MEMORIA ESCRITA DEL PROYECTO

CFGS Desarrollo de Aplicaciones Web

**Muñardones Games**

**Autor:** Daniel Rodríguez Navarro

**Tutor:** Nombre del tutor

**Fecha de entrega:** 11/06/2024

**Convocatoria:** 2S2024 (semestre de la convocatoria)

**Documentos del proye****cto:** https://drive.google.com/drive/folders/1P-WW0nwC\_vy-4YigNBIPd\_u-eAq2-F\_E?usp=sharing

Índice de contenidos

[**1. Introducción 3**](#_heading=h.1fob9te)

[**1.1. Motivación 3**](#_heading=h.3znysh7)

[**1.2. Abstract 3**](#_heading=h.2et92p0)

[**1.3. Objetivos propuestos (generales y específicos) 3**](#_heading=h.tyjcwt)

[**2. Estado del Arte 4**](#_heading=h.3dy6vkm)

[**3. Metodología usada 5**](#_heading=h.1t3h5sf)

[**4. Tecnologías y herramientas utilizadas en el proyecto 6**](#_heading=h.4d34og8)

[**5. Planificación, Diagnóstico y Contexto Laboral 7**](#_heading=h.2s8eyo1)

[**6. Análisis del proyecto 8**](#_heading=h.17dp8vu)

[**7. Diseño del proyecto 9**](#_heading=h.3rdcrjn)

[**8. Despliegue y pruebas 10**](#_heading=h.26in1rg)

[**9. Conclusiones 11**](#_heading=h.lnxbz9)

[**10. Vías futuras 12**](#_heading=h.35nkun2)

[**11. Bibliografía/Webgrafía 13**](#_heading=h.1ksv4uv)

[**12. Anexos 14**](#_heading=h.44sinio)

[**1.1. Manual de Usuario 14**](#_heading=h.2jxsxqh)

# Introducción

## Motivación

Históricamente, los juegos de mesa surgieron en la antigua Mesopotamia alrededor de los 5.000 años AC. Se les llamó "juegos de pasaje del alma", los cuales eran enterrados junto con sus antiguos dueños para acompañarlos en el más allá.

Según llego a conocerse, estos juegos eran considerados posesiones de alto valor sentimental para los ciudadanos, ya que servían diversión y entretenimiento entre amigos o miembros de una misma familia.

Actualmente los juegos de mesa son una interesante opción de ocio para grupos de personas. No solo estos pueden ser jugados en tu hogar, sino que también existen multitud de locales y cafeterías especializadas donde facilitan el espacio para desarrollar esta actividad.



*https://rotaaldia.com/upload/images/05\_2023/8394\_parchis.jpg*

En lo que se refiere a mis los juegos de tablero siempre han supuesto un papel importante en mi vida y es por eso que me gustaría desarrollar esta idea que tanta ilusión me hace. Cabe mencionar también que mi desarrollo laboral ha tornado a las vertientes de economía e informática por lo que creo que podría mezclar estos ingredientes para crear una aplicación y posteriormente poseer los conocimientos necesarios para constituir y gestionar una empresa.

Por último, me gustaría terminar esta sección con una recomendación de juegos de mesa para todo tipo de usuarios y dificultades.

* **Principiantes:** Risk**,** Survive, Black histories, Carcassonne. Arcadia quest.
* **Intermedios:** Terraforming Mars, Wonders Duels, Coimbra, Mansiones de la locura.
* **Avanzados:** Twilight imperium**,** Civilization, Gloomheaven, Brass.

En la siguiente web se puede ver el ranking de los juegos mejores valorados por los usuarios.

<https://boardgamegeek.com/browse/boardgame?sort=rank>

## Abstract

The goal of this project is to create a web application about a board game shop. This full stack project will consist in a backend with a microservices architecture developed with java spring boot. Persistence is going to be applied with JPA, this will create all our tables in our Postgres SQL database.

On the other hand, we have the frontend part which will be developed with React framework for JavaScript.

In this way, a customer could enter into the web, register, select one or many items to purchase and finally pay them.

## Objetivos propuestos (generales y específicos)

Los objetivos generales tienen como finalidad crear una web amigable para el cliente donde este de una forma cómoda pueda realizar sus compras. Para esto el diseño debe ser el adecuado para todo tipo de usuarios.

Paralelamente, respecto a los objetivos específicos es necesario definir el comportamiento esperado, así como las funciones que debe seguir la aplicación web.

Los objetivos generales son:

1. La **creación de una** **web intuitiva** de fácil acceso para que todos los usuarios puedan realizar correctamente sus pedidos como parte de diseño de UX/UI.
2. Una **buena impresión del cliente**. Si el cliente percibe una grata experiencia de compra es muy posible que vuelvan a realizar más compras en el futuro. Esta es una de las claves para un negocio fructífero.

Los objetivos específicos son:

1. **Acceso a la aplicación:** mediante esta acción los clientes pueden acceder y registrarse en caso de no tener cuenta.
2. **Consultar el catálogo**. el cliente tiene la capacidad de consultar un catálogo, buscar y ver de esta manera los detalles de los productos.
3. **Gestión del carrito de compras**, son todas aquellas acciones por las que los clientes son capaces de agregar, modificar y eliminar productos del carrito de la compra.
4. **Pagar**, todos los clientes podrán finalizar el proceso compra y proceder finalmente al pago de sus artículos.
5. **Editar el catálogo**, el administrador puede añadir además de modificar productos al catálogo.
6. **Administración de usuarios**, el administrador será capaz de editar los datos de los usuarios además de poder también eliminarlos.

# Estado del Arte

# A lo largo de la historia los juegos de mesa han resultado ser una medida de entretenimiento popular entre la sociedad. Su surgimiento se remonta a la antigua Mesopotamia y estos han evolucionado continuamente hasta los juegos modernos de la actualidad los cuales son de diversas categorías: estrategia, wargames, gestión de recursos, etc.

# Los primeros juegos de mesa que se tienen constancia son “Senet” jugado en Egipto el cual se trataba de un tablero con treinta casillas y piezas de madera. Se piensa que se trataba de un juego con carácter religioso que simbolizaba el viaje del alma después de la muerte.

# Por otro lado, en Mesopotamia se jugaba al “Royal Game of Ur” que se jugaba con dados y un tablero de veinte casillas. Este era un juego de estrategia usado para entrenamiento militar y también usado para instruir en comercio.

# Ya en la Edad Media, los más populares cobraron vida en Europa como puede ser el “Ajedrez”, proveniente de la India, y el “Juego de la Oca”, europeo, ambos se convirtieron en los favoritos entre la realeza y nobleza.

# Más tarde en el siglo XIX los juegos de tablero comenzaron a ser producidos con más intensidad y ya eran jugados por la clase media, entre ellos podemos encontrar el “Scrabble” y, como no, el “Monopoly”.

# En la actualidad continúan siendo un gran pasatiempo para multitud de personas siendo además muy accesibles para todo el mundo y con un gran catalogo para todos los gustos.

# La creación y el diseño de los actuales juegos de mesa están influenciados por tecnologías de diseño las cuales permiten crear los componentes con gran detalle. Algunas de estas herramientas son Adobe Ilustrator, TinkerCAD, SelfCAD, Blender. Adicionalmente se ha popularizado el uso de impresoras 3D para la creación de componentes físicos que permiten crear muestras antes de la fabricación en masa.

# **Adobe Illustrador**

-Ventajas:

* Herramientas de diseño vectorial: es una herramienta que permite crear gráficos vectoriales que permite la escalabilidad sin pérdida de calidad.
* Ecosistema Adobe: es capaz de integrarse perfectamente con otros productos de Adobe.
* Herramientas y funciones: ofrece una extensa variedad de herramientas de dibujo, efectos y funcionalidades avanzadas.

-Desventajas:

* Curva de aprendizaje: difícil de aprender al poseer muchas funcionalidades.
* Costo: requiere suscripción.
* Requiere hardware potente: requiere hardware con prestaciones técnicas altas.

**TinkderCAD**

-Ventajas:

* Fácil: posee una interfaz intuitiva que permite a los usuarios crear modelos 3D.
* Basado en navegador: se puede acceder a través de cualquier navegador web.
* Educativo: resulta ser una herramienta excelente para enseñar conceptos básicos de diseño 3D.

-Desventajas:

* Limitado en funciones avanzadas: presenta carencias en funciones avanzadas para el modelaje 3D.
* Dependencia de Internet: se necesita conexión a Internet.
* Opciones limitadas de exportación: sus opciones de exportación son limitadas en comparación con la competencia.

**SelfCAD**

-Ventajas:

* Amplia gama de funciones: combina herramientas de modelado, esculpido y renderizado.
* Fácil: posee una interfaz sencilla y herramientas intuitivas apto para los más novatos.
* Basado en navegador: se puede acceder a través de cualquier navegador web.

-Desventajas:

* Limitado en funciones avanzadas: presenta carencias en funciones avanzadas para el modelaje 3D, aunque algo menos que TrinkerCAD.
* Costo: requiere suscripción.
* Dependencia de Internet: se necesita conexión a Internet.

**Blender**

-Ventajas:

* Código Abierto: es gratuito y está abierto a toda la comunidad.
* Funciones avanzadas: amplia y potente gama para modelado, esculpido, animación, renderizado y simulación.
* Comunidad activa: tiene una comunidad activa que resuelven dudas.

-Desventajas:

* Curva de aprendizaje: difícil de aprender al poseer muchas funcionalidades.
* Requiere hardware potente: requiere hardware con prestaciones técnicas altas.
* Interfaz no intuitiva: la interfaz puede resultar no agradable para los usuarios

# Respecto a los materiales más usados para su fabricación encontramos el cartón como pilar indiscutible, su versatilidad y capacidad para adaptarse a las plataformas lo hace el principal recurso para los fabricantes. La madera y el metal es usado para dotar de más autenticidad a las piezas. Por ultimo. el plástico es otro de los materiales que más podemos encontrar en la mayoría de los componentes y es muy usado en los wargames.

# Por otro lado, en estos últimos años han surgido una gama de juegos de mesa digitales. Existen aplicaciones móviles y plataformas que ofrecen una experiencia de juego online con personas de todo el mundo, ampliando así las posibilidades de crecimiento y amplitud de este mercado.

# Paralelamente aparecieron plataformas de crowfunding como Kickstarter que han revolucionado la distribución y financiación de los proyectos. En estas plataformas se ponen en comunicación a los diferentes diseñadores para mostrar sus proyectos e ideas a los usuarios, y de esta manera estos pueden colaborar mediante donaciones o recomprando el juego.

# Alrededor de esto se han creado una comunidad online, foros y canales de YouTube donde reseñar los juegos de tablero. El foro más famoso es BoardGameGeek donde los usuarios valoran, descubren y discuten sobre ellos. Esto resulta muy útil a nivel de marketing para que los potenciales clientes conozcan nuevas adquisiciones.

Como hemos mencionado antes muchas editoriales están creando versiones digitales de sus juegos para móviles y aplicaciones de escritorio. Esta nueva forma de jugar que se asemeja a un videojuego abre la puerta a nuevos jugadores que se sienten más cómodos con un entorno más digital.

# Metodología usada

Para el desarrollo de este proyecto se ha optado por hacer uso de la metodología Ágil con el fin ofrecer control sobre el ciclo de vida del proyecto al ser una metodología de trabajo muy usada actualmente en el mundo de IT.

Dicha metodología trata sobre un conjunto de buenas prácticas para optimizar el trabajo en equipo y coordinar de mejor manera a los integrantes. Esta metodología se encuentra especializada en la resolución de entornos dinámicos y complejos de manera que se pueden obteniendo resultados con rapidez. La innovación, competitividad, flexibilidad y productividad son cruciales para lograr esto.

Ahora es necesario diferenciar las distintas fases del ciclo de vida del proyecto:

* Planificación: se trata de una fase donde es necesario definir cuáles serán los objetivos, definir los recursos y marcar un plan de trabajo.
* Análisis y diseño: en esta fase procederemos a analizar cada una de las tareas a hacer. Comenzaremos definiendo los casos de uso, posteriormente diseñar el patrón de diseño de la aplicación que en este caso se ha elegido la de microservicios para la parte de backend. Se creará un modelo de datos para la base de datos de PostgreSQL y finalmente se diseñará la parte de frontend con JS y React.
* Implementación: pasemos a la fase de implementación donde se desarrollará la estructura del sistema mediante Java Spring Boot y los componentes y servicios con JS + React y se integrará mediante JPA para la persistencia en BBDD. Para lograr esto con Agile se deberán seguir las instrucciones funcionales y técnicas previamente definidas en las actividades de nuestro tablón.
* Pruebas: una vez hayamos terminado con la implementación daremos paso a las pruebas funcionales, para que de esta manera se pueda garantizar que el sistema cumpla con las necesidades del cliente y tenga un correcto funcionamiento.
* Despliegue: esta fase se corresponde con la explotación de nuestro modelo corriendo la aplicación previamente desde un entorno local/develop/aceptance a un entorno final de producción y, posteriormente, garantizar su correcto funcionamiento.
* Uso y Mantenimiento: en esta fase se hace un seguimiento continuo de la mejora del sistema. Aquí se asegurará el correcto funcionamiento de la aplicación y la mejora continua. En caso se surgir anomalías (bugs), se deberá crear una nueva tarea en el tabón y acto después se corregirán los problemas y se procederá a la actualización de componente.

Las fases definidas se llevarán a cabo mediante sprints de, aproximadamente, dos semanas cada uno. Al principio de cada día se hace una exposición de los puntos realizados ayer, en qué se trabajará hoy y si tenemos algún bloqueo con actividad. Al final de cada sprint se realizará una revisión y se reajusta el plan de trabajo para el siguiente sprint. Con esta metodología podemos hacer un seguimiento personal diario que de tal forma se podrán identificar rápidamente los problemas que puedan surgir, también se garantiza una mayor eficiencia en el desarrollo del proyecto.

Hay que tener en cuenta que al surgir errores de estimaciones en las actividades es posible que cambie el scope y con ello no se pueda llegar a los tiempos de entrega que se están especificando. En caso de que esto ocurra se deberá ampliar el tiempo de desarrollo pactando previamente el número de sprints adicionales requeridos para llevarlo a cabo.

# Tecnologías y herramientas utilizadas en el proyecto

A continuación, se listan las tecnologías elegidas en el proyecto:



**Java** es un conocido lenguaje de programación de alto nivel y orientado a objetos que se utiliza para desarrollar aplicaciones tanto para dispositivos móviles como para sistemas empresariales. Ha evolucionado desde sus humildes comienzos hasta impulsar una gran parte del mundo digital actual, ya que es una plataforma fiable en la que se crean muchos servicios y aplicaciones. Los nuevos e innovadores productos y servicios digitales diseñados para el futuro también siguen basándose en Java.

En el proyecto se establecerá Java como lenguaje de programación principal de desarrollo para la parte de backend. Gracias a que posee una amplia variedad de librerias, frameworks y es compatible con el resto de teconologias a elegidas.



Java **Spring Boot** es una herramienta que hace que el desarrollo de aplicaciones web y microservicios con Spring Framework sea más rápido y fácil a través de tres funcionalidades principales:

* Configuración automática.
* Gestión de dependencias.
* Arranque rápido.
* Monitorización y análisis.

Estas características funcionan juntas para brindarle una herramienta que le permite configurar una aplicación basada en Spring con una configuración y preparación mínimas.

Su función será la de simplificar el proceso de desarrollo de la aplicación, automatizando la configuración, la gestión de dependencias y reduciendo los tiempos de arranque.



**PostgreSQL** se trata de un sistema de gestión de bases de datos relacional (aunque sería posible también no relacionales) de código abierto que se utiliza para almacenar y gestionar grandes volúmenes de datos. Nació a mediados de la década de 1980 a partir de Ingres, pero la primera versión al público estuvo disponible en el año 1989.

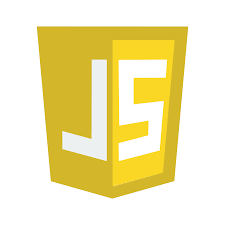
Como SGBD su función en la tienda de juegos de mesa será la de almacenar la información sobre los juegos de mesa disponibles: nombre, descripción y precio.



**JPA** es una especificación que indica cómo se debe realizar la persistencia (almacenamiento) de los objetos en programas Java. Fíjate en que destaco la palabra "Especificación" porque JPA no tiene una implementación concreta, sino que, como veremos enseguida, existen diversas tecnologías que implementan JPA para darle concreción.

JPA forma parte de Java empresarial desde su versión 5, en el año 2006. Desde entonces han aparecido varias versiones de la especificación, que sigue el proceso de la comunidad Java.

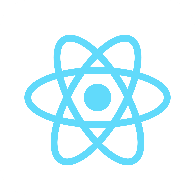
Usaremos JPA para realizar de almacenamiento de objetos en la base de datos de PostgreSQL, proporcionando una capa de abstracción entre la base de datos y el código, ofrecerá también independencia a la base de datos y un mapeo objeto-relacional.



**JavaScript** se trata de uno de los lenguajes de programación más usados actualmente en entorno web, siendo multiplataforma y contando con gran abanico de librerías. Se caracteriza por ser versátil, sencillo y de tipado débil.

Cabe destacar su compatibilidad con todos los navegadores y que sus scripts pueden ejecutarse en el navegador y no en el servidor.

Haremos uso de JS para crear la estructura de nuestro proyecto al ser un lenguaje orientado a frontend.



**React** es una librería de JS que se usa para la creación de interfaces de usuarios declarativas, que se ha vuelto muy popular al ser muy creativa. Varios de los elementos que hacen posible esto son: componentes, estados y desarrollo declarativo, virtual DOM, JSX, unidireccionalidad de datos y herramientas complementarias.

Su función en el proyecto será la de facilitar el proceso de desarrollo como por ejemplo creando componentes y servicios.

# Planificación, Diagnóstico y Contexto Laboral

Pasemos ahora a la planificación de las distintas tareas que se han diseñado previamente según nuestra metodología de trabajo Agile y constara de dos sprints de dos semanas cada uno. Para ello vamos a hacer eso de un diagrama de Gantt:

1. Análisis de requisitos y diseño de la arquitectura de microservicios.
2. Diseño e implementación de la base de datos con PostgreSQL.
3. Desarrollo del backend utilizando Java Spring Boot y JPA.
4. Desarrollo del frontend utilizando JS y React.
5. Corrección de bugs.
6. Pruebas funcionales.
7. Mejora del formato del código.
8. Documentación del proyecto.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **16/04/2023** | **17/04/2023** | **18/04/2023** | **19/04/2023** | **20/04/2023** | **21/04/2023** | **22/04/2023** | **23/04/2023** | **24/04/2023** | **25/04/2023** | **26/04/2023** | **27/04/2023** | **28/04/2023** | **29/04/2023** | **30/04/2023** | **1/05/2023** | **2/05/2023** | **3/05/2023** | **4/05/2023** | **5/05/2023** | **6/05/2023** | **7/05/2023** | **8/05/2023** | **9/05/2023** | **10/05/2023** | **11/05/2023** | **12/05/2023** | **13/05/2023** | **14/05/2023** | **15/05/2023** | **16/05/2023** |
| Act |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **01** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **02** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **03** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **04** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **05** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **06** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **07** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **08** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

No obstante, tras completar los dos sprints se ha optado por ampliar el tiempo de desarrollo pues no pudo ser suficiente al no definir correctamente las tareas en la fase de analisis Agile. Es por este motivo que se contará con dos semanas más de desarrollo.

De esta forma quedaría ahora así reflejado:

1. Desarrollo del backend utilizando Java Spring Boot y JPA.
2. Desarrollo del frontend utilizando JS y React.
3. Corrección de bugs.
4. Pruebas funcionales.
5. Documentación del proyecto.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **17/05/2023** | **18/05/2023** | **19/05/2023** | **20/05/2023** | **21/05/2023** | **22/05/2023** | **23/05/2023** | **24/05/2023** | **25/05/2023** | **26/05/2023** | **27/05/2023** | **28/05/2023** | **29/05/2023** | **30/05/2023** | **31/05/2023** | **01/06/2023** | **02/06/2023** | **03/06/2023** | **04/06/2023** | **05/06/2023** | **06/06/2023** | **07/06/2023** | **08/06/2023** | **09/06/2023** | **10/06/2023** | **11/06/2023** | **12/06/2023** | **13/06/2023** | **14/06/2023** | **15/06/2023** |
| Act |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **01** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **02** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **03** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **04** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **05** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Vemos por tanto que con este diagrama de Gantt podemos organizar de una manera practica el desarrollo de las actividades además de ver una trazabilidad de ellas.

Una vez planificado el tiempo de actuación, debemos aplicar un estudio previo a la explotación empresarial donde podamos ver el análisis externo e interno de la empresa (fortalezas, oportunidades, debilidades y amenazas). Es por ello que se definirá a continuación un análisis DAFO



Ahora sí podemos concluir que, aunque existen una gran barrera de entrada al existir grandes competidores en el sector seguiría siendo atractivo entrar puedas la demanda de este tipo de productos aún está en auge

# Análisis del proyecto

Los requisitos funcionales y no funcionales que se deben cumplir en el proyecto de la tienda de juegos de mesa son los siguientes:

**Requisitos funcionales:**

* Registro de usuarios: la aplicación debe permitir a los usuarios registrarse en la tienda mediante un formulario que incluya campos como nombre, email y password.
* Inicio de sesión: una vez que el usuario se haya registrado, debe poder iniciar sesión en la tienda con su correo electrónico y contraseña.
* Búsqueda de juegos: la aplicación debe permitir a los usuarios buscar juegos por nombre o categoría.
* Compra de juegos: los usuarios deben poder comprar juegos seleccionando el juego deseado y realizando el pago.
* Gestión de pedidos: los usuarios deben poder ver el historial de pedidos y el estado actual de los mismos.
* Gestión de stock: la aplicación debe mantener un registro actualizado del stock de juegos disponibles para la venta.
* Gestión de usuarios: la aplicación debe permitir a los administradores gestionar los usuarios registrados, incluyendo la eliminación de cuentas de usuario.
* Administración de juegos: los administradores deben poder agregar, editar y eliminar juegos de la tienda.

**Requisitos no funcionales:**

* Seguridad: la aplicación debe ser segura y proteger los datos de los usuarios, incluyendo la información de pago y las contraseñas.
* API: vamos a usar un api RestFull (aunque más bien se define como un protocolo) que nos permita consumir nuestros endpoints.
* Disponibilidad: la aplicación debe estar disponible para su uso en todo momento de manera que no cese la actividad empresarial
* Testing: se deben de implementar diferentes test para comprobar la calidad del código así ahorrar en costes futuros de mantenimiento, siendo igual alcanzar una cobertura de al menos un 80% de líneas de código.
* Manuales y guías: se debe acompañar a la aplicación con un manual de instalación, así como una guía de usuario.

Ya definidos los requisitos funcionales y no funcionales, para lograr una mayor organización de lo que vamos a representar mediante código, es crucial representar mediante diagramas relacionales las entidades que la forman y cómo se comportan entre ellas.

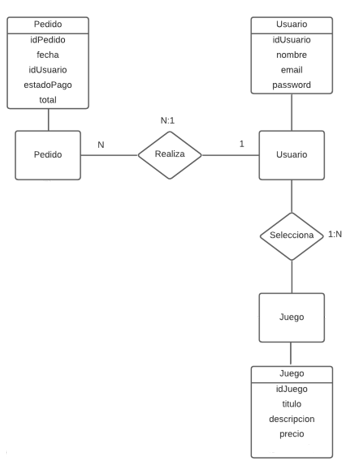
En lo que a BBDD relacional se refiere, debemos crear un diagrama **entidad-relación.** Este tipo de diagrama se caracteriza por relacionar las diferentes entidades que la componen con la cardinalidad que presentan. En nuestro caso vamos a definir cuatro entidades: Usuario, Pedido y Juego.

La cardinalidad se presenta de esta manera:

* La entidad **Pedido** posee una relación de uno a muchos con la entidad Usuario, ya que un usuario puede tener muchos pedidos, pero cada pedido sólo puede estar asociado a un único usuario.
* Por último, la entidad **Usuario** posee una cardinalidad de uno a muchos con Juego, pues cada usuario puede tener varios juegos, pero ese juego solo lo puede tener un usuario.

Cada entidad posee asociados un conjunto de atributos representados en las tablas mostradas.

Por ende, el diagrama entidad-relación quedaría de la siguiente manera:



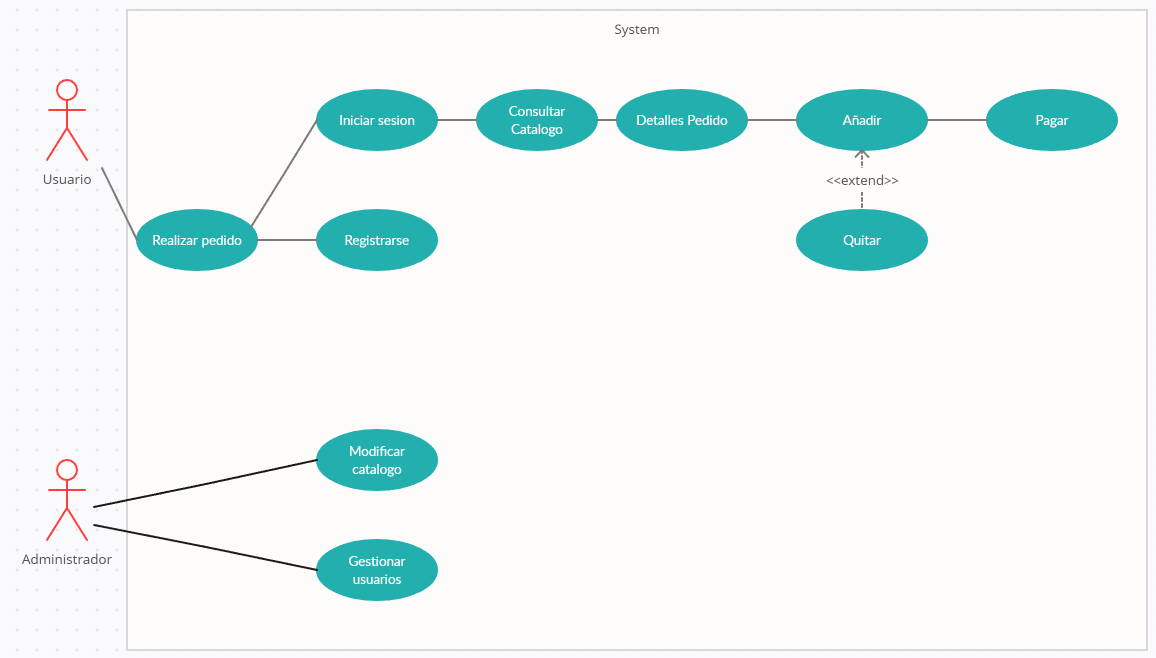
Ahora pasemos a definir el diagrama **UML** el cual nos ayudará a modelar sistemas de diversas formas con los actores y funciones de nuestra aplicación. Estos diagramas permiten de manera grafica conocer un proyecto complejo. También son usados para modelas flujos de trabajo y procesos empresariales.

Podemos encontrar dos tipos de diagramas UML: los estructurales y los de comportamiento.

**Estructurales**: diagrama de objetos, diagrama de componentes, diagrama de estructuras compuestas, diagrama de implementación, diagrama de paquetes, diagrama de clases.

**Comportamiento**: diagrama de actividades, diagrama de comunicación, diagrama de interacciones, diagrama de secuencia, diagrama de máquina de estados, diagrama de temporización, diagrama de casos de uso.

En nuestra aplicación se ha optado por usar uno de tipo comportamiento: casos de uso. Con él esperamos relacionar los diferentes usuarios o actores dentro de un sistema para así representar gráficamente las actividades que podrán llevar a cabo.



Tenemos a los actores: Usuario y Administrador. Las funciones que realizan son:

* Realizar pedido **(Usuario)**
* Registrarse **(Usuario)**
* Iniciar sesión **(Usuario)**
* Consultar Catálogo **(Usuario)**
* Detalles Pedido **(Usuario)**
* Añadir **(Usuario)**
* Quitar **(Usuario)**
* Pagar **(Usuario)**
* Modificar catálogo **(Administrador)**
* Gestionar usuarios **(Administrador)**

**Casos de Uso para Usuario para realizar un pedido**

1. Registrarse: crear una nueva cuenta en la aplicación.

2. Inicia sesión: accede con sus credenciales a su cuenta

3. Consulta Catálogo: elige un artículo de entre todos los que se encuentran en la lista

4. Detalles Pedido: puede ver las propiedades de los artículos seleccionados

5. Añadir: puede añadir artículos al carrito de la compra

6. Quitar: puede eliminar artículos de su carrito de la compra

**Casos de Uso para Administrador**

1. Modificar catálogo: puede crear, modificar o eliminar artículos del catalogo
2. Gestionar usuarios: opciones de configuración para editar o eliminar usuarios

Pasemos ahora a crear los diagramas donde estableces los nombres de las clases e interfaces con los atributos y métodos.

Diagrama de clase para entidad Juego:

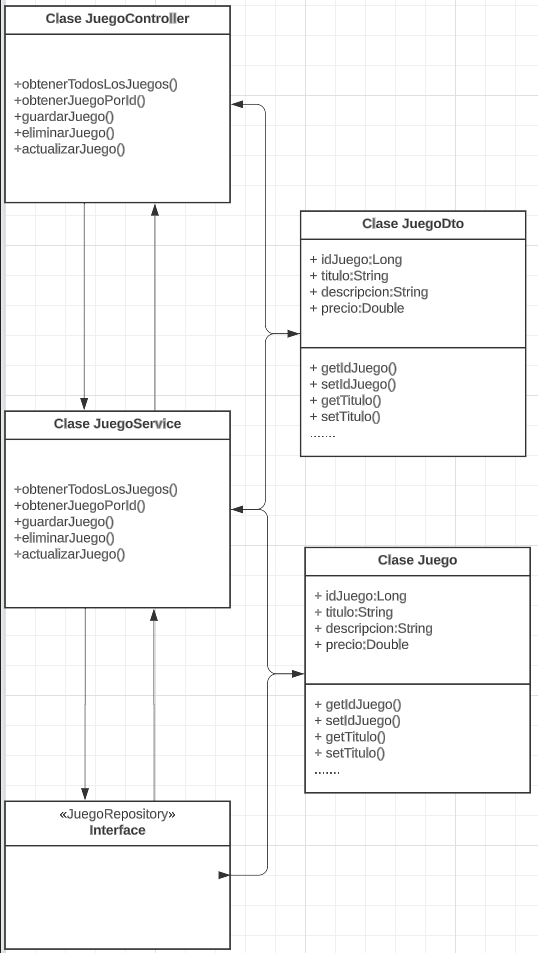
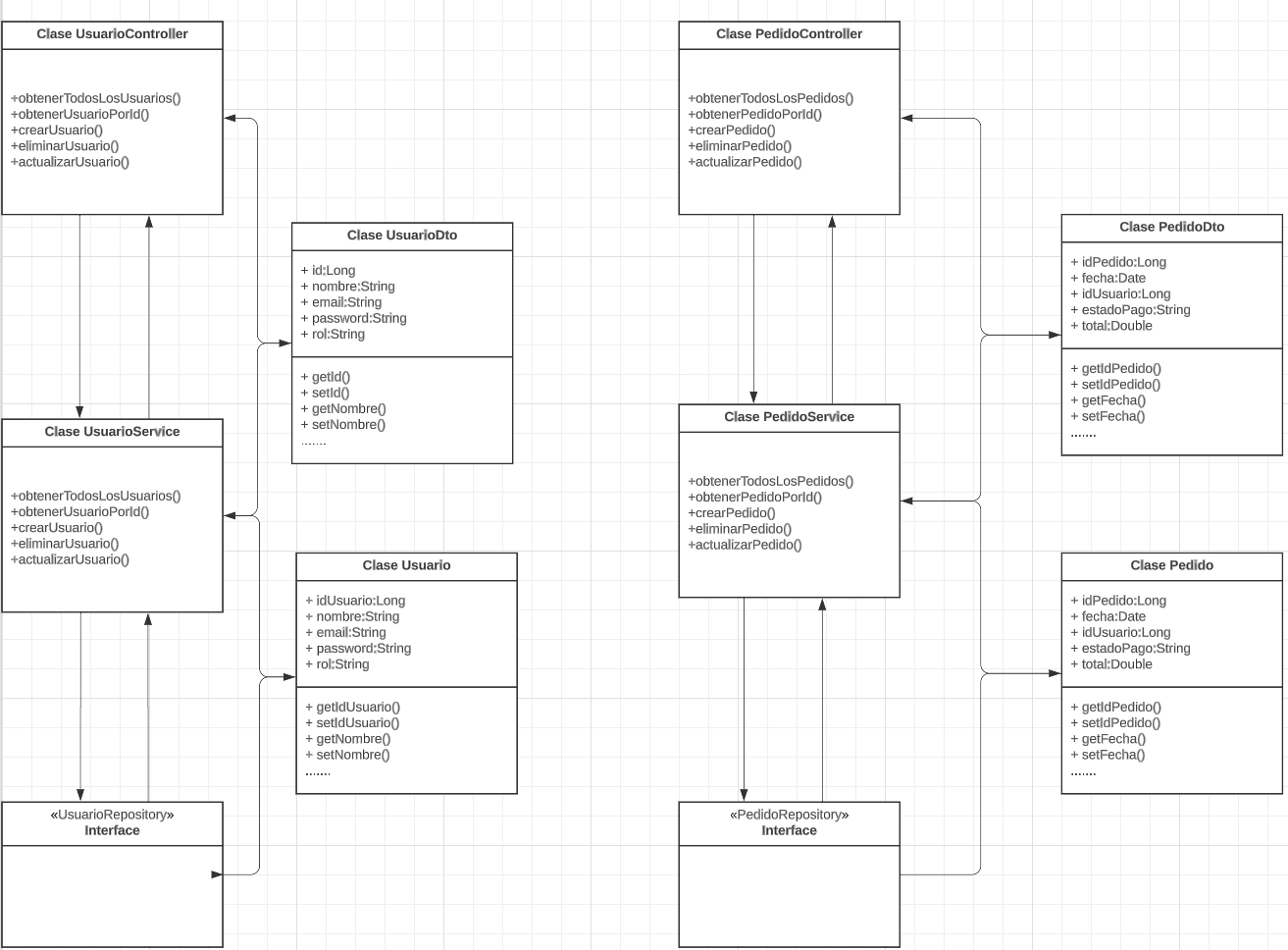


Diagrama de clase para entidades Usuario y Pedido:

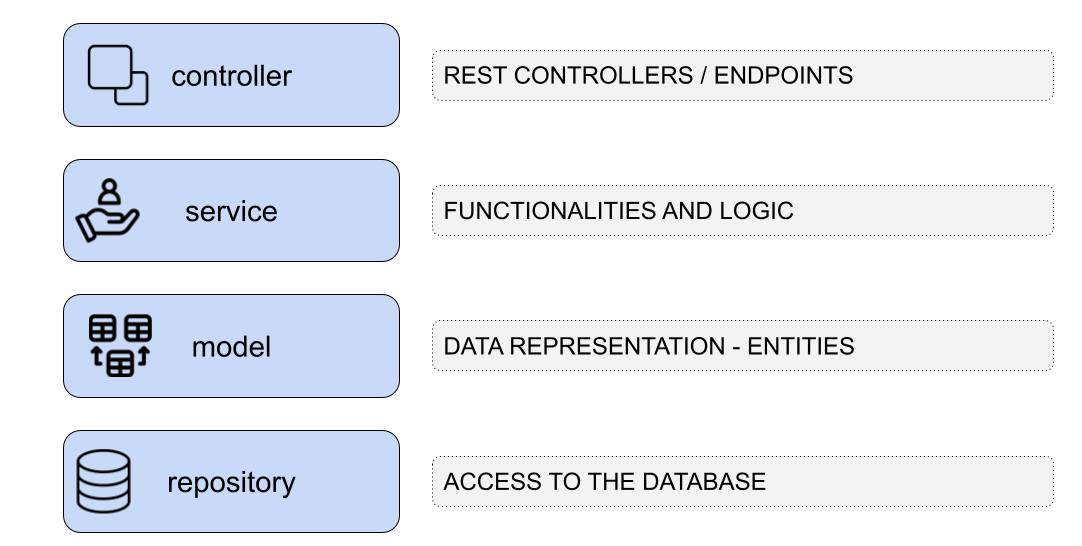


# Diseño del proyecto (6-14 páginas)

Pasemos ahora a hablar sobre el diseño estructural de la aplicación. En primer lugar, echemos un primer vistazo al patrón de diseño elegido: microservicios. Este tipo de arquitectura se basa en tres capas principales: **Controller**, **Service** y **Repository**.

Como sus principales ventajas podemos hablar de:

* Gran capacidad de ajuste: a medida que aumenta la demanda de ciertos servicios puede realizar implementaciones en distintos servidores e infraestructuras.
* Capacidad de recuperación: si un componente falla no debe alterar el funcionamiento de los demás al estar más desacoplado.
* Facilidad de implementación: supone un avance respecto a una arquitectura monolítica al ser modulares y más pequeñas.
* Accesibilidad: resulta más fácil de entender y actualizar para el desarrollador.
* Conexión de los microservicios: se enlazan entre ellos a través de una API gateway.



[*https://gustavopeiretti.com/estructura-de-paquetes-spring-boot/spring-boot-estructura-paquetes.png*](https://gustavopeiretti.com/estructura-de-paquetes-spring-boot/spring-boot-estructura-paquetes.png)

En nuestro caso se han creado tres **entidades**: Usuario, Juego y Pedido.

# Despliegue y pruebas

Para asegurarnos de que la aplicación cumple con las funcionalidades esperadas debemos plantear una serie de casos de uso y hacer unas pruebas de caja negra. Este tipo de pruebas se basan en que dado un input definimos un comportamiento esperado.

Resulta necesario también especificar un plan de actuación que determine como se gestionará la aplicación en un entorno de producción:

1. Instalación y creación de BBDD en PostgreSQL
2. Lanzar la aplicación en Backend y en Frontend
3. Verificación del correcto funcionamiento post-despliegue
4. Establecer herramientas de monitorización
5. Servicio de incidencias

A continuación, pasemos a definir el plan de pruebas a través de los casos de uso que previamente hemos mencionado.

1. Usuario puede crear cuenta e iniciar sesión en la plataforma.
2. Usuario desde el Home Page puede acceder al catálogo y añadir un juego al carrito.
3. Usuario puede visualizar su pedido y eliminar juegos de su carrito de compras.
4. Usuario puede cerrar sesión.
5. Administrador puede crear una cuenta, editarla, eliminarla y obtener una lista de usuarios
6. Administrador puede crear un juego, actualizarlo, eliminarlo y obtener una lista de todos los juegos y un juego por id.

Primer caso de uso:

**Objetivo probado:** Usuario puede crear cuenta e iniciar sesión en la plataforma.

**Input**: El usuario pueden ver la pantalla de login con los campos Email y Password y sus correspondientes cajas, además puede ver los botones Login y Register. Al pulsar el botón Register se carga una ventana nueva donde deben aparecer los campos Nombre, Email y Password que al rellenarlos y pulsar el botón Register debe mostrar el mensaje “Usuario Registrado”. Acto seguido si volvemos a la ventana Login podemos acceder ingresando las credenciales.

**Output**: aparecen todos los campos, los botones te redirigen a las ventanas esperadas y el usuario se crea correctamente en la base de datos.

Segundo caso de uso:

**Objetivo probado:** Usuario desde el Home Page puede acceder al catalogo y añadir un juego al carrito de la compra.

**Input**: En la Home Page vemos una imagen situada en el centro, debajo el botón cerrar sesión y los botones Home, Catalogo, Carrito y Editar Cuenta en el header. Pulsando el botón Catalogo es redirigido a una nueva pagina donde aparecen todos los juegos disponibles. Para cada uno de ellos aparece el nombre, descripción y precio, además de un botón Añadir al carrito. Al pulsarlo es añadido al carrito de la compra.

**Output**: al pulsar sobre el botón Catalogo redirige correctamente, aparece una lista de juegos con la información de cada uno. Cuando pulsamos sobre el botón Añadir al carrito y luego nos desplazamos con el botón Carrito estos aparecen en el pedido.

Tercer caso de uso:

**Objetivo probado:** Usuario puede visualizar su pedido y eliminar juegos de su carrito de compras.

**Input**: En la ventana Carrito deben aparecer los ítems previamente añadidos mostrando su información y cada uno tiene un botón Eliminar. Además, puede ver el total a pagar y se modificará en función de si añadimos/eliminamos ítems.

**Output**: se muestran los ítems correctamente, el botón eliminar funciona satisfactoriamente, el total hace la suma correcta del pedido.

Cuarto caso de uso:

**Objetivo probado:** Usuario puede cerrar sesión.

**Input**: Al pulsar sobre el botón cerrar sesión este es sacado de la aplicación.

**Output**: al pulsar sobre el botón nos cierra la sesión correctamente.

Quinto caso de uso:

**Objetivo probado:** Administrador puede crear una cuenta, editarla, eliminarla y obtener una lista de todos los usuarios

**Input**: En la Home Page vemos situado en el header el botón Editar Cuenta, en el podemos ver los campos ID Usuario, Nombre, Email y Password. Al rellenarlos y pulsar el botón Crear usuario nos muestra el mensaje “Se ha creado el usuario”, podemos modificar los datos de una cuenta pulsando el botón Actualizar usuario nos devuelve “Se ha actualizado el usuario”- Podemos eliminar un usuario proporcionando una id y nos arroja el mensaje “Eliminado usuario con ${id}”. Se puede obtener una lista de todos los usuarios registrados al pulsar sobre el botón Obtener todos los usuarios.

**Output**: el botón Crear usuario crea el registro en la BBDD, el botón Actualizar usuario actualiza en BBDD los campos con la información introducida, el botón Eliminar usuario borra el registro correctamente en la BBDD y el botón Obtener todos los usuarios muestra correctamente todos los registros de la tabla usuario.

Sexto caso de uso:

**Objetivo probado:** Administrador puede crear un juego, actualizarlo, eliminarlo y obtener una lista de todos los juegos y un juego por id.

**Input**: En Catalogo vemos situado en el header el botón Editar Catalogo, en el podemos ver los campos ID Juego, Titulo, Descripción y Precio. Al rellenarlos y pulsar el botón Crear juego nos muestra el mensaje “Juego creado”, podemos modificar los datos de una cuenta pulsando el botón Actualizar usuario nos devuelve “Juego actualizado”- Podemos eliminar un usuario proporcionando una id a través del botón Eliminar juego y nos arroja el mensaje “Juego ${id} eliminado”. El botón Obtener juego por ID nos devuelve el juego deseado proporcionando una id. Se puede obtener una lista de todos los juegos al pulsar sobre el botón Obtener todos los juegos.

**Output**: el botón Crear juego crea el registro en la BBDD, el botón Actualizar juego actualiza en BBDD los campos con la información introducida, el botón Eliminar juego borra el registro correctamente en la BBDD, el botón Obtener juego por ID trae la información respectiva del usuario requerido, el botón Obtener todos los juegos muestra correctamente todos los registros de la tabla usuario.

# Conclusiones

Llevando a cabo el desarrollo hemos aprendido sobre uno de los patrones de diseño más utilizados en los proyectos de desarrollo reales del mundo laboral.

Por otro lado, en la parte de backend hemos podido profundizar en el sistema de anotaciones que ofrece Java Spring boot, así como las que usa JPA para crear las tablas de forma automatizada en nuestra base de datos. Paralelamente, en la parte de frontend se ha intentado hacer un diseño actual con React.js a la hora de desarrollo de los diferentes componentes que recibe la aplicación.

A nivel de estructura en la parte de back hay ligeros aspectos a mejorar como podría ser la creación de casos de uso para los diferentes endpoints. En la parte de frontend sería necesario mejorar la parte de acceso a la aplicación para personas con discapacidad.

Como aprendizaje me gustaría recalcar la mejora en el diseño del archivo App.css

Respecto a las dificultades encontradas, las derivadas del desarrollo de los componentes y servicios.

Como conclusión investigar y profundizar en las tecnologías usadas ha resultado ser muy interesante de cara a que son frameworks y herramientas las cuales son de las más usadas actualmente en el mundo laboral y pienso que pueden blindar de muy buena manera mi valor como trabajador.

# Vías futuras

Tras finalizar el desarrollo se han detectado aspectos a mejorar y características, que, por motivo de una incorrecta estimación del tiempo necesitado, no se han podido completar.

Estos son los siguientes:

Las medidas que no se cumplieron fueron:

Uno de los objetivos no cumplidos sería la implementación de la plataforma final de pago.

Mejora del diseño UX/UI.

A mejorar:

Implementación de autenticación mediante un sistema de roles gestionado con Spring Security y Keycloak por su facilidad de uso y manipulación. Por ejemplo, se podrían restringir la funcionalidad (endpoints) de los usuarios con la anotación @Secured o @RolesAllowed que se configura en la clase con la anotación @Configured y @EnableGlobalMethodSecurity en base a los diferentes usuarios como Admin, Regular…etc.

Optimización de la aplicación para dispositivos móviles y tablets.

Expansión internacional, lo que conllevará traducir a varios idiomas el contenido web.

Otro tema interesante sería incluir un chat de IA para guiar en su compra al usuario, así como actuar como soporte técnico en caso de necesidad.

# Bibliografía/Webgrafía

Pérez, J. (2021, 15 de marzo). El mercado de los juegos de mesa en España sigue fuerte. La Publicidad. https://lapublicidad.net/el-mercado-de-los-juegos-de-mesa-en-espana-sigue-fuerte/

Rodríguez, A. (2023, 18 de junio). ¿Qué es Scrum? Proyectos Ágiles. https://proyectosagiles.org/que-es-scrum/

Oracle Corporation. (s.f.). ¿Qué es Java? Ayuda de Java. https://www.java.com/es/download/help/whatis\_java.html

Rivas, E. (s.f.). Servidor PostgreSQL. Blog de Infranetworking. <https://blog.infranetworking.com/servidor-postgresql/>

IBM Corporation. (s.f.). Java Spring Boot. IBM. https://www.ibm.com/mx-es/topics/java-spring-boot

CampusMVP. (s.f.). La API de persistencia de Java: ¿Qué es JPA? JPA vs. Hibernate vs. EclipseLink vs. Spring JPA. CampusMVP https://www.campusmvp.es/recursos/post/la-api-de-persistencia-de-java-que-es-jpa-jpa-vs-hibernate-vs-eclipselink-vs-spring-jpa.aspx

Miro. (s.f.). ¿Qué es un diagrama UML? Miro. https://miro.com/es/diagrama/que-es-diagrama-uml/

Baeldung. (s.f.). Baeldung. https://www.baeldung.com

# Anexos

# Manual de Usuario

