

Proyecto de Fin de Ciclo

# Sesión Cero

*Profesor Félix de Pablo Lobo*

---



Proyecto de Fin de Ciclo centrado en el desarrollo de una aplicación web para la creación intuitiva de personajes de Dungeons and Dragons.

*De Val Navares, Beatriz  
Menéndez Sanz, Juan Carlos  
Rojas Lendines, Sergio*



---

## AGRADECIMIENTOS

---

**S:** Ahora, llegando al final de este camino, quisiera expresar mi agradecimiento a todas las personas que han sido parte esencial de este recorrido. En primer lugar a mi pareja, Marina, quien me motivó a dar un giro a mi vida y cuyo apoyo incondicional y paciencia han sido fundamentales. A mi familia, por su constante comprensión con este nuevo objetivo. A todos mis compañeros, en especial a Carol, Ainara, Manuel y Beatriz, los eternos agonías, quienes han sido mis compañeros de grupo durante el curso y han hecho que esta experiencia sea mucho más amena, enriquecedora y divertida. Y por supuesto a Juan Carlos y Beatriz (de nuevo), con quienes he compartido este proyecto final. Por último, pero no menos importante, quiero agradecer al profesorado y equipo de UNIRFP por su apoyo y guía durante todo este proceso.

Por todo lo vivido, **GRACIAS**.

**B:** Después de estos dos años me gustaría dar las **GRACIAS**, en mayúsculas, a mi familia por hacerme ver que era capaz de esto, por comprenderme y animarme siempre. A mi pareja por tener fé ciega, y a mis amigos Jorge y Antía, quienes me dieron el empujón en esta aventura. Y sobre todo, a mis compañeros de proyecto, Juan Carlos y Sergio, con quienes he tenido la oportunidad de compartir esta frustrada (pero muy recompensada) experiencia, y a Carol, Manu y Ainara, quienes han ejercido de amigos, profesores y por qué no, psicólogos. Sin olvidarme de todo el profesorado, por su tiempo, cercanía y conocimientos.

**JC:** Quiero agradecer a mis profesores por su dedicación y enseñanzas, a mis compañeros por su colaboración y apoyo, y a mi familia por su constante aliento y comprensión. Sin ellos, este proyecto no hubiera sido posible. Su ayuda y respaldo han sido fundamentales en todo momento.

---

## ÍNDICE

---

<b>1.- Resumen - Definición del Proyecto</b>	<b>5</b>
1.1.- Summary - Project Definition	6
<b>2.- Objetivos del proyecto</b>	<b>7</b>
<b>3.- Situación actual</b>	<b>8</b>
3.1.- ¿Quiénes son nuestros competidores?	8
<b>4.- ¿Qué asignaturas apoyan el proyecto?</b>	<b>10</b>
4.1.- Programación	10
4.2.- Bases de datos	10
4.3.- Entornos de desarrollo	11
4.4.- Diseño de interfaces	11
4.5.- Desarrollo web en entorno cliente	11
4.6.- Desarrollo web en entorno servidor	11
<b>5.- Lenguajes empleados</b>	<b>12</b>
5.1.- Java	12
5.2.- TypeScript	12
5.3.- SQL	12
5.4.- HTML	13
5.5.- CSS	13
<b>6.- Herramientas y tecnologías</b>	<b>14</b>
6.1.- EclipseIDE	14

6.2.- Visual Studio Code	15
6.4.- Spring Framework	16
6.5.- Spring Boot	16
6.6.- Spring Data JPA	16
6.7.- Angular	16
6.8.- Figma	17
6.9.- GIT	17
6.10.- GitHub	17
<b>7.- Fases del proyecto</b>	<b>18</b>
7.1.- La idea	18
7.2.- Design Thinking	18
7.3.- Casos de uso	21
7.4.- Guía de estilos y wireframing	23
7.5.- Modelado de datos	25
7.6.- Angular	27
7.7.- Spring	30
<b>8.- Conclusiones y mejoras del proyecto</b>	<b>36</b>
<b>9.- Bibliografía</b>	<b>38</b>
<b>10.- Anexos</b>	<b>39</b>



**ROLL D20 TO CONTINUE**

## 1.- Resumen - Definición del Proyecto

---

### ¿Qué es y de qué trata nuestro proyecto?

El proyecto se ha centrado en diseñar y desarrollar una aplicación web con el objetivo de simplificar y mejorar la creación de personajes para el juego de rol Dungeons and Dragons (D&D). Permite a los usuarios crear múltiples personajes, personalizando cada uno según su clase, ascendencia, habilidades y otras estadísticas que definen su identidad en el juego. La aplicación ofrece una experiencia intuitiva y eficiente, facilitando tanto a principiantes como a jugadores experimentados la creación de personajes únicos para sus aventuras en el mundo de D&D.

La elección de desarrollar una aplicación para la creación de personajes de D&D surgió de la popularidad y el atractivo duradero de este juego de rol en la comunidad de aficionados a los juegos de mesa y rol. D&D es conocido por su profundidad y complejidad en la creación de personajes, lo que puede resultar abrumador para algunos jugadores, especialmente aquellos nuevos en el juego. La motivación principal fue simplificar y agilizar este proceso, permitiendo a los usuarios disfrutar más rápidamente de la parte creativa y emocionante del juego.

La aplicación está dirigida a una amplia variedad de jugadores y aficionados a Dungeons and Dragons. Esto incluye tanto a jugadores novatos que buscan una herramienta fácil de usar para crear sus primeros personajes, como a jugadores experimentados que desean una forma rápida y conveniente de generar nuevos personajes para sus campañas o partidas. También es útil para directores de juego que necesiten crear personajes no jugadores de manera eficiente.

## 1.1.- Summary - Project Definition

---

### What is our project about?

The project has focused on designing and developing a web application aimed at simplifying and enhancing character creation for the role-playing game Dungeons and Dragons (D&D). It allows users to create multiple characters, customizing each according to their class, ancestry, skills, and other statistics that define their identity in the game. The application offers an intuitive and efficient experience, facilitating both beginners and experienced players in creating unique characters for their adventures in the world of D&D.

The decision to develop an application for D&D character creation arose from the popularity and enduring appeal of this role-playing game in the tabletop and role-playing game community. D&D is known for its depth and complexity in character creation, which can be overwhelming for some players, especially those new to the game. The main motivation was to simplify and streamline this process, allowing users to more quickly enjoy the creative and exciting part of the game.

The application is aimed at a wide variety of Dungeons and Dragons players and enthusiasts. This includes both novice players looking for an easy-to-use tool to create their first characters, as well as experienced players who want a quick and convenient way to generate new characters for their campaigns or games. It is also useful for game masters who need to efficiently create non-player characters.

## 2.- Objetivos del proyecto

El proyecto que se ha desarrollado, pretende alcanzar estos objetivos y cubrir las necesidades que se enumeran a continuación:

- Simplificar el proceso de creación de personajes, facilitando a los usuarios la creación de personajes ofreciendo una interfaz intuitiva.
- Personalización detallada, permitiendo a los usuarios personalizar completamente sus personajes según sus preferencias, incluyendo clases, ascendencias, habilidades y otras estadísticas relevantes.
- Facilitar la creación, diseñando la aplicación para que sea accesible tanto para principiantes como para jugadores experimentados.
- Acelerar el proceso de creación, reduciendo el tiempo necesario para crear personajes, permitiendo así a los usuarios generar personajes rápidamente para iniciar sus aventuras sin demoras.
- Ofrecer una experiencia intuitiva, desarrollando una interfaz fácil de navegar y utilizar, asegurando una experiencia fluida y agradable.
- Soportar la creación de múltiples personajes, permitiendo a los usuarios la preparación para diferentes campañas y partidas.
- Almacenamiento y recuperación, dando opción a los usuarios de guardar sus personajes y acceder a ellos en cualquier momento.
- Proporcionar recursos descargables para las partidas, ofreciendo material adicional útil para enriquecer las sesiones de juego.

## 3.- Situación actual

### 3.1.- ¿Quiénes son nuestros competidores?

En el mercado actual, varias herramientas como "nivel20" y "D&D Beyond" compiten en la creación de personajes para Dungeons and Dragons, ofreciendo diversas características y enfoques. Esto plantea un desafío para diferenciarse, requiriendo no solo una interfaz intuitiva y características únicas, sino también una experiencia de usuario superior y un enfoque innovador para atraer y retener usuarios.

#### Nivel20 - <https://nivel20.com>

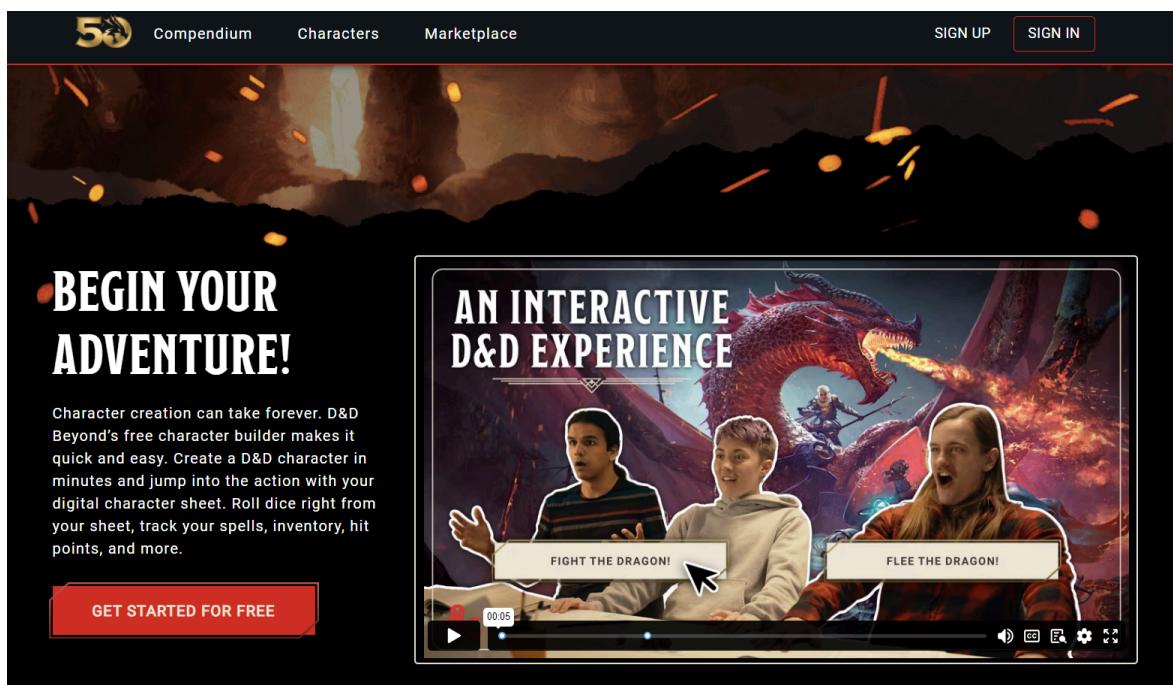
Es una plataforma web gratuita que facilita la creación, gestión y compartición de personajes y campañas para juegos de rol como Dungeons and Dragons, Pathfinder y El Resurgir del Dragón. Ofrece herramientas para jugadores y directores de juego, además de proporcionar recursos específicos para la 5<sup>a</sup> edición de D&D, como una biblioteca de conjuros. La plataforma cuenta con una comunidad activa en Discord, donde los usuarios pueden resolver dudas, buscar partidas y conectarse con otros aficionados al rol.



**D&D Beyond - <https://www.dndbeyond.com/>**

D&D Beyond es una plataforma digital oficial diseñada para facilitar la creación y gestión de personajes y campañas en la quinta edición de Dungeons & Dragons (D&D). Ofrece una variedad de herramientas que incluyen un constructor de personajes intuitivo, una biblioteca de contenido digital con libros oficiales, y herramientas para directores de juego.

Estas herramientas permiten a los jugadores personalizar personajes, gestionar inventarios y habilidades, y a los directores de juego gestionar combates y seguir el progreso de las campañas de manera eficiente. Además, la plataforma está disponible tanto en navegadores web como en dispositivos móviles, asegurando accesibilidad desde cualquier lugar.



## 4.- ¿Qué asignaturas apoyan el proyecto?

A lo largo del proyecto se reflejaron los diversos conocimientos adquiridos durante los dos años de formación. A continuación, se citan las asignaturas que más presencia tuvieron, así como los módulos más relevantes:

### 4.1.- Programación

El desarrollo de la aplicación incluyó el uso de los conocimientos adquiridos sobre Java. Se emplearon estructuras y bucles condicionales para establecer las reglas específicas que determinan las estadísticas de los personajes, basándose en sus habilidades y orígenes.

Además, se aplicaron los principios de la Programación Orientada a Objetos (POO) para modelar las diferentes entidades del juego, organizando el código de manera eficiente y flexible. También se utilizaron interfaces y Objetos de Acceso a Datos (DAO), en este caso llamados “Services”, facilitando la abstracción y el manejo de datos, lo que permitió una interacción fluida y organizada con la base de datos.

### 4.2.- Bases de datos

El proyecto incluyó la creación de una base de datos relacional, definiendo su estructura y características del modelo. La base de datos fue poblada y manipulada utilizando las sentencias del lenguaje del gestor de la propia base. Además, se diseñó el modelo relacional normalizado mediante la interpretación de diagramas de entidad/relación.

#### **4.3.- Entornos de desarrollo**

Se generó documentación de código a través de JavaDoc para la utilización del código a posteriori, así como para facilitar su lectura y comprensión por parte de terceros.

También se utilizaron las herramientas de Git y GitHub para llevar a cabo un eficaz control de versiones por parte de todos los integrantes.

#### **4.4.- Diseño de interfaces**

La interfaz de la aplicación web se desarrolló acorde a las especificaciones de diseño correspondientes a la temática elegida. Se crearon los elementos necesarios y se aplicaron las guías de estilo más adecuadas, teniendo en cuenta los criterios de accesibilidad y usabilidad.

#### **4.5.- Desarrollo web en entorno cliente**

El proyecto se dotó de funcionalidad a través de herramientas destinadas para la ejecución de la aplicación por navegadores en entornos web.

#### **4.6.- Desarrollo web en entorno servidor**

Se desarrolló una aplicación web con acceso a una base de datos utilizando lenguajes, objetos de acceso y herramientas de mapeo adecuadas.

## 5.- Lenguajes empleados

---

Antes de comenzar a hablar sobre las herramientas y tecnologías utilizadas, es conveniente mencionar los lenguajes de programación que han estado presentes en el proyecto:

### 5.1.- Java

Lenguaje de programación de alto nivel, orientado a objetos y basado en clases, diseñado para tener la menor cantidad posible de dependencias de implementación. Fue creado por James Gosling en Sun Microsystems y lanzado en 1995. La capacidad de Java de "escribir una vez, ejecutar en cualquier lugar" (WORA) significa que el código compilado en Java puede ejecutarse en cualquier plataforma que soporte Java, gracias a la Máquina Virtual de Java (JVM).

[Fuente - Encyclopedia Britannica](#)

### 5.2.- TypeScript

Lenguaje de programación de código abierto, orientado a objetos y desarrollado por Microsoft. Es un superconjunto tipado de JavaScript que agrega tipos estáticos, clases y otras características orientadas a objetos, permitiendo la verificación de tipos en tiempo de compilación. Esto ayuda a detectar errores antes de ejecutar el código, mejorando así la calidad y mantenibilidad del mismo.

[Fuente - Typescriptlang.org](#)

### 5.3.- SQL

(Structured Query Language) es un lenguaje de programación específico de dominio utilizado para gestionar y manipular datos en un sistema de gestión de bases de datos relacional (RDBMS).

Desarrollado en la década de 1970, SQL permite realizar operaciones de consulta, inserción, actualización y eliminación de datos de una manera eficiente y estructurada.

[Fuente - Amazon Web Services](#)

#### **5.4.- HTML**

(HyperText Markup Language) es el estándar de marcado utilizado para crear y estructurar páginas web. Define la estructura básica de una página mediante el uso de etiquetas y elementos, que indican al navegador cómo mostrar el contenido, como encabezados, párrafos, enlaces, imágenes y otros componentes multimedia.

[Fuente - W3Schools](#)

#### **5.5.- CSS**

(Cascading Style Sheets) es un lenguaje de hojas de estilo utilizado para describir la presentación de un documento escrito en HTML o XML. CSS controla cómo se muestran los elementos en pantalla. Con CSS se pueden definir estilos para elementos como colores, fuentes, espaciado, posicionamiento y disposición de los elementos en la página.

[Fuente - W3Schools](#)

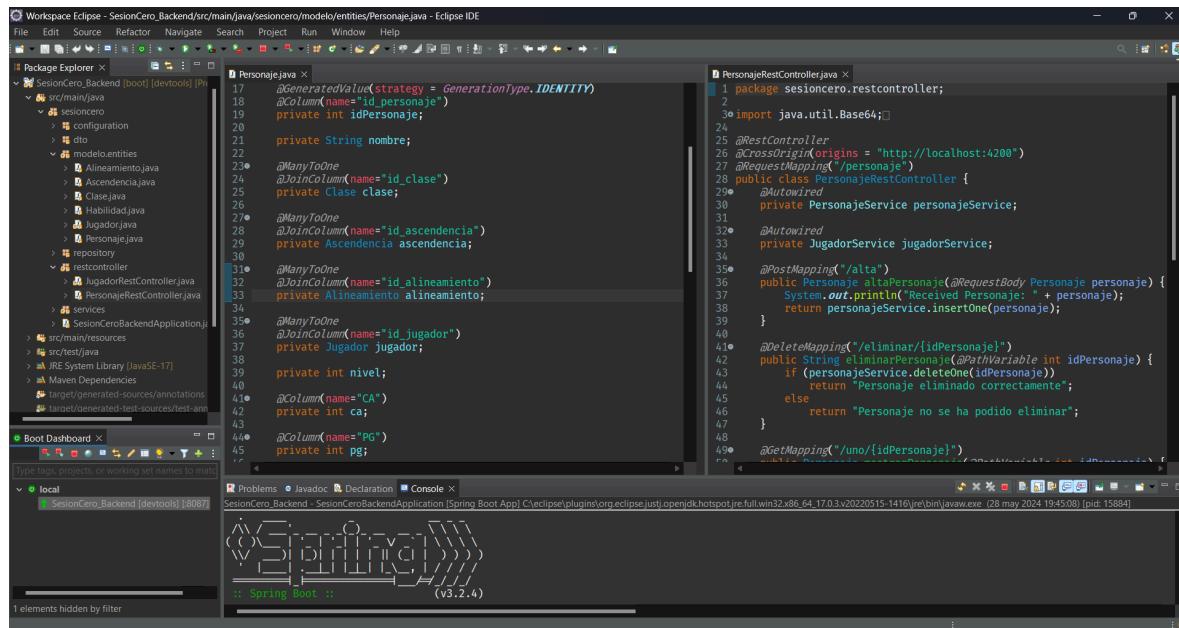
## 6.- Herramientas y tecnologías

Para la realización de la aplicación web se utilizaron diversas tecnologías destinadas al desarrollo tanto del *back* como del *front*. La elección de las mismas se llevó a cabo tras realizar un análisis y comparativa entre ellas, optando por aquellas que resultaron más útiles para el desarrollo del proyecto web, teniendo siempre en cuenta la funcionalidad:

### 6.1.- Eclipse IDE

Entorno de desarrollo integrado (IDE) utilizado principalmente para el desarrollo de aplicaciones en Java. Desarrollado originalmente por IBM y luego gestionado por la Fundación Eclipse, Eclipse es conocido por su arquitectura extensible basada en complementos (plug-ins), lo que permite a los desarrolladores agregar funcionalidades adicionales según sus necesidades específicas.

[Fuente - The Eclipse Foundation](#)



## 6.2.- Visual Studio Code

(VS Code) es un editor de código fuente desarrollado por Microsoft que se ejecuta en Windows, macOS y Linux. Es gratuito, de código abierto y ampliamente utilizado por desarrolladores debido a su rendimiento y extensibilidad. VS Code incluye soporte incorporado para JavaScript, TypeScript y Node.js, y cuenta con una rica oferta de extensiones para otros lenguajes y herramientas de desarrollo.

## Fuente - VS Code

```
1 import { Component, OnInit } from '@angular/core';
  2 import { FormBuilder, FormGroup, Validators, FormArray } from '@angular/forms';
  3 import { ascendencias } from '../dataConstants/ascendencias';
  4 import { clases } from '../dataConstants/clases';
  5 import { alineamientos } from '../dataConstants/alineamientos';
  6 import { habilidadesPorClase, todasLasHabilidades } from '../dataConstants/habilidades';
  7 import { PersonajeService } from '../services/personaje.service';
  8 import { Personaje } from '../interfaces/personaje.interface';
  9 import { HabilidadService } from '../services/habilidad.service';
 10 import { Router } from '@angular/router';

11 @Component({
12   selector: 'app-form',
13   templateUrl: './form.component.html',
14   styleUrls: ['./form.component.css'],
15 })
16 export class FormComponent implements OnInit {
17   formulario: FormGroup;
18   ascendencia = ascendencias;
19   clases = clases;
20   alineamientos = alineamientos;
21   habilidadesPorClase = habilidadesPorClase;
22   todasLasHabilidades = todasLasHabilidades;
23   habilidadesFijasSeleccionadas: string[] = [];

24   caracteristicas = [
25     { key: 'fuerza', label: 'Fuerza' },
26     { key: 'destreza', label: 'Destreza' },
27     { key: 'constitucion', label: 'Constitución' },
28     { key: 'inteligencia', label: 'Inteligencia' },
29     { key: 'sabiduria', label: 'Sabiduría' },
30     { key: 'carisma', label: 'Carisma' },
31   ];
32 }

33 private basePG = 0;
34 private modConstitucion = 0;
35 private idClaseAnteriorPG = 0;
36 private baseCA = 0;
```

```
1 form.component.html
  2 <div class="container mt-4 mb-5">
  3   <form [formGroup]="formulario" (ngSubmit)="onSubmit()">
  4     <div class="row justify-content-center">
  5       <!-- Columna para detalles -->
  6       <div class="col-md-3 mx-md-2">
  7         <h2 class="title">DETALLES</h2>
  8         <div class="form-group">
  9           <label for="nombre">Nombre</label>
 10           <input
 11             type="text"
 12             class="form-control form-control-sm"
 13             id="nombre"
 14             formControlName="nombre"
 15           />
 16           <div *ngIf="campoInvalido('nombre')" class="text-danger">
 17             El nombre es obligatorio.
 18           </div>
 19         </div>
 20         <div class="form-group">
 21           <label for="id_ascendencia">Ascendencia</label>
 22           <select
 23             class="form-control form-control-sm"
 24             id="id_ascendencia"
 25             formControlName="id_ascendencia"
 26           >
 27             <option *ngFor="let asc of ascendencias" [value]="
 28               {{ asc.name }}"
 29             >{{ asc.name }}
 30           </select>
 31           <div *ngIf="campoInvalido('id_ascendencia')" class="text-danger">
 32             La ascendencia es obligatoria.
 33           </div>
 34         </div>
 35         <div class="form-group">
 36           <label for="id_clase">Clase</label>
 37           <select
 38             class="form-control form-control-sm"
 39           >
```

## 6.3.- MySQL Workbench

Herramienta visual unificada para arquitectos, desarrolladores y administradores de bases de datos.

Fuente - Simplilearn

A screenshot of the MySQL Workbench interface. The title bar says "MySQL Workbench". The menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, Help. The toolbar has icons for Home, SesionCero, New, Open, Save, Print, Copy, Paste, Find, Replace, Undo, Redo, and others. The left sidebar shows "Navigator" and "SCHEMAS". Under SCHEMAS, there is a tree view with "cajero\_2023" expanded, showing "dungeons\_and\_dragons" which is further expanded to show "Tables" containing "alineamientos" and "ascendencias". A script editor window titled "DMD Script\_modificado\_final\*" is open, showing the following SQL code:

```
1 -- Crear la base de datos
2 • DROP DATABASE IF EXISTS dungeons_and_dragons;
3 • CREATE DATABASE dungeons_and_dragons;
4 • USE dungeons_and_dragons;
5
```

The status bar at the bottom says "Limit to 1000 rows".

## 6.4.- Spring Framework

Plataforma para desarrollar aplicaciones Java que ofrece soporte integral de infraestructura. En su núcleo, Spring se centra en la inyección de dependencias, lo que permite a los desarrolladores crear aplicaciones modulares con componentes acoplados de manera flexible.

[Fuente - Spring Documentation](#)

## 6.5.- Spring Boot

Extensión de Spring diseñada para simplificar la configuración y el despliegue de aplicaciones. Permite configurar automáticamente una aplicación con las dependencias y configuraciones necesarias, acelerando el desarrollo. Además, permite crear aplicaciones autónomas con un servidor web embedido, eliminando la necesidad de desplegar la aplicación en un servidor externo.

[Fuente - IBM](#)

## 6.6.- Spring Data JPA

Proyecto dentro del ecosistema de Spring que simplifica el acceso a bases de datos mediante el uso de JPA (Java Persistence API). Proporciona una capa de abstracción que facilita la interacción con bases de datos relacionales, permitiendo a los desarrolladores escribir consultas y operaciones de persistencia sin necesidad de manejar SQL.

[Fuente - Spring.io](#)

## 6.7.- Angular

Framework desarrollado por Google para la creación de aplicaciones web de una sola página (SPA). Utiliza TypeScript como lenguaje y ofrece una arquitectura basada en componentes, lo que facilita la creación y mantenimiento de aplicaciones escalables y modulares.

[Fuente - Simplilearn](#)

## 6.8.- Figma

Herramienta de diseño y prototipado basada en la nube, utilizada para proyectos digitales como la creación de interfaces de usuario (UI) y experiencias de usuario (UX). Lanzada en 2016, Figma permite a los usuarios colaborar en tiempo real, lo que facilita la revisión y el desarrollo conjunto de diseños.

[Fuente - Designshack](#)

## 6.9.- GIT

Sistema de control de versiones distribuido que permite a los desarrolladores gestionar y registrar cambios en el código fuente a lo largo del tiempo. Fue creado por Linus Torvalds en 2005 y es ampliamente utilizado en el desarrollo de software por su eficiencia y confiabilidad en la gestión de proyectos grandes y pequeños.

[Fuente - Git](#)

## 6.10.- GitHub

Plataforma de alojamiento de código que utiliza Git para el control de versiones. Proporciona una interfaz web para facilitar la colaboración entre desarrolladores, permitiendo compartir y trabajar en proyectos de código de manera más eficiente.

[Fuente - GitHub Docs](#)

## 7.- Fases del proyecto

### 7.1.- La idea

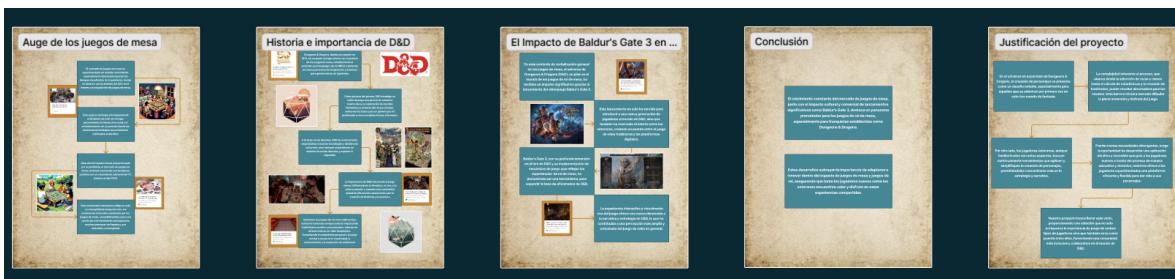
Durante la fase inicial del proyecto, se propusieron diversas ideas:

1. En primer lugar, crear una **Enciclopedia del universo Marvel**, que habría implicado desarrollar una base de datos extensa con información detallada sobre personajes, eventos y lugares, proporcionando un recurso interactivo para los fans.
2. Otra propuesta fue diseñar un **Tablero de Mando** para una flota espacial basado en el universo de Dune. Este tablero permitiría la gestión y control de una flota, incluyendo la administración de recursos, ofreciendo una experiencia inmersiva inspirada en el universo de Dune.
3. Finalmente, se decidió desarrollar una **Aplicación para la Creación de Personajes de Dungeons & Dragons**. Esta elección se basó en la popularidad de D&D y la necesidad de herramientas que simplifiquen y mejoren el proceso de creación de personajes.

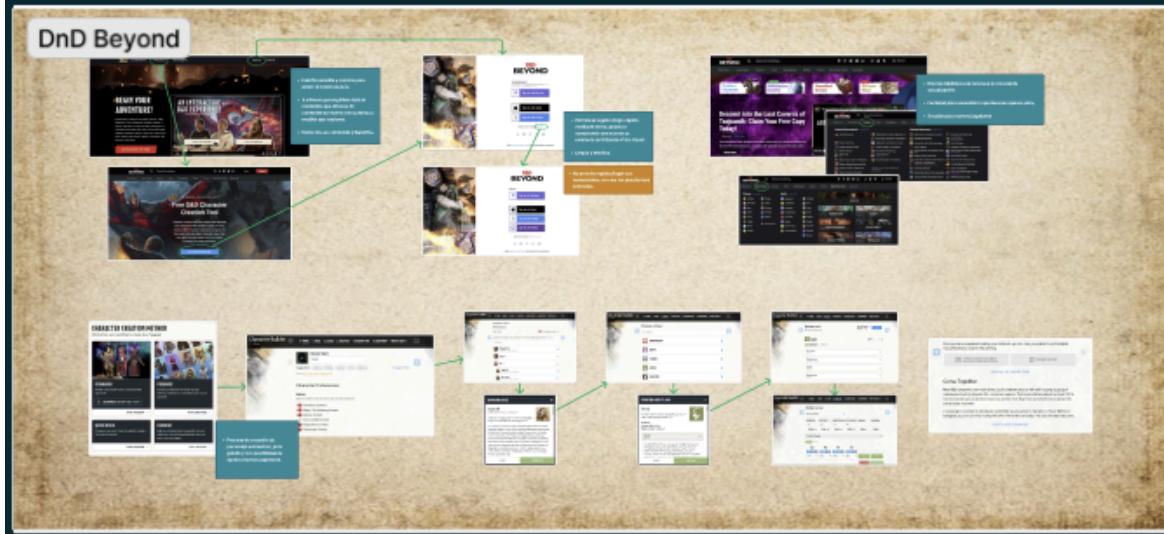
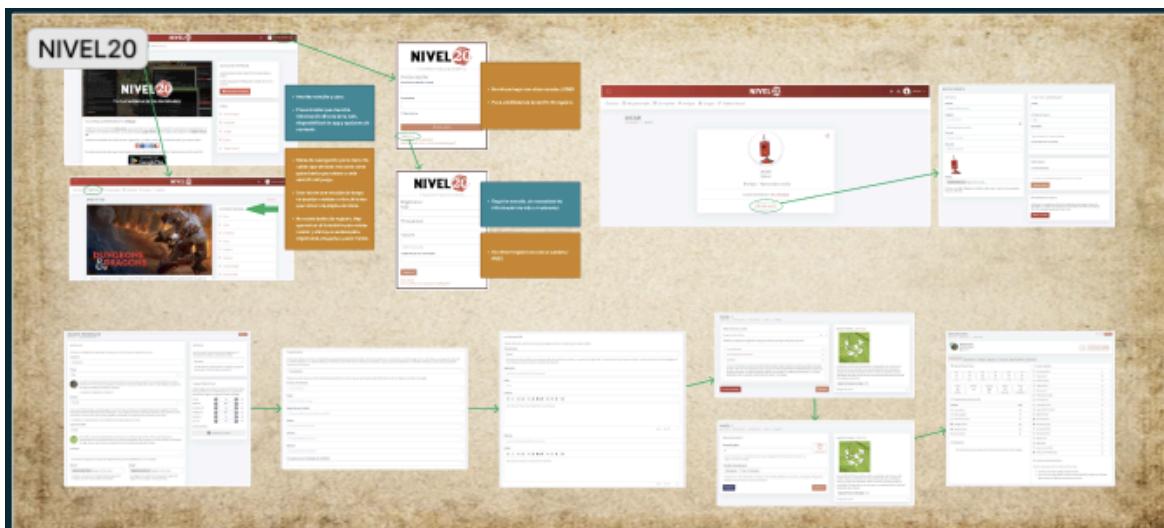
La decisión final se tomó considerando la viabilidad y el valor añadido que la aplicación podría ofrecer a la comunidad de D&D.

### 7.2.- Design Thinking

Para comprender mejor a qué usuario queremos dirigirnos, se utilizaron las técnicas aprendidas en la asignatura de Diseño de Interfaces Web. Se empleó FigJam para realizar un proceso de análisis y posteriormente crear una ficha de perfil del usuario típico.



Overview investigación



Overview benchmarking

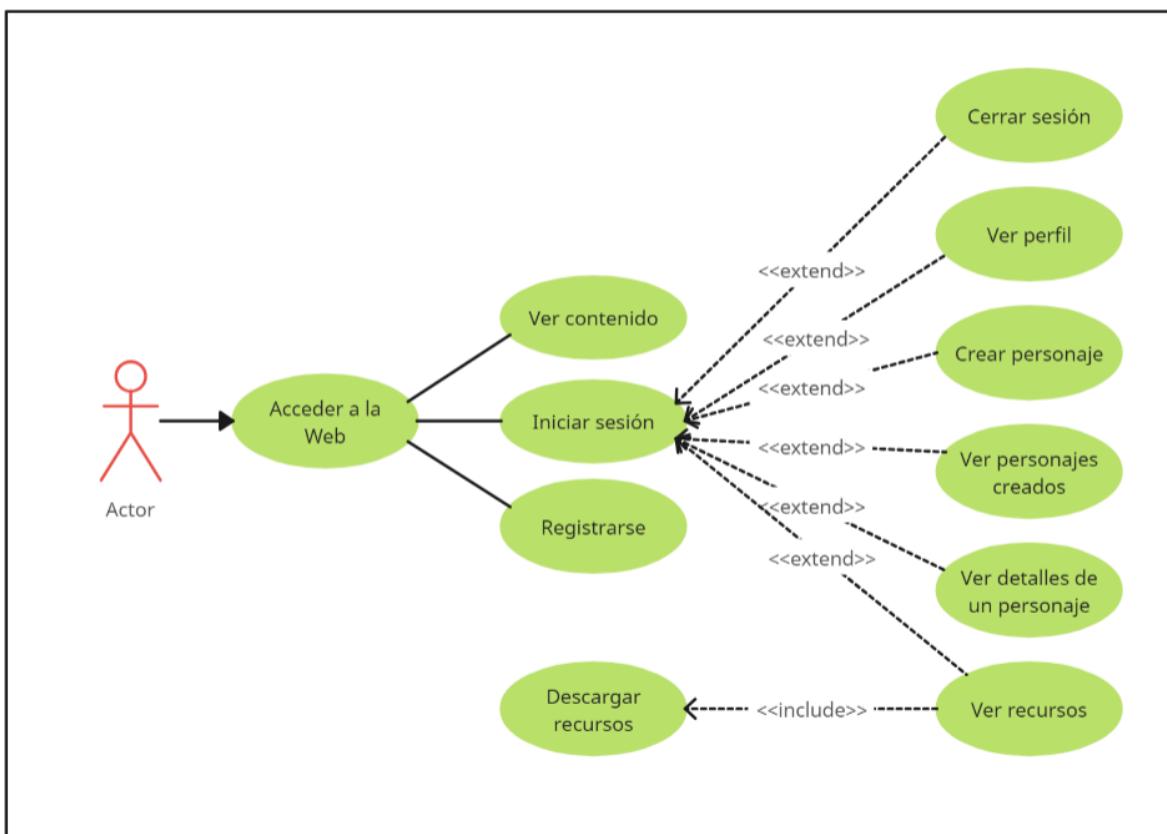
Este perfil se centra en jugadores de rol de entre 20 y 40 años, iniciados o apasionados por Dungeons & Dragons, que buscan herramientas para facilitar la creación de personajes. Son usuarios que desean invertir más tiempo en la aventura y menos en la preparación, o que se ven abrumados por la cantidad de cosas a tener en cuenta para jugar.

DATOS PERSONALES		
 JOHN DOE NOMBRE DEL PERSONAJE	24 EDAD	178CM ALTURA
 APARIENCIA DEL PERSONAJE	2 OJOS	SI PIEL
		75KG PESO
		TAMBIÉN CABELLO
<ul style="list-style-type: none"> <li>Joven Adulto/a</li> <li>En la universidad o recién ingresado en el mundo laboral, con un interés creciente en los juegos de rol como una forma de ocio y conexión social.</li> <li>Disfruta de la fantasía, videojuegos, y literatura. Busca actividades que estimulen su creatividad y le permitan escapar de la rutina diaria.</li> <li>Ha oido hablar de Dungeons &amp; Dragons y está intrigado por el concepto, pero se siente abrumado por la complejidad del juego y no sabe por dónde empezar.</li> </ul>		
<b>INVESTIGACIÓN DE USUARIO:</b> <b>PERFIL GENERAL</b>  <b>SÍMBOLO</b>		
<p>Recién graduado en Diseño Gráfico, trabaja como freelancer.</p> <p>John es un joven creativo y entusiasta de la cultura pop, especialmente interesado en la fantasía, los videojuegos y el arte digital.</p> <p>Aunque ha escuchado sobre Dungeons &amp; Dragons a través de amigos y redes sociales, nunca ha tenido la oportunidad de jugarlo debido a la percepción de que es demasiado complejo y requiere un compromiso de tiempo que no cree tener.</p> <p>Sin embargo, la idea de crear personajes únicos y sumergirse en aventuras épicas le atrae mucho.</p> <p>John busca formas de socializar en un entorno creativo y expandir su círculo social.</p>	<p><b>OBJETIVOS</b></p> <ul style="list-style-type: none"> <li>Aprender a jugar D&amp;D de una manera fácil y accesible.</li> <li>Encontrar una comunidad con la que pueda compartir su interés por la fantasía y el rol.</li> <li>Desarrollar su creatividad y aplicar sus habilidades de diseño en la creación de personajes.</li> </ul> <p><b>MOTIVACIONES</b></p> <ul style="list-style-type: none"> <li>El deseo de escapar de la rutina diaria a través de la inmersión en mundos de fantasía.</li> <li>La búsqueda de una actividad que combine su amor por la historia, la fantasía y el arte</li> </ul> <p><b>RASGOS Y ATRIBUTOS ADICIONALES</b></p> <p><b>POV 1:</b> John necesita una manera de aprender D&amp;D que se ajuste a su horario irregular de trabajo freelance sin sentir que está comprometiendo demasiado tiempo de antemano.</p> <p><b>POV 2:</b> Busca una herramienta que le permita expresar su creatividad en la creación de personajes, integrando su habilidad y pasión por el diseño gráfico.</p> <p><b>POV 3:</b> Desea una plataforma que no solo sea educativa sino también social, permitiéndole conectarse con otros jugadores que puedan ayudarlo a integrarse en la comunidad de D&amp;D.</p>	<p><b>FRUSTRACIONES</b></p> <ul style="list-style-type: none"> <li>Se siente abrumado por la cantidad de reglas y la complejidad de D&amp;D.</li> <li>Le preocupa no tener suficiente tiempo para el aprendizaje del juego.</li> </ul> <p><b>TESORO</b></p>

### 7.3.- Casos de uso

Una vez se han identificado los diferentes tipos de usuarios a los que se dirige la aplicación, y definido el user persona sobre el que se trabaja, se comienza el diseño. Antes de proceder, es esencial elaborar los casos de uso para los usuarios de la aplicación web.

Un diagrama de casos de uso es una representación gráfica utilizada en la ingeniería del software para describir las interacciones entre los actores externos (usuarios u otros sistemas) y un sistema. Este diagrama detalla las diversas funciones (casos de uso) que el sistema proporciona, mostrando cómo los actores interactúan con el sistema para lograr un objetivo específico. Cada caso de uso representa un conjunto de actividades o acciones que un actor puede realizar dentro del sistema.



1. **Entrar en la Web:** el usuario, al acceder a la web, tiene la opción de ver el contenido estático, registrarse o iniciar sesión:

- **Ver contenido estático:** el usuario puede acceder a información disponible en la web sin necesidad de registrarse ni iniciar sesión.
- **Registrarse:** si el usuario no está registrado, deberá completar un formulario de registro para crear una cuenta.
- **Iniciar sesión:** si ya está registrado, podrá iniciar sesión ingresando su usuario y contraseña.

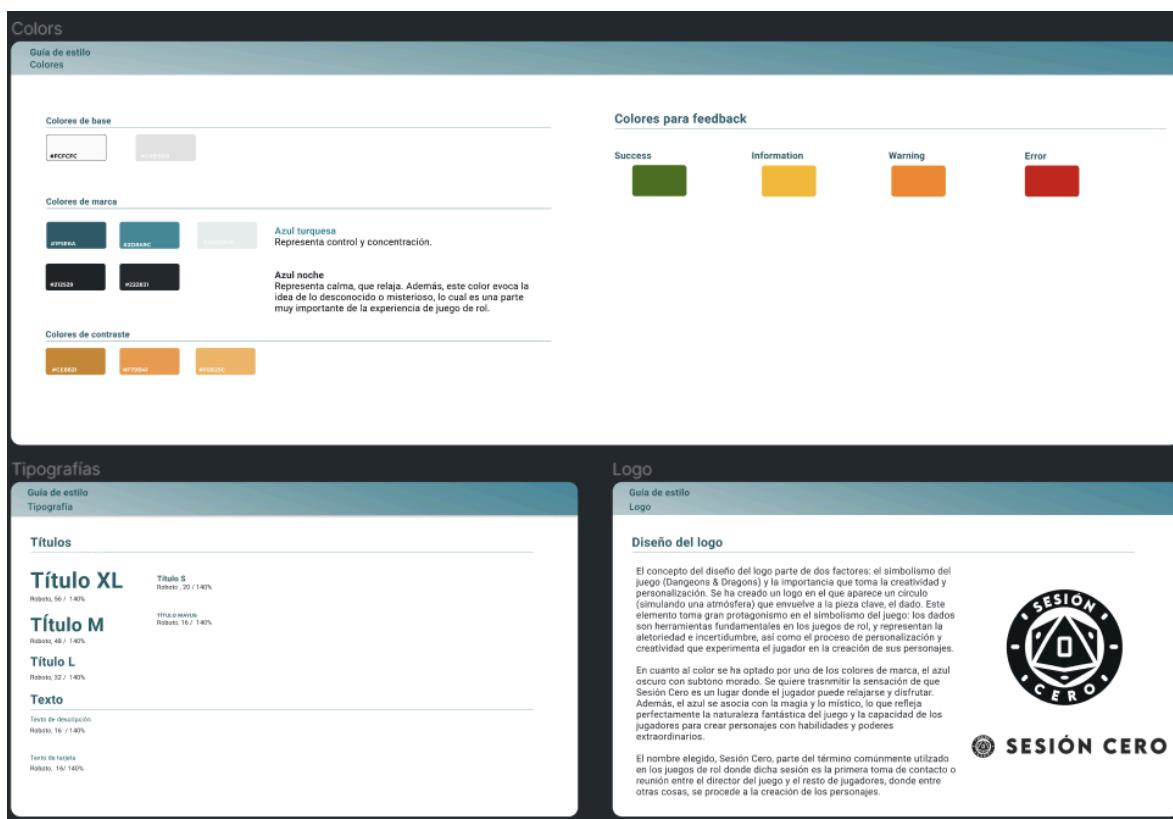
2. **Iniciar sesión:** una vez iniciada la sesión, el usuario puede realizar diversas acciones:

- **Cerrar sesión:** el usuario puede optar por cerrar su sesión en cualquier momento.
- **Crear personajes:** el usuario puede comenzar a crear nuevos personajes para sus aventuras de D&D.
- **Ver perfil:** el usuario puede acceder y revisar su perfil, incluyendo información personal y configuraciones.
- **Ver todos los personajes:** el usuario puede ver una lista de todos los personajes que ha creado.
- **Ver detalles de un personaje:** el usuario puede seleccionar un personaje específico para ver sus detalles completos.
- **Acceder a sección de recursos:** el usuario puede acceder a una sección dedicada a recursos adicionales.
  - **Descargar recursos:** dentro de la sección de recursos, el usuario puede descargar materiales para el juego.

## 7.4.- Guía de estilos y wireframing

Una guía de estilos en el desarrollo de interfaces web es un documento o conjunto de herramientas que establece directrices y normas visuales y de comportamiento para asegurar la coherencia y uniformidad en el diseño de una aplicación o sitio web. Esta guía abarca aspectos como la tipografía, paleta de colores, iconografía, espaciado, botones, formularios y otros elementos de la interfaz de usuario.

Su propósito es proporcionar un marco común que facilite la colaboración entre diseñadores y desarrolladores, mejorando la experiencia del usuario y manteniendo una identidad visual consistente a lo largo del proyecto.



**Colors**

Guía de estilo  
Colores

**Colores de base**

- #E9E9E9
- CCCCCC

**Colores de marca**

- #0072BC
- #008399
- CCCCCC
- #212121
- #E9E9E9

Azul turquesa  
Representa control y concentración.

Azul noche  
Representa calma, que relaja. Además, este color evoca la idea de lo desconocido o misterioso, lo cual es una parte muy importante de la experiencia de juego de rol.

**Colores de contraste**

- #E9E9E9
- #F7996U
- #E6E6E6

**Colores para feedback**

- Success
- Information
- Warning
- Error

**Tipografías**

Guía de estilo  
Tipografía

**Títulos**

Título XL	Título S
Rubato, 56 / 140%	Rubato, 20 / 140%

Título M	Título M
Rubato, 46 / 140%	Rubato, 16 / 140%

Título L	Título L
Rubato, 32,7 / 140%	Rubato, 16 / 140%

**Texto**

Texto de descripción	Texto de descripción
Rubato, 16 / 140%	Rubato, 16 / 140%

Texto de tabla	Texto de tabla
Rubato, 16 / 140%	Rubato, 16 / 140%

**Logo**

Guía de estilo  
Logo

**Diseño del logo**

El concepto del diseño del logo parte de dos factores: el simbolismo del juego (Dungeons & Dragons) y la importancia que toma la creatividad y personalización. Se ha creado un logo en el que aparecen un díctyon (simbolo de los jugadores) y un dado en su parte clara del logo. Este elemento toma gran protagonismo en el simbolismo del juego: los dados son herramientas fundamentales en los juegos de rol, y representan la aleteridad e incertidumbre, así como el proceso de personalización y creatividad que experimenta el jugador en la creación de sus personajes.

En cuanto al color se ha optado por uno de los colores de marca, el azul oscuro con subtono morado. Se quiere transmitir la sensación de que Sesión Cero es una actividad divertida y emocionante de disfrutar. Además, el azul se sincroniza con la magia y lo misterio, lo que refleja perfectamente la naturaleza fantástica del juego y la capacidad de los jugadores para crear personajes con habilidades y poderes extraordinarios.

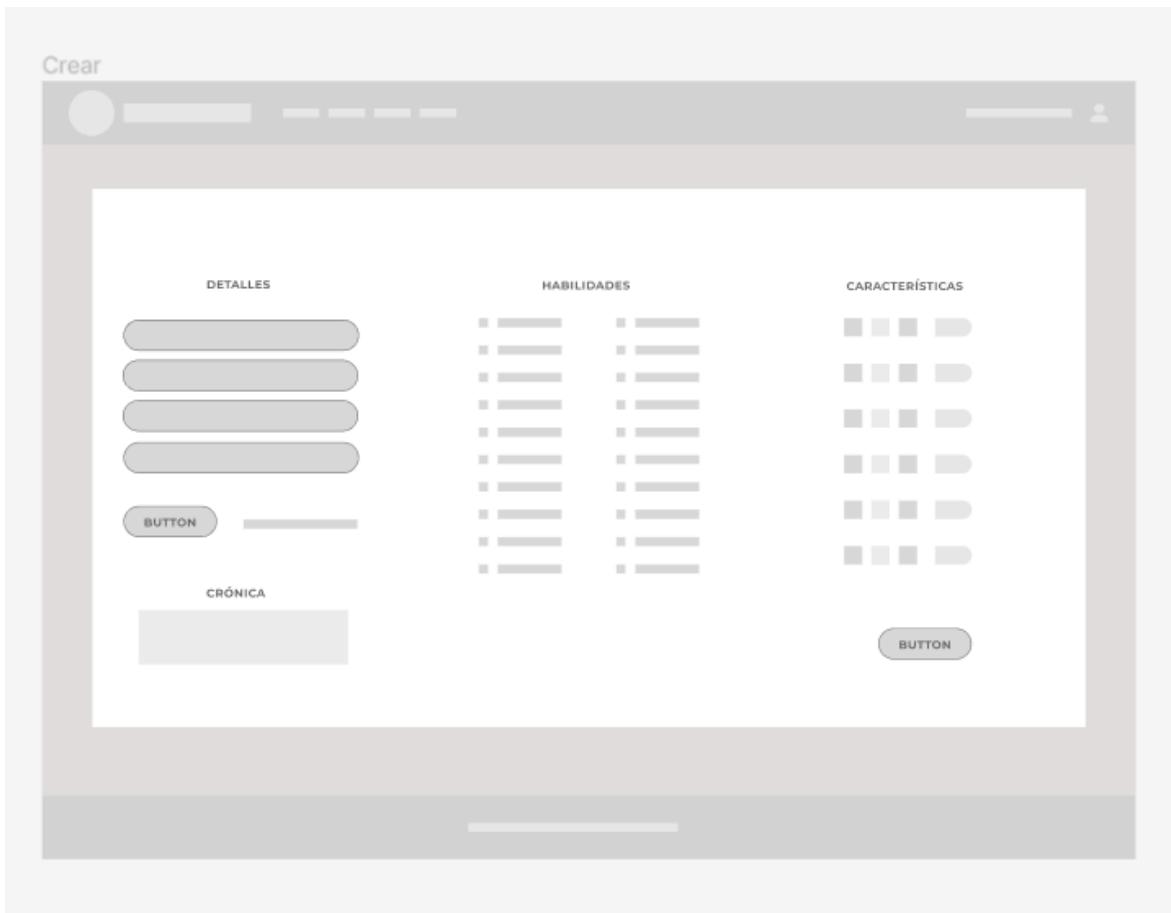
El nombre elegido, Sesión Cero, parte del término comúnmente utilizado en los juegos de rol donde dicha sesión es la primera toma de contacto o reunión entre el director del juego y el resto de jugadores, donde entre otras cosas, se procede a la creación de los personajes.

 SESIÓN CERO

Overview guía de estilos

Wireframing es una práctica esencial en el desarrollo de aplicaciones web y móviles. Un wireframe es una representación visual simplificada de la estructura y funcionalidad de una página web o una aplicación, actuando como un esqueleto que muestra la disposición de los elementos en la interfaz sin entrar en detalles de diseño como colores, gráficos o contenido específico.

[Fuente - Interaction Design Foundation](#)



*Ejemplo de wireframe*

## 7.5.- Modelado de datos

El modelado de datos es el proceso de crear una representación visual de un sistema de información para comunicar las conexiones entre diferentes puntos de datos y estructuras. Este proceso implica diagramas que representan entidades, atributos y relaciones, ayudando a organizar los datos de manera que reflejen las necesidades y faciliten su análisis.

Una base de datos es un sistema organizado para almacenar, gestionar y recuperar datos de manera eficiente. Pueden ser relacionales, estructurando los datos en tablas, o no relacionales, utilizando modelos como documentos o pares clave-valor.

Para esta aplicación, se ha utilizado una base de datos relacional. Esta elección se debe a su capacidad para manejar datos estructurados de manera eficiente, garantizando la integridad y consistencia de la información mediante el uso de relaciones y claves.

[Fuente - Simplilearn](#)

```

1      -- Crear la base de datos
2 •  DROP DATABASE IF EXISTS dungeons_and_dragons;
3 •  CREATE DATABASE dungeons_and_dragons;
4 •  USE dungeons_and_dragons;
5
6      -- Crear la tabla jugadores
7 •  CREATE TABLE `jugadores` (
8          `id_jugador` int NOT NULL AUTO_INCREMENT,
9          `nombre` varchar(100) NOT NULL,
10         `apellido1` varchar(100) NOT NULL,
11         `apellido2` varchar(100) DEFAULT NULL,
12         `email` varchar(100) NOT NULL,
13         `contraseña` varchar(250) NOT NULL,
14         PRIMARY KEY (`id_jugador`),
15         UNIQUE KEY `email` (`email`)
16     );

```

*Fragmento del script utilizado*

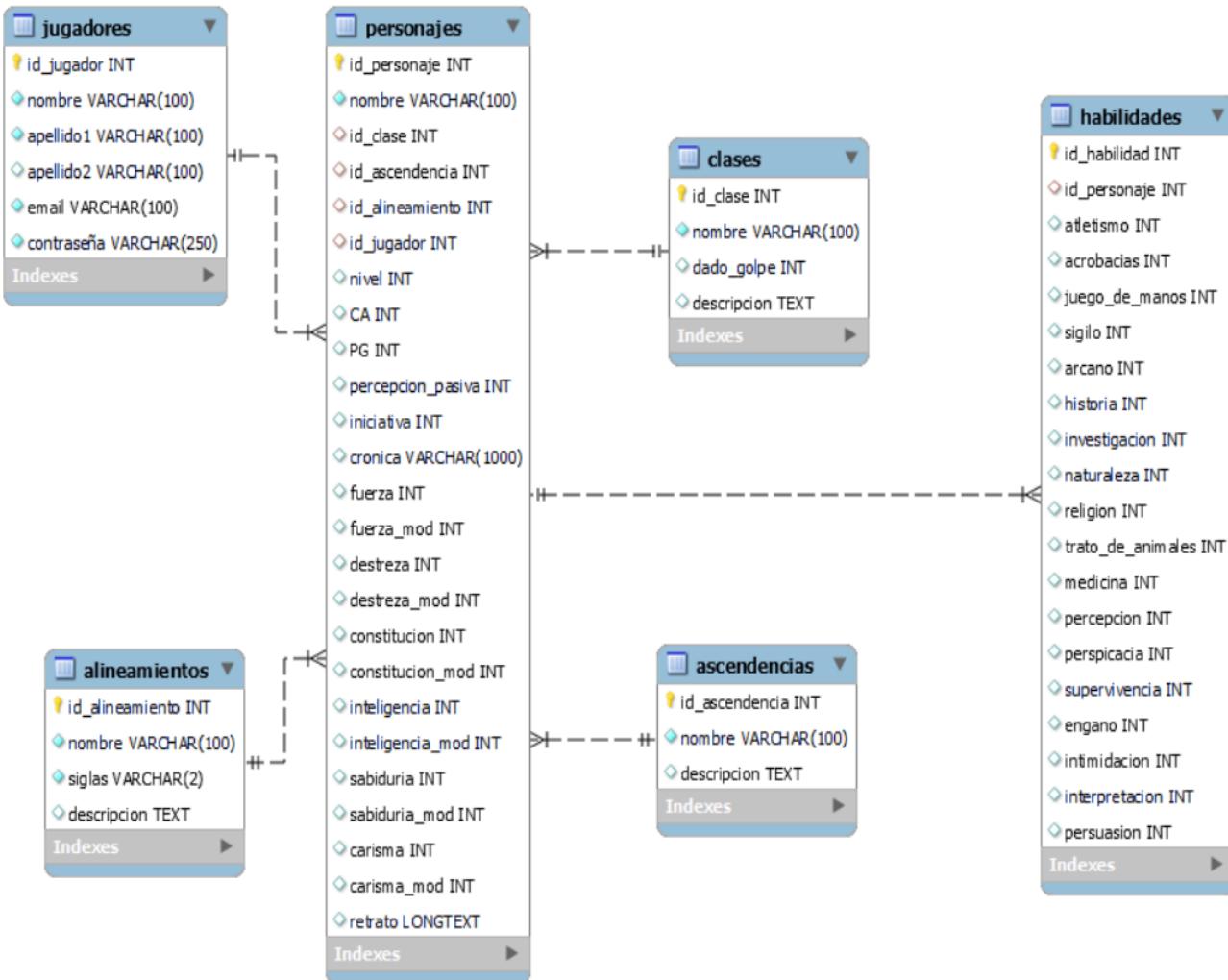


Diagrama Entidad-Relación

## 7.6.- Angular

Para el comienzo del desarrollo de la parte front con Angular, se empezaron creando los primeros componentes que compondrían la vista de la aplicación, específicamente la página principal o landing page, el nav y el footer. Estos componentes iniciales establecieron la estructura básica de la interfaz de usuario, permitiendo definir y organizar los elementos visuales esenciales que percibirá el usuario en su primer vistazo a la web. Para ello, se utilizaron HTML y CSS, que proporcionaron la estructura y el estilo necesarios.

En los siguientes pasos del proyecto, se crearon más componentes para agregar funciones y mejorar la experiencia del usuario.

```

1 <div class="body">
2 <div class="container mt-5 home-component">
3   <div id="carouselExampleControls" class="carousel slide" data-bs-ride="carousel">
4     <div class="carousel-inner">
5       <div class="carousel-item active">
6         
7       </div>
8       <div class="carousel-item">
9         
10      </div>
11      <div class="carousel-item">
12        
13      </div>
14    </div>
15    <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleControls" data-bs-slide="prev">
16      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
17      <span class="visually-hidden">Previous</span>
18    </button>
19    <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleControls" data-bs-slide="next">
20      <span class="carousel-control-next-icon" aria-hidden="true"></span>
21      <span class="visually-hidden">Next</span>
22    </button>
23  </div>
```

```

1 /* Estilos para los márgenes laterales */
2 .container {
3   margin-right: auto;
4   margin-left: auto;
5 }
6
7 /* Estilos para la alineación vertical de los elementos del carrusel */
8 .carousel-inner img {
9   margin: auto;
10 }
11
12 /* Estilos para las tarjetas */
13 .card {
14   margin-top: 20px;
15 }
```

Ejemplo HTML y CSS

Para optimizar el rendimiento y evitar llamadas continuas a la base de datos, se tomó la decisión de servir ciertos grupos de datos desde el front. En concreto, se almacenaron de forma estática y en constantes dentro del front valores como la información sobre clases, ascendencias y alineamientos, ya que son datos que no sufren modificaciones. Estas constantes se utilizan para poblar los campos correspondientes en los desplegables dentro del formulario de creación, mejorando así el rendimiento de la aplicación al reducir la carga sobre la base de datos.

Además, debido a la falta de un servidor de datos, se decidió guardar una serie de archivos en el front para proporcionarlos a través de una página de recursos. Estos archivos están disponibles para descarga, ofreciendo a los usuarios materiales esenciales para sus partidas. Esta solución permite mantener la funcionalidad y accesibilidad del contenido sin necesidad de infraestructura adicional.

Para gestionar la descarga de estos archivos, fue necesario crear un servicio específico, el DownloadService, que maneja las solicitudes de descarga y facilita la distribución de los recursos almacenados en el front.

```

1 import { Injectable } from '@angular/core';
2
3 /**
4  * Este es un servicio para descargar archivos.
5  *
6  * La anotación @Injectable indica que este servicio se puede injectar en cualquier parte
7  * de la aplicación. Al poner providedIn: 'root', Angular se encarga de crear una única instancia
8  * de este servicio que se puede usar en toda la app.
9  */
10@Injectable({
11  providedIn: 'root'
12})
13export class DownloadService {
14
15  constructor() { }
16
17  /**
18   * Este método permite descargar un archivo.
19   *
20   * @param {string} filePath - La URL o la ruta donde se encuentra el archivo que queremos descargar.
21   * @param {string} fileName - El nombre que queremos darle al archivo cuando lo guardemos en nuestra computadora.
22   *
23   * - Creamos un elemento <a> (un enlace) usando document.createElement('a').
24   * - Le asignamos la URL del archivo a link.href.
25   * - Le decimos cómo queremos que se llame el archivo descargado con link.download.
26   * - Finalmente, simulamos un clic en ese enlace con link.click() para que el navegador comience la descarga.
27   */
28  downloadFile(filePath: string, fileName: string) {
29    const link = document.createElement('a');
30    link.href = filePath;
31    link.download = fileName;
32    link.click();
33  }
34}

```

*DownloadService*

```

17  export class FormComponent implements OnInit {
18    formulario: FormGroup;
19    ascendencias = ascendencias;
20    clases = clases;
21    alineamientos = alineamientos;
22    habilidadesPorClase = habilidadesPorClase;
23    todasLasHabilidades = todasLasHabilidades;
24    habilidadesFijasSeleccionadas: string[] = [];
25
26    caracteristicas = [
27      { key: 'fuerza', label: 'Fuerza' },
28      { key: 'destreza', label: 'Destreza' },
29      { key: 'constitucion', label: 'Constitución' },
30      { key: 'inteligencia', label: 'Inteligencia' },
31      { key: 'sabiduria', label: 'Sabiduría' },
32      { key: 'carisma', label: 'Carisma' },
33    ];
34

```

```

144  private calcularPG(): void {
145    const claseId = this.formulario.get('id_clase')?.value;
146    const constitucion = this.formulario.get('constitucion')?.value;
147
148    if (this.idClaseAnteriorPG !== claseId) {
149      this.idClaseAnteriorPG = claseId;
150      this.basePG = this.obtenerValorBaseClase(claseId, 'dadoGolpe');
151    }
152
153    this.modConstitucion = this.calcularModificador(constitucion);
154    this.formulario.patchValue({ pg: this.basePG + this.modConstitucion });
155  }
156

```

### Ejemplos del formulario y métodos

```

1  // Clases
2  export const clases = [
3    { id: 1, name: 'Picaro', dadoGolpe: 8, description: 'Maestro del sigilo y el engaño.' },
4    { id: 2, name: 'Mago', dadoGolpe: 6, description: 'Manipulador de la magia arcana.' },
5    { id: 3, name: 'Druida', dadoGolpe: 8, description: 'Guardián de la naturaleza con habilidades místicas.' },
6    { id: 4, name: 'Brujo', dadoGolpe: 8, description: 'Hace pactos con seres poderosos para obtener magia.' },
7    { id: 5, name: 'Hechicero', dadoGolpe: 6, description: 'Innato lanzador de hechizos.' },
8    { id: 6, name: 'Explorador', dadoGolpe: 10, description: 'Maestro de la supervivencia y la exploración.' },
9    { id: 7, name: 'Guerrero', dadoGolpe: 10, description: 'Experto en combate y tácticas marciales.' },
10   { id: 8, name: 'Clerigo', dadoGolpe: 8, description: 'Sirviente de una deidad, dotado de poder divino.' },
11   { id: 9, name: 'Paladin', dadoGolpe: 10, description: 'Caballero sagrado con habilidades divinas.' },
12   { id: 10, name: 'Barbaro', dadoGolpe: 12, description: 'Feroz guerrero que entra en un frenesí.' },
13   { id: 11, name: 'Monje', dadoGolpe: 8, description: 'Maestro en artes marciales y disciplinas espirituales.' },
14   { id: 12, name: 'Artifice', dadoGolpe: 8, description: 'Experto en la creación de objetos mágicos y artilugios.' },
15   { id: 13, name: 'Bardo', dadoGolpe: 8, description: 'Entrenedor mágico y habilidoso en muchas áreas.' }
16 ];

```

### Ejemplos de constante con datos estáticos

## 7.7.- Spring

Para comenzar un proyecto con Spring, se suele recomendar el uso de Spring Boot, ya que simplifica considerablemente el proceso de configuración y despliegue de aplicaciones basadas en Spring. Spring Boot proporciona una manera rápida y preconfigurada de crear aplicaciones listas para producción. Este framework se basa en el Spring Framework y favorece la "convención sobre configuración", lo que significa que viene con configuraciones predeterminadas para muchas tareas comunes, reduciendo así la necesidad de configuración manual.

[Fuente - Spring.io](#)

La creación de la parte back del proyecto se realiza de manera guiada en Eclipse IDE, mediante la opción “New -> Spring Starter Project”. Esto permite configurar todos los aspectos del proyecto, e incluir las dependencias necesarias para el mismo. En este caso, las dependencias utilizadas son:

- **Spring-boot-starter-data-jpa:** proporciona integración con JPA (Java Persistence API) para realizar operaciones de acceso a datos en bases de datos relacionales mediante Spring Data JPA.
- **Spring-boot-starter-web:** incluye todo lo necesario para crear aplicaciones web, incluida la configuración automática de Spring MVC y el servidor embebido Tomcat.
- **Spring-boot-devtools:** herramientas de desarrollo para mejorar la experiencia de desarrollo, como recarga automática y otras características que facilitan el trabajo en aplicaciones Spring Boot.
- **Mysql-connector-j:** controlador JDBC para la base de datos MySQL, que permite la conexión y ejecución de consultas SQL en MySQL desde aplicaciones Java.
- **Lombok:** biblioteca para simplificar el código Java mediante anotaciones que generan automáticamente métodos comunes como getters, setters, constructores, y más.

- **Spring-boot-starter-security:** incluye todo lo necesario para añadir seguridad a las aplicaciones Spring Boot, como autenticación y autorización.
- **Jwt:** biblioteca para crear y manejar JSON Web Tokens (JWT), que se utilizan comúnmente en aplicaciones para autenticar usuarios de manera segura.
- **Spring-boot-starter-validation:** proporciona herramientas para validar datos de entrada en aplicaciones Spring Boot, facilitando la implementación de validaciones estándar y personalizadas.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
5     <modelVersion>4.0.0</modelVersion>
6     <parent>
7       <groupId>org.springframework.boot</groupId>
8       <artifactId>spring-boot-starter-parent</artifactId>
9       <version>3.2.4</version>
10      <relativePath /> 
11    </parent>
12    <groupId>SesionCero_Backend</groupId>
13    <artifactId>SesionCero_Backend</artifactId>
14    <version>0.0.1-SNAPSHOT</version>
15    <name>SesionCero_Backend</name>
16    <description>Backend para gestión de personajes de D&D con Spring Boot,
17      incluyendo servicios RESTful y conexión a MySQL para CRUD y lógica de
18      juego</description>
19    <properties>
20      <java.version>17</java.version>
21    </properties>
22    <dependencies>
23      <dependency>
24        <groupId>org.springframework.boot</groupId>
25        <artifactId>spring-boot-starter-data-jpa</artifactId>
26      </dependency>
27    </dependencies>

```

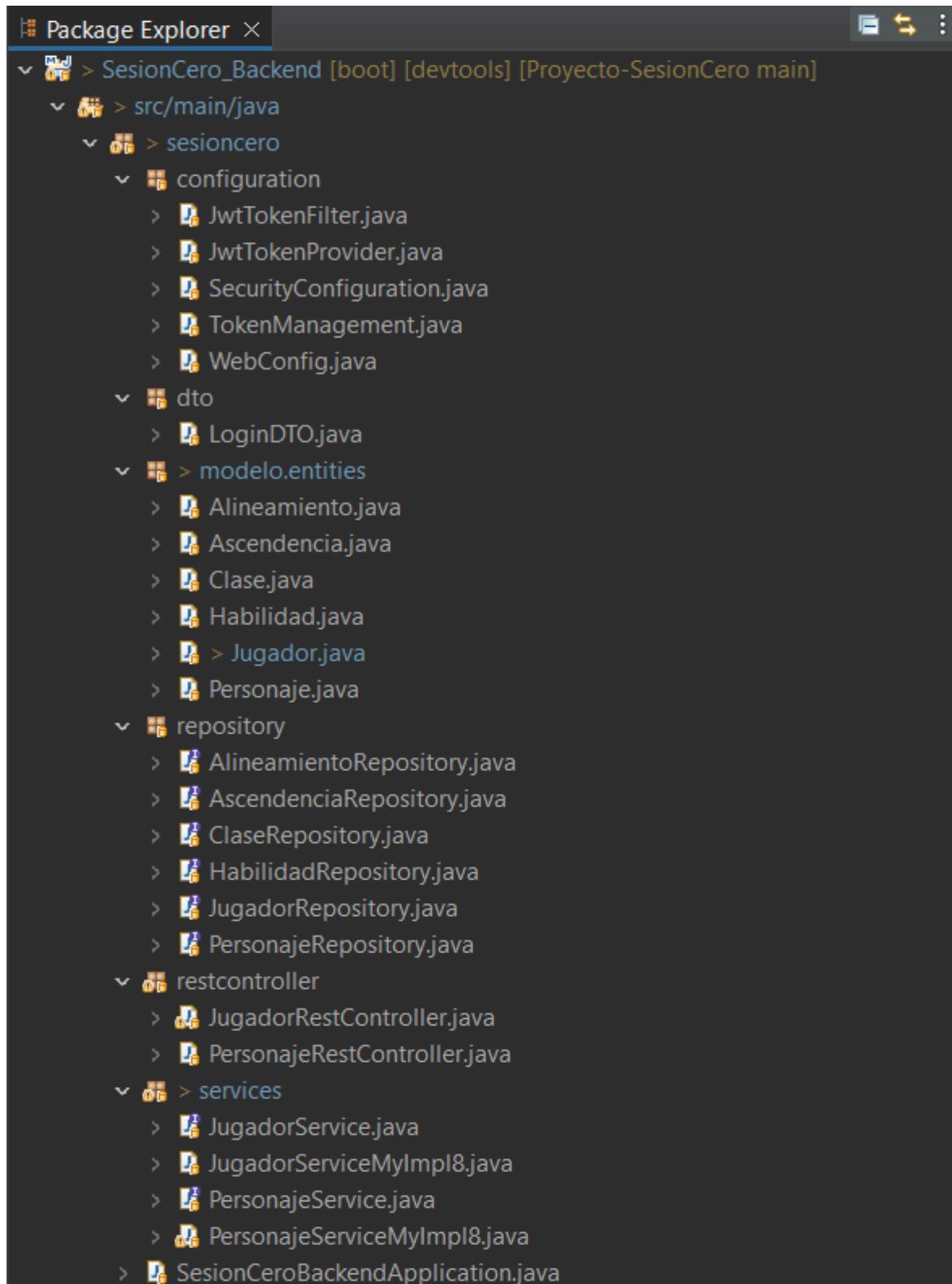
POM.xml

```

1 spring.application.name=SesionCero_Backend
2 server.port=8087
3
4 # para mysql 8 bbdd proyecto_cero
5 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
6 spring.datasource.url=jdbc:mysql://localhost:3306/dungeons_and_dragons?serverTimezone=UTC
7
8 spring.datasource.username=sesioncero
9 spring.datasource.password=sesioncero
10
11 spring.jpa.generate-ddl=false
12 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
13 spring.jpa.show-sql=true
14 spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
15 spring.main.allow-circular-references=true

```

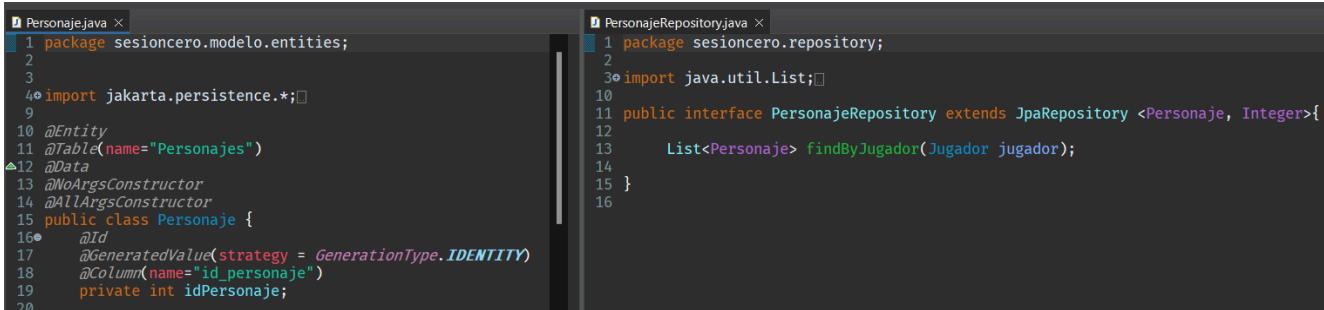
Application.properties



Estructura de paquetes

En el proyecto, se utilizan entidades, servicios y sus implementaciones para organizar y gestionar los datos y la lógica:

- **Entities:** clases que representan las tablas de la base de datos, mapeando los datos a objetos Java. Por ejemplo, la entidad Personaje corresponde a la tabla de "Personajes".
- **Services:** Interfaces que definen las operaciones que se pueden realizar sobre las entidades.
- **ServiceImpl:** implementaciones que contienen el código para realizar las operaciones definidas en los servicios, interactuando directamente con la base de datos mediante Spring Data JPA para operaciones CRUD.
- **Repositories:** interfaces que extienden de JpaRepository, proporcionando métodos predefinidos para operaciones básicas de acceso a datos y permitiendo definir métodos personalizados. Facilitan la interacción sencilla con la base de datos.

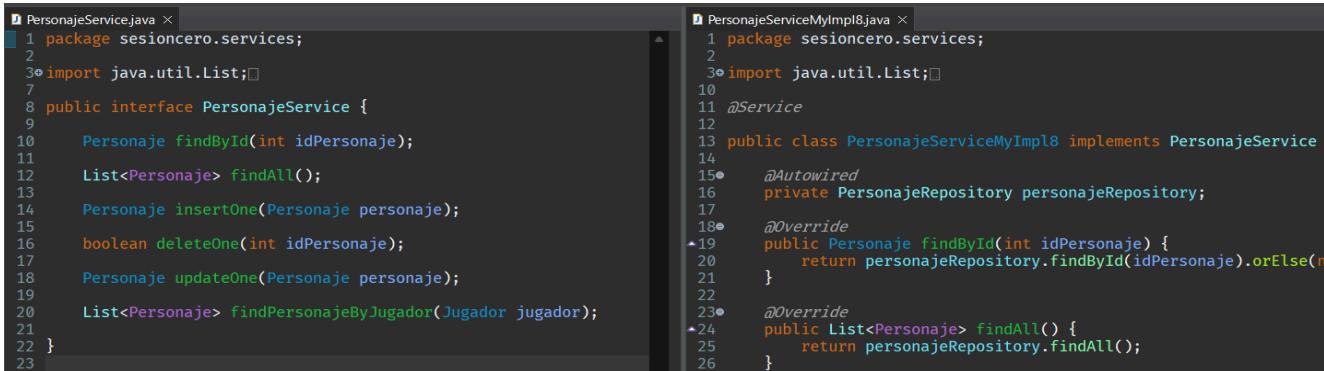


```

  Personaje.java
  package sesioncero.modelo.entities;
  ...
  import jakarta.persistence.*;
  ...
  @Entity
  @Table(name="Personajes")
  @Data
  @NoArgsConstructor
  @AllArgsConstructor
  public class Personaje {
    ...
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id_personaje")
    private int idPersonaje;
  }

  PersonajeRepository.java
  package sesioncero.repository;
  ...
  import java.util.List;
  ...
  public interface PersonajeRepository extends JpaRepository <Personaje, Integer>{
    ...
    List<Personaje> findByJugador(Jugador jugador);
  }

```



```

  PersonajeService.java
  package sesioncero.services;
  ...
  import java.util.List;
  ...
  public interface PersonajeService {
    ...
    Personaje findById(int idPersonaje);
    List<Personaje> findAll();
    Personaje insertOne(Personaje personaje);
    boolean deleteOne(int idPersonaje);
    Personaje updateOne(Personaje personaje);
    List<Personaje> findPersonajeByJugador(Jugador jugador);
  }

  PersonajeServiceMyImpl8.java
  package sesioncero.services;
  ...
  import java.util.List;
  ...
  @Service
  public class PersonajeServiceMyImpl8 implements PersonajeService {
    ...
    @Autowired
    private PersonajeRepository personajeRepository;
    ...
    @Override
    public Personaje findById(int idPersonaje) {
      ...
      return personajeRepository.findById(idPersonaje).orElse(null);
    }
    ...
    @Override
    public List<Personaje> findAll() {
      ...
      return personajeRepository.findAll();
    }
  }

```

Entidad, Repositorio y Servicio para Personajes

Para la comunicación de la parte back con el front, se utiliza un controlador. Un controlador actúa como un intermediario que maneja las solicitudes HTTP entrantes desde el front y envía las respuestas adecuadas. Son esenciales para establecer la comunicación entre el back-end y el front-end.

El controlador se define utilizando la anotación `@RestController` y mapea las solicitudes a métodos específicos mediante `@RequestMapping` o `@GetMapping`, `@PostMapping`, entre otros. Estos métodos manejan las solicitudes HTTP, realizan las operaciones necesarias (como llamadas a servicios o repositorios) y devuelven los datos en formato JSON, que Angular puede consumir fácilmente. Las anotaciones empleadas en este proyecto son:

- **@RestController:** anotación que indica que la clase maneja solicitudes HTTP y devuelve datos en formato JSON o XML.
- **@CrossOrigin:** permite solicitudes de dominios diferentes al especificado, configurando CORS (Cross-Origin Resource Sharing).
- **@RequestMapping:** mapea las solicitudes HTTP a métodos específicos de un controlador, pudiendo configurar la ruta, el método HTTP, y otros atributos.
- **@GetMapping:** mapea solicitudes HTTP GET a un método específico en el controlador, utilizado para recuperar datos.
- **@PostMapping:** mapea solicitudes HTTP POST a un método específico en el controlador, utilizado para crear nuevos recursos.
- **@PutMapping:** mapea solicitudes HTTP PUT a un método específico en el controlador, utilizado para actualizar recursos existentes.
- **@DeleteMapping:** mapea solicitudes HTTP DELETE a un método específico en el controlador, utilizado para eliminar recursos existentes.

Usando como ejemplo el método para el alta de un personaje:

- En Angular, se define un servicio (PersonajeService) que maneja las solicitudes HTTP hacia el back-end. Este servicio utiliza HttpClient para enviar una solicitud POST al servidor cuando se quiere dar de alta un nuevo personaje. El método altaPersonaje envía los datos del personaje al endpoint /alta del controlador REST del back-end.

```

6  @Injectable({
7    providedIn: 'root'
8  })
9  export class PersonajeService {
10
11    private baseUrl = 'http://localhost:8087/personaje';
12
13    constructor(private http: HttpClient) { }
14
15    altaPersonaje(personaje: Personaje): Observable<Personaje> {
16      return this.http.post<Personaje>(`${this.baseUrl}/alta`, personaje);
17    }
  
```

- En el back-end, se define un controlador (PersonajeRestController) que escucha las solicitudes entrantes en el endpoint /personaje/alta. El método altaPersonaje del controlador recibe los datos del personaje enviados desde el front-end, los procesa y persiste los datos en la base de datos mediante el PersonajeService del back.

```

25 @RestController
26 @CrossOrigin(origins = "http://localhost:4200")
27 @RequestMapping("/personaje")
28 public class PersonajeRestController {
29   @Autowired
30   private PersonajeService personajeService;
31
32   @Autowired
33   private JugadorService jugadorService;
34
35   @PostMapping("/alta")
36   public Personaje altaPersonaje(@RequestBody Personaje personaje) {
37     System.out.println("Received Personaje: " + personaje);
38     return personajeService.insertOne(personaje);
39   }
  
```

## 8.- Conclusiones y mejoras del proyecto

---

A continuación, se detallan nuestras principales conclusiones y áreas de mejora identificadas.

### Satisfacción con el Proceso de Desarrollo

A pesar de los desafíos, estamos contentos con cómo se desarrolló el proyecto. Pudimos aplicar lo que aprendimos durante el curso y también adquirimos nuevas habilidades. Usar Spring y Angular nos permitió comprender mejor estas tecnologías y mejorar nuestras capacidades de desarrollo.

### Retos con Spring y Angular

Optar por Spring para el back y Angular para el front presentó varios retos. Al inicio, nuestro conocimiento de Angular era básico, lo que nos obligó a enfrentar una curva de aprendizaje pronunciada. Sin embargo, esta experiencia nos ayudó a mejorar nuestras habilidades en Angular y a aprender a resolver problemas y adaptarnos a nuevas tecnologías. Los desafíos nos fueron útiles para nuestro crecimiento y nos prepararon mejor para futuros proyectos.

### Problemas con el Sistema de Login

Uno de los mayores problemas que enfrentamos fue el sistema de login. Nos costó mucho saltar el login preestablecido de Spring y recibimos errores 403 en cada intento. Pero finalmente y tras muchos esfuerzos conseguimos implementar la autenticación utilizando JSON Web Tokens (JWT).

**Áreas de Mejora:**

- **Optimización del código:** A medida que profundizamos en Angular, podemos refactorizar y optimizar el código para mejorar la legibilidad, el rendimiento y la mantenibilidad.
- **Visualización de imágenes en la ficha de detalles:** Para la ficha de detalles del personaje, hemos guardado la imagen que sube el usuario en base64 en la base de datos, pero no hemos conseguido que se muestre. Es algo que debemos solucionar.
- **Visualización de habilidades en la ficha de detalles:** En la ficha de detalles aún no se muestra la lista de habilidades seleccionadas por el usuario, ya que están configuradas de manera diferente a como se deberían guardar en la tabla de habilidades de la base de datos. Es un punto a mejorar modificando la forma en que se almacenan los detalles de las habilidades extra seleccionadas.
- **Visualización de mensaje de error en la ficha de detalles y ficha de usuario y redirección:** Tanto en la ficha de detalles del personaje como en la del perfil del jugador deberían de mostrarse por pantalla un mensaje de confirmación/error cuando el usuario procede a eliminar dicho personaje/jugador. Se implantará posteriormente para mejorar la experiencia del usuario, así como la redirección pertinente después de ejecutar la acción.
- **Manejo de cierre de sesión tras eliminar un jugador:** Pese a eliminar de manera correcta un jugador desde su perfil, la sesión se mantiene activa. Punto importante a mejorar.

En resumen, aunque enfrentamos varios desafíos, el desarrollo de este proyecto ha sido una experiencia gratificante. Aprendimos a superar dificultades técnicas y ganamos confianza en nuestras habilidades para manejar proyectos complejos utilizando estas tecnologías.

## 9.- Bibliografía

En la elaboración de este proyecto, se utilizaron diversas fuentes tecnológicas para obtener información. A continuación, se detallan las principales fuentes consultadas:

- **Repositorios de GitHub:** Se consultaron múltiples repositorios públicos en GitHub para ejemplos de implementación y mejores prácticas en el uso de Spring y Angular.

Página principal: [GitHub](#)

- **StackOverflow:** Esta plataforma fue fundamental para resolver dudas específicas y problemas técnicos que surgieron durante el desarrollo.

Página principal: [Stack Overflow](#)

- **Material proporcionado por el instituto:** Utilizamos los apuntes, guías y recursos proporcionados por nuestros profesores y el instituto a lo largo del curso.

- **Trabajos realizados en clase:** Los proyectos y ejercicios realizados durante el curso sirvieron como base y referencia para el desarrollo de nuestra aplicación. Estos trabajos nos ayudaron a consolidar nuestro conocimiento y aplicar lo aprendido en un contexto práctico.

## 10.- Anexos

---

Como anexos, se proporcionan los siguientes enlaces con los recursos del proyecto:

**Repositorio del proyecto:** [Proyecto - SesionCero - GitHub](#)

**FigJam:** [Proyecto - SesionCero - FigJam](#)

**Figma:** [Proyecto - SesionCero - Figma](#)