



PROYECTO 1 DEWC



07 DE DICIEMBRE DE 2025
AKETZA GONZALEZ, ANDONI MONTERO Y LIERNI SARRAOA

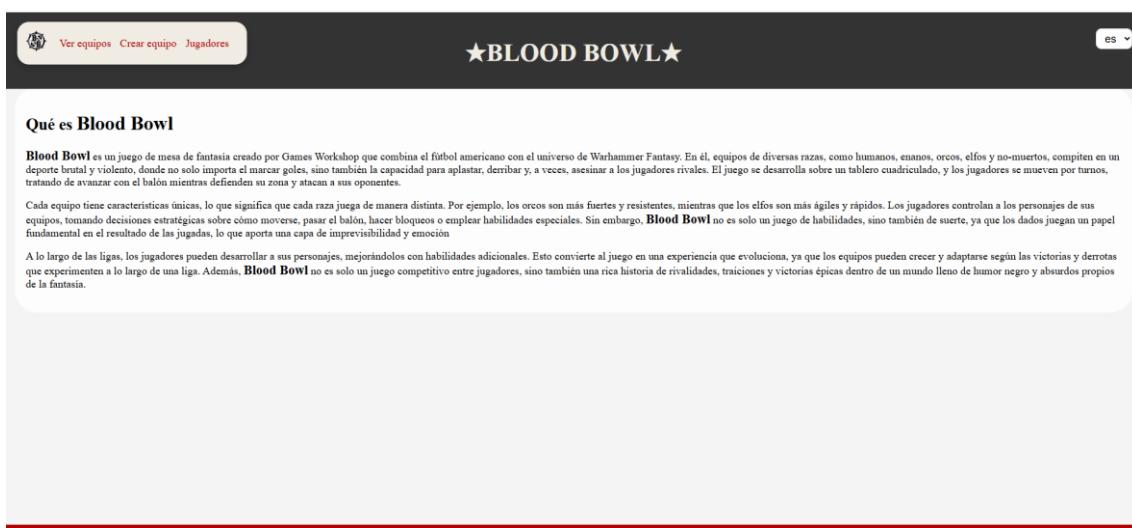
INDICE

FUNCIONAMIENTO DE LA APLICACIÓN WEB	2
COMUNICACIÓN CON LA API	10
GET	11
POST	16
PUT	18
PATCH	20
DELETE	22
FUNCIONALIDADES EXTRA	24
MULTIDIOMA.....	24
FUNCIONALIDAD EN LA INTERFAZ DE USUARIO (UI).....	25
GENERAR PDF.....	26
FUNCIONALIDAD EN LA INTERFAZ DE USUARIO (UI).....	38

FUNCIONAMIENTO DE LA APLICACIÓN WEB

El tema que hemos elegido a sido crear una página para jugar a **Blood Bowl**. **Blood Bowl** es un juego de fantasía creado por **Games Workshop** que combina el **fútbol americano** con el universo de **Warhammer Fantasy**. En él, equipos de diversas razas, como **humanos, enanos, orcos, elfos y no-muertos**, compiten en un **deporte brutal y violento**, donde no solo importa el marcar goles, si no también la capacidad para **aplastar, derribar** y, a veces, **asesinar** a los jugadores rivales. El juego se desarrolla sobre un **tablero cuadriculado**, y los jugadores se mueven por **turno**, tratando de avanzar con el balón mientras **defiendes** su zona y **atacan** a sus oponentes.

Ahora que sabemos el tema podemos ver el funcionamiento de la aplicación web. En nuestro caso la **página principal** consta de un texto **informativo** sobre el juego **Blood Bowl**. En la **parte de arriba** nos encontramos un **ícono** a la parte **derecha** y en la parte **izquierda** nos encontramos con **un seleccionador**. Si el **ratón** se pone **encima del ícono** aparecerá un menú con **3 opciones, ver equipos, crear equipo y jugadores**. El **seleccionador de la izquierda** es para seleccionar el idioma preferido entre las **3 opciones** disponibles que son **el español, el euskera y el inglés**.



Cuando se selecciona en el menú **ver equipos** podemos ver **2 secciones**, una sección es donde **aparecen los jugadores del equipo seleccionado** y donde se puede **crear tu propio equipo**, la otra sección es para poder **cambiar el valor del reroll del equipo seleccionado**.

La primera sección, como antes he dicho, va sobre los jugadores del equipo seleccionado, para ello primero tenemos **un seleccionador** donde aparecen **las opciones de los equipos disponibles**. Cuando se selecciona **un equipo** se actualiza automáticamente la **tabla** donde aparecen los jugadores, en **la tabla de los jugadores** aparecen los siguientes **datos** de cada jugador: **cantidad, alineación, tags, precio, mv, fu, ag, ps, ar, habilidades (en una lista), pri y sec.** Debajo de la tabla de los jugadores aparecen **más datos** sobre el equipo como el **precio del reroll**, si es **apotecario** o no y las **normas especiales**, aparte de ello aparece un **botón** con el texto **CREA TU PROPIO EQUIPO** que clicando en el se dirigirá a la página para **crear un equipo**.

The screenshot displays two separate instances of the 'Creador de Equipos' application window. Both windows have a dark header bar with the title '★CREADOR DE EQUIPOS★' and a language switcher 'es'. The top window shows a table titled 'Humans' with columns for CANTIDAD, ALINEACION, TAGS, PRECIO, MV, FU, AG, PS, AR, HABILIDADES, PRI, and SEC. It lists various human units like Lineman, Hopeful, Catcher, Thrower, Blitzer, and Ogre, each with specific stats and abilities. The bottom window shows a similar table for 'Right Stuff' units, also listing Human Catcher, Thrower, Blitzer, and Ogre units. At the bottom of both windows, there are status messages: 'Poyesto DWEC - 2025', '18:35 07/12/2025', and a button labeled 'Crea tu propio equipo'.

Humans											
CANTIDAD	ALINEACION	TAGS	PRECIO	MV	FU	AG	PS	AR	HABILIDADES	PRI	SEC
0-16	Human Lineman	-Human -Lineman	50k	6	3	3+	4+	9+	-	G	A,D,S
0-3	Halfling Hopeful	-Halfling -Lineman	30k	5	2	3+	4+	7+	-Stunty -Dodge -Right Stuff	A	D,G,S
0-2	Human Catcher	-Human -Catcher	75k	8	3	3+	4+	8+	-Catch -Dodge	A,G	P,S,D
0-2	Human Thrower	-Human -Thrower	75k	6	3	3+	3+	9+	-Sure Hands -Pass	G,P	A,D,S
0-2	Human Blitzer	-Human -Blitzer	85k	7	3	3+	4+	9+	-Block -Tackle	G,S	A,D
0-1	Ogre	-Ogre -Big guy	140k	5	5	4+	5+	10+	-Bone Head -Loner (3+) -Mighty Blow -Thick Skull -Throw Team-mate	S	A,G,M

Right Stuff											
CANTIDAD	ALINEACION	TAGS	PRECIO	MV	FU	AG	PS	AR	HABILIDADES	PRI	SEC
0-2	Human Catcher	-Human -Catcher	75k	8	3	3+	4+	8+	-Catch -Dodge	A,G	P,S,D
0-2	Human Thrower	-Human -Thrower	75k	6	3	3+	3+	9+	-Sure Hands -Pass	G,P	A,D,S
0-2	Human Blitzer	-Human -Blitzer	85k	7	3	3+	4+	9+	-Block -Tackle	G,S	A,D
0-1	Ogre	-Ogre -Big guy	140k	5	5	4+	5+	10+	-Bone Head -Loner (3+) -Mighty Blow -Thick Skull -Throw Team-mate	S	A,G,M

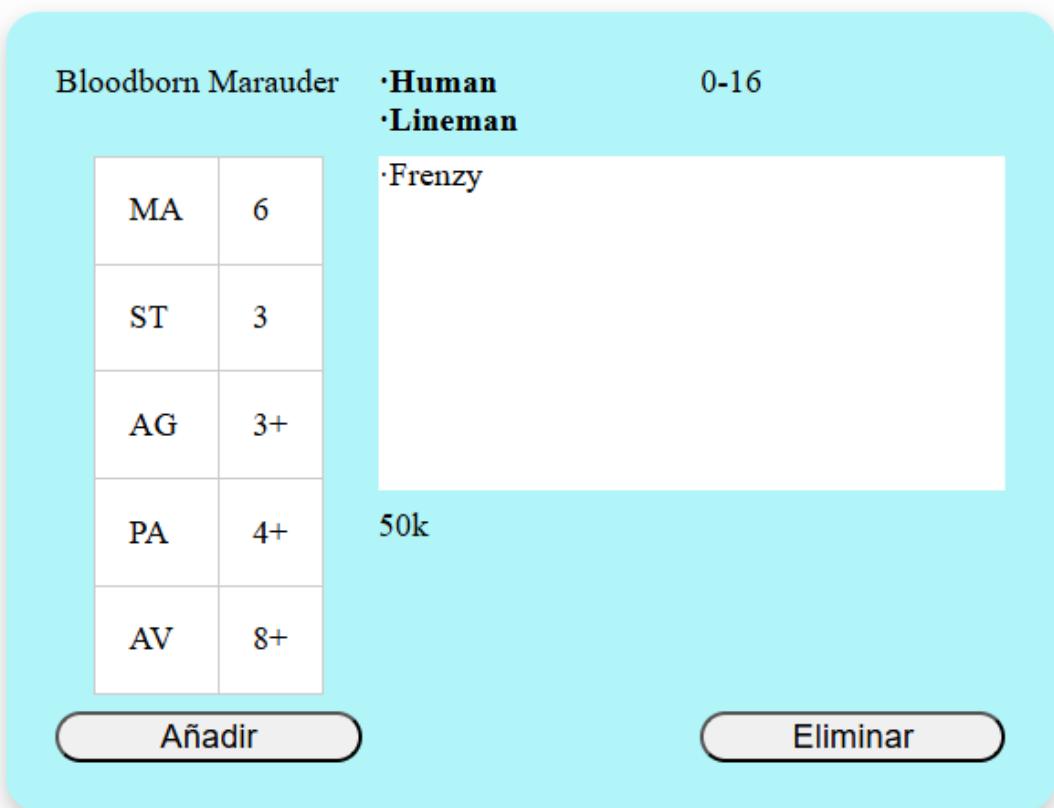
Reroll: 0-8 50k
Apotecario: si
Normas Especiales: Team Captain
[Crea tu propio equipo](#)

La segunda sección, consta de un **seleccionador** para poder seleccionar el equipo que se quiere **actualizar**, debajo de ello hay un campo para insertar el **nuevo valor**, cuando se quiera actualizar se clica en el botón con el texto **ACTUALIZAR** y se **actualizará**.



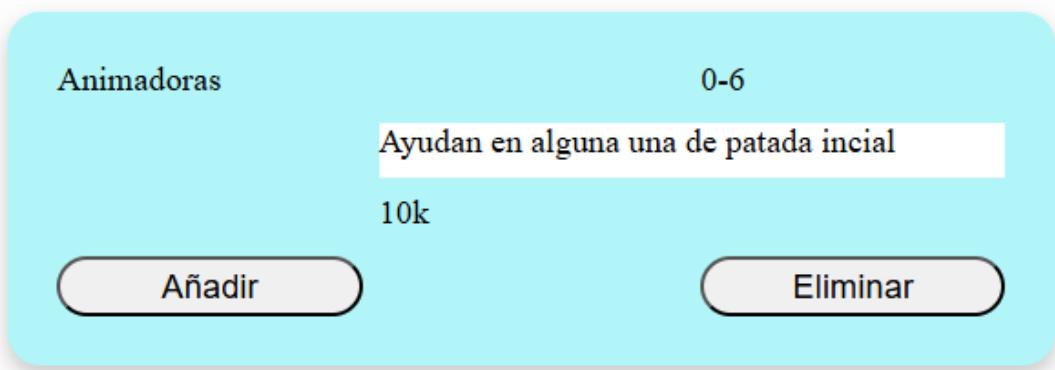
Como antes he dicho hay un botón donde se puede **crear tu propio equipo**, para acceder a esta página se puede **de 2 maneras** una desde el **botón de la página anterior** y la otra desde el **menú** con el texto **CREAR EQUIPO**.

Esta página consta de un campo para insertar **el nombre del equipo**. Sin este campo **NO** se podrá crear el equipo. Después del campo hay un **seleccionador** donde las opciones que aparecen son **los equipos disponibles**, al seleccionar un equipo aparecen en **forma de tarjeta** los **jugadores y extras** disponibles. **Las tarjetas de los jugadores** tienen el siguiente formato.



En esta tarjeta aparece el **nombre del jugador**, la **posición**, la **cantidad** de jugadores que se puede elegir, **una tabla** donde aparecen los siguientes datos: **ma, st, ag, pa y av**. Y, por último, aparecen tanto las **habilidades** como el **precio** del jugador. Para poder añadir este jugador se clica en el **botón con el texto AÑADIR**, y para eliminar el jugador se clica el **botón con el texto ELIMINAR**.

Las tarjetas de los extras tienen el siguiente formato.



En estas tarjetas aparece **el título del extra**, una **descripción**, la **cantidad** posible y el **precio**. Para añadir los extras se clica en **el botón con el texto AÑADIR**, y para eliminar el extra se clica en **el botón con el texto ELIMINAR**.

Para poder crear el equipo hay un **botón con el texto VALIDAR** en la parte de **debajo de la tabla** donde aparecen los **jugadores y extras**. Antes de darle al botón de validar, hay que tener en cuenta que **NO** se va a poder crear el equipo **si la cantidad de jugadores es menor a 11**, y el **nombre del equipo se encuentra vacío**. Al crear el equipo se crea **un archivo PDF**, con la información de cada **jugador y extra añadido** al equipo creado. El nombre del archivo es el nombre del equipo creado.

The screenshot shows the '★CREADOR DE EQUIPOS★' application. It has a sidebar with categories: Bloodspawn, Animadoras, Rolll, Fan Factor, Ayudantes de entrenador, and Apotecario. Each category has a card with details like stats, descriptions, and 'Añadir' or 'Eliminar' buttons. At the bottom is a 'Validar' button.

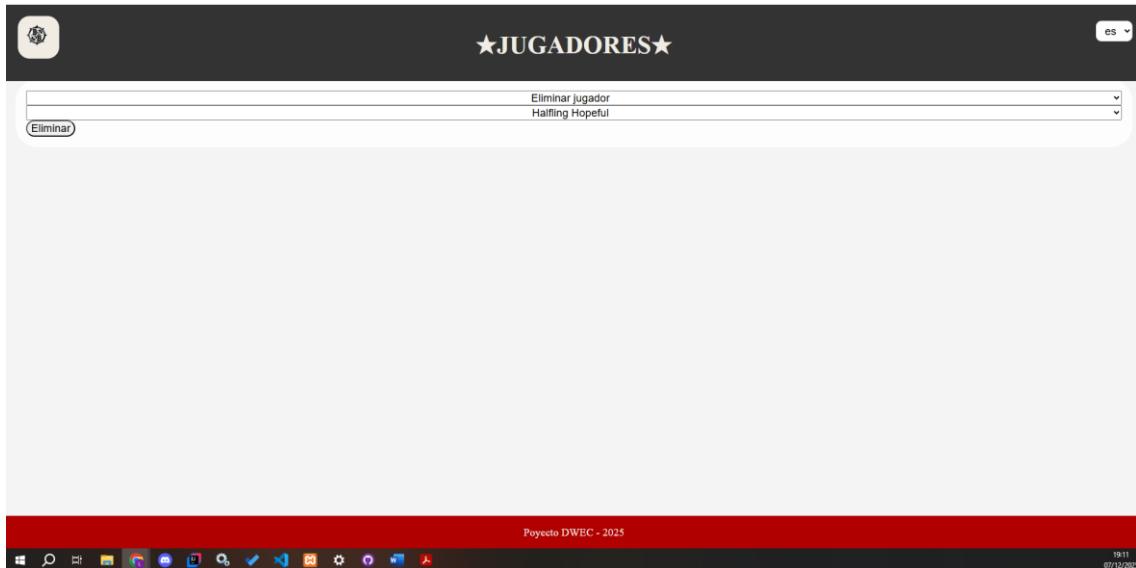
The screenshot shows a PDF titled 'Poyecto DEWC - 2025' with a table for 'Equipo: Prueba'. The table lists 11 players with their stats and abilities. Below the table is a section for 'Extras' with two entries: 'Ayudantes de entrenador' and 'Animadoras'. A sidebar on the right contains an AI assistance feature.

La última opción de esta página es **crear, modificar y eliminar jugadores**. En esta página hay un seleccionador con 3 opciones, una **crear jugador**, la segunda **modificar jugador** y la última **eliminar jugador**.

Al seleccionar **crear jugador**, aparece un **formulario** para poder insertar todos los valores a los siguientes datos: **alineación, tags, cantidad, precio, mv, fu, ag, ps, ar, habilidades, pri y sec**. Después se podrá guardar el jugador clicando en el botón con el texto **GUARDAR**.

Al seleccionar **modificar jugador**, automáticamente aparece un **seleccionador** para poder ver los **jugadores disponibles para modificarlos**, al seleccionar un jugador, **aparecen los valores de cada campo**. Para poder actualizarlo se cambia el valor que se quiera cambiar y se clica en el botón con el texto **GUARDAR**.

Al seleccionar **eliminar jugador**, aparece automáticamente un seleccionador donde se puede **elegir el jugador a eliminar**, para poder eliminar el jugador seleccionado se clica en **el botón con el texto ELIMINAR.**



COMUNICACIÓN CON LA API

En nuestro proyecto hemos utilizado **un servidor json** para poder publicar nuestra **db.json** en un entorno local, el entorno que hemos utilizado es **Node.js**, este permite usar **JavaScript** en el **servidor**. Está basado en el motor **V8 de Chrome**, funciona de manera **asíncrona**, y se utiliza para crear **aplicaciones web, APIs y servicios en tiempo real**.

Para poder comunicarnos entre el servidor y la aplicación web vamos a utilizar **la función async**. **Fetch** puede resultar **difícil de leer** por su naturaleza **asíncrona**, por eso, **async** surgió con el objetivo de mejorar la **gestión de funciones asíncronas**, intentando hacer el **código más legible y menos propenso a errores**.

Los métodos que hemos utilizado son **GET, POST, PUT, PATCH Y DELETE**, a continuación, se explicará cada método con ejemplos de la aplicación.



GET

Esta petición se usa para **pedir datos** al servidor **sin modificar nada**, se utiliza cuando quieras **consultar información**, por ejemplo, cuando quieras ver todos los usuarios o un producto concreto.

Ejemplos de la aplicación: Para **obtener los equipos**.

```
//funcion que realiza una petición get de los equipos y devuelve
un array de objetos con campos
//(id (string),nombre (string), normasEspeciales (string[]),
reroll (int), apotecario (boolean))
export async function getEquipos() {
    try{
        const response = await
fetch("http://localhost:3000/equipos"); //Se crea la petición
        if(!response.ok) throw new Error ("GET erróneo");
        const equipos = await response.json();
        //Retorna el array de equipos
        return equipos
    }catch(error){
        console.log(error);
    }
}
```

Devolvería los siguientes datos en **JSON**.

```
[
  {
    "id": "1",
    "nombre": "Humans",
    "normasEspeciales": ["Team Captain"],
    "reroll": 50,
    "apotecario": true
  },
  {
    "id": "2",
    "nombre": "Orcs",
    "normasEspeciales": ["Team Captain", "Brawlin Brutes"],
    "reroll": 60,
    "apotecario": true
  }
]
```

Ejemplos de la aplicación: Para obtener los jugadores.

```
//funcion que realiza una petición get de los jugadores y
devuelve un array de objetos con campos
//(id (string), posicion (string), tags (string[]), cantidad
(int), coste (int), MA (int), FU (int), AG (int),
//PA (int), AR (int), Habilidades (string[]), Pri (string[]), Sec
(string[]), Equipos (string[]))
export async function getJugadores() {
    try{
        const response = await
fetch("http://localhost:3000/jugadores");//Se crea la petición
        if(!response.ok) throw new Error ("GET erróneo");
        const jugadores = await response.json();
        //Retorna un array de jugadores
        return jugadores
    }catch(error){
        console.log(error);
    }
}
```

Devolvería los siguientes datos en **JSON**.

```
[
  {
    "id": "1",
    "posicion": "Lineman",
    "tags": ["Human", "Lineman"],
    "cantidad": 16,
    "coste": 50,
    "MA": 6,
    "FU": 3,
    "AG": 3,
    "PA": 4,
    "AR": 9,
    "Habilidades": [],
    "Pri": ["G"],
    "Sec": ["A", "D", "S"],
    "Equipos": ["1"]
  }, ... (Todos los registros que aparezcan)
]
```

Ejemplos de la aplicación: Para obtener los jugadores de un equipo.

```
//funcion que realiza una petición get de los jugadores de un
equipo y devuelve un array de objetos con campos
//(id (string), posicion (string),tags (string[]),cantidad
(int),coste (int),MA (int),FU (int),AG (int),
//PA (int),AR (int),Habilidades (string[]),Pri (string[]),Sec
(string[]),Equipos (string[])
export async function getJugadoresEquipo(idEquipo) {
    try{
        const response = await
fetch("http://localhost:3000/jugadores");
        if(!response.ok) throw new Error ("GET erróneo");
        const jugadores = await response.json();
        let arrDevolver=[];
        jugadores.forEach(jugador => {
            if(jugador.Equipos.includes(idEquipo+"")){
                arrDevolver.push(jugador);
            }
        });
        return arrDevolver;
    }catch(error){
        console.log(error);
    }
}
```

Devolvería los siguientes datos en **JSON**.

```
[
  {
    "id": "1",
    "posicion": "Lineman",
    "tags": ["Human", "Lineman"],
    "cantidad": 16,
    "coste": 50,
    "MA": 6,
    "FU": 3,
    "AG": 3,
    "PA": 4,
    "AR": 9,
    "Habilidades": [],
    "Pri": ["G"],
    "Sec": ["A", "D", "S"],
    "Equipos": ["1"]
  },... (Todos los registros que aparezcan del equipo
seleccionado)
]
```

Ejemplos de la aplicación: Para obtener un solo jugador.

```
//funcion que realiza una petición get de un jugador y devuelve
un objeto jugador con los campos
//(id (string), posicion (string),tags (string[]),cantidad
(int),coste (int),MA (int),FU (int),AG (int),
//PA (int),AR (int),Habilidades (string[]),Pri (string[]),Sec
(string[]),Equipos (string[]))
export async function getJugador(id) {
    try{
        const response = await
fetch("http://localhost:3000/jugadores/"+id);
        if(!response.ok) throw new Error ("GET erróneo");
        const jugador = await response.json();
        return jugador
    }catch(error){
        console.log(error);
    }
}
```

Devolvería los siguientes datos en **JSON**.

```
[
  {
    "id": "1",
    "posicion": "Lineman",
    "tags": ["Human", "Lineman"],
    "cantidad": 16,
    "coste": 50,
    "MA": 6,
    "FU": 3,
    "AG": 3,
    "PA": 4,
    "AR": 9,
    "Habilidades": [],
    "Pri": ["G"],
    "Sec": ["A", "D", "S"],
    "Equipos": ["1"]
  }
]
```

Ejemplos de la aplicación: Para obtener un solo equipo.

```
//funcion que realiza una petición get de un equipo y lo
devuelve como objeto
//(id (string), nombre (string), normasEspeciales (string[]),
reroll (int), apotecario (boolean))
export async function getEquipo(id) {
    try{
        const response = await
fetch("http://localhost:3000/equipos/" + id);
        if(!response.ok) throw new Error ("GET erróneo");
        const equipos = await response.json();
        return equipos
    }catch(error){
        console.log(error);
    }
}
```

Devolvería los siguientes datos en **JSON**.

```
[
    {
        "id": "1",
        "nombre": "Humans",
        "normasEspeciales": [
            "Team Captain"
        ],
        "reroll": "20",
        "apotecario": true
    }
]
```

POST

Esta petición sirve para añadir un recurso nuevo al **archivo json**, se utiliza cuando se quiere **registrar** un objeto nuevo, ya sea un usuario, un producto, etc. Hay que tener en cuenta de que, si no se ponen todos los campos del objeto se **añadirá sin información**, es decir, si se añade un usuario sin el nombre, pero si con la edad, el nombre aparecería vacío y la edad con el dato introducido, otra cosa a tener en cuenta que no es obligatorio tener el campo id, pero si es recomendable si se quiere manipular esos datos. En la petición **POST** no hace falta añadir el id, ya que **json-server** lo generará automáticamente cuando se creen nuevas peticiones **POST**.

Ejemplo de aplicación: Para **añadir** un **nuevo jugador**.

```
// función que recibe un jugador y lo inserta en la db. Recibe
// un objeto jugador, que tiene el siguiente aspecto:
// (id (string), posicion (string),tags (string[]),cantidad
// (int),coste (int),MA (int),FU (int),AG (int),
// PA (int),AR (int),Habilidades (string[]),Pri (string[]),Sec
// (string[]),Equipos (string[]))
export async function postJugador(jugador) {
    try {
        //Se crea la petición con algunos parámetros
        //Si no se pone nada por defecto es GET
        const response = await
fetch("http://localhost:3000/jugadores", {
            method: "POST", //Poner el tipo de método POST
            headers: {
                //El contenido que se va a subir va a ser tipo json
                "Content-Type": "application/json"
            },
            //Todo lo que este en body se envía al servidor
            body: JSON.stringify(jugador)
        });
        if (!response.ok) {
            throw new Error("Error al guardar el jugador");
        }
    } catch (error) {
        console.log("Error:", error);
    }
}
```

Si todo ha ido **bien** el servidor **guarda el nuevo jugador** y responde con el **objeto creado**.

```
{  
    "id": "2",  
    "posicion": "Marta",  
    "tags": "25",  
    "cantidad": 16,  
    "coste": 30,  
    "MA": 5,  
    "FU": 2,  
    "AG": 3,  
    "PA": 4,  
    "AR": 7,  
    "Habilidades": ["Stunty", "Dodge", "Right Stuff"],  
    "Pri": ["A"],  
    "Sec": ["D", "G", "S"],  
    "Equipos": ["1"]  
}
```

PUT

Esta petición actualiza un recurso **ENTERO**, sobrescribiendo **TODOS** sus campos. **PUT** se utiliza cuando se quiere cambiar **TODOS** los datos de un registro/objeto. En este caso hay que poner si o sí **TODOS** los campos que el objeto puede llegar a tener, si tiene **15** campos en el **body** habrá que poner **15 campos**. Para saber que objeto se quiere modificar se pone **el ID en la URL**.

Ejemplo de aplicación: Para **actualizar todos** los campos **de un jugador**.

```
// función que recibe un jugador y un id para actualizar todo el
// objeto en la db.
// Recibe un identificador para saber cual es el jugador a
// actualizar
// y también recibe un objeto jugador, que tiene el siguiente
// aspecto:
// posicion (string),tags (string[]),cantidad (int),coste
// (int),MA (int),FU (int),AG (int),
// PA (int),AR (int),Habilidades (string[]),Pri (string[]),Sec
// (string[]),Equipos (string[])
export async function putJugador(id_jugador, jugador) {
    try {
        //Se crea la petición con algunos parámetros
        //Si no se pone nada por defecto es GET

        const response = await
fetch("http://localhost:3000/jugadores/" + id_jugador, {
            method: "PUT",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify(jugador)
        });
        if (!response.ok) {
            throw new Error("Error al guardar el jugador");
        }
    } catch (error) {
        console.log("Error:", error);
    }
}
```

Si todo ha ido **bien** el servido **guarda los datos del usuario** y responde con el **objeto actualizado**.

```
{  
    "id": "2",  
    "posicion": "Troll",  
    "tags": "25",  
    "cantidad": 20,  
    "coste": 300,  
    "MA": 5,  
    "FU": 2,  
    "AG": 3,  
    "PA": 4,  
    "AR": 7,  
    "Habilidades": ["Stunty", "Dodge", "Right Stuff"],  
    "Pri": ["A"],  
    "Sec": ["D", "G", "S"],  
    "Equipos": ["2"]  
}
```

PATCH

Esta petición actualiza solo **ALGUNOS** campos de un **recurso/objeto**.

PATCH se utiliza cuando se quiere **cambiar un dato** concreto sin tocar el resto, esta no es la única diferencia entre **PATCH** y **PUT**, con **PATCH** también puedes **añadir campos extra**, por ejemplo, si tenemos un json con los **campos id, nombre y edad** con **PATCH** podemos **añadir otro campo** que se llame por ejemplo **email**, pero solo se añadirá ese campo al objeto que se quiere actualizar, ya que a la hora de especificar que objeto se quiere actualizar es la misma manera que **PUT**.

Ejemplo de aplicación: Para **actualizar algunos** campos de **un equipo**.

```
// función que recibe un equipo con los campos que se quieran
// actualizar en la db.
// Recibe un identificador para saber cual es el equipo a
// actualizar
// y también recibe un objeto equipo, que tiene el siguiente
// aspecto (según los campos a actualizar):
// reroll (int)
export async function patchEquipo(id_equipo, equipo) {
    try {
        const response = await
fetch("http://localhost:3000/equipos/" + id_equipo, {
            method: "PATCH",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify(equipo)
        });
        if (!response.ok) {
            throw new Error("Error al guardar el jugador");
        }
    } catch (error) {
        console.log("Error:", error);
    }
}
```

Si todo ha ido bien el servidor guarda los datos del equipo y responde con el objeto actualizado.

```
{  
    "id": "1",  
    "nombre": "Humans",  
    "normasEspeciales": [  
        "Team Captain"  
    ],  
    "reroll": "20",  
    "apotecario": true  
}
```

DELETE

Esta petición **borra** un recurso de la base de datos. **DELETE** se utilizar cuando quieras eliminar a un usuario, producto, etc. En esta petición también hay que **especificar** que objeto queremos **eliminar**. Se especifica de la misma manera que las anteriores, poniendo el **id correspondiente en la URL**.

Ejemplos de aplicación: Para **eliminar un jugador** en concreto.

```
// función que recibe un identificador y elimina al jugador de la bd.  
// Recibe un identificador para saber cual es el jugador a eliminar.  
export async function deleteJugador(id_jugador) {  
    try {  
        //Se crea la petición con algunos parámetros  
        //Si no se pone nada por defecto es GET  
        const response = await  
fetch("http://localhost:3000/jugadores/" + id_jugador, {  
            method: "DELETE",  
            headers: {  
                "Content-Type": "application/json"  
            }  
        });  
        if (!response.ok) {  
            throw new Error("Error al guardar el jugador");  
        }  
    } catch (error) {  
        console.log("Error:", error);  
    }  
}
```

Si todo va bien devolverá el **elemento/objeto eliminado**, pero en el archivo **json** no aparecerá ese elemento/objeto.

```
{  
    "id": "2",  
    "posicion": "Humano",  
    "tags": "250",  
    "cantidad": 20,  
    "coste": 350,  
    "MA": 5,  
    "FU": 2,  
    "AG": 3,  
    "PA": 4,  
    "AR": 7,  
    "Habilidades": ["Stunty", "Dodge", "Right Stuff"],  
    "Pri": ["A"],  
    "Sec": ["D", "G", "S"],  
    "Equipos": ["2"]  
}
```

FUNCIONALIDADES EXTRA

Para la mejora de nuestra aplicación hemos decidido **implementar 2 funcionalidades extra**. Una es la funcionalidad de **Multidioma** y la otra implementación es **generar un PDF**.

MULTIDIOMA

Para la facilidad de los usuarios hemos decidido implementar la funcionalidad **Multidioma**. La aplicación tendrá **3 opciones** en la parte **derecha del menú**, estas opciones representan las **banderas/ikurriñas** de cada idioma, en nuestro caso, **español, inglés y euskera**. Al clicar en alguna de estas opciones se **cambiará automáticamente los textos a mostrar**.

Internamente la aplicación llama al **método loadLanguage** que se le pasará como parámetro el código **lang del idioma seleccionado**, dentro de ese método llama a otro que se llama **applyStrings**, este es el que se encarga de que **aplicar los strings a los componentes**.

```
let strings = {};  
// aquí se guardarán los textos cargados  
export async function loadLanguage(lang) {  
    //meter el idioma en la sesión  
    sessionStorage.setItem("idioma", lang);  
    // carga el idioma seleccionado  
    const res = await fetch(`../lang/${lang}.json`);  
    const localStrings = await res.json();  
    // carga español como fallback  
    const fallbackRes = await fetch("../lang/es.json");  
    const fallbackStrings = await fallbackRes.json();  
    // mezcla: si falta algo en el idioma elegido, usa español  
    strings = { ...fallbackStrings, ...localStrings };  
    //Llama al método applyStrings() para aplicar los textos  
    applyStrings();  
}  
  
function applyStrings() {  
    //Obtiene los keys de los strings  
    const ids = Object.keys(strings);  
    //Recorre cada key  
    ids.forEach(id => {  
        //Obtiene el elemento que tiene el mismo id que el key  
        const el = document.getElementById(id);  
        if (el) {  
            el.innerText = strings[id];  
        }  
    });  
}
```

FUNCIONALIDAD EN LA INTERFAZ DE USUARIO (UI)

Para poder **cambiar el idioma** hay un **select** en la parte de **arriba a la derecha**, donde aparecen **los 3 idiomas disponibles** que son **euskera, inglés y español**, en cuanto se selecciona una opción se traducen todos los textos al **idioma seleccionado**. Esta opción es **accesible desde cualquier página**.

Qué es Blood Bowl

Blood Bowl es un juego de mesa de fantasía creado por Games Workshop que combina el fútbol americano con el universo de Warhammer Fantasy. En él, equipos de diversas razas, como humanos, enanos, orcos, elfos y no-muertos, compiten en un deporte brutal y violento, donde no solo importa el marcar goles, sino también la capacidad para aplastar, derribar y, a veces, asesinar a los jugadores rivales. El juego se desarrolla sobre un tablero cuadrulado, y los jugadores se mueven por turnos, tratando de avanzar con el balón mientras defienden su zona y atacan a sus oponentes.

Cada equipo tiene características únicas, lo que significa que cada raza juega de manera distinta. Por ejemplo, los orcos son más fuertes y resistentes, mientras que los elfos son más ágiles y rápidos. Los jugadores controlan a los personajes de sus equipos, tomando decisiones estratégicas sobre cómo moverse, pasar el balón, hacer bloques o emplear habilidades especiales. Sin embargo, **Blood Bowl** no es solo un juego de habilidades, sino también de suerte, ya que los dados juegan un papel fundamental en el resultado de las jugadas, lo que aporta una capa de imprevisibilidad y emoción.

A lo largo de las ligas, los jugadores pueden desarrollar a sus personajes, mejorándolos con habilidades adicionales. Esto convierte al juego en una experiencia que evoluciona, ya que los equipos pueden crecer y adaptarse según las victorias y derrotas que experimentan a lo largo de una liga. Además, **Blood Bowl** no es solo un juego competitivo entre jugadores, sino también una rica historia de rivalidades, traiciones y victorias épicas dentro de un mundo lleno de humor negro y absurdos propios de la fantasía.

Zer da Blood Bowl

Blood Bowl Games Workshop-ek sortutako fantasiako mahai-gaineko joko bat da, futbola eta Warhammer Fantasy unibertsioan uztartzen dicensa. Bertan, hainbat arraza (gizonezkoak, nanoak, orkak, elfos eta hildakoak) taldeak lehastzen dira kirol boritz eta indartsu batean, non golak sartzea garantitza dena, baixa boia ere, aurkarien jokalariek zapaltzea, lurerratea eta batzuetan hiltszea ere. Jokoak taula koadratuan jokatzen da, eta jokalariek txandaka mugitzen dute euren txandetan, baloia aurteratzeko, berenik erremua defendatze eta aurkaririk erasotuz.

Talde bakoiatzak ezagutegi bereziki ditu, horrek esan nahi du arraza bakoiatzak modu desberdinarekin jokatzen duela. Adibidez, orkak indartsuagoak eta iraunkorragoak dira, elfoak berriz azkarragoak eta maluguagoak. Jokalariek euren taldeko pertsonaiak kontrolatzeko dituzte, mugitzeko, baloia pasatzeko, blokeatzeko edo gaitasun berezik erabiltzeko estrategiak hartuz. Hala ere, **Blood Bowl** ez da soilik trebezia joko bat, baizik eta zorionta ere izan behar duzu, dideo jokatzen baitute jokaldien emaitza, eta hori ezezonkorrasun eta emozio geruza bat ekarren du.

Ligetan zehar, jokalariek euren pertsonaiak garatu ditzakete, gaitasun gehigarriak lortuz. Horrek jokoaren esperientzia ebolutiboa bilurtzen du, taldeek euren garapenean eta egokiztean aurrera egun ahalia, garaien eta porrotak izan dituztela. Horrez gain, **Blood Bowl** ez da soilik jokalarien arteko lehiaketa bat, baizik eta izpiritutako beltzezko eta fantasiako absurduruen unibertsioa, lehia, traizio eta garaien epikoen istorio aberatua.

What is Blood Bowl

Blood Bowl es un fantasy tabletop game created by Games Workshop that combines American football with the Warhammer Fantasy universe. In it, teams of various races, such as humans, dwarfs, orcs, elves, and undead, compete in a brutal and violent sport where scoring goals is important, but so is the ability to crush, knock down, and sometimes kill opposing players. The game is played on a grid board, with players moving in turns, trying to advance with the ball while defending their area and attacking their opponents.

Each team has unique characteristics, meaning each race plays differently. For example, orcs are stronger and more resilient, while elves are more agile and fast. Players control their team's characters, making strategic decisions on how to move, pass the ball, block, or use special abilities. However, **Blood Bowl** is not only a game of skill but also of luck, as dice play a key role in the outcome of plays, adding a layer of unpredictability and excitement.

Throughout the leagues, players can develop their characters, improving them with additional skills. This makes the game an evolving experience, as teams can grow and adapt based on the victories and defeats they experience over the course of a league. Additionally, **Blood Bowl** is not only a competitive game between players but also a rich story of rivalries, betrayals, and epic victories within a world filled with dark humor and absurdities typical of fantasy.

GENERAR PDF

Para poder explicar bien este método, hay que tener en cuenta cuando se puede utilizar esta funcionalidad. Esta aplicación te permite **crear equipos** para poder jugar, desde esta página se podrá acceder a la función **generarPDF**, también se encontrarán **todos los jugadores y extras** del **equipo seleccionado en el select**, este encuentra en la parte de arriba. Se pueden seleccionar todos los jugadores que se quiera, teniendo en cuenta que el **mínimo son 11 jugadores y el máximo 16**, y que el coste **máximo es de 1000k**. Cuando se haya terminado de seleccionar todos los jugadores, **en la parte de abajo**, hay un botón donde pone **validar**, este es el que se encarga de llamar al **método de generarPDF**, donde se pasarán como parámetros **todos los elementos del equipo en un mapa**, toda la **información de los jugadores en un array** y por último **el nombre del equipo**.

Si nos metemos en el código veremos los siguientes métodos sobre **generar un PDF**. El primero es **accionGenerarPDF**, este es el evento del **botón**. Lo que hace es **previene que se recargue el formulario**, verifica que los **jugadores totales están por encima de 11** y que el **dinero es menor o igual a 1000**, antes de verificar la cantidad de jugadores y de dinero gastado, **verifica si el nombre se ha introducido o no**, si **está todo correcto** directamente llamará al método **generarPDF**, si **no está todo correcto** mostrará un mensaje de **error** dependiendo **del idioma seleccionado**.

[EL CÓDIGO SE ENCUENTRA EN LA SIGUIENTE PÁGINA]

```
// función para generar el PDF
function accionGenerarPDF(event) {
    event.preventDefault();
    let nombreEquipo =
document.getElementById("txtNombreEquipo").value.trim();
    if (nombreEquipo != "") {
        if (dineroGastado <= 1000 && jugadoresTotales >= 11) {
            generarPDF(nombreEquipo, mapaElementosEquipo, infoJugadores)
        } else {
            switch (sessionStorage.getItem("idioma")) {
                case "en":
                    alert("You must have at least 11 players an a max of
1000k of spent treasury.");
                    break;
                case "eu":
                    alert("11 jokalari gutxienez izan behar dituzu eta
gehienez 1000k gastatuta.");
                    break;
                case "es":
                    alert("Debes tener un mínimo de 11 jugadores y un
máximo de 1000k gastados.");
                    break;
            }
        }
    } else {
        switch (sessionStorage.getItem("idioma")) {
            case "en":
                alert("You must to write a name for your team.");
                break;
            case "eu":
                alert("Zure taldearen izena idatzi behar duzu.");
                break;
            case "es":
                alert("Debes insertar un nombre para tu equipo.");
                break;
        }
    }
}
```

Este **método** es el que se encarga **realmente de crear el documento PDF**. Para empezar, **importa jsPDF**, es una librería de **JavaScript** que permite **generar archivos PDF directamente en el navegador**, sin necesidad de un servidor. Después de importar **jsPDF**, empieza con la creación de ello con algunos parámetros, por ejemplo, la orientación **es horizontal** y el **tamaño de los textos será de 18**, después **verifica que tipo de idioma** se está utilizando para poner un nombre u otro, para **poner texto en el PDF** se hace con el método **.text**. Después empieza con **la creación de las tablas** con el método **.autoTable**, pero antes de ello **modifica** algunos campos de los jugadores ya que viene **con un solo dato** y en el **PDF le queremos dar algún tipo de formato**, con esto quiero decir que **por ejemplo** en el **campo de precio solo saldría el valor numérico**, pero nosotros le queremos **añadir al valor la letra k** para que el usuario sepa **de que unidad estamos hablando**, también **el nombre varía según el idioma seleccionado**... A continuación, cuando se tengan todos los campos como se quiere ver en el **PDF se procede a crear la tabla**, este método tiene que tener unos campos como, **startY, head, body, theme, headStyles, tableWidth y styles**. **StartY** es donde **se empezará a dibujar la tabla**, **Head** es donde se insertan las **cabeceras** que en nuestro caso **varía según el idioma seleccionado**, **Body** es el contenido que va a tener la tabla en nuestro caso es **el array de jugadores anteriormente modificado**, **Theme** es donde se configura el **estilo de la tabla** en nuestro caso **grid**, **HeadStyles** es donde se insertan los **estilos para la cabecera**, **TableWidth** es el **tamaño que se le quiera dar a la tabla** y por último **Styles** donde se insertan **los estilos que se quieran**. Después de configurar la **tabla de jugadores**, se empieza a configurar la **tabla de los extras (Si es que existen)**, este proceso **es igual al anterior**, **primero se modifican los campos** con los valores que se quieran mostrar en el **PDF**, después con **.autoTable**, con **todos los campos rellenos** y teniendo en cuenta que en nuestro caso en el **Head** aparecerá **un texto u otro**, **se crea la tabla de los extras**. Y, por último, **guarda el PDF y lo descarga automáticamente**.

[EL CÓDIGO SE ENCUENTRA EN LA SIGUIENTE PÁGINA]

```
// import jsPDF
const { jsPDF } = window.jspdf;

//declarar let infoJugadores = {}; Tener en cuenta que aqui va la informacion de los jugadores y los extras
//al crear las tarjetas/crear la tabla hacer (ejemplo de uso, los valores del objeto, son lo que hay que reemplazar):
// infoJugadores["linea"] = {
//     posicion:"linea",
//     nombre:"LINEA",
//     tags:["Human","Linea"],
//     limite:16,
//     precio:50,
//     mv:6,
//     fu:3,
//     ag:3,
//     pa:3,
//     ar:9,
//     habilidades:[],
//     pri:[ "G" ],
//     sec:[ "A", "P" ]
// };

// variable para almacenar los jugadores y extras seleccionados:
// let mapaElementosEquipo = new Map(); (key: nombre del elemento, value: cantidad de dicho elemento)
// este mapa guarda informacion sobre todos los elementos añadidos al crear el equipo
// ejemplo: mapaElementosEquipo.set("linea",4);

//funcion para generar un pdf del equipo
//nombreEquipo: Nombre del equipo
//mapaElementosEquipo:
//infoJugadores: mapa con informacion de los jugadores y extras
export function generarPDF(nombreEquipo, mapaElementosEquipo, infoJugadores) {
    const doc = new jsPDF({ orientation: "landscape" });
    doc.setFontSize(18);
    switch (sessionStorage.getItem("idioma")) {
        case "en":
            doc.text(`Team: ${nombreEquipo}` , 14, 15);
            break;
        case "eu":
            doc.text(`Taldea: ${nombreEquipo}` , 14, 15);
            break;
        case "es":
            doc.text(`Equipo: ${nombreEquipo}` , 14, 15);
            break;
    }
}
```

```
// tabla de jugadores
let dorsal = 1;
const jugadores = [];
for (let [nombre, cantidad] of mapaElementosEquipo.entries()) {
  const j = infoJugadores[nombre];
  // Jugadores = elementos con estadísticas (es decir, mv != "-")
  if (cantidad > 0 && j && j.mv != "-") {
    for (let i = 0; i < cantidad; i++) {
      jugadores.push([
        dorsal,
        j.posicion,
        j.tags.join("\n"),
        generarNombre(),
        j.precio + "k",
        j.mv,
        j.fu,
        j.ag + "+",
        j.pa + "+",
        j.ar + "+",
        j.habilidades.join("\n"),
        j.pri,
        j.sec
      ]);
      dorsal++;
    }
  }
}
doc.autoTable({
  startY: 25,
  //cabeceras
  head: () => {
    switch (sessionStorage.getItem("idioma")) {
      case "en":
        return [
          'Back Number',
          'Position',
          'Tags',
          'Name',
          'Price',
          'MV',
          'ST',
          'AG',
          'PA',
          'AR',
          'Skills',
          'Pri',
          'Sec'
        ];
    }
  }
});
```

```
        case "eu":
            return [
                'Atzera Zenbakia',
                'Posizioa',
                'Etiketak',
                'Izena',
                'Prezioa',
                'MU',
                'IN',
                'AG',
                'PA',
                'AR',
                'Gaitasunak',
                'Leh',
                'Big'
            ];
        case "es":
            return [
                'Dorsal',
                'Posición',
                'Etiquetas',
                'Nombre',
                'Precio',
                'MV',
                'FU',
                'AG',
                'PA',
                'AR',
                'Habilidades',
                'Pri',
                'Sec'
            ];
    }
})(),
//contenido
body: jugadores,
theme: 'grid',
headStyles: { fillColor: [50, 50, 50], textColor: 255 },
styles: { fontSize: 9, cellPadding: 2 },
tableWidth: '100%'
});
// Tabla de extras
const extras = [];
for (let [nombre, cantidad] of mapaElementosEquipo.entries()) {
    const j = infoJugadores[nombre];
    if (cantidad > 0 && j && j.mv == "-") {
        extras.push([
            j.posicion,
            cantidad,

```

```
j.limite,
j.precio,
j.habilidades
]);
}
}
doc.autoTable({
    startY: doc.lastAutoTable.finalY + 10,
    head: () => {
        switch (sessionStorage.getItem("idioma")) {
            case "en":
                return [
                    'Extra',
                    'Quantity',
                    'Limit',
                    'Price',
                    'Description'
                ];
            case "eu":
                return [
                    'Extra',
                    'Kantitatea',
                    'Muga',
                    'Prezioa',
                    'Deskribapena'
                ];
            case "es":
                return [
                    'Extra',
                    'Cantidad',
                    'Límite',
                    'Precio',
                    'Descripción'
                ];
        }
    })(),
    body: extras,
    theme: 'grid',
    headStyles: { fillColor: [30, 30, 30], textColor: 255 },
    styles: { fontSize: 9, cellPadding: 2 },
    tableWidth: '100%'
});
// guardar el pdf
doc.save(`$nombreEquipo}.pdf`);
```

En el método de **generarPDF** hay una llamada a un método llamado **generarNombre**, este lo que hace es **devolver un nombre compuesto por un nombre y un mote**. Este nombre es sacado **aleatoriamente de 2 arrays uno de nombres y el otro de motes**, pero **teniendo en cuenta el idioma seleccionado**.

```
//funcion que devuelve un nombre (nombre y mote) aleatorio
//sacado de 2 arrays constantes (NOMBRES y MOTES)
function generarNombre() {
    const NOMBRES = [
        "Thorgar",
        "Morgul",
        "Balin",
        "Gorim",
        "Skarn",
        "Fendrel",
        "Ulric",
        "Drakar",
        "Hroth",
        "Varg",
        "Ragnar",
        "Kroth",
        "Dagnar",
        "Thrain",
        "Orgrim",
        "Brugor",
        "Falkor",
        "Gundar",
        "Haldor",
        "Irik",
        "Jorvik",
        "Keldor",
        "Lothar",
        "Morgrim",
        "Nargul",
        "Oskar",
        "Pelor",
        "Quarn",
        "Rethor",
        "Skarok",
        "Thalgrim",
        "Urden",
        "Vorgath",
        "Wulfric",
        "Xandor",
        "Yrgoth",
        "Zarvik",
        "Brogor",
```

```
"Durnan",
"Eldric",
"Fargus",
"Gorath",
"Hrogar",
"Ithran",
"Jarkor",
"Korrin",
"Lurgan",
"Mordek",
"Norrik",
"Othran"
];
const MOTES_EN = [
  "\The Prole\",
  "\The Destroyer\",
  "\Quick Hands\",
  "\The Unstoppable\",
  "\Iron Claws\",
  "\Deadly Roar\",
  "\Fist of Stone\",
  "\Night Shadow\",
  "\Bloody Fang\",
  "\The Invincible\",
  "\Eagle Eye\",
  "\War Hammer\",
  "\Swift Spear\",
  "\Night Hunter\",
  "\Steel Storm\",
  "\Hawk Eye\",
  "\Furious Beast\",
  "\Lethal Claw\",
  "\The Ravager\",
  "\Cold Blood\",
  "\Iron Fist\",
  "\The Relentless\",
  "\Dreams Destroyer\",
  "\Dark Hawk\",
  "\Deadly Lightning\",
  "\Stone Shield\",
  "\Quick Slash\",
  "\Deadly Shadow\",
  "\Dragon's Fury\",
  "\Silent Destroyer\",
  "\Ghost Hand\",
  "\Roaring Colossus\",
  "\Thunder Fist\",
  "\Lone Wolf\",
  "\The Voracious\"
```

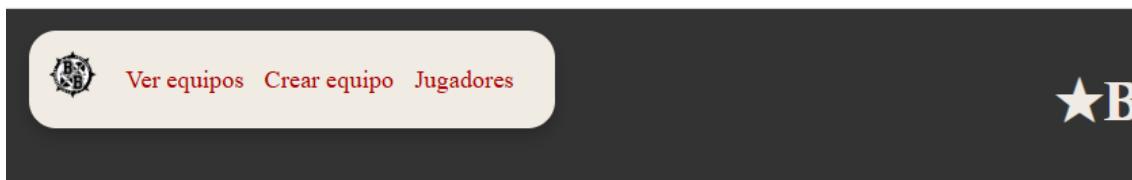
```
    "\"Deadly Hammer\"",
    "\"Sabre Tooth\"",
    "\"Fury Lightning\"",
    "\"Tiger Eye\"",
    "\"The Violent\"",
    "\"Shadow Claws\"",
    "\"The Thunderer\"",
    "\"Steel Roar\"",
    "\"Swift Saber\"",
    "\"Blade of the Night\"",
    "\"Men Destroyer\"",
    "\"Fist of Steel\"",
    "\"Unstoppable Fury\"",
    "\"The Annihilator\"",
    "\"Lethal Shadow\"",
    "\"Hawk Eye\""
];
const MOTES_ES = [
    "\"La Prole\"",
    "\"El Destructor\"",
    "\"Manos Rápidas\"",
    "\"El Imparable\"",
    "\"Garras de Hierro\"",
    "\"Rugido Mortal\"",
    "\"Puño de Piedra\"",
    "\"Sombra Nocturna\"",
    "\"Colmillo Sangriento\"",
    "\"El Invencible\"",
    "\"Ojo de Águila\"",
    "\"Martillo de Guerra\"",
    "\"Lanza Veloz\"",
    "\"Cazador Nocturno\"",
    "\"Tormenta de Acero\"",
    "\"Ojo de Halcón\"",
    "\"Bestia Furiosa\"",
    "\"Garra Letal\"",
    "\"El Devastador\"",
    "\"Sangre Fría\"",
    "\"Puño de Hierro\"",
    "\"El Implacable\"",
    "\"Destruidor de Sueños\"",
    "\"Halcon Oscuro\"",
    "\"Rayo Mortal\"",
    "\"Escudo de Piedra\"",
    "\"Corte Rápido\"",
    "\"Sombra Mortal\"",
    "\"Furia del Dragón\"",
    "\"Destructor Silencioso\"",
    "\"Mano Fantasma\"",
```

```
    "\"Coloso Rugiente\"",
    "\"Puño del Trueno\"",
    "\"Lobo Solitario\"",
    "\"El Voraz\",
    "\"Martillo Mortal\",
    "\"Diente de Sable\",
    "\"Furia Relámpago\",
    "\"Ojo de Tigre\",
    "\"El Violento\",
    "\"Garras Sombrias\",
    "\"El Tronador\",
    "\"Rugido de Acero\",
    "\"Sable Veloz\",
    "\"Espada de la Noche\",
    "\"Destruktor de Hombres\",
    "\"Puño de Acero\",
    "\"Furia Imparable\",
    "\"El Aniquilador\",
    "\"Sombra Letal\",
    "\"Ojo de Halcón\""
];
const MOTES_EU = [
    "\"Proletarioa\",
    "\"Suntsitzalea\",
    "\"Esku Azkarrak\",
    "\"Gelditu Ezina\",
    "\"Burdin Harkaitza\",
    "\"Hiltzaileen Oihuak\",
    "\"Harrizko Ukabila\",
    "\"Gaueko Itzala\",
    "\"Odol Kolpatu\",
    "\"Gailentza Gabea\",
    "\"Etxe Adarra\",
    "\"Gerra Martillo\",
    "\"Asto Lasterra\",
    "\"Gaueko Ehiztaria\",
    "\"Altzairuzko Ekaitza\",
    "\"Harrigorri Begia\",
    "\"Zaldi Sutsu\",
    "\"Azken Gorria\",
    "\"Suntsitzalea\",
    "\"Garaile Iguzkia\",
    "\"Burden Bakarra\",
    "\"Deabruen Deia\",
    "\"Gaueko Oihua\",
    "\"Buruzko Oihua\",
    "\"Pena Bastardoa\",
    "\"Herri Bat Makina\",
    "\"Suntsitzalea Maite\"",
```

```
"\"Berdintasunari\",
\"Beratasuna Edatetikak\",
\"Burdinezko Zuloak\",
\"Miasms Diefernatch\"
];
switch (sessionStorage.getItem("idioma")) {
    case "en":
        return NOMBRES[Math.floor(Math.random() *
NOMBRES.length)] + " " + MOTES_EN[Math.floor(Math.random() *
MOTES_EN.length)]
    case "eu":
        return NOMBRES[Math.floor(Math.random() *
NOMBRES.length)] + " " + MOTES_EU[Math.floor(Math.random() *
MOTES_EU.length)]
    case "es":
        return NOMBRES[Math.floor(Math.random() *
NOMBRES.length)] + " " + MOTES_ES[Math.floor(Math.random() *
MOTES_ES.length)]
}
```

FUNCIONALIDAD EN LA INTERFAZ DE USUARIO (UI)

Para poder **crear el PDF** primero hay que **crear un equipo**, para acceder a esa página desde el **menú** que se encuentra **arriba a la izquierda** tendremos un acceso a **crear equipo**.



Qué es Blood Bowl

Blood Bowl es un juego de mesa de fantasía creado por Games Workshop que combina el fútbol americano deporte brutal y violento, donde no solo importa el marcar goles, sino también la capacidad para aplastar, derribando de avanzar con el balón mientras defienden su zona y atacan a sus oponentes.

Cada equipo tiene características únicas, lo que significa que cada raza juega de manera distinta. Por ejemplo, equipos, tomando decisiones estratégicas sobre cómo moverse, pasar el balón, hacer bloqueos o emplear habilidades fundamentales en el resultado de las jugadas, lo que aporta una capa de imprevisibilidad y emoción.

A lo largo de las ligas, los jugadores pueden desarrollar a sus personajes, mejorándolos con habilidades adicionales que experimenten a lo largo de una liga. Además, **Blood Bowl** no es solo un juego competitivo entre jugadores de la fantasía.

En esta página nos encontramos con **el primer campo que es el nombre del equipo, sin él no se podrá crear el PDF** y en la parte de abajo aparecen **todos los jugadores y extras del equipo seleccionado** desde el **select** que aparece justo arriba de la tabla. Cuando se seleccionan los jugadores mínimos que son **11** se podrá **validar el equipo y se creará el PDF**.

The screenshot shows the 'Creador de Equipos' application interface. At the top, it says '★CREADOR DE EQUIPOS★'. Below that, there's a section titled 'Crea tu equipo' with a 'Nombre del equipo' input field containing 'Prueba'. A dropdown menu shows 'Humans' selected. The 'Tesorería gastada: 850/1000k' and 'Jugadores: 11/16' are displayed. The main area contains three player cards:

- Human Lineman**: 2-16. Stats: MA 6, ST 3, AG 3+, PA 4+, AV 9+. Cost: 50k. Buttons: Añadir, Eliminar.
- Halfling Hopeful**: 2-3. Stats: MA 5, ST 2, AG 3+, PA 4+, AV 7+. Abilities: Stunty, Dodge, Right Stuff. Cost: 30k. Buttons: Añadir, Eliminar.
- Human Catcher**: 2-2. Stats: MA 8, ST 3, AG 3+, PA 4+, AV 8+. Abilities: Catch, Dodge. Cost: 75k. Buttons: Añadir, Eliminar.

At the bottom, there's a red bar with the text 'Proyecto DWEC - 2025' and a timestamp '17:25 07/10/2025'.

The screenshot shows the 'Creador de Equipos' application interface after validation. At the top, it says '★CREADOR DE EQUIPOS★'. Below that, there are four sections with cards:

- Reroll**: 0-8. Abilities: Repetir tiradas. Cost: 50k. Buttons: Añadir, Eliminar.
- Ayudantes de entrenador**: 5-6. Abilities: Ayudan en una tirada de patada inicial. Cost: 10k. Buttons: Añadir, Eliminar.
- Animadoras**: 3-6. Abilities: Ayudan en alguna una de patada inicial. Cost: 10k. Buttons: Añadir, Eliminar.
- Fan Factor**: 0-6. Abilities: Ayudan en alguna una de patada inicial y en la tirada de ingresos en liga. Cost: 10k. Buttons: Añadir, Eliminar.
- Apotecario**: 0-1. Abilities: Permite repetir una tirada de lesión. Cost: 50k. Buttons: Añadir, Eliminar.

Below these sections is a long button labeled 'Validar'. At the bottom, there's a red bar with the text 'Proyecto DWEC - 2025' and a timestamp '17:25 07/10/2025'.

The screenshot shows a POS system interface. At the top, there's a file menu with "DISEÑO", "Maps", "HDFull - (El Real) Tu...", "LEG", and a download icon. A download dialog box is open, showing "Prueba.pdf" (42,6 KB), "Hecho". Below the menu, the POS logo "POS★" is visible. In the center, there's a product card for a product labeled "PA 5+" and "AV 10+", with a price of "140k". Below the card are buttons for "Iniciar", "Añadir", and "Eliminar".

Equipo: Prueba												
Dorsal	Posición	Etiquetas	Nombre	Precio	MV	FU	AG	PA	AR	Habilidades	Pri	Sec
1	Human Lineman	Human Lineman	Drakar "El Invencible"	50k	6	3	3+	4+	9+	-	G	A.D.S
2	Human Lineman	Human Lineman	Otrran "El Implacable"	50k	6	3	3+	4+	9+	-	G	A.D.S
3	Halfling Hopeful	Halfling Lineman	Mordek "Furia Imparable"	30k	5	2	3+	4+	7+	Stunty Dodge Right Stuff	A	D.G.S
4	Halfling Hopeful	Halfling Lineman	Gorath "Puño de Hierro"	30k	5	2	3+	4+	7+	Stunty Dodge Right Stuff	A	D.G.S
5	Human Catcher	Human Catcher	Vorgath "Ojo de Halcón"	75k	8	3	3+	4+	8+	Catch Dodge	A.G	P.S.D
6	Human Catcher	Human Catcher	Xandor "Destruidor de Sustos"	75k	8	3	3+	4+	8+	Catch Dodge	A.G	P.S.D
7	Human Thrower	Human Thrower	Brogor "Puño de Piedra"	75k	6	3	3+	3+	9+	Sure Hands Pass	G.P	A.D.S
8	Human Thrower	Human Thrower	Duman "El Invencible"	75k	6	3	3+	3+	9+	Sure Hands Pass	G.P	A.D.S
9	Human Blitzer	Human Blitzer	Lurgan "Furia del Dragón"	85k	7	3	3+	4+	9+	Block Tackle	G.S	A.D
10	Human Blitzer	Human Blitzer	Norkik "Furia Imparable"	85k	7	3	3+	4+	9+	Block Tackle	G.S	A.D
11	Ogre	Ogre Big guy	Skarok "Sable Veloz"	140k	5	5	4+	5+	10+	Bone Head Loner (3+) Bone Thick Skull Throw Team-mate	S	A.G.M

Below the main table is a "Extra" section:

Extra	Cantidad	Límite	Precio	Descripción
Ayudantes de entrenador	5	6	10	Ayudan en una tirada de patada inicial
Animadoras	3	6	10	Ayudan en alguna una de patada inicial