTO

El sistema incluye las siguientes DTOs clave:

- Usuario: contiene campos como ID, nombre, correo y contraseña.
- Pokemon: almacena datos como ID, nombre, altura, peso, habilidades, generación, tipo, etc.
- Equipo: entidad que representa un equipo Pokémon asociado a un usuario.
- Tipo, TipoEficaz, PokemonTipo, EquiposPokemon: entidades auxiliares para representar relaciones entre tipos, Pokémon y equipos.

iv. Controladores personalizados y servicios

La lógica de negocio principal se ha delegado a:

- **Providers**: por ejemplo, EquipoCollectionProvider para obtener todos los equipos de un usuario.
- **Processors**: como GuardarEquipoProcessor que guarda un equipo enviado desde el frontend.

Los servicios utilizan inyección de dependencias y se registran automáticamente en el contenedor de Symfony.

vi. Pruebas de funcionalidad

Se utilizaron herramientas como Postman para probar todos los endpoints de la API:

- /api/pokemons, /api/equipos, /api/tipos/{id}/efectividad, etc.
- Verificación de respuestas, códigos HTTP y datos devueltos.

También se emplearon pruebas manuales desde el frontend y tests unitarios

vii. Estructura modular y buenas prácticas

- Separación de lógica en controladores, servicios y DTOs.
- Uso de serialización personalizada para mejorar las respuestas.
- Utilización de grupos de serialización (@Groups) para definir los datos que se exponen en cada operación.
- Comentarios en las clases, propiedades y métodos explicando su funcionalidad, autoría y fecha.

7. Despliegue

El despliegue se ha realizado mediante un entorno local. Por parte del backend se ha utilizado Docker, abriendo el puerto 8080 para la base de datos y el 8000/api para utilizar Symfony y API plataform. Por el contrario, para el front utilizamos Node.js para ejecutar código en el servidor abriendo el puerto 4200 por defecto.

Entorno Local

Se ha utilizado Docker para virtualizar los siguientes servicios:

- Symfony (PHP8.2)
- API-Plataform
- MySQL
- PHPMyAdmin

Se ha utilizado Node.js para ejecutar angular en el servidor:

- Angular (CLI 17.3)

Con estas metodologías hemos podido replicar el entorno, pudiendo hacer pruebas tanto de seguridad como de depuración de código.

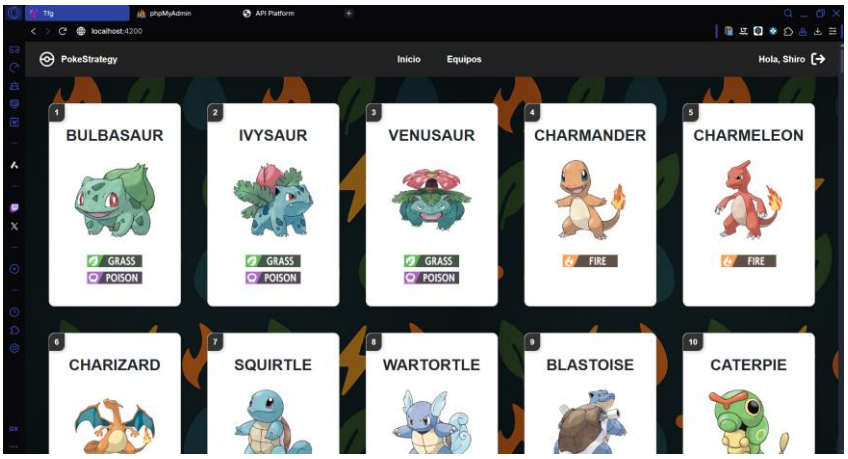
Ficheros de configuración utilizados

- docker-compose.yml: definición de los servicios.
- nginx/default.conf: configuración personalizada del proxy inverso.
- .env.local de Symfony: configuración de conexión a la base de datos.
- php.ini: configuración del intérprete PHP para desarrollo.
- angular.json y environment.ts: configuración del entorno frontend.

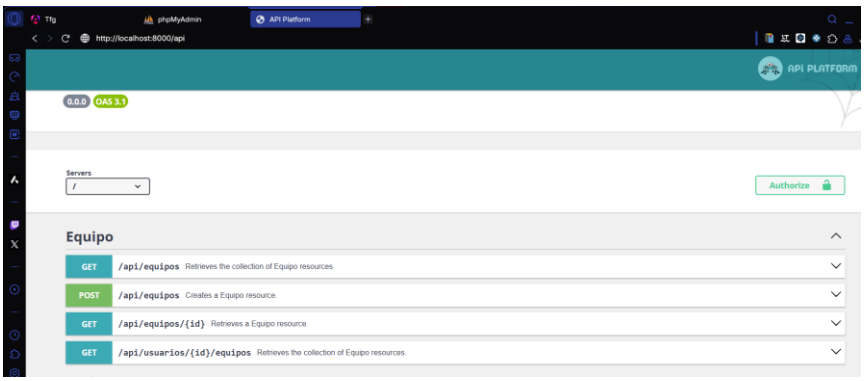
Acceso y pruebas del proyecto

Al ejecutarse de forma local, el proyecto se ha probado accediendo a:

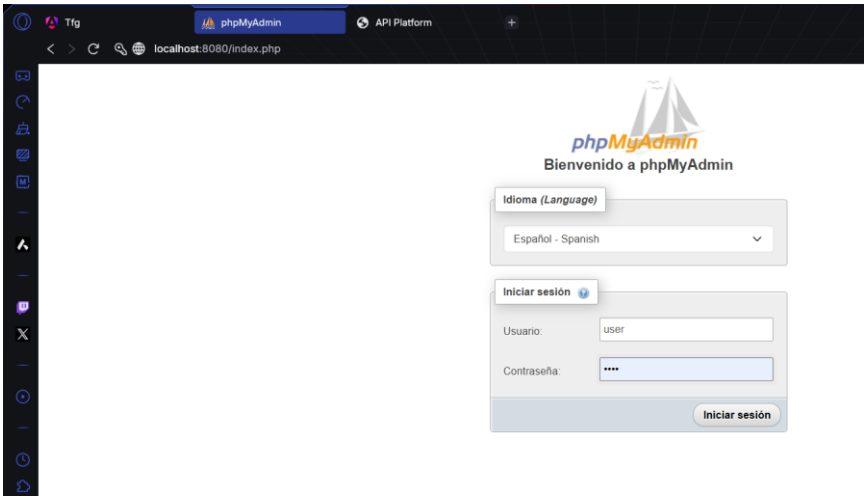
- http://localhost:4200 → Interfaz de usuario (Angular)
  Comando: npm start o ng serve


TFg
phpMyAdmin
API Platform
localhost:4200
PokeStrategy
Inicio
Equipos
Hola, Shiro
1
BULBASAUR
GRASS
POISON
2
IVYSAUR
GRASS
POISON
3
VENUSAUR
GRASS
POISON
4
CHARMANDER
FIRE
5
CHARMELEON
FIRE
6
CHARIZARD
7
SQUIRTLE
8
WARTORTLE
9
BLASTOISE
10
CATERPIE

- http://localhost:8000/api → API REST de Symfony
  Comando: docker-compose up -d –build



- http://localhost:8080 → phpMyAdmin para gestión de MySQL
  Comando: docker-compose up -d –build



8. Herramientas de apoyo

a. Control de versiones.

Se ha utilizado **Git** como sistema de control de versiones y **GitHub** como plataforma de alojamiento del repositorio remoto. Durante la creación del proyecto, se han ido haciendo “git pull” a este repositorio junto con el uso del **historial de commits** para documentar el progreso. Gracias a todo ello, ha permitido mantener el control sobre el estado del código durante el proceso de creación.

b. Sistemas de integración continua.

Durante el desarrollo, se ha seguido un flujo de trabajo basado en los principios de integración continua. Se han seguido buenas prácticas de trabajo y control de versiones con el uso de Git y GitHub, como se comentó anteriormente.

Además, se han realizado pruebas manuales en cada cambio antes de subir al repositorio, comprobando el funcionamiento general y validado el frontend y el backend en entornos locales con el fin de mantener la estabilidad del proyecto y evitar errores.

c. Gestión de pruebas.

Para garantizar la calidad y estabilidad de la aplicación, se han realizado pruebas tanto a nivel de unidad como de integración tanto del frontend como del backend.

Por parte del **frontend**, se han hecho pruebas de funcionalidad y diseño en diferentes navegadores, se ha verificado el comportamiento de formularios, servicios, enrutamiento y componentes. Por parte del **backend**, se han hecho pruebas manuales de endpoints a través de Postman y validado la autenticación y las respuestas HTTP.

En cuanto a las **pruebas de integración**, se verificó la correcta integración entre el frontend y en backend (registro, login, creación y visualización de equipos) y se validó la persistencia de datos en la base de datos y su correcta recuperación por parte del frontend.

Gracias a todas estas pruebas, se ha podido validar el correcto funcionamiento de las funcionalidades implementadas en toda la aplicación.

9. Conclusiones.

a. Conclusiones sobre el trabajo realizado

El desarrollo de este Trabajo de Fin de Grado (TFG) ha tenido como resultado una aplicación web funcional y moderna que combina las características de una Pokédex con un sistema de construcción y análisis de equipos Pokémon. A lo largo del proyecto se ha trabajado con tecnologías como Angular, Symfony, API Platform y Docker, reforzando conocimientos tanto del lado cliente como del lado servidor.

El resultado ha sido una herramienta interactiva que permite al usuario autenticarse, consultar Pokémon, crear equipos personalizados y analizar automáticamente su efectividad. Se ha logrado una integración completa entre frontend y backend, con componentes reutilizables y persistencia de datos mediante base de datos relacional (MySQL).

Este proyecto destaca frente a otras herramientas actuales relacionadas con Pokémon, ya que a pesar de la existencia de plataformas como Pokémon Showdown o My Pokémon Team, ninguna de ellas ofrece persistencia de datos en base de datos ni funcionalidades personalizadas por usuario, como las que en PokeStrategy se han implementado.

b. Conclusiones personales

Este proyecto ha sido un gran reto, pero también una oportunidad de crecimiento académico y profesional. Gracias al desarrollo de la aplicación se han puesto en práctica competencias adquiridas durante el grado, como el diseño de bases de datos o el diseño frontend. También se han implementado competencias aprendidas de manera autodidacta y las adquiridas durante mi periodo de prácticas, como ha sido el uso de los frameworks Angular y Symfony y la creación de una API desde 0.

Además, ha sido una buena experiencia a nivel personal, ya que el tema principal es del universo Pokémon. El haber construido un sistema completo desde cero y haber creado una herramienta práctica y funcional, que integra distintos aspectos del desarrollo web full-stack, aporta una gran satisfacción personal y profesional, sirviendo como base sólida para futuros proyectos y pudiendo ser un proyecto profesional para el portfolio.

c. Posibles ampliaciones y mejoras

Aunque la aplicación desarrollada cumple con todos los objetivos iniciales, con el fin de enriquecer aún más la web, se plantea una futura ampliación de funcionalidades. Las propuestas son las siguientes:

- Implementar una lista de favoritos para que cada usuario pueda marcar y acceder rápidamente a sus Pokémon seleccionados.
- Ampliar la Pokédex, ofreciendo más generaciones Pokémon.
- Permitir la edición y eliminación de equipos guardados por parte del usuario.
- Añadir visualizaciones estadísticas como, por ejemplo, gráficas de efectividad por equipo.
- Implementar un sistema de combates entre usuarios.
- Implementar en la BBDD los movimientos de cada Pokémon.

10.Bibliografía

a. Artículos

Durante el TFG se han consultado distintos artículos técnicos y recursos online que han servido como guía para la implementación de funcionalidades clave del proyecto.

- SymfonyCasts. *DTO Data Transformer*.

  https://symfonycasts.com/screencast/api-platform2-extending/data-transformer

  Este artículo ayudó a estructurar correctamente los DTOs para separar las entidades de las respuestas de la API y aplicar buenas prácticas de desacoplamiento.

- SymfonyCasts Blog. *API Platform: A REST and GraphQL framework on top of Symfony*.

  https://api-platform.com/docs/core/dto/

  Permitió entender la integración de DTOs con API Platform, definir operaciones personalizadas y controlar serialización mediante @Groups.

b. Direcciones web

- PokéAPI.

  https://pokeapi.co/

  PokéAPI ha sido la fuente de datos primaria del proyecto. A través de sus endpoints REST se han importado datos sobre Pokémon, como, por ejemplo, sus tipos y evoluciones, para añadir dicha información a la base de datos del proyecto.

- API Platform.

  https://api-platform.com/

  Ha sido la base sobre el que se ha estructurado el backend del proyecto. Su generación automática de endpoints ha facilitado un desarrollo ágil de la API.

- Docker.

  https://www.docker.com/

  Herramienta utilizada para los distintos servicios del sistema. Docker ha permitido crear un entorno de desarrollo replicable y aislado con Symfony, MySQL y phpMyAdmin.

- My Pokémon Team.

  https://mypokemonteam.com/

  Herramienta online de construcción de equipos Pokémon. Se consultó para analizar su enfoque visual y estructura, lo cual inspiró la interfaz del TeamBuilderComponent.

- Pokémon Showdown.

  https://pokemonshowdown.com/

  Plataforma de simulación de combates Pokémon. Aunque más centrado en el aspecto competitivo, sirvió de referencia para entender cómo presentar fortalezas y debilidades por tipos de manera clara.

- WikiDex y Pokémon Wiki.

  https://www.wikidex.net/

  https://pokemon.fandom.com/es/wiki/Pok%C3%A9mon

  Enciclopedias sobre el universo Pokémon. Se utilizó como fuente complementaria para verificar información textual y visual que no estaba disponible de forma directa en PokéAPI.