



Introducción a la programación  
de dispositivos móviles.

Profesor: Mauricio Arroqui.

# Trabajo Final: Ofertapp

*Mastieri Julian: [julimastieri@gmail.com](mailto:julimastieri@gmail.com)  
Goicoechea, Maria Pia: [maripigoico@gmail.com](mailto:maripigoico@gmail.com)*

## Indice

Introducción.....	3
Base de datos y localhost.....	3
Funcionamiento .....	4
Conclusión.....	11

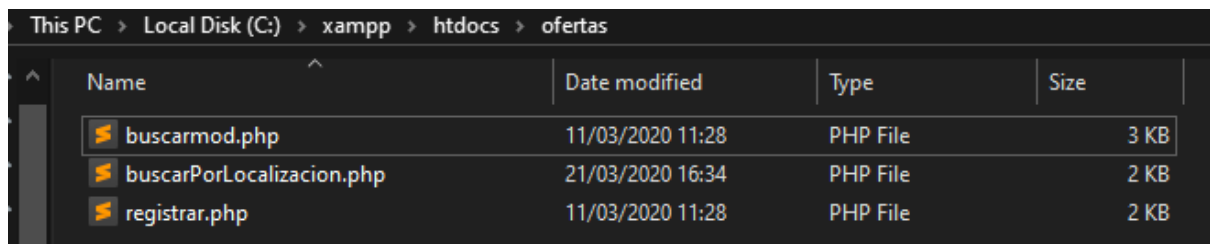
## Introducción

En el presente informe se detalla el desarrollo de la aplicación Android llamada Ofertapp. La misma cuenta con tres funcionalidades principales: agregar una oferta a una base de datos, buscar ofertas con la posibilidad de aplicar filtros y envío de notificaciones con las ofertas más cercanas a la posición actual.

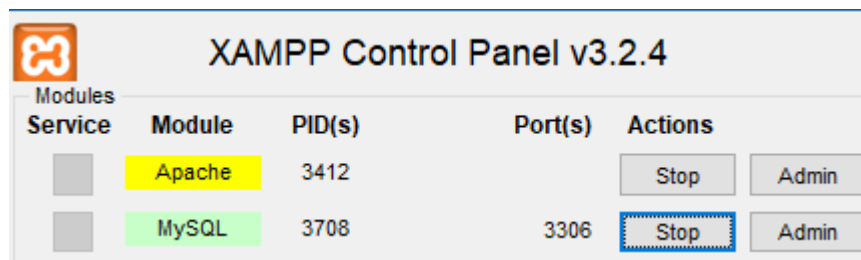
## Base de datos y localhost

Para que la aplicación funcione correctamente se debe de contar con la herramienta XAMPP. La misma nos permite establecer un localhost donde estará ubicada la base de datos de la aplicación.

Luego de instalarla, hay que colocar los tres archivos PHP (buscarmod.php, buscarPorLocalizacion.php, registrar.php) en la siguiente ruta: **C:\xampp\htdocs\ofertas**; puede ser que la misma varíe dependiendo de cada computadora, pero básicamente es dentro de la carpeta de XAMPP buscamos htdocs y creamos una carpeta llamada ofertas donde pegaremos los archivos.



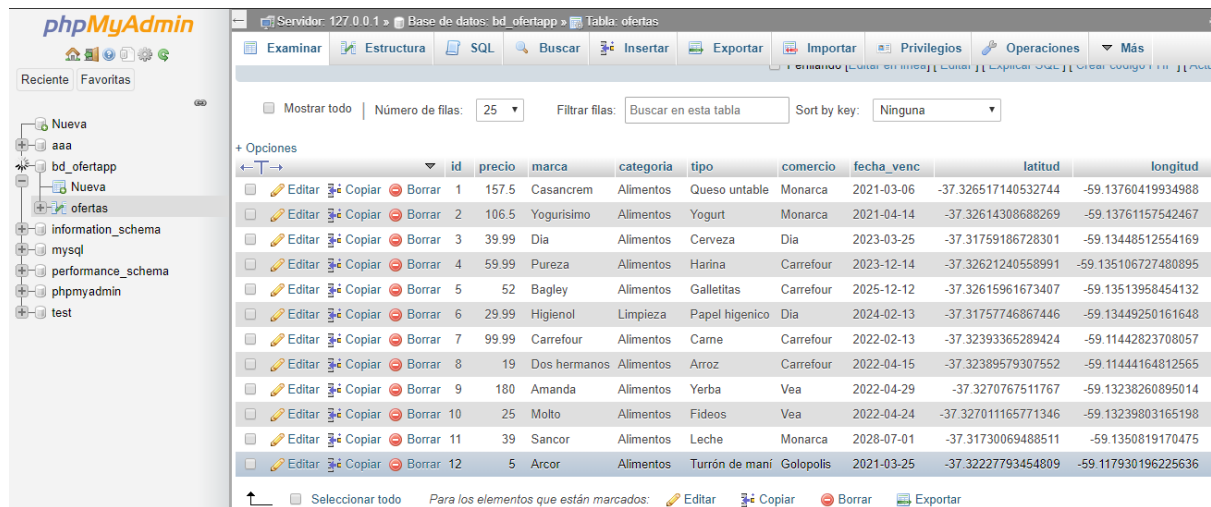
Luego abrimos la aplicación e iniciamos Apache y MySQL y cliqueamos en “Admin” de esta última.



Se nos abrirá **phpMyAdmin**, donde crearemos una nueva base de datos llamada “bd\_ofertapp”.



E importaremos en la base de datos la tabla del archivo “bd\_ofertapp.sql”.



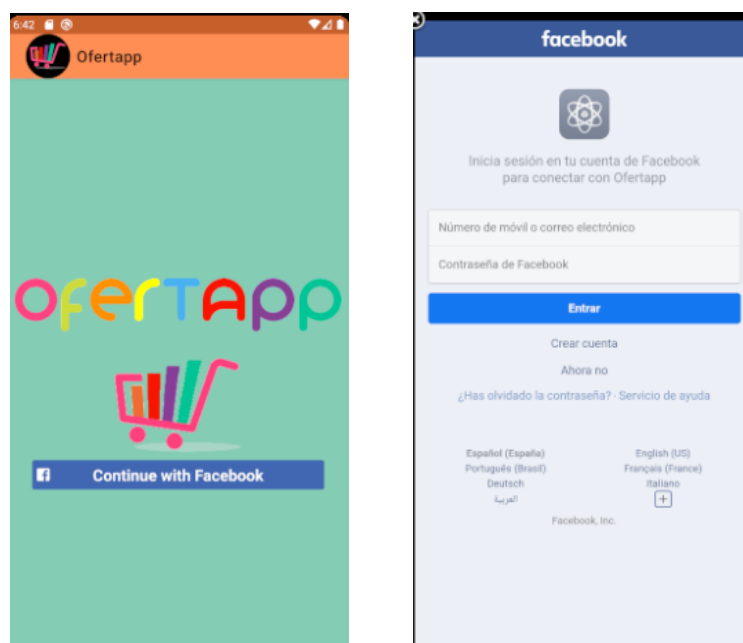
	id	precio	marca	categoria	tipo	comercio	fecha_venc	latitud	longitud
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	1	157.5	Casancrem	Alimentos	Queso untable	Monarca	2021-03-06	-37.326517140532744	-59.13760419934988
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	2	106.5	Yogurísimo	Alimentos	Yogurt	Monarca	2021-04-14	-37.32614308688269	-59.13761157542467
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	3	39.99	Dia	Alimentos	Cerveza	Dia	2023-03-25	-37.31759186728301	-59.13448512554169
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	4	59.99	Pureza	Alimentos	Harina	Carrefour	2023-12-14	-37.32621240558991	-59.135106727480895
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	5	52	Bagley	Alimentos	Galletitas	Carrefour	2025-12-12	-37.32615961673407	-59.13513958454132
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	6	29.99	Higienol	Limpieza	Papel higienico	Dia	2024-02-13	-37.31757746867446	-59.13449250161648
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	7	99.99	Carrefour	Alimentos	Carne	Carrefour	2022-02-13	-37.32393365289424	-59.11442823708057
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	8	19	Dos hermanos	Alimentos	Arroz	Carrefour	2022-04-15	-37.32389579307552	-59.11444164812565
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	9	180	Amanda	Alimentos	Yerba	Vea	2022-04-29	-37.3270767511767	-59.13238260895014
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	10	25	Molto	Alimentos	Fideos	Vea	2022-04-24	-37.327011165771346	-59.13239803165198
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	11	39	Sancor	Alimentos	Leche	Monarca	2028-07-01	-37.31730069488511	-59.1350819170475
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	12	5	Arcor	Alimentos	Turrón de mani	Golopolis	2021-03-25	-37.32227793454809	-59.117930196225636

Finalmente, en las clases **LocationService** línea 158, **BuscarOferta** línea 68 y **AgregarOferta** línea 142 hay que colocar nuestra dirección IP para poder realizar las consultas a la base de datos mediante los archivos php antes mencionados.

## Funcionamiento

En este apartado nos dedicaremos a explicar cómo fue implementado y desarrollado cada aspecto de la aplicación, desde la composición de las activitys hasta el funcionamiento y relación entre las mismas. Una aclaración importante es que la aplicación fue testada en un Google Pixel 3a con API 29.

En primer lugar, se nos abrirá la primer activity de la aplicación llamada **MainActivity**. La misma está compuesta por dos Imageview y un botón de login de Facebook, el cual nos abre una ventana para iniciar cesión con esta red social y pasa a la siguiente activity. El inicio de cesión no es obligatorio para el correcto funcionamiento de la aplicación.



Luego nos encontraremos con la activity **Menu** que está compuesta por tres botones en donde cada uno nos conducirá a una activity distinta, para realizar diversas acciones. A continuación, explicaremos cada una en detalle.



El primer botón, nos lleva a la activity **AgregarOferta** que, como se puede ver, nos permite agregar una oferta especificando su precio, marca, categoría, tipo, comercio, vencimiento y localización.

A screenshot of the 'Agregar Oferta' (Add Offer) form in the Ofertapp application. The form is displayed on a teal background. It features several input fields with placeholder text and examples: 'Precio' (5), 'Ej: \$2.5', 'Marca' (Arcor), 'Ej: Arcor-Nike-Pepsi', 'Categoría' (Alimentos), 'Ej: Alimentos-Librería-Tecnología', 'Tipo' (Turrón de mani), 'Ej: Harina-Agua-Azúcar', 'Comercio' (Golopolis), 'Ej: Monarca-Carrefour-Dia', 'Vencimiento' (2021/3/25), and 'Fecha de finalización de la oferta'. Below the form is a yellow button with a location pin icon and the text 'AGREGAR'.

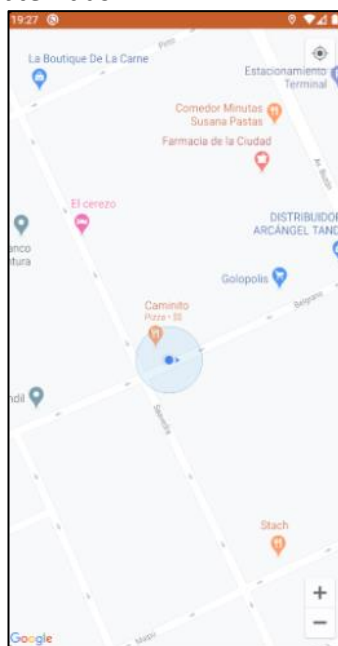
Las primeras 5 características se introducen en varios **editText** mediante el **teclado**, pero no sucede lo mismo para las últimas dos.

Al cliquear en la casilla de vencimiento, se nos abrirá un **DatePickerDialog** el cual nos permite introducir la fecha con la ayuda de un calendario; esto se realizó así ya que, si fuera por teclado, el usuario podría no respetar el formato de la fecha, además de que mejora la usabilidad de la aplicación porque es más fácil y cómodo introducir la fecha de vencimiento.



Luego tenemos un **ImageButton** que nos permite, mediante el método **startActivityForResult**, iniciar **MapsActivity** y cuando el usuario seleccione una ubicación volver a la anterior activity automáticamente y poder usar la ubicación para cargarla en la base de datos.

Esto se realizó así ya que se requería que el usuario ingrese la ubicación del comercio, pero de un modo amigable, ya que introducir la latitud y longitud con **editText** nos pareció muy complicado y tedioso. En cambio, de esta manera se abre una activity que muestra un mapa y nos ubica automáticamente donde estemos nosotros (pidiendo los permisos correspondientes para hacerlo) y luego de que el usuario mantenga pulsado sobre una ubicación en el mapa, se vuelve a la activity que lo llamó, especificando en **onActivityResult** la copia de la latitud y longitud obtenidas.



Una vez que el usuario introdujo valores en los campos se pulsa agregar y se ingresa la oferta en la base de datos mediante el **registrar.php**.

Por otro lado, la activity **BuscarOferta** nos da la posibilidad de especificar filtros y buscar las ofertas en la base de datos. Nuevamente cada filtro se introduce en un **editText** mediante el teclado.

Vale aclarar que, en caso de no especificar ningún filtro, se buscan todas las ofertas de la base de datos, es decir, no son obligatorios.



Al presionar en el botón **buscar**, se recuperan los datos mediante un **StringRequest** (especificándole la URL con el **buscarmod.php** y demás parámetros) y los métodos **onResponse** y **onErrorResponse**. En caso de error se muestra un mensaje con el error correspondiente. Caso contrario, se crea un **JSONArray** inicializándolo con el string de devuelto en la respuesta. Luego se recorre dicho arreglo construyendo una lista de ofertas y se inicializa la activity **Listar\_ofertas** pasándole dicha lista.



Esta activity está compuesta por un **RecyclerView**, que nos lista las ofertas recuperadas desde la base de datos.

Para esto, se crea un **Adapter** al cual se le pasa la lista de ofertas como parámetro y se configura que al hacer click en alguna tarjeta nos lleva a otra activity, pasando, a su vez, las características de la oferta.

El adaptador se asocia a un **RecyclerView** mediante el método **setAdapter**, permitiendo personalizar el diseño de cada ítem de la lista. El mismo consta de un **CardView** que contiene el tipo, precio y un texto ver detalles que nos indica que al pulsar la tarjeta podemos ver en detalle las características de la oferta.

El adaptador, que extiende de **RecyclerView.Adapter**, administra los objetos contenedores de vistas. El adaptador crea contenedores de vistas, según sea necesario, y los vincula con los datos de las ofertas. Para hacerlo, se le asigna el contenedor de vistas a una posición y llama al método **onBindViewHolder** del adaptador. Este método usa la posición del contenedor de vistas para determinar cuál debería ser el contenido, en función de su posición en la lista.

Al clicar en una tarjeta, se abre la activity **seeDetails** que nos muestra los atributos de la oferta junto con su localización en el mapa. Se utilizó un **TableLayout** para ubicar los **ImageView** y **TextView** por filas y abajo un mapa provisto por Google.

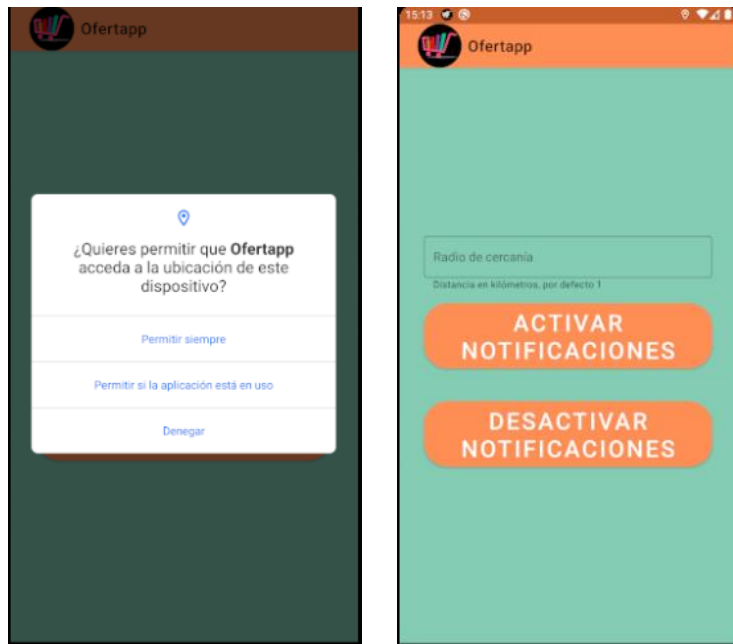


Finalmente pulsando el botón de notificaciones nos dirigimos a **notificationsActivity**. El propósito de la misma es poder realizar la activación o desactivación de las notificaciones periódicas de ofertas que se encuentran próximas a nuestra posición actual.

La activity cuenta con un **editText** para especificar el radio en kilómetros a considerar a la hora de buscar ofertas cercanas (por defecto es 1km), junto con dos botones para activar y desactivar las notificaciones.

A su vez, se le solicita al usuario que permita el uso de su ubicación para poder recomendar las ofertas.



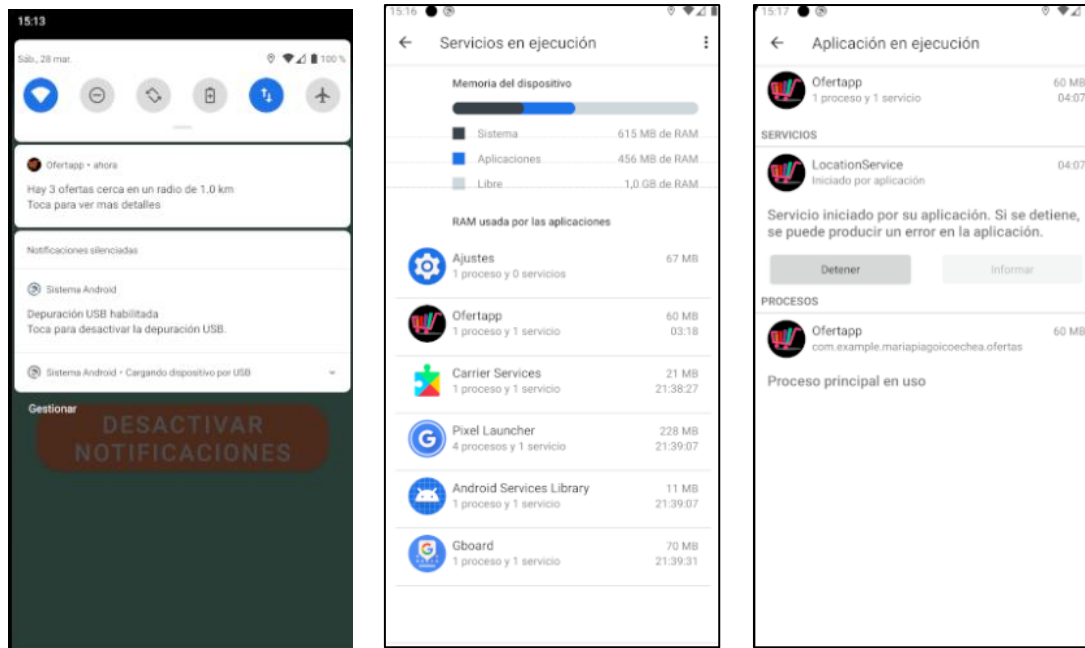


Al presionar el botón de activación, se da comienzo a el servicio de localización llamado **LocationService**; el cual se encarga de obtener la localización periódicamente y hacer una consulta en la base de datos; del mismo modo que cuando se buscan ofertas por filtro, pero en este caso según la latitud, longitud y distancia máxima a la oferta, por lo que también se utiliza otro archivo (**buscarPorLocalizacion.php**) para la consulta.

Vale aclarar que el intervalo de tiempo entre notificaciones se especifica en la constante **UPDATE\_INTERVAL\_IN\_MIL**, cuyo valor es 10 segundos.

La notificación se configura en el método **getNotification**, en donde se declara un **pendingIntent** para que al pulsar sobre la misma nos lleve a **Listar\_ofertas**, mostrar las ofertas obtenidas y también poder ver los detalles si así se desea. De esta forma se reutilizan clases ya creadas, sin la necesidad de crear nuevas activities similares.

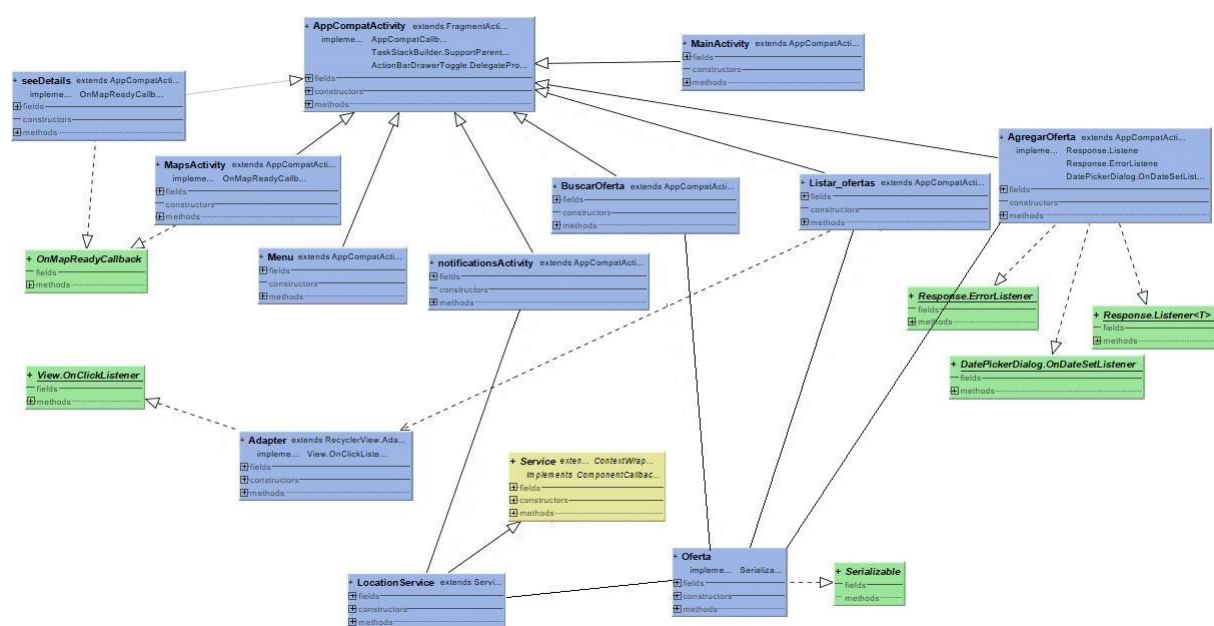
Esta funcionalidad fue implementada a través de un servicio ya que nos permite, aún sin la aplicación abierta o una determinada activity en primer plano, obtener actualizaciones sobre la ubicación del usuario y, por ende, de ofertas cercanas al mismo. Como se puede ver en las siguientes imágenes, el servicio se ejecuta en segundo plano, aún con la aplicación cerrada y nos llegan las notificaciones al dispositivo. En la notificación se muestra la cantidad de ofertas y el radio de cercanía en kilómetros. Si modificamos la ubicación del emulador la notificación se actualizará.



## Diagrama de clases

A continuación, se puede observar el diagrama de clases de la aplicación. Las **activities** heredan de la clase **AppCompatActivity**, las dos que utilizan mapas implementan de **onMapReadyCallback**, **Adapter** implementa de **View.OnClickListener** para poder tener la funcionalidad de al tocar una tarjeta realizar una acción y el servicio **LocationService** hereda de **Service**.

Con respecto a la clase **Oferta** y las relaciones con la misma, primero decir que implementa **Serializable** para poder pasar una lista de ofertas entre **activities**; **LocationService** y **BuscarOferta** crean una lista de ofertas mediante una consulta a la base de datos, **AgregarOferta** justamente ingresa una oferta a la BD y **Listar\_ofertas** despliega tarjetas con ofertas en la pantalla.



## Conclusión

En conclusión, mediante este trabajo se desarrollaron los contenidos vistos en la materia, desde los más básicos como crear y diseñar un activity hasta la utilización de servicios para procesos en segundo plano, entre otros conceptos vistos en el desarrollo de este trabajo como iniciar una actividad para obtener un resultado. Se trabajó además con la conexión a una base de datos para almacenar las ofertas y se explicó en detalle el funcionamiento y composición de la aplicación.