El recomendador de smartphones

weizy

Consultor: Carlos Casado Martínez
Profesor: Ignasi Lorente Puchades
Fecha de entrega: 11/06/2018

Diego Ruiz Álvarez . Grado en Multimedia . TFG Ingeniería Web

FICHA DEL TRABAJO FINAL

- **Título del trabajo:** Weizy – Recomendador de Smartphones
- **Nombre del autor:** Diego Ruiz Álvarez
- **Nombre del consultor:** Ignasi Lorente Puchades
- **Nombre del PRA:** Carlos Casado Martínez
- **Fecha de entrega (mm/aaaa):** 06/2018
- **Titulación:** Grado en Multimedia
- **Área del Trabajo Final:** Ingeniería Web
- **Idioma del trabajo:** Castellano
- **Palabras clave:** recomendador, app, smartphones

Resumen del Trabajo:

Hoy en día, casi todos los seres humanos del planeta utilizan un smartphone. ¿Pero qué ocurre a la hora de adquirir uno nuevo? ¿Saben los usuarios qué están comprando realmente? Weizy es una aplicación que permite a los consumidores encontrar el teléfono móvil que más se ajusta a sus necesidades.

Primeramente, se ha recopilado información acerca de aplicaciones similares y su funcionamiento. Una vez decididas las funcionalidades, se ha empezado a trabajar en la fase de diseño de la interfaz (UX/UI). El siguiente paso ha sido el desarrollo de la aplicación. Para crear la susodicha aplicación, se ha utilizado una base de datos MySQL, código PHP 5, HTML5, CSS3 y JavaScript, además de otras API. Finalmente, en la fase de testeo se han comprobado los posibles errores, antes de la entrega final.

El resultado final: Una aplicación web **smartphones.weizyapp.com** que se compone de dos partes. Los usuarios visitaran este sitio, donde podrán interactuar con una interfaz que les irá guiando durante todo el proceso hasta mostrar los teléfonos ideales por pantalla, pudiendo aplicar ciertos filtros a los resultados finales. Desde el panel de control, el administrador podrá hacer las modificaciones oportunas para la correcta funcionalidad del sitio (a través de una interfaz gráfica).

Considero que ha sido una tarea compleja crear esta herramienta online, debido a la gran cantidad de funciones que se dispone (desplegando todos los conocimientos aprendidos a lo largo del grado y yendo más allá). Ha sido un reto completar todos estos procesos en el tiempo establecido.

Abstract:

Nowadays, almost all humans on the planet use a smartphone. But what happens when it comes the time to buy a new one? Do users know what they are really buying? Weizy is an application that allows consumers to find the mobile phone that best suits their needs.

First, information has been gathered about similar applications and their operation. Once the functionalities have been decided, work has begun on the design phase of the interface (UX / UI). The next step has been the development of the application. To create the application, we have used a MySQL database, PHP 5, HTML5, CSS3 and JavaScript code, in addition to other APIs. Finally, in the testing phase, possible errors have been checked before the final delivery.

The end result: A web application **smartphones.weizyapp.com** that is made up of two parts. The users will visit this site, where they will be able to interact with an interface that will guide them throughout the process until the ideal telephones are shown, being able to apply certain filters to the final results. In the control panel, the administrator can make the appropriate modifications for the correct functionality of the site (through a graphical interface).

I think it has been a complex task to create this online tool, due to the large number of functions available (displaying all the knowledge learned throughout the degree and even going further). It has been a challenge to complete all these processes in the established time.

ÍNDICE

1 . Introducción

1.1 Contexto y justificación del Trabajo

Actualmente, casi todos los seres humanos utilizamos un teléfono móvil a diario. Tanto para escribir mensajes, ver vídeos, entrar en redes sociales, trabajar, etc. Pero ¿Qué ocurre a la hora de comprar un nuevo Smartphone? ¿Saben los usuarios lo que realmente necesitan? ¿Son realmente capaces de interpretar las especificaciones técnicas de un producto?

Weizy ayuda a todas esas personas que no entienden de tecnología a encontrar el Smartphone que más se ajusta a sus necesidades.

Contestando a una serie de sencillas preguntas, la app se encarga de traducir todas esas incomprensibles especificaciones técnicas en palabras que la gente corriente pueda entender y mostrar el móvil que más se ajusta a las necesidades de un usuario específico.

Los potenciales clientes son todas aquellas personas que necesiten comprarse un nuevo móvil, y que no tienen por qué saber qué es una memoria RAM o un procesador. Y puedan hacerlo en cualquier momento y cualquier parte desde su Smartphone o el PC.

1.2 Objetivos del Trabajo

Debe ser una aplicación moderna, sencilla, eficiente, eficaz e inmediata.

Se trata de recopilar información de los mejores sitios web para recomendarle a los usuarios los mejores productos que mejor se ajusten a sus necesidades. Así, podemos sustituir al típico amigo al que todos recurrimos para que nos haga una recomendación.

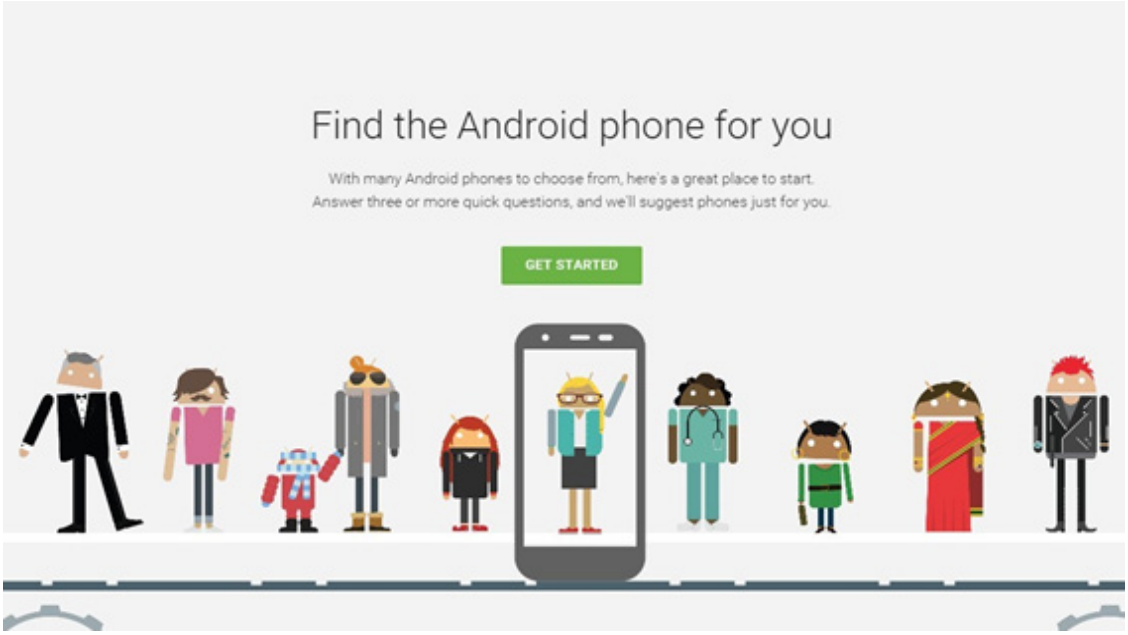
Por lo tanto, el usuario podrá: acceder a una aplicación web (a través de un dominio), seleccionar entre las diferentes secciones el uso que le va a dar a su dispositivo, visualizará categorías y subcategorías de opciones, deberá seleccionar al menos tres categorías, al final del proceso podrá visualizar por pantalla los Smartphone que más se ajustan a sus necesidades, podrá ver los diferentes precios y opciones de compra, podrá filtrar teniendo en cuenta diferentes criterios, finalmente también se guardará en la base de datos información sobre los gustos de los usuarios.

Objetivos de Weizy:

- **Traducir las especificaciones técnicas**
- **Facilitar una compra eficiente**
- **Objetividad**
- **Inmediatez**
- **Imparcialidad entre marcas y comercios**

1.3 Enfoque y método seguido

La idea es desarrollar un producto nuevo en el mercado europeo, tomando como referencia una versión creada por Google (Which Phone). Esta app ofrecía un servicio similar, aunque no ofertaba dispositivos IOS. Por otra parte, tampoco ofrecía los comercios disponibles. Además, solamente estaba disponible en el mercado de EEUU.



Entonces, por ejemplo, si un usuario desea comprar un Smartphone, la aplicación le preguntará acerca de:

¿Estás pensando en viajar con él?

¿También vas a usar la cámara para grabar vídeo?

Después de estos pasos, la aplicación muestra al usuario los móviles que mejor se adaptan a él y también las diferentes tiendas en línea donde podría comprarlos. Finalmente, el usuario puede comparar los diferentes precios, tiendas y condiciones. Podríamos obtener una comisión con cada venta individual.

A modo de conclusión, la idea es desarrollar una plataforma web que siga los siguientes pasos:

1. Sección de preguntas acerca del uso que se le va a dar al Smartphone.
2. Existirán subcategorías con preguntas adicionales.
3. Una vez acabado el proceso, se muestra al usuario los móviles por pantalla.
4. El usuario puede hacer filtrado por precio, categoría, tiendas, etc.
5. Se podrá recopilar información en la base de datos acerca de los gustos de los usuarios.

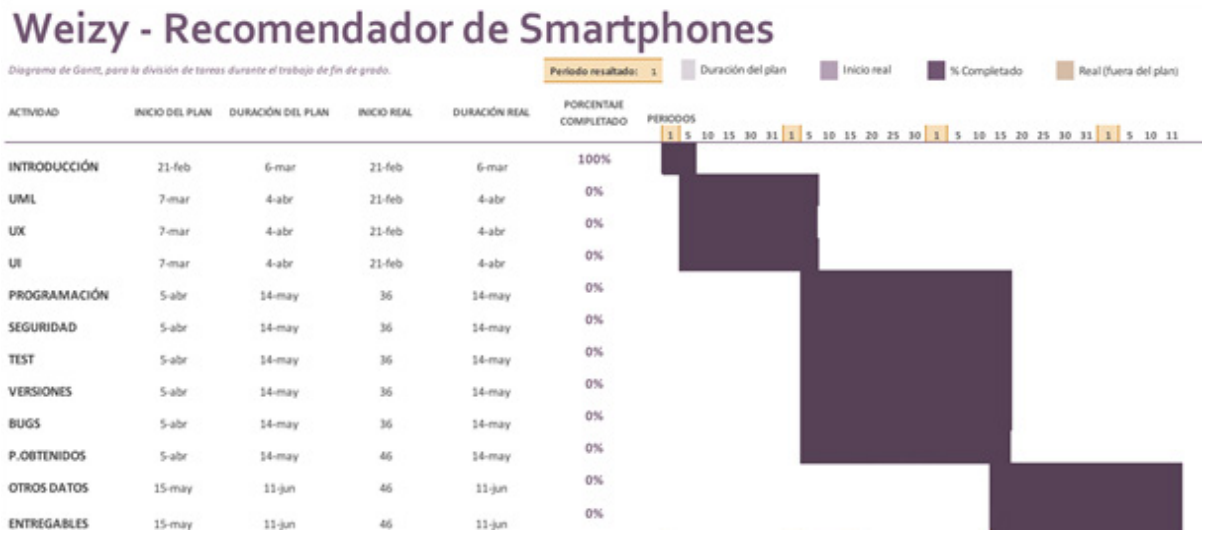
Considero que esta estrategia es la más apropiada, puesto que se ha comprobado que una empresa puntera en el sector de la tecnología ha desarrollado software similar. La idea, es mejorar la versión propuesta por Google y adaptarla al mercado español.

1.4 Planificación del Trabajo

Las fases a realizar podrían resumirse en 4 grandes grupos:

- **Introducción:** desarrollo de la idea
- **Diseño:** UX/UI y diseño de la interfaz
- **Desarrollo:** Completar la funcionalidad de la aplicación utilizando PHP y Javascript, etc.
- **Testeo:** Comprobación de posibles errores

A continuación en el diagrama de Gantt se amplían los tareas a realizar con más detalle y los tiempos establecidos para cada tarea.



1.5 Breve sumario de productos obtenidos

Tras todo el proceso, el producto obtenido es una aplicación web dividida en dos grandes grupos: El front office del site, y el panel de administración o back office. Se detallan todos los archivos de la aplicación.

Para el Front Office:

- **index.php >** Página principal de la app para usuarios.
- **pregunta.php >** Sección de subpreguntas al usuario.
- **final.php >** Página de resultados finales, se muestran las recomendaciones al usuario.

Para el Back Office:

- **admin.php >** Página inicial del panel de administración (página estática).
- **adminproducto.php >** Página para la configuración y borrado de productos existentes.
- **adminpuntuaciones.php >** Página para la puntuación y categorización de los productos.
- **adminsincronizacion.php >** Página para el sincronizado total, hora y frecuencia.

- **agregar.php >** Página para agregar un nuevo producto a la base de datos.
- **amazon.php >** Funciones para obtener los precios de la base de datos de Amazon.
- **bd.php >** Funciones clave para el correcto funcionamiento de todo el sistema.
- **composer.json >** Archivo composer para la utilización de la API de Amazon
- **composer.lock >** Archivo composer para la utilización de la API de Amazon
- **comprar.php >** Algoritmo para el conteo de las ventas realizadas.
- **config.php >** Conexión con la base de datos, y definición de la puntuación máxima.
- **fechapanelcontrol.php >** Obtenemos la fecha actual para el panel de control.
- **login.php >** Página de login al panel de control.
- **logout.php >** Página de logout del panel de control.
- **menu.php >** Guardamos aquí el menú principal para posteriormente hacer un include.
- **puntuarproducto.php >** Página para el redireccionamiento a adminpuntuaciones.
- **sincronizacion.php >** Sincronización manual de precios para cada producto.
- **sincronizacionprogramada.php >** Utilizamos el crontab para actualizar automáticamente.
- **sincronizar_producto.php >** Utilizamos este archivo para la sincronización de los precios.
- **usuarionav.php >** Guardamos la barra de navegación superior, con usuario y cerrar sesión.
- **utils.php >** Funciones necesarias para el correcto funcionamiento de la aplicación.
- **otros >** Archivos css, api Amazon, js, imágenes, y fuentes adicionales.

1.6 Breve descripción de los capítulos de la memoria

2. Diseño de la aplicación

2.1 Diagramas UML - No es necesario el uso de UML en este caso, no hay POO.
2.2 Diagramas de flujo - Representación de los diagramas de flujo.
2.3 Casos de uso - Representación de los casos de uso en formato tabla y diagrama.
2.4 Diagrama ER - Representación del diagrama Entidad - Relación.
2.5 Historias de Usuario - Representación de las historias de usuario.
2.6 Usabilidad/UX (DCU) - Muestra y explicación de cada interfaz.
2.7 Arquitectura - Datos técnicos sobre el proyecto.

3. Desarrollo - Despliegue del conjunto de pasos realizados en la fase de desarrollo.

3.1 Extractos de código - Explicación de los fragmentos de código por partes.
3.1.1 Zona Usuario (Front Office) - Código para el Front Office.
3.1.2 Panel de administración (Back Office) - Código para el Back Office.
3.1.3 Otros - Otros archivos con código relevante.

2. Diseño de la aplicación

2.1 Diagramas UML

PARA ESTE PROYECTO NO USAREMOS UN DIAGRAMA DE CLASES, PUESTO QUE NO USAMOS ORIENTACIÓN A OBJETOS PARA ACCEDER A LA BASE DE DATOS. EN SU LUGAR, EMPLEAREMOS SQL DIRECTAMENTE. DE ESTE MODO, HEMOS ELABORADO SOLAMENTE EL MODELO RELACIONAL.

Existirán dos tipos de usuario en esta aplicación:

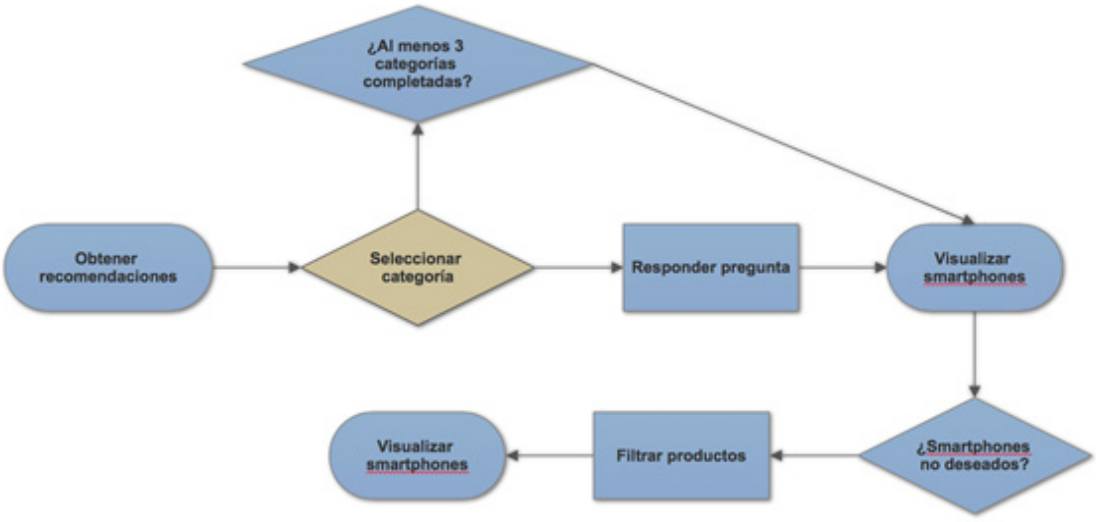
- **El usuario final**
- **El administrador de la aplicación**

Para ello, se desarrollarán dos interfaces diferentes: el front office (donde el usuario final podrá recibir las recomendaciones personalizadas de los Smartphone que más se ajustan a sus necesidades) y el back office (donde el administrado podrá hacer una seria de configuraciones que afectarán directamente a la interfaz principal de la aplicación: subir un nuevo producto, configurar la sincronización de precios, etc).

2.2 Diagramas de flujo

Usuarios:

El usuario deberá completar al menos tres categorías, respondiendo a sus correspondientes preguntas para poder visualizar los smartphones recomendados en la pantalla final. Si el usuario no está satisfecho con las recomendaciones, podrá utilizar el filtro concretando algunos datos más.

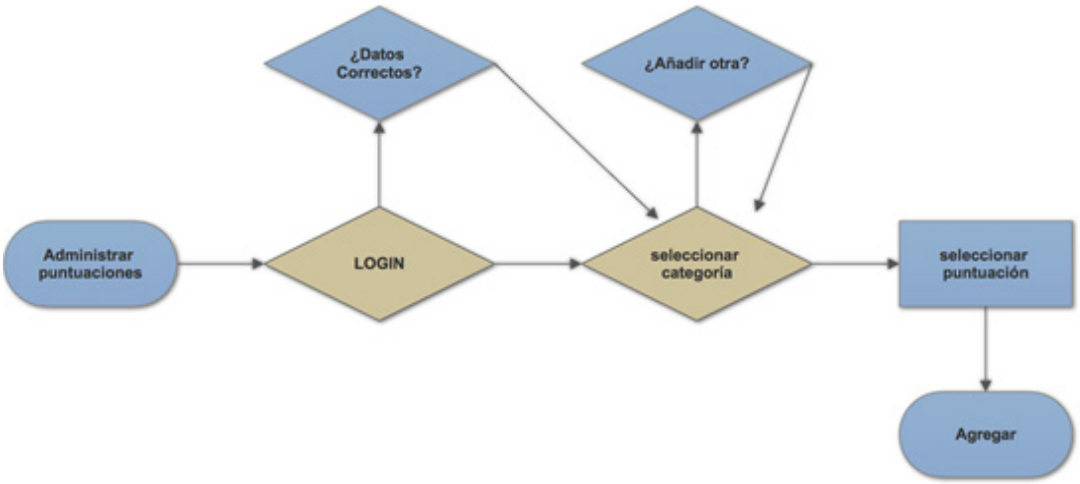


Administrador:

Subir nuevo producto: Para subir un nuevo producto, el administrador deberá loguearse con su usuario y su contraseña. Si los datos son correctos, podrá acceder al formulario para rellenar los datos requeridos. Una vez rellenados satisfactoriamente podrá insertar el nuevo producto en la base de datos.

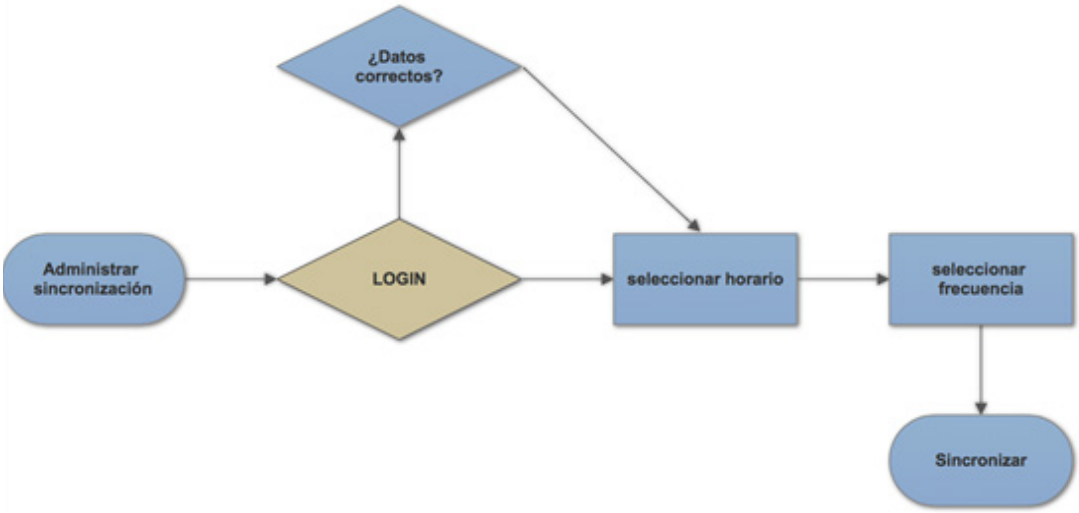
Administrar categorías y puntuaciones:

Para administrar categorías y puntuaciones, el administrador deberá loguearse con su usuario y su contraseña. Si los datos son correctos, podrá acceder al formulario para seleccionar tres categorías como máximo, con sus correspondientes puntuaciones. Una vez completados los datos, podrá agregar estas características.

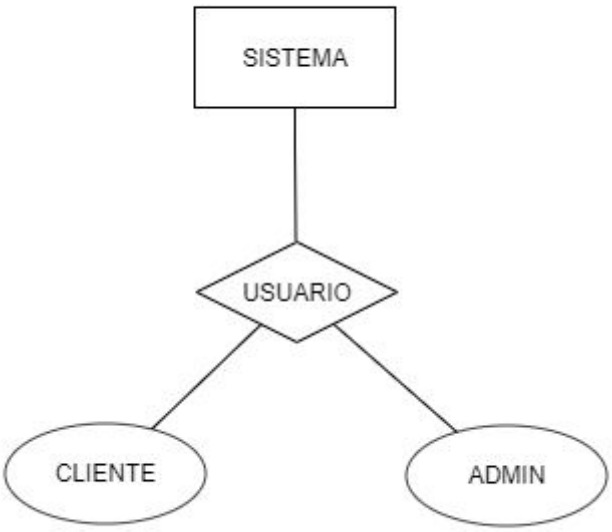


Administrar sincronización:

Para administrar la sincronización, el administrador deberá loguearse con su usuario y su contraseña. Si los datos son correctos, podrá acceder al formulario para seleccionar el momento de la sincronización y la frecuencia de repeticiones. Una vez completados los datos, los precios se podrán sincronizar en los tiempos establecidos.



Actores del Sistema



2.3 Casos de Uso

Caso de uso número 1: “ Contestar preguntas“

Resumen de la funcionalidad: El cliente contesta a las preguntas planteadas

Actores: Cliente

Casos de uso relacionados: Recibir recomendaciones

Pre-condición: El cliente debe completar al menos tres categorías principales con sus subsecciones.

Post-condición: Una vez ccompletado puede acceder a la página final de recomendaciones.

Alternativas de proceso y excepciones: El usuario también podrá realizar modificaciones en las categorías completadas, o completar alguna más.

Caso de uso número 2: “Recibir recomendaciones“

Resumen de la funcionalidad: El usuario final recibirá las recomendaciones por pantalla

Actores: Cliente

Casos de uso relacionados: Contestar preguntas

Pre-condición: El cliente recibe por pantalla las recomendaciones.

Post-condición: Se pulsará el botón comprar para finalizar el proceso.

Alternativas de proceso y excepciones: Se podrán aplicar ciertos filtros en el caso de que el usuario quiera ajustar más las recomendaciones originales.

Caso de uso número 3: “ Login y logout“

Resumen de la funcionalidad: El administrador se logeará para acceder al panel y podrá cerrar sesión.

Actores: Administrador

Casos de uso relacionados: (panel de control)

Pre-condición: El administrador completa los campos para acceder al panel

Post-condición: El administrador cierra sesión para salir del panel

Alternativas de proceso y excepciones:

Caso de uso número 4: “ Administrar categorías“

Resumen de la funcionalidad: El administrador podrá administrar las categorías

Actores: Administrador

Casos de uso relacionados: (panel de control)

Pre-condición: El administrador modificará o eliminará un producto

Post-condición: Se muestra por pantalla un mensaje de respuesta

Alternativas de proceso y excepciones: Se debe tener en cuenta que son dos funciones diferentes, el administrador podrá elegir entre ambas.

Caso de uso número 5: “ Agregar un producto“

Resumen de la funcionalidad: El administrador podrá agregar un nuevo producto

Actores: Administrador

Casos de uso relacionados: (panel de control)

Pre-condición: El administrador rellenará los campos a completar con la información

Post-condición: Se mostrará un mensaje por pantalla

Alternativas de proceso y excepciones: En caso de que el ASIN ya exista en la base de datos, no se podrá agregar el producto.

Caso de uso número 6: “ Administrar Sincronización“

Resumen de la funcionalidad: El administrador podrá dministrar la sincronización de los precios.

Actores: Administrador

Casos de uso relacionados: (panel de control)

Pre-condición: El administrador deberá seleccionar entre sincronización manual o automática.

Post-condición: Se mostrará un mensaje por pantalla, los precios se sincronizarán.

Alternativas de proceso y excepciones: Estas secciones no son complementarias, por lo que el administrador podrá realizar cualquiera de ellas en el momento que estime oportuno.

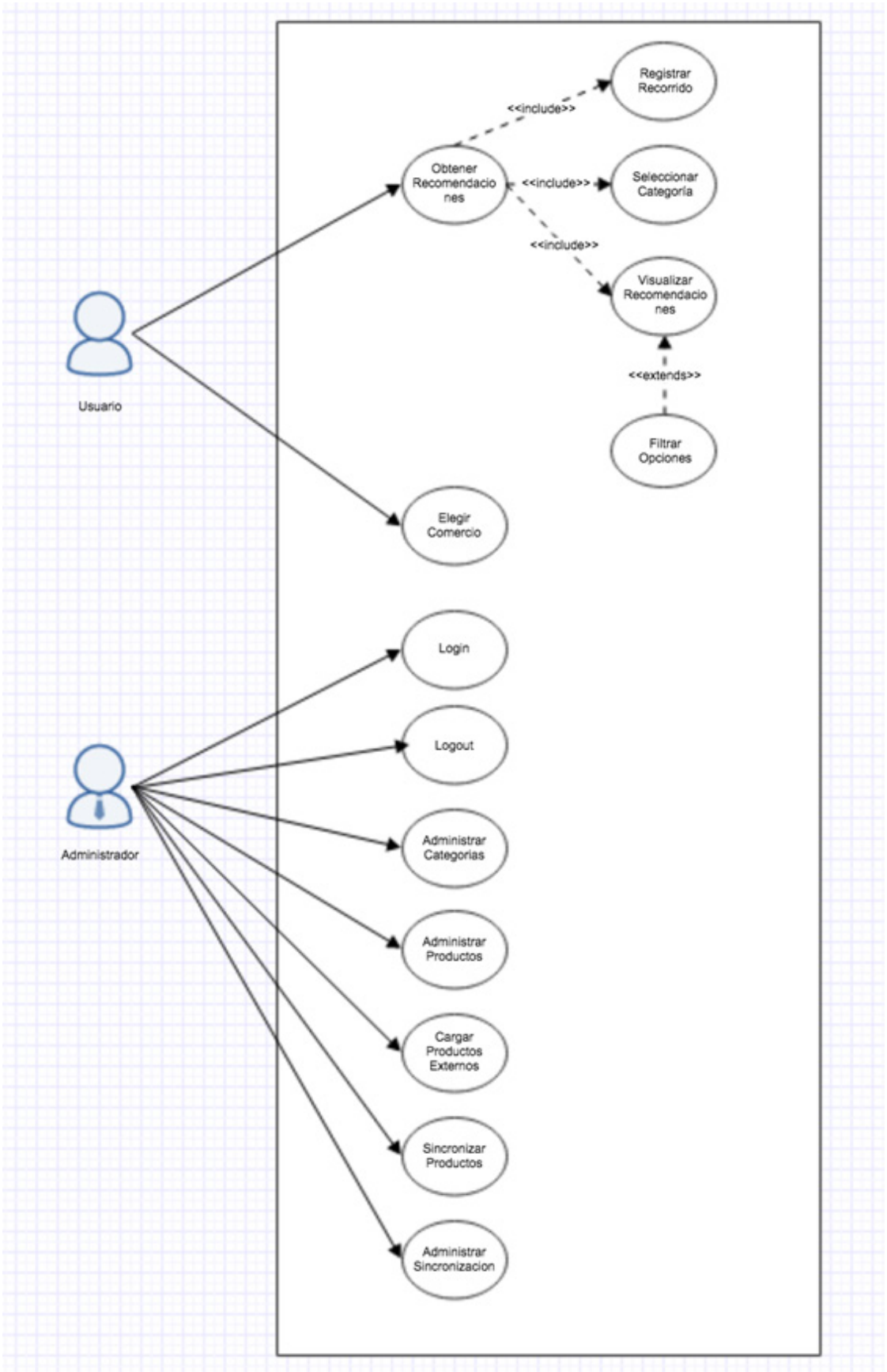
A continuación, se muestran los dos tipos de actores, y las funciones que podrán realizar cada uno de los usuarios dependiendo de su rol:

Usuario principal:

Se puede apreciar en el diagrama como este usuario puede realizar dos tareas principales (obtener recomendaciones de los Smartphone deseados y elegir el comercio que más se adecúe a sus necesidades.) Para poder obtener recomendaciones, el usuario deberá seguir una serie de pasos obligatoriamente: registrar recorrido de las opciones deseadas, seleccionando las categorías. Una vez realizado este proceso, el usuario final tendrá la opción de poder filtrar los productos mostrados. Más tarde detallaremos con más profundidad las posibilidades de cada pantalla.

Administrador de Weizy:

Por otra parte, el administrador podrá realizar muchas más tareas: Conectarse a un panel de administración utilizando un usuario y una contraseña (Login y Logout), subir un nuevo producto (se mostrará dentro de las recomendaciones finales al usuario final), sincronizar los precios con la base de datos de Amazon, administrar la configuración de la sincronización (horario de la sincronización, frecuencia, etc), categorizar cada producto (es decir, determinar cuándo se mostrará un Smartphone determinado por pantalla al recibir una recomendación), configurar un producto (hacer modificaciones o eliminarlo directamente). Más tarde detallaremos con más profundidad las posibilidades de cada pantalla.


Registrar Recorrido
<<include>>
Obtener Recomendaciones
<<include>>
Seleccionar Categoría
<<include>>
Visualizar Recomendaciones
<<extends>>
Filtrar Opciones
Usuario
Elegir Comercio
Login
Logout
Administrar Categorias
Administrar Productos
Cargar Productos Externos
Sincronizar Productos
Administrar Sincronizacion
Administrador

2.4 Diagrama ER

A continuación, se muestra el diagrama de entidad-relación que describe la estructura final de la base de datos de Weizy.

- **Por un lado, guardaremos un único usuario (el administrador).**

No necesitaremos guardar más usuarios, porque no hay la posibilidad de hacer un registro para los usuarios finales. Este administrador tendrá una id, con usuario y una contraseña.

- **Categorías:**

Se trata del elemento básico y principal de la aplicación. Esto guardará una id, un icono y un nombre para cada categoría que pueda elegir el usuario. Otros elementos necesarios irán asociados a esta tabla.

Para cada categoría, habrá preguntas y respuestas distintas.

Cada respuesta tendrá un valor diferente que, al final, servirá para determinar los smartphones más adecuados. Tanto las preguntas como las respuestas guardarán datos similares (el nombre, el orden, una id, etc.)

- **Puntuación:**

Utilizamos este espacio en la base de datos para categorizar cada Smartphone. Es decir, al final del proceso de selección de respuestas, la aplicación guardará una puntuación. Dependiendo de esa puntuación se mostrará un tipo de smartphone u otro.

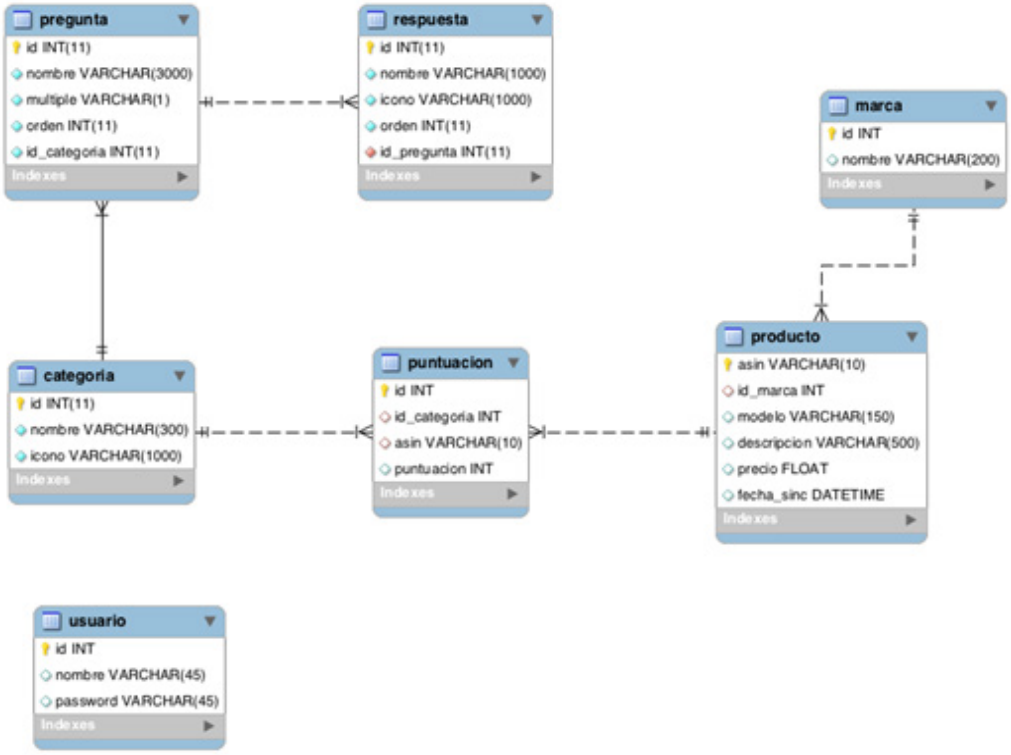
- **El producto:**

Aquí se guardarán todas aquellas características relacionadas con el smartphone, un ASIN (Amazon Standard Identification Number) que servirá de id, el modelo, descripción, etc.

- **La marca estará asociada al producto:**

Cada una tendrá una id y un nombre asociado. Más tarde podremos seleccionar la marca en un desplegable a la hora de subir un nuevo producto desde la interfaz de administración.

Más tarde detallaremos con más profundidad las posibilidades de cada pantalla, donde se podrán apreciar con más facilidad los elementos mostrados en este diagrama.



2.5 Historias de Usuario

Número: 1

Usuario: Usuario final, cliente
Nombre de la historia: Registrar recorrido
Prioridad en negocio: alta
Riesgos en desarrollo: medio
Programador responsable: Diego Ruiz
Descripción: Quiero seleccionar el uso que le voy a dar a mi nuevo smartphone: Necesito una buena cámara, voy a viajar y me importa mucho el diseño.
Validación: A través de las categorías y subcategorías, el usuario podrá detallar con preguntas y respuestas sencillas el uso que le va a dar a su smartphone.

Número: 2

Usuario: Usuario final, cliente
Nombre de la historia: Visualizar recomendaciones
Prioridad en negocio: alta
Riesgos en desarrollo: alto
Programador responsable: Diego Ruiz
Descripción: Quiero que me recomienden un teléfono según los usos que he tenido que describir en las categorías.
Validación: Se mostrará por pantalla los teléfonos que más se ajustan a las necesidades descritas del usuario.

Número: 3

Usuario: Usuario final, cliente
Nombre de la historia: Filtrar recomendaciones
Prioridad en negocio: medio
Riesgos en desarrollo: bajo
Programador responsable: Diego Ruiz
Descripción: Los smartphones mostrados me parecen demasiado caros, quiero poder encontrar los más baratos y que se ajusten a mis necesidades.
Validación: Una vez mostrados los smartphones por pantalla, el usuario podrá filtrar los resultados por precio, comercio, marca, etc.

Número: 4

Usuario: Administrador
Nombre de la historia: Acceder a la plataforma de administración
Prioridad en negocio: alta
Riesgos en desarrollo: alto
Programador responsable: Diego Ruiz
Descripción: Como administrador quiero poder ser el único que accede al panel de administración.
Validación: Mediante un formulario de acceso con dos campos (usuario y contraseña) el administrador podrá acceder a la interfaz de administración de forma segura.

Número: 5

Usuario: Administrador
Nombre de la historia: Subir un nuevo producto
Prioridad en negocio: media
Riesgos en desarrollo: medio
Programador responsable: Diego Ruiz
Descripción: Quiero poder ir subiendo los últimos smartphones que salen al mercado, para tener la base de datos actualizada.
Validación: El administrador podrá acceder a la sección “subir nuevo producto”. Aquí podrá subir un nuevo producto a través de un formulario donde tendrá que cubrir los datos requeridos.

Número: 6

Usuario: Administrador
Nombre de la historia: Categorizar y puntuar
Prioridad en negocio: media
Riesgos en desarrollo: medio
Programador responsable: Diego Ruiz
Descripción: Quiero poder ordenar los smartphones según las categorías que les correspondan, y también poder puntuarlos en función de las respuestas.
Validación: Existirá una sección especial solamente para este proceso, donde el administrador podrá seleccionar hasta tres categorías y dar la puntuación correspondiente de 1 a 10 para cada una de ellas.

Número: 7

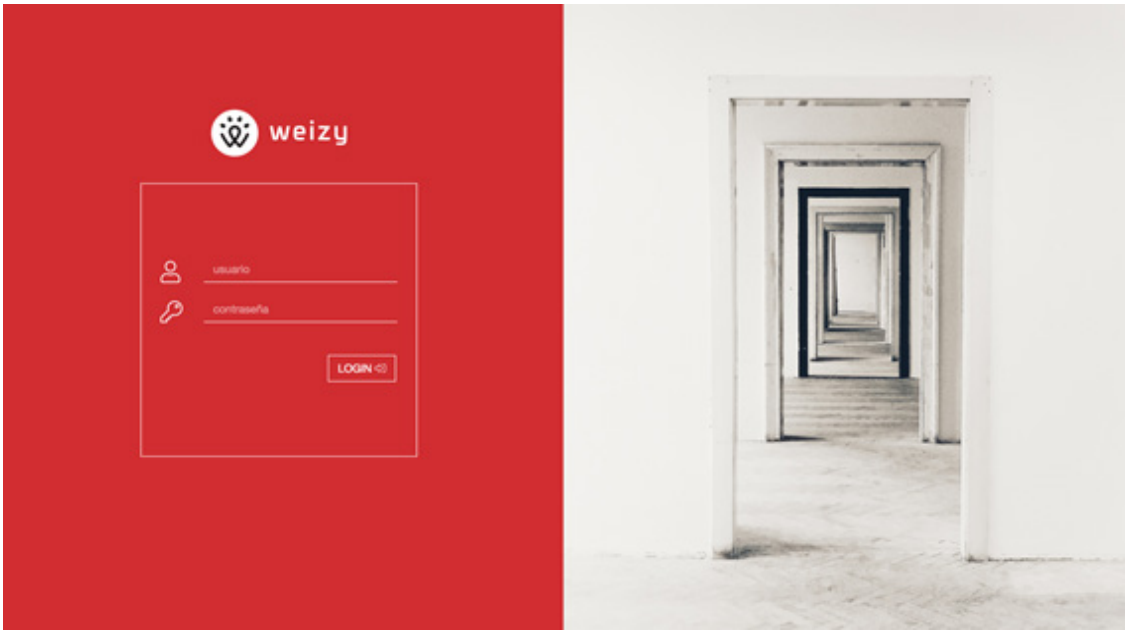
Usuario: Administrador
Nombre de la historia: Administrar sincronización
Prioridad en negocio: alta
Riesgos en desarrollo: bajo
Programador responsable: Diego Ruiz
Descripción: Quiero poder mostrar en el front office el precio de los productos sincronizado con la base de datos de Amazon.
Validación: Desde la sección administrar sincronización, el administrador podrá seleccionar el momento del día en que se sincronizan los precios, la frecuencia o también podrá hacer una sincronización manual.

2.6 Usabilidad/UX (DCU)

INTERFAZ DE ADMINISTRACIÓN

LOGIN:

El administrador podrá loguearse con su usuario y su contraseña, para poder acceder al panel de control.

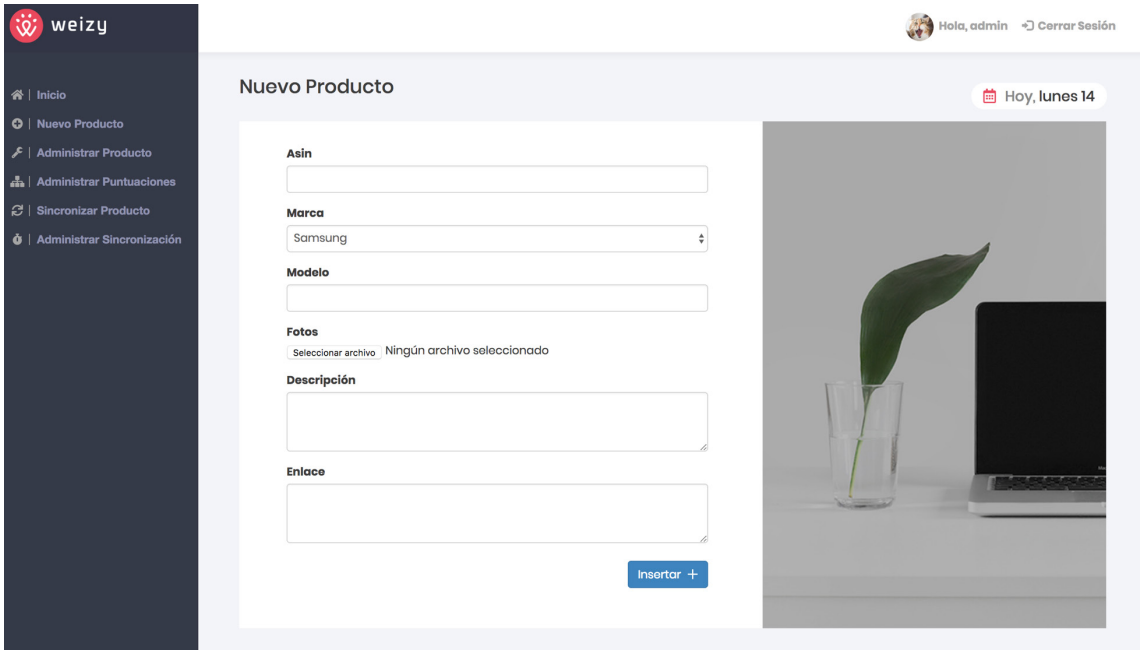


INICIO/HOME:

Esta es la página inicial, la que se muestra por defecto al loguearse en el panel. De momento, no tiene ninguna funcionalidad (simplemente estética). En el futuro podría hacerse que los gráficos representasen dicha información de forma dinámica.

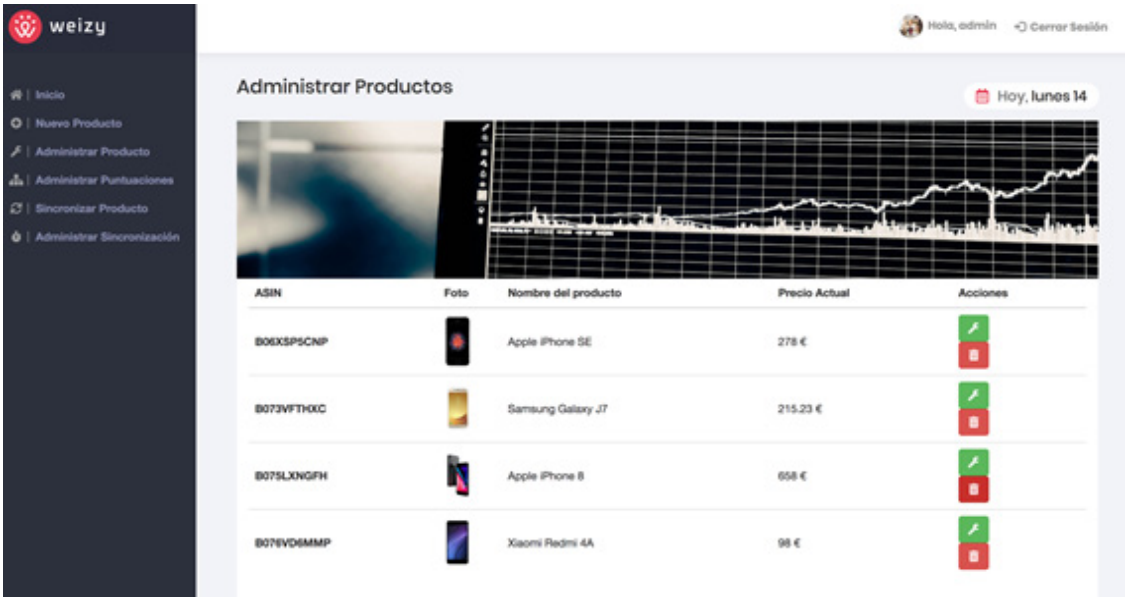
NUEVO PRODUCTO:

Esta es la sección para agregar un nuevo teléfono. El administrador deberá rellenar todos los datos requeridos e insertar. Si hay algún error se mostrará por pantalla.



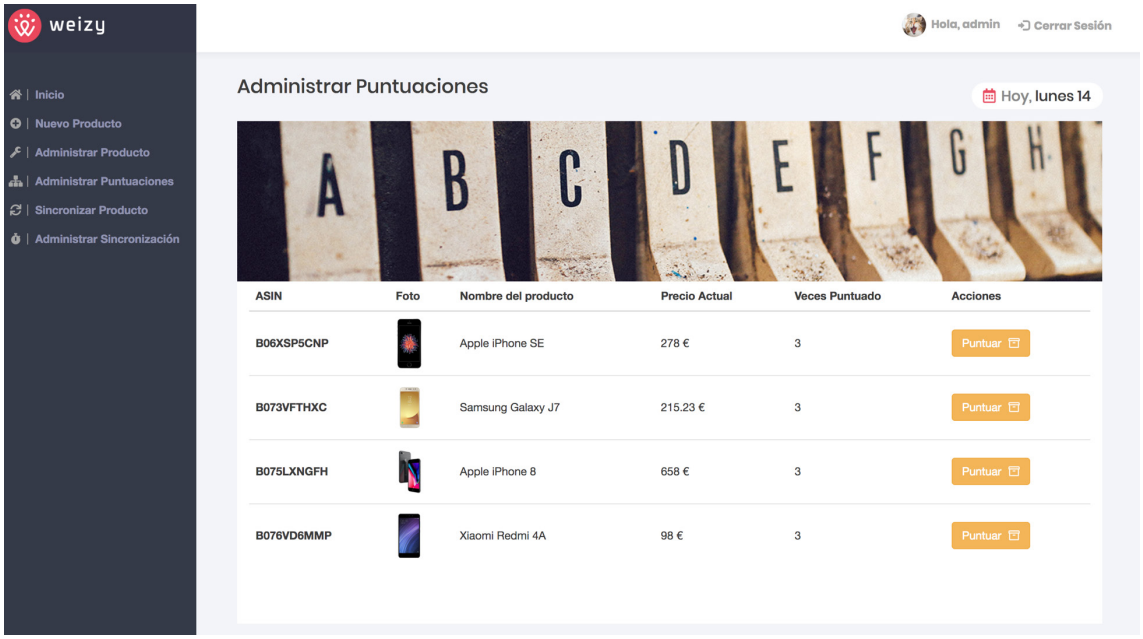
NUEVO PRODUCTO:

El administrador podrá actualizar los datos de cada producto (faltaría añadir esta funcionalidad en un futuro, puesto que ahora mismo solamente es posible utilizar el botón de eliminar) en la base de datos, y podrá eliminarlo directamente si lo desea.



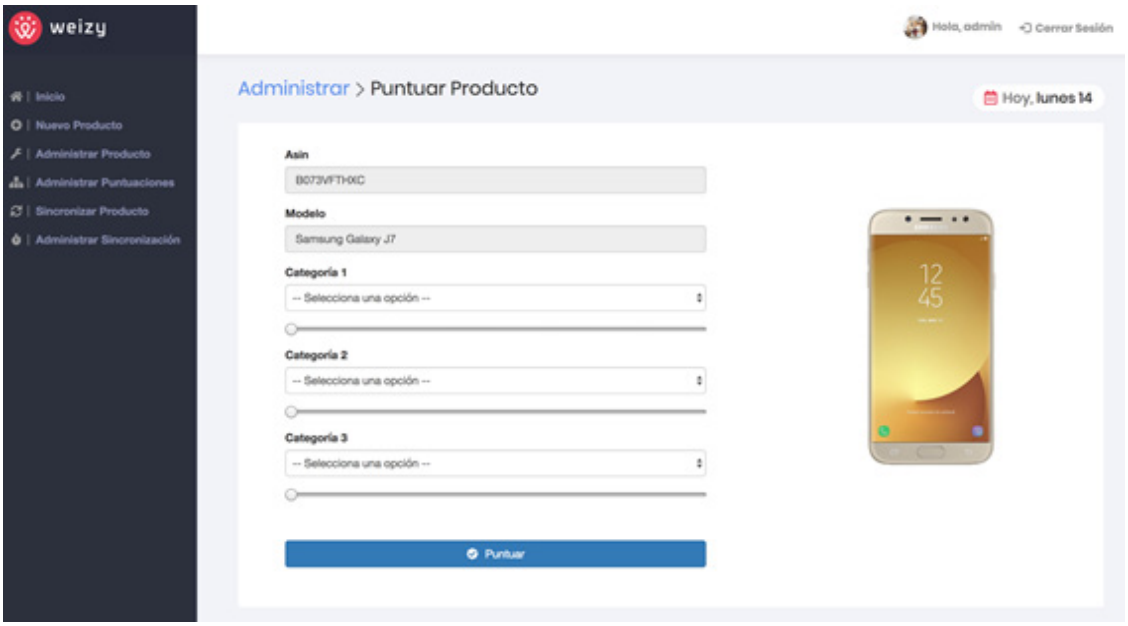
ADMINISTRAR PUNTUACIONES:

El administrador podrá puntuar y categorizar cada uno de los productos contenidos dentro de la base de datos.



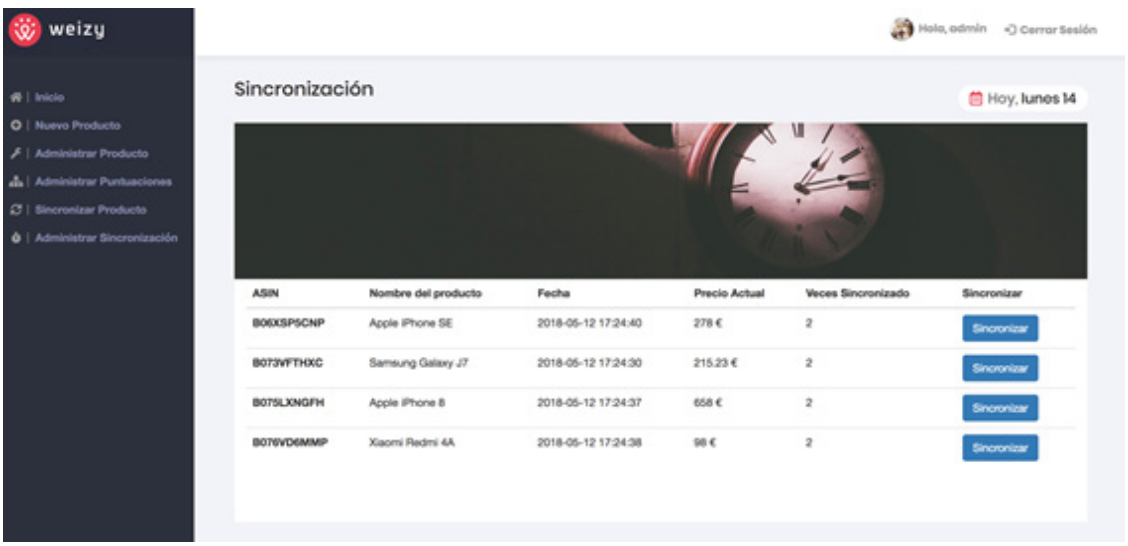
PUNTUAR:

Este apartado es crucial para el funcionamiento de la aplicación, puesto que esta configuración es la que permite que se recomiende un dispositivo u otro en las recomendaciones finales. Se muestra la fotografía del producto, su asin correspondiente y el modelo. El administrador puede seleccionar hasta cinco categorías (estas son las que ve el usuario en la página inical). Por otro lado, cada categoría contiene una puntuación (de 1 a 10). Estas puntuaciones hacen referencia a las subcategorías, e indican el grado de intensidad de cada una de ellas. Por ejemplo, asignaríamos a un smartphone la categoría “con buena cámara” porque es una de las 5 cosas que más destacan y una puntuación de 10, por ejemplo, para indicar que dentro de los móviles con buena cámara, este es de los mejores. Las subcategorías (front office) también están planteadas en función de este grado de intensidad, pero enfocadas de una forma distinta y más sencilla de entender para el usuario.



ADMINISTRAR SINCRONIZACIÓN:

El usuario podrá ver la fecha de la última sincronización, sincronizar manualmente, ver el precio actual y la cantidad de veces que se ha sincronizado cada producto. Tendrá el ASIN y el nombre como referencia.



INTERFAZ PARA EL USUARIO

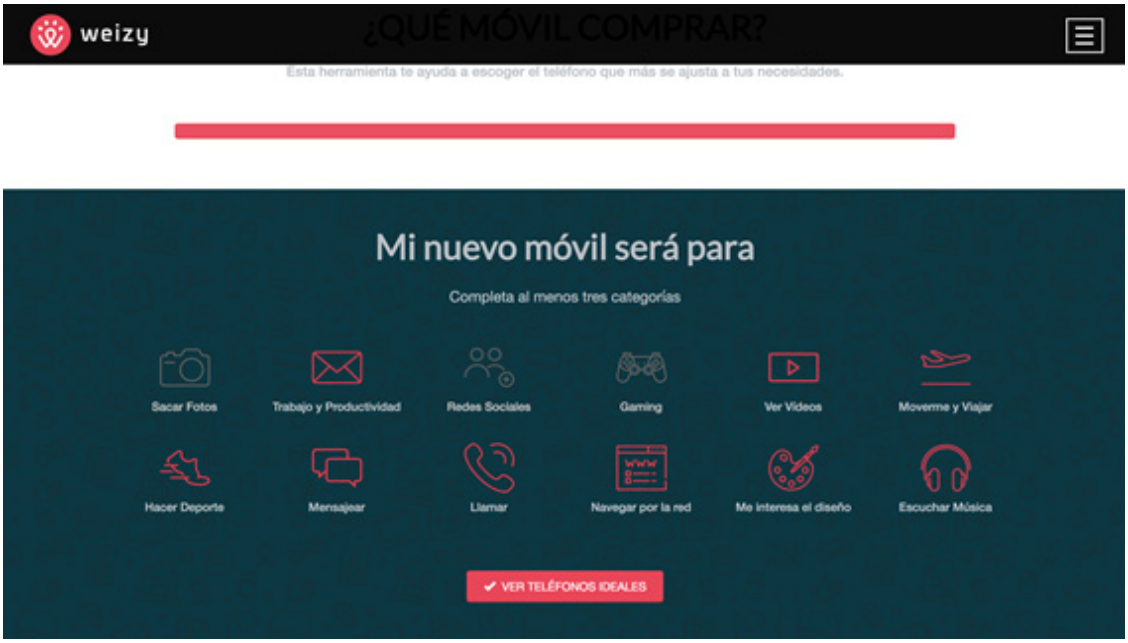
A continuación, se muestran las pantallas que verá el usuario final.

PÁGINA INICIO/HOME:

A continuación, se muestra la interfaz de categorías principal. En esta pantalla, el usuario podrá seleccionar el uso que piensa darle a su nuevo smartphone. Debe completar al menos tres categorías, con sus correspondientes subcategorías para que se muestre por pantalla el botón: “mostrar smartphones ideales”. Además, una barra de progreso (desarrollada con Javascript), irá mostrando en qué fase del proceso se encuentra el usuario en cada momento.

PÁGINA INICIO/HOME CON “VISITED”:

En esta pantalla se puede apreciar por los iconos en color gris, como el usuario ha completado tres categorías con sus correspondientes preguntas. Este era el requisito mínimo para poder finalizar el proceso y ver los teléfonos ideales. Podemos ver como la barra de progreso se muestra al 100%, indicando que se han completado el mínimo de pasos requeridos.

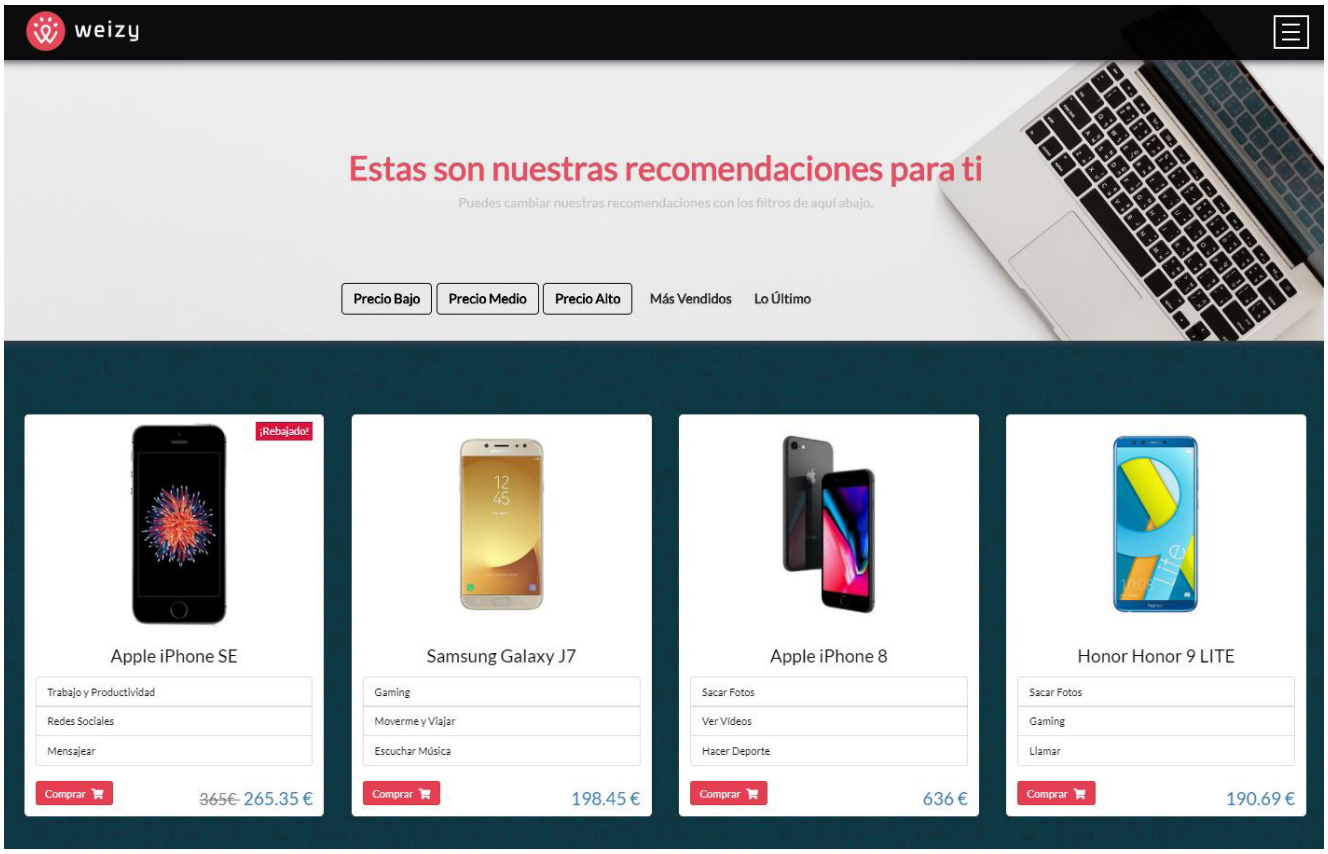


PÁGINA INICIO/HOME DENTRO DE UNA SUBCATEGORÍA:

En esta pantalla se muestra lo que vería un usuario al seleccionar una de las categorías principales. Esto sería una subcategoría, donde se muestra una de las preguntas con sus correspondientes respuestas. Cada icono, con su nombre hace referencia a una puntuación concreta. Esta puntuación servirá para determinar posteriormente qué smartphone se ajusta más a las necesidades del usuario. Por otra parte, este podrá volver sobre sus pasos en todo momento, pulsando el botón “volver al paso anterior”. Una categoría puede contener una o varias subcategorías, con sus correspondientes preguntas y respuestas.

PÁGINA FINAL CON LOS RESULTADOS:

Una vez completado todo el proceso, se muestra por pantalla el botón “ver smartphones ideales”. Esta pantalla mostraría los teléfonos que más se ajustan a sus necesidades. Habiendo completado las categorías necesarias, se toma la puntuación final recopilada durante todo el proceso. Cada smartphone está asociado a una serie de categorías y unas puntuaciones, al igual que las categorías. Esto hace que el algoritmo pueda mostrar los teléfonos más adecuados. El usuario podrá ir haciendo scroll para ver todas las posibilidades y comprobar los detalles de cada smartphone (el resto de información que se guarda en la base de datos acerca del producto). Por otro lado, también podrá filtrar los smartphones mostrados por precio, tipo de comercio, marca, etc.



2.7 Arquitectura

Información detallada sobre la arquitectura del proyecto:

- **Cliente:** HTML5, CSS y JAVASCRIPT
- **Servidor:** PHP 5
- **Bases de datos:** Linux, MySQL, phpMyAdmin
- **Frameworks:** Bootstrap 4, Fontawesome
- **Dominio:** **smartphones.weizyapp.com**
- **APIs:** API afiliación de Amazon

3 . DESARROLLO

3.1 Extractos de código

A ontinuación, se detallarán los fragmentos de código más relevantes para el correcto funcionamiento de la aplicación. Estos se dividen en tres grandes grupos, por zonas: La parte que verá el usuario final, el panel de administración, otros algoritmos que son complementarios.

3.1.1 Zona Usuario (Front Office)

INDEX.PHP:

En este documento PHP creamos la conexión con la base de datos y tomamos los datos utilizando SELECT, mediante SQL. De forma que se muestren las correspondientes categorías. Las categorías seleccionadas comienzan a guardarse en sesión.

```
// Recopilamos las respuestas del usuario, y registramos las categorías que
han sido completadas.



$conn = crearConexion();

if (isset($_GET['id_categoria'])) {


    $id_categoria = $_GET['id_categoria'];
    $_SESSION['finalizadas'][$id_categoria] = true;

    if(isset($_GET['orden']) && isset($_GET['id_resp']) && isset($_GET['orden_
resp']) && isset($_GET['orden_pregunta'])){
        $orden = $_GET['orden'];
        $id_resp = $_GET['id_resp'];
        $orden_resp = $_GET['orden_resp'];
        $orden_pregunta = $_GET['orden_pregunta'];

       $sql = "SELECT id, nombre, multiple, orden FROM pregunta WHERE id_cate-
goria=" . $id_categoria . " AND orden=" . $orden;
        $result = $conn->query($sql);
        if ($result->num_rows > 0) {

            $pregunta = $result->fetch_assoc();
        }
```

```
        $total_preguntas = 0;
        $sql= “SELECT COUNT(*) as total FROM `pregunta` WHERE id_categoria=” .
$id_categoria ;
        $result = $conn->query($sql);
        if ($result->num_rows > 0) {
            // output data of each row
            $res = $result->fetch_assoc();
            $total_preguntas = $res[“total”];
        }

        $total_respuestas = 0;
        $sql= “SELECT COUNT(*) as total FROM `respuesta` WHERE id_pregunta=” .
$pregunta[“id”] ;
        $result = $conn->query($sql);
        if ($result->num_rows > 0) {
            // output data of each row
            $res = $result->fetch_assoc();
            $total_respuestas = $res[“total”];
        }

        if (!isset($_SESSION[‘respuestas’][$id_categoria])) {
             $_SESSION[‘respuestas’][$id_categoria] = array();
        }
        $_SESSION[‘respuestas’][$id_categoria][$orden] = $id_resp;
```

```
        // Calcular la puntuación de la respuesta elegida por el usuario

        if (!isset($_SESSION[‘puntuaciones’][$id_categoria])) {
             $_SESSION[‘puntuaciones’][$id_categoria] = array();
        }


        $puntos_pregunta = ((float)PUNTUACION_MAX / $total_preguntas);
       $puntos_respuesta = ((float)$puntos_pregunta * $orden_resp / $total_res-
puestas);
```

```
       // Guardamos los puntos para cada respuesta

      $_SESSION[‘puntuaciones’][$id_categoria][$orden_pregunta] = $puntos_res-
puesta;

    }
}
```

```
// Mostrar las categorías por pantalla, tomando los datos con una SELECT e in-
yectando PHP en las etiquetas html, después cerramos la conexión. Se calcula la
clase que se debe aplicar para cada categoría en función de si ha sido comple-
tada.


$RUTA_IMG = “img/”;


$sql = “SELECT id, nombre, icono FROM categoria”;
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {

    while($categoria = mysqli_fetch_assoc($result)) {

        $completado = false;

        if (finalizada($categoria[“id”])){

            $completado = true;

        }

<div class=”col-md-2 elemento animated fadeInUp” <?= $completado ? “style=’fil-
ter:grayscale(100%);’” : “” ?>>

   <a href=”pregunta.php?id_categoria=<?=$categoria[“id”]?>&orden=1”>
        <img alt=”” class=”center-block hvr-float” src=”<?= $RUTA_IMG . $catego-
ria[‘icono’]?>” />
        <h4 class=”text-center”><?= $categoria[“nombre”]?></h4>
   </a>

</div>


<?php


    }
}

$conn->close();
```

```
// Para poder mostrar el botón “ver teléfonos ideales” por pantalla, se deben
completar al menos tres categorías


        <?php

            if(total_finalizadas() >= 3){

        ?>

        <div class=”col-md-12 text-center”>
             <a href=”final.php”>
             <button type=”submit” class=”btn miboton btn-default”>
               <span class=”glyphicon glyphicon-ok”></span>
                 Ver teléfonos ideales
             </button>
            </a>
        </div>

        <?php

            }

        ?>
```

PREGUNTA.PHP:

Este algoritmo representa a la subsección que muestra las preguntas por pantalla. A su vez, cada pregunta está relacionada con varias respuestas (con un valor determinado). Tanto las preguntas como las respuestas se toman de la base de datos. Las respuestas se guardan en la sesión hasta el proceso final.

```
// Hacemos cálculos para la barra de progreso y guardamos la respuesta para la
pregunta anterior.


$RUTA_IMG = “img/”;

if (isset($_GET[‘id_categoria’]) && isset($_GET[‘orden’])) {

    $id_categoria = $_GET[‘id_categoria’];
    $orden = $_GET[‘orden’];
```

```
   $sql = “SELECT id, nombre, multiple, orden FROM pregunta WHERE id_categoria=”
. $id_categoria . “ AND orden=” . $orden;
    $result = $conn->query($sql);
    if ($result->num_rows > 0) {

        $pregunta = $result->fetch_assoc();
    }

    $total_preguntas = 0;
    $sql= “SELECT COUNT(*) as total FROM `pregunta` WHERE id_categoria=” . $id_
categoria ;
    $result = $conn->query($sql);
    if ($result->num_rows > 0) {

        $res = $result->fetch_assoc();
        $total_preguntas = $res[“total”];
    }

    $total_respuestas = 0;
    $sql= “SELECT COUNT(*) as total FROM `respuesta` WHERE id_pregunta=” . $pre-
gunta[“id”] ;
    $result = $conn->query($sql);
    if ($result->num_rows > 0) {

        $res = $result->fetch_assoc();
        $total_respuestas = $res[“total”];
    }


   if(isset($_GET[‘id_resp’]) && isset($_GET[‘orden_resp’]) && isset($_GET[‘or-
den_pregunta’])){

        $id_resp = $_GET[‘id_resp’];
        $orden_resp = $_GET[‘orden_resp’];
        $orden_pregunta = $_GET[‘orden_pregunta’];

        if (!isset($_SESSION[‘respuestas’][$id_categoria])) {
            $_SESSION[‘respuestas’][$id_categoria] = array();
        }
        $_SESSION[‘respuestas’][$id_categoria][$orden] = $id_resp;
```

```
      $_SESSION['puntuaciones'][$id_categoria][$orden_pregunta] = $puntos_res-
puesta;

    $maximo = 0;
    $sql= "SELECT IFNULL(MAX(orden),0) as orden FROM `pregunta` WHERE id_cate-
goria=" . $id_categoria ;
    $result = $conn->query($sql);
    if ($result->num_rows > 0) {

        $res = $result->fetch_assoc();
        $maximo = $res["orden"];
    }

     // Muestra la pregunta con sus posibles respuestas.

    if (isset($pregunta)) {
        // Para la pregunta
    ?>
        <div class="col-md-12">
              <h2 class="text-center pregunta minuevosera animated fadeInUp"><?=
$pregunta["nombre"] ?></h2>
            <?php
                if ($pregunta["multiple"] == "S"){
            ?>

                <h3 class="text-center masdeuna">Elige una opción</h3>

            <?php
                }
            ?>

               <div class="row row-centered">
    <?php

           $sql = "SELECT id, nombre, icono, orden FROM respuesta WHERE id_pre-
gunta=" . $pregunta["id"] ." order by orden";
            $result = $conn->query($sql);

            if ($result->num_rows > 0) {

                $url = "";

                if($pregunta["orden"] < $maximo){

                    $url = "pregunta.php?id_categoria=" . $id_categoria . "&or-
den=" . ($pregunta["orden"]+1) . "&orden_pregunta=" . $pregunta["orden"];

                } else {

                    $url = "index.php?id_categoria=" . $id_categoria . "&orden="
. ($pregunta["orden"]) . "&orden_pregunta=" . $pregunta["orden"];
                }
```

```
    while($respuesta = $result->fetch_assoc()) {

    ?>

                <div class="col-md-2 elemento respuestaElemento col-centered
animated fadeInUp">
                 <a href="<?=$url?>&id_resp=<?=$respuesta['id']?>&orden_res-
p=<?=$respuesta['orden']?>">
                        <img alt="" class="center-block hvr-float" src="<?=
$RUTA_IMG . $respuesta['icono']?>" />
                        <h4 class="text-center"><?= $respuesta["nombre"]?></
h4></a>
                </div>
```

FINAL.PHP:

Este es el algoritmo que mediante las puntuaciones recopiladas, muestra por pantalla los teléfonos que coinciden con dichas puntuaciones.

```
// Contando las puntuaciones recopiladas y si se ha seleccionado algún filtro,
mostramos los smartphone que se ajustan a estas condiciones.

            $conn = crearConexion();

            $sql = "SELECT DISTINCT p.asin, p.modelo, p.descripcion, p.precio,
p.precio_anterior, p.fecha_sinc, p.foto, p.enlace, m.nombre as nombre_marca FROM
producto p, marca m, puntuacion pu WHERE p.id_marca = m.id AND p.asin = pu.asin";

            $where = " AND ( ";
            $i = 1;

            foreach (calcularPuntuacion() as $id_categoria => $puntuacion) {

                if ($i != 1) {

                    $where .= " OR ";
                }

                $where .= " (pu.id_categoria = " . $id_categoria . " AND pu.pun-
tuacion >= " . $puntuacion . ") ";

                $i++;

            }
```

```
$where .= " ) ";

            if ($preciobajo){

                $where .= " AND p.precio <= 200 ";
            }

            if ($preciomedio){

                $where .= " AND p.precio > 200 AND p.precio <= 500 ";
            }

            if ($precioalto){

                $where .= " AND p.precio > 500 ";
            }


            $sql .= $where;

            if ($loultimo){

                $sql .= " ORDER BY fecha_alta DESC";
            }

            if ($masvendidos){

                $sql .= " ORDER BY p.ventas DESC";
            }

            $result = mysqli_query($conn, $sql);

            if (mysqli_num_rows($result) > 0) {
                // output data of each row
                while($producto = mysqli_fetch_assoc($result)) {

            ?>
```

3.1.2 Panel de Administración (Back Office)

LOGIN.PHP:

Este algoritmo comprueba que el usuario y contraseña del administrador son los que figuran en la base de datos. En caso afirmativo inicia la sesión que permite acceder al panel de control Además, se guarda en sesión los datos relacionados con el administrador (foto, nombre).

```
// Hacemos el login utilizando el usuario y la contraseña definida en la base de
datos. Con el algoritmo SHA2 de 256 bits guardamos la contraseña cifrada en la
base de datos.


$error = false;

if (isset($_POST[“usuario”]) && isset($_POST[“password”])) {
     $usuario = $_POST[“usuario”];
     $password = $_POST[“password”];

    $sql= “SELECT usuario, foto  FROM `usuario` WHERE usuario=’” . $usuario . “’
and pwd=SHA2(‘” . $password . “’, 256)” ;
    $result = $conn->query($sql);
    if ($result->num_rows == 1) {

        $usuario = $result->fetch_assoc();
          // Guardamos el usuario en sesión
        $_SESSION[‘usuario’] = $usuario[“usuario”];
          // Guardamos la foto en sesión
        $_SESSION[‘foto’] = $usuario[“foto”];

          header(“location: admin.php”);

    } else{

          $error = true;

     }
```

LOGOUT.PHP:

Este algoritmo destruye la sesión para poder salir de forma segura del panel de control.

```
<?php

// Start the session
session_start();

session_destroy();

header(“location: login.php”);

?>
```

AGREGAR.PHP:

Este formulario recopila y valida que todos los datos estén completos. El elemento “marca” se selecciona de una lista desplegable, tomando los diferentes nombres de la base de datos. Si todo está correcto se hace un INSERT en las tablas correspondientes. En caso contrario, muestra un mensaje de error por pantalla.

```
//  Insertamos en la base de datos los campos rellenados de un producto, sincro-
nizando el precio con Amazon. Por otra parte, mostramos el formulario de inser-
ción de un nuevo producto.

   if(isset($_POST[“asin”]) && isset($_POST[“id_marca”]) && isset($_POST[“mode-
lo”]) && isset($_FILES[“foto”]) && isset($_POST[“descripcion”]) && isset($_POS-
T[“enlace”])){

        $target_dir = “productos/”;
        $target_file = $target_dir . basename($_FILES[“foto”][“name”]);
        $mensaje = “”;
        $uploadOk = 1;
        $imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));

        if(isset($_POST[“submit”])) {
            $check = getimagesize($_FILES[“foto”][“tmp_name”]);
            if($check !== false) {
                $uploadOk = 1;
            } else {
                $mensaje .= “La foto no es una imagen valida.<br>”;
                $uploadOk = 0;
            }
        }

        if (file_exists($target_file)) {
            $mensaje .= “La foto ya existe.<br>”;
            $uploadOk = 0;
        }

        if ($_FILES[“foto”][“size”] > 5000000) {
            $mensaje .= “La foto es demasiado grande.<br>”;
            $uploadOk = 0;
        }

        if($imageFileType != “jpg” && $imageFileType != “png” && $imageFileType
!= “jpeg”
        && $imageFileType != “gif” ) {
            $mensaje .= “La imagen no es: JPG, JPEG, PNG & GIF.<br>”;

  $uploadOk = 0;
        }

        if ($uploadOk == 0) {
            $mensaje .= “La imagen no se ha subido.<br>”;

        } else {
            if (move_uploaded_file($_FILES[“foto”][“tmp_name”], $target_file)) {
                $uploadOk = 1;
            } else {
                $mensaje .= “La imagen no se ha subido.<br>”;
                $uploadOk = 0;
            }        }
```

```
if ($uploadOk == 1){

            $precio = obtenerAmazonPrecio($_POST[“asin”]);

            if (is_null($precio)){

                $mensaje .= “No hay precio disponible. <br>”;
            }

           $sql = “INSERT INTO producto (asin, id_marca, modelo, foto, descrip-
cion, enlace, precio)
            VALUES (‘” . $_POST[“asin”] . “’, “ . $_POST[“id_marca”] . “, ‘” .
$_POST[“modelo”] . “’, ‘” . $target_file . “’, ‘” . $_POST[“descripcion”] . “’,
‘” . $_POST[“enlace”].”’, “ . $precio .” )”;

            if ($conn->query($sql) === TRUE) {
                $mensaje .= “Producto agregado correctamente.<br>”;
            } else {

                $mensaje .= “Error al crear un nuevo producto.<br>”;
                $error = true;
            }

        } else {

            $mensaje .= “Error al crear un nuevo producto.<br>”;
            $error = true;
        }

    }

    $sql = “SELECT id, nombre FROM marca”;
    $marcas = mysqli_query($conn, $sql);

<form method=”post” class=”agregar” enctype=”multipart/form-data”>

          <div class=”form-group”>
            <label for=”asin”>Asin</label>
             <input type=”text” class=”form-control” id=”asin” name=”asin” pla-
ceholder=”asin”>
          </div>
          <div class=”form-group”>
            <label for=”id_marca”>Marca</label>
            <select class=”form-control” id=”id_marca” name=”id_marca”>
    <?php

        while($marca = mysqli_fetch_assoc($marcas)) {

    ?>

                    <option value=”<?= $marca[‘id’]?>”><?= $marca[‘nombre’]?></
option>
```

ADMINPRODUCTO.PHP:

Utilizando DELETE FROM producto WHERE, se elimina el producto de la base de datos. Si es posible, para la entrega final intentaremos crear la posibilidad de hacer un UPDATE de los elementos guardados en la tabla producto.

```
// Eliminamos el producto de la base de datos, relacionado con su ASIN corres-
pondiente.

if(isset($_POST[“asin”]) && isset($_POST[“accion”])){

    $asin = $_POST[“asin”];
    $accion = $_POST[“accion”];

    if ($accion == “eliminar”) {

        $sql = “DELETE FROM producto WHERE asin = ‘” . $asin . “’”;

        if ($conn->query($sql) === TRUE) {

            $mensaje = “El producto con ASIN “ . $asin . “ se ha eliminado co-
rrectamente.”;

        } else {

           $mensaje = “El producto con ASIN “ . $asin . “ no se ha eliminado.”;

        }

$sql = “SELECT p.asin, p.modelo, p.descripcion, p.precio, p.fecha_sinc, p.foto,
p.enlace, p.veces_sincronizado, m.nombre as nombre_marca FROM producto p, marca
m WHERE p.id_marca = m.id”;
    $result = mysqli_query($conn, $sql);

    if (mysqli_num_rows($result) > 0) {

        while($producto = mysqli_fetch_assoc($result)) {

    ?>

<input type=”hidden” name=”asin” value=”<?=$producto[“asin”]?>” />
                        <input type=”hidden” name=”accion” value=”eliminar” />
                         <button class=”btn btn-primary btn-danger” type=”sub-
mit”>
                          <i class=”fas fa-trash-alt”></i>
```

ADMINPUNTUACIONES.PHP:

Este algoritmo muestra una tabla con varios datos acerca de un producto, haciendo una SELECT para mostrarlos (ASIN, foto, nombre, etc). Además, muestra el precio sincronizado y el número de veces que se ha categorizado cada teléfono.

```
// Mostramos por pantalla una tabla con los detalles de los productos y el número de veces que han sido puntuados.

 $sql = “SELECT p.asin, p.modelo, p.descripcion, p.precio, p.fecha_sinc, p.foto, p.enlace, p.veces_sincronizado, m.nombre as nombre_marca, (SELECT COUNT(*) FROM `puntuacion` WHERE asin = p.asin) AS veces_puntuado FROM producto p, marca m WHERE p.id_marca = m.id”;
    $result = mysqli_query($conn, $sql);

    if (mysqli_num_rows($result) > 0) {

        while($producto = mysqli_fetch_assoc($result)) {

    ?>

            <tr class=”alineacionmediotabla”>
              <th scope=”row”><?=$producto[“asin”]?></th>
              <td class=”fotoadministrar”>
                      <span><img class=”img-responsive” src=”<?=$producto[“foto”]?>”></span>
              </td>
            <td><?=$producto[“nombre_marca”] . “ “ . $producto[“modelo”]?></td>
              <td><?=$producto[“precio”]?> €</td>
              <td><?=$producto[“veces_puntuado”]?></td>
              <td>
                     <a href=”puntuarproducto.php?asin=<?=$producto[“asin”]?>” class=”btn btn-primary btn-warning”> Puntuar <i class=”far fa-archive iconomargenizquierdo”></i></a>
```

PUNTUARPRODUCTO.PHP:

Este algoritmo permite que se seleccione una o más categorías, con sus correspondientes puntuaciones de 1 a 10. Si los campos han sido cubiertos, la información se guarda en la tabla correspondiente, dentro de la base de datos.

```
if (isset($_POST[“asin”])) {

    $asin = $_POST[“asin”];

    if (isset($_POST[“id_categoria1”]) && isset($_POST[“puntuacion1”])){

        $sql = “INSERT INTO puntuacion (asin, id_categoria, puntuacion)
        VALUES (‘” . $_POST[“asin”] . “’, “ . $_POST[“id_categoria1”] . “, “ .
$_POST[“puntuacion1”] . “)”;

        if ($conn->query($sql) === TRUE) {
            $mensaje .= “Producto puntuado correctamente.<br>”;
        } else {

            $mensaje .= “Error al puntuar el producto.<br>”;
            $error = true;
        }

    }

    if (isset($_POST[“id_categoria2”]) && isset($_POST[“puntuacion2”])){

        $sql = “INSERT INTO puntuacion (asin, id_categoria, puntuacion)
        VALUES (‘” . $_POST[“asin”] . “’, “ . $_POST[“id_categoria2”] . “, “ .
$_POST[“puntuacion2”] . “)”;

        if ($conn->query($sql) === TRUE) {
            $mensaje .= “Producto puntuado correctamente.<br>”;
        } else {

            $mensaje .= “Error al puntuar el producto.<br>”;
            $error = true;
        }
```

```
   if (isset($_POST[“id_categoria3”]) && isset($_POST[“puntuacion3”])){

        $sql = “INSERT INTO puntuacion (asin, id_categoria, puntuacion)
        VALUES (‘” . $_POST[“asin”] . “’, “ . $_POST[“id_categoria3”] . “, “ .
$_POST[“puntuacion3”] . “)”;

        if ($conn->query($sql) === TRUE) {
            $mensaje .= “Producto puntuado correctamente.<br>”;
        } else {

            $mensaje .= “Error al puntuar el producto.<br>”;
            $error = true;
if (isset($_GET[“asin”])) {

    $asin = $_GET[“asin”];

    $sql = “SELECT p.asin, p.modelo, p.descripcion, p.precio, p.fecha_sinc, p.
foto, p.enlace, p.veces_sincronizado, m.nombre as nombre_marca FROM producto p,
marca m WHERE p.id_marca = m.id AND p.asin = ‘” . $asin . “’”;

    $result = mysqli_query($conn, $sql);

    if (mysqli_num_rows($result) > 0) {

        $producto = mysqli_fetch_assoc($result);
```

```
$sql = “SELECT id, nombre FROM categoria”;

$resultadoCategorias = mysqli_query($conn, $sql);

$categorias = mysqli_fetch_all($resultadoCategorias, MYSQLI_ASSOC);
```

SINCRONIZACIÓN.PHP:

Este algoritmo interactúa con bd.php y amazon.php para poder tomar el precio de la API y poder hacer un UPDATE con este dentro de la base de datos. En caso de que haya un error, se muestra un mensaje por pantalla.

```
if(isset($_POST[“asin”])){

    $asin = $_POST[“asin”];

    $precio = obtenerAmazonPrecio($asin);

    if (is_null($precio)) {

        $mensaje = “El producto con ASIN “ . $asin . “ no se ha sincronizado.”;

    } else {

       $sql = “UPDATE producto SET precio = “ . $precio . “, fecha_sinc = NOW(),
veces_sincronizado = veces_sincronizado + 1 WHERE asin = ‘” . $asin . “’”;

        if ($conn->query($sql) === TRUE) {

            $mensaje = “El producto con ASIN “ . $asin . “ se ha sincronizado
correctamente.”;

        } else {

            $mensaje = “El producto con ASIN “ . $asin . “ no se ha sincroniza-
do.”;
```

```
$sql = “SELECT p.asin, p.modelo, p.descripcion, p.precio, p.fecha_sinc, p.foto,
p.enlace, p.veces_sincronizado, m.nombre as nombre_marca FROM producto p, marca
m WHERE p.id_marca = m.id”;
    $result = mysqli_query($conn, $sql);

    if (mysqli_num_rows($result) > 0) {

        while($producto = mysqli_fetch_assoc($result)) {

    ?>

            <tr>
              <th scope=”row”><?=$producto[“asin”]?></th>
            <td><?=$producto[“nombre_marca”] . “ “ . $producto[“modelo”]?></td>
              <td><?=$producto[“fecha_sinc”]?></td>
              <td><?=$producto[“precio”]?> €</td>
              <td><?=$producto[“veces_sincronizado”]?></td>
              <td>
                  <form action=”sincronizacion.php” method=”post”>
                      <input type=”hidden” name=”asin” value=”<?=$producto[“a-
sin”]?>” />
                       <button class=”btn btn-primary” type=”submit”>Sincroni-
zar</button>
                  </form>
              </td>
            </tr>
```

ADMINSINCRONIZACIÓN.PHP:

Sincroniza todos los precios al pulsar un botón manualmente y permite indicar la hora de la sincronización automática.

```
if (isset($_GET[“actualizarPrecios”])) {

    sincronizarPrecios();
    $mensaje = “Se han actualizado los precios en la base de datos”;

}

if (isset($_POST[“hora”])) {

    $hora = $_POST[“hora”];

    $error = !actualizarValorPropiedad(“sincronizacion.hora”, $hora);
    $mensaje = “Se ha actualizado la hora en la base de datos”;
}
```

```
if (isset($_POST[“frecuencia”])) {

    $frecuencia = $_POST[“frecuencia”];

     $error = !actualizarValorPropiedad(“sincronizacion.frecuencia”, $frecuencia);
    $mensaje = “Se ha actualizado la frecuencia en la base de datos”;
}
```

3.1.3 Otros

BD.PHP:

Este es el algoritmo que permitirá hacer la conexión con la base de datos en todos los documentos php que lo necesiten. Además, incluye funciones cruciales para el funcionamiento de la aplicación, que más tarde serán utilizados en otros algoritmos.

```
// Nos conectamos a la base de datos
function crearConexion(){


    $conn = new mysqli(SERVERNAME, USERNAME, PASSWORD, DBNAME);

    if ($conn->connect_error) {
        die(“Connection failed: “ . $conn->connect_error);
    }
    mysqli_set_charset($conn,”utf8”);

    return $conn;
}
```

```
// Obtenemos las categorías

function obtenerCategorias($asin){

    $conn = crearConexion();

    $sql = “SELECT c.id, c.nombre, c.icono, c.alias, c.icono_final FROM categoria c, puntuacion p WHERE p.id_categoria = c.id AND p.asin=’” . $asin .”’”;

    $result = $conn->query($sql);

    $conn->close();

    return $result;
}
```

```
// Contador de ventas realizadas

function incrementarVentas($asin) {

    $exito = false;

    $conn = crearConexion();

   $sql = “UPDATE producto SET ventas = ventas + 1 WHERE asin = ‘” . $asin . “’”;

    if ($conn->query($sql) === TRUE) {

        $exito = true;

    }
    $conn->close();
    return $exito;
}
```

```
// Actualizamos el valor de la propiedad

function actualizarValorPropiedad($propiedad, $valor) {

    $exito = false;

    $conn = crearConexion();

    $sql = “UPDATE configuracion SET valor = ‘” . $valor . “’ WHERE propiedad =
‘” . $propiedad . “’”;

    if ($conn->query($sql) === TRUE) {

        $exito = true;

    }
    $conn->close();
    return $exito;
}
```

```
// Obtenemos el valor de la propiedad

 function obtenerValorPropiedad($propiedad){

    $valor = null;

    $conn = crearConexion();

   $sql = “SELECT valor FROM configuracion WHERE propiedad=’” . $propiedad . “’”;

    $result = mysqli_query($conn, $sql);

    if (mysqli_num_rows($result) > 0) {

        $configuracion = mysqli_fetch_assoc($result);
        $valor = $configuracion[“valor”];

    }

    $conn->close();

    return $valor;
}
```

```
// Obtenemos la respuesta seleccionada por el usuario

function nombreRespuesta($id_respuesta){

    $nombre = “”;

    $conn = crearConexion();

    $sql = “SELECT nombre FROM respuesta WHERE id=” . $id_respuesta;
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {

        $categoria = $result->fetch_assoc();
        $nombre = $categoria[“nombre”];
    }

    $conn->close();

    return $nombre;
}
```

CONFIG.PHP:

Este es el algoritmo que permitirá hacer la conexión con la base de datos en todos los documentos php que lo necesiten. Además, incluye funciones cruciales para el funcionamiento de la aplicación, que más tarde serán utilizados en otros algoritmos.

```
// Definimos puntuación máxima a 10 y los datos de acceso a la base de datos
define('PUNTUACION_MAX', 10);
define('SERVERNAME', "localhost");
define('USERNAME', "");
define('PASSWORD', "");
define('DBNAME', "");
```

AMAZON.PHP:

Este es el algoritmo que interactúa directamente con la API de Amazon. Y que permite tomar el precio de su base de datos. Además, incluimos 2 funciones para sincronizar y actualizar los precios en la base de datos.

```
// Obtener el precio de la base de datos de Amazon

function obtenerAmazonPrecio($asin){

    $precio = null;
    // Datos para la conexión a la API Amazon
    $keyId = '';
    $secretKey = '';
    $associateId = '';

    // Setup a new instance of the AmazonUrlBuilder with your keys
    $urlBuilder = new AmazonUrlBuilder(
        $keyId,
        $secretKey,
        $associateId,
        'es'
    );

    // Setup a new instance of the AmazonAPI and define the type of response
    $amazonAPI = new AmazonAPI($urlBuilder, 'simple');

    $items = $amazonAPI->ItemLookUp($asin);

    if (count($items) > 0){

        $precio = $items[0]["lowestPrice"];

    }
    return $precio;
```

```
//Sincronizar el precio de todos los productos

function sincronizarPrecios() {

    $conn = crearConexion();

    $sql = “SELECT p.asin FROM producto p”;
    $result = mysqli_query($conn, $sql);

    if (mysqli_num_rows($result) > 0) {

        while($producto = mysqli_fetch_assoc($result)) {

            $precio = obtenerAmazonPrecio($producto[“asin”]);
            actualizarPrecio($producto[“asin”], $precio);
        }
    }
    $conn->close();

}
```

```
// Actualizar el precio de todos los productos

function actualizarPrecio($asin, $precio) {

    $exito = false;

    $conn = crearConexion();

    $sql = “UPDATE producto SET precio = “ . $precio . “, precio_anterior = pre-
cio, fecha_sinc = NOW(), veces_sincronizado = veces_sincronizado + 1 WHERE asin
= ‘” . $asin . “’”;

    if ($conn->query($sql) === TRUE) {

        $exito = true;

    }
    $conn->close();
    return $exito;
}
```

UTILS.PHP:

En estas funciones, calculamos la puntuación y contamos el total de respuestas finalizadas.

```
// Comprobamos si la clave está dentro de un array asociativo

  function findKey($match, array $array) {
    foreach ($array as $key => $value){
        if ($key == $match) {
            return true;
        }
    }
    return false;
  }
```

```
// Esta funcion devuelve un array con la puntuación para cada categoría


function calcularPuntuacion() {
    $puntos = array();
    foreach ($_SESSION['puntuaciones'] as $id_categoria => $puntos_preguntas){
       $puntos_categoria = 0;
       foreach ($puntos_preguntas as $orden_pregunta => $puntos_respuesta){

           $puntos_categoria += $puntos_respuesta;
       }

        $puntos[$id_categoria] = $puntos_categoria;
    }
    return $puntos;
}
```

```
// Contamos el total de categorías finalizadas

function total_finalizadas(){

        return count($_SESSION['finalizadas']);
    }
```

SINCRONIZACIONPROGRAMADA.PHP:

Este es un programa que se ejecuta cada minuto, gracias al crontab (programador de tareas). Solamente sincronizará a la hora guardada en la base de datos.

```
// Con crontab -e
// * * * * * curl http://localhost/weizy/sincronizacionprogramada.php
// * * * * * cd /Applications/XAMPP/xamppfiles/htdocs/weizy && php sincroniza-
cionprogramada.php >> /var/log/sincronizacionprogramada.log 2>&1
require("bd.php");
require("amazon.php");

$minutosMargen = 5;

$fechaActual = date("Y-m-d H:i:s");
$timeActual = strtotime($fechaActual);

$time = $timeActual - ($minutosMargen * 60);
$fechaIni = date("Y-m-d H:i:s", $time);

$time = strtotime($fechaActual);
$time = $timeActual + ($minutosMargen * 60);
$fechaFin = date("Y-m-d H:i:s", $time);

$horaSincro = obtenerValorPropiedad('sincronizacion.hora');
$soloFechaActual = date("Y-m-d", $timeActual);

$timeSincro = strtotime($soloFechaActual . " " . $horaSincro . ":00");
$fechaSincro = date('Y-m-d H:i:s',$timeSincro);

echo $fechaSincro . "<br>";
echo $fechaIni . "<br>";
echo $fechaActual . "<br>";
echo $fechaFin;

if ($fechaSincro >= $fechaIni &&  $fechaSincro <= $fechaFin) {
    echo "Sincronizamos.";
    sincronizarPrecios();
} else {
    echo "NO Sincronizamos.";
}
```

3.2 Seguridad

Por la tipología de esta aplicación, no existen grandes riesgos de seguridad.

En la sección del front office con la que interactúa el usuario, no hemos estimado oportuno hacer ningún tipo de registro de usuario, por lo que no se estaría procesando ni guardando ningún tipo de dato sensible.

Por otro lado, en el caso del panel de administración, sí que tendríamos este problema. Nos encontramos con un formulario de acceso, aunque no hemos desarrollado un formulario de registro.

El resto de las páginas del panel de administración están protegidas por la misma sesión. Es decir, es necesario que el administrador acceda mediante el formulario de LOGIN, para poder utilizar cualquiera de las herramientas del panel.

```
// Fragmento de inicio de sesión para una página del panel de administración

     <?php

     session_start();

     if (!isset($_SESSION[“usuario”])) {

          header(“location: login.php”);

     }

     ?>
```

Para este formulario hemos creado el documento login.php y logout.php, de los que hablábamos anteriormente. Este algoritmo comprueba que en la base de datos existe el usuario y su correspondiente contraseña, para poder iniciar la sesión. Por otra parte, podemos ver en el código como la contraseña está encriptada para proteger los datos en caso de acceso indeseado a la base de datos. Finalmente guardamos los datos relacionados con el administrador en sesión. Utilizamos el método SHA2 de 256 bits.

```
if (isset($_POST[“usuario”]) && isset($_POST[“password”])) {
     $usuario = $_POST[“usuario”];
     $password = $_POST[“password”];

     // Creamos la conexión

    $conn = crearConexion();
    $sql= “SELECT usuario, foto  FROM `usuario` WHERE usuario=’” . $usuario . “’
and pwd=SHA2(‘” . $password . “’, 256)” ;
    $result = $conn->query($sql);
    if ($result->num_rows == 1) {

     $usuario = $result->fetch_assoc();

// Guardamos el usuario en sesión

        $_SESSION[‘usuario’] = $usuario[“usuario”];
```

```
// Guardamos la foto en sesión

        $_SESSION['foto'] = $usuario["foto"];

          header("location: admin.php");

          // Mostramos un error en caso contrario

    } else{

          $error = true;

     }

<form class="form-horizontal animated fadeInLeft" action="login.php" metho-
d="POST">
```

Utilizamos el método POST para el envío de los datos del formulario de forma más segura. Tanto el usuario como la contraseña son campos requeridos obligatoriamente.

3.3 Test

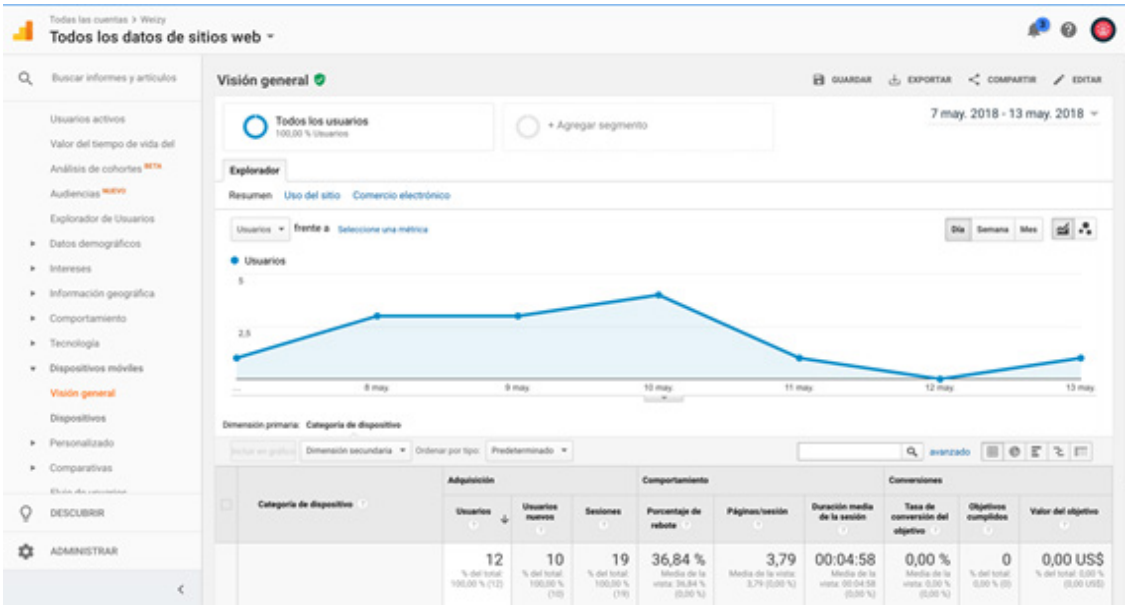
Se han creado diferentes test, que a continuación se detallarán, para comprobar el correcto desarrollo de la aplicación.

3.3.1 Usuario

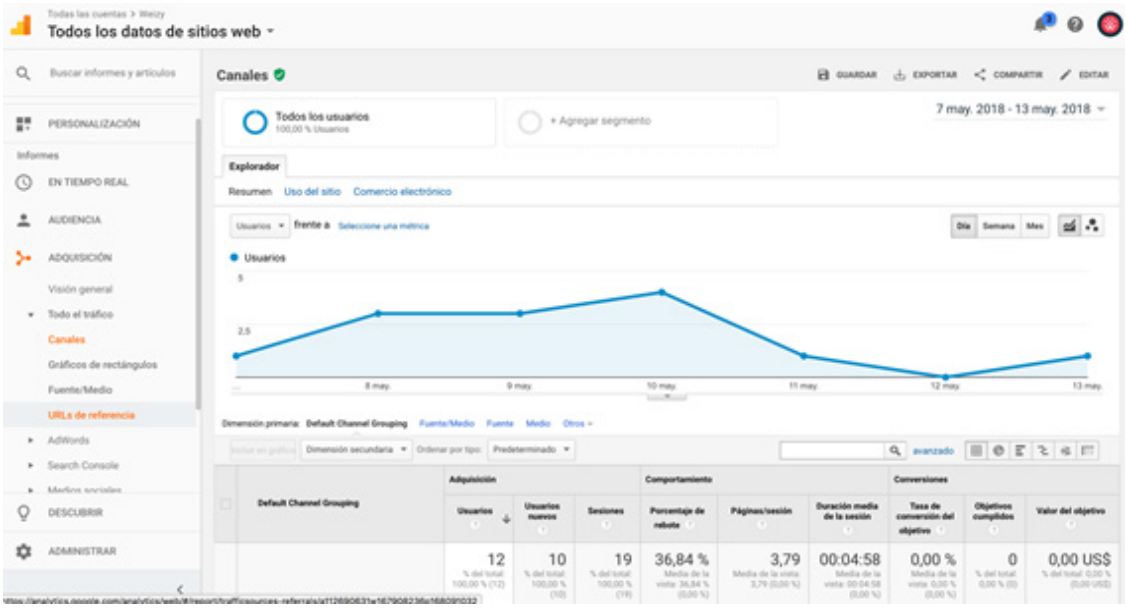
Prueba con Google Analytics: Mediante el uso de esta herramienta hemos querido realizar un análisis acerca de la audiencia.

La tasa de rebote: es el porcentaje de usuarios que visitan única y exclusivamente una página. Es decir, cuando en un sitio web se produce una sesión de una sola página.

Audiencia > Móvil > Visión General



Adquisición > Todo el tráfico > Canales



Un canal o un canal de marketing es un grupo de varias fuentes de tráfico con el mismo medio. En el primer apartado, dispositivos móviles, podemos ver que la tasa de rebote es del 45,45%. Y que la duración media de la sesión es de 4 minutos y 18 segundos. Por otro lado, en la sección de canales se muestra un gráfico de usuarios y aquellos diferentes canales de los que provienen: búsqueda orgánica, social, directa o referral. Si prestamos atención a la información inferior, podemos ver la segmentación de la tasa de rebote por canales: 25% búsqueda orgánica, 100% social, 66.67% directos, y 0% referral. Además, también podemos ver la duración media de la sesión. De modo que podemos intuir qué tipo de público (de dónde procede) es el que interactúa o se interesa más por nuestro contenido en smartphones y recomendaciones.

Estos datos son importantes porque permiten detectar qué usuarios no ven interesante el contenido del sitio, o no han encontrado el teléfono que buscaban puesto que entran en la app y sin navegar a través de ninguna categoría, abandonan el sitio. De modo que podría servir como indicador para medir el nivel de satisfacción de los usuarios. Es decir, a lo mejor deberíamos prestar más atención en los contenidos de nuestro sitio para que los usuarios interactúen más.

3.3.2 Usabilidad

Utilizamos este script para para poder hacer el seguimiento de los usuarios dentro de nuestra aplicación, utilizando la herramienta Hotjar.

```
<script>
    (function(h,o,t,j,a,r){
        h.hj=h.hj||function(){(h.hj.q=h.hj.q||[]).push(arguments)};
        h._hjSettings={hjid:876419,hjsv:6};
        a=o.getElementsByTagName('head')[0];
        r=o.createElement('script');r.async=1;
        r.src=t+h._hjSettings.hjid+j+h._hjSettings.hjsv;
        a.appendChild(r);
    })(window,document,'https://static.hotjar.com/c/hotjar-','.js?sv=');
</script>
```

El mapa de calor nos permite medir el modo en el que los usuarios interactúan con los elementos contenidos dentro de una página web determinada. De forma que, mediante un código de colores podemos interpretar qué partes de esta llaman más la atención del usuario. El código de colores utilizado es el mismo en todas las pruebas realizadas. Desde el análisis de clics, a el de movimiento, etc. El código de colores representa desde el tono rijo al azulado. Donde los colores más cálidos hacen referencia a aquellas partes donde se concentran la mayor parte de las interacciones de los usuarios, mientas que las zonas frías hacen referencia a lo contrario.

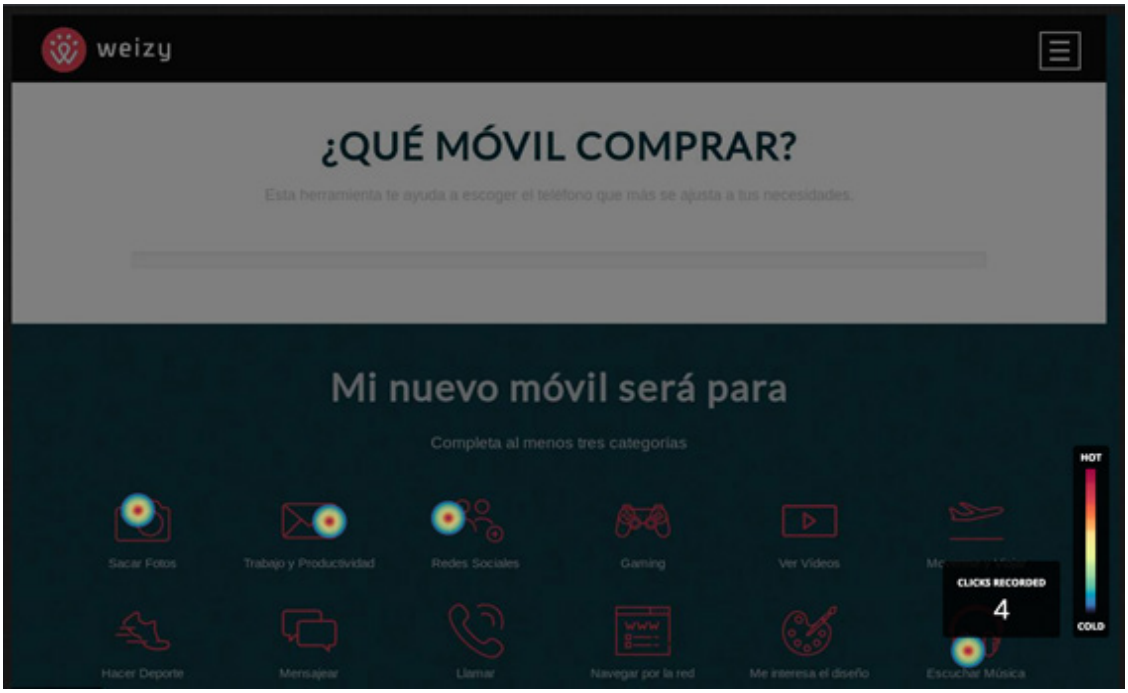
Análisis del mapa de movimientos de cursor:

Recoge el cambio de posición del ratón, registrando tanto cuando está en movimiento como cuando está parado. De este modo, podemos hacer un seguimiento de los movimientos del usuario.

En la siguiente captura podemos ver el en acción el código de colores del que hablábamos antes. Se aprecia claramente como las zonas que despiertan más interés en los usuarios, son aquellas situadas coincidiendo con la zona de lectura de categorías y preguntas, y con mayor intensidad en los enlaces.

Podríamos suponer que la mirada del usuario sigue el mismo trayecto que el cursor del ratón y cuando su mirada está entrada en alguna parte de la página, el cursor está también alrededor de es mismo lugar. Se aprecia menor movimiento en las fotografías o zonas sin ningún elemento.

Análisis sobre los clics:



Seleccionamos la opción “clics”, y en este mapa se recogen los puntos de la página web donde los usuarios han hecho clic con el ratón. Podemos comprobar como se mantiene el código de colores también, en esta herramienta.

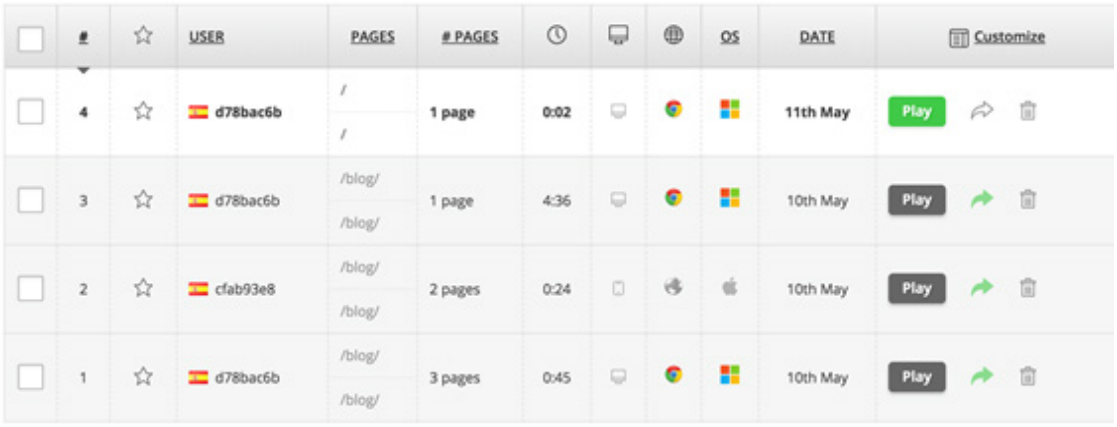
Al final de la página, se hace un recuento total del número de clics, y si situamos el cursor por encima de alguna de las zonas calientes, la herramienta nos hace una media del porcentaje de clics en cada zona. Parece que además esta herramienta recoge clics de otros elementos como imágenes o texto sin enlaces.

Análisis sobre las grabaciones de sesiones:

Esta es una sección a parte, es necesario activarla también si se desea que se hagan seguimientos. La herramienta no da la posibilidad de descargar los vídeos de las sesiones, pero permite compartirlos con terceros dando una autorización previa.

Esta sección muestra vídeos en tiempo real de las visitas a una página por cada usuario. De modo que se puede saber exactamente los puntos recorridos con el cursor, y aquellos elementos con los que ha interactuado.

También, tenemos información acerca de cada usuario: código y nacionalidad, páginas visitadas, número de visitas, el tiempo de la sesión, antigüedad, sistema operativo y navegador utilizados. Podemos ver como en estas sesiones, se demuestra un poco las sospechas que teníamos con los primeros análisis de mapas de calor. Las zonas con texto, y zonas con enlaces son las que más llaman la atención de los usuarios.



3.3.3 Seguridad

En cuanto a la seguridad, hemos realizado varias pruebas de simulación de acceso a través del único formulario de la aplicación (panel de administración). Hemos corregido los posibles agujeros de seguridad.

3.4 Versiones de la aplicación

Hemos querido seguir desarrollando constantemente la aplicación hasta haber saltado directamente a la versión Beta. De este modo, hemos podido realizar los primeros testeos con usuarios reales, elemento necesario para poder hacer los test requeridos en la fase de testeo.

La versión Beta recopilaba las elecciones del usuario durante el proceso de categorías y respuestas. Aunque no existía ningún panel de administración, ni tampoco mostraba los smartphones ideales por pantalla al finalizar el proceso. Como se ve a continuación, la fase Beta incluía un formulario donde el usuario podía rellenar su email y añadir más datos. A partir de ahí, se le enviarían las recomendaciones personalizadas por correo electrónico.

Durante la PEC 3 del trabajo de fin de grado, hemos desarrollado la versión 1.0 de la aplicación. En este caso, el usuario ya podrá ver los teléfonos ideales por pantalla y el administrador podrá hacer las configuraciones previstas. Gracias a las correcciones de esta PEC por parte del profesor, desarrollaremos la versión final, perfectamente funcional y lista para el día de la entrega del trabajo de fin de grado.

3.5 Bugs

No se han detectado errores importantes en las pruebas realizadas con usuarios.

Por otro lado, sí que hemos tenido una gran cantidad de errores que corregir durante la fase de desarrollo de la aplicación:

Errores de lógica:

- Rescribir funciones existentes de PHP
- No haber utilizado funciones para separar lógicamente la parte de PHP de la parte de HTML.
- Haber olvidado el inicio de sesión, creando un fallo de seguridad grande al permitir que cualquier usuario acceda a una parte determinada del panel de administración.
- No cambiar los datos de conexión a la base de datos al exportarla al hosting web.

Errores de sintaxis:

- Problemas al comparar: el operador de comparación de igual es == y el símbolo = es el indicador de asignación de variable.
- Programar en PHP como si fuera otro lenguaje
- No hacer una revisión de código (no cerrar una llave, por ejemplo)

Todos los errores encontrados han sido corregidos rápidamente.

4 . Conclusiones

He de decir que ha sido un trabajo complejo, debido a la gran cantidad de funciones a desarrollar. Considero que es una aplicación ambiciosa, con posibilidades de futuro y de continuar desarrollando.

Las asignaturas más relevantes para el desarrollo de esta aplicación han sido:

Aquellas relacionadas con la especialización en Ingeniería Web

- Ingeniería del software
- Programación web avanzada
- Diseño y programación orientada a objetos
- Análisis y diseño de patrones

Otras

- Lenguajes y estándares web
- Diseño gráfico
- Arquitectura de la infromación
- Diseño de bases de datos

Por otra parte, como comento en el vídeo, esta aplicación podría extrapolarse a otros sectores como el de la cosmética o la alimentación. Y continuar desarrollando otras funcionalidades como la configuración de productos, etc.

Por mi parte, una vez finalizado el grado, me gustaría poder continuar con mi especialidad y seguir desarrollando aplicaciones que resuelvan un problema concreto de los usuarios, y que al mismo tiempo resulten rentables.

4 . Glosario

CDN: Una red de distribución de contenidos (CDN, content delivery network en inglés) es una red superpuesta de computadoras que contienen copias de datos, colocados en varios puntos de una red con el fin de maximizar el ancho de banda para el acceso a los datos de clientes por la red. Un cliente accede a una copia de la información cerca del cliente, en contraposición a todos los clientes que acceden al mismo servidores central, a fin de evitar embudos cerca de ese servidor.

PHP: acrónimo recursivo en inglés de PHP Hypertext Preprocessor (procesador de hipertexto), es un lenguaje de programación de propósito general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en un documento HTML en lugar, de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera el HTML resultante.

API: La interfaz de programación de aplicaciones, abreviada como API del inglés: Application Programming Interface,1 es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Front Office: Es un término que traducido literalmente significa: oficina de delante. Viene indicado como el conjunto de las estructuras de una organización que gestionan la interacción con el cliente.

Back Office: En la gestión empresarial el back office (en español significa literalmente oficina trasera u oficina de trastienda) es el conjunto de actividades de apoyo al negocio. Es decir, la parte de las empresas que realizan las tareas destinadas a gestionar la propia empresa y que no tienen contacto directo con el cliente, como las labores informáticas de comunicaciones, de gestión de recursos humanos, contabilidad o finanzas.

Boostrap: Es un framework web o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. A diferencia de muchos frameworks web, solo se ocupa del desarrollo front-end.

5 . Bibliografía

Front Office y Back Office, Fundación politécnica de la comunidad valenciana

https://www.antiguosupv.org/innovaaccion/vocabulario/back-office-y-front-office

CDN, Red de distribución de contenidos, Wikipedia

https://es.wikipedia.org/wiki/Red_de_distribuci%C3%B3n_de_contenidos

API, Interfaz de programación de aplicaciones, Wikipedia

https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones

PHP, Wikipedia

https://es.wikipedia.org/wiki/PHP

Bootstrap, Bootstrap (framework), Wikipedia

https://es.wikipedia.org/wiki/Bootstrap_(framework)

Monjo Palau, Tona. Diseño centrado en el usuario. (Módulo 2. Asignatura: Diseño de interfaces multimedia). Universitat Oberta de Catalunya. Barcelona: FUOC.

Monjo Palau, Tona. Usabilidad. (Módulo 3. Asignatura: Diseño de interfaces multimedia). Universitat Oberta de Catalunya. Barcelona: FUOC.

Pradel i Miquel, Jordi; Raya Martos, José. Análisis UML. (Módulo 4. Asignatura: Ingeniería del software). Universitat Oberta de Catalunya. Barcelona: FUOC.

Pradel i Miquel, Jordi; Raya Martos, José. Requisitos. (Módulo 3. Asignatura: Ingeniería del software). Universitat Oberta de Catalunya. Barcelona: FUOC.

Patricia Gil, Eva; de Lera Tatjer, Eva; Monjo Palau, Antónia. Usuarios y sistemas interactivos (Módulo anexo. Asignatura: Arquitectura de la Información). Universitat Oberta de Catalunya. Barcelona: FUOC.

Morville, Peter; Rosenfeld, Louis (2006). Arquitectura de la información para la World Wide Web. California (USA): O'Reilly Media Inc.

Costal Costa, Dolors. Introducción al diseño de bases de datos (Módulo anexo. Asignatura: Uso de Bases de Datos). Universitat Oberta de Catalunya. Barcelona: FUOC.