PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

APLICACIÓN MÓVIL
PARA COMPARAR PRECIOS DE SUPERMERCADOS
"COTIZAPP"

FRANCISCO JAVIER CANALES GUTIERREZ
JAVIER OMID ROJAS CORNEJO

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO DE EJECUCIÓN EN INFORMÁTICA

AGOSTO 2017

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

APLICACIÓN MÓVIL
PARA COMPARAR PRECIOS DE SUPERMERCADOS
“COTIZAPP”

FRANCISCO JAVIER CANALES GUTIERREZ
JAVIER OMID ROJAS CORNEJO

Profesor Guía : **Iván Mercado Bermúdez**
Profesor Co-referente : **Ismael Figueroa Palet**

Carrera: **Ingeniería de Ejecución en Informática**

AGOSTO 2017

Índice

Resumen

Las tendencias actuales en tecnología están dominadas por la ubicuidad de la información, donde tiene un rol predominante el uso de dispositivos móviles, desde los cuales se les ofrece a los usuarios realizar nuevas actividades cada día, desde juegos hasta formas de movilizarse por la ciudad.

En el presente informe, se presenta una aplicación móvil que tiene por objetivo permitir a los usuarios comparar los precios entre las principales cadenas de supermercados del país a fin de que puedan decidir donde les es más conveniente realizar sus compras. Se expone sobre el modelo de proceso seleccionado, la arquitectura del sistema, incluyendo la aplicación en Android, un servicio web y la base de datos, la cual será generada mediante procedimientos de w*eb scraping*, utilizando un *framework* para Python denominado Beautiful Soup.

Palabras-claves: Android, Web Scraping, Python, Beautiful Soup.

Abstract

Current trends in technology are dominated by the ubiquity of information, where the predominant role is the use of mobile devices, from which users are offered new activities every day, from games to forms of mobilization around the city.

In this report, we present a mobile application which aims to enable users to compare prices between the main supermarket chains in the country, in order to decide where it would be most convenient for them to get their shopping list. It discusses the development methodology, the system architecture, which includes the android application, a web service and the database, this last one will be generated using web scraping procedures, using a framework for Python called Beautiful Soup.

Keywords: Android, Web Scraping, Python, Beautiful Soup.

Índice de Figuras

Índice de Tablas

1 Introducción

Dada la casi nula existencia de aplicaciones móviles para cotizar el precio de un producto en distintos supermercados de manera centralizada, es que se realizará un cotizador de precios para productos disponibles en tales lugares. Esto para teléfonos inteligentes con sistema operativo Android [1].

Esta aplicación permitirá al usuario realizar consultas por unidad o indicando un listado de productos, para los cuales la aplicación entregará su precio en los distintos supermercados seleccionados para el desarrollo, con lo cual el usuario podrá tomar la decisión de donde le es conveniente adquirir tal producto.

En primer lugar, se expondrán cuáles son los requerimientos tanto funcionales como no funcionales sobre los cuales se extiende el concepto y desarrollo de la aplicación. Luego, se explicará el modelo de proceso a aplicar durante el desarrollo del proyecto, indicando la razón de su elección. Posteriormente, se expondrá sobre el w*eb scraping* [2], dando una definición de este, sus aplicaciones y formas de utilización.

Seguidamente, se entrega una descripción general de la arquitectura del sistema, para continuar con una revisión a los diagramas utilizados para orientar el desarrollo. Luego se muestran algunas capturas de pantalla y descripciones de parte destacada del código de la aplicación. Posteriormente, se enumeran algunas consideraciones y supuestos que se tuvieron en cuenta al desarrollar este proyecto, y se expone el plan de pruebas utilizado. Por último, se expondrán las conclusiones adquiridas durante el desarrollo de este informe.

1 - Se desarrollará para el sistema operativo Android debido a que representa una cuota de aproximadamente un 75% del mercado móvil en el mundo, y en Chile se estima que el porcentaje de teléfonos con este sistema operativo es de un 84,7%.

2 - Web scraping es una técnica utilizada mediante programas de software para extraer información de sitios web.

2 Objetivos

Los objetivos principales de este trabajo se enfocan en cumplir con los requerimientos establecidos para el proyecto de *software* concebido y su correcto funcionamiento.

2.1 Objetivo General

- Desarrollar una aplicación móvil que permita comparar de manera simple los precios de las páginas web de distintos supermercados.

2.2 Objetivos Específicos

- Realizar un estudio sobre técnicas y herramientas de w*eb scraping* y seleccionar la adecuada para el desarrollo del proyecto.
- Implementar una herramienta de w*eb scraping* para obtener las categorías y la información de productos desde las páginas de los supermercados seleccionados.
- Desarrollar un prototipo funcional de aplicación móvil, que posibilite la consulta de la información obtenida mediante w*eb scraping*.
- Validación de la aplicación con potenciales usuarios.

3 Contextualización

No queda duda de que la tecnología hoy es cada vez más indispensable para las personas. Las distintas funciones que permiten realizar un sinnúmero de aplicaciones en nuestros *smartphones* hacen de estos unos objetos de enorme necesidad para ser parte del mundo actual. Solo en Chile, se estima que existen 132 celulares por cada 100 habitantes, y que 2 de cada 3 de estos son teléfonos inteligentes [1].

Para los *smartphones* existe una innumerable cantidad de aplicaciones que buscan entretener o satisfacer alguna necesidad de las personas, las que van desde aplicaciones para hacer deporte, afinadores para instrumentos musicales y miles de juegos, hasta laboratorios científicos de bolsillo. El mercado es enorme, y las posibilidades son amplias para desarrollar y ofrecer nuevas alternativas a los usuarios.

Como una manera de aprovechar este mercado, se planteó el desarrollo de una aplicación móvil, la que se denominó CotizApp. El proyecto consiste en ofrecer a los usuarios una forma fácil para comparar precios de listas y productos entre distintos supermercados del país a través de sus celulares.

Actualmente, no existe una manera extendida para que los usuarios puedan obtener y comparar los precios entre distintas tiendas o supermercados. Debido a esto, para que los usuarios puedan conocer los valores de los productos antes de ir de comprar, deben consultar en las revistas o páginas web de cada uno de los supermercados, lo que en la práctica es poco eficiente. Con la aplicación propuesta, se ofrecerá una alternativa que permita obtener esta información de manera rápida, contribuyendo así a un consumo informado.

El objetivo principal es lograr una cliente móvil que permita al usuario poder ver el precio de un producto en los tres supermercados seleccionados para el desarrollo del proyecto (Jumbo, Tottus y Líder) en una sola consulta. El usuario también podrá seleccionar un listado de productos, y así comparar el precio completo que tendría en cada uno de estos supermercados.

La aplicación móvil se desarrollará inicialmente para dispositivos Android, dada su mayor presencia en el mercado, además de facilidad de acceso a herramientas de desarrollo. En lo que respecta a la información de los supermercados, esta se obtendrá desde los sitios web de cada uno de los supermercados, utilizando para esto técnicas de *web scraping*.

4 Estado del Arte

Actualmente en el mercado nacional no hay una aplicación móvil de renombre que se enfoque en la comparación de precios de supermercados. Existen diversas alternativas enfocadas hacia el área de los “metabuscadores”, es decir, programas que no tienen una base de datos propia, sino que buscan en la web coincidencias a partir de lo que indique el usuario. A continuación se mencionan algunas de estas.

- Buscapé: *Software* de comparación de productos, creado en 1999 en Brasil. Buscapé organiza y agrupa los productos, haciendo más rápido y fácil el proceso de compra [2].

- Trivago: Es un buscador de precios de hoteles *online*. Busca y compara los precios ofrecidos por las principales agencias de viajes en Internet para una habitación de hotel en la ciudad indicada. Fue fundada en Alemania el 2005, hoy está presente en más de 20 países [3].

- SoloTodo: Página chilena que está enfocada principalmente al mercado de productos electrodomésticos. El sitio permite filtrar las ofertas según el precio máximo que se desea pagar, y entrando a búsqueda avanzada, según las características específicas del producto, como marca, tamaño, etc.

- MiPrecioJusto: Plataforma chilena donde se comparan los precios de grandes tiendas como Falabella, Ripley, etc. Cuando se busca un artículo, la plataforma entrega diferentes alternativas, ya sea de exactamente lo mismo que se buscaba o productos similares. Además, la página ofrece a los usuarios la opción de comprar algunos productos directamente.

- Onyougo: Es una aplicación para móviles que compara el precio de un producto entre más de 18 millones de referencias online y en tiendas físicas con solo acercar el código de barras al móvil. Además, permite encontrar una tienda física en las cercanías del usuario, contando con cerca de 80 mil establecimientos localizables. Esta aplicación tiene un carácter colaborativo, ya que los usuarios también pueden subir el precio de los productos, ayudando a mantener la base de datos actualizada.

- Cornershop: Es una aplicación y un sitio web mediante el cual se ofrece un servicio de entrega de una lista de compras a domicilio. Actualmente permite realizar compras en Jumbo, Líder, Gourmeat entre otros.

5 Requerimientos del Sistema

En esta sección se mencionarán los requerimientos que se han definido para el sistema propuesto.

5.1 Requerimientos Funcionales

Tabla 1.1 – Requerimientos funcionales

RF-CC : Requerimientos Funcionales “CotizApp”	
RF-01	La aplicación debe mostrar el precio para los supermercados seleccionados a la vez, permitiendo su comparación.
RF-02	La aplicación debe permitir consultar el precio de un producto.
RF-03	La aplicación debe permitir consultar el precio total de un conjunto de productos.
RF-04	La aplicación debe tener una opción de ayuda donde se expliquen las opciones disponibles.
RF-05	El sistema debe considerar una capa intermedia para comunicar la aplicación con la base de datos.
RF-06	La base de datos debe generarse y actualizarse respecto a los precios publicados en los sitios web de los supermercados seleccionados.
RF-07	Los precios de los productos deben ser mostrados a través de una ventana emergente.
RF-08	Los listados, tanto de categorías como de productos, deben permitir ingresar nuevos valores directamente en la base de datos y reflejarlos en el aplicativo, sin necesidad de volver a instalarlo.

5.2 Requerimientos No Funcionales

Tabla 1.2 – Requerimientos No Funcionales

RF-CC : Requerimientos No Funcionales “CotizApp”	
RNF-01	La aplicación debe tener una interfaz intuitiva.
RNF-02	El código debe estar ordenado y comentado.
RNF-03	La aplicación debe ser desarrollada con Android Studio.
RNF-04	El tiempo de demora de inicio de la aplicación no debe ser mayor a un minuto.
RNF-05	El tiempo de respuesta de la capa intermedia debe ser menor a 30 segundos.
RNF-06	Los supermercados seleccionados para trabajar son Líder, Tottus y Jumbo.
RNF-07	El servidor de base de datos debe estar disponible en una red local.

6 Modelo de Proceso

Para el desarrollo de la aplicación móvil propuesta se seleccionó el modelo de proceso *Prototype*. El modelo de prototipos permite que todo el sistema, o algunas de sus partes, se construyan rápidamente para comprender con facilidad y aclarar ciertos aspectos en los que se aseguren que el desarrollador, el usuario y el cliente estén de acuerdo en lo que se necesita, así como también la solución que se propone para dicha necesidad, y de esta forma minimizar el riesgo y la incertidumbre en el desarrollo.

Este modelo se utiliza principalmente cuando un cliente define un conjunto de objetivos generales para el *software* a desarrollarse, sin delimitar detalladamente los requisitos de entrada, procesamiento y salida, es decir, cuando el responsable no está seguro de la eficacia de un algoritmo, de la adaptabilidad del sistema, o de la forma en que interactúa el hombre y la máquina. Este modelo se encarga, principalmente, de ayudar al ingeniero y al cliente a entender de mejor manera cuál será el resultado de la construcción cuando los requisitos estén satisfechos [4].

La razón por la cual se seleccionó este modelo, es la facilidad que nos da para ir refinando requerimientos, vistas y otros, puesto que los requerimientos del proyecto pueden modificarse frente a distintas situaciones, además de poder ir evaluando la utilidad y el fácil entendimiento de la interfaz de usuario a medida que se desarrolla.

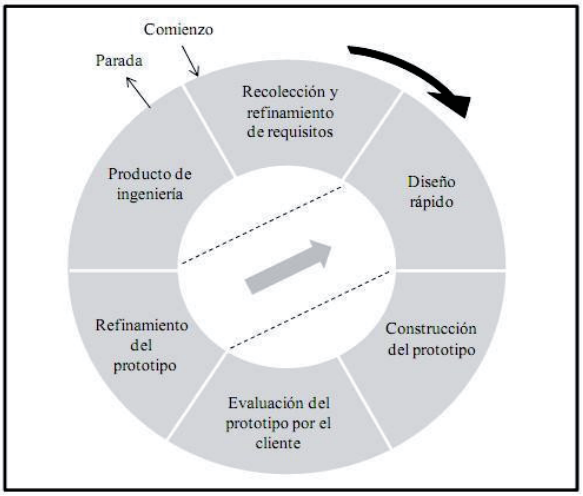

Figura 1 : Modelo de Proceso Prototype

7 Web Scraping

Para comprender como se obtendrá la información de las bases de datos del proyecto, es importante entender que es el w*eb scraping*. Una traducción literal es “raspado web”. En informática, se utiliza este término cuando se desea extraer información específica a partir de grandes volúmenes de datos disponibles en Internet.

7.1 Definición

Web scraping es el proceso por medio del cual un usuario puede hacer una copia de información contenida en las páginas web que tenga acceso, descartando aquella que no le resulta útil. Para el caso de la aplicación que se expone en este informe, esta técnica se utiliza para obtener la información disponible en las páginas web de los supermercados respecto a los productos, realizando mediante esta técnica la obtención de datos como nombre, marca, descripción, precio, imagen y código de cada producto.

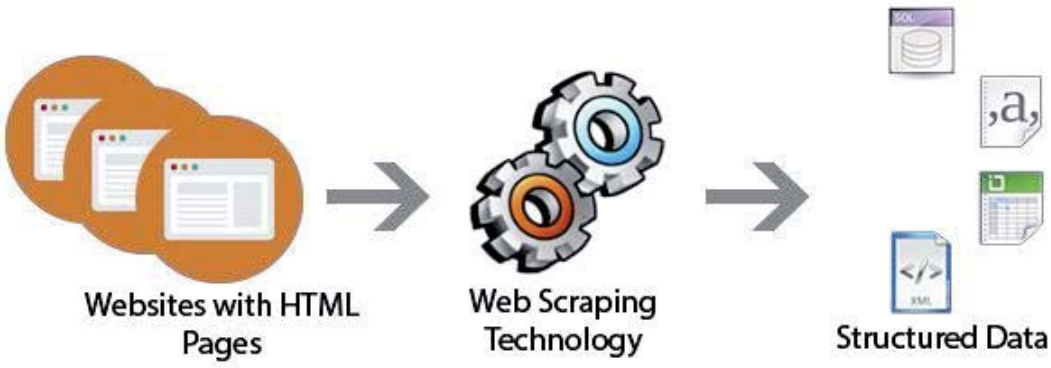

Figura 2 : Proceso de Web Scraping

7.2 Aplicaciones

Cuando deseamos obtener ciertos datos dentro de la enorme fuente de información que es Internet, debemos enfocar nuestros esfuerzos en la manera en que tales datos se pueden obtener. Realizarlo de manera convencional resulta un proceso bastante engorroso y, por sobre todo, lento.

El w*eb scraping* es una tecnología de obtención de datos relativamente nueva (desde el 2000 se registran las primeras investigaciones extensas); a pesar de lo anterior, resulta una técnica bastante útil para obtener información, dado que podemos elegir su periodicidad, además de poder seleccionar sólo las páginas que nos interesan. Para realizar w*eb scraping* se pueden utilizar programas con interfaz gráfica, tales como Irobosoft e Import.io con versiones gratuitas, o bien *software* más poderoso y de pago, como Octoparse y Web Harvest. Otras alternativas son utilizar *scripts* en algún lenguaje de programación.

Para el desarrollo del sistema CotizApp se decidió utilizar una serie de *scripts*, ya que brinda mayor flexibilidad al poder programar cada una de las tareas que se quieren realizar, tanto para obtener la información como para procesarla. Para esto, se utiliza un framework para Python denominado Beautiful Soup, el cual permite obtener el código detrás de un sitio web e inspeccionar sus componentes de manera simple. En el caso de sitios con Javascript, se debe utilizar un navegador web, y para realizar el s*craping* se utilizó PhantomJS, un *headless browser* gratuito (navegador sin interfaz gráfica, especial para tareas automatizadas).

Al realizar w*eb scraping*, se debe considerar la carga que se realizará sobre los servidores del sitio al cual se le realiza la extracción, evitando que los accesos sean excesivos en un periodo de tiempo, lo que podría bloquear algunos sitios o provocar que la dirección IP del equipo haciendo el *scraping* sea inhabilitada de todo acceso, al levantar alertas de seguridad. Algunos sitios web tienen archivos disponibles donde indican como realizar extracción de información de manera prudente.

Nos parece conveniente utilizar esta técnica para obtener los datos, debido a la facilidad de mantener la información actualizada para mostrarsela al usuario según lo definido, ya que, por ejemplo, podemos obtener los datos de los precios una vez por día, y de esta manera el cliente verá información actualizada y fiable, dado que viene directamente desde los sitios web que los supermercados tienen disponible para sus clientes.

7.3 Aspectos Legales

Existen aspectos legales que se deben considerar al realizar *web scraping*. Dado que no toda la información disponible puede que sea de uso público, se debe tener cuidado con cuáles páginas se van a inspeccionar, además de qué datos o imágenes se van a mostrar efectivamente en la aplicación. Si bien hay datos que son posibles de encontrar de manera pública por cualquier usuario, existe la posibilidad de que el administrador del sitio web o las empresas que originan tal información, podrían reclamar sobre la naturaleza de la esta, poniendo en entredicho que pueda ser utilizada por terceros.

Un conocido caso sobre s*craping* involucró a la aerolínea American Airlines y a una empresa llamada FareChase. La aerolínea acusaba a la empresa de *software* de ingresar sin autorización en sus servidores a realizar la recolección mediante *scraping*. La aerolínea ganó esta batalla, haciendo que la empresa creadora del servicio tuviera que quitar los datos o no podría continuar vendiendo su *software*, el cual ofrecía a los usuarios comparar tarifas de vuelos en línea, incluyendo entre otros el sitio de American Airlines. Recientemente, LinkedIn a indicado a sus usuarios que no se puede realizar *scraping* de su sitio web sin permiso [6], sin embargo, la justicia de EEUU dictaminó en su contra, ya que no se accede a sus servidores [7].

Los casos aquí comentados nos indican que al utilizar este tipo de técnicas, se debe de consultar y conocer sus potenciales implicancias legales, así como utilizar buenas prácticas de *web scraping* [8], a fin de no poner en riesgo el funcionamiento normal de los sitios web accedidos.

8 Arquitectura del Sistema

El modelo considerado para este proyecto es uno de 3 capas, como lo muestra la siguiente figura:

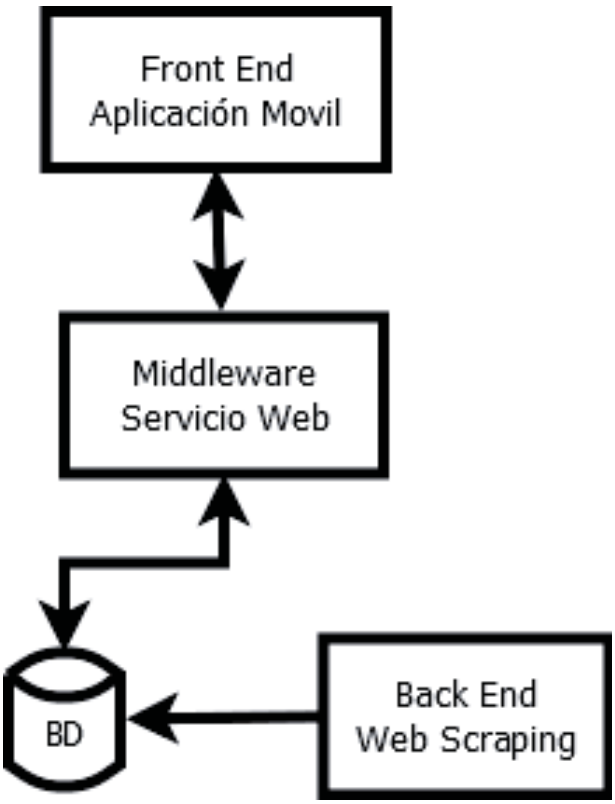

Figura 3 : Arquitectura del Sistema

- Front End: Corresponde a la aplicación móvil, mediante la cual el usuario puede interactuar con el sistema, mediante una interfaz donde se presentan las acciones y la información disponible.

- Capa intermedia: Consiste en el servicio web. En esta capa se realiza la comunicación entre la aplicación móvil y la base de datos del sistema.

- Back End: Contiene la base de datos y su sistema de mantención, generación y actualización, estos dos últimos correspondientes a las herramientas de *web scraping*.

9 Diseño del Proyecto

Los diagramas expuestos a continuación describirán la organización y las relaciones que existen entre los actores y las clases que concurren en la utilización del sistema móvil a desarrollar.

9.1 Diagramas de Casos de Uso

A continuación, se presentan dos diagramas de casos de uso, el primero correspondiente a una vista general de la aplicación móvil, mientras que el segundo abarca el comportamiento de tales acciones en el siguiente nivel de detalle.

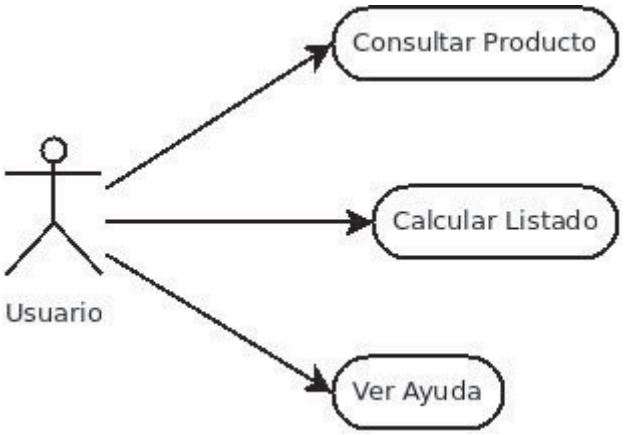

Figura 4 : Diagrama de Casos de Uso de Alto Nivel

En la Figura 4 se observa el diagrama de contexto, o bien diagrama de caso de uso de alto nivel. En él, se aprecian las tres acciones disponibles que puede realizar un usuario para interactuar con la aplicación. En la Figura 5 se indica en mayor detalle que ocurre en cada una de estas acciones.

La opción “Consultar Producto”, mostrará la lista de categorías, luego se mostrará la lista de productos pertenecientes a la misma, a fin de que el usuario pueda seleccionar uno de estos para consultar su precio, los cuales aparecerán en una ventana emergente. De manera similar, la opción “Seleccionar Listado” permite sumar el precio de varios productos indicados y comparar el precio total del conjunto ingresado.

Para la funcionalidad “Ver Ayuda”, el usuario accederá a una pantalla donde se describirán las características principales de la aplicación y su modo de empleo. Luego de esto, el usuario puede volver al inicio, opción que está siempre disponible.

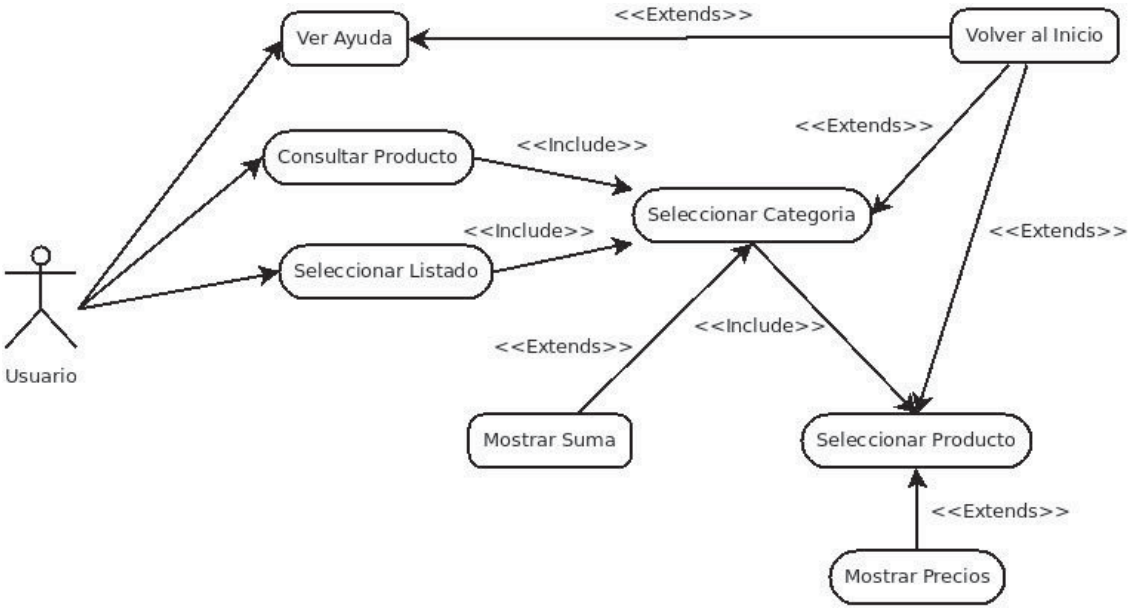

Figura 5 : Diagrama de Casos de Uso de Nivel 1

9.2 Diagrama de Clases

A continuación se expone el diagrama donde se aprecian las clases que se identificaron para la aplicación desarrollada.

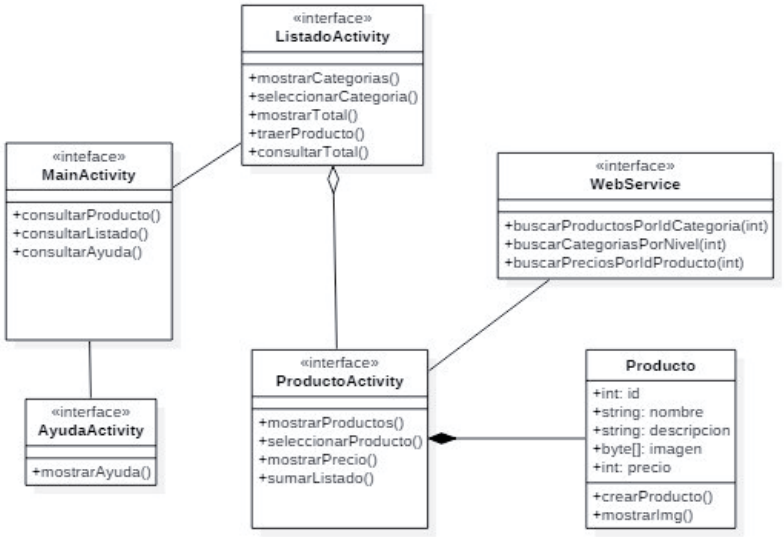

Figura 6 : Diagrama de Clases

9.3 Diagramas de Secuencia

A continuación se muestran los diagramas de secuencia desarrollados para describir la interacción que existe entre el actor y los principales módulos del sistema, esto al realizar alguna de las acciones disponibles en el aplicativo.

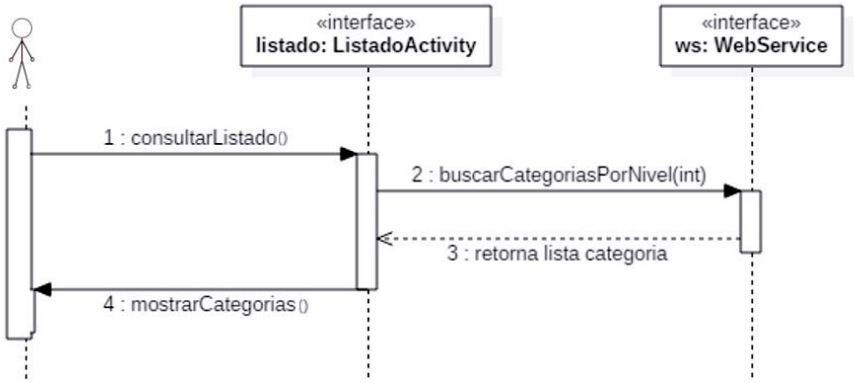

Figura 7 : Diagrama de Secuencia Seleccionar Categoría

En la Figura 7 se indica el proceso que ocurre al presionar la opción de consultar listado (o consultar producto), se muestra al usuario una lista con las categorías disponibles, en las cuales se agrupan distintos productos.

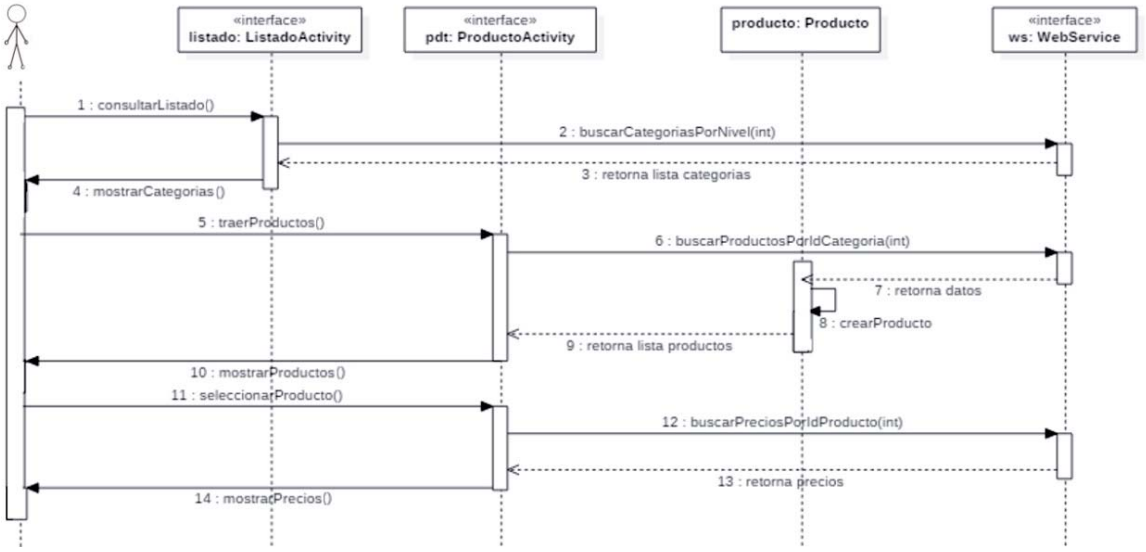

Figura 8 : Diagrama de Secuencia Seleccionar Producto

Al igual que en la Figura 7, para el proceso que se indica en la Figura 8, el usuario en primer lugar accede a la lista de categorias, selecciona una de estas, donde ahora le aparecerá

un listado de productos. El usuario debe indicar el item deseado, provocando que la aplicación envíe el identificador al servicio web para obtener el precio desde la base datos.

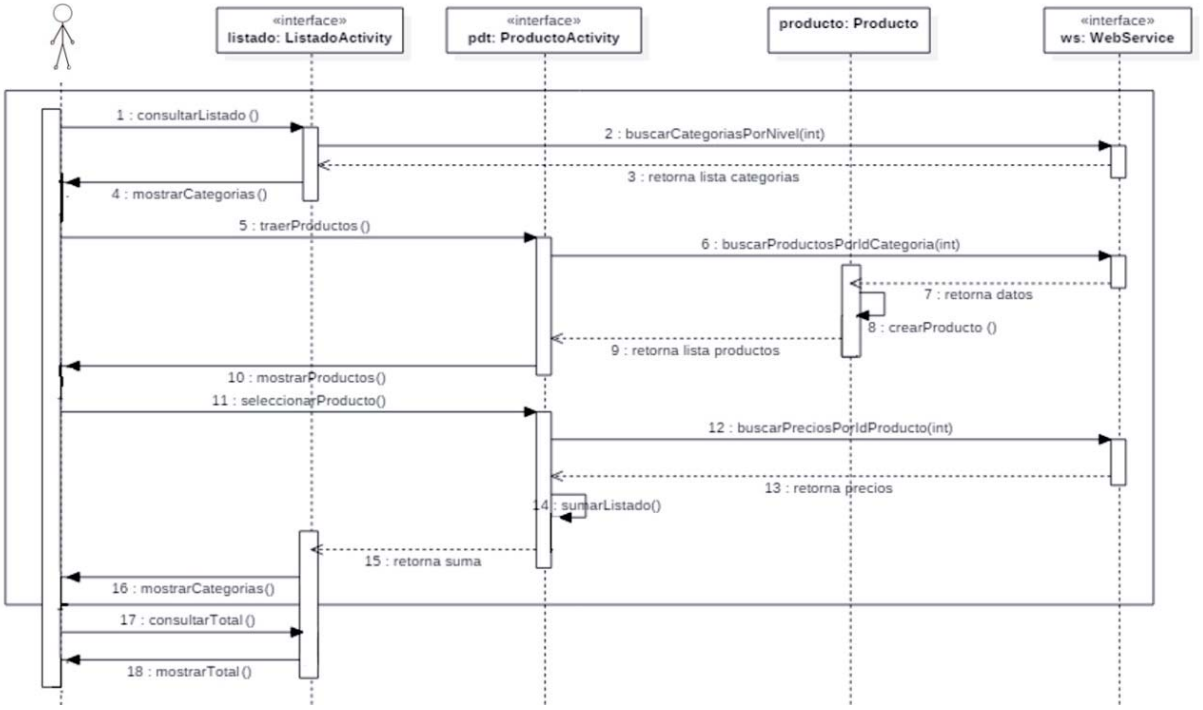

Figura 9: Diagrama de Secuencia Seleccionar Listado de Productos

Un proceso similar se realiza al consultar un listado de productos, como se muestra en la Figura 9. La diferencia es que ahora la aplicación irá guardando la suma de precios de los productos seleccionados y mostrará el total al presionar el boton de cálculo.

9.4 Diagramas de Entidad - Relación

En este punto se presentan los diagramas de Entidad-Relación pertenecientes a las bases de datos de la aplicación, CotizApp y WebScrap (donde se almacenan los resultados del *web scraping).* Es importante mencionar que al utilizar el motor PostgreSQL, en su última versión existe la opción de utilizar tablas foráneas para conectar distintas bases de datos alojadas en el mismo servidor, lo que permite un acceso con mejor rendimiento que el obtenido mediante la utilización de vistas, como se suele realizar con otros motores.

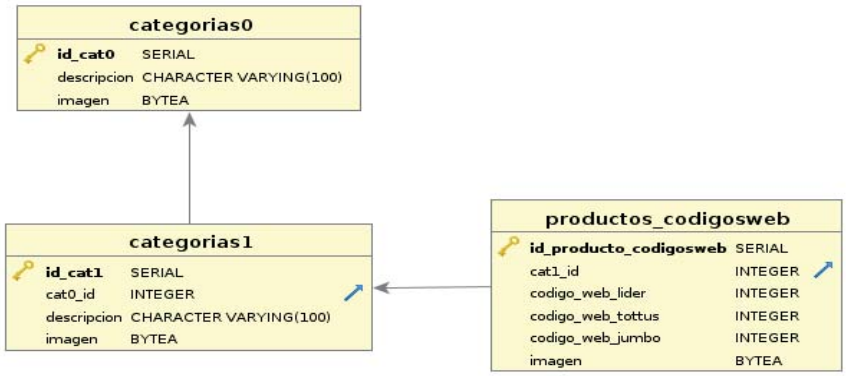

Figura 10 : Diagrama Entidad-Relación CotizApp

La figura 11 corresponde al modelo de datos definido, donde se almacenan la información obtenida con los procedimientos de *web scraping*.

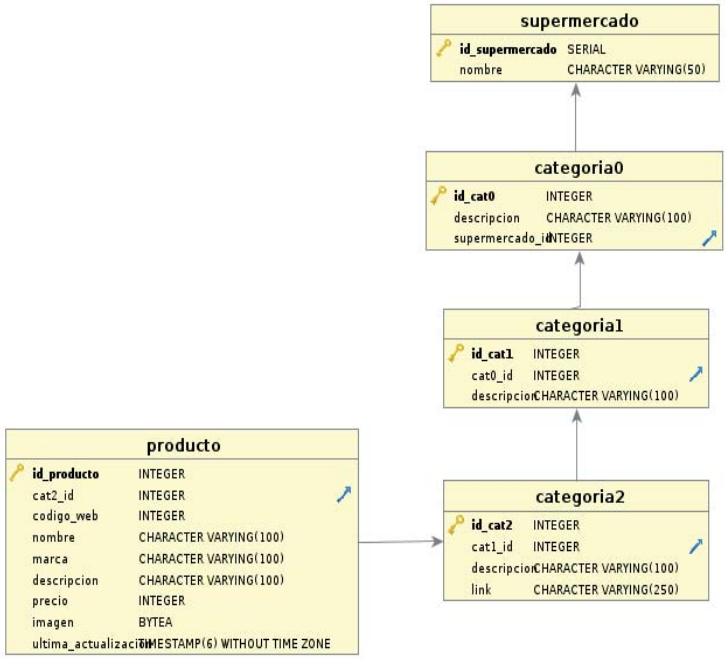

Figura 11 : Diagrama Entidad-Relación WebScrap

10 Implementación de la Aplicación

En este apartado se expondrán algunas capturas de pantalla de la aplicación, junto con extractos destacados del código desarrollado.

Pantalla Principal:

La siguiente imagen muestra la primera pantalla existente en el aplicativo. El código adjunto muestra la creación de la instancia de los 3 botones propuestos, indicando el método de invocación de uno de ellos.

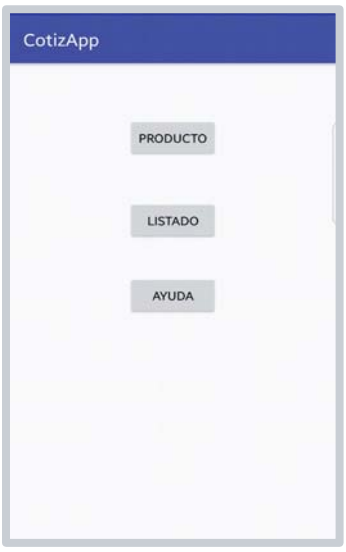

Figura 12 : Pantalla de Inicio

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ActionBar actionBar = getSupportActionBar();
        actionBar.setDisplayHomeAsUpEnabled(true);

        producto = (Button) findViewById(R.id.btn_pdt);
        listado = (Button) findViewById(R.id.btn_lista);
        ayuda = (Button) findViewById(R.id.btn_ayuda);

        siguiente.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v){
                Intent sgt = new Intent(MainActivity.this,
ProductoActivity.class);
                startActivity(sgt);
            }
        }
);
```

Pantalla Lista de Categorías:

Se muestra al seleccionar el botón “listado” (para el botón “productos” se ve la misma vista sin el boton calcular). En el código adjunto, se muestra la creación de la vista, obteniendo el listado de categorías.

Figura 13: Pantalla de Lista de Categorías

```
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.categorias_activity);

      CategoriasContatenados(num);
         CategoriasContatenadosView();
   }

   private void CategoriasContatenados(num)
   {
      WebServiceCotizApp WebS = new WebServiceCotizApp();

      try {
         final List<String> resultado =
WebS.traerProductosPorIdCategoria1(num);
         mCategorias = new ArrayList<>();
         for(Object datos : resultado){
            String data = String.valueOf(datos);
            Categorias cat = new Categorias(data, obtenerImg(num));
            mCategorias.add(prod);
         }
      } catch (IOException e) {
         e.printStackTrace();
      } catch (XmlPullParserException e) {
         e.printStackTrace();
      }
```

Pantalla Lista de Productos:

Se muestra al seleccionar una categoría. En el código adjunto se muestra la creación de a vista, en base a los datos obtenidos del WebService previamente al presionar en la categoría.

Figura 14: Pantalla de Lista de Productos

```
private void ProductosContatenadosView()
   {
      ArrayAdapter<Productos> adapter = new MyListAdapter();
ListView list = (ListView) findViewById(R.id.lv_arroz);
list.setAdapter(adapter);
```

Pop-Up Precios:

Al seleccionar un producto, se muestra el Pop-Up con el precio del producto en los 3 supermercados seleccionados. En el código adjunto se muestra la creación del Pop-Up.

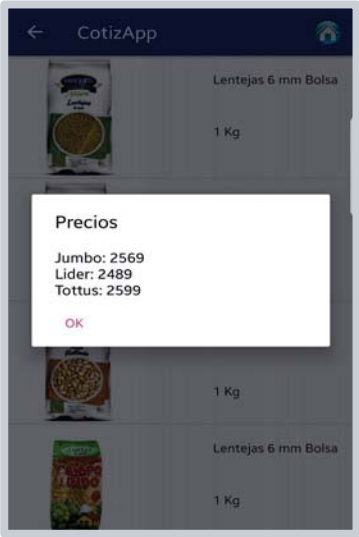

Figura 15 : Pantalla con Pop-Up de Precios

```
AlertDialog.Builder builder = new
AlertDialog.Builder(ProductoAdapter.this);
builder.setMessage("Jumbo: " + valorNumericoJumbo + "\nLider: "
+ valorNumericoLider + "\nTottus: " + valorNumericoTottus)
.setTitle("Precios")
.setCancelable(false)                    .setNeutralButton("OK",
new DialogInterface.OnClickListener()
{                                  public void onClick(DialogInterface
dialog, int id) {                          dialog.cancel();
}                                });
AlertDialog alert = builder.create();
alert.show();
}
```

11 Consideraciones Importantes

En los siguientes puntos se enumeran algunas consideraciones que se tuvieron durante el desarrollo de este proyecto.

- La aplicación desarrollada considera durante esta etapa solamente la utilización de los precios publicados por las cadenas de supermercados en sus sitios web, sin considerar descuentos u ofertas especiales.

- Se seleccionaron los supermercados: Líder, Jumbo y Tottus. Esto para poder enfocar el desarrollo, además de que otros supermercados presentan mayores dificultades para realizar *web scraping* al utilizar, por ejemplo, *flash* o imágenes en vez de texto para indicar los precios de los productos.

- Aquellos productos que no estén en el catálogo *online* del supermercado no serán considerados en la aplicación; de la misma manera, aquellas ofertas que hayan sido publicadas de manera física en los locales, pero no subidas a su página oficial tampoco serán parte de los precios que mostrará la aplicación.

- Si el usuario desea saber el costo de un listado de productos, la aplicación mostrará el precio de la lista ya sumada. No mostrará el detalle del precio de los productos. Es decir, si se ingresan 3 productos, mostrará cual es el valor de comprar los 3 productos juntos en un supermercado, no cuál es el valor de cada producto y el valor sumado.

10 Plan de Pruebas

Tabla 2.1 – Plan de Pruebas

Análisis y Diseño							
N°	**Objetivo Caso de Prueba**	**Condiciones de la Prueba**	**Acciones**	**Caso de Prueba Precedente**	**Resultado Esperado**	**Fecha Ejecución**	**Responsable Ejecución**
1	Botón de listado funciona correctamente	**Se realizará en ambiente de productivo no masivo**	Seleccionar el boton de listado		Que este despliegue la lista de categorías	**TBD**	**Javier Rojas / Francisco Canales**
2	Botón de producto funciona correctamente	**Se realizará en ambiente de productivo no masivo**	Seleccionar el boton de producto		Que este despliegue la lista de categorías	**TBD**	**Javier Rojas / Francisco Canales**
3	Que pueda entrar a la lista de productos	**Se realizará en ambiente de productivo no masivo**	Una vez en el menú de categorías, se debe seleccionar una	1 y 2	Que al seleccionar una categoría, se pueda entrar a la vista que contiene la lista de productos	**TBD**	**Javier Rojas / Francisco Canales**
4	Validar que el push-up funciona	**Se realizará en ambiente de productivo no masivo**	Seleccionar un producto	3	Que al seleccionar un producto, este me muestre el push-up	**TBD**	**Javier Rojas / Francisco Canales**
5	Validar la correcta obtención de los datos	**Se realizará en ambiente de productivo no masivo**	Seleccionar un producto	4	Que los precios mostrados en el push-up sean los que en la base de datos están almacenados	**TBD**	**Javier Rojas / Francisco Canales**
6	Validar que el botón Home funciona correctamente	**Se realizará en ambiente de productivo no masivo**	Seleccionar el botón home		Que al seleccionar el botón se vuelva al menú principal, matando los activitys seleccionados previamente	**TBD**	**Javier Rojas / Francisco Canales**
7	Validar que el boton atras funciona correctamente	**Se realizará en ambiente de productivo no masivo**	Seleccionar el boton atrás		Que al seleccionar el botón se vuelva a la vista anterior, matando los activitys seleccionados previamente	**TBD**	**Javier Rojas / Francisco Canales**

13 Conclusiones

En este informe se revisó el desarrollo de la aplicación móvil “CotizApp”, mediante la exposición del modelo de proceso del desarrollo, la arquitectura utilizada, y los diagramas que orientaron el diseño del *software*. También se indicaron aspectos de la aplicación móvil y las bases de datos asociadas al sistema.

Respecto al desarrollo de la aplicación móvil, las primeras versiones se realizaron utilizando un Activity principal junto a un conjunto de Fragments[3], pero debido a dificultades encontradas al momento de pasar de un Fragment a otro, se decidió modificar el código y utilizar solamente Activities[4] para cada acción requerida, lo que permitió alcanzar un prototipo funcional del *software*. Respecto al diseño, la interfaz que presenta la aplicación es básica, pero que cumple con los requerimientos del sistema y los objetivos propuestos. Cabe mencionar que existen plantillas prediseñadas de aplicaciones móviles en sitios como GitHub, las cuales respetando las licencias, pueden ser utilizadas para otros desarrollos. Esta opción no se utilizó, dado que no se tenía el conocimiento suficiente de Android para poder adaptar código ajeno.

En la capa del backend, correspondiente a la generación de las bases de datos, en primer lugar se realizó una investigación sobre distintas aplicaciones de *scraping* en el mercado, a fin de encontrar la más adecuada para el desarrollo. Se utilizó el programa iRobosoft, pero se descartó su uso debido a que éste requería aprender un lenguaje específico (HTQL) y utilizaba mucha memoria en la máquina. Luego se decidió realizar una serie de *scripts*, ya que la investigación nos indicó que estos brindan mayor flexibilidad al poder programar cada una de las tareas que se quieren realizar tanto para obtener la información como para procesarla.

Para implementar los *scripts*, se optó por el lenguaje Python, y se probó con algunos *frameworks* para esta función, entre ellos Scrapy y Beautiful Soup. Se optó por utilizar este último, debido a sus capacidades y facilidad de uso. Una vez probado y terminado los *scripts* de *web scraping*, se generaron las bases de datos y se realizó la integración del sistema mediante el *web service*. En base a esta experiencia, se recomienda el uso de Beutiful Soup junto a Selenium y PhantomJS, o algun otro *headless browser* que permita la ejecución de Javascript y la navegación en las páginas web sobre las que se realice el *scraping*.

Respecto al desarrollo de la aplicación móvil, se recomienda investigar sobre cuales son las mejores arquitecturas para su desarrollo según las versiones de Android con las que se espera sea compatible el *software*. En el caso de este proyecto, se fue aprendiendo y

3 - Fragment : Representa un comportamiento o una parte de la interfaz de usuario en un Activity.

4 - Activity : Es una clase que engloba alguna acción que puede realizar un usuario. Se puede trabajar en módulos o fragments.

desarrollando en base a prototipos, con el foco en lograr las funcionalidades requeridas, pero a costa de una utilización poco prolija de los recursos del *smartphone*.

Una vez alcanzada la versión final del sistema, es decir, la aplicación y el *web service* funcionando, además de las bases de datos completadas con información obtenida con *web scraping*, se procedió a realizar pruebas con usuarios, para validar la aceptación del producto ofrecido. En general se recibieron comentarios positivos respecto a la idea de comparar precios de supermercados, así como sugerencias respecto al diseño, así como otras ideas para incorporar nuevas funcionalidades, como el uso de geolocalización, el manejo de sesiones personalizadas, entre otras.

Como proyecciones del sistema, existen diversas opciones y mejoras que se pueden incorporar en la aplicación móvil, tales como entregar mayor información sobre los productos, permitir el ingreso o actualización de precios por parte de los usuarios, utilización de memoria caché para menú e imagenes, y mejoras en el aspecto visual. Por el lado del backend, además se pueden obtener datos estadísticos sobre los precios, los productos más consultados en un periodo de tiempo, entre otros, lo que puede constituir información útil para las mismas cadenas de supermercados, y un potencial foco de modelo de negocios.

14 Bibliografía

[1] La Tercera. (2016). Subtel: 77% de las conexiones a internet son a través del celular - LA TERCERA. [online] Disponible en: http://www.latercera.com/noticia/subtel-77-de-las-conexiones-a-internet-son-a-traves-del-celular/ [Consultado el 15 Marzo 2016].

[2] Buscape.com.ar. (2016). Buscapé - Conoce Buscapé. [online] Disponible en: http://www.buscape.com.ar/conoce-buscape [Consultado el 12 Febrero 2016].

[3] Es.wikipedia.org. (2017). Trivago. [online] Disponible en : https://es.wikipedia.org/wiki/Trivago [Consultado el 20 Abril 2016].

[4] G. y V. (2011). MODELO DE PROTOTIPO. [online] Gestionrrhhusm.blogspot.cl. Disponible en: http://gestionrrhhusm.blogspot.cl/2011/05/modelo-de-prototipo.html [Consultado el 21 Abril 2016].

[5] Ine.cl. (2016). Nueva Canasta IPC [online] Disponible en: http://www.ine.cl/canales/sala_prensa/revistaseconomicas/presentaciones/pdf/nuevacanastaipc.pdf [Consultado el 21 Abril 2016].

[6] Burnham, K. (2017). LinkedIn Sues After Scraping Of User Data - InformationWeek. [online] Disponible en: https://www.informationweek.com/software/social/linkedin-sues-after-scraping-of-user-data/d/d-id/1113362 [Consultado el 8 de Agosto de 2017].

[7] Lee, T. (2017). Court rejects LinkedIn claim that unauthorized scraping is hacking - Ars Technica. [online]. Disponible en: https://arstechnica.com/tech-policy/2017/08/court-rejects-linkedin-claim-that-unauthorized-scraping-is-hacking/ [Consultado el 15 de Agosto de 2017].

[8] Koshy, J. (2016). Web Scraping: Best Practices to Follow - PromptCloud [online] Disponible en: https://www.promptcloud.com/blog/web-scraping-best-practices [Consultado el 15 de Agosto de 2017].