**Dragon Ball Legends**



**Jose Carlos Baque Meneses**

**2º DAM**

Índice

[Índice…………………………………………………………………………………………………………………………………Pág.2](#Indice)

[Descripción del programa………………………………………………………………………………………………….Pág.3](#Descripcion_Proyecto)

[Diagrama de Clases……………………………………………………………………………………………………………Pág.3](#Diagrama_de_Clases)

[Diagrama de Flujo………………………………………………………………………………………………………………Pág.4](#Diagrama_de_Flujo)

[Base de Datos…………………………………………………………………………………………………………………….Pág.4](#Base_de_Datos)

[Wireframes y Mapa de Rutas…………………………………………………………………………………………….Pág.5](#Wireframes)

[(/index)………………………………………………………………………………………………………………….Pág.5](#Indice)

[(/registrar) ……………………………………………………………………………………………………………Pág.5](#registrar)

[(/login) ………………………………………………………………………………………………………………….Pág.6](#login)

[(/menu) ………………………………………………………………………………………………………………..Pág.6](#menu)

[(/banner) ………………………………………………………………………………………………………………Pág.7](#banner)

[(/personajes) ………………………………………………………………………………………………………..Pág.7](#personajes)

[(/equipo) ………………………………………………………………………………………………………………Pág.8](#equipo)

[(/evento) ………………………………………………………………………………………………………………Pág.8](#evento)

[(/nivel) ………………………………………………………………………………………………………………….Pág.9](#nivel)

[(/tienda) ……………………………………………………………………………………………………………….Pág.9](#tienda)

[(/bolsa) ……………………………………………………………………………………………………………….Pág.10](#bolsa)

[(/configuracion) ………………………………………………………………………………………………….Pág.10](#configuracion)

[(/modificacion) ……………………………………………………………………………………………………Pág.11](#modificacion)

[Mapa de Rutas…………………………………………………………………………………………………….Pág.11](#Mapa_de_Rutas)

[Capas……………………………………………………………………………………………………………………………….Pág.12](#Capas)

[Dominio……………………………………………………………………………………………………………….Pág.12](#Dominio)

[Data…………………………………………………………………………………………………………………….Pág.12](#Data)

[Negocio……………………………………………………………………………………………………………….Pág.12](#Negocio)

[Servlets……………………………………………………………………………………………………………….Pág.13](#servlets)

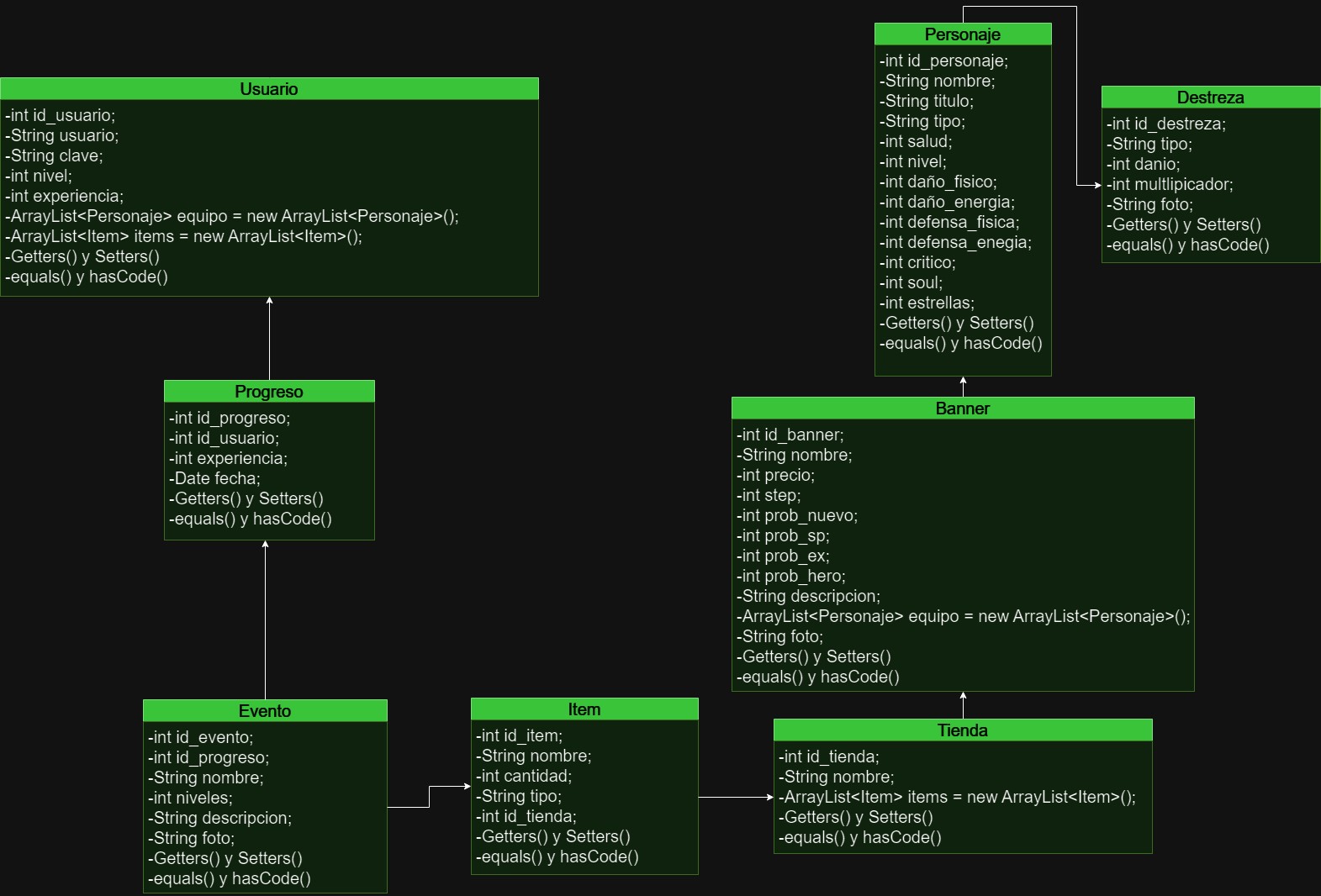
[Jsp´s…………………………………………………………………………………………………………………….Pág.13](#jsps)

[JSP´S…………………………………………………………………………………………………………………………………Pág.14](#jsps)

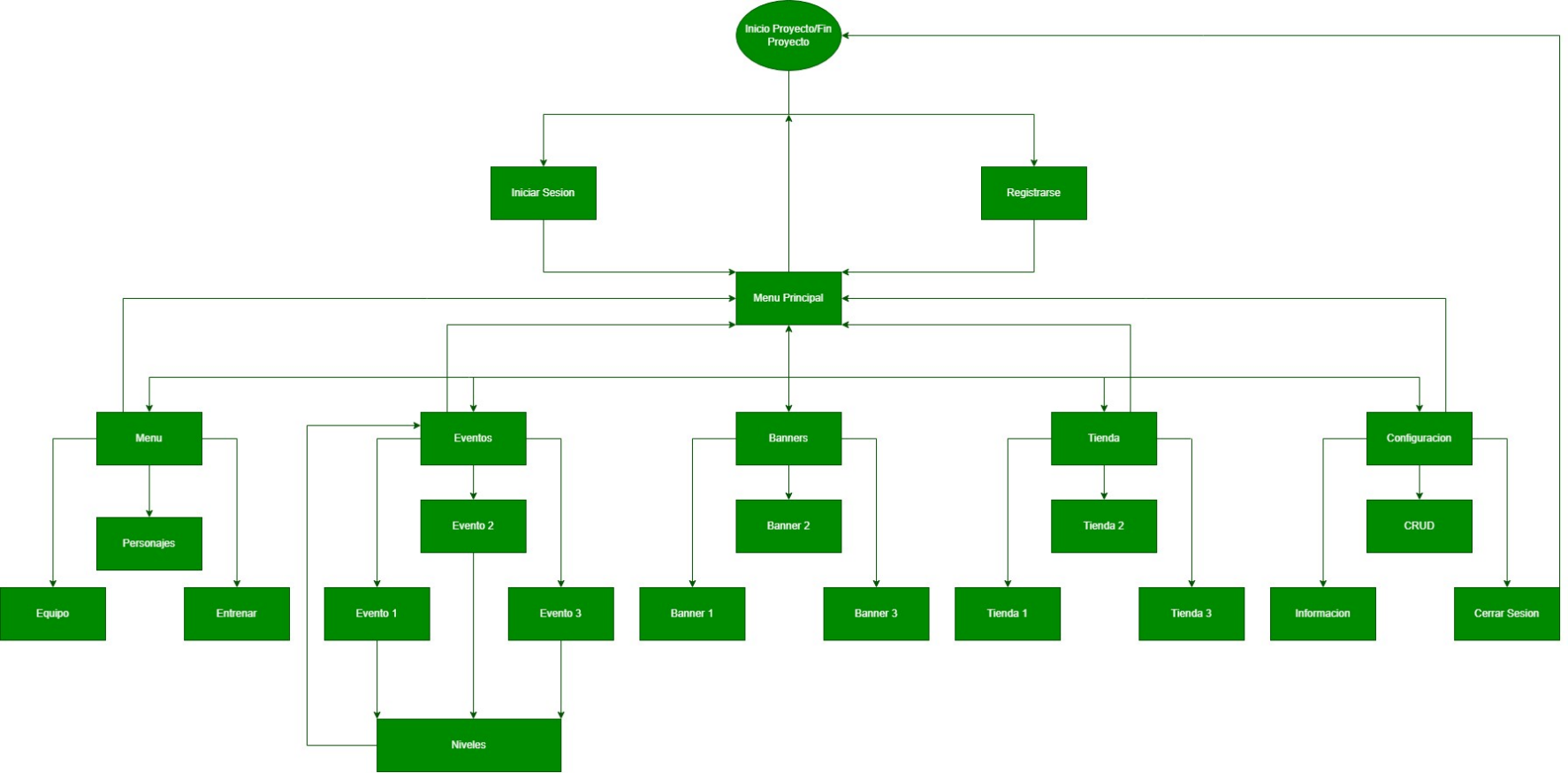
[Descripción del Proyecto](#Indice)

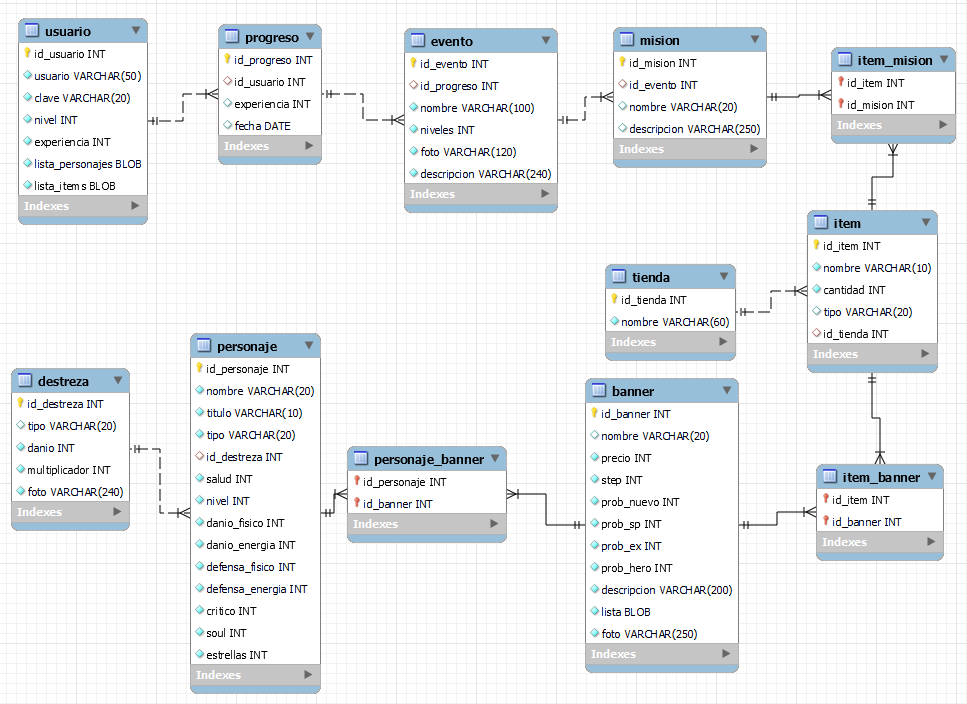
Dragon Ball Legends es un videojuego basado en un sistema Gacha, un sistema que se basa en la obtención de un objeto tipo moneda para intercambiar por la probabilidad de la obtención de un personaje, también tiene niveles en los que enfrentas contra la IA para la obtención de varios objetos como puede ser para subir personajes de nivel, para comprar dichos objetos de entrenamiento y subida de nivel y por supuesto para tirar por personajes. Es un juego perfecto para pasar el rato en un largo viaje.

[Diagrama de Clases](#Indice)

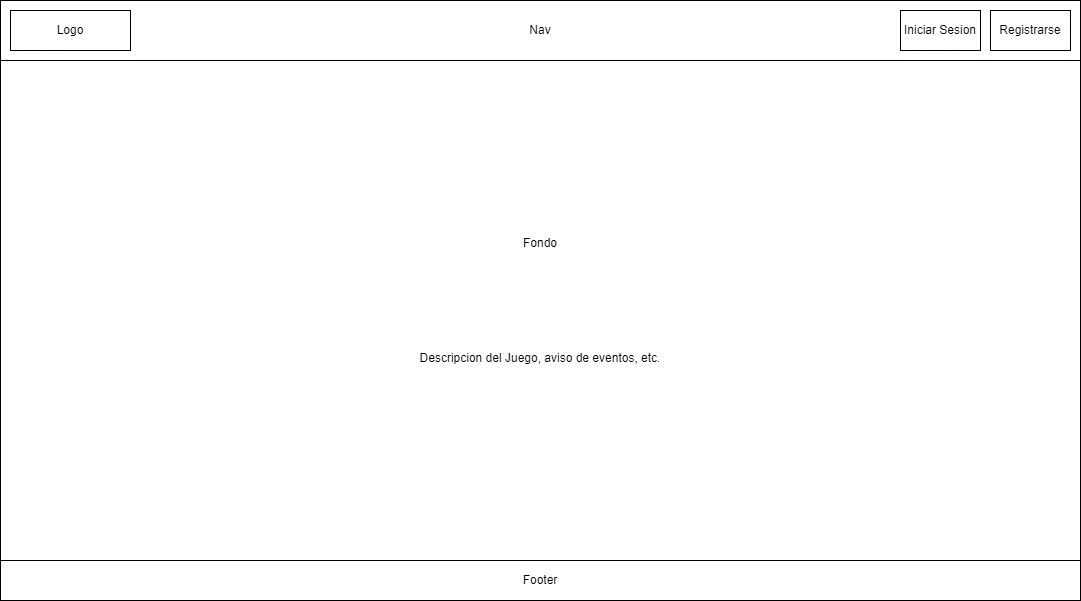


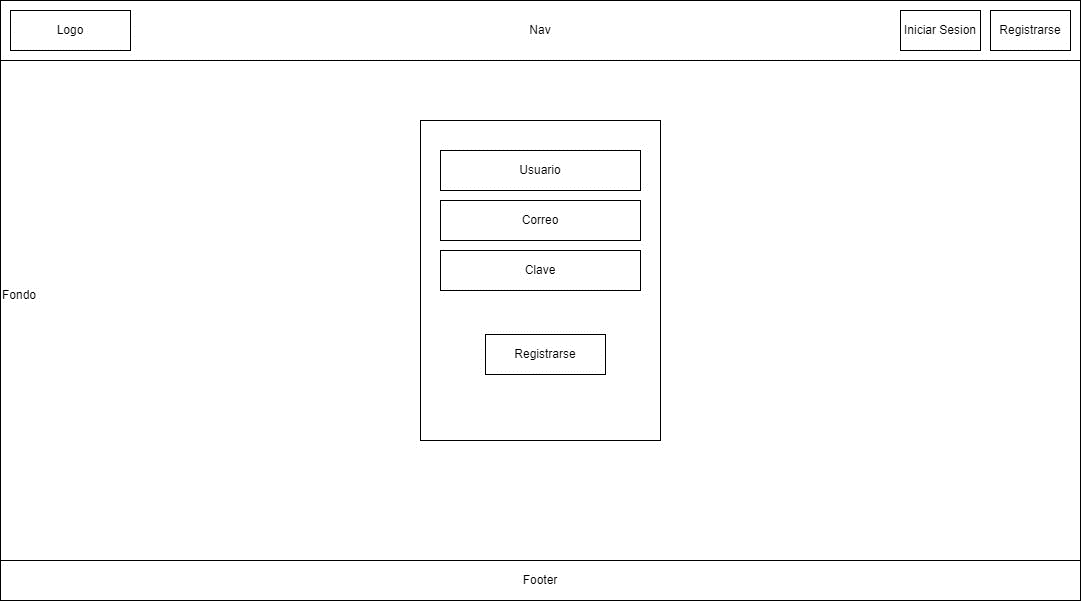
[Diagrama de Flujo](#Indice)

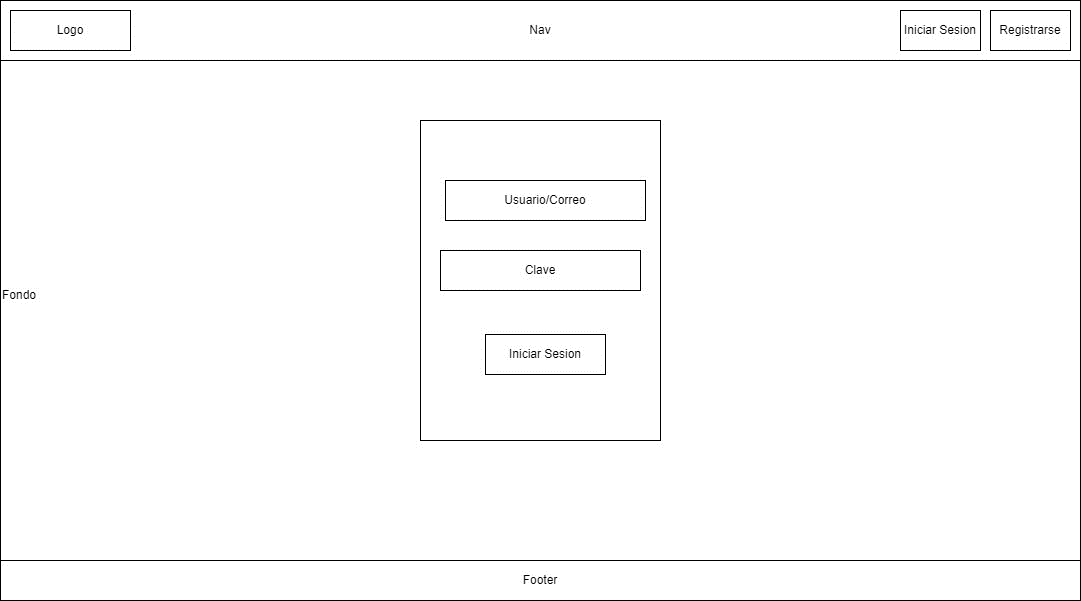


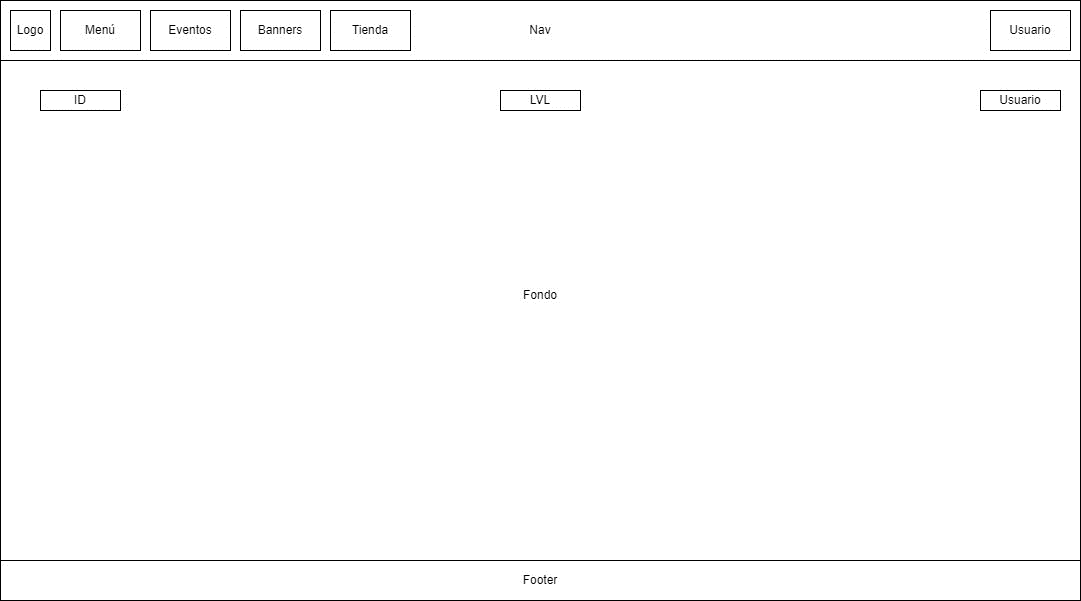
[Base de Datos](#Indice)

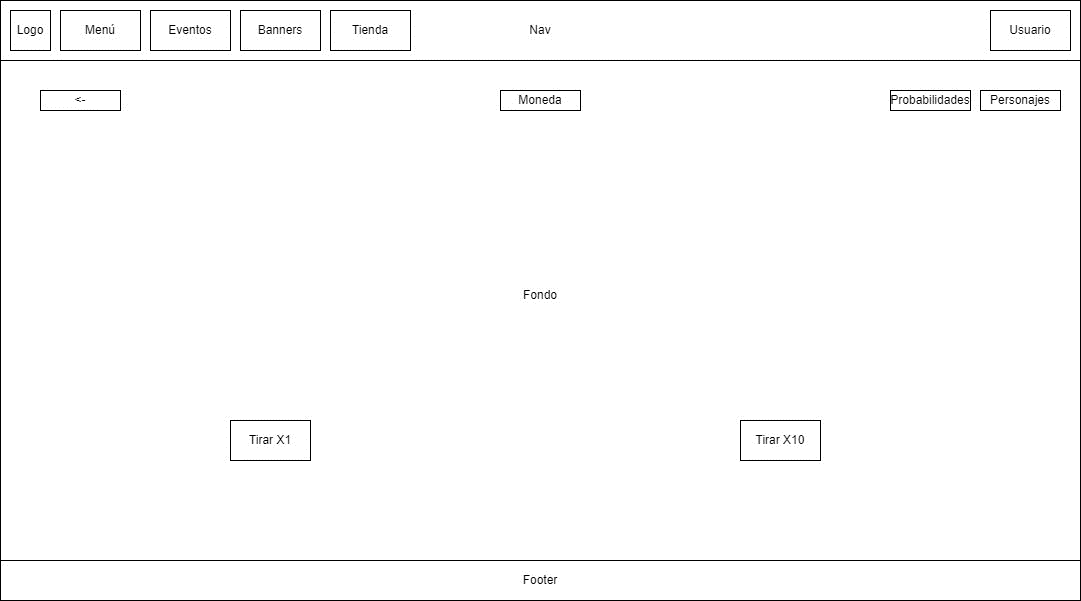
[Wireframes y Mapa de Rutas](#Indice)

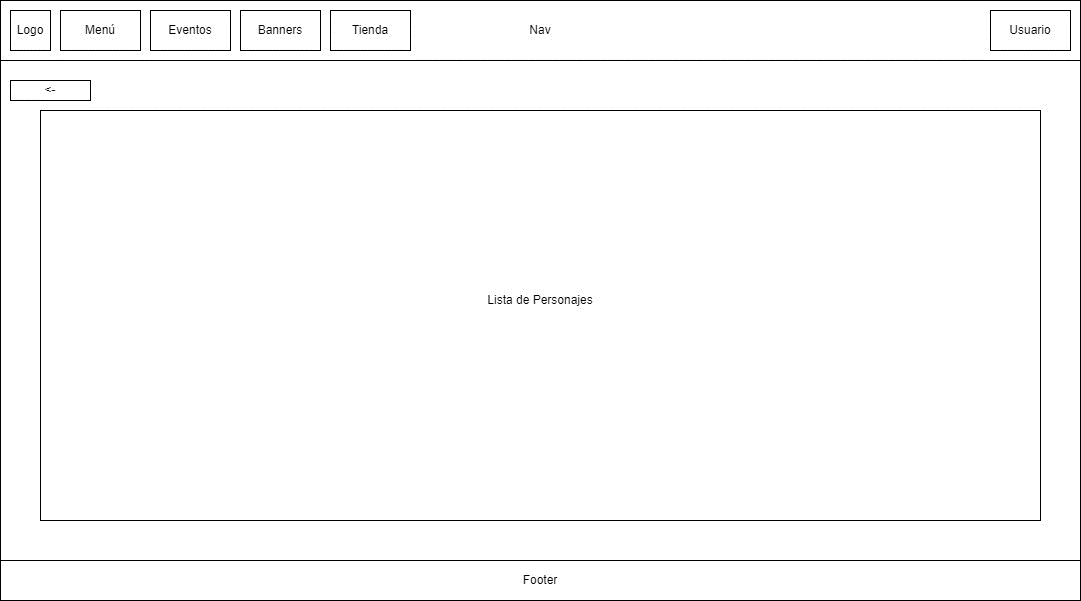
(/index):

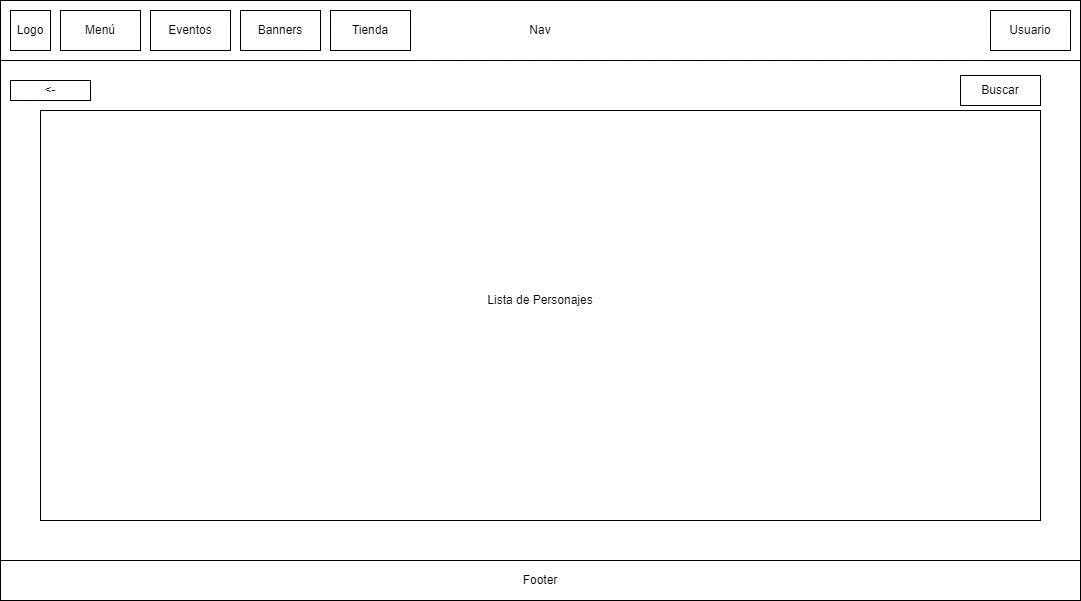
(/registrar):

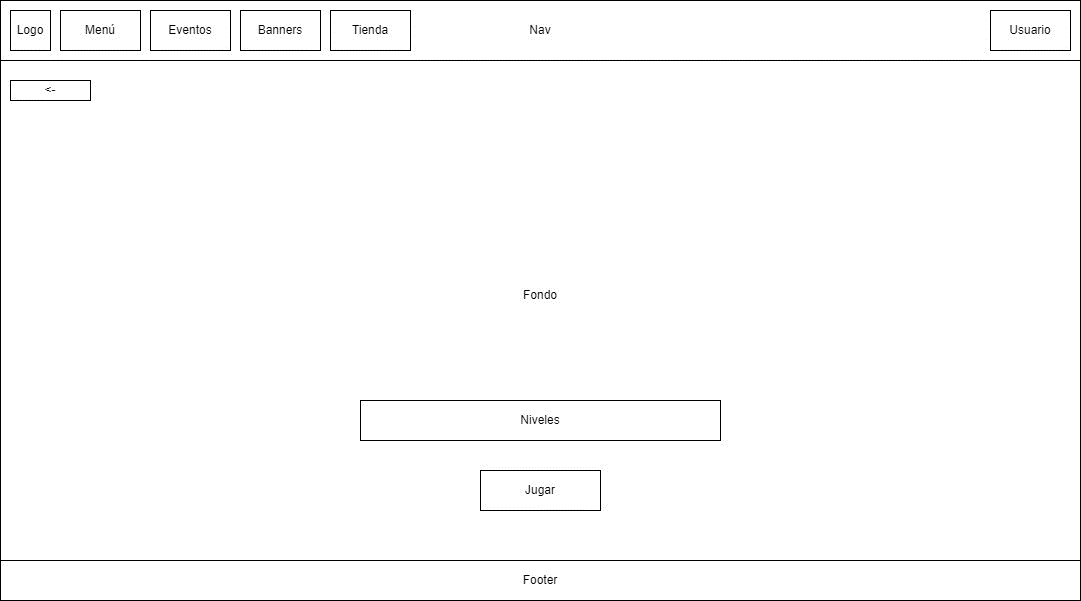
(/login):

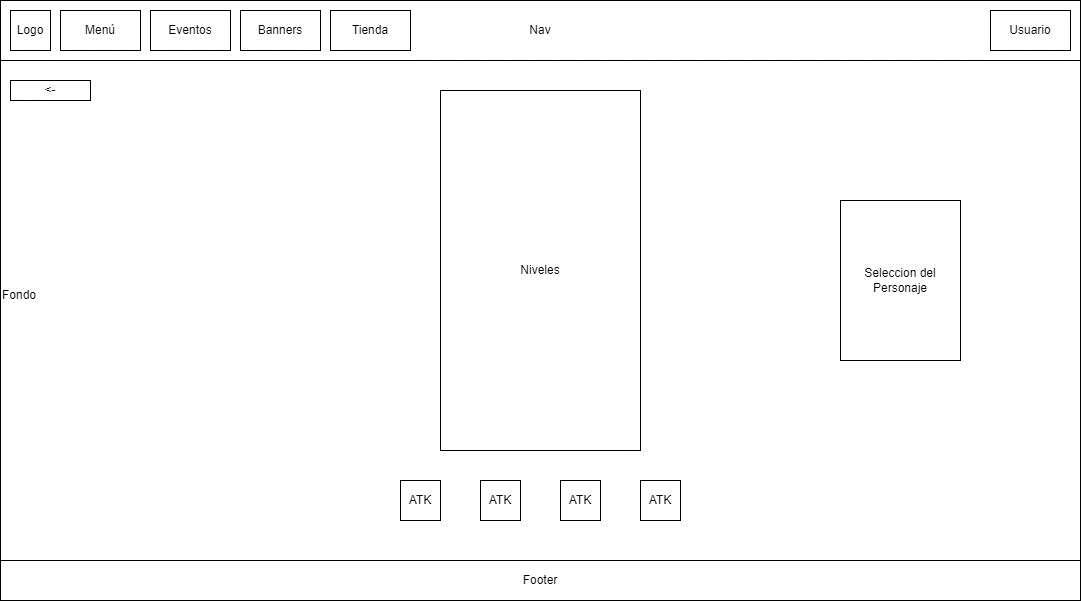
(/menu):

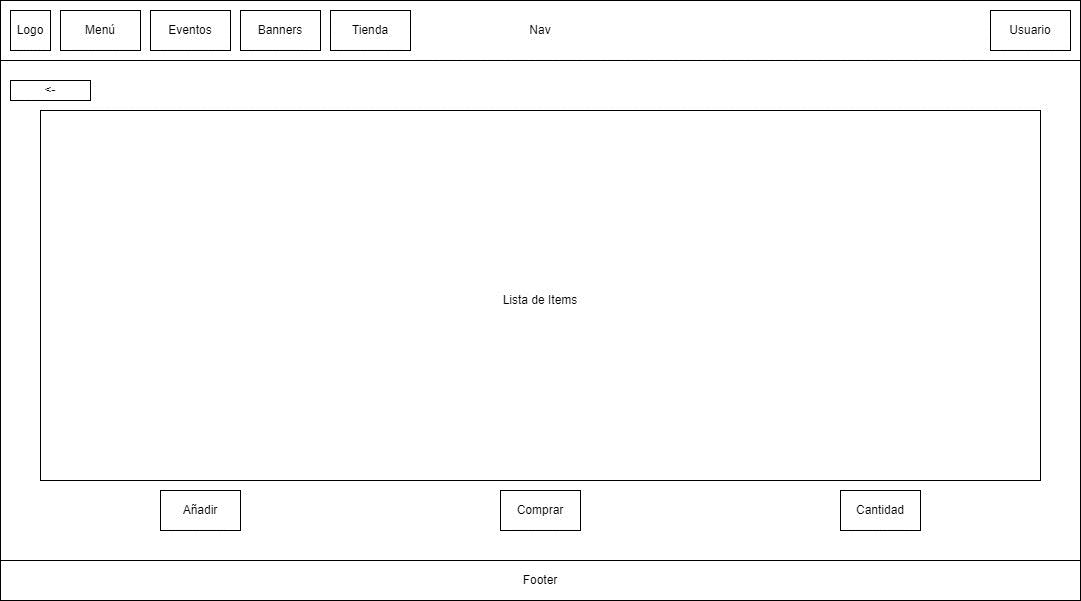
(/banner):

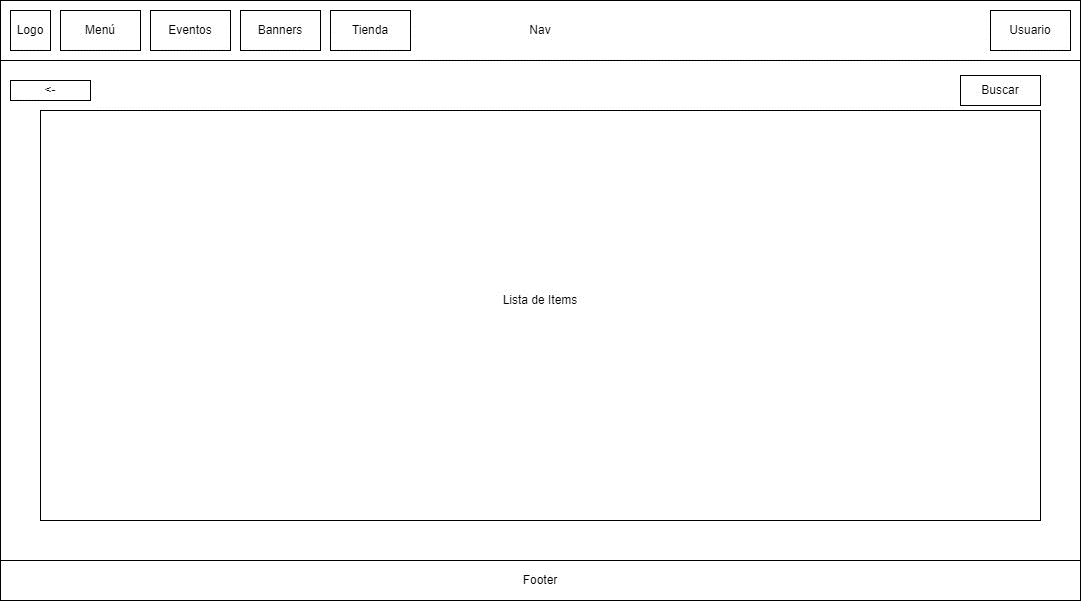
(/personajes):

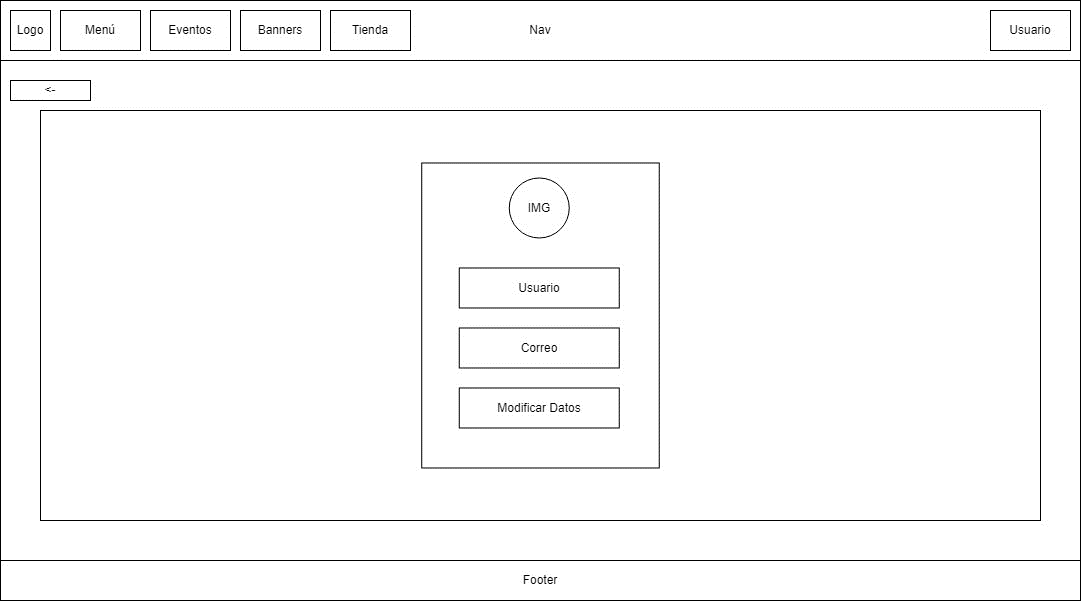
(/equipo):

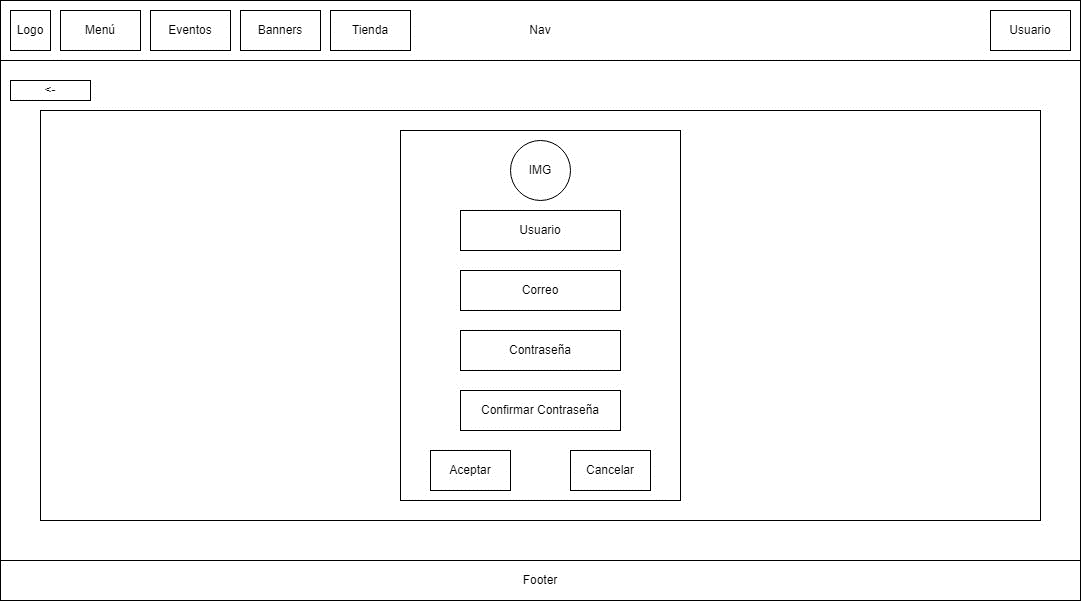
(/evento):

(/nivel):

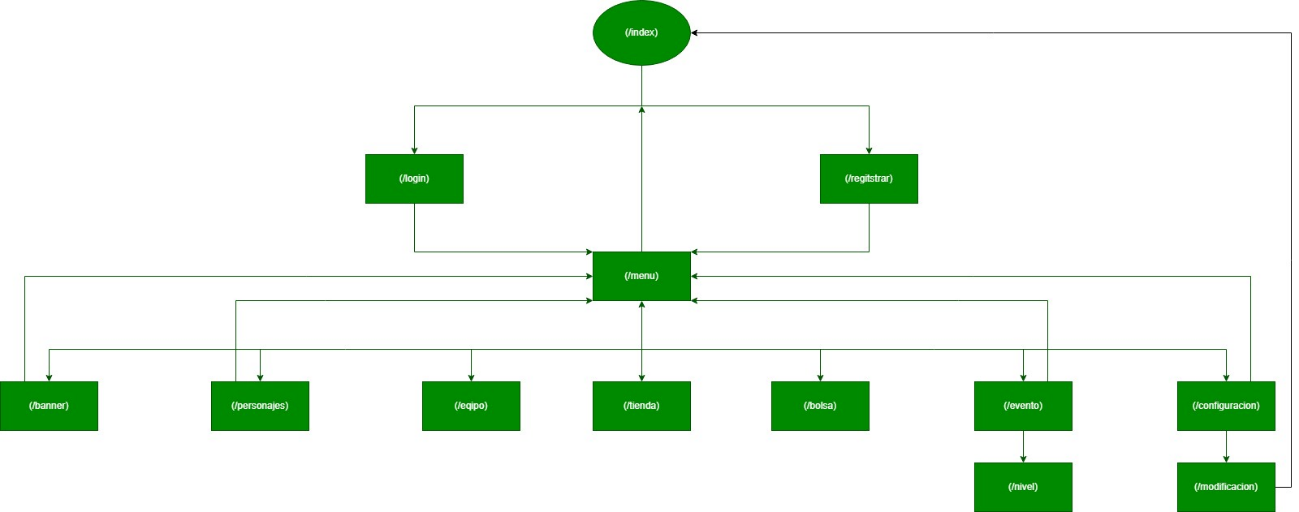
(/tienda):

(/bolsa):

(/configuracion):

(/modificacion):

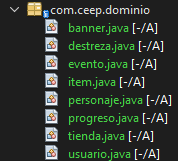
Mapa de Rutas:



[Capas](#Indice)

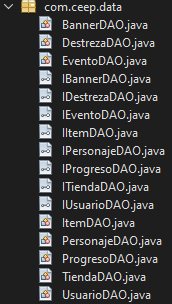
Dominio:

Clases de los objetos del programa



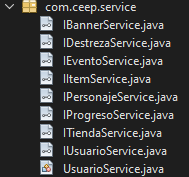
Data:

Clases que trabajan con la base de datos del programa



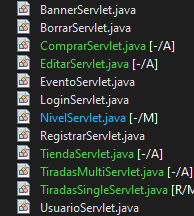
Negocio:

Clase de programa que trabajan con el programa



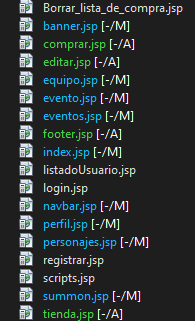
Servlets:

DoGet y DoPost



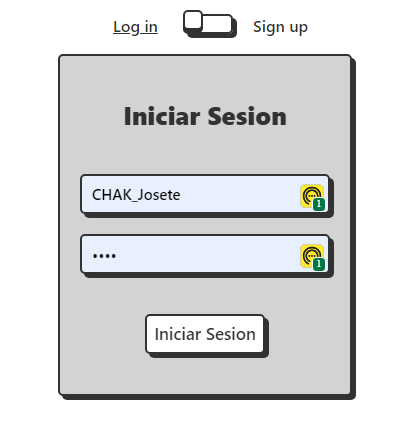
Jsp´S:

Visuales



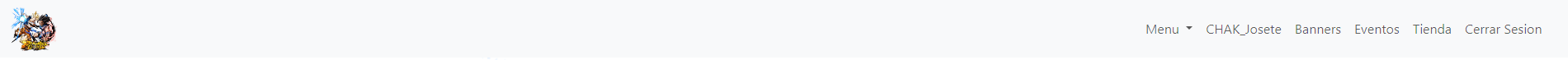
[JSP´S](#Indice)







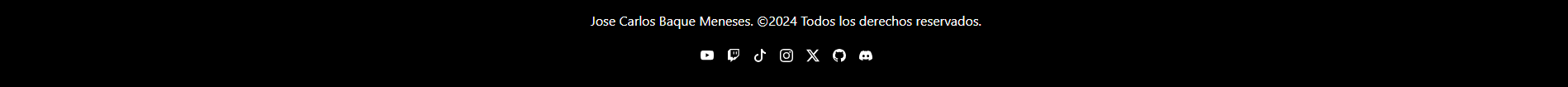
NAVBAR:







Footer:















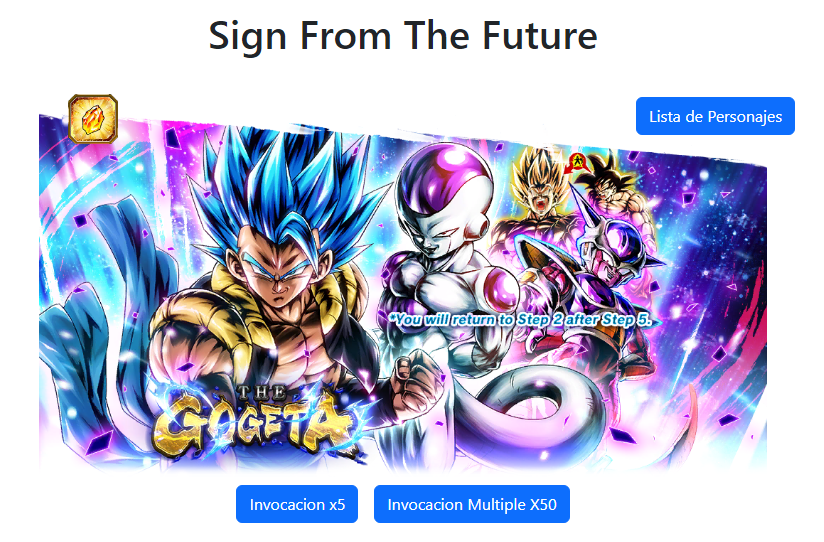












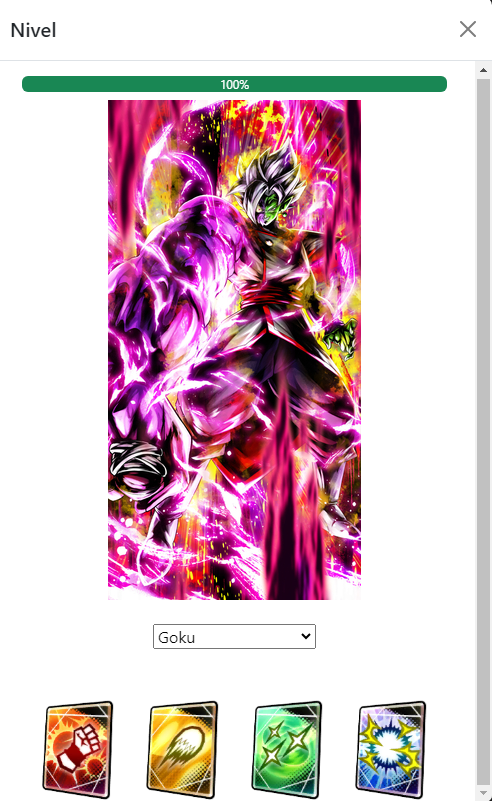








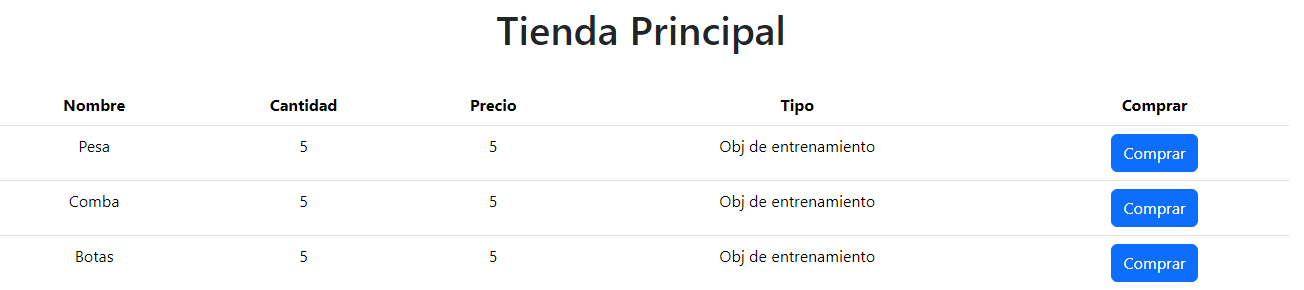
Nivel:











Métodos

deserializar\_pjs:

1. **Función de deserialización**: Utiliza la clase **ObjectInputStream** para deserializar los datos contenidos en el arreglo de bytes. La deserialización es el proceso de reconstruir un objeto a partir de su representación en forma de bytes.
2. **ByteArrayInputStream**: Se utiliza para leer los datos del arreglo de bytes en la memoria. El **ObjectInput** se conecta a este **ByteArrayInputStream** para leer los datos serializados de él.
3. **Conversión a lista de objetos**: Después de que la deserialización se completa, el método convierte los datos deserializados en una lista de objetos **personaje** mediante la llamada al método **readObject()** de **ObjectInput**.

serializar\_pjs:

1. **Función de serialización**: Utiliza la clase **ObjectOutputStream** para serializar la lista de **personajes**. La serialización es el proceso de convertir un objeto en una secuencia de bytes que pueden ser almacenados o transmitidos a través de la red y luego reconstruidos para recrear una copia exacta del objeto original.
2. **ByteArrayOutputStream**: Se utiliza para escribir datos en un arreglo de bytes en la memoria. El **ObjectOutput** se conecta a este **ByteArrayOutputStream** para escribir los datos serializados en él.
3. **Conversión a arreglo de bytes**: Después de que la serialización se completa, el método convierte los datos serializados en un arreglo de bytes utilizando el método **toByteArray()** de **ByteArrayOutputStream**.

deserializar\_items:

1. **Función de deserialización**: Utiliza la clase **ObjectInputStream** para deserializar los datos contenidos en el arreglo de bytes. La deserialización es el proceso de reconstruir un objeto a partir de su representación en forma de bytes.
2. **ByteArrayInputStream**: Se utiliza para leer los datos del arreglo de bytes en la memoria. El **ObjectInput** se conecta a este **ByteArrayInputStream** para leer los datos serializados de él.
3. **Conversión a lista de objetos**: Después de que la deserialización se completa, el método convierte los datos deserializados en una lista de objetos **item** mediante la llamada al método **readObject()** de **ObjectInput**.

serializar\_items:

1. **Función de serialización**: Utiliza la clase **ObjectOutputStream** para serializar la lista de **items**. La serialización es el proceso de convertir un objeto en una secuencia de bytes que pueden ser almacenados o transmitidos a través de la red y luego reconstruidos para recrear una copia exacta del objeto original.
2. **ByteArrayOutputStream**: Se utiliza para escribir datos en un arreglo de bytes en la memoria. El **ObjectOutput** se conecta a este **ByteArrayOutputStream** para escribir los datos serializados en él.
3. **Conversión a arreglo de bytes**: Después de que la serialización se completa, el método convierte los datos serializados en un arreglo de bytes utilizando el método **toByteArray()** de **ByteArrayOutputStream**.