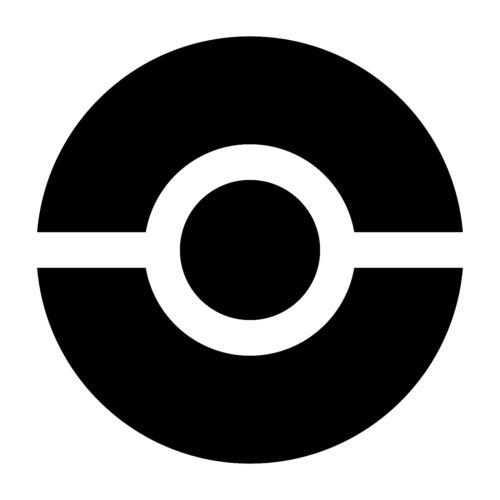
Grupo 2:

Centro Pokémon



UNAI CANO

IGOR BUENO

CLEMENTE RODRIGUEZ

# Índice

[**Índice 1**](#_heading=h.gjdgxs)

[**Introducción 2**](#_heading=h.30j0zll)

[**Objetivos 2**](#_heading=h.3znysh7)

[**Definición de la herramienta 3**](#_heading=h.tyjcwt)

[Bases de datos 3](#_heading=h.3dy6vkm)

[Definición del modelo 3](#_heading=h.1t3h5sf)

[Usuarios y roles 3](#_heading=h.4d34og8)

[Procesos automáticos 3](#_heading=h.2s8eyo1)

[Programación 4](#_heading=h.3rdcrjn)

[Pojos 4](#_heading=h.26in1rg)

[Ventanas 4](#_heading=h.lnxbz9)

[Lógica 4](#_heading=h.35nkun2)

[Lenguaje de marcas (sólo DAM) 5](#_heading=h.1ksv4uv)

[Sistemas informáticos 6](#_heading=h.44sinio)

[Entornos de desarrollo 7](#_heading=h.2jxsxqh)

[Diagrama de Clases 7](#_heading=h.z337ya)

[Diagrama de Paquetes 7](#_heading=h.3j2qqm3)

[Diseño final 7](#_heading=h.1y810tw)

[Pruebas JUnit 7](#_heading=h.4i7ojhp)

[**Conclusiones 8**](#_heading=h.2xcytpi)

Todo el proyecto debe estar en lenguaje impersonal (3ª persona)

Los márgenes deben estar justificados

# Introducción

Se va a desarrollar un sistema de almacenamiento de Pokémon, emulando la idea original de los propios juegos, en el cual el usuario tendrá un equipo de 6 pokémon en el inventario y un sistema de almacenamiento de estos, para poder almacenar los excedentes del equipo en un “PC”. Así mismo, en el propio pc podrá manipular la posición de estos, añadir más Pokémon a la lista, liberarlos (eliminarlos) de dicha caja o moverlos de posición en esta desde una interfaz gráfica y fácil de entender.

Así mismo se hará un menú de administración el cual se ha bautizado como “Profesor” cuya función es modificar los datos existentes en los pokémon, desde sus estadísticas hasta sus parámetros básicos como nombre, etc.

# Objetivos

Es un servicio que ofrece a los jugadores la opción de almacenar los pokémon que hayan obtenido de forma que no se pierdan, que sea modificable y de fácil manejo.

# 

# Definición de la herramienta

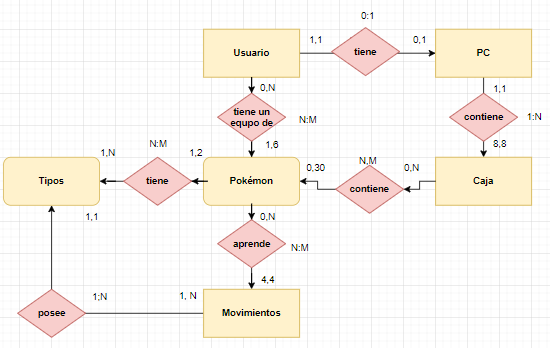
En este apartado se va a definir la funcionalidad que se va a desarrollar durante el reto, sin meterse en cómo se va a realizar técnicamente. Es necesario definir el alcance que va a tener en cada uno de los apartados:

## Bases de datos

Deberá incluir los siguientes puntos.

### Definición del modelo

Este es el modelo E/R básico



Incluir también las vistas que se van a crear.

–

### Usuarios y roles

Definición de usuarios y roles, junto con permisos a nivel general que van a tener.

### Procesos automáticos

Listado y breve explicación de los procesos automáticos (procedimientos, funciones y triggers)

**Máximo 3 páginas**

## 

## Programación

En este apartado se explicará la funcionalidad que va a cubrir el desarrollo, incluyendo los siguientes apartados:

### Pojos

Las entidades que se van a utilizar en nuestro proyecto serán las siguientes:

-**Usuario**: clase abstracta que se divide en 2

-**Profesor**: clase de administrador, solo puede modificar los datos de su generación (Ya que los pokémon disponen de generación propia)

-**Jugador**: clase que utiliza el emulador, dispone de un pc y hasta 6 pokémon

-**MiPc**: clase que almacena las cajas contenedoras de pokemon, solo se le asigna una a cada jugador

-**Cajas**: contenedores individuales que contienen hasta 30 pokemon cada una.

-**Pokemon:** entidades con una serie de atributos únicos.

-**Movimientos**: cada Pokémon dispone de 4 movimientos

-**Tipo**: cada movimiento tiene un solo tipo y cada pokemon tiene de 1 a 2 tipos.

### Ventanas

Se dispondrá, en principio, de 6 ventanas:

**-Inicio**: una página de Inicio, dispone de 2 opciones: iniciar sesión o registrarse.

**-Login:** En esta ventana se pide el usuario y contraseña.

-**Registro**: una página de registro, solo se podrá crear un usuario **Jugador** desde aquí.

-**Página de administrador:** Solo se puede acceder mediante el usuario **Profesor**. Aquí se podrán modificar los valores de los pokémon a nivel general, sus estadísticas, nombre, etc…

**-Página de PC:** Aqui solo se podrá acceder mediante el usuario **Jugador**, se dispone de una lista contenedora para el equipo del jugador, que contiene hasta 6 pokémon, y otra para el almacenamiento de estos externamente, la cual se puede rellenar hasta 30 unidades por caja, y en cada posición se almacena un pokemon, dispone de una serie de opciones para gestionar, desde cambiar la caja en uso, eliminarlos de la lista, ver sus datos o pasarlos al aquí.

**-Página de Perfil:** En esta ventana el usuario tiene la posibilidad de cambiar sus datos y darse de baja si quiere.

### Lógica

La funcionalidad que se va a implementar en cada uno de los componentes, y la navegación entre ellos.

**-Inicio:** Desde aquí se puede llegar a Login y Registro con los botones respectivos.

**-Registro:** Tiene campos de texto para rellenar con los datos y un botón para verificar los datos introducidos y registrarse.

**-Login:** Tiene campos de texto para el usuario y contraseña y un botón para validar lo introducido.

-**Página de administrador:** dispone de una tabla de pokemons, al seleccionar uno tiene la posibilidad de cambiar sus estadísticas y hacer ese cambio permanente.

**-Página de Perfil:** campos de texto con sus datos para cambiarlos si asi lo quiere o darse de baja.

**-Página de PC:** tiene una tabla con celdas que pueden contener pokemon, botones para sacar, meter, registrar y revisar pokémon, también unos botones para cambiar de caja del pc.

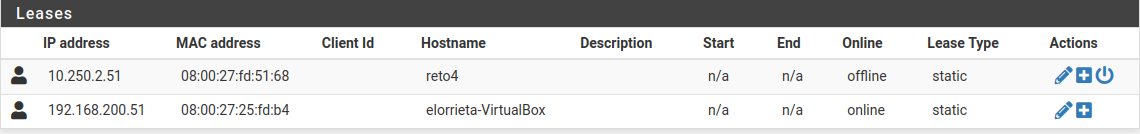
**Máximo 2 páginas**

## Lenguaje de marcas (sólo DAM)

Se deberá añadir una imagen de la jerarquía XML

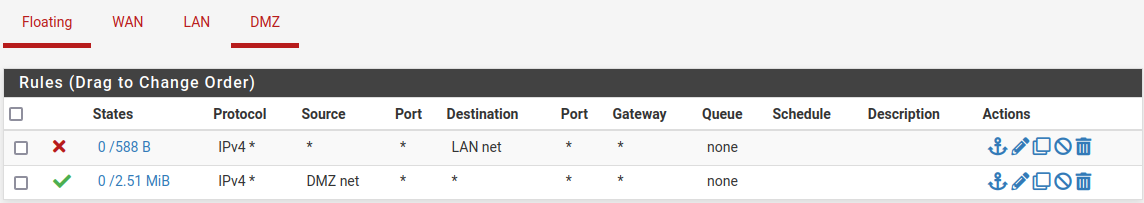
**Máximo 1 página**

## Sistemas informáticos



→reto4 : ubuntu server en red DMZ

→elorrieta-VirtualBox: ubuntu desktop en red LAN



Reglas DMZ.

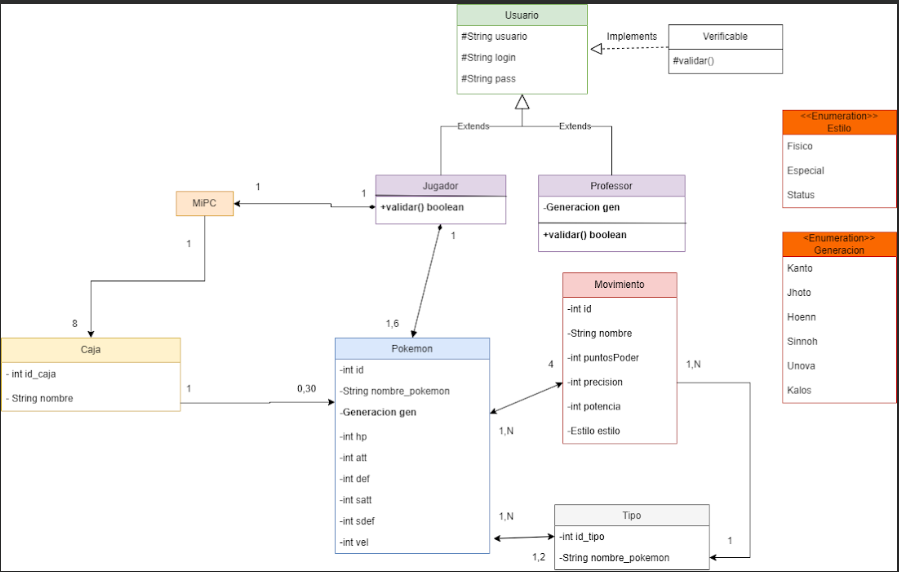
-Regla 1: de WAN no se puede conectar a ningun equipo LAN

-Regla 2: permiso a DMZ para conectar a cualquier destino

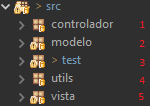
## Entornos de desarrollo

Añadir los diagramas de diseño.

### Diagrama de Clases



### Diagrama de Paquetes



1→ Aquí en el controlador tenemos la clase de métodos.

2→ Aquí en modelo tenemos todas las clases básicas, abstractas, enumeraciones e interfaces.

3→ Aquí tenemos los tests a los que sometemos a las clases y los métodos.

4→ Aquí en utils tenemos la conexión con la base de datos y las rutas a las imágenes y dinamizadores de estas.

5→ Por último, en vista guardamos todas las ventanas gráficas.

### Diseño final

Deberá incluir:

* Documento de Diseño Inicial, pero actualizado a su diseño definitivo.
* Resumen de Cambios significativos
* Conclusiones sobre el proceso de diseño.

### Pruebas JUnit

Se deberá añadir una captura con la explicación correspondiente de los siguientes componentes:

* Operaciones Select, Insert, Update, Delete (CRUD) sobre la tabla usuario y administrador
* Operaciones Select, Insert, Update, Delete (CRUD) sobre la Tabla Asignada a cada Alumno

**Máximo ??? páginas**