UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Desarrollo de un portal web para el comercio electrónico

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Jin Li Hu

Tutor: Germán Francisco Vidal Oriola

2016 - 2017

Resumen

En este proyecto se plantea desarrollar un portal *web* de carácter comercio electrónico enfocado a empresas pequeñas. El portal se va a desarrollar con el entorno de trabajo *Bootstrap*, el lenguaje de programación *PHP* y la base de datos *MySQL*. Se va a centrar en conseguir un sitio *web* fácil de usar y que sea accesible desde cualquier dispositivo tecnológico.

Palabras clave: portal web, bootstrap, comercio electrónico, usabilidad, accesibilidad.

Abstract

In this project, we are going to develop a ecommerce website which is focused on small size companies. The website is going to be developed with the framework Bootstrap, server side programing language PHP and the database MySQL. We are going to focus on getting a user friendly website and it must be accessible from any technological device.

Keywords : web portal, bootstrap, ecommerce, usability, accessibility.

Tabla de contenidos

1. Introducción

El presente documento describe el proceso de implementación de un portal *web* de carácter comercio electrónico.

1.1 Motivación

En actualidad la tecnología de internet web está disponible en todos lados, en el trabajo, en el hogar y en cualquier otro lado a través de los dispositivos móviles y no móviles, y en cualquier momento. El mercado se extiende más allá de los límites tradicionales y se elimina de una ubicación temporal y geográfica. Como consecuencia surge el comercio electrónico, con el que se puede realizar compras en cualquier parte. Se mejora la conveniencia para el cliente y se reducen los costos de compra.

Pequeños comercios han visto en la tienda virtual una ventana de escape a sus productos, debido a la ausencia de locales físicas y la reducción de costes de mantenimiento.

Como el desarrollador de este proyecto, me gustaría extenderlo y convertirlo en una aplicación web real que pueda ser desarrollada para las empresas pequeñas que deseen ampliar sus negocios por Internet.

1.2 Objetivo

El objetivo de este proyecto es crear un portal *web* para el comercio electrónico enfocado a empresas pequeñas, un portal *web* con el que los clientes puedan realizar compras de manera simple y sencilla, y que puedan acceder el portal *web* desde cualquier dispositivo.

El portal *web* va a tener los siguientes componentes esenciales:

- Estructura responsiva
- Sistema de cesta
- Sistema de mensajería en línea
- Sistema de identificación y registro
- Sistema de procesamiento de pago en línea

1.3 Estructura del documento

Este documento se presenta siguiendo una estructura de fases, algo común en el desarrollo de proyectos *software*.

A continuación veamos las diferentes fases de desarrollo del portal *web*.

- Especificación de requisitos: Se recolectan todos aquellos requisitos funcionales y visuales que debe de tener el portal *web*.
- Análisis: Se estudia y analiza factores importantes que afectan nuestro portal *web*.
- Diseño: Se describen el proceso de diseño el portal web teniendo en cuenta los resultados obtenidos de la fase de análisis.
- Implementación: Se describe la implementación de los componentes esenciales del portal *web*.
- Conclusiones: Se señalan los objetivos conseguidos y las conclusiones obtenidos del proyecto.
- Bibliografía: Se listan las bibliografías utilizadas durante la elaboración de este documento.
- Anexos: Se incluye información extra acerca de la implementación del portal *web*.

2. Especificación de requisitos

2.1. Introducción

En esta sección se recopilan las características y condiciones que el portal *web* debe cumplir para conseguir un funcionamiento adecuado y satisfactorio para los usuarios.

Para elaborar esta especificación de requisitos se ha seguido como modelo el estándar de IEEE 830, el cual nos proporciona una estructura clara de cómo ha de ser un documento ERS.

2.1.1 Propósito

La finalidad de la especificación de requisitos es conocer las funciones y las limitaciones del software en cuestión.

2.1.2 Ámbito

El objetivo del proyecto es crear un portal *web* para el comercio electrónico, en nuestro caso, se ha enfocado en un comercio que vende dispositivos móviles, tabletas, relojes inteligentes, y accesorios de los tres anteriores.

Para la implementación del portal *web*, se ha registrado el nombre de sub-dominio *eGadgets.uphero.com* en la plataforma de *hosting* gratuito *000webhost*.

El *web* en cuestión ha de ser responsivo, debe ofrecer las características esenciales de un *web e-Commerce*, entre ellas caben destacar, el sistema de identificación, el sistema de carrito, pago seguro, soporte técnico mediante mensajería en línea y un buscador de productos.

2.1.3 Definiciones, acrónimos y abreviaturas

- **IEEE:** Son las siglas de *Institute of Electrical and Electronics Engineers* es una asociación mundial de ingenieros dedicadas a la estandarización y el desarrollo en áreas técnicas.
- **ERS:** La especificación de requisitos de *software*, es una descripción completa del comportamiento del sistema que se va a desarrollar.
- **Subdominio:** Un grupo o subclasificación del nombre de dominio, que podría considerarse como un dominio de segundo nivel.

- **Web Hosting:** El servicio que provee a los usuarios de *Internet* un sistema para poder almacenar información, imágenes, vídeo, o cualquier contenido accesible vía *web*.
- **Diseño web Responsivo:** Una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las páginas *web* al dispositivo que se esté utilizando para visitarlas.
- **e-Commerce:** También es conocido como comercio electrónico, que consiste en la compra y venta de productos o de servicios a través de medios electrónicos, tales como *Internet* y otras redes informáticas.
- **Front-end:** En la programación *web*, *front-end* son todas aquellas tecnologías que corren del lado del cliente, normalmente es el que se encarga de estilizar la página de tal manera que la página pueda quedar cómoda para la persona que la ve.
- **Back-end:** En la programación *web*, *back-end* es la parte que procesa la entrada desde el *front-end* son todas aquellas tecnologías del lado del servidor.
- **Framework:** Una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de *software*, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado para ayudar a desarrollar y unir los diferentes componentes de un proyecto.
- **Bootstrap:** Un conjunto de herramientas de código abierto para diseño de sitios y aplicaciones *web*. Contiene plantillas de diseño con tipografía, formulario, menú de navegación y otros elementos de diseño basado en *HTML* y *CSS*, así como, extensiones de *JavaScript*.
- **HTML:** El lenguaje de marcado para elaboración de páginas *web*, cuyas siglas son *HyperText Markup Language*.
- **CSS:** Es conocido como hojas de estilo en cascada, cuyas siglas en inglés son *Cascading StyleSheets*, es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado.
- **Javascript:** Un lenguaje de programación interpretado, implementado como parte de un navegador *web* permite mejoras en la interfaz de usuario y páginas *web* dinámicas.
- **PHP:** Un lenguaje de programación de programación de uso general de código del lado del servidor, diseñado para el desarrollo *web* de contenido dinámico.
- **Stripe:** Una plataforma que ofrece servicio de procesamiento de pagos a través de *Internet* de forma segura.
- **PhpMyAdmin:** Una herramienta escrita en *PHP* con la intención de manejar la administración de *MySQL* a través de páginas *web*, utilizando *Internet*.

- **MySQL:** Un sistema de gestión de bases de datos relacional desarrollado por la empresa *Oracle Corporation*.
- **Tawk.to:** Una platamaforma que ofrece servico de mensajería en línea a través de *internet*.
- **Back-up:** Copia de seguridad o más archivos informáticos, que se hace, generalmente, para prevenir posibles pérdidas de información.

2.1.3 Visión general

A continuación se realiza una descripción general y otra más detallada de los los requisitos específicos del portal *web* que se va a desarrollar.

2.2. Descripción general

2.2.1. Perspectiva del producto

La aplicación se desarrolla en dos partes separadas, *front-end* y *back-end*.

La parte *front-end* de la aplicación se implementa con el *framework Bootstrap*, un entorno de trabajo de *HTML*, *CSS* y *Javascript* que ofrece una gran flexibilidad y una gran variedad de opciones de diseño de páginas *web*. Y la parte *back-end* se desarrolla con *PHP* y *MySQL* los lenguajes de programación más populares del lado de servidor.

El sitio *web* que se desarrolla no es totalmente independiente, se integra dos servicios de terceros. Uno de los servicios integrados es Tawk.to, una plataforma gratuita de mensajería en línea, la cual nos permite monitorizar y chatear con los visitantes de nuestro portal *web*. Y el otro es Stripe, una plataforma de procesamiento de pago seguro a través de *internet*.

2.2.2. Funcionamiento del producto

El portal *web* se dispone de las siguientes funciones:

Función de búsqueda: Esta función permite a los usuarios buscar de manera fácil y rápido los productos deseados. Es un buscador de tamaño considerable situado en el centro de la barra de navegación que permite ser visualizado de manera inmediata.

Función de compra: Esta función está compuesta por tres elementos muy importantes, uno de ellos es el sistema de carrito que permite tanto a los usuarios no identificados como los identificados almacenar los productos que

desean comprar en un carro virtual, con el cuál se les permite visualizar los detalles y modificar la cantidad de los productos añadidos.

El segundo elemento es el sistema de identificación, para poder llevar a cabo el proceso de compra el usuario debe estar identificado con una cuenta registrada previamente con nuestra aplicación, gracias a este sistema los usuarios pueden hacer un seguimiento a sus pedidos, modificar, cancelar o hacer una devolución de los productos comprados.

El último elemento es el sistema de procesamiento de pago a través de *Internet*, es un servicio de terceros que se ha de integrar de la compañía Stripe. Los usuarios pueden pagar sus pedidos con tarjetas crédito o débito de manera segura y rápida, sin necesidad de tener una cuenta con la plataforma de procesamiento de pago, como es el caso de Paypal.

Función de soporte: Para facilitar el uso de nuestra aplicación a los usuarios, a parte de tener un soporte técnico mediante teléfonos, nuestro sitio *web* también dispone de un sistema de mensajería en línea, con el que los usuarios pueden resolver sus dudas a través de este sistema, es un servicio integrado de terceros, de la compañía Tawk.to.

Función de mantenimiento: El administrador puede gestionar tanto los pedidos de los usuarios, como el stock de los productos a través de la interfaz de PhpMyAdmin que sirve para gestionar la base de datos.

2.2.3. Características del usuario

Se distingue entre tres tipos de usuario en nuestra aplicación:

- **Anónimo:** aquel usuario que no está identificado, pueden explorar nuestra *web*, visualizar los productos y usar nuestro sistema de soporte técnico pero no pueden comprar.
- **Usuario registrado:** este tipo usuario a parte de tener los mismos privilegios que los usuarios anónimos puede también realizar compras, y hacer modificaciones sobre su pedido.
- **Soporte técnico:** aquel que interactúa con los demás usuarios a través de la interfaz que proporciona la plataforma de mensajería en línea integrada para resolver posibles dudas.
- **Administrador de la base de datos:** aquellos que se encarga de gestionar la base de datos de la aplicación.

2.2.4. Restricciones

En esta subsección se va a describir las limitaciones que se impone sobre los desarrolladores del producto.

Los lenguajes usados para llevar a cabo el desarrollo de esta aplicación son:

- Front-end:
  - HTML
  - CSS
  - JAVASCRIPT
- Back-end:
  - PHP
  - MySQL
  - JSON

A parte de las interfaces propias de la aplicación también se dispone de las interfaces externas proporcionadas por los dos servicios integrados que son el procesamiento de pago seguro a través de *Internet* y el sistema de mensajería en línea que solo tienen acceso a ellas los soportes técnicos de la tienda virtual.

2.2.5. Supuestos y dependencias

Para conseguir un correcto funcionamiento en nuestra *web*, es necesario que los navegadores de los usuarios tenga la opción de visualización de *Javascript* habilitada, ya que ciertas funciones de nuestra aplicación están desarrolladas con dicho lenguaje de programación.

Se debe de realizar una *Back-up* a la base de datos periódicamente para prevenir la pérdida de datos por ataques de *hackers* o la mala gestión por parte de los administradores de la base de datos.

2.3. Requisitos específicos

2.3.1. Interfaces externas

2.3.1.1. Interfaz de software

Dependiendo del dispositivo que se usa el usuario el sitio *web* ha de adaptarse su diseño de manera adecuada e intuitiva, sean dispositivos móviles, tabletas, portátiles o ordenadores de sobremesa.

2.3.1.2. Interfaz de comunicación

Debe de haber mensajes de interacción entre el usuario y el sitio *web*, cuando el usuario se haya realizado acciones erróneas, que la *web* mostrase mensajes explícitos e intuitivos para que el usuario pueda entender lo que está pasando y corregir los errores.

2.3.2. Funcionales

Dependiendo del tipo de usuario se permite realizar diferentes tipos de funciones. A continuación se detallan las posibles funciones.

2.3.2.1. Anónimo

Función	**Consultar categoría**
Entrada	Selección de la categoría de producto
Proceso	Acceso de lectura a la base de datos
Salida	Todos los productos de la categoría en cuestión

Función	**Visualizar producto**
Entrada	Selección del producto
Proceso	Acceso de lectura a la base de datos
Salida	Los detalles del producto en cuestión

Función	**Filtrar productos**
Entrada	Selección de los filtros
Proceso	Acceso de lectura a la base de datos
Salida	Todos los productos relacionados con los atributos del filtro

Función	**Buscar producto**
Entrada	Introducción de palabras claves relacionadas con el producto en cuestión
Proceso	Acceso de lectura a la base de datos
Salida	Los detalles del producto en cuestión

Función	**Añadir al carrito**
Entrada	Selección del producto
Proceso	Acceso de escritura a variable de sesión de PHP
Salida	Producto añadido al carrito

Función	**Comprar**
Entrada	Selección del producto
Proceso	Acceso de escritura a variable de session de PHP y redirección a la interfaz del carrito
Salida	Producto añadido al carrito, página redirigida al carrito

Función	**Chatear**
Entrada	Introducción del texto
Proceso	Procesamiento de texto por parte de la plataforma de mensajería
Salida	Mensaje enviado al soporte técnico

2.3.2.2. Usuario identificado

Función	**Consultar categoría**
Entrada	Selección de la categoría de producto
Proceso	Acceso de lectura a la base de datos
Salida	Todos los productos de la categoría en cuestión

Función	**Visualizar producto**
Entrada	Selección del producto
Proceso	Acceso de lectura a la base de datos
Salida	Los detalles del producto en cuestión

Función	**Filtrar productos**
Entrada	Selección de los filtros
Proceso	Acceso de lectura a la base de datos
Salida	Todos los productos relacionados con los atributos del filtro

Función	**Buscar producto**
Entrada	Introducción de palabras claves relacionadas con el producto en cuestión
Proceso	Acceso de lectura a la base de datos
Salida	Los detalles del producto en cuestión

Función	**Añadir al carrito**
Entrada	Selección del producto
Proceso	Acceso de escritura a variable de sesión de PHP
Salida	Producto añadido al carrito

Función	**Comprar**
Entrada	Selección del producto
Proceso	Acceso de escritura a variable de session de PHP y redirección a la interfaz del carrito
Salida	Producto añadido al carrito, página redirigida al carrito

Función	**Chatear**
Entrada	Introducción del texto
Proceso	Procesamiento de texto por parte de la plataforma de mensajería
Salida	Mensaje enviado al soporte técnico

Función	**Pagar**
Entrada	Introducción de datos bancarias
Proceso	Procesamiento de pago por parte de la plataforma Stripe
Salida	Pedido realizado

Función	**Modificar datos personales**
Entrada	Introducción de datos
Proceso	Acceso de escritura a la base de datos
Salida	Cambios realizados

Función	**Visualizar pedidos**
Entrada	Selección de la visualización de pedidos
Proceso	Acceso de lectura a la base de datos
Salida	Todos los pedidos del usuario en cuestión

Función	**Cancelar pedido**
Entrada	Selección del pedido
Proceso	Acceso de escritura a la base de datos
Salida	Estado del pedido modificado

Función	**Solicitar devolución**
Entrada	Selección del pedido
Proceso	Acceso de escritura a la base de datos
Salida	Estado del pedido modificado

2.3.2.3. Soporte técnico

Función	**Chatear**
Entrada	Introducción del texto
Proceso	Procesamiento de texto por parte de la plataforma de mensajería
Salida	Mensaje enviado al cliente

2.3.2.4. Administrador de la base de datos

Función	**Gestión de base de datos**
Entrada	Selección de la base de datos
Proceso	Leer, escribir, actualizar o eliminar los datos
Salida	Base de datos actualizada

2.3.3. Restricciones del diseño

El sitio *web* se va a diseñar siguiendo la filosofía de desarrollo centrado en el usuario, que tiene por objetivo la creación de productos que resuelvan las necesidades concretas de sus usuarios finales. Se va a hacer hincapié en conseguir un nivel de usabilidad y accesibilidad adecuada para que la aplicación pueda ser utilizada por los usuarios de manera eficaz y sencilla.

2.3.4. Atributos del sistema

Respecto a la información de los clientes registrados, teniendo en cuenta que en el ámbito de la red existen grandes posibilidades de ataques e intrusiones, no se va a almacenar las contraseñas tal como las han puesto los usuarios, sino que se va a realizar una serie de medidas de seguridad para garantizar la protección de los datos de los usuarios y de la propia aplicación *web*.

Se va a realizar una copia de seguridad de la base de datos periódicamente para prevenir las posibles pérdidas de información.

3. Análisis

3.1. Introducción

El análisis de la aplicación *web* se va a realizar siguiendo las metodologías de DCU, desarrollo centrado en el usuario, como objetivo principal se va a construir una aplicación *web* que puedan ser utilizada por los usuarios de manera eficaz y sencilla.

Para conseguir el objetivo de nuestro proyecto, se va a analizar las necesidades de usuario, y se va a estudiar las siguientes cuestiones como estudio previo para la fase de análisis:

- ¿Qué usuarios van a utilizar la aplicación?
- ¿Qué aptitudes tienen?
- ¿Qué necesitan de la aplicación?

3.1.1. ¿Qué usuarios van a utilizar la aplicación?

No es una pregunta tan obvia como para responder con pocas palabras, y es que dependiendo de las diferentes factores se puede distinguir entre tres tipos de usuarios:

- **Usuarios primarios:** aquellos que tienen un uso directo con la aplicación y con una alta frecuencia de uso.
- **Usuarios secundarios:** aquellos que usan la aplicación ocasionalmente o a través de otro.
- **Usuarios terciarios:** son los que se ven afectados por el uso o la compra del producto.

En este proyecto, el dueño de la tienda virtual representa el usuario terciario, los clientes anónimos y los clientes registrados representan los usuarios primarios y, los técnicos de mantenimiento del sitio *web* representan los usuarios secundarios.

3.1.2. ¿Qué aptitudes tienen?

Las diferentes aptitudes de los usuarios van a condicionar el diseño de nuestro *producto software*. Se va a crear usuarios de diferentes tipos considerando las aptitudes detectadas que representan.

Para averigual las actitudes de nuestros usuarios, se ha considerado aspectos importantes como las habilidades del usuario, sus conocimientos, su edad y género, sus hábitos y preferencia de trabajo.

En este proyecto se ha identificado tres tipos de usuario teniendo en cuenta la experiencia de uso: principiantes, usuarios intermedios y usuarios avanzados, se va a centrar en optimizar el diseño para usuarios intermedios, debido a que nadie quiere ser un principiante para siempre y que pocos se convierten en expertos.

3.1.3. ¿Qué necesitan de la aplicación?

Generalmente los usuarios no son capaces de contar qué necesitan de la aplicación ni tampoco saber qué es posible de conseguir con una aplicación software. Por lo tanto se va a centrar en entender qué tareas realizan en el trabajo diario de los usuarios, y qué tareas les gustarían poder realizar adicionalmente.

Una vez estudiadas estas cuestiones, se va a proceder a analizar las necesidades de usuario siguiendo los siguientes pasos:

- **Investigación cualitativa:** Identificar los usuarios y sus necesidades.
- **Definición de Personas:** Crear usuarios ficticios que representan diferentes agrupaciones de comportamientos, actitudes, aptitudes, objetivos y motivaciones.
- **Descripción de Escenarios:** Refinar los objetivos en descripciones que indican cómo conseguirlos.

3.2. Investigación cualitativa

La investigación cualitativa consiste en la recolección de datos que son no cuantitativos, con el propósito de explorar y describir la realidad tal como la experimentan sus correspondientes protagonistas.

Dicha investigación requiere un profundo entendimiento del comportamiento humano y las razones que lo gobiernan. A diferencia de la investigación cuantitativa, la investigación cualitativa busca explicar las razones de los diferentes aspectos de tal comportamiento.

Investiga el por qué y el cómo se tomó una decisión, en contraste con la investigación cuantitativa, que busca responder preguntas tales como cuál, dónde, cuándo y cuánto.

Entre las técnicas más importantes a la hora de recopilar información cualitativa caben destacar:

- Entrevistas
- Cuestionarios
- Grupos de interés y talleres
- Observación
- Estudio de documentación

3.2.1. Entrevista

Entrevista es una técnica tradicional muy usada, es adecuada para las primeras tomas de contacto. No se trata de una técnica sencilla, ya que requiere de experiencia y habilidad de comunicación. Pero a pesar de su dificultad de manejo es muy útil para entender los problemas actuales y extraer los objetivos reales.

A la hora de preparar una entrevista hay que tener en cuenta el tipo de preguntas y el tipo de entrevista que se va a realizar.

3.2.1.1. Tipos de pregunta

- Preguntas cerradas: aquellas que tienen una respuesta predeterminada (e.g. SI/NO).
- Preguntas abiertas: aquellas que permiten contestar al entrevistado libremente.

3.2.1.2. Tipos de Entrevista

Entrevista estructurada

Este tipo de entrevista se basa en una serie de preguntas predeterminadas. Se puede decir que es una de las más rígidas ya que no se permite salirse del guión preestablecido. Proporciona precisión y seguridad como ventajas y, como inconveniente, impide conversaciones amigables. Se trata de una técnica fácil de replicar de un proyecto a otro, aunque las respuestas obtenidas aportan menos información.

Entrevista no estructurada

La entrevista no estructurada también es conocido como entrevista libre. En ella se trabaja con preguntas abiertas, sin un orden preestablecido, adquiriendo las características de conversación y permitiendo la espontaneidad.

Comparando con la entrevista estructurada la no estructurada se obtiene más respuestas más ricas pero es mucho más difícil de replicar.

3.2.2. Cuestionarios

La técnica de cuestionarios es entrevista estructurada escrita en papel u otro medio, con esta técnica el entrevistado puede tomarse más tiempo para responder y hacerlo en un momento y lugar adecuado, e incluso, remotamente.

Esta técnica es muy útil en dominios donde se requiere recolectar un gran número de enfoques de una situación o para contrastar opiniones. Es necesario conocer suficientemente el dominio para preparar las preguntas.

A la hora de elaborar los cuestionarios hay que tener en cuenta los siguientes factores:

- Contenido del cuestionario: establecer los focus, y para cada foco una o varias preguntas.
- Forma de las preguntas: pueden ser preguntas cerradas o abiertas.
- Formato del cuestionario: hay que planificar la cantidad total de preguntas, el orden de las preguntas.

3.2.3. Grupos de interés y talleres

Esta técnica permite la participación de todos los implicados en el desarrollo: analistas, diseñadores, usuarios, etc. La ventaja de usar esta técnica es que recoge muchos puntos de vista, resalta las áreas de consenso y de conflicto y fuerza el contacto entre desarrolladores y usuarios.

3.2.4. Observación

La técnica observación consiste en recopilar la información necesaria a través de la observación de los usuarios.

Hay cuatro diferentes formas de observación:

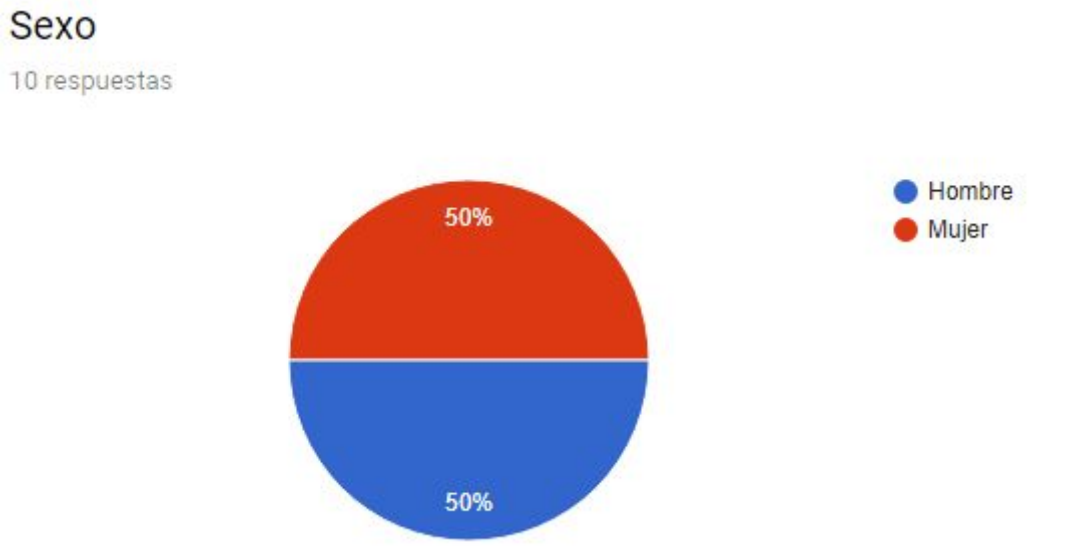
- Observación pasiva: el observador no participa en la organización estudiada. Se usan cintas de audio, video, cámaras de seguridad, etc.
- Observación activa: el observador participa directamente. Puede hacer preguntas para clarificar las tareas realizadas.
- Observación explicativa: el usuario va narrando y explicando las tareas realizadas.
- Etnografía: el observador participa activamente en las tareas ayudando o realizando directamente algunas de las tareas.

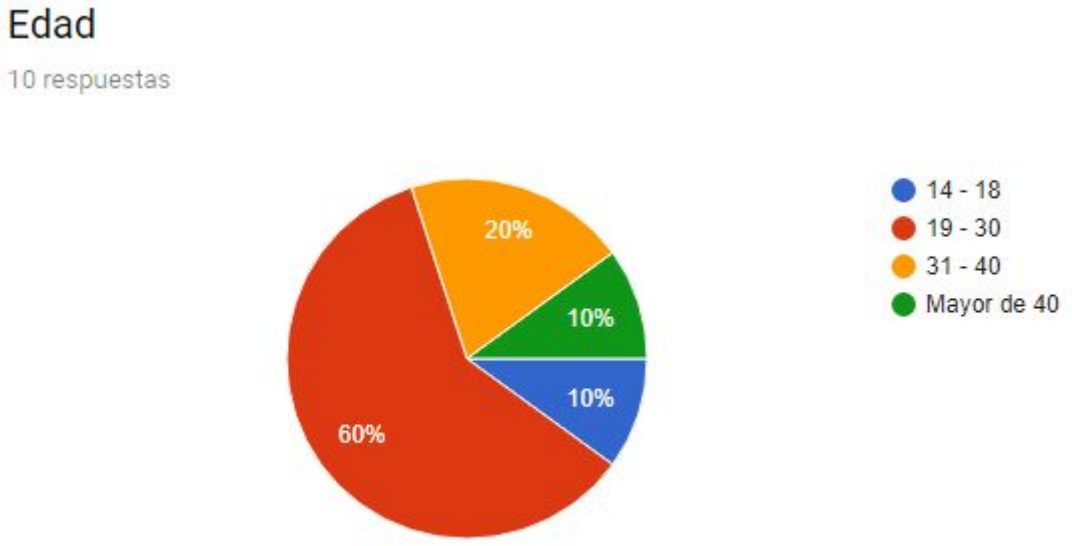
3.2.5. Estudio de documentación

Consiste en analizar la documentación utilizada en la organización, que pueden ser formularios, facturas, contratos, información financiera o información de mercado. Como ventaja de utilizar este tipo de técnica es que no se necesita tiempo de los usuarios, y es bueno para aprender los procedimientos, reglas y estándares.

3.2.6. Técnica utilizada para el proyecto

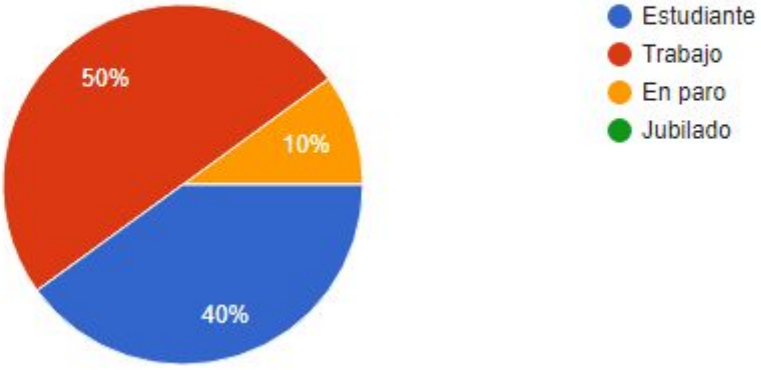
En este proyecto se ha seguido la técnica Cuestionarios para proceder a la Investigación cualitativa. Se ha hecho un cuestionario de diez preguntas cerradas, y se ha obtenido los siguientes resultados:





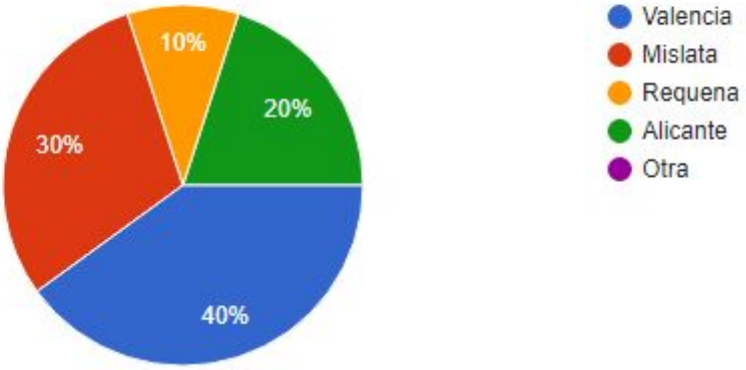
Ocupación

10 respuestas



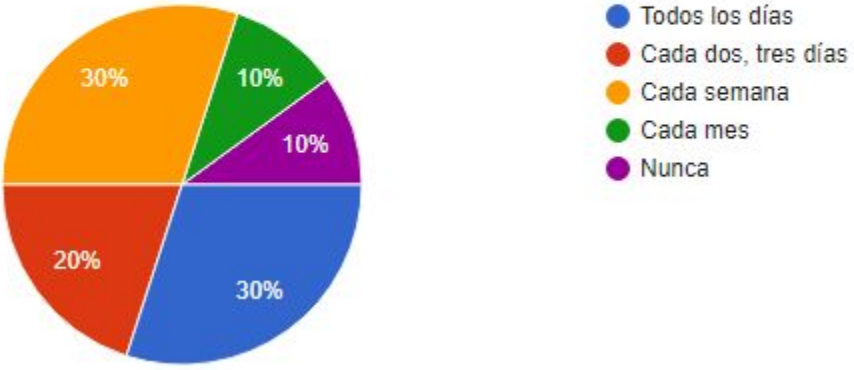
Ciudad de residencia

10 respuestas



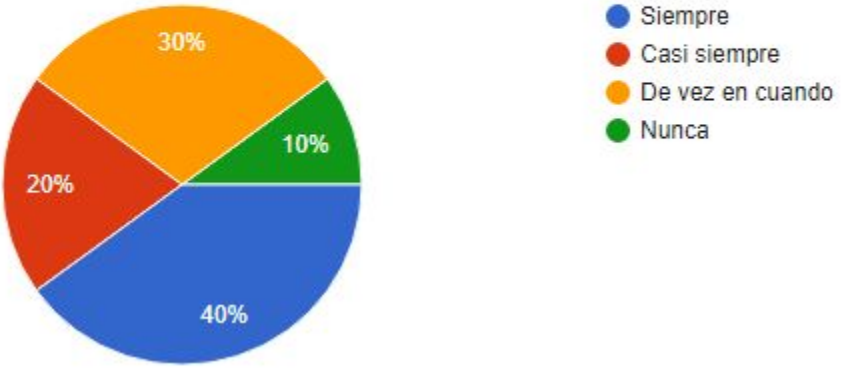
¿Con qué frecuencia visitas a tiendas virtuales?

10 respuestas



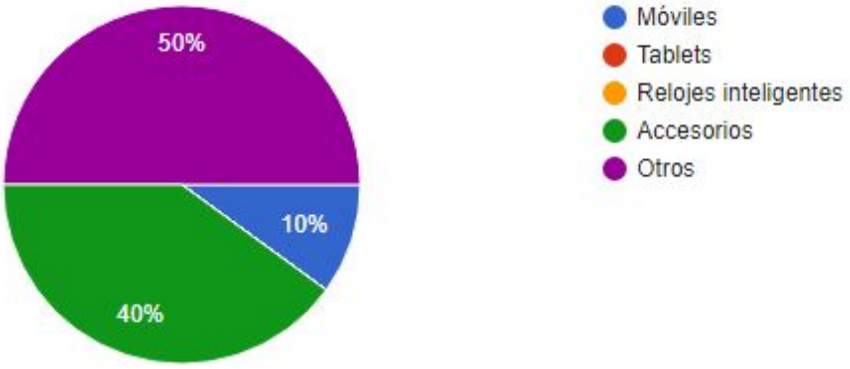
¿Con qué frecuencia realizas compras a través de Internet?

10 respuestas



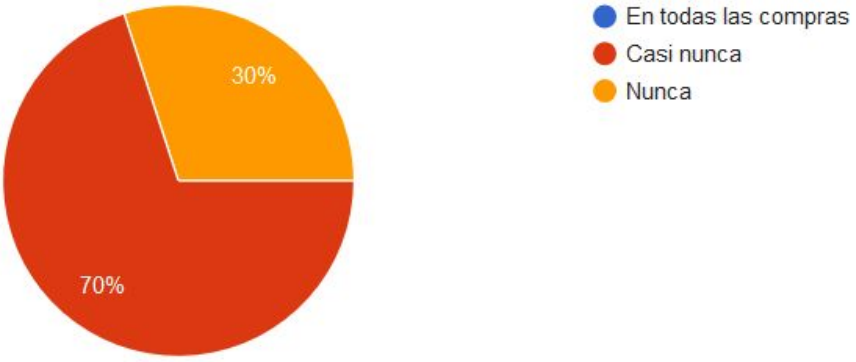
¿Qué tipo de productos sueles comprar a través de Internet?

10 respuestas



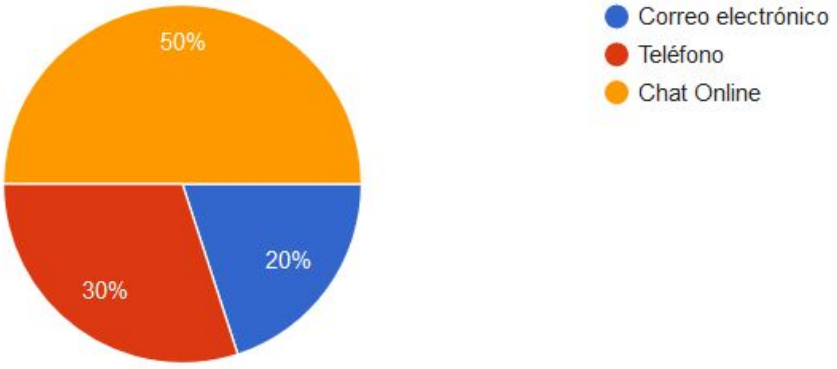
¿Con qué frecuencias sueles realizar devoluciones?

10 respuestas



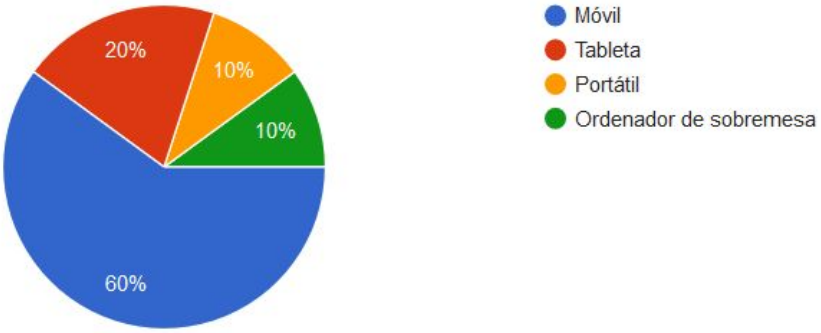
¿Qué tipo de herramienta usarías para contactar?

10 respuestas



¿Qué tipo de dispositivos sueles usar para visitar las tiendas virtuales?

10 respuestas



3.3. Personas

Una Persona es una descripción de un personaje ficticio para el cual debe ser diseñado el producto *software*. La función principal de una Persona es alcanzar los objetivos establecidos, la mayoría de los objetivos deben ser relevantes en relación con el producto desarrollado, aunque algunos pueden ser más generales, incluso relacionados con sus forma de vida, que ayuden contextualizar a la Persona.

3.3.1. Persona creada

Basando en los resultados obtenido en la fase de investigación cualitativa, se ha creado la siguiente Persona:

Sergio García

Biografía:

- 26 años de edad.
- Lleva viviendo en Valencia desde hace 15 años.
- Aún vive con sus padre.
- Tiene una mascota, un perro de la raza *Pug.*
- Le encanta conducir, tanto conducir coche como moto.
- Trabaja temporalmente en una pizzería como repartidor de pizzas.
- Es graduado en bellas artes.
- Sabe hablar inglés a parte de castellano y valenciano.
- Había estudiado informática durante un curso, pero lo dejo por bellas artes porque le resultaba difícil avanzar.

Objetivos:

- Independizarse antes de los 28 años.
- Comprar un piso cerca de sus padres.
- Encontrar un trabajo que corresponde a sus estudios.

Salud:

- Tiene miopía desde pequeño.
- Tiene una lesión permanente en su mano izquierda, se la hizo cuando jugaba baloncesto de pequeño en la calle.

Tecnología:

- No es un experto con los dispositivos tecnológicos, pero sabe manejarlos sin problemas.
- Es un fanático de los productos de apple.

- Tiene un MacBook Pro.
- Tiene un *iPhone 5*, se lo regalaron sus padres cuando se graduó en bellas artes.
- Recientemente ha comprado una tableta *Kindle Fire* en la página de Amazon, porque es barato y está ahorrando dinero para comprar un piso. Lo usa principalmente para ver películas y visitar páginas *web.*

3.4. Escenarios

Un escenario es una descripción de un diseño desde el punto de vista de una Persona específica.

Con el desarrollo de escenarios se puede identificar aspectos importantes que afectan a la utilización de un producto en el mundo real y que no se pueden identificar ni tenerse en cuenta de otro modo.

3.4.1. Escenario creado

Juntando la Persona creada y los resultados obtenidos en la fase de investigación cualitativa se ha creado el siguiente escenario:

Sergio lleva meses buscando un móvil nuevo para sustituir su viejo *iPhone 5*, busca un móvil de gama media y que tenga una pantalla mayor de 4 pulgadas, Sergio usa su móvil para hacer prácticamente todo, y no lo pierde de vista nunca, pero un móvil de 4 pulgadas ya no le es suficiente, y necesita algo más.

Sergio coge su *Kindle Fire* y empieza a buscar móviles en *Google*, le sale tantas opciones que no sabe cual escoger, al final entra a la página de *Amazon* para buscar, ya que está bastante familiarizado con esta página.

Lo primero que hace es buscar entre las categorías la opción de *Smartphones*, luego hace clic la opción de ordenar los productos por precio de bajo a alto, ya que tiene un presupuesto ajustado, ve una variedad de móviles, pero se frustra porque la mayoría de ellos no son lo que buscaba, para que la búsqueda sea más precisa, Sergio acude a la función de filtro que dispone la página de *Amazon*, entre las opciones del filtro Sergio escoge el tamaño de pantalla mayor de 4 pulgadas, y sistema operativo *Android*, ha decidido cambiar de *iOS* a *Android* porque le gustaría probar cosas diferentes, y con su nueva tableta *Kindle Fire* se está gustando mucho el sistema *Android*.

Con todas las opciones que le muestra tras aplicar el filtro, parece que por fin Sergio encuentra lo que buscaba, un móvil de gama media de un precio que encaja con su situación económica actual, para saber más detalles de ese

móvil Sergio hace clic en la imagen y entra en la página específica de ese móvil, lo primero que ve cuando entra a la página son las letras rojas de aviso de que ese producto está agotado, pero no pone en ningún lado de cuándo estará disponible de nuevo, Sergio se siente frustrado de nuevo, pero lo primero que ha pensado ante esta situación es contactar con la atención al cliente de *Amazon*, como a Sergio nunca le ha gustado contactar con la atención al cliente mediante teléfono porque siempre tiene que esperar mucho, entonces decide hacerlo mediante *chat online*, trás contactar con la atención al cliente de *Amazon* Sergio ha realizado una pre-reserva para que cuando el móvil esté disponible se realiza el pedido de manera automática.

Cinco días después, Sergio cambia de opinión y decide cancelar el pedido. Entra a la página de *Amazon* y se identifica con la cuenta creada cuando hizo la pre-reserva, y después en la sección de mis pedidos Sergio selecciona el pedido y hace clic al botón de cancelar el pedido.

4. Diseño

4.1. Introducción

La importancia del diseño de un producto *software* se basa en que éste será el que modela la interacción entre el usuario y la aplicación y por tanto posibilitará o no la consecución de los objetivos perseguidos por el usuario.

En este proyecto, se ha dado mucha importancia un diseño que centre en la usabilidad y accesibilidad del mismo, es decir conseguir que nuestra aplicación *web* sea evidente, clara y fácil de entender.

Según la definición ofrecida por *ISO*, la usabilidad es el "*grado de eficacia, eficiencia y satisfacción con la que usuarios específicos pueden lograr objetivos específicos, en contextos de uso específicos*".

Como se indica en la definición, la usabilidad de una aplicación debe ser entendida siempre en relación con la forma y condiciones de uso por parte de sus usuarios, así como con las características y necesidades propias de estos usuarios.

El concepto de accesibilidad es un concepto íntimamente ligado al de usabilidad. Éste ya no se refiere a la facilidad de uso, sino a la posibilidad de acceso. En concreto a que el diseño, como prerrequisito imprescindible para ser usable, posibilite el acceso a todos sus potenciales usuarios, sin excluir a aquellos con limitaciones individuales o limitaciones derivadas del contexto de acceso.

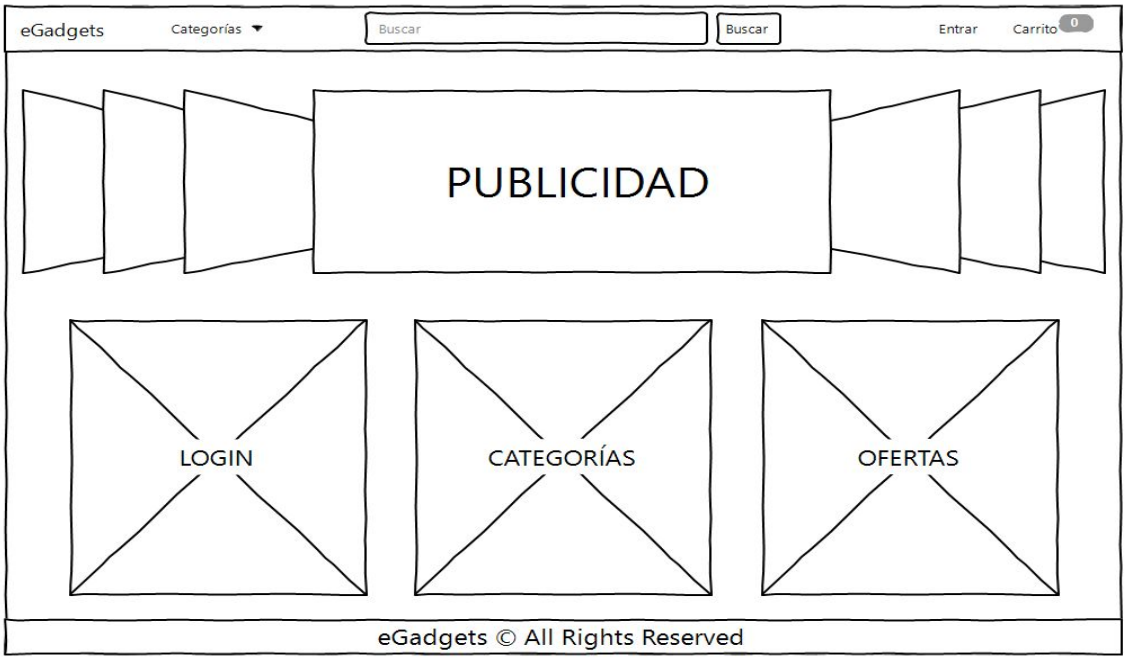
4.2. Bocetos

Los bocetos también son conocidos en inglés como *Sketches* o *Wireframes*, son formas de representar las primeras ideas, ideas sobre lo que se pretende representar, sobre las funcionalidades concretas de la aplicación *web*.

Son usadas en la etapa del diseño, con la finalidad de recoger las primeras impresiones del espacio de trabajo de la interacción.

A continuación vamos a ver los diferentes bocetos realizados para este proyecto, que corresponden a las diferentes páginas del sitio *web*.

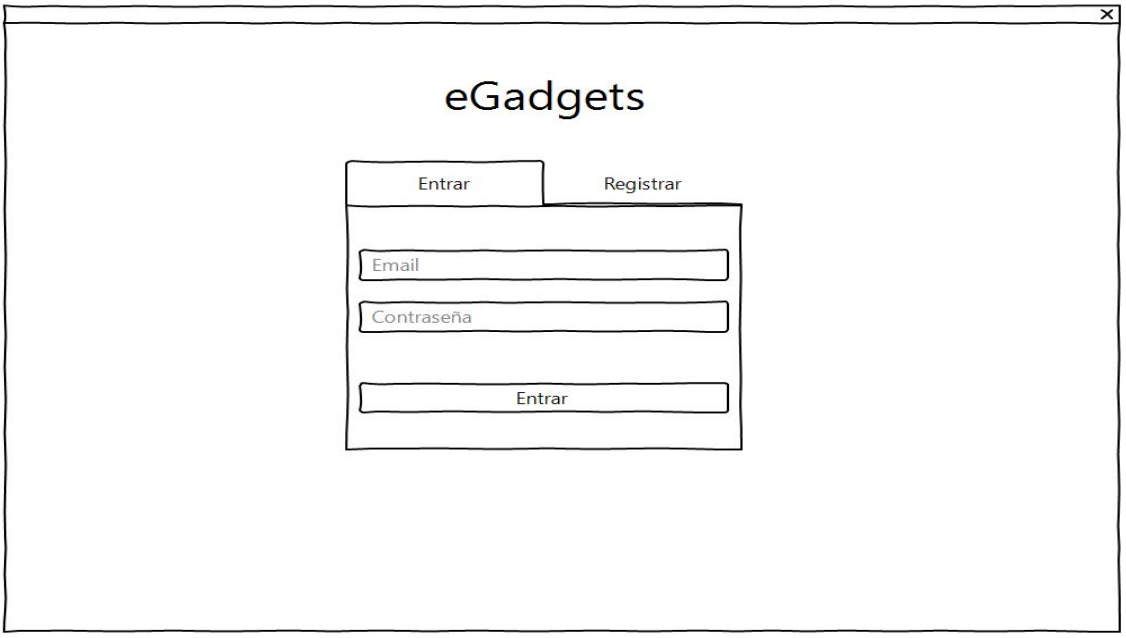
4.2.1. Página principal



La página principal está compuesta por cinco elementos importantes:

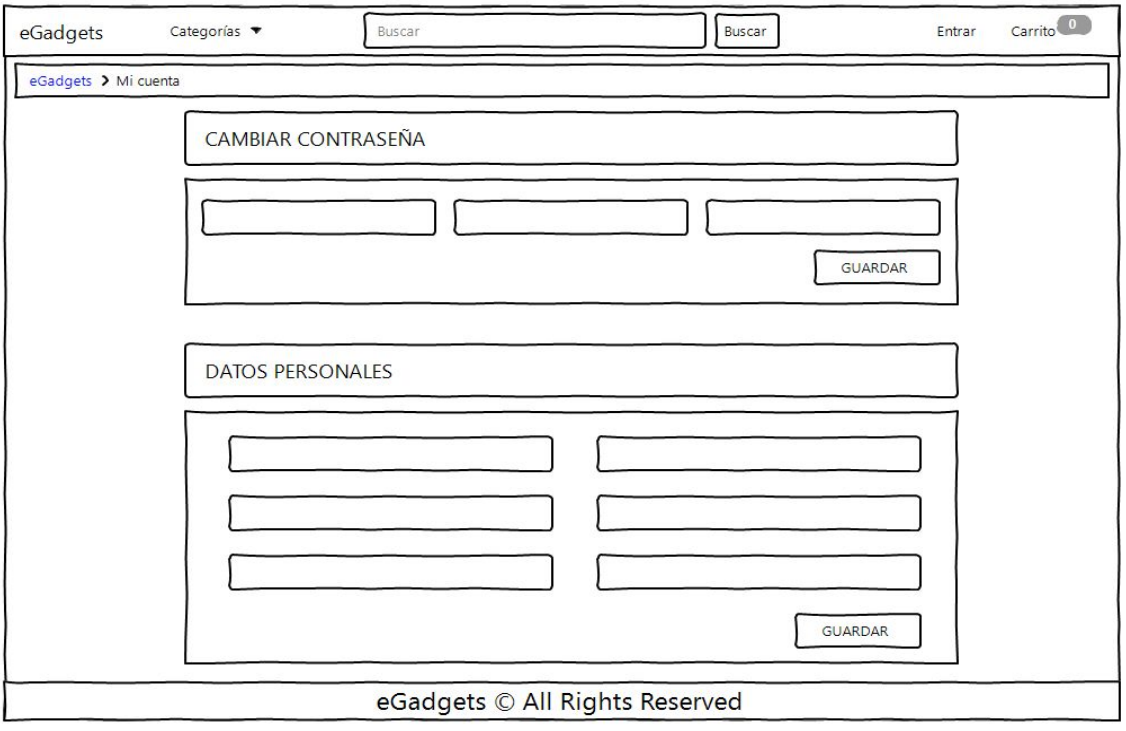
- Barra de navegación, cuya función principal es informar a los usuarios sobre lo que el sitio contiene, les guía y sitúa en el sitio.
- Sección de publicidad, donde muestran las informaciones más relevantes sobre el sitio y sobre el contiene del sitio.
- Sección de identificación.
- Sección de categorías.
- Sección de ofertas, en la que se mostrarían los productos con descuentos o productos nuevos.

4.2.2. Página de identificación



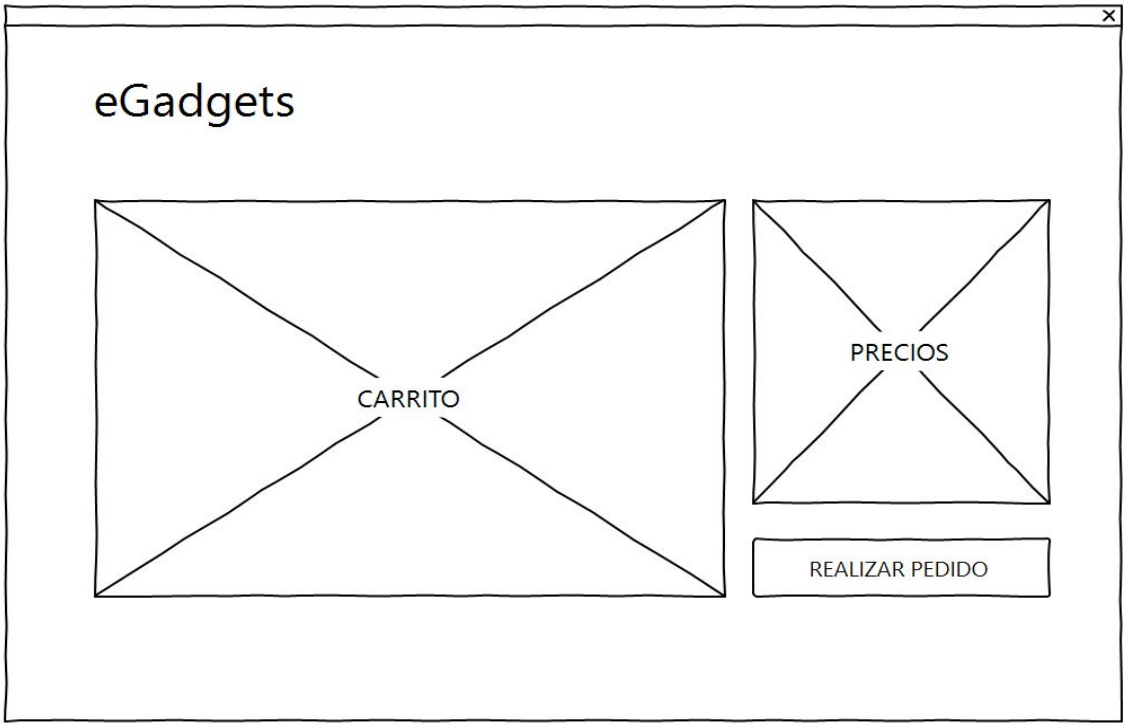
A través de esta página, los usuarios pueden identificarse si ya tiene una cuenta con la aplicación o crear una nueva cuenta si no la tienen.

4.2.3. Configuración de la cuenta

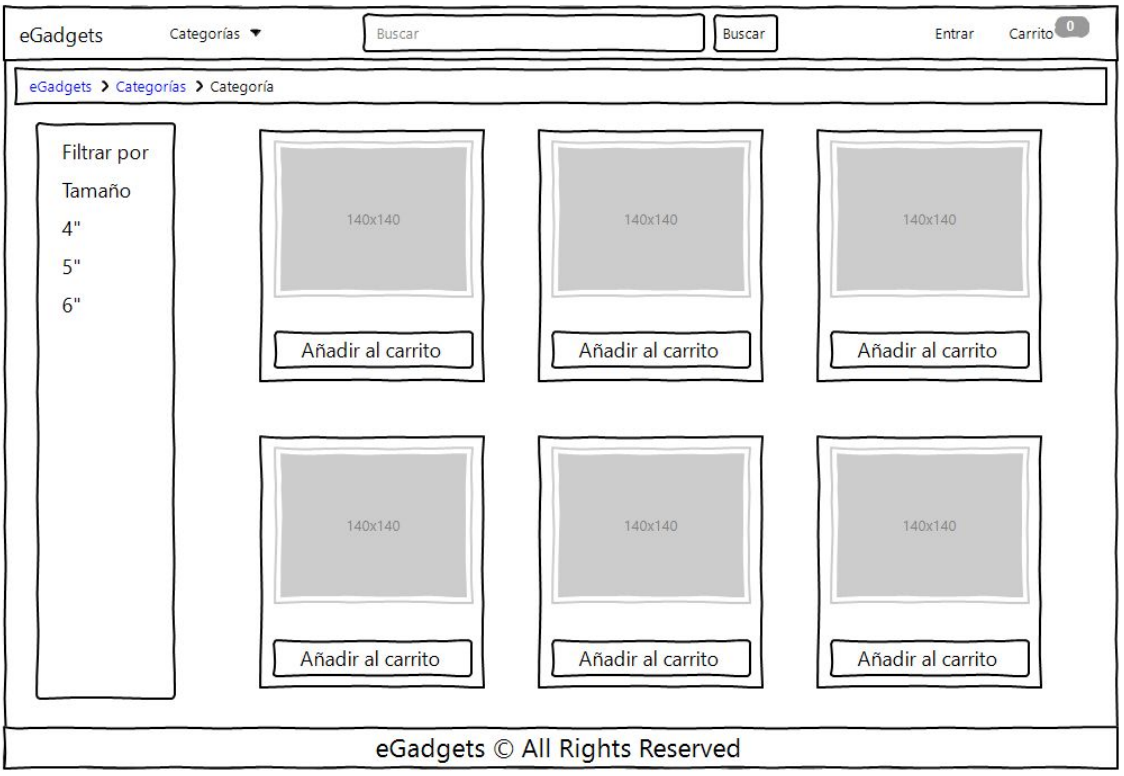


Una vez haya identificado el usuario, se puede modificar la contraseña de la cuenta o añadir datos personales a la cuenta a través de esta página.

4.2.4. Interfaz del carrito



4.2.5. Productos por categorías

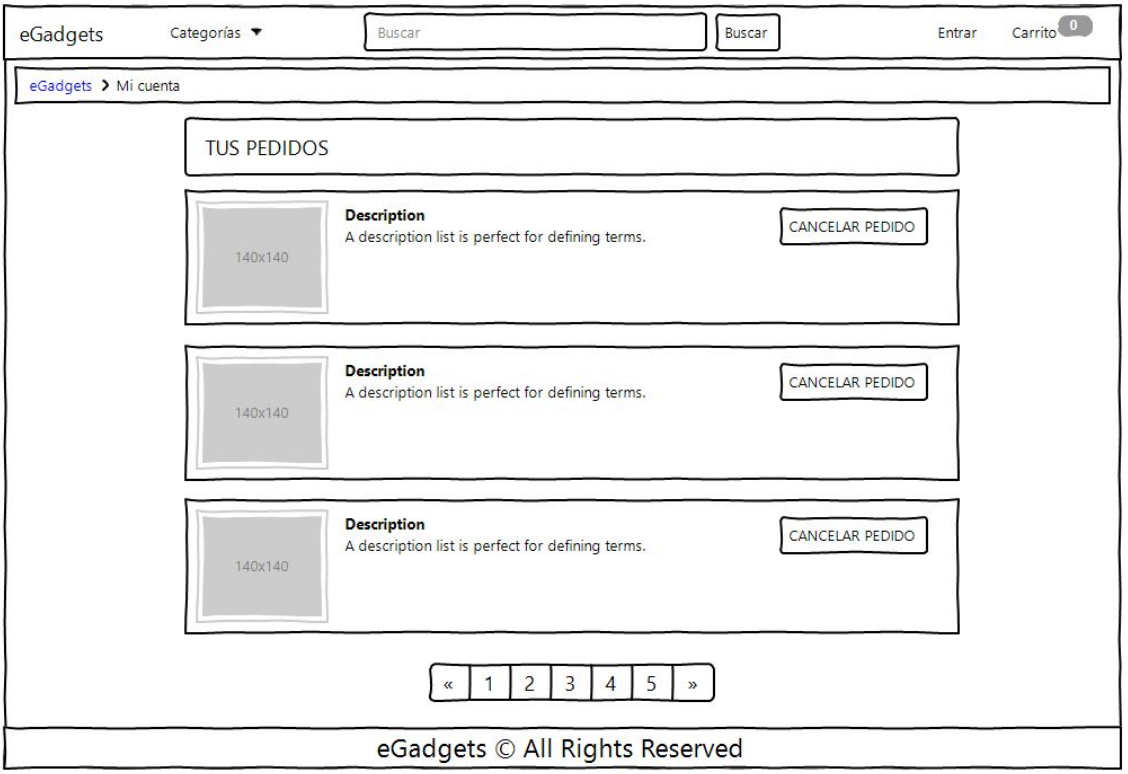


En esta página se muestran los productos según el tipo de categoría seleccionada por el usuario.

A diferencia de la página principal a parte de la barra de navegación esta página ofrece al usuario las siguientes funciones:

- Migas de pan, para que el usuario pueda saber en cualquier momento su ubicación actual dentro del sitio *web*.
- Filtro, para realizar una búsqueda más precisa de productos según las características específicas de los productos.
- Añadir los productos en el carrito para su posterior procesamiento.

4.2.6. Visualización de pedidos



Cuando el usuario haya realizado una compra, a través de esta página se puede hacer un seguimiento de su pedido, también puede cancelar o devolver el pedido si se cambia de opinión.

Al igual que las páginas anteriores, en ésta también se puede encontrar la función migas de pan, y la barra de navegación ambas son considerados dos elementos esenciales para aumentar la usabilidad del sitio *web*.

4.2.7. Detalles del producto

Con esta página el usuario puede saber toda aquella información acerca de un producto concreto, tal como la lista de las especificaciones del producto, y el estado de stock. También puede añadir el producto en el carrito a través de esta página si se desea.

4.3. Arquitectura de la información

La usabilidad de la aplicación no depende solo del diseño de la interfaz, sino también de su arquitectura, la estructura y organización del mismo.

La arquitectura de la información es el resultado de organizar, clasificar, ordenar estructurar y describir los contenidos de producto software, con el fin de que los usuarios puedan satisfacer sus necesidades informativas con el menor esfuerzo posible.

En este proyecto se ha identificado la siguiente estructura de la información:

- Home
    - Barra de navegación
        - Buscador
        - Categorías

        - Smartphones
        - Tablets
        - Smartwatches
        - Accesorios
    - Login
        - Mi cuenta
        - Mis pedidos
    - Carrito
- Categorías
    - Smartphones
    - Tablets
    - Smartwatches
    - Accesorios
- Login
    - Mi cuenta
    - Mis pedidos
- Ofertas

4.4. Diseño de interacción

El diseño de interacción es el que decide las formas de operar la interfaz, los flujos de operación y las respuestas del sistema. En definitiva, pone el foco en el contacto entre el usuario y la aplicación.

En este proyecto, para que las respuestas del sistema sean más intuitivas se ha desarrollado la validación en tiempo real para los formularios de identificación y de registro, que corresponden a las páginas de *login* y mi cuenta.

La validación en tiempo real consiste en la notificación al usuario mediante mensajes, los posibles datos erróneos, cuando el usuario interactúa con los formularios, es decir la introducción de datos en los campos de los formularios.

Ilustración 4-1-Validación en tiempo real de un formulario

5. Implementación

5.1. Introducción

Para la implementación del proyecto se tiene en cuenta todos aquellos requerimientos principales, las necesidades del sitio en referencia al diseño y las interacciones que deben realizarse con el sistema.

La aplicación se ha desarrollado por dos partes separadas, que son *front-end* el lado del cliente y *back-end* el lado del servidor.

Para el lado del cliente se ha usado como principal herramienta de desarrollo el *framework Bootstrap*, un entorno de trabajo de código abierto, que es muy conocido por su sistema GRID también es llamado como sistema de columnas, con el que se puede maquetar sitios *web* por columnas de manera muy sencilla, y es gracias a eso podemos crear sitios *web* responsivos que se adaptan su apariencia según el tamaño de la pantalla de los dispositivos.

Para el lado del servidor, se ha hecho uso del lenguaje de programación PHP, el cual fue diseñado principalmente para desarrollo de páginas *web*.

5.1. Tecnologías de implementación

5.1.1. Front-end

Las tecnologías utilizadas para el desarrollo del lado del cliente son las siguientes:

5.1.1.1. Bootstrap

Bootstrap, es un *framework* originalmente creado por *Twitter*, que permite crear interfaces web con *CSS* y *JavaScript*, cuya particularidad es la de adaptar la interfaz del sitio *web* al tamaño del dispositivo en que se visualice. Es decir, el sitio web se adapta automáticamente al tamaño de un móvil, una tableta u otro dispositivo.

5.1.1.2. HTML

Es conocido como el lenguaje de marcado para hipertextos, es el elemento de construcción más básico de una página *web*, y se usa para crear y representar visualmente una página *web*. Determina el contenido de la página *web*, pero

no su funcionalidad. Los hipertextos son enlaces que conectan a una página *web* con otra, ya sea dentro de una página web o entre diferentes sitios *web*.

5.1.1.3. CSS

Hojas de Estilo en Cascada (*Cascading Style Sheets*) es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura.

CSS es utilizado para dar estilo a documentos *HTML* y *XML*, separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos de una página *web*, cualquier cambio en el estilo marcado para un elemento en la *CSS* afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

5.1.1.4. JavaScript

JavaScript es un lenguaje ligero e interpretado orientado a objetos con funciones de primera clase, más conocido como el lenguaje de *script* para páginas *web*, permite mejoras en la interfaz de usuario y página web dinámicas, pero también es usado en muchos entornos sin navegador, tales como *node.js* o *Apache CouchDB*. Es un lenguaje script multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa.

5.1.2. Back-end

Las siguientes son las tecnologías utilizadas para el desarrollo del lado del servidor:

5.1.2.1. PHP

PHP es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo *web* y que puede ser incrustado en *HTML*.

En lugar de usar muchos comandos para mostrar *HTML* como en *C* o en *Perl*, las páginas *PHP* contiene *HTML* con código incrustado que hace “algo”. El código de *PHP* está encerrado entre las etiquetas especiales de comienzo y final *<?php* y *?>*.

Lo que distingue a *PHP* de algo del lado del cliente como *JavaScript* es que el código es ejecutado en el servidor, generando *HTML* y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que

procese todos los ficheros *HTML* con *PHP*, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga.

5.1.2.2. phpMyAdmin

phpMyAdmin es una herramienta de libre distribución en *PHP*, creado por una comunidad sin ánimo de lucro. Es una programa muy completa que permite acceder a todas las funciones típicas de la base de datos *MySQL* a través de una interfaz web muy intuitiva.

La herramienta en sí no es más que un conjuntos de archivos escritos en PHP que podemos copiar en un directorio de nuestro servidor web, de modo que cuando accedemos a esos archivos, nos muestran unas páginas donde podemos encontrar las bases de datos a las que tenemos acceso en nuestro servidor de base de datos y todas sus tablas.

phpMyAdmin nos permite crear tablas, insertar datos en las tablas existentes, navegar por los registros de las tablas, editarlos y borrarlos, borrar tablas y un largo etcétera, incluso ejecutar sentencias *SQL* y hacer un *backup* de la base de datos.

5.1.2.3. MySQL

MySQL es un sistema de gestión de base de datos relacional muy utilizado en aplicaciones *web*. Su popularidad como aplicación web está muy ligada a *PHP*, que a menudo aparece en combinación con *MySQL*.

5.1.2.4. JSON

JSON, acrónimo de *JavaScript Object Notation*, es un formato de texto ligero para el intercambio de datos.

JSON está construido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

5.2. Implementación detallada

A continuación se va a explicar en detalle la implementación de los componentes esenciales del sitio *web*.

5.2.1. Front-end

5.2.1.1. Estructura responsiva

Es este proyecto se ha usado el sistema *Grid* de Bootstrap para conseguir una estructura responsiva de nuestro sitio *web.*

¿Qué es un sistema *Grid*?

Se trata de una serie de *clases* predefinidas que tiene sus propias declaraciones *CSS* establecidas en la librería de *CSS* que nos proporciona *Bootstrap*.

Una *clase* es un atributo que puede tener cualquier elemento *HTML*, y es con la que podemos estilizar los elementos *HTML* a través de las declaraciones de CSS.

El sistema *Grid* de *Bootstrap* escala el contenido del sitio *web* hasta 12 columnas a medida que aumenta el tamaño del dispositivo o de la ventana de visualización.

Para conseguir la adaptabilidad que buscamos hay que usar las siguientes *clases* para los contenidos de nuestro sitio web:

- ***.container***, *clase* contenedor cuya anchura es fija, que tiene una margen preestablecida en el lado izquierdo y el lado derecho.
- ***.container-fluid***, *clase* contenedor cuya anchura es completa a la pantalla del dispositivo o la ventana de visualización.
- ***.row***, clase fila.
- ***.col-xs-****, clase columna, el prefijo xs indica que es para los dispositivos extra pequeños como los móviles cuya pantalla es menor que 768 píxeles.
- ***.col-sm-****, clase columna, el prefijo sm indica que es para los dispositivos pequeños, como los tablets cuya pantalla es mayor o igual que 768 píxeles.
- ***.col-md-****, clase columna, el prefijo md indica que es para los dispositivos medianos, como los portátiles cuya pantalla es mayor o igual que 992 píxeles.

- ***col-lg-****, clase columna, el prefijo lg indica que es para los dispositivos grandes como los ordenadores de sobremesa cuya pantalla es mayor o igual que 1200 píxeles.

¿Pero cómo funcionan exactamente esas *clases* del sistema *Grid*?

Cuando queremos que un contenido de nuestro sitio web sea adaptativo, ese contenido tiene que irse dentro de una *clase* columna (.col-*-*), a su vez la *clase* columna tiene que estar dentro de una *clase* fila (.row) , y esta *clase* fila tiene que permanecer dentro de una *clase* de contenedor ya sea un contenedor de anchura fija (.container) o un contenedor de anchura completa (.container-fluid).

Veamos un ejemplo de la implementación responsiva de un contenido de la página principal de este proyecto:

```
41      <div class="container" id="info-wraper">
42          <!-- Row 2 -->
43          <div class="col-xs-12 col-sm-6 col-md-4 col-lg-4 mainSesion-wraper">
73          </div>
74          <div class="col-xs-12 col-sm-6 col-md-4 col-lg-4 categories-wraper">
116         </div>
117         <div class="col-xs-12 col-sm-12 col-md-4 col-lg-4 ofertas-wraper">
143         </div>
144         <!-- Row 2 -->
145     </div>
```

Ilustración 5-1 Ejemplo de la implementación responsiva con Bootstrap

Con las *clases .col-xs-12 .col-sm-6 .col-md-4* y *col-lg-4* estamos diciendo que cuando la pantalla del dispositivo es menor que 768 píxeles cada *div* ocuparía una fila entera de espacio, es decir 12 columnas, cuando la pantalla del dispositivo es mayor o igual que 768 píxeles y menor que 992 píxeles cada *div* ocuparía un espacio de 6 columnas, que es equivalente la mitad de una fila, y por último cuando la pantalla del dispositivo es mayor o igual que 992 píxeles cada *div* ocuparía un tercio de una fila, es decir 4 columnas.

Podemos ver el resultado de esta implementación adaptativa en las siguientes imágenes:

Los *divs* mencionados anteriormente corresponden a las secciones de identificación, Categorías y Ofertas.

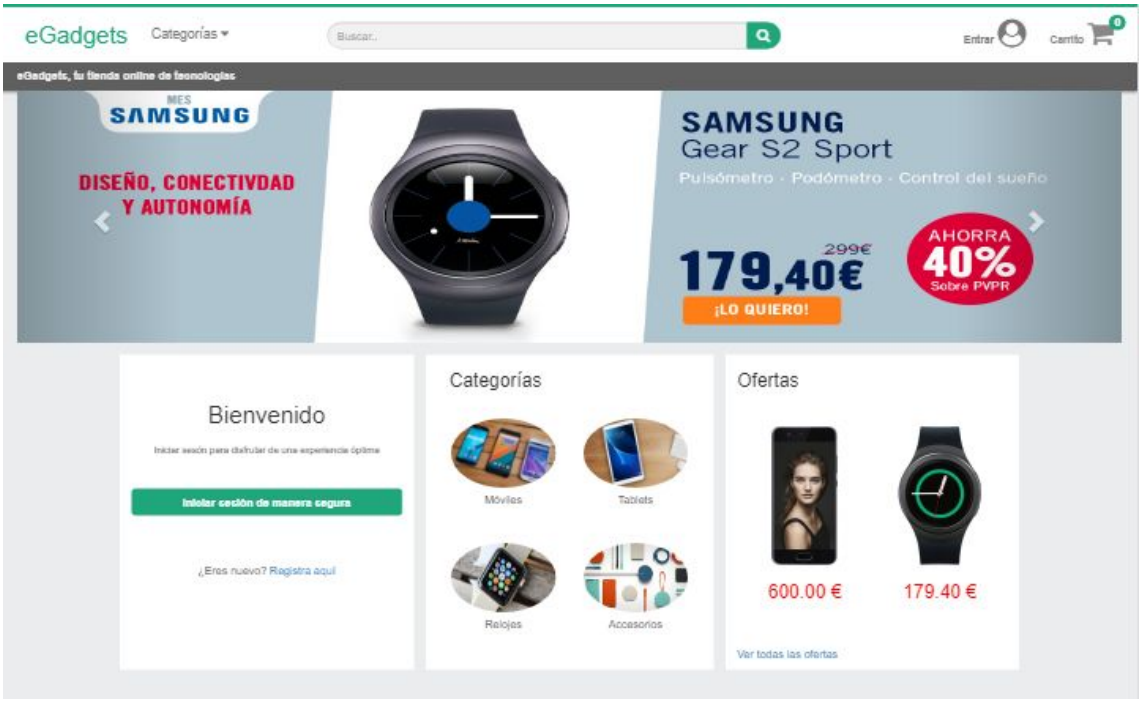


Ilustración 5-2 Vista de la estructura responsiva desde un ordenador

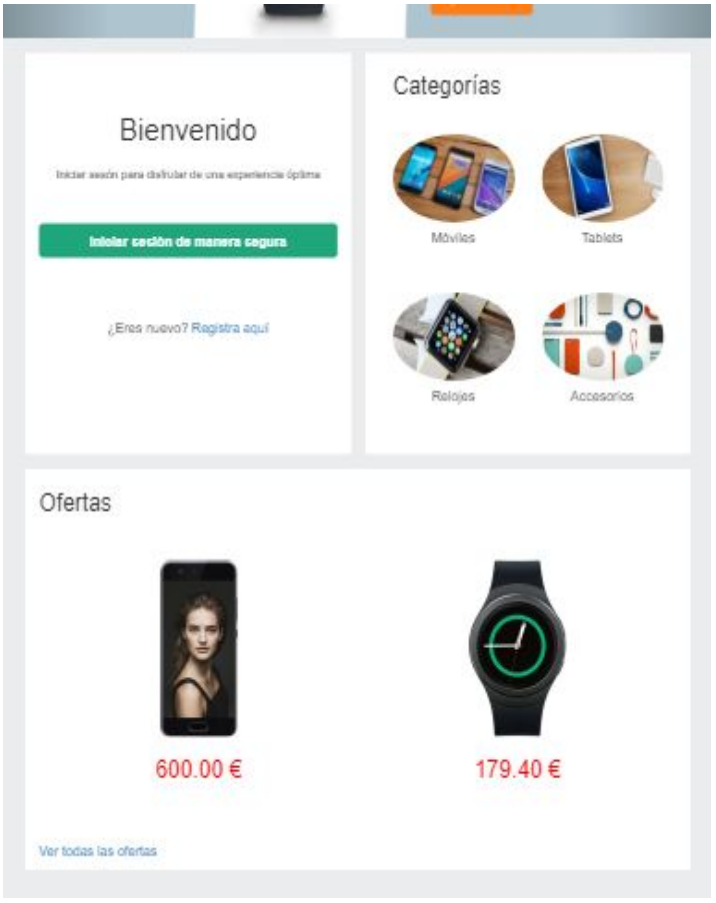


Ilustración 5-3 Vista de la estructura responsiva desde una tableta

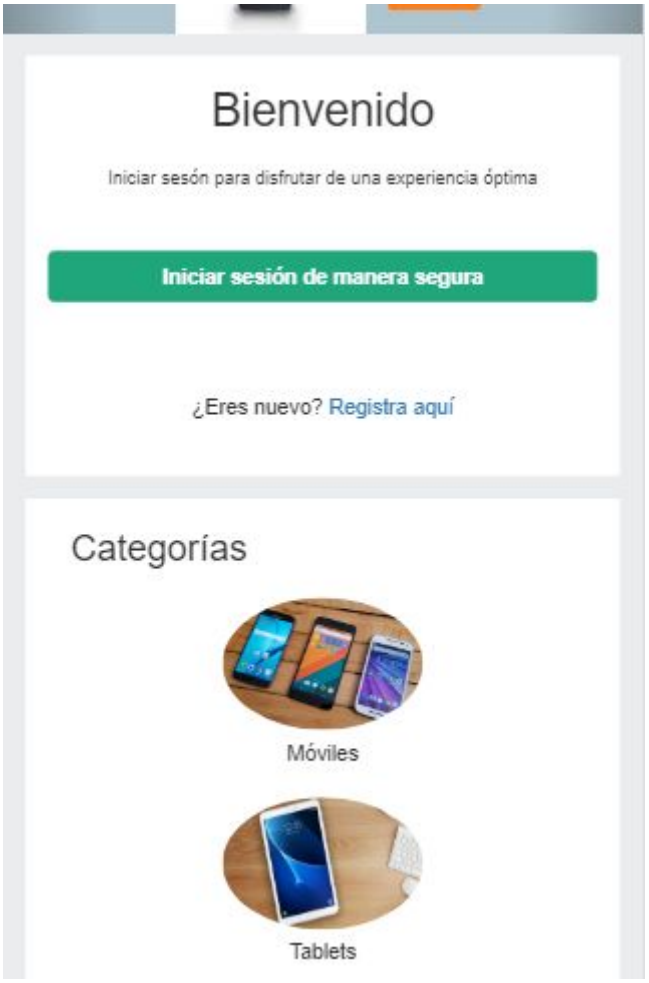


Ilustración 5-4 Vista de la estructura responsiva desde un dispositivo móvil

5.2.1.2. Barra de navegación

A parte del sistema *Grid, Bootstrap* también nos ofrece plantillas de diseño. En este proyecto se ha desarrollado la barra de navegación usado una de las plantillas de diseño.

La razón por la que se ha usado esta plantilla de barra de navegación es debido a que también ofrece adaptabilidad, cuando la pantalla es menor que 768 píxeles, todas las opciones del menú se colapsan, y se mostrarían cuando le damos al botón del menú colapsado.

Para usar la plantilla de barra de navegación basta con llamar las clases predefinidas *.nav* y *.nav-default* en nuestro código HTML.

Para que el menú se colapse de manera automática es necesario añadir las *clases .collapse* y *. navbar-collapse* en el código *HTML* de nuestra barra de navegación.

```
34    <!-- Fixed Navbar -->
35    <nav class="nav navbar-default navbar-fixed-top" id="miNavbar">
36      <div class="container-fluid">
37
38        <!-- Navbar-header 1 -->
39        <div class="navbar-header">
68        </div>
69        <!-- End Navbar-header 1 -->
70
71        <!-- Navbar-header 2 -->
72        <div class="navbar-header pull-right right_brands">
79        </div>
80        <!-- End Navbar-header 2 -->
81
82        <!-- Menu -->
83        <div class="collapse navbar-collapse" id="menuColapsado">
167       </div>
168       <!-- End menu -->
169     </div>
170     <!-- End Container -->
171   </nav>
172   <!-- End Fixed Navbar -->
```

Ilustración 5-5 Código HTML de la barra de navegación

Se puede observar en la imagen anterior una estructura jerárquica clara de la barra de navegación, la barra de navegación en sí, las cabeceras y el menú.

Una barra de navegación en *Bootstrap* puede contener una o varias cabeceras, normalmente se usa para colocar el logo del sitio *web*, el contenido de la cabecera no se colapsa, se muestra siempre, da igual el tamaño de la pantalla con la que estamos visualizando el sitio *web*.

En este proyecto se ha implementado dos cabeceras, una para situar el logo de la tienda virtual y otra para situar el carrito de nuestra tienda, ya que el carrito es un componente esencial de la tienda virtual y debe de estar en presente en todo momento. También se ha implementado un menú colapsado, que contiene las diferentes categorías de la tienda y la opción que da acceso a la identificación de usuario.

Ahora veamos el resultado final de la implementación de nuestra barra de navegación:

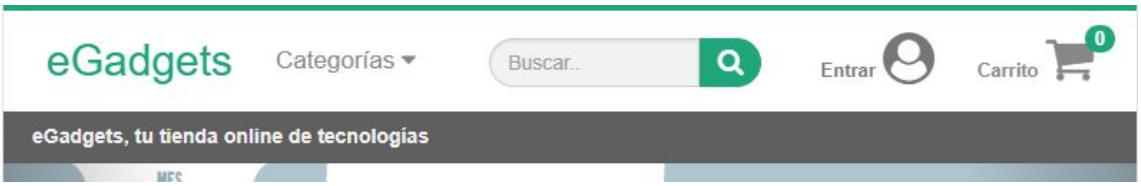


Ilustración 5-6 Vista de la barra de navegación desde una pantalla mayor que 768 píxeles

Ilustración 5-7 Vista de la barra de navegación colapsada

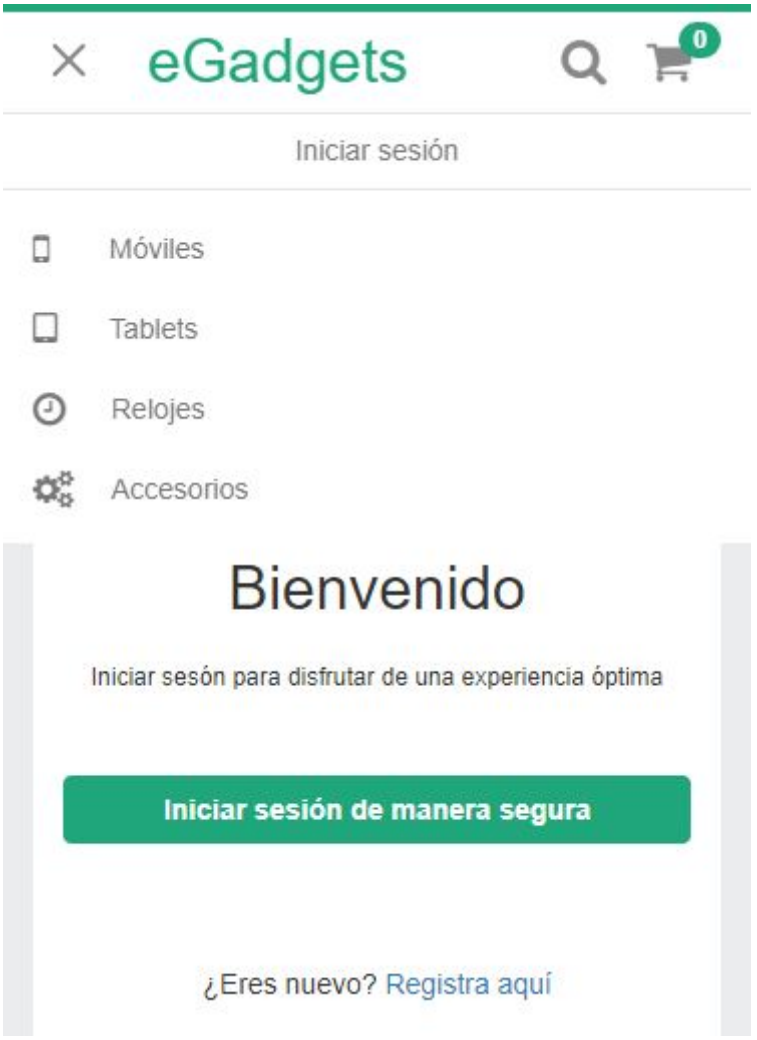


Ilustración 5-8 Vista del menú colapsado abierto

5.2.1.3. Sección de publicidad

Otro de los componentes esenciales del sitio *web*, es la sección de publicidad, con la que se da información relevante sobre productos destacados de la tienda virtual.

Se ha desarrollado esta sección utilizando el componente JavaScript llamado *Carousel.js* que nos proporciona Bootstrap, que también es conocido como *Slider*.

Un *slider* es una forma espectacular de transición y animación para las imágenes.

Básicamente el *Slider* está compuesto por tres elementos, el contenedor de *slides*, el indicador de *slides* y el controlador para avanzar or retroceder los *slides*.

Para usar este componente de *JavaScript* de *Bootstrap*, es necesario añadir las siguientes *clases* en nuestro código *HTML*:

- *.carousel*
- *.slide*
- *.carousel-indicators*
- *.carousel-inner*
- *.left*
- *.right*
- *.carousel-control*

Veamos a continuación la implementación de la sección de publicidad:

```
<div class="carousel-inner">
    <div class="item active">
        <a href="fichaAmpliada.php?id=9"><img class="slide-image"
        src="images/slide1.jpg" alt=""></a>
    </div>
    <div class="item">
        <a href="fichaAmpliada.php?id=7"><img class="slide-image"
        src="images/slide2.jpg" alt=""></a>
    </div>
    <div class="item">
        <a href="fichaAmpliada.php?id=10"><img class="slide-image"
        src="images/slide3.jpg" alt=""></a>
    </div>
</div>
```

Ilustración 5-9 Código HTML del contenedor de slides

```
<ol class="carousel-indicators">
    <li data-target="#mySlider" data-slide-to="0" class="active"></
    li>
    <li data-target="#mySlider" data-slide-to="1"></li>
    <li data-target="#mySlider" data-slide-to="2"></li>
</ol>
```

Ilustración 5-10 Código HTML del indicador de slides

```
<a class="left carousel-control" href="#mySlider" data-slide="prev"
>
    <span class="glyphicon glyphicon-chevron-left"></span>
</a>
<a class="right carousel-control" href="#mySlider" data-slide="next
">
    <span class="glyphicon glyphicon-chevron-right"></span>
</a>
```

Ilustración 5-11 Código HTML del controlador del slider

Ilustración 5-12 Vista final del slider

5.2.1.4. Visualizador de imágenes

El componente *Modal* de *Bootstrap* también es conocido como ventana emergente es el que se ha usado en este proyecto para crear un visualizador de imagenes para los productos de la tienda virtual.

Al igual que un *Slider*, el Modal es un componente *JavaScript*, para usarlo es necesario incluir las *clases* predefinidas de *Bootstrap* en nuestro código *HTML*.

El visualizador está compuesto por dos elementos diferenciables, la cabecera del *Modal* que contiene el título del producto, y la función de cancelación de la ventana emergente, y el cuerpo del *Modal*, que contiene las imágenes del producto en cuestión.

Se ha creado un pequeño *script* para dar la función de cambiar de imágenes al visualizador.

Veamos a continuación la implementación del visualizador:

```
<div class="modal-header">
    <button type="button" class="close" data-dismiss="modal"
     aria-label="Close"><span aria-hidden="true">&times;</
    span></button>
    <h4 class="modal-title" id="myModalLabel">Galería</h4>
</div>
```

Ilustración 5-13 Código HTML de la cabecera del Modal

```
<div class="modal-body">
    <div class="container-fluid">
        <div class="row">
            <div id="gallery-wraper" class="col-xs-12
            gallery-wraper" style="background-image: url('
                <?php echo $row['imagen2']; ?>');">
            </div>
            <div class="col-xs-12 gallery-thumbs">
                <a href="#" onclick="bg1()">
                    <div class="thumb" style="
                        background-image: url('<?php echo $
                        row['imagen2']; ?>');"></div>
                </a>
                <a href="#" onclick="bg2()">
                    <div class="thumb" style="
                        background-image: url('<?php echo $
                        row['imagen3']; ?>');"></div>
                </a>
            </div>
        </div>
    </div>
</div>
```

Ilustración 5-14 Código HTML del cuerpo del Modal

```
<script type="text/javascript">
    function bg1(){
        document.getElementById("gallery-wraper").style.
            backgroundImage = "url('<?php echo $row['imagen2
            ']; ?>')";
    }
    function bg2(){
        document.getElementById("gallery-wraper").style.
            backgroundImage = "url('<?php echo $row['imagen3
            ']; ?>')";
    }
</script>
```

Ilustración 5-15 Código JavaScript que implementa la función de cambio de imágenes

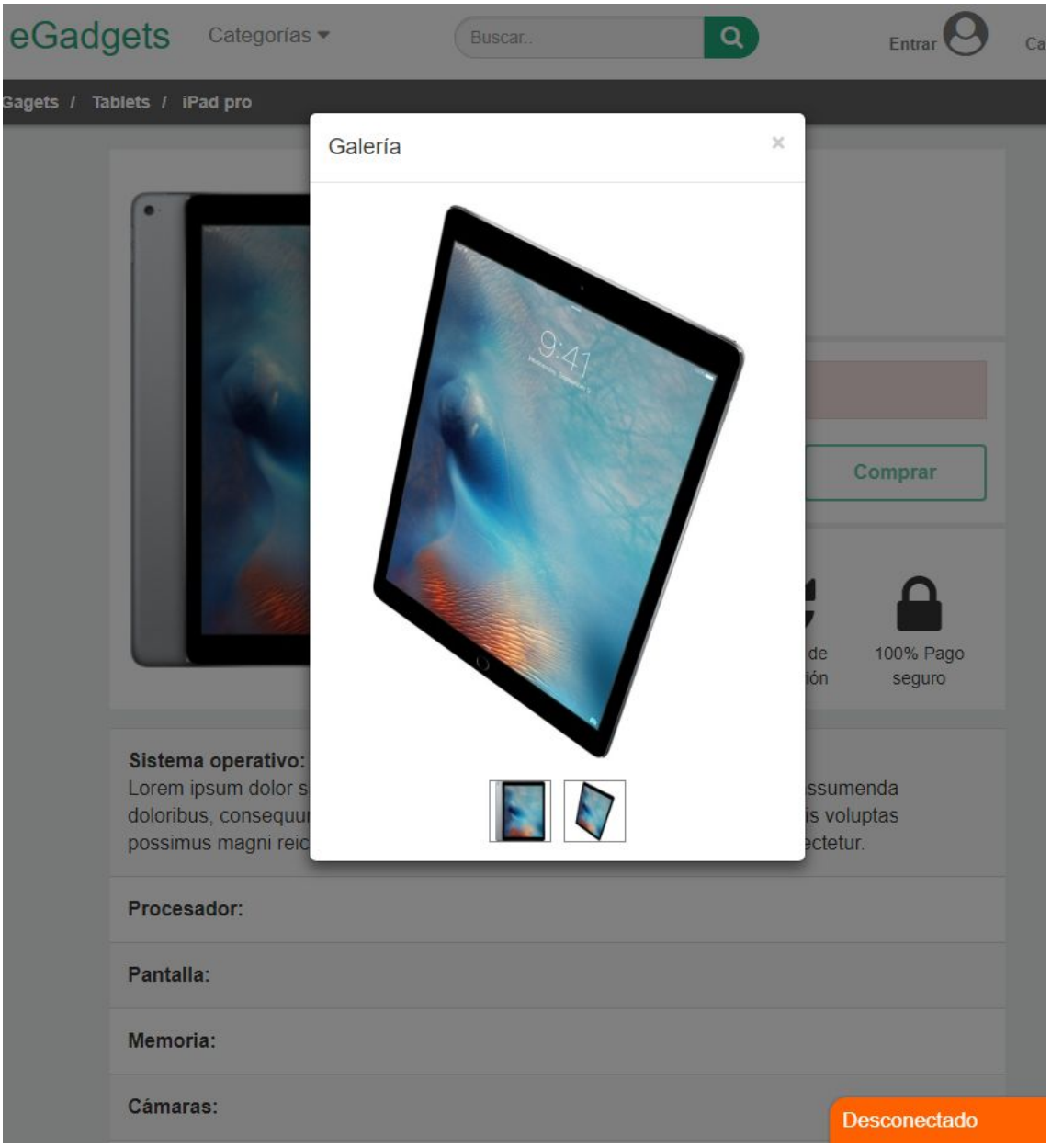


Ilustración 5-16 Vista final del visualizador de imágenes

5.2.1.5. Validación de formularios

Para crear el sitio *web* de carácter comercio electrónico se ha hecho uso de formularios en varias ocasiones, y como uno de los objetivos obtenidos tras la fase de diseño es conseguir una interacción intuitiva entre el usuario y el sistema, se ha implementado un sistema de validación en tiempo real.

Se ha usado un *plugin* hecho con *JavaScript* llamado *Bootstrap Validator* para crear dicho sistema.

Al igual que *Bootstrap* este *plugin* ofrece clases predefinidas con las que se puede conseguir las funciones que se desea obtener.

Para habilitar las funciones de este *plugin*, se puede hacer via *data-api* o via *JavaScript*. En este proyecto se ha seguido el método de *data-api* para habilitar las funciones del *plugin Validator,* se consigue añadiendo el atributo *data-toggle= "validator"* a nuestro formulario.

A la hora de desarrollar este sistema, es importante tener en cuenta que todos los campos *input* deben estar dentro de la clase *.form-group* para poder visualizar de forma correcta los mensajes de error, y si se quiere mostrar mensajes personalizados, hace falta incluir las *clases .has-feedback* y *.data-*-error* en los elementos *inputs.*

La mayoría de las reglas de validación están basadas en los atributos del estándar de HTML5:

- *type = "email"*
- *type = "url"*
- *type = "number"*, con restricciones adicionales vía atributos *max*, *min* y step.
- *pattern = "Reg(ular)?Exp(ression)"*, comparación de patrones usando las expresiones regulares, es aplicable para los tipos *text*, *search*, *tel*, *url* y *email*.

Reglas de validación basadas en atributos no estándares:

- *data-match = "#inputToMatch"*, se usa para confirmación de contraseñas.
- *data-minlength = "5"*, para forzar la mínima cantidad de caracteres.

Veamos a continuación la implementación del formulario de la página de identificación con la validación en tiempo real.

```
<form data-toggle="validator" role="form" action="login.php
" method="post">

    <div class="alert alert-success" id="registro-exito"
    role="alert">Has registrado con exito!</div>
    <div class="alert alert-danger" id="registro-fallido"
    role="alert">El correo está usado!</div>
    <div class="alert alert-danger" id="login-fallido" role
    ="alert">Contraseña incorrecta o usuario no está
    registrado!</div>

    <div class="form-group has-feedback">
        <input type="text" pattern="^[a-zA-Z0-9_.+-]+@[
        a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$" class="form-control"
         placeholder="Email*" data-required-error="Campo
        obligatorio" data-pattern-error="Formato incorrecto
        " required name="lemail">
        <span class="glyphicon form-control-feedback"
        aria-hidden="true"></span>
        <div class="help-block with-errors"></div>
    </div>

    <div class="form-group has-feedback">
        <input type="password" class="form-control"
        placeholder="Contraseña*" required
        data-required-error="Campo obligatorio" name="lpsw"
        >
        <span class="glyphicon form-control-feedback"
        aria-hidden="true"></span>
        <div class="help-block with-errors"></div>
    </div>

    <div class="form-group">
        <button type="submit" class="btn btn-default">
        Entrar</button>
    </div>

</form>
```

Ilustración 5-17 Código HTML de la validación en tiempo real

5.2.2. Back-end

A continuación veamos la implementación del lado del servidor de la tienda virtual.

5.2.2.1. Acceso a MySQL

El presente proyecto tiene una base de datos que está formada por tres tablas de datos, se encarga de almacenar la información relevante de la tienda virtual. Veamos a continuación las sentencias SQL usadas para crear dichas tablas.

```
CREATE TABLE `users` (
  `id` int(10) UNSIGNED NOT NULL,
  `user` varchar(128) COLLATE utf8_spanish_ci NOT NULL,
  `psw` varchar(128) COLLATE utf8_spanish_ci NOT NULL,
  `nombre` varchar(255) COLLATE utf8_spanish_ci DEFAULT NULL,
  `apellidos` varchar(255) COLLATE utf8_spanish_ci DEFAULT NULL,
  `direccion` varchar(255) COLLATE utf8_spanish_ci DEFAULT NULL,
  `provincia` varchar(255) COLLATE utf8_spanish_ci DEFAULT NULL,
  `poblacion` varchar(255) COLLATE utf8_spanish_ci DEFAULT NULL,
  `cp` int(255) DEFAULT NULL,
  `tel` int(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;

CREATE TABLE `productos` (
  `id` int(8) NOT NULL,
  `tipo` varchar(255) COLLATE utf8_spanish_ci NOT NULL,
  `nombre` varchar(255) COLLATE utf8_spanish_ci NOT NULL,
  `imagen1` varchar(255) COLLATE utf8_spanish_ci DEFAULT NULL,
  `imagen2` varchar(255) COLLATE utf8_spanish_ci DEFAULT NULL,
  `precio` double(10,2) NOT NULL,
  `marca` varchar(255) COLLATE utf8_spanish_ci NOT NULL,
  `cantidad` int(255) NOT NULL,
  `estado` varchar(255) COLLATE utf8_spanish_ci NOT NULL,
  `imagen3` varchar(255) COLLATE utf8_spanish_ci NOT NULL,
  `key_words` varchar(255) COLLATE utf8_spanish_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;

CREATE TABLE `pedidos` (
  `id` int(8) NOT NULL,
  `product_id` int(8) NOT NULL,
  `user` varchar(255) COLLATE utf8_spanish_ci NOT NULL,
  `date` varchar(255) COLLATE utf8_spanish_ci NOT NULL,
  `price` double(10,2) NOT NULL,
  `status` varchar(255) COLLATE utf8_spanish_ci NOT NULL,
  `quantity` int(8) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;
```

Para acceder a la base de datos MySQL y obtener información acerca de los productos de la tienda lo primero que hay que hacer es establecer una conexión el servidor web con la base de datos.

Se ha creado un archivo llamado *database.php* a propósito para almacenar las variables que contienen las credenciales con las que se puede establecer la conexión el servidor con la base de datos.

```php
<?php
    $db_hostname = 'localhost';
    $db_database = 'egadgets';
    $db_username = 'root';
    $db_password = '';
?>
```

Ilustración 5-18 Fichero database.php

Para establecer la conexión basta con incluir este archivo en la página que quiera establecer dicha conexión y posteriormente llamando la función *mysqli_connect()* de *PHP* con las credenciales obtenidas.

```php
require_once 'includes/database.php';
$db_server = mysqli_connect($db_hostname, $db_username, $
  db_password, $db_database);
```

Ilustración 5-19 Código PHP: establecer conexión con MySQL

Una vez establecida la conexión, las consultas, modificaciones, actualizaciones y eliminaciones se realizan mediante la función *mysqli_query()* de *PHP*.

Veamos un ejemplo de la consulta a la base de datos mediante código *PHP*:

```php
$query = "SELECT * FROM users WHERE user = '$user'";
$result = mysqli_query($db_server, $query);
$row = mysqli_fetch_row($result);
```

Ilustración 5-20 Código PHP: Consulta SQL

La función *mysqli_fetch_row()* genera un *array* con los resultados obtenidos tras la consulta a la base de datos para su posterior procesamiento.

5.2.2.2. Transferencia de datos

Para implementar la transferencia de datos entre el lado del cliente y el lado del servidor de nuestro sitio *web*, se ha hecho uso de dos métodos de *HTTP, GET y POST.*

HTTP es conocido como el Protocolo de Hipertexto de Transferencia, es diseñado para establecer conexión entre clientes y servidores.

HTTP funciona como un protocolo de solicitud-respuesta entre un cliente y un servidor.

La diferencia entre envío de datos vía *GET* y *POST* es que con el método *POST* los datos enviados no son visibles para los clientes ni tampoco se almacenan en el *cache* del navegador, mientras con el método *GET* sí son visibles y los datos enviados son almacenados en el cache del navegador, por lo tanto es más seguro usar el método *POST*, cuando se trata de datos sensibles y privados.

Para poder el servidor procesar los datos enviados por el cliente, *PHP* el lenguaje del lado del servidor tiene dos variables reservadas específicamente para este uso, que son *$_GET[ ]* y *$_POST[ ]*, dos variables de tipo array.

5.2.2.3. Sesión

Una sesión en *PHP* es una forma de almacenar información en variables para ser usada a través de diferentes páginas de un sitio *web*.

A diferencia de *cookie*, la información no es almacenada en el ordenador de los usuarios sino en el servidor.

Cuando trabajamos con una aplicación web, la abrimos, hacemos cambios en ella y la cerramos. Eso es una sesión realizada. El ordenador sabe quién eres, sabe cuándo la abriste y cuándo la cerraste, pero existe un problema cuando se trata del mundo de *Internet*, el problema radica en que el servidor *web* no sabe quién eres ni qué haces con la *web*, ya que la dirección *HTTP* no se mantiene estados.

Para solucionar el problema mencionado, se han surgido las variables de sesión una de las características más importantes de *PHP*.

Las variables de sesión almacenan la información del usuario para ser usada a través de múltiples páginas de un sitio *web*. Por defecto, las variables de sesión se caducan cuando el usuario cierra el navegador.

Para que nuestra aplicación web pueda iniciar una sesión o acceder las variables de sesión ya creadas, es necesario que el servidor web llame la función *session_start()* de PHP antes de imprimir la página web.

Las variables de sesión son establecidas mediante la variable global *$_SESSION*, se trata de una variable de tipo array, que almacena el valor de todas aquellas variables de sesión.

Veamos a continuación la implementación de dos sistemas esenciales de este proyecto que se ha hecho uso de sesiones de *PHP*.

5.2.2.4. Carrito de compra

El sistema de carrito de compra se presenta en nuestra tienda virtual mediante dos componentes, un indicador situado en la barra de navegación que sirve para indicar la cantidad de productos que hay en la cesta del usuario y una interfaz en la que muestra en detalle todos aquellos productos en la cesta del usuario, que también ofrece la opción de proceder al proceso de compra de dichos productos.

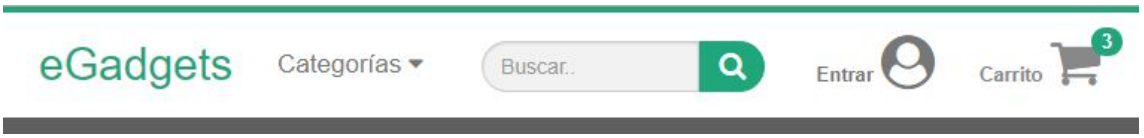


Ilustración 5-21 Vista del indicador del carrito

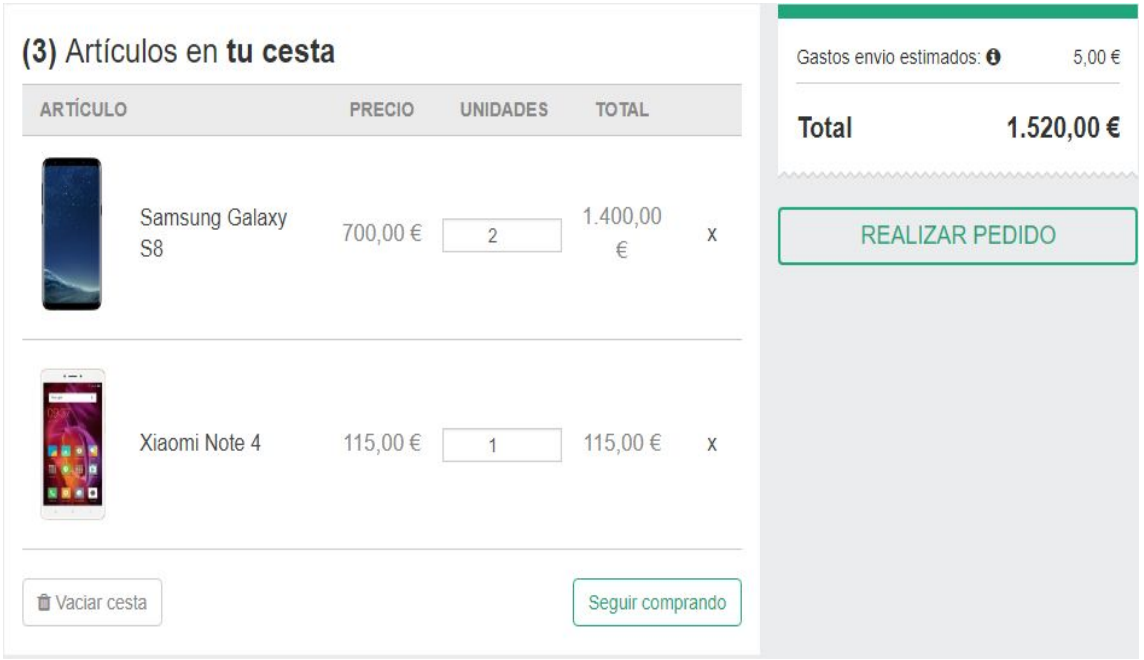


Ilustración 5-22 Vista de la interfaz del carrito

Para la implementación del sistema de carrito se ha creado una variable de sesión *$_SESSION[“cart_items”]* para almacenar los productos, cada producto es almacenado como un *array* en la variable de sesión, cada array contiene el nombre del producto, su precio, su cantidad, su identificador y la ubicación de las imágenes correspondientes del producto.

Para obtener la información de la cesta del usuario basta recorrer el array bidimensional *$_SESSION[“cart_items”].*

La interfaz de la cesta no solo ofrece la función de visualización de los productos sino también la modificación de los mismos. Se puede modificar la cantidad de cada producto de la cesta, eliminar un producto concreto o simplemente vaciar la cesta entera. La implementación de todas estas funciones se realiza a través del procesamiento de la variable de sesión *$_SESSION[“cart_items”].*

5.2.2.5. Identificación y Registro

Otro de los sistemas esenciales de nuestra tienda virtual es el sistema de registro e identificación.

Cuando un usuario se registra, se le envía al servidor mediante el protocolo *HTTP* todos sus datos a través de un formulario, el servidor recoge esos datos y hace una comparación con los usuarios existentes de la base de datos mediante una consulta *SQL*, si resulta que el usuario que intenta registrar ya está registrado se le notifica al usuario mediante un mensaje de aviso en la propia interfaz del sistema de registro, el caso contrario se almacena el nombre y la contraseña del usuario en la base de datos.

Cuando un usuario se identifica, al igual que el proceso de registración, se le envía al servidor el nombre del usuario y la contraseña mediante el protocolo *HTTP,* el servidor coge el nombre del usuario, hace una consulta SQL a la base de datos para comprobar la existencia de dicho usuario, si el usuario existe, se hace una comprado de la contraseña introducida por el usuario y la contraseña almacenada en la base de datos, si las contraseñas coinciden, se crea una variable de sesión *$_SESSION[“user”]* para almacenar el identificador del usuario identificado.

Una vez se ha identificado el usuario, ya puede realizar las operaciones deseadas con su sesión.

Veamos ahora el código de implementación del sistema de registro e identificación.

```php
<?php
if (isset($_POST['lemail']) && isset($_POST['lpsw'])) {
    $user = $_POST['lemail'];
    $password = $_POST['lpsw'];
    $query = "SELECT * FROM users WHERE user = '$user'";
    $result = mysqli_query($db_server, $query);
    $row = mysqli_fetch_row($result);

    if ($row[2] == $password){
        $_SESSION['user'] = $row[1];
        if (isset($_SESSION['cart_items'])) {
?>
        <script type="text/javascript">
            window.location.replace("carrito.php");
        </script>
<?php
        }else{
?>
        <script type="text/javascript">
            window.location.replace("index.php");
        </script>
<?php
        }
    }else{
?>

<script type="text/javascript">
    document.getElementById("login-fallido").style.display = "
        block";
</script>

<?php
    }
}
?>
```

Ilustración 5-23 Código PHP: Sistema de identificación

```php
<?php
if (isset($_POST['remail']) && isset($_POST['rpsw'])) {

    $user =  $_POST['remail'];
    $password = $_POST['rpsw'];
    $query = "SELECT * FROM users WHERE user = '$user'";
    $result = mysqli_query($db_server, $query);
    $row = mysqli_fetch_row($result);

    if($row[1] == $user){
?>

<script type="text/javascript">
    document.getElementById("registro-fallido").style.
        display = "block";
</script>

<?php
    }else{
        $query = "INSERT INTO users (user, psw) VALUES ('$
            user', '$password')";
        mysqli_query($db_server, $query);
?>

<script type="text/javascript">
    document.getElementById("registro-exito").style.display
        = "block";
</script>

<?php
    }
}
?>
```

Ilustración 5-24 Código PHP: Sistema de registro

5.2.2.6. Procesamiento de pago

El sistema de procesamiento de pago está implementado integrando el servicio de terceros llamado *Stripe*.

Stripe es una plataforma de procesamiento de pago, que permite a particulares y empresas aceptar pagos a través de *Internet*. Se enfoca en proveer la infraestructura técnica, de prevención de fraude y bancaria necesaria para operar sistemas de pago en línea.

La plataforma *Stripe* ofrece una variedad de métodos de integración de su servicio. En este proyecto se ha optado por usar el método de Checkout y *PHP.*

Para poder usar el servicio de procesamiento de pago de *Stripe,* las empresas deben de tener una cuenta registrada con la compañía Stripe y una cuenta bancaria ligada con dicha cuenta.

Sin embargo, los particulares a la hora de pagar usando este método de pago no es necesario que tenga una cuneta de *Stripe*, sino basta con tener una tarjeta de crédito o débito.

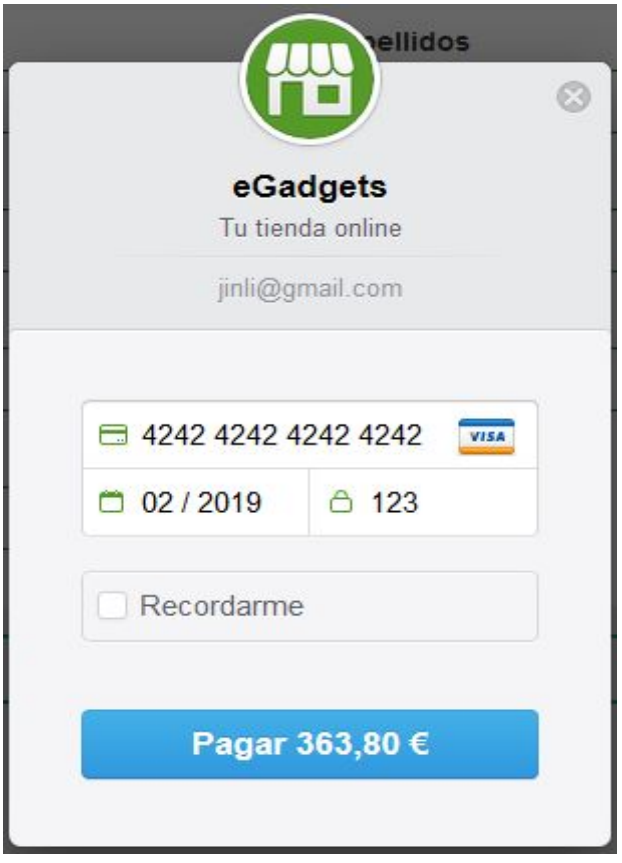


Ilustración 5-25 Vista de la ventana emergente de cobro

Una vez el particular haya pagado, la empresa puede gestionar el pago recibido a través de la página de *Stripe* usando la cuenta registrada previamente.

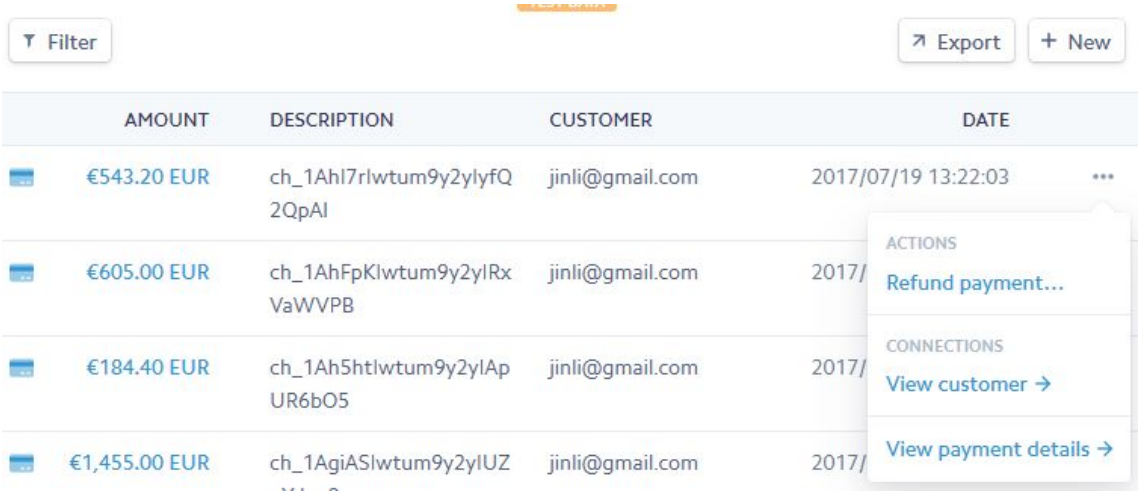


Ilustración 5-26 Vista de la interfaz del panel de control de Stripe

Para integrar el servicio de procesamiento de pago de *Stripe* en nuestro proyecto, se ha descargado la librería *PHP "stripe-php-5.1.1"* de Stripe e incluido en nuestra aplicación *web*.

Se ha creado el archivo de configuración *config.php* para vincular nuestra tienda virtual con la cuenta registrada de *Stripe*.

```php
<?php
require_once 'stripe-php-5.1.1/init.php';

$stripe = array(
  "secret_key"      => "**********************************",
  "publishable_key" => "pk_test_ZUYiCmK1ObyOehsdYXF08wEn"
);

\Stripe\Stripe::setApiKey($stripe['secret_key']);
?>
```

Ilustración 5-27 Fichero config.php

La clave privada y la clave pública se obtiene en la sección *API* de nuestra cuenta de *Stripe*.

En el código *HTML* de la página de confirmación de pago se ha incluido el siguiente *Script* para que salga la ventana emergente de cobro de Stripe cuando el usuario haga clic en el botón de "Pago seguro".

```
<script>
var handler = StripeCheckout.configure({
  key: 'pk_test_ZUYiCmK1ObyOehsdYXF08wEn',
  image: 'https://stripe.com/img/documentation/checkout/
      marketplace.png',
  locale: 'auto',
  token: function(token) {
        $("#stripeToken").val(token.id);
        $("#payForm").submit();
  }
});

document.getElementById('customButton').addEventListener('
    click', function(e) {
  // Open Checkout with further options:
  handler.open({
    name: 'eGadgets',
    description: 'Tu tienda online',
    zipCode: false,
    amount: <?php echo $total * 100 ?>,
    currency: 'eur',
    email: '<?php echo $row['user'] ?>'
  });
  e.preventDefault();
});

// Close Checkout on page navigation:
window.addEventListener('popstate', function() {
  handler.close();
});
</script>
```

Ilustración 5-28 Script: Activación de la ventana emergente de cobro

Se ha creado el archivo *charge.php* que funciona como el intermediario entre la ventana emergente de cobro y el servidor de Stripe. Es el encargado de enviar los datos bancarios y los datos del pedido del comprador al servidor Stripe para que posteriormente pueda proceder a cobrar el dinero correspondiente de la compra.

```php
<?php
  session_start();
  require_once 'includes/database.php';
  $db_server = mysqli_connect($db_hostname, $db_username, $
    db_password, $db_database);
  require_once 'config.php';

  $token = $_POST['stripeToken'];
  $total = $_POST['total'];
  $email = $_POST['customer'];

  $customer = \Stripe\Customer::create(array(
      'email' => $email,
      'source'  => $token
  ));

  $charge = \Stripe\Charge::create(array(
      'customer' => $customer->id,
      'amount'   => $total,
      'currency' => 'eur'
  ));

  unset($_SESSION['cart_items']);
  header('Location: afterPay.php');
?>
```

Ilustración 5-29 Fichero charge.php

5.2.2.7. Chat online

Se ha implementado un sistema de mensajería en línea integrando un como la sección de soporte técnico a nuestra tienda virtual.

Se trata de un servicio de terceros que se ha decidido a integrar en este proyecto. Se llama *tawk.to* es una plataforma que ofrece servicio de mensajería en línea para las aplicaciones *web*, para usar su servicio al igual que *Stripe* las empresas deben de tener una cuenta de *tawk.to*.

El proceso de integración del servicio *chat online* no es tan complicado como es con el *Stripe*, cuando la empresa se haya registrado y obtenido una cuenta de *tawk.to*, para integrarlo basta con copiar el código *script* del componente

gráfico del *chat* desde el panel de administración de la cuenta de *tawk.to* e incluirlo en el código *HTML* de las páginas de nuestro sitio *web*.

```
<!--Start of Tawk.to Script-->
<script type="text/javascript">
var Tawk_API=Tawk_API||{}, Tawk_LoadStart=new Date();
(function(){
var s1=document.createElement("script"),s0=document.get
  ElementsByTagName("script")[0];
s1.async=true;
s1.src='https://embed.tawk.to/**************/default';
s1.charset='UTF-8';
s1.setAttribute('crossorigin','*');
s0.parentNode.insertBefore(s1,s0);
})();
</script>
<!--End of Tawk.to Script-->
```

Ilustración 5-30 Script: Componente gráfico del chat

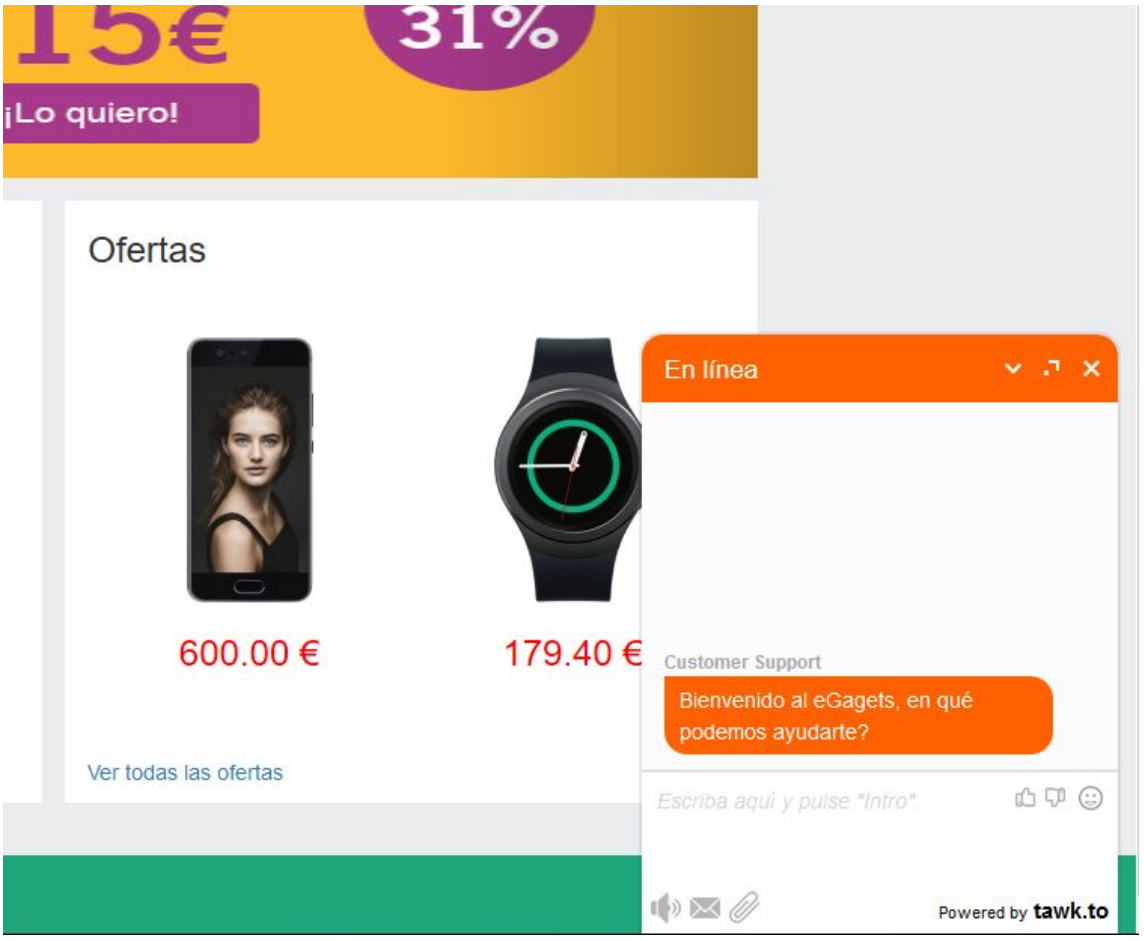


Ilustración 5-31 Vista del sistema chat online

A parte de la función de enviar mensajes a través de Internet, la plataforma también nos permite realizar análisis estadístico a los visitantes de nuestra aplicación *web*.

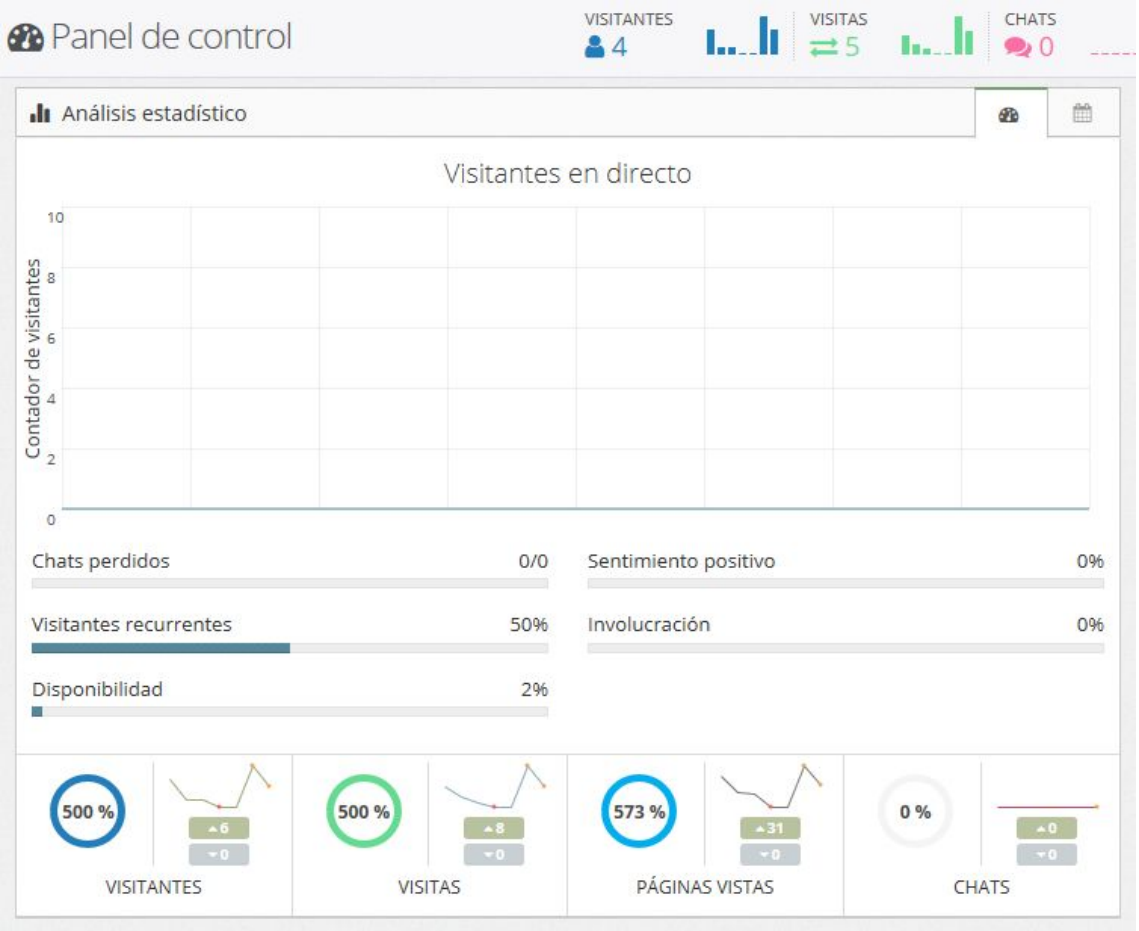


Ilustración 5-32 Vista del panel de control de la plataforma tawk.to

6. Conclusiones

Durante estos cuatro años de estudio he podido probar diferentes lenguajes de programación y metodologías de diseño e implementación. Me ha marcado mucho la experiencia de prácticas en empresas realizadas durante este último curso, mi trabajo consistía en desarrollo de páginas *web*, aunque la mayoría de los trabajos eran solamente del lado del cliente y se trabajaba con el *framework Bootstrap*, pero también en escasas veces he podido probar lenguajes de programación del lado del servidor como es el caso de *PHP,* haciendo páginas web de prueba con el sistema de gestión de contenido WordPress.

Tras un largo periodo de práctica, me he dado cuenta de que es cierto que a la hora de desarrollar un sitio *web* de carácter comercio electrónico es mucho más fácil y rápido si se emplea un *CMS* como WordPress, pero me he enamorado de la flexibilidad que ofrece el hecho de programar a mano sin usar temas pre-implementados.

La razón por la que he escogido *Bootstrap* como herramienta principal de desarrollo es que a diferencia de los *CMS*, un *framework* no tiene tantas restricciones.

Una posible futura mejora del sitio *web* desarrollado es añadir el sistema de comentarios para los diferentes productos, y un aumento de posicionamiento SEO de la aplicación *web*.

El resultado final del producto ha sido bueno, se han alcanzado los requisitos de la etapa de planificación y he podido aplicar los conocimientos adquiridos durante la carrera en el desarrollo de este proyecto.

7. Bibliografía

Asignatura Desarrollo centrado en el usuario. Tema 2 “Análisis de necesidades del usuario” Curso 2016 - 2017 Universidad Politécnica de Valencia

Asignatura Desarrollo centrado en el usuario. Tema 3 “Diseño con prototipos” Curso 2016 - 2017 Universidad Politécnica de Valencia

Cina Saffary. Validator, for Bootstrap 3. (2016). [En línea] http://1000hz.github.io/bootstrap-validator/ [Accedido: 15 de agosto 2017]

Mark otto & Jacobod Thornton. Bootstrap. [En línea] https://getbootstrap.com/docs/3.3/ [Accedido: 4 de agosto 2017]

Miguel Angel Alvarez. (2002). phpMyAdmin. [En línea] https://desarrolloweb.com/articulos/844.php [Accedido: 20 de agosto 2017]

Robin Nixon. (2009). Learning PHP, MySQL, and JavaScript. United States of America: O’Reilly.

W3C España. Guía breve de CSS. [En línea] http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo [Accedido: 22 de agosto 2017]

W3Schools. HTTP Methods GET vs POST. [En línea] https://www.w3schools.com/tags/ref_httpmethods.asp [Accedido: 22 de agosto 2017]

W3Schools. PHP 5 Sessions. [En línea] https://www.w3schools.com/php/php_sessions.asp [Accedido: 21 de agosto 2017]

WDN Web docs. JavaScript. [En línea] https://developer.mozilla.org/es/docs/Web/JavaScript [Accedido: 18 de agosto 2017]

Yusef Hassan & Francisco J. Martín Fernández & Ghzala Iazza. (2004). Diseño Web Centrado en el Usuario: Usabilidad y Arquitectura de la Información. [En línea] https://www.upf.edu/hipertextnet/numero-2/diseno_web.html [Accedido: 15 de agosto 2017]