Escuela Politécnica Superior de Linares

Grado en ingeniería telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

VENTA DE ARTÍCULOS ONLINE DE DISCOVER ÚBEDA

Alumno: **Noelia Toral Jiménez**

Tutor: D. Ángel Inocencio Aguilera García
Depto.: Departamento de informática

Junio, 2018

Agradecimientos

Está terminando una etapa que ha sido muy importante para mí. Es la que me ha dado trabajo para mantenerme y muchos amigos que espero que sean amigos para toda la vida.

En primer lugar, me gustaría agradecerles a mis padres todo el esfuerzo que han hecho para que pudiese estudiar este grado y el ánimo que me han dado durante los momentos difíciles de la carrera. Sin ellos no hubiera conseguido nada.

A todos mis compañeros de la universidad por haber pasado con ellos unos años geniales.

A la universidad de Jaén ya que, sin ella no estuviera escribiendo esto ahora mismo.

Por último, y en especial, a mi novio Héctor que me ha estado apoyando y ayudando en todo momento y aguantándome en esta última etapa sobre todo cuando no me salía algo.

Índice

Índice de ilustraciones

Índice de tablas

Resumen

Este proyecto consiste en desarrollar una aplicación web y móvil de una tienda física llamada Discover Úbeda, destinada al turismo en Úbeda y venta de productos típicos de la zona.

La aplicación constará de una sección con todos los productos que se encuentran en la tienda, típicos de esta ciudad. Cada producto deberá tener una descripción, su precio, detalles técnicos (material, dimensiones, etc.). Esta sección estará organizada por categorías para que resulte muy fácil buscar cualquier producto.

Cualquier producto se podrá comprar a través de la aplicación web.

Para que cualquier persona pueda conocer al máximo Úbeda durante el tiempo que este en esta ciudad existirán una sección llamada visitas guiadas en la que se organizaran pequeños grupos con un guía que irán por todo el casco urbano y por la compra de cualquier producto de la tienda se descontarán un tanto por ciento en el precio de las visitas.

1. INTRODUCCIÓN

El mundo está cambiando muy rápido a lo largo de los años. Siempre buscamos poder realizar todo de forma más cómoda. Inventamos cosas para vivir mejor.

Hace aproximadamente 300 años las personas iban andando o en caballo a los sitios. Ahora existen muchos medios de transporte para poder ir a los sitios. Esto mismo ha pasado con la tecnología, desde que se desarrolló ha crecido demasiado rápido en tan solo unos años.

Con la evolución de las personas y las tecnologías se ha ido cambiando la sociedad y con ella sus hábitos y la forma de ver las cosas o de pensar. Por lo que, para poder comprender el porqué de este proyecto tenemos que pensar en todo esto y tenemos que situarnos en cómo vivimos actualmente.

Hoy en día no soltamos el móvil para nada. Esto puede tener sus ventajas y desventajas. Una ventaja buena es que puedes estar al día en todo. Otra ventaja es que una buena forma de atraer a personas a cualquier lugar es mediante la publicidad a través de Internet.

Con esto se puede entender que el objetivo es de este proyecto es atraer a las personas a donde yo quiero en este caso a una tienda de recuerdos a la vez que le estás haciendo que sea fácil poder ver los productos.

Si se consigue un buen marketing de la tienda online, atraerá a más usuarios y la tienda obtendrá más beneficios.

También se tiene que comentar que la tienda no solo se centra en vender productos típicos de Úbeda hay ropa y joyería. Por lo que esto le puede interesar no solo a turistas de Úbeda si no a cualquier persona, en este caso tener una página web accesible a todo el mundo viene muy bien.

1.1. Antecedentes

El problema del mercado del turismo es que prácticamente todas las tiendas que se dedican a esto venden los mismos productos y generalmente bastante más caro que en otras tiendas normales.

Otro problema es que cuando se va a visitar alguna ciudad no vas entrando en todas las tiendas viendo lo que hay ya que interesa ver más el patrimonio que tiene la ciudad que a ir tienda por tienda viendo generalmente los mismos productos como se ha comentado anteriormente.

Por tanto, hay que cambiar el punto de vista. Si se quiere destacar en un campo hay que realizar cosas innovadoras y distintas a la competencia.

Tampoco nos podemos quedar esperando a que todo venga hecho hay que echarse para adelante y realizar cosas nuevas.

1.2. Motivación

La idea de este trabajo es poder ayudar a una persona a meterse en el mundo de las tecnologías y dar su tienda a conocer a cualquier persona del mundo.

La tienda está situada en Úbeda que es Ciudad Patrimonio de la Humanidad, por lo que vienen muchos turistas a visitar esta bonita ciudad. Lo que interesa es que se incrementen las ventas gracias a la publicidad de llevar la tienda física al mundo de Internet. Ya que, cualquier persona podrá comprar un producto no solamente los turistas que vayan a visitar la ciudad.

Otra oportunidad que se recoge en este proyecto es poder investigar en diferentes lenguajes de programación y tecnologías con las que actualmente hay mucha demanda para trabajar en el ámbito de la programación.

1.3. Objetivos

Los objetivos de este proyecto son:

- El desarrollo de una web y aplicación móvil en la que se puedan ver todos los productos disponibles en la tienda.
- Diseño de una base de datos relacional en donde estarán todos los datos guardados.
- Elaboración de un proyecto de tecnología de la aplicación tecnológica donde están incluidas las fases de análisis, diseño planificación, codificación y prueba.
- Estudio de las diferentes herramientas que permitan el desarrollo de la herramienta propuesta y la puesta en marcha de la misma.

1.4. Estructura del documento

La Ingeniería del Software es la aplicación práctica del conocimiento científico en el diseño y construcción de los programas y la documentación requerida para su desarrollo, operación y mantenimiento. (Boehm 1976)

Este documento se va a basar en las diferentes etapas que existen en la Ingeniería del Software para realizar el proyecto. El modelo más conocido que utiliza la Ingeniería del Software y el que se va a seguir para realizar este proyecto es el modelo en cascada que consiste en cuatro fases. Se denomina en cascada porque cuando termina una fase se empieza con la siguiente ya que, el resultado de cada fase sirve para la siguiente.

La primera fase es la de análisis y se estudiara en el capítulo 2. En esta fase hay que definir muy bien los objetivos que desea el cliente y buscar que posibilidades existen en el mercado para llevar el proyecto de una forma u otra. También hay que dejar bien claro los requisitos funcionales y no funcionales que se llevarán a cabo. Estos requisitos se hablarán en el punto 2.4.1 y 2.4.2.

La segunda fase es la denominada fase de diseño que determina como se va a realizar el proyecto para cumplir todos los objetivos requeridos, es decir, cuáles son las tecnologías utilizadas, cuál va a ser la arquitectura del software, como va a ser la base de datos, etc.

La penúltima fase es la de codificación o implementación. Aquí se desarrolla todo el código junto con la base de datos.

Por último, la última fase es la de pruebas en la que una vez que ha sido todo implementado hay que probar todo el sistema para comprobar que todo funciona correctamente tal y como queremos o por el contrario si vemos errores corregirlos.

2. Análisis

Como se ha comentado anteriormente se va a realizar una aplicación web y móvil para la venta online de productos.

Esta aplicación está basada en una tienda física donde la principal línea de negocio es vender recuerdos de Úbeda, un pueblo situado en la provincia de Jaén. También vende otros tipos de productos como ropa, complementos y comida entre otros.

En este capítulo se va investigar sobre el análisis del mercado para comprobar si es factible o no tener una página web y una aplicación móvil y el mantenimiento que conlleva esto.

Por otro lado, se analizará cual es la mejor forma de realizar la aplicación, es decir, si utilizar puramente código o algún programa que realice casi todo el trabajo.

Por último, se describirán los requisitos del sistema establecidos por el cliente.

2.1. Análisis del mercado

2.1.1 Introducción

En este apartado vamos a estudiar la competencia que existe en el mercado de este proyecto. Debemos saber que realiza nuestra competencia para aprender de ella y buscar puntos débiles en los que sobresalir nuestra aplicación.

2.1.2 Estudio del mercado

Actualmente no existen ninguna plataforma que venda online recuerdos de Úbeda. Por tanto, esto es una gran ventaja para este proyecto.

Sobre plataformas para conocer Úbeda hay una que es www.ubedaybaezaturismo.com en la que se puede encontrar todo tipo de información para las personas que quieran visitar Úbeda y Baeza. La web está muy completa. Viene clasificada en distintas categorías como lugares de interés, gastronomía, eventos culturales, alojamiento, restauración, etc. El inconveniente que se encuentra es que se centran en dos ciudades: Úbeda y Baeza. Con lo cual, si alguien solo quiere estar visitar una de las dos por qué tiene que estar viendo información de la otra ciudad también. Ventaja que se puede sacar de esta página, es la mayoría de información que tiene.

Existe otra página web que es www.turismodeubeda.com en la que está enfocada al turismo está muy completa y se centra solo en Úbeda. La ventaja que se puede sacar al

igual que la anterior plataforma es que la información que viene está muy completa. El inconveniente es que cualquier turista mirará esa página para visitar Úbeda ya que hasta en el nombre pone turismo.

La última aplicación web que existe es www.ubedabaeza10.com pero esta consiste en ofertas en bares, comercios, servicios turísticos, pubs y cafeterías sobre las dos ciudades. Por lo que con mi plataforma no tiene mucho que ver.

Se puede obtener como conclusión que al mezclar el turismo con la venta de productos se conseguirá tener posición en el mercado y se puede aprovechar bastante la información que hay relacionada con Úbeda.

Sobre aplicaciones android hay dos:

- Úbeda: solo hay cuatro itinerarios que te tienes que descargar y solo da información de donde se encuentran en un mapa. Por lo que sirve de poca utilidad.
- Guía oficial de Úbeda: está muy bien organizada por categorías, pero solo se limita a poner la ubicación de los distintos comercios, museos, biblioteca municipal, planetario, etc.

En conclusión, las dos aplicaciones de android existentes básicamente no sirven para nada por lo que esto da una gran ventaja para realizar la aplicación Android.

2.2. Análisis del Arte tecnológico

2.2.1. Introducción

En este apartado vamos a investigar sobre las tecnologías que existen y con cuales se puede realizar una aplicación web y una aplicación móvil. Explicando las ventajas y desventajas de cada una.

2.2.2. CMS

CMS son las siglas de Content Management System en español conocido como gestor de contenidos es un software que se utiliza para desarrollar contenidos que pueda realizar y manejar cualquier persona sin necesidad de tener conocimientos de programación.

En términos generales, los programadores no tienen conocimientos de diseño web, es decir, de estilos. Si se hace una página web desde cero hay que enfrentarse al problema de que puede funcionar perfectamente pero no atraemos mucho tráfico a nuestra página porque no llama la atención y al final acabará en el olvido. Para disuadir este problema se puede conseguir la ayuda de un experto en diseño web o bien utilizar las plantillas de los CMS que se pueden personalizar. Con estas plantillas no quedará feo ni raro.

Por otra parte, los CMS están orientados a la realización de páginas web como foros, comercio electrónico, blogs, etc. Todo esto sin la necesidad de saber programar.

Una gran ventaja que tiene utilizar un CMS es el posicionamiento SEO. El posicionamiento SEO es a la altura en la que queda la página en las búsquedas de Google. Con cualquier página hecha a mano desde cero es muy difícil y laborioso poder superar el SEO que tiene un CMS. Con el gestor de contenidos es muy fácil realizar el SEO.

Otra ventaja que existe es el tiempo, es decir, al utilizar un gestor de contenidos se tarda menos del doble que realizar la página web a mano.

Detrás de los gestores de contenido hay un equipo de personas que van sacando actualizaciones por lo que también se está muy seguro.

A continuación, vamos a comparar los CMS más utilizados para el comercio electrónico:

- **WordPress**: es el más utilizado por la comunidad de usuarios. Empezó para la creación de blogs, pero ha ido evolucionando y hoy en día se puede realizar cualquier cosa a través de sus plugin. La mayoría de las páginas tanto de comercio electrónico como de otra categoría están realizadas con este gestor de contenidos. A continuación, podemos observar su logo.

Figura 1. Logo de WordPress

- **PrestaShop**: la principal característica de este CMS es la creación de tiendas online de comercio electrónico. Es de código abierto, escrito en PHP y se base en MySQL para la gestión de su base de datos.

  Hay que destacar el evento “Prestashop day”. Es un evento que se organiza todos los años en diversos países entre ellos España en el que

expertos del sector del comercio electrónico se juntan para dar charlas y conferencias.

La figura que hay a continuación es el logo de PrestaShop.

Figura 2. Logo de PrestaShop

- **Magento**: este es otro gestor de contenidos código abierto escrito en PHP. Lo que principalmente se destaca de Magento son los temas en el que puedes editar las páginas mediante PHP, HTML y CSS o bien instalarse los temas sin cambiar la visualización ni la funcionalidad.

  Magento está divido en módulos o plugin que aumentan su funcionalidad. La última característica es la integración donde los usuarios pueden integrar los nombres de dominio en un panel de controlar y gestionarlos desde un panel de administración.

  Hay que destacar un evento que hace Magento todos los años el denominado Imagine eCommerce. Es un congreso anual de comercio electrónico en el que se comparten ideas de comercio electrónico y se establece sesiones de oportunidad.

  A continuación, se muestra el logo de Magento.

Figura 3. Logo de Magento.

2.2.3. Framework

Un framework es un conjunto de normas, estandarizaciones y esquemas o patrones que se siguen para realizar el desarrollo de aplicaciones.

Se va a comentar un ejemplo para dejar más clara su definición. Una persona que realiza una web. No importa que lenguaje de programación utilice, solo que la aplicación la hace sin seguir ninguna norma haciéndolo como esa persona considere oportuno. Si tuviéramos que reutilizar ese código para cambiar alguna funcionalidad de la aplicación y no se puede hablar con el autor del código, el trabajo de cambiar el código resultará bastante tedioso. Nadie programa de la misma forma y al no seguir ninguna norma sería un poco caos. Por ello se utilizan los framework para seguir unas pautas a la hora de programar.

Los frameworks más utilizados son:

- **Spring MVC**: separa la aplicación en tres partes: modelo, vista y controlador. Se estudiará detenidamente más adelante.
- **Java Server Faces**: es un framework de SUN basado en la arquitectura modelo, vista y controlador. Simplifica el desarrollo de interfaces de usuario en aplicaciones JAVA EE.
- **Hibernate**: se utiliza para el mapeo de base de datos relaciones. Transforma la base de datos a objetos. Se utiliza en el lenguaje orientado a objetos de java y realiza cualquier operación en base de datos.
- **Struts 2**: es un framework para construir aplicaciones web de Java basadas en la filosofía MVC.
- **Laravel**: es de código abierto para realizar aplicaciones y servicios web en PHP. Proporciona bibliotecas orientadas a objetos, hash Bcrypt, restablecimiento de contraseñas, migración a base de datos, protección CSRF (Cross-Site Request Forgery), soporte MVC.

Como ventajas de utilizar framework para los proyectos de programación es que se tiene el esqueleto y solo hay que rellenarlo siguiendo unas pautas.

Es fácil encontrar diversas herramientas y librerías que se puedan incorporar a la aplicación y facilita el desarrollo de mantenimiento.

Para concluir hay que elegir entre realizar una aplicación mediante un gestor de contenidos o un framework.

La conclusión es que para una persona que no tenga conocimiento en lenguajes de programación o no tenga demasiado tiempo para programar le recomiendo que utilice un gestor de contenido. Por el contrario, para quien le guste programar y tenga tiempo le digo que siga investigando en las diferentes tecnologías que existen en el mercado. Nunca paran de evolucionar y salir nuevas y tenemos que estar al día de todo.

2.2.4. Arquitectura MVC

La arquitectura Modelo-Vista-Controlador de ahora en adelante MVC, es en términos generales unos patrones o reglas a seguir de software para desarrollar aplicaciones web.

Principalmente separa el software en tres capas: modelo, vista y controlador. Las capas son totalmente independientes entre sí, por tanto, si se modifica una parte de una capa este cambio no afecta a las demás. Por ejemplo, si cambiamos la base de datos solo se debe modificar la capa que se encarga de controlar los datos, las demás capas no hay que tocarlas. A continuación, se explicará la misión de cada capa.

La capa de modelo o capa de datos es la encargada de conectarse a la base de datos y recoger toda la información que haya en ella. Esta capa envía a la vista la información que el usuario o cliente desea visualizar en pantalla.

La capa controlador o la capa de negocio es un enlace entre la capa de modelo y la capa de vista ya que se encarga de procesar las acciones del usuario en la capa de la vista y pedir información a la capa de datos. Por ejemplo, supongamos que tenemos una tienda online en la que podemos insertar productos para vender. El formulario para insertar productos se encuentra en la vista y cuando rellenemos todos los datos y pulsemos el botón de enviar o insertar, el controlador recogerá todos los datos del formulario y los llevará a la capa de datos para insertarlos en la base de datos.

La última capa es la capa de la vista o la capa de presentación que es la interfaz gráfica que ve el usuario, es decir, absolutamente todo lo que ve el usuario.

En la siguiente imagen se muestra visualmente las diferencias entre las tres capas:

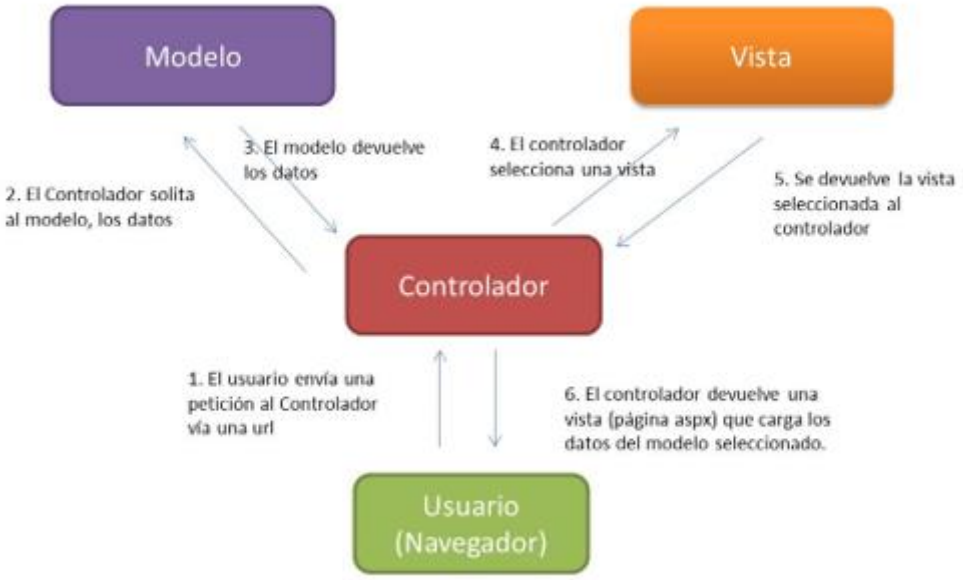


Figura 4. Modelo-Vista-Controlador

Las ventajas de la arquitectura MVC son las siguientes:

- La realización del código se realiza de forma modular. Como se ha comentado anteriormente esto es una gran ventaja ya que si tenemos que cambiar o modificar alguna parte solo nos centramos en esa parte y sin tener que tocar lo demás.
- Al ser modular es muy fácil de mantener la aplicación.
- Escalabilidad al contar con una base estructurada y sólida es fácil añadir nuevas funcionalidades.

Las desventajas de la arquitectura MVC son las siguientes:

- Los archivos para mantener y desarrollar se incrementan considerablemente
- Aprender a utilizar esta arquitectura puede ser más costosa que utilizar otros modelos sencillos.

2.2.5. Aplicaciones móviles

Para introducir este capítulo se va a explicar que es una aplicación móvil y que aplicaciones existen en el mercado.

Una aplicación móvil es una aplicación o programa informático que se ejecuta en un teléfono inteligente o tablet. Las aplicaciones permiten al usuario un entretenimiento, educación, acceso a servicios, etc.

Existen tres tipos de aplicaciones móviles: nativas, híbridas y aplicaciones web.

Los sistemas operativos más utilizados que utilizan las aplicaciones son Android, iOS y Windows Phone. En la siguiente imagen se puede observar los porcentajes con los distintos sistemas operativos que utiliza las personas sacado de una encuesta del 2017.

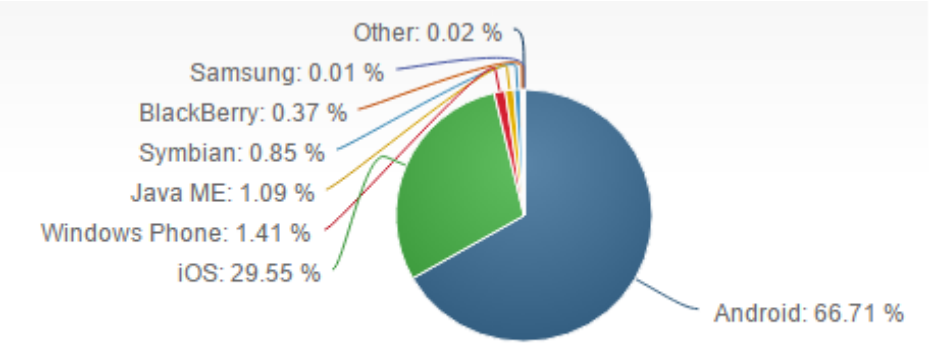


Figura 5. Utilización de los sistemas operativos móviles

Las aplicaciones nativas son las que se crean para un determinado sistema operativo. Es decir, si hay que crear una aplicación nativa para aprender inglés y queremos que en móviles con Android e iOS hay que realizarla dos veces una concretamente en Android y otra en iOS.

Esto implica que el coste y del desarrollo de la aplicación va a ser alto, no se puede reutilizar código entre las dos plataformas. Pero también existen ventajas como que la actualización de la aplicación es constante puede ser publicada en tiendas para su comercialización, la mayoría no necesitan estar conectadas a internet para que funcione.

Las aplicaciones web son las que se desarrollan usando lenguajes de programación como JavaScript, CSS, HTML y frameworks como jQuery mobile. Estas aplicaciones pueden ser ejecutadas desde cualquier sistema operativo su coste es mucho menor que las nativas, no necesitan instalación, no se pueden publicar en plataformas para su distribución y se necesitan conexión a Internet para utilizarlas.

Por último, las aplicaciones híbridas son es una mezcla de las dos anteriores. Se crean con framework como Phonegap o Ionic que utilizan CSS, HTML entre otros. Las ventajas que tiene son que pueden utilizar los recursos del sistema operativo, el coste es mucho menor que una nativa, son multiplataforma es decir se puede ejecutar en cualquier sistema operativo y permite su distribución.

2.3. Análisis del arte legislativo

El pasado 25 de mayo del 2018 entro en vigor la ley GDPR, General Data Protection Regulation, que en español es Regulación General de Protección de Datos. Es el nuevo reglamento para la protección de datos. Esta ley engloba muchos ámbitos en este documento se va a centrar en lo que concierne sobre la venta online.

Si un cliente se registra por primera vez en la aplicación se debe notificar a la Agencia Española de Protección de Datos (AGDP) con los documentos en los que se encuentra la información detallada de todos los datos personales de la persona en cuestión.

Asimismo, si se obtiene y se guarda información de terceros, las tiendas online deben contar con el consentimiento de la persona o empresa. Por eso siempre habrá que poner en algún sitio la posibilidad de aceptar o recabar el consentimiento de que se a obtener información personal para ser almacenados.

Un cliente debe poder en cualquier momento manipular sus datos desde que se registra por primera vez hasta el envío de un paquete.

Los ficheros en donde se encuentren todos los datos deben estar protegidos ante posibles hackeos y solo se podrá acceder por personas autorizadas.

Cualquier persona tiene derecho a poder borrar todos los datos personales que se hayan almacenado.

Hay que dejar claro cuánto tiempo se va a mantener almacenado los datos personales y para que, ya que no se pueden mantener para siempre.

En el caso en el que se rompa la seguridad hay un plazo máximo de 72 horas para avisar a los clientes de lo que ha ocurrido.

Todo formulario tiene que tener una explicación de que objetivo tiene y al final del formulario hay que poner una casilla de consentimiento de que el cliente va a enviar sus datos personales.

2.4. Requisitos del sistema

El objetivo en este punto es recopilar todos los requisitos, tanto funcionales como no funcionales, que debe cumplir el proyecto.

Por requisitos se entiende como las condiciones exigidas por el cliente para realizar el proyecto.

2.4.1. Requisitos funcionales

Los requisitos funcionales definen la funcionalidad del proyecto. En otras palabras, los objetivos que pone el cliente para que el sistema se comporte de una forma u otra.

Para el proyecto propuesto los requisitos funcionales son:

- **Autenticación de usuarios**: cualquier usuario que acceda a la página debe poder acceder a la aplicación con un usuario y contraseña.
- **Autenticación de empleados**: como el punto anterior cualquier empleado puede acceder a la aplicación con usuario y contraseña.
- **Gestión de productos**: solo los empleados y administrador podrá acceder a insertar productos.
- G**estión del carrito** de compra: cualquier usuario podrá ser capaz de insertar y eliminar productos de su carrito de la compra.
- **Gestión de venta**: el sistema tiene que ser capaz de comprar a través de algún método de pago por Internet en este caso mediante Paypal.
- **Gestión de categorías**: todos los proyectos deben estar clasificados por categorías y tener una sección para seleccionar la categoría que se desee y poder ver los productos solo de esa categoría.
- **Búsqueda de productos**: la aplicación tendrá un filtro para poder buscar cualquier producto que se desee.
- **Tipos de usuarios**: tienen que existir tres tipos de usuario bien diferenciados administrador, empleado y usuario.
    - El administrador es la persona que realiza todo el proyecto y el que se dedicará al mantenimiento de la aplicación.
    - Los empleados son las personas que trabajan en la tienda. Podrán subir productos.

- Los usuarios son los clientes que acceden a la plataforma para comprar los productos.

2.4.2. Requisitos no funcionales

Los requisitos no funcionales son características de funcionamiento o dicho de otra forma atributos de calidad. Estos requisitos los pone el cliente y son restricciones o condiciones que este impone para el proyecto en cuestión como por ejemplo tiempo de entrega, cantidad de usuarios, etc.

Para este proyecto vamos a definir los siguientes requisitos no funcionales:

- **Usabilidad**: debe ser fácil de utilizar e intuitivo para cualquier usuario.
- **Confiabilidad**: al realizar pagos económicos por Internet el usuario debe esta estar seguro de que no va a pasar nada pagando a través de nuestra página.
- **Diseño adaptable**: la aplicación debe poder visualizarse correctamente en cualquier dispositivo como tablet, smartphone y ordenador.
- **Disponibilidad**: el sistema no se puede colgar y debe poder soportar una tasa elevada de clientes.
- **Compatibilidad**: la aplicación deberá poder ejecutarse en cualquier sistema operativo

2.4.3. Restricciones

Las restricciones son las condiciones de desarrollo que impone el sistema. A continuación, vamos a comentar las restricciones necesarias para este proyecto.

Hardware

- Para el desarrollo del proyecto será necesario disponer de un ordenador y un Smartphone. No se va a entrar a detallar las características de los componentes ya que, dependerá de la carga de usuarios que tenga que soportar la aplicación.

Software

- La base de datos será MySql.
- Se debe disponer de un programa FTP para mover los ficheros al servidor en este caso se utilizará filezilla.
- Para visualizar la aplicación se necesitará conexión a Internet y un navegador web o un smartphone.

- Para la aplicación web se utilizará como tecnologías Spring e Hibernate y para la aplicación móvil se utilizará Phonegap estas dos irán conectadas a una base de datos en MySQL.

2.5 Análisis de viabilidad

En este capítulo vamos a estudiar si merece la pena realizar una aplicación web y una aplicación móvil para aumentar las ventas de la tienda.

Primero vamos a calcular el presupuesto que llevaría realizar el proyecto empezando por el software.

Software	Precio/mes	Duración (Meses)	Precio total
Eclipse	N/A	N/A	N/A
Phonegap	N/A	N/A	N/A
MySQL Server	N/A	N/A	N/A
Sublime Text	6 €	12	72 €
Navegador web	N/A	N/A	N/A
Xampp	N/A	N/A	N/A
Precio total			**72 €**

Tabla 1. Presupuesto del software

En segundo lugar, se calcularán los gastos en hardware

Hardware	Precio	Duración (Meses)	Precio total
Servidor	10 €/mes	12	120 €
Dominio	15 €/anuales	12	15 €
Smartphone	100 €	12	100 €
Ordenador	800 €	12	800 €
Total			**1035 €**

Tabla 2. Presupuesto del hardware

Por último, se calcula el sueldo de los trabajadores que han colaborado en el proyecto.

Trabajadores	Salario al año	Horas totales	Coste total
Programador Junior	15000€	300h	15000 €
Total			**1500 €**

Tabla 3. Presupuesto de los trabajadores

El presupuesto total del primer año es de 2.907 €.

Gastos	Precio
Software	72 €
Hardware	1035 €
Gastos de personal	15000 €
Total	2.607 €

Tabla 4. Presupuesto total

Para el segundo año y posteriores hay que quitar el precio del ordenador y del smartphone ya que se va a utilizar el mismo que el del primer año por lo que se quedaría en 1707€

$$(1)\ Coste\ total\ segundo\ año = 2.607€ - 800€ - 100€ = 1707€$$

Conclusión, merece la pena gastarse sobre 2600€ para realizar las dos aplicaciones ya que cara está muy bien de precio para ser dos aplicaciones y va a ser una importante entrada de beneficios a la tienda.

Conclusión, merece la pena gastarse sobre 2600€ para realizar las dos aplicaciones ya que cara está muy bien de precio para ser dos aplicaciones y va a ser una importante entrada de beneficios a la tienda.

2.6 Planificación temporal

El proyecto se ha divido en seis fases para realizar la planificación temporal. En la ilustración siguiente muestra todas las fases su duración, fecha de inicio y fecha final.

Nombre de la tarea	Fecha de Inicio	Fecha final
Búsqueda de información	**01/04/18**	**10/04/18**
Análisis	**10/04/18**	**23/04/18**
Estudio del mercado	10/04/18	10/04/18
Estudio del arte tecnológico	10/04/18	13/04/18
Estudio del arte legislativo	18/04/18	18/04/18
Requisitos del sistema	18/04/18	20/04/18
Análisis de viabilidad	23/04/18	23/04/18
Diseño del sistema	**24/04/18**	**02/05/18**
Implementación	**03/05/18**	**13/06/18**
Pruebas	**14/06/18**	**18/06/18**
Redacción manual de usuario	**18/06/17**	**20/06/17**
Revisión y finalización memoria	**20/06/18**	**21/06/18**

Figura 6. Planificación temporal

Con el siguiente diagrama de Gantt se puede observar mediante una línea temporal la ejecución del proyecto, cuanto tiempo se ha tardado para cada tarea.

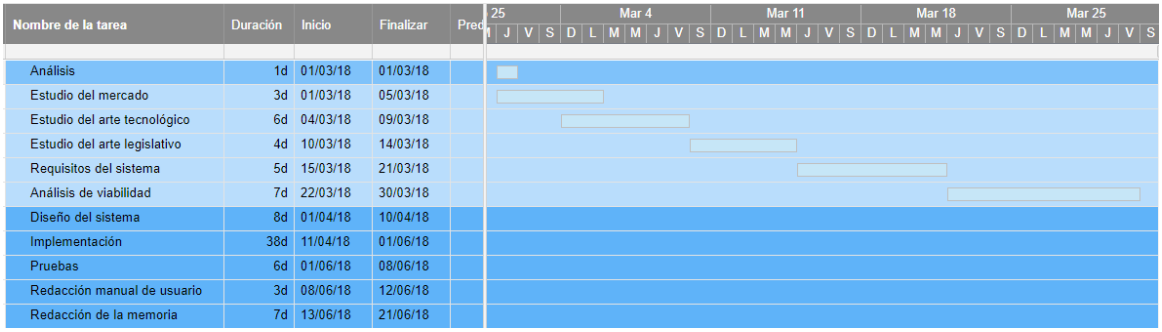


Figura 7. Planificación temporal Marzo

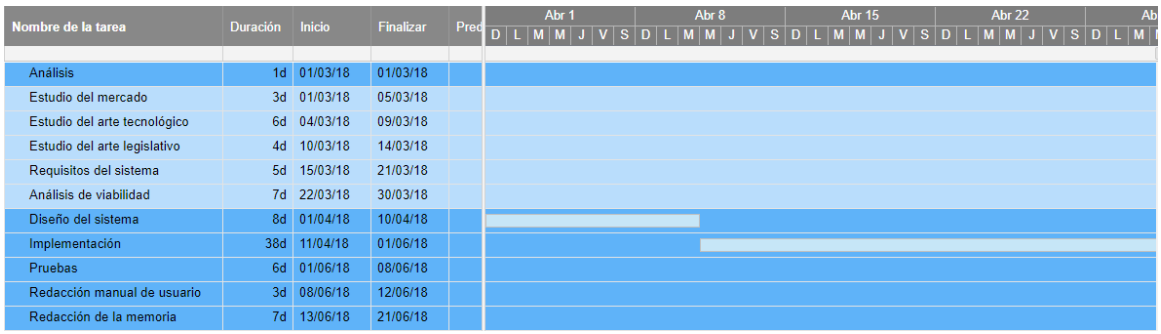


Figura 8. Planificación temporal Abril

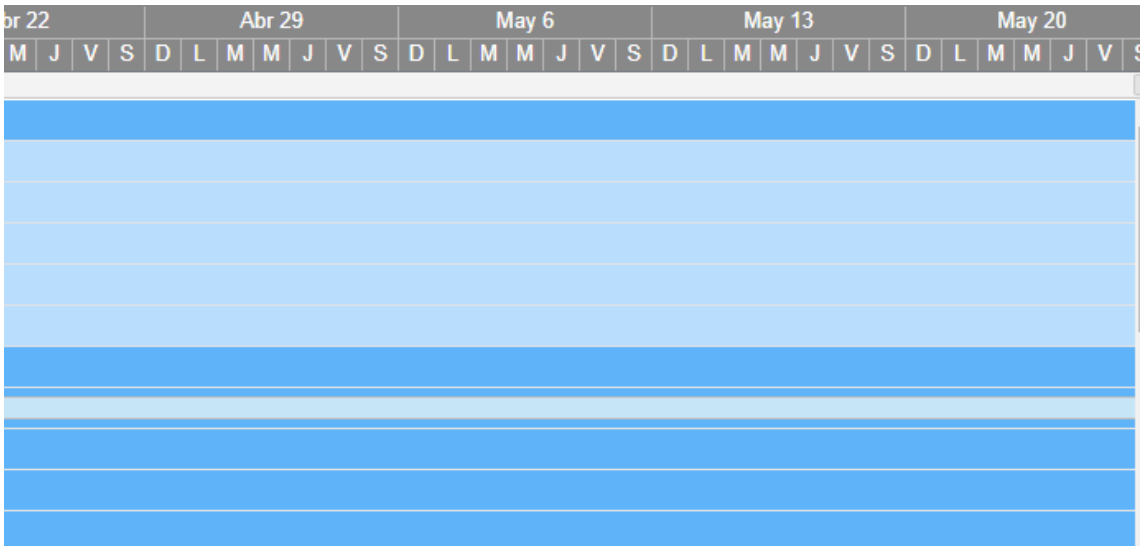


Figura 9. Planificación temporal Mayo

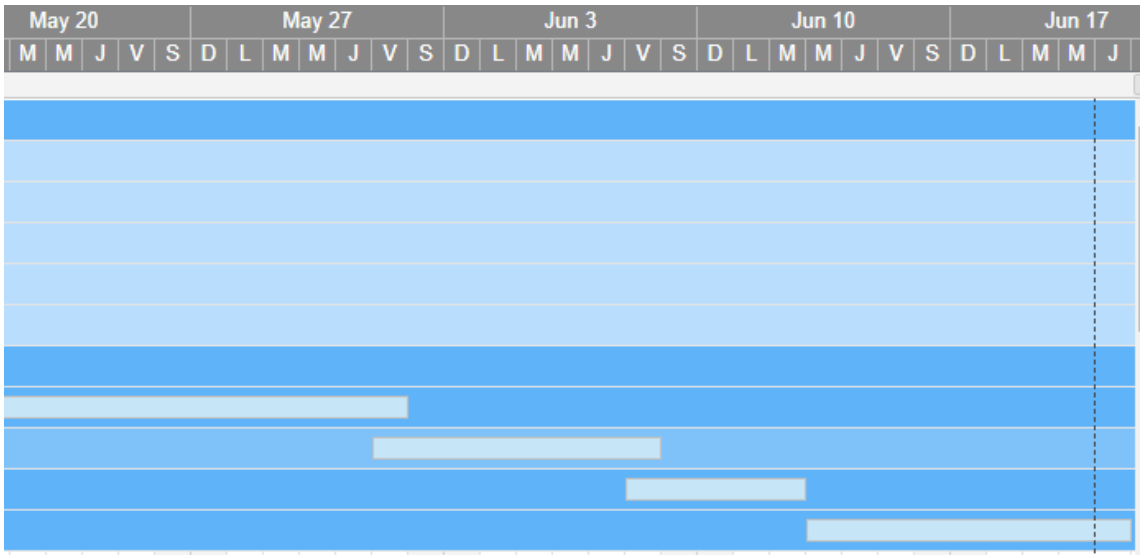


Figura 10. Planificación temporal Junio

3. Diseño del sistema

3.1. Tecnologías utilizadas

Vamos a ver que popularidad tienen las tecnologías que se han utilizado en este proyecto.

En la gráfica de abajo se muestran las tecnologías Spring, Spring Security, Hiberante, Java EE y MySQL. Esta gráfica representa desde el año 2009 hasta el año 2017 la popularidad que han tenido las diferentes tecnologías, es decir, si los programadores la han utilizado o no.

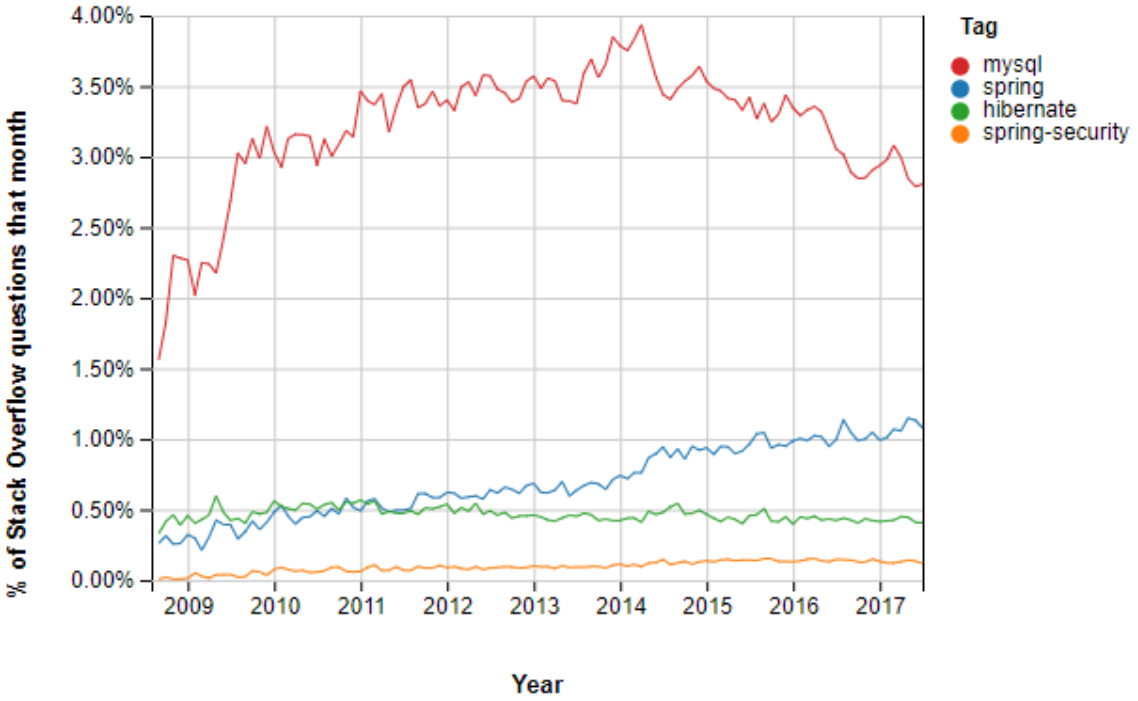


Figura 11. Evolución de las tecnologías

Cómo podemos observar MySQL supera a las demás y tiene algunos picos importantes. Esto es debido es la tecnología que más años lleva en el mercado las demás son muy recientes y es muy popular para el desarrollo con base de datos.

Se observar que Spring Security va aumentando su popularidad poco a poco, pero va aumentando e Hibernate se va manteniendo.

La tecnología que sin duda está creciendo y haciéndose cada vez más famosa es Spring. Hoy en día es la que más se utiliza.

La gráfica esta sacada de la página stack overflow trends en la que actualmente es fiable ya que hay más de ocho mil usuarios registrados.

En los siguientes capítulos, explícitamente capítulo 4, 5, 6, 7, 8, 9, 10 y 11 se va a entrar en detalle sobre las tecnologías utilizadas en este proyecto.

4. Spring

Spring es un framework que está basado en la arquitectura MVC explicada en el punto 2.2.4. y utiliza java como lenguaje de programación. Este framework proporciona crear aplicaciones web.

Figura 12. Logo de Spring

4.1 Arquitectura de Spring

Spring está dividido en una serie de módulos que se explicaran a continuación. En la siguiente imagen se muestran los módulos utilizados.

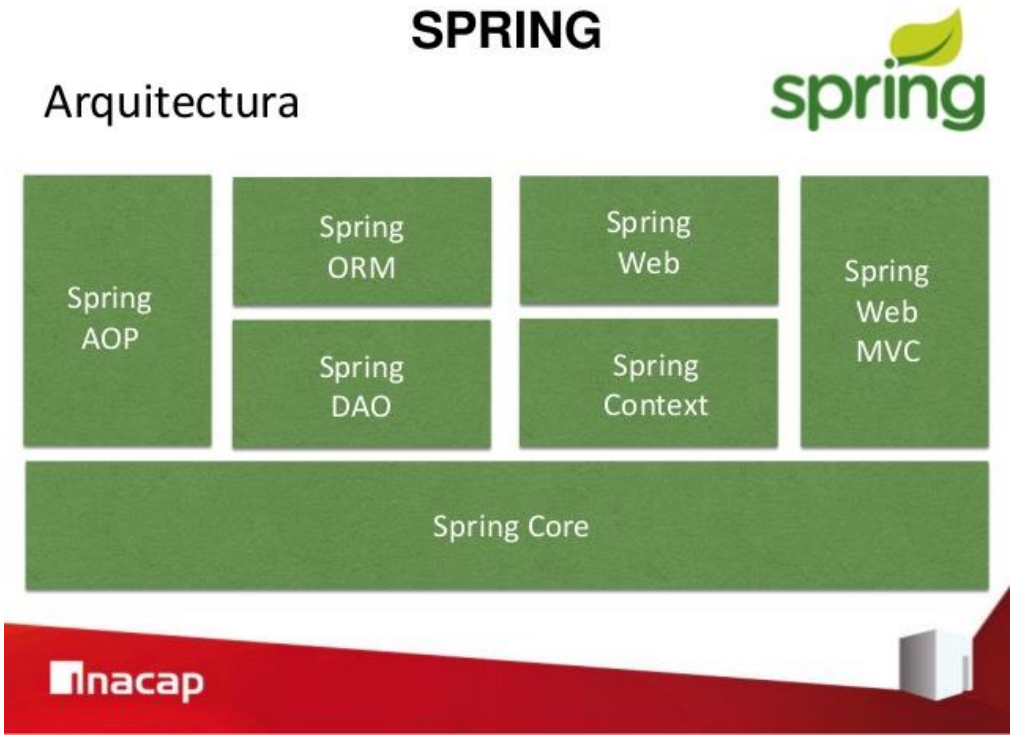


Figura 13. Arquitectura de Spring

A continuación, se va a explicar los módulos que tiene Spring.

- **Spring Core**: es el núcleo principal de Spring. Es un contenedor que está basado en un patrón de diseño llamado Inversión de Control, IoC, utilizado mediante la inyección de dependencias.
  La inversión de control se basa en el principio de Hollywood: "No nos llames; nosotros te llamaremos." El IoC invierte como se ejecuta normalmente el

programa con respecto a los métodos de programación tradicionales de ahí su nombre, es decir, el contenedor es el encargado en controlar el flujo de la aplicación.
La Inversión de Control se utiliza mediante la inyección de dependencias que se pueden realizar en un constructor, en un setter o un método. Esto ocurre cuando un objeto necesita de otro para realizar alguna acción.

- **Spring ORM**: las sigas ORM significan Object Relational Mapping que en castellano es Mapeo de Objeto Relacional. En otras palabras, es el mapeo de nuestra base de datos en código. En mi caso para esto he utilizado hibernate que explicaré en el punto 5.
- **Spring DAO**: las siglas DAO significan Data Access Object que quiere decir objeto de acceso a datos. Este se encargará de realizar las transacciones a la base de datos.
- **Spring context**: es la configuración de Spring a través de xml o anotaciones. Yo he utilizado anotaciones como iré enseñando a lo largo de esta memoria.
- **Spring Web MVC**: sigue la arquitectura modelo-vista-controlador explicada en el punto anterior.

Para realizar una aplicación en Spring es necesario seguir los siguientes pasos:

1. Creación de un proyecto con Maven y descarga de las dependencias que se necesiten
2. Desarrollo de la aplicación
3. Despliegue de la aplicación en un servidor

La misión de principal de Spring es simplificar el desarrollo del código de las aplicaciones. Esto lo hace posible gracias a la cantidad de módulos que tiene y a sus interfaces. Spring se acopla a la aplicación sin tener que modificar un código ya escrito.

Una de las ventajas de Spring es la Inyección de Dependencias o IoC (Inversion of Control). La Inyección de Dependencias es cuando se instancia o inicializa un objeto en tiempo de creación. Se puede ver la definición en el siguiente ejemplo.

Se tiene la clase Elfo que implementa la interfaz Guerrero

```
public class Elfo implements Guerrero {
  private Arma arma;

  public Elfo(){
    arma = new Arco();
  }

  @Override
  public void ataca() {
    arma.dispara();
  }
```

```
}
```

La clase Elfo crea una instancia arma de la clase Arma. Esto provoca que la clase Arco este ligada a la clase Elfo. Por ahora el Elfo solo puede utilizar un arco, pero ¿y si se quiere utilizar un cuchillo?

De la manera en la que está ahora mismo el código no se podría, la solución para esto la Inyección de Dependencias. Con el IoC el código quedaría de la siguiente forma:

```
public class Elfo implements Guerrero {
  private Arma arma;

  public Elfo(Arma a){
    arma = a;
  }

  @Override
  public void ataca() {
    arma.dispara();
  }
}
```

Ahora se crea una instancia del arma cuando se llama al método y se ha solucionado el problema que había. El elfo ya puede utilizar cualquier arma.

Spring utiliza una programación orientada a aspectos o AOP. Esta programación consiste en añadir funcionalidad a las clases y métodos sin tener que cambiar o modificar el código original. Por ejemplo, si se quiere que el método sea listar los productos de una tienda el método solo se debe preocupar de listar los productos no tiene que ver si el usuario tiene permisos o no.

Otra ventaja de Spring es la flexibilidad que tiene con la integración de otras herramientas como son Hibernate y Struts. Cómo se ha comentado con anterioridad al estar en módulos e interfaces es muy sencilla la integración con estas herramientas, simplemente habrá que utilizar algunas anotaciones de Spring.

Por último, comentar que también es muy fácil de gestionar con librerías se ponen las independencias y se descargar automáticamente.

4.2. Spring Security

Se encarga de dar seguridad a las aplicaciones basadas en Java EE en especial a las realizadas con Spring.

Una de las características que destaca Spring Security es que al contrario que ocurren con los Servlet de Java o la Especificación EJB es que si el entorno del navegador cambiar no hay que volver a reconfigurar todo como pasa en los Servlets de Java y en la especificación EJB.

Spring Security se encarga de dos áreas principales que son autenticación y autorización. Autenticación es el proceso que dice si un usuario es quien dice ser y autorización es si el proceso de decisión de realizar acciones o no.

Spring Security tiene una amplia variedad de modelos de autenticación, lo más destacados son encabezados de autenticación HTTP Basic, autenticación automática de "recordarme", LDAP, autenticación basada en formularios, entre otros.

Otra forma de ver Spring Security es la facilidad que tiene de implementación. Se puede integrar perfectamente en cualquier entorno por eso se está haciendo tan popular hoy en día.

Toda la configuración de seguridad que se realice es portable de un servidor a otro porque se encuentra dentro del EAR o el WAR de las aplicaciones.

Spring Security se creó en el año 2003. Por aquel entonces la comunidad de Spring era muy pequeña comparada con la de hoy en día. Spring Security surgió porque uno de los desarrolladores de Spring mando un correo preguntando si se había dado alguna consideración a una implementación de seguridad basada en Spring.

La ilustración siguiente es el logo de Spring Security.

Figura 14. Logo de Spring Security

5. Hibernate

Hibernate es una herramienta de mapeo de objetos relacionales para la plataforma Java y .Net. Esta herramienta transforma las bases de datos relacionales a la filosofía de objetos de Java mediante mapeo de atributos.

Esta herramienta es de software libre y está distribuida bajo los términos de licencia GNU LGPL.

Para realizar las consultas a la base de datos se puede realizar de dos formas mediante un lenguaje llamado HQL, Hibernate Query Languaje o mediante una API para construir las consultas denominado criteria.

La configuración de Hibernate se puede realizar mediante archivos XML o mediante anotaciones.

Una característica de Hibernate es la rapidez de desarrollo para las aplicaciones ya que utiliza la persistencia de datos que es la capacidad de recuperar y almacenar los datos de forma transparente al desarrollador.

Para utilizar Hibernate es necesario tener algunos conocimientos mínimos de SQL y Java.

Esta herramienta es la que se ha elegido para gestionar en la aplicación la base de datos. En el capítulo 13.1.1. se realizará la configuración de Hibernate mediante anotaciones.

Figura 15. Logo de Hibernate

5.1. Arquitectura de Hibernate

En la siguiente imagen se muestra la arquitectura de Hibernate.

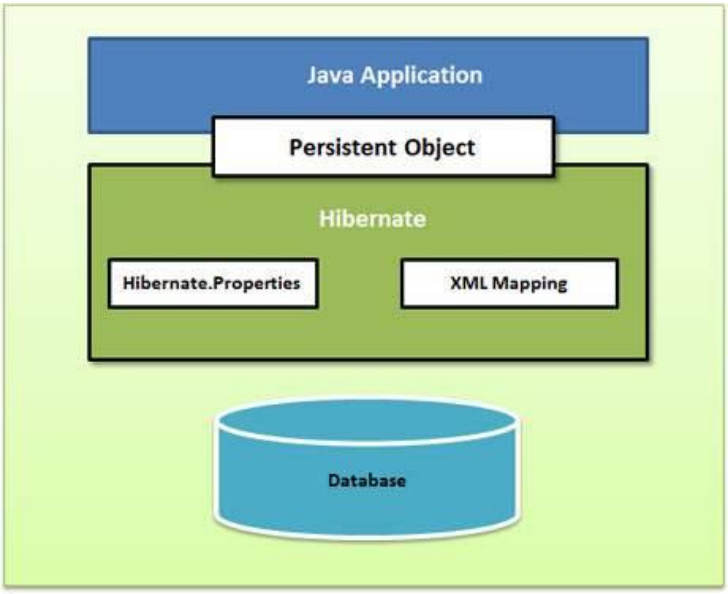


Figura 16. Arquitectura hibernate

Hibernate se comunica mediante la interfaz Session. Para cada tipo de base de datos debe existir una instancia de sesión.

Tiene dos tipos de objetos. El objeto de tipo transient que sólo existe en memoria y no en un almacén de datos y el objeto de tipo peristent que es un objeto ya almacenado y por tanto persistente.

Una sesión pertenece a una misma hebra de ejecución. Por lo que para crear instancia de sesiones y realizar operaciones se utiliza la interfaz SessionFactory.

6. PHP

Figura 17. Logo de PHP

Para mi aplicación móvil he decidido utilizar como lenguaje de programación PHP, ya que la plataforma elegida para hacer la aplicación ha sido Phonegap y en esta se puede utilizar PHP. Su logo se muestra en la imagen de arriba.

PHP es un lenguaje de código abierto que se suele utilizar para el desarrollo de aplicaciones web dinámicas con información almacenada en una base de datos.

Las partes de cliente y servidor están muy diferenciadas. El código de PHP es ejecutado en el lado del servidor y para recibir información en la parte de cliente se utiliza JavaScript.

Un dato importante es que el código escrito en PHP es invisible a los navegadores web y al cliente ya que se ejecuta el código en el servidor y envía la información al HTML del navegador.

Como se ha comentado anteriormente PHP se utiliza en el lado del servidor y se puede hacer cosas como recoger información de formularios, enviar y recibir cookies o generar páginas con contenidos dinámicos. No solamente queda en esto este lenguaje, existen tres campos principales donde se utilizan scripts de PHP que son:

- Scripts en el lado del servidor. Es donde más se utiliza. Para que funcione se necesita un analizador de PHP, un servidor web y un navegador web.
- Scripts desde la línea de comandos. Aquí se puede ejecutar sin necesidad de un servidor o navegador al contrario que el que se ha comentado anteriormente.
- Se pueden escribir aplicaciones de escritorio. Para estas aplicaciones se suele utilizar PHP-GTK que sirve para escribir los programas con interfaz gráfica.

La ventaja de PHP es que se puede emplear con todos los sistemas operativos principales y cualquier servidor web. También se puede utilizar con diversas bases de datos.

Este lenguaje es fácil de utilizar para personas que no tiene demasiado conocimientos de programación ya que es fácil de leer y entender esto no quita que sea un lenguaje potente. Algunos ejemplos de uso de PHP son generar archivos de texto o PDF o creación de imágenes.

7. Phonegap

La siguiente imagen muestra el logo de Phonegap.

Figura 18. Logo de Phonegap

Phonegap fue desarrollado por Nitobi con licencias de software libre pero en el 2011 Adobe obtuvo la adquisición de Nitobi por lo que PhoneGap paso a ser controlado por Adobe. Entonces Adobe dono PhoneGap a la Apache y con esto conservo Phonegap su integridad libre.

Phonegap es un framework utilizado para realizar aplicaciones móviles híbridas. Este framework ejecuta código desarrollado en HTML, CSS, JavaScript y PHP como si fueran aplicaciones nativas para móviles o tablets.

Estas aplicaciones se pueden publicar en las tiendas de aplicaciones como Google Play, Windows Store o App Store de Apple y puede utilizan los recursos de los dispositivos en que se ejecutan como por ejemplo la cámara o la geolocalización. Phonegap utiliza Apis para tener acceso a los recursos del dispositivo en que se encuentre.

He elegido este framework para realizar la aplicación móvil porque se puede realizar para todos los sistemas operativos y actúa como una nativa. Y el código PHP con el que se desarrolla es fácil y rápido de aprender.

Otra razón que viene vinculada con la primera es que fácil de mantener para todos los móviles ya que si hubiese sido una aplicación nativa tienes que actualizar tantos códigos como en sistemas operativos tengas la aplicación. Aparte haciendo una aplicación híbrida llega a todo el mundo si se elige una aplicación nativa solo va a llegar a las personas que tengan el sistema operativo que se elija para desarrollar.

8. MySQL

La siguiente imagen muestra el logo de MySQL.

Figura 19. Logo de MySQL

MySQL es el sistema elegido parar realizar la gestión de la base de datos relacional de la aplicación. Es un gestor de base de datos en el que las funcionalidades principales son el multihilo y multiusuario por lo que permite que varias personas entren a la vez y realicen transacciones sin ningún problema.

Fue desarrollada en el 1995 por la empresa MySQL AB. Unos años después en el 2008 esta empresa fue adquirida por Sun Microsystems y a su vez comprada por Oracle Corporation en el 2010.

Este gestor se utiliza mucho hoy en día porque es muy fácil y sencillo de manejar y funciona perfectamente con base de datos relacionales. Una base de datos relacional es aquella en que la información se guarda en tablas separadas conectadas a través de algunos datos. Esto permite gran velocidad y flexibilidad.

MySQL permite su utilización junto con los lenguajes de programación más demandados actualmente como PHP y Java.

Actualmente es la base de datos más popular del todo el mundo. Aplicaciones reconocidas como Facebook, Twitter o Youtube la utilizan.

9. Bootstrap

La siguiente imagen muestra el logo de Bootsrap.

Figura 20. Logo de Bootsrapp

Bootstrap es un framework de código abierto que se utiliza para el diseño web de aplicaciones.

Está basado en HTML, CSS y JavaScript. Este framework crea unas plantillas con el diseño ya realizado para todo tipo de requerimientos como formularios botones, modelos responsive, menús de navegación, etc.

Este framework fue creado por dos trabajadores de Twitter Mark Otto y Jacob Thornton. Se creó con la idea de acelerar el desarrollo de sus proyectos.

En 2011 Twitter liberó a Bootsrap como código abierto y se convirtió en el proyecto más popular de GitHub.

Se ha elegido este framework para el diseño gráfico ya que es muy fácil de utilizar y combina perfectamente todos los colores y formas.

10. JSON

JavaScript Object Notation son las siglas de JSON, en español objetos de notación de JavaScript. JSON es un formato estándar para representar los datos. Este formato es solo de tipo texto y es muy útil para intercambiar información entre navegador y servidor.

También es muy fácil de tratar mediante JavaScript.

JSON tiene el siguiente formado predefinido [{"nombre":"valor","nombre":"valor”}].

A continuación, se muestra un ejemplo sacado del proyecto.

```
[
	{“nombreProducto”: “Abanico”,
	“descripcionCorta”: “Abanico con flores”,
	“url”: “static/imgProductos/11/1.png”,
	“precio” : “3.5”,
	“idProductos”: “11”}
	,
	{“nombreProducto”: “Azulejo”,
	“descripcionCorta”: “Azulejo para cualquier sitio”,
	“url”: “static/imgProductos/12/1.png”,
	“precio” : “3.5”,
	“idProductos”: “12”}

}]
```

11. Paypal

La siguiente imagen muestra el logo de Bootsrap.

Figura 21. Logo de Paypal

Paypal es una empresa estadounidense que tiene un sistema por todo el mundo para realizar transacciones de dinero entre usuario y el sitio online para comprar.

Inicialmente Paypal se utiliza como servicio para pagar transferencias entre dispositivos PDAs pero el pago por web se fue haciendo más importante

En 2002 eBay compró a Paypal teniendo más del 50% de los usuarios de eBay. Pero unos años después, en el 2015, eBay se separó de Paypal.

Paypal no es reconocido como banco por lo que no se rige por las mismas leyes que las entidades bancarias. No obstante, tiene que obedecer las reglas del Departamento del Tesoro de los Estados Unidos y de la Autoridad de Servicios Financieros de la Unión Europea. Esto es debido para evitar transacciones no autorizadas o corrupción con el dinero.

Una de las principales características de este método de pago y de porque es tan popular es porque tiene una política de protección al comprador en la que si algún producto no es recibido o es muy distinto al descrito Paypal se encarga de hablar con la empresa vendedora, devolver el producto y que devuelvan el dinero sin que el usuario tenga que realizar nada.

Paypal no cobra por meter dinero en su cuenta o realizar el pago a otra persona u empresa pero si cobra una pequeña comisión al vendedor.

Con respecto a este proyecto se ha elegido Paypal como método de pago ya que se pueden crear unos botones crear unos botones en HTML para que los desarrolladores puedan implementarlo en su página web, realizar pruebas y fiarse de este método. Esta técnica es utilizada para atraer a más empresas ya que si los desarrolladores pueden implementarlo fácilmente y realizar pruebas se van a fiar de este método.

11.1. Botones de Paypal

En este capítulo se va a explicar cómo realizar e implementar los botones de Paypal. Pero antes vamos a ver que botones proporciona Paypal.

Paypal ofrece a los desarrolladores cuatro botones para que los implementen en sus códigos. Son los siguientes:

- Carrito de la compra: tiene una opción opcional para realizar seguimiento e inventario dentro de la plataforma de Paypal.
- Comprar ahora: al igual que el carrito de la compra, tiene una opción opcional para realizar seguimiento e inventario dentro de la plataforma de Paypal.
- Suscripciones: se introduce el nombre del artículo, la facturación de cada ciclo y cada cuánto.
- Donativos: hay que introducir el nombre de organización o servicio

Los cuatro botones se pueden poner para que el código este protegido.

Para que el pedido se introduzca en la base de datos de la aplicación hay que configurar un archivo en PHP en la notificación de pago instantánea. Este archivo se conectará a la base de datos recogerá los atributos por POST de la compra y los insertará en la base de datos.

12. Diseño

12.1. Recogida de datos

Para realizar el diseño de la aplicación y la base de datos hay que dejar bien claro que datos queremos obtener y guardar. No se puede dejar nada ambiguo porque el sistema podría quedar en algunas partes inconsistente.

La tienda online se va a dedicar a la venta de recuerdos de Úbeda. Estos productos hay que clasificarlos por categorías para que estén agrupados y sea más fácil para el usuario buscar que producto se quiere. Las principales características que se van a llevar a cabo son artesanía, complementos, heráldica, recuerdos y regalos.

También se tiene que diferenciar los roles. En este caso existirán tres tipos de roles: administrador, empleado y usuario. El administrador mantiene la web. El empleado es el encargado de gestionar la web subiendo productos, gestionando los pedidos, etc. Y los usuarios que son las personas que van a realizar la comprar por Internet.

Por último, indicar que se ha tenido que realizar fotos de los productos de la tienda física para subirlos a la aplicación.

12.2. Modelo entidad-relación

Una vez que se ha analizado los datos pasamos a realizar el modelo o diagrama entidad-relación.

El modelo entidad-relación es una herramienta para representar los datos recogidos en el punto anterior de forma que se pueda establecer en una base de datos real.

En este proyecto el modelo entidad-relación queda de la siguiente forma:

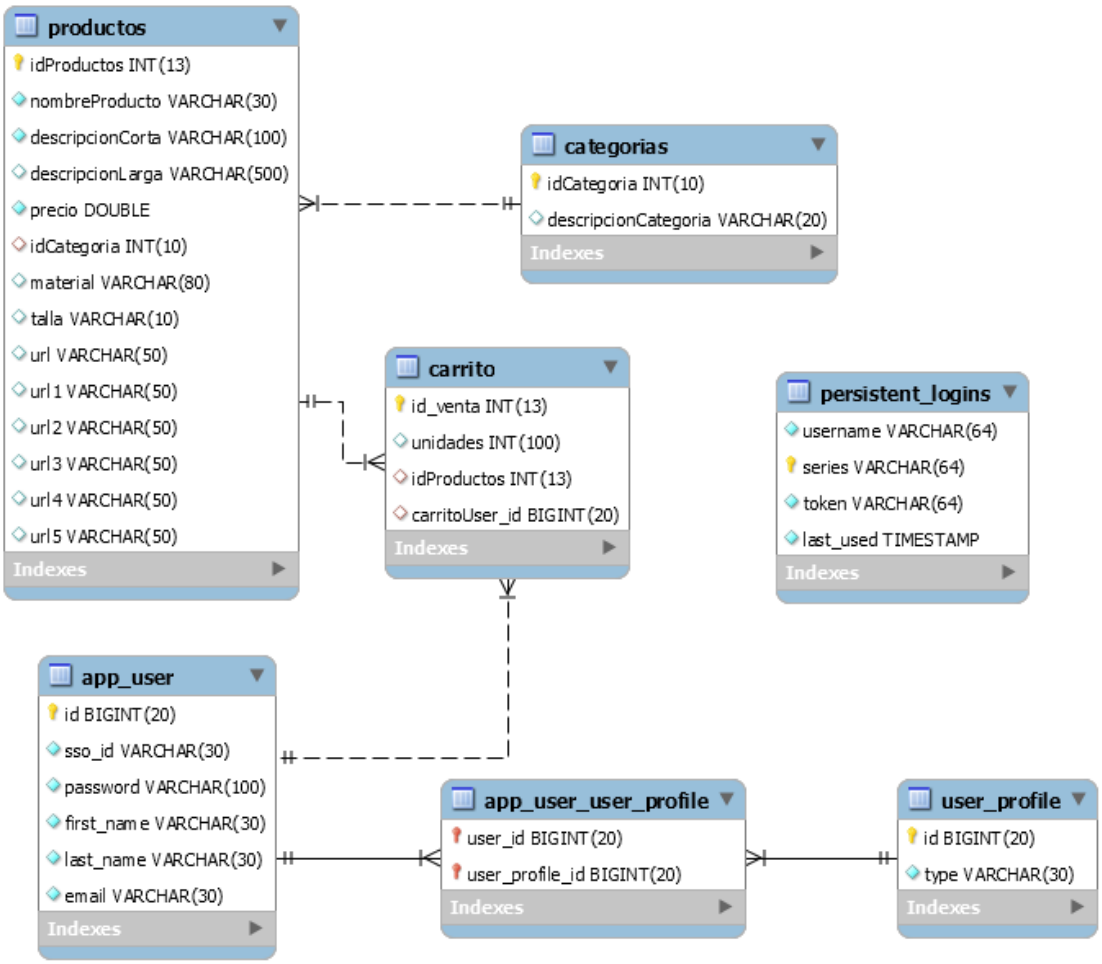


Figura 22. Modelo de base de datos

En la tabla productos se va a guardar todos los productos que vende la tienda. Tiene una relación de uno a muchos con la tabla categorías porque una categoría puede ser de muchos productos, pero un producto solo va a pertenecer a una categoría.

La tabla carrito se utiliza para ver los productos que existen en el típico carrito de la compra. Esta unida con la tabla productos y usuarios con una relación de uno a muchos ya que, un usuario puede tener muchos productos en el carrito, pero el usuario solo va a tener un carrito.

La tabla user_profile tiene guardados los tres tipos de roles que existen, administrador, empleado y usuario.

La tabla denominada app_user_user_profile une las tablas de user y user_profile con una relación de uno a muchos para que el usuario tenga un perfil asociado.

Por último, la tabla persisten_logins se utiliza en la web para guardar el token del usuario.

12.3. Paso a tablas

Las sentencias MySQL para pasar del diagrama entidad-relación a las tablas de la base de datos son las siguientes:

- **APP_USER**:

```
CREATE TABLE APP_USER (
  id BIGINT NOT NULL AUTO_INCREMENT,
  sso_id VARCHAR(30) NOT NULL,
  password VARCHAR(100) NOT NULL,
  first_name VARCHAR(30) NOT NULL,
  last_name  VARCHAR(30) NOT NULL,
  email VARCHAR(30) NOT NULL,
  PRIMARY KEY (id),
  UNIQUE (sso_id)
);
```

- **USER_PROFILE**:

```
CREATE TABLE USER_PROFILE(
  id BIGINT NOT NULL AUTO_INCREMENT,
  type VARCHAR(30) NOT NULL,
  PRIMARY KEY (id),
  UNIQUE (type)
);
```

- **APP_USER_USER_PROFILE**:

```
CREATE TABLE APP_USER_USER_PROFILE (
   user_id BIGINT NOT NULL,
   user_profile_id BIGINT NOT NULL,
   PRIMARY KEY (user_id, user_profile_id),
   CONSTRAINT FK_APP_USER FOREIGN KEY (user_id)
REFERENCES APP_USER (id),
   CONSTRAINT FK_USER_PROFILE FOREIGN KEY
(user_profile_id) REFERENCES USER_PROFILE (id)
);
```

- **PERSISTEN_LOGINS**:

```
CREATE TABLE PERSISTENT_LOGINS(
   username VARCHAR(64) NOT NULL,
   series VARCHAR(64) NOT NULL,
   token VARCHAR(64) NOT NULL,
   last_used TIMESTAMP NOT NULL,
   PRIMARY KEY (series)
);
```

- **CATEGORIAS**:

```
CREATE TABLE CATEGORIAS(
        idCategoria int(10) auto_increment primary key,
        descripcionCategoria varchar (20)
);
```

- **PRODUCTOS**:

```
CREATE TABLE PRODUCTOS(
   idProductos int(13) not null auto_increment,
   nombreProducto varchar(30) not null,
   descripcionCorta varchar(100) not null,
   descripcionLarga varchar(500),
   precio double not null,
   idCategoria int(10) DEFAULT 6,
   material varchar(80),
   talla varchar(10),
   url varchar(50),
   url1 varchar(50),
   url2 varchar(50),
   url3 varchar(50),
   url4 varchar(50),
   url5 varchar(50),
   PRIMARY KEY (idProductos),
   CONSTRAINT FK_ID_CATEGORIA FOREIGN KEY (idCategoria)
REFERENCES CATEGORIAS (idCategoria)
);
```

- **CARRITO**

```
CREATE TABLE CARRITO(
   id_venta int(13) not null auto_increment,
   unidades int(100),
   idProductos int(13),
   carritoUser_id BIGINT,
  PRIMARY KEY (id_venta),
CONSTRAINT FK_ID_PRODUCTOS FOREIGN KEY (idProductos)
REFERENCES PRODUCTOS (idProductos),
CONSTRAINT FK_ID_USER FOREIGN KEY (carritoUser_id)
REFERENCES APP_USER (id)
  );
```

13. Implementación

13.1 Configuración de la aplicación web

El primer paso para la realización de la aplicación web en Spring junto con Hibernate es crear el arquetipo del proyecto. Para realizar el arquetipo se ha creado un proyecto Maven.

Una vez creado el proyecto Maven el primer paso es indicar las dependencias en el archivo pom.xml. Las más importantes son:

```
<dependencies>
	<!-- Spring -->
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-core</artifactId>
		<version>${springframework.version}</version>
	</dependency>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-web</artifactId>
		<version>${springframework.version}</version>
	</dependency>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-webmvc</artifactId>
		<version>${springframework.version}</version>
	</dependency>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-tx</artifactId>
		<version>${springframework.version}</version>
	</dependency>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-orm</artifactId>
		<version>${springframework.version}</version>
	</dependency>
	<!-- MySQL -->
	<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<version>${mysql.connector.version}</version>
	</dependency>
</dependencies>
```

El siguiente paso es crear el archivo ApplicationContext o AppConfig. En él hay que indicarle que Spring tiene que escanear todo el proyecto o la parte que se requiera con la etiqueta @ComponentScan(basePackages = “com.tienda.springmvc”). Lo que va entre comillas son las carpetas de java de Spring indica que ahí es donde tiene que buscar los archivos de spring.

Hay que poner la etiqueta @Configuration y @EnableWebMvc para permitir la arquitectura Modelo-Vista-Controlador y con la de configuration indicamos que es un archivo de configuración.

Para que Spring reconozca los archivos .jsp hay que indicarlo con las siguientes líneas.

```
@Override
public void configureViewResolvers(ViewResolverRegistry registry) {

		InternalResourceViewResolver viewResolver = new
InternalResourceViewResolver();
		viewResolver.setViewClass(JstlView.class);
		viewResolver.setPrefix("/WEB-INF/views/");
		viewResolver.setSuffix(".jsp");
		registry.viewResolver(viewResolver);
}
```

Para los estáticos serán estas:

```
@Override

public void addResourceHandlers(ResourceHandlerRegistry registry) {

	registry.addResourceHandler("/static/**").addResourceLocations("/static/");
}
```

Para tener un archivo de mensajes que se muestren por pantalla hay que poner las siguientes líneas:

```
@Bean
public MessageSource messageSource() {
	ResourceBundleMessageSource messageSource = new
ResourceBundleMessageSource();
	messageSource.setBasename("messages");
	return messageSource;
}
```

13.1.1. Modelo

En la arquitectura MVC que está siguiendo este proyecto el modelo es la implementación de la base de datos para realizar consultas. Para realizar esto se ha utilizado hibernate así que en este capítulo vamos a ver parte del código de como configurar Hibernate en nuestro proyecto. La configuración se realizará mediante anotaciones.

Lo primero que hay que hacer para implementar Hibernate es colocar las dependencias en al fichero pom.xml. Que serán las siguientes:

```
<dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>4.3.11.Final</version>
</dependency>
<dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-validator</artifactId>
        <version>5.1.3.Final</version>
</dependency>
```

El siguiente paso es crear los Bean, mediante la anotación @Bean, necesarios para que encuentre el fichero de configuración de la base de datos y hay que instanciar el objeto sessionfactory y las transacciones para que pueda realizar consultas a la base de datos.

```
@Bean
public LocalSessionFactoryBean sessionFactory() {
      LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();
      sessionFactory.setDataSource(dataSource());
      sessionFactory.setPackagesToScan(new String[] { "com.tienda.springmvc.model"
});
      sessionFactory.setHibernateProperties(hibernateProperties());
      return sessionFactory;
}
@Bean
@Autowired
public HibernateTransactionManager transactionManager(SessionFactory s) {
     HibernateTransactionManager txManager = new HibernateTransactionManager();
     txManager.setSessionFactory(s);
     return txManager;
}
```

En el fichero de configuración se encontrará la URL de la base de datos, el usuario y contraseña, el driver de MySQL y algunas propiedades que nos interesen.

```
jdbc.driverClassName = com.mysql.jdbc.Driver
jdbc.url = jdbc:mysql://localhost:3306/discoverubeda
jdbc.username = usuario
jdbc.password = contraseña
hibernate.dialect = org.hibernate.dialect.MySQLDialect
hibernate.show_sql = true
hibernate.format_sql = true
```

La línea 1 corresponde a los Drivers de MySQL. Url, username y password corresponde como su nombre indica a la url donde se encuentra la base de datos, el usuario y la contraseña y las propiedades que nos han interesado poner han sido que cada vez que realice una consulta MySQL lo ponga en la consola mediante formato SQL y que se va a realizar las consultas con el dialecto de MySQL.

El siguiente paso es realizar las clases POJO del modelo de datos. Hibernate a cada tabla en la base de datos la llama entidad con la anotación @Entity.

Otras etiquetas utilizadas en este proyecto han sido:

- @Id: para indicar que el atributo es la clave primaria
- @GeneratedValue: cuando el valor es autoincremento
- @Column(name=”valor”): indica el nombre en la base de datos que tiene el atributo.
- @ManyToOne: cuando la relación es de muchos a uno
- @OneToOne: cuando la relación es de uno a uno
- @JoinColumn: para unir tablas mediante claves foráneas
- @Temporal: para el mapeo de fechas

Un ejemplo con estas anotaciones es la tabla carrito que quedaría así:

```
@Entity
@Table(name="CARRITO")
public class Carrito implements Serializable{

	private static final long serialVersionUID = 1L;

	@Id @GeneratedValue(strategy=GenerationType.IDENTITY)
	private int id_venta;

	@Column(name="unidades")
	private int unidades;

	@ManyToOne
	@JoinColumn(name="idProductos")
	private Productos productos;

	@OneToOne
	@JoinColumn(name="carritoUser_id")
	private User user;


	public Carrito() {

	}

	public Carrito(int id_venta, int unidades, Productos productos) {
		super();
		this.id_venta = id_venta;
		this.unidades = unidades;
		this.productos = productos;
	}

	public int getId_venta() {
```

```
		return id_venta;
	}

	public void setId_venta(int id_venta) {
		this.id_venta = id_venta;
	}

	public int getUnidades() {
		return unidades;
	}

	public void setUnidades(int unidades) {
		this.unidades = unidades;
	}

	public Productos getProductos() {
		return productos;
	}

	public void setProductos(Productos productos) {
		this.productos = productos;
	}

	public User getUser() {
		return user;
	}

	public void setUser(User user) {
		this.user = user;
	}

}
```

Como esta clase esta realizado para todas las tablas que existen en la base de datos.

El siguiente paso es crear los objetos DAO, Data Access Object, en español objetos de acceso a datos. Estos objetos son los que interactúan con la base de datos y realizan consultas en esta. Veamos un ejemplo.

```
public List<Productos> listarProductosCategorias(int idCategoria) {


		Criteria criteria = createEntityCriteria().addOrder(Order.asc("nombreProducto")).add(Restrictions.eq("categoria.idCategoria", idCategoria));

		List<Productos> productos = (List<Productos>) criteria.list();

		return productos;
```

```
}
```

Aquí devuelve una lista de productos cuando el id de la categoría sea el mismo que el que se le pasa.

```
public int maxIdProductos() {

		Criteria criteria = createEntityCriteria().setProjection(Projections.max("idProductos"));

		int maximo = (int) criteria.uniqueResult();

		return maximo;

}
```

En este método se obtiene el número máximo de productos que existen en la base de datos.

En todas las clases DAO hay que poner la anotación @Repository(“nombreDeLaClase”) para indicar que se van a crear objetos DAO.

Por último, para llamar a los DAO se necesitan los servicios. Los servicios son clases de Java que llaman a los métodos de los DAO. Los servicios utilizan la anotación @Service(“nombreDeLaClase”) y con la anotación @Autowired instanciamos los objetos DAO dentro de los servicios.

A continuación, se mostrará un ejemplo de cómo llamar a un objeto DAO.

```
@Autowired
private ProductosDao productosDao;
public List<Productos> listarProductosCategorias(int idCategoria){
		return productosDao.listarProductosCategorias(idCategoria);
}
public int maxIdProductos() {
		return productosDao.maxIdProductos();
}
```

13.1.2 Vista

La vista es lo que ve el usuario, es decir la interfaz gráfica de la aplicación. Esta se realiza mediante archivos con extensión .jsp.

Los archivos .jsp son archivos que se escriben en HTML y pueden utilizar CSS, JavaScript, JQuery, etc. Estos archivos son generados por el servidor y esté los convierte a archivos HTML por lo que el usuario simplemente visualiza un archivo HTML junto con sus estilos y funcionalidades.

Para enviar y recoger información de la vista se va a explicar en qué consiste el atributo o anotación ModelAttribute. Este atributo se puede utilizar en tres casos bien diferenciados.

- Para enviar información a la vista. Por ejemplo:

```
public String listUsers(ModelMap model) {

        List<User> users = userService.findAllUsers();
        model.addAttribute("users", users);
        model.addAttribute("loggedinuser", getPrincipal());
        return "userslist";
}
```

En estas líneas de código vemos que users es una lista con todos los usuarios que se encuentran en la base de datos y se pasa a la vista mediante model.addAttribute. Hay que declarar el model en el argumento del método con ModelMap para poder utilizarlo.
Estas líneas de código se escriben en el controlador que es el que envía la información a la vista.

- Cómo argumento en los métodos como por ejemplo recoger información de formularios. En el siguiente ejemplo se puede observar como recoger la información de un profesor.

```
public String submit(@ModelAttribute("profesor") Profesor profesor) {
                return "vistaProfesor";
}
```

- En los formularios. En este caso cuando tenemos un objeto por ejemplo Productos con modelAttribute comprueba que los parámetros que se cogen son solo de la clase producto.

```
<form:form method="POST" modelAttribute="productos"
action="insertarProductos?${_csrf.parameterName}=${_csrf.token}"
enctype="multipart/form-data" class="form-horizontal">
        <div class="form-group">
                <label class="control-label col-sm-4"
for="nombreProducto">Nombre del producto:</label>
    <div class="col-sm-4">
                <form:input type="text" class="form-control"
placeholder="Nombre del producto" path="nombreProducto" />
```

```
            </div>
        </div>
        <div class="form-group">
            <label class="control-label col-sm-4"
for="descripcionCorta">Descripción corta:</label>
            <div class="col-sm-4">
                <form:input type="text" class="form-control" placeholder="Descripción
corta" path="descripcionCorta" />
            </div>
        </div>
</form:form>
```

En la declaración del formulario se pone de que clase son los datos que se van a pasar en este caso es la clase productos modelAttribute=″productos″ y con la etiqueta path se hace referencia a los atributos de la clase productos. En este ejemplo se hace referencia a nombre del producto y la descripción corta path="nombreProducto"y path="descripcionCorta.

Hemos visto como recoger información y como enviarla, pero, ¿cómo se muestra la información a los usuarios? Para mostrar los datos en la vista se utilizan con las etiquetas de la tecnología JSTL, JavaServer Pages Standard Tag Library.

Para que se comprenda vamos a poner como se muestran todos los usuarios.

```
<c:forEach items="${users}" var="user">
    <tr>
        <td>${user.firstName}</td>
        <td>${user.lastName}</td>
        <td>${user.email}</td>
        <td>${user.ssoId}</td>
        <td><a href="<c:url value='/edit-user-${user.ssoId}' />"
class="btn btn-success custom-width">edit</a></td>
        <td><a href="<c:url value='/delete-user-${user.ssoId}' />"
class="btn btn-danger custom-width">delete</a></td>

    </tr>

</c:forEach>
```

Con el bucle for each recorrer todo la lista de usuarios y mediante ${clase.atributo} muestra la información deseada.

13.1.3. Controlador

Como se ha comentado en otros capítulos, el controlador es quien gestiona el modelo y las vistas. A continuación, se mostrará con ejemplos la implementación de este proyecto.

Para definir una URL se utiliza la etiqueta @RequestMapping. En ella hay que poner / y la URL deseada y el método deseado, POST o GET. Por ejemplo

```
@RequestMapping(value = "/login", method = RequestMethod.GET)
	public String loginPage() {
		if (isCurrentAuthenticationAnonymous()) {
			return "login";
		} else {
			return "redirect:/list";
		}
}
```

La etiqueta siempre va delante de un método ya que sirve para llamar al método en este caso si ponemos /login llamamos al método loginPage con el método GET. Este método comprueba si hay alguien registrado o no. Si hay alguien registrado le devuelve la lista de usuarios si no aparecerá la pantalla para iniciar sesión.

Para mostrar la vista hay que poner return y entre comillas el nombre de la vista sin la extensión. Por ejemplo, **return** "login"; devuelve la vista que tiene el nombre login.

Para llamar a los servicios explicados en el capítulo 13.1.1 se utiliza la etiqueta @Autowired este crea una instancia del bean estos objetos. A continuación, se comentará un ejemplo.

```
@Autowired
ProductosService productosService;

@RequestMapping(value= { "/listarCategorias-{descripcion}" }, method = RequestMethod.GET)
	public String listadoCategorias(@PathVariable String descripcion,ModelMap model)
{

		//Buscamos el id de la categoria elegida
		int idCategoria = categoriasService.findByDescripcion(descripcion);
		//Sacamos los productos que tengan esa categoria
		List<Productos> listadoProductos = productosService.listarProductosCategorias(idCategoria);

		model.addAttribute("listado", listadoProductos);

		return "listadoProductos";
}
```

Cuando se pone @PathVariable quiere decir que en la url estamos pasando algo que nos interesa recoger de la vista, en este caso es la descripción. Despues se pasa la descripción por el método para encontrar el id de la categoría que ha elegido el usuario en la vista. Una vez encontrado los productos se pasan a la vista denominada listadoProductos.jsp.

Por último, indicar que spring reconoce a la clase controlador con la anotación @Controller

13.1.4. Implementación de Paypal

Implementar el botón de comprar ahora de Paypal es muy sencillo. Simplemente hay que poner el siguiente formulario

```
<form action="https://www.sandbox.paypal.com/cgi-bin/webscr" method="post" target="_top">
	<input type="hidden" name="cmd" value="_xclick">
	<input type="hidden" name="business" value="ntj00003-buyer@red.ujaen.es">
	<input type="hidden" name="lc" value="ES">
	<input type="hidden" name="item_name" value="hola">
	<input type="hidden" name="item_number" value="5">
	<input type="hidden" name="amount" value="${precio}">
	<input type="hidden" name="currency_code" value="EUR">
	<input type="hidden" name="button_subtype" value="services">
	<input type="hidden" name="no_note" value="0">
	<input type="hidden" name="cn" value="Añadir instrucciones especiales para el vendedor:">
	<input type="hidden" name="no_shipping" value="2">
	<input type="hidden" name="tax_rate" value="2.000">
	<input type="hidden" name="shipping" value="5.00">
	<input type="hidden" name="bn" value="PP-BuyNowBF:btn_buynowCC_LG.gif:NonHosted">
	<input type="image" src="https://www.sandbox.paypal.com/es_ES/ES/i/btn/btn_buynowCC_LG.gif" border="0" name="submit" alt="PayPal, la forma rápida y segura de pagar en Internet.">
	<img alt="" border="0" src="https://www.sandbox.paypal.com/es_ES/i/scr/pixel.gif" width="1" height="1">
</form>
```

Se va a explicar algunos parámetros importantes del formulario.

- Amount: es el precio total.
- Item_name: el nombre del pedido
- Item_number: el id del pedido
- business: a que correo va dirigido el dinero
- currency_code: moneda que se utiliza para comprar
- tax_rate: taxa de impuestos
- shipping: gastos de envio

13.1.5. Configuración de Spring Security

Para finalizar la configuración del proyecto se va a explicar cómo implementar Spring Security.

1. El primer paso es insertar las dependencias en el archivo pom.xml.

```
<dependency>
```

```
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-web</artifactId>
            <versión>4.0.4.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-config</artifactId>
            <versión>4.0.4.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-taglibs</artifactId>
            <version>4.0.4.RELEASE<version>
        </dependency>
```

2. A continuación, se crean unos filtros conocidos como springSecurityFilterChain que son responsables de toda la seguridad de tu aplicación como por ejemplo proteger las URL y validar los usuarios y contraseñas, etc.

```
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {
@Bean
        public DaoAuthenticationProvider authenticationProvider() {
            DaoAuthenticationProvider authenticationProvider = new
DaoAuthenticationProvider();

        authenticationProvider.setUserDetailsService(userDetailsService);
            authenticationProvider.setPasswordEncoder(passwordEncoder());
            return authenticationProvider;
        }
}
```

3. A parte del código de arriba tenemos que registrar los filtros, en la arquitectura MVC se realiza así:

```
import
org.springframework.security.web.context.AbstractSecurityWebApplicationIniti
alizer;

public class SecurityWebApplicationInitializer extends
AbstractSecurityWebApplicationInitializer {



}
```

4. Spring requiere que todos los usuarios sean autenticados y admite autenticación basada en formularios. Esto lo hace a través de las siguientes líneas de código:

```
@Override
    protected void configure(HttpSecurity http) throws Exception {
```

```
http.authorizeRequests()
                .antMatchers("/carrito").access("hasRole('ADMIN') or
hasRole('EMPLEADOS') or hasRole('USUARIOS')")
                .antMatchers("/list").access("hasRole('ADMIN') or
hasRole('EMPLEADOS')")
                .antMatchers("/logout").access("hasRole('ADMIN') or
hasRole('EMPLEADOS')")

        .antMatchers("/insertarProductos").access("hasRole('ADMIN') or
hasRole('EMPLEADOS')")

        .antMatchers("/insertarArchivos").access("hasRole('ADMIN') or
hasRole('EMPLEADOS')")
                .antMatchers("/comprarProducto-
*").access("hasRole('ADMIN') or hasRole('EMPLEADOS') or hasRole('USER')")
                .antMatchers("/edit-user-*")
                .access("hasRole('ADMIN') or
hasRole('EMPLEADOS')").and().formLogin().loginPage("/login")

        .loginProcessingUrl("/login").usernameParameter("ssoId").passwordParameter("
password").and()
                .rememberMe().rememberMeParameter("remember-
me").tokenRepository(tokenRepository)

        .tokenValiditySeconds(86400).and().csrf().and().exceptionHandling().accessDen
iedPage("/Access_Denied");

        }
```

Estas líneas aseguran que cualquier solicitud a nuestra aplicación requiere que el usuario sea autenticado, permitiendo a los usuarios autenticarse con el inicio de sesión basado en formulario y autenticación HTTP básica.

Explicaremos a continuación lo que quiere decir cada cosa.

- authorizeRequests() .antMatchers("/carrito") indica que para la url carrito se van a poner unas restricciones
- .access indica que rol puede acceder a la url
- .formLogin().loginPage indica que los usuarios tienen autenticación básica de HTTP mediante formulario. Este método permite dar acceso a todos los usuarios para todas las URLS de nuestra aplicación a través de formularios.
- .rememberMe().rememberMeParameter("remember-tokenRepository(tokenRepository) se utiliza para recordar la contraseña.
- .tokenValiditySeconds(86400).and().csrf().and().exceptionHandling().accessDen iedPage("/Access_Denied") se utiliza para cuando han pasado esos segundo que la sesión termine automáticamente.

A continuación, se muestra un formulario que utiliza la autenticación básica de Spring Security.

```
<c:url var="loginUrl" value="/login" />
        <form action="${loginUrl}" method="post">
        <c:if test="${param.error != null}">
                <div class="alert alert-danger">
                        <p>Invalid username and password.</p>
                </div>
        </c:if>
        <c:if test="${param.logout != null}">
                <div class="alert alert-success">
                        <p>You have been logged out successfully.</p>
                </div>
        </c:if>
                <div class="form-group">
                        <label for="exampleInputEmail1">Dirección de correo</label>
                        <input type="text" class="form-control" id="username"
name="ssoId" placeholder="Enter Username" required>
                </div>
                <div class="form-group">
                        <label for="password">Dirección de correo</label>
                        <input type="password" class="form-control" id="password"
name="password" placeholder="Enter contraseña" required>
                </div>
                <div class="input-group input-sm">
                <div class="form-check">
```

```
                <input type="checkbox" class="form-check-input" id="rememberme"
name="remember-me">

                <label class="form-check-label" for="rememberme">Recordar mis
datos</label>

              </div>

            </div>

            <input type="hidden" name="${_csrf.parameterName}"

                                                value="${_csrf.token}" />
            <div class="form-actions">

                    <input type="submit"
      class="btn btn-block btn-primary btn-default" value="Acceder">

            </div>

      </form>
```

En este formulario una petición POST intentará autenticar al usuario. Si el parámetro da error es que la autenticación fallo si por el contrario el parámetro da distinto de nulo es que el usuario se desconectó correctamente.

El nombre de usuario y contraseña se tienen que llamar sí o sí username y password respectivamente.

Con el token se recuerdan los datos autenticados durante un tiempo determinado.

5. Faltan por especificar unos vean.

El siguiente vean especifica el método de encriptado a la base de datos, en este caso será Sha-1

```
@Bean
public ShaPasswordEncoder passwordEncoder() {
      return new ShaPasswordEncoder();

}
```

Para recordar los datos se sesión se realiza de la siguiente manera:

```
@Bean
public PersistentTokenBasedRememberMeServices
getPersistentTokenBasedRememberMeServices() {
            PersistentTokenBasedRememberMeServices tokenBasedservice = new
PersistentTokenBasedRememberMeServices(
                        "remember-me", userDetailsService, tokenRepository);
```

```
        return tokenBasedservice;

    }
```

Y con este se indica si el usuario es anónimo o no, es decir, si ya está autenticado o no.

```
    @Bean
    public AuthenticationTrustResolver getAuthenticationTrustResolver() {
        return new AuthenticationTrustResolverImpl();

    }
```

13.2. Configuración de la aplicación móvil

EL primer paso para la configuración de la aplicación móvil es descargarse e instalarse de la página oficial, el framework elegido para realizar la aplicación móvil. La url es la siguiente https://phonegap.com/getstarted/.

También hay que descargarse la aplicación en el smartphone que se vaya a utilizar para realizar las pruebas.

Una vez instalado en ambos dispositivos se creará un proyecto nuevo desde el ordenador. Como muestra la siguiente imagen.

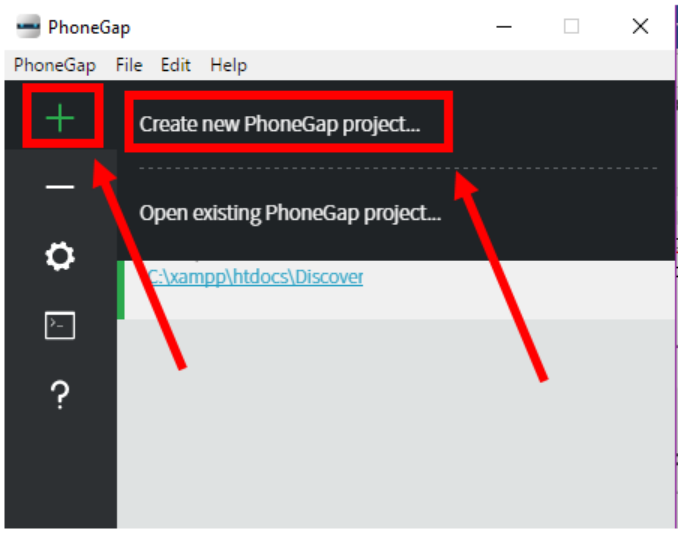


Figura 23. Creación de un proyecto en Phonegap

En el siguiente paso hay que rellenar donde se quiere guardar el proyecto, hay que ponerle un nombre y un id parecido al de Spring.

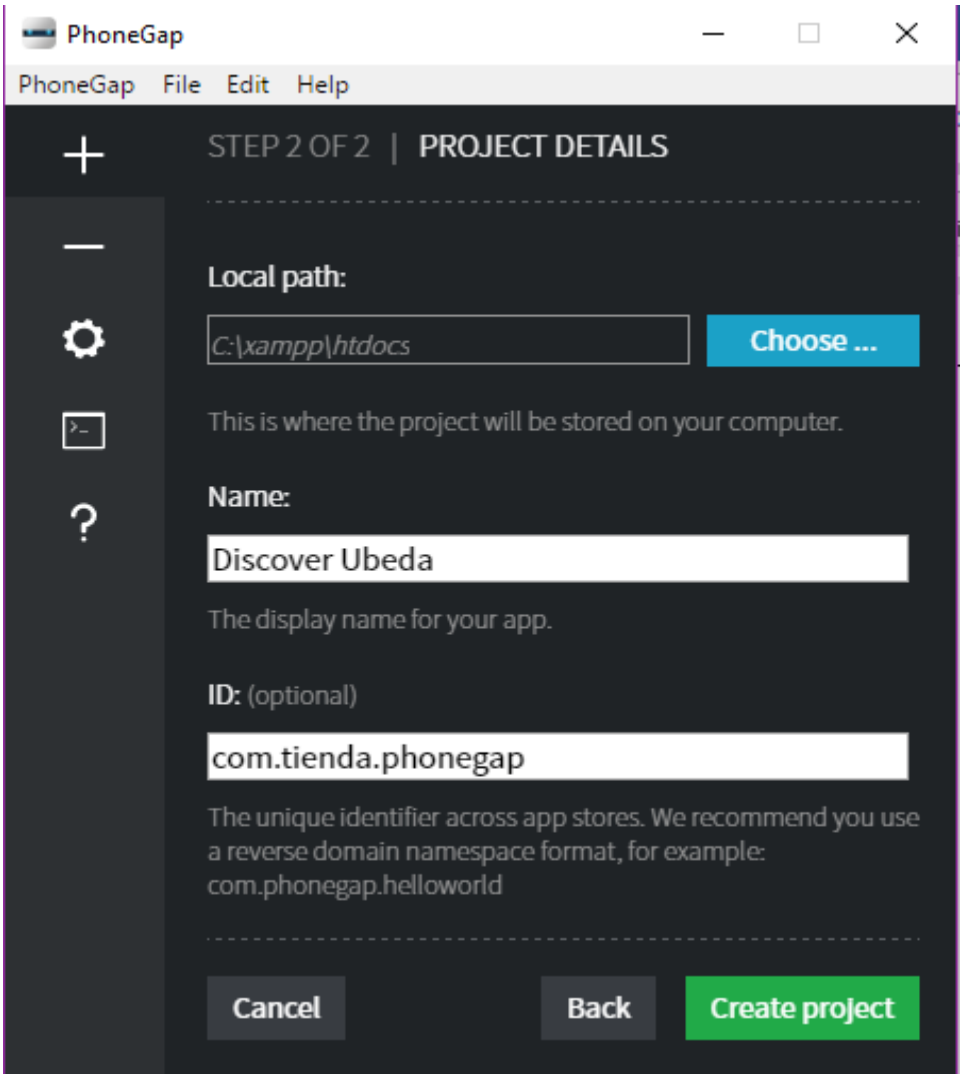


Figura 24. Datos para la creación de un proyecto en Phonegap

Al pinchar en create proyect, como podemos observar en la imagen de arriba, Phonegap creará los siguientes directorios del proyecto:

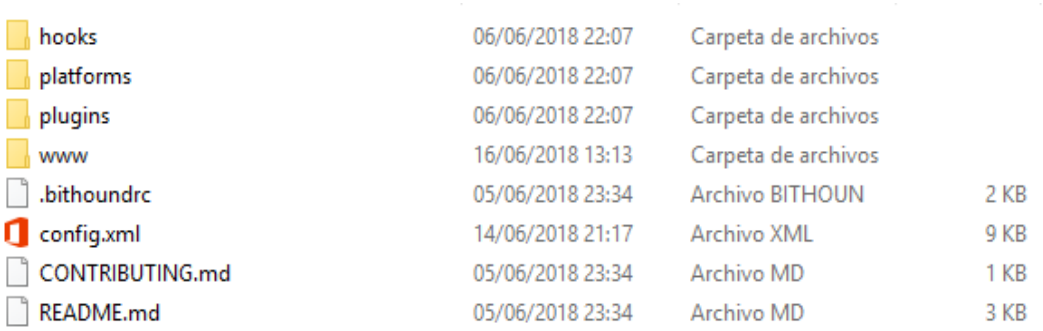


Figura 25. Directorio de un de Phonegap

En la carpeta www es donde hay que poner nuestros archivos HTML, PHP y JS para que se ejecuten.

Para iniciar la aplicación necesitamos tener un archivo llamado index.html en la ruta citada anteriormente ya que es el primer archivo que coge Phonegap.

En el móvil se tiene que conectar poniendo la dirección IP que tiene el ordenador como muestra la ilustración de abajo.

Figura 26. Inicio de aplicación desde un smartphone

Por último, para que todo funcione el móvil y el ordenador deberán estar conectados a la misma red local.

A continuación, se va a explicar cómo listar los productos. Primero se hará un archivo php que se conecte a la base de datos y obtenga los datos en formato JSON.

```
<?php
header("Cache-Control: no-store, no-cache, must-revalidate, max-age=0");
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache");
$conexion = mysqli_connect('dirección IP del servidor','user','password','base de datos');
mysqli_set_charset($conexion, 'utf8');
if(!$conexion){
	die('La base de datos no se ha podido conectar por: '.mysqli_connect_error());
}
$consulta = "SELECT * FROM PRODUCTOS";
$sql = mysqli_query($conexion,$consulta);
```

```
$datos= array();
while ($obj = mysqli_fetch_object($sql)) {
        $datos[] = array('nombreProducto' => $obj->nombreProducto,
                                        'descripcionCorta' => $obj->descripcionCorta,
                                        'url' => $obj->url,
                                        'precio' => $obj->precio,
                                        'idProductos' => $obj->idProductos,

        );
}
echo " . json_encode($datos) . ";
mysqli_close($conexion);
header('Content-type: application/json');
header("Access-Control-Allow-Origin: *");
?>
```

Primero establecemos la conexión a la base de datos con mysqli_connect. Esta conexión deberemos ponera a utf8 con mysqli_set_charset para que nos reconozca todas las tildes. Realizamos la consulta a la base de datos y obtenemos los datos con un array. Mediante json_encode($datos) los datos se transforman solos a formato JSON.

Para recoger estos datos se ha utilizado JavaScript.

```
var url = 'http://dirección IP del servidor/nombreArchivo.php';
xhttp = new XMLHttpRequest();
xhttp.open("GET", url, true);
xhttp.send();
xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4) {
     if (xhttp.status == 200) {
            var data = JSON.parse(xhttp.responseText);
                if (data!=''){
                        var tam = data.length;
                        var iter;
                        var html='<div class="contenedorProductos">';
                        for (iter=0; iter<tam; iter++){
                                var nombreProducto = data[iter].nombreProducto;
                                var descripcionCorta = data[iter].descripcionCorta;
                                var url = data[iter].url;
                                var precio = data[iter].precio;
                                var idProductos = data[iter].idProductos;
                                var html_user = '<div class="w3-cell style="width:30%"
">'+
                                                        '<img style="width:100%""
src="http:188.166.174.201/'+url+'">'+
                                                        '</div>'+
                                                        '<div class="w3-cell w3-
container">'+
                                                        '<h3>'+nombreProducto+'</h3>'+
                                                        '<p>'+descripcionCorta+'</p>'+
                                                        '<p>'+precio+'€</p>'+
```

```
                                                    '<input type="button" class="btn
btn-danger" value="Ver más" onclick="verProducto('+idProductos+');">'+
                                                        '</div>'+
                                                        '<hr>';

                            var html2 = '</div>';

                            html=html+html_user+html2;

                        }//Fin for

                html1 = '<footer class="w3-container w3-theme"><a
href="index.html">'+
                            '<h3 class="text-center">Volver al inicio</h3>'+
                                '</a></footer>';
                        html = html + html1;

                }//fin if(data!='')
     }//fin if(xhttp.status == 200)
    }//fin xhttp.readyState == 4)
    document.getElementById("listadoProductos").innerHTML = html;
}
```

Con este código recogemos los datos en formato JSON y los mostramos mediante lenguaje HTML.

Ahora se va a proceder a ver un ejemplo de un formulario en HTML y recoger los datos mediante el método POST en PHP.

```
<form action="http://dirección IP del servidor/nombreArchivo.php" method="post">
        <div class="form-group">
        <label class="control-label col-sm-4" for="sso_id">Nombre de usuario:</label>
        <div class="col-sm-4">
                        <input type="text" class="form-control" placeholder="Nombre de
usuario" name="sso_id"/>
                </div>
        </div>
        <div class="form-group">
     <label class="control-label col-sm-4" for="first_name">Primer apellido</label>
     <div class="col-sm-4">
                        <input type="text" class="form-control" placeholder="Primer
apellido" name="first_name" />
     </div>
   </div>
   <div class="form-group">
     <label class="control-label col-sm-4" for="last_name">Segundo apellido</label>
     <div class="col-sm-4">
                        <input type="text" class="form-control" placeholder="Segundo
apellido" name="last_name" />
     </div>
   </div>
```

```
   <div class="form-group">
    <label class="control-label col-sm-4" for="email">Correo electrónico</label>
    <div class="col-sm-4">
                        <input type="email" class="form-control" placeholder="Correo
electrónico" name="email" />
    </div>
   </div>
   <div class="form-group">
    <label class="control-label col-sm-4" for="password">Contraseña</label>
    <div class="col-sm-4">
                        <input type="password" class="form-control"
placeholder="Contraseña" name="password"/>
    </div>
   </div>
   <div class="form-group">
    <div class="col-sm-offset-7 col-sm-10">
     <button type="submit" class="btn btn-default">Enviar</button>
     <button type="reset" class="btn btn-default">Limpiar</button>
        </div>
   </div>
</form>
```

Y el archivo php en cuestión es el siguiente:

```
<?php
header("Cache-Control: no-store, no-cache, must-revalidate, max-age=0");
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache");
        $conexion =
mysqli_connect('188.166.174.201','Noelia','soytonta468','discoverubedapruebas');
        mysqli_set_charset($conexion, 'utf8');
        if(!$conexion){
                die('La base de datos no se ha podido conectar por: '.mysql_error());
        }

        //Nombres variables con metodo POST
        $sso_id = $_POST['sso_id'];
        $first_name = $_POST['first_name'];
        $last_name = $_POST['last_name'];
        $email = $_POST['email'];
        $password = sha1($_POST['password']);
        $sqlInsert = "INSERT INTO APP_USER
(sso_id,first_name,last_name,email,password)
VALUES('$sso_id','$first_name','$last_name','$email','$password')" or
die(mysql_error());
        mysqli_query($conexion,$sqlInsert);
        $resultado = "SELECT id FROM APP_USER WHERE sso_id = '".$sso_id."' " or
die(mysql_error());


        mysqli_query($conexion,$resultado);
        $result = mysqli_query($conexion, $resultado);
```

```
        if (mysqli_num_rows($result) > 0) {
          while($row = mysqli_fetch_assoc($result)) {
            //echo "ID: " . $row["id"]. "<br>";
            $res = $row["id"];
          }
        } else {
          echo "0 results";
        }
          $sql = "INSERT INTO APP_USER_USER_PROFILE (user_id,user_profile_id)
VALUES($res,'1')" or die(mysql_error());
          mysqli_query($conexion,$sql);
          header('Location: http://188.166.174.201/Discover/www/index.html');
          //Cierro la conexión
          mysqli_close($conexion);
?>
```

14. Casos de uso

Los casos de uso en ingeniera del software son la secuencia de acciones que hay que llevar a cabo en un sistema para producir una acción.

En este capítulo se va a proceder a mostrar mediante casos de uso las diferentes funcionalidades que se pueden realizar con la aplicación web y la aplicación móvil.

Caso de uso I – Búsqueda de productos	
Actor primario	Usuario
Actores participantes	Base de datos y usuario
Condiciones previas	Ninguna
Acciones o flujo	1. El usuario entra en la aplicación 2. El usuario acceder a la URL para donde se encuentran todos los producots 3. El usuario escribe el nombre de un producto
Resultado	Se muestran los productos que se llamen con el nombre que se ha insertado

Tabla 5. Caso de uso I

Caso de uso II – Listado de productos por categorías	
Actor primario	Usuario
Actores participantes	Base de datos y usuario
Condiciones previas	Ninguna
Acciones o flujo	4. El usuario entra en la aplicación 5. El usuario acceder a la URL para donde se encuentran las categorías 6. El usuario selecciona una categoría
Resultado	Se muestran todos los productos de una categoría en el navegador

Tabla 6. Caso de uso II

Caso de uso III – Listado de productos	
Actor primario	Usuario
Actores participantes	Base de datos y usuario
Condiciones previas	Ninguna
Acciones o flujo	7. El usuario entra en la aplicación 8. El usuario acceder a la URL para listar los productos
Resultado	Se muestran todos los productos en el navegador

Tabla 7. Caso de uso III

Caso de uso IV – Insertar productos	
Actor primario	Usuario

Actores participantes	Base de datos y usuario
Condiciones previas	EL usuario está registrado en la aplicación como admin o empleado
Acciones o flujo	1. El usuario accede a la URL para insertar productos 2. El usuario rellena todos los datos que le pide el formulario 3. El usuario envía el formulario
Resultado	Se ha insertado el producto

Tabla 8. Caso de uso IV

Caso de uso V – Crear cuenta	
Actor primario	Usuario
Actores participantes	Base de datos y usuario
Condiciones previas	EL usuario no está registrado en la base de datos
Acciones o flujo	4. El usuario entra en la aplicación 5. El usuario acceder a la URL para crear una cuenta 6. El usuario rellena todos los datos que le pide el formulario 7. El usuario envía el formulario
Resultado	El usuario se ha registrado en la aplicación

Tabla 9. Caso de uso V

Caso de uso VI – Inicio de sesión	
Actor primario	Usuario
Actores participantes	Base de datos y usuario
Condiciones previas	EL usuario no está logueado en la aplicación
Acciones o flujo	1. El usuario entra en la aplicación 2. El usuario acceder a la URL para iniciar sesión 3. El usuario escribe su usuario y contraseña en el formulario que aparece 4. El usuario envía el formulario
Resultado	El usuario ha iniciado sesión en la aplicación

Tabla 10. Caso de uso VI

Caso de uso VII – Carrito de la compra	
Actor primario	Usuario
Actores participantes	Base de datos y usuario
Condiciones previas	El usuario tiene que estar logueado en la aplicación
Acciones o flujo	1. El usuario entra en la aplicación

	2. El usuario acceder a la URL donde se encuentran los productos 3. El usuario selecciona añadir al carrito
Resultado	El producto se ha añadido al carrito

Tabla 11. Caso de uso VII

Caso de uso VIII – Compra de productos	
Actor primario	Usuario
Actores participantes	Base de datos y usuario
Condiciones previas	Tener productos en el carrito y estar logueado en la aplicación
Acciones o flujo	1. El usuario entra en la aplicación 2. El usuario acceder a la URL del carrito de la compra 3. El usuario selecciona el botón de paypal de comprar 4. El usuario paga el dinero correspondiente a través de paypal
Resultado	Se ha comprado correctamente

Tabla 12. Caso de uso VIII

15. Conclusiones y áreas de mejora

Han sido unos meses duros pero una vez finalizado y realizado todo el proyecto cualquier persona involucrada estará orgulloso/a de sí mismo/a de ver que el proyecto ha conseguido terminar bien.

Es una gran satisfactoria poder elegir un proyecto que guste y se sienta interés de verdad por realizarlo, eso es mi importante. Nadie se esfuerza lo mismo a realizar algo que le gusta o realizar algo que no gusta.

Este proyecto ha abierto una buena puerta para salir al mercado y buscar un trabajo ya que, las ofertas de trabajo ahora mismo para Java y Spring están muy elevadas.

Ahora solo queda esperar que la aplicación consiga atraer a más personas y aumentar los beneficios de la tienda.

Por supuesto la aplicación no se puede quedar como esta. Se proponen las siguientes mejoras:

- Permitir que los usuarios que anteriormente hayan comprado algún producto de la tienda puedan poner comentarios sobre el estado del producto que compraron.
- Al pagar por Internet realizar una factura para que el cliente este más seguro y certifique que ha comprado productos.
- Como el nicho de mercado para esta tienda son turistas una idea es ponerla en inglés por la gran cantidad de ingleses que vienen a este pueblo, Úbeda.
- A traer a los usuarios mediante ofertas y descuentos
- Realizar notificaciones para avisar a los usuarios de cosas como ofertas, actualizaciones en la página, productos nuevos, etc.

Todo este proyecto ha sido posible gracias al gran avance de la tecnología en los últimos años.

En definitiva, he aprendido mucho con este proyecto y seguiré esforzándome porque salga hacia adelante.

16. Bibliografía

[1] https://www.arsys.es/blog/programacion/diseno-web/que-es-phonegap/

[2] https://prezi.com/zcsejbxmrjgm/modelo-vista-controlador/

[3] https://codigofacilito.com/articulos/mvc-model-view-controller-explicado

[4] http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/122

[5] https://www.tutorialspoint.com/spring/spring_overview.htm

[6] http://www.davidvalverde.com/blog/inversion-de-control/

[7] https://docs.spring.io/spring/docs/5.1.0.BUILD-SNAPSHOT/spring-framework-reference/core.html#aop

[8] http://cursohibernate.es/doku.php?id=unidades:01_introduccion_orm:06_intro_orm

[9] https://docs.spring.io/spring-security/site/docs/5.0.5.RELEASE/reference/htmlsingle/#getting-started

[10] https://www.youtube.com/watch?v=LG4GjdPXqz0

[11] http://php.net/manual/es/intro-whatis.php

[12] https://www.arsys.es/blog/programacion/diseno-web/que-es-phonegap/

[13] https://jmacuna.tecnoblog.guru/2017/03/sistemas-operativos-moviles.html

Anexo I

Manual de usuario de la aplicación web

En este anexo se va a proceder a ofrecer al lector una guía básica con las acciones que puede realizar en la aplicación web.

Página principal

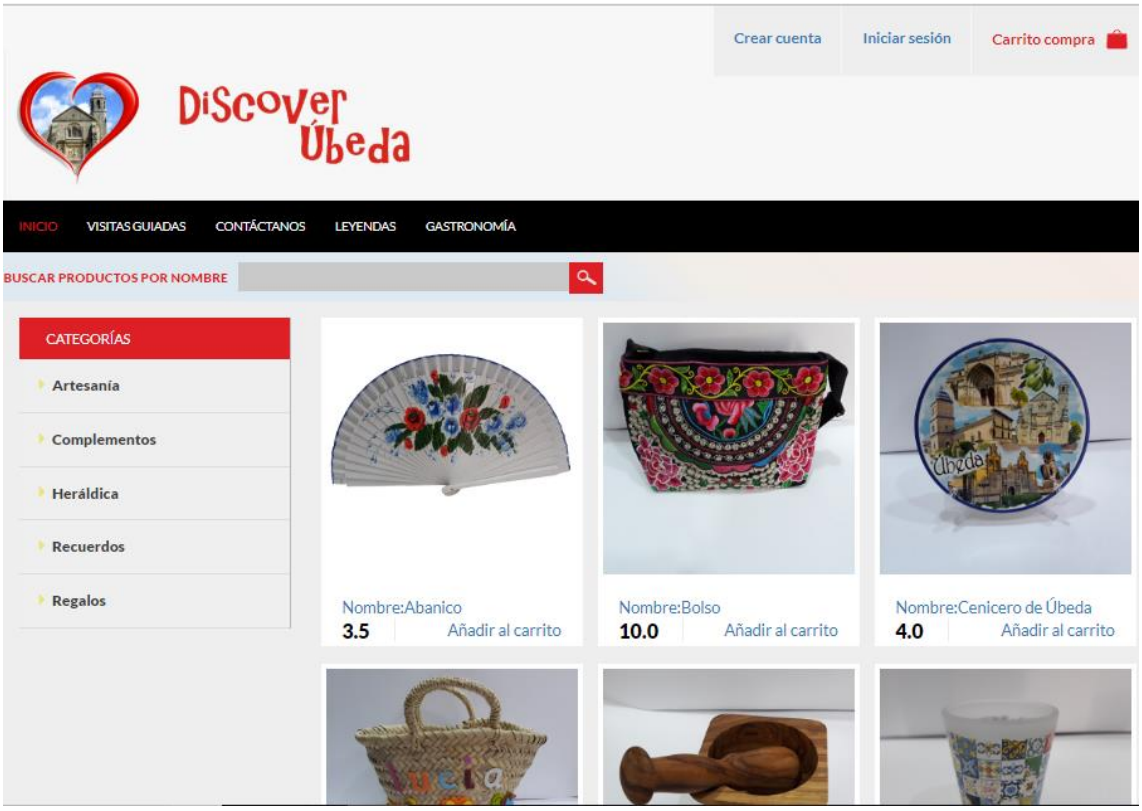


Figura 26. Página principal

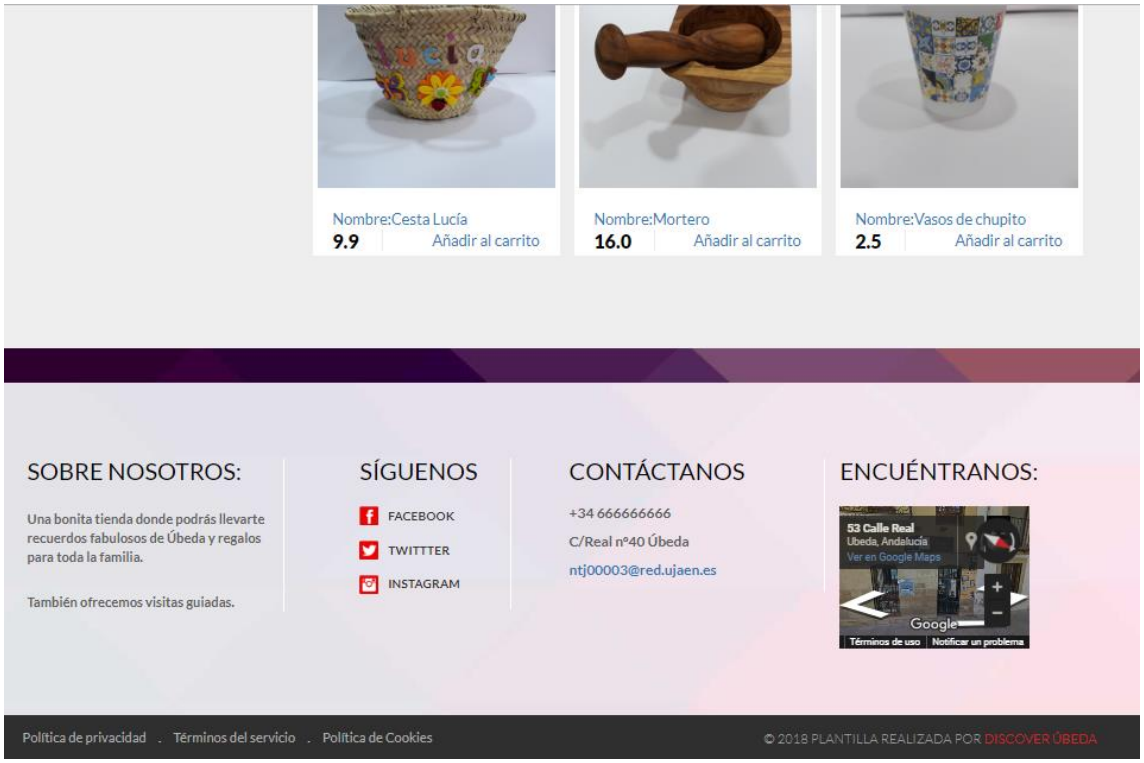


Figura 27. Página principal pie de página

Como se puede observar en la página principal aparecen el listado de los productos existentes de todas las categorías.

Arriba a la derecha aparece un menú para iniciar sesión, crear cuenta y ver el carrito de la compra.

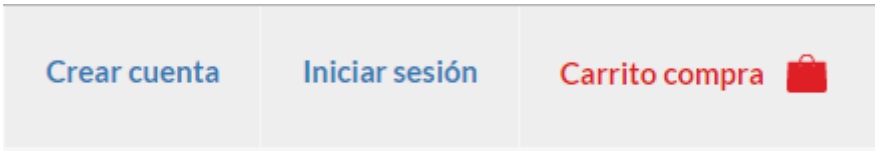


Figura 28 Menú de arriba

Debajo del logo de la tienda aparece otro menú. Este se pondrá las letras en rojo cuando nos encontremos situado en dicha página, es decir, si estamos en la página principal las letras rojas serán las de inicio, si estamos en la página de contáctanos las letras rojas será contáctanos.

Existe un menú princpal en el que podemos observar que esta la pagína principal, visitas guiadas que acceder a una url donde estan los precios de las visitas guiadas que realiza la tienda. Las páginas de leyendas y gastronomía esta en construcción pero aquí ira historias antiguas sobre Úbeda y los platos típicos de esta ciudad. Por último, en la página de contáctanos esta todo lo referente para encontrar la tienda y poder enviar un correo a la tienda por si hubiera cualquier duda o pregunta.

Figura 29. Menú principal

En la página de inicio podemos buscar cualquier producto por su nombre mediante el siguiente filtro:

Figura 30. Filtro buscar producto

Los productos están clasificados en categorías. Si se pincha en una categoría se podrán solamente los productos asociados a esa categoría.

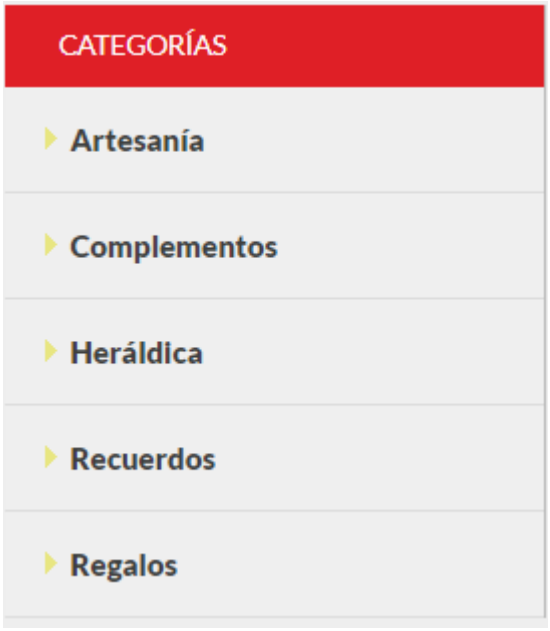


Figura 31. Filtro categorías

En el footer se puede encontrar información de la tienda, sus redes sociales, la política de privacidad, términos del servicio y política de cookies.

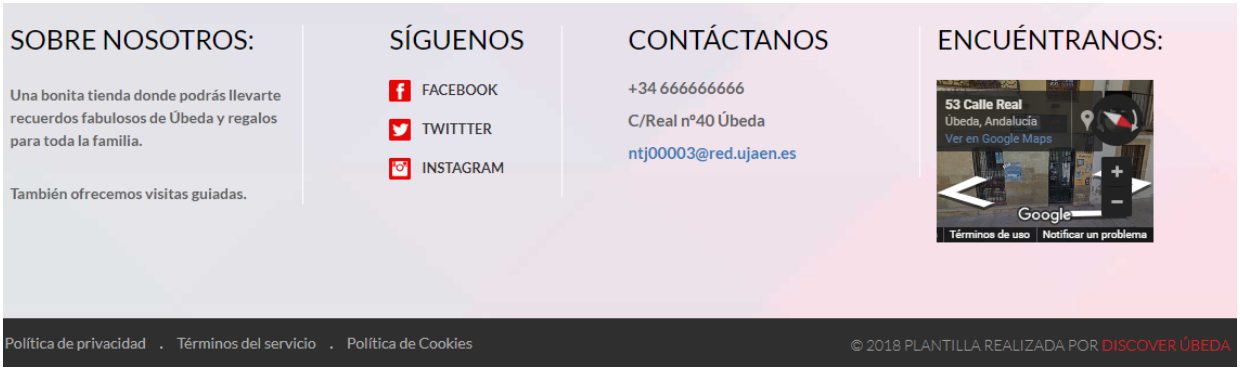


Figura 32. Footer

El menú principal, el footer y el menú de arriba a la derecha son comunes en todas las páginas.

Productos individuales

Si se pincha sobre un producto iremos a otra página donde se encontrará más información del producto.

En esta pantalla podremos añadir al carrito el producto, ver su precio, una descripción más larga, ver todas las imágenes que tiene no solamente una como en la página principal y una información adicional como dimensiones, material, etc.

Figura 33. Producto individual

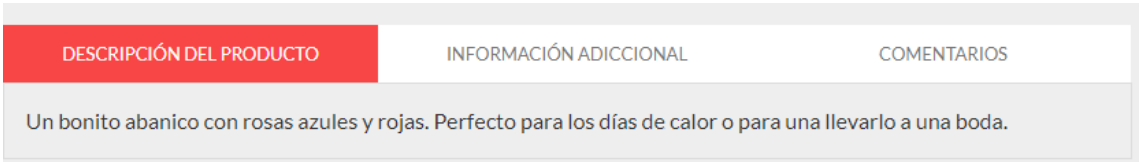


Figura 34. Información adicional

Se puede realizar zoom sobre todas las imágenes para verlas más grandes tal y como muestra la siguiente imagen.

Figura 35. Zoom de los productos

Contáctanos

En esta página se podrá enviar un correo electrónico a la tienda para cualquier duda o sugerencia de los usuarios. También contiene información sobre done está situada la tienda física. En la siguiente imagen podemos verlo.

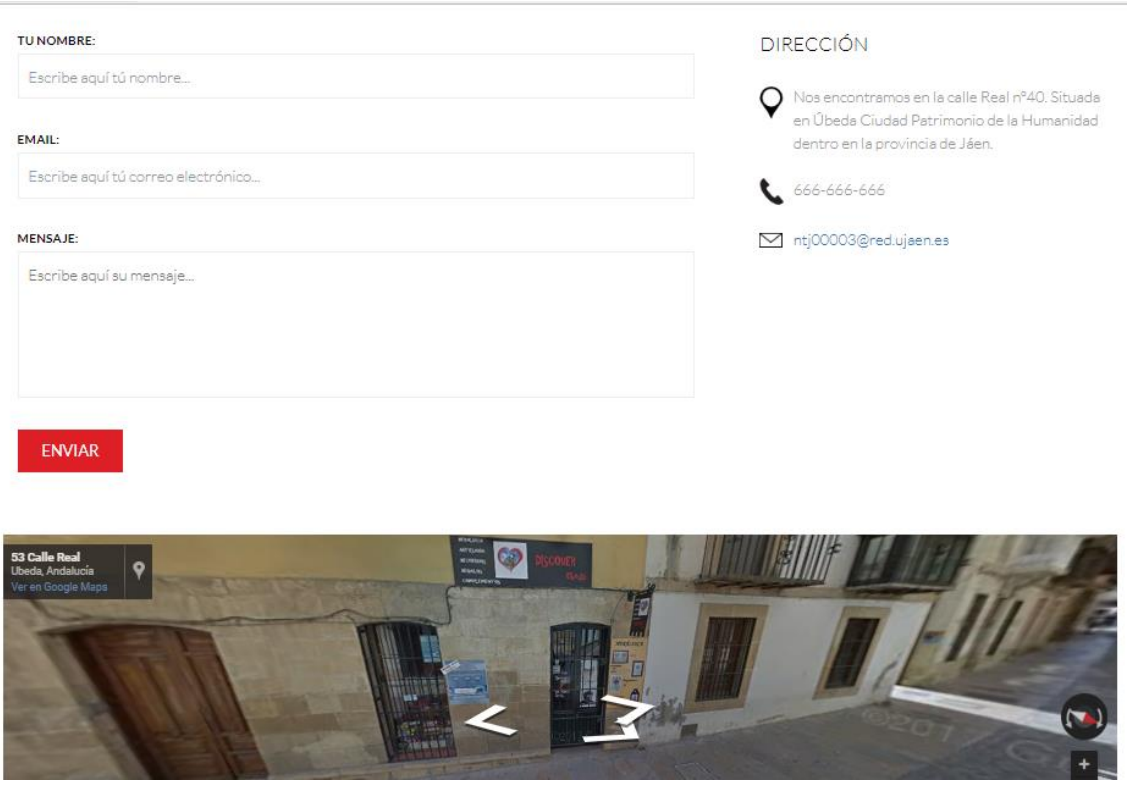


Figura 36. Página contáctanos

Crear cuenta

Cómo muestra la imagen de abajo, esta página muestra un formulario que hay que rellenar todos los datos para crear una cuenta.

Figura 36. Crear cuenta

Iniciar sesión

Para iniciar sesión hace falta el nombre de usuario y la contraseña. También se permite recordar las credenciales.

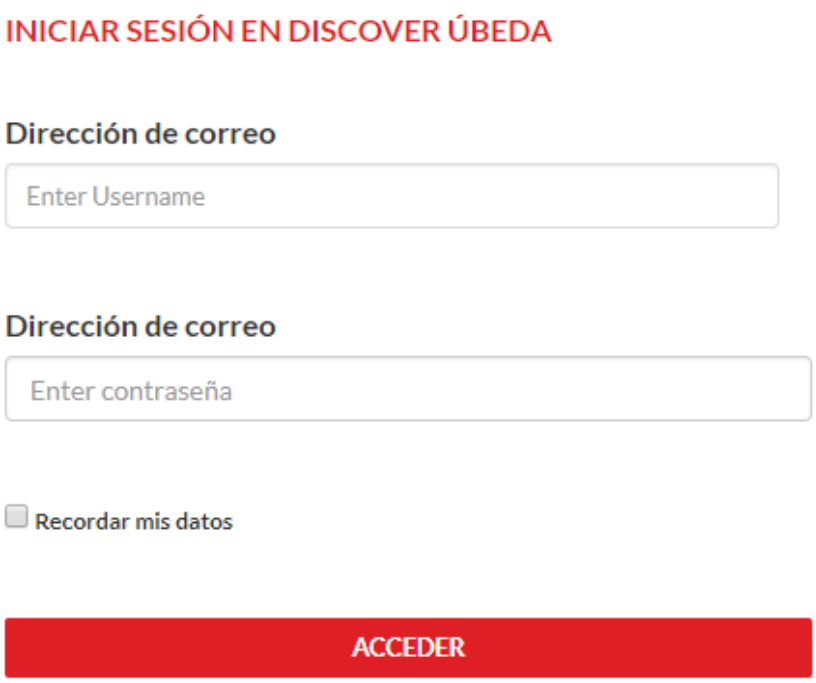


Figura 37. Iniciar sesión

Una vez iniciada la sesión, en el menú principal pueden aparecer una o dos pestañas más. Si se entra como admin aparecerá insertar productos y cerrar sesión como muestra la imagen de abajo. Por el contrario, si se entra como usuario solo aparecerá cerrar sesión

Figura 38. Menú principal

Si se loguea como admin podemos ver los usuarios que hay, editarlos y borrarlos.

Figura 39. Lista de usuarios

Cerrar sesión

Al cerrar la sesión se muestra un mensaje por pantalla diciendo que se ha cerrado la sesión correctamente.

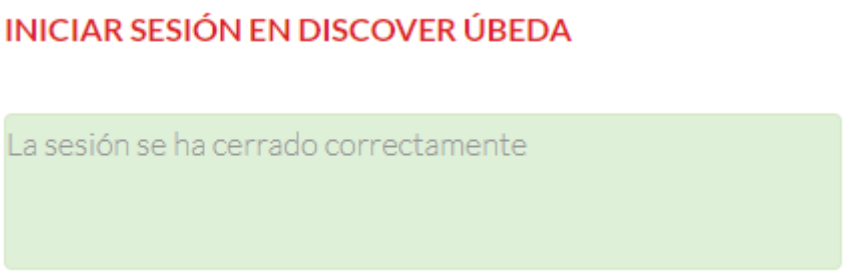
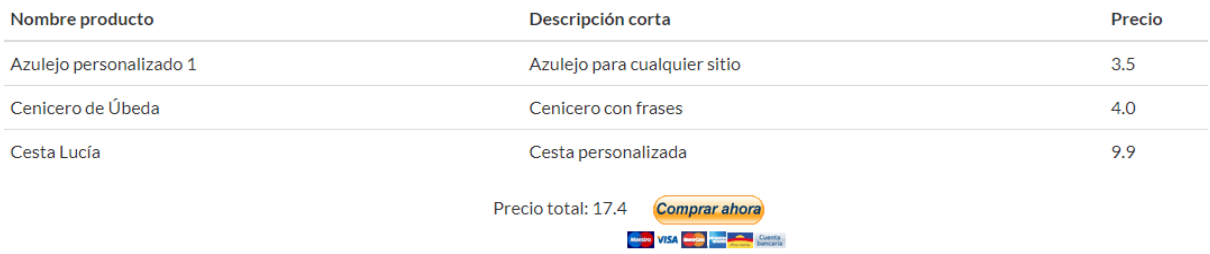


Figura 40. Sesión cerrada

Carrito de la compra

Si pinchamos sobre el carrito de la compra podremos ver todos los productos que tenemos con el precio total y un botón para comprar a través de Paypal.



Nombre producto	Descripción corta	Precio
Azulejo personalizado 1	Azulejo para cualquier sitio	3.5
Cenicero de Úbeda	Cenicero con frases	4.0
Cesta Lucía	Cesta personalizada	9.9

Precio total: 17.4 Comprar ahora

Figura 41. Carito de la compra

Anexo II
Manual de usuario de la aplicación móvil

En este anexo se va a proceder a ofrecer al lector una guía básica con las acciones que puede realizar en la aplicación móvil.

Página principal

En ella se muestra el logo de la tienda y un menú desplegable con las acciones que se pueden realizar.

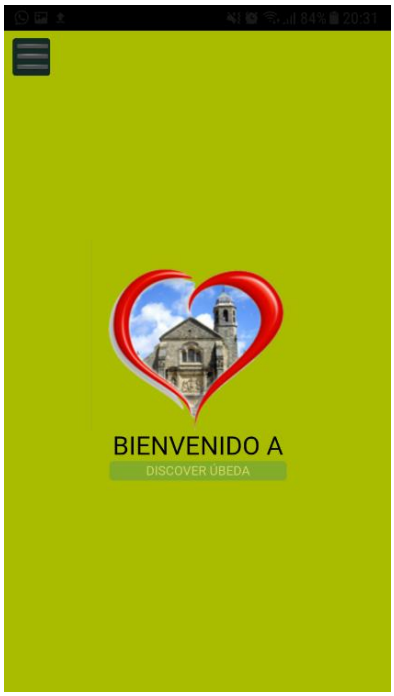


Figura 41. Página web

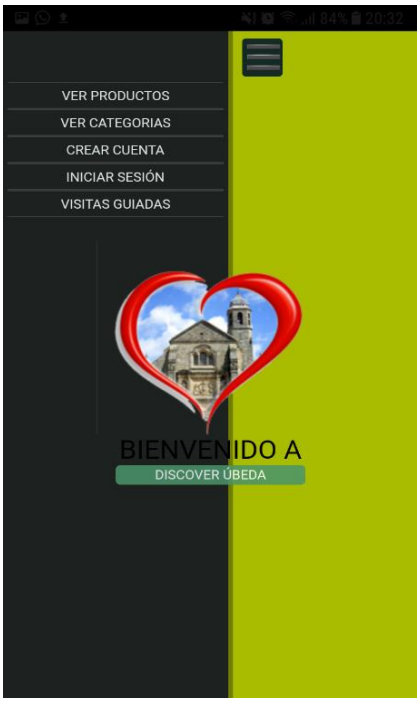


Figura 42. Menú desplegable

Ver productos

Como pasa en la aplicación web primero se muestran todos los productos y si se pincha en el botón de “Ver más” se puede observar información más detallada sobre el producto.

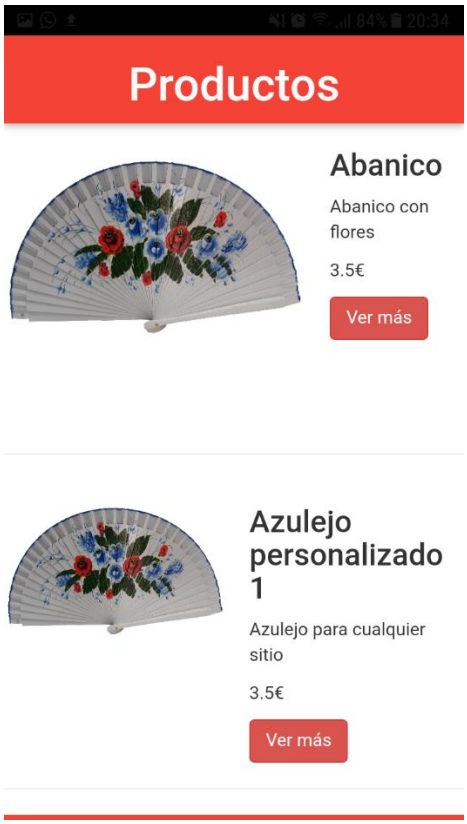


Figura 43. Listado de productos

Figura 43. Detalle producto

Categorías

En esta sección se muestran todas las categorías en las que se clasifican los productos. Si se pincha en una aparecen los productos asociados.

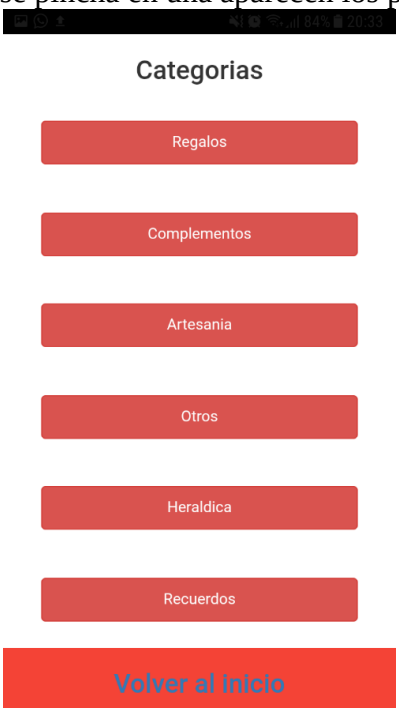


Figura 45. Listado categorías

Crear cuenta

Se puede crear una cuenta rellenado el siguiente formulario:

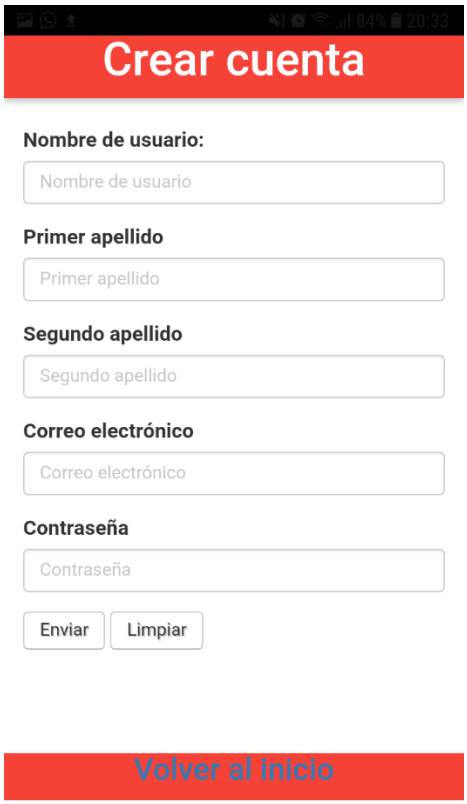


Figura 46. Crear cuenta