Universidad Autónoma Metropolitana

Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Reporte Final del Proyecto de Integración

Licenciatura en Ingeniería en Computación

Modalidad de Proyecto Tecnológico

Aplicación móvil para la recomendación de productos en el comercio electrónico

Karina Guzmán Villanueva

207301255

Asesor: María Lizbeth Gallardo López

14-O

Fecha de entrega:

12 de enero de 2015

Yo, María Lizbeth Gallardo López, *declaro que aprobé el contenido* del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Yo, Karina Guzmán Villanueva, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Resumen

Los sistemas de recomendación frecuentemente son integrados en plataformas de comercio electrónico. Las recomendaciones suelen considerar dos aspectos: 1) los productos que un usuario a adquirido previamente; formando un perfil de consumo; y 2) basados en las opiniones de otros usuarios; formando comunidades de personas que comparten en un cierto grado las mismas preferencias. Debido al éxito que se ha tenido con el comercio electrónico se desarrolló el proyecto "Aplicación móvil para la recomendación de productos en el comercio electrónico", el cual busca que las recomendaciones de equipos de cómputo sean más adecuadas a las necesidades, expresadas a través de un conjunto de características y su ponderación correspondiente. La ponderación es definida por el consumidor. Para realizar las recomendaciones sobre el equipo de cómputo, se implementó un algoritmo heurístico llamado *Logic Scoring of Preferences* (Puntuación lógica de preferencias).

Índice

Contenido

Índice de figuras

Índice de tablas

1. Introducción

El avance en las tecnologías de la información han permitido, entre otras cosas: la comunicación a través de dispositivos móviles y el comercio electrónico. En efecto, el uso de los dispositivos móviles se ha extendido en el mundo (celulares, tablets), y día tras día el número de usuarios de estos dispositivos va creciendo de tal forma que se han hecho indispensables en la vida de muchas personas. El uso que se da a los dispositivos móviles es diverso, entre otros: entretenimiento y negocios. Por su parte, el comercio electrónico permite realizar compras en línea proponiendo a los consumidores otra manera de encontrar los productos que necesitan, un ejemplo es Mercadolibre, el cual proporciona un medio para que los vendedores y consumidores de distintos lugares geográficos se pongan en contacto para intercambiar productos [1]; también se encuentra Amazon [2], quizá el más conocido a nivel mundial; y otro que puede mencionarse es eBay [3] que se dedica a hacer subastas de diferentes artículos.

El comercio electrónico, está haciendo uso de sistemas de recomendación. Un sistema de recomendación predice el grado de preferencia que una persona puede tener sobre un objeto, tales como libros, música o películas [4]. Por ejemplo, Amazon, recomienda automáticamente nuevos productos a sus clientes, basado en las compras que éste y otros clientes han realizado; además, de considerar los votos (a nivel de satisfacción) que proporcionan sobre los productos. Este proyecto terminal busca experimentar otra manera de realizar la recomendación de productos, proponiendo una herramienta llamada RECOM, basado en el algoritmo heurístico *LSP,* por sus siglas en inglés *Logic Scoring of Preferences* para soportar el análisis de decisión para las recomendaciones. LSP requiere definir un conjunto de características para el producto y posteriormente, solicitar al usuario una ponderación para cada una de ellos. Este algoritmo podría integrarse a la técnica basada en un perfil de consumo, o a la basada en comunidades de preferencias similares, para mejorar la precisión de las recomendaciones que una plataforma de comercio electrónico puede ofrecer.

RECOM realiza recomendaciones de equipos de cómputo, señalando las características más importantes, a través de una ponderación que proporcione un consumidor. Esta ponderación es la entrada del algoritmo heurístico LSP. Es importante señalar que los componentes de RECOM son: una aplicación web, un servicio web y una aplicación cliente, esta última diseñada para dispositivos móviles con sistema operativo *Android*.

2. Antecedentes

Los siguientes proyectos son una muestra de trabajos internos y externos relacionados con el presente proyecto:

2.1. Referencias internas

Aplicación Colaborativa para Dispositivos Móviles con Sistema Operativo Android [5]. Esta aplicación consiste en un pizarrón colaborativo donde varios usuarios pueden editar figuras geométricas desde diferentes computadoras; el requisito es que estén conectadas a Internet. La Aplicación Colaborativa y RECOM operan en dispositivos móviles con sistema operativo Android; sin embargo, RECOM fue diseñada para un consumidor de equipo de cómputo a quien le hace recomendaciones en base a preferencias; el consumidor deberá conectarse a un servicio web para obtener tales recomendaciones.

Gestión de calificaciones de cursos mediante servicios Web [6]. Esta aplicación plantea el uso de servicios web para gestionar calificaciones. RECOM y este gestor de calificaciones son servicios web, pero tienen propósitos diferentes, ya que RECOM recomienda productos de cómputo y la otra aplicación gestiona calificaciones.

Interfaz para la administración en línea de cursos presenciales [7]. Este proyecto tiene como objetivo crear una plataforma virtual que administre cursos presenciales, es similar a RECOM porque ambos gestionan recursos en una base de datos, pero RECOM cuenta con un servicio web.

2.2. Referencias externas

Mercadolibre [1]. Es una aplicación web por medio del cual los usuarios pueden vender y comprar cosas, en América Latina. RECOM solo ofrece la recomendación de equipos de cómputo. Además no se observa que Mercadolibre ofrezca una ponderación de sus productos.

Amazon [2]. Es una tienda de Estados Unidos de comercio electrónico, la cual tiene páginas sede para diferentes países donde se venden todo tipo de artículos, en cambio RECOM solo recomienda equipos de cómputo.

Dell [8]. La página de Dell cuenta con una opción en la cual pueden ser ajustadas algunas de las características de los equipos que ofrece, como son: el tipo de pantalla, memoria RAM, etc. RECOM hace algo similar, pero tomando en cuenta la ponderación del comprador sobre cada característica.

3. Justificación

Los sistemas de recomendación frecuentemente son integrados en plataformas de comercio electrónico. Las recomendaciones suelen considerar dos aspectos: 1) los productos que un usuario a adquirido previamente; formando un perfil de consumo; y 2) basados en las

opiniones de otros usuarios; formando comunidades de personas que comparten en un cierto grado las mismas preferencias.

Debido al éxito que se ha tenido con el comercio electrónico se realizó este proyecto, el cual busca que las recomendaciones de equipos de cómputo sean más adecuadas a las necesidades, expresadas a través de un conjunto de características y su ponderación correspondiente. La ponderación es definida por el consumidor.

Construiremos un servicio web que proporcione recomendaciones de equipos de cómputo, el cual es accedido por dispositivos móviles; con el propósito de ofrecer a los usuarios de estos dispositivos una forma de ponderar las características de un producto, de acuerdo a sus necesidades.

La idea que se implementó está enfocada a equipos de cómputo, sin embargo, podría extenderse a cualquier producto.

4. Objetivos

4.1 Objetivo general

Implementar un algoritmo heurístico que permita la recomendación de compra de productos, en un servicio de compras por Internet.

4.2 Objetivos específicos

- Adaptar e implementar el algoritmo heurístico de lógica de puntuación de preferencias (LSP) en el proceso de recomendación de productos que vende una empresa de computadoras.

- Diseñar e implementar un módulo que permita dar de alta, baja, modificar y consultar productos.

- Diseñar e implementar una base de datos que contenga la información de los productos a vender.

- Diseñar e implementar un módulo que permita a un consumidor emitir un voto sobre el producto de la elección del consumidor.

5. Marco teórico

5.1. Comercio electrónico

El comercio electrónico es definido por los estudios de la Organización para la Cooperación y el Desarrollo Económicos (OCDE) como el proceso de compra, venta o intercambio de bienes, servicios e información a través de las redes de comunicación. Representa una gran variedad de posibilidades para adquirir bienes o servicios ofrecidos por proveedores en diversas partes del mundo. Las compras de artículos y servicios por internet (también llamadas en línea) pueden resultar atractivas por la facilidad para realizarlas, sin embargo, es importante que los ciberconsumidores tomen precauciones para evitar ser víctimas de prácticas comerciales fraudulentas [9].

5.2. Algoritmo LSP

La figura 1 muestra el algoritmo heurístico LSP [10], el cual considera una serie de criterios x_i que son evaluados a través de una función de normalización $G_i(x_i)$ para obtener un valor de preferencia E_i comprendida entre [0..1]. Finalmente los E_i son agregados en la función $L(E_1, E_2,\ldots, E_k)$ que obtiene el grado de satisfacción del conjunto de criterios definidos para un producto.

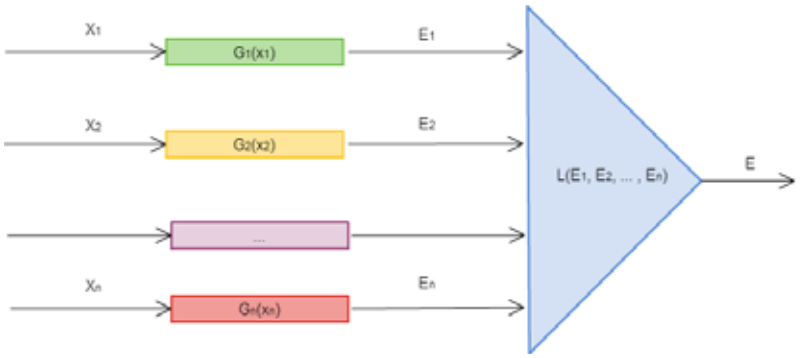

Figura 1. Diagrama del algoritmo de puntuación lógica de preferencias LSP [10]

5.3. *Web service*.

Un *web service* [11] es un componente de software que puede ejecutar tanto procesos como funciones, las cuales pueden ser consumidas por otro módulo de software, aunque este otro modulo este programado en otro lenguaje de programación; esto se logra por el intercambio de información mediante XML o JSON.

XML (en español lenguaje de marcas extensible) es una tecnología sencilla que se complementa por más tecnologías. Se basa en el uso de etiquetas como HTML, con diversos propósitos, por ejemplo configurar un programa o enviar información dentro de un servicio web.
JSON (por sus siglas Java Script Object Notation) es un formato ligero para envio y recepción de datos, ya que este se compone de cadenas, gracias a sus caracteres especiales pueden ser enviados objetos, arreglos o valores simples.

El protocolo REST, sirve para tener una comunicación sencilla entre un servidor web y los clientes que consumen este servicio. Este protocolo, principalmente se basa en 4 operaciones básicas, a saber: PUT, GET, POST y DELETE. Puede intercambiar XML o JSON dependiendo de cómo configuremos el servicio web.

6. Desarrollo del proyecto

6.1. Metodología de desarrollo.

La metodología que se siguió fue el proceso unificado para el desarrollo de software, el cual se caracteriza por ser iterativo e incremental. Consiste en cuatro fases: Inicio, elaboración, construcción y transición. Estas fases incluyen la realización de siete disciplinas. A continuación se describirán las disciplinas junto con su realización dentro de este proyecto.

Modelo del negocio. En esta etapa se debe de comprender la lógica del negocio. Al inicio del proyecto, se planteó el problema y se definieron los objetivos.

Requisitos. Consiste en determinar las necesidades del cliente. En esta etapa se identificaron a los actores principales, y se iniciaron los casos de uso de texto.

Análisis. Consiste en realizar el documento de especificaciones, donde se describe lo que hace el sistema. En esta etapa se concluyeron los casos de uso de texto, y se mejoraron en dos iteraciones.

Diseño. En esta disciplina se construye un modelo de solución. En esta etapa se creó el documento de diseño, a través de los siguientes artefactos: casos de uso de texto, diagramas de robustez, diagrama de arquitectura del sistema, diagrama entidad-relación para la base de datos.

Implementación. El diseño pasa a ser programado. En esta disciplina se inició la implementación del sistema, iniciando por la base de datos, luego se continuó por implementar el algoritmo; y por último se creó el servicio web, junto con el cliente que consume este servicio.

Pruebas. Consiste en diseñar pruebas para observar el comportamiento del sistema. En esta etapa se realizaron pruebas de unidad para cada módulo que se desarrolló; posteriormente, se realizaron pruebas de sistema, una vez que los módulos fueron integrados.

Implantación. Colocar el sistema en producción, para que el consumidor pueda utilizarlo. Esta etapa no forma parte de los alcances de este proyecto de integración.

6.2 Diseño del sistema

6.2.1. Diagrama de casos de uso

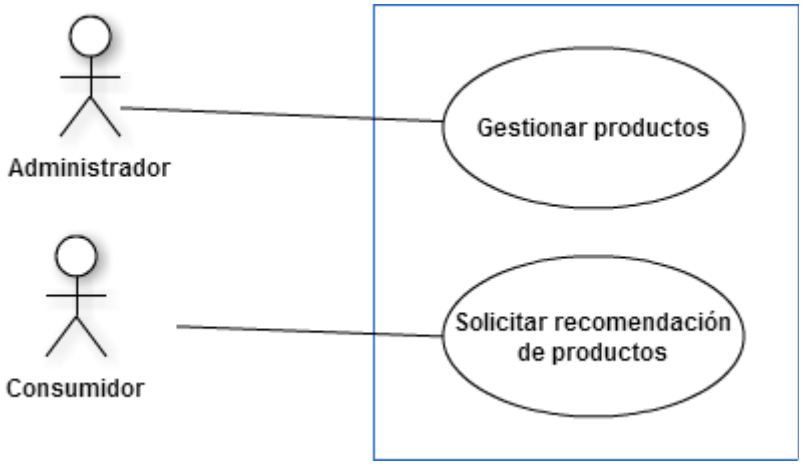

Figura 2. Diagrama de casos de uso general

El diagrama de casos de uso general para el sistema RECOM se muestra en la figura 3, los actores primarios son:

1. Administrador: Es el rol que tiene los permisos suficientes y necesarios para hacer cambios en los productos que tiene la base de datos, a saber: Agregar productos, eliminar productos, cambiar las propiedades de los productos y hacer consultas de los productos.

2. Consumidor: Es el rol que desea comprar un producto, así que por hace una elección de las características del producto que desea comprar y les da un peso; de esta forma el sistema le devuelve los productos en un orden lo más cercano posible a lo que el consumidor solicitó.

Los módulos principales del sistema son: Gestionar los productos y comprar productos. El módulo de gestionar productos incluye las operaciones de: Agregar productos, eliminar productos, cambiar las propiedades de los productos y hacer consultas de los productos.

1. El módulo de solicitar recomendaciones incluye las operaciones de: seleccionar las características del producto y seleccionar una ponderación a cada una de ellas; además de recomendar los productos cuyas características correspondan a la petición del consumidor.

En este momento RECOM tiene como productos únicamente equipos de cómputo: PC y laptop. Las características del equipo de cómputo con los que opera RECOM son: Disco Duro, RAM, Tamaño de pantalla, Tipo de equipo de cómputo, Popularidad entre los consumidores, Precio, Color, Marca, Procesador y Sistema Operativo.

Para ejemplificar el desarrollo de la aplicación RECOM, explicaremos uno de los principales casos de uso de texto, a saber: solicitar recomendación de productos.

Caso de uso	**Solicitar recomendación de productos**
Actor	Consumidor
Resumen	El consumidor desea recibir recomendaciones sobre un equipo de cómputo que cumpla algunas características. Las recomendaciones son presentadas en el orden de importancia que determinó el consumidor sobre tales características.
Precondición	El consumidor se encuentra en la aplicación RECOM
Poscondición	El consumidor recibe un conjunto de recomendaciones sobre equipo de cómputo
Disparador	El consumidor ingresa a la aplicación móvil RECOM instalada previamente en su dispositivo móvil.

Descripción

Acción del actor	**Respuesta del sistema**
1. El consumidor se encuentra en la interfaz principal de RECOM	2. RECOM muestra una lista de características para el equipo de cómputo, a saber: Disco Duro, RAM, Tamaño de pantalla, Tipo de equipo de cómputo, Popularidad entre los consumidores, Precio, Color, Marca, Procesador y Sistema Operativo Para cada una de estas características proporciona su importancia, a saber: • Imprescindible • Necesario • Me gustaría • No necesario • Prescindible
3. El consumidor elige las características que busca en el equipo de cómputo y la importancia de cada una de ellas.	
4. El consumidor da clic en el botón buscar.	5. RECOM recibe la ponderación del usuario y los transmite a un servicio Web,

para que realice el cálculo con el algoritmo LSP.
6. El servicio Web devuelve una lista de los equipos de cómputo que cumplen con las características y los ordena por el grado de importancia obtenido por el algoritmo LSP, a partir de la ponderación que registró del consumidor.

A continuación se explicará el proceso del algoritmo LSP para determinar la ponderación del equipo de cómputo; es decir el orden en que se propondrán al consumidor.

Se toma en cuenta las características que el usuario haya seleccionado como preferenciales, por ejemplo: Tipo de equipo de cómputo con un valor preferencial prescindible, color azul imprescindible y popularidad prescindible; las cuáles serán las entradas de nuestro algoritmo. Después se tendrá una función $G_i(x_i)$ encargada de normalizar el valor de x_i, el valor normalizado corresponde a E_i. La función de normalización $G_i(x_i)$ que emplearemos es una recta que va de 0 a 1:

$$G_i(x_i) = mx_i + b$$

Donde *m* (la pendiente de la recta) tiene un valor de 1, y *b* es 0 ya que la recta no se mueve de su eje, en la figura 3 podemos ver una gráfica de nuestra recta.

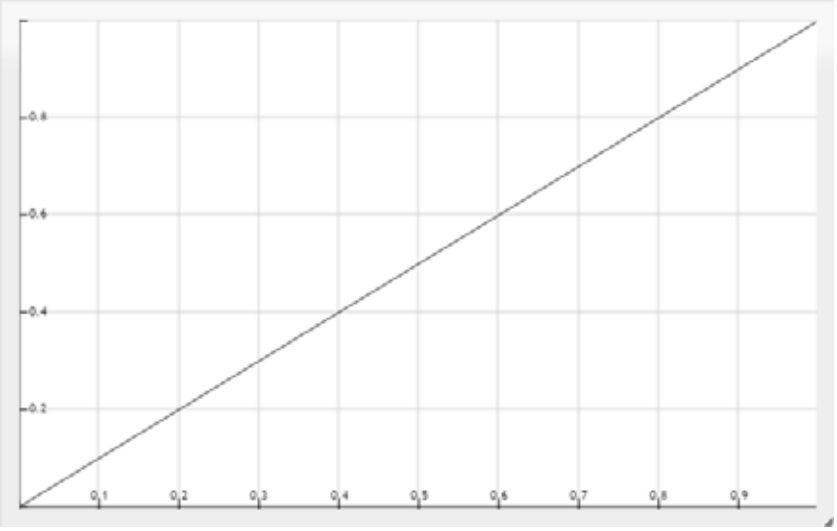

Figura 3. Gráfica de la función $G_i(x_i)$

Los valores de $G_i(x_i)$ varían de 0 a 1; donde 1 significa que la característica del equipo de cómputo es exactamente igual a la característica elegida por el consumidor; por ejemplo, si el consumidor eligió una memoria RAM de 8GB, y la memoria RAM del equipo de cómputo que se está evaluando es 8GB. En cambio, 0 significa que la característica del equipo de cómputo no corresponde con la característica elegida por el consumidor; por ejemplo, si el consumidor eligió una memoria RAM de 2GB, pero el equipo de cómputo

tiene 8GB ó 16GB en RAM. Un valor en el rango (0-1) significa que la característica del equipo de cómputo es similar a la característica elegida por el consumidor; por ejemplo, si el consumidor eligió el color gris, y el equipo de cómputo es de color plata, se considera que hay una similitud entre ellos. E_i se calcula a través de la comparación de la característica del producto que se está evaluando contra la característica que el consumidor proporcionó. Cada valor normalizado corresponde a una E_i que sirve de entrada a una función de agregación (ver Figura 4) para realizar la evaluación global de las características de un producto.

Figura 4.Modelo de Agregación

Cada E_i es agregada a la función $L(E_1, E_2, \ldots, E_k)$ para obtener finalmente el grado de satisfacción de las características de un producto sobre los criterios definidos por un consumidor. Para realizar el cálculo de la función de agregación se usa la fórmula siguiente:

$$L(r) = (w_1 E_1^r + w_2 E_2^r + \cdots + w_k E_k^r)^{\frac{1}{r}}$$

El valor de r depende del número de E_i involucrados en la función de agregación L, como mínimo puede tener 2 y como máximo 5. En una agregación, las relaciones pueden ser definidas por: fuerte quasi-disyunción, media quasi-disyunción, débil quasi-disyunción, débil quasi-conjunción, media quasi-disyunción y fuerte quasi-disyunción tal como lo muestra la tabla 1. Las relaciones de una agregación quedan definidas por las ponderaciones proporcionadas por el consumidor.

Por ejemplo, si para el consumidor es `imprescindible` el tipo de equipo de cómputo, y `necesario` el dispositivo óptico, entonces la relación entre ambas características sería una `media quasi-conjunción`. Dada la escala nominal que se le dió a cada característica, se observa que ambas son de una alta importancia para el consumidor. Por el contrario, si no fuesen importantes estas características para el consumidor, la función de agregación sería una disyunción. Establecida la función de agregación, el valor de r (de acuerdo a la tabla 1) sería -0.720, ya que la relación tiene dos características, se toma el valor que está en la columna r(2) y la fila de media quasi-conjunción.

Operador	Símbolo	r(2)	r(3)	r(4)	r(5)
Fuerte quasi-disyunción	D+	9.521	11.095	12.270	13.325
Media quasi-disyunción	DA	3.929	4.450	4.825	5.11
Débil quasi-disyunción	D-	2.018	2.187	2.302	2.384
Débil quasi-conjunción	C-	0.261	0.192	0.153	0.129
Media quasi-conjunción	CA	-0.720	-0.732	-0.721	-0.707
Fuerte quasi-conjunción	C+	-3.510	-3.114	-2.823	-2.605

Tabla 1. Valores posibles de r en las distintas relaciones

6.2.2. Diagrama de dominio de RECOM

El diagrama de dominio de la figura 5 es el resultado del caso de uso de texto: Solicitar recomendación de productos y gestionar productos. Las entidades principales son: `Producto, ProductoEvaluado y Ponderación.`

`Producto`. La clase producto contiene los atributos: Id_producto, Nombre, Color, RAM, Dispositivo Óptico, Disco duro, Procesador, Pantalla, Marca, Precio, Popularidad y Sistema Operativo, se utiliza para manipular los productos en el algoritmo, también se usa para dar de alta, dar de baja, modificar y consultar productos.

`ProductoEvaluado`. Esta clase hereda de producto sus atributos y agrega uno extra, a saber: evaluación, el cual se utiliza para guardar la evaluación del producto al ser comparado contra la ponderación que da el usuario, esta clase es utilizada por el algoritmo, donde se devuelve una lista de productos con la calificación que da el algoritmo, y se utiliza para dar un orden a la lista que el servidor regresará al consumidor.

`Ponderacion`. Se utiliza para recibir una lista de ponderaciones del usuario, por cada característica se utiliza un objeto tipo ponderación, de modo que el algoritmo recibe una lista de ponderaciones, se hizo de esta forma para facilitar la parte en la que el algoritmo forma el árbol de decisiones para evaluar las características.

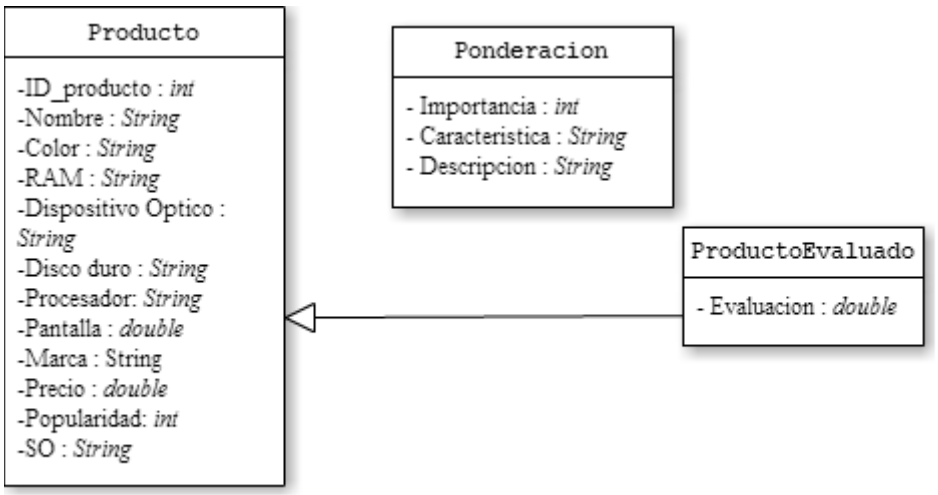

Figura 5. Diagrama de dominio

6.2.2. Diagrama de clases de software de RECOM

La figura 6 muestra el diagrama de clases de software, el cual incluye las clases correspondientes a las entidades identificadas en el modelo de dominio y las clases controladoras del sistema. Las clases controladoras son: `Eliminar`, `Editar`, `Agregar`, `Obtener`, `Consultar`, `ComparaPonderacion`, `Ordena-Resultado` y `Algoritmo`. A continuación se explica cada una de las clases controladoras de RECOM.

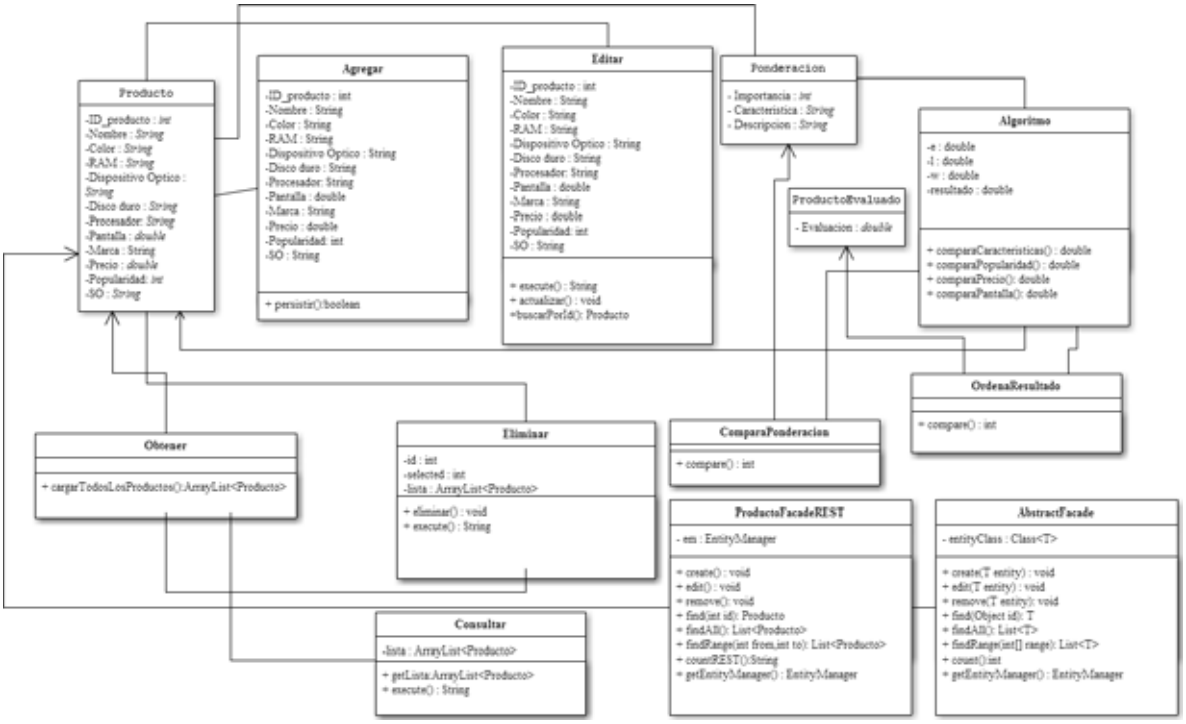

Figura 6. Diagrama de clases

`Algoritmo`. Se utiliza para evaluar la ponderación de los productos; por lo tanto, contiene las operaciones necesarias para usar el algoritmo LSP.

`Eliminar`. Se utiliza para eliminar un equipo de cómputo de la base de datos.

`Editar`. Se utiliza para editar un equipo de cómputo de la base de datos.

`Agregar`. Se utiliza para agregar un equipo de cómputo de la base de datos.

`Obtener`. Se utiliza para obtener un listado de los equipos de cómputo de la base de datos.

`Consultar`. Se utiliza para consultar un equipo de cómputo de la base de datos.

`ComparaPonderacion`. Se utiliza para comparar cada una de las características que el consumidor seleccionó.

`OrdenaResultado`. Se utiliza para ordenar los productos evaluados.

`ProductoFacadeREST`. Se utiliza para programar las operaciones básicas de REST, a saber : PUT, GET, DELETE y POST.

`AbstractFacade`. Esta clase crea los mismos métodos que `ProductoFacadeREST` pero de una manera genérica es que produce los objetos.

6.2.4. Arquitectura del sistema

RECOM está formado por una aplicación web destinada a gestionar los productos de cómputo, un servicio web destinado a proporcionar la lista del equipo de cómputo registrado, y una aplicación cliente que hace las peticiones al servicio web proporciona las recomendaciones, ver figura 7. A continuación se explicará cada uno de ellos.

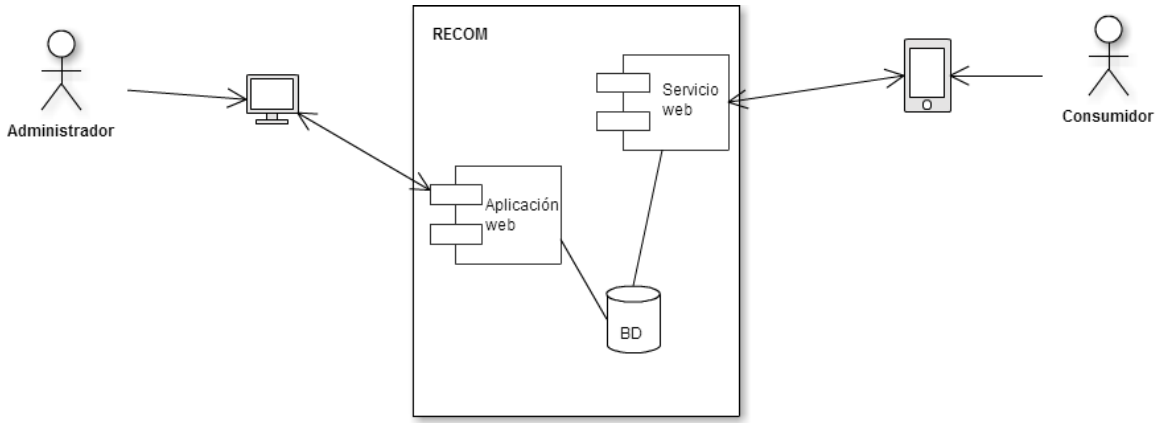

Figura 7. Esquema de la arquitectura general de RECOM

Aplicación web para gestionar los equipos de cómputo.

Esta aplicación web emplea el *framework struts 2*, por lo tanto se implementó en base al modelo MVC (Modelo-Vista-Controlador). Los paquetes que conforman la aplicación se muestran en la figura 8.

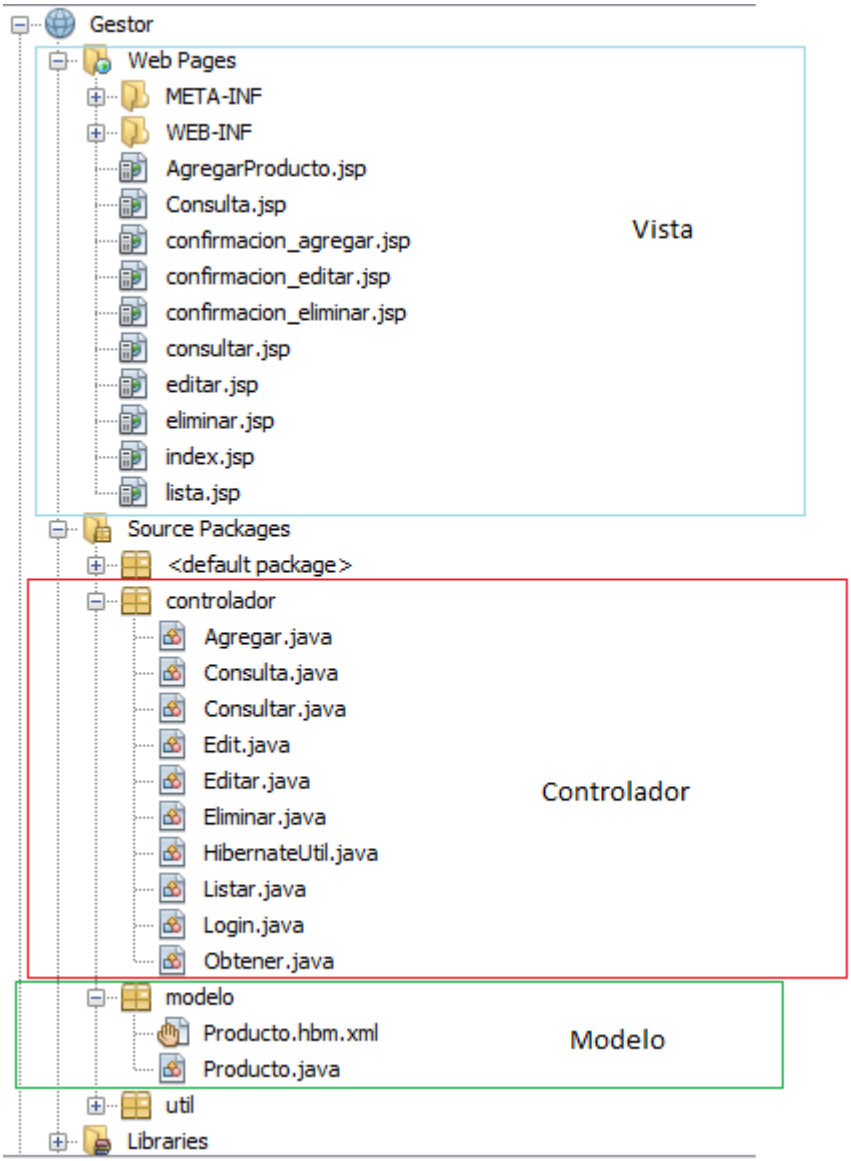

Figura 8. Paquetes de la aplicación web.

Modelo. Formado por dos clases: `Producto`, `Producto.hbm.xml`, las cuales permiten registrar los datos de los equipos de cómputo.

Vista. Está formada por archivos *jsp* que dan origen a páginas web desde las cuales el administrador del sistema puede interactuar con RECOM. Los archivos son: `AgregarProducto`, `Consulta`, `confirmación_agregar`, `confirmación_editar`, `confirmación_eliminar`, `consultar`, `editar`, `eliminar`, `lista` e `index`.

Controlador. Este paquete contiene las clases necesaria para realizar las operaciones CRUD (*Create, Read, Update, Delete*), a saber: `Agregar, consulta, consultar, eliminar, edit, editar, lista, login, obtener, HibernateUtil`.

Servicio web para proponer recomendaciones de equipos de cómputo.

Este servicio web está formado por dos paquetes: *model* y *service* ver figura 9. La comunicación entre el servicio web y la aplicación del cliente (dispositivo móvil) se realiza a través del protocolo REST.

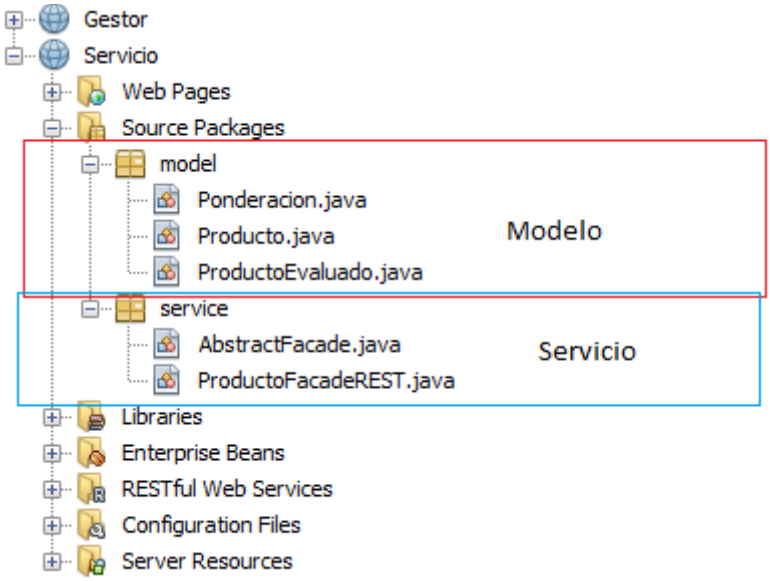

Figura 9. Paquetes del servicio web.

Model. Formado por tres clases, a saber: `Producto, ProductoEvaluado, Ponderacion`, estas clases conforman la capa de datos del servicio.

Service. Formado por dos clases, las cuales realizan las operaciones básicas de REST: PUT, GET, POST y DELETE.

Aplicación cliente

La aplicación cliente, ver figura 10, contiene las clases necesarias para realizar una petición de equipos de cómputo al servicio web y posteriormente aplicar el algoritmo LSP para obtener una lista ordenada de acuerdo a un grado de preferencia dado por un consumidor.

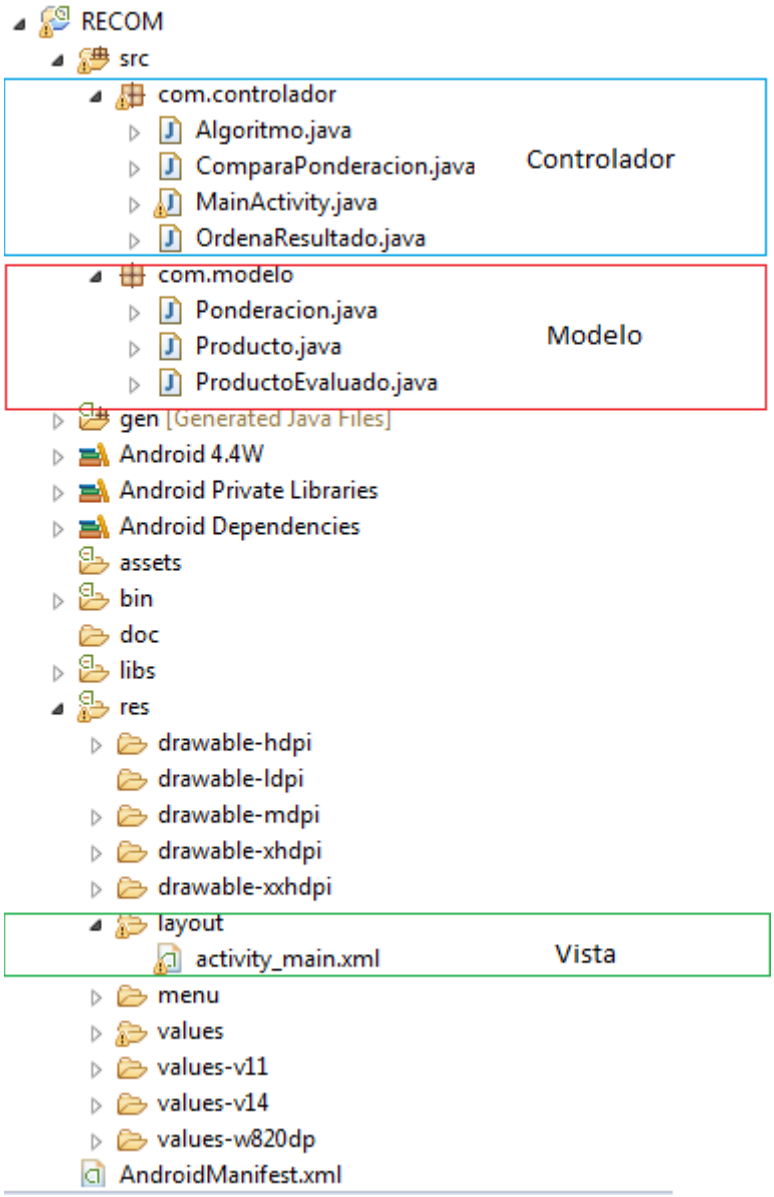

Figura 10. Paquetes de aplicación cliente.

Vista. Contiene un archivo *activity_main.xml* que permite visualizar la interfaz en el dispositivo móvil.

Controlador. Contiene las clases *OrdenaResultado*, *MainActivity*, *ComparaPonderacion* y *Algoritmo*, las cuales son necesarias para aplicar el algoritmo LSP sobre los equipos de cómputo recuperados y obtener la ponderación del consumidor, a través de la petición de los productos al servicio web.

Modelo. Este paquete contiene la capa de datos de la aplicación móvil, la cual se utiliza por el controlador. Contiene las clases *Producto*, *Ponderacion* y *ProductoEvaluado*, las cuales son necesarias para la interacción con la información obtenida del servicio web.

La arquitectura que se ha explicado previamente corresponde con la implementación del sistema RECOM. Sin embargo, es importante señalar que idealmente la aplicación cliente debería tener únicamente los elementos de la vista y un pequeño módulo que recupere la ponderación del usuario (grado de preferencia) sobre las características del equipo de cómputo, para enviarlo al servicio web. De tal manera, que el servicio web se encargue de aplicar el algoritmo LSP, y luego ordenar la lista de equipos de cómputo para posteriormente responder a la aplicación cliente. La premura de tiempo, y la falta de experiencia en la tecnología para servicios web, provocó que no se realizara de esta manera.

6.2.5. Estructura de la base de datos

La base de datos cuenta con una tabla llamada `producto`, la cual contiene los datos de todos los productos disponibles en el servicio web de RECOM, ver la figura 11. La tabla 1 describe los campos que la conforman.

Figura 11. Esquema de la base de datos

Campo	Descripción
ID_producto	Identificador para cada una de los equipos de cómputo, el cual es un número entero único auto-incrementable.
Nombre	Nombre del equipo de cómputo.
Tipo	Tipo de equipo de cómputo, existen 3 tipos, a saber: laptop, notebook y PC
Color	Color del equipo de cómputo, existen varios colores, a saber: Negro, blanco, rojo, plateado, gris, azul, rosa y morado.
Marca	Símbolo del equipo de cómputo, que nos indica cual es la compañía que construyó esta misma. Solo se manejan 4 marcas, a saber: Acer, HP, Sony y Dell.
SO	Sistema Operativo del equipo de cómputo en cuestión; se manejan 5 sistemas operativos, a saber: Windows 7 Home Basic, Windows 7 Starter, Free DOS, Windows 8 y Windows 8 Pro.
RAM	Memoria de trabajo para el sistema operativo; se tienen varios

	tamaños, desde 2 GB hasta 16 GB.
Disco Duro	Memoria que contiene el equipo de cómputo para guardar información; se manejan diferentes tamaños, desde 250 GB hasta 1 TB
Dispositivo Óptico	Dispositivo de almacenamiento óptico; se manejan 2 tipos: Reproductor/quemador de DVD y no tiene.
Procesador	Procesador del equipo de cómputo, se manejan diferentes tipos que no será enlistados aquí por tratarse de una lista exhaustiva; la lista de procesadores se encuentra en el Apéndice A.
Pantalla	Tamaño de la pantalla del equipo de cómputo, se maneja en pulgadas y hay de diferentes tamaños, entre las 10" y las 16".
Precio	Monto que debe pagarse por un equipo de cómputo, se tienen diferentes precios, desde $4,999 hasta $50,000.
Popularidad	La popularidad de un equipo de cómputo se determina a través de estrellas, donde 5 estrellas la define como muy popular, y 1 estrella la define como poco popular. Las estrellas son asignadas por el consumidor, en un esquema de voto, a partir del grado de satisfacción que le proporcionó el producto.

Tabla 2.Características a evaluar en los equipo de cómputo

6.3 Uso del sistema.

El uso de RECOM se explicará a partir de los casos de uso: Solicitar recomendación de un producto y gestionar productos.

Solicitar recomendación de un producto.

RECOM opera desde un dispositivo móvil. Para iniciar la aplicación damos clic en el ícono RECOM, como se puede observar en la figura 12.

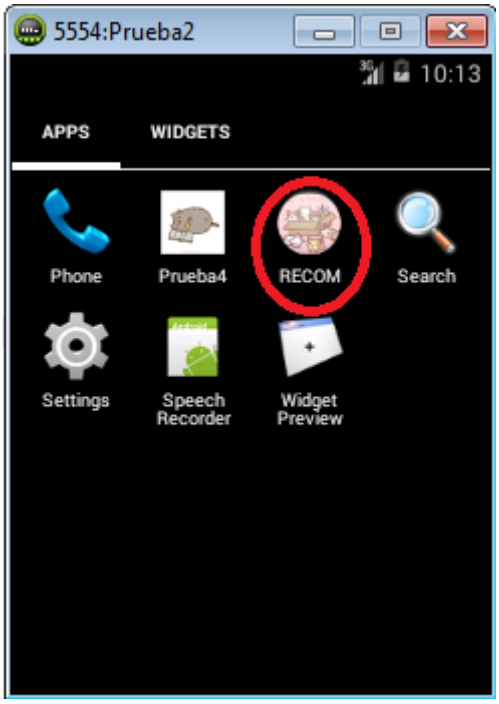

Figura 12. Icono de RECOM

Al ingresar se muestra una lista de las características de estos equipos de cómputo, a saber: color, disco duro, marca, RAM, popularidad, procesador, sistema operativo, tamaño de pantalla, tipo de equipo, precio y dispositivo óptico ver Figura 13.

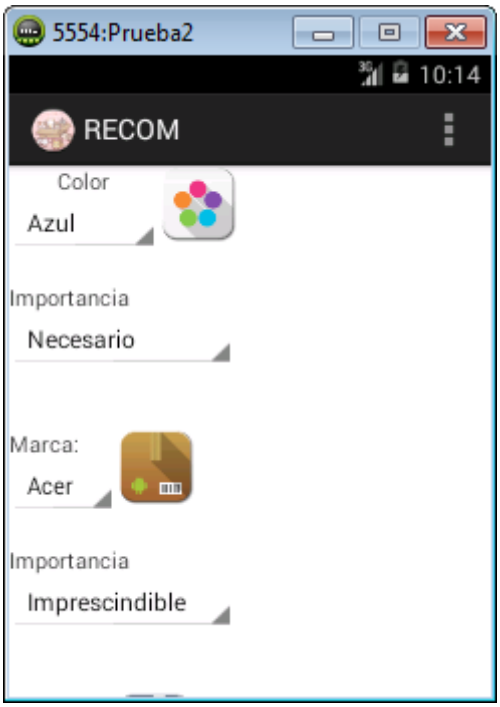

Figura 13. Pantalla principal de RECOM

Debajo de cada característica, se debe marcar el grado de importancia que para el consumidor tiene esa característica, a saber: Imprescindible, Necesario, Me gustaría, No necesario, y Prescindible. Popularidad se maneja como número, en un rango de (1 a 10) porque es equivalente al voto que un consumidor asigna a un equipo de cómputo.

Una vez realizado esto da clic en buscar como se puede ver en la figura 14.

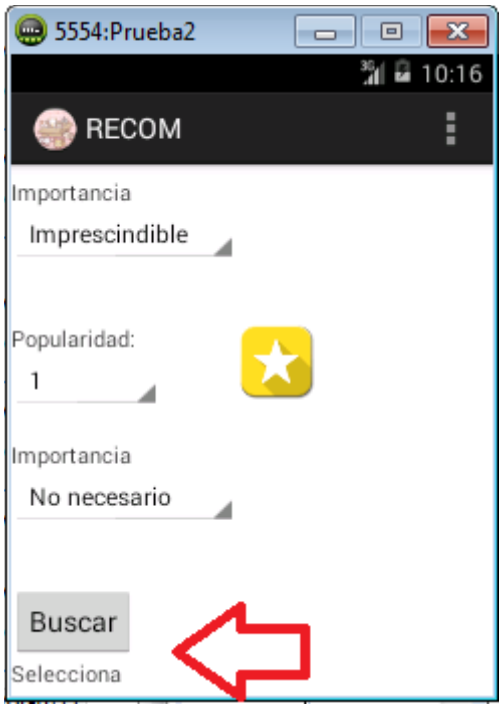

Figura 14. Botón buscar de RECOM

Cuando el consumidor ha terminado de marcar sus preferencias y da clic en buscar, la aplicación cliente (del dispositivo móvil) envía una petición al servicio Web para recuperar la lista de los equipos de cómputo disponibles, a través del protocolo REST. Una vez obtenida la lista, la aplicación cliente aplica el algoritmo LSP para obtener una lista de aquellos equipos de cómputo que más se aproximan a las preferencias del consumidor. Al consumidor se presenta la lista de los equipos por prioridad: desde los que más se aproximan a las características requeridas por el consumidor hasta los que menos se acercan.

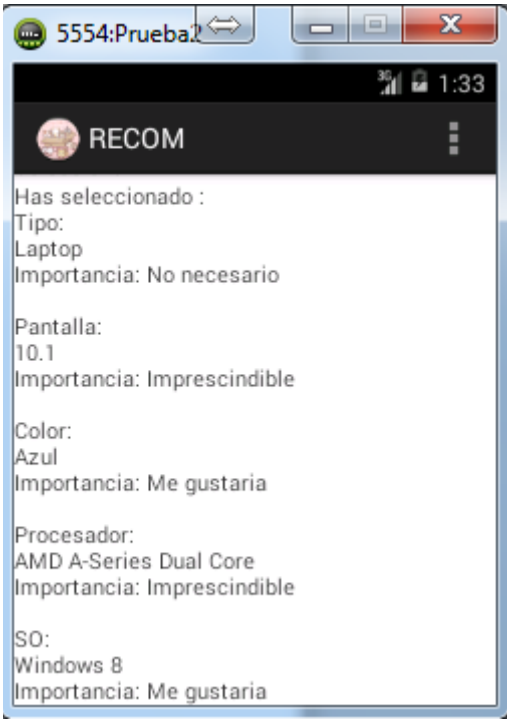

Figura 15. Ponderación del consumidor (primera parte)

Figura 16. Ponderación del consumidor (segunda parte)

En las figuras 15 y 16 se puede observar la ponderación realizada por el consumidor sobre las características del equipo de cómputo. En la figura 19 y 20, se pueden observar los

resultados del algoritmo, cada equipo de cómputo aparece separado de los demás por una línea. El formato en el que aparece es el siguiente: i) Nombre del equipo de cómputo, ii) marca y iii) color.

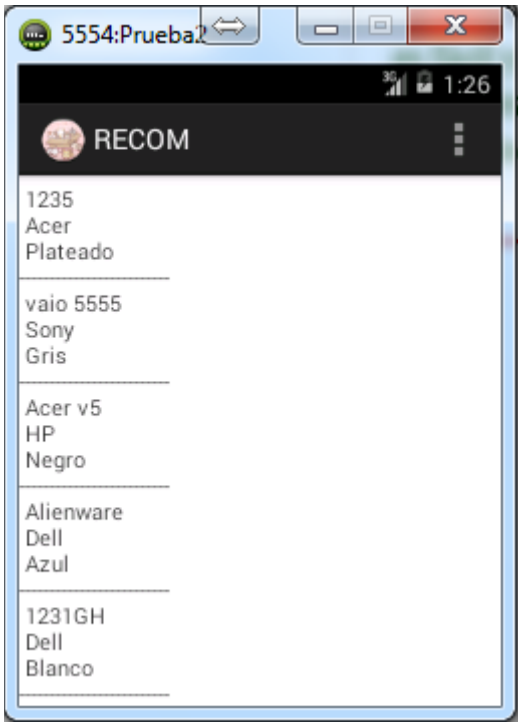

Figura 17. Resultados que se acercan más a la ponderación del consumidor.

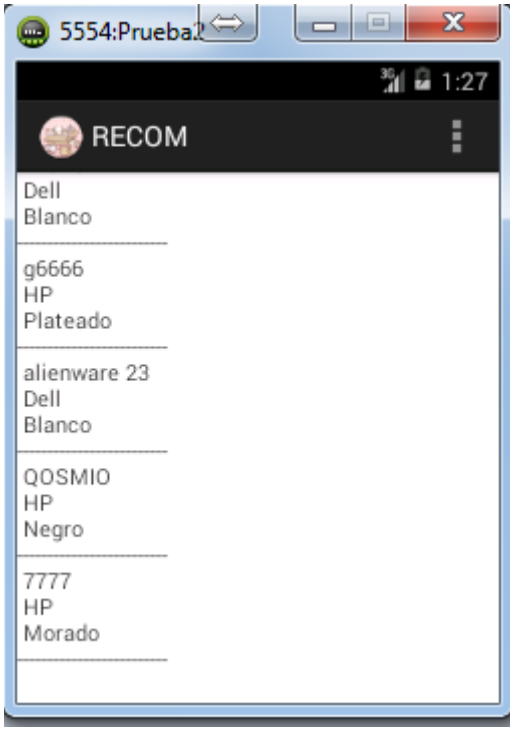

Figura 18. Resultados que menos se acercan a la ponderación del consumidor.

Los equipos de cómputo se presentan en orden de prioridad, la primera lista (Figura 17) favorece a las computadoras de marca Acer, porque esta es una característica impresindible por el consumidor. Mientras que las computadoras de la segunda lista (Figura 18) muestra otras marcas pero con características que se aproximan a las preferencias, a saber: la memoria RAM de 2GB, precio máximo $5000, dispositivo óptico, entre otros.

Gestionar productos: Editar.

RECOM cuenta con una aplicación web destinada al administrador del sistema. Esta aplicación web permite editar, modificar, eliminar y consultar los equipos de cómputo, que después utiliza el servicio web para hacer recomendaciones. A continuación se explica la operación de editar.

La pantalla principal se muestra en la figura 19, formada por la lista de los equipos de cómputo y sus características registrados hasta este momento. Delante de cada registro se observan dos ligas: una para eliminarlo y la otra para editarlo.

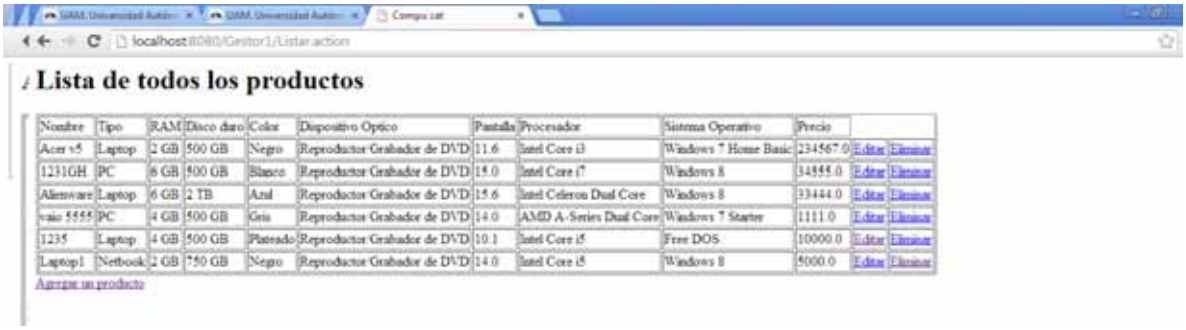

Figura 19. Pantalla inicial para el administrador.

Al escoger la opción de editar para el registro de la computadora "Laptop1", aparece la lista de las características del equipo en modo de edición, ver Figura 20. El administrador puede modificar los campos que sean necesarios, para luego presionar el botón de guardar cambios. Para este ejemplo se modificó únicamente el campo de color, de negro a azul.

Figura 20. Pantalla de edición de un equipo de cómputo.

En la Figura 21 se observa el cambio cuando regresamos a la pantalla principal, en rojo se señala el cambio realizado a "Laptop1".

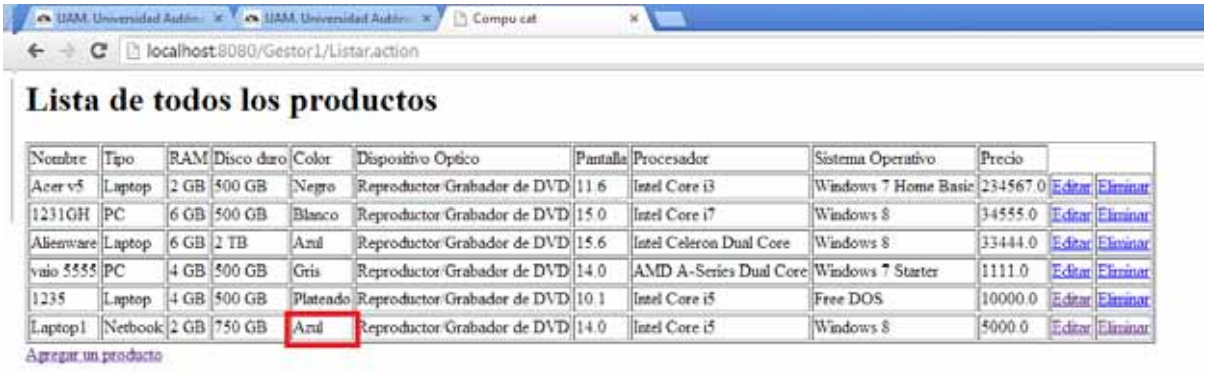

Figura 21. Pantalla de lista de equipos de cómputo actualizados.

6.4. Hardware y software necesarios

6.4.1 Software

A continuación se exponen brevemente las distintas herramientas que se utilizaron para el desarrollo de la aplicación RECOM.

6.4.1.1 Tecnología para el desarrollo de RECOM

Eclipse ADT [10]. Este ADT de eclipse incluye todo lo necesario para empezar a desarrollar
aplicaciones para Android, a saber:

- Eclipse + ADT plugin.
- Android SDK.
- Herramientas para la plataforma Android.
- Una imagen ISO del sistema operativo Android para el emulador.

Netbeans 7.1 [15] es un entorno de desarrollo por medio del cual se pueden crear aplicaciones web y aplicaciones de escritorio.

6.4.1.2 Tecnología para la instalación y puesta en marcha de RECOM

Apache Tomcat 6 [14]. Apache Tomcat es un servidor de aplicaciones web de código abierto que implementa Java Servlets y páginas web con tecnología JavaServer. Se optó por la versión 6, porque implementa las especificaciones de Servlet 3.0, el cual es indispensable para definir el Servicio Web (SW) con tecnología REST.

MySQL 5 [13]. MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario. Se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia.

Glassfish 3.1.2 [12] es un servidor de aplicaciones web de código abierto. Se optó por la versión 3.1.2 porque soporta las operaciones de REST.

Java JDK 7 [11] Es una herramienta que permite desarrollar y desplegar aplicaciones Java en equipos de escritorio y servidores. Java ofrece rendimiento, versatilidad y portabilidad.

6.4.2 Hardware

Para el desarrollo de RECOM se empleó una computadora de escritorio con las siguientes especificaciones:

- Un procesador de 1.40 GHz
- 4 GB de memoria RAM.
- Disco duro de 500 GB.
- Un monitor de 14 pulgadas.

7. Resultados

En este proyecto de integración se desarrolló la aplicación RECOM que compuesto por un módulo para gestionar los productos (equipos de cómputo) de una empresa y un servicio web para proponer recomendaciones de esos mismos productos; estos dos módulos atienden a los objetivos particulares mencionados en la sección 4.2.

El módulo gestionar los productos permite a un administrador de los productos (equipos de cómputo): dar de alta, baja, editar y consultar los productos que tienen a la venta. Este módulo de RECOM se diseñó e implementó como una aplicación web.

Como parte del módulo proponer recomendaciones de los productos, se implementó el algoritmo LSP en una aplicación cliente (para dispositivo móvil). El consumidor proporciona una ponderación en cada una de las características que desea de su producto, en la aplicación cliente; posteriormente ésta se comunica con el servicio web para solicitar la lista de los equipos de cómputo disponibles. Una vez obtenida la lista, la aplicación cliente aplica el algoritmo LSP para obtener una lista de aquellos equipos de cómputo que más se aproximan a las preferencias del consumidor. Al consumidor se presenta la lista de los equipos por prioridad: desde los que más se aproximan a las características requeridas por el consumidor hasta los que menos se acercan.

Se diseñó e implementó una base de datos que contiene los productos que vende la empresa, desde esta base de datos se pueden realizar todas las consultas que hace tanto la aplicación web para la gestión, como el servicio web para realizar la recomendación de equipos de cómputo.

8. Análisis y discusión de resultados

Soluciones

La aplicación RECOM desarrollado en este proyecto permite tanto proponer recomendaciones de productos a un consumidor, como gestionar los productos a un administrador.

La aplicación web que permite gestionar los productos fue diseñada empleando el patrón MVC, algunas ventajas fueron: i) RECOM fue implementado modularmente, y ii) sus vistas muestran en todo momento información actualizada.

La aplicación web, fue realizada con *struts* 2 para seguir el patrón MVC, en la vista se tienen las páginas web, en el controlador se tienen las clases que controlan la funcionalidad de las páginas web y el modelo se opera la capa de datos. Se utilizó *Hibernate*, el cual realiza el mapeo objeto-relacional de los datos.

El servicio Web que permite recuperar la lista de equipos de cómputo disponible fue realizado con el protocolo *REST*, el cual es un protocolo sencillo y eficiente, el intercambio de información se hace mediante *JSON* que hace una transferencia ligera de datos. La aplicación móvil recibe los datos que *JSON* provee y los muestra de una forma entendible y adecuada para los consumidores.

Cada módulo que conforma a la aplicación RECOM fue diseñado, implementado y sometido a pruebas unitarias. Posteriormente, los módulos fueron integrados y sometidos a pruebas de sistema. En las pruebas unitarias, se probaron por separado las operaciones CRUD en la aplicación web. Para probar el servicio web se construyó un cliente provisional para probar la correcta obtención de los equipos de cómputo. En la aplicación cliente se realizaron diversas combinaciones de ponderaciones para comprobar el resultado obtenido por el algoritmo LSP.

A nivel de las pruebas de sistema se probó la integración de los módulos de RECOM. En la aplicación web, se hicieron varias pruebas que consistieron en agregar productos, editarlos, eliminarlos; en cada operación se hicieron varias consultas de la lista de productos para validar que efectivamente las operaciones se llevaron a cabo con éxito. En la aplicación cliente y el servicio web se hicieron varias ejecuciones para observar la respuesta del servicio web, con respecto a la lista del equipo de cómputo, para luego observar el comportamiento del algoritmo LSP en aplicación móvil.

Limitaciones

En el diseño inicial, el servicio web debía contener el algoritmo LSP, pero por falta de tiempo y falta de experiencia en la tecnología de servicios web, el algoritmo se encuentra

en la aplicación cliente, y el servicio web únicamente proporciona la lista de productos que se encuentran en la base de datos.

La aplicación cliente muestra los resultados del algoritmo LSP en un *label* lo cual no es adecuado para correcta visualización de la información en el dispositivo móvil. Es conveniente mostrarlos en un *listview* pero tampoco fue posible completar este aspecto.

El módulo que permitiría a un consumidor emitir un voto sobre el producto de la elección del consumidor, no se alcanzó; se trata de una funcionalidad donde el consumidor pueda dar una calificación (por ejemplo de cero a diez). Esta calificación servirá para designar la popularidad del producto, y será un elemento más para afinar la recomendación de los equipo de cómputo que RECOM pueda ofrecer.

9. Conclusiones

RECOM es una aplicación para recomendaciones de compras en equipos de cómputo, las recomendaciones se dan por medio de una ponderación que hacen los consumidores, esta ponderación es evaluada por el algoritmo LSP y arroja las recomendaciones de compra. RECOM está compuesto por los siguientes módulos: 1) Gestión de productos; 2) Recomendación de equipo de cómputo. El primero opera en una aplicación web y el segundo opera en una aplicación móvil y un servicio web.

La aplicación web está basada en dos tecnologías: *Hibernate* y *struts2*; *hibernate* se utiliza para realizar la comunicación de la aplicación web con la base de datos; por otro lado *struts2*, es la tecnología que se encarga del manejo de la arquitectura modelo-vista-controlador de la aplicación web.

El servicio web está basado en la arquitectura REST; el consumidor lo opera desde una aplicación cliente (que se ejecuta en un dispositivo móvil), la cual solicita a un servidor remoto la lista del equipo de cómputo disponible.

La aplicación cliente fue implementado para dispositivos móviles con sistema operativo *Android*. Esta aplicación interactúa con el consumidor, y se comunica con el servicio web a través del protocolo REST.

El proyecto fue un reto, principalmente porque involucró varias tecnologías, a saber: *Struts2, Hibernate, REST* y *Android*, las cuales debían ser estudiadas a medida que se avanzaba en el diseño del sistema.

El Proceso Unificado marcó las fases de todo el desarrollo: primero se estudió el algoritmo de lógica de puntuación de preferencias. Se estudió la tecnología para desarrollar el proyecto; se instaló la tecnología: *plugins*, bibliotecas. Se inició la programación del

algoritmo de lógica de puntuación de preferencias. Posteriormente se diseñó e implementó la programación de la aplicación web para el módulo de gestión de productos; finalmente, se diseño y se implementó el servicio web para el módulo de recomendaciones de productos; además, se implementó la aplicación cliente sobre un dispositivo móvil con sistema operativo Android.

A continuación se menciona cada uno de los objetivos específicos y su grado de completud.

- Adaptar e implementar el algoritmo heurístico de lógica de puntuación de preferencias (LSP) en el proceso de recomendación de productos que vende una empresa de computadoras.

  Este objetivo se cumplió cabalmente, el algoritmo LSP forma parte de la aplicación cliente para un dispositivo móvil con sistema operativo Android.

- Diseñar e implementar un módulo que permita dar de alta, baja, modificar y consultar productos.

  Todas las secciones de este módulo se efectúan correctamente, el objetivo se cumplió en su totalidad.

- Diseñar e implementar una base de datos que contenga la información de los productos a vender.

  Este objetivo se cumplió cabalmente, se implementó la base de datos para RECOM y cuenta con los datos para algunos equipos de cómputo. Esta base de datos es empleada tanto por la aplicación web como por el servicio web.

- Diseñar e implementar un módulo que permita a un consumidor emitir un voto sobre el producto de la elección del consumidor.

  Este objetivo no se cumplió, el consumidor no puede emitir un voto sobre los equipos de cómputo.

10. Perspectivas del proyecto

RECOM puede ser mejorada en los módulos actuales y puede ser extendida a nuevas funcionalidades.

En el módulo de la aplicación cliente, se puede implementar la funcionalidad para recuperar el voto del consumidor sobre los equipos de cómputo.

En el módulo del servicio web debería implementarse el proceso de recomendación, a través del algoritmo LSP, y no solo la consulta de los equipos. Actualmente, es la aplicación cliente la encargada de aplicar el algoritmo LSP.

En el módulo de la aplicación móvil, podría realizarse una mejora visual, para que el consumidor interactúe con la aplicación de una mejor manera. Por ejemplo, agregar una descripción más detallada sobre el equipo de cómputo.

RECOM podría tener una aplicación cliente para computadoras personales. También podría ampliarse la gama de productos que se recomiendan.

Referencias bibliográficas

[1] http://www.mercadolibre.com. Fecha de consulta: 24 de marzo de 2013, hora: 03:09 P M.
[2] http://www.amazon.com. Fecha de consulta: 24 de marzo de 2013, hora: 05:45 P M.
[3] http:// www.ebay.com. Fecha de consulta: 24 de marzo de 2013, hora: 09:32 P M.
[4] Herlocker, J. L, *Evaluating collaborative filtering recommender systems*. ACM
[5] J. P. Alcántara Olivares y J. D. López Jaimes, "Aplicación Colaborativa para
[6]S. E. Avendaño Méndez, "Gestión de calificaciones de cursos mediante servicios Web.", Proyecto terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México, 2012.
[7] D. Gutiérrez Trevilla,"Interfaz para la administración en línea de cursos presenciales.", Proyecto terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México, 2010.
[8]http://www.dell.com.mx. Fecha de consulta: 23 de marzo de 2013, hora: 10:03 A M.
Transactions Information System, vol. 22., pp. 5-53, enero 2004.
Dispositivos Móviles con Sistema Operativo Android.", Proyecto terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México, 2012.
[9] http:// age.ieg.csic.es/metodos/2010_Sevilla/ponencia2/BUSTOS.pdf
[10] J.J. Dujmovic, /A Method for Evaluation and Selection of Complex //Hardware and Software Systems/, The 22nd International Conference for the Resource Management and Performance Evaluation of Enterprise CS. CMG, 96 Proceedings, Vol. 1, pp.368-378, 1996.
[11]http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/. Fecha de consulta: 8 de abril de 2013, hora: 02:00 P M.
[12] https://eclipse.org/ /. Fecha de consulta: 7 de enero de 2015, hora: 02:00 A M.
[13]https://netbeans.org//. Fecha de consulta: 7 de enero de 2015, hora: 09:40 A M.
[14]http://tomcat.apache.org/download-60.cgi/. Fecha de consulta: 7 de enero de 2015, hora: 09:39 A M.
[15]http://www.mysql.com//. Fecha de consulta: 7 de enero de 2015, hora: 09:35 A M.
[16]https://glassfish.java.net/es/. Fecha de consulta: 7 de enero de 2015, hora: 09:33 A M.
[17]http://www.oracle.com/technetwork/java/javase/overview/index.html. Fecha de consulta: 7 de enero de 2015, hora: 09:30 A M.

Apéndice A. Listado de procesadores

- AMD E2-1800
- Intel Core i3
- Intel Core i5
- Intel Core i7
- Intel Celeron Dual Core
- AMD A-Series Dual Core
- Intel Atom N2600
- AMD E2-3200
- Intel Atom D425
- AMD Fx-4100
- AMD E-300 Dual Core
- Intel Pentium G2020
- Intel Celeron G465

Apéndice B. Entregable: Listado del API del código fuente desarrollado

Aplicación web

Paquete controlador

Agregar

```
package controlador;

import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.Preparable;
import com.opensymphony.xwork2.validator.annotations.*;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;
import modelo.Producto;
import org.hibernate.Session;
import org.hibernate.Transaction;

/**
 * Esta clase sirve para agregar productos a la lista de equipos de computo.
 * @author Karina Guzmán Villanueva
 */
public class Agregar extends ActionSupport implements Preparable{

    private String nombre;
    private String color;
    private String ram;
```

```
    private String discoduro;
    private String dispositivooptico;
    private double precio;
    private double pantalla;
    private String marca;
    private String tipo;
    private String procesador;
    private String so;
    private int popularidad;



    /**
     * Constructor de la clase.
     */
    public Agregar() {

    }

    /**
     * Obtiene la popularidad del equipo de computo en cuestion.
     * @return popularidad del equipo.
     */
    public int getPopularidad() {
        return popularidad;
    }

    /**
     * Define la popularidad del equipo de computo en cuestion.
     * @param popularidad Este parametro es el que se pondra como valor en la
popularidad.
     */
    public void setPopularidad(int popularidad) {
        this.popularidad = popularidad;
    }


    /**
     * Obtiene el nombre del equipo de computo en cuestion.
     * @return nombre del equipo.
     */
    public String getNombre() {
        return nombre;
    }

    /**
     * Obtiene el color del equipo de computo en cuestion.
```

```
     * @return color del equipo.
     */
    public String getColor() {
        return color;
    }

    /**
     * Obtiene la ram del equipo de computo en cuestion.
     * @return ram del equipo.
     */
    public String getRam() {
        return ram;
    }

    /**
     * Obtiene el disco duro del equipo de computo en cuestion.
     * @return disco duro del equipo.
     */
    public String getDiscoduro() {
        return discoduro;
    }

    /**
     * Obtiene el dispositivo optico del equipo de computo en cuestion.
     * @return dispositivo optico del equipo.
     */
    public String getDispositivooptico() {
        return dispositivooptico;
    }

    /**
     * Obtiene el precio del equipo de computo en cuestion.
     * @return precio del equipo.
     */
    public double getPrecio() {
        return precio;
    }

    /**
     * Obtiene la pantalla del equipo de computo en cuestion.
     * @return pantalla del equipo.
     */
    public double getPantalla() {
        return pantalla;
    }

    /**
```

```
     * Obtiene el sistema operativo del equipo de computo en cuestion.
     * @return sistema operativo del equipo.
     */
    public String getSo() {
        return so;
    }

    /**
     * Obtiene la marca del equipo de computo en cuestion.
     * @return marca del equipo.
     */
    public String getMarca() {
        return marca;
    }

    /**
     * Obtiene el tipo del equipo de computo en cuestion.
     * @return tipo del equipo.
     */
    public String getTipo() {
        return tipo;
    }

    /**
     * Obtiene el procesador del equipo de computo en cuestion.
     * @return procesador del equipo.
     */
    public String getProcesador() {
        return procesador;
    }


    /**
     * Define el nombre del equipo de computo en cuestion.
     * @param nombre Este parametro es el que se pondra como valor en el nombre.
     */
    @RequiredStringValidator(message="El nombre es obligatario", trim = true)
    @StringLengthFieldValidator(message="El nombre debe contener al  menos 3
caracteres", trim=true, minLength="3")
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    /**
     * Define el color del equipo de computo en cuestion.
     * @param color Este parametro es el que se pondra como valor en el color.
     */
    @StringLengthFieldValidator(message="Debe seleccionar un color", trim=true,
```

```
minLength="3")
        public void setColor(String color) {
        this.color = color;
    }

    /**
     * Define el sistema operativo del equipo de computo en cuestion.
     * @param so Este parametro es el que se pondra como valor en el sistema operativo.
     */
    @StringLengthFieldValidator(message="Debe seleccionar el sistema operativo",
trim=true, minLength="3")
    public void setSo(String so) {
        this.so = so;
    }
    /**
     *  Define la ram del equipo de computo en cuestion.
     * @param ram Este parametro es el que se pondra como valor en la ram.
     */
    @StringLengthFieldValidator(message="Debe seleccionar el tamaño de la RAM",
trim=true, minLength="3")
    public void setRam(String ram) {
        this.ram = ram;
    }


    /**
     * Define el tipo del equipo de computo en cuestion.
     * @param tipo Este parametro es el que se pondra como valor en el tipo.
     */
    @StringLengthFieldValidator(message="Debe seleccionar el tipo de computadora",
trim=true, minLength="3")
    public void setTipo(String tipo) {
        this.tipo = tipo;
    }

    /**
     * Define el disco duro del equipo de computo en cuestion.
     * @param discoduro Este parametro es el que se pondra como valor en el disco
duro.
     */
    @StringLengthFieldValidator(message="Debe seleccionar el tamaño deldisco duro",
trim=true, minLength="3")
    public void setDiscoduro(String discoduro) {
        this.discoduro = discoduro;
    }

    /**
```

```
     * Define el dispositivo optico del equipo de computo en cuestion.
     * @param dispositivooptico Este parametro es el que se pondra como valor en el
dispositivo optico.
     */
    @StringLengthFieldValidator(message="Debe seleccionar el tipo de dispositivo
optico", trim=true, minLength="3")
    public void setDispositivooptico(String dispositivooptico) {
        this.dispositivooptico = dispositivooptico;
    }

    /**
     * Define el procesador del equipo de computo en cuestion.
     * @param procesador Este parametro es el que se pondra como valor en el
procesador.
     */
    @StringLengthFieldValidator(message="Debe seleccionar el procesador", trim=true,
minLength="3")
    public void setProcesador(String procesador) {
        this.procesador = procesador;
    }
    /**
     * Define el precio del equipo de computo en cuestion.
     * @param precio Este parametro es el que se pondra como valor en el precio.
     */
    @DoubleRangeFieldValidator(message="El precio es obligatorio", minInclusive =
"999.99")
    public void setPrecio(double precio) {
        this.precio = precio;
    }

    /**
     * Define la pantalla del equipo de computo en cuestion.
     * @param pantalla Este parametro es el que se pondra como valor en la pantalla.
     */
    @DoubleRangeFieldValidator(message="Seleccione el tamaño de pantalla",
minInclusive =  "9.9")
    public void setPantalla(double pantalla) {
        this.pantalla = pantalla;
    }

    /**
     * Define la marca del equipo de computo en cuestion.
     * @param marca Este parametro es el que se pondra como valor en la marca.
     */
    @StringLengthFieldValidator(message="Debe seleccionar la marca", trim=true,
minLength="3")
    public void setMarca(String marca) {
```

```
        this.marca = marca;
    }

    /**
     * Este metodo ejecuta la clase.
     * @return Regresa el resultado de la ejecucion.
     */
    @Override
    public String execute() {
        boolean guardado = persistir();
        if (guardado) {
            return SUCCESS;
        } else {
            return INPUT;
        }
    }

    /**
     * Este metodo guarda el nuevo equipo de computo.
     * @return Regresa el resultado de la ejecucion.
     */
    private boolean persistir() {
        Producto lista = new
Producto(nombre,color,ram,discoduro,dispositivooptico,precio,pantalla,marca,tipo,proc
esador,so,popularidad);
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();

        Serializable result = session.save(lista);
        tx.commit();
        session.close();
        return result != null;


    }

    /**
     * Metodo para preparar la transaccion.
     * @throws Exception Se lanza esta excepcion si no se puede ejecutar el metodo.
     */
    @Override
    public void prepare() throws Exception {


    }
}
```

Edit

```
package controlador;


import static com.opensymphony.xwork2.Action.INPUT;
import static com.opensymphony.xwork2.Action.SUCCESS;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.validator.annotations.RequiredFieldValidator;
import com.opensymphony.xwork2.validator.annotations.RequiredStringValidator;
import com.opensymphony.xwork2.validator.annotations.StringLengthFieldValidator;
import modelo.Producto;

import org.apache.struts2.ServletActionContext;
import org.hibernate.Session;
import org.hibernate.Transaction;

/**
 * Esta clase sirve para editar productos de la lista de equipos de computo.
 * @author Karina Guzmán Villanueva
 */
public class Edit extends ActionSupport{
    private Session session;

    private int id;
    private Producto lista;
    private String tipo;
    private String color;
    private String nombre;
    private String ram;
    private String discoduro;
    private String dispositivooptico;
    private String marca;
    private double pantalla;
    private String procesador;
    private String so;
    private double precio;
    private int popularidad;

    /**
     * Constructor de la clase.
     */
    public Edit() {
        session = HibernateUtil.getSessionFactory().openSession();
        id = Integer.parseInt(ServletActionContext.getRequest().getParameter("id"));
        lista = buscarPorId();
    }

    /**
```

```
 * Obtiene el id de un equipo de computo.
 * @return id de un equipo.
 */
public int getId()
{
   return id;
}

/**
 * Obtiene la lista de equipos de computo.
 * @return lista de equipos.
 */
public Producto getLista(){
   return lista;
}

/**
 * Obtiene el nombre del equipo de computo en cuestion.
 * @return nombre del equipo.
 */
public String getNombre(){
   return nombre;
}

/**
 * Obtiene el color del equipo de computo en cuestion.
 * @return color del equipo.
 */
public String getColor(){
   return color;
}
/**
 * Obtiene la ram del equipo de computo en cuestion.
 * @return ram del equipo.
 */
public String getRam(){
   return ram;
}
/**
 * Obtiene el tipo del equipo de computo en cuestion.
 * @return tipo del equipo.
 */
public String getTipo(){
   return tipo;
}
/**
 * Obtiene el disco duro del equipo de computo en cuestion.
```

```
     * @return disco duro del equipo.
     */
    public String getHDD(){
        return discoduro;
    }
    /**
     * Obtiene el dispositivo optico del equipo de computo en cuestion.
     * @return dispositivo optico del equipo.
     */
    public String getDispositivoOptico(){
        return dispositivooptico;
    }
    /**
     * Obtiene la marca del equipo de computo en cuestion.
     * @return marca del equipo.
     */
    public String getMarca(){
        return marca;
    }
    /**
     * Obtiene la pantalla del equipo de computo en cuestion.
     * @return pantalla del equipo.
     */
    public double getPantalla(){
        return pantalla;
    }
    /**
     * Obtiene el procesador del equipo de computo en cuestion.
     * @return procesador del equipo.
     */
    public String getProcesador(){
        return procesador;
    }
    /**
     * Obtiene el sistema operativo del equipo de computo en cuestion.
     * @return sistema operativo del equipo.
     */
    public String getOs(){
        return so;
    }

    /**
     * Obtiene el precio del equipo de computo en cuestion.
     * @return precio del equipo.
     */
    public double getPrecio(){
        return precio;
```

```
    }

    /**
     * Obtiene la popularidad del equipo de computo en cuestion.
     * @return popularidad del equipo.
     */
    public int getPopularidad(){
        return popularidad;
    }


    /**
     * Define el nombre del equipo de computo en cuestion.
     * @param nombre Este parametro es el que se pondra como valor en el nombre.
     */
    @RequiredStringValidator(message="El nombre es obligatario", trim = true)
    @StringLengthFieldValidator(message="El nombre debe contener al  menos 3
caracteres", trim=true, minLength="3")
    public void setNombre(String nombre){
        this.nombre = nombre;
    }
    /**
     * Define el color del equipo de computo en cuestion.
     * @param color Este parametro es el que se pondra como valor en el color.
     */
    public void setColor(String color){
        this.color = color;
    }

    /**
     * Define el precio del equipo de computo en cuestion.
     * @param precio Este parametro es el que se pondra como valor en el precio.
     */
    public void setPrecio(double precio){
        this.precio = precio;
    }
    /**
     *  Define la ram del equipo de computo en cuestion.
     * @param ram Este parametro es el que se pondra como valor en la ram.
     */
    public void setRam(String ram){
        this.ram=ram;
    }
    /**
      * Define el tipo del equipo de computo en cuestion.
     * @param tipo Este parametro es el que se pondra como valor en el tipo.
     */
```

```
    public void setTipo(String tipo){
        this.tipo = tipo;
    }
    /**
     * Define el disco duro del equipo de computo en cuestion.
     * @param discoduro Este parametro es el que se pondra como valor en el disco
duro.
     */
    public void setDiscoduro(String discoduro){
        this.discoduro = discoduro;
    }
    /**
      * Define el dispositivo optico del equipo de computo en cuestion.
     * @param dispositivooptico Este parametro es el que se pondra como valor en el
dispositivo optico.
     */
    public void setDispositivooptico(String dispositivooptico){
        this.dispositivooptico = dispositivooptico;
    }
    /**
     * Define la marca del equipo de computo en cuestion.
     * @param marca Este parametro es el que se pondra como valor en la marca.
     */
    public void setMarca(String marca){
        this.marca = marca;
    }
    /**
     * Define la pantalla del equipo de computo en cuestion.
     * @param p Este parametro es el que se pondra como valor en la pantalla.
     */
    public void setPantalla(double p){
        this.pantalla = p;
    }
    /**
     * Define el procesador del equipo de computo en cuestion.
     * @param procesador Este parametro es el que se pondra como valor en el
procesador.
     */
    public void setProcesador(String procesador){
        this.procesador = procesador;
    }
    /**
      * Define el sistema operativo del equipo de computo en cuestion.
     * @param so Este parametro es el que se pondra como valor en el sistema operativo.
     */
    public void setSo(String so){
        this.so = so;
```

```
    }

    /**
     * Define la popularidad del equipo de computo en cuestion.
     * @param popularidad Este parametro es el que se pondra como valor en la
popularidad.
     */
    public void setPopularidad(int popularidad){
        this.popularidad = popularidad;
    }


    /**
     * Define el id del equipo de computo en cuestion.
     * @param id Este parametro es el que se pondra como valor en el id.
     */
    @RequiredFieldValidator(fieldName = "id")
    public void setId(int id){
        this.id = id;
    }

    /**
     * Este metodo ejecuta la clase.
     * @return Regresa el resultado de la ejecucion.
     */
    @Override
    public String execute(){
        if (id > 0){
            actualizar();
            return SUCCESS;
        }
        else{
            return INPUT;
        }
    }
    /**
     * Este metodo actualiza los datos de un equipo de computo.
     */
    private void actualizar(){
        Transaction tx = session.beginTransaction();
        Producto e = (Producto) session.get(Producto.class, id);
        e.setNombre(nombre);
        e.setColor(color);
        e.setRam(ram);
        e.setTipo(tipo);
        e.setDiscoduro(discoduro);
        e.setDispositivooptico(dispositivooptico);
```

```
        e.setPantalla(pantalla);
        e.setProcesador(procesador);
        e.setSo(so);
        e.setPrecio(precio);
        e.setPopularidad(popularidad);

        session.update(e);
        tx.commit();
        session.flush();

    }
    /**
     * Este metodo buscar un equipo por Id.
     * @return Regresa el equipo de computo con el id soicitado.
     */
    private Producto buscarPorId(){
        Transaction tx = session.beginTransaction();
        Producto e = (Producto) session.get(Producto.class, id);
        tx.commit();
        return e;
    }

}
```

Eliminar

```
package controlador;
import static com.opensymphony.xwork2.Action.INPUT;
import static com.opensymphony.xwork2.Action.SUCCESS;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.validator.annotations.RequiredFieldValidator;
import com.opensymphony.xwork2.validator.annotations.RequiredStringValidator;
import com.opensymphony.xwork2.validator.annotations.StringLengthFieldValidator;
import java.util.ArrayList;
import java.util.List;
import modelo.Producto;
import org.hibernate.Session;
import org.hibernate.Transaction;

/**
 * Esta clase sirve para eliminar productos de la lista de equipos de computo.
 * @author Karina Guzmán Villanueva
 */
public class Eliminar extends ActionSupport{
     private int id;
    private int selected;
    private ArrayList<Producto> lista;
```

```
/**
 * Constructor de la clase.
 */
public Eliminar() {
   selected = 1;
   lista = Obtener.cargarTodosLosProductos();
}

/**
 * Obtiene la lista de equipos de computo.
 * @return lista de equipos.
 */
public ArrayList<Producto> getLista()
{
   return lista;
}

/**
 * Obtiene el equipo seleccionado.
 * @return equipo seleccionado.
 */
public int getSelected()
{
   return selected;
}

/**
 * Obtiene el id de un equipo de computo.
 * @return id de un equipo.
 */
public int getId()
{
   return id;
}

/**
 * Define el id de un equipo de computo.
 * @param id
 */
@RequiredFieldValidator(fieldName = "id")
public void setId(int id){
   this.id = id;
}

/**
  * Este metodo ejecuta la clase.
 * @return Regresa el resultado de la ejecucion.
```

```
     */
    @Override
    public String execute(){

        if (id > 0){
            eliminar();
            return SUCCESS;
        }

        else{
            return INPUT;
        }
    }
    /**
     * Este metodo elimina equipo.
     */
    private void eliminar(){
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction t = session.beginTransaction();
        Producto temp = (Producto) session.get(Producto.class, id);
        session.delete(temp);
        t.commit();
        session.flush();
        session.close();
    }

}
```

HibernateUtil

```
package controlador;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;

/**
 * Hibernate Utility class with a convenient method to get Session Factory
 * object.
 *
 * @author Karina Guzmán Villanueva
 */
public class HibernateUtil {

    private static final SessionFactory sessionFactory;

    static {
        try {
            // Create the SessionFactory from standard (hibernate.cfg.xml)
```

```
            // config file.
            sessionFactory = new
AnnotationConfiguration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            // Log the exception.
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    /**
     * Metodo para crear una sesion de hibernate.
     * @return
     */
    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

Listar

```
package controlador;

import modelo.Producto;
import static com.opensymphony.xwork2.Action.SUCCESS;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.Preparable;
import java.util.ArrayList;

/**
 * Esta clase sirve para eliminar productos de la lista de equipos de computo.
 * @author Karina Guzmán Villanueva
 */
public class Listar  extends ActionSupport{

    private ArrayList<Producto> listas;

    /**
     * Constructor de la clase.
     */
    public Listar(){
        listas = Obtener.cargarTodosLosProductos();
    }

    /**
     * Obtiene la lista de equipos de computo.
     * @return lista de equipos.
     */
```

```
    public ArrayList<Producto> getListas()
    {
        return listas;
    }


    /**
     * Este metodo ejecuta la clase.
     * @return Regresa el resultado de la ejecucion.
     */
    @Override
    public String execute(){
        return SUCCESS;
    }

}
```

Login

```
package controlador;
import com.opensymphony.xwork2.ActionSupport;
import java.util.ArrayList;
import modelo.Producto;

/**
  * Esta clase sirve para iniciar sesion en la aplicacion.
 * @author Karina Guzmán Villanueva
 */

public class Login extends ActionSupport {
    private ArrayList<Producto> listas;

    /**
     * Constructor de la clase.
     */
    public Login(){
        listas = Obtener.cargarTodosLosProductos();
    }

    /**
      * Obtiene la lista de equipos de computo.
     * @return lista de equipos.
     */
    public ArrayList<Producto> getListas()
    {
        return listas;
    }
```

```
    /**
     * Este metodo ejecuta la clase.
     * @return Regresa el resultado de la ejecucion.
     * @throws Exception Lanza esta excepcion si no puede ejecutar este metodo.
     */
    @Override
public String execute() throws Exception {
System.out.println("Validating login");
if(!getUsername().equals("Admin") || !getPassword().equals("Admin")){
addActionError("Nombre de usuario invalido o password invalido");
return ERROR;
}else{
return SUCCESS;
}
}

private String username = null;
/**
* Obtiene el nombre del usuario.
* @return nombre del usuario.
*/
public String getUsername() {
return username;
}
/**
* Define el nombre del usuario.
* @param value nombre del usuario.
*/
public void setUsername(String value) {
username = value;
}

private String password = null;
/**
* Obtiene la contraseña del usuario.
* @return contraseña del usuario.
*/
public String getPassword() {
return password;
}
/**
*  Define el contraseña del usuario.
* @param value contraseña del usuario.
*/
public void setPassword(String value) {
password = value;
}
```

```
}
```

Obtener

```
package controlador;
import java.util.ArrayList;
import java.util.List;
import modelo.Producto;
import org.hibernate.Criteria;
import org.hibernate.Hibernate;
import org.hibernate.Session;
/**
 * Esta clase sirve para obtener todos los equipos de computo.
 * @author Karina Guzmán Villanueva
 */
public class Obtener {
    /**
     * Carga todos los equipos de computo.
     * @return lista de equipos.
     */
    public static ArrayList<Producto> cargarTodosLosProductos(){
        Session session = HibernateUtil.getSessionFactory().openSession();
        Criteria c = session.createCriteria(Producto.class);
        ArrayList<Producto> lista = (ArrayList<Producto>) c.list();
        return lista;}
}
```

Paquete modelo

Producto

```
package modelo;
// Generated 10/01/2015 06:33:33 PM by Hibernate Tools 3.2.1.GA



/**
 * Producto generated by hbm2java
 * @author Karina Guzmán Villanueva
 */

public class Producto  implements java.io.Serializable {


     private Integer idProducto;
     private String nombre;
     private String color;
     private String ram;
     private String discoduro;
     private String dispositivooptico;
```

```
    private double precio;
    private double pantalla;
    private String marca;
    private String tipo;
    private String procesador;
    private String so;
    private int popularidad;

    /**
     * Constructor de la clase.
     */
    public Producto() {
    }

    /**
     * Constructor de la clase.
     * @param nombre nombre del equipo de computo.
     * @param color color del equipo de computo.
     * @param ram ram del equipo de computo.
     * @param discoduro disco duro del equipo de computo.
     * @param dispositivooptico dispositivo optico del equipo de computo.
     * @param precio precio del equipo de computo.
     * @param pantalla pantalla del equipo de computo.
     * @param marca marca del equipo de computo.
     * @param tipo tipo del equipo de computo.
     * @param procesador procesador del equipo de computo.
     * @param so sistema operativo del equipo de computo.
     * @param popularidad popularidad del equipo de computo.
     */
    public Producto(String nombre, String color, String ram, String discoduro, String
dispositivooptico, double precio, double pantalla, String marca, String tipo, String
procesador, String so, int popularidad) {
        this.nombre = nombre;
        this.color = color;
        this.ram = ram;
        this.discoduro = discoduro;
        this.dispositivooptico = dispositivooptico;
        this.precio = precio;
        this.pantalla = pantalla;
        this.marca = marca;
        this.tipo = tipo;
        this.procesador = procesador;
        this.so = so;
        this.popularidad = popularidad;
    }

    /**
```

```
 * Obtiene el id de un equipo de computo.
 * @return id de un equipo.
 */
public Integer getIdProducto() {
   return this.idProducto;
}

/**
 * Define el id de un equipo de computo.
 * @param idProducto id de un equipo que se definira.
 */
public void setIdProducto(Integer idProducto) {
   this.idProducto = idProducto;
}
/**
 * Obtiene el nombre de un equipo de computo.
 * @return nombre de un equipo.
 */
public String getNombre() {
   return this.nombre;
}

/**
 * Define el nombre de un equipo de computo.
 * @param nombre nombre de un equipo que se definira.
 */
public void setNombre(String nombre) {
   this.nombre = nombre;
}
/**
 * Obtiene el color de un equipo de computo.
 * @return color de un equipo.
 */
public String getColor() {
   return this.color;
}

/**
 * Define el color de un equipo de computo.
 * @param color color de un equipo que se definira.
 */
public void setColor(String color) {
   this.color = color;
}
/**
 Obtiene la ram de un equipo de computo.
 * @return ram de un equipo.
```

```
 */
public String getRam() {
   return this.ram;
}

/**
 * Define la ram de un equipo de computo.
 * @param ram ram de un equipo que se definira.
 */
public void setRam(String ram) {
   this.ram = ram;
}
/**
 * Obtiene el disco duro de un equipo de computo.
 * @return disco duro de un equipo.
 */
public String getDiscoduro() {
   return this.discoduro;
}

/**
 * Define el disco duro de un equipo de computo.
 * @param discoduro disco duro de un equipo que se definira.
 */
public void setDiscoduro(String discoduro) {
   this.discoduro = discoduro;
}
/**
 * Obtiene el dispositivo optico de un equipo de computo.
 * @return dispositivo optico de un equipo.
 */
public String getDispositivooptico() {
   return this.dispositivooptico;
}

/**
 * Define el dispositivo optico de un equipo de computo.
 * @param dispositivooptico  dispositivo optico de un equipo que se definira.
 */
public void setDispositivooptico(String dispositivooptico) {
   this.dispositivooptico = dispositivooptico;
}
/**
 * Obtiene el precio de un equipo de computo.
 * @return precio de un equipo.
 */
public double getPrecio() {
```

```
    return this.precio;
}

/**
 * Define el precio de un equipo de computo.
 * @param precio precio de un equipo que se definira.
 */
public void setPrecio(double precio) {
    this.precio = precio;
}
/**
 * Obtiene la pantalla de un equipo de computo.
 * @return pantalla de un equipo.
 */
public double getPantalla() {
    return this.pantalla;
}

/**
 * Define la pantalla de un equipo de computo.
 * @param pantalla pantalla de un equipo que se definira.
 */
public void setPantalla(double pantalla) {
    this.pantalla = pantalla;
}
/**
 * Obtiene la marca de un equipo de computo.
 * @return marca de un equipo.
 */
public String getMarca() {
    return this.marca;
}

/**
 * Define la marca de un equipo de computo.
 * @param marca marca de un equipo que se definira.
 */
public void setMarca(String marca) {
    this.marca = marca;
}
/**
 * Obtiene el tipo de un equipo de computo.
 * @return tipo de un equipo.
 */
public String getTipo() {
    return this.tipo;
}
```

```
/**
 * Define el tipo de un equipo de computo.
 * @param tipo tipo de un equipo que se definira.
 */
public void setTipo(String tipo) {
   this.tipo = tipo;
}
/**
 * Obtiene el procesador de un equipo de computo.
 * @return procesador de un equipo.
 */
public String getProcesador() {
   return this.procesador;
}

/**
 * Define el procesador de un equipo de computo.
 * @param procesador procesador de un equipo.
 */
public void setProcesador(String procesador) {
   this.procesador = procesador;
}
/**
 * Obtiene el sistema operativo de un equipo de computo.
 * @return sistema operativo de un equipo.
 */
public String getSo() {
   return this.so;
}

/**
 * Define el sistema operativo de un equipo de computo.
 * @param so sistema operativo de un equipo
 */
public void setSo(String so) {
   this.so = so;
}
/**
 * Obtiene la popularidad de un equipo de computo.
 * @return popularidad de un equipo.
 */
public int getPopularidad() {
   return this.popularidad;
}

/**
```

```
    * Define la popularidad de un equipo de computo.
    * @param popularidad popularidad de un equipo.
    */
   public void setPopularidad(int popularidad) {
      this.popularidad = popularidad;
   }
}
```

Paquete vista

AgregarProducto

```
<%--
   Document   : AgregarProducto
   Created on : 20/06/2013, 12:01:09 AM
   Author     : Karina Guzmán Villanueva
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>

          <meta charset="UTF-8" />
          <title>RECOM</title>
          <link rel="stylesheet" type="text/css" href="css/style.css" />

 <link rel="shortcut icon" href="favicon.ico" >
  <link rel="icon" type="image/gif" href="animated_favicon1.gif" >
</head>

<!DOCTYPE html>

   <body>
      <h1>Agregar un producto</h1>
      <table border="1">
            <tr>
            <td>
   <s:form action="Agregar">

      <s:label name="nombrel" value="Nombre"/> <s:textfield name="nombre"/>
            </td>
            <td>
         <s:select label="Color"
                  headerKey="-1" headerValue="--- Select ---"
            list="{'Negro', 'Blanco', 'Plateado', 'Gris', 'Azul', 'Rosa', 'Morado'}"
            name="color"
```

```
            />
          </td>
          <td>
        <s:select label="Memoria RAM"
                headerKey="-1" headerValue="--- Select ---"
           list="{'2 GB', '4 GB', '6 GB', '8 GB'}"
           name="ram"
                 />
          </td>
          <td>
        <s:select label="Tipo"
                headerKey="-1" headerValue="--- Select ---"
           list="{'Netbook', 'Laptop', 'PC'}"
           name="tipo"
                 />
          </td>

          <td>
    <s:label name="preciol" value="Precio"/> <s:textfield name="precio"/>
          </td>
          <td>
          <s:select label="Disco Duro"
                headerKey="-1" headerValue="--- Select ---"
           list="{'250 GB', '500 GB', '750 GB', '1 TB', '2 TB'}"
           name="discoduro"
                 />
          </td>
          <td>
    <s:select label="Dispositivo Optico"
                headerKey="-1" headerValue="--- Select ---"
           list="{'No tiene', 'Reproductor/Grabador de DVD'}"
           name="dispositivooptico"
                 />
          </td>
          <td>
         <s:select label="Marca"
                headerKey="-1" headerValue="--- Select ---"
           list="{'Acer', 'HP', 'Dell', 'Sony'}"
           name="marca"
                 />
          </td>
          <td>

    <s:select label="Pantalla"
                headerKey="-1" headerValue="--- Select ---"
           list="{'10.1', '11.6', '13.3', '14', '15', '15.6', '18.5', '20'}"
           name="pantalla"
```

```
                    />
            </td>
            <td>
        <s:select label="Procesador"
                    headerKey="-1" headerValue="--- Select ---"
             list="{'AMD E2-1800', 'Intel Core i3', 'Intel Core i5', 'Intel Core i7', 'Intel
Celeron Dual Core', 'AMD A-Series Dual Core', 'Intel Atom N2600', 'AMD E2-3200',
'Intel Atom D425', 'AMD Fx-4100', 'AMD E-300 Dual Core', 'Intel Pentium G2020',
'Intel Celeron G465'}"
             name="procesador"
                    />
            </td>
            <td>
       <s:select label="Sistema Operativo"
                    headerKey="-1" headerValue="--- Select ---"
             list="{'Free DOS', 'Windows 7 Home Basic', 'Windows 7 Starter', 'Windows
8', 'Windows 8 Pro'}"
             name="so"
                    />
            </td>
            </tr>
       </table>

       <s:submit value="Agregar producto"/>
    </s:form>

    </body>
</html>
```

Lista

```
<%--
    Document   : Lista
    Created on : 17/09/2013, 11:22:53 AM
    Author     : Karina Guzmán Villanueva
--%>
<%@page import="Controlador.Consultar"%>
<%@page import="Modelo.Lista"%>
<%@page import="java.util.List"%>
<%@taglib uri="/struts-tags" prefix="s" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Consulta</title>
    </head>
    <body>
```

```
      <h1>Lista de productos</h1>
     <s:iterator value="lista">
       <s:property list="idProducto" name="id" listKey="idProducto"
listValue="nombre"/>
           </s:iterator>
  </body>
</html>
```

confirmacion_agregar

```
<%--
   Document   : confirmacion_agregar
   Created on : 20/06/2013, 12:05:21 AM
   Author     : Karina Guzmán Villanueva
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib uri="/struts-tags" prefix="s" %>
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Agregar producto</title>
   </head>
   <body>
      <h1>Producto agregado <s:property value="nombre"/> exitosamente</h1>
      <p><a href="/RECOM/Listar.action">Volver</a></p>

   </body>
</html>
```

confirmacion_editar

```
        <%--
           Document   : confirmacion_editar
           Created on : 9/09/2013, 02:06:39 AM
           Author     : Karina Guzmán Villanueva
        --%>

        <%@page contentType="text/html" pageEncoding="UTF-8"%>
        <%@taglib uri="/struts-tags" prefix="s" %>
        <!DOCTYPE html>
        <html>
           <head>
              <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
              <title>Compucat</title>
           </head>
           <body>
              <p>El producto <strong><s:property value="nombre" /></strong> ha sido
editado </p>
```

```
            <ul>
                <li><a href="/RECOM/Listar.action"><img
src="../botones/regresar.png">Volver</a></li>

            </ul>

        </body>
    </html>
```

confirmacion_eliminar

```
    <%--
        Document   : confirmacion_eliminar
        Created on : 9/09/2013, 02:05:53 AM
        Author     : Karina Guzmán Villanueva
    --%>

    <%@page contentType="text/html" pageEncoding="UTF-8"%>
    <%@taglib uri="/struts-tags" prefix="s" %>
    <!DOCTYPE html>
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
            <title>Eliminacion completa</title>

        </head>
        <body>
            <h1>Se ha eliminado el producto satisfactoriamente</h1>
            <a href="/RECOM/Listar.action">Volver</a></li>
        </body>
    </html>
```

editar

```
    <%--
        Document   : editar
        Created on : 9/09/2013, 02:07:15 AM
        Author     : Karina Guzmán Villanueva
    --%>

    <%@page contentType="text/html" pageEncoding="UTF-8"%>
    <%@taglib uri="/struts-tags" prefix="s" %>
    <!DOCTYPE html>
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
            <title>Edicion de datos de producto</title>
        </head>
        <body>
            <h1><s:property value="lista.nombre" /></h1>
```

```
            <s:form action="Edit">
              <s:textfield name="id" label="Id" readonly="true"
value="%{lista.idProducto}" />

              <s:textfield name="nombre" label="Nombre" value="%{lista.nombre}"
/>
              <s:select label="Color"
                    headerKey="%{lista.color}" headerValue="%{lista.color}"
                list="{'Negro', 'Blanco', 'Plateado', 'Gris', 'Azul', 'Rosa', 'Morado'}"
                name="color"
                    />
              <s:select label="Memoria RAM"
                    headerKey="%{lista.ram}" headerValue="%{lista.ram}"
                list="{'2 GB', '4 GB', '6 GB', '8 GB'}"
                name="ram"
                    />
              <s:select label="Tipo"
                    headerKey="%{lista.tipo}" headerValue="%{lista.tipo}"
                list="{'Netbook', 'Laptop', 'PC'}"
                name="tipo"
                    />


             <s:textfield name="precio" label = "Precio" value = "%{lista.precio}"/>
            <s:select label="Disco Duro"
                    headerKey="%{lista.discoduro}"
headerValue="%{lista.discoduro}"
                list="{'250 GB', '500 GB', '750 GB', '1 TB', '2 TB'}"
                name="discoduro"
                    />
            <s:select label="Dispositivo Optico"
                    headerKey="%{lista.dispositivooptico}"
headerValue="%{lista.dispositivooptico}"
                list="{'No tiene', 'Reproductor/Grabador de DVD'}"
                name="dispositivooptico"
                    />
             <s:select label="Marca"
                    headerKey="%{lista.marca}" headerValue="%{lista.marca}"
                list="{'Acer', 'HP', 'Dell', 'Sony'}"
                name="marca"
                    />
            <s:select label="Pantalla"
                    headerKey="%{lista.pantalla}" headerValue="%{lista.pantalla}"
                list="{'10.1', '11.6', '13.3', '14', '15', '15.6', '18.5', '20'}"
                name="pantalla"
                    />
```

```
                <s:select label="Procesador"
                        headerKey="%{lista.procesador}"
headerValue="%{lista.procesador}"
                    list="{'AMD E2-1800', 'Intel Core i3', 'Intel Core i5', 'Intel Core i7',
'Intel Celeron Dual Core', 'AMD A-Series Dual Core', 'Intel Atom N2600', 'AMD E2-
3200', 'Intel Atom D425', 'AMD Fx-4100', 'AMD E-300 Dual Core', 'Intel Pentium
G2020', 'Intel Celeron G465'}"
                    name="procesador"
                        />
                <s:select label="Sistema Operativo"
                        headerKey="%{lista.so}" headerValue="%{lista.so}"
                    list="{'Free DOS', 'Windows 7 Home Basic', 'Windows 7 Starter',
'Windows 8', 'Windows 8 Pro'}"
                    name="so"
                        />

                  <s:submit value="Guardar cambios" />
               </s:form>
            </body>
         </html>
```

eliminar

```
         <%--
            Document   : eliminar
            Created on : 9/09/2013, 02:07:38 AM
            Author     : Karina Guzmán Villanueva
         --%>
         <%@page import="controlador.Obtener"%>
         <%@page import="modelo.Producto"%>
         <%@page import="java.util.List"%>
         <%@page contentType="text/html" pageEncoding="UTF-8"%>
         <%@taglib uri="/struts-tags" prefix="s" %>
         <!DOCTYPE html>
         <html>
            <head>
               <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
               <title>Lista de productos</title>
            </head>
            <body>
               <div>
                  <s:form action="Eliminar">

                     <s:iterator value="lista">
                        <s:radio list="idProducto" name="id" listKey="idProducto"
listValue="nombre"/>
                     </s:iterator>
```

```
                    <s:submit value="Eliminar" />
                </s:form>
            </div>
        </body>
    </html>
```

index

```
<!DOCTYPE html>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<html>
<head>

        <meta charset="UTF-8" />
        <title>RECOM</title>
        <link rel="stylesheet" type="text/css" href="css/style.css" />

</head>
<body>
        <div id="page">
                <div id="header">
                        <div>

                        </div>

                </div>




                        <div id="home">
                                <div>


                                        <s:form action="doLogin"
method="POST">
                                <center>
                                <table border ="1">
                        <tr>
                        <td colspan="2">
                        Login
                        </td>
                        </tr>
                        <tr>
                        <td colspan="2">
```

```
                        <s:actionerror />
                        <s:fielderror />
                        </td>
                        </tr>
                        <s:textfield name="username" label="Login name"/>
                        <s:password name="password" label="Password"/>
                        <s:submit value="Login" align="center"/>
                        </s:form>

                            </table></center>
                              </div>


                </div>
                <div id="footer">
                        <div>
                                <div>


                                </div>

                        </div>
                </div>
        </div>

</body>
</html>
```

lista

```
<%@page import="java.util.List"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>

        <meta charset="UTF-8" />
        <title>RECOM</title>
        <link rel="stylesheet" type="text/css" href="css/style.css" />

 <link rel="shortcut icon" href="favicon.ico" >
   <link rel="icon" type="image/gif" href="animated_favicon1.gif" >
</head>

<!DOCTYPE html>
   <body>
      <h1>Equipos de cómputo</h1>
```

```
<div>
  <table border="1">
    <tr>
    <td>
      Nombre
    </td>
    <td>
      Tipo
    </td>
    <td>
      RAM
    </td>
    <td>
      Disco duro
    </td>
    <td>
      Color
    </td>
    <td>
      Dispositivo Optico
    </td>
    <td>
      Pantalla
    </td>
    <td>
      Procesador
    </td>
    <td>
      Sistema Operativo
    </td>
    <td>
      Precio
    </td>
    </tr>
     <s:iterator value="listas">

          <td>
       <s:property value="nombre" />
       </td>
       <td>
       <s:property value="tipo" />
       </td>
       <td>
       <s:property value="ram" />
       </td>
       <td>
       <s:property value="discoduro" />
```

```
                        </td>
                        <td>
                        <s:property value="color" />
                        </td>
                        <td>
                        <s:property value="dispositivooptico" />
                        </td>
                        <td>
                        <s:property value="pantalla" />
                        </td>
                        <td>
                           <s:property value = "procesador" />
                        </td>
                        <td>
                        <s:property value="so" />
                        </td>
                        <td>
                        <s:property value="precio" />
                        </td>


                        <td>
                           <a href='/RECOM/Edit.action?id=<s:property
value="idProducto" />'>Editar</a>

                        </td>


                        <td>
                           <a href='/RECOM/Eliminar.action?id=<s:property
value="idProducto" />'>Eliminar</a>

                        </td>
                        </tr>

                     </s:iterator>
                  </table>
                  <a href="/RECOM/AgregarProducto.jsp"> Agregar un producto</a>
               </div>
            </body>
         </html>
```

Servicio web

Paquete model

```
package model;

/**
```

```
 * Esta clase sirve para recibir una ponderacion dada por un consumidor.
 * @author Karina Guzmán Villanueva
 */
public class Ponderacion {

   private int importancia;
   private String Caracteristica;
   private String Descripcion;

   /**
    * Constructor de la clase.
    * @param importancia importancia que se da a la caracteristica.
    * @param Caracteristica Caracteristica que se califica.
    * @param Descripcion descripcion de la caracteristica.
    */
   public Ponderacion(int importancia, String Caracteristica, String Descripcion)
{
      this.importancia = importancia;
      this.Caracteristica = Caracteristica;
      this.Descripcion = Descripcion;
   }

   /**
    * Obtiene la importancia de la ponderacion.
    * @return Importancia de la ponderacion.
    */
   public int getImportancia() {
      return importancia;
   }

   /**
    *Define la importancia de la ponderacion.
    * @param importancia importancia de la ponderacion.
    */
   public void setImportancia(int importancia) {
      this.importancia = importancia;
   }

   /**
    * Obtiene la caracteristica de la ponderacion.
    * @return caracteristica de la ponderacion.
    */
   public String getCaracteristica() {
      return Caracteristica;
   }

   /**
```

```
     * Define la caracteristica de la ponderacion.
     * @param Caracteristica caracteristica de la ponderacion.
     */
    public void setCaracteristica(String Caracteristica) {
        this.Caracteristica = Caracteristica;
    }

    /**
     * Obtiene la descripcion de la ponderacion.
     * @return descripcion de la ponderacion.
     */
    public String getDescripcion() {
        return Descripcion;
    }

    /**
     * Define la descripcion de la ponderacion.
     * @param Descripcion descripcion de la ponderacion.
     */
    public void setDescripcion(String Descripcion) {
        this.Descripcion = Descripcion;
    }


}
```

Producto

```
package model;

import java.io.Serializable;
import javax.persistence.*;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;

/**
 * Esta clase realiza la comunicacion con la base de datos.
 * @author Karina Guzmán Villanueva
 */
@Entity
@Table(name = "producto")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Producto.findAll", query = "SELECT p FROM
Producto p"),
    @NamedQuery(name = "Producto.findByIdProducto", query = "SELECT p
```

```
FROM Producto p WHERE p.idProducto = :idProducto"),
        @NamedQuery(name = "Producto.findByNombre", query = "SELECT p
FROM Producto p WHERE p.nombre = :nombre"),
        @NamedQuery(name = "Producto.findByColor", query = "SELECT p FROM
Producto p WHERE p.color = :color"),
        @NamedQuery(name = "Producto.findByRam", query = "SELECT p FROM
Producto p WHERE p.ram = :ram"),
        @NamedQuery(name = "Producto.findByDiscoduro", query = "SELECT p
FROM Producto p WHERE p.discoduro = :discoduro"),
        @NamedQuery(name = "Producto.findByDispositivooptico", query =
"SELECT p FROM Producto p WHERE p.dispositivooptico = :dispositivooptico"),
        @NamedQuery(name = "Producto.findByPrecio", query = "SELECT p
FROM Producto p WHERE p.precio = :precio"),
        @NamedQuery(name = "Producto.findByPantalla", query = "SELECT p
FROM Producto p WHERE p.pantalla = :pantalla"),
        @NamedQuery(name = "Producto.findByMarca", query = "SELECT p
FROM Producto p WHERE p.marca = :marca"),
        @NamedQuery(name = "Producto.findByTipo", query = "SELECT p FROM
Producto p WHERE p.tipo = :tipo"),
        @NamedQuery(name = "Producto.findByProcesador", query = "SELECT p
FROM Producto p WHERE p.procesador = :procesador"),
        @NamedQuery(name = "Producto.findBySo", query = "SELECT p FROM
Producto p WHERE p.so = :so"),
        @NamedQuery(name = "Producto.findByPopularidad", query = "SELECT p
FROM Producto p WHERE p.popularidad = :popularidad")})
    public class Producto implements Serializable {
        private static final long serialVersionUID = 1L;
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Basic(optional = false)
        @NotNull
        @Column(name = "id_producto")
        private Integer idProducto;
        @Basic(optional = false)
        @NotNull
        @Size(min = 1, max = 60)
        @Column(name = "nombre")
        private String nombre;
        @Basic(optional = false)
        @NotNull
        @Size(min = 1, max = 10)
        @Column(name = "color")
        private String color;
        @Basic(optional = false)
        @NotNull
        @Size(min = 1, max = 5)
        @Column(name = "RAM")
```

```
    private String ram;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 7)
    @Column(name = "discoduro")
    private String discoduro;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 60)
    @Column(name = "dispositivooptico")
    private String dispositivooptico;
    @Basic(optional = false)
    @NotNull
    @Column(name = "precio")
    private double precio;
    @Basic(optional = false)
    @NotNull
    @Column(name = "pantalla")
    private double pantalla;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 5)
    @Column(name = "marca")
    private String marca;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 15)
    @Column(name = "tipo")
    private String tipo;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 30)
    @Column(name = "procesador")
    private String procesador;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 20)
    @Column(name = "so")
    private String so;
    @Basic(optional = false)
    @NotNull
    @Column(name = "popularidad")
    private int popularidad;

    /**
     * Constructor de la clase.
     */
```

```
    public Producto() {
    }

    /**
     * Constructor de la clase.
     * @param idProducto id del equipo de computo.
     */
    public Producto(Integer idProducto) {
       this.idProducto = idProducto;
    }

    /**
     * Constructor de la clase.
     * @param idProducto id del equipo de computo.
     * @param nombre nombre del equipo de computo.
     * @param color color del equipo de computo.
     * @param ram ram del equipo de computo.
     * @param discoduro disco duro del equipo de computo.
     * @param dispositivooptico dispositivo optico del equipo de computo.
     * @param precio precio del equipo de computo.
     * @param pantalla pantalla del equipo de computo.
     * @param marca marca del equipo de computo.
     * @param tipo tipo del equipo de computo.
     * @param procesador procesador del equipo de computo.
     * @param so sistema operativo del equipo de computo.
     * @param popularidad popularidad del equipo de computo.
     */
    public Producto(Integer idProducto, String nombre, String color, String ram,
String discoduro, String dispositivooptico, double precio, double pantalla, String marca,
String tipo, String procesador, String so, int popularidad) {
       this.idProducto = idProducto;
       this.nombre = nombre;
       this.color = color;
       this.ram = ram;
       this.discoduro = discoduro;
       this.dispositivooptico = dispositivooptico;
       this.precio = precio;
       this.pantalla = pantalla;
       this.marca = marca;
       this.tipo = tipo;
       this.procesador = procesador;
       this.so = so;
       this.popularidad = popularidad;
    }

    /**
     * Obtiene el id de un equipo de computo.
```

```
     * @return id de un equipo.
     */
    public Integer getIdProducto() {
        return idProducto;
    }

    /**
     * Define el id de un equipo de computo.
     * @param idProducto id de un equipo que se definira.
     */
    public void setIdProducto(Integer idProducto) {
        this.idProducto = idProducto;
    }

    /**
     * Obtiene el nombre de un equipo de computo.
     * @return nombre de un equipo.
     */
    public String getNombre() {
        return nombre;
    }


    /**
     * Define el nombre de un equipo de computo.
     * @param nombre nombre de un equipo que se definira.
     */
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

   /**
     * Obtiene el color de un equipo de computo.
     * @return color de un equipo.
     */
    public String getColor() {
        return color;
    }

    /**
     * Define el color de un equipo de computo.
     * @param color color de un equipo que se definira.
     */
    public void setColor(String color) {
        this.color = color;
    }
```

```
/**
 *Obtiene la ram de un equipo de computo.
 * @return ram de un equipo.
 */
public String getRam() {
   return ram;
}

/**
 * Define la ram de un equipo de computo.
 * @param ram ram de un equipo que se definira.
 */
public void setRam(String ram) {
   this.ram = ram;
}

/**
 * Obtiene el disco duro de un equipo de computo.
 * @return disco duro de un equipo.
 */
public String getDiscoduro() {
   return discoduro;
}

/**
 * Define el disco duro de un equipo de computo.
 * @param discoduro disco duro de un equipo que se definira.
 */
public void setDiscoduro(String discoduro) {
   this.discoduro = discoduro;
}

/**
 * Obtiene el dispositivo optico de un equipo de computo.
 * @return dispositivo optico de un equipo.
 */
public String getDispositivooptico() {
   return dispositivooptico;
}

/**
 * Define el dispositivo optico de un equipo de computo.
 * @param dispositivooptico  dispositivo optico de un equipo que se definira.
 */
public void setDispositivooptico(String dispositivooptico) {
   this.dispositivooptico = dispositivooptico;
}
```

```
/**
 * Obtiene el precio de un equipo de computo.
 * @return precio de un equipo.
 */
public double getPrecio() {
   return precio;
}

 /**
 * Define el precio de un equipo de computo.
 * @param precio precio de un equipo que se definira.
 */
public void setPrecio(double precio) {
   this.precio = precio;
}

/**
 * Obtiene la pantalla de un equipo de computo.
 * @return pantalla de un equipo.
 */
public double getPantalla() {
   return pantalla;
}

/**
 * Define la pantalla de un equipo de computo.
 * @param pantalla pantalla de un equipo que se definira.
 */
public void setPantalla(double pantalla) {
   this.pantalla = pantalla;
}

 /**
 * Obtiene la marca de un equipo de computo.
 * @return marca de un equipo.
 */
public String getMarca() {
   return marca;
}

 /**
 * Define la marca de un equipo de computo.
 * @param marca marca de un equipo que se definira.
 */
public void setMarca(String marca) {
   this.marca = marca;
```

```
    }

    /**
     * Obtiene el tipo de un equipo de computo.
     * @return tipo de un equipo.
     */
    public String getTipo() {
        return tipo;
    }

    /**
     * Define el tipo de un equipo de computo.
     * @param tipo tipo de un equipo que se definira.
     */
    public void setTipo(String tipo) {
        this.tipo = tipo;
    }

    /**
     * Obtiene el procesador de un equipo de computo.
     * @return procesador de un equipo.
     */
    public String getProcesador() {
        return procesador;
    }

    /**
     * Define el procesador de un equipo de computo.
     * @param procesador procesador de un equipo.
     */
    public void setProcesador(String procesador) {
        this.procesador = procesador;
    }

    /**
     * Obtiene el sistema operativo de un equipo de computo.
     * @return sistema operativo de un equipo.
     */
    public String getSo() {
        return so;
    }

     /**
     * Define el sistema operativo de un equipo de computo.
     * @param so sistema operativo de un equipo
     */
    public void setSo(String so) {
```

```
        this.so = so;
    }

    /**
     * Obtiene la popularidad de un equipo de computo.
     * @return popularidad de un equipo.
     */
    public int getPopularidad() {
        return popularidad;
    }

    /**
     * Define la popularidad de un equipo de computo.
     * @param popularidad popularidad de un equipo.
     */
    public void setPopularidad(int popularidad) {
        this.popularidad = popularidad;
    }

    /**
     * Obtiene el identificdor de un equipo de computo.
     * @return identificdor de un equipo de computo.
     */
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idProducto != null ? idProducto.hashCode() : 0);
        return hash;
    }

    /**
     * Compara un par de objetos si son iguales
     * @param object objeto  para comparar.
     * @return verdadero si son iguales.
     */
    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not
set
        if (!(object instanceof Producto)) {
            return false;
        }
        Producto other = (Producto) object;
        if ((this.idProducto == null && other.idProducto != null) || (this.idProducto
!= null && !this.idProducto.equals(other.idProducto))) {
            return false;
        }
```

```
        return true;
    }

    /**
     * Metodo que imprime un objeto.
     * @return
     */
    @Override
    public String toString() {
        return "model.Producto[ idProducto=" + idProducto + " ]";
    }

}
```

ProductoEvaluado

```
package model;

/**
 * Esta clase realiza la evaluacion de un equipo de computo.
 * @author Karina Guzmán Villanueva
 */
public class ProductoEvaluado {

    private Integer idProducto;
    private String nombre;
    private String color;
    private String ram;
    private String discoduro;
    private String dispositivooptico;
    private double precio;
    private double pantalla;
    private String marca;
    private String tipo;
    private String procesador;
    private String so;
    private int popularidad;
    private double evaluacion;

    /**
     * Constructor de la clase.
     * @param p equipo de computo a evaluar.
     * @param ev Evaluacion.
     */
    public ProductoEvaluado(Producto p, double ev) {
        this.idProducto = p.getIdProducto();
        this.nombre = p.getNombre();
        this.color = p.getColor();
```

```
        this.ram = p.getRam();
        this.discoduro = p.getDiscoduro();
        this.dispositivooptico = p.getDispositivooptico();
        this.precio = p.getPrecio();
        this.pantalla = p.getPantalla();
        this.marca = p.getMarca();
        this.tipo = p.getMarca();
        this.procesador = p.getProcesador();
        this.so = p.getSo();
        this.popularidad = p.getPopularidad();
        this.evaluacion = ev;
    }

    /**
     * Constructor de la clase.
     * @param idProducto id del equipo de computo.
     * @param nombre nombre del equipo de computo.
     * @param color color del equipo de computo.
     * @param ram ram del equipo de computo.
     * @param discoduro disco duro del equipo de computo.
     * @param dispositivooptico dispositivo optico del equipo de computo.
     * @param precio precio del equipo de computo.
     * @param pantalla pantalla del equipo de computo.
     * @param marca marca del equipo de computo.
     * @param tipo tipo del equipo de computo.
     * @param procesador procesador del equipo de computo.
     * @param so sistema operativo del equipo de computo.
     * @param popularidad popularidad del equipo de computo.
     */
    public ProductoEvaluado(Integer idProducto, String nombre, String color,
String ram, String discoduro, String dispositivooptico, double precio, double pantalla,
String marca, String tipo, String procesador, String so, int popularidad, double
evaluacion) {
        this.idProducto = idProducto;
        this.nombre = nombre;
        this.color = color;
        this.ram = ram;
        this.discoduro = discoduro;
        this.dispositivooptico = dispositivooptico;
        this.precio = precio;
        this.pantalla = pantalla;
        this.marca = marca;
        this.tipo = tipo;
        this.procesador = procesador;
        this.so = so;
        this.popularidad = popularidad;
        this.evaluacion = evaluacion;
```

```
}

/**
 *  Constructor de la clase.
 */
public ProductoEvaluado() {
}

/**
 * Obtiene el id de un equipo de computo.
 * @return id de un equipo.
 */
public Integer getIdProducto() {
   return idProducto;
}

/**
 * Define el id de un equipo de computo.
 * @param idProducto id de un equipo que se definira.
 */
public void setIdProducto(Integer idProducto) {
   this.idProducto = idProducto;
}

/**
 * Obtiene el nombre de un equipo de computo.
 * @return nombre de un equipo.
 */
public String getNombre() {
   return nombre;
}


/**
 * Define el nombre de un equipo de computo.
 * @param nombre nombre de un equipo que se definira.
 */
public void setNombre(String nombre) {
   this.nombre = nombre;
}

/**
 * Obtiene el color de un equipo de computo.
 * @return color de un equipo.
 */
public String getColor() {
   return color;
```

```
}

/**
 * Define el color de un equipo de computo.
 * @param color color de un equipo que se definira.
 */
public void setColor(String color) {
   this.color = color;
}

/**
 *Obtiene la ram de un equipo de computo.
 * @return ram de un equipo.
 */
public String getRam() {
   return ram;
}

/**
 * Define la ram de un equipo de computo.
 * @param ram ram de un equipo que se definira.
 */
public void setRam(String ram) {
   this.ram = ram;
}

/**
 * Obtiene el disco duro de un equipo de computo.
 * @return disco duro de un equipo.
 */
public String getDiscoduro() {
   return discoduro;
}

/**
 * Define el disco duro de un equipo de computo.
 * @param discoduro disco duro de un equipo que se definira.
 */
public void setDiscoduro(String discoduro) {
   this.discoduro = discoduro;
}

/**
 * Obtiene el dispositivo optico de un equipo de computo.
 * @return dispositivo optico de un equipo.
 */
public String getDispositivooptico() {
```

```
    return dispositivooptico;
}

/**
 * Define el dispositivo optico de un equipo de computo.
 * @param dispositivooptico  dispositivo optico de un equipo que se definira.
 */
public void setDispositivooptico(String dispositivooptico) {
    this.dispositivooptico = dispositivooptico;
}

/**
 * Obtiene el precio de un equipo de computo.
 * @return precio de un equipo.
 */
public double getPrecio() {
    return precio;
}

 /**
 * Define el precio de un equipo de computo.
 * @param precio precio de un equipo que se definira.
 */
public void setPrecio(double precio) {
    this.precio = precio;
}

/**
 * Obtiene la pantalla de un equipo de computo.
 * @return pantalla de un equipo.
 */
public double getPantalla() {
    return pantalla;
}

/**
 * Define la pantalla de un equipo de computo.
 * @param pantalla pantalla de un equipo que se definira.
 */
public void setPantalla(double pantalla) {
    this.pantalla = pantalla;
}

/**
 * Obtiene la marca de un equipo de computo.
 * @return marca de un equipo.
 */
```

```
    public String getMarca() {
        return marca;
    }

    /**
     * Define la marca de un equipo de computo.
     * @param marca marca de un equipo que se definira.
     */
    public void setMarca(String marca) {
        this.marca = marca;
    }

    /**
     * Obtiene el tipo de un equipo de computo.
     * @return tipo de un equipo.
     */
    public String getTipo() {
        return tipo;
    }

    /**
     * Define el tipo de un equipo de computo.
     * @param tipo tipo de un equipo que se definira.
     */
    public void setTipo(String tipo) {
        this.tipo = tipo;
    }

    /**
     * Obtiene el procesador de un equipo de computo.
     * @return procesador de un equipo.
     */
    public String getProcesador() {
        return procesador;
    }

    /**
     * Define el procesador de un equipo de computo.
     * @param procesador procesador de un equipo.
     */
    public void setProcesador(String procesador) {
        this.procesador = procesador;
    }

    /**
     * Obtiene el sistema operativo de un equipo de computo.
     * @return sistema operativo de un equipo.
```

```
     */
    public String getSo() {
        return so;
    }

     /**
     * Define el sistema operativo de un equipo de computo.
     * @param so sistema operativo de un equipo
     */
    public void setSo(String so) {
        this.so = so;
    }

    /**
     * Obtiene la popularidad de un equipo de computo.
     * @return popularidad de un equipo.
     */
    public int getPopularidad() {
        return popularidad;
    }

    /**
     * Define la popularidad de un equipo de computo.
     * @param popularidad popularidad de un equipo.
     */
    public void setPopularidad(int popularidad) {
        this.popularidad = popularidad;
    }
    /**
     * Define la evaluacion de un equipo de computo.
     * @param evaluacion evaluacion de un equipo.
     */
    public void setEvaluacion(double evaluacion) {
        this.evaluacion = evaluacion;
    }



    /**
     *Obtiene la marca de un equipo de computo.
     * @return marca de un equipo.
     */
    public double getEvaluacion() {
        return evaluacion;
    }

}
```

Servicio
AbstractFacade

```
package service;

import java.util.List;
import javax.persistence.EntityManager;

/**
 * Esta clase defdine los metodos genericos de REST.
 * @param <T> Entidad.
 * @author Karina Guzmán Villanueva
 */public abstract class AbstractFacade<T> {
   private Class<T> entityClass;

   /**
    * Constructor de la clase.
    * @param entityClass
    */
   public AbstractFacade(Class<T> entityClass) {
      this.entityClass = entityClass;
   }

   /**
    * Obtiene un objeto.
    * @return objeto.
    */
   protected abstract EntityManager getEntityManager();

   /**
    * Crea un objeto
    * @param entity  Datos del objeto.
    */
   public void create(T entity) {
      getEntityManager().persist(entity);
   }

   /**
    * Edita un objeto
    * @param entity objeto a editar.
    */
   public void edit(T entity) {
      getEntityManager().merge(entity);
   }

   /**
    * Elimina un objeto
    * @param entity objeto a eliminar.
```

```
     */
    public void remove(T entity) {
        getEntityManager().remove(getEntityManager().merge(entity));
    }

    /**
     * Encuentra un objeto por id
     * @param id id objeto.
     * @return
     */
    public T find(Object id) {
        return getEntityManager().find(entityClass, id);
    }

    /**
     * regresa una lista de todos los objetos.
     * @return lista de objetos.
     */
    public List<T> findAll() {
        javax.persistence.criteria.CriteriaQuery cq =
getEntityManager().getCriteriaBuilder().createQuery();
        cq.select(cq.from(entityClass));
        return getEntityManager().createQuery(cq).getResultList();
    }

    /**
     * Regresa los objetos entre cierto rango.
     * @param range rango entre el que se encuentran los objetos.
     * @return
     */
    public List<T> findRange(int[] range) {
        javax.persistence.criteria.CriteriaQuery cq =
getEntityManager().getCriteriaBuilder().createQuery();
        cq.select(cq.from(entityClass));
        javax.persistence.Query q = getEntityManager().createQuery(cq);
        q.setMaxResults(range[1] - range[0]);
        q.setFirstResult(range[0]);
        return q.getResultList();
    }

    /**
     * Cuenta el numero de objetos
     * @return
     */
    public int count() {
        javax.persistence.criteria.CriteriaQuery cq =
getEntityManager().getCriteriaBuilder().createQuery();
```

```
        javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
        cq.select(getEntityManager().getCriteriaBuilder().count(rt));
        javax.persistence.Query q = getEntityManager().createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    }

}
```

ProductoFacadeREST

```
package service;

import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.ws.rs.*;
import model.Producto;

/**
 * Esta clase defdine los metodos de REST con los equipos de computo.
 * @param <T> Entidad.
 * @author Karina Guzmán Villanueva
 */
@Stateless
@Path("model.producto")
public class ProductoFacadeREST extends AbstractFacade<Producto> {
    @PersistenceContext(unitName = "RESTPrueba5PU")
    private EntityManager em;

    /**
     * * Constructor de la clase.
     */
    public ProductoFacadeREST() {
        super(Producto.class);
    }

    /**
     * Crea un equipo de computo.
     * @param entity Datos del equipo.
     */
    @POST
    @Override
    @Consumes({"application/xml", "application/json"})
    public void create(Producto entity) {
        super.create(entity);
    }
```

```
    /**
     * Edita un equipo de computo.
     * @param entity objeto a editar.
     */
    @PUT
    @Override
    @Consumes({"application/xml", "application/json"})
    public void edit(Producto entity) {
        super.edit(entity);
    }

    /**
     * Elimina un equipo de computo.
     * @param id id del equipo.
     */
    @DELETE
    @Path("{id}")
    public void remove(@PathParam("id") Integer id) {
        super.remove(super.find(id));
    }

    /**
     * Obtiene un equipo de computo por id.
     * @param id id del equipo.
     * @return Equipo de computo.
     */
    @GET
    @Path("{id}")
    @Produces({"application/xml", "application/json"})
    public Producto find(@PathParam("id") Integer id) {
        return super.find(id);
    }

    /**
     * Obtiene todos los equipos de computo.
     * @return Lista de equipos de computo.
     */
    @GET
    @Override
    @Produces({"application/json"})
    public List<Producto> findAll() {
        return super.findAll();
    }

    /**
     * Obtiene todos los equipos de computo en un rango de ids.
     * @param from Inicio de rango.
```

```
     * @param to Fin de rango.
     * @return
     */
    @GET
    @Path("{from}/{to}")
    @Produces({"application/xml", "application/json"})
    public List<Producto> findRange(@PathParam("from") Integer from,
@PathParam("to") Integer to) {
        return super.findRange(new int[]{from, to});
    }

    /**
     * Cuenta todos los equipos de computo.
     * @return Numero de equipos de computo.
     */
    @GET
    @Path("count")
    @Produces("text/plain")
    public String countREST() {
        return String.valueOf(super.count());
    }

    /**
     * Obtiene un entidad en forma de equipos de computo
     * @return
     */
    @java.lang.Override
    protected EntityManager getEntityManager() {
        return em;
    }

}
```

Aplicación cliente

Paquete Vista

activity_main.xml

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/ScrollView1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical" >
```

```
        <RelativeLayout
            android:id="@android:id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent" >

            <TextView
                android:id="@+id/label_a"
                android:layout_width="50dp"
                android:layout_height="wrap_content"
                android:layout_alignRight="@+id/button1"
                android:text="Color" />

    <Spinner
        android:id="@+id/sel_color"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_below="@+id/label_a"
         />
    <ImageView android:id="@+id/imageView1"
         android:layout_width="50dp"
         android:layout_height="50dp"
         android:layout_toRightOf="@+id/sel_color"
         android:src="@drawable/color" />

    <View android:id="@+id/view1"
        android:layout_width="fill_parent"
        android:layout_height="20dp"
         android:layout_below="@+id/sel_color"
         android:background="#FFFFFF"/>


    <TextView
         android:id="@+id/label_ai"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="Importancia"
         android:layout_below="@+id/view1"/>
    <Spinner
        android:id="@+id/sel_importancia"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_below="@+id/label_ai"
         />

    <View android:id="@+id/view2"
        android:layout_width="fill_parent"
        android:layout_height="40dp"
```

```
        android:layout_below="@+id/sel_importancia"
        android:background="#FFFFFF"/>

    <TextView
        android:id="@+id/label_b"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/view2"
        android:text="Marca:"/>
    <Spinner
       android:id="@+id/sel_marca"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_below="@+id/label_b"
        />
    <ImageView android:id="@+id/imageView2"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_toRightOf="@+id/sel_marca"
        android:layout_below="@+id/view2"
        android:src="@drawable/mark" />

    <View android:id="@+id/view3"
       android:layout_width="fill_parent"
       android:layout_height="20dp"
        android:layout_below="@+id/sel_marca"
        android:background="#FFFFFF"/>


    <TextView
        android:id="@+id/label_bi"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Importancia"
        android:layout_below="@+id/view3"/>
    <Spinner
       android:id="@+id/sel_importanciam"
       android:layout_width="150dp"
       android:layout_height="wrap_content"
       android:layout_below="@+id/label_bi"
        />

    <View android:id="@+id/view4"
       android:layout_width="fill_parent"
       android:layout_height="40dp"
        android:layout_below="@+id/sel_importanciam"
        android:background="#FFFFFF"/>
```

```
 <TextView
     android:id="@+id/label_c"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
      android:layout_below="@+id/view4"
     android:text="RAM"/>
<Spinner
    android:id="@+id/sel_ram"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/label_c"
     />
<ImageView android:id="@+id/imageView3"
     android:layout_width="50dp"
     android:layout_height="50dp"
     android:layout_toRightOf="@+id/sel_ram"
     android:layout_below="@+id/view4"
     android:src="@drawable/ddr2" />

<View android:id="@+id/view5"
    android:layout_width="fill_parent"
    android:layout_height="20dp"
     android:layout_below="@+id/sel_ram"
     android:background="#FFFFFF"/>


<TextView
     android:id="@+id/label_ci"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="Importancia"
     android:layout_below="@+id/view5"
    />
<Spinner
    android:id="@+id/sel_importanciaram"
    android:layout_width="150dp"
    android:layout_height="wrap_content"
    android:layout_below="@+id/label_ci"
     />

<View android:id="@+id/view6"
    android:layout_width="fill_parent"
    android:layout_height="40dp"
     android:layout_below="@+id/sel_importanciaram"
     android:background="#FFFFFF"/>
<TextView
```

```
        android:id="@+id/label_d"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/view6"
        android:text="SO:"></TextView>
    <Spinner
       android:id="@+id/sel_so"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_below="@+id/label_d"
        />
    <ImageView android:id="@+id/imageView4"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_toRightOf="@+id/sel_so"
        android:layout_below="@+id/view6"
        android:src="@drawable/so" />

    <View android:id="@+id/view7"
       android:layout_width="fill_parent"
       android:layout_height="20dp"
        android:layout_below="@+id/sel_so"
        android:background="#FFFFFF"/>


    <TextView
        android:id="@+id/label_di"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Importancia"
        android:layout_below="@+id/view7"
        />
    <Spinner
       android:id="@+id/sel_importanciaso"
       android:layout_width="150dp"
       android:layout_height="wrap_content"
       android:layout_below="@+id/label_di"
        />

    <View android:id="@+id/view8"
       android:layout_width="fill_parent"
       android:layout_height="40dp"
        android:layout_below="@+id/sel_importanciaso"
        android:background="#FFFFFF"/>
    <TextView
        android:id="@+id/label_e"
        android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_below="@id/view8"
        android:text="Disco Duro:"/>
    <Spinner
       android:id="@+id/sel_hdd"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_below="@+id/label_e"
        />
    <ImageView android:id="@+id/imageView5"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_toRightOf="@+id/sel_hdd"
        android:layout_below="@+id/view8"
        android:src="@drawable/hdd" />

    <View android:id="@+id/view9"
       android:layout_width="fill_parent"
       android:layout_height="20dp"
        android:layout_below="@+id/sel_hdd"
        android:background="#FFFFFF"/>


    <TextView
        android:id="@+id/label_ei"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Importancia"
        android:layout_below="@+id/view9"
        />
    <Spinner
       android:id="@+id/sel_importanciahdd"
       android:layout_width="150dp"
       android:layout_height="wrap_content"
       android:layout_below="@+id/label_ei"
        />

    <View android:id="@+id/view10"
       android:layout_width="fill_parent"
       android:layout_height="40dp"
        android:layout_below="@+id/sel_importanciahdd"
        android:background="#FFFFFF"/>
    <TextView
        android:id="@+id/label_f"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/view10"
```

```
        android:text="Tipo:"/>
    <Spinner
       android:id="@+id/sel_tip"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_below="@+id/label_f"
        />
    <ImageView android:id="@+id/imageView6"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_toRightOf="@+id/sel_tip"
        android:layout_below="@+id/view10"
        android:src="@drawable/type" />

    <View android:id="@+id/view11"
       android:layout_width="fill_parent"
       android:layout_height="20dp"
        android:layout_below="@+id/sel_tip"
        android:background="#FFFFFF"/>


    <TextView
        android:id="@+id/label_fi"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Importancia:"
        android:layout_below="@+id/view11"
        />
    <Spinner
       android:id="@+id/sel_importanciatip"
       android:layout_width="150dp"
       android:layout_height="wrap_content"
       android:layout_below="@+id/label_fi"
        />

    <View android:id="@+id/view12"
       android:layout_width="fill_parent"
       android:layout_height="40dp"
        android:layout_below="@+id/sel_importanciatip"
        android:background="#FFFFFF"/>


    <TextView
        android:id="@+id/label_g"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/view12"
```

```
        android:text="Disp. óptico:"/>
    <Spinner
       android:id="@+id/sel_do"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_below="@+id/label_g"
        />

    <ImageView android:id="@+id/imageView7"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_toRightOf="@+id/sel_do"
        android:layout_below="@+id/view12"
        android:src="@drawable/dop" />

    <View android:id="@+id/view13"
       android:layout_width="50dp"
       android:layout_height="50dp"
        android:layout_below="@+id/sel_do"
        android:background="#FFFFFF"/>


    <TextView
        android:id="@+id/label_gi"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Importancia"
        android:layout_below="@+id/view13"
       />
    <Spinner
       android:id="@+id/sel_importanciadop"
       android:layout_width="150dp"
       android:layout_height="wrap_content"
       android:layout_below="@+id/label_gi"
        />

    <View android:id="@+id/view14"
       android:layout_width="fill_parent"
       android:layout_height="40dp"
        android:layout_below="@+id/sel_importanciadop"
        android:background="#FFFFFF"/>


    <TextView
        android:id="@+id/label_h"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```

```
        android:layout_below="@id/view14"
        android:text="Procesador:"/>
    <Spinner
       android:id="@+id/sel_proc"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_below="@+id/label_h"
        />
    <ImageView android:id="@+id/imageView8"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_toRightOf="@+id/sel_proc"
        android:layout_below="@+id/view14"
        android:src="@drawable/processor" />

    <View android:id="@+id/view15"
       android:layout_width="fill_parent"
       android:layout_height="20dp"
        android:layout_below="@+id/sel_proc"
        android:background="#FFFFFF"/>


    <TextView
        android:id="@+id/label_hi"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Importancia"
        android:layout_below="@+id/view15"
        />
    <Spinner
       android:id="@+id/sel_importanciaproc"
       android:layout_width="150dp"
       android:layout_height="wrap_content"
       android:layout_below="@+id/label_hi"
        />

    <View android:id="@+id/view16"
       android:layout_width="fill_parent"
       android:layout_height="40dp"
        android:layout_below="@+id/sel_importanciaproc"
        android:background="#FFFFFF"/>

    <TextView
        android:id="@+id/label_i"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/view16"
```

```
        android:text="Pantalla:"/>
    <Spinner
       android:id="@+id/sel_pan"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_below="@+id/label_i"
        />

    <ImageView android:id="@+id/imageView9"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_toRightOf="@+id/sel_pan"
        android:layout_below="@+id/view16"
        android:src="@drawable/screen" />

    <View android:id="@+id/view17"
       android:layout_width="fill_parent"
       android:layout_height="20dp"
        android:layout_below="@+id/sel_pan"
        android:background="#FFFFFF"/>


    <TextView
        android:id="@+id/label_ii"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Importancia"
        android:layout_below="@+id/view17"/>
    <Spinner
       android:id="@+id/sel_importanciapan"
       android:layout_width="150dp"
       android:layout_height="wrap_content"
       android:layout_below="@+id/label_ii"
        />

    <View android:id="@+id/view18"
       android:layout_width="fill_parent"
       android:layout_height="40dp"
        android:layout_below="@+id/sel_importanciapan"
        android:background="#FFFFFF"/>
    <TextView
        android:id="@+id/label_j"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/view18"
        android:text="Precio:"/>
    <Spinner
```

```
    android:id="@+id/sel_prec"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/label_j"
     />
<ImageView android:id="@+id/imageView10"
     android:layout_width="50dp"
     android:layout_height="50dp"
     android:layout_toRightOf="@+id/sel_prec"
     android:layout_below="@+id/view18"
     android:src="@drawable/price" />

<View android:id="@+id/view19"
    android:layout_width="fill_parent"
    android:layout_height="20dp"
     android:layout_below="@+id/sel_prec"
     android:background="#FFFFFF"/>


<TextView
     android:id="@+id/label_ji"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="Importancia"
     android:layout_below="@+id/view19"
     />
<Spinner
    android:id="@+id/sel_importanciaprec"
    android:layout_width="150dp"
    android:layout_height="wrap_content"
    android:layout_below="@+id/label_ji"
     />

<View android:id="@+id/view20"
    android:layout_width="fill_parent"
    android:layout_height="40dp"
     android:layout_below="@+id/sel_importanciaprec"
     android:background="#FFFFFF"/>


<TextView
     android:id="@+id/label_k"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:layout_below="@id/view20"
     android:text="Popularidad:"/>
<Spinner
```

```
        android:id="@+id/sel_pop"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_below="@+id/label_k"
         />
    <View android:id="@+id/view25"
        android:layout_width="50dp"
        android:layout_height="50dp"
         android:layout_toRightOf="@+id/sel_pop"
         android:layout_below="@+id/view20"
         android:background="#FFFFFF"/>

    <ImageView android:id="@+id/imageView11"
         android:layout_width="50dp"
         android:layout_height="50dp"
         android:layout_toRightOf="@+id/view25"
         android:layout_below="@+id/view20"
         android:src="@drawable/pop" />

      <View android:id="@+id/view21"
        android:layout_width="fill_parent"
        android:layout_height="20dp"
         android:layout_below="@+id/sel_pop"
         android:background="#FFFFFF"/>


      <TextView
         android:id="@+id/label_ki"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="Importancia"
         android:layout_below="@+id/view21"/>
      <Spinner
        android:id="@+id/sel_importanciapop"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_below="@+id/label_ki"
         />

      <View android:id="@+id/view22"
        android:layout_width="fill_parent"
        android:layout_height="40dp"
         android:layout_below="@+id/sel_importanciapop"
         android:background="#FFFFFF"/>
```

```
  <Button
     android:id="@+id/button1"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
      android:layout_below="@+id/view22"
     android:text="Buscar" />

  <TextView
      android:id="@+id/Resultados"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_below="@id/button1"
      android:text="Selecciona"/>
<TextView
      android:id="@+id/Prueba"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_below="@id/Resultados"
      android:text=" "/>

<TextView
      android:id="@+id/tREST"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_below="@id/Prueba"
      android:text=" "/>

<TextView
      android:id="@+id/tRESTi"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_below="@id/tREST"
      android:text=" "/>
      </RelativeLayout>

   </LinearLayout>

</ScrollView>
```

Paquete modelo

Producto

```
package com.modelo;

// TODO: Auto-generated Javadoc
/**
 *   Clase Producto.
 */
```

```
public class Producto {

	/** id el equipo de computo. */
	private Integer idProducto;

	/**
	 * Obtiene  id producto.
	 *
	 * @return   id producto
	 */
	public Integer getIdProducto() {
		return idProducto;
	}

	/**
	 * Define   id producto.
	 *
	 * @param idProducto   new id producto
	 */
	public void setIdProducto(Integer idProducto) {
		this.idProducto = idProducto;
	}

	/**
	 * Obtiene   nombre.
	 *
	 * @return   nombre
	 */
	public String getNombre() {
		return nombre;
	}

	/**
	 * Define   nombre.
	 *
	 * @param nombre   new nombre
	 */
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}

	/**
	 * Obtiene    color.
	 *
	 * @return   color
	 */
	public String getColor() {
```

```
        return color;
    }

    /**
     * Define   color.
     *
     * @param color   new color
     */
    public void setColor(String color) {
        this.color = color;
    }

    /**
     * Obtiene    ram.
     *
     * @return   ram
     */
    public String getRam() {
        return ram;
    }

    /**
     * Define   ram.
     *
     * @param ram   new ram
     */
    public void setRam(String ram) {
        this.ram = ram;
    }

    /**
     * Obtiene    disco duro.
     *
     * @return   disco duro
     */
    public String getDiscoduro() {
        return discoduro;
    }

    /**
     * Define   disco duro.
     *
     * @param discoduro   new disco duro
     */
    public void setDiscoduro(String discoduro) {
        this.discoduro = discoduro;
    }
```

```
/**
 * Obtiene   dispositivo optico.
 *
 * @return   dispositivooptico
 */
public String getDispositivooptico() {
        return dispositivooptico;
}

/**
 * Define   dispositivo optico.
 *
 * @param dispositivooptico   new dispositivooptico
 */
public void setDispositivooptico(String dispositivooptico) {
        this.dispositivooptico = dispositivooptico;
}

/**
 * Obtiene    precio.
 *
 * @return   precio
 */
public double getPrecio() {
        return precio;
}

/**
 * Define   precio.
 *
 * @param precio   new precio
 */
public void setPrecio(double precio) {
        this.precio = precio;
}

/**
 * Obtiene   pantalla.
 *
 * @return   pantalla
 */
public double getPantalla() {
        return pantalla;
}

/**
```

```
     * Define   pantalla.
     *
     * @param pantalla   new pantalla
     */
    public void setPantalla(double pantalla) {
            this.pantalla = pantalla;
    }

    /**
     * Obtiene   marca.
     *
     * @return   marca
     */
    public String getMarca() {
            return marca;
    }

    /**
     * Define   marca.
     *
     * @param marca   new marca
     */
    public void setMarca(String marca) {
            this.marca = marca;
    }

    /**
     * Obtiene   tipo.
     *
     * @return   tipo
     */
    public String getTipo() {
            return tipo;
    }

    /**
     * Define   tipo.
     *
     * @param tipo   new tipo
     */
    public void setTipo(String tipo) {
            this.tipo = tipo;
    }

    /**
     * Obtiene   procesador.
     *
```

```
     * @return   procesador
     */
    public String getProcesador() {
            return procesador;
    }

    /**
     * Define   procesador.
     *
     * @param procesador   new procesador
     */
    public void setProcesador(String procesador) {
            this.procesador = procesador;
    }

    /**
     * Obtiene   sistema operativo.
     *
     * @return   sistema operativo
     */
    public String getSo() {
            return so;
    }

    /**
     * Define   so.
     *
     * @param so   new so
     */
    public void setSo(String so) {
            this.so = so;
    }

    /**
     * Obtiene   popularidad.
     *
     * @return   popularidad
     */
    public int getPopularidad() {
            return popularidad;
    }

    /**
     * Define   popularidad.
     *
     * @param popularidad   new popularidad
     */
```

```
	public void setPopularidad(int popularidad) {
		this.popularidad = popularidad;
	}

	/** nombre. */
	private String nombre;

	/** color. */
	private String color;

	/** ram. */
	private String ram;

	/** disco duro. */
	private String discoduro;

	/** dispositivo optico. */
	private String dispositivooptico;

	/** precio. */
	private double precio;

	/**  pantalla. */
	private double pantalla;

	/**  marca. */
	private String marca;

	/**  tipo. */
	private String tipo;

	/**  procesador. */
	private String procesador;

	/**  sistema operativo. */
	private String so;

	/** popularidad. */
	private int popularidad;
}
```

ProductoEvaluado

```
package com.modelo;

// TODO: Auto-generated Javadoc
/**
 *  Clase Producto Evaluado.
```

```
 */
public class ProductoEvaluado {

	/**  id producto. */
	private Integer idProducto;

	/**  nombre. */
	private String nombre;

	/**  color. */
	private String color;

	/**  ram. */
	private String ram;

	/**  discoduro. */
	private String discoduro;

	/**  dispositivooptico. */
	private String dispositivooptico;

	/**  precio. */
	private double precio;

	/**  pantalla. */
	private double pantalla;

	/**  marca. */
	private String marca;

	/**  tipo. */
	private String tipo;

	/**  procesador. */
	private String procesador;

	/**  so. */
	private String so;

	/**  popularidad. */
	private int popularidad;

	/**  evaluacion. */
	private double evaluacion;

	   /**
	 * Instancia un producto evaluado.
```

```
     *
     * @param p producto
     * @param ev  evauacion
     */
    public ProductoEvaluado(Producto p, double ev) {
        this.idProducto = p.getIdProducto();
        this.nombre = p.getNombre();
        this.color = p.getColor();
        this.ram = p.getRam();
        this.discoduro = p.getDiscoduro();
        this.dispositivooptico = p.getDispositivooptico();
        this.precio = p.getPrecio();
        this.pantalla = p.getPantalla();
        this.marca = p.getMarca();
        this.tipo = p.getMarca();
        this.procesador = p.getProcesador();
        this.so = p.getSo();
        this.popularidad = p.getPopularidad();
        this.evaluacion = ev;
    }

    /**
     * Instancia un producto evaluado.
     *
     * @param idProducto  id producto
     * @param nombre  nombre
     * @param color  color
     * @param ram  ram
     * @param discoduro  discoduro
     * @param dispositivooptico  dispositivooptico
     * @param precio  precio
     * @param pantalla  pantalla
     * @param marca  marca
     * @param tipo  tipo
     * @param procesador  procesador
     * @param so  so
     * @param popularidad  popularidad
     * @param evaluacion  evaluacion
     */
    public ProductoEvaluado(Integer idProducto, String nombre, String
color, String ram, String discoduro, String dispositivooptico, double precio, double
pantalla, String marca, String tipo, String procesador, String so, int popularidad, double
evaluacion) {
        this.idProducto = idProducto;
        this.nombre = nombre;
        this.color = color;
        this.ram = ram;
```

```
        this.discoduro = discoduro;
        this.dispositivooptico = dispositivooptico;
        this.precio = precio;
        this.pantalla = pantalla;
        this.marca = marca;
        this.tipo = tipo;
        this.procesador = procesador;
        this.so = so;
        this.popularidad = popularidad;
        this.evaluacion = evaluacion;
    }

    /**
 * Instancia un producto evaluado.
 */
public ProductoEvaluado() {
    }

    /**
 * Define  id producto.
 *
 * @param idProducto   id producto
 */
public void setIdProducto(Integer idProducto) {
        this.idProducto = idProducto;
    }

    /**
 * Define  nombre.
 *
 * @param nombre   nombre
 */
public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    /**
 * Define  color.
 *
 * @param color   color
 */
public void setColor(String color) {
        this.color = color;
    }

    /**
 * Define  ram.
```

```
     *
     * @param ram   ram
     */
    public void setRam(String ram) {
        this.ram = ram;
    }

    /**
     * Define  discoduro.
     *
     * @param discoduro   discoduro
     */
    public void setDiscoduro(String discoduro) {
        this.discoduro = discoduro;
    }

    /**
     * Define  dispositivooptico.
     *
     * @param dispositivooptico   dispositivooptico
     */
    public void setDispositivooptico(String dispositivooptico) {
        this.dispositivooptico = dispositivooptico;
    }

    /**
     * Define  precio.
     *
     * @param precio   precio
     */
    public void setPrecio(double precio) {
        this.precio = precio;
    }

    /**
     * Define  pantalla.
     *
     * @param pantalla   pantalla
     */
    public void setPantalla(double pantalla) {
        this.pantalla = pantalla;
    }

    /**
     * Define  marca.
     *
     * @param marca   marca
```

```
 */
public void setMarca(String marca) {
        this.marca = marca;
    }

    /**
 * Define  tipo.
 *
 * @param tipo   tipo
 */
public void setTipo(String tipo) {
        this.tipo = tipo;
    }

    /**
 * Define  procesador.
 *
 * @param procesador   procesador
 */
public void setProcesador(String procesador) {
        this.procesador = procesador;
    }

    /**
 * Define  so.
 *
 * @param so   so
 */
public void Defineo(String so) {
        this.so = so;
    }

    /**
 * Define  popularidad.
 *
 * @param popularidad   popularidad
 */
public void setPopularidad(int popularidad) {
        this.popularidad = popularidad;
    }

    /**
 * Define  evaluacion.
 *
 * @param evaluacion   evaluacion
 */
public void setEvaluacion(double evaluacion) {
```

```
        this.evaluacion = evaluacion;
    }

    /**
 * Obtiene  id producto.
 *
 * @return  id producto
 */
public Integer getIdProducto() {
        return idProducto;
    }

    /**
 * Obtiene  nombre.
 *
 * @return  nombre
 */
public String getNombre() {
        return nombre;
    }

    /**
 * Obtiene  color.
 *
 * @return  color
 */
public String getColor() {
        return color;
    }

    /**
 * Obtiene  ram.
 *
 * @return  ram
 */
public String getRam() {
        return ram;
    }

    /**
 * Obtiene  discoduro.
 *
 * @return  discoduro
 */
public String getDiscoduro() {
        return discoduro;
    }
```

```
    /**
 * Obtiene  dispositivooptico.
 *
 * @return  dispositivooptico
 */
public String getDispositivooptico() {
        return dispositivooptico;
    }

    /**
 * Obtiene  precio.
 *
 * @return  precio
 */
public double getPrecio() {
        return precio;
    }

    /**
 * Obtiene  pantalla.
 *
 * @return  pantalla
 */
public double getPantalla() {
        return pantalla;
    }

    /**
 * Obtiene  marca.
 *
 * @return  marca
 */
public String getMarca() {
        return marca;
    }

    /**
 * Obtiene  tipo.
 *
 * @return  tipo
 */
public String getTipo() {
        return tipo;
    }

    /**
```

```
     * Obtiene  procesador.
     *
     * @return  procesador
     */
    public String getProcesador() {
        return procesador;
      }

      /**
     * Obtiene  so.
     *
     * @return  so
     */
    public String Obtieneo() {
        return so;
      }

      /**
     * Obtiene  popularidad.
     *
     * @return  popularidad
     */
    public int getPopularidad() {
        return popularidad;
      }

      /**
     * Obtiene  evaluacion.
     *
     * @return  evaluacion
     */
    public double getEvaluacion() {
        return evaluacion;
      }
}
```

Ponderacion

```
package com.modelo;

// TODO: Auto-generated Javadoc
/**
 * Clase Ponderacion.
 */
public class Ponderacion {

    /**  importancia. */
    private int importancia;
```

```
	/**  Caracteristica. */
	private String Caracteristica;

	/**  Descripcion. */
	private String Descripcion;

	/**
	 * Instanciat una  ponderacion.
	 *
	 * @param importancia  importancia
	 * @param Caracteristica  caracteristica
	 * @param Descripcion  descripcion
	 */
	public Ponderacion(int importancia, String Caracteristica, String
Descripcion) {
		this.importancia = importancia;
		this.Caracteristica = Caracteristica;
		this.Descripcion = Descripcion;
	}

	/**
	 * Obtiene  importancia.
	 *
	 * @return  importancia
	 */
	public int getImportancia() {
		return importancia;
	}

	/**
	 * Sets  importancia.
	 *
	 * @param importancia   importancia
	 */
	public void setImportancia(int importancia) {
		this.importancia = importancia;
	}

	/**
	 * Obtiene  caracteristica.
	 *
	 * @return  caracteristica
	 */
	public String getCaracteristica() {
		return Caracteristica;
	}
```

```
	/**
	 * Sets  caracteristica.
	 *
	 * @param Caracteristica   caracteristica
	 */
	public void setCaracteristica(String Caracteristica) {
		this.Caracteristica = Caracteristica;
	}

	/**
	 * Obtiene  descripcion.
	 *
	 * @return  descripcion
	 */
	public String getDescripcion() {
		return Descripcion;
	}

	/**
	 * Sets  descripcion.
	 *
	 * @param Descripcion   descripcion
	 */
	public void setDescripcion(String Descripcion) {
		this.Descripcion = Descripcion;
	}
}
```

Paquete controlador
Algoritmo

```
package com.controlador;

import java.util.ArrayList;
import java.util.Collections;
import static java.lang.Math.pow;
import com.modelo.*;

// TODO: Auto-generated Javadoc
/**
 *  Clase Algoritmo.
 */
public class Algoritmo {

	/**
	 * Lsp.
	 *
```

```
                 * @param prod producto
                 * @param u  ponderacion
                 * @return evaluacion
                 */
                public double LSP(Producto prod, ArrayList<Ponderacion> u) {
            double resultado = 0; //Variable para regresar el resultado final del
algoritmo
            double[] e = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
            double [] l = {0,0,0,0,0,0,0};
            double [] w ={0,0,0,0,0,0,0,0,0,0,0,0};
            Collections.sort(u, new ComparaPonderacion());//Aqui ordenamos las
caracteristicas que da el usuario
            //En esta seccion comparamos las caracteristicas que el usuario nos solicitó
con las de los productos
            for (int i = 0; i < u.size() - 1; i++) {
                if (u.get(i).getCaracteristica().equals("popularidad")) {
                    e[i] = comparaPopularidad(prod.getPopularidad(),
Integer.parseInt(u.get(i).getDescripcion()));
                    System.out.println(prod.getPopularidad() + " = " +
u.get(i).getDescripcion() + " ----> " + e[i]);
                } else if (u.get(i).getCaracteristica().equals("precio")) {

                    e[i] = comparaPrecio(prod.getPrecio(),
Double.parseDouble(u.get(i).getDescripcion()));
                    System.out.println(prod.getPrecio() + " = " + u.get(i).getDescripcion()
+ " ----> " + e[i]);

                } else if (u.get(i).getCaracteristica().equals("pantalla")) {
                    e[i] = comparaPantalla(prod.getPantalla(),
Double.parseDouble(u.get(i).getDescripcion()));
                    System.out.println(prod.getPantalla() + " = " +
u.get(i).getDescripcion() + " ----> " + e[i]);

                } else if (u.get(i).getCaracteristica().equals("color")) {
                    e[i] = comparaCaracteristicas(prod.getColor(),
u.get(i).getDescripcion());
                    System.out.println(prod.getColor() + " = " + u.get(i).getDescripcion()
+ " ----> " + e[i]);

                } else if (u.get(i).getCaracteristica().equals("tipo")) {
                    e[i] = comparaCaracteristicas(prod.getTipo(),
u.get(i).getDescripcion());
                    System.out.println(prod.getTipo() + " = " + u.get(i).getDescripcion() +
" ----> " + e[i]);

                } else if (u.get(i).getCaracteristica().equals("dispositivoOptico")) {
                    e[i] = comparaCaracteristicas(prod.getDispositivooptico(),
```

```
u.get(i).getDescripcion());
                    System.out.println(prod.getDispositivooptico() + " = " +
u.get(i).getDescripcion() + " ----> " + e[i]);

                } else if (u.get(i).getCaracteristica().equals("procesador")) {
                    e[i] = comparaCaracteristicas(prod.getProcesador(),
u.get(i).getDescripcion());
                    System.out.println(prod.getProcesador() + " = " +
u.get(i).getDescripcion() + " ----> " + e[i]);

                } else if (u.get(i).getCaracteristica().equals("ram")) {
                    e[i] = comparaCaracteristicas(prod.getRam(),
u.get(i).getDescripcion());
                    System.out.println(prod.getRam() + " = " + u.get(i).getDescripcion() +
" ----> " + e[i]);

                } else if (u.get(i).getCaracteristica().equals("hdd")) {
                    e[i] = comparaCaracteristicas(prod.getDiscoduro(),
u.get(i).getDescripcion());
                    System.out.println(prod.getDiscoduro() + " = " +
u.get(i).getDescripcion() + " ----> " + e[i]);

                } else if (u.get(i).getCaracteristica().equals("marca")) {
                    e[i] = comparaCaracteristicas(prod.getMarca(),
u.get(i).getDescripcion());
                    System.out.println(prod.getMarca() + " = " + u.get(i).getDescripcion()
+ " ----> " + e[i]);

                } else if (u.get(i).getCaracteristica().equals("so")) {
                    e[i] = comparaCaracteristicas(prod.getSo(), u.get(i).getDescripcion());
                    System.out.println(prod.getSo() + " = " + u.get(i).getDescripcion() + "
----> " + e[i]);

                }

            }
            //Aqui asignaremos los pesos
            w[0]=0.6;
            w[1]=0.3;
            w[2]=0.1;
            w[3]=0.6;
            w[4]=0.3;
            w[5]=0.1;
            w[6]=0.6;
            w[7]=0.3;
            w[8]=0.1;
            w[9]=0.7;
```

```
    w[10]=0.3;

    //Multiplicando por r
    e[0] =  w[0]*pow(e[0],11.095);
    e[1] =  w[1]*pow(e[1],11.095);
    e[2] =  w[2]*pow(e[2],11.095);
    e[3] =  w[3]*pow(e[3],2.187);
    e[4] =  w[4]*pow(e[4],2.187);
    e[5] =  w[5]*pow(e[5],2.187);
    e[6] =  w[6]*pow(e[6],0.192);
    e[7] =  w[7]*pow(e[7],0.192);
    e[8] =  w[8]*pow(e[8],0.192);
    e[9] =  w[9]*pow(e[9],-3.510);
    e[10] = w[10]*pow(e[10],-3.510);

    //Arbol de operaciones
    l[0] =e[0]+e[1]+e[2];
    l[0] = pow (l[0],(1/11.095));
    l[1] =e[3]+e[4]+e[5];
    l[1] = pow (l[1],(1/2.187));
    l[2] =e[6]+e[7]+e[8];
    l[2] = pow (l[2],(1/0.192));
    l[3] =e[9]+pow(e[10],-3.510);
    l[3] = pow (l[3],(1/-3.510 ));
    l[4] = (w[9]*pow(l[0],9.521))+(w[10]*pow(l[1],9.521));
    l[4] = (pow(l[4],9.521));
    l[5] = (w[9]*pow(l[2],-3.510))+(w[10]*pow(l[3],-3.510));
    l[5] = (pow(l[5],-3.510));
    l[6] = (w[9]*pow(l[4],0.261))+(w[10]*pow(l[5],0.261));
    l[6] = (pow(l[6],0.261));
    resultado = l[6];

    return resultado;
}

/**
 * Compara caracteristicas.
 *
 * @param car1  caracteristica 1
 * @param car2  caracteristica 2
 * @return  double
 */
public double comparaCaracteristicas(String car1, String car2) {
    double comparacion = 0;
    if (car1.equals(car2)) {
        comparacion = 1;
    } else if ((car1.equals("gris") && car2.equals("plateado")) ||
```

```
(car2.equals("gris") && car1.equals("plateado"))) {
                    comparacion = 0.7;
                } else if ((car1.equals("rosa") && car2.equals("rojo")) ||
(car2.equals("rosa") && car1.equals("rojo"))) {
                    comparacion = 0.5;
                } else if ((car1.equals("netbook") && car2.equals("laptop")) ||
(car2.equals("netbook") && car1.equals("laptop"))) {
                    comparacion = 0.7;
                }
                return comparacion;
            }

            /**
             * Compara popularidad.
             *
             * @param pop1  popularidad 1
             * @param pop2  popularidad 2
             * @return  double
             */
            public double comparaPopularidad(int pop1, int pop2) {
                double compara = 0;
                if (pop1 == pop2) {
                    compara = 1;
                } else if ((pop1 == 1 && pop2 == 2) || (pop2 == 1 && pop1 == 2)) {
                    compara = 0.6;
                } else if ((pop1 == 3 && pop2 == 2) || (pop2 == 3 && pop1 == 2)) {
                    compara = 0.6;
                } else if ((pop1 == 3 && pop2 == 4) || (pop2 == 3 && pop1 == 4)) {
                    compara = 0.6;
                } else if ((pop1 == 5 && pop2 == 4) || (pop2 == 5 && pop1 == 4)) {
                    compara = 0.6;
                }

                return compara;
            }

            /**
             * Compara precio.
             *
             * @param precio1  precio 1
             * @param precio2  precio 2
             * @return  double
             */
            public double comparaPrecio(double precio1, double precio2) {
                double compara = 0;
                if ((precio2 / precio1) == 1) {
                    compara = 1;
```

```
            }
            else if(precio1==5000 && precio2<5001){
               compara= 1;
            }
            else if(precio1==10000 && precio2>5000 && precio2<10001){
               compara= 1;
            }
            else if(precio1==20000 && precio2>10000 && precio2<20001){
               compara= 1;
            }
            else if(precio1==30000 && precio2>20000){
               compara= 1;
            }

            return compara;
         }

         /**
          * Compara pantalla.
          *
          * @param pant1  pantalla 1
          * @param pant2  pantalla 2
          * @return  double
          */
         public double comparaPantalla(double pant1, double pant2) {
            double compara = 0;
            if (pant1 / pant2 == 1) {
               compara = 1;
            }
            return compara;
         }
      }
```

ComparaPonderacion

```
      package com.controlador;

      import java.util.Comparator;

      import com.modelo.Ponderacion;

      // TODO: Auto-generated Javadoc
      /**
       * Clase Compara Ponderacion.
       */
      public class ComparaPonderacion implements Comparator<Ponderacion>{

         /* (non-Javadoc)
```

```
             * @see java.util.Comparator#compare(java.lang.Object, java.lang.Object)
             */
            public int compare(Ponderacion t, Ponderacion t1) {
               return new Integer(t.getImportancia()).compareTo(new
Integer(t1.getImportancia()));
                        }
        }
```

MainActivity

```
        package com.controlador;

        import java.io.IOException;
        import java.io.InputStream;
        import java.util.ArrayList;
        import java.util.Collections;
        import android.support.v7.app.ActionBarActivity;
        import android.os.AsyncTask;
        import android.os.Bundle;
        import android.view.Menu;
        import android.view.MenuItem;
        import android.view.View;
        import android.view.View.OnClickListener;
        import android.widget.AdapterView;
        import android.widget.ArrayAdapter;
        import android.widget.Button;
        import android.widget.Spinner;
        import android.widget.TextView;
        import android.widget.AdapterView.OnItemSelectedListener;
        import org.apache.http.HttpEntity;
        import org.apache.http.HttpResponse;
        import org.apache.http.client.HttpClient;
        import org.apache.http.client.methods.HttpGet;
        import org.apache.http.impl.client.DefaultHttpClient;
        import org.apache.http.protocol.BasicHttpContext;
        import org.apache.http.protocol.HttpContext;
        import org.apache.http.util.EntityUtils;
        import org.json.JSONArray;
        import org.json.JSONException;
        import org.json.JSONObject;
        import com.example.compras.R;
        import com.modelo.Ponderacion;
        import com.modelo.Producto;
        import com.modelo.ProductoEvaluado;


        // TODO: Auto-generated Javadoc
        /**
```

```
 *  Class MainActivity.
 */
public class MainActivity extends ActionBarActivity {

	/**  button buscar. */
	Button buttonJugar = null;

	/**  sistema operativo. */
	String so;

	/**  color. */
	String color;

	/**  marca. */
	String marca;

	/**  precio. */
	String precio;

	/**  procesador. */
	String procesador;

	/**  popularidad. */
	String popularidad;

	/**  pantalla. */
	String pantalla;

	/**  dispositivo optico. */
	String doptico;

	/**  hdd. */
	String hdd;

	/**  ram. */
	String ram;

	/**  tipo. */
	String tipo;

	/**  sistema operativo importancia. */
	int soi;

	/**  color importancia. */
	int colori;

	/**  marca importancia. */
```

```
int marcai;

/**  precio importancia. */
int precioi;

/**  procesador importancia. */
int procesadori;

/**  popularidad importancia. */
int popularidadi;

/**  pantalla importancia. */
int pantallai;

/**  doptico importancia. */
int dopticoi;

/**  disco duro importancia. */
int hddi;

/**  ram importancia. */
int rami;

/**  tipo importancia. */
int tipoi;

/**  sistema operativo importancia. */
String sois;

/**  color importancia. */
String coloris;

/**  marca importancia. */
String marcais;

/**  precio importancia. */
String preciois;

/**  procesador importancia. */
String procesadoris;

/**  popularidad importancia. */
String popularidadis;

/**  pantalla importancia. */
String pantallais;
```

```
		/**  dispositivo optico importancia. */
		String dopticois;

		/**  disco duro importancia. */
		String hddis;

		/**  ram importancia. */
		String ramis;

		/**  tipo importancia. */
		String tipois;

		/* (non-Javadoc)
		 * @see
android.support.v7.app.ActionBarActivity#onCreate(android.os.Bundle)
		 */
		@Override
		protected void onCreate(Bundle savedInstanceState) {
			super.onCreate(savedInstanceState);
			setContentView(R.layout.activity_main);
			//Declaración de los Spinners
					Spinner spinner_color = (Spinner)
findViewById(R.id.sel_color);
					Spinner spinner_marca = (Spinner)
findViewById(R.id.sel_marca);
					Spinner spinner_prec= (Spinner)
findViewById(R.id.sel_prec);
					Spinner spinner_proc = (Spinner)
findViewById(R.id.sel_proc);
					Spinner spinner_pop = (Spinner)
findViewById(R.id.sel_pop);
					Spinner spinner_pan = (Spinner)
findViewById(R.id.sel_pan);
					Spinner spinner_do = (Spinner)
findViewById(R.id.sel_do);
					Spinner spinner_hdd = (Spinner)
findViewById(R.id.sel_hdd);
					Spinner spinner_ram = (Spinner)
findViewById(R.id.sel_ram);
					Spinner spinner_tip = (Spinner)
findViewById(R.id.sel_tip);
					Spinner spinner_so = (Spinner)
findViewById(R.id.sel_so);
					//Spinners de Importancia
					Spinner spinner_colori = (Spinner)
findViewById(R.id.sel_importancia);
					Spinner spinner_marcai = (Spinner)
```

```
findViewById(R.id.sel_importanciam);
                                        Spinner spinner_preci= (Spinner)
findViewById(R.id.sel_importanciaprec);
                                        Spinner spinner_proci = (Spinner)
findViewById(R.id.sel_importanciaproc);
                                        Spinner spinner_popi = (Spinner)
findViewById(R.id.sel_importanciapop);
                                        Spinner spinner_pani = (Spinner)
findViewById(R.id.sel_importanciapan);
                                        Spinner spinner_doi = (Spinner)
findViewById(R.id.sel_importanciadop);
                                        Spinner spinner_hddi = (Spinner)
findViewById(R.id.sel_importanciahdd);
                                        Spinner spinner_rami = (Spinner)
findViewById(R.id.sel_importanciaram);
                                        Spinner spinner_tipi = (Spinner)
findViewById(R.id.sel_importanciatip);
                                        Spinner spinner_soi = (Spinner)
findViewById(R.id.sel_importanciaso);

                                        //Llenado de los adapters de características
                                        ArrayAdapter spinner_adapter =
ArrayAdapter.createFromResource( this, R.array.colores ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter2 =
ArrayAdapter.createFromResource( this, R.array.marca ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter3 =
ArrayAdapter.createFromResource( this, R.array.DO ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter4 =
ArrayAdapter.createFromResource( this, R.array.HDD ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter5 =
ArrayAdapter.createFromResource( this, R.array.pantalla ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter6 =
ArrayAdapter.createFromResource( this, R.array.Popularidad ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter7 =
ArrayAdapter.createFromResource( this, R.array.precio ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter8 =
ArrayAdapter.createFromResource( this, R.array.Procesador ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter9 =
ArrayAdapter.createFromResource( this, R.array.RAM ,
```

```
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter10 =
ArrayAdapter.createFromResource( this, R.array.SO ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter11 =
ArrayAdapter.createFromResource( this, R.array.tipo ,
android.R.layout.simple_spinner_item);
                                        //Llenado de los adapters de Importancia
                                        ArrayAdapter spinner_adapter12 =
ArrayAdapter.createFromResource( this, R.array.importancia ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter13 =
ArrayAdapter.createFromResource( this, R.array.importancia ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter14 =
ArrayAdapter.createFromResource( this, R.array.importancia ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter15 =
ArrayAdapter.createFromResource( this, R.array.importancia ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter16 =
ArrayAdapter.createFromResource( this, R.array.importancia ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter17 =
ArrayAdapter.createFromResource( this, R.array.importancia ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter18 =
ArrayAdapter.createFromResource( this, R.array.importancia ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter19 =
ArrayAdapter.createFromResource( this, R.array.importancia ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter20 =
ArrayAdapter.createFromResource( this, R.array.importancia ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter21 =
ArrayAdapter.createFromResource( this, R.array.importancia ,
android.R.layout.simple_spinner_item);
                                        ArrayAdapter spinner_adapter22 =
ArrayAdapter.createFromResource( this, R.array.importancia ,
android.R.layout.simple_spinner_item);



        spinner_adapter.setDropDownViewResource(android.R.layout.simple_spinner_
dropdown_item);
```

```
        spinner_adapter2.setDropDownViewResource(android.R.layout.simple_spinner
_dropdown_item);

        spinner_adapter3.setDropDownViewResource(android.R.layout.simple_spinner
_dropdown_item);

        spinner_adapter4.setDropDownViewResource(android.R.layout.simple_spinner
_dropdown_item);

        spinner_adapter5.setDropDownViewResource(android.R.layout.simple_spinner
_dropdown_item);

        spinner_adapter6.setDropDownViewResource(android.R.layout.simple_spinner
_dropdown_item);

        spinner_adapter7.setDropDownViewResource(android.R.layout.simple_spinner
_dropdown_item);

        spinner_adapter8.setDropDownViewResource(android.R.layout.simple_spinner
_dropdown_item);

        spinner_adapter9.setDropDownViewResource(android.R.layout.simple_spinner
_dropdown_item);

        spinner_adapter10.setDropDownViewResource(android.R.layout.simple_spinne
r_dropdown_item);

        spinner_adapter11.setDropDownViewResource(android.R.layout.simple_spinne
r_dropdown_item);

        spinner_adapter12.setDropDownViewResource(android.R.layout.simple_spinne
r_dropdown_item);

        spinner_adapter13.setDropDownViewResource(android.R.layout.simple_spinne
r_dropdown_item);

        spinner_adapter14.setDropDownViewResource(android.R.layout.simple_spinne
r_dropdown_item);

        spinner_adapter15.setDropDownViewResource(android.R.layout.simple_spinne
r_dropdown_item);

        spinner_adapter16.setDropDownViewResource(android.R.layout.simple_spinne
r_dropdown_item);

        spinner_adapter17.setDropDownViewResource(android.R.layout.simple_spinne
r_dropdown_item);
```

```
        spinner_adapter18.setDropDownViewResource(android.R.layout.simple_spinne
r_dropdown_item);

        spinner_adapter19.setDropDownViewResource(android.R.layout.simple_spinne
r_dropdown_item);

        spinner_adapter20.setDropDownViewResource(android.R.layout.simple_spinne
r_dropdown_item);

        spinner_adapter21.setDropDownViewResource(android.R.layout.simple_spinne
r_dropdown_item);

        spinner_adapter22.setDropDownViewResource(android.R.layout.simple_spinne
r_dropdown_item);

                                        //Asignación de Adapters a los Spinners
                                        spinner_color.setAdapter(spinner_adapter);
                                        spinner_marca.setAdapter(spinner_adapter2);
                                        spinner_do.setAdapter(spinner_adapter3);
                                        spinner_hdd.setAdapter(spinner_adapter4);
                                        spinner_pan.setAdapter(spinner_adapter5);
                                        spinner_pop.setAdapter(spinner_adapter6);
                                        spinner_prec.setAdapter(spinner_adapter7);
                                        spinner_proc.setAdapter(spinner_adapter8);
                                        spinner_ram.setAdapter(spinner_adapter9);
                                        spinner_so.setAdapter(spinner_adapter10);
                                        spinner_tip.setAdapter(spinner_adapter11);
                                        spinner_colori.setAdapter(spinner_adapter12);
                                        spinner_marcai.setAdapter(spinner_adapter13);
                                        spinner_preci.setAdapter(spinner_adapter14);
                                        spinner_proci.setAdapter(spinner_adapter15);
                                        spinner_popi.setAdapter(spinner_adapter16);
                                        spinner_pani.setAdapter(spinner_adapter17);
                                        spinner_doi.setAdapter(spinner_adapter18);
                                        spinner_hddi.setAdapter(spinner_adapter19);
                                        spinner_rami.setAdapter(spinner_adapter20);
                                        spinner_tipi.setAdapter(spinner_adapter21);
                                        spinner_soi.setAdapter(spinner_adapter22);

                                                spinner_color.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){
                                                        color =
parentView.getSelectedItem().toString();
                                                }
```

```
                                        @Override
                                        public void
onNothingSelected(AdapterView<?> arg0) {
                                                // TODO Auto-generated method
stub

                                        }
                                }

                                                );

                                        spinner_colori.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){

        if(parentView.getSelectedItem().toString().equals("Imprescindible"))
                                                        colori = 1;
                                                        else
if(parentView.getSelectedItem().toString().equals("Necesario"))
                                                        colori = 2;
                                                        else
if(parentView.getSelectedItem().toString().equals("Me gustaria"))
                                                        colori = 3;
                                                        else
if(parentView.getSelectedItem().toString().equals("No necesario"))
                                                        colori = 4;
                                                        else
if(parentView.getSelectedItem().toString().equals("Prescindible"))
                                                        colori = 5;
                                                        coloris
=parentView.getSelectedItem().toString();
                                                }

                                                @Override
                                                public void
onNothingSelected(AdapterView<?> arg0) {
                                                        // TODO Auto-generated
method stub

                                                }
                                        }

                                                        );

                                        spinner_marca.setOnItemSelectedListener(
```

```
new OnItemSelectedListener(){
                                                    public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){
                                                            marca =
parentView.getSelectedItem().toString();
                                                    }

                                                    @Override
                                                    public void
onNothingSelected(AdapterView<?> arg0) {
                                                            // TODO Auto-generated
method stub

                                                    }
                                            }

                                                            );

                                            spinner_marcai.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                            public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){

        if(parentView.getSelectedItem().toString().equals("Imprescindible"))
                                                                    marcai = 1;
                                                                    else
if(parentView.getSelectedItem().toString().equals("Necesario"))
                                                                            marcai
= 2;
                                                                    else
if(parentView.getSelectedItem().toString().equals("Me gustaria"))
                                                                            marcai
= 3;
                                                                    else
if(parentView.getSelectedItem().toString().equals("No necesario"))
                                                                            marcai
= 4;
                                                                    else
if(parentView.getSelectedItem().toString().equals("Prescindible"))
                                                                            marcai
= 5;
                                                                        marcais
=parentView.getSelectedItem().toString();
                                                            }

                                                            @Override
                                                            public void
```

```
onNothingSelected(AdapterView<?> arg0) {
                                                        // TODO Auto-
generated method stub

                                                    }
                                                }

                                                        );

                                            spinner_do.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){
                                                    doptico =
parentView.getSelectedItem().toString();
                                                }

                                                @Override
                                                public void
onNothingSelected(AdapterView<?> arg0) {
                                                    // TODO Auto-generated
method stub

                                                }
                                            }

                                                        );

                                            spinner_doi.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                    public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){

        if(parentView.getSelectedItem().toString().equals("Imprescindible"))
                                                                dopticoi = 1;
                                                                else
if(parentView.getSelectedItem().toString().equals("Necesario"))

        dopticoi = 2;
                                                                else
if(parentView.getSelectedItem().toString().equals("Me gustaria"))

        dopticoi = 3;
                                                                else
if(parentView.getSelectedItem().toString().equals("No necesario"))

        dopticoi = 4;
```

```
                                                                    else
if(parentView.getSelectedItem().toString().equals("Prescindible"))

        dopticoi = 5;
                                                                        dopticois
=parentView.getSelectedItem().toString();
                                                                }

                                                                @Override
                                                                public void
onNothingSelected(AdapterView<?> arg0) {
                                                                        // TODO Auto-
generated method stub

                                                                }
                                                        }

                                                                );
                                                spinner_hdd.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                        public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){
                                                                hdd =
parentView.getSelectedItem().toString();
                                                        }

                                                        @Override
                                                        public void
onNothingSelected(AdapterView<?> arg0) {
                                                                // TODO Auto-generated
method stub

                                                        }
                                                }

                                                                );

                                                spinner_hddi.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                                public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){

        if(parentView.getSelectedItem().toString().equals("Imprescindible"))
                                                                        hddi = 1;
                                                                        else
if(parentView.getSelectedItem().toString().equals("Necesario"))
                                                                    hddi =
```

```
2;
                                                                    else
if(parentView.getSelectedItem().toString().equals("Me gustaria"))
                                                                        hddi =
3;
                                                                    else
if(parentView.getSelectedItem().toString().equals("No necesario"))
                                                                        hddi =
4;
                                                                    else
if(parentView.getSelectedItem().toString().equals("Prescindible"))
                                                                        hddi =
5;
                                                                hddis
=parentView.getSelectedItem().toString();
                                                            }

                                                            @Override
                                                            public void
onNothingSelected(AdapterView<?> arg0) {
                                                                // TODO Auto-
generated method stub

                                                            }
                                                        }

                                                                );
                                                    spinner_pan.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                        public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){
                                                            pantalla =
parentView.getSelectedItem().toString();
                                                        }

                                                        @Override
                                                        public void
onNothingSelected(AdapterView<?> arg0) {
                                                            // TODO Auto-generated
method stub

                                                        }
                                                    }

                                                            );

                                                    spinner_pani.setOnItemSelectedListener(
```

```
new OnItemSelectedListener(){
                                                    public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){

        if(parentView.getSelectedItem().toString().equals("Imprescindible"))
                                                                pantallai = 1;
                                                                else
if(parentView.getSelectedItem().toString().equals("Necesario"))

        pantallai = 2;
                                                                else
if(parentView.getSelectedItem().toString().equals("Me gustaria"))

        pantallai = 3;
                                                                else
if(parentView.getSelectedItem().toString().equals("No necesario"))

        pantallai = 4;
                                                                else
if(parentView.getSelectedItem().toString().equals("Prescindible"))

        pantallai = 5;
                                                                pantallais
=parentView.getSelectedItem().toString();
                                                    }

                                                    @Override
                                                    public void
onNothingSelected(AdapterView<?> arg0) {
                                                            // TODO Auto-
generated method stub

                                                    }
                                                }

                                                    );
                                            spinner_pop.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){
                                                    popularidad =
parentView.getSelectedItem().toString();
                                                }

                                                @Override
                                                public void
onNothingSelected(AdapterView<?> arg0) {
```

```
                                                        // TODO Auto-generated
method stub

                                                }
                                        }

                                                );

                                        spinner_popi.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){

        if(parentView.getSelectedItem().toString().equals("Imprescindible"))
                                                        popularidadi
= 1;
                                                        else
if(parentView.getSelectedItem().toString().equals("Necesario"))

        popularidadi = 2;
                                                        else
if(parentView.getSelectedItem().toString().equals("Me gustaria"))

        popularidadi = 3;
                                                        else
if(parentView.getSelectedItem().toString().equals("No necesario"))

        popularidadi = 4;
                                                        else
if(parentView.getSelectedItem().toString().equals("Prescindible"))

        popularidadi = 5;
                                                                popularidadis
=parentView.getSelectedItem().toString();
                                                }

                                                @Override
                                                public void
onNothingSelected(AdapterView<?> arg0) {
                                                        // TODO Auto-
generated method stub

                                                }
                                        }

                                                );
                                        spinner_prec.setOnItemSelectedListener(
```

```
new OnItemSelectedListener(){
                                                    public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){
                                                            precio =
parentView.getSelectedItem().toString();
                                                    }

                                                    @Override
                                                    public void
onNothingSelected(AdapterView<?> arg0) {
                                                            // TODO Auto-generated
method stub

                                                    }
                                            }

                                                            );

                                            spinner_preci.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                            public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){

        if(parentView.getSelectedItem().toString().equals("Imprescindible"))
                                                                    precioi = 1;
                                                                    else
if(parentView.getSelectedItem().toString().equals("Necesario"))
                                                                            precioi
= 2;
                                                                    else
if(parentView.getSelectedItem().toString().equals("Me gustaria"))
                                                                            precioi
= 3;
                                                                    else
if(parentView.getSelectedItem().toString().equals("No necesario"))
                                                                            precioi
= 4;
                                                                    else
if(parentView.getSelectedItem().toString().equals("Prescindible"))
                                                                            precioi
= 5;
                                                                            preciois
=parentView.getSelectedItem().toString();
                                                            }

                                                            @Override
                                                            public void
```

```
onNothingSelected(AdapterView<?> arg0) {
                                                        // TODO Auto-
generated method stub

                                                    }
                                                }

                                                        );
                                            spinner_proc.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){
                                                    procesador =
parentView.getSelectedItem().toString();
                                                }

                                                @Override
                                                public void
onNothingSelected(AdapterView<?> arg0) {
                                                    // TODO Auto-generated
method stub

                                                }
                                            }

                                                    );

                                            spinner_proci.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                    public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){

        if(parentView.getSelectedItem().toString().equals("Imprescindible"))
                                                                procesadori =
1;
                                                                else
if(parentView.getSelectedItem().toString().equals("Necesario"))

        procesadori = 2;
                                                                else
if(parentView.getSelectedItem().toString().equals("Me gustaria"))

        procesadori = 3;
                                                                else
if(parentView.getSelectedItem().toString().equals("No necesario"))

        procesadori = 4;
```

```
                                                                    else
if(parentView.getSelectedItem().toString().equals("Prescindible"))

        procesadori = 5;
                                                                        procesadoris
=parentView.getSelectedItem().toString();
                                                                }

                                                                @Override
                                                                public void
onNothingSelected(AdapterView<?> arg0) {
                                                                        // TODO Auto-
generated method stub

                                                                }
                                                        }

                                                                );
                                                spinner_so.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                        public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){
                                                                so =
parentView.getSelectedItem().toString();
                                                        }

                                                        @Override
                                                        public void
onNothingSelected(AdapterView<?> arg0) {
                                                                // TODO Auto-generated
method stub

                                                        }
                                                }

                                                                );

                                                spinner_soi.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                                public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){

        if(parentView.getSelectedItem().toString().equals("Imprescindible"))
                                                                                soi = 1;
                                                                                else
if(parentView.getSelectedItem().toString().equals("Necesario"))
                                                                                        soi =
```

```
2;
                                                                    else
if(parentView.getSelectedItem().toString().equals("Me gustaria"))
                                                                        soi =
3;
                                                                    else
if(parentView.getSelectedItem().toString().equals("No necesario"))
                                                                        soi =
4;
                                                                    else
if(parentView.getSelectedItem().toString().equals("Prescindible"))
                                                                        soi =
5;
                                                                sois
=parentView.getSelectedItem().toString();
                                                            }

                                                            @Override
                                                            public void
onNothingSelected(AdapterView<?> arg0) {
                                                                // TODO Auto-
generated method stub

                                                            }
                                                        }

                                                                );
                                                    spinner_ram.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                        public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){
                                                            ram =
parentView.getSelectedItem().toString();
                                                        }

                                                        @Override
                                                        public void
onNothingSelected(AdapterView<?> arg0) {
                                                            // TODO Auto-generated
method stub

                                                        }
                                                    }

                                                            );

                                                    spinner_rami.setOnItemSelectedListener(
```

```
new OnItemSelectedListener(){
                                                    public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){

        if(parentView.getSelectedItem().toString().equals("Imprescindible"))
                                                                rami = 1;
                                                                else
if(parentView.getSelectedItem().toString().equals("Necesario"))
                                                                        rami =
2;
                                                                else
if(parentView.getSelectedItem().toString().equals("Me gustaria"))
                                                                        rami =
3;
                                                                else
if(parentView.getSelectedItem().toString().equals("No necesario"))
                                                                        rami =
4;
                                                                else
if(parentView.getSelectedItem().toString().equals("Prescindible"))
                                                                        rami =
5;
                                                                ramis
=parentView.getSelectedItem().toString();
                                                        }

                                                        @Override
                                                        public void
onNothingSelected(AdapterView<?> arg0) {
                                                                // TODO Auto-
generated method stub

                                                        }
                                                }

                                                                );
                                        spinner_tip.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){
                                                        tipo =
parentView.getSelectedItem().toString();
                                                }

                                                @Override
                                                public void
onNothingSelected(AdapterView<?> arg0) {
```

```
                                                        // TODO Auto-generated
method stub

                                                }
                                        }

                                                        );

                                        spinner_tipi.setOnItemSelectedListener(
new OnItemSelectedListener(){
                                                        public void
onItemSelected(AdapterView<?> parentView, View selectedItem,int position, long id){

        if(parentView.getSelectedItem().toString().equals("Imprescindible"))
                                                                tipoi = 1;
                                                                else
if(parentView.getSelectedItem().toString().equals("Necesario"))
                                                                        tipoi =
2;
                                                                else
if(parentView.getSelectedItem().toString().equals("Me gustaria"))
                                                                        tipoi =
3;
                                                                else
if(parentView.getSelectedItem().toString().equals("No necesario"))
                                                                        tipoi =
4;
                                                                else
if(parentView.getSelectedItem().toString().equals("Prescindible"))
                                                                        tipoi =
5;
                                                                tipois
=parentView.getSelectedItem().toString();
                                                        }

                                                        @Override
                                                        public void
onNothingSelected(AdapterView<?> arg0) {
                                                                // TODO Auto-
generated method stub

                                                        }
                                                }

                                                        );

                                        buttonJugar =
```

```
(Button)findViewById(R.id.button1);
                                        buttonJugar.setOnClickListener(new
OnClickListener(){
                                          //@Override
                                          public void onClick(View v) {
                                             TextView p =
(TextView)findViewById(R.id.Prueba);
                                            Button b = (Button)findViewById(R.id.button1);
                                            new LongRunningGetIO().execute();
                                                b.setClickable(false);
                                                   //so = (String) t.getText();
                                                   p.setText("Has seleccionado
:\nTipo:\n"+tipo + " \nImportancia: "+ tipois+
                                                                        "\n\nPantalla:\n"+pantalla +
" \nImportancia: "+ pantallais+
                                                                        "\n\nColor:\n"+color + "
\nImportancia: "+ coloris+

        "\n\nProcesador:\n"+procesador + " \nImportancia: "+ procesadoris+
                                                                        "\n\nSO:\n"+so + "
\nImportancia: "+ sois+
                                                                        "\n\nDispositivo
óptico:\n"+doptico + " \nImportancia: "+ dopticois+
                                                                        "\n\nDisco Duro:\n"+hdd + "
\nImportancia: "+ hddis+
                                                                        "\n\nMemoria
RAM:\n"+ram + " \nImportancia: "+ ramis+
                                                                        "\n\nPrecio:\n"+precio + "
\nImportancia: "+ preciois+
                                                                        "\n\nMarca:\n"+marca + "
\nImportancia: "+ marcais+

        "\n\nPopularidad:\n"+popularidad + " \nImportancia: "+ popularidadis);
                                          }
                                        });


                }

                /* (non-Javadoc)
                 * @see
android.app.Activity#onCreateOptionsMenu(android.view.Menu)
                 */
                @Override
                public boolean onCreateOptionsMenu(Menu menu) {
                        // Inflate the menu; this adds items to the action bar if it is
present.
```

```
                        getMenuInflater().inflate(R.menu.main, menu);
                        return true;
                }

                /* (non-Javadoc)
                 * @see
android.app.Activity#onOptionsItemSelected(android.view.MenuItem)
                 */
                @Override
                public boolean onOptionsItemSelected(MenuItem item) {
                        // Handle action bar item clicks here. The action bar will
                        // automatically handle clicks on the Home/Up button, so long
                        // as you specify a parent activity in AndroidManifest.xml.
                        int id = item.getItemId();
                        if (id == R.id.action_settings) {
                                return true;
                        }
                        return super.onOptionsItemSelected(item);
                }

        /**
         * Clase LongRunningGetIO.
         */
        private class LongRunningGetIO extends AsyncTask <Void, Void, String> {

                        /**
                         * Obtiene el contenido ASCII desde la endidad.
                         *
                         * @param entity entidad.
                         * @return contenido ASCII desde la endidad.
                         * @throws Lanza excepcion de estado ilegal.
                         * @throws Lanza excepcion de entrada de informacion.
                         */
                        protected String getASCIIContentFromEntity(HttpEntity entity)
throws IllegalStateException, IOException {
                      InputStream in = entity.getContent();
                        StringBuffer out = new StringBuffer();
                        int n = 1;
                        while (n>0) {
                           byte[] b = new byte[4096];
                           n =  in.read(b);
                           if (n>0) out.append(new String(b, 0, n));
                        }
                        return out.toString();
                    }

                        /* (non-Javadoc)
```

```
                         * @see
android.os.AsyncTask#doInBackground(java.lang.Object[])
                         */
                        @Override
                        protected String doInBackground(Void... params) {
                                HttpClient httpClient = new DefaultHttpClient();
                                HttpContext localContext = new BasicHttpContext();
                HttpGet httpGet = new
HttpGet("http://10.0.2.2:8080/Servicio/resources/model.producto");
                httpGet.setHeader("content-type", "application/json");
                String text = null;
                try {
                    HttpResponse response = httpClient.execute(httpGet);
                     text = EntityUtils.toString(response.getEntity());

                } catch (Exception e) {
                         return e.getLocalizedMessage();
                }
                return text;
                        }

                        /* (non-Javadoc)
                         * @see android.os.AsyncTask#onPostExecute(java.lang.Object)
                         */
                        protected void onPostExecute(String results) {
                                TextView t = (TextView)findViewById(R.id.tREST);
                                if (results!=null) {

                                        try {
                                                JSONArray Datos = new
JSONArray(results);
                                                ArrayList<Producto> p =
(ArrayList<Producto>) new ArrayList<Producto>();
                                                ArrayList<Ponderacion> po =
(ArrayList<Ponderacion>) new ArrayList<Ponderacion>();
                                            ArrayList<ProductoEvaluado> pe =
(ArrayList<ProductoEvaluado>) new ArrayList<ProductoEvaluado>();

                                                for(int i = 0 ; i<Datos.length();i++){
                                                        Producto pAux = new Producto();
                                                        JSONObject jObj =
Datos.getJSONObject(i);


        pAux.setNombre(jObj.getString("nombre"));

        pAux.setColor(jObj.getString("color"));
```

```
        pAux.setRam(jObj.getString("ram"));

        pAux.setDiscoduro(jObj.getString("discoduro"));


        if(jObj.getString("precio").equals("De $0 - $5,000"))
                                                {
                                                        pAux.setPrecio(5000);
                                                }

        if(jObj.getString("precio").equals("De $5,000 - $10,000"))
                                                {
                                                        pAux.setPrecio(10000);
                                                }

        if(jObj.getString("precio").equals("De $15,000 - $20,000"))
                                                {
                                                        pAux.setPrecio(20000);
                                                }

        if(jObj.getString("precio").equals("De $20,000 - en adelante"))
                                                {
                                                        pAux.setPrecio(30000);
                                                }

        pAux.setMarca(jObj.getString("marca"));

        pAux.setTipo(jObj.getString("tipo"));

        pAux.setProcesador(jObj.getString("procesador"));
                                                pAux.setSo(jObj.getString("so"));

        pAux.setDispositivooptico(jObj.getString("dispositivooptico"));

        pAux.setPantalla(Double.parseDouble(jObj.getString("pantalla")));

        pAux.setPopularidad(Integer.parseInt(jObj.getString("popularidad")));
                                                p.add(pAux);
                                        }

                                        Algoritmo a = new Algoritmo();
                                        po.add(new
Ponderacion(colori,"color",color));
                                         po.add(new Ponderacion(tipoi, "tipo",
tipo));
                                      po.add(new Ponderacion(dopticoi,
```

```
"dispositivoOptico", doptico));
                                        po.add(new Ponderacion(pantallai, "pantalla",
pantalla));
                                        po.add(new Ponderacion(popularidadi,
"popularidad", popularidad));
                                        po.add(new Ponderacion(procesadori,
"procesador", procesador));
                                        po.add(new Ponderacion(rami, "ram", ram));
                                        po.add(new Ponderacion(precioi, "precio",
"5000"));
                                        po.add(new Ponderacion(hddi, "hdd", hdd));
                                        po.add(new Ponderacion(marcai, "marca",
marca));
                                        po.add(new Ponderacion(soi, "so",so));
                                        for (Producto l : p) {
                                           double res = a.LSP(l, po);
                                           pe.add(new ProductoEvaluado(l, res));
                                           Collections.sort(pe, new
OrdenaResultado());
                                        }
                                        for (ProductoEvaluado prod : pe) {
                                           t.setText(t.getText()+"\n
"+prod.getNombre()+"\n "+prod.getMarca()
                                                   +"\n "+prod.getColor()+"\n--
----------------------");
                                        }

                                    } catch (JSONException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                    }
                                }
                                Button b = (Button)findViewById(R.id.button1);
                                b.setClickable(true);
                            }
        }

        }
```

OrdenaResultado

```
        package com.controlador;

        import java.util.Comparator;

        import com.modelo.ProductoEvaluado;

        // TODO: Auto-generated Javadoc
```

```
        /**
         * The Clase OrdenaResultado.
         */
        public class OrdenaResultado implements Comparator<ProductoEvaluado>{

            /* (non-Javadoc)
             * @see java.util.Comparator#compare(java.lang.Object, java.lang.Object)
             */
            @Override
            public int compare(ProductoEvaluado t1, ProductoEvaluado t) {
                return new Double(t.getEvaluacion()).compareTo(new
Double(t1.getEvaluacion()));

            }
        }
```

Apéndice C. Entregable: Manual del usuario

Introducción

El manual redactado en este documento tiene como propósito orientar a los usuarios que deseen usar la aplicación RECOM, de forma que no sea complicado operar la aplicación.

Nuestra aplicación se conforma de dos componentes: una aplicación web que puede ser utilizada por un administrador desde cualquier computadora personal y una aplicación cliente que puede ser utilizada por un consumidor para solicitar recomendaciones de productos.

Uso del sistema

Para poder utilizar el gestor de equipos de cómputo es necesario el abrir una ventana de cualquier navegador, posteriormente escribiremos la dirección del gestor: http://localhost:8080/Gestor, cabe mencionar que esta dirección cambiará de acuerdo al lugar donde se aloje esta aplicación web; una vez ingresada esta dirección nos aparecerá un formulario para iniciar sesión, una vez realizado esto, se nos mostrará todos los equipos que se encuentran actualmente dados de alta en RECOM, así como las operaciones de editar, eliminar y agregar de algún nuevo equipo de cómputo, como se muestra en la figura 22.

Figura 22. Pantalla inicial del gestor de base de datos

Para agregar un nuevo producto, damos clic en la liga que se señaló en la imagen con una flecha verde, para eliminar cualquiera de los productos en la lista se da clic en la liga que esta frente al producto señalada con una flecha color rojo, y finalmente para editar un producto, dar clic en la liga frente al producto, la cual está señalada con una flecha en color azul.

La aplicación cliente se inicia cuando en el dispositivo móvil se da clic sobre el ícono de RECOM, como se puede ver en la figura 23.

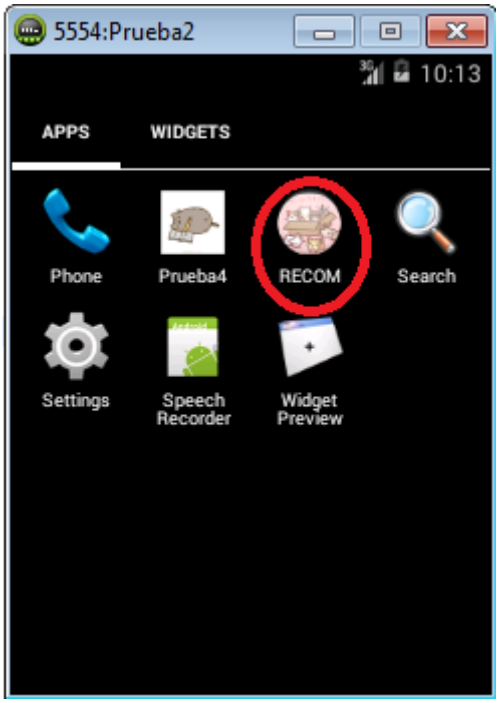

Figura 23. Icono de RECOM

Después se muestra la pantalla de inicio, ver figura 24, en la cual se pueden seleccionar las características con las cuales se requiere el equipo, debajo de la característica se puede

seleccionar la importancia que el consumidor le da a esa característica en el equipo de cómputo.

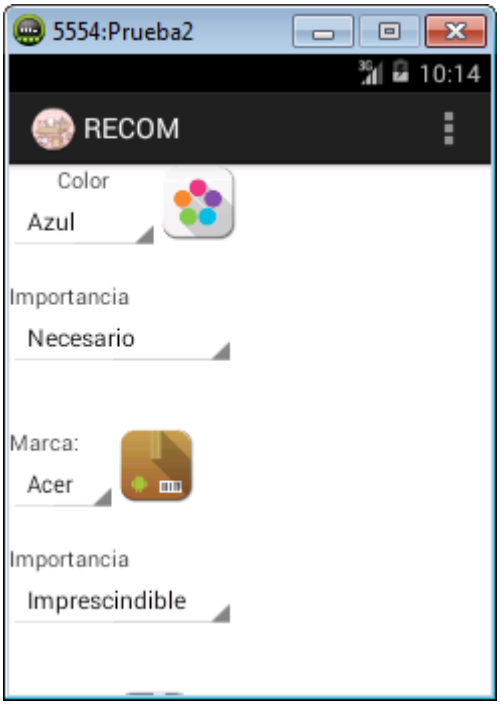

Figura 24. Pantalla inicial de RECOM.

Una vez seleccionadas las características del equipo de cómputo, se da clic en el botón buscar, el cual puede verse en la figura 25.

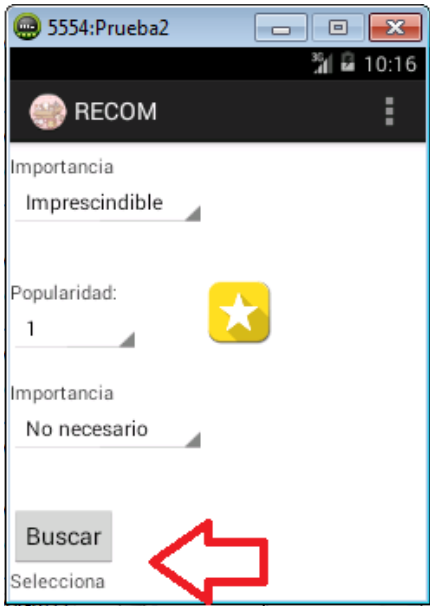

Figura 25. Botón buscar de RECOM.

Una vez que se ha dado clic en buscar aparecerán los resultados de las recomendaciones de equipo de cómputo, como se puede observar en la figura 26.

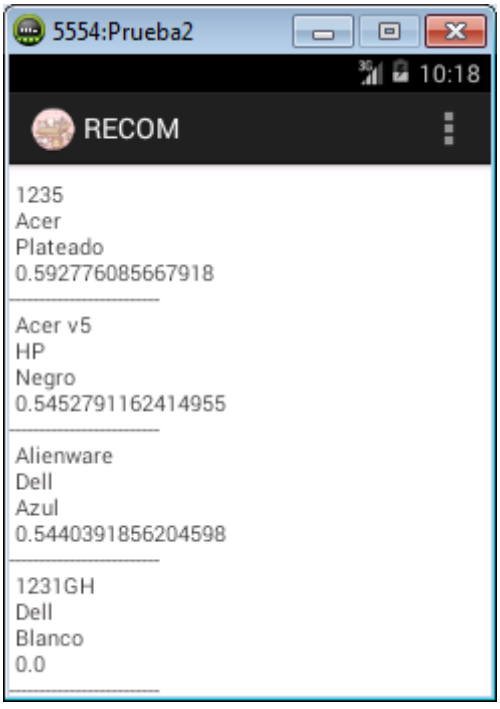

Figura 26. Resultados arrojados por RECOM.