



# Warp Zone Wonders



## Identificación de las necesidades del proyecto

Este proyecto apunta a las necesidades que tienen los aficionados de los videojuegos de asegurarse que los juegos que van a comprar merezcan la pena teniendo en cuenta las opiniones de la comunidad.

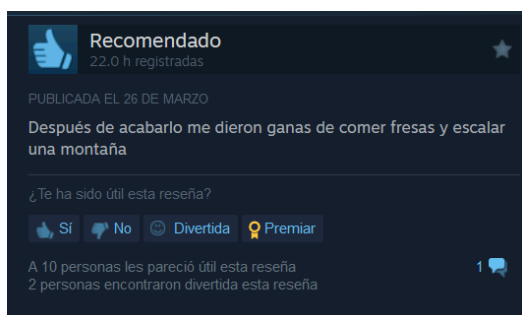
## Breve análisis/comparativa con las alternativas del mercado

Como competidores potenciales podemos encontrar a [Steam](#) e [IGN](#).

IGN suele contratar game journalists, reporteros que se dedican a jugar videojuegos antes de su salida para escribir reseñas, pero al tener tan poco tiempo para jugar antes del lanzamiento puede hacer que sus reseñas sean demasiado apresuradas.

Steam lo hace mejor en comparación ya que son los propios usuarios los que escriben las propias reseñas, por lo que suelen salir más tarde pero si encuentras un usuario que tenga gustos parecidos a los tuyos puedes encontrar nuevos juegos interesantes de forma rápida.

La mejora que me gustaría implementar para competir contra este tipo de reseñas es un sistema de comunidades en el que se pueden buscar reseñas según la comunidad por la que te sientas más identificado.



## **Justificación del proyecto**

Este proyecto tiene grandes ventajas sobre las alternativas ya que al tener reseñas escritas por los propios usuarios le da mucha credibilidad y autenticidad a las reseñas que nos encontraremos.

El filtrado por comunidades nos da un filtrado personalizado muy en conexión con los grupos con los que se identifican los usuarios, lo cual hace que tiendan a ser más activos.

Debido a todo esto mejorará la capacidad de decisión de nuestros usuarios a la hora de comprar juegos nuevos haciendo que ahorren tiempo y dinero.

## **Uso de stack tecnológico**

MySQL (BBDD):

Mi decisión al usar sql se basa en que permite manejar grandes volúmenes de datos de manera eficiente y la capacidad de almacenar imágenes, esto es prioritario en un sistema en el que se necesite manejar una gran cantidad de datos, en nuestro caso una gran cantidad de reseñas.

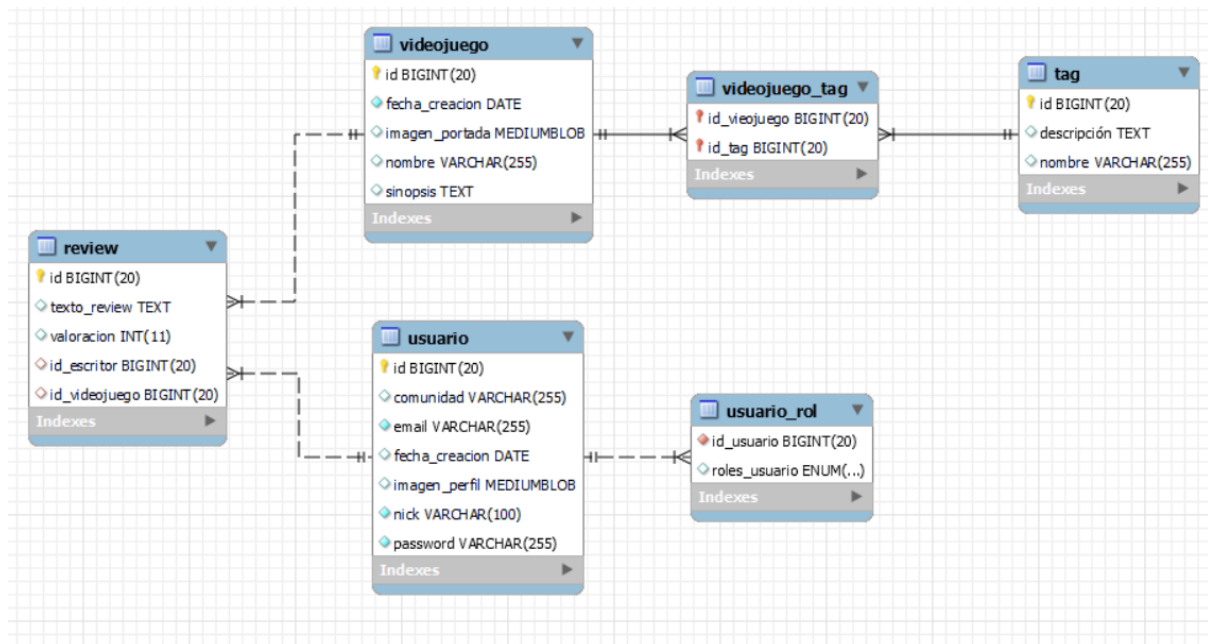
Spring Boot (Backend):

Mi decisión al usar Spring Boot se basa en la integración sencilla con MySQL, las características de seguridad gracias a la configuración de web security basada en tokens y el uso de Java por su programación orientada a objetos.

Angular (Frontend):

Mi decisión al usar Spring Boot se basa en la capacidad de usar componentes para poder reutilizar código de forma eficiente y tiene la ventaja sobre otros modelos basados en JavaScript, ya que al usar TypeScript permite el tipado de los objetos y permite un entendimiento más rápido de los componentes generados previamente.

## Esquema de la BD



## Prototipo en Figma



[Enlace a figma](#)

## Definición API REST

La [documentación de la Api](#) la podemos encontrar en el [repositorio de github del proyecto](#).

Aquí tenemos una tabla de los endpoints de la api:

Metodo	Ruta	Permisos	Descripción
POST	api/v1/auth/login	Pública	Obtenemos el token del usuario a partir de su usuario y contraseña.  Body: <pre>{   "email": "admin@admin.com",   "password": "passwordadmin" }</pre>
POST	api/v1/auth/signup	Pública	Creamos un usuario y obtenemos directamente el token tras la creación.  Body: <pre>{   "nick": "usuario",   "email": "usuario@usuario",   "password": "passwordusuario" }</pre>
POST	api/v1/auth/loginToken	Usuario	Obtenemos un token a partir de la sesión que ya tenemos activa.
GET	api/v1/user/roles	Usuario	Obtenemos los roles del usuario que tenemos en la sesión activa.
GET	api/v1/user	Admin	Obtenemos el listado de todos los usuarios.
GET	api/v1/videojuego/\${nombre}	Pública	Obtenemos los datos en detalle de un videojuego.
GET	api/v1/videojuego	Pública	Obtenemos los datos resumidos de todos los videojuegos, mediante el parámetro opcional nombre podemos filtrar mediante nombre parcial.

POST	api/v1/videojuego	Admin	Endpoint para crear videojuegos.  Body: <pre>{   "nombre": "VideojuegoTest",   "fechaCreacion": "2024-02-19",   "sinopsis": "Sinopsis",   "tags": ["Accion"] }</pre>
GET	api/v1/review	Pública	Obtenemos todas las reseñas sin filtrar
GET	api/v1/review/\${nombre}	Pública	Obtenemos todas las reseñas del videojuego que se encuentre en la url.
POST	api/v1/review	Usuario	Endpoint para crear reseñas.  Body: <pre>{   "nombreUsuario": "admin",   "nombreVideojuego": "Tomb Raider",   "valoracion": "3",   "textoReview": "Ni fu ni fa" }</pre>

## Manual de despliegue

Para desplegar esta aplicación usaremos docker, un paso previo antes de poder usar el docker-compose es compilar la api.

Una forma sencilla de hacerlo es abriendo una consola en la carpeta raíz del proyecto y ejecutando estos comandos:

- `cd src-api`
- `mvn clean`
- `mvn install`

Como paso extra, podemos configurar algunas variables de entorno en el docker-compose, como por ejemplo en nombre de la bbdd o las credenciales del usuario administrador.











```
sql:
  image: mysql:latest
  ports:
    - "3306:3306"
  environment:
    MYSQL_ROOT_PASSWORD: root
    MYSQL_DATABASE: wzw
```

```
api:
  build: src-api
  ports:
    - "8080:8080"
  environment:
    DB_URL: jdbc:mysql://sql:3306/wzw
    DB_USERNAME: root
    DB_PASSWORD: root
    ADMIN_NICK: admin
    ADMIN_EMAIL: admin@admin.com
    ADMIN_PASSWORD: passwordadmin
```

Tras compilar la api en un jar volveremos a la carpeta raíz y ejecutamos estos comandos:

- `docker-compose build`
- `docker-compose up -d`

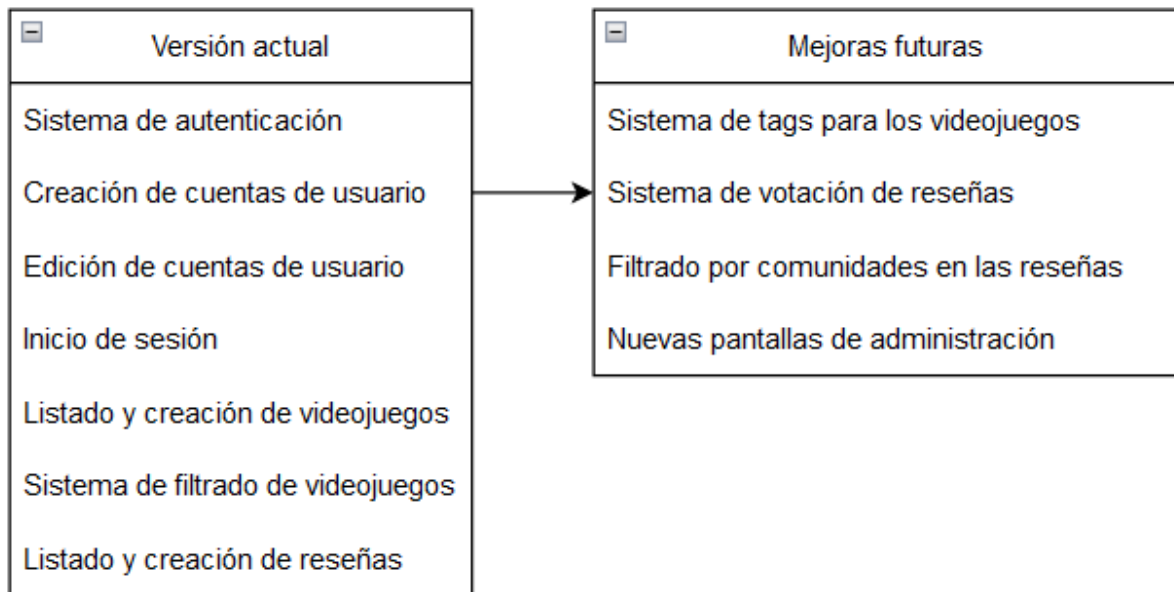
Con esto tendremos desplegada nuestra aplicación en el puerto 4201

▼	 <b>warp-zone-wonders</b>	Running (3/3)		
	<b>api-1</b> cab643a67936 	<a href="#">warp-zone-wonders-api</a>	Running	<a href="#">8080:8080</a> 
	<b>frontend-1</b> 7fd5729f07f6 	<a href="#">warp-zone-wonders-frontend</a>	Running	<a href="#">4201:4200</a> 
	<b>sql-1</b> a51745cd58cd 	<a href="#">mysql:latest</a>	Running	<a href="#">3306:3306</a> 

## Conclusiones y Postmortem

Al desarrollar este proyecto he encontrado muchas dificultades al momento de guardar imágenes en la base de datos, quizás usando servicios de cloud storage hubiese sido más sencillo tratar con las imágenes para no poner tanta carga en la api.

Me ha parecido que he sido muy ambicioso con el alcance del proyecto y esto ha hecho que no estén desarrolladas todas las funcionalidades a las que este proyecto puede llegar, aquí podemos encontrar un análisis de funcionalidades actuales y futuras.



Me parece que el proyecto tiene muchas salidas a futuro ya que se puede aplicar monetización gracias a las comunidades que se pueden formar junto a los videojuegos, algunos ejemplos de esto son:

- Anuncios: la gran cantidad de usuarios traerá anunciantes a nuestra página.
- Eventos en directo: las fechas de salida de videojuegos nuevos hacen períodos en los que los usuarios buscan opiniones sobre los nuevos juegos, podemos aprovechar esta oportunidad para hacer streaming en plataformas como [Youtube](#) o [Twitch](#).
- Merchandising: una vez que nuestra aplicación crezca lo suficiente, algunos usuarios pueden estar interesados en comprar merchandising como camisetas, tazas o fundas para el móvil.

En conclusión, me parece que con suficiente tiempo este proyecto se puede transformar en una gran oportunidad lucrativa y de aprendizaje.