VIVERSIDAD

DESARROLLO DE APLICACIONES MOVILES

Antonio Lorenzana Fabio Enrique.
Luna Vazquez Gustavo
Pelcastre Vargas Uriel

12/12/2020

INDICE

Introducción.

El proyecto está basado en definir y mostrar cómo funciona una aplicación para dispositivos móviles (smartphone) identificando cada uno de sus componentes que la integran, así como su diseño y funcionalidad.

Para el diseño de aplicaciones se usa el termino técnico de (UI) el cual hace referencia a la interfaz de usuario, para el comercio de plantas “Viversidad” al ser una aplicación móvil el tipo de diseño de UI a implementar es de software la cual se basa en brindar información relevante acerca de la aplicación para ello también se implementan widgets, los cuales son aplicaciones miniatura que se muestran dentro de la UI en los cuales el comercio de plantas visualizara información que sea de su interés, como pueden ser, inventario, pedidos, contactos, transacciones.

De igual manera el desarrollo de la UI se implementan características como es la **Claridad** donde la interfaz muestra información de manera precisa para evitar que el encargado del comercio de plantas cometa errores durante la interacción, es **Concisa** y se da al encargado solo la información que necesita ver de la aplicación ya sea ingresando a ella o a través de un widget, es **Flexible** donde la interfaz permite que el dueño del comercio de plantas restaure elementos y deshaga acciones dentro de la aplicación.

Mientras que la parte funcional de la aplicación se muestra cómo se lleva a cabo el control de servicios ofrecidos del comercio de plantas “Viversidad “generando diferentes módulos los cuales permitirán al dueño tener un mejor manejo de los datos como son sus clientes, pedidos, inventario, métodos de pago.

Objetivo general.

Desarrollar una aplicación móvil para smartphones que permita el control del servicio de un comercio de plantas “Viversidad”, mediante Android Studio y Firebase para que el dueño del comercio lleve sus registros dentro de su dispositivo móvil como es su inventario, contacto cos sus clientes, pedidos, compras realizadas, envíos.

Justificación.

Las aplicaciones móviles son actualmente indispensables en los smartphones debido a esto el diseño de una aplicación administrativa que lleve el registro de un comercio de plantas, tiene el propósito de contribuir al manejo de información dentro de un smartphone, para integrar el comercio al mundo digital. La realización del proyecto contribuirá así mismo a mejorar el manejo de información de manera más sencilla por parte del dueño y optimizar sus procesos administrativos del comercio de plantas, atenderá a una demanda y una necesidad de poder llevar la información de manera de manera portable en cuando a su inventario, pedidos, y los pagos realizados dentro del comercio.

Es conveniente que se lleve a cabo y finalice este proyecto para optimizar el manejo de la información de manera más simple y sin pérdida de la misma de los clientes que hasta el momento han realizado alguna compra, así como el inventario restante que queda dentro del comercio.

Manual técnico.

Requerimientos técnicos.

Software.

- **Firebase:** Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles desarrollada por Google
- **Android Studio:** Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android.

Hardware.

- **Dispositivo Android:** Dispositivo Android que pueda correr correctamente las aplicaciones Android.

Implementaciones.

- **Analytics:** Solución gratuita de medición de apps que proporciona estadísticas sobre el uso de las apps y la participación de los usuarios.
- **Database:** Almacena y sincroniza datos con nuestra base de datos NoSQL alojada en la nube. Los datos se sincronizan con todos los clientes en tiempo real
- **Design:** Recomendadas para la estructuración de los datos JSON de tu Firebase Realtime Database.

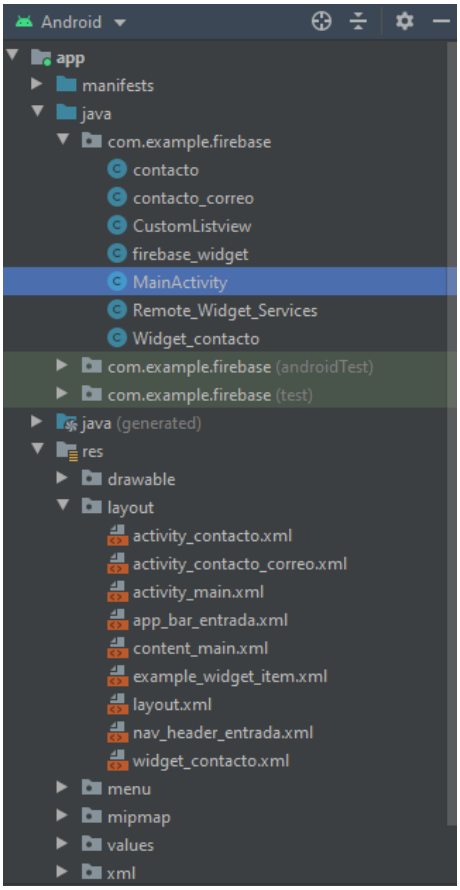
Archivos necesarios.

- **Google-services:** Archivo de configuración de Firebase para Android a la app.

Android Studio.

- **Activity.**

  Se muestran los Activitys necesarios, así como sus diseños que compone la aplicación Viversidad.



- **Contacto.**

  Se visualiza el método que nos trae los datos de la base de datos con sus respectivos campos y se guarda en un ListView.

```
mref.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(@NonNull DataSnapshot dataSnapshot, @Nullable String s) {
        String value = dataSnapshot.child("nombre").getValue(String.class);
        String value2 = dataSnapshot.child("email").getValue(String.class);
        String value3 = dataSnapshot.child("mensaje").getValue(String.class);
        myarray.add(value+" ;"+value2+" ;"+value3);
        notification();
        myadapter.notifyDataSetChanged();
    }

    @Override
    public void onChildChanged(@NonNull DataSnapshot dataSnapshot, @Nullable String s) {
        myadapter.notifyDataSetChanged();
```

- **Firebase_widget.**

  Se obtiene los datos desde firebase para que desde el widget pueda visualizarse.

```
public class firebase_widget {
    DatabaseReference mref;
    String []myarrays ={"","","","","",""} ;
    private ArrayList<String> myarray = new ArrayList<>();
    firebase_widget(){
        mref = FirebaseDatabase.getInstance().getReference().child("users-list");
       mref.addChildEventListener(new ChildEventListener() {
            @Override
            public void onChildAdded(@NonNull DataSnapshot dataSnapshot, @Nullable String s) {
                String value = dataSnapshot.child("nombre").getValue(String.class);
                String value2 = dataSnapshot.child("email").getValue(String.class);
                String value3 = dataSnapshot.child("mensaje").getValue(String.class);
                myarray.add(value+" ;"+value2+" ;"+value3);
               myarray.toArray(myarrays);

            }
```

- **Remote_widget_services.**

  Permite insertar la consulta de la base de datos en las tarjetas del diseño widget a través de un arreglo.

```
public class Remote_Widget_Services extends RemoteViewsService {

    public RemoteViewsFactory onGetViewFactory(Intent intent) {
        return new ExampleWidgetItemFactory(getApplicationContext(), intent);
    }
    class ExampleWidgetItemFactory implements RemoteViewsFactory {
        firebase_widget f = new firebase_widget();
        private Context context;
        private int appWidgetId;
        private String[] exampleData = f.myarrays;


        ExampleWidgetItemFactory(Context context, Intent intent) {
            this.context = context;
            this.appWidgetId = intent.getIntExtra(AppWidgetManager.EXTRA_APPWIDGET_ID,
                    AppWidgetManager.INVALID_APPWIDGET_ID);
        }
        @Override
        public void onCreate() {
            //connect to data source
            SystemClock.sleep( ms: 3000);
        }
```

- **Widget_contacto.**

  Esta es la ventana que corresponde al widget con los Intent y dirigiéndonos a la vista de servicios de widget.

```
static void updateAppWidget(Context context, AppWidgetManager appWidgetManager,
                            int appWidgetId) {


    // Construct the RemoteViews object
    RemoteViews views = new RemoteViews(context.getPackageName(), R.layout.widget_contacto);
     views.setTextViewText(R.id.example_widget_text, text: "Viversidad");

    Intent serviceIntent = new Intent(context, Remote_Widget_Services.class);
    serviceIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, appWidgetId);
    serviceIntent.setData(Uri.parse(serviceIntent.toUri(Intent.URI_INTENT_SCHEME)));
    views.setRemoteAdapter(R.id.example_widget_stack_view,serviceIntent);
    views.setEmptyView(R.id.example_widget_stack_view,R.id.example_widget_empty_view);

    // Instruct the widget manager to update the widget
    appWidgetManager.updateAppWidget(appWidgetId, views);
}
```

- **Contacto_correo.**

  Permite el consumo de servicio de correo electrónico en una API a través de una petición post con el protocolo http.

```
public void sendPost(final String nom, final String email, final String mensaje) {
    Thread thread = new Thread((Runnable) () -> {
            try {
                URL url = new URL(urlAdress);
                HttpURLConnection conn = (HttpURLConnection) url.openConnection();
                conn.setRequestMethod("POST");
                conn.setRequestProperty("Content-Type", "application/json");
                conn.setDoOutput(true);
                conn.setDoInput(true);

                JSONObject jsonParam = new JSONObject();
                jsonParam.put( name: "nombre",nom);
                jsonParam.put( name: "email",email) ;
                jsonParam.put( name: "mensaje", mensaje);


                Log.i( tag: "JSON", jsonParam.toString());
                DataOutputStream os = new DataOutputStream(conn.getOutputStream());
                //os.writeBytes(URLEncoder.encode(jsonParam.toString(), "UTF-8"));
                os.writeBytes(jsonParam.toString());

                os.flush();
                os.close();

                Log.i( tag: "STATUS", String.valueOf(conn.getResponseCode()));
                Log.i( tag: "MSG" , conn.getResponseMessage());
```

- **MainActivity.**

  Nos redirige a la vista de contacto además declaramos el uso del menú lateral, además de una imagen que se observa en dicha lista.

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

    foto1=(ImageView)findViewById(R.id.foto1);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    final View header =((NavigationView)findViewById(R.id.nav_view)).getHeaderView( index: 0);
    final ImageView photo=(ImageView)header.findViewById(R.id.imageView);
    ((TextView)header.findViewById(R.id.textus)).setText("Fabio Antonio");
    ((TextView)header.findViewById(R.id.textmail)).setText("ing.fabio.a@gmail.com");
    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
            activity: this, drawer, toolbar, "Open navigation drawer", "Close navigation drawer");
    drawer.addDrawerListener(toggle);
    toggle.syncState();



    NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
    navigationView.setNavigationItemSelectedListener(this);
```

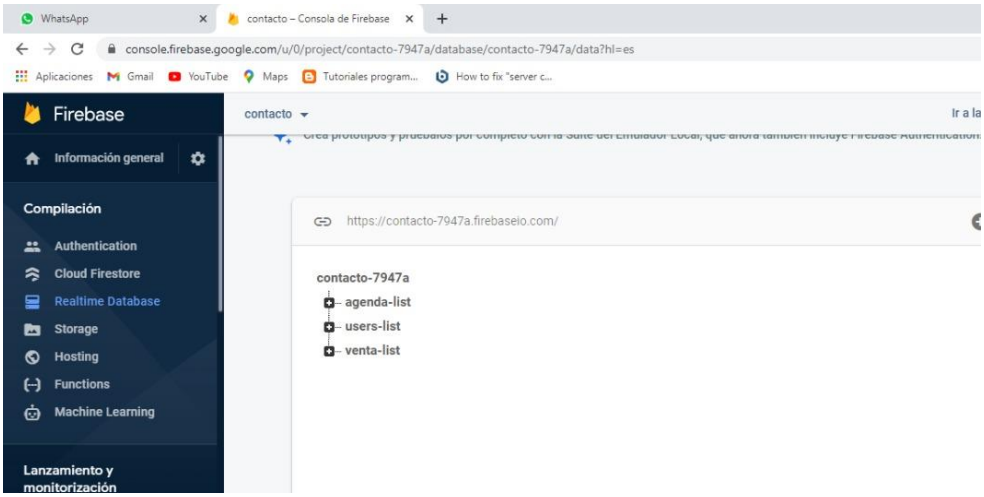
- **AndroidManifest.**

  En el manifest tendremos que darle permiso a internet, información sobre redes, que la aplicación lea desde un almacenamiento externo y que escriba desde un almacenamiento externo y que se pueda ejecutar en primer plano la notificación.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firebase">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="firebase"
```

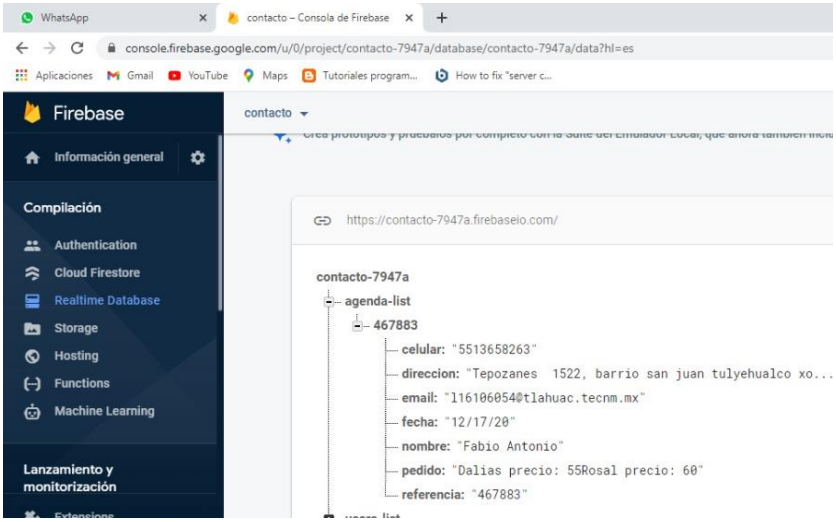
Firebase.

Se muestran los apartados de la base de datos en firebase Realtime Database en donde se alojarán los pedidos, usuarios y ventas de nuestra página web y aplicación móvil.



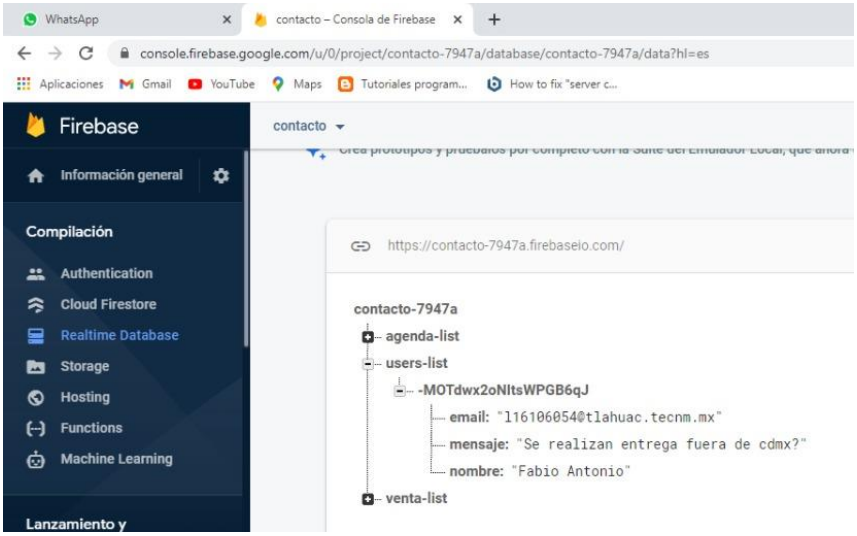
- **Agenda-list.**

  Se mostrarán los datos del cliente, así como su referencia de pago anexando el pedido que el cliente desea. Esta información deberá de llegar a esta base de datos, en el apartado de agenda-list y en la aplicación móvil.



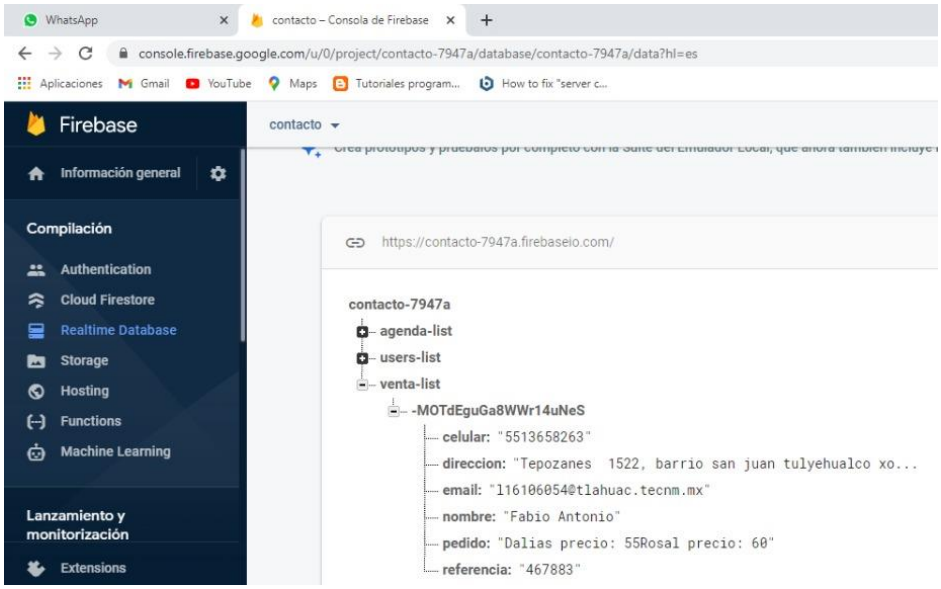
- **Users-list.**

  Se muestran los datos del cliente, además de dudas, preguntas o mensajes que los clientes quieran enviar al usuario vendedor.



- **Venta-list.**

  En el apartado de ventas se observará las ventas que se han hecho, este contara con la información del comprador y el pedido que el requirió.



Pruebas y resultados.

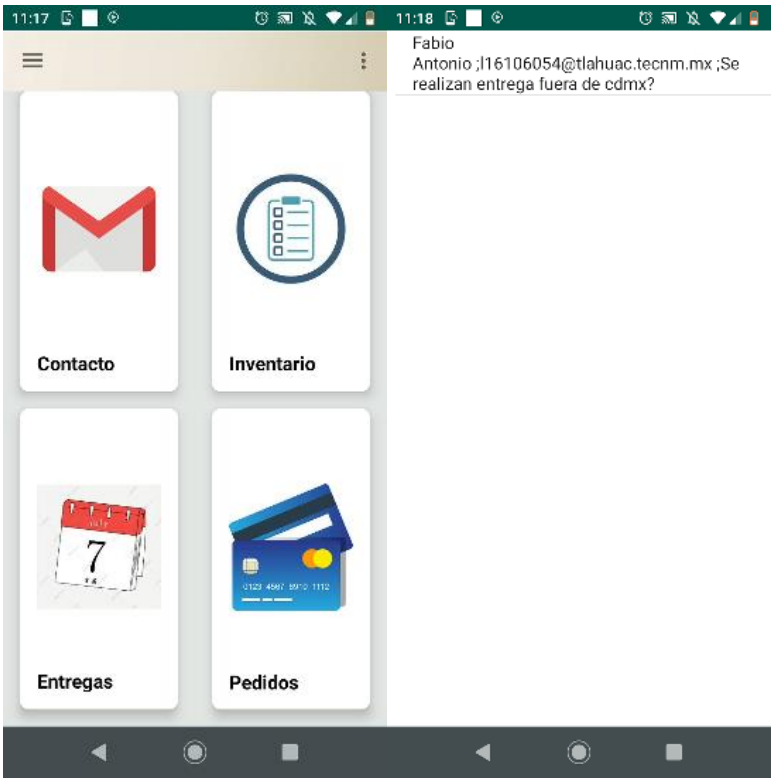
Unidad 3: Construcción y diseño de la interfaz de usuarios

En el presente apartado se realizan las pruebas de funcionamiento de los diferentes componentes de la interfaz de usuario de la aplicación, en los que se destaca el uso de eventos, widgets, fragmentos y la implementación de la herramienta Material Design para el desarrollo de una interfaz más amigable.

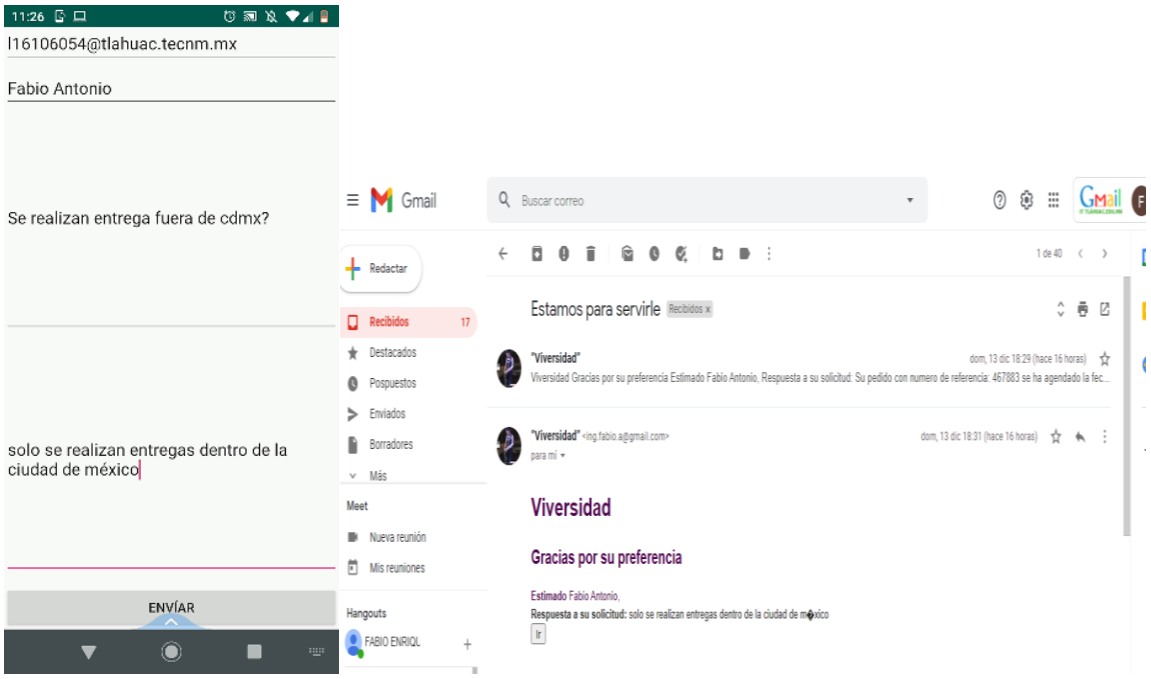
- **Eventos**

  El desarrollo de la presente aplicación cuanta con tres diferentes tipos de eventos que se distribuyen a través de los Activitys, cumplen con diferentes funciones como navegación, activación de fragmentos, consultas a la base de datos y el consumo de servicios. Estas funciones son explicadas a detalle en las siguientes pruebas de eventos.

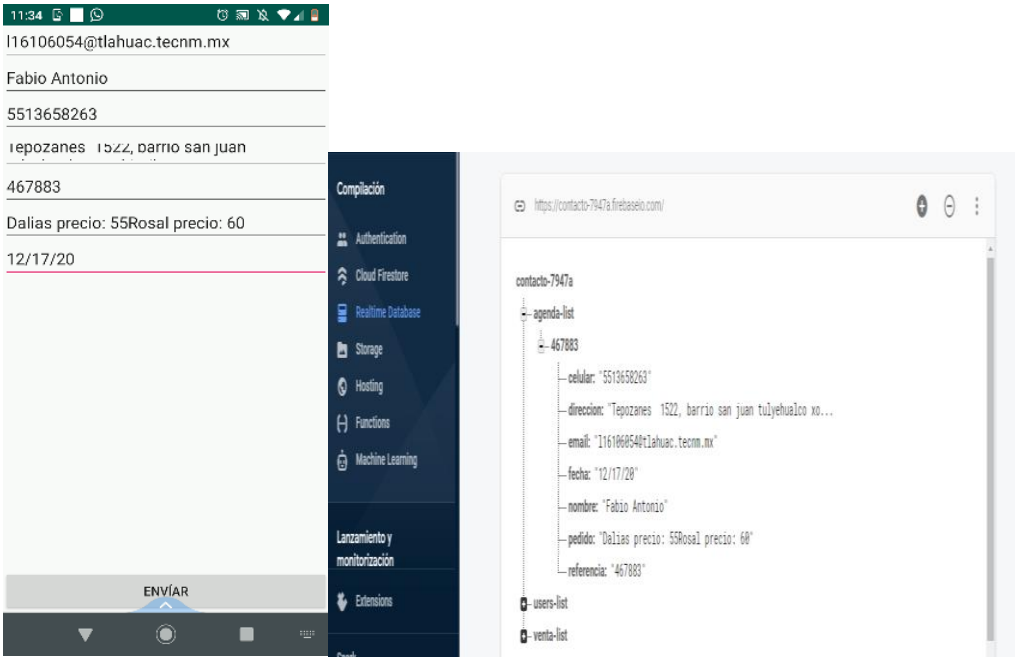
OnClick: Este evento es utilizado a través de la aplicación para tres diferentes propósitos, el primero de ellos, consiste en la navegación de los diferentes Activitys relacionados con las tarjetas implementadas en el menú principal. En la se muestra las siguientes imágenes un ejemplo de su funcionamiento en la tarjeta de contacto.



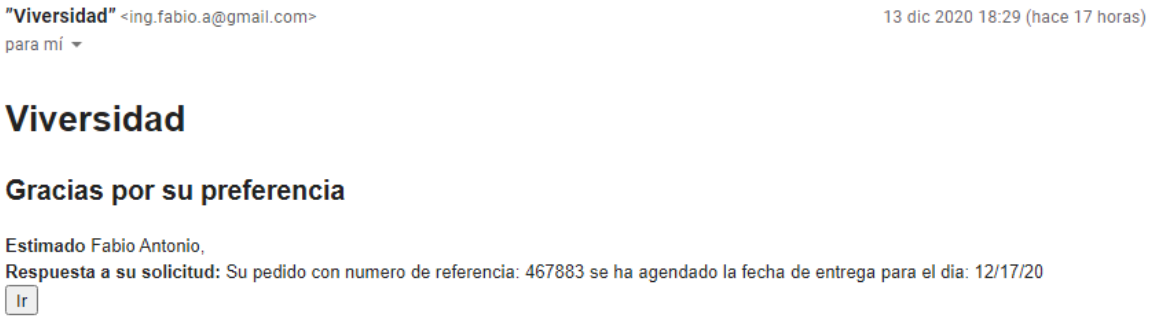
Otra de las funciones que realiza este evento en la aplicación, es el envío de datos en formato Json para el consumo de un servicio de correo electrónico a través de una API, en este caso es utilizado para contestar a las preguntas realizadas por el cliente, como se muestra en la siguiente prueba de las siguientes imágenes.



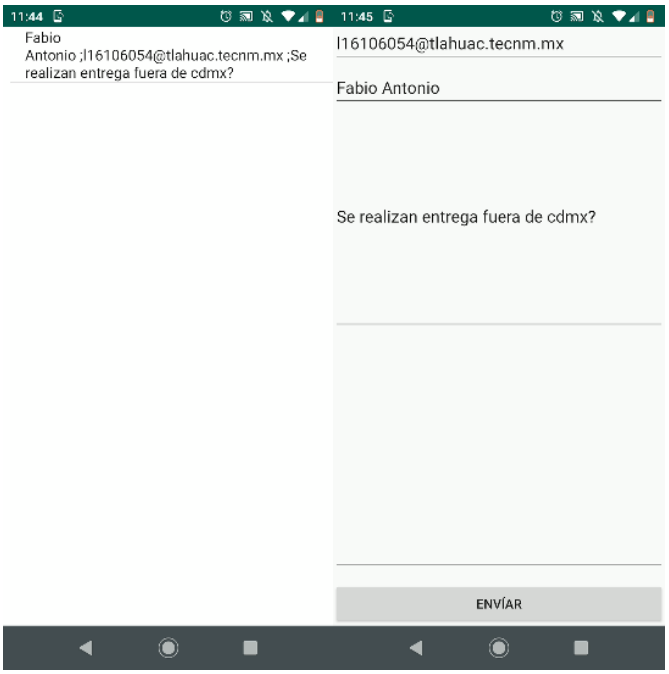
La última función del evento onClick dentro de la aplicación es la inserción en la base de datos remota, específicamente en el Activity que se encarga de asignar una fecha de entrega a un pedido pagado, como se muestra en la prueba de las siguientes imágenes.



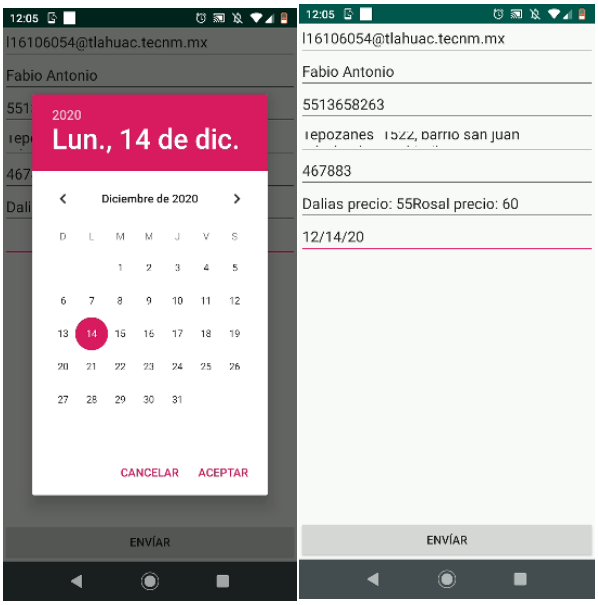
A su vez, esta función también consume el servicio de correo electrónico como se muestra en la siguiente imagen.



SetOnItemClickListener: Este evento es utilizado para realizar una acción al pulsar una opción de una lista en un ListView, específicamente en la aplicación es utilizado para enviar los datos de una opción seleccionada y enviarlos al siguiente Activity, como se muestra en la prueba de las siguientes imágenes.



setOnFocusChangeListener: Este evento es utilizado para ejecutar una acción cuando un determinado campo de un formulario es seleccionado, en el caso de esta aplicación se utiliza para desplegar un fragment para la selección de fechas para un pedido, como se muestra en la prueba de las siguientes imágenes.

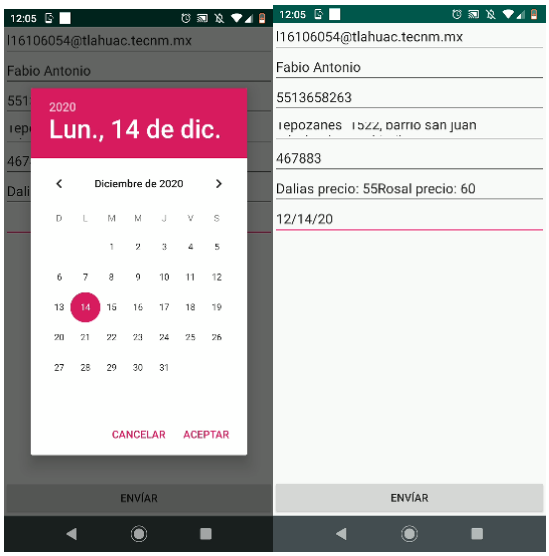


- **Widgets.**

  Esta aplicación cuenta con la implementación de un widget, el cual tiene la función de mostrar al usuario una lista con las cinco últimas inserciones en el módulo de preguntas y respuestas en el escritorio del dispositivo, de esta manera se tiene una visualización previa de las últimas preguntas realizadas por los clientes antes entrar a la aplicación, como se muestra en la prueba de la siguiente imagen.

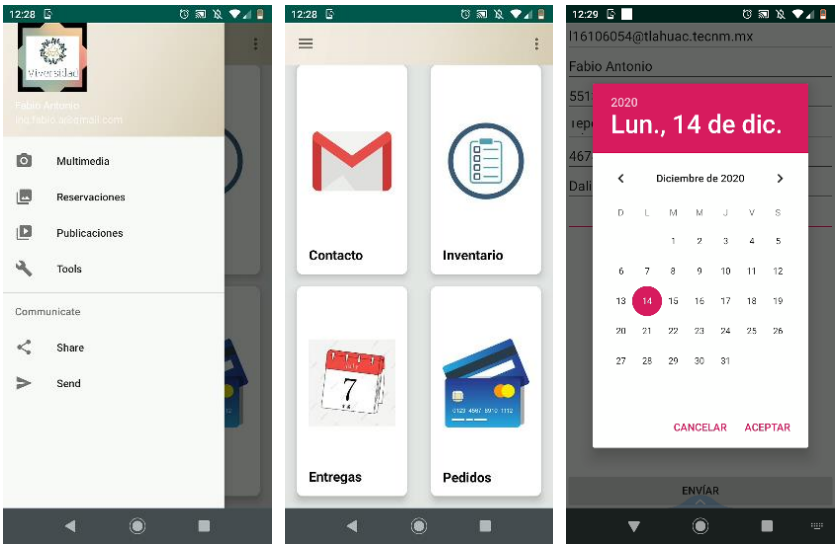
- **Fragment**

  Para el desarrollo de esta aplicación solo se tiene implementado el uso de un fragment, se encuentra en el Activity de asignación de fecha para un pedido. El fragmento se despliega al seleccionar el campo de fecha y permite al usuario seleccionar la fecha de entrega de manera amigable, como se muestra en la prueba de las siguientes imágenes.



- **Material Design.**

  El uso de la herramienta Material Design permite la implementación de diseños visualmente más amigables para el usuario. El objetivo principal es mejorar la experiencia de usabilidad de una manera más eficiente. En el caso de esta aplicación, el uso de esta herramienta se centra en tres diferentes componentes: El uso de CardViews para la navegación a los diferentes módulos, la implementación del menú lateral y el diseño del fragment para la selección de fechas a un pedido. Esto se puede ver en las pruebas de las siguientes imágenes.



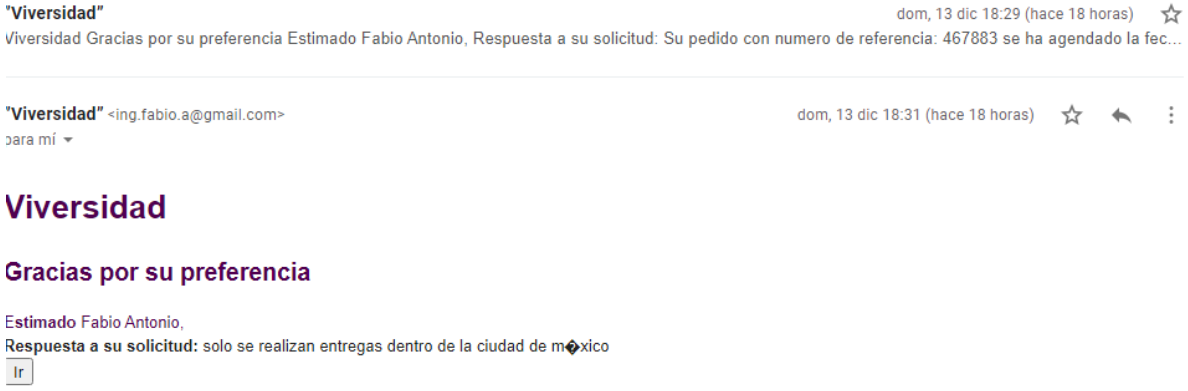
Unidad 4: Funcionalidad del Dispositivo

En el presente apartado se realizan las pruebas correspondientes a los puntos de la unidad 4, en donde se centra en el uso de servicios web, el tráfico de datos y la seguridad en las transacciones en la red.

- **Servicios Web.**

  En el desarrollo de esta aplicación solo se cuenta con el consumo de un servicio que se encuentra en red, corresponde al servicio para el envío de correo electrónico a los clientes de la tienda en línea, ya sea para responder sus preguntas o asignarles una fecha de compra. Este servicio es proporcionado por una API REST y es contactada por medio de una petición post por el protocolo HTTP y el envío de datos en formato Json, como se muestra en la prueba de las siguientes imágenes.

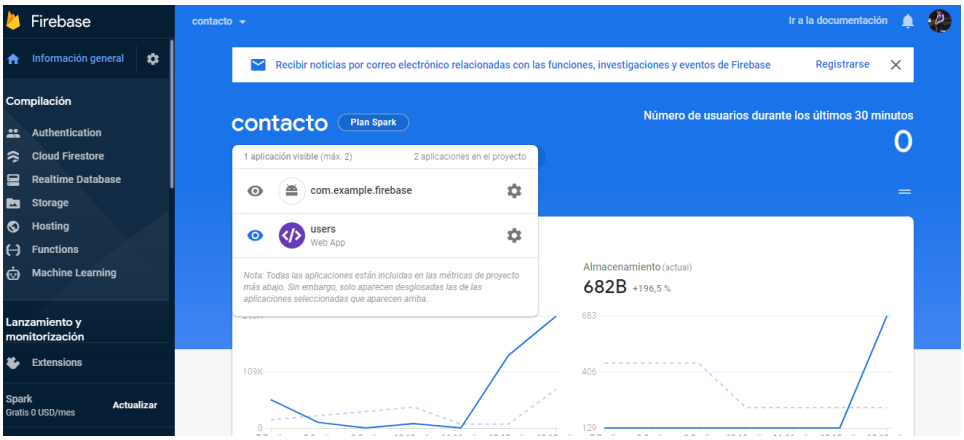
```
public void sendPost() {
    final String nom = nombre.getText().toString();
    final String emails = email.getText().toString();
    final String mensaje = "Su pedido con numero de referencia: "
            +referencia.getText().toString()+" se ha agendado la fecha de entrega para el dia: "+fecha.getText().toString();
    Thread thread = new Thread((Runnable) () → {
            try {
                URL url = new URL(urlAdress);
                HttpURLConnection conn = (HttpURLConnection) url.openConnection();
                conn.setRequestMethod("POST");
                conn.setRequestProperty("Content-Type", "application/json");
                conn.setDoOutput(true);
                conn.setDoInput(true);

                JSONObject jsonParam = new JSONObject();
                jsonParam.put( name: "nombre",nom);
                jsonParam.put( name: "email",emails) ;
                jsonParam.put( name: "mensaje", mensaje);


                Log.i( tag: "JSON", jsonParam.toString());
                DataOutputStream os = new DataOutputStream(conn.getOutputStream());
                //os.writeBytes(URLEncoder.encode(jsonParam.toString(), "UTF-8"));
                os.writeBytes(jsonParam.toString());

                os.flush();
                os.close();

                Log.i( tag: "STATUS", String.valueOf(conn.getResponseCode()));
                Log.i( tag: "MSG" , conn.getResponseMessage());
```

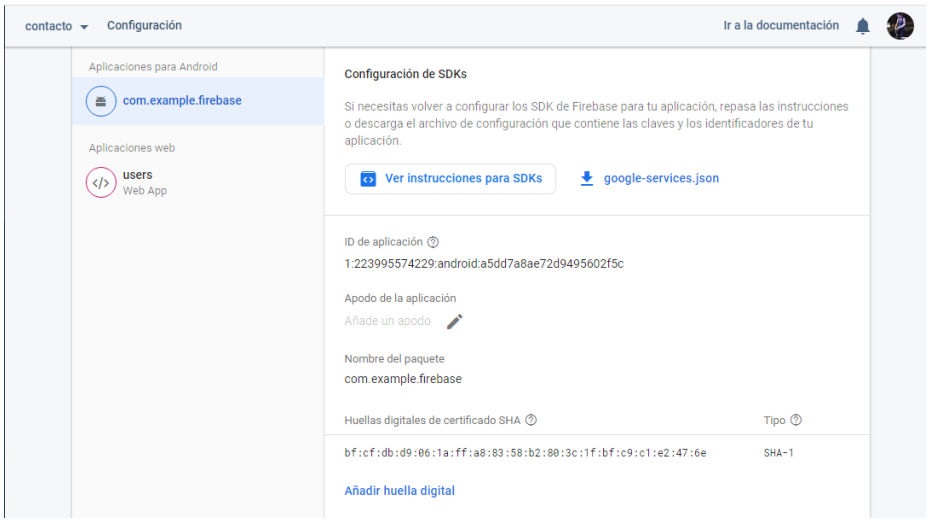


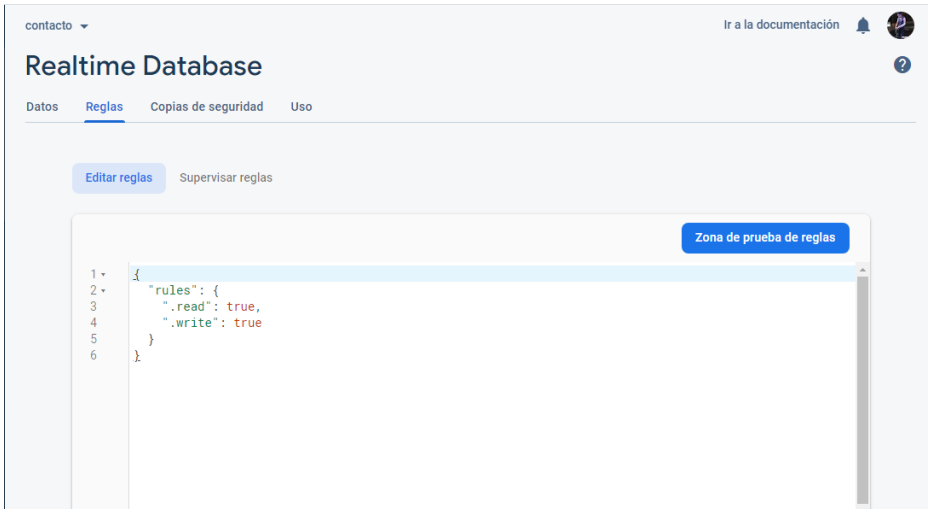
- **Seguridad de Red.**

  La seguridad implementada en esta aplicación está dada principalmente por el protocolo HTTP y su cifrado de datos SSL/TLS, el cual permite transacciones seguras entre los servicios de Google que corresponden a las consultas de las bases de datos y servicio de correo electrónico.

Para el uso seguro de la base de datos en tiempo real para el almacenamiento en la nube que ofrece Google, cuenta con algunos filtros de seguridad adicionales, como el uso de una llave para habilitar la conexión a diferentes aplicaciones y el uso de reglas que habilitan la lectura y escritura en la base de datos y certificado SHA, como se muestra en la prueba de las siguientes imágenes.

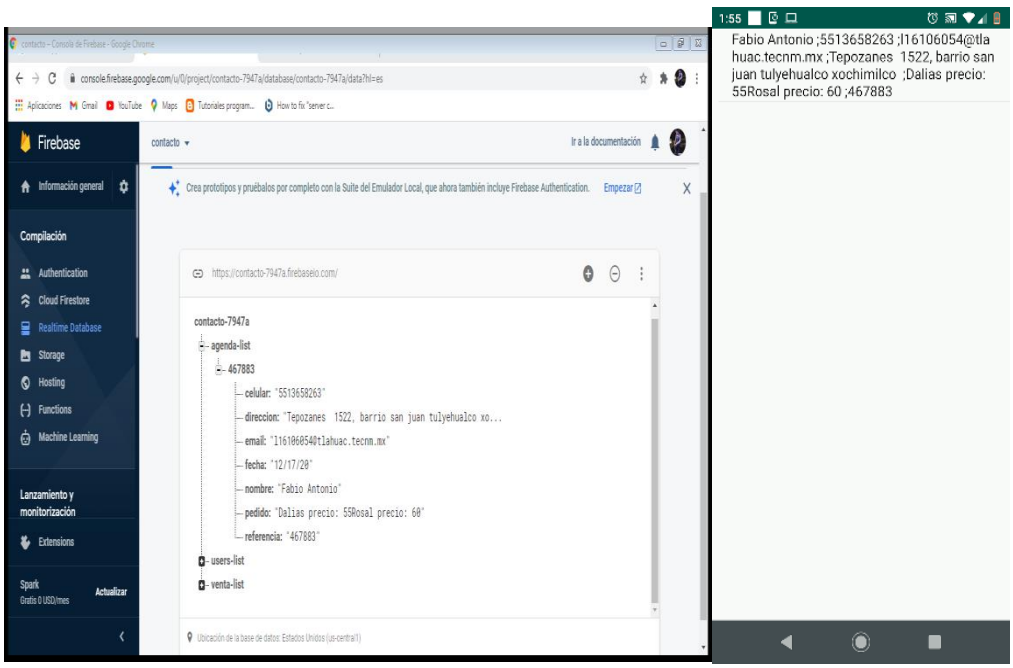




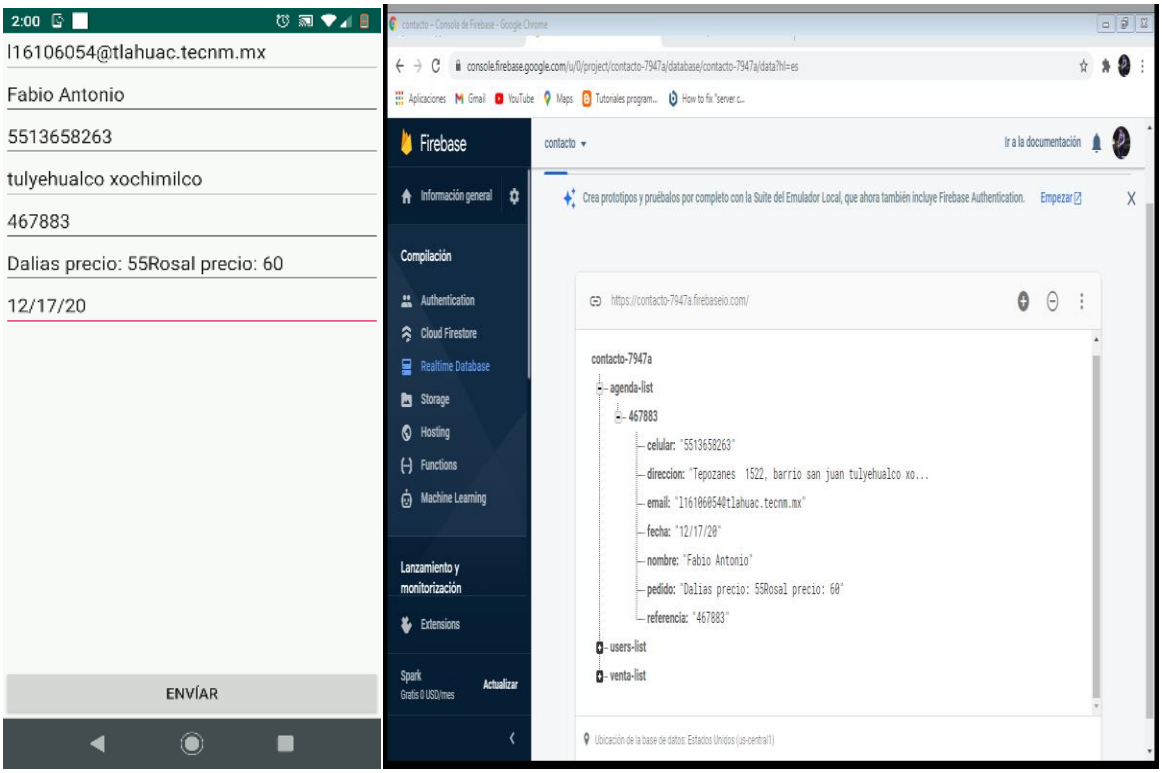


Unidad 5: Almacenamiento de información

En este apartado se presentan las pruebas correspondientes a los puntos de la unidad 5, en donde se abordan los temas referentes al almacenamiento de información de manera remota. En el caso de esta aplicación, la información se almacena en un servicio de almacenamiento en la nube, con protocolos en tiempo real de Google llamado firebase. La aplicación realiza dos tipos de consulta con esta base de datos, la primera corresponde a la consulta de los diferentes registros como se muestra en la prueba de las siguientes imágenes.



La segunda consulta que realiza la aplicación con la base de datos de firebase, es la inserción de datos, específicamente cuando se asigna una fecha a un pedido realizado por el cliente a través de la tienda en línea, como se muestra en la prueba de las siguientes imágenes.



Conclusión.

Conforme se realizó el proyecto se observan circunstancias que no se consideraron desde un principio. La importancia de saber las necesidades secundarias del dueño como las ventas totales realizadas en un periodo mensual para así llevar un reporte de ventas dentro de la aplicación, se detectó también puntos clave para afianzar muchos procesos, detectar áreas de oportunidad para mejorar la aplicación como es una sección de noticias de la información más utilizada por parte del dueño sin la necesidad de ingresar a ella mediante una nueva ventana.

La información es uno de los recursos más importantes que tienen los comercios de cualquier tipo hablando en general para cualquier ámbito y en ocasiones solo se basan en aplicaciones para los clientes pero las de tipo administrativas ayudan a los dueños de los comercios a llevar un mejor manejo de la información que tienen.

Hay muchas cosas por mencionar las cuales dejaron un aprendizaje a lo largo del proyecto, las más importantes son el tomar en cuenta todas las posibles necesidades del comercio por más mínimas que estas sean, de igual forma llevar a cabo antes que nada una planeación de lo que se quiere realizar y que se espera obtener cuando se lleve a cabo un proyecto, por ende, se debe desarrollar una evaluación correcta de las posibles alternativas que se tengan antes de iniciar cualquier cosa.