UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de un portal web destinado a la compraventa de videojuegos

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Daniel Sánchez López

Tutor: Sergio Sáez Barona

Curso 2020-2021

Resum

La contínua evolució d'Internet i de les tecnologies de la informació han afavorit l'auge del comerç electrònic. Hui dia, un dels seus màxims exponents són les pàgines de compravenda d'articles de segona mà, que ofereixen més possibilitats i una major flexibilitat que les botigues de segona mà convencionals. A partir d'això, es planteja com TFG la creació d'un portal dedicat a la compravenda de videojocs de segona mà. Durant aquesta memòria s'explicarà tot el procés seguit per a desenvolupar aquesta aplicació. En ella, els usuaris podran registrar els videojocs que ja no desitgen per a vendre'ls i, al seu torn, podran buscar videojocs en els quals estiguen interessats – a través dels múltiples filtres que inclou l'aplicació – per a la seua posterior sol·licitud i compra.

Paraules clau: usuari, videojoc, sol·licitud, compra, framework, aplicació

Resumen

La continua evolución de Internet y de las tecnologías de la información han favorecido el auge del comercio electrónico. Hoy en día, uno de sus máximos exponentes son las páginas de compraventa de artículos de segunda mano, que ofrecen más posibilidades y una mayor flexibilidad que las tiendas de segunda mano convencionales. A partir de ello, se plantea como TFG la creación de un portal dedicado a la compraventa de videojuegos de segunda mano. Durante esta memoria se explicará todo el proceso seguido para desarrollar dicha aplicación. En ella, los usuarios podrán registrar los videojuegos que ya no deseen para venderlos y, a su vez, podrán buscar videojuegos en los que estén interesados – a través de los múltiples filtros que incluye la aplicación – para su posterior solicitud y compra.

Palabras clave: usuario, videojuego, solicitud, compra, framework, aplicación

Abstract

The continuous evolution of the Internet and information technologies have favoured the rise of electronic commerce. Nowadays, one of its greatest exponents are the second-hand sale pages which offer more possibilities and flexibility than conventional second-hand stores. Keeping in mind this information, the creation of a web portal dedicated to the sale of second-hand video games has been proposed to develop this TFG. During this report, the entire process followed to develop this application will be explained. In this application, users will be able to register the video games to sell. Besides, they will be able to search for those video games which they are interested - through the multiple filters included in the application - for later request and purchase.

Key words: user, videogame, request, purchase, framework, application

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

CAPÍTULO 1
Introducción

Cuando una persona, antes de que el comercio electrónico existiera, quería vender un videojuego que ya no usaba o comprar uno a un precio más barato que el original, acudía a una tienda de segunda mano. A la hora de vender, era el establecimiento el que imponía el precio, siendo este no negociable y sin dejar al vendedor establecer por cuanto quería vender su producto. El ir a este tipo de tiendas físicas se debe al hecho de que no existía otro método para la compraventa. Sin embargo, con el auge del comercio *online*, aparecieron portales web dedicados a este tipo de negocio. En ellos, son los propios vendedores los que establecen el precio por el que quieren vender su producto. A diferencia de las tiendas físicas, en estas plataformas *online* los compradores pueden intentar regatear el precio al vendedor, siendo este quien decida si acepta la oferta o si se mantiene fiel a su oferta inicial. Asimismo, al no haber intermediario, los vendedores se llevan el importe integro del producto. Todas estas ventajas han hecho que este tipo de plataformas ganen popularidad en relativamente poco tiempo y se conviertan en una de las grandes potencias del comercio *online*.

1.1 Motivación

La motivación para realizar este proyecto se debe a distintos factores. Desde que cursé la asignatura 'Desarrollo web' en tercero de carrera, tuve claro que hacia donde quería enfocar mi carrera profesional era a la creación de aplicaciones web. Por ello, esta ha sido una de las principales motivaciones para realizar una aplicación como TFG. Además, de esta manera he podido seguir mejorando mis habilidades como programador y también he podido utilizar las tecnologías que he ido empleando durante mi periodo de prácticas en empresa, demostrando así lo que he aprendido. Asimismo, desde hace bastante tiempo, el sector del comercio electrónico me ha ido generando cada vez más interés, llegando a estar al tanto de muchas de las noticias y avances tecnológicos sobre este ámbito. Por este motivo, me pareció interesante basar la aplicación sobre este sector. Por último, pero no por ello menos importante, los videojuegos han formado parte de mi vida desde que era pequeño y tenía claro que quería que formaran parte de mi TFG. Juntando esta idea y las dos anteriores, surgió el tema de este proyecto: una aplicación dedicada a la compraventa de videojuegos de segunda mano.

1.2 Objetivos

El objetivo principal de este TFG es el desarrollo de una aplicación destinada a la compraventa de videojuegos de segunda mano. En ella, los usuarios podrán vender los

videojuegos al precio que deseen, para así recuperar parte de la inversión que hicieron en su momento. Por otro lado, los usuarios podrán solicitar la compra de un videojuego al vendedor pudiendo ofrecer una contraoferta al precio establecido. Otro de los objetivos que se quiere conseguir es que un usuario pueda buscar los videojuegos que desee con la mayor precisión posible, para que pueda encontrar exactamente lo que está buscando.

1.3 Estructura de la memoria

Esta memoria estará dividida en siete capítulos. En el primero, se muestran las motivaciones que han hecho que este proyecto se lleve a cabo. A su vez, se presentan los objetivos a cumplir a partir de este. En el segundo capítulo se realizará un análisis sobre diferentes aplicaciones existentes en el mercado y se compararán con esta propuesta. En el tercero, se hará un análisis del problema a resolver. Para ello se especificarán una serie de requisitos que la aplicación deberá cumplir, así como, casos de uso y un diagrama UML explicativo. En el cuarto capítulo se explicará la arquitectura elegida para desarrollar este proyecto junto con las herramientas utilizadas para ello y un conjunto de *mock-ups* que servirán de referencia para diseñar la aplicación. En el quinto, se explicará cómo se ha desarrollado el proyecto, analizando cada una de las partes en las que está dividida y su funcionamiento en la aplicación. El sexto capítulo estará formado por pruebas a partir de las cuales se validará el correcto funcionamiento de la aplicación. Por último, en el séptimo capitulo, se presentarán las conclusiones de este proyecto, así como, mejoras futuras planteadas para este.

CAPÍTULO 2
Estado del arte

Hoy en día la compra *online* se encuentra en pleno auge, y con ella, los portales web destinados a la venta de artículos de segunda mano no tardaron en aparecer. Actualmente, algunas de las páginas más influyentes son las siguientes:

Ebay: Página web fundada en 1995 en California, Estados Unidos. Cuenta con 50 millones de usuarios registrados. Lo que caracteriza a Ebay y diferencia de otros portales webs es la posibilidad que tienen los usuarios de hacer subastas. Los vendedores fijan un precio inicial y, tras un periodo de tiempo establecido, el usuario que haya hecho la puja más alta ganará y estará obligado a continuar con el proceso de compra, completando así la transacción por el producto deseado.[1]

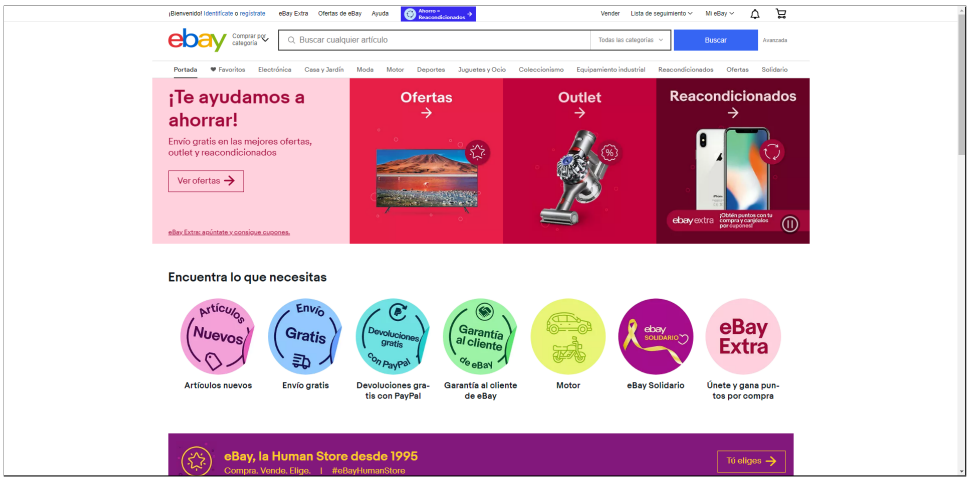

Figura 2.1: Página principal de Ebay

Vinted: Portal web fundado en 2012 en Lituania. Actualmente, cuenta con más de 45 millones de usuarios registrados. Su objetivo principal es la venta de artículos de moda de segunda mano: ropa, calzado, accesorios, etc. Aunque también se pueden encontrar artículos del hogar, juguetes o libros. Uno de sus puntos fuertes es la seguridad que es ofrecida al usuario a la hora de realizar la compra. Hasta que el comprador no haya recibido el producto y verificado que se encuentra en buenas condiciones, Vinted retendrá el cobro al vendedor, disminuyendo así la posibilidad de estafas. [2]

Wallapop: Página web fundada en España en 2013. Hoy en día cuenta con más de 180 millones de productos ofertados y más de 15 millones de usuarios registrados. Una de las características de Wallapop es el uso de la geolocalización. Su objetivo principal es mostrar al usuario los productos que otras personas han subido a la plataforma y listarlos en función de su cercanía.[3, 4]

2.1 Diferencias y similitudes entre los portales de compraventa actuales

Los portales web mencionados en el apartado anterior presentan ciertas similitudes entre ellos. Se hace uso de un sistema de categorías donde se encuentran clasificados los productos según el artículo que se está anunciando: motor, informática, imagen y sonido... En el caso de Vinted, al estar dirigida a un ámbito más específico, las categorías se centran en el tipo de artículo de moda: ropa, calzado, bolsos, accesorios...

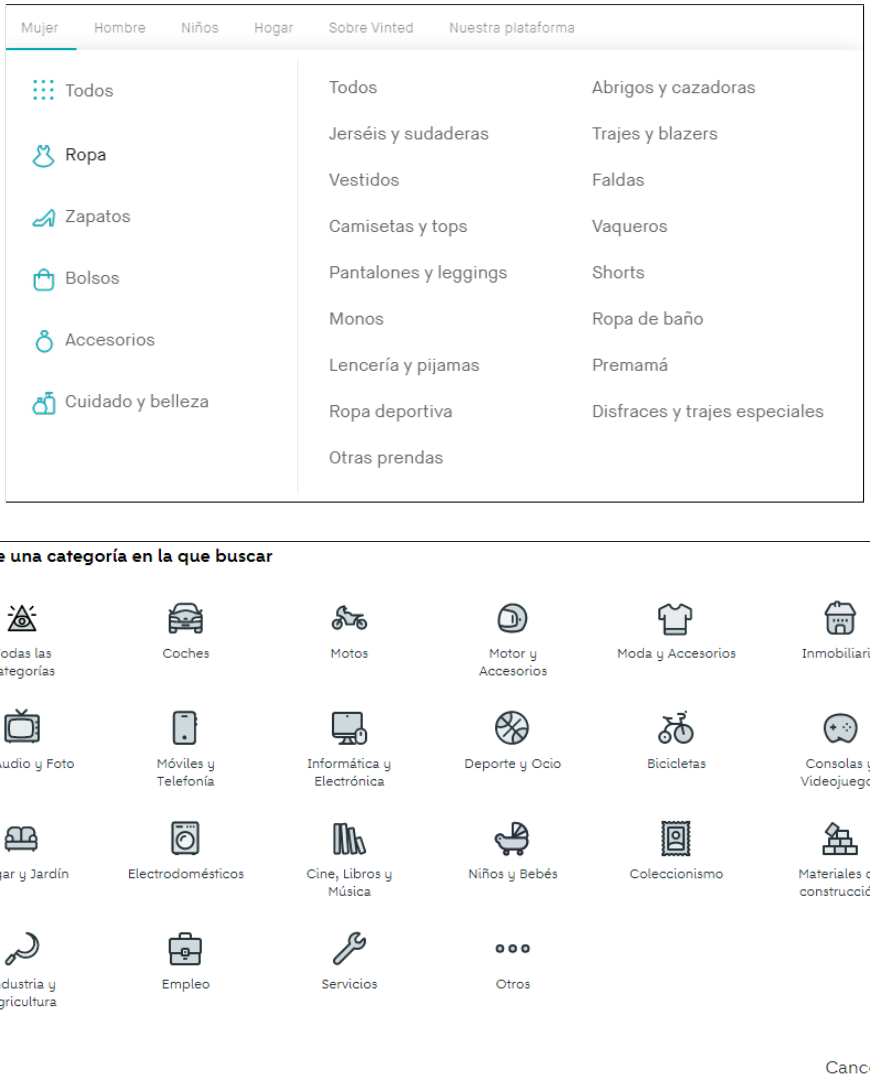

Figura 2.2: Categorías de Vinted vs Categorías de Wallapop

A su vez, estas webs utilizan una serie de filtros y ordenaciones que ayudan a buscar de forma más precisa. Algunos filtros recurrentes son el filtrado por precio – dando la posibilidad de encontrar lo que más se ajusta al presupuesto del usuario–, el filtrado por lugar – en caso de querer especificar la provincia deseada – o el estado del producto, entre otros. Respecto a las ordenaciones, las más comunes son: por precio – ascendente o descendente –, por últimas novedades, por distancia y por relevancia.

También estará a disposición del usuario un perfil donde los datos personales pueden ser gestionados, así como, una lista de productos favoritos donde se encontrarán todos los productos que el usuario desee

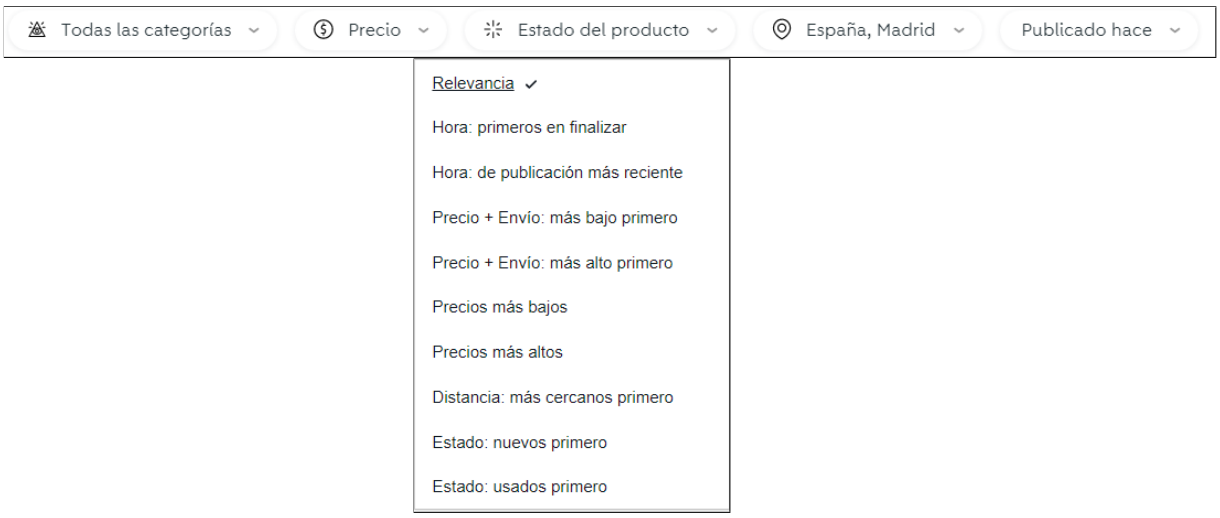

Figura 2.3: Filtros de Wallapop y Ordenaciones de Ebay

Por lo que se refiere a las diferencias, estas se basan en funcionalidades que algunas de las webs tienen y otras no, en el contenido que se puede vender en la página y en quién puede vender en dichas páginas.

Respecto a las diferencias entre las funcionalidades, tanto Vinted como Wallapop cuentan con un chat en vivo donde los usuarios pueden conversar para llegar, o no, a un acuerdo. Sin embargo, en Ebay si se quiere hablar con el vendedor tendrá que ser vía email. Por último, Ebay cuenta con el apartado de subastas, característica fundamental que lo diferencia de las otras webs mencionadas.

En relación con el contenido que se puede ofertar en la web, tanto Ebay como Wallapop permiten la venta de artículos referentes a una gran cantidad de categorías, como pueden ser: automóviles, electrodomésticos, *smartphones*... Sin embargo, Vinted se reduce principalmente a artículos de moda, aunque existe la posibilidad de anunciar algunos productos fuera de este ámbito, pero no se pueden comparar con la gran cantidad de productos en los otros dos portales webs mencionados.

Por último, Ebay ofrece la posibilidad de vender artículos en calidad de empresa en lugar de un particular, característica que ni en Vinted ni Wallapop poseen.

2.2 Solución propuesta

Una vez vistas las similitudes y diferencias entre los portales de compraventa ya mencionados, queda por analizar en que va a diferir esta propuesta de las demás.

Principalmente los cambios serán dos. Primero, este proyecto se va a centrar única y exclusivamente en la compraventa de videojuegos, por lo que no será posible la venta de otros artículos como si ocurre – en mayor o menor medida – en los portales anteriores. Y, en segundo lugar, se va a permitir la búsqueda de videojuegos mediante filtros específicos que no se encuentran en los portales anteriores, aunque algunos de ellos también permitan la venta de videojuegos. La existencia de estos filtros viene dada por uno de los objetivos de este proyecto, el hacer posible que alguien interesado en cierta consola y/o un género determinado pueda encontrar los videojuegos que busca con la mayor precisión posible, posibilidad que tampoco ofrecen las webs ya mencionadas.

CAPÍTULO 3
Analisis del problema

Una vez vistas las diferentes webs existentes en el mercado y aclarado en que se va a distinguir esta propuesta de las demás, es necesario hacer un análisis del problema a desarrollar. A continuación, se realizará una especificación de requisitos que deberá cumplir la aplicación, seguido de un conjunto de casos de uso y finalizando con un diagrama E-R.

3.1 Especificación de requisitos de software según el estándar IEEE 830/98

3.1.1. Introducción

En este apartado se realizará una especificación de requisitos de software (ERS) para el portal de compraventa de videojuegos de segunda mano que se va a desarrollar en este TFG. Esta especificación se ha estructurado a partir del estándar IEEE 830/98.

Propósito

El propósito de esta especificación es definir los requisitos, tanto funcionales como no funcionales, que la aplicación deberá cumplir para que funcione tal y como se espera. En un ámbito profesional, estaría dirigido al cliente y a los desarrolladores, pero en este caso será utilizado por el estudiante y los profesores.

Ámbito del sistema

Este sistema permitirá a los usuarios vender los videojuegos que ya no utilizan o comprar aquellos en los que puedan estar interesados. Sin embargo, la compra será ficticia, al quedarse el desarrollo de una pasarela de compra segura fuera del ámbito de este TFG.

Abreviaturas utilizadas

- ERS –> Especificación de requisitos de software
- RF –> Requisito funcional
- RNF –> Requisito no funcional

Visión general de la sección

Esta sección contará con tres apartados. En el primero se hace una introducción a esta sección y se proporciona una aproximación de que es lo que hará esta aplicación. En el segundo apartado se hace una descripción general del sistema donde se comentará, sin entrar en detalles, las principales funciones que realizará la aplicación, así como características de los usuarios que utilizarán el producto, restricciones y suposiciones que afectarán al desarrollo del proyecto. Por último, el tercer apartado será en el que se definan los requisitos tanto funcionales como no funcionales que la aplicación deberá cumplir.

3.1.2. Descripción general

En esta sección se describirán todos los factores que afectarán a la aplicación. Se dividirá en los siguientes subapartados: funciones del producto, características de los usuarios, restricciones y suposiciones.

Funciones del producto

La aplicación se encargará de mostrar a los usuarios una lista de los videojuegos disponibles para comprar. A su vez, lo usuarios podrán filtrar sobre estos para buscar de una forma mas precisa. Asimismo, se permitirá a los usuarios la creación de solicitudes sobre los videojuegos en los que estén interesados. Estas solicitudes podrán ser aprobadas, rechazadas o borradas. Por último, una vez una solicitud sea aprobada, el sistema permitirá al usuario proceder a la compra del videojuego.

Características de los usuarios

En esta aplicación no existirán diferentes tipos de usuarios, pero cabe mencionar que todos los usuarios pueden adoptar dos roles distintos:

- Comprador: Será aquel que realice las solicitudes y proceda a la compra de los artículos. Será necesario de un nivel educacional básico y no es necesaria experiencia previa en otro tipo de sistemas.
- Vendedor: será aquel que ponga los videojuegos a la venta y apruebe las solicitudes para la compra de los artículos. Será necesario de un nivel educacional básico y no es necesaria experiencia previa en otro tipo de sistemas.

Restricciones

Las restricciones que habrá en el desarrollo del proyecto serán las siguientes:

- Protocolos de comunicación: La aplicación deberá comunicarse con el servidor mediante el protocolo HTTP.
- Seguridad: El sistema deberá ser seguro, no permitiendo la entrada a la aplicación a ningún usuario no registrado o que no haya iniciado sesión correctamente.
- Tecnologías utilizadas: deberá utilizarse la arquitectura MEAN Stack.

Suposiciones

A partir de esta definición de requisitos existen ciertas suposiciones que deben ser cumplidas para el buen funcionamiento de la aplicación. Primero, el sistema operativo en el que deberá ejecutarse la aplicación será Windows y, segundo, el navegador que deberá ser utilizado para su uso deberá ser Firefox o Google Chrome.

3.1.3. Requisitos Específicos

En este apartado se expondrán los diferentes requisitos, tanto funcionales como no funcionales, que la aplicación debe cumplir.

Requisitos funcionales

- **Creación y edición de usuarios [RF 01]**: El sistema debe ser capaz de crear y actualizar usuarios de la aplicación.
- **Creación y edición de videojuegos [RF 02]**: El sistema debe ser capaz de crear, listar, actualizar y borrar videojuegos de la aplicación.
- **Validar el acceso a la aplicación [RF 03]**: El sistema deberá validar las credenciales de inicio de sesión de los usuarios y obligarles a iniciar sesión antes de entrar en la aplicación, usando un servicio de autenticación.
- **Listar videojuegos en venta [RF 04]**: El sistema permitirá al usuario visualizar los videojuegos que los demás usuarios hayan puesto a la venta.
- **Filtrado de videojuegos [RF 05]**: El usuario podrá ordenar la lista de videojuegos a la venta según una serie de preferencias. Por un lado, se podrá ordenar según el precio (ascendente o descendente) o bien por últimos añadidos (opción por defecto). Por otro lado, se podrá filtrar según el género, videoconsola, si es multijugador o no, año de publicación, nombre, precio y/o PEGI.
- **Visualizar los propios videojuegos [RF 06]**: El usuario podrá visualizar los videojuegos que tenga ahora mismo en venta.
- **Creación de una solicitud de compra [RF 07]**: El sistema permitirá al usuario crear una solicitud de compra a un videojuego donde el usuario quedará a la espera de que el vendedor apruebe la compra.
- **Administrar solicitudes del usuario [RF 08]**: El sistema permitirá al usuario, por un lado, poder revisar el estado de las solicitudes de videojuegos que él mismo haya realizado o borrarlas si así lo desea, y, por otro lado, podrá revisar y administrar solicitudes que él haya recibido sobre videojuegos que tenga a la venta.
- **Compra de videojuegos [RF 09]**: El usuario podrá proceder a la compra del videojuego una vez que el vendedor lo haya autorizado. (Para este proyecto se tratará de una compra ficticia, ya que el desarrollo de una pasarela de compra segura queda fuera del ámbito de este TFG)
- **Visualizar compras del usuario [RF 10]**: El sistema permitirá al usuario visualizar el histórico de compras que haya realizado.
- **Visualizar ventas del usuario [RF 11]**: El sistema permitirá al usuario visualizar el histórico de ventas que haya realizado.

Requisitos no funcionales

- **Seguridad de la aplicación [RNF 01]:** El acceso a esta se hará mediante un *token* cifrado por lo que los usuarios no registrados serán incapaces de entrar a la aplicación.
- **Usabilidad [RNF 02]:** El sistema debe ser lo suficientemente intuitivo como para que al usuario no tenga, o apenas tenga dudas sobre el funcionamiento de la aplicación.
- **Usabilidad II [RNF 03]:** El sistema mostrará mensajes de error si algún proceso no se ha efectuado correctamente o mensajes satisfactorios indicando que si se han procesado correctamente.
- **Disponibilidad [RNF 04]:** El sistema estará disponible la mayor parte del tiempo.

3.2 Casos de uso

En esta sección se procederá a describir los diferentes casos de uso que deberán cumplirse para que la aplicación funcione correctamente y que sea acorde a los requisitos establecidos.

3.2.1. Registrar un usuario: Válido

- **Prerequisito:** El usuario no se ha registrado
- **Rol:** Comprador/Vendedor
- **Requisitos cubiertos:** RF 01, RNF 03
- **Pasos:**
  1. El usuario inicia la aplicación
  2. El usuario selecciona la pestaña de registro.
  3. . El usuario rellena todos los datos necesarios para completar el registro, indicando un nombre de usuario, correo electrónico y/o teléfono que no pertenezca a otro ya registrado.
  4. El usuario pulsa el botón de registrarse.
- **Salida:** La aplicación vuelve a la pestaña de iniciar sesión y muestra una notificación indicando que el usuario se ha registrado con éxito.

3.2.2. Registrar un usuario: Inválido

- **Prerequisito:** El usuario no se ha registrado
- **Rol:** Comprador/Vendedor
- **Requisitos cubiertos:** RF 01, RNF 03
- **Pasos:**
  1. El usuario inicia la aplicación.
  2. El usuario selecciona la pestaña de registro.

3. El usuario rellena todos los datos necesarios para completar el registro, indicando un nombre de usuario, correo electrónico y/o teléfono que ya pertenezca a otro ya registrado.
4. El usuario pulsa el botón de registrarse.

- **Salida:** La aplicación se mantiene en la pestaña de registro y muestra una notificación indicando el error al usuario.

3.2.3. Iniciar sesión: Válido

- **Prerequisito:** El usuario se ha registrado previamente
- **Rol:** Comprador/Vendedor
- **Requisitos cubiertos:** RF 03, RNF 01, RNF 03
- **Pasos:**

1. El usuario inicia la aplicación
2. El usuario escribe su nombre de usuario y su contraseña en el formulario, siendo ambos campos correctos.
3. El usuario pulsa el botón de iniciar sesión.

- **Salida:** La aplicación se mantiene en la pestaña de iniciar sesión y muestra una notificación indicando el error al usuario.

3.2.4. Iniciar sesión: Inválido

- **Prerequisito:** El usuario se ha registrado previamente
- **Rol:** Comprador/Vendedor
- **Requisitos cubiertos:** RF 03, RNF 01, RNF 03
- **Pasos:**

1. El usuario inicia la aplicación
2. El usuario escribe su nombre de usuario y su contraseña en el formulario, siendo alguno de ellos, o ambos, incorrectos.
3. El usuario pulsa el botón de iniciar sesión.

- **Salida:** La aplicación se mantiene en la pestaña de iniciar sesión y muestra una notificación indicando el error al usuario.

3.2.5. Filtrar videojuegos

- **Prerequisito:** El usuario ha iniciado sesión
- **Rol:** Comprador
- **Requisitos cubiertos:** RF 04, RF 05
- **Pasos:**

1. El usuario pulsa el botón de inicio

2. El usuario selecciona la opción de 'Filtros Avanzados'.
3. El usuario rellena todos los campos por los que quiera filtrar un videojuego
4. El usuario pulsa el botón de filtrar o en el botón representado por una lupa.

- **Salida:** La aplicación muestra todos los videojuegos que coinciden con los campos rellenados por el usuario.

3.2.6. Realizar una solicitud: Válido

- **Prerequisito:** El usuario aún no ha solicitado el videojuego en cuestión.
- **Rol:** Comprador
- **Requisitos cubiertos:** RF 07, RNF 03
- **Pasos:**

1. El usuario presiona el botón de inicio
2. El usuario presiona el botón de solicitar en el videojuego que desee.
3. El usuario seleccionará si quiere realizar una oferta o no, indicará el nuevo precio si corresponde y presionará el botón de aceptar.

- **Salida:** La aplicación se mantiene en la pestaña de inicio y muestra una notificación indicando que la solicitud se ha realizado correctamente.

3.2.7. Realizar una solicitud: Inválido

- **Prerequisito:** El usuario ya ha solicitado el videojuego en cuestión.
- **Rol:** Comprador
- **Requisitos cubiertos:** RF 07, RNF 03
- **Pasos:**

1. El usuario presiona el botón de inicio
2. El usuario presiona el botón de solicitar en el videojuego que desee.
3. El usuario seleccionará si quiere realizar una oferta o no, indicará el nuevo precio si corresponde y presionará el botón de aceptar.

- **Salida:** La aplicación se mantiene en la pestaña de inicio y muestra una notificación de error indicando que el usuario ya ha solicitado el videojuego.

3.2.8. Editar perfil: Válido

- **Prerequisito:** El usuario no se ha registrado
- **Rol:** Comprador/Vendedor
- **Requisitos cubiertos:** RF 01, RNF 03
- **Pasos:**

1. El usuario presiona el botón 'Mi perfil'.

2. El usuario presiona el 'Editar'.
3. El usuario cambiará los datos que desee en el formulario mostrado.
4. El usuario presiona el botón aceptar.

- **Salida:** La aplicación vuelve a la pestaña del perfil del usuario con los datos actualizados.

3.2.9. Editar perfil: Inválido

- **Prerequisito:** El usuario ha iniciado sesión.
- **Rol:** Comprador/Vendedor
- **Requisitos cubiertos:** RF 01, RNF 03
- **Pasos:**

1. El usuario presiona el botón 'Mi perfil'.
2. El usuario presiona el 'Editar'.
3. El usuario cambiará los datos que desee en el formulario mostrado, incluyendo un número de teléfono, correo electrónico y/o un nombre de usuario que ya pertenecen a otro usuario registrado.
4. El usuario presiona el botón aceptar.

- **Salida:** La aplicación vuelve a la pestaña del perfil del usuario mostrando los mismos datos y un mensaje de error explicando el error en cuestión.

3.2.10. Aprobar solicitud

- **Prerequisito:** El usuario ha iniciado sesión y tiene una o más solicitudes recibidas.
- **Rol:** Vendedor
- **Requisitos cubiertos:** RF 08, RNF 03
- **Pasos:**

1. El usuario presiona el botón 'Solicitudes recibidas'.
2. El usuario presiona el botón 'Aceptar' de la solicitud que quiera aceptar.
3. El usuario confirmará que quiere aceptar la solicitud en la ventana emergente que aparecerá.

- **Salida:** La aplicación se mantiene en la pestaña de solicitudes recibidas mostrando esta solicitud como aceptada y mostrando las solicitudes restantes sobre el mismo videojuego como suspendidas.

3.2.11. Rechazar solicitud

- **Prerequisito:** El usuario ha iniciado sesión y tiene una o más solicitudes recibidas.
- **Rol:** Vendedor
- **Requisitos cubiertos:** RF 08, RNF 03

- **Pasos:**
  1. El usuario presiona el botón 'Solicitudes recibidas'.
  2. El usuario presiona el botón 'Rechazar' de la solicitud que quiera aceptar.
  3. El usuario confirmará que quiere rechazar la solicitud en la ventana emergente que aparecerá.
- **Salida:** La aplicación se mantiene en la pestaña de solicitudes recibidas sin mostrar ahora la solicitud que ha sido rechazada.

3.2.12. Borrar solicitud

- **Prerequisito:** El usuario ha iniciado sesión y ha realizado una o más solicitudes.
- **Rol:** Comprador
- **Requisitos cubiertos:** RF 08, RNF 03
- **Pasos:**
  1. El usuario presiona el botón 'Solicitudes realizadas'.
  2. El usuario presiona el 'Borrar de la solicitud que quiera aceptar.
  3. El usuario confirmará que quiere borrar la solicitud en la ventana emergente que aparecerá.
- **Salida:** La aplicación se mantiene en la pestaña de solicitudes realizadas sin mostrar ahora la solicitud que ha sido borrada.

3.2.13. Registrar videojuegos

- **Prerequisito:** El usuario ha iniciado sesión.
- **Rol:** Vendedor
- **Requisitos cubiertos:** RF 02, RF 06, RNF 03
- **Pasos:**
  1. El usuario presiona el botón 'Registra un videojuego'.
  2. El usuario rellena los datos del formulario necesarios para el registro de este.
  3. El usuario presiona el botón 'Crear'.
- **Salida:** La aplicación desplaza al usuario a la pestaña 'Mis videojuegos a la venta' junto con una notificación indicando que el videojuego se ha registrado correctamente.

3.2.14. Editar videojuegos

- **Prerequisito:** El usuario ha iniciado sesión y tiene uno o más videojuegos a la venta.
- **Rol:** Vendedor
- **Requisitos cubiertos:** RF 02, RF 06, RNF 03
- **Pasos:**

1. El usuario presiona el botón 'Mis Videojuegos a la venta'.
2. El usuario presiona el 'Editar' del videojuego que quiera editar.
3. La aplicación mostrará un formulario donde aparecerán los datos del videojuego a editar.
4. El usuario cambiará los datos que quiera del formulario y presionará en el botón 'Editar'.

- **Salida:** La aplicación se mantiene en la misma pestaña mostrando ahora el videojuego en cuestión con los datos cambiados.

3.2.15. Comprar videojuego

- **Prerequisito:** El usuario ha iniciado sesión y cuenta con una o más solicitudes aceptadas.
- **Rol:** Comprador
- **Requisitos cubiertos:** RF 09, RF 10, RF 11, RNF 03
- **Pasos:**

1. El usuario presiona el botón 'Solicitudes realizadas'.
2. El usuario presiona el 'Comprar' del videojuego que quiera comprar.
3. El usuario confirmará que quiere comprar el videojuego en la ventana emergente que aparecerá.
4. La aplicación mostrará un formulario donde el usuario deberá revisar sus datos y los del videojuego a comprar.
5. El usuario presiona el botón 'Comprar'.

- **Salida:** La aplicación mostrará un mensaje de compra satisfactoria junto con un botón para redirigir al usuario a la pestaña 'Mis Compras' y se mostrará una notificación indicando que el videojuego ha sido comprado correctamente.

3.3 Diagrama E-R

En este diagrama se pueden observar los distintos objetos de negocio con los que se va a operar en la aplicación junto con las acciones con las que interactuarán entre sí.

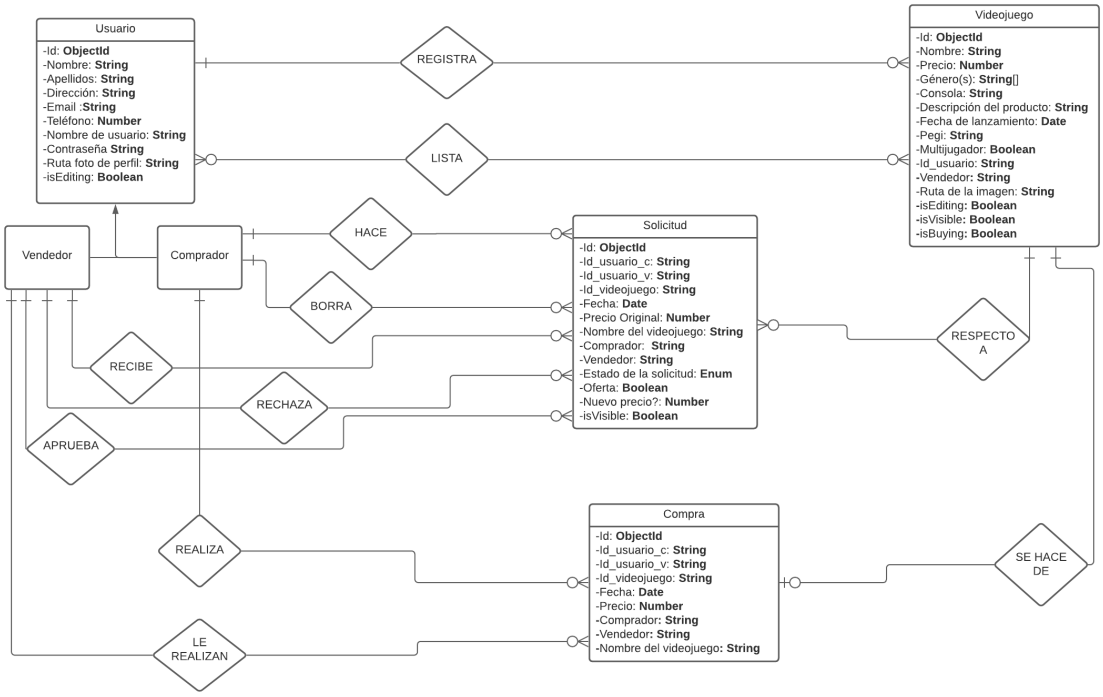

Figura 3.1: Diagrama E-R

Como se puede apreciar en el diagrama, la aplicación cuenta con 4 objetos fundamentales: **usuarios, videojuegos, solicitudes** y **compras**. Los usuarios serán los encargados de registrar los videojuegos que quieran vender, rellenando los datos necesarios en el formulario dedicado al registro de videojuegos. A su vez, estos videojuegos serán listados a los usuarios en la página principal de la aplicación. En cuanto a las solicitudes, estas serán realizadas por los usuarios en calidad de comprador, o recibidas cuando sean los vendedores del videojuego en cuestión. Estas solicitudes mantendrán su atributo 'Estado de la solicitud' con un valor igual a 'En Espera' hasta que el vendedor decida si la aprueba o la rechaza. Si se aprueba la solicitud, su estado cambiará a 'Aprobada' mientras que si se rechaza cambiará a 'Rechazada'. Por otro lado, el usuario que ha realizado la solicitud tendrá la opción de borrarla. Una vez una solicitud haya sido aprobada por el vendedor, el usuario que la haya realizado podrá proceder a la compra del videojuego por el cual se ha realizado la solicitud.

CAPÍTULO 4
Diseño de la solución

En el capítulo anterior se pudo observar tanto los requisitos que la aplicación debe cumplir como las diferentes acciones que el usuario puede realizar, así como, un diagrama UML con el objetivo de aclarar el funcionamiento de la aplicación. Por lo tanto, en este capítulo se procederá a explicar cuál será la arquitectura de la aplicación y el conjunto de tecnologías y herramientas utilizadas para su desarrollo.

4.1 Arquitectura del sistema

Para desarrollar esta infraestructura se ha elegido una arquitectura basada en MEAN *stack*. MEAN es un acrónimo que hace referencia a las tecnologías utilizadas en esta arquitectura: Mongo DB, Express, Angular.js y Node.js. [6]

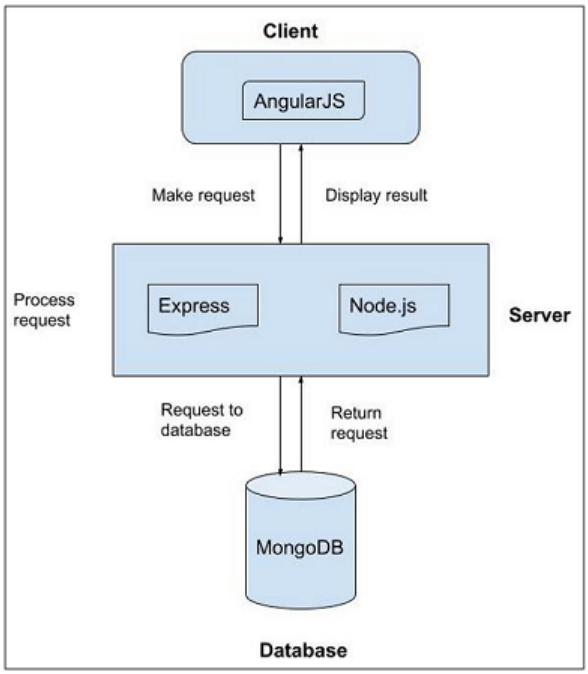

Figura 4.1: Arquitectura MEAN *Stack*

MEAN destaca por ser una arquitectura homogénea, ya que todas las tecnologías utilizadas hacen uso de Javascript, en lugar de un lenguaje distinto en cada parte de la aplicación. Esto a su vez favorece al buen estado del trabajador ya que todo es desarrollado bajo unos mismos conceptos y no es necesario dominar diferentes lenguajes para poder participar en todo el proyecto. [5]

4.2 Tecnologías utilizadas

En esta sección se explicarán las tecnologías utilizadas en MEAN *stack*, comenzando por MongoDB, siguiendo con Express.js y Nest.js, Angular y finalmente Node.js

4.2.1. Mongo DB

MongoDB es un sistema de gestión de bases de datos no relacional. La información almacenada en Mongo son objetos, en lugar de filas. A estos objetos se les conoce como "documentos". El formato de almacenamiento de los objetos es del estilo de JSON

Un objeto JSON es una colección no ordenada de pares (nombre, valor). En un objeto el contenido se encuentra separado por comas y delimitado por llaves. [7]


```
{
    "_id": {
        "$oid": "60e0a97fbeb7fa2cdf0696d2"
    },
    "name": "Daniel",
    "surnames": "Sánchez López",
    "address": "Av Tarongers, 2",
    "email": "daniel@upv.es",
    "password": "daniel",
    "phone": "689745123",
    "username": "daniel",
    "filePath": "4-8fc64.jpg",
    "isEditing": false,
    "__v": {
        "$numberInt": "0"
    }
}
```


Figura 4.2: Objeto JSON de un usuario en MongoDB

MongoDB permite filtrar y ordenar por cualquier campo independientemente de cómo se encuentre en el documento. Además, las consultas sobre la base de datos son también en formato JSON, por lo que son fáciles de aprender y de programar. [8]

Para utilizar MongoDB se puede instalar e iniciar un servidor en el ordenador personal o, por el contrario, se puede utilizar el servicio que Mongo proporciona en la nube. Esta última opción ha sido la elegida para este proyecto sobre todo por la comodidad y para disminuir la carga de trabajo del PC.

Mongoose

Trabajar directamente con MongoDB puede ser complicado. Para ello, se ha utilizado Mongoose. Se trata de un ODM (*Object Data Mapper*) que simplifica la lógica de negocio en las aplicaciones de Node.js. A través de Mongoose se podrá conectar la aplicación a la base de datos y actuar como intermediario entre la base de datos y el servidor, utilizando su sistema de modelos y esquemas. Estos esquemas servirán para definir la estructura que tendrán los objetos una vez se encuentren en la base de datos.

```
@Schema({collection: 'users'})
export class User {
    @Prop()
    username!: string;

    @Prop()
    password!: string;

    @Prop()
    email!: string;

    @Prop()
    address!: string;

    @Prop()
    phone!: string;

    @Prop()
    name!: string;

    @Prop()
    surnames!: string;

    @Prop()
    filePath!: string;

    @Prop()
    isEditing!: boolean;
}
```

Figura 4.3: Representación de un esquema en Mongoose.

Una vez el esquema esté listo, habrá que convertirlo a un modelo con el que se pueda trabajar. A partir de estos modelos se realizarán las operaciones sobre la base de datos. Como se puede observar en la siguiente imagen, a través del método '*save*()' se guardará el nuevo objeto creado a partir del modelo en la base de datos. [23]

```
public async createUser(userDto: UserDto): Promise<User> {
  const newUser = await new this.userModel(userDto).save();
  return newUser;
}
```

Figura 4.4: Operación de crear un usuario en la base de datos.

4.2.2. Express.js y Nest.js

Express es definido, según su página web, como: "una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles." [9] Hoy en día es uno de los *frameworks* más famosos y utilizados para desarrollar aplicaciones web y ha sentado la base de muchos *frameworks*

posteriores. Proporciona mecanismos para manejar peticiones HTTP, definir rutas y también permite el uso de *middlewares* (porciones de código que se pueden colocar entre las peticiones y permiten tratar el código, por ejemplo: verificar la autenticación del usuario). [5, 10]

Por otro lado, Nest.js es un *framework* utilizado para desarrollar aplicaciones del lado del servidor eficientes y escalables. Al estar Nest.js basado en Express, implementa sus funcionalidades y, además, adopta otras que han hecho que esta sea la herramienta elegida para usar en el proyecto. Al encontrarse influenciado en gran medida por Angular, aprovecha muchos de sus conceptos como son los módulos, los controladores y la inyección de dependencias. Además, permite la opción de programar también en Typescript, potenciando la homogeneidad anteriormente mencionada. [11] A continuación, se pasará a definir las partes básicas en las que se divide una aplicación de Nest.js: [24]

- Controladores: se encargan de gestionar las solicitudes y devolver los datos al cliente. La misión principal del controlador es recibir las solicitudes de la aplicación. El mecanismo de enrutamiento determina qué controlador recibirá según que peticiones. Cada controlador podrá tener más de una ruta y múltiples rutas podrán realizar varias acciones, según el tipo de petición. Para crear un controlador básico, Nest usa clases y decoradores. Estos decoradores asocian las clases con los metadatos necesarios y permiten que Nest construya un mapa de enrutamiento. Para definir métodos HTTP para una determinada ruta, existen los siguientes decoradores: *@Get()* , *@Post()*, *@Put()*, *@Delete()*, etc.

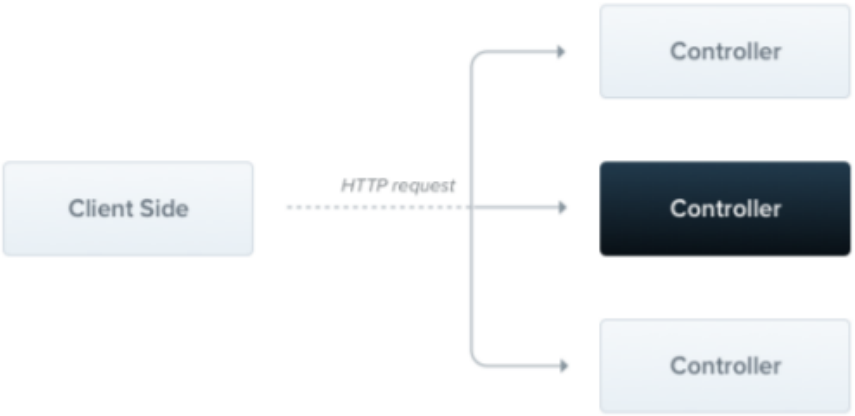

Figura 4.5: Funcionamiento de los controladores en Nest.js

- Modulo: Un módulo es la clase donde se conectan los servicios y los controladores que tienen relación entre si. Sirven para separar las funcionalidades básicas de la aplicación en distintos bloques. Estos módulos utilizarán el decorador *@Module()*.

- Proveedores: Se trata de una clase anotada con el decorador *@Injectable()*. La idea principal de un proveedor es que puede inyectar dependencias, es decir, que los objetos pueden crear varias relaciones entre sí.

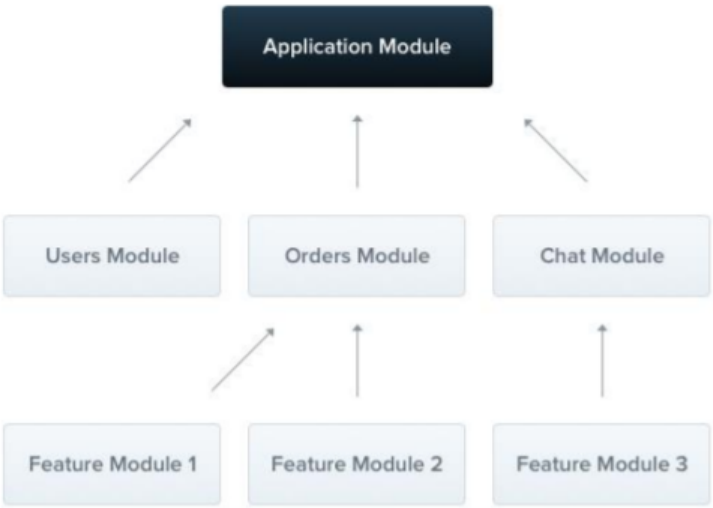

Figura 4.6: Funcionamiento de los módulos en Nest.js

- Servicios: Los servicios son los responsables de operar en la base de datos y son administrados por los controladores. Un servicio se define también como un tipo de proveedor por lo que también utilizaran el decorador *@Injectable()*.

Por último, cabe destacar que el uso del *framework* Express.js podría haber sido usado en este proyecto sin ningún problema. El uso de Nest.js en su lugar ha sido una decisión totalmente subjetiva y personal. Esto no significa que sea un mejor o peor *framework* que la opción elegida.

4.2.3. Angular.js

Angular es un *framework* utilizado para el desarrollo *front-end* de aplicaciones web que está basado en JavaScript.

El desarrollo de aplicaciones web con Angular se basa en el uso de componentes. Estos componentes se constituyen de código en Typescript con un decorador *@Component*, una plantilla HTML y estilos. El decorador *@Component* especifica la siguiente información [16]:

- Un selector CSS que define como el componente se usará en una plantilla HTML. Los elementos HTML que coinciden con el selector se convierten en instancias del componente.
- Una plantilla HTML que le indica a Angular cómo renderizar el componente.
- Un conjunto opcional de estilos CSS que definen la apariencia de los elementos HTML de la plantilla.

Una aplicación compleja se construirá a partir de diferentes componentes. Uno de los puntos fuertes de Angular es la reutilización. Una vez creado el componente, puede ser instanciado en cualquier parte del proyecto a través de su decorador CSS. [17] Además, potenciando esta reutilización, Angular permite la inyección de datos a la hora de instanciar un componente. Estos serán recibidos a través del decorador *@Input()* y podrán

ser utilizados como una variable más dentro del componente. De esta manera se podrán usar como condiciones para controlar que es lo que se quiere mostrar en cada una de las vistas.

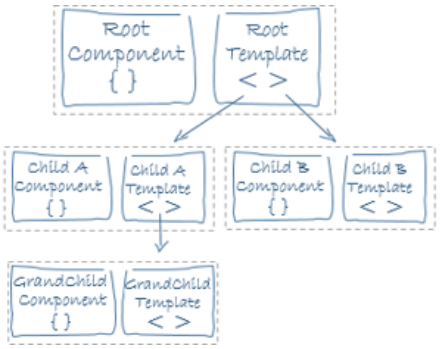

Figura 4.7: Aplicación construida a partir de distintos componentes.

Por otro lado, las aplicaciones de Angular son modulares. Estos módulos consolidan componentes, directivas y servicios en bloques cohesivos de funcionalidad, cada uno de los cuales se centra en un área de funcionamiento. Cada aplicación Angular tiene al menos un módulo, el módulo raíz, que se denomina convencionalmente *AppModule* y reside en un archivo llamado 'app.module.ts'. Si bien una aplicación pequeña puede tener solo un módulo, la mayoría de las aplicaciones tienen muchos más módulos entre los cuales se reparten las funcionalidades de la aplicación. El módulo raíz de una aplicación se llama así porque puede incluir módulos secundarios en una jerarquía de cualquier profundidad. [18, 19]

Redux y NGRX

Redux es una librería que ayuda a tratar y guardar la información de las aplicaciones desarrolladas. En lugar de estar haciendo múltiples llamadas a un servidor o guardar la información en ficheros, Redux permite manejar el estado de la aplicación (información del usuario, datos obtenidos del servidor, estado de la vista...) a través de sus tres piezas fundamentales: *Store*, *Actions* y *Reducers*. Asimismo, en este proyecto se hará uso de NGRX, un *framework* empleado para construir aplicaciones en Angular bajo el estándar que proporciona Redux.

El *Store* es un contenedor donde se encapsulará el estado de nuestra aplicación. Normalmente, cada componente tendrá la información que vaya a utilizar, pero existen casos en el que la misma información es tratada por componentes diferentes. Estos datos deberán estar sincronizados en todos los componentes que los utilicen mediante el uso de eventos. La ventaja que ofrece Redux es la centralización de los datos en el *Store*, por lo que una vez que los datos sean actualizados, se sincronizarán automáticamente en todos los componentes que los utilicen.

A su vez, presenta ventajas respecto a la escalabilidad ya que al no hacer uso de los eventos – y de las muchas confusiones que estos conllevan – no importa el número de componentes que deban acceder a la misma información.

Las acciones representan operaciones que ocurren en nuestra aplicación. Estas acciones pueden ser interacciones del usuario con la página, interacciones externas a través de solicitudes de red o interacciones directas con el API del proyecto, entre otros. A través de ellas se desencadenarán los cambios en el *Store* y/o las llamadas a servidores externos o a una API.

Los *reducers* son los responsables de manejar las transiciones de forma síncrona de un estado a otro de la aplicación. Cabe destacar que un estado no se modifica, sino que, al realizar algún cambio, se crea un estado nuevo. Cada función del reducer recoge la última acción emitida junto con el estado actual y se encargará de determinar el nuevo estado a devolver. Existen ocasiones donde si no se cumplen ciertas condiciones implementadas en la función no se podrá pasar a un nuevo estado, por lo que deberá devolverse el estado actual. A su vez, es importante destacar el papel fundamental que realizan tanto los selectores como los efectos.

Los selectores son funciones utilizadas para obtener ciertas partes del estado de nuestra aplicación. El componente se encargará de llamar a esta función para obtener la porción de estado deseada.

Los efectos proporcionan una vía para realizar las llamadas a los servicios que se comunicarán con el API o el servidor, consiguiendo así, separar esta funcionalidad de los componentes (como es el caso de Angular) y permitiendo la existencia de componentes mas "puros".

Por último, cabe destacar el ciclo de vida del estado de la aplicación en NGRX, desde que el componente realiza una acción hasta que se llega al siguiente estado.

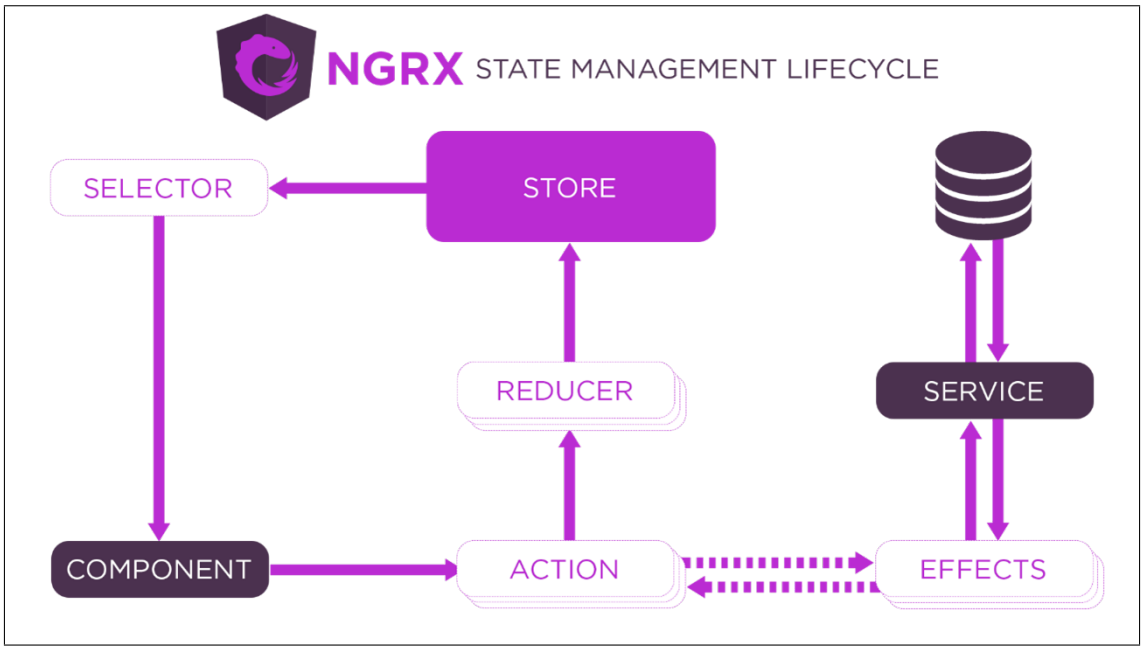

Figura 4.8: Ciclo de vida de un estado en NGRX

Inicialmente el componente se encargará de emitir la acción. Una vez esta operación haya ocurrido, se activará la función correspondiente en el *reducer* y/o en el efecto. En el caso en el que se haya accionado el efecto, este se encargará de llamar al servicio pertinente para obtener datos del servidor. Una vez haya obtenido la respuesta, ejecutará otra acción que podrá activar de nuevo tanto una función del reducer como otra función del mismo efecto. El reducer será el encargado de comprobar los parámetros que le llegan a través de la acción y, si las comprobaciones son correctas, creará un nuevo estado a partir de los parámetros obtenidos y el estado original. Este nuevo estado se guardará en el *Store* y a través de los selectores, los componentes recibirán la parte del estado que necesiten, completando así el ciclo.

4.2.4. Node.js

Node.js es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Usa un modelo de E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente. El ecosistema de paquetes de Node.js, npm, es el ecosistema más grande de librerías de código abierto en el mundo. Concebido como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node está diseñado para construir aplicaciones en red escalables. Por cada conexión se ejecutará un *callback*, sin embargo si no hay trabajo que hacer Node estará durmiendo.

Esto contrasta con el modelo de concurrencia más común hoy en día, donde se usan hilos del Sistema Operativo. Las operaciones de redes basadas en hilos son relativamente ineficientes y son muy difíciles de usar. Además, los usuarios de Node están libres de preocupaciones sobre el bloqueo del proceso, ya que no existe. Casi ninguna función en Node realiza I/O directamente, así que el proceso nunca se bloquea. Debido a que no hay bloqueo es muy razonable desarrollar sistemas escalables en Node. [25]

4.3 Herramientas utilizadas

Para desarrollar este proyecto se han hecho uso de distintas aplicaciones:

- Visual Studio Code es un editor de código fuente disponible para Windows, macOS y Linux. Junto con su instalación viene incorporado soporte para JavaScript, TypeScript y Node.js y también cuenta con un rico ecosistema de extensiones para otros lenguajes, como C++, Java y Python, entre otros. [27]
- Postman es una plataforma utilizada para el desarrollo de API. A través de esta herramienta se pueden probar las rutas y las peticiones HTTP que hayan sido creadas y comprobar si funcionan o no de la manera esperada, pudiendo establecer parámetros, cabeceras, etc. [26]
- GitLab es un sistema para administrar repositorios de Git. Está escrito en Ruby y permite al usuario implementar un control de versiones para su código de manera sencilla. [28] Se publicó por primera vez en GitHub en octubre de 2011 y desde entonces se ha convertido en una herramienta poderosa. GitLab se publica bajo la licencia MIT.
- Marvel App es una herramienta para crear prototipos interactivos de plataformas digitales. Los prototipos se pueden adaptar para todo tipo de dispositivos y sistemas operativos. Dentro de una página, se pueden agregar áreas de interacción, transiciones y gestos, para un resultado más realista. [29]
- Lucidchart es una herramienta de diagramación basada en la web, que permite a los usuarios colaborar y trabajar juntos en tiempo real, creando diagramas de flujo, esquemas de sitios web, diseños UML... Está construida con estándares web, como HTML5 y JavaScript, Asimismo, funciona en todos los navegadores web modernos, como Google Chrome, Firefox, Safari e Internet Explorer 8+. [30]

4.4 Mock-Ups

En este apartado se mostrarán los diferentes prototipos que se han realizado con el fin de sentar una base sobre el diseño de la aplicación antes de comenzar a trabajar en ella.

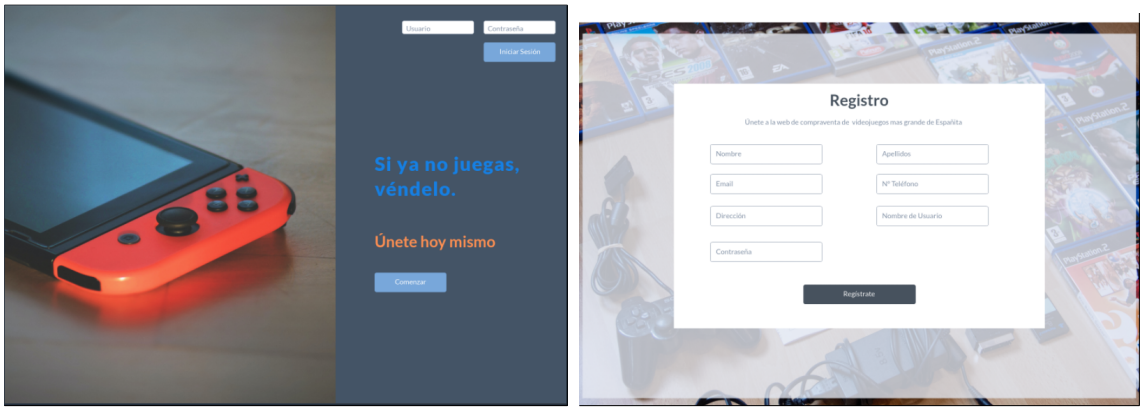

Figura 4.9: Inicio de sesión y registro

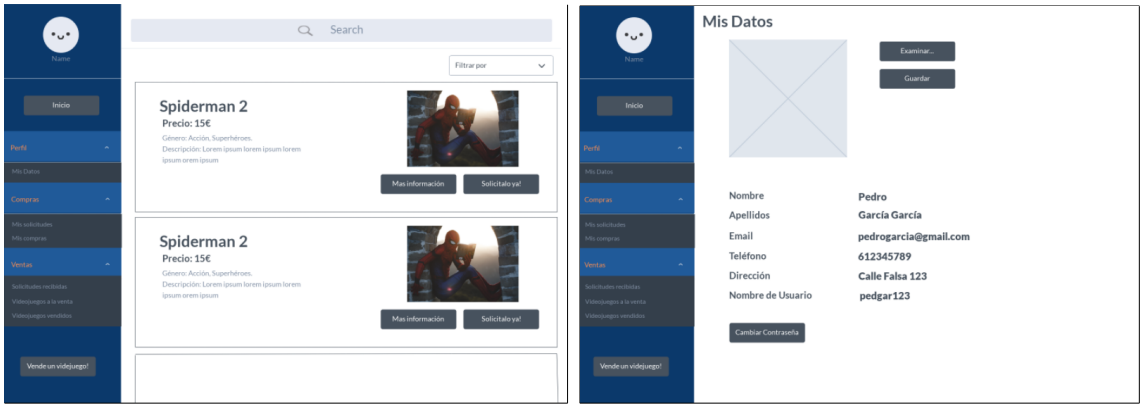

Figura 4.10: Página de inicio y perfil

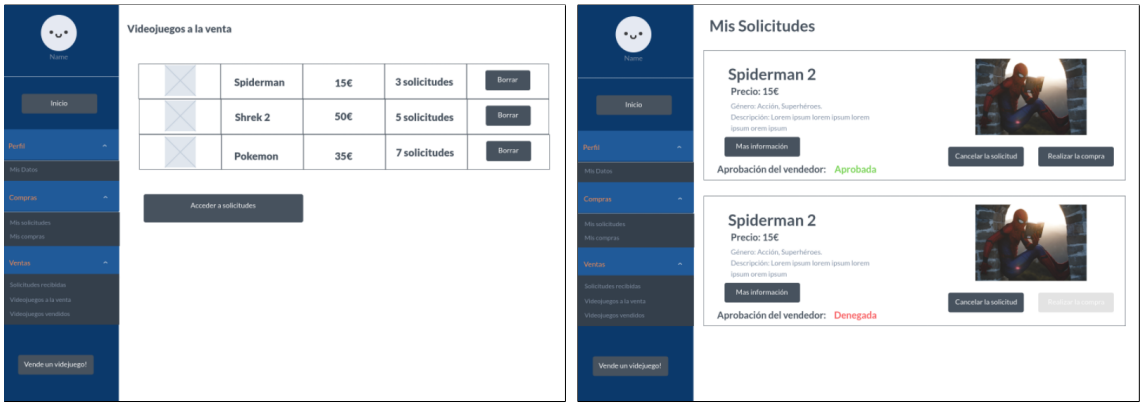

Figura 4.11: Videojuegos a la venta y mis solicitudes

Figura 4.12: Solicitudes de compra y mis compras

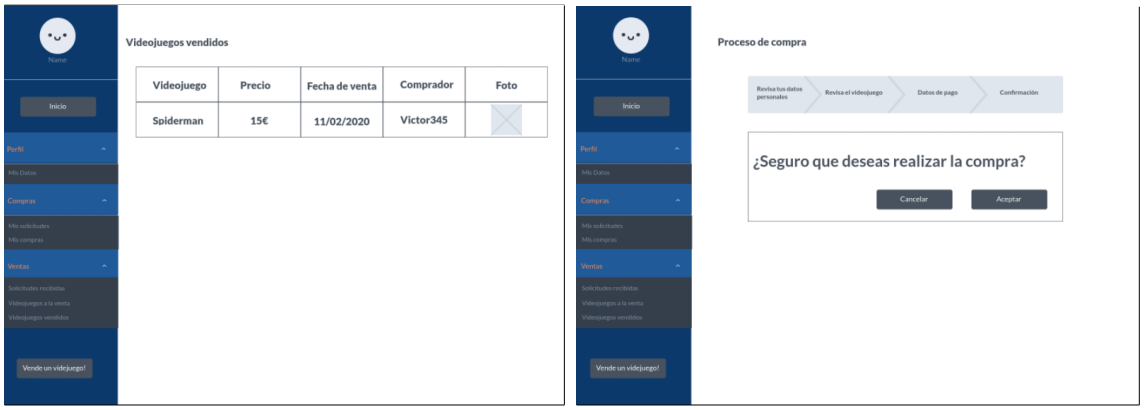

Figura 4.13: Videojuegos vendidos y compra

CAPÍTULO 5
Desarrollo e implementación

En este capítulo se explicará cómo se ha desarrollado la aplicación. Se comentará como se ha estructurado cada una de las secciones en las que se encuentra dividida y su funcionalidad. Sin embargo, habrá ocasiones en las que se expliquen varias de las funciones de manera más extensa que las demás. El motivo viene dado por su complejidad y por la importancia que tienen en la aplicación, diferenciándolas así, de otras funciones consideradas más básicas (como puede ser una consulta a la base de datos).

5.1 Consideraciones iniciales

En esta sección se tratarán algunas consideraciones que hay que tener en cuenta antes de empezar a programar.

Inicialización de los proyectos

Antes de empezar a programar, se deben realizar algunos pasos previos. Primero, es necesario la instalación de Node.js y de npm. A partir de este manejador de paquetes, se podrán instalar los intérpretes por línea de comandos (CLI) de Angular y de Nest.js. Con estos CLI los proyectos se pueden generar y ejecutar de una forma muy sencilla.

```
1  npm -i @angular/cli
2  npm -i @nest/cli
```

Una vez instalados estos intérpretes, se procederá a la generación de los proyectos a partir de los siguientes comandos:

```
1  ng new my-app
2  nest new project-name
```

Con lo que respecta al proyecto de angular, lo único que le hace falta es la instalación del *framework* NGRX. Para ello, simplemente se ejecutará:

```
1  npm install @ngrx/store --save
2  npm install @ngrx/effects --save
```

Por último, queda configurar el proyecto de Nest para que se conecte a la base de datos. Como se ha comentado en el apartado anterior, se va a utilizar MongoDB Atlas, la base de datos de Mongo en la nube.

Una vez creada, tanto la cuenta como el proyecto en Atlas, se deberá crear un usuario. En esta ocasión, el usuario se llamará 'user-TFG' y tendrá todos los permisos disponibles.

Después, Mongo nos proporcionará un enlace (en el que hay datos que deben ser cambiados por el usuario) a través del cual la aplicación *Back-end* podrá conectarse a la base de datos.

Para crear la conexión entre la base de datos y el servidor, se hará uso de Mongoose, anteriormente explicado.

```
//Conexión a la BBDD
const user = 'user_TFG';
const password = 'tiB49BozWueMlMSW';
const bbdd = 'TFG'
const uri = `mongodb+srv://${user}:${password}@cluster0.5ixwk.mongodb.net/${bbdd}?retryWrites=true&w=majority`;
```

```
@Module({
  imports: [MongooseModule.forRoot(uri,{ useNewUrlParser: true, useUnifiedTopology: true })
```

Figura 5.1: Conexión del proyecto de Nest a la base de datos

Módulos

Como se ha comentado anteriormente, tanto Angular como Nest son modulares, por lo que cada uno de los proyectos estarán divididos en módulos. El proyecto de angular estará dividido en dos módulos: el módulo 'login' y el módulo 'cvv'. En el módulo 'login', se encontrará todo lo relacionado con el registro, inicio de sesión y autenticación de usuarios. Por otro lado, el módulo 'cvv' estará compuesto por el resto de la aplicación, es decir, por la parte referente a la compraventa de videojuegos. La razón por la que el proyecto de Angular se encuentra dividido en 2 módulos, y no en más o menos módulos, viene dada por separar ambas funcionalidades, ya que son independientes entre si. A continuación, se presenta como se encuentra dividido un módulo en Angular y se explicará que función tiene cada parte.

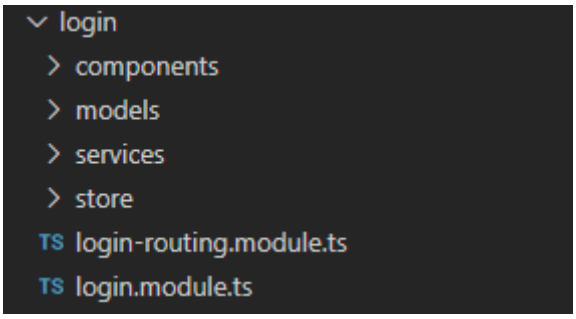

Figura 5.2: Estructura de un módulo de Angular

En la carpeta 'components' se encontrarán los componentes que formarán parte de este módulo. En la carpeta 'models' estarán definidas las entidades, los *mocks* y los enumeradores. Los servicios que se comunicarán con el servidor se encuentran en la carpeta 'services'. Por último, en la carpeta 'store' se encontrará todo lo relacionado con el *Store* de NGRX, es decir, acciones, efectos, *reducers* y selectores, cuya función ya se explicó en el tercer capítulo de esta memoria. El archivo 'xx-routing.module.ts' será donde estén definidas las diferentes rutas referentes a ese módulo. Por último, el archivo 'xx.module.ts' será el archivo donde se encuentren las dependencias del módulo, es decir, donde se importen, exporten y se declaren componentes, servicios, etc.

Por otro lado, el proyecto de Nest contará con un módulo por objeto de negocio. Por lo tanto, existirán cuatro módulos: usuarios, videojuegos, compras y solicitudes. De esta manera se encontrarán separados y ordenados todos los controladores, servicios, etc. que

tengan que ver con un objeto de negocio en específico. A continuación, se explicará cómo se encuentra dividido un módulo en Nest.

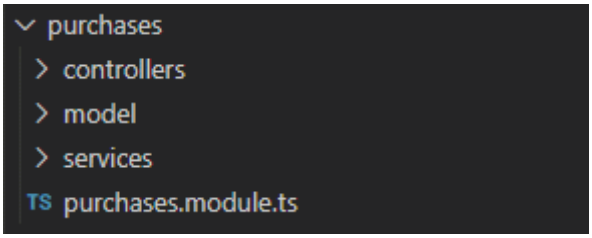

Figura 5.3: Estructura de un módulo de Nest

En la carpeta 'controllers', se encontrarán los controladores que se encargarán de recibir las peticiones. En la carpeta 'model', se encontrarán las entidades que se utilizarán, así como, los diferentes enumeradores que puedan existir y la definición del esquema que representará el objeto de negocio en la base de datos. Los servicios que se encargarán de realizar las operaciones sobre la base de datos se encuentran en la carpeta 'services'. Por último, al igual que ocurría en el proyecto de Angular, el archivo 'xx.module.ts' será el encargado de albergar las dependencias del módulo.

5.2 Registro

La vista de la página de registro está formada por un formulario con los campos necesarios para registrar un usuario. Estos están dotados de validadores que, en caso de estar vacíos o tener un formato erróneo (ej. escribir caracteres en el campo destinado al número de teléfono), harán que el botón de registro se mantenga inactivo. Una vez se proceda al registro, se ejecutará una acción que, como se explicó anteriormente en el ciclo de vida de NGRX [4.8], terminará con la ejecución de una petición POST al servidor *Back-end*. Cuando el servidor reciba la petición deberá comprobar que no existe ningún usuario con un nombre de usuario, teléfono y/o correo electrónico iguales a los que se reciben en el cuerpo de la petición. Si esto ocurriera, se responderá a la petición con un código de error y un mensaje explicativo sobre dicho problema. En caso contrario, el usuario será creado en la base de datos y se enviará como respuesta un código HTTP 200 OK indicando que la operación se ha realizado satisfactoriamente.

Subida de imágenes

En el caso en el que el usuario quiera ponerse una foto de perfil o adjuntar una foto del videojuego que quiere poner a la venta, deberá estar implementado un sistema para poder subir archivos al servidor. Para ello es necesario el uso del paquete 'Multer'. Este es un *middleware* de Node.js utilizado para poder subir archivos de todo tipo al *Back-end*. [14, 15] Una vez se reciba la petición de registrar o editar un usuario o videojuego, se podrá extraer la imagen del cuerpo de la petición a partir del interceptor *FileInterceptor()*.

```
@Post('/createUser')
@UseInterceptors(FileInterceptor('file', {
  storage: diskStorage({
    destination: './upload',
    filename: renameImage,
  }),
  fileFilter: fileFilter,

}))
```

Figura 5.4: Interceptor de Multer

Asimismo, este interceptor tiene una serie de opciones que se pueden configurar para, entre otras cosas, elegir el destino de las imágenes, el nombre que estas tendrán y/o los formatos de archivos que son admitidos. Para controlar que las imágenes no puedan tener el mismo nombre y se sobrescriban, se utilizará la función *renameImage()*. Esta renombrará la imagen a partir de un código aleatorio y su nombre original.

```
export const renameImage = (req, file, callback) => {
    const name = file.originalname.split('.')[0];
    const fileName = file.originalname;
    const randomName = Array(4).fill(null).map(() => Math.round(Math.random() * 16).toString(16)).join('');
    callback(null, `${name}-${randomName}${fileName}`);
}
```

Figura 5.5: Función renameImage.

Por otro lado, la función *fileFilter()* servirá para comprobar que el archivo subido es una imagen. En caso contrario, lanzará un error.

```
export const fileFilter = (req, file, callback) => {
    if(!file.originalname.match(/\.(jpg|jpeg|png|gif|PNG)$/)){
        return callback( new Error('Invalid format type'), false)
    }

    callback(null, true);
}
```

Figura 5.6: Función renameImage.

5.3 Inicio de sesión

La página de inicio de sesión contará con un simple formulario donde el usuario deberá introducir sus credenciales, es decir, su nombre de usuario y su contraseña. Una vez se valide el formulario, se realizará la petición al servidor que se encargará de comprobar si el nombre de usuario y la contraseña son correctos.

5.3.1. Autenticación

Para realizar la comprobación de los usuarios se hará uso de una librería específica: Passport. Este es un *middleware* de autenticación para Node.js. Se encargará de autenticar

al usuario a partir de sus credenciales y lo hará a través de un conjunto de complementos conocidos como 'estrategias'. [12, 13]

A su vez, otra pieza fundamental para la autenticación serán los *guards*. Estos son clases cuya única función será determinar si una petición acabará siendo procesada o no, dependiendo de una serie de condiciones. [12]

Por último, también hará falta hacer uso del paquete *passport-jwt*. Este paquete permitirá cifrar *tokens* a partir de los datos del usuario e implementará la estrategia *JWT* que posibilitará la protección de rutas. [12]

```
@UseGuards(AuthGuard('local'))
@Post('auth/login')
async login(@Request() req) {
  return this.authService.login(req.user);
}
```

Figura 5.7: Función de autenticación

Una vez se envíe la petición para iniciar sesión desde el *Front-end* se ejecutará el *guard* contenido en el decorador *@UseGuards*. Esta clase *AuthGuard* actuará como una barrera, permitiendo ejecutar el resto del código si la estrategia a la que hace referencia devuelve una respuesta satisfactoria o no permitiéndolo en caso de que devuelva un error. Esta estrategia se encargará de comprobar que el usuario es válido y generará un error en caso contrario.

```
export class LocalStrategy extends PassportStrategy(Strategy) {
  constructor(private authService: AuthService) {
    super();
  }

  async validate(username: string, password: string): Promise<any> {
    const user = await this.authService.validateUser(username, password);
    if (!user) {
      throw new UnauthorizedException();
    }
    return user;
  }
}
```

Figura 5.8: Función de autenticación

A partir del usuario devuelto por la estrategia y de la librería *JWT* se procederá al cifrado de la información del usuario para generar un *token* que será devuelto como respuesta de la petición.

```
async login(user: any) {
  const realUser = user._doc;
  const payload = { username: realUser.username, sub: realUser._id };
  const access_token =  this.jwtService.sign(payload);
  const token : Login = {
    auth_token: access_token,
  }
  return token;
}
```

Figura 5.9: Cifrado de un *token* a partir de la información del usuario

El *Front-end*, al recibir el *token*, lo guardará en el *localStorage* del navegador y a su vez, lo incorporará en las cabeceras de las próximas peticiones que realice. Para ello, se hará uso de la librería *HttpInterceptor* y del siguiente método *intercept()*:

```
intercept(req: any, next: any) {
  const token = localStorage.getItem('token');
  const headerTokenized = req.clone({
    setHeaders: {
      Authorization: `Bearer ${token}`,
    },
  });
  return next.handle(headerTokenized);
}
```

Figura 5.10: Funcion intercept() que añadirá el *token* a las cabeceras de las peticiones.

Al incorporar el *token* en las cabeceras de las peticiones, el servidor será capaz de descifrar el *token* a partir de la estrategia *JWT* y validar que el usuario que ha hecho la petición es un usuario válido, protegiendo así las rutas del servidor.

Por otro lado, al incorporar el *token* al *localStorage* del navegador, es posible proteger las rutas de la aplicación *Front-end*. A través de la directiva *canActivate* se pueden ejecutar *guards* al navegar entre las páginas.

```
{
  path: 'login',
 loadChildren: () => import('./login/login.module').then((m) => m.LoginModule),
  canActivate:[LoggedGuard]

},
{
  path: 'cvv',
  loadChildren: () => import('./cvv/cvv.module').then((m) => m.CvvModule),
  canActivate: [AuthGuard]
},
```

Figura 5.11: Guards que controlan la navegación entre rutas en el *Front-end*

Si se intenta acceder a alguna ruta de la aplicación en la que sea necesario haber iniciado sesión, se comprobará si en el *localStorage* existe el *token*. Por el contrario, si se ha

iniciado sesión y se intenta volver a la página de registro o inicio de sesión, se comprobará que el *token* no exista.

5.4 Panel de acceso rápido

El panel de acceso rápido es un elemento que estará siempre disponible para el usuario desde el momento en que inicie sesión. En ella se encuentra la foto de perfil del usuario, así como, accesos directos que lo redirigirán a las diferentes pestañas de la aplicación. Es importante destacar que cada vez que se entre en una de las pestañas se mandará una petición al servidor para recuperar los datos relativos a ella. Ej. cuando se entre en 'Inicio' se recuperarán los videojuegos que se encuentran a la venta. De esta manera los datos se mantendrán lo más actualizados posible.

5.5 Mi perfil

En este apartado el usuario podrá consultar su información personal y actualizarla si así lo desea. Cuando se presione el botón 'Editar', la vista cambiará para mostrar un formulario con los datos actuales del usuario que podrán ser modificados siguiendo las reglas vistas en el registro. En el caso de querer cambiar la contraseña, se abrirá un cuadro de dialogo donde se deberá indicar la contraseña actual y la nueva contraseña deseada. Al ejecutarse la acción de edición, el servidor recibirá la petición *PUT* para actualizar el usuario. De la misma manera que en el registro, se comprobará que tanto el correo electrónico, como el teléfono y el nombre de usuario, no pertenezcan a otra persona ya registrada. Una vez comprobado que todo está en orden, el servidor llevará acabo la modificación del usuario en la base de datos.

Asimismo, es necesario destacar el uso del atributo *isEditing* del usuario para controlar la edición de datos. Existe la posibilidad que el usuario haya iniciado sesión en dos navegadores y se intente editar de forma simultánea. Para evitar esta situación, cuando se intente acceder a la edición, se comprobará, a través de una petición al *Back-end*, que este campo es falso. En ese caso, el campo cambiará a verdadero y el usuario podrá modificar su información. En caso contrario, el formulario no se activará y se mostrará un mensaje de error.

5.6 Registrar un videojuego y mis videojuegos a la venta

El apartado destinado a registrar un videojuego es muy similar al registro de usuarios. Esta vista contará con un formulario donde rellenar los datos relativos al videojuego que se quiere poner a la venta. Al registrar el videojuego, el servidor será el encargado de recibir los datos y crear el objeto en la base de datos. Una vez el servidor devuelva una respuesta satisfactoria, la aplicación redirigirá al usuario a la pestaña 'Mis videojuegos a la venta'. Esta página, como su nombre indica, mostrará una lista con todos los videojuegos que el usuario tenga a la venta. Esta lista se basará en una iteración de un mismo componente que representará al videojuego donde podrán observarse todos los detalles de este. En el caso en el que se quiera modificar alguno de los datos sobre el videojuego se hará uso de la misma mecánica vista en el apartado 'Mi perfil' para evitar que se pueda editar un videojuego en dos pestañas al mismo tiempo. A su vez, se reutilizará el formulario utilizado al registrar videojuegos para mostrar los datos de este y poder modificarlos.

Por último, los videojuegos podrán ser borrados si el usuario así lo desea. En caso de haber recibido solicitudes sobre el videojuego que se vaya a borrar, el estado de estas pasará a 'Rechazada'.

5.7 Página de inicio

La vista de la página de inicio está compuesta por diferentes componentes. Por un lado, se encuentra el componente dedicado al filtrado de videojuegos. Se trata de un formulario donde el usuario puede indicar uno o más campos por los que quiere buscar videojuegos. Estos campos, como ya se vio en el apartado 'Registro de usuarios', contienen validadores que deshabilitarán el botón de filtrado cuando tengan un formato erróneo. Una vez que el usuario pulse el botón 'Filtrar', se enviará una petición al servidor. Esta petición contendrá en su cuerpo el conjunto de atributos por los que se requiere buscar. El servidor se encargará de recibir la petición, ejecutar la operación de consulta a la base de datos y devolver como respuesta una lista de los videojuegos encontrados. En caso de no encontrar ninguno, devolverá una lista vacía.

Por otro lado, la página también cuenta con una lista de videojuegos. Esta lista se forma a partir de la iteración de un mismo componente – que representa al videojuego – y se repite un número de veces igual al número de videojuegos recuperados por el *Back-end*. Cabe destacar que el formato de este componente diferirá en función de si se muestran todos los videojuegos o si se ha filtrado por algún campo. En el caso en el que se muestren todos los videojuegos, se mostrará con un componente representado por la foto del videojuego, junto con su nombre, su precio y un botón de solicitud. Si se quiere más detalle de este, basta con pulsar en la foto o en el nombre para redirigir al usuario a una página donde ver un componente más detallado del videojuego – componente reutilizado del apartado 'Mis Videojuegos' –. Sin embargo, si se ha filtrado por algún campo, el componente utilizado en la lista será el utilizado en la vista de detalle. Esta decisión ha sido tomada en base a que puede ser contraproducente que el usuario se encuentre una gran cantidad de videojuegos con mucho detalle nada más iniciar sesión. Por otro lado, cuando realice una búsqueda, el número de videojuegos a mostrar se reducirá considerablemente por lo que se podrá mostrar al usuario una lista detallada sin producir un efecto de agobio.

Una vez que el usuario haya encontrado el videojuego que le interesa podrá crear una solicitud sobre él. El servidor recibirá la petición y registrará la solicitud en la base de datos. Si el usuario ya hubiera realizado una petición sobre ese mismo videojuego, la petición devolvería un mensaje de error.

5.8 Solicitudes realizadas y solicitudes recibidas

Tanto la página de 'solicitudes realizadas' como la de 'solicitudes recibidas' reutilizarán el mismo componente. Se podrán realizar diferentes acciones sobre las solicitudes según en la vista en la que se encuentren. En la vista de 'solicitudes recibidas', el usuario podrá aceptar o rechazar solicitudes. Sin embargo, en la vista de 'solicitudes realizadas', el usuario podrá borrar las solicitudes o proceder – en caso de que se pudiera – a la compra del videojuego. A continuación, se procederá a explicar cada una de las acciones que se pueden hacer sobre las solicitudes (sin incluir la compra) y la implicación que pueden tener sobre otras solicitudes y sobre los videojuegos.

- **Aceptar una solicitud:** al aceptar una solicitud, su estado cambiará a 'aprobada'. Esta acción tendrá las siguientes repercusiones: El usuario que haya realizado la solicitud podrá acceder a la compra del videojuego. Si el usuario vendedor había recibido alguna solicitud más por ese videojuego, estas cambiaran su estado a 'suspendida' y el botón de aceptar se bloqueará. El videojuego sobre el que se ha aprobado la solicitud desaparecerá de las búsquedas y de las páginas de inicio de todos los usuarios de la aplicación para así evitar recibir más solicitudes sobre él. Por último, no se podrá ni editar ni borrar los videojuegos que tengan una solicitud aceptada. De esta manera se protege al comprador para evitar que el vendedor pueda cambiar los datos del producto entre que la solicitud es aceptada y se realiza la compra.

- **Rechazar una solicitud**: al rechazar una solicitud, su estado cambiará a 'rechazada'. Esta acción implicará lo siguiente: la solicitud desaparecerá de la lista de solicitudes recibidas del vendedor, pero seguirá estando en la lista del comprador como rechazada. Así, el usuario comprador sabrá que su solicitud ha sido denegada y no desaparecerá de su lista directamente. Si la solicitud que se rechaza estaba previamente aprobada, todas las que estaban suspendidas, pasarán de nuevo a su estado original. Por lo tanto, el vendedor podrá elegir –o no – entre otra de las solicitudes ya existentes. Hasta que se vuelva a aprobar otra solicitud, el videojuego volverá a ser visible para el resto de los usuarios. Finalmente, volverá a ser posible la edición y el borrado del videojuego.

- **Eliminar una solicitud**: al eliminar una solicitud se borrará directamente de la base de datos. Con lo cual, desaparecerá directamente de la lista de solicitudes realizadas y recibidas. Si el estado de la solicitud era 'aprobada', el efecto sobre el resto de solicitudes y del videojuego, será el mismo que si se hubiera rechazado una solicitud aprobada.

5.9 Compra

Como se especificó en los requisitos, la implementación de una pasarela de compra segura no iba a formar parte de este proyecto. Por lo tanto, el proceso de compra será ficticio.

Una vez una solicitud sea aprobada, el usuario que la ha realizado podrá acceder a la compra del videojuego. Al pulsar en el botón 'comprar', se comprobará mediante una petición al servidor que la solicitud siga teniendo el estado 'Aprobada' o que no se haya borrado. Esta verificación debe realizarse para asegurar que la solicitud no haya sido borrada desde otra pestaña o que el vendedor haya anulado la aprobación de esta justo antes de comenzar el proceso de compra. A pesar de ser situaciones con un bajo índice de probabilidad su comprobación es tan necesaria como las demás. Una vez realizada esta comprobación, aparecerá ante el usuario un formulario con tres partes. En la primera, aparecerán sus datos personales que podrá cambiar si así lo desea (por ejemplo, si quiere que el envío se haga a nombre de otra persona). Seguidamente deberá revisar que los datos del videojuego sean los correctos. Es responsabilidad del comprador asegurarse de ello antes de efectuar la compra. Por último, deberá rellenar los datos de su tarjeta de crédito/débito para efectuar la compra. Una vez los datos del formulario sean correctos, el botón de compra será habilitado. Al pulsar en él, volverán a comprobarse las condiciones del principio y si está todo correcto, el pedido se efectuará. Tras ello aparecerá un mensaje de confirmación y un botón que redirigirá al usuario al apartado 'Mis Compras'.

5.10 Mis compras y mis ventas

Como ocurría en las 'solicitudes realizadas' y 'solicitudes recibidas', se reutilizará el mismo componente para ambas vistas. Estas páginas serán informativas y mostrarán un histórico de las compras o las ventas que el usuario ha hecho a lo largo del tiempo.

CAPÍTULO 6
Pruebas

Para realizar las pruebas se procederá a hacer un recorrido a lo largo de la aplicación para comprobar que tanto los requisitos como los casos de uso expuestos en el segundo capítulo de esta memoria se han cumplido. Para ello, hará falta ir cambiando de usuario a lo largo del recorrido. De esta manera se podrá comprobar que todas las funcionalidades se ejecutan correctamente.

En primer lugar, se comenzará registrando un usuario. Para ello se rellanarán los datos que se observan en el formulario. En este caso, se intentará registrar con un nombre de usuario ya perteneciente a otra persona. Con ello se comprueba el caso de uso 'Registrar un usuario: inválido'.

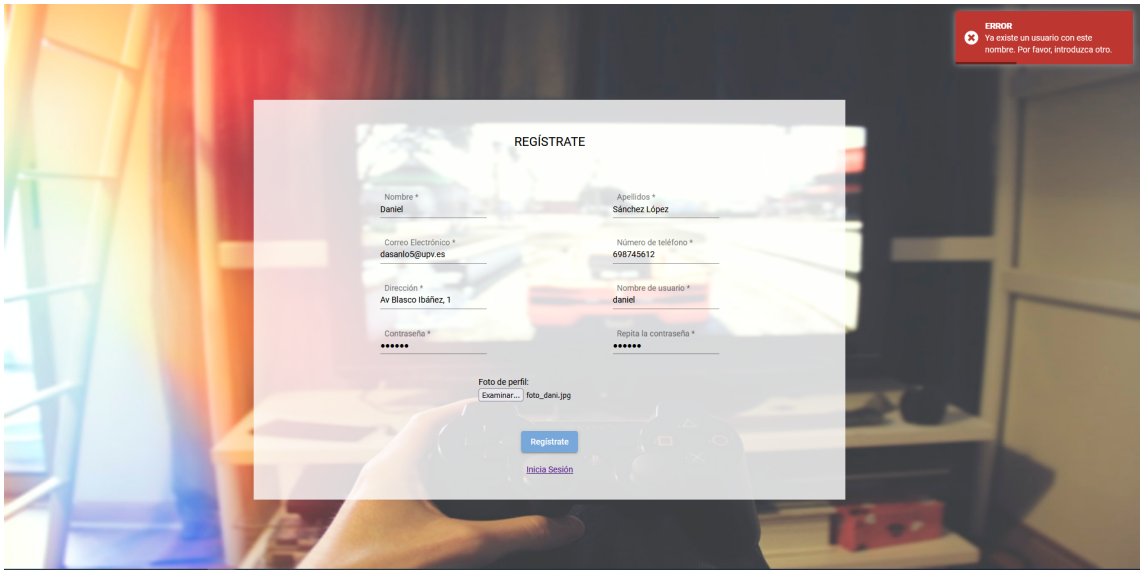

Figura 6.1: Registro fallido de un usuario en la aplicación.

De esta manera se comprueba que no es posible registrarse en la aplicación si ya existe un usuario con ese nombre de usuario. Ocurriría de la misma manera con el teléfono y con el correo electrónico. Ahora se procederá a registrar el usuario con un nombre de usuario válido, probando así el caso de uso 'Registrar un usuario: válido'.

Figura 6.2: Registro satisfactorio de un usuario en la aplicación.

El usuario se registra de forma satisfactoria tal y como se puede ver en la notificación y se redirige al usuario a la pantalla de inicio de sesión. Si se inicia sesión con credenciales erróneas, no se permitirá al usuario a acceder a la aplicación, comprobando así que el caso de uso 'Iniciar sesión: inválido' se cumple.

Figura 6.3: Inicio de sesión fallido.

A continuación, se procederá a iniciar sesión con las credenciales escogidas para este usuario, verificando el caso de uso 'Iniciar sesión: valido'. Seguidamente, se redirigirá al usuario a la página principal.

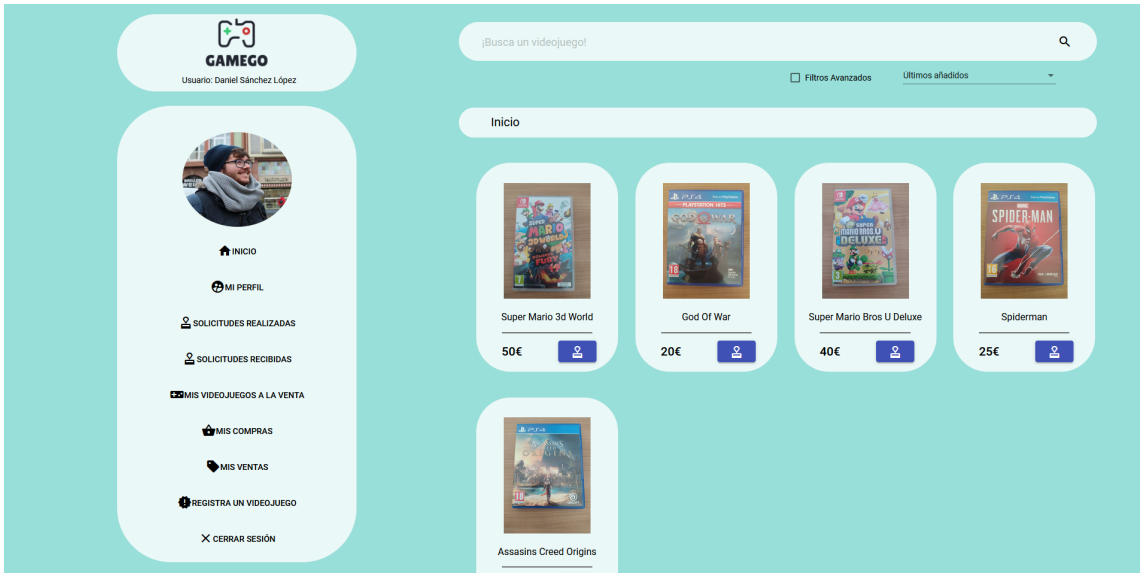

Figura 6.4: Página principal de la aplicación.

En la pantalla de inicio aparecen los videojuegos que se encuentran disponibles para solicitar. Haciendo clic en la foto se mostrará una pantalla donde aparece más información del videojuego en cuestión.

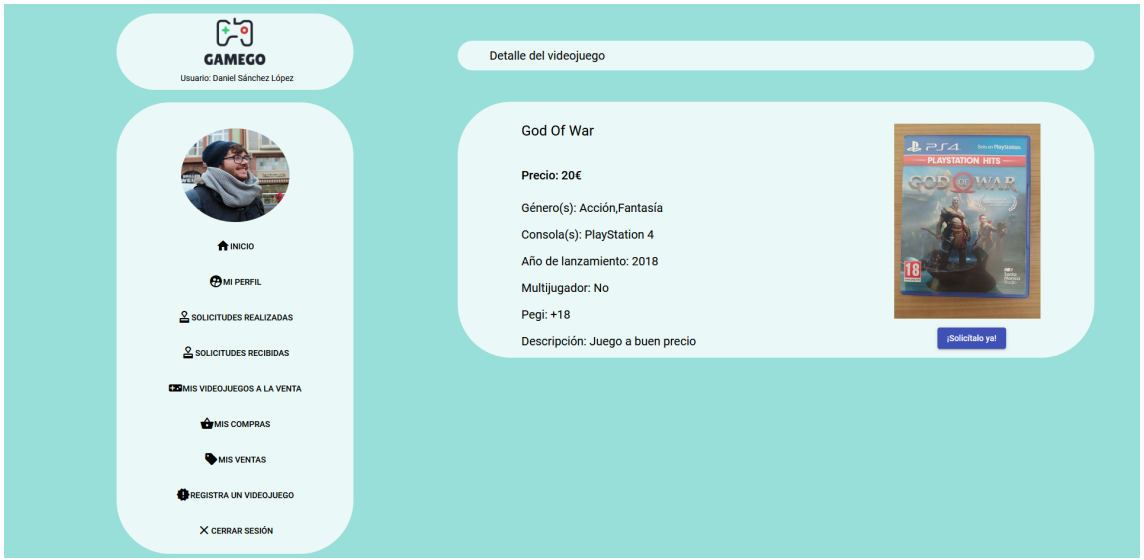

Figura 6.5: Página de detalle de un videojuego

Una vez el usuario quiera realizar una búsqueda más extensa, se dirigirá a los filtros. Por ejemplo, en este caso se quieren filtrar los videojuegos que contengan el género de plataformas y sea para la consola 'Nintendo Switch'.

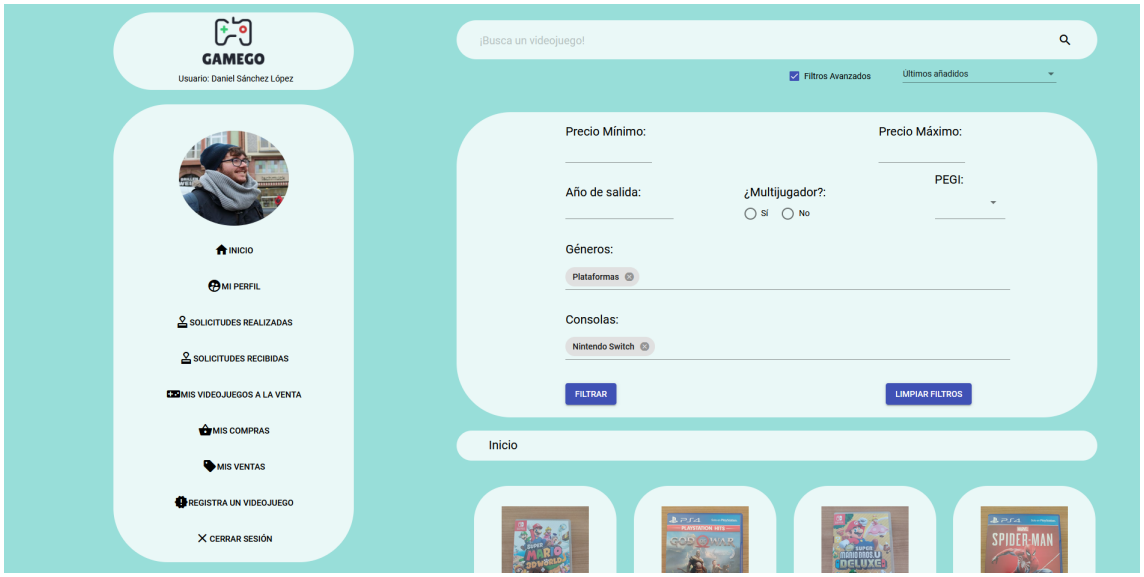

Figura 6.6: Filtros

Los videojuegos obtenidos de este filtrado contarán con un formato igual al de la pestaña de detalle. De esta manera se probará el caso de uso 'Filtrar videojuegos'.

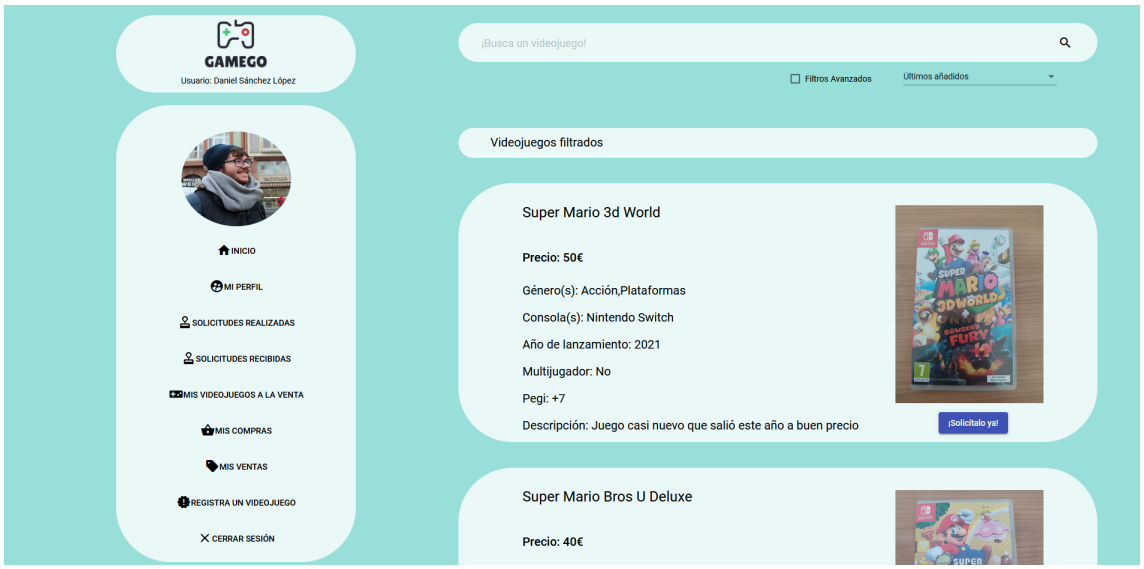

Figura 6.7: Resultado del filtrado

Si el usuario está interesado en un videojuego podrá solicitarlo. En este caso se solicitarán los videojuegos 'God Of War' y 'Spiderman'. Se realizará una oferta de 15€ y 20€ respectivamente.

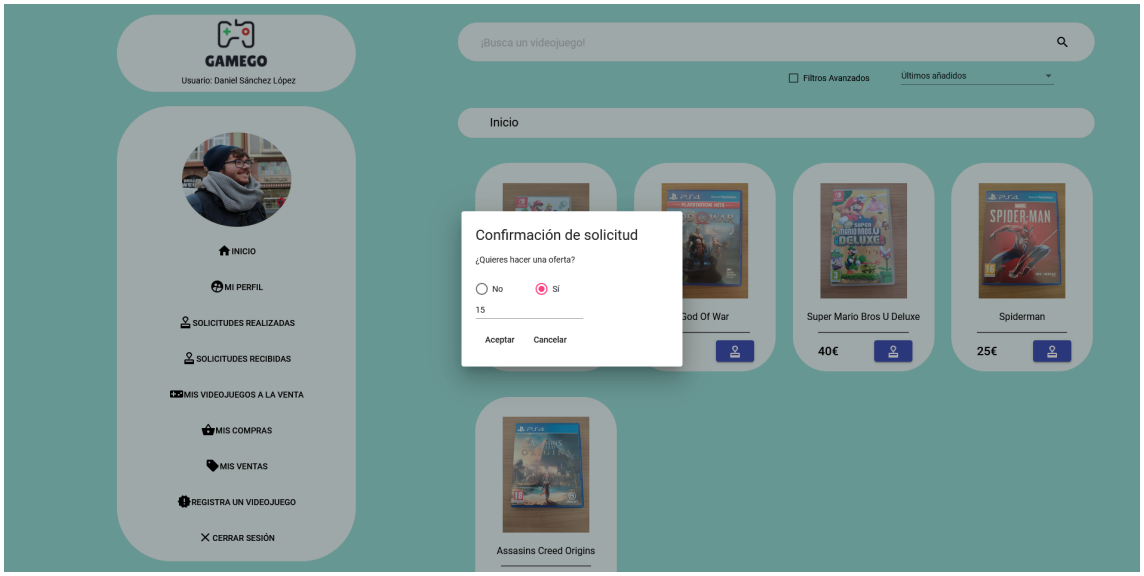

Figura 6.8: Solicitar un videojuego

Al no haberse realizado previamente una solicitud sobre estos videojuegos, estas se crearán correctamente y se redirigirá al usuario a la pestaña de 'Solicitudes Realizadas' donde se encontrarán las solicitudes que acaba de efectuar. De este modo, se comprueba que el caso de uso 'Realizar una solicitud: válido' se cumple.

Figura 6.9: Apartado de solicitudes realizadas

Por otro lado, si ya se hubiera realizado una solicitud, la aplicación mostraría un mensaje de error indicándolo, verificando así el caso de uso 'Realizar una solicitud: inválido'.

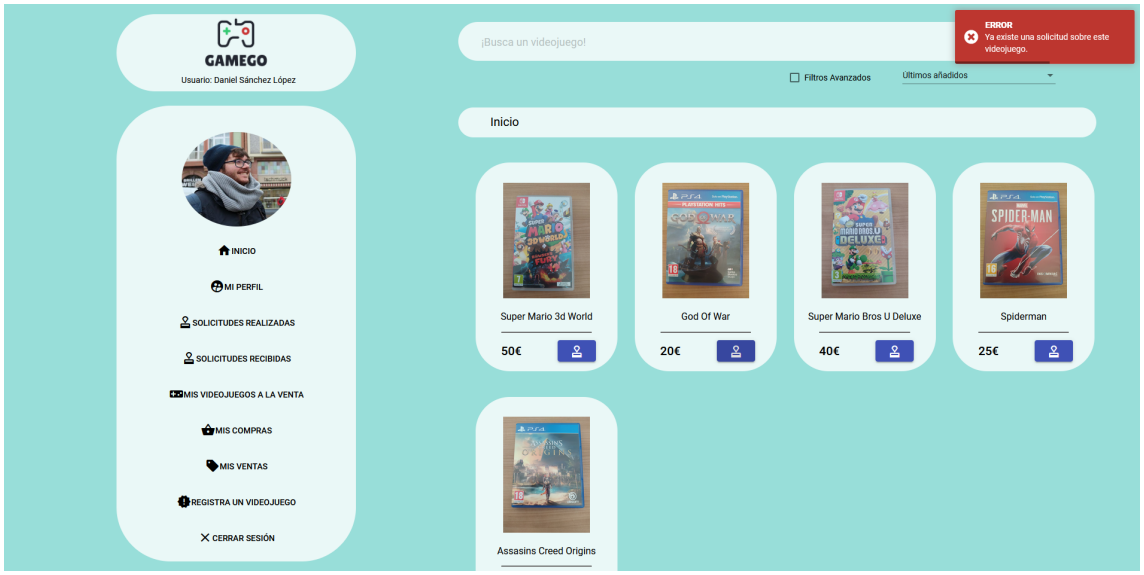

Figura 6.10: Fallo producido por intentar solicitar un videojuego ya solicitado

Como se ha observado en la imagen 6.9, el vendedor del videojuego 'God of War' es el usuario 'joan1'. Si se inicia sesión como este usuario, se puede observar como, en su apartado de 'Solicitudes recibidas', aparece la solicitud que se acaba de realizar junto a otra sobre el mismo videojuego.

Figura 6.11: Apartado de solicitudes recibidas

En este caso se procederá a aprobar la solicitud del usuario que acaba de ser creado, probado así el caso de uso 'Aprobar solicitud'. Como resultado, esta solicitud cambiará su estado a 'Aprobada' mientras que la otra solicitud cambiará su estado a 'Suspendida'.

Figura 6.12: Aprobar solicitud

Asimismo, se rechazará la solicitud que no ha sido aprobada. De esta manera se comprobará que la solicitud desaparece de la pestaña 'Solicitudes recibidas' del vendedor' pero continúa apareciendo en la pestaña de 'Solicitudes realizadas' del comprador, ahora siendo su estado 'Rechazada'. A su vez, se probará el caso de uso 'Rechazar solicitud'.

Figura 6.13: Rechazar una solicitud I

Figura 6.14: Rechazar una solicitud II

Ahora que la solicitud ha sido aprobada, el usuario que ha sido creado al principio podrá realizar la compra del videojuego. Una vez se inicie el procedimiento, aparecerá un formulario donde deberá revisar sus datos y confirmar la compra.

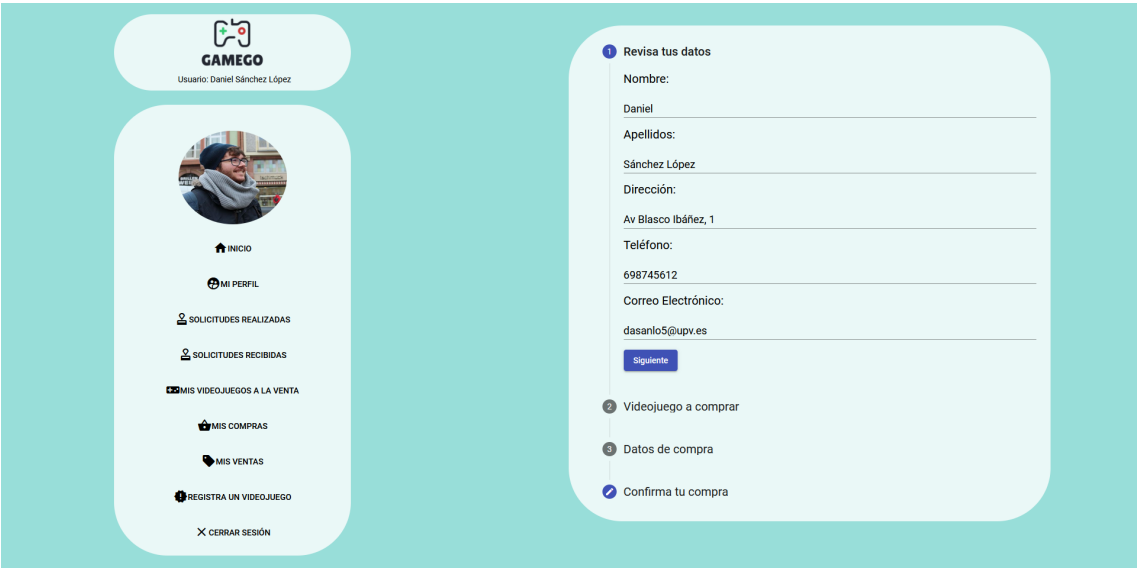

Figura 6.15: Proceder a la compra

Cuando se autorice la compra del videojuego, aparecerá una ventana informativa para informar al usuario de que la compra se ha realizado con éxito. A partir de ahora esta compra figurará en el apartado 'Mis Compras' de la aplicación. De esta manera se compruba que el caso de uso 'Comprar videojuego' se cumple.

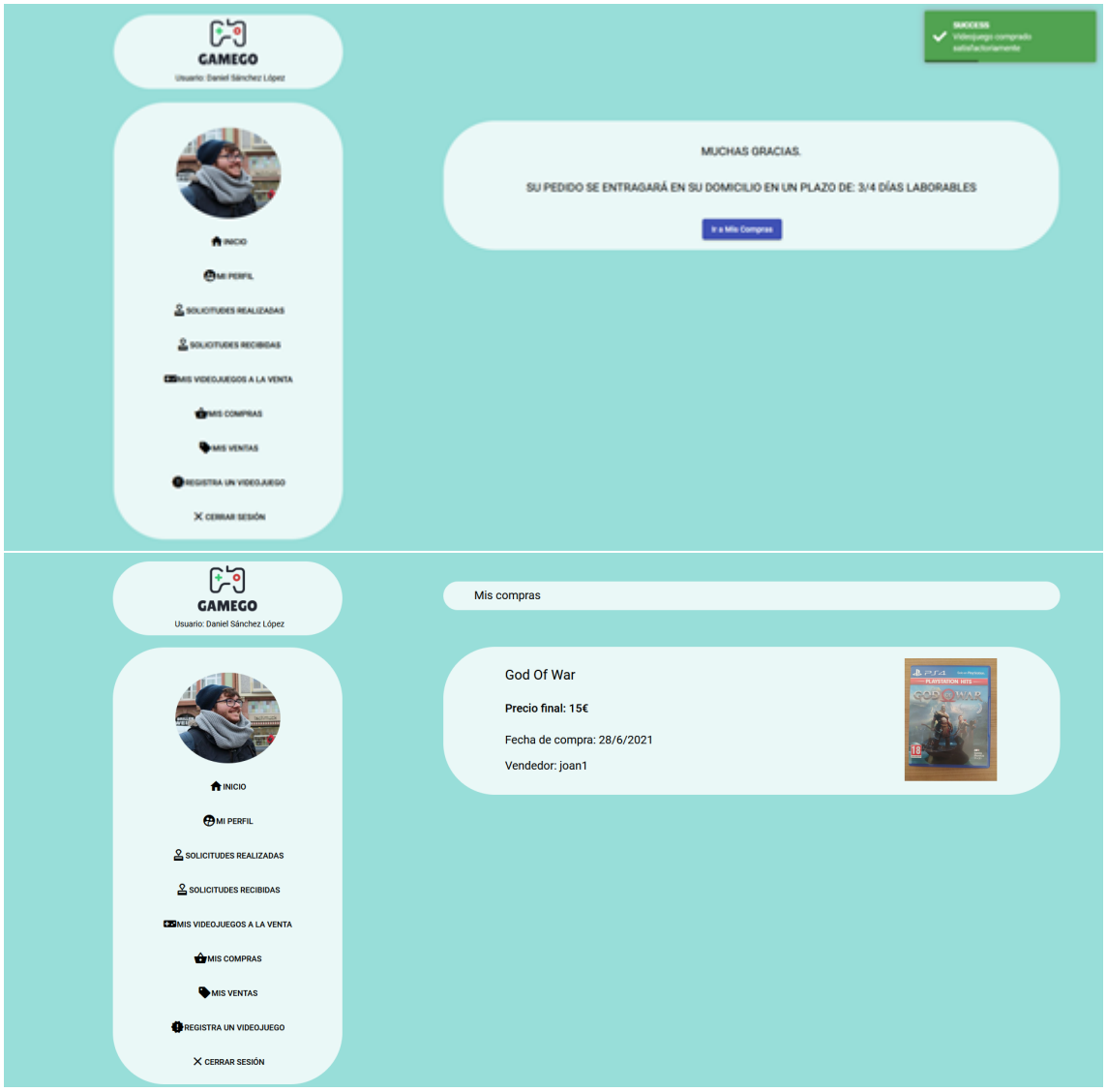

Figura 6.16: Compra Realizada y apartado Mis Compras

Una vez realizada la compra, en el apartado de 'Solicitudes realizadas' del usuario solamente se encontrará la solicitud realizada al videojuego 'Spiderman'. Ahora se procederá al borrado de esta solicitud, comprobando el caso de uso 'Borrar solicitud'.

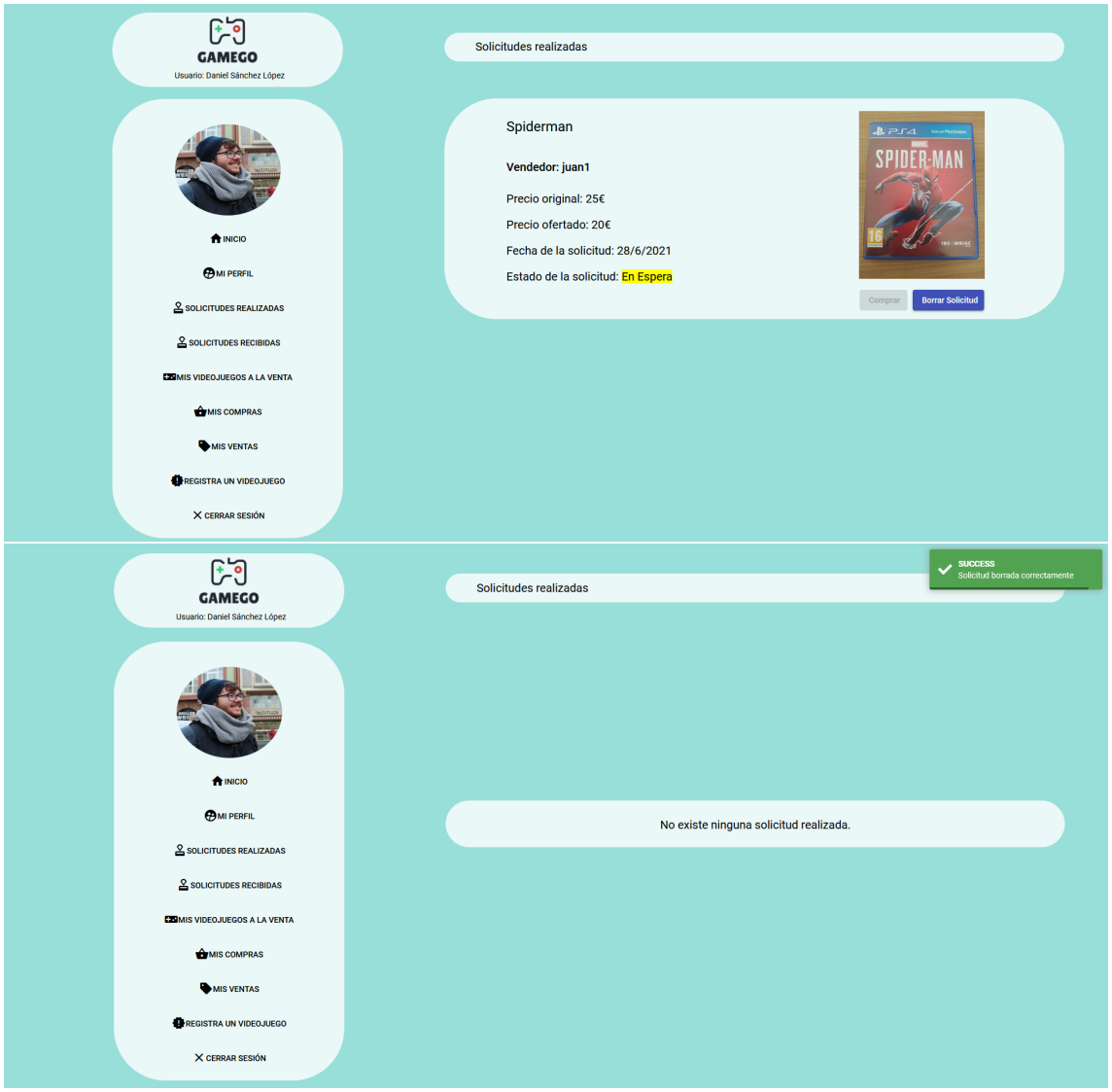

Figura 6.17: Borrado de una solicitud

A continuación, se procederá a poner a la venta un videojuego, verificando así el cumplimiento del caso de uso 'Registrar videojuegos'. Para ello se rellenarán sus datos en el formulario que se muestra a continuación.

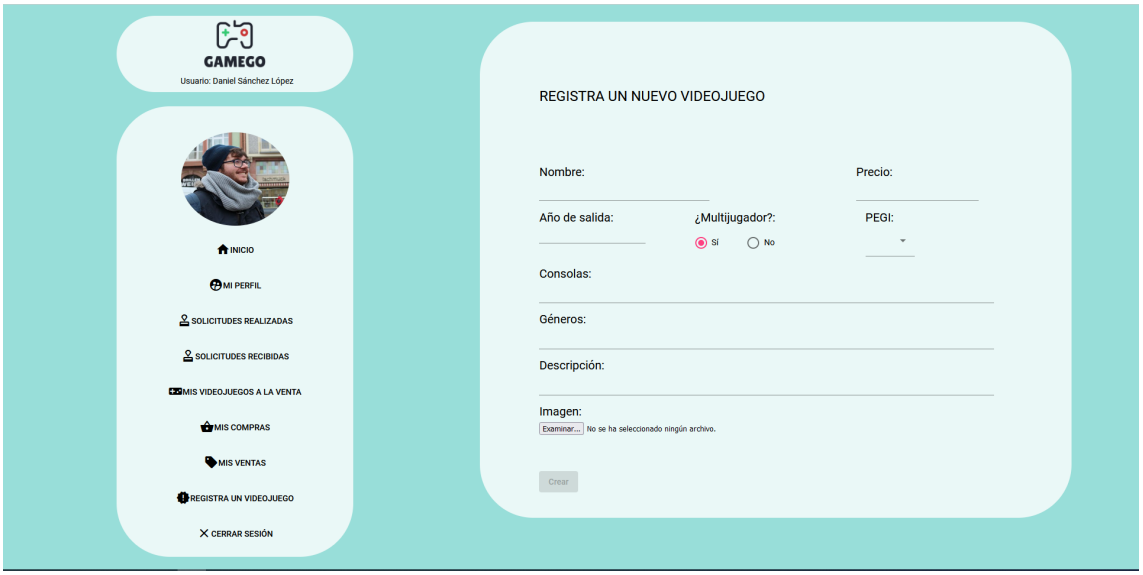

Figura 6.18: Registrar un videojuego

Una vez se registre el videojuego, se redirigirá al usuario a la página 'Mis Videojuegos a la venta'.

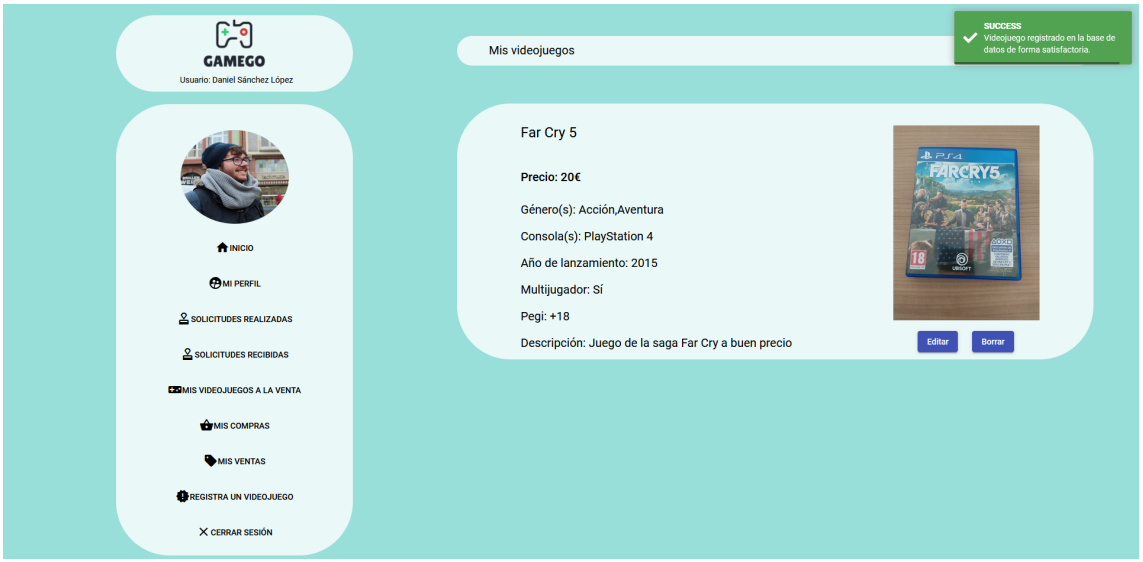

Figura 6.19: Apartado Mis videojuegos a la venta

Si el usuario quiere modificar algún dato del videojuego, se podrá acceder al formulario de edición a través del botón 'Editar'. En este saldrán los datos del videojuego utilizando el formulario anteriormente visto para el registro. Se procederá a cambiar el precio de 20€ a 15€ y se editará el juego. De este modo se prueba que el caso de uso 'Editar videojuegos' se cumple.

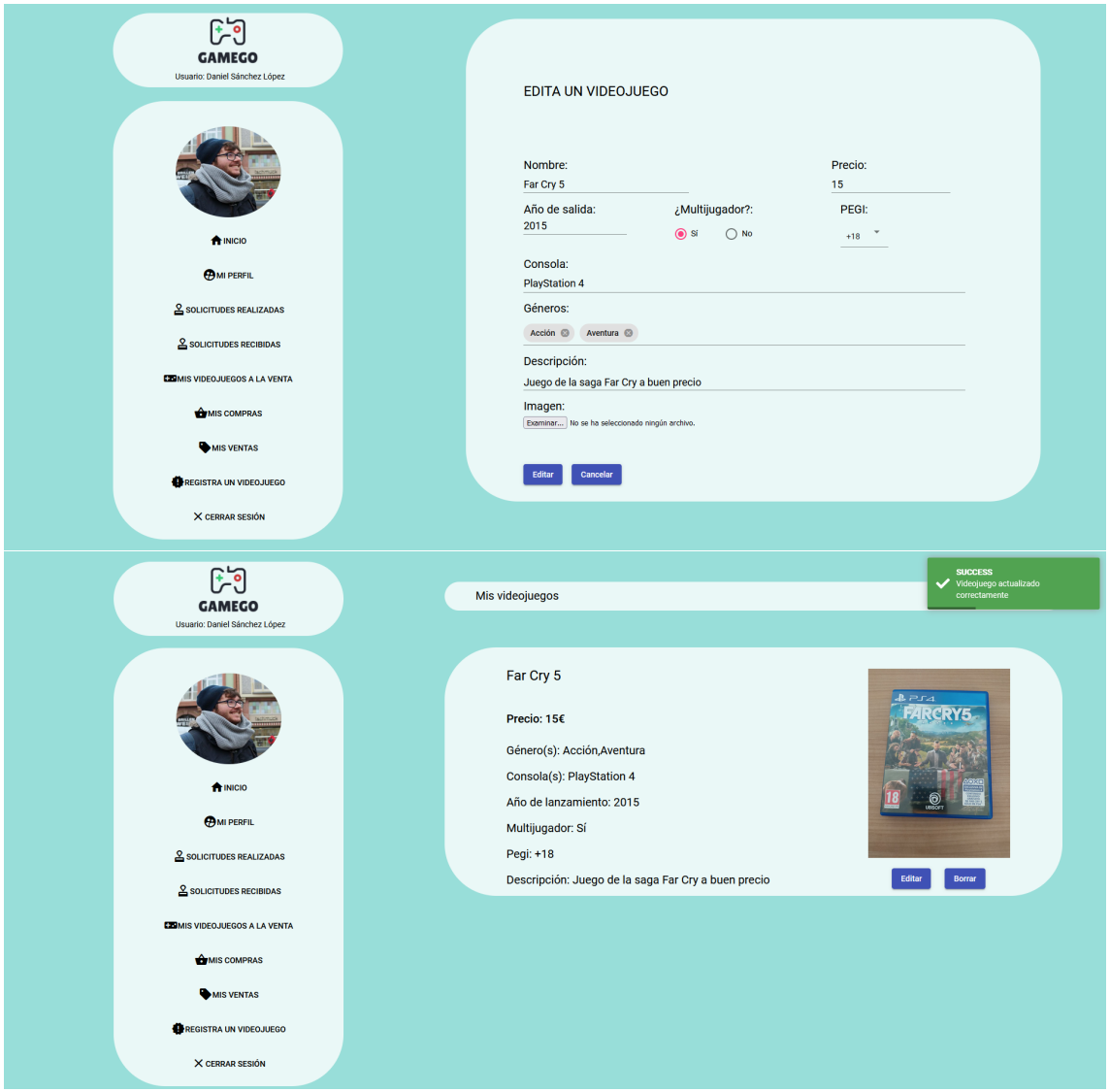

Figura 6.20: Editar un videojuego

Una vez el usuario venda este videojuego, aparecerá en su pestaña de 'Mis ventas'

Figura 6.21: Apartado Mis ventas

El último apartado que queda por repasar en este recorrido es el apartado 'Mi perfil'. Aquí se podrán observar y editar los datos del usuario.

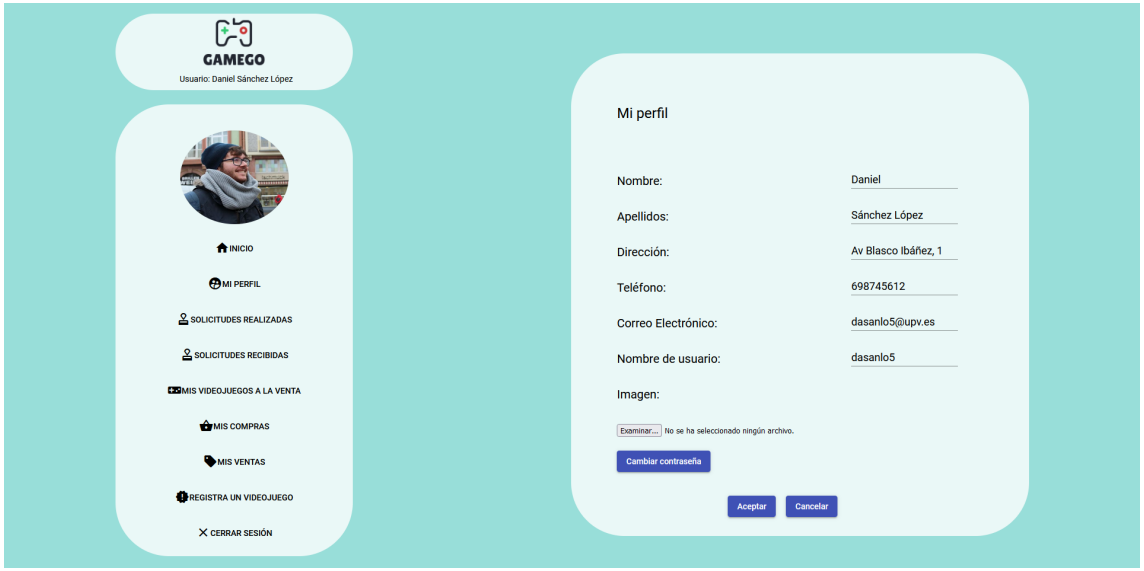

Figura 6.22: Apartado Mi perfil

A continuación, se editará el correo electrónico del usuario cambiándolo por el de un usuario ya existente. Comprobando así el caso de uso 'Editar perfil: inválido'.

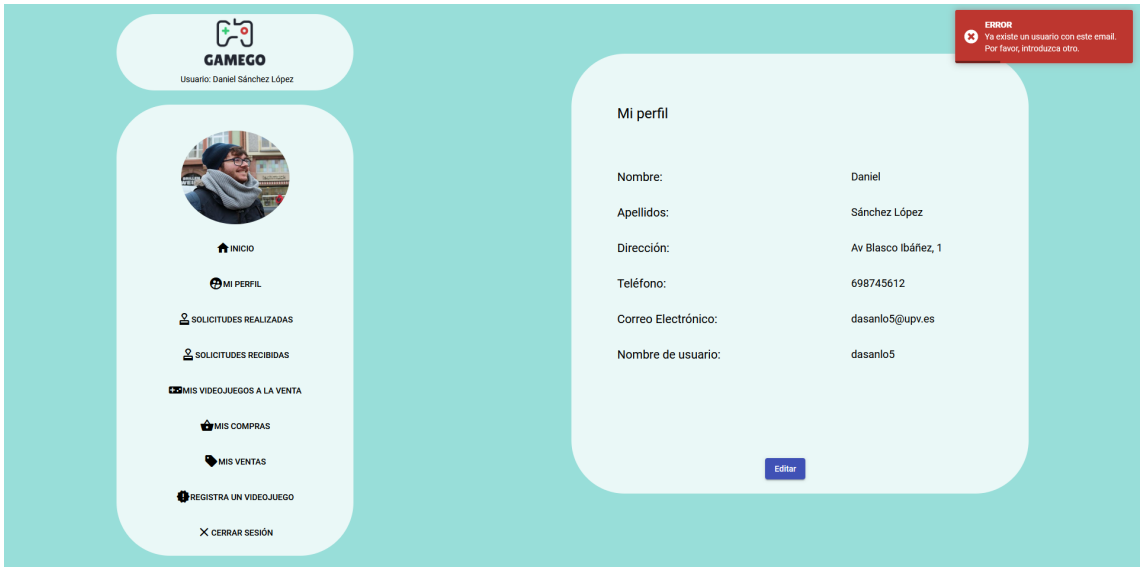

Figura 6.23: Fallo al editar un usuario

Al ya existir un usuario con este correo electrónico, los datos se mantendrán igual y aparece el mismo error que aparecía en el registro de usuarios. Por último, se realizará el cambio de correo electrónico del usuario por uno que sí que sea válido, comprobando así que el caso de uso 'Editar perfil: válido' se cumple.

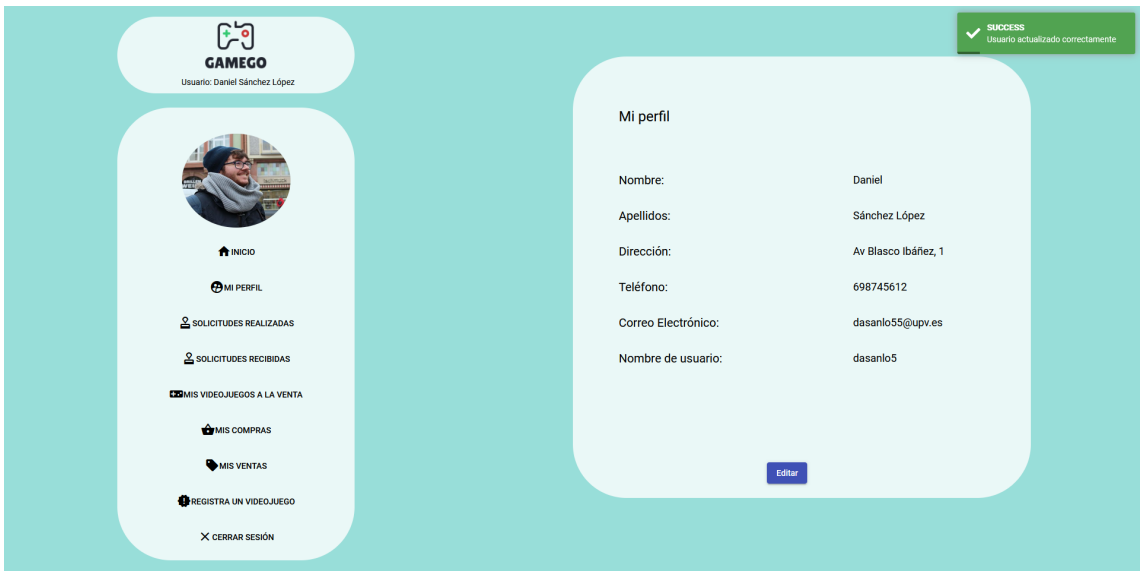

Figura 6.24: Usuario editado satisfactoriamente

CAPÍTULO 7
Conclusiones

En este TFG se ha desarrollado un portal web dedicado a la compraventa de videojuegos de segunda mano. En ella, los usuarios pueden comprar videojuegos a través de un sistema de solicitudes o poner a la venta juegos que ya no utilizan.

Una vez finalizado el proyecto, se puede concluir que el objetivo principal, crear una pagina web, se ha conseguido. A su vez, como se puede observar en el capítulo dedicado a las pruebas, los requisitos que figuran en el capítulo tres de esta memoria se han cumplido en su totalidad. Asimismo, se han añadido funcionalidades que no se contemplaban en los requisitos. Por ejemplo, no permitir que un usuario pueda ser editado en dos pestañas simultáneamente.

También cabe destacar la experiencia que ha aportado la realización de este proyecto de cara a futuros trabajos. Al ser la primera vez que se realizaba una aplicación de estas dimensiones, han ido surgiendo problemas relacionados con la programación de funciones concretas. Sin embargo, esto también ha servido para superarse a uno mismo y, a través de investigación, conseguir resolver el problema. Por otro lado, otro de los problemas que merece la pena mencionar, ha sido la falta de conocimiento sobre diseño gráfico. Esto ha requerido la modificación de la interfaz de la aplicación en distintas ocasiones hasta llegar a un punto en la que esta fuera óptima.

En conclusión, a pesar de los problemas que han ido surgiendo, el desarrollo de este TFG ha sido una experiencia muy fructífera que ha aportado conocimientos que podrán ser empleados de nuevo en el futuro, tanto profesional como académicamente.

7.1 Trabajos futuros

De cara al futuro, se han planteado una serie de adiciones con el fin de mejorar el funcionamiento de la aplicación. Una de las mejoras clave sería el desarrollo de una pasarela de compra segura. Otra mejora sería la implementación de un chat entre vendedor y comprador para mejorar la comunicación entre ambos. También sería recomendable poder enviar correos electrónicos cuando se realiza una compra, a modo de confirmación y la generación de un documento PDF que reflejase la factura de la compra. Asimismo, se planteará el desarrollo de la aplicación para dispositivos móviles.

7.2 Relación del trabajo desarrollado con los estudios cursados y consideraciones sobre el grado

Durante la realización de este proyecto se han puesto en práctica conocimientos obtenidos en distintas asignaturas cursadas en el grado de Ingeniería Informática. Dichas asignaturas son las siguientes:

- Interfaces persona computador (11556)
- Bases de datos y sistemas de información (11548)
- Ingeniería del software (11555)
- Gestión de proyectos (11554)
- Desarrollo web (11610)
- Tecnología de bases de datos (11612)

Por último, lo único que he echado en falta en el grado ha sido la profundización en tecnologías actuales cuyo conocimiento y experiencia son una parte importante de la mayoría de ofertas de trabajo en el mercado laboral actual.

Bibliografía

[1] Valdatti, Bianca. Guía sobre el funcionamiento de las pujas de eBay. *Geekno*. 28 de noviembre de 2019. https://www.geekno.com/guia-sobre-el-funcionamiento-de-las-pujas-de-ebay.html

[2] Vinted. Acerca de Vinted. Recuperado el 22 de abril de 2021. https://www.vinted.es/about

[3] Crear Comunidad. ¿QUÉ ES WALLAPOP Y CÓMO FUNCIONA?. 8 de octubre de 2018. https://crearcomunidad.com/2014/10/08/que-es-wallapop/

[4] Wallapop. Historia de Wallapop. Recuperado el 22 de abril de 2021. https://about.wallapop.com/

[5] Aggarwal, Sanchit, and Jyoti Verma. Comparative analysis of MEAN stack and MERN stack. *International Journal of Recent Research Aspects* 5.1 (2018): 127-32.

[6] Álvarez Caules, Cecilio. Arquitecturas Web y MEAN Stack. *Cantabriatic*. 14 de Agosto de 2014. http://www.cantabriatic.com/arquitecturas-web-y-mean-stack/

[7] Usaola, Macario Polo. *MongoDB: gestión, administración y desarrollo de aplicaciones.* Macario Polo Usaola, 2015.

[8] MongoDB,INC. What Is MongoDB? Recuperado el 24 de abril de 2021. https://www.mongodb.com/es/what-is-mongodb

[9] OpenJS Foundation. Express, Infraestructura web rápida, minimalista y flexible para Node.js Recuperado el 24 de abril de 2021. https://expressjs.com/es/

[10] Rosa, José Manuel *QUÉ ES EXPRESS* Youtube, subido por OpenWebinars. 5 de febrero de 2018. https://www.youtube.com/watch?v=OMzOK7V0k3Q

[11] Trillon.io Nest.js introduction. Recuperado el 24 de abril de 2021 https://docs.nestjs.com/

[12] Trillon.io Nest.js authentication. Recuperado el 25 de junio de 2021 https://docs.nestjs.com/security/authentication

[13] Jared Hanson Passport. Recuperado el 25 de junio de 2021 https://github.com/jaredhanson/passport

[14] Trillon.io File Upload Recuperado el 25 de junio de 2021 https://docs.nestjs.com/techniques/file-upload

[15] MIT Multer. Recuperado el 25 de junio de 2021 https://github.com/expressjs/multer

[16] Google,INC. What is Angular? Recuperado el 24 de abril de 2021. https://angular.io/guide/what-is-angular

[17] Google,INC. Introduction to components and templates Recuperado el 25 de junio de 2021. https://angular.io/guide/architecture-components

[18] Google,INC. Introduction to modules Recuperado el 25 de junio de 2021. https://angular.io/guide/architecture-modules

[19] Google,INC. NgModules Recuperado el 25 de junio de 2021. https://angular.io/guide/ngmodules

[20] NGRX Community What is NgRx? Recuperado el 25 de abril de 2021. https://ngrx.io/docs

[21] Coalla, José Luis Introducción a Redux *Tribalyte Technologies* 8 de noviembre de 2019. https://tech.tribalyte.eu/blog-introduccion-a-redux

[22] Coalla, José Luis Introducción a NgRx *Tribalyte Technologies* 21 de noviembre de 2019. https://tech.tribalyte.eu/blog-introduccion-ngrx

[23] MIT Mongoose. Recuperado el 25 de junio de 2021 https://mongoosejs.com/

[24] Pham, Anh Duc. "Developing back-end of a web application with NestJS framework: Case: Integrify Oy's student management system."(2020).

[25] Tudela Peñarrubia, José Joaquín. "Diseño e implementación con Node. js de una aplicación web para el seguimiento y evaluación del aprendizaje."(2017).

[26] Postman Team The Postman API Platform. Recuperado el 6 de mayo de 2021 https://www.postman.com/api-platform/

[27] Microsoft Visual Studio Code Overview. Recuperado el 6 de mayo de 2021 https://code.visualstudio.com/docs

[28] Hethey, Jonathan M. GitLab Repository Management. Packt Publishing Ltd, 2013.

[29] idaBlog. Ventajas de Marvel App en la creación de prototipos. 4 de mayo de 2017. https://blog.ida.cl/diseno/ventajas-marvel-app-prototipos/

[30] Wikipedia, La enciclopedia libre. Lucidchart. 6 de abril de 2021. https://es.wikipedia.org/w/index.php?title=Lucidchart