uc3m | Universidad Carlos III de Madrid

Doble grado en Ingeniería Informática y Administración de Empresas
2021-2022

Trabajo Fin de Grado

"Diseño y desarrollo de un portal de e-commerce de productos deportivos de segunda mano"

Carlos Salinero Sánchez

Tutor/es

Alejandro Rey López

Colmenarejo, marzo de 2022

Lista de figuras

Lista de tablas

1. INTRODUCCIÓN

1.1 Contexto

En la actualidad, el consumo tan masivo que se hace de internet provoca la necesidad de que todo se requiera al instante, desde leer el periódico hasta realizar pagos a través del portal online del banco en tan solo unos segundos. Casi cualquier tipo de negocio de venta necesita un portal de venta online como respuesta a esta demanda de rapidez y comodidad cada vez más frecuente. La importancia de estos portales en el sector del consumo es algo innegable y se evidencia en datos como que más de 25,8 millones de personas en España (lo que supone el 76% de los usuarios habituales de internet en el país) son compradores habituales en este tipo de portales según el "estudio E-commerce 2021" de IAB Spain. Este estudio también revela que estos usuarios realizan una media de 3,8 compras mensuales por internet en 2021, lo que evidencia un fuerte crecimiento frente a 2019 donde las compras mensuales se situaban en 3 [1].

Un tipo especial de portal de venta online es aquel en el que cualquier usuario puede ser tanto comprador como vendedor. Este tipo de negocio está generalmente orientado a la venta de productos de segunda mano y por lo tanto difiere de un portal común de e-commerce, en que todos los usuarios pueden sacar rendimientos económicos a través de la venta de sus productos. En este trabajo se pretende profundizar en las tecnologías necesarias para crear un portal de venta online de este tipo.

1.2 Idea general y objetivos (revisar y orientar al problema de necesidad de mercado)

El presente proyecto se basa en el diseño y desarrollo de un portal de e-commerce de productos deportivos de segunda mano.

Los objetivos principales del proyecto son los siguientes:

- Realizar un portal de e-commerce que permita a los usuarios dar una segunda vida a aquellos productos deportivos que no utilicen y obtener así un rendimiento económico.
- Realizar un portal de e-commerce que permita:
    - Registrarse e iniciar sesión en el sistema.
    - Visualizar y modificar los datos con los que el usuario se registró.
    - Subir productos añadiendo detalles del mismo. Modificar esos detalles o eliminar el producto.

    - Realizar búsquedas tanto simples en un buscador como avanzadas filtrando por diversos campos.
    - Añadir productos al carrito.
    - Finalizar el proceso de compra.
- Realizar un diseño que permita la escalabilidad de la aplicación.
- Desarrollar un proceso de autenticación robusto que incremente la seguridad de la aplicación.

1.3 Motivación

La elección de este tema está basada en la motivación por ampliar conocimientos acerca de las tecnologías web y de su uso a nivel profesional. Cualquier persona que lea este documento ha realizado con total seguridad una compra a través de internet, pero es posible no sepa cómo se crea un portal web o como se realiza la comunicación entre el navegador y el servidor. Es por ello, que considero muy interesante profundizar en algo tan cotidiano, pero a la vez tan indispensable a día de hoy.

Considero importante destacar que durante el paso por la universidad he adquirido conocimientos sobre distintas áreas como programación, protocolos de comunicación, la importancia de la eficiencia y el rendimiento, la gestión de proyectos software etc y este trabajo ofrece la oportunidad de aunar todos esos conocimientos y ponerlos en común con el fin de sacar adelante este proyecto.

1.4 Estructura del documento

A continuación, se expone con más detalle el contenido que se incluye en cada una de las partes del presente documento.

1. *Introducción.* En este primer capítulo se ha definido la idea general del proyecto y el contexto en el que se encuadra, además de la motivación de la que nace.
2. *Estado del arte.* En este capítulo se presentan diferentes alternativas tecnológicas que, sin entrar en el detalle de la aplicación, podrían ser utilizadas para el desarrollo del portal. Estas tecnologías se presentan agrupadas según la función que realizan dentro de una aplicación.
3. *Planteamiento del problema.* Se presentará la necesidad detectada en el mercado y que soluciones, totales o parciales, existen en la actualidad. Se definirá que es lo que el software debe y no debe hacer, se detallarán las restricciones a las que hacer frente y

se analizarán las características de los usuarios potenciales del sistema. Como parte principal de esta sección se encuentran los requisitos de usuario y de sistema.

4. *Justificación de la solución.* En este capítulo se justificará la elección de las tecnologías previamente mencionadas y se expondrá el diseño completo del sistema, ofreciendo un alto grado de detalle en todos los aspectos. Se incluye, además, una sección dedicada a la evaluación del sistema para verificar su funcionalidad.
5. *Marco regulador.* Se analizará la legislación aplicable a la solución desarrollada, se expondrán cuestiones relacionadas con licencias de productos utilizados y propiedad intelectual, y se mencionarán qué estándares técnicos han sido utilizados.
6. *Entorno socio*-económico. Se presentará el presupuesto monetario de la elaboración del proyecto, como si hubiese sido realizado por una empresa profesional, y se discutirá el impacto socio-económico que este proyecto tiene en la sociedad.

1.5 Glosario

En esta sección se presentan una serie de términos técnicos mencionados con frecuencia en el presente documento, con el objetivo de facilitar la lectura de este último.

- **Página web:** Se trata de un documento que permite incluir diferentes elementos como texto, imágenes, video, sonidos, enlaces y que es presentado al usuario.
- **Sitio web:** Se compone de un conjunto de páginas web.
- **Dominio:** Se trata del nombre único que se asocia a un sitio web y lo identifica.
- **IP:** Se trata de un conjunto de números que identifica una interfaz de red de un sistema electrónico que utilice el protocolo de comunicación TCP/IP.
- **DNS (Domain Name System):** Se trata de un sistema que asocia un nombre de dominio de un sitio web, con una IP.
- **Cliente:** Aplicación informática que necesita conectarse a otra, llamada servidor y que está ubicada en otra máquina, para utilizar un servicio.
- **Servidor:** Aplicación informática que ofrece un servicio consumido por otras aplicaciones desde otras máquinas.
- **Navegador:** Software que permite visualizar páginas web, y accede a ellas a través de internet.
- **URL:** Mecanismo utilizado para ubicar los recursos publicados en Internet.
- **SQL:** Lenguaje utilizado por las bases de datos relacionales para el tratamiento de los datos.

- **NoSQL**: De esta manera se nombran a las bases de datos no relacionales y que por lo tanto no utilizan SQL como lenguaje de tratamiento de datos.
- **IoT (Internet of Things):** Se trata de la interconexión existente entre objetos físicos que disponen de software y que se comunican a través de Internet.
- **Kernel:** Núcleo de un sistema, que contiene la parte principal de un software, generalmente de un sistema operativo.
- **Plugin:** Se trata de aplicaciones que pueden acoplarse a otras ya existentes con el objetivo de ampliar funcionalidad.
- **Cloud:** Procede del término *cloud computing* y consiste en un conjunto de servidores que ofrecen entornos de ejecución en remoto.
- **Cluster:** Consiste en un conjunto de servidores que operan conjuntamente, ofreciendo la sensación de la existencia de un único servidor.
- **Stakeholder:** Cualquier persona potencialmente involucrada en un proyecto, ya sean proveedores, clientes, usuarios finales, trabajadores etc.
- **IDE:** Se trata de un entorno de desarrollo.
- **Base 64:** Método de codificación que permite representar una cadena de caracteres en código ASCII.

2. ESTADO DEL ARTE

En esta sección se van a exponer diferentes tecnologías que se utilizan a nivel profesional en el desarrollo de aplicaciones web y se analizarán distintas alternativas valorando sus ventajas e inconvenientes. A continuación, se definirá con más detalle el alcance del proyecto y se analizarán distintas soluciones existentes en el mercado.

2.1 Paradigma cliente-servidor

Cuando un usuario quiere acceder a un sitio web, lo realiza por medio de un **cliente**, generalmente un navegador como Google Chrome o Mozilla Firefox, que se encarga de presentar la interfaz de usuario de la web después de haber solicitado la información correspondiente al servidor. Un **servidor** es otro ordenador o un conjunto de ellos que es capaz de soportar un gran número de solicitudes de información por parte de los navegadores. Una de las razones principales del éxito de este paradigma es la gran capacidad de almacenamiento que tienen los servidores. Al navegar por diferentes sitios web, el navegador está constantemente recibiendo información de la parte servidora y de no ser así, la parte cliente debería contar con esa información para presentarla en la interfaz [2].

El proceso de carga de una página web sería el siguiente: El navegador realiza una petición mediante una *URL* y el *DNS (Domain Name System)* traduce esa *URL* en una IP asociada al servidor que tiene alojada la página web en cuestión. En este punto y según el tipo de sitio web tenemos 2 escenarios distintos:

- Sitios web estáticos, donde el código de la página web se envía al navegador para que este lo ejecute y muestre la interfaz al usuario.
- Sitios web dinámicos, donde antes de enviar la página al navegador, se realizan una serie de operaciones en una base de datos como consultas, inserciones y borrados, con el fin de enviar el contenido de la página web tras alguna modificación. Estas operaciones se realizan por lo tanto desde la parte servidora.

Esta estructura sirve para separar la lógica de la aplicación, situada en la parte servidora, de su presentación al usuario, situada en la parte cliente..

2.3 Tecnologías backend

PHP

PHP nace en 1995 como un simple intérprete utilizado para procesar formularios y que vio la luz bajo el nombre "Personal Home Page". Tras la liberación del código fuente por su autor Rasmus Lerdof, y la colaboración de programadores a lo largo del mundo, PHP 3 aparece como la primera versión de este lenguaje con cierta similitud a la versión actual y actualizando su nombre a *Hypertext PreProcessor* [3].

Con la versión PHP 4 se incrementó el rendimiento del lenguaje, pero no fue hasta PHP 5 cuando este lenguaje consiguió equipararse a otros lenguajes backend punteros como Java. La versión actual es PHP 7 y está disponible en Windows, Linux y Mac.

PHP permite la interacción con bases de datos y editar archivos del servidor con el fin de construir sitios web dinámicos. Es un lenguaje libre y orientado a objetos con la posibilidad de aplicar ciertos patrones de diseño y arquitectura.

Existe además de una amplia documentación online tanto oficial como procedente de programadores independientes debido a que se trata de un lenguaje con cierta antigüedad.

Java EE

Java Enterprise Edition se compone de un conjunto de especificaciones que las aplicaciones desarrolladas con el lenguaje de programación Java deben seguir para ser consideradas como aplicaciones Java EE.

La arquitectura Java EE se basa en 3 conceptos: [4].

- **Servicios**. Albergan la funcionalidad de la aplicación y son proporcionados por un contendor.
- **Contenedores**. Son entornos donde se ejecutan las aplicaciones. Varios contenedores forman un servidor de aplicaciones.
- **Componentes.** Hacen uso de los servicios para construir una funcionalidad determinada que se ejecuta en un contenedor u otro en función del tipo de componente.

Java EE se compone de varias tecnologías como: [5].

- **Java Server Pages (JSP).** Se trata de una página HTML con código Java incrustado que se ejecuta del lado del servidor. Existen etiquetas que determinan el inicio y el fin de sentencias Java y son denominadas *Scriptlets* [6]. Un ejemplo de utilidad de esta

tecnología es su uso en la página del "carrito" de una tienda online, donde el servidor devuelve la página tras realizar la suma de los precios de los productos y mostrar ese resultado en un campo de la estructura HTML. Esa suma de los precios se realizaría dentro de un scriplet y sería ejecutado por el servidor antes de enviar la información al navegador. Un scriplet consta de una etiqueta de apertura y otra de cierre entre las que se incluye código Java.

- **Java Message Service (JMS).** Se trata de un API que permite la comunicación entre aplicaciones Java de manera que dos componentes puedan estar conectados mediante el envío de mensajes. Una aplicación JMS consta de clientes que envían o reciben mensajes, los propios mensajes y los denominados objetos administrados, hacia los que se dirigen las comunicaciones.
- **Java Persistence API (JPA).** Ofrece un modelo de persistencia basado en el mapeo de las "tablas" de las bases de datos relacionales en POJO's (Plain Old Java Object), una clase Java que representa la semántica recogida en una tabla [7].

Python

Python es un lenguaje de programación creado a comienzos de la década de los 90 por Guido van Rossum y caracterizado por ser un lenguaje libre con un elevado número de librerías que facilitan la tarea de los programadores. Además, es un lenguaje que requiere menos líneas de código que otros para realizar la misma funcionalidad, lo cual es un gran atractivo para los desarrolladores [8]. Cabe destacar el gran crecimiento que Python está teniendo desde principios de la década pasada hasta llegar a situarse al mismo nivel de usuarios que lenguajes históricamente dominantes como Java o JavaScript.

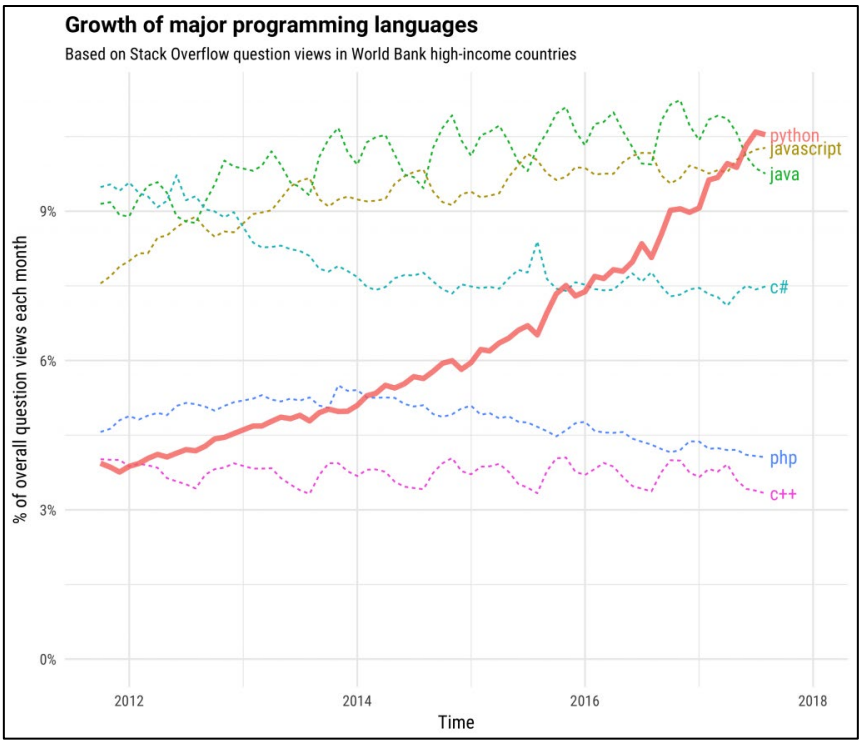


Fig. 2.1 Evolución del uso de los lenguajes de programación [9]

2.4 Tecnologías frontend

Son utilizadas para realizar la interfaz con la que el usuario interactúa y establecer comunicaciones con el servidor.

HTML

HTML O *HyperText Markup Language es* un lenguaje de programación que define la estructura de un sitio web y está basado en etiquetas con diferentes significados como títulos, párrafos o imágenes. *HyperText* o hipertexto hace referencia a la vinculación entre diferentes páginas web, es decir, a los enlaces existentes entre ellas [10].

Actualmente se trabaja con la versión HTML5, pero la gran importancia de internet y los sitios web provoca que esté en constante actualización.

CSS

CSS o *Cascading Style Sheets* se trata de un lenguaje que permite personalizar el diseño del sitio web, proporcionando un estilo determinado a alguna etiqueta HTML o a un conjunto de ellas. Sin estas hojas de estilo la página web puede seguir siendo visitada, pero su aspecto será muy básico y nada elaborado.

JavaScript

Es un lenguaje que permite dotar a los sitios web de mayor dinamismo e interactividad. Esto se consigue mediante animaciones, cambios de imágenes cada ciertos segundos o control de errores en formularios. Se trata de una de las principales tecnologías web en la actualidad.

Cualquier sitio web profesional combina estos 3 lenguajes, pero existen frameworks que facilitan el desarrollo de esta parte frontend y que están muy extendidos a nivel profesional. Destaca en este apartado, Angular, compatible con Mozilla Firefox, Google Chrome, Microsft Edge, Andorid e iOS. Por otro lado, React es una librearía de código abierto que facilita la programación de esta parte cliente [11].

2.5 Bases de datos

Otro aspecto fundamental en el desarrollo de aplicaciones es la base de datos. A continuación, se va a hacer un análisis de las diferentes opciones de bases de datos más extendidas a nivel profesional.

2.5.1 Bases de datos relacionales

Los principios de las bases de datos relacionales nacieron en IBM en 1970 de la mano de Edgar Frank Codd como respuesta a un paradigma poco eficiente de las bases de datos. Cada sistema disponía de su propia base de datos, pero no había un estándar sobre el diseño de la estructura, lo que dificultaba el mantenimiento de las aplicaciones al tratarse de un conocimiento muy específico. Las bases de datos relacionales supusieron un nuevo paradigma en este ámbito al proporcionar un estándar de almacenamiento y acceso a la información eficiente. [12].

Están basadas en tablas, que representan entidades. Estas tablas están formadas por columnas que definen los atributos de la entidad que se quieren almacenar. De esta manera, cada fila de una tabla se corresponde con un registro con una clave identificadora única de manera que no pueden existir dos filas iguales.

Una gran virtud de este modelo es el uso de SQL (Structured Query Language), un lenguaje de consulta propio de las bases de datos relacionales basado en el álgebra relacional y que resulta intuitivo y eficiente [12]. Este lenguaje ofrece mecanismos de integridad referencial de manera que si se elimina un registro (una fila de una tabla), se eliminan todos los registros dependientes de este, situados en otras tablas y que carecen de sentido si ese primer registro se elimina. Como

inconvenientes, cabe destacar que su uso no está optimizado para datos multimedia y que, ante consultas complejas, el rendimiento disminuye notablemente.

Las bases de datos relacionales ofrecen las propiedades conocidas como ACID [13].

- La atomicidad hace referencia a la necesidad de que una transacción se ejecute por completo o no se ejecute.
- La coherencia supone que los datos se reflejan en la base de datos de acuerdo a su esquema, una vez realizada la transacción.
- El aislamiento supone que las transacciones se realicen por separado.
- La durabilidad hace referencia a la persistencia de los datos ante cualquier situación, ya sea un corte eléctrico o un error inesperado.

Dentro de los sistemas gestores de bases de datos relacionales existen diversas opciones como las que se exponen a continuación.

MySQL

MySQL es un sistema gestor de bases de datos de código abierto desarrollado por MySQL AB, la cual fue comprada por Sun MicroSystems en 2008 y esta a su vez por Oracle Corporation en 2010. Algunas de las ventajas de MySQL son la existencia de desecadenantes para automatizar labores, o la maximización del número de transacciones por segundo, siendo una transacción, un conjunto de operaciones que se ejecutan en conjunto o no se ejecutan, con el objetivo de garantizar la integridad de la base de datos.[14]

Oracle Database

Se trata del sistema gestor desarrollado por Oracle y el históricamente más utilizado a nivel profesional. Cuenta con 7 ediciones de las cuales solo una, la Express Edition es gratuita [15]. Destaca la existencia de SQL Plus como gestor basado en línea de comandos o el framework SQL Developer con una interfaz amigable.

SQL Server

El sistema gestor de bases de datos desarrollado por Microsoft cuenta también con una interfaz por línea de comandos y otra interfaz gráfica y desde 2016 también puede ser utilizado en sistemas GNU/Linux. Como desventajas cabe destacar que soporta menos lenguajes de programación que sus competidores [16].

2.5.2 Bases de datos no relacionales

Las bases de datos no relacionales surgieron por la necesidad de gestionar datos no estructurados o semi-estructurados y difieren de las bases de datos relacionales, tanto en la forma de almacenar los datos como en la manera a de acceder a los mismos. Este tipo de bases de datos no utilizan SQL como medio de interacción con los datos sino que existen diversos lenguajes para ello, si bien no existe ninguno que se considere como estándar. Por otro lado, estas bases de datos no cumplen con las propiedades ACID debido a que relajan estas restricciones para conseguir una mayor escalabilidad, lo que hace que sean una opción interesante para aplicaciones de alto rendimiento.

Dentro de las bases de datos no relacionales existen diversos tipos entre los que destacan los siguientes: [17]

- **Clave-valor.** Son muy utilizadas en el ámbito de IoT o de los videojuegos online debido a su baja latencia (escaso tiempo de respuesta entre la acción física del usuario y la respuesta del sistema). En ellas cada clave se corresponde con un identificador único, a la que se le asigna un valor.
- **Documentos.** Están basadas en documentos JSON que son por naturaleza flexibles ya que no todos los JSON deben ser iguales, pero si disponen de cierta estructura debido a que se trata de un formato de texto estandarizado. Por analogía, cada documento se corresponde con un registro de las bases de datos relaciones, si bien estos registros tienen un tamaño fijo.
- **Gráficos.** Son utilizadas para conjuntos de datos relacionados entre sí, con aplicaciones en redes sociales o sistemas de recomendación.
- **En memoria.** Ofrecen un tiempo de respuesta muy bajo, lo que permite su aplicación en tablas de apuestas o análisis en tiempo real.

A modo de resumen, se podría señalar que las principales ventajas de las bases de datos no relacionales frente a las relacionales radican en su flexibilidad, su escalabilidad y su alto rendimiento. Las desventajas se resumen en que no existe un lenguaje estandarizado de consultas, no cumplen con las propiedades ACID, y existe menos documentación debido a que son relativamente nuevas [18].

Como gestores de este tipo de bases de datos encontramos:

MongoDB

Se trata de una base de datos basada en documentos y distribuida, es decir, está formada por una serie de bases de datos situadas en distintos puntos que se conectan mediante un sistema de comunicaciones.

Como ventajas cabe destacar que se trata de una herramienta de código abierto, de manera que el precio que se paga por la licencia es solo por el soporte ofrecido, es una base de datos NoSQL muy popular por lo que la documentación es extensa y permite realizar consultas complejas con buen rendimiento. Además, permite tener clusters distribuidos por todo el mundo de manera que si la aplicación tiene usuarios de varias partes del mundo se reduce el problema de alta latencia entre el servicio y la base de datos [19].

Couchbase

Se trata de otra base de datos documental y distribuida cuya primera versión vió la luz en 2010. Es una herramienta similar a MongoDB, si bien utiliza un sistema de cachés que permite una escalabilidad aún mayor además de proporcionar una configuración más simple y centralizada en su *Admin Console*. Como punto negativo cabe destaca que son menos los lenguajes de programación que permiten la interacción con Couchbase [20].

2.6 Arquitectura del sistema

Otro aspecto importante que se debe tener en cuenta en el diseño de una aplicación orientada a e-commerce es cómo se va a estructurar el software, por lo que a continuación se van a exponer diferentes opciones valorando sus ventajas e inconvenientes.

Arquitectura monolítica

Se trata de una arquitectura poco novedosa, donde todos los componentes de la aplicación están conectados entre ellos y desarrollados de tal forma que todos deben ser ejecutados simultáneamente en el mismo servidor. Esta arquitectura simplifica el despliegue de la aplicación en el servidor puesto que solo se necesita realizar una vez y reduce la complejidad técnica [21], [22].

Por otro lado, los principales problemas radican en la dificultad de escalabilidad, debido a que se debe de dotar de más recursos a toda la aplicación y no solo a una parte, como en otras

arquitecturas. Otro problema relevante es el difícil mantenimiento debido a la dificultad para localizar errores a medida que se va añadiendo código al mismo módulo constantemente [21].

Arquitectura de micro-kernel

Este patrón se basa en agrupar una serie de tareas críticas que se realizan de manera recurrente en el denominado *kernel* de la aplicación, mientras que la lógica específica de la aplicación se implementa en los denominados *plugins*. El *kernel* es el encargado de tareas como la carga de plugins o la comunicación entre ellos cuando existe dependencia.

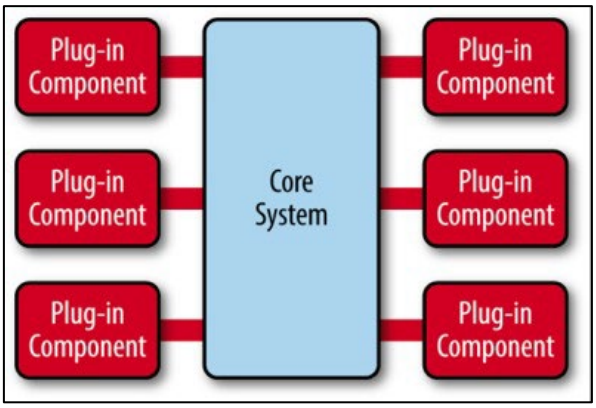


Fig. 2.2 Arquitectura microkernel [23]

La principal ventaja de este patrón reside en la escalabilidad del sistema, puesto que para expandir o mejorar un aspecto como el registro de usuarios únicamente haría falta implementar lógica en el plugin que contenga esta funcionalidad, siendo este proceso transparente a otras partes de la aplicación.

El principal problema radica en la difícil decisión de qué partes de la aplicación incluir en el kernel debido a la complejidad que supone realizar cambios en él. Como consecuencia, este patrón suele estar más orientado a aplicaciones de escritorio que a aplicaciones web [24].

Arquitectura de micro-servicios

Se trata de una arquitectura nacida a partir del concepto de microservicio, definido por primera vez por Martin Fowler en 2014 como una unidad independiente que realiza una tarea determinada y se comunica con otros microservicios mediante mecanismos ligeros, generalmente a través de una API y el protocolo HTTP.

Esta arquitectura puede parecerse a la anterior, pero tiene una clara orientación a la computación en la nube o *cloud,* donde cada uno de los microservicios que componen una aplicación puede estar contenido en diferentes cluster, lo que permite dotar a un microservicio de mayores recursos si la situación lo requiere [24]. Un ejemplo puede ser el caso del *black friday*

en una tienda de e-commerce, de manera que se puedan dotar de más recursos al microservicio encargado del proceso de compra.

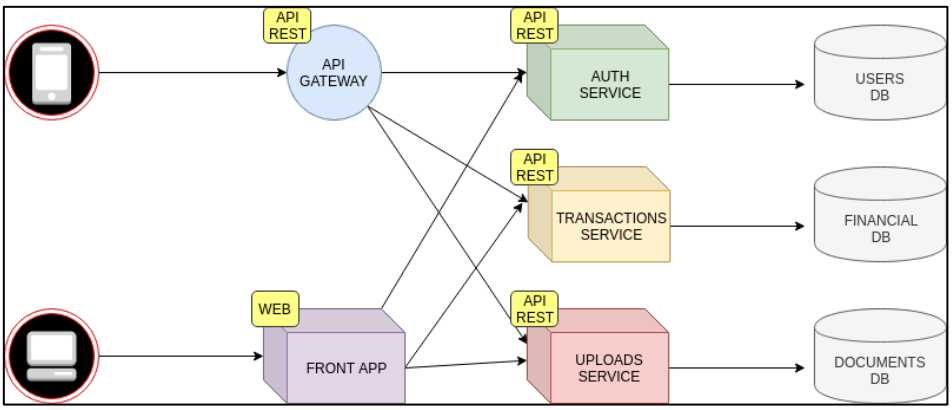


Fig. 2.3 Arquitectura basada en microservicios [24]

En esta imagen se puede observar cómo en función de la acción que se esté realizando en el sistema, se realizan llamadas al microservicio encargado de esa tarea mediante la API del microservicio y este accede únicamente a la parte de la base de datos del sistema relacionada con su funcionalidad.

Como principales ventajas de esta arquitectura cabe destacar:

- Es posible escalar cada microservicio de manera individual si la situación lo requiere, tal y como se ha comentado en el ejemplo del *black Friday*. Además, en caso de tener que añadir nuevas funcionalidades, se simplifica el proceso de identificación de los componentes afectados.
- Mantenimiento más sencillo y aislamiento de errores. Si por algún motivo el servidor que aloja un microservicio deja de funcionar, el resto de la aplicación puede seguir funcionando con normalidad. [25]

2.7 Metodologías de trabajo

Una vez estudiadas las herramientas tecnológicas existentes, es conveniente estudiar las diferentes metodologías de trabajo existentes. La existencia de un procedimiento para actuar minimiza los errores y maximiza la eficacia y eficiencia de un equipo, por lo que conocer las principales metodologías que se aplican a nivel profesional es un factor que todas las empresas de desarrollo software buscan en un empleado.

A continuación, se van a exponer el funcionamiento de las metodologías tradicionales y las nuevas metodologías ágiles, y se detallarán las ventajas y desventajas de cada una en relación a su ajuste al presente proyecto.

Metodologías tradicionales

Como su propio nombre indica se trata de las metodologías que se han utilizado en el mundo del software durante décadas y que se caracterizan por ser poco flexibles y en consecuencia, responder de manera ineficiente a los cambios.

Desarrollo en cascada o Waterfall. Se trata de la principal metodología a destacar en este grupo y determina una serie de actividades a realizar de manera secuencial ya que el resultado de una tarea es necesaria para la siguiente [26]. Las fases de las que consta son las siguientes:

- **Análisis de requerimientos.** En esta fase inicial del ciclo de desarrollo, se realiza la toma de requisitos del proyecto con el cliente, determinando qué es exactamente lo que tiene que hacer el sistema sin entrar en detalles de implementación. Es importante destacar que esta toma de requisitos se realiza únicamente en esta fase. A continuación, se realiza un análisis de viabilidad del proyecto para determinar la rentabilidad del mismo y la decisión de llevarlo a cabo o no.
- **Diseño del sistema.** En esta fase se formula una solución específica, detallando la arquitectura del sistema, la tecnología a utilizar y aspectos técnicos en función a la tecnología escogida como librerías o frameworks.
- **Implementación.** Se trata de la fase de desarrollo del sistema, cubriendo todas las funcionalidades que según los requisitos debe tener.
- **Verificación.** Se trata de comprobar que el sistema, una vez desarrollado, cumple con los requerimientos del cliente y que en su conjunto funciona correctamente. Cabe destacar que este término de *verificación* es diferente de la *validación*, la cual consiste en comprobar con el cliente que lo que se va a hacer es lo que se ha pedido.
- **Mantenimiento.** En esta fase, una vez el software es entregado al cliente, se realizan cambios o mejoras en el sistema debido a fallos o posibles ampliaciones. Según Borja Santos, responsable de la compañía *Stripe* de España y Portugal, los desarrolladores dedican la mitad de su tiempo al mantenimiento del software.

Existen otras metodologías tradicionales con mejor respuesta ante los cambios como las siguientes.

Prototipado. Se basa en la construcción rápida del sistema de manera que los usuarios finales puedan utilizarlo y aportar *feedback* con el objetivo de adaptar el producto al cliente.

Incremental. En este caso, se construye el sistema con algunas de las funcionalidades requeridas y se entrega el producto para que empiece a ser utilizado. En sucesivas entregas posteriores se van añadiendo el resto de funcionalidades pedidas [25].

Metodologías ágiles

Estas metodologías se consideran adaptativas, es decir, se ajustan bien a los cambios que surgen a lo largo de las fases de un proyecto y se benefician de ese cambio para crear una solución final más completa. En ellas se trabaja con la documentación mínima necesaria ya que se centra en el desarrollo del proyecto.

El origen de estas metodologías se remonta a 1994, cuando *The Standish Group* publicó "The Chaos Report", un informe sobre diversos proyectos software donde se señalaba que sólo el 16% de estos proyectos eran considerados exitosos. Se señalaba, además, que el hecho de que la calificación del 84% restante fuera de "desafiante" o "cancelado" era consecuencia del cambio de los requerimientos en diferentes etapas del proyeto y de la falta de involucramiento de los diversos *stakeholders* [27]. A partir de entonces surgieron varias metodologías que se recogieron en el "Manifiesto Ágil" y entre las que destacan las siguientes.

SCRUM

Esta metodología promueve el desarrollo incremental del producto, dando mayor prioridad a lo que tiene más valor para el cliente y está basada en iteraciones o *sprints*. Un *sprint* es un ciclo de desarrollo de entre las 2 y 4 semanas en las que el equipo de desarrollo trabaja para proporcionar un incremento en el producto final que es entregado al final de ese periodo. Los *sprint* dentro de un proyecto son de duración fija y suponen un límite para entregar una funcionalidad determinada [28].

En esta metodología destaca el *Product Owner* o cliente, que se encarga de definir lo que espera del producto y ayuda en la creación de las "historias de usuario", una forma de representar los requisitos de manera estandarizada. Por otro lado, se encuentra el *Scrum Master* que traslada los objetivos del *sprint* al equipo de desarrollo y coordina el desarrollo del proyecto. Son habituales las reuniones diarias del *Scrum Master* con el equipo de desarrollo para valorar el estado del desarrollo y alinear los objetivos [29].

Lean

Esta metodología nació como una estrategia de fabricación en la compañía japonesa *Toyota* y actualmente se trata de una de las principales metodologías ágiles fundamentada en 7 principios: [30]

- Eliminar desperdicios como burocracia o falta de comunicación.
- Amplificar el aprendizaje mediante reuniones periódicas con el cliente o la integración continua de funcionalidades verificadas.
- Decidir lo más tarde posible para reducir errores si el enfoque del proyecto no es claro.
- Entregar lo más rápido posible una versión del producto final para poder recibir feedback y realizar ajustes.
- Capacitar al equipo de manera que se les permita expresar sus opiniones acerca del proyecto, así como hacerles sentir apoyados en momentos críticos.
- Construir con integridad de manera que el cliente perciba el producto final como el resultado que esperaba.
- Ver el sistema en su conjunto, teniendo en cuenta no solo los componentes sino también las interacciones entre ellos, adelantándose así a posibles errores.

3. PLANTEAMIENTO DEL PROBLEMA

3.1 Descripción general

3.1.1 Necesidad existente en el mercado

Es un hecho que la pandemia del Covid-19 ha cambiado muchos hábitos, y es que a raíz de las restricciones a las que la sociedad se ha visto sometida, algunas actividades han entrado en auge. El deporte se ha convertido en una vía de escape al estrés y una forma de ocio que se ha extendido aún más durante estos últimos dos años. Esto ha provocado que la compra de productos deportivos se haya disparado, hasta el punto de incrementarse un 79% durante el año 2020 [31]. Pasado un tiempo después de la compra de estos productos, hay algunos que ya no encajan con los propietarios, bien porque no se ajustan al nivel requerido por el deportista o simplemente porque se desea un cambio o renovación en ese material. En este punto es habitual el uso de aplicaciones de venta de productos de segunda mano, donde cada usuario puede subir sus productos con el objetivo de ser vendidos.

A continuación, se va a realizar un análisis sobre las distintas aplicaciones existentes en el mercado de venta de ropa de segunda mano, con el objetivo de analizar como de necesaria es la aplicación que se expone en este proyecto.

3.1.2 Soluciones existentes en el mercado

Wallapop

Se trata de una empresa española fundada en 2013 que ofrece una plataforma de compraventa de productos de segunda mano que funciona como un mercado en el que los compradores pueden chatear con los vendedores sobre el estado del producto, el precio y otros detalles. Esta aplicación utiliza la tecnología GPS disponible en los dispositivos móviles con el objetivo de ofrecer productos a los usuarios en un radio cercano.

Navegando por la web se puede observar cómo ofrece búsquedas en base a categorías o búsquedas simples a través de un buscador. Cabe destacar que para realizar funciones como subir un producto o chatear con los vendedores es necesario iniciar sesión, lo cual garantiza un mínimo de seguridad en el proceso de compra.

Fig. 3.1 Portal web de Wallapop

Las ventajas principales de Wallapop son la gran variedad de categorías de producto disponibles, unas funcionalidades e interfaz intuitivas y su gran poder de mercado. La plataforma cuenta tanto con versión web como con app móvil para Android e iOS.

El principal inconveniente, teniendo en cuenta el enfoque de este proyecto, es que no se centra en productos deportivos, sino que ofrece una gran variedad de productos. El hecho de no estar especializada puede influir en la decisión de los usuarios acerca de la aplicación en la que comprar productos deportivos.

Vinted

Vinted nació en Lituania en el año 2008 ofreciendo un sitio web de venta de ropa de segunda mano orientado al público femenino. Su expansión ha sido notablemente rápida, alcanzando Estados Unidos en 2010, publicando su versión app para móviles en 2012 e incluso comprando en 2019, a la competidora española por aquel entonces, Chicfy, por unos 10 millones de euros.

Vinted, al ofrecer únicamente productos de vestimenta, permite una búsqueda por categorías de hombre, mujer, niño y hogar, además de un buscador por palabras clave. Permite también el chat entre usuarios y ofrece una sección de novedades en su página principal.

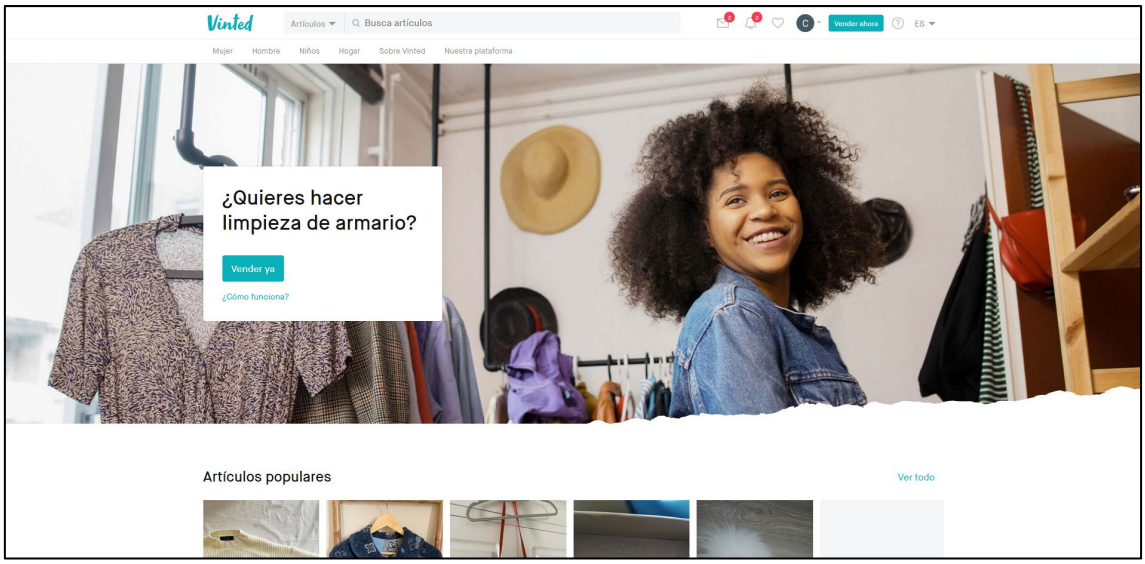


Fig. 3.2 Portal web de Vinted

A diferencia de Wallapop, el vendedor no tiene que acordar un punto de encuentro con el comprador, sino que simplemente tiene que llevar el artículo al punto de envío.

Como ventajas destaca la especialización en prendas de ropa, y una comodidad extra para ambas partes de la compraventa debido a que el vendedor lleva el producto al punto de envío y el comprador lo recibe donde él elija.

Como inconvenientes, al igual que en Wallapop, no existe una especialización en productos deportivos.

Amazon

Fundada en 1994 por Jeff Bezos, es una de las compañías más grandes a nivel mundial dedicada al comercio electrónico y a la computación en la nube. Se trata de una tienda online donde los usuarios solo pueden comprar los productos ofertados, ya que es el propio Amazon quien pone los productos a la venta. Ofrece tal variedad de productos que no ofrece búsqueda por categorías mas allá de “Lo mas vendido” o “Últimas novedades”, si bien si que despone de un buscador clásico por palabras clave.

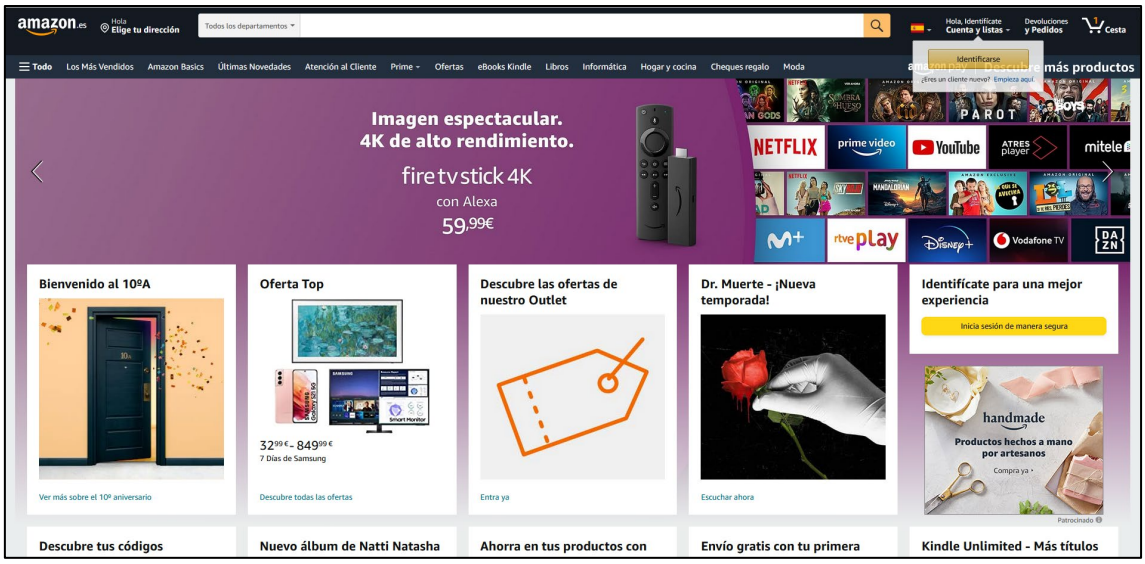


Fig. 3.2 Portal web de Amazon

Como novedad frente a los ejemplos anteriores, proporciona un "carrito" donde los usuarios pueden ir añadiendo productos con el fin de finalizar la compra de todos los productos simultáneamente. Para finalizar la compra también es necesario iniciar sesión en el portal. Dispone además de varias imágenes por cada producto y permite dejar comentarios valorando el producto.

Como ventajas destaca la existencia de una muy amplia variedad de productos a estrenar. Esto unido a la gran experiencia de usuario que se tiene al interactuar con la web provoca que muchos clientes confíen en Amazon como primer recurso para hacer sus compras.

Como inconvenientes, destacar que no se trata de un portal de compraventa de productos de segunda mano y no está especializado en productos deportivos.

3.1.3 Alcance del proyecto

Una vez detectada la necesidad existente en el mercado y tras haber realizado un análisis sobre las distintas alternativas que se pueden encontrar, se va a detallar la funcionalidad del sistema.

El sistema a desarrollar recibe el nombre de **Sportunity** y se trata de un sitio web de compra y venta de productos deportivos de segunda mano. Si bien el sistema estará disponible para cualquier usuario, se permitirá el registro de usuarios y su inicio de sesión para el acceso a la totalidad de las funcionalidades del sistema. En función de si se ha iniciado sesión en el sistema, el usuario podrá realizar diversas acciones:

- **No ha iniciado sesión.** En este caso se podrán realizar búsquedas de productos a través de palabras clave y se podrá acceder al detalle del producto.
- **Si ha iniciado sesión.** Además de las funcionalidades del caso anterior, el usuario podrá realizar búsquedas filtrando por distintos campos y poner productos a la venta completando un pequeño formulario con los detalles del producto. Los usuarios podrán modificar tanto sus datos de registro como los de sus productos, añadir productos a un "carrito" y finalizar el proceso de compra.

Por otro lado, cabe destacar lo que el sistema no permite:

- El sistema no permite chat entre usuarios.
- No permite el pago mediante otro método que no sea una tarjeta bancaria.
- No se desarrollarán versiones para móviles.

3.1.4 Restricciones generales

Para el funcionamiento del sistema es necesario un servidor de aplicaciones que permita la ejecución de la lógica de la aplicación y el acceso a los datos que deben persistir en una base de datos.

Además, es necesario un cliente para que muestre la interfaz del sistema y permita al usuario interactuar con él. Dado que el presente proyecto es una prueba de concepto y el sitio web no va a ser accesible desde internet se utilizará el dominio localhost de un ordenador personal de manera que la aplicación se ejecute en este ordenador en su totalidad. Las características del ordenador son las siguientes:

- Ordenador ASUS VivoBook S14.
    - Procesador Intel Core i5-8250U de octava generación.
    - 8GB de memoria RAM.
    - 256GB de disco duro SSD.
    - Sistema operativo Windows 10.

Dado que el volumen de operaciones que se van a realizar en el sistema es muy reducido al tratarse de una prueba de concepto y ser el autor del presente documento el único usuario que interactúe con el sistema, se considera aceptable este escenario.

El tiempo de desarrollo de este proyecto es entorno a 5 meses con una dedicación diaria entre 1 y 3 horas debido a su compatibilidad con otros proyectos de igual envergadura, lo que supone un incremento de la dificultad del proyecto.

3.1.5 Características de los usuarios

En este apartado se van a exponer lo que los usuarios podrán hacer dentro sistema junto con sus características, como nivel educativo y frecuencia estimada de uso de la aplicación, con el objetivo de adecuar las funcionalidades al tipo de usuario que el sistema va a tener. Dado que cualquier usuario puede ser a la vez comprador y vendedor, únicamente existe un perfil que detallar. El análisis aparece en la siguiente tabla.

Capacidades	**Características**
✓ Registrarse en la aplicación. ✓ Iniciar sesión. ✓ Modificar datos de registro. ✓ Subir productos. ✓ Modificar productos. ✓ Realizar búsquedas. ✓ Añadir productos al carrito. ✓ Comprar productos. ✓ Revisar sus compras y ventas.	✓ Conocimiento tecnológico nivel usuario. ✓ Frecuencia de uso de la aplicación ocasional, en torno a 1 vez semanal. ✓ Edades comprendidas entre los 15 y los 65 años.

Tabla 3.1: Capacidades y características de los usuarios

Sportunity está dirigido a personas que practiquen deporte de manera ocasional o regular, lo cual crea un *target* muy amplio de usuarios, aunque existen diferentes *stakeholders* dentro de este proyecto:

- El autor y desarrollador de este proyecto.
- Usuarios potenciales:
    - Deportistas habituales que necesitan renovar su equipamiento con regularidad y recurren a la aplicación para ver productos ofertados.
    - Deportistas ocasionales que no deseen realizar una inversión en material original y prefieren adquirir material de segunda mano a un precio menor.
- Posibles inversores que consideren invertir en el proyecto financiando el desarrollo o una ampliación del mismo.

3.2 Análisis de requerimientos

El proceso de análisis, documentación y validación de las funcionalidades que un sistema debe proveer es conocido como ingeniería de requisitos, y constituye una fase clave en el desarrollo del proyecto, tal y como se refleja en los siguientes gráficos.

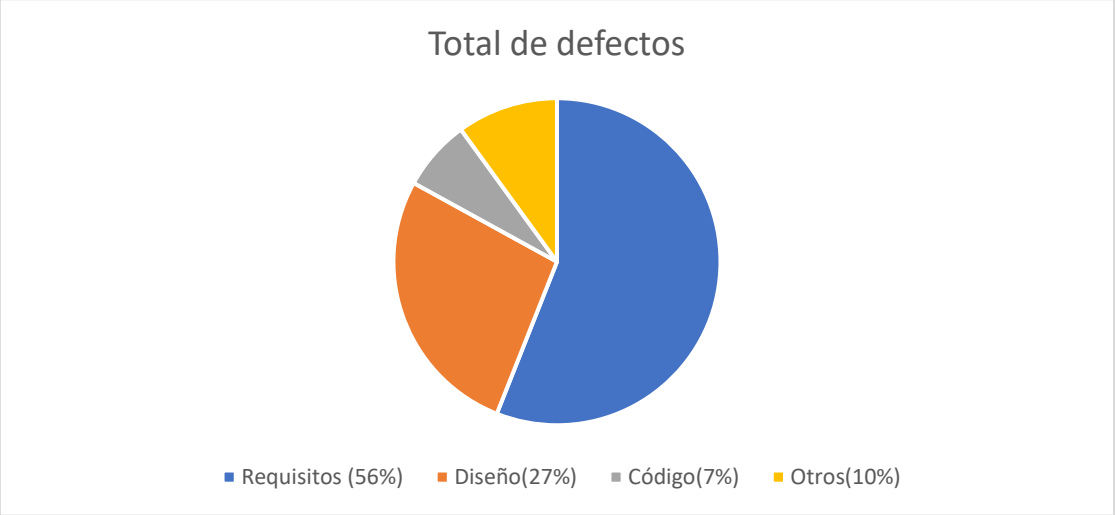


Fig. 3.3. Causas de los defectos del software [32]

Según un estudio realizado en 1984 por James Martin, y publicado bajo el nombre de "An Information Systems Manifesto“ el 56% de los defectos existentes en los productos finales provenían de los requisitos, es decir, de la definición de lo que el sistema debía hacer.

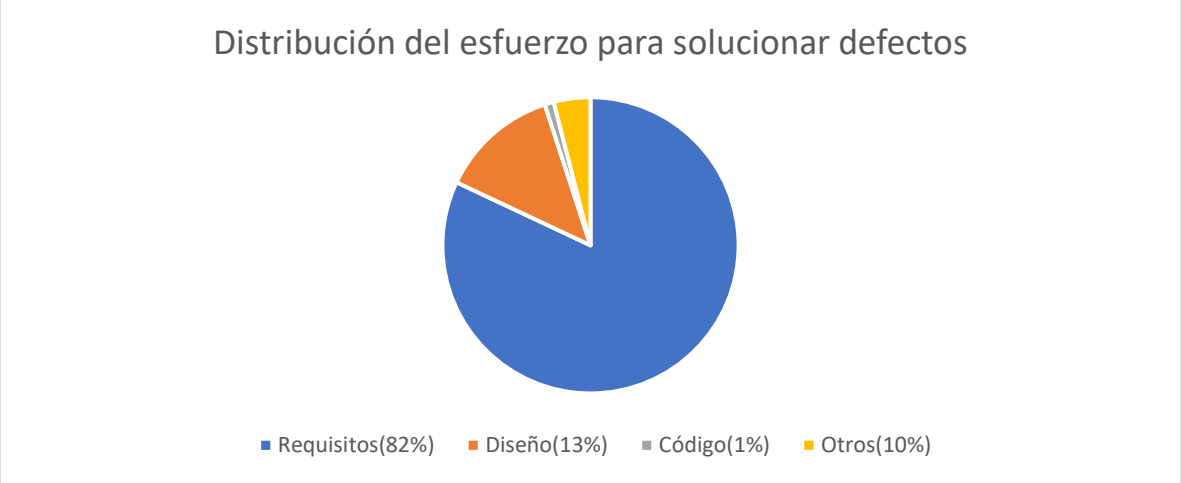


Fig. 3.4. Distribución del esfuerzo para solucionar defectos del software [32]

En ese mismo manifiesto se destaca que el 82% del tiempo dedicado a solucionar defectos, se emplea en errores surgidos debidos a una deficiente elicitación de requisitos. De esta manera se observa cómo además de ser la principal causa de defectos, estos errores son más difíciles de solventar que otros como los de diseño.

El Instituto de Ingenieros en Electricidad y Electrónica (IEEE), la organización técnica más prestigiosa del mundo, diseñó el estándar IEEE 830-1998 donde recoge ciertas directrices sobre cómo debe ser una buena especificación de requisitos, entre las que destacan las siguientes:

- Completa: Deben reflejarse todos los requerimientos.
- Consistente: Debe existir consistencia entre requisitos y otras partes del documento.
- Inequívoca: La redacción debe carecer de ambigüedades.
- Correcta: El software debe adecuarse a los requisitos.
- Trazable: Debe ser posible verificar su función en el sistema.
- Priorizable: Los requisitos deben poder ordenarse jerárquicamente.
- Los requisitos deben ser fácilmente modifcables.
- Deben redactarse en un lenguaje claro y fácilmente entendible.

La ingeniería de requisitos es un proceso, y como tal, requiere de unas fases que completar para que sea exitoso. En este ámbito se deben destacar dos tipos de requisitos, los **requisitos de usuario** y los **requisitos de sistema**. Los primeros se centran en definir en lenguaje natural pequeñas funciones que el sistema debe realizar, y deben ser entendidos por cualquier persona que los lea. Los del sistema tratan esas mismas funciones, pero a un nivel más bajo, es decir, ofrecen un detalle muy definido de qué pequeñas tareas debe realizar el sistema para completar esa función, además de definir cómo deben hacerse esas tareas.

3.2.1 Requisitos de usuario

En esta sección se van a definir una plantilla que se utilizará para recoger los requisitos de usuario de manera sencilla, clara y ordenada, y se expondrán los diferentes requisitos de este tipo.

Con el objetivo de conseguir una correcta elicitación de requisitos de acuerdo a las directrices del IEEE, es necesario definir una serie de atributos que deberán ser completados para cada requisito.

- **ID:** Se trata de un código que identifica de manera unívoca un requisito y se utilizará el siguiente patrón para su creación: RU-XX, donde RU hace referencia a los requisitos de usuario. XX hace referencia a la secuencia numérica de los requisitos, que empezarán por el código 01.
- **Nombre del requisito:** Breve nombre que sea descriptivo en cuanto a la necesidad reflejada en el requisito.

- **Estado:** Representa cómo se encuentra el requisito en relación a su validez, pudiendo tomar los valores de *propuesto*, *validado* o *cancelado*. Los requisitos son "validados" con el cliente con el objetivo de conocer si la especificación del requisito encaja con lo que se espera del sistema, si bien en el ámbito de este proyecto no existe la figura de un cliente y los requisitos serán validados por el autor del proyecto. Únicamente los requisitos validados pueden ser implementados.
- **Versión:** Detalla la fecha de la última modificación realizada en la descripción del requisito.
- **Prioridad:** Especifica cierto orden en el que los requisitos deben ser tenidos en cuenta para elaborar el sistema, pudiendo tomar los valores de *baja*, *media* o *alta*. Se trata de un elemento importante debido a que facilita la planificación del proyecto.
- **Relevancia:** Acepta los valores *esencial*, *deseable* u *opcional* y hace referencia a la importancia del requisito en cuestión.
- **Verificabilidad:** Se trata de un atributo binario con dos posibilidades, *Verificable* o *No verificable* y señala la posibilidad de probar que el sistema cumple con la especificación del requisito. El escenario ideal es la verificabilidad de todos los requisitos del sistema, para asegurar que el sistema cumple con su especificación.
- **Autor:** Nombre del autor del requisito.
- **Descripción:** Detalle de la función que debe cumplir el sistema

La plantilla sobre la que se recogerán los distintos requisitos y que contiene los atributos señalados, es la presentada a continuación.

<ID>	**<Nombre del requisito>**
Estado	Propuesto |Validado | Cancelado
Versión	Fecha última de modificación
Prioridad	Baja |Media | Alta
Relevancia	Esencial | Deseable |Opcional
Verificabilidad	Verificable |No verificable
Autor	Nombre del autor
Descripción	

Tabla 3.2. Plantilla de requisitos de usuario.

3.2.1.1 Relacionados con el registro de usuario

RU-01	**Registro de usuario**
Estado	Validado
Versión	15/10/2021
Prioridad	Alta
Relevancia	Esencial

Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario debe poder registrarse en el sistema.

Tabla 3.3. Requisitos de usuario 01.

RU-02	**Inicio de sesión del usuario**
Estado	Validado
Versión	15/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario debe poder iniciar sesión en el sistema.

Tabla 3.4. Requisitos de usuario 02.

RU-03	**Visualización de los datos del usuario**
Estado	Validado
Versión	15/10/2021
Prioridad	Media
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario, tras haber iniciado sesión, debe poder visualizar sus datos de registro.

Tabla 3.5. Requisitos de usuario 03.

RU-04	**Modificación datos del usuario**
Estado	Validado
Versión	15/10/2021
Prioridad	Baja
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario, tras haber iniciado sesión, debe poder modificar los datos con los que realizó el registro.

Tabla 3.6. Requisitos de usuario 04.

RU-05	**Cierre de sesión del usuario**
Estado	Validado
Versión	15/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario puede cerrar la sesión una vez la haya iniciado.

Tabla 3.7. Requisitos de usuario 05.

RU-06	Baja del usuario en el sistema
Estado	Validado
Versión	15/10/2021
Prioridad	Media
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario, tras haber iniciado sesión, debe poder darse de baja en el sistema.

Tabla 3.8. Requisitos de usuario 06.

3.2.1.2 Relacionados con los productos

RU-07	Puesta a la venta de un producto
Estado	Validado
Versión	15/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario, tras haber iniciado sesión, debe poder poner a la venta un producto.

Tabla 3.9. Requisitos de usuario 07.

RU-08	Consulta de los productos propios
Estado	Validado
Versión	15/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario, tras haber iniciado sesión, debe poder visualizar qué productos tiene a la venta.

Tabla 3.10. Requisitos de usuario 08.

RU-09	Modificación de los datos de un producto
Estado	Validado
Versión	15/10/2021
Prioridad	Media
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario, tras haber iniciado sesión, debe poder modificar los datos de los productos de su propiedad.

Tabla 3.11. Requisitos de usuario 09.

RU-10	Eliminar un producto

Estado	Validado
Versión	15/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario, tras haber iniciado sesión, debe poder eliminar un producto de la aplicación.

Tabla 3.12. Requisitos de usuario 10.

RU-11	**Búsqueda simple**
Estado	Validado
Versión	15/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario debe poder realizar una búsqueda por palabras clave en un buscador.

Tabla 3.13. Requisitos de usuario 11.

RU-12	**Búsqueda avanzada**
Estado	Validado
Versión	15/10/2021
Prioridad	Media
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario debe poder realizar una búsqueda mediante el filtro de varios campos en un formulario.

Tabla 14. Requisitos de usuario 12.

RU-13	**Detalle del producto**
Estado	Validado
Versión	15/10/2021
Prioridad	Baja
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario debe poder visualizar las características de un producto.

Tabla 3.15. Requisitos de usuario 13.

3.2.1.3 Relacionados con el carrito

RU-14	**Añadir productos al carrito**
Estado	Validado

Versión	15/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario debe poder añadir productos al carrito de la compra si ha iniciado sesión.

Tabla 16. Requisitos de usuario 14.

RU-15	**Consultar el carrito**
Estado	Validado
Versión	15/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario debe poder observar los productos existentes en el carrito si ha iniciado sesión.

Tabla 17. Requisitos de usuario 15.

RU-16	**Eliminar productos del carrito**
Estado	Validado
Versión	15/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario debe poder eliminar productos del carrito si ha iniciado sesión.

Tabla 3.18. Requisitos de usuario 16.

3.2.1.4 Relacionados con el proceso de compra

RU-17	**Finalizar el proceso de compra**
Estado	Validado
Versión	15/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario debe poder realizar el pago de los productos del carrito si ha iniciado sesión.

Tabla 3.19. Requisitos de usuario 17.

RU-18	**Envío de confirmaciones**
Estado	Validado
Versión	15/10/2021

Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	Una vez se haya realizado el pago se mostrará un mensaje de confirmación al comprador y se enviará un mensaje de aviso al vendedor indicando que su producto ha sido comprado.

Tabla 3.20. Requisitos de usuario 18.

3.2.1.5 Relacionados con el histórico de compra-venta

RU-19	**Visualización de las compras realizadas**
Estado	Validado
Versión	15/10/2021
Prioridad	Media
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario deberá poder acceder a un histórico de las compras realizadas, si ha iniciado sesión.

Tabla 3.21. Requisitos de usuario 19.

RU-20	**Visualización de las ventas realizadas**
Estado	Validado
Versión	15/10/2021
Prioridad	Media
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario deberá poder acceder a un histórico de las ventas realizadas, si ha iniciado sesión.

Tabla 3.22. Requisitos de usuario 20.

3.2.2 Requisitos del sistema

3.2.2.1 Casos de uso

Los casos de uso constituyen una representación visual a alto nivel sobre las acciones que el software debe llevar a cabo para realizar las funciones definidas en los requisitos de usuario, mostrando así la interacción entre los diferentes actores del sistema y el propio sistema. Existen 2 tipos de relaciones entre estas acciones. Por un lado, la relación *include* define la realización de una acción si la acción principal de la que depende es realizada. Estas acciones reciben el nombre de casos de uso. Por otro lado, la relación *extend* señala que, si un caso de uso tiene lugar, otros pueden, o no, tener lugar si están relacionados con el primero con este tipo de relación, definiendo así una implicación opcional.

La representación de estos casos de uso se realiza mediante un diagrama donde los casos de uso se representan con óvalos con un pequeño nombre descriptivo en su interior y los actores con un pequeño dibujo de una persona. En el caso de las relaciones *include* la flecha entre ambos casos de uso parte del caso de uso principal hacia los casos de uso dependientes, mientras que en el caso de las relaciones *extend*, la flecha parte del caso de uso dependiente hacia el principal.

A continuación, se presentan los casos de uso organizados en las mismas categorías que se describieron en los requisitos de usuario.

Relacionados con el registro de usuario

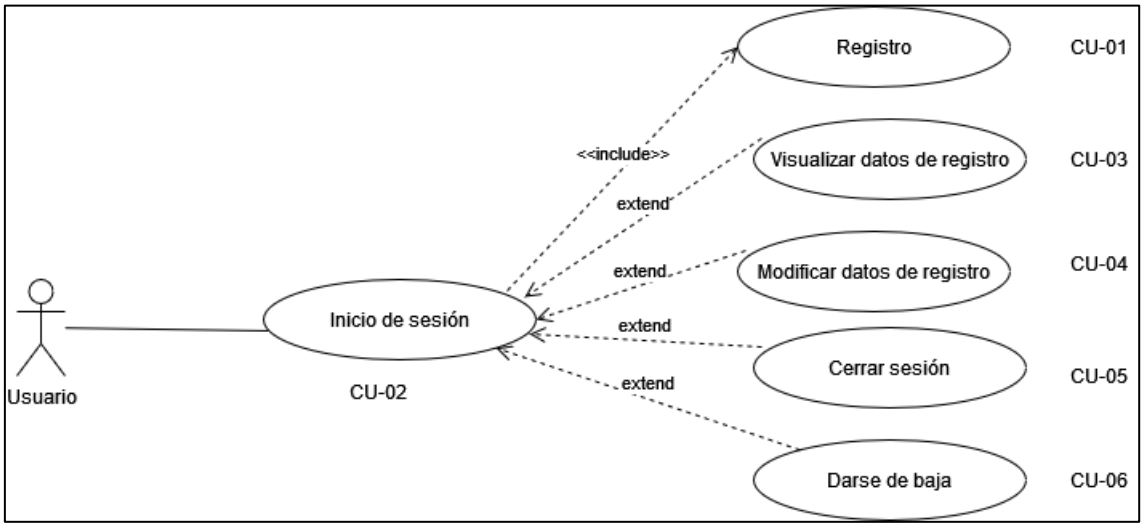


Fig. 3.5. Casos de uso relacionados con el registro de usuario.

Relacionados con los productos

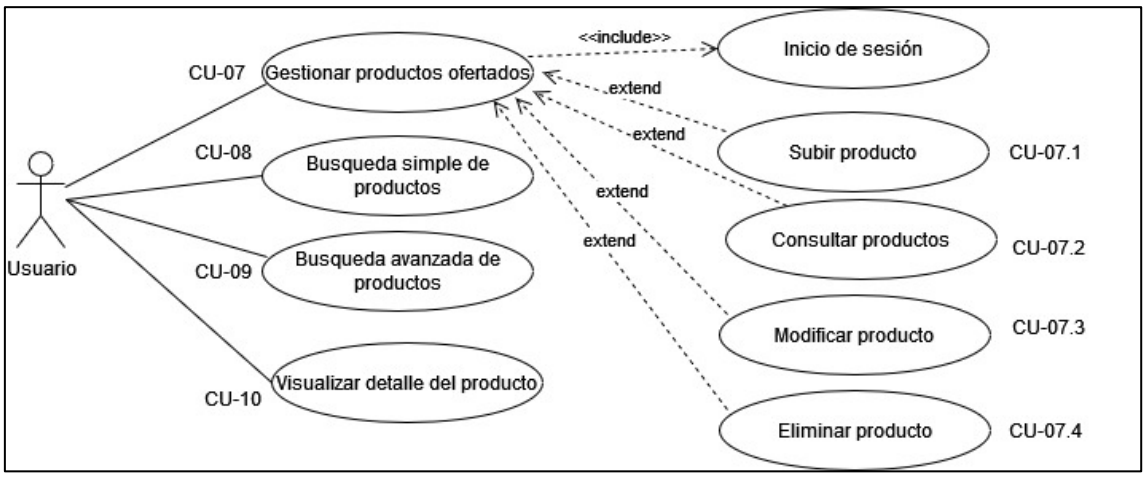


Fig. 3.6. Casos de uso relacionados con los productos.

Relacionados con el carrito

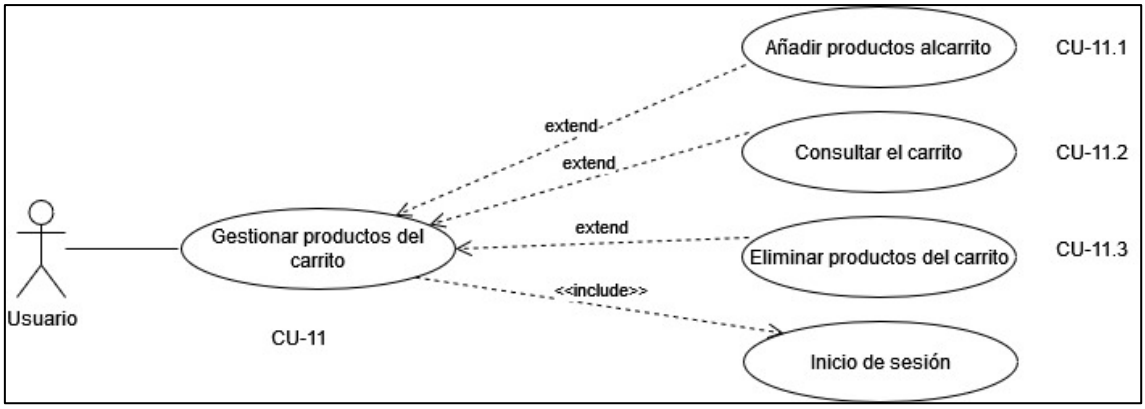


Fig. 3.7. Casos de uso relacionados con el carrito.

Relacionados con el proceso de compra

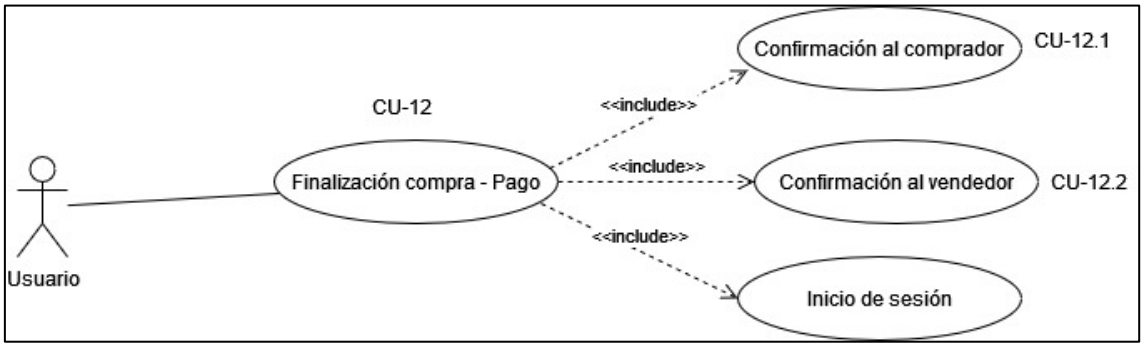


Fig. 3.8. Casos de uso relacionados con las compras.

Relacionados con el histórico de compra-venta

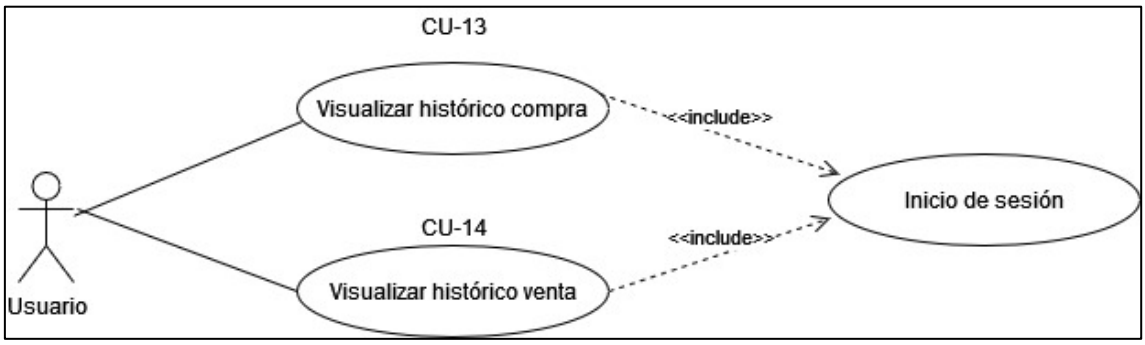


Fig. 3.9. Casos de uso relacionados con el histórico de compras.

Especificación de los requisitos del sistema

Como se ha mencionado anteriormente, los requisitos del sistema cubren desde el punto de vista técnico lo que los requisitos de usuario señalan que los usuarios pueden realizar en el sistema. Supone el segundo paso del proceso de elicitación de requisitos y se trata de un

aspecto fundamental, pues detalla cómo esas funciones definidas en los requisitos de usuario deben llevarse a cabo. Dentro de esta categoría existen dos tipos de requisitos; requisitos funcionales y no funcionales, cuyo detalle se señala a continuación.

- **Requisitos funcionales:** Definen las operaciones que el software debe realizar para ofrecer las funcionalidades especificadas en los requisitos de usuario.
- **Requisitos no funcionales:** Definen las restricciones con las que trata el software, relativas al equipo utilizado, compatibilidades con otros software, etc.

Con el objetivo de realizar una especificación de requisitos clara y coherente, se va a utilizar una plantilla similar a la utilizada para los requisitos funcionales con las siguientes modificaciones:

- **ID:** El identificador del requisito utilizará el siguiente patrón: RS-XX-YY, donde RS hace referencia a los requisitos del sistema. XX puede tomar los valores F para los requisitos funcionales o NF para los no funcionales. Por último, YY hace referencia a la secuencia numérica de los requisitos, que empezarán por el código 01.
- Se añadirá un campo **RU** que servirá para conocer en qué requisito de usuario se basa el requisito del sistema en cuestión.
- Se añadirá un campo **CU** que servirá para conocer con qué caso de uso está asociado el requisito en cuestión.

La plantilla, por lo tanto, queda de la siguiente manera:

<ID>	**<Nombre del requisito>**
Estado	Propuesto |Validado | Cancelado
Versión	Fecha última de modificación
Prioridad	Baja |Media | Alta
Relevancia	Esencial | Deseable |Opcional
Verificabilidad	Verificable |No verificable
Autor	Nombre del autor
RU	Requisito de usuario involucrado
CU	Caso de uso involucrado
Descripción	

Tabla 3.23. Plantilla de requisitos de sistema.

3.2.2.2 Requisitos funcionales

En esta sección se van a detallar los requisitos funcionales siguiendo la clasificación realizada con los requisitos de usuario.

3.2.2.2.1 Relacionados con el registro de usuario

RS-F-01	**Registro de usuario**
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-01
CU	CU-01
Descripción	El sistema de registro requerirá los siguientes campos: nombre, apellidos, email, contraseña y ciudad de residencia. Estos campos deberán estar almacenados en la base de datos.

Tabla 3.24. Requisito de sistema 01.

RS-F-02	**Inicio de sesión del usuario (I)**
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-02
CU	CU-02
Descripción	El inicio de sesión en el sistema requerirá introducir email y contraseña.

Tabla 3.25. Requisito de sistema 02.

RS-F-03	**Inicio de sesión del usuario (II)**
Estado	Validado
Versión	19/10/2021
Prioridad	Media
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-02
CU	CU-02
Descripción	El sistema mostrará una advertencia en caso de error en el proceso de autenticación del usuario.

Tabla 3.26. Requisito de sistema 03.

RS-F-04	**Visualización de los datos del usuario**
Estado	Validado
Versión	19/10/2021
Prioridad	Media
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-03
CU	CU-03
Descripción	El sistema debe ofrecer una sección donde visualizar los datos de registro del usuario tras consultarlos en la base de datos. Para esta acción se requiere inicio de sesión.

Tabla 3.27. Requisito de sistema 04.

RS-F-05	**Modificación datos del usuario**
Estado	Validado
Versión	19/10/2021
Prioridad	Baja
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-04
CU	CU-04
Descripción	El sistema deberá permitir modificar los datos de registro de un usuario. Para ello, se le mostrará el formulario de registro con sus datos rellenos y se permitirá la modificación de cualquiera de los datos y su correspondiente actualización en la base de datos. Para esta acción se requiere inicio de sesión.

Tabla 3.28. Requisito de sistema 05.

RS-F-06	**Cierre de sesión del usuario**
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-05
CU	CU-05
Descripción	El sistema deberá permitir cerrar sesión únicamente si se ha iniciado la sesión previamente.

Tabla 3.29. Requisito de sistema 06.

RS-F-07	Baja del usuario en el sistema
Estado	Validado
Versión	19/10/2021
Prioridad	Media
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-06
CU	CU-06
Descripción	El sistema debe permitir al usuario darse de baja en la aplicación. Esta acción elimina los datos del usuario en la base de datos además de cualquier registro de su actividad en la aplicación. Para esta acción se requiere inicio de sesión.

Tabla 3.30. Requisito de sistema 07.

3.2.2.2.2 Relacionados con los productos

RS-F-08	Puesta a la venta de un producto
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-07
CU	CU-07.1
Descripción	El sistema debe mostrar un formulario con una serie de campos para rellenar con las características del producto y que tendrán su reflejo en la base de datos. Para esta acción se requiere inicio de sesión.

Tabla 3.31. Requisito de sistema 08.

RS-F-09	Consulta de los productos propios
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-08
CU	CU-07.2
Descripción	El sistema debe mostrar al usuario los productos que tiene a la venta en una sección en la que únicamente aparezcan sus productos. Para esta acción se requiere inicio de sesión.

Tabla 3.32. Requisito de sistema 09.

RS-F-10	Modificación de los datos de un producto
Estado	Validado
Versión	19/10/2021
Prioridad	Media
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-09
CU	CU-07.3
Descripción	El sistema debe permitir modificar la información de un producto. Para ello, se le mostrará al usuario el formulario que rellenó para poner a la venta del producto, con los datos rellenos y se permitirá la modificación de cualquiera de los datos y su correspondiente actualización en la base de datos. Para esta acción se requiere inicio de sesión.

Tabla 3.33. Requisito de sistema 10.

RS-F-11	Eliminar un producto
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-10
CU	CU-07.4
Descripción	El sistema debe permitir al usuario dar de baja un producto de su propiedad de la aplicación, eliminado así sus datos de la base de datos además de cualquier registro en el que se vea involucrado el producto. Para esta acción se requiere inicio de sesión.

Tabla 3.34. Requisito de sistema 11.

RS-F-12	Búsqueda simple
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-11
CU	CU-08
Descripción	El sistema debe permitir realizar una búsqueda por palabras clave. El usuario introducirá una o varias palabras y se le mostrarán como resultados una serie de productos que contengan esas palabras en el nombre o en la descripción.

Tabla 3.35. Requisito de sistema 12.

RS-F-13	Búsqueda avanzada
Estado	Validado
Versión	19/10/2021
Prioridad	Media
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-12
CU	CU-09
Descripción	El sistema debe permitir realizar una búsqueda mediante la selección de varios campos en un formulario. El usuario completará los datos que considere en el formulario y tras enviar el formulario se le mostrarán como resultados una serie de productos que sean compatibles con los datos seleccionados.

Tabla 3.36. Requisito de sistema 13.

RS-F-14	Detalle del producto
Estado	Validado
Versión	19/10/2021
Prioridad	Baja
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-13
CU	CU-10
Descripción	El sistema debe permitir visualizar la información detallada de un producto, tras su consulta a la base de datos.

Tabla 3.37. Requisito de sistema 14.

3.2.2.2.3 Relacionados con el carrito

RS-F-15	Añadir productos al carrito (I)
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-14
CU	CU-11.1
Descripción	El sistema debe permitir añadir productos al carrito de la compra pulsando el botón habilitado para ello, tanto desde la descripción del mismo como desde la página de resultados. Para esta acción se requiere inicio de sesión.

Tabla 3.38. Requisito de sistema 15.

RS-F-16	Añadir productos al carrito (II)
Estado	Validado
Versión	19/10/2021
Prioridad	Media
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-14
CU	CU-11.1
Descripción	El producto añadido al carrito no debe mostrarse en las páginas de resultados. La información del carrito debe persistir en la base de datos.

Tabla 3.39. Requisito de sistema 16.

RS-F-17	Consultar el carrito
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-15
CU	CU-11.2
Descripción	El sistema debe permitir observar los productos existentes en el carrito del usuario mediante una consulta a la base de datos. Para esta acción se requiere inicio de sesión.

Tabla 3.40. Requisito de sistema 17.

RS-F-18	Eliminar productos del carrito (I)
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-16
CU	CU-11.3
Descripción	El sistema debe permitir eliminar productos del carrito. Para esta acción se requiere inicio de sesión.

Tabla 3.41. Requisito de sistema 18.

RS-F-19	Eliminar productos del carrito (II)
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial

Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-16
CU	CU-11.3
Descripción	El producto eliminado del carrito debe poder visualizarse de nuevo en las páginas de resultados. Debe actualizarse en consecuencia la base de datos.

Tabla 3.42. Requisito de sistema 19.

3.2.2.2.3 Relacionados con el proceso de compra

RS-F-20	**Finalizar el proceso de compra**
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-17
CU	CU-12
Descripción	El sistema debe permitir finalizar la compra de los productos del carrito mediante el pago con tarjeta bancaria. Los detalles de la compra persistirán en la base de datos. Para esta acción se requiere inicio de sesión.

Tabla 3.43. Requisito de sistema 20.

RS-F-21	**Confirmaciones al comprador**
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-18
CU	CU-12.1
Descripción	Una vez se haya realizado el pago se mostrará un mensaje de confirmación al comprador, mostrando como resumen de la compra, el nombre y precio de cada artículo comprado, así como el total abonado.

Tabla 3.44. Requisito de sistema 21.

RS-F-22	**Notificación al vendedor**
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable

Autor	Carlos Salinero Sánchez
RU	RU-18
CU	CU-12.2
Descripción	Una vez se haya producido una compra, al vendedor del producto se le mostrará un mensaje indicando que su producto ha sido vendido cuando inicie sesión en el sistema.

Tabla 3.45. Requisito de sistema 22.

3.2.2.2.4 Relacionados con el histórico de compra-venta

RS-F-23	**Visualización de las compras realizadas**
Estado	Validado
Versión	19/10/2021
Prioridad	Media
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-19
CU	CU-13
Descripción	El sistema deberá mostrar un registro de las compras realizadas en el que se detallará la fecha de la compra, el email del vendedor, los artículos adquiridos y el precio pagado por cada artículo. Para esta acción se requiere inicio de sesión.

Tabla 3.46. Requisito de sistema 23.

RS-F-24	**Visualización de las ventas realizadas**
Estado	Validado
Versión	19/10/2021
Prioridad	Media
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
RU	RU-20
CU	CU-14
Descripción	El sistema deberá mostrar un registro de las ventas realizadas en el que se detallará la fecha de la compra, el email del comprador, los artículos vendidos y el precio pagado por cada artículo. Para esta acción se requiere inicio de sesión.

Tabla 3.47. Requisito de sistema 24.

3.2.2.3 Requisitos no funcionales

A continuación, se presentan los requisitos no funcionales, los cuales representan una serie de condiciones o restricciones que debe cumplir el sistema.

RS-NF-01	**Navegación en la web**
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El usuario, si no ha iniciado sesión, podrá acceder a la web y navegar por ella pero con acceso limitado a las funciones de búsqueda simple y detalles del producto, correspondientes a los requisitos de usuario; RU-11 y RU-13

Tabla 3.48. Requisito no funcional 01.

RS-NF-02	**Navegadores compatibles**
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El sistema debe funcionar en los navegadores Mozilla Firefox y Google Chrome.

Tabla 3.49. Requisito no funcional 02.

RS-NF-03	**Información actualizada**
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	La información que aparezca en la interfaz de usuario debe ser coherente con la que persiste en la base de datos.

Tabla 3.50. Requisito no funcional 03.

RS-NF-04	**Entorno de ejecución**
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El sistema debe poder ser ejecutado y debe permitir realizar todas las pruebas definidas, utilizando un único ordenador.

Tabla 3.51. Requisito no funcional 04.

RS-NF-05	**Persistencia de los datos**
Estado	Validado
Versión	19/10/2021
Prioridad	Alta
Relevancia	Esencial
Verificabilidad	Verificable
Autor	Carlos Salinero Sánchez
Descripción	El sistema debe hacer uso de una base de datos para la persistencia de la información.

Tabla 3.52. Requisito no funcional 05

4. JUSTIFICACIÓN DE LA SOLUCIÓN

En esta sección se incluye una justificación completa de la solución al problema planteado, tratando la elección de las tecnologías a utilizar, el diseño de la arquitectura del sistema y el detalle de la implementación.

4.1 Elección de las tecnologías utilizadas

En esta sección se pretende justificar la elección de las tecnologías que se utilizaran en el desarrollo del proyecto siguiendo el orden en el que fueron expuestas, con el objetivo de facilitar la comprensión.

Para el desarrollo de la parte ***backend*** se hará uso de *Java Enterprise Edition* debido a que se trata de una tecnología multiplataforma que permite su uso en cualquier servidor o dispositivo y permite la implementación de múltiples arquitecturas debido a su versatilidad, además de ser utilizado en aplicaciones web de diversas empresas como Orange o Dell [33], [34]. Java cuenta con *frameworks* especializados como *Spring*, el cual se va a utilizar en el desarrollo del proyecto. La principal ventaja de este entorno de trabajo es que permite a los desarrolladores centrar su actividad en la lógica de negocio mediante la minimización de errores, y la automatización de tareas repetitivas [35]. *Spring* facilita la utilización de patrones arquitectónicos, proporciona mecanismos de persistencia de los datos y de acceso a los mismos y permite la integración de pruebas unitarias en el código del sistema.

LOGO SPRING

Pese a que Python es otra opción totalmente válida para el desarrollo del *backend* de este proyecto, la madurez del lenguaje Java permite que haya mayor cantidad de documentación accesible desde internet, lo que facilita la tarea del desarrollador. Además, permite el desarrollo de la parte *frontend* en el propio entorno de trabajo, tal y como se expone a continuación.

La parte ***frontend*** también va a estar basada en la tecnología *Java Enterprise Edition*, concretamente en las *Java Server Pages* o JSP, que son páginas HTML con código Java incrustado que se ejecuta del lado del servidor y que permiten la creación de páginas web dinámicas. Estas páginas pueden incluir también elementos CSS y código JavaScript para dotar al sitio web de un aspecto más profesional y una calidad percibida mayor. Cabe destacar que la parte *frontend* también cuenta con cierta lógica de la aplicación y por tanto también utiliza el propio lenguaje Java, tal y como se expone en la sección Detalles de implementación.

Esta elección se basa en la simplificación de desarrollo que proporciona *Spring* al permitir el desarrollo de la totalidad de la aplicación en un mismo entorno sin prescindir de calidad o robustez en el producto final.

Para la persistencia de los datos de la aplicación se hará uso de una **base de datos relacional.** Las bases de datos no relacionales son muy útiles en escenarios donde no se es consciente de qué tipo de información se va a querer almacenar o no es fácil estructurar estos datos, pero en el ámbito del presente proyecto si que se conoce qué información debe persistir ya que a grandes rasgos se pueden identificar unidades de persistencia básicas como usuarios y productos, ambos con una serie de detalles fácilmente identificables. En el caso de los usuarios, por ejemplo, se querrá almacenar el nombre, los apellidos, el correo electrónico, la contraseña de acceso y quizá algún campo más, pero eso corresponde ya al diseño de la base de datos propiamente dicho. Las bases de datos relacionales resultan apropiadas en estos casos debido a que además de almacenar los datos, ofrecen las ya mencionadas propiedades ACID, que garantizan una mayor robustez [36].

Se hará uso de **MySQL** como sistema gestor de bases de datos debido a que se trata de un sistema multiplataforma de manera que pueda ser utilizado independientemente del ordenador en el que se trabaje. Cuenta con gran cantidad de documentación accesible en internet, necesita de pocos recursos de memoria RAM y CPU, ofrece un buen rendimiento, similar al de otros sistemas gestores y cuenta con el *framework* MySQL Workbench, que permite una gestión simple e intuitiva de la base de datos [37]–[39].

Con el objetivo de construir una aplicación robusta y escalable la arquitectura del sistema va a estar basada en **microservicios**, de manera que la lógica de negocio se separe en diferentes módulos claramente diferenciados e independientes entre sí, dotando así al sistema de las dos grandes ventajas mencionadas anteriormente de este tipo de arquitectura: capacidad para escalar el módulo de la aplicación necesario (el del registro, el de proceso de compra… etc) y un aislamiento de errores que permite una actuación más rápida y eficaz. *Spring* cuenta con la herramienta ***Spring Boot***, que permite la creación de aplicaciones Java con una configuración simplificada, minimiza la gestión de dependencias y por lo tanto permite centrar la labor en el desarrollo del sistema sin invertir demasiado tiempo en aspectos que realmente no aportan valor al producto final [40].

Spring Boot permite la elección de “starters”, un conjunto de dependencias que suelen ir relacionadas, de manera que no sea necesario buscar e importar una gran cantidad de dependencias, sino un conjunto de ellas directamente. Además, el despliegue de los

microservicios en el servidor de aplicaciones es tan sencillo como pulsar un botón, de manera que todas estas características en conjunto hacen que sea una tecnología adecuada para este proyecto.

Como servidor que permite la ejecución de la aplicación se ha escogido **Glassfish**, debido a que se trata de la implementación por referencia de las aplicaciones J2EE, soportando así todas las tecnologías utilizadas en el ámbito de este proyecto [41]. Además, su configuración es sencilla y la experiencia de uso en proyectos anteriores es positiva.

La naturaleza de este proyecto provoca que sea el autor de este documento el único diseñador y desarrollador del sistema. Esto hecho, unido a un alcance bastante delimitado, limita la posibilidad de trabajar en base a una metodología ágil caracterizada por responder de manera eficaz ante los cambios y pensada para una acción conjunta de un equipo. Por otro lado, el desarrollo en cascada ofrece un proceso demasiado rígido al no permitir cambios en los requisitos en ninguna fase posterior al análisis de los requerimientos del proyecto, lo que supone una restricción demasiado fuerte para este proyecto. Por este motivo, se hará uso de una **metodología** que combine la eficacia de un desarrollo secuencial, de toma de requisitos, diseño, implementación y verificación, con la flexibilidad de que otorgan las metodologías ágiles respecto a la modificación o ampliación de requisitos. Una metodología que agrupa estas características es la denominada **Incremental**, que se basa en agregar funcionalidades en cada etapa de desarrollo y comprobar el estado de la aplicación, de manera que los requisitos pueden ser modificados durante el periodo de implementación del proyecto.

4.2 Diseño de la base de datos

El primer paso en la construcción del sistema es el diseño de la base de datos, y en esta sección se muestra el detalle de este diseño. Como punto de partida, se muestra en la imagen que se encuentra a continuación, el esquema relacional que representa la base de datos utilizada en este proyecto.

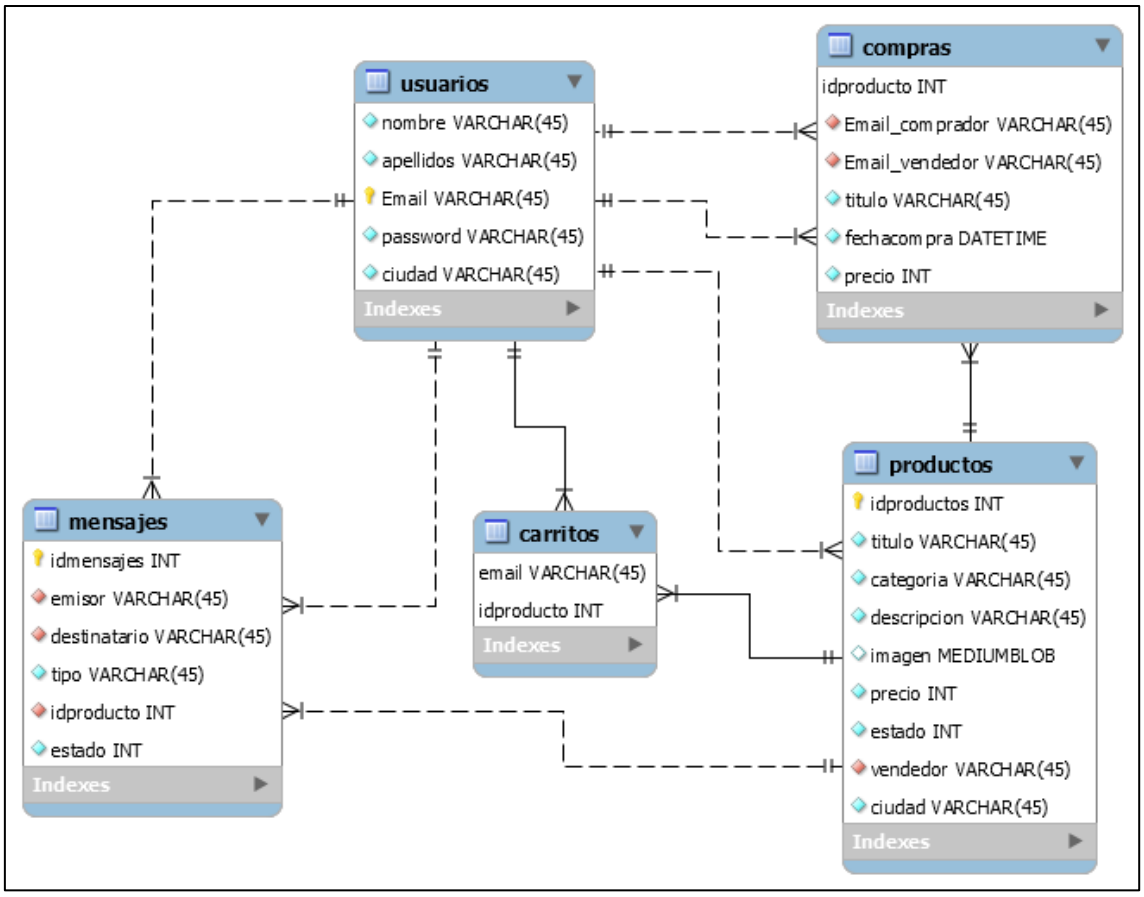
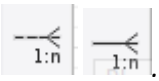


Fig. 4.1. Esquema relacional

Nota aclarativa del esquema: Este diagrama es extraído del *framework MySQL Workbench*, y las relaciones entre tablas se representan mediante líneas continuas, si en ambas partes de la relación se ven involucradas las claves primarias de ambas tablas, como entre productos y compras, donde la relación se establece entre el id del producto, siendo éste clave primaria en ambas relaciones. Por otro lado, las líneas discontinuas señalan que al menos en un lado de la relación, el atributo partícipe de la relación no es clave primaria, como por ejemplo entre mensajes y usuarios, donde los atributos emisor y destinatario de la tabla mensajes no forman parte de la clave primaria.

Por otro lado, la cardinalidad de las relaciones se puede identificar con esta pequeña leyenda extraída del propio *framework*. 

Por último, cabe destacar que el diagrama que se extrae de este entorno de trabajo tiene un error en la relación entre las tablas productos y compras, donde la relación debe ser 1 a 1 debido a que la tabla compras referencia a la tabla productos por el atributo id de producto, siendo éste clave primaria en ambas relaciones.

A continuación, se expone la estructura de las tablas, con las aclaraciones correspondientes para facilitar su comprensión.

- **Tabla Usuarios.** Se almacenan los siguientes datos básicos de un usuario.
    - **Nombre** del usuario.
    - **Apellidos** del usuario.
    - **Email**, que será la clave primaria de la tabla al identificar unívocamente a un usuario.
    - **Password**, necesaria junto al Email para la autenticación del usuario.
    - **Ciudad** en la que reside el usuario.
- **Tabla Productos.** Se almacenan una serie de datos relativos a un producto que lo caracterizan:
    - **Idproductos**, representado por un número entero que se incrementa con cada nueva entrada en la tabla y que define unívocamente a un producto, de manera que se considera como clave primaria.
    - **Título**, que resume el producto ofertado.
    - **Categoría**, que tendrá un valor dentro del siguiente dominio; {Padel, Tenis, Fútbol, Baloncesto, Ciclismo, Natación, Hípica, Running, Boxeo Otros}
    - **Descripción,** que amplía la definición del producto.
    - **Imagen,** definida con el tipo de datos MEDIUMBLOB, usado comúnmente para almacenar imágenes o sonidos.
    - **Precio.**
    - **Estado,** representado por un número entero donde un 1 significa que el producto está disponible y un 0 que el producto ha sido vendido.
    - **Vendedor**¸ forma parte de una clave ajena que referencia al atributo “Email” de la tabla usuarios, de manera que si se elimina un usuario, se eliminan todos los productos de los que este usuario es propietario.
    - **Ciudad** en la que se encuentra el producto, con el objetivo de que sea un filtro más en las búsquedas avanzadas.
- **Tabla Carritos.** Está formada por los siguientes atributos:
    - **Email**, que referencia al atributo “Email” de la tabla usuarios de manera que si se borra un usuario se borran todas las filas de esta tabla que contengan el email del usuario.

- **Idproducto**, que referencia al atributo “idproductos” de la tabla productos de manera que si se borra un producto se borran todas las filas de esta tabla que contengan el id de ese producto.

Ambos atributos forman parte de la clave primaria de esta tabla, de manera que todas las filas que contengan el email de un usuario forman el carrito de ese usuario.

- **Tabla Compras.** Esta tabla almacena las compras realizadas en la aplicación.
  - **Idproducto** referencia al atributo “idproductos” de la tabla productos y constituye la clave primaria de la tabla.
  - **Email_comprador e Email_vendedor** referencian a la tabla usuarios de manera que si un usuario se elimina de la base de datos, se eliminan todas las filas de la tabla compras en las que aparezcan.
  - **Título**, que será el del producto adquirido.
  - **FechaCompra,** que almacena la fecha en la que se realizó la transacción.
  - **Precio** del producto comprado.

  De esta manera, una compra estará formada por múltiples filas de esta tabla, donde coincidan el email del comprador y la fecha
- **Tabla Mensajes.** Esta tabla almacena los mensajes de confirmación de compra y de notificación al vendedor, enviados una vez se haya completado el proceso de compra. Entre los atributos que se muestran, cabe destacar los siguientes:
  - **Idmensajes,** representado por un número entero que se incrementa con cada entrada nueva en la tabla y que define unívocamente a un mensaje, de manera que se considera como clave primaria.
  - **Emisor y destinatario** se relacionan con la tabla usuarios.
  - **Tipo,** que puede contener el texto “resumenCompra” o “confirmacionVendedor” indicando así, si se trata de un mensaje dirigido a resumir la compra o si por el contrario, se dirige al vendedor de un producto a modo de notificación de su venta.
  - **Idproducto,** se relaciona con la tabla productos.
  - **Estado,** representado por un número entero donde un 1 significa que el mensaje ha sido leído y un 0 que está pendiente de visualizar. El motivo de este atributo es el hecho de que los mensajes solo pueden visualizarse una vez, de manera que una vez han sido leídos, dejan de persistir en la base de datos.

4.3 Arquitectura

En esta sección se va a detallar el modelo arquitectónico diseñado para la implementación del portal siguiendo el modelo de las 4C; contexto, contenedores, componentes y código, que permiten definir un detalle a diferentes niveles de este modelo arquitectónico. Cabe destacar que las vistas de componentes y código se han desarrollado juntas al considerar que de esta manera se proporciona una explicación más didáctica.

4.3.1 Contexto

En este punto se incluye un diagrama que muestra el sistema dentro de su entorno operacional, con un detalle a alto nivel sobre la función principal del sistema.

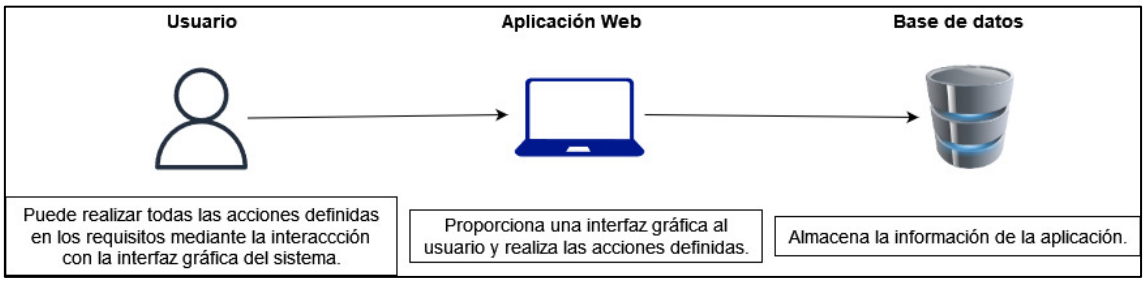


Fig. 4.2. Diagrama de contexto

4.3.2 Contenedores

En este punto se incluye un diagrama que muestra a alto nivel, pero con un mayor grado de detalle, el funcionamiento del sistema, agrupando las distintas partes de este funcionamiento en contenedores. Un contenedor es cualquier parte del sistema que contenga datos o código, de manera que este diagrama muestra un reparto de las responsabilidades dentro del sistema y está dirigido a conocer cómo los elementos internos y externos de la aplicación se relacionan entre sí.

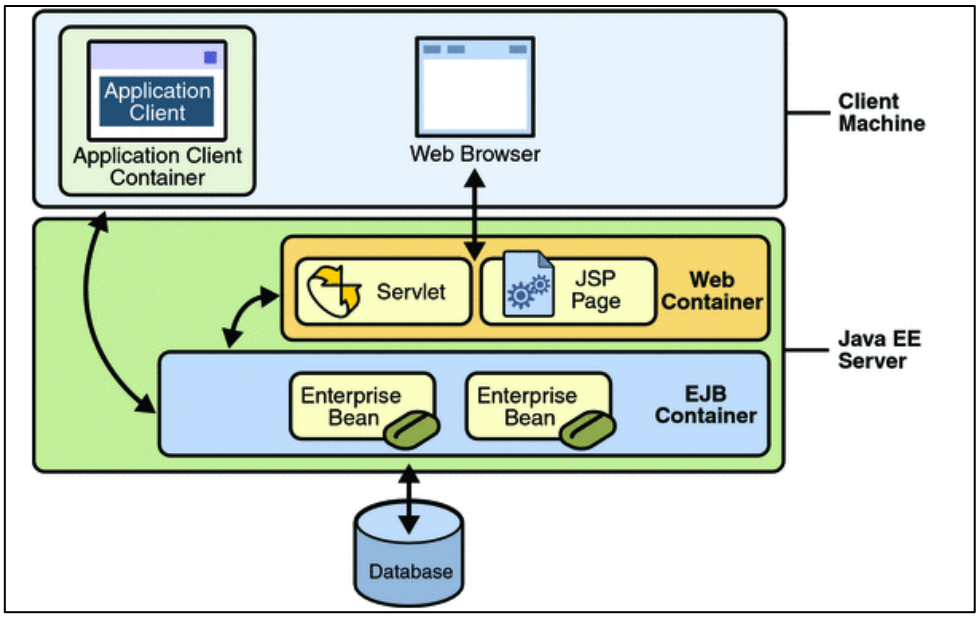


Fig. 4.3. Diagrama de contenedores [42]

Este diagrama obtenido de la página oficial de Oracle muestra dos contenedores claramente diferenciados. En el primero, el usuario final, utiliza un navegador web desde su ordenador personal para interactuar con el sistema, que se encuentra en ejecución en el segundo contenedor, que en este caso se trata del servidor Glassfish que soporta el estándar Java EE.

Glassfish es un servidor de aplicaciones que consta de un contenedor web, que supone un entorno de ejecución para las distintas partes de la aplicación (explicadas en detalle en la siguiente vista de la arquitectura), y contiene además un contenedor EJB, encargado del tratamiento de las distintas unidades que persisten en la base de datos y que tienen su correspondencia con los objetos con los que se trata en código Java [43]. Por último, cabe destacar que el sistema accede a la base de datos, la cual se considera como un contenedor a parte.

4.3.3 Componentes

Como se ha mencionado, la arquitectura elegida para este proyecto es la basada en microservicios, la cual permite separar la lógica de la aplicación en diferentes módulos totalmente independientes llamados microservicios, los cuales son invocados desde la parte *frontend* de la aplicación para realizar diversas tareas relacionadas principalmente con el tratamiento de la información de la base de datos. La comunicación mediante el *frontend* y el *backend* se realiza mediante llamadas a servicios **REST** utilizando el formato JSON para el intercambio de datos**.** Esta comunicación se basa en el protocolo HTTP, en el que interviene

únicamente un objeto request que contiene los datos de la petición, y un objeto response que contiene los datos de respuesta del servidor. Esta tecnología permite el intercambio de información entre el cliente y el servidor mediante el uso de URL´s y su uso se explicará posteriormente en la sección Postman y servicios REST.

Los microservicios diseñados e implementados para realizar la lógica de negocio de la aplicación son los que se presentan a continuación junto con las funcionalidades que realizan:

4.3.3.1 Sportunity-ms-Users

Es el microservicio encargado de la gestión de los usuarios dentro del sistema y realiza las operaciones CRUD correspondientes. Estas operaciones son la creación (Create), la lectura (Read), la modificación (Update) y el borrado (Delete) de usuarios en la base de datos, con el fin de cubrir las funcionalidades descritas en los requisitos relativas al registro, el inicio de sesión etc…

La estructura del microservicio sigue el patrón arquitectónico **MVC,** basado en la separación del tratamiento de los datos del sistema (Modelo), el control del flujo de ejecución (Controlador) y la presentación al usuario (Vista), siendo esta última, parte de la aplicación *frontend* y por lo tanto no está presente en el microservicio. La estructura definida es la siguiente:

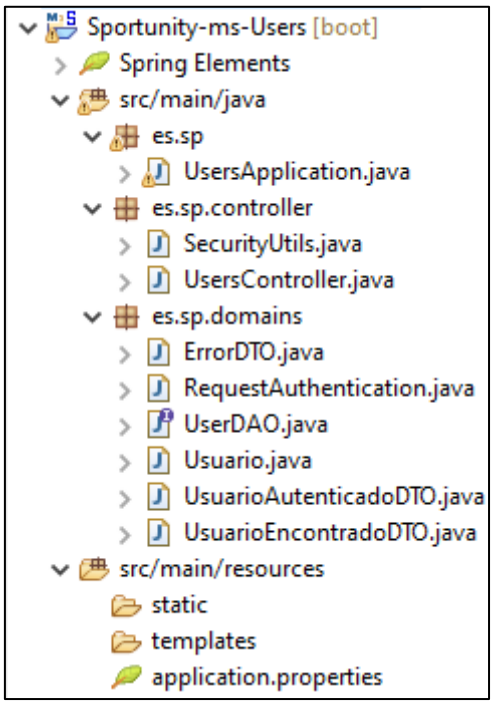


Fig. 4.4. Estructura de código del componente Sportunity-ms-Users.

Se dispone de 3 paquetes, cuya nomenclatura tiene la raíz “es.sp”, derivado de España y Sportunity, con el objetivo de definir una nomenclatura estándar para el proyecto. A

continuación, se expone la función de cada clase dentro del microservicio siguiendo la paquetería expuesta

- **es.sp.** La clase *UsersApplication* es generada automáticamente por SpringBoot y es la encargada de ejecutar el microservicio.
- **es.sp.controller.** Contiene la clase *UsersController*, que implementa cada operación CRUD y la asocia a una URL, de manera que cuando el *frontend* realice una llamada a alguna de las URL´s definidas, con los parámetros requeridos por cada una de ellas, se realizará la función definida para esa URL. Cabe destacar que estas URL´s se denominan ***endpoints*** y pueden existir dos iguales debido a que SpringBoot permite tener endpoints iguales si se utiliza un método HTTP distinto. Los métodos HTTP utilizados en este desarrollo han sido; POST para crear un nuevo usuario, GET para obtener un usuario, PUT para modificar un usuario y DELETE para eliminarlo. Por otro lado, la clase *SecurityUtils* contiene las funciones necesarias para la creación y validación de un token de sesión, como parte del proceso de autenticación definido en la sección Seguridad con JSON Web Token.
- **es.sp.domains.** Contiene las clases que definen el modelo del sistema. Por un lado, la clase *Usuario* contiene la definición de un usuario, con los mismo campos que los definidos en el modelo relacional presentado anteriormente, con el objetivo de realizar una correspondencia objeto-relacional, es decir, lo que comúnmente se conoce como un "mapeo" entre una entidad de la base de datos y un objeto Java. Para automatizar ese mapeo, Spring ofrece las anotaciones "@Entity" y "@Table" que permiten identificar una clase Java como una unidad de persistencia y asociarla a una tabla de la base de datos. Por otro lado, la clase *UserDAO* proporciona el acceso a la base de datos mediate las operaciones CRUD definidas en la interfaz pública CrudRepository. El resto de clases corresponden a elementos que forman las peticiones y respuestas de diversos servicios definidos en la clase *UsersController.*
- El fichero **application.properties** contiene una serie de propiedades esenciales para el funcionamiento del microservicio, como la url de acceso a la base de datos, el usuario y contraseña para acceder a esta base de datos, y el puerto en el que el microservicio se despliega en el servidor local. Este puerto debe ser diferente en cada microservicio para que todos puedan ser desplegados simultáneamente.
- Cabe destacar que cada microservicio cuenta con un archivo **pom.xml**, que contiene una serie de dependencias externas usadas por SpringBoot, entre las cuales se deben incluir "Spring JPA Data", que permite el acceso a la base de datos mediante la ya

mencionada tecnología JPA y “Spring Web”, que permite la implementación del patrón MVC.

4.3.3.2 Sportunity-ms-Products

Es el microservicio encargado de la gestión de los productos dentro del sistema y realiza las operaciones CRUD correspondientes utilizando el patrón MVC y siguiendo la misma estructura que la mostrada en el microservicio de usuarios. Con el fin de no repetir lo comentado en el caso anterior, únicamente se muestra la estructura dentro del entorno de desarrollo.

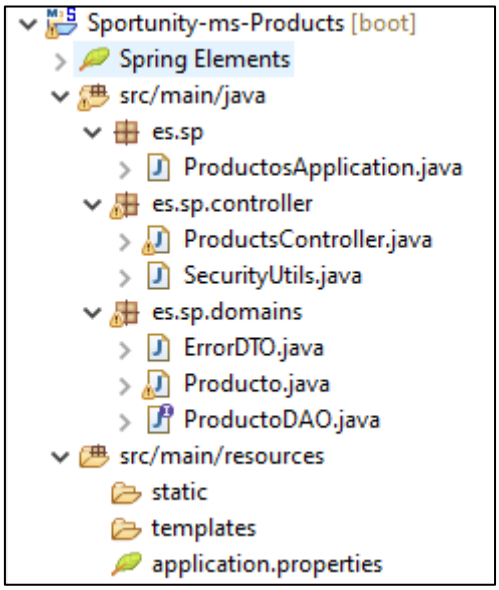


Fig. 4.5. Estructura de código del componente Sportunity-ms-Products

4.3.3.3 Sportunity-ms-Compras_Carrito

Es el microservicio encargado de la gestión de las compras y del carrito de la compra dentro del sistema y realiza las operaciones CRUD correspondientes utilizando el patrón MVC y siguiendo la misma estructura que la mostrada en los anteriores microservicios. Con el fin de no repetir lo comentado en los casos anteriores, únicamente se muestra la estructura dentro del entorno de desarrollo.

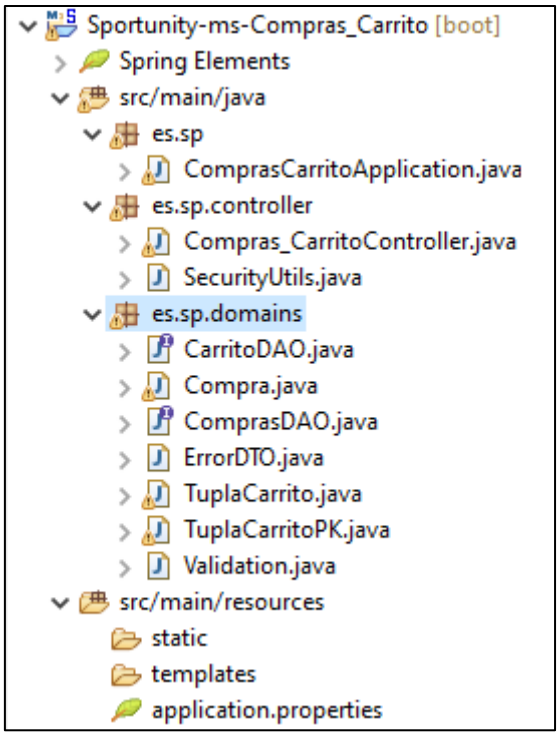


Fig. 4.6. Estructura de código del componente Sportunity-ms-Compras_Carrito

En este caso, es necesario crear una clase llamada TuplaCarritoPK, que contiene los dos atributos de cada entrada de la tabla Carrito; email e id del producto. El motivo de esta clase es generar el atributo identificativo de cada entrada de esta tabla, pues ambos campos son parte de la clave primaria de la tabla y de esta manera poder definir un atributo de tipo TuplaCarritoPK con la anotación @EmbeddedId en la clase TuplaCarrito, que es la que se corresponde con la tabla carritos de la base de datos.

4.3.3.4 Sportunity-ms-Messages

Es el microservicio encargado del envío de mensajes de confirmación al comprador y al vendedor dentro de una operación de compra de los artículos del carrito. Se encarga de las operaciones CRUD correspondientes utilizando el patrón MVC y siguiendo la misma estructura que la mostrada en los anteriores microservicios.

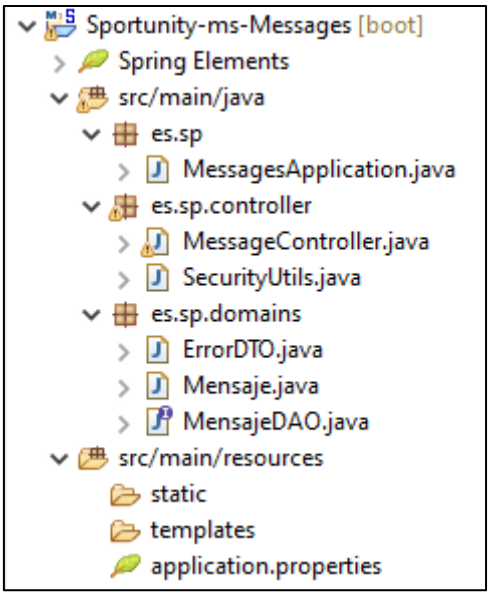


Fig. 4.7. Estructura de código del componente Sportunity-ms-Messages.

4.3.3.4 Sportunity-User-App

Una vez detallada la parte *backend* del sistema, a continuación, se presenta el detalle de la parte *frontend*. Se trata de una aplicación de usuario de tipo "Dynamic Web Project", es decir, un sitio web dinámico que permite al servidor realizar modificaciones en el código HTML antes de devolverlo con el fin de conseguir una interfaz con la que el usuario pueda interactuar. Este funcionamiento fue explicado en la sección Paradigma cliente-servidor.

Esta aplicación recibe el nombre de ***Sportunity-User-App*** y presenta la siguiente estructura:

- **es.sp.controller.** Contiene la clase principal de la aplicación, el controlador o **Web Servlet**, que es el encargado de atender las peticiones HTTP desencadenadas por las acciones del usuario. Este controlador es el encargado de asignar una petición a un manejador determinado, que será el encargado de realizar las llamadas a los microservicios con el fin de realizar las modificaciones requeridas en la base de datos. Estos manejadores se localizan en los diferentes paquetes de este proyecto y están agrupados según la funcionalidad que desarrollen. Contiene además la definición de una interfaz que es implementada por cada manejador.
- Los paquetes **es.sp.login, es.sp.products, es.sp.messages y es.sp.compras_carrito** contienen los diferentes manejadores que se encargan de realizar las llamadas a los microservicios.
- **es.sp.persitence.** Contiene exactamente las mismas clases que representan las unidades de persistencia y que están incluidas en los microservicios, con el objetivo de poder realizar el envío y la recepción de los datos en las llamadas a los servicios REST.

- En la carpeta **WebContent** se encuentran todas las páginas que componen el sitio web y que tienen extensión .jsp debido a que se trata de las ya mencionadas "Java Server Pages" que permiten incluir código Java en el propio HTML. En esta carpeta también se incluye el archivo *web.xml*, que permite definir entre otras cosas, la página de inicio del sitio web y el archivo *glassfish-web.xml* que identifica el proyecto a desplegar. Cabe destacar en este punto, que los archivos con extensión .jsp tienen su origen en una **plantilla *frontend*** obtenida de https://templatemo.com/tm-559-zay-shop tal y como se expone en la sección Licencias y propiedad intelectual y que ha permitido simplificar el desarrollo de la parte visual de la aplicación, pues el foco del proyecto está en el diseño del sistema.

A continuación, se presenta un diagrama que muestra la interacción de los distintos componentes del sistema.

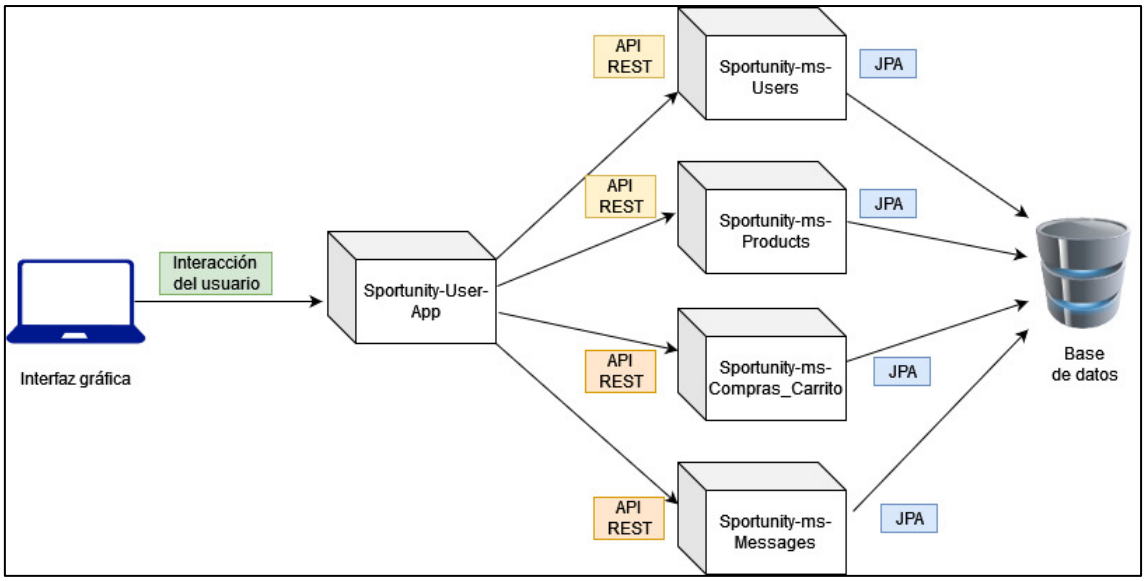
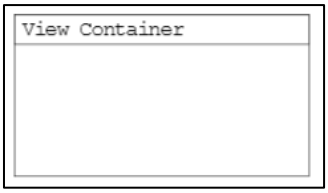


Fig. 4.8. Diagrama de componentes

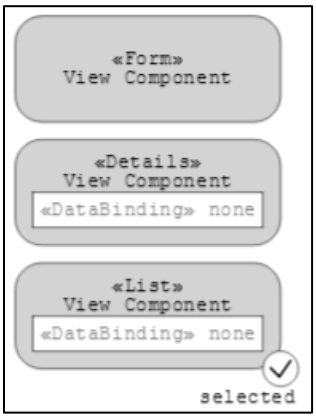
En el se observa cómo el usuario interactúa directamente con la parte *frontend* del sistema, la cual determina el tipo de acción solicitada y pide la acción del microservicio adecuado. Todos los microservicios acceden a la misma base de datos, si bien desde cada microservicio solo se accede a una parte determinada de esta base de datos, con el fin de conseguir una buena distribución del código y facilitar así el mantenimiento y la escalabilidad.

4.4 Modelado de la interfaz



En esta sección se va a hacer uso del lenguaje IFML (*Interaction Flow Modeling Language*) para modelar el flujo de información entre las distintas interfaces de la aplicación.

En este lenguaje de modelado se distinguen diferentes elementos:



- **View Container:** Cualquier contenedor de elementos, ya sea texto, imágenes etc.

Fig. 4.9. IFML- View container

- **View component:** Elemento que se le muestra al usuario, ya sea un menú, una lista de productos o imágenes.

- **Data Binding:** Representa una entidad de persitencia en el *backend*.

Fig. 4.10. IFML- View component

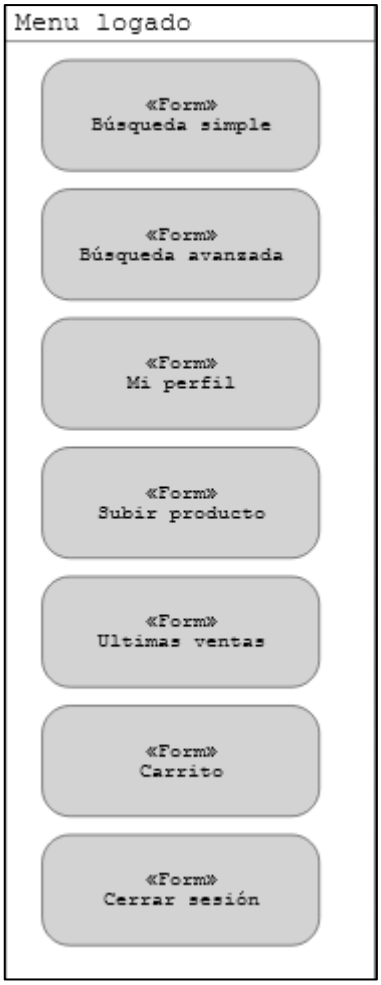
- **Evento:** Representa una acción ejecutada en el sistema, como seleccionar una opción de menú concreta.

Fig. 4.11. IFML- Event

- **Acción**: Se trata de un proceso desarrollado en el *backend* y que es provocado por un evento.

Fig. 4.12. IFML- Action

A continuación, se presenta el modelado completo de las interfaces de la aplicación, si bien cabe destacar, que por motivos de espacio y claridad, en el diseño completo se hacen referencia a dos componentes denominados *Menu logado* y *Menu sin logar*, que se muestran a continuación, y no en el diseño completo, para facilitar la claridad y comprensión del modelado.



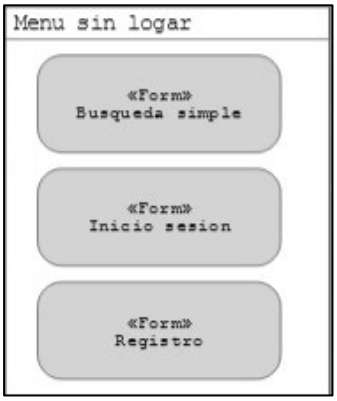


Fig. 4.13. IFML- Menú visible si se ha iniciado sesión Fig. 4.14. IFML - Menú visible sin iniciar sesión

Cabe destacar, además, que las páginas accesibles desde estos menús únicamente se han señalado desde las páginas *Inicio sin logar* e *Inicio logado*, si bien estos menús tienen la misma funcionalidad en cualquiera de las interfaces que se muestran en el diseño completo a continuación.

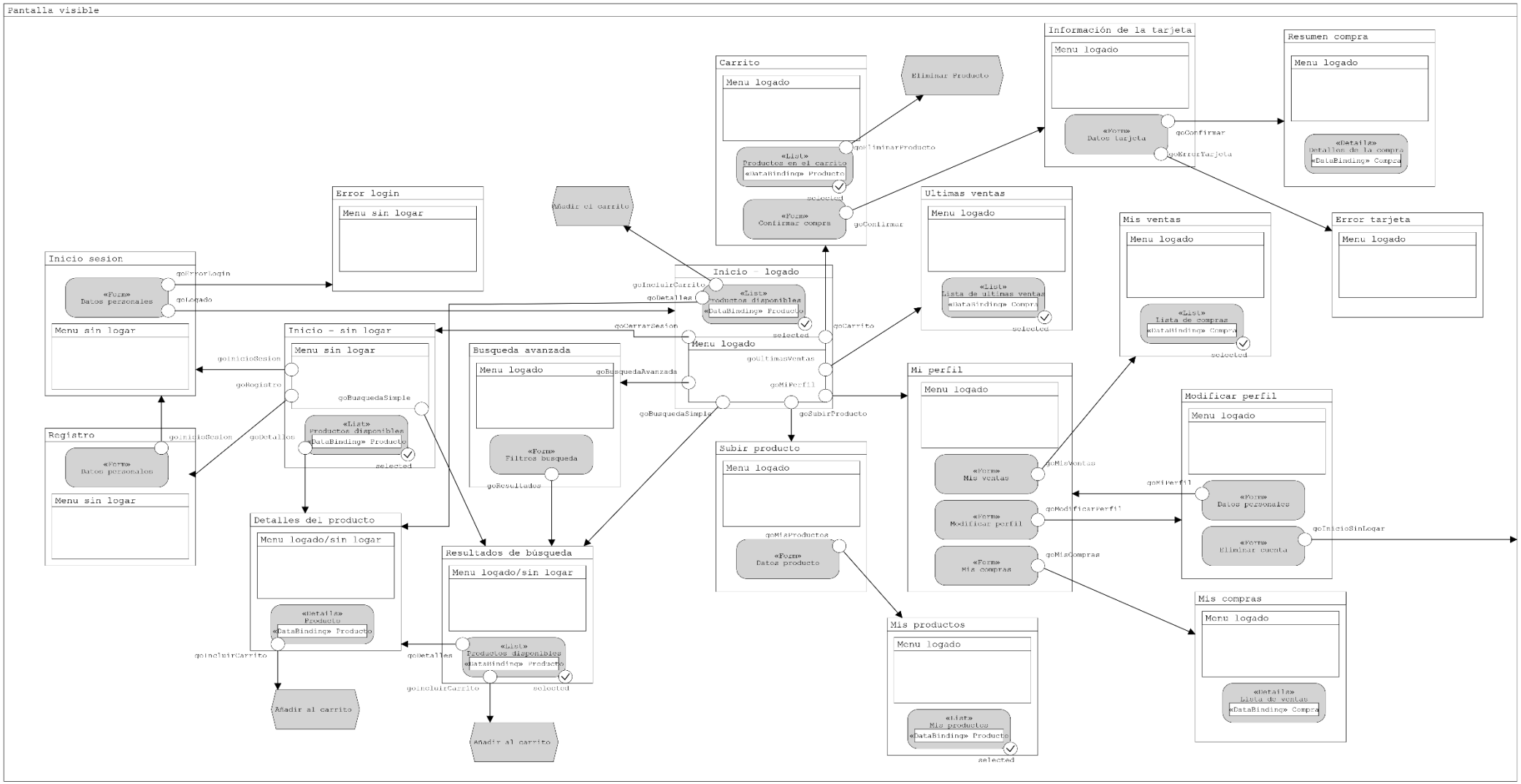


Fig. 4.15. IFML- Diagrama completo

Para facilitar la comprensión de este diagrama, se va a explicar página *Inicio-Logado* junto con sus interacciones asociadas.

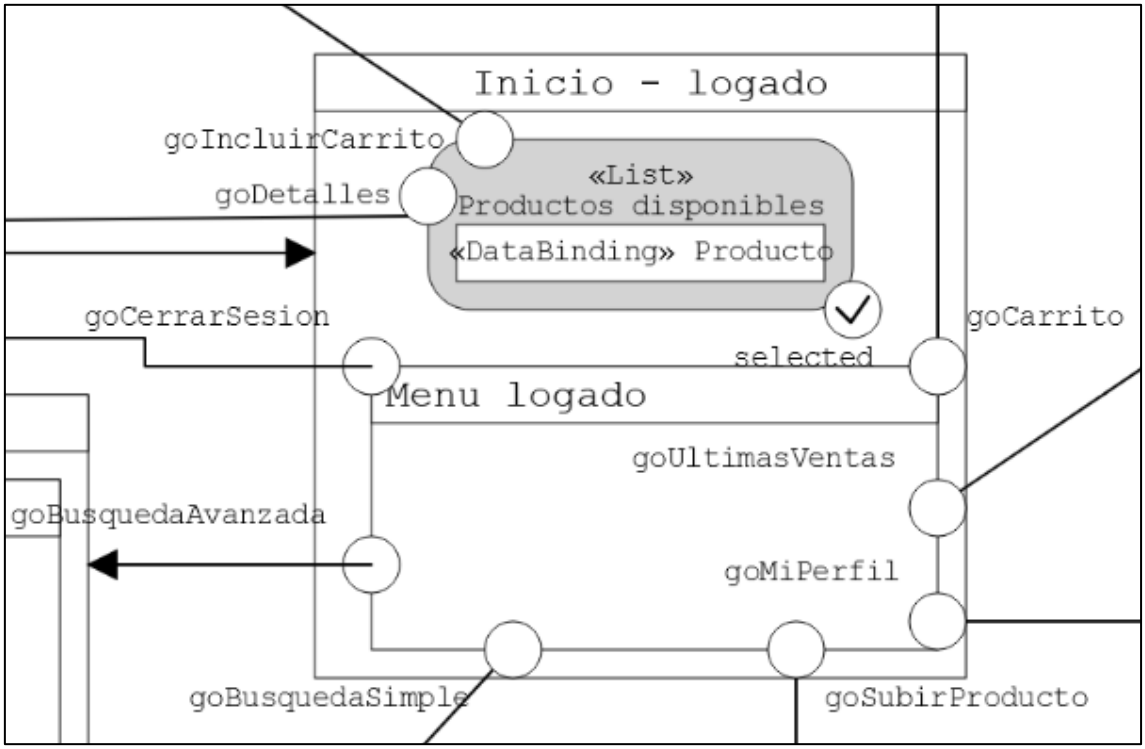


Fig. 4.16. IFML- Pantalla inicio-logado

La página *Inicio Logado* se corresponde con la página mostrada al usuario una vez ha iniciado sesión y está formada por el menú de navegación *Menú logado* y una lista de productos disponibles. A continuación, se muestra una captura de la página final de la aplicación, donde se observan estos elementos.

Fig. 4.17. Pantalla inicio-logado

Como se puede observar, en el modelado de esta página aparece el menú estándar una vez el usuario ha iniciado sesión en la aplicación, el cual contiene los posibles eventos goBusquedaSimple, goBusquedaAvanzada, goSubirProducto, goUltimasVentas, goMiperfil, goCarrito y goCerrarSesion que se corresponden con las opciones de navegación que aparecen en la interfaz final mostrada en la Fig. 4.17, donde algunas opciones aparecen textualmente en el menú y otras como iconos representativos de la acción a realizar.

Cada una de estas opciones lleva al usuario a otra pantalla de la aplicación, tal y como se puede observar en el diagrama completo de la Fig. 4.15.

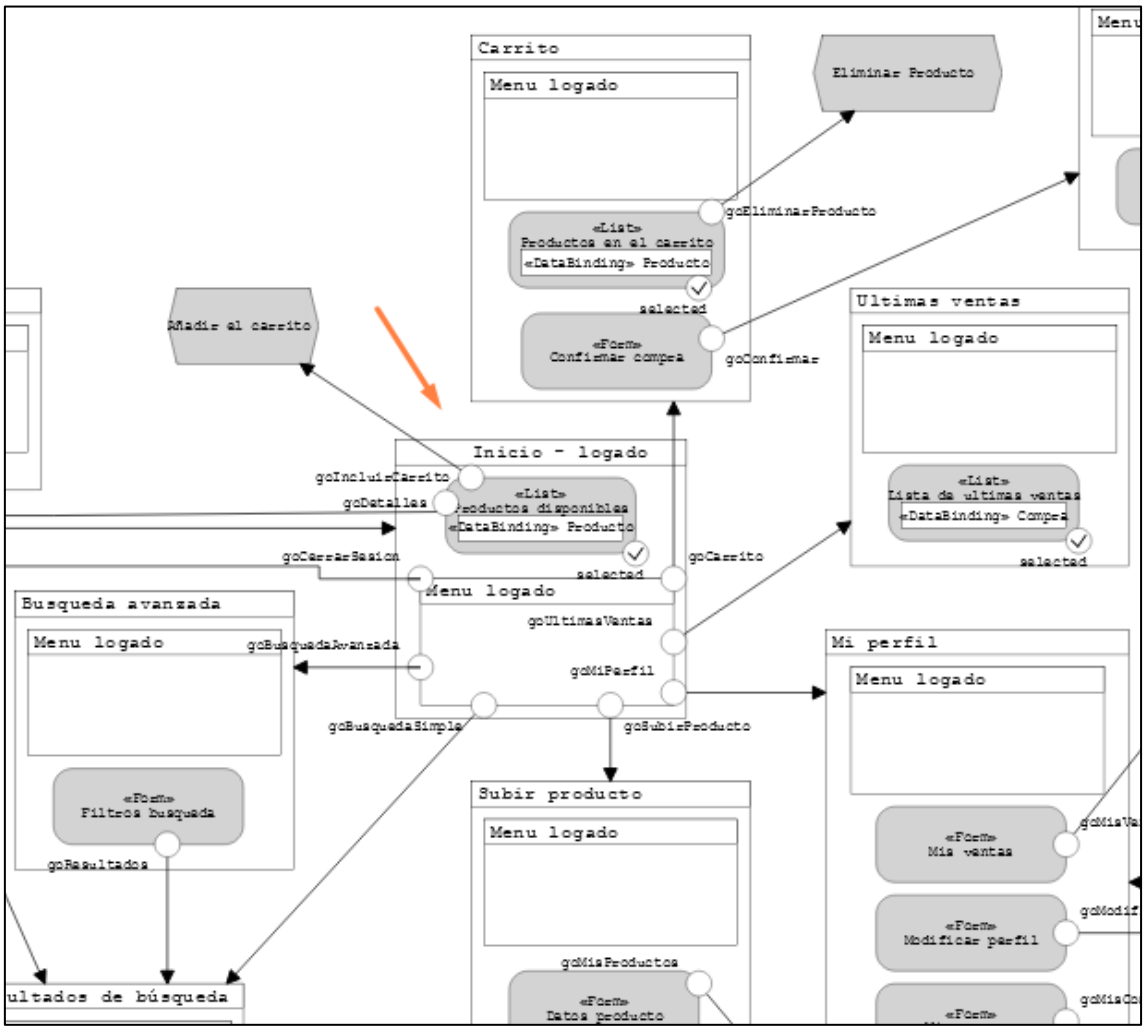


Fig. 4.18. Pantalla inicio-logado e interacciones

La siguiente imagen corresponde con la parte inferior de la misma página de inicio, donde se muestran una serie de productos sobre los que realizar las dos acciones señaladas en el modelado; mostrar los detalles del producto (goDetalles) y añadir el producto al carito (goInlcuirCarrito).

La acción goIncluirCarrito desencadena la acción de guardar ese producto dentro del carrito del usuario y se desarrolla en el *backend*, sin mostrar nada por pantalla, si bien la opción goDetalles, lleva al usuario a la página de detalles del producto, cambiando así la vista de la aplicación.

Con el objetivo de mostrar el resultado completo de la aplicación, en el ANEXO B se incluye una guía de usuario, que permite analizar la interfaz de usuario final del resto de pantallas.

4.5 Detalles de implementación

4.5.1 Servidor de aplicaciones

A continuación, se muestran una serie de imágenes que muestran la configuración realizada en el servidor Glassfish para la creación de la conexión del servidor a la base de datos MySQL. Para poder realizar esta configuración son necesarias las siguientes acciones:

- Tener creada la base de datos en MySQL.
- Tener cargado el conector MySQL para la plataforma Java en el entorno de desarrollo.
- Acceder al "Admin Console" del servidor Glassfish para realizar las siguientes acciones.

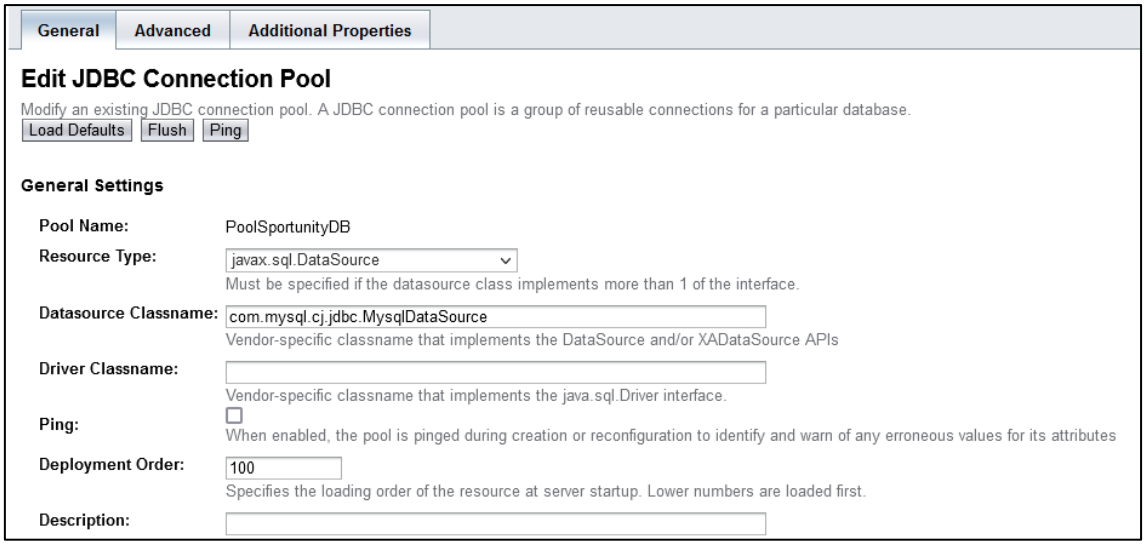


Fig. 4.19. Configuración del servidor Gassfish (I)

Esta imagen muestra la configuración del pool de conexiones a una base de datos de MyQL. Se debe escoger un nombre, el tipo de recurso, que debe de ser sql y el "Datasource Classname" que debe ser el mostrado en la imagen para la conexión a una base de datos MySQL

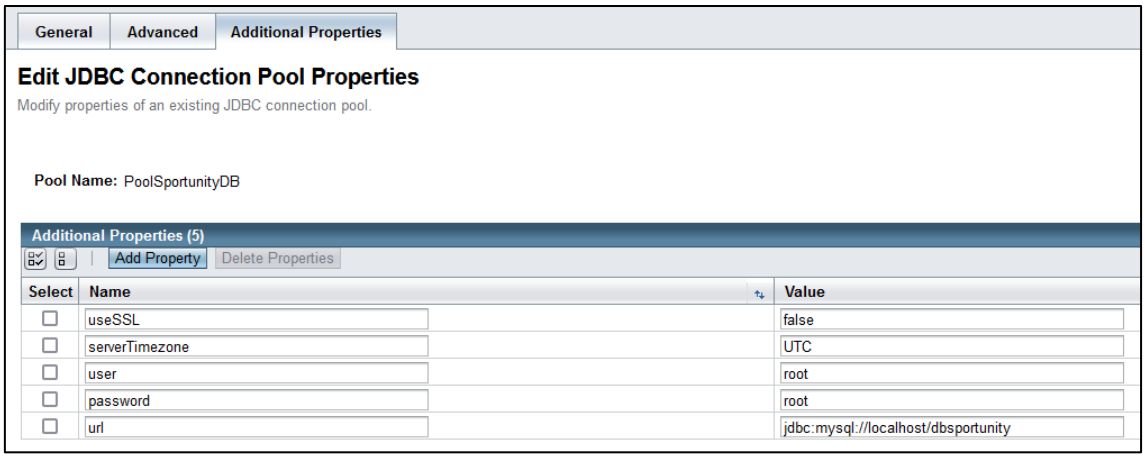


Fig. 4.20. Configuración del servidor Gassfish (II)

En la imagen previa se muestran las propiedades de ese pool de conexiones, entre las que destacan el usuario, la contraseña y la url de acceso a la base de datos. La propiedad useSSL se marca a false debido a que las conexiones no dispondrán de la seguridad SSL y serverTimezone indica el uso horario del servidor.

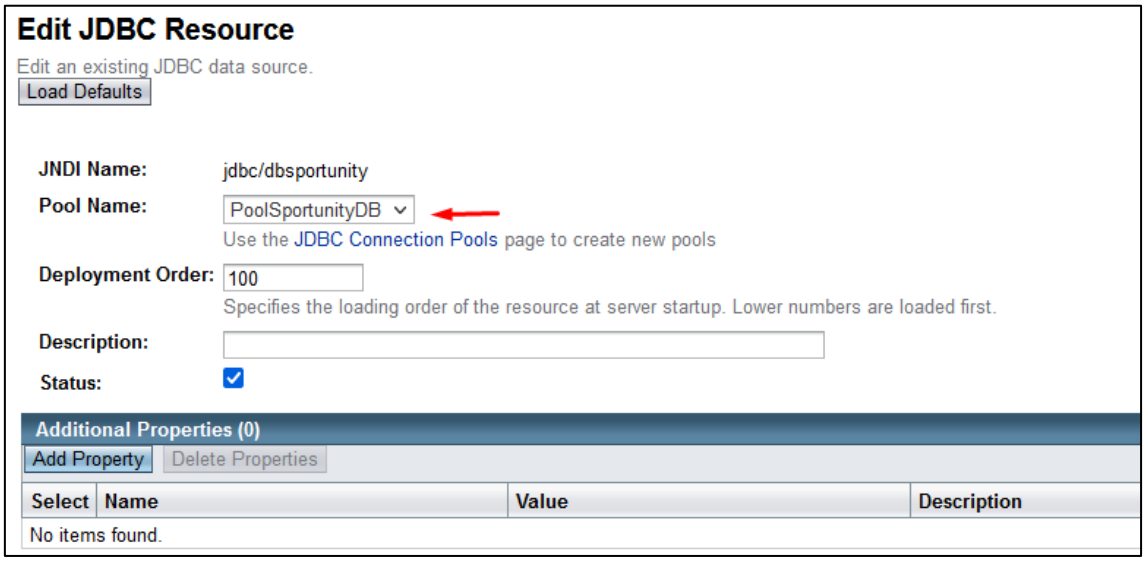


Fig. 4.21. Configuración del servidor Gassfish (III)

Por último, es necesario crear un recurso jdbc que use ese pool de conexiones creado. Se escoge un nombre para este recurso y se asocia con el pool creado anteriormente, de manera que la variable “Pool Name” de este recurso coincide con la misma variable definida en la configuración del pool de conexiones.

4.5.2 Postman y servicios REST

Con el objetivo de mostrar el correcto funcionamiento de estos microservicios sin necesidad de interactuar con la interfaz real del sistema, se ha utilizado el software **Postman**, que

permite realizar peticiones HTTP a las APIs definidas en los microservicios. A continuación, se muestra un ejemplo de su uso con una petición al *endpoint* que permite recuperar los datos de un producto y cuya definición se muestra a continuación:

```
@GetMapping(value = "/product/{id_product}")
public @ResponseBody ResponseEntity<?> getProducts(@PathVariable("id_product") int id)
```

Fig. 4.22. Ejemplo de *endpoint* para recuperar un producto en base a su id

En Postman, se define la petición GET a la URL http://localhost:10604/product/25 donde localhost hacer referencia al servidor local y 10604 hace referencia al puerto en el que el microservicio relativo al tratamiento de productos (*Sportunity-ms-Products*) está desplegado en este servidor local. La última parte de la URL hace referencia al endpoint definido, siendo 25 el id de producto que identifica de manera unívoca a un producto de la base de datos. La petición en Postman quedaría de la siguiente manera:

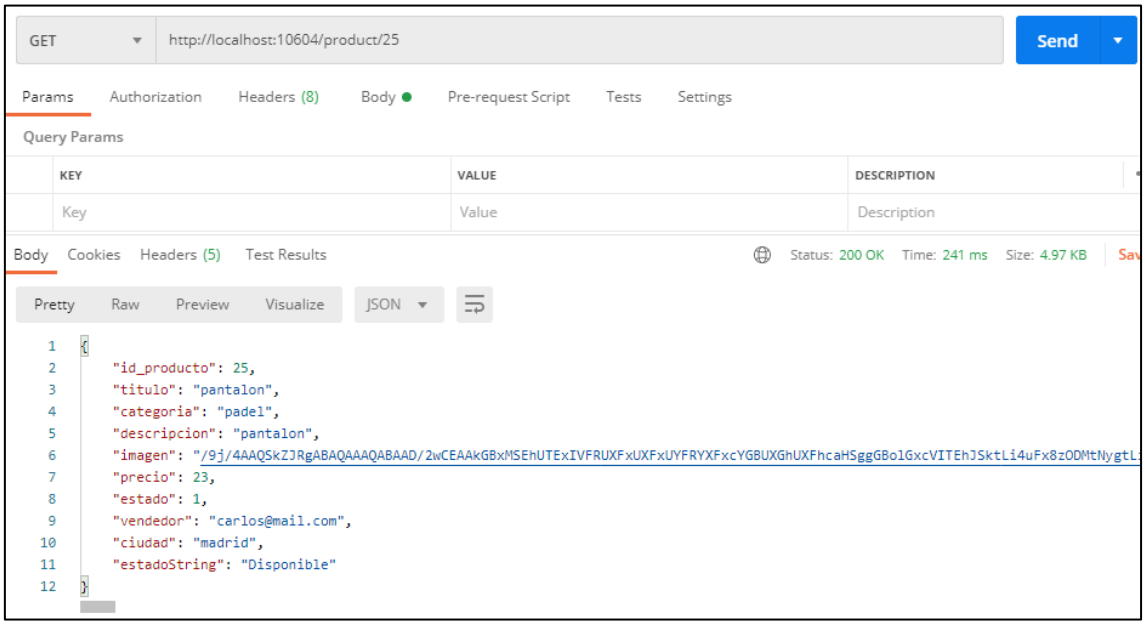


Fig. 4.23. Respuesta del servicio REST de recuperar un producto mediante Postman.

Se observa como en la parte superior de la imagen está definida la petición GET, de manera que al pulsar sobre el botón “Send” situado en la esquina superior derecha, se obtiene la respuesta en la que aparecen los datos del producto que persisten en la base de datos debido a esta correspondencia definida entre una tabla de la base de datos y un objeto Java. Como se puede observar, la respuesta contiene la información en formato JSON, debido a que es el que se ha definido en las propias peticiones dentro del código Java, si bien, estos servicios REST permiten también el intercambio de información en formato XML. Por último, cabe destacar que en la parte derecha de la imagen se observa cómo se recibe el “status” o código HTTP de la petición, indicando el código 200 que la petición se realizó correctamente.

A continuación, se muestra otro ejemplo utilizando el método POST para almacenar un nuevo usuario en la base de datos, utilizando el endpoint que se muestra a continuación:

```
@PostMapping(value = "/user/insert")
public @ResponseBody ResponseEntity<?> saveUser(@RequestBody Usuario user) {
    try {
        usuarioDAO.save(user);
        return new ResponseEntity<>(HttpStatus.OK);
    }catch(Exception e) {
        return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Fig. 4.24. Ejemplo de *endpoint* para insertar un usuario en la base de datos.

En Postman, se define la petición POST a la URL http://localhost:10605/user enviando en el cuerpo de la petición los datos del usuario a insertar en formato JSON, quedando de la siguiente manera:

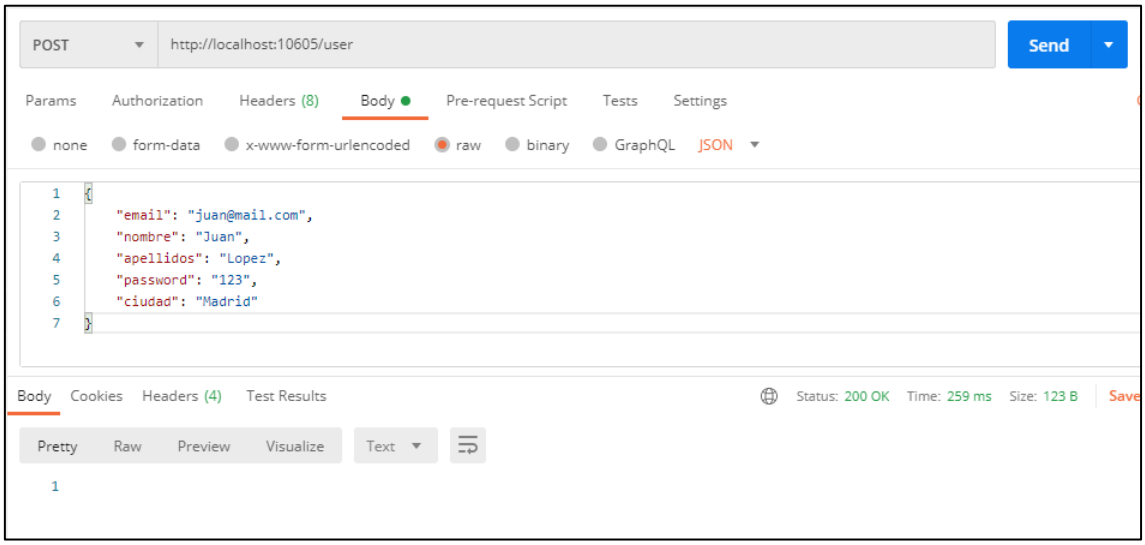


Fig. 4.25. Respuesta del servicio REST de insertar un usuario mediante Postman

Por otro lado, se observa el estado de la base de datos antes de la petición.

	nombre	apellidos	Email	password	ciudad
▶	Carlos	Salinero	carlos@mail.com	123	Madrid
*	NULL	NULL	NULL	NULL	NULL

Fig. 4.26 .Tabla usuarios (I)

Y después, donde queda registrado el nuevo usuario.

	nombre	apellidos	Email	password	ciudad
▶	Carlos	Salinero	carlos@mail.com	123	Madrid
	Juan	Lopez	juan@mail.com	123	Madrid
*	NULL	NULL	NULL	NULL	NULL

Fig. 4.27. Tabla usuarios (II)

Cabe destacar que la definición completa de las APIs de los microservicios se encuentra en la sección API de los microservicios.

4.5.2 Seguridad con JSON Web Token (JWT)

Para incrementar la seguridad de la aplicación, se ha hecho uso de la autenticación de las APIS mediante tokens de acceso denominados JSON Web Token. Se trata del estándar RFC7519 que permite añadir seguridad a las APIs de las aplicaciones y su funcionamiento es el explicado a continuación.

Un token JWT es una cadena de caracteres codificada en base64(glosario) y que consta de 3 partes: [44]

- **Header:** Contiene el algoritmo de cifrado utilizado para cifrar la firma o "signature".
- **Payload:** Contiene datos relativos al usuario al que se concede este token.
- **Signature:** Contiene el header y el payload en base64, concatenados y cifrados de la siguiente manera: HMACSHA512(codificación_base64(header)+"."+ codificación_base64(payload), clave_cifrado). HMACSHA512 hacer referencia a una función Hash cifrada con el algoritmo SHA de 512 bits y clave_cifrado es la clave utilizada para cifrar la *signature.*

A continuación, se muestra la utilización del sitio web https://jwt.io/ para la decodificación de un token generado en el proceso de autenticación de la aplicación:

Fig. 4.28. JWT decodificado (I).

En esta imagen se observa el token descompuesto en las 3 partes anteriormente mencionadas. Por un lado, la parte roja corresponde con el ***header***, que señala que el algoritmo de cifrado utilizado para cifrar la *signature* es HS512 (SHA512). Por otro lado, la parte morada corresponde con el ***payload***, que contiene información acerca de la procedencia del token (jti), del usuario autenticado (sub y authorities) y acerca de la concesión y expiración del token (iat y exp). Por último, la parte azul corresponde con la firma o *signature* del token y contiene la información mencionada anteriormente y que se muestra en la siguiente figura.

Fig. 4.29. JWT decodificado (II)

Una pregunta lógica acerca de este proceso de autenticación mediante token consiste en cuestionar **donde reside la seguridad del proceso**, si la información del token es fácilmente decodificable con una herramienta como este sitio web, y la **respuesta** reside en esta firma o ***signature***. El proceso de autenticación no comprueba únicamente que los datos del *payload* sean correctos, sino además, que la firma sea correcta, de manera que un token compuesto por el mismo *header y payload*, pero con una firma distinta se identifica como inválido e imposibilita la autenticación del usuario. Es por ello, que la clave de cifrado, debe mantenerse en secreto, ya que únicamente los tokens firmados con la clave original serán válidos. En este punto cabe mencionar que “mySecretKey” es la clave de cifrado utilizada en el propio código para firmar este token y que se expone en este documento con fines informativos, si bien está información debe mantenerse en secreto en la aplicación y nunca ser revelada.

Para comprender mejor está explicación, a continuación se expone dos tokens que contienen la misma información pero firmados con claves distintas.

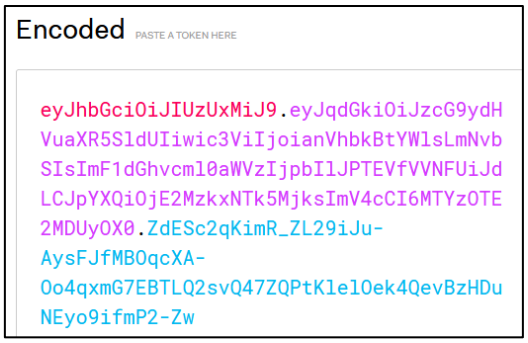


Fig. 4.30. JWT clave original

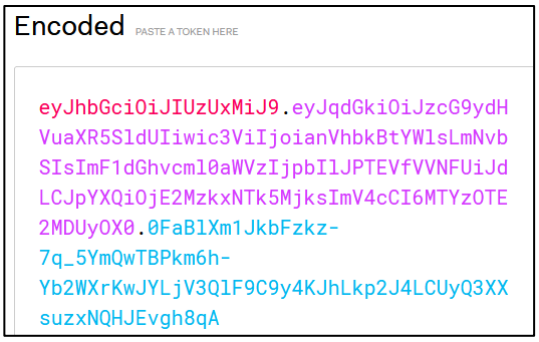


Fig. 4.31. JWT clave falsa

El segundo token se identificaría como inválido, una vez el proceso de autenticación trate de descifrar la *signatura* con la clave orginal "mySecretKey" y no lo consiga pues fue cifrado con otra. Destacar que el algoritmo de cifrado utilizado hace uso de la misma clave para cifrar y descifrar, por lo que se resalta una vez más, la importancia del secreto de esa clave.

Una vez se conoce la estructura de un token JWT, se procede a explicar el **funcionamiento completo del proceso de autenticación**, el cual queda resumido en la siguiente imagen.

Fig. 4.32. Funcionamiento de un token JWT. [45]

En primer lugar, se realiza una petición POST con el usuario y contraseña que introduce el usuario (1). Si ese usuario existe en la base de datos y su contraseña es correcta, se genera un token JWT cifrado con la clave secreta (2) y se devuelve en la respuesta de la petición POST (3). Para cualquier otra petición a las APIS definidas de la sección API de los microservicios se envía el token en la cabecera de la petición precedido de la palabra "Bearer", a modo de identificación del funcionamiento del token (4). Cabe destacar que algun *endpoint* de los

definidos en las APIS, no necesita autenticación, pues es usado en ciertas funciones de la aplicación que no requieren de un inicio de sesión.

Por último, una vez realizada una petición que requiere autenticación, el servidor comprueba que la firma del token es válida, que el token aún no ha expirado y que el usuario que envía ese token tiene acceso al recurso solicitado. A continuación, se ejemplifica esté proceso mediante Postman:

El usuario con email jorge@mail.com trata de iniciar sesión y se realiza la petición POST que se muestra a continuación, en la que se observa el token devuelto en la respuesta.

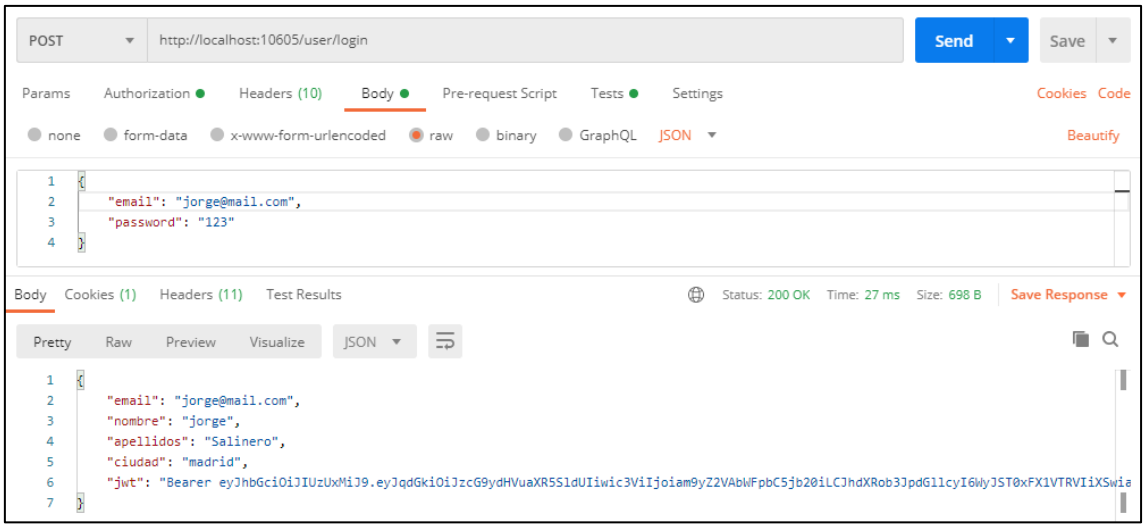


Fig. 4.33. Respuesta del endpoint de login.

Si este usuario desea modificar un producto del que es propietario, se realizaría la siguiente petición en la que se observa una respuesta JSON y un código HTTP 200 indicando un éxito en la petición.

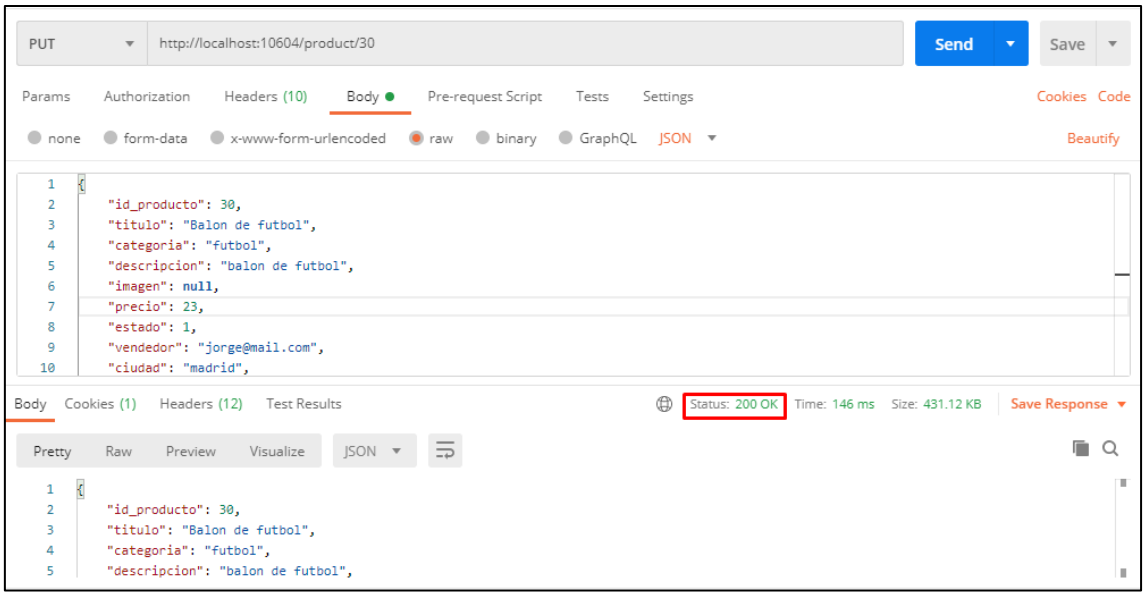


Fig. 4.34. Respuesta de endpoint con autenticación requerida.

En la sección *Headers* se incluye la cabecera "Authorization" con el token previamente generado:

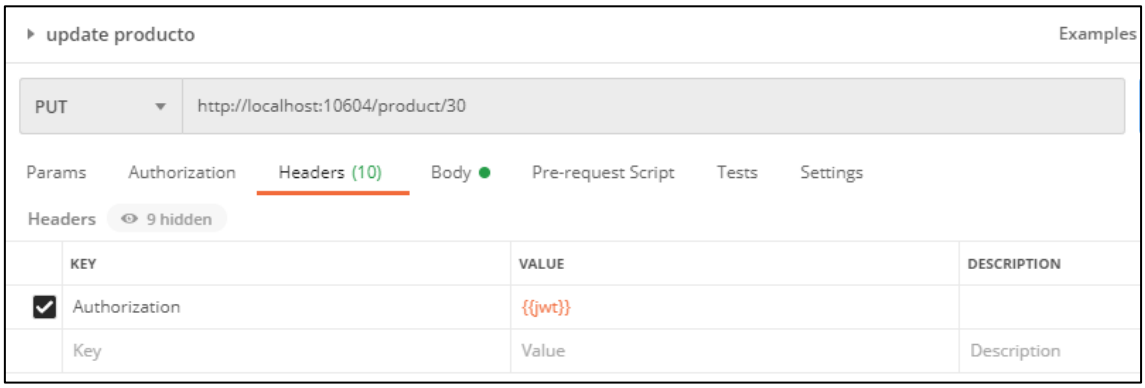


Fig. 4.35. Cabecera de una petición con autenticación requerida.

Si esta cabecera no se incluye o se incluye un token de otro usuario (un usuario no propietario del producto), el servidor devuelve una respuesta vacía con el código HTTP 403 Forbidden indicando que el usuario no está autorizado a usar ese recurso.

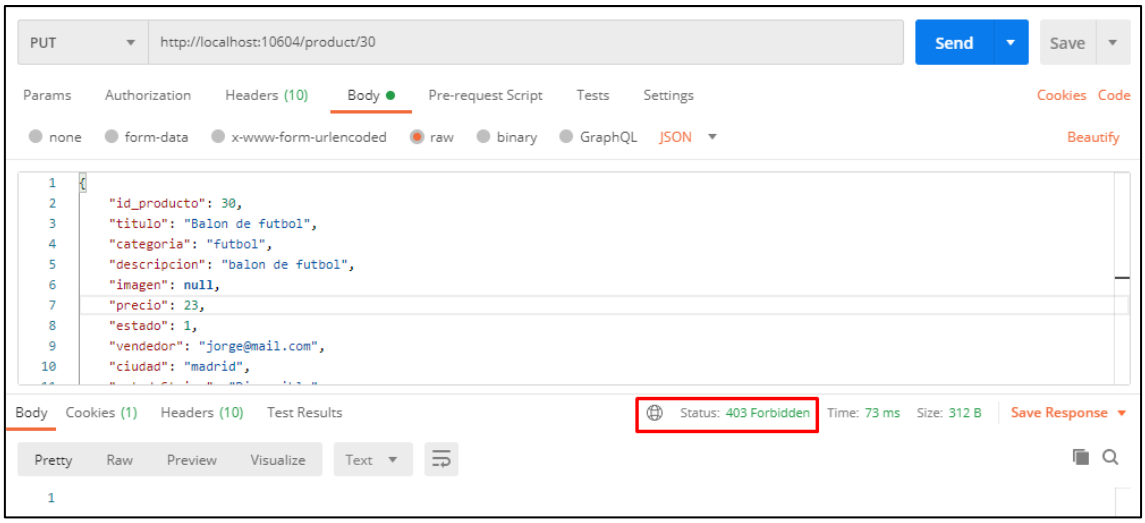


Fig. 4.36. Respuesta de un servicio con autenticación requerida y fallida.

4.6 Evaluación

En esta sección se va evaluar la adecuación de la solución desarrollada de acuerdo a los requisitos de sistema definidos en la sección Análisis de requerimientos. Para ello, se presentarán varios test que comprobarán varios de estos requisitos de manera simultánea, y se mostrará una matriz de trazabilidad entre las pruebas realizadas y los requisitos del sistema.

4.6.1 Tests

Los elementos que componen cada test son los siguientes:

- **ID**, que identifica a un test de manera unívoca y que sigue el patrón T-XX, donde XX corresponde a la secuencia numérica de test que comienza por 01.
- **Nombre**.
- **Requisitos cubiertos** y que quedan verificados si el test es exitoso.
- **Precondiciones**, que definen la situación en la que debe ejecutarse el test.
- **Descripción,** que señala las acciones en las que consiste el test.
- **Resultados** esperados de la ejecución del test
- **Estado**, que admite los valores de “Verificado” o “Fallido”.

La plantilla diseñada para cada test y que incluye los elementos anteriores es la siguiente:

<ID>	**<Nombre>**
Requisitos cubiertos	
Precondiciones	
Descripción	
Resultados	
Estado	

Tabla 4.1. Plantilla de tests.

Relacionados con el registro de usuario

T-01	**Registro e inicio de sesión**
Requisitos cubiertos	RS-F-01, RS-F-02
Precondiciones	Un usuario no registrado, debe acceder a la página principal de la aplicación.
Descripción	1. El usuario pulsará el botón de registro y aparecerá una pantalla con un formulario donde introducirá los datos pedidos. 2. Se redigirá al usuario a la pantalla de inicio de sesión y accederá con el email y contraseña que introdujo en el registro.
Resultados	El usuario visualiza un nuevo menú de navegación donde puede acceder a distintas funcionalidades exclusivas de este modo de navegación, como la visualización de su perfil.
Estado	Verificado.

Tabla 4.2. Test 01.

T-02	**Fallo en el inicio de sesión**
Requisitos cubiertos	RS-F-03
Precondiciones	Un usuario se encuentra en la pantalla de inicio de sesión de la aplicación.
Descripción	1. El usuario tratará de acceder con un email y contraseña incorrectos, bien porque el email no está registrado en la base de datos, bien porque la contraseña introducida no se corresponde con el email.
Resultados	El usuario visualiza una pantalla de error, donde se le indica que las credenciales introducidas no son las correctas.
Estado	Verificado.

Tabla 4.3. Test 02.

T-03	**Visualización y modificación de los datos del usuario**
Requisitos cubiertos	RS-F-04, RS-F-05
Precondiciones	Un usuario ha iniciado sesión en la aplicación.
Descripción	1. El usuario selecciona la opción de navegación “Mi perfil” y en la nueva pantalla, visualiza sus datos de registro. 2. En esta nueva pantalla selecciona la opción “modificar perfil” y visualiza un formulario con sus datos rellenos, donde modifica, por ejemplo, la ciudad de residencia.

	3. El usuario envía el formulario.
Resultados	Se redirige al usuario a la página “Mi perfil” donde se observa que la ciudad de residencia ha sido actualizada.
Estado	Verificado.

Tabla 4.4. Test 03.

T-04	**Cierre de sesión**
Requisitos cubiertos	RS-F-06
Precondiciones	Un usuario ha iniciado sesión en la aplicación.
Descripción	1. El usuario selecciona la opción “Log out”.
Resultados	Se redirige al usuario a la página de inicio de la aplicación, donde las opciones del menú de navegación se limitan a registro e inicio de sesión y las funcionalidades permitidas son la búsqueda rápida y la visualización de los detalles de un producto.
Estado	Verificado.

Tabla 4.5. Test 04.

T-05	**Baja de la aplicación**
Requisitos cubiertos	RS-F-07
Precondiciones	Un usuario ha iniciado sesión en la aplicación.
Descripción	1. El usuario selecciona la opción de navegación “Mi perfil”. 2. En esta nueva pantalla selecciona la opción “modificar perfil”. 3. En esta nueva pantalla selecciona la opción “Eliminar cuenta”.
Resultados	Se redirige al usuario a la página de inicio de la aplicación. Si se intenta acceder con las credenciales del usuario eliminado, salta un error de inicio de sesión.
Estado	Verificado.

Tabla 4.6. Test 05.

Relacionados con los productos

T-06	**Puesta a la venta de un producto**
Requisitos cubiertos	RS-F-08
Precondiciones	Un usuario ha iniciado sesión en la aplicación.
Descripción	El usuario selecciona la opción de navegación “Subir producto” y rellena un formulario con la información del producto.
Resultados	• Se redirige al usuario a la página donde se muestran todos sus productos que tiene a la venta. • Este producto es accesible desde las distintas opciones de búsqueda que ofrece la aplicación.
Estado	Verificado.

Tabla 4.7. Test 06.

T-07	**Consulta de productos propios y modificación de un producto**
Requisitos cubiertos	RS-F-09, RS-F-10
Precondiciones	Un usuario ha iniciado sesión en la aplicación.
Descripción	1. El usuario selecciona la opción de navegación “Mi perfil”

	2. En esta nueva pantalla, el usuario selecciona la opción “Mis productos”, donde visualiza todos los productos que tiene a la venta. 3. Al pulsar sobre la opción “Modificar producto” en el producto deseado, aparece un formulario con los datos del producto rellenos donde se modifica, por ejemplo, el precio.
Resultados	Se muestran los productos que el usuario tiene a la venta, con el cambio en el precio en el producto modificado.
Estado	Verificado.

Tabla 4.8. Test 07.

T-08	**Eliminar un producto**
Requisitos cubiertos	RS-F-11
Precondiciones	Un usuario ha iniciado sesión en la aplicación.
Descripción	1. El usuario selecciona la opción de navegación “Mi perfil” 2. En esta nueva pantalla, el usuario selecciona la opción “Mis productos”. 3. El usuario selecciona la opción “Eliminar producto” en el producto deseado.
Resultados	• Se muestran los productos que el usuario tiene a la venta, donde ya no se encuentra el producto eliminado. • Este producto ya no es accesible a través de las búsquedas.
Estado	Verificado.

Tabla 4.9. Test 08.

T-09	**Búsqueda simple y detalle del producto**
Requisitos cubiertos	RS-F-12, RS-F-14
Precondiciones	
Descripción	1. El usuario selecciona el icono de la lupa, en la esquina superior derecha de la interfaz y escribe la palabra “padel”.
Resultados	• Se muestran como resultados, los productos que contienen “padel” en su título o descripción. • Al pulsar sobre la opción “Detalles” de un producto, se visualiza una pantalla donde se detallan las características del producto.
Estado	Verificado.

Tabla 4.10. Test 09.

T-10	**Búsqueda avanzada**
Requisitos cubiertos	RS-F-13
Precondiciones	Un usuario ha iniciado sesión en la aplicación.
Descripción	1. El usuario selecciona la opción de navegación “Búsqueda avanzada” donde aparecen varios filtros a aplicar. 2. Selecciona, por ejemplo, la categoría “padel”, y envia el formulario.
Resultados	Se muestran como resultados, los productos que tienen “padel” como categoría definida.
Estado	Verificado.

Tabla 4.11. Test 10.

Nota: Se ha comprobado la funcionalidad de todos los filtros existentes (ciudad, categoría, título, precio desde y precio hasta, pero no se incluye una prueba por cada uno de ellos para no extender el plan de pruebas.)

Relacionados con el carrito

T-11	**Añadir productos al carrito**
Requisitos cubiertos	RS-F-15, RS-F-16, RS-F-17
Precondiciones	1. Un usuario ha iniciado sesión en la aplicación. 2. Realiza una búsqueda de algún tipo.
Descripción	El usuario pincha en el botón de añadir al carrito relativo a un producto determinado de los resultados de la búsqueda.
Resultados	Al pinchar en el botón de consultar el carrito, aparece el producto añadido, junto a los ya existentes si los hubiera.
Estado	Verificado.

Tabla 4.12. Test 11.

T-12	**Eliminar productos del carrito**
Requisitos cubiertos	RS-F-17, RS-F-18, RS-F-19
Precondiciones	1. Un usuario ha iniciado sesión en la aplicación. 2. Tiene algún producto en el carrito.
Descripción	El usuario pincha en el botón de eliminar del carrito relativo a un producto existente en el carrito.
Resultados	El producto deja de aparecer entre los existentes en el carrito.
Estado	Verificado.

Tabla 4.13. Test 12.

Relacionados con el proceso de compra

T-13	**Finalización de una compra**
Requisitos cubiertos	RS-F-20, RS-F-21, RS-F-22
Precondiciones	1. Un usuario ha iniciado sesión en la aplicación. 2. Tiene algún producto en el carrito.
Descripción	1. El usuario pincha en el botón de confirmar la compra. 2. Se le pide que introduzca la información relativa a su tarjeta de crédito y que acepte la compra de los productos del carrito.
Resultados	1. Al comprador le aparece una pantalla informándole del resumen de su compra. 2. El vendedor, al iniciar sesión, puede consultar en la sección de "Ultimas ventas" la venta realizada. Esta venta también aparecerá en la sección "Mi perfil"→"Mis ventas". 3. Los productos comprados, dejan de aparecer en las búsquedas de cualquier usuario y se eliminan además del carrito de cualquier usuario.
Estado	Verificado.

Tabla 4.14. Test 13.

Relacionados con el histórico de compra-venta

T-14	Consulta de las compras realizadas
Requisitos cubiertos	RS-F-23
Precondiciones	Un usuario ha iniciado sesión en la aplicación.
Descripción	El usuario accede a la sección “Mi perfil” →”Mis compras”
Resultados	Al usuario le aparece un histórico de sus compras realizadas.
Estado	Verificado.

Tabla 4.15. Test 14.

T-15	Consulta de las ventas realizadas
Requisitos cubiertos	RS-F-24
Precondiciones	Un usuario ha iniciado sesión en la aplicación.
Descripción	El usuario accede a la sección “Mi perfil” →”Mis ventas”
Resultados	Al usuario le aparece un histórico de sus ventas.
Estado	Verificado.

Tabla 4.16. Test 15.

Requisitos no funcionales

T-16	Navegación en la web
Requisitos cubiertos	RS-NF-01
Precondiciones	
Descripción	Un usuario que no ha iniciado sesión navega por la web.
Resultados	1. Este usuario solo encuentra disponibles las opciones de realizar una búsqueda rápida, y acceder a los detalles de un producto. 2. Tiene la opción, además, de registrarse o iniciar sesión.
Estado	Verificado.

Tabla 4.17. Test 16.

T-17	Navegadores compatibles
Requisitos cubiertos	RS-NF-02
Precondiciones	
Descripción	Una vez desplegada la aplicación de usuario Sportunity-User-App en un navegador Mozilla Firefox, se copia la url, y se pega en un navegador Google Chrome.
Resultados	La aplicación funciona de igual manera en ambos navegadores
Estado	Verificado.

Tabla 4.18. Test 17.

T-18	Información actualizada
Requisitos cubiertos	RS-NF-03
Precondiciones	Un usuario ha iniciado sesión en la aplicación.
Descripción	El usuario accede a “Mi perfil”→”Modificar perfil” y actualiza su ciudad de residencia.
Resultados	Al volver a consultar esta sección se muestra la información ya actualizada con la existente en la base de datos. Es necesario, por

	lo tanto, recargar una página del sitio web para observar los cambios realizados en ella.
Estado	Verificado.

Tabla 4.19. Test 18.

T-19	**Entorno de ejecución**
Requisitos cubiertos	RS-NF-04
Precondiciones	
Descripción	Todo el desarrollo del sistema, incluidas las pruebas y la ejecución de la aplicación, se lleva a cabo en el mismo ordenador personal.
Resultados	
Estado	Verificado.

Tabla 4.20. Test 19.

T-20	**Persistencia de los datos**
Requisitos cubiertos	RS-NF-05
Precondiciones	
Descripción	El modelo de persistencia utilizado es el de una base de datos relacional, tal y como se ha mencionado con anterioridad.
Resultados	
Estado	Verificado.

Tabla 4.21. Test 20.

4.6.2 Matriz de trazabilidad

A continuación, se muestra la matriz de trazabilidad entre los requisitos del sistema y los test propuestos, para asegurar la cobertura de la totalidad de los requerimientos. Con el objetivo de conseguir una visualización más clara, la matriz se ha dividido en dos.

	T-01	**T-02**	**T-03**	**T-04**	**T-05**	**T-06**	**T-07**	**T-08**	**T-09**	**T-10**
RS-F-01	■									
RS-F-02	■									
RS-F-03		■								
RS-F-04			■							
RS-F-05			■							
RS-F-06				■						
RS-F-07					■					
RS-F-08						■				
RS-F-09							■			
RS-F-10							■			
RS-F-11								■		
RS-F-12									■	
RS-F-13										■
RS-F-14									■	

Tabla 4.22. Matriz de trazabilidad test – requisitos de sistema (I).

	T-11	T-12	T-13	T-14	T-15	T-16	T-17	T18	T-19	T-20	T-21
RS-F-15											
RS-F-16											
RS-F-17											
RS-F-18											
RS-F-19											
RS-F-20											
RS-F-21											
RS-F-22											
RS-F-23											
RS-F-24											
RS-NF-01											
RS-NF-02											
RS-NF-03											
RS-NF-04											
RS-NF-05											

Tabla 4.23. Matriz de trazabilidad test – requisitos de sistema (II).

5. PLAN DE PROYECTO

En esta sección se muestra el proceso seguido en el desarrollo del proyecto. Se definirán las tareas realizadas, y se mostrará un diagrama de Gantt que ilustre el desarrollo de las tareas en una línea temporal, que permita observar la evolución de proyecto y la simultaneidad de tareas.

- **Planificación general:** Fase inicial donde se realiza una planificación temporal de las tareas a realizar, se crea el diagrama de GANTT, y se realiza un esquema con la estructura del documento.
- **Estado del arte:** Fase del proyecto durante la que se recopila información acerca de tecnologías útiles para el desarrollo del proyecto.
- **Planteamiento del problema:** Fase donde se describe con exhaustividad la necesidad existente en el mercado, junto con las soluciones existentes y el alcance provisional de proyecto.
- **Análisis de requerimientos:** En esta fase se distinguen dos partes consecutivas.
    - Requisitos de usuario.
    - Requisitos de sistema.
- **Diseño y arquitectura:** Fase determinada por la elección de las tecnologías a utilizar y el diseño de la arquitectura del sistema.
- **Implementación:** Fase centrada en el desarrollo del código fuente de la aplicación. Se diferencia una primera fase relativa al microservicio de usuarios, otra consecutiva relativa al microservicio de productos y un desarrollo simultáneo de los demás microservicios.
- **Evaluación en integración:** Consiste en verificar las funcionalidades desarrolladas y transcurre simultáneamente con el periodo de implementación, al realizar estas comprobaciones con cada funcionalidad añadida.
- **Evaluación final:** Consiste en la ejecución del plan de pruebas definido en la sección de Evaluación y que verifica de nuevo la totalidad de las funcionalidades.
- **Marco regulador:** Etapa dedicada a la investigación acerca de legislaciones, licencias y estándares aplicables al proyecto.
- **Entorno socio-económico:** Elaboración del presupuesto económico y análisis del impacto socio-económico en la sociedad.
- **Conclusiones:** Fase del proyecto dedicada a la reflexión acerca de las conclusiones finales.

- **Revisión general:** Fase final del proyecto dedicada a la revisión de la documentación desarrollada.

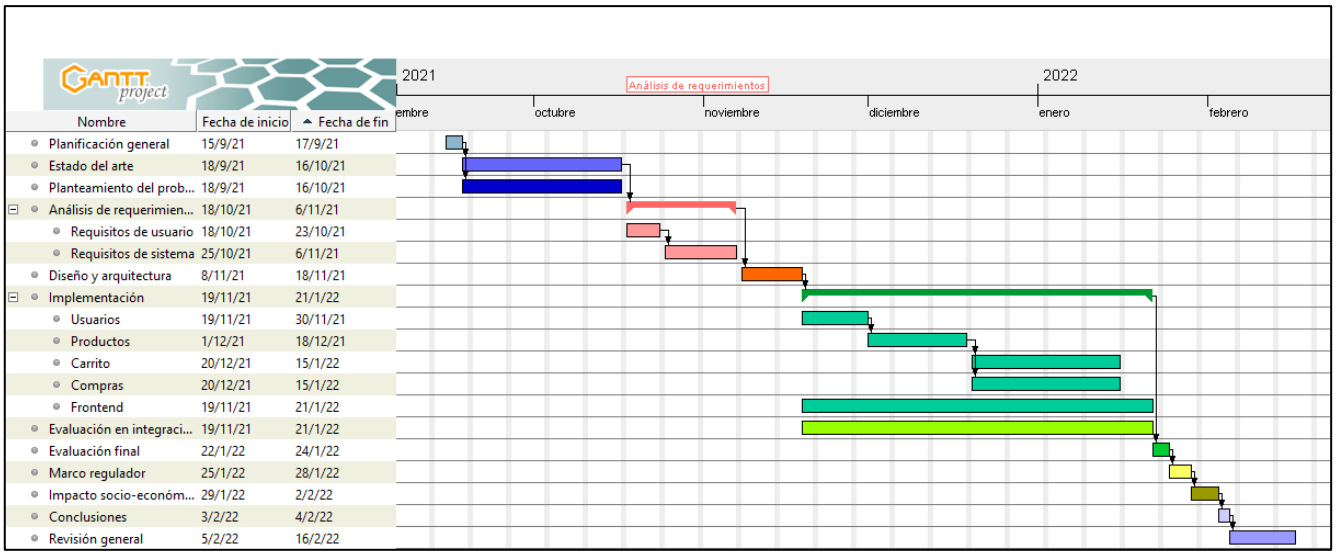


Fig. 4.18 Diagrama de Gantt

En este diagrama se puede observar la evolución temporal del proyecto así como la simultaneidad de tareas, con unas fechas comprendidas entre el 15 de septiembre de 2021 y el 15 de febrero de 2022, resultando un total de 5 meses de trabajo.

6. MARCO REGULADOR

6.1 Legislación aplicable

En esta sección se detallarán los aspectos más relevantes en cuanto a la legislación aplicable a la implementación señalada [46]. Cabe señalar que el proyecto desarrollado es una prueba de concepto y el portal no se encuentra accesible desde internet, por lo que la legislación aplicable únicamente tiene sentido si este proyecto finalmente se lanza y esa accesible desde internet.

- **Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales**. (LOPDGDD) [47] ,que sustituye a la anterior "Ley Orgánica de Protección de Datos (LOPD) y que tiene por finalidad proteger la privacidad e integridad del individuo de acuerdo con el artículo 18.4 de la Constitución Española. Su aplicación directa en este proyecto es el aviso a los nuevos usuarios de la necesidad de otorgar el consentimiento para el tratamiento de sus datos personales, ya que esto es necesario para el funcionamiento del sistema. Este aviso debe producirse en La interfaz de usuario, como paso previo a confirmar el registro en el sistema, de manera que el usuario sea libre de aceptar y poder registrarse o rechazar y continuar utilizando las funcionalidades que no requieren ese registro. Entre los derechos reconocidos a los usuarios, destacan los siguientes:
    - Derecho al acceso a la información cedida.
    - Derecho a la modificación de esa información.
    - Derecho de eliminación de sus datos del sistema de gestión.
    - Derecho a limitar el tratamiento ejercido sobre sus datos.
    - Derecho a la portabilidad o traslado de sus datos a otra entidad de gestión.
    - Derecho a la oposición a recibir cierta información de marketing.

- **Ley 9/2014, de 9 de mayo, General de Telecomunicaciones** [48], por la cual se permite cifrar cualquier tipo de información que se transmita por redes de comunicación y que se incluye en esta sección porque ayuda al cumplimiento de la anterior ley expuesta, al garantizar que la información no sea filtrada cuando se produzca una intercepción de comunicaciones.
- **Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico** (LSSI) [49], por la que se regulan las actividades económicas a

través de internet, y que señala la obligación, por parte del prestador del servicio a informar sobre:

  - Datos relativos al prestador del servicio, como NIF/CIF y el correo electrónico.
  - Condiciones del servicio, donde se debe señalar:
    - El aviso legal, que identifica al prestador del servicio.
    - La política de privacidad.
    - La política de cookies y la opción al usuario de otorgar el consentimiento.
  - El estado del producto y los métodos de pago.

  En definitiva, esta ley asegura que las operaciones comerciales virtuales sean equivalentes a las realizadas en un entorno físico.

- **Real Decreto Legislativo 1/2007, de 16 de noviembre**, que establece un marco de protección general para consumidores y usuarios.
- **Ley 7/1998, de 13 de abril, que regula las condiciones generales de contratación** y establece una serie de directrices en relación a las cláusulas abusivas.

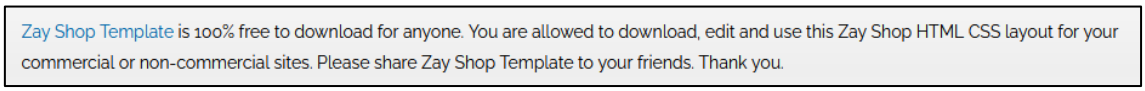
Como se ha comentado, este marco legal debe tenerse en cuenta si el sitio web se publica finalmente y es accesible desde internet. Sin embargo, al tratarse de una prueba de concepto, varios de estos aspectos no se han tratado en la implementación del proyecto, si bien es necesario tenerlos presentes.

6.2 Licencias y propiedad intelectual

El software desarrollado utiliza varias tecnologías de otros autores para su función completa, por lo que en esta sección se detallarán cuales son estas tecnologías y las licencias que aplican.

- **Spring Tool Suite.** Se trata del *framework* principal del proyecto sobre el que se ha desarrollado el código fuente del proyecto. Se trata de un IDE gratuito, de código abierto y que permite su uso para cualquier propósito comercial o no, al estar bajo la licencia Apache 2.0.
- **MySQL Workbench.** Para el desarrollo de este proyecto se ha hecho uso de la versión *Communnity*, la única que se ofrece de código abierto por parte del proveedor, y que se distribuye bajo la licencia GPL (*General Public License*), que permite su uso para fines comerciales.
- **Plantilla *frontend*.** Como se ha comentado anteriormente, para el desarrollo de este proyecto, se ha hecho uso de una plantilla para la parte visual de la web, la cual

muestra la interfaz al usuario. Esta plantilla se ha obtenido del sitio web https://templatemo.com/tm-559-zay-shop donde se señala claramente que esta plantilla puede ser usada y modificada con cualquier fin.



- **Glassfish**. El servidor de aplicaciones utilizado es de software gratuito, de código abierto y permite su uso para cualquier fin, comercial o no, al estar distribuido bajo las licencias GPL y CCDL.

Por último, cabe destacar que como medida de protección frente a futuros problemas legales o económicos, y como paso previo a una publicación del portal en internet, se registrará la marca *Sportunity* ante la Oficina Española de Patentes y Marcas (OEPM). Con esta medida se adquieren los siguientes derechos:

- Vender, alquilar, o decidir quién usa la marca.
- Tener el derecho exclusivo de uso sobre la marca *Sportunity.*
- Prohibir su uso.

Además, este registro evita que otro sujeto, con una marca registrada igual o similar, prohíba al autor de este proyecto el uso de *Sportunity* como marca, evitando así posibles sanciones económicas derivadas de ese uso.

6.3 Estándares técnicos

Al utilizar Java EE como tecnología principal para el desarrollo de este proyecto, se utilizan por defecto, una serie de estándares definidos, que componen esta tecnología Java y entre los que destacan:

- Java Persistence API, que estandariza el acceso a la base de datos con una correspondencia objeto-tabla.
- Java Server Pages, que estandariza la actualización de código HTML del lado servidor.
- Java Servlet Technology, que estandariza la respuesta a peticiones HTTP y permite implementar el patrón de arquitectura, ya comentado, MVC.

Cabe destacar también el uso del estándar RFC7519 debido al uso de la autenticación basada en JSON Web Tokens.

Por otro lado, cabe mencionar que los servicios implementados en los microservicios han sido desarrollados según las restricciones de un servicio REST y las buenas prácticas definidas para ellos, lo que facilita el mantenimiento y la posible escalabilidad del código [50].

Por último, y en relación al lenguaje de programación Java, el código fuente de la aplicación se ha desarrollado siguiendo el estándar y las buenas prácticas propias de este lenguaje [51].

7. ENTORNO SOCIO-ECONÓMICO

En esta sección, se va a detallar el coste monetario incurrido en la realización del proyecto en relación a recursos software, hardware, humanos y otro tipo de costes derivados. Además, se analizará el impacto económico y social, que se espera que este proyecto tenga en los usuarios finales, posibles socios o inversores y en la sociedad en general.

7.1 Presupuesto

El presente proyecto ha sido desarrollado en un plazo de 5 meses comprendidos entre finales de septiembre de 2021 y finales de febrero de 2022, en los cuales se ha trabajado un total de 125 días, teniendo en cuenta días de descanso. La dedicación media por día ha sido de 2,5 horas, obteniendo así un total de 312,5 horas trabajadas. La siguiente tabla resume esta información.

Dedicación del proyecto	
Meses	5
Días	125
Dedicación diaria media	2,5 horas
Dedicación total	312,5 horas

Tabla 7.1. Resumen temporal de dedicación.

Para la correcta imputación de los costes es importante tener en cuenta la amortización del activo en cuestión, cuyo cálculo se realiza de la siguiente manera:

$$Amortización = (precio\ del\ activo * \%\ de\ amortización\ anual) * \left(\frac{Nº\ de\ meses\ de\ uso}{Nº\ meses\ del\ año}\right)$$

Recursos software

A continuación, se resumen los costes derivados del software utilizado. En ese punto, cabe destacar que la amortización se establece en un 33% como nivel máximo anual según define la Agencia Tributaria española [52] para “Sistemas y programas informáticos”.

Coste del software		
Recurso software	**Precio de compra**	**Amortización**
Microsoft Windows 10 – Home Edition	131 €	18 €
Microsoft Office 365 – Estudiantes	0 €	0 €
IDE Spring Tool Suite	0 €	0 €
MySQL Workbench 8.0 CE	0 €	0 €
Servidor Glassfish	0 €	0 €
Plantilla *frontend*	0 €	0 €
Total		**18 €**

Tabla 7.2. Coste de recursos software.

Recursos hardware

En el caso de los recursos hardware, el nivel máximo amortizable en un periodo anual es de un 20%, según lo establecido para los "Equipos electrónicos".

Coste del hardware		
Recurso hardware	**Precio de compra**	**Amortización**
Ordenador personal	580	48,33333333
Monitor	90	7,5
Total		**55,83333333**

Tabla 7.3. Coste de recursos hardware.

Recursos humanos

El cálculo del coste de los recursos humanos empleados se ha realizado como si el proyecto hubiera sido realizado por una empresa, donde diferentes profesionales especializados en áreas complementarias hubieran contribuido a su desarrollo. Los roles involucrados, junto a su salario y el coste total de estos recursos se presentan en la siguiente tabla.

Recursos humanos				
Rol	Jefe de proyecto	Analista	Programador	Pruebas
Días trabajados	125	125	110	15
Horas diarias	1	1	2,5	2,5
Sueldo anual medio	40.000,00 €	35.000,00 €	30.000,00 €	30.000,00 €
Salario/hora	19,23 €	16,83 €	14,42 €	14,42 €
Total por trabajador	2.403,75 €	2.103,75 €	3.965,50 €	540,75 €
Total	**9.013,75 €**			

Tabla 7.4. Coste de recursos humanos.

Nota: Los salarios medios corresponden a diferentes fuentes consultadas, como la siguiente [53], donde se reflejan cantidades similares en todas ellas. La transformación de salario anual a salario por hora se ha realizado mediante un recurso web [54]. Al jefe de proyecto y al analista se les imputan menos horas diarias de trabajo al considerar este proyecto compatible con otros dada la carga de trabajo para estos tipos de perfil.

En cuanto a los días trabajados, se consideran los primeros quince únicamente para el jefe de proyecto y el analista, encargados del análisis y la viabilidad del proyecto. A partir de ese momento, y hasta el momento de la entrega, el programador está presente en todo el proyecto, si bien, el responsable de pruebas, lo está únicamente en días puntales para comprobar la funcionalidad desarrollada.

Costes indirectos

A continuación, se presentan los costes indirectos, en los que se ha incurrido durante los 5 meses de duración del proyecto.

Costes indirectos		
Descripcion	**Coste mensual**	**Total**
Luz	10,00 €	50,00 €
Agua	5,00 €	25,00 €
Internet	10,00 €	50,00 €
Total	**125,00 €**	

Tabla 7.5. Costes indirectos.

Resumen de los costes totales

En la siguiente tabla, se muestra un resumen de las partidas principales que conforman el coste total del proyecto, sin tener en cuenta el IVA aplicable, ni el margen de beneficio.

Resumen de costes	
Descripción	**Total**
Software	18 €
Hardware	55,8333333
RRHH	9.013,75 €
Costes indirectos	125,00 €
Coste del proyecto (sin IVA)	**9.212,58 €**

Tabla 7.6. Resumen de costes.

Beneficio y coste total

En esta última tabla, se presenta el coste total del proyecto una vez aplicado el IVA del 21% y un margen de beneficio del 20% que se considera razonable para un proyecto de desarrollo de software.

Costes totales	
Descripción	**Total**
Coste del proyecto (sin IVA)	9.212,58 €
Beneficio(20%)	1.842,52 €
Total sin IVA	11.055,1 €
IVA	2.321,57 €
Total	**13.376,67 €**

Tabla 7.7. Costes totales.

7.2 Impacto socio-económico

8. CONCLUSIONES

ANEXOS

ANEXO A: API de los microservicios

A continuación, se muestra la API de los microservicios, donde quedan reflejadas todas las llamadas a servicios REST que utiliza la aplicación *frontend.*

Sportunity-ms-Users

POST	`/user/login`	
Recibe	@RequestBody (RequestAuthentication request – objeto formado por user y password)	
Resultado	200	• Si encuentra al usuario devuelve un objeto UsuarioAutenticadoDTO con el token y los datos del usuario. • Si el usuario y contraseña no concuerdan devuelve un mensaje informativo.
	500	Error del servidor en la autenticación.

Tabla A.1. Servicio 1 - Sportunity-ms-Users.

GET	`/user/{email}`	
Recibe	@PathVariable (String email)	
Resultado	200	• Devuelve el usuario asociado al email en un objeto de tipo usuarioDTO. • Si no encuentra el usuario devuelve un mensaje informativo en un objeto ErrorDTO.
	403	El usuario que solicita la llamada no el mismo del que se pretende obtener los datos
	500	Error en el servidor.

Tabla A.2. Servicio 2 - Sportunity-ms-Users.

GET	`/user/exists/{email}`	
Recibe	@PathVariable (String email)	
Resultado	200	• Devuelve el email de un usuario si lo encuentra en un objeto usuarioEncontradoDTO. • Si no encuentra el usuario devuelve un mensaje informativo en un objeto ErrorDTO.
	500	Error en el servidor.

Tabla A.3. Servicio 3 - Sportunity-ms-Users.

POST	`/user`	
Recibe	@RequestBody (Usuario usuario)	
Resultado	200	Inserta un usuario.
	500	Error en el servidor.

Tabla A.4. Servicio 4 - Sportunity-ms-Users.

DELETE	`/user/{email}`

Recibe	@PathVariable (String email)	
Resultado	200	• Elimina al usuario con email igual al pasado por parámetro. • Si el usuario no existe devuelve un mensaje informativo en un objeto ErrorDTO.
	403	El usuario que solicita la llamada no el mismo que se pretende eliminar
	500	Error en el servidor.

Tabla A.5. Servicio 5 - Sportunity-ms-Users.

PUT	`/user/{email}`	
Recibe	@PathVariable (String email) @RequestBody (Usuario usuario)	
Resultado	200	• Actualiza los datos del usuario con email igual al pasado por parámetro, con los datos del usuario pasado en el cuerpo de la petición. • Si el usuario no existe devuelve un mensaje informativo en un objeto ErrorDTO.
	403	• El usuario que solicita la llamada no el mismo que se pretende actualizar.
	500	Error en el servidor.

Tabla A.6. Servicio 6 - Sportunity-ms-Users.

Sportunity-ms-Products

GET	`/product/{id_product}`	
Recibe	@PathVariable (int `id_product`)	
Resultado	200	Devuelve el producto asociado a ese id, en un objeto Producto.
	500	Error en el servidor.

Tabla A.7. Servicio 1 - Sportunity-ms-Products.

PUT	`/product/updateState/{id_product}`	
Recibe	@PathVariable (int `id_product`)	
Resultado	200	Actualiza el estado a vendido (definido con un 0) al producto con id igual al pasado por parámetro.
	403	El usuario que solicita la llamada no ha iniciado sesión.
	500	Error en el servidor.

Tabla A.8. Servicio 2 - Sportunity-ms-Products.

GET	`/product/available`	
Recibe		
Resultado	200	Devuelve una lista de tipo Producto con todos los productos disponibles (no vendidos).
	500	Error en el servidor.

Tabla A.9. Servicio 3 - Sportunity-ms-Products.

GET	`/product`

Recibe	@RequestParam (String email)	
Resultado	200	Devuelve una lista de tipo Producto con los productos que un usuario tiene a la venta.
	403	El usuario que solicita la llamada no es el mismo del que se pretenden recuperar los productos
	500	Error en el servidor.

Tabla A.10. Servicio 4 - Sportunity-ms-Products.

GET	`/product/quickSearch`	
Recibe	@RequestParam(String entrada)	
Resultado	200	Devuelve una listade tipo Producto con los productos que contengan el texto de la variable “entrada” en el título o en la descripción.
	500	Error en el servidor.

Tabla A.11. Servicio 5 - Sportunity-ms-Products.

GET	`/product/advancedSearch`	
Recibe	@RequestParam (String ciudad), @RequestParam (String categoría), @RequestParam (String título), @RequestParam (int precio_desde), @RequestParam (int precio_hasta)	
Resultado	200	Devuelve una lista de tipo Producto con los productos que cumplan los criterios establecidos.
	403	El usuario que solicita la llamada no ha iniciado sesión
	500	Error en el servidor.

Tabla A.12. Servicio 6 - Sportunity-ms-Products.

POST	`/product`	
Recibe	@RequestBody (Producto producto)	
Resultado	200	Inserta un producto.
	403	El usuario que solicita la llamada no coincide con el vendedor del producto a insertar
	500	Error en el servidor.

Tabla A.13. Servicio 7 - Sportunity-ms-Products.

DELETE	`/product/{id_product}`	
Recibe	@PathVariable (int id_producto)	
Resultado	200	• Elimina el producto con id igual al pasado por parámetro. • Si no encuentra el producto devuelve un mensaje informativo en un objeto ErrorDTO.
	403	El usuario que solicita la llamada no es el propietario del producto que se desea eliminar.
	500	Error en el servidor.

Tabla A.14. Servicio 8 - Sportunity-ms-Products.

PUT	`/93roducto/{id_product}`

Recibe	@PathVariable(int id_producto), @RequestBody(Producto producto)	
Resultado	200	• Actualiza los datos del producto con id igual al pasado por parámetro, con los datos el producto pasado en el cuerpo de la petición. • Si el producto no existe devuelve un mensaje informativo en un objeto ErrorDTO.
	403	El usuario que solicita la llamada no es el propietario del producto que se desea modificar
	500	Error en el servidor.

Tabla A.15. Servicio 9 - Sportunity-ms-Products.

Sportunity-ms-Compras_Carrito

DELETE	`/carrito`	
Recibe	@RequestParam (int id_product)	
Resultado	200	Elimina de todos los carritos, el producto con id igual al pasado por parámetro.
	403	El usuario que solicita la llamada no ha iniciado sesión.
	500	Error en el servidor.

Tabla A.16. Servicio 1 - Sportunity-ms-Compras-carrito.

DELETE	`/carrito/{email}/{id_producto}`	
Recibe	@PathVariable (String email), @PathVariable (int id_product)	
Resultado	200	• Elimina del carrito del usuario con email igual al pasado por parámetro, el producto con id igual al pasado por parámetro. • Si el producto no existe en el carrito de ese usuario, devuelve un mensaje informativo en un objeto ErrorDTO.
	403	El usuario que solicita la llamada no es el propietario del carrito.
	500	Error en el servidor.

Tabla A.17. Servicio 2 - Sportunity-ms-Compras-carrito.

GET	`/carrito`	
Recibe	@RequestParam (String email)	
Resultado	200	Recupera todos los productos del carrito de un usuario en una lista de tipo TuplaCarrito.
	403	El usuario que solicita la llamada no es el propietario del carrito.
	500	Error en el servidor.

Tabla A.18. Servicio 3 - Sportunity-ms-Compras-carrito.

POST	`/carrito`

Recibe	@RequestBody (TuplaCarrito tupla)	
Resultado	200	Inserta un nuevo producto en el carrito de un usuario. Para ello, se inserta el objeto tuplaCarrito, que es el que se mapea con la tabla carritos de la base de datos.
	403	El usuario que solicita la llamada no es el propietario del carrito.
	500	Error en el servidor.

Tabla A.19. Servicio 4 - Sportunity-ms-Compras-carrito.

POST	`/compra`	
Recibe	@RequestBody (Compra compra)	
Resultado	200	Inserta una compra.
	403	El usuario que solicita la llamada no es el comprador de la transacción.
	500	Error en el servidor.

Tabla A.20. Servicio 5 - Sportunity-ms-Compras-carrito.

GET	`/compra`	
Recibe	@RequestParam (String email)	
Resultado	200	Recupera una lista de tipo Compra con las compras del usuario.
	403	El usuario que solicita la llamada no es el comprador de la transacción.
	500	Error en el servidor.

Tabla A.21. Servicio 6 - Sportunity-ms-Compras-carrito.

GET	`/venta`	
Recibe	@RequestParam (String email)	
Resultado	200	Recupera una lista de tipo Compra con las ventas del usuario.
	403	El usuario que solicita la llamada no es el vendedor de la transacción.
	500	Error en el servidor.

Tabla A.22. Servicio 7 - Sportunity-ms-Compras-carrito.

GET	`/banking`	
Recibe	@RequestParam (String numtarjeta), @RequestParam (String cvv), @RequestParam (String year_caducidad), @RequestParam (String mes_caducidad)	
Resultado	200	Comprueba correctamente la validez de la tarjeta de crédito.
	403	El usuario que solicita la llamada no ha iniciado sesión.
	402	Tarjeta de crédito no válida.

Tabla A.23. Servicio 8 - Sportunity-ms-Compras-carrito.

Sportunity-ms-Messages

POST	`/message`	
Recibe	@RequestBody(Mensaje mensaje)	
Resultado	200	Inserta un mensaje.
	403	El usuario que solicita la llamada no es el emisor del del mensaje.
	500	Error en el servidor.

Tabla A.24. Servicio 1 - Sportunity-ms-Messages.

DELETE	`/message`	
Recibe		
Resultado	200	Elimina los mensajes leídos
	403	El usuario que solicita la llamada no ha iniciado sesión en la aplicación y esta acción se realiza una vez se lee un mensaje.
	500	Error en el servidor.

Tabla A.25. Servicio 2 - Sportunity-ms-Messages.

PUT	`/message`	
Recibe	@PathVariable int idmensaje	
Resultado	200	Actualiza un mensaje a leído.
	403	El usuario que solicita la llamada no es el destinatario del mensaje.
	500	Error en el servidor.

Tabla A.26. Servicio 3 - Sportunity-ms-Messages.

GET	`/message/receipt`	
Recibe	@RequestParam String email	
Resultado	200	Recupera el mensaje de confirmación de compra en un objeto de tipo Mensaje.
	403	El usuario que solicita la llamada no es el destinatario del del mensaje.
	500	Error en el servidor.

Tabla A.27. Servicio 4 - Sportunity-ms-Messages.

GET	`/message/confirmation`	
Recibe	@RequestParam String destinatario	
Resultado	200	Recupera un mensaje asociado a una venta reciente en un objeto de tipo Mensaje.
	403	El usuario que solicita la llamada no es el emisor del del mensaje.
	500	Error en el servidor.

Tabla A.28. Servicio 5 - Sportunity-ms-Messages.

Cabe destacar que la formación de las URLs ha sido realizada de acuerdo a las buenas prácticas de los servicios REST, donde se han tenido en cuenta las siguientes consideraciones:

- Misma raíz de la URL para los *endpoints* de un microservicio.

- Uso de @PathVariable para aquellos *endpoints* donde se pase un atributo que identifica al objeto en cuestión. Por ejemplo, en la petición GET `/user/{email}` se utiliza este modelo de URL porque el email identifica a un usuario. La URL quedaría de la siguiente manera: `/user/carlos@mail.com`
- Uso de @RequestParam en aquellos casos donde se realice un filtrado que no sea por clave primaria, como en el caso de la petición DELETE `/carrito`, donde se pasa por parámetro un email, al tratarse de un elemento no identificativo de la tabla carritos. La URL quedaría de la siguiente manera: `/carrito?email=carlos@mail.com`

ANEXO B: Guía de usuario

En este anexo se detalla el uso de la aplicación, con el objetivo de observar un uso cotidiano de la misma, al mismo tiempo que permite observar el resultado final desarrollado.

Registro e inicio de sesión

La página de inicio de la aplicación es la presentada a continuación, donde se observa un menú que permite registrarse o iniciar sesión, además de la opción de realizar una búsqueda rápida.

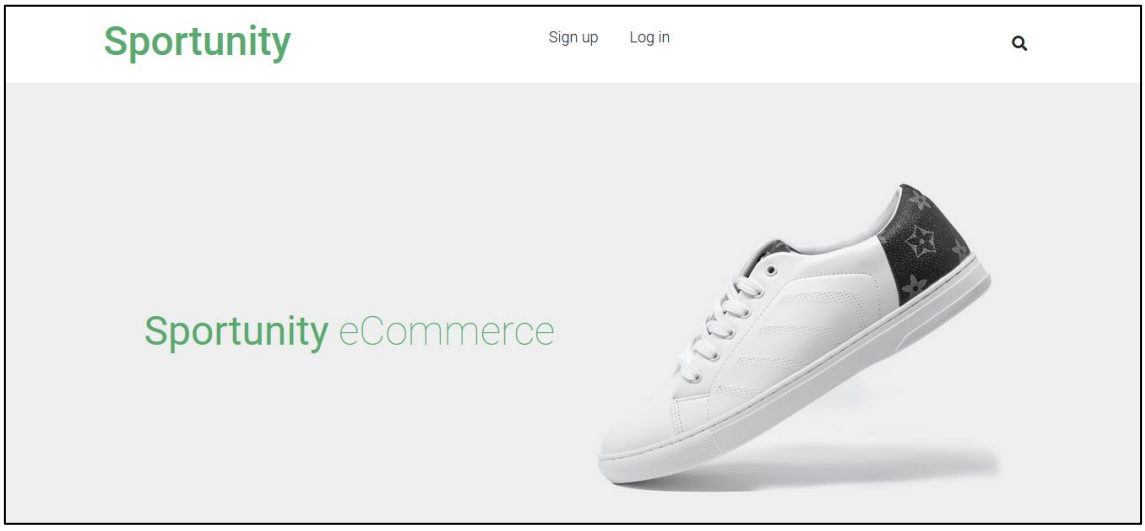


Fig. A.1. Página de inicio.

Se supone que se quiere realizar una búsqueda rápida de una pala de padel, para lo cual se pulsa sobre el icono de la lupa y se introduce la palabra “pala”.

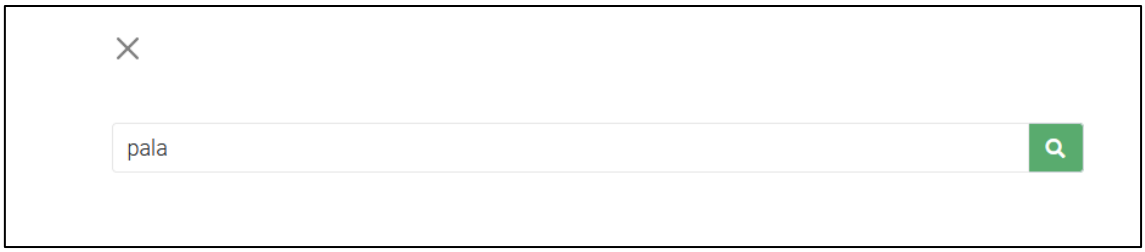


Fig. A.2. Buscador simple.

A continuación, se muestran al usuario los resultados obtenidos de su búsqueda.

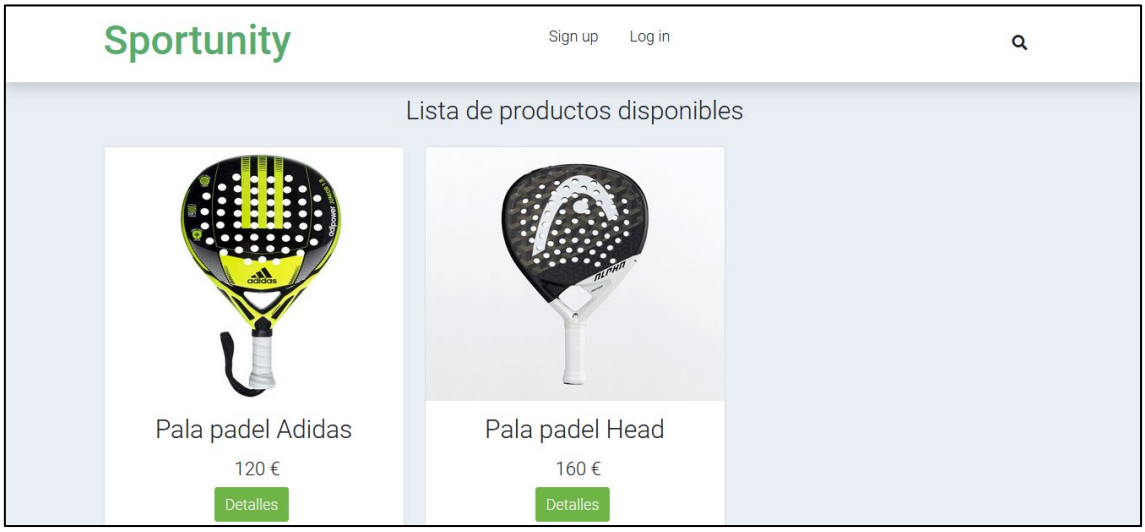


Fig. A.3. Resultados de una búsqueda sin iniciar sesión.

El usuario puede pulsar el botón de *detalles* para acceder a una descripción más exhaustiva del producto.

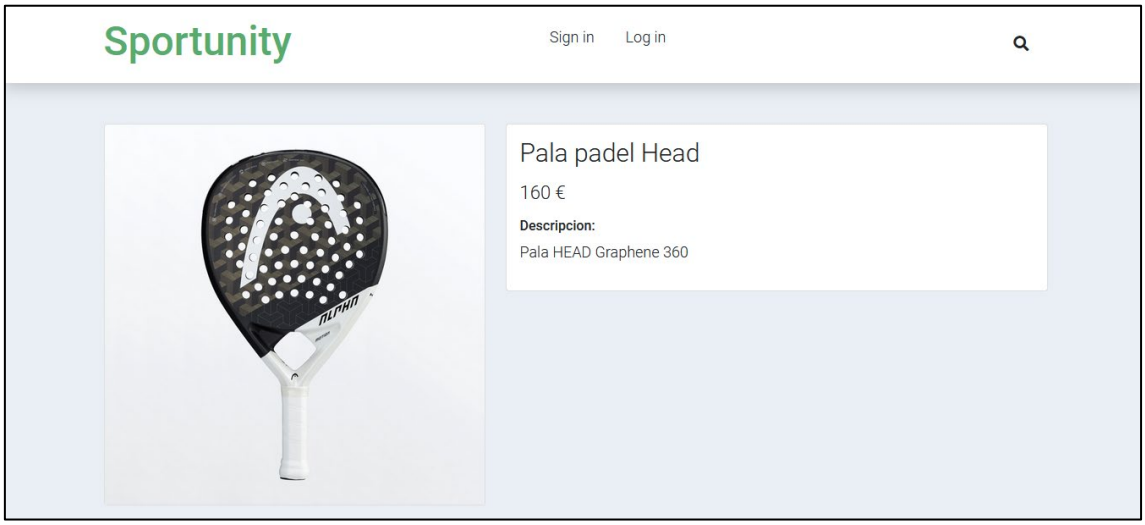


Fig. A.4. Detalles de un producto.

Se supone ahora, que el usuario está interesado en comprar uno de estos productos y para ello debe registrarse, pulsando la opción “Sign up” existente en el menú de cualquier pantalla si no se ha iniciado sesión todavía y que muestra al usuario el siguiente formulario de registro.

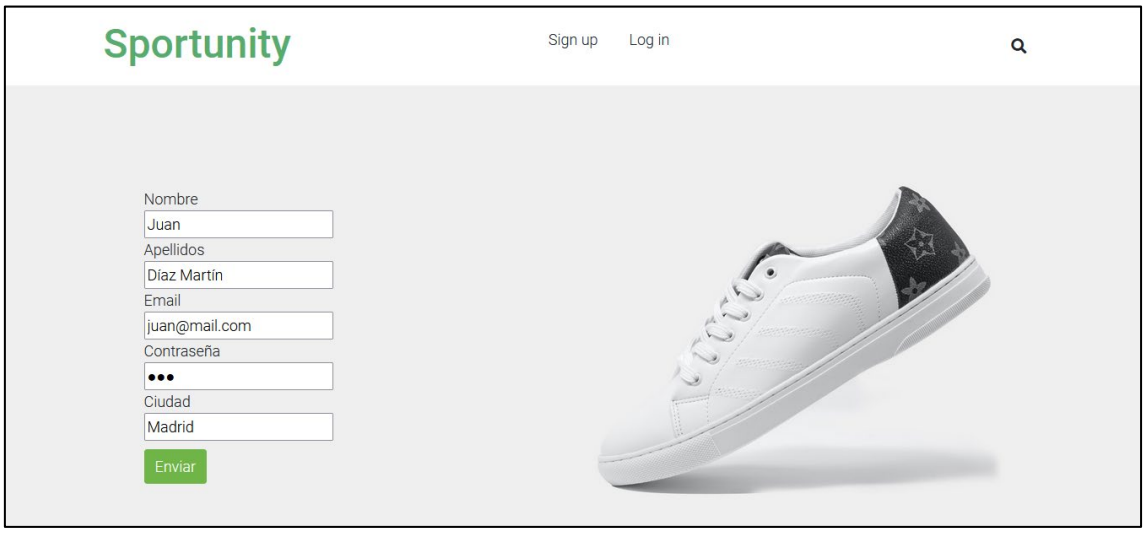


Fig. A.5. Formulario de registro.

Una vez se envía el formulario, los datos son almacenados en la base de datos y se redirige al usuario a la pantalla de *login*.

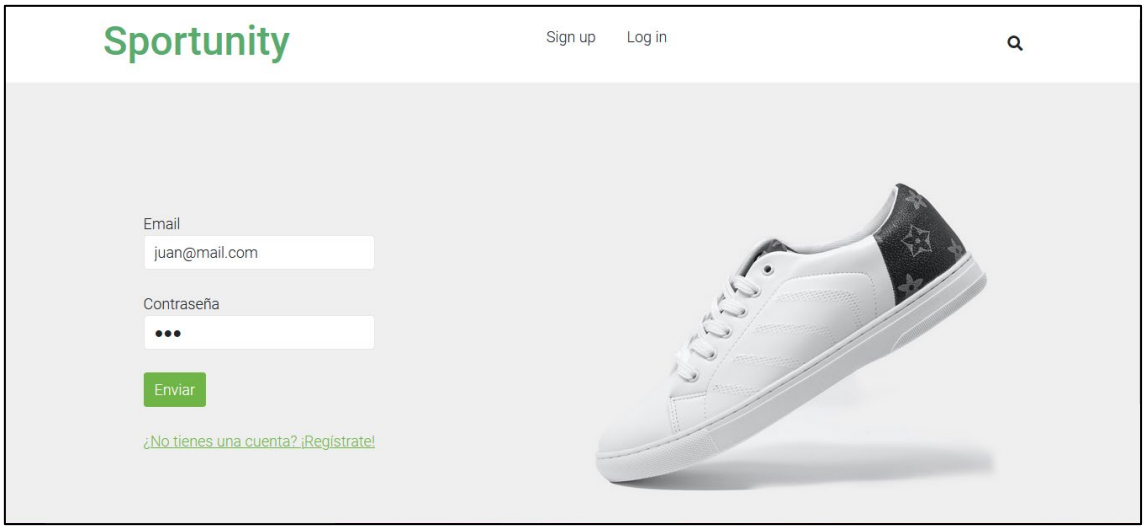


Fig. A.5. Formulario de inicio de sesión.

Cuando el usuario introduce su email y contraseña para acceder existen dos posibles escenarios:

- El email no está registrado, o el email y la contraseña no concuerdan, para lo cual se muestra el siguiente mensaje informativo:

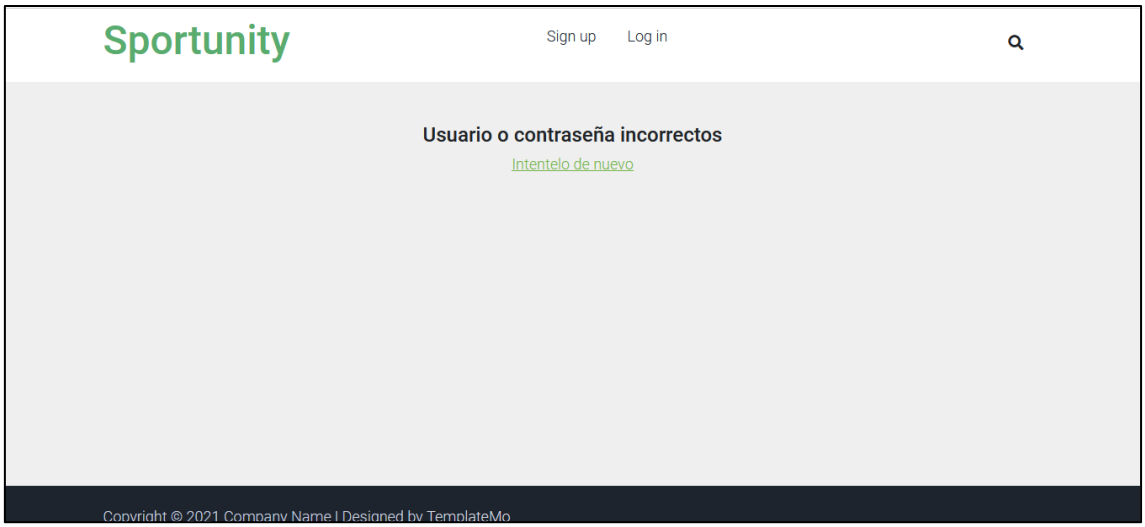


Fig. A.7. Página de error en el inicio de siesión.

- El email y la contraseña concuerdan y se redirige al usuario a la pantalla principal de la aplicación una vez se ha iniciado sesión. En ella se muestra un formulario fácilmente accesible para realizar una búsqueda avanzada, y deslizando el ratón hacía abajo aparecen varios de los productos existentes en la aplicación.

Fig. A.8. Página principal tras iniciar de sesión (I).

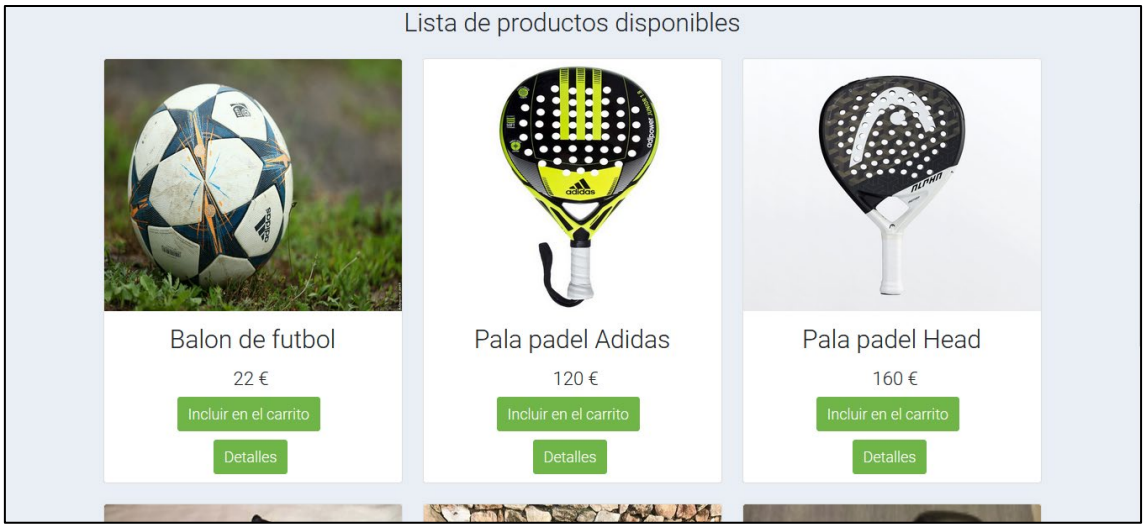


Fig. A.8. Página principal tras iniciar de sesión (II).

Cambio de los datos de registro

El usuario puede modificar sus datos personales seleccionando las opciones *Mi perfil* → *Modificar perfil*, donde se mostrará una pantalla con el mismo formulario de registro pero con los datos rellenos. Al enviar el formulario, se redigirá al usuario a la página principal de la aplicación.

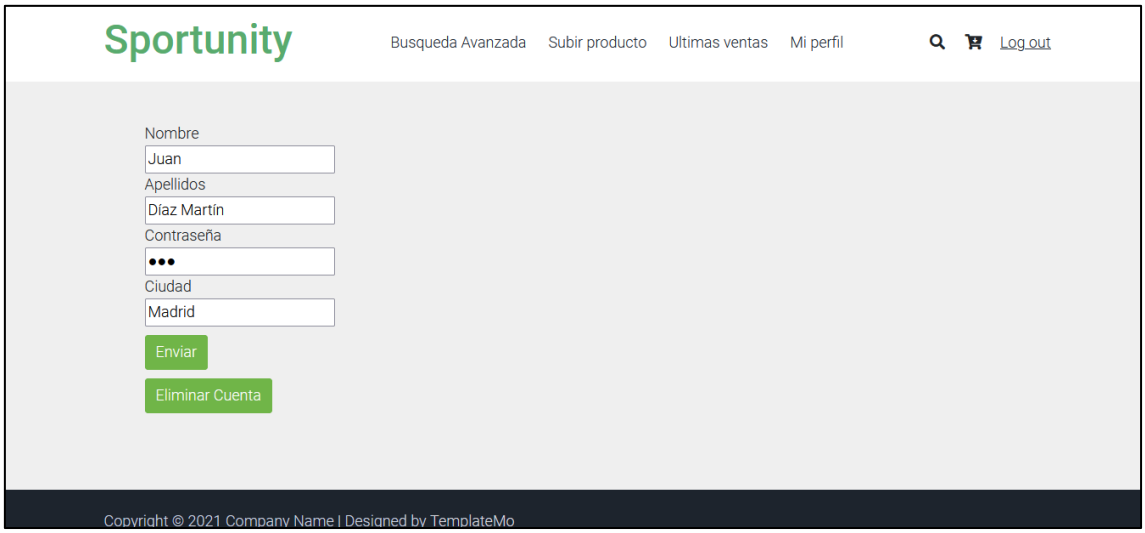


Fig. A.10. Formulario de modificación de perfil.

Cierre de sesión y eliminación de perfil

Para cerrar sesión basta con pulsar la opción *Log out* en la esquina superior derecha. Para eliminar la cuenta de la aplicación, se debe seleccionar la opción *Mi perfil* → *Modificar perfi* → *Eliminar cuenta*. En ambos casos se redigirá al usuario a la página inicial de la aplicación.

Búsquedas

Una vez el usuario ha iniciado sesión tiene la opción de realizar una búsqueda avanzada, bien desde la página principal, bien desde la sección "Búsqueda avanzada" del menú de navegación presente en cualquier pantalla si se ha iniciado sesión. El usuario rellenará los filtros que considere y se le mostrarán aquellos resultados que concuerden con el filtrado deseado.

Fig. A.11. Formulario de búsqueda avanzada.

Fig. A.12. Resultados de una búsqueda tras iniciar sesión.

Al haber iniciado sesión, además del botón de *detalles* relativo a cada producto, también existe el botón *Incluir en el carrito*, que permite guardar ese producto en la sección del carrito de la compra del usuario.

Subir un producto

Para subir un producto, se debe pulsar sobre el botón *Subir producto* del menú superior, y se redigirá al usuario a un formulario que rellenar con varios datos relativos al producto en cuestión.

Fig. A.13. Formulario de subida de un producto.

Al enviar el formulario, se redigirá al usuario a una página que muestre todos los productos que tiene a la venta.

Fig. A.14. Lista de productos a la venta de un usuario.

Modificar los datos de un producto

Para modificar un producto se debe pulsar sobre las opciones *Mi perfil* → *Mis productos* y sobre el producto deseado pulsar *Modificar producto*, y se mostrará una pantalla con el mismo formulario de subida de producto pero relleno con los datos actuales del producto, salvo la imagen y la categoría.

Al enviar el formulario, se redigirá al usuario a una página que muestre todos los productos que tiene a la venta.

Finalizar el proceso de compra

Para finalizar el proceso de compra, el usuario debe pulsar sobre el icono del carrito en la esquina superior derecha y se le redigirá a consultar el estado de su carrito de la compra.

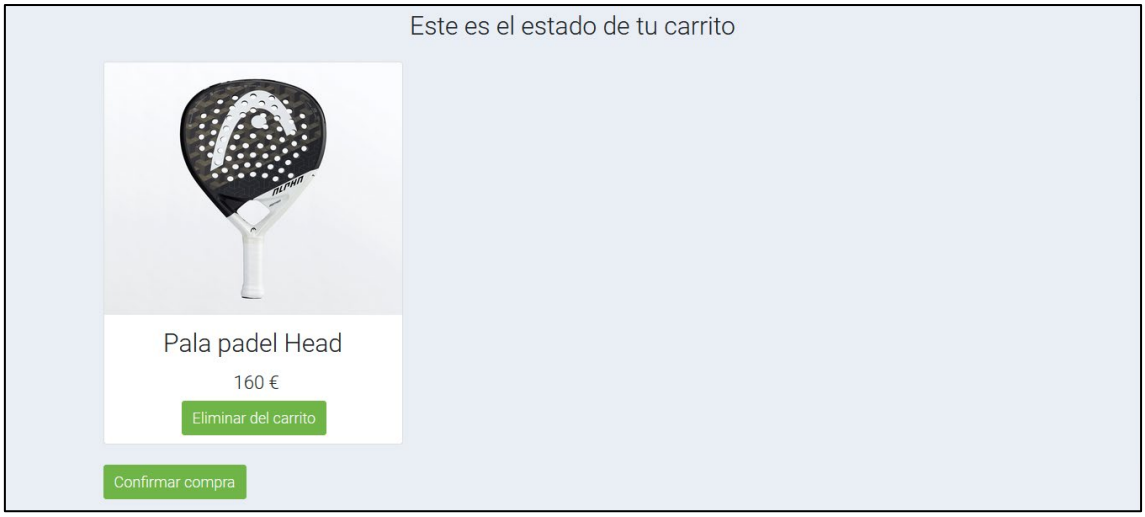


Fig. A.15. Estado del carrito de un usuario.

A continuación, debe pulsar sobre el botón *Confirmar compra* y aparecerá una pantalla para introducir los datos de la tarjeta con la que realizar el pago.

Fig. A.16. Datos de la tarjeta para efectuar el pago.

Una vez introducidos lo datos de la tarjeta y pulsado el botón de *Confirmar*, se habrá realizado la transacción y aparecerá una pantalla indicando el resumen de la compra.

Fig. A.17. Resumen de la compra.

El usuario vendedor, tras iniciar sesión puede ver en la sección *Ultimas ventas*, la venta realizada. Cabe destacar que solo podrá visualizar la venta en esta sección una sola vez, si bien puede consultar su historial de ventas en la sección *M perfil→Mis ventas.*

BIBLIOGRAFÍA

[1] "El ecommerce gana adeptos: crecerá casi un 25% en 2021 - elEconomista.es." https://www.eleconomista.es/empresas-finanzas/noticias/11328299/07/21/El-ecommerce-gana-adeptos-crecera-casi-un-25-en-2021.html (accessed Oct. 06, 2021).

[2] "Los fundamentos de desarrollo web y sus principios - ingeniovirtual.com." https://www.ingeniovirtual.com/fundamentos-de-desarrollo-web/ (accessed Oct. 06, 2021).

[3] "PHP." https://desarrolloweb.com/home/php (accessed Oct. 06, 2021).

[4] "Java EE - Wikipedia, la enciclopedia libre." https://es.wikipedia.org/wiki/Java_EE (accessed Oct. 06, 2021).

[5] "¿Qué es J2EE en programación Java?" https://formatalent.com/que-es-j2ee-en-programacion-java/ (accessed Oct. 06, 2021).

[6] "Java Server Pages (JSP)." https://www.it.uc3m.es/labttlat/2007-08/material/jsp/ (accessed Oct. 06, 2021).

[7] "¿Qué es un POJO, EJB y un Bean? | Carlos Pesquera Nieto." https://carlospesquera.com/que-es-un-pojo-ejb-y-un-bean/ (accessed Oct. 06, 2021).

[8] "Python: El lenguaje del futuro - BAOSS." https://www.baoss.es/python-el-lenguaje-del-futuro/ (accessed Oct. 06, 2021).

[9] "Python se ha convertido en el lenguaje de programación que crece más rápido." https://www.genbeta.com/actualidad/python-se-ha-convertido-en-el-lenguaje-de-programacion-que-crece-mas-rapido (accessed Nov. 19, 2021).

[10] "HTML: Lenguaje de etiquetas de hipertexto | MDN." https://developer.mozilla.org/es/docs/Web/HTML (accessed Oct. 06, 2021).

[11] "¿Cuáles son las mejores tecnologías para desarrollo web?" https://www.armadilloamarillo.com/blog/cuales-son-las-mejores-tecnologias-para-desarrollo-web/ (accessed Oct. 06, 2021).

[12] "Qué es una base de datos relacional | Oracle España." https://www.oracle.com/es/database/what-is-a-relational-database/ (accessed Oct. 06, 2021).

[13] "Bases de datos SQL | AWS." https://aws.amazon.com/es/relational-database/ (accessed Oct. 06, 2021).

[14] "Qué es MySQL: Características y ventajas | OpenWebinars." https://openwebinars.net/blog/que-es-mysql/ (accessed Oct. 06, 2021).

[15] "Oracle Database - Wikipedia, la enciclopedia libre." https://es.wikipedia.org/wiki/Oracle_Database (accessed Oct. 06, 2021).

[16] "Microsoft SQL Server vs. MySQL vs. Oracle Comparison." https://db-engines.com/en/system/Microsoft+SQL+Server%3BMySQL%3BOracle (accessed Oct. 06, 2021).

[17] “Base de datos no relacional. ¿Qué es? Características y ejemplos | Ayuda Ley Protección Datos.” https://ayudaleyprotecciondatos.es/bases-de-datos/no-relacional/#Ventajas (accessed Oct. 06, 2021).

[18] “Bases de datos no relacionales | Bases de datos de gráficos | AWS.” https://aws.amazon.com/es/nosql/ (accessed Oct. 06, 2021).

[19] “La base de datos líder del mercado para aplicaciones modernas | MongoDB.” https://www.mongodb.com/es (accessed Oct. 06, 2021).

[20] “The Essential Difference Between Couchbase & MongoDB | by Cubet Techno Labs | Medium.” https://medium.com/@Cubet_Techno_Labs/the-essential-difference-between-couchbase-mongodb-5b4d7d192cff (accessed Oct. 06, 2021).

[21] “Arquitecturas monolíticas o arquitectura de microservicios.” https://www.ilimit.com/blog/arquitecturas-monoliticas-o-arquitectura-de-microservicios-ventajas-e-inconvenientes/ (accessed Oct. 06, 2021).

[22] “Arquitectura de microservicios vs arquitectura monolítica - Viewnext.” https://www.viewnext.com/arquitectura-de-microservicios-vs-arquitectura-monolitica/ (accessed Oct. 06, 2021).

[23] “Emanuel Goette, alias Crespo: El patrón de arquitectura microkernel.” https://emanuelpeg.blogspot.com/2017/07/el-patron-de-arquitectura-microkernel.html (accessed Nov. 19, 2021).

[24] “Patrones de arquitectura de software Desarrolladores Páginas Web.” https://www.dreams.es/transformacion-digital/desarrolladores-paginas-web/patrones-de-arquitectura-de-software (accessed Oct. 06, 2021).

[25] “Metodologías de desarrollo software | Blog Becas Santander.” https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html (accessed Oct. 06, 2021).

[26] “Desarrollo en cascada - Wikipedia, la enciclopedia libre.”

[27] “El origen de las metodologías ágiles | by Andrés Salcedo | Andrés Salcedo | Medium.” https://medium.com/andressalcedo/el-origen-de-las-metodolog%C3%ADas-%C3%A1giles-2104558b8b9f (accessed Oct. 06, 2021).

[28] “Qué es SCRUM – Proyectos Ágiles.” https://proyectosagiles.org/que-es-scrum/ (accessed Oct. 06, 2021).

[29] “Scrum (desarrollo de software) - Wikipedia, la enciclopedia libre.” https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software) (accessed Oct. 06, 2021).

[30] “Lean software development - Wikipedia, la enciclopedia libre.” https://es.wikipedia.org/wiki/Lean_software_development (accessed Oct. 06, 2021).

[31] “El ecommerce de artículos deportivos se dispara un 79% en el año del Covid-19 | Palco23.” https://www.palco23.com/entorno/el-ecommerce-de-articulos-deportivos-se-dispara-un-79-en-el-ano-del-covid-19.html (accessed Oct. 06, 2021).

[32] James Martin, *An Information Systems Manifesto*. 1984.

[33] “Expertos en desarrollo Java Enterprise Edition - 3digits.” https://www.3digits.es/desarrollo/Desarrollo_de_software_y_programacion_de_aplicaciones_web_con_JEE_-_Java_Enterprise_Edition.html (accessed Oct. 07, 2021).

[34] “Python vs Java in 2021: Comparison, Features & Applications.” https://hackr.io/blog/python-vs-java (accessed Oct. 07, 2021).

[35] “Qué es Spring Framework y por qué usarlo | OpenWebinars.” https://openwebinars.net/blog/conoce-que-es-spring-framework-y-por-que-usarlo/ (accessed Oct. 07, 2021).

[36] “Bases de datos relacionales vs. no relacionales: ¿qué es mejor? - Aukera.” https://aukera.es/blog/bases-de-datos-relacionales-vs-no-relacionales/ (accessed Oct. 07, 2021).

[37] “Por qué elegir el gestor de base de datos MySQL | FP Online.” https://fp.uoc.fje.edu/blog/por-que-elegir-el-gestor-de-base-de-datos-mysql/ (accessed Oct. 07, 2021).

[38] “MySql: uno de los gestores de base de datos más utilizado – Fx2 – Software House.” https://fx2.com.uy/blog/mysql-un-aliado-para-la-gestion-de-base-de-datos/ (accessed Oct. 07, 2021).

[39] “Razones por la que utilizar MySQL | Instituto FOC - FP Informática Online.” http://www.foc.es/2013/04/11/988-razones-por-la-que-utilizar-mysql.html (accessed Oct. 07, 2021).

[40] “¿Qué es Spring Boot? - Arquitectura Java.” https://www.arquitecturajava.com/que-es-spring-boot/ (accessed Oct. 08, 2021).

[41] “Tomcat vs Glassfish - Tech blog for developers | FacilcloudTech blog for developers | Facilcloud.” https://www.facilcloud.com/noticias/tomcat-vs-glassfish/ (accessed Oct. 30, 2021).

[42] “Java EE Containers - The Java EE 5 Tutorial.” https://docs.oracle.com/javaee/5/tutorial/doc/bnabo.html (accessed Nov. 19, 2021).

[43] “Sesión 1: Introducción a la tecnolgía EJB.” http://www.jtech.ua.es/j2ee/2003-2004/abierto-j2ee-2003-2004/ejb/sesion01-apuntes.htm (accessed Oct. 30, 2021).

[44] “Autenticación de APIs basada en tokens con Spring y JWT.” https://blog.softtek.com/es/autenticando-apis-con-spring-y-jwt (accessed Dec. 09, 2021).

[45] “Qué es Json Web Token y cómo funciona | OpenWebinars.” https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona/ (accessed Dec. 09, 2021).

[46] “¿Qué aspectos legales debe cumplir una página web? - Vadillo Asesores.” https://www.grupovadillo.com/aspectos-legales-pagina-web/ (accessed Dec. 17, 2021).

[47] “BOE.es - BOE-A-2018-16673 Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.”

https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673&p=20181206&tn=2 (accessed Nov. 05, 2021).

[48] “BOE.es - BOE-A-2014-4950 Ley 9/2014, de 9 de mayo, General de Telecomunicaciones.” https://www.boe.es/buscar/act.php?id=BOE-A-2014-4950#a1 (accessed Nov. 05, 2021).

[49] “BOE.es - BOE-A-2002-13758 Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico.” https://www.boe.es/buscar/act.php?id=BOE-A-2002-13758#cii (accessed Nov. 05, 2021).

[50] UC3M, *RestfulWS - class material*.

[51] “Java Foundations: Java - Estándares de programación.” http://javafoundations.blogspot.com/2010/07/java-estandares-de-programacion.html#2_javadoc (accessed Nov. 07, 2021).

[52] “Tabla de coeficientes de amortización lineal. - Agencia Tributaria.” https://www.agenciatributaria.es/AEAT.internet/Inicio/_Segmentos_/Empresas_y_profesionales/Empresas/Impuesto_sobre_Sociedades/Periodos_impositivos_a_partir_de_1_1_2015/Base_imponible/Amortizacion/Tabla_de_coeficientes_de_amortizacion_lineal_.shtml (accessed Nov. 07, 2021).

[53] “Salario Medio en España (Sueldos 2021) | Jobted.es.” https://www.jobted.es/salario (accessed Nov. 08, 2021).

[54] “Convertir salario por hora (sueldo horario) a sueldo anual (salario por año).” https://convertir-sueldo-hora-ano.appspot.com/ (accessed Dec. 18, 2021).