UNIVERSIDAD NACIONAL DE INGENIERIA

FACULTAD DE ELECTROTECNIA Y COMPUTACIÓN

Manual de usuario y programador de Aplicación Móvil para la Ubicación de Rutas del Transporte Urbano e Interurbano de Managua

Presentado por:

Br. Claudia Patricia Rivera Irías 2008-24148

Br. Daniel Guillermo Torres Martínez 2007-22305

Tutor:

Msc.Ing. Gloria Talía Flores Quintana

Managua, Nicaragua, Viernes 21 de Junio de 2019

I. Introducción:

El siguiente documento describe aspectos para el uso técnico de la Aplicación Móvil para teléfonos con sistema operativo Android bajo el nombre “RUTANIC” que es una Aplicación Móvil para la Ubicación de Rutas del Transporte Urbano e Interurbano de Managua, la cual brinda información sobre el transporte urbano colectivo de Managua e interurbano tales como buses que salen hacia los departamentos.

A continuación se detallan los pasos para el uso de la aplicación, las generalidades del diseño de la aplicación y explicación de códigos fuentes principales dentro de cada modulo que constituye la app asi también como el Diseño y funcionalidad del sitio administrativo para agregar, eliminar rutas etc. Los comentarios dentro del código fuente de las clases explican el uso de cada segmento de código y funciones específicas.

II. Manual de Usuario

II.1. Descripción:

RutaNic es una aplicación creada para teléfonos con sistema operativo Android, la cual te permite conocer cuáles son los trayectos y los lugares por los que pasan los buses del transporte urbano en Managua e interurbanos que salen hacia los departamentos.

Por medio de esta app podras tener a la mano información de una gran cantidad de rutas urbanas e interurbanas tales como el precio del pasaje, trayectoria que sigue la ruta, paradas etc.

Requerimientos:

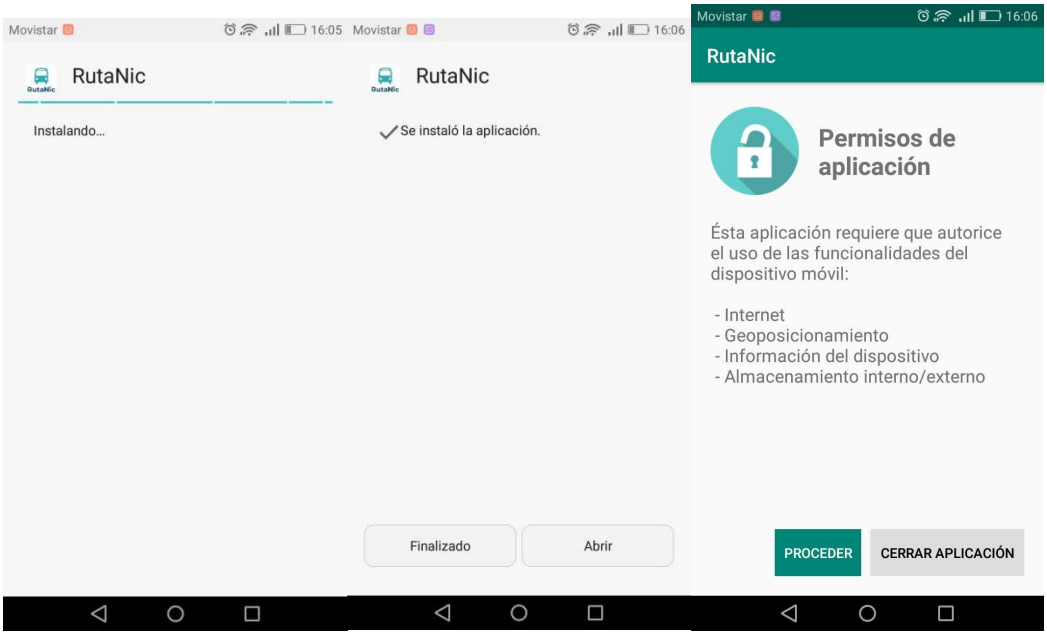
- Teléfono con sistema operativo Android versión 4.0.3 o superior
- Conexión a internet por medio de WiFi o paquete de datos.

II.2. Instalación

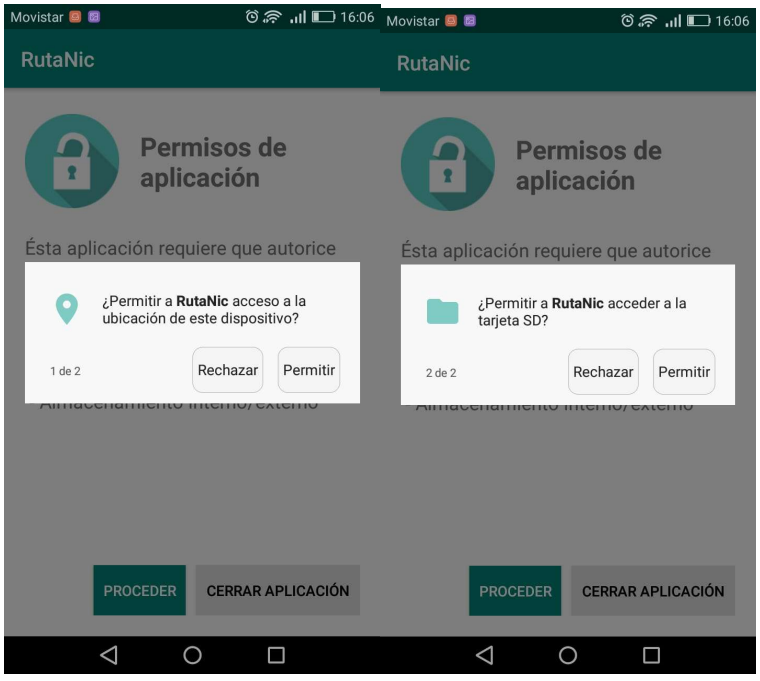
Para descargar la aplicación, se necesita acceder a la tienda de aplicaciones “Play Store” desde el dispositivo Android en donde se va a instalar. La gran mayoría de los teléfonos traen esta tienda de aplicaciones por defecto.

En ella se puede encontrar esta aplicación buscando “RutaNic” una vez que tenemos la app procedemos a instalar como cualquier otra app. Nos pedirá permiso para acceder a internet y al gps solo se los concedemos para poder usar la app.

Como las imágenes a continuación:

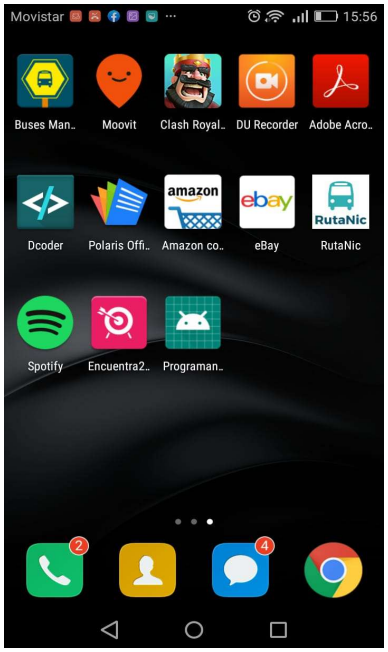


1. *Instalacion del APK*



2. *Concede Permisos*

Una vez descargada e instalada la aplicación, se puede acceder a ella desde el menú principal de nuestro dispositivo Android.



3. RutaNic Instalada

II.3. Como usar la aplicación

4. Interfaz del menú principal.

Menú Principal (Módulos):

Cómo Llegar: Búsqueda por petición de usuario, redireccióna en el mapa a solicitud.

Rutas Managua: Lista de Rutas de las trayectorias de transporte en el casco de Managua y lo muestra en el mapa.

Terminales: Información y ubicación en el mapa de las 5 terminales de las rutas interurbanas para la visualización de las paradas en el mapa, itinerarios y precios de Trayectoria de transporte hacia los departamentos.

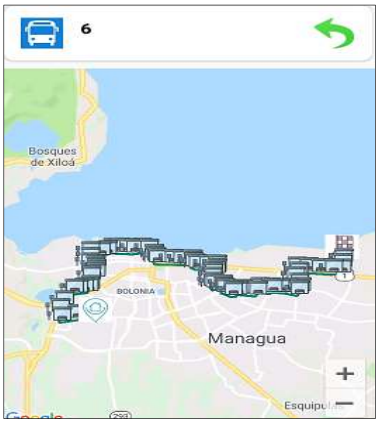
Acerca de: información de contacto de soporte de la app.

- Slider de imágenes ilustrativas relacionadas a las ciudades de Nicaragua.

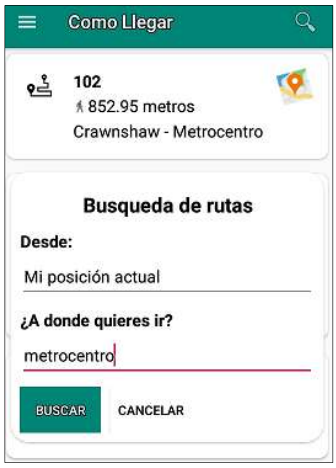
Interfaz Principal de la Aplicación:

- Como Llegar
- Rutas Managua
- Terminales
- Acerca

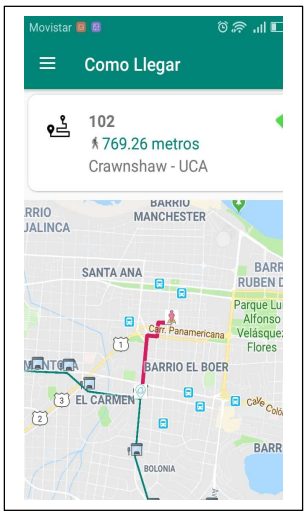
En la Opción del menú de Rutas Managua, se abrirá la interfaz que lista los buses de Managua, en la pestaña Rutas.

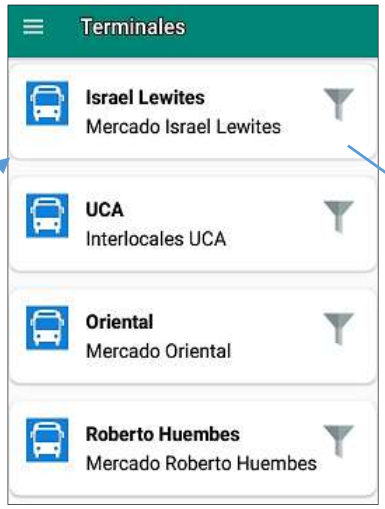


Aquí se puede ver en el mapa toda la trayectoria de la ruta seleccionada.



En la opción como llegar se abre una pantalla ingresamos un punto de partida y un punto de llegada la app calcula que rutas pasan cerca de tu ubicación e indica de manera informativa desde donde debes caminar hasta la parada o estación de bus. En la imagen se muestra en color magenta.





Retornando a la pantalla principal, seleccionamos la opción del menú Terminales y nos muestra una pantalla con la lista de las terminales de las rutas que van hacia de los departamentos para seleccionar el destino con la trayectoria en el mapa, clasificándolos en ordinario, expreso o interlocal según preferencia del usuario, en el cual se desplegara la siguiente pantalla con el itinerario el precio de la ruta y la trayectoria de las paradas. En la opción acerca solo se muestra información de los desarrolladores de la app.

II.4. Preguntas más frecuentes:

- ¿Cómo se si mi teléfono es compatible con esta aplicación?

Si la aplicación aparece en la tienda Play Store, significa que el dispositivo es compatible para instalarla.

-¿Por qué no encuentro algunas rutas en específico?

Introducir los datos dentro de la base de datos de buses conlleva un gran tiempo para recolectarlos y guardarlos. Con el tiempo se irán agregando más rutas a la aplicación.

-¿Por qué a veces toma tiempo en cargar cuando busco como llegar de un lugar a otro?

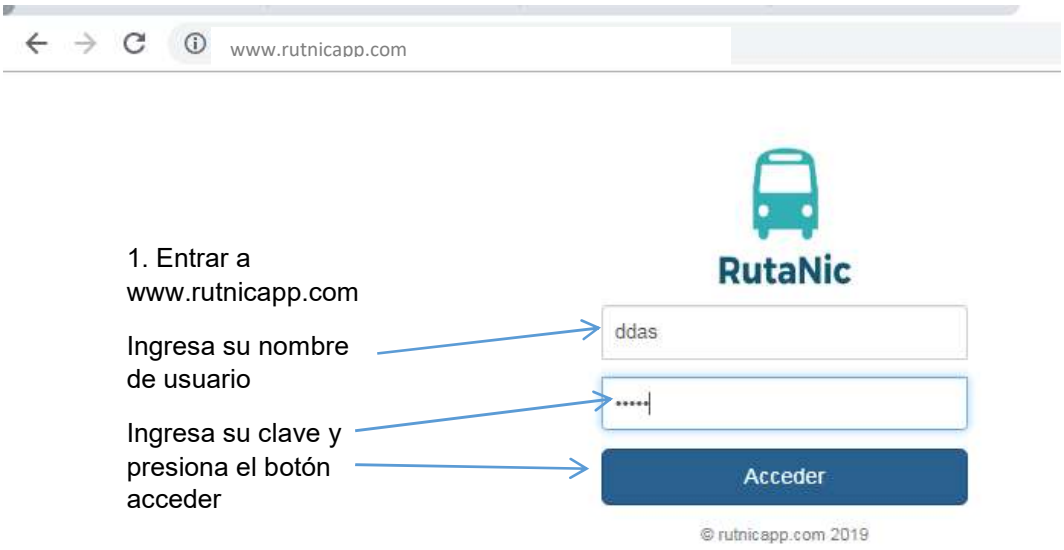
Buscar rutas usando un radio alto (1 km o más) puede conllevar a que la búsqueda tome un poco más de tiempo.

-¿Por qué no puedo usar la aplicación sin acceso a internet?

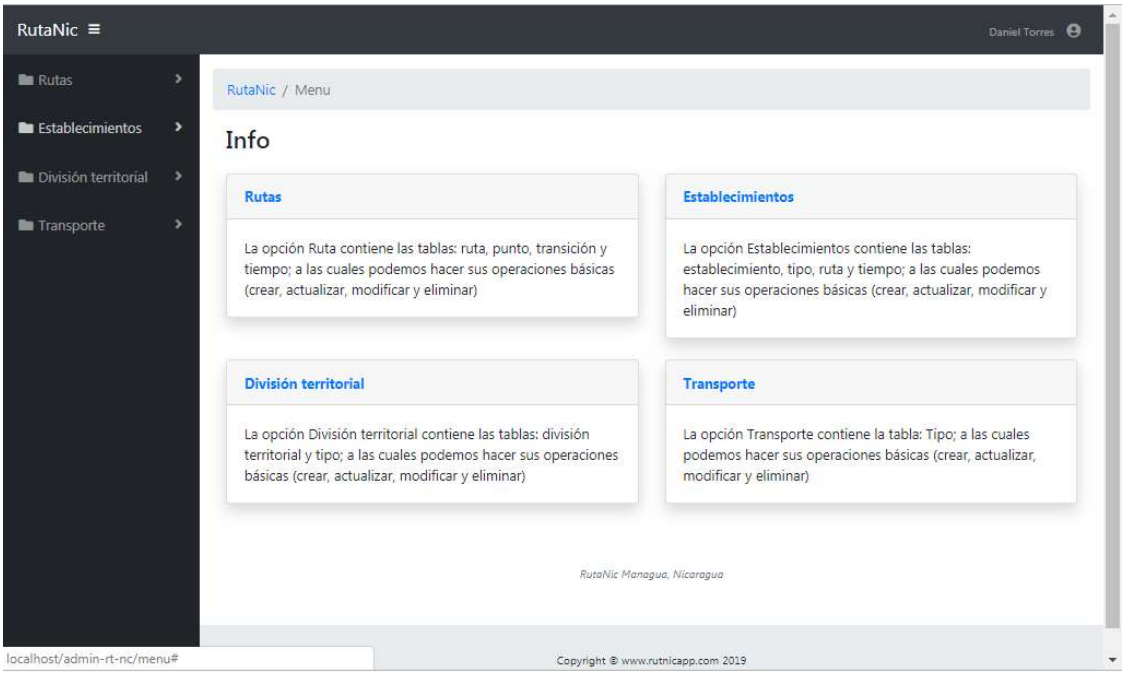
Para devolver los resultados más acertados, la aplicación tiene que estar siempre conectada al internet, ya que es algo que no se puede hacer solamente con el teléfono y ya que toda su información la obtiene de un servicio web.

III. Manual de usuario del sitio administrativo

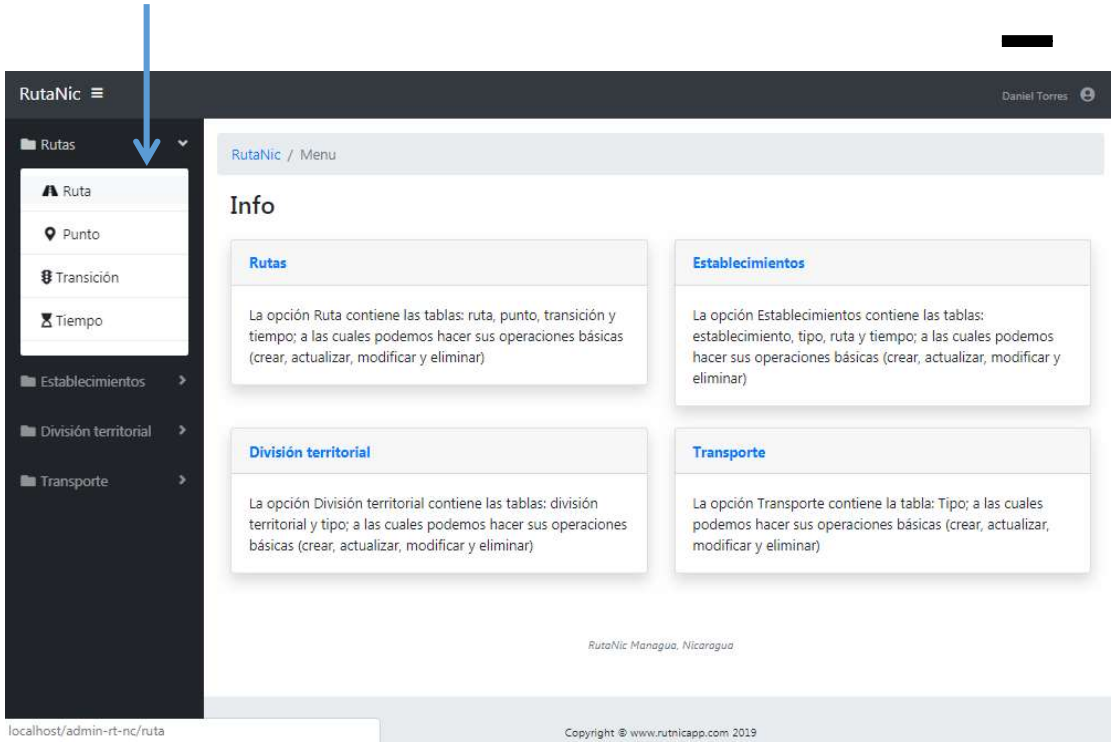
Para poder gestionar todos los módulos de la Aplicación en la base de datos fue necesario la creación de un sitio web administrativo que ayudara a alimentar los datos para crear, actualizar o eliminar una ruta y luego ser consumido a través de un web Services por la aplicación, se creo un acceso Login por motivos de seguridad y evitar que intruso manipulen los datos, también cuenta con un diseño sencillo que sea más amigable al usuario administrador.



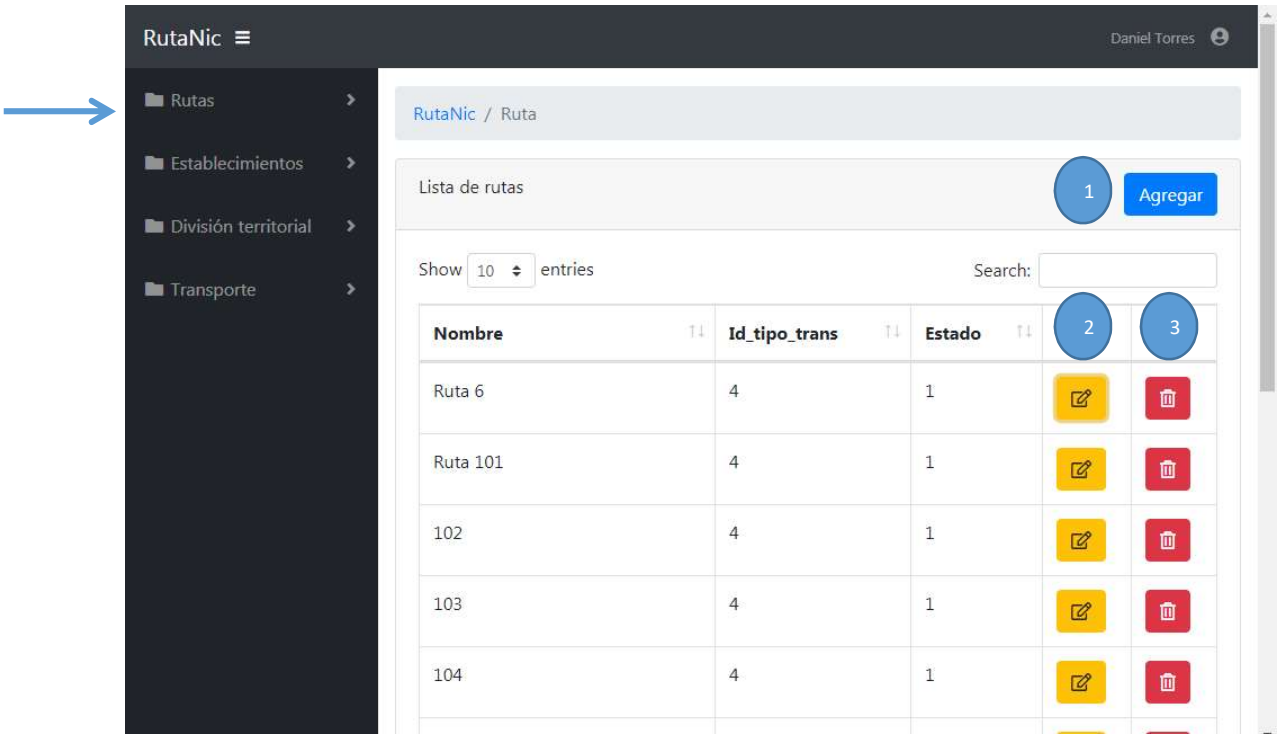
2.Luego accede a la pantalla principal:



3. En esta pantalla se muestran cuatro modulos cada uno de ellos contiene las operaciones crud básicas para cada tabla contenida en la base de datos:



Por ejemplo el usuario puede seleccionar la opción Ruta en la carpeta Rutas y se le mostrara la siguiente información:

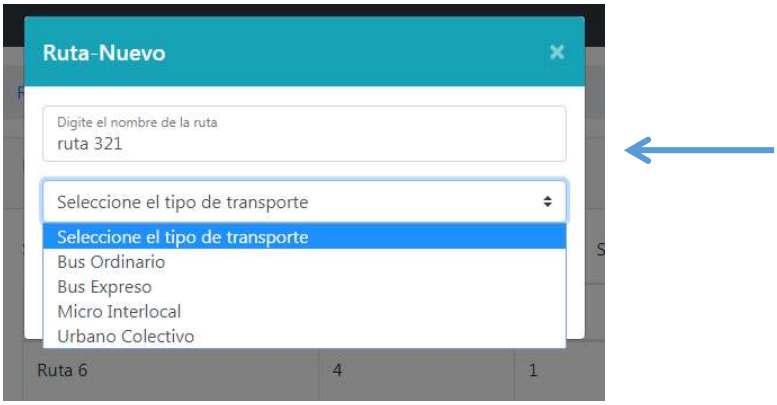


1-Boton para agregar registros

2-Boton para acceder a editar un registro

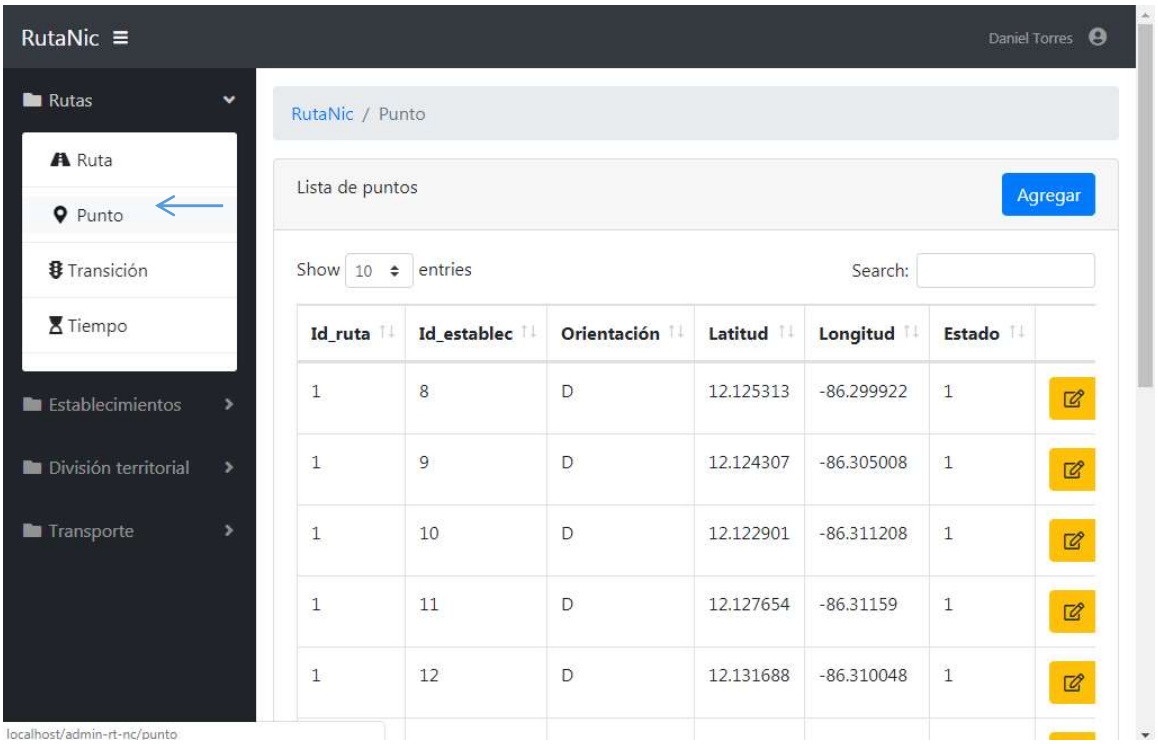
3-Boton para eliminar registros

1-**Agregar ruta**



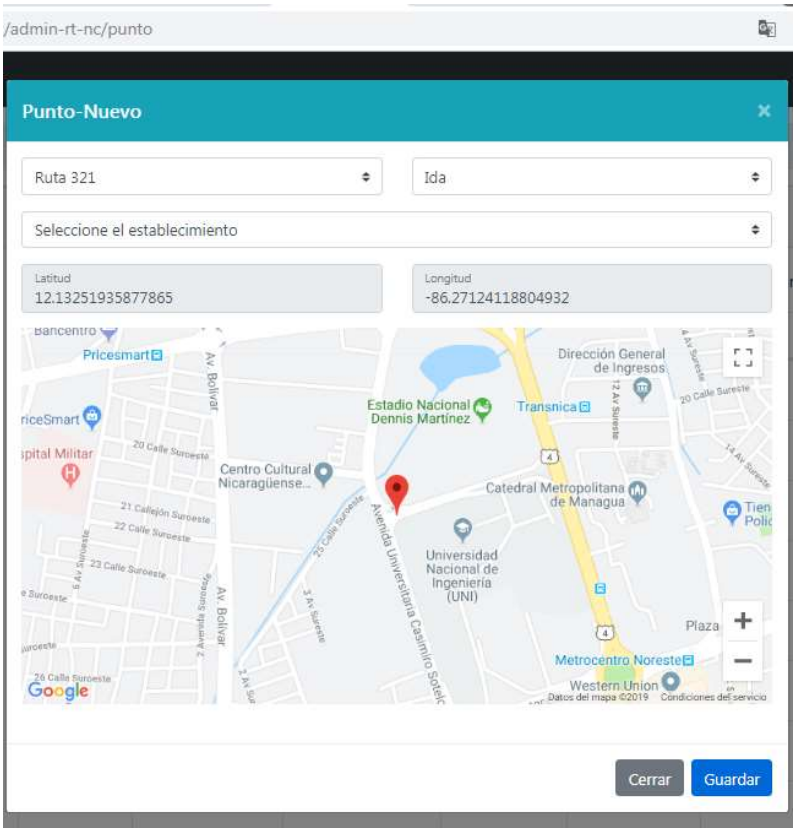
Aquí se agrega la nueva ruta se digita el nombre y selecciona el tipo de transporte a la que va a pertenecer. Una vez agregada la información el usuario procede a guardar el registro

Luego se procede a guardar los puntos por donde pasa la ruta eso lo hacemos en el siguiente modulo.



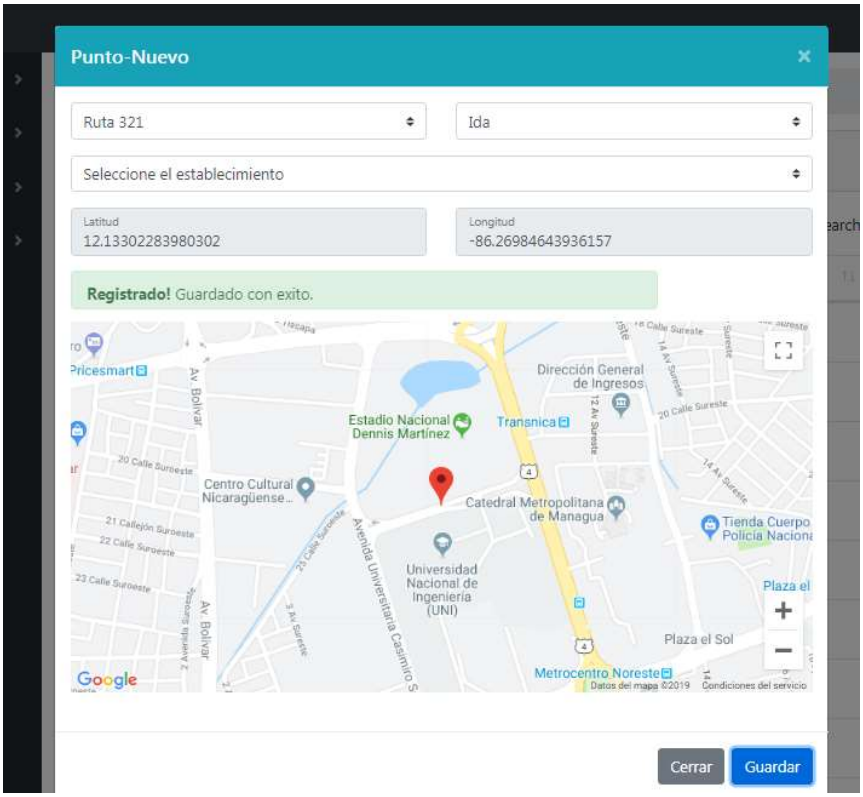
2-En Rutas-Punto agregamos los puntos por donde pasa la ruta mientras mas puntos agregamos mejor queda la línea de la ruta

Este modulo también cuenta con las operaciones básicas (crud)

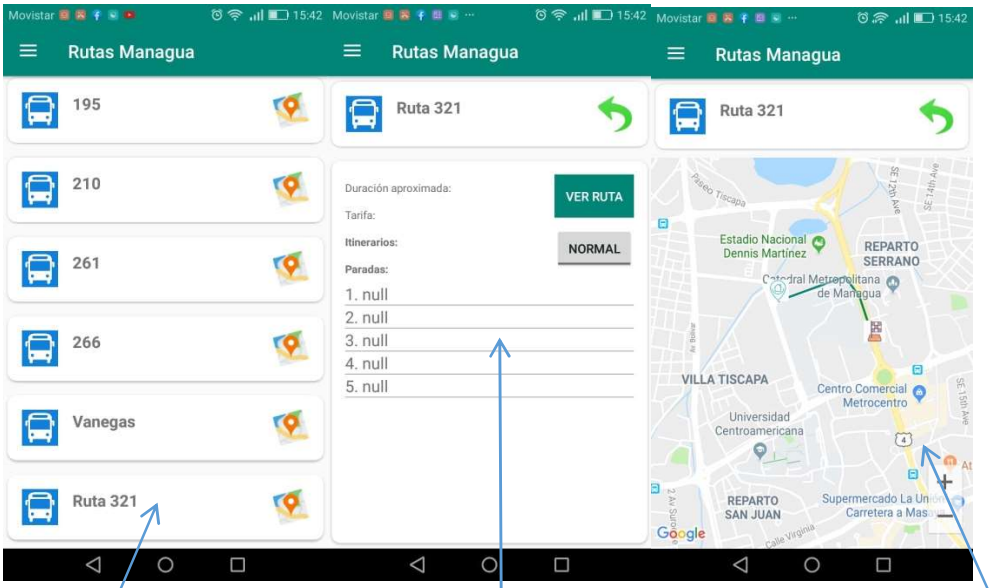


- ✓ El campo ruta 321 es la ruta agregada anteriormente, seleccionamos si es de ida o regreso, el tipo de establecimiento es opcional.
- ✓ La latitud y longitud se cargan al mover el marcador en el mapa.
- ✓ Agregamos todos los puntos para la ruta en su trayectoria de ida o de regreso.
- ✓ Una vez hecho esto ya tenemos una ruta de tipo urbano para la ciudad de Managua.

Mensaje de Guardado con éxito el Registro:



Visualizacion desde la app la ruta nueva:



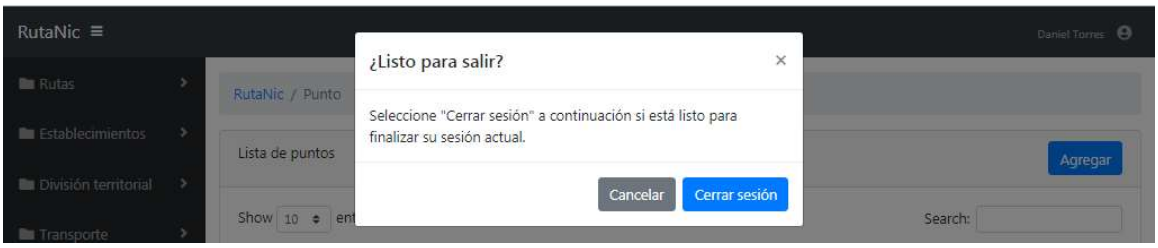
En rutas Managua ya aparece nuestra nueva ruta

Detalle de paradas, no aparece nombre por que no agregamos el establecimiento ya que es opcional.

Podemos visualizar en el mapa los puntos guardados.

De igual manera se realiza el procedimiento para los buses interurbanos.

Al finalizar podemos cerrar sesión. Solo damos clic en el icono a la derecha de la pantalla justo a la par del nombre de usuario logueado.



IV. Manual del programador

IV.1. Diseño de la interfaz.

Para el diseño de la interfaz se utilizó Android Studio, generando código java y xml. La interfaz de la Aplicación móvil está estructurada de la siguiente manera:

Icono principal de la Aplicación

Menú Principal (Módulos):

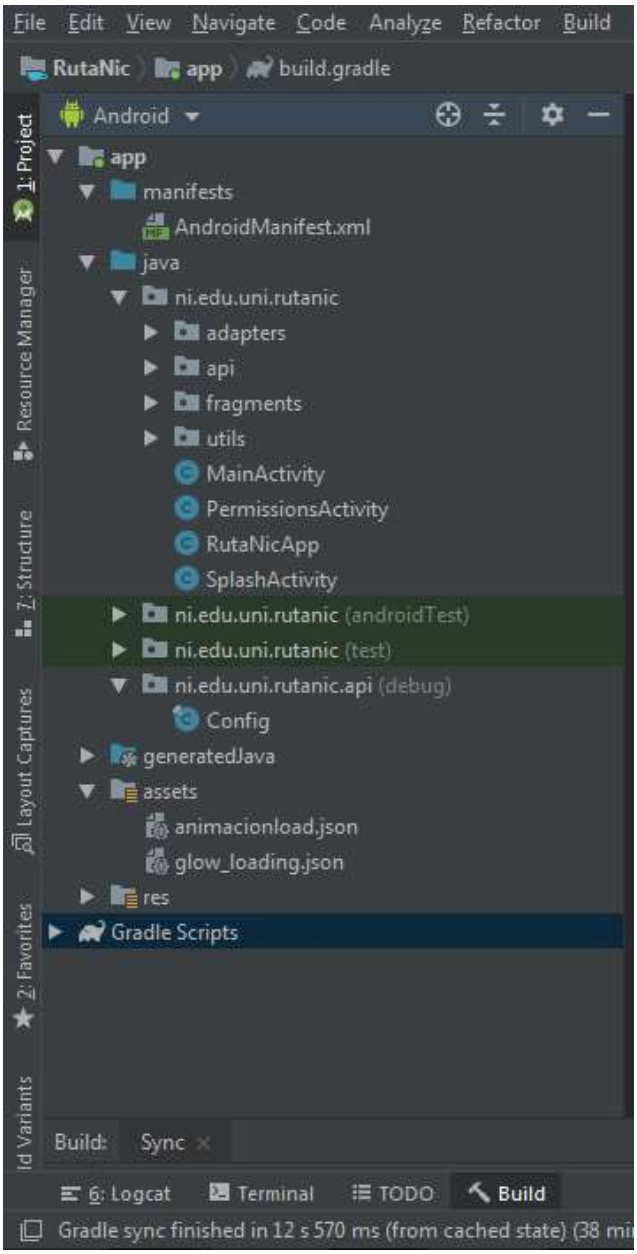
Cómo Llegar: Búsqueda por petición de usuario, redirecciónа en el mapa a solicitud.

Rutas Managua: Lista de Rutas de las trayectorias de transporte en el casco de Managua y lo muestra en el mapa.

Terminales: Información y ubicación en el mapa de las 5 terminales de las rutas interurbanas para la visualización de las paradas en el mapa, itinerarios y precios de Trayectoria de transporte hacia los departamentos.

Acerca de: información de contacto de

IV.2 Estructura de la aplicación en el entorno de desarrollo android



AndroidManifest.xml:

Contiene toda la información necesaria para saber como trabajara nuestra app tal como el formulario de inicio y permisos entre otras configuraciones.

Paquete-ni.edu.ni.rutanic:

Contine otros paquetes y clases las cuales se verán más a detalle mas adelante.

Paquete-ni.edu.uni.rutanic.api:

Este paquete contiene una única clase de configuración en la cual indicamos la dirección del servidor que nos brinda la información necesaria para utilizar la app.

El recurso Assets:

Este recurso contiene un archivo json utilizado para la animación que se muestra en el splash screen de la app.

El recurso res:

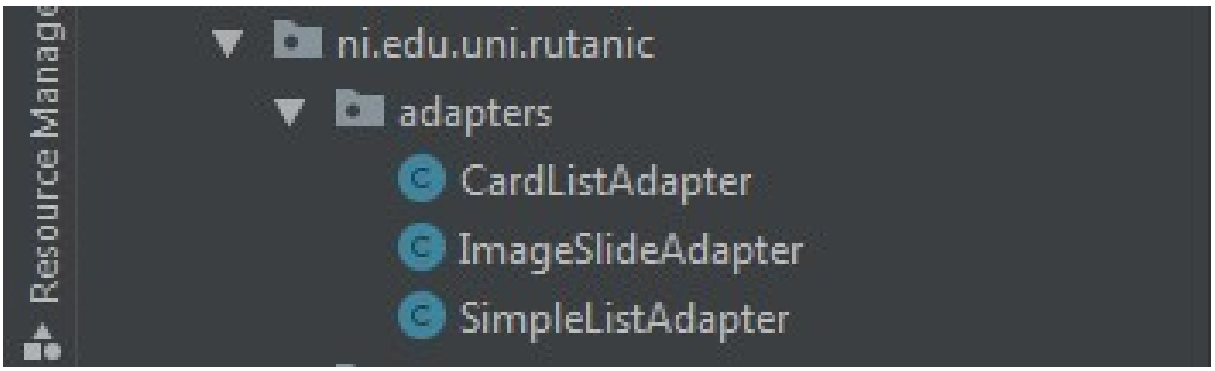
Contiene todas la interfaces o pantallas usadas por la app en archivos de tipo xml asi como strings u otros valores que se utilizan en la capa de diseño.

Gradle Scripts:

Este contiene un archivo en el cual se agregan las dependencias del proyecto asi como librerías entre otros.

IV.3 El Paquete Adapters:

Este paquete contiene las clases usadas para llenado de lista como ReciclerView, CardView e ImageSlider cada uno de ellos se usan para la lista de rutas, terminales, slider de la pantalla principal.



IV.4 Clase CardListAdapter

```
package ni.edu.uni.rutanic.adapters;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.widget.AppCompatImageView;
import androidx.recyclerview.widget.RecyclerView;

import java.util.List;

import ni.edu.uni.rutanic.R;
import ni.edu.uni.rutanic.api.models.ApiModel;
import ni.edu.uni.rutanic.api.models.RutaEstablecimiento;
import ni.edu.uni.rutanic.fragments.FragmentInteractionListener;

public class CardListAdapter extends
```

```
RecyclerView.Adapter<CardListAdapter.CardListHolder>{
    private List<ApiModel> mDataSet;
    private Context context;
    private FragmentInteractionListener listener;
    private int rscIcon = 0;

    public CardListAdapter(Context context, FragmentInteractionListener listener,
List<ApiModel> myDataSet) {
        this.mDataSet = myDataSet;
        this.context = context;
        this.listener = listener;
    }

    public CardListAdapter(Context context, FragmentInteractionListener listener,
List<ApiModel> myDataSet, int rscIcon) {
        this.mDataSet = myDataSet;
        this.context = context;
        this.listener = listener;
        this.rscIcon = rscIcon;
    }

    @NonNull
    @Override
    public CardListAdapter.CardListHolder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {
        View v =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_card_list, parent,
false);
        return new CardListAdapter.CardListHolder(v);
    }

    @Override
    public void onBindViewHolder(@NonNull CardListAdapter.CardListHolder
holder, int position) {
        final ApiModel model = mDataSet.get(position);
        holder.textView1.setText(model.getDescription());
        if (model.getDescriptionDetail() == null) {
            holder.textView2.setVisibility(View.GONE);
        } else {
            holder.textView2.setVisibility(View.VISIBLE);
            holder.textView2.setText(model.getDescriptionDetail());
        }
        if (holder.tvDistance != null) {
            if (model instanceof RutaEstablecimiento) {
                holder.tvDistance.setVisibility(View.VISIBLE);
                holder.tvDistance.setText(context.getString(
```

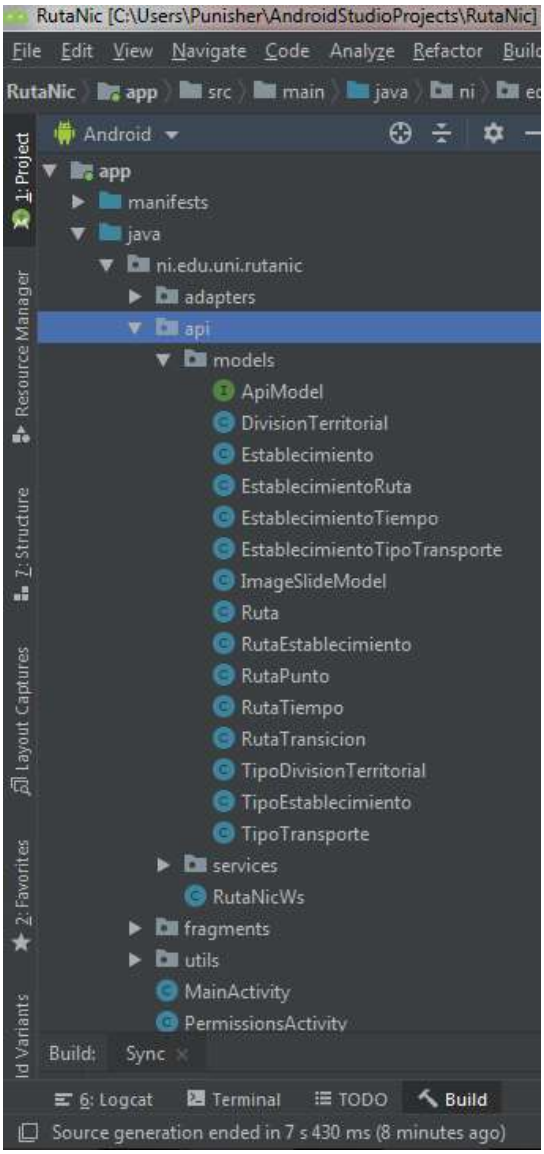
```
                    R.string.app_distancia_metros, ((RutaEstablecimiento)
model).getDistanciaPieTotal()
                ));
                holder.imgWlk.setVisibility(View.VISIBLE);
            } else {
                holder.tvDistance.setVisibility(View.GONE);
                holder.imgWlk.setVisibility(View.GONE);
            }
        }

holder.imageView.setImageDrawable(context.getResources().getDrawable(mode
l.getIconResource()));
        if (rscIcon > 0) {

holder.imageButton.setImageDrawable(context.getResources().getDrawable(rscI
con));
        }
        holder.imageButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (rscIcon > 0) {
                    listener.onFragmentInteraction(2, model);
                } else {
                    listener.onFragmentInteraction(3, model);
                }
            }
        });
    }

    @Override
    public int getItemCount() {
        return mDataSet.size();
    }

    static class CardListHolder extends RecyclerView.ViewHolder {
        TextView textView1;
        TextView textView2;
        TextView tvDistance;
        View imgWlk;
        ImageView imageView;
        AppCompatImageView imageButton;

        CardListHolder(View v) {
            super(v);
            textView1 = v.findViewById(R.id.textView1);
            textView2 = v.findViewById(R.id.textView2);
```

```
            tvDistance = v.findViewById(R.id.tvDistance);
            imageView = v.findViewById(R.id.imageView1);
            imageButton = v.findViewById(R.id.imageButton1);
            imgWlk = v.findViewById(R.id.imgWlkCn);
        }
    }
}
```

IV.5 Clase ImageSliderAdapter

```
package ni.edu.uni.rutanic.adapters;

import android.content.Context;
import android.os.Parcelable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

import androidx.annotation.NonNull;
import androidx.viewpager.widget.PagerAdapter;

import java.util.ArrayList;

import ni.edu.uni.rutanic.R;
import ni.edu.uni.rutanic.api.models.ImageSlideModel;

public class ImageSlideAdapter extends PagerAdapter {
    private ArrayList<ImageSlideModel> imageModelArrayList;
    private LayoutInflater inflater;

    public ImageSlideAdapter(Context context, ArrayList<ImageSlideModel>
imageModelArrayList) {
        this.imageModelArrayList = imageModelArrayList;
        inflater = LayoutInflater.from(context);
    }

    @Override
    public void destroyItem(@NonNull ViewGroup container, int position,
@NonNull Object object) {
        container.removeView((View) object);
    }

    @Override
```

```
    public int getCount() {
        return imageModelArrayList.size();
    }

    @NonNull
    @Override
    public Object instantiateItem(@NonNull ViewGroup view, int position) {
        View imageLayout = inflater.inflate(R.layout.item_image_slide, view, false);
        if (imageLayout != null) {
            ImageView imageView = imageLayout.findViewById(R.id.image);

imageView.setImageResource(imageModelArrayList.get(position).getImage_dra
wable());
            view.addView(imageLayout, 0);
        } else {
            imageLayout = new View(inflater.getContext());
        }
        return imageLayout;
    }

    @Override
    public boolean isViewFromObject(@NonNull View view, @NonNull Object
object) {
        return view.equals(object);
    }

    @Override
    public void restoreState(Parcelable state, ClassLoader loader) {
    }

    @Override
    public Parcelable saveState() {
        return null;
    }
}
```



Contiene la interface ApiModel y sus clases a como se ven en la imagen de la izquierda.

Todas implementan de la interface ApiModel y cada una hace referencia a un método usado en la ApiRest (Mas adelante se explica este servicio).

El objetivo de este paquete es tener todos los objetos que nuestra app usa para poder ser manipulados de manera más sencilla dentro de la misma.

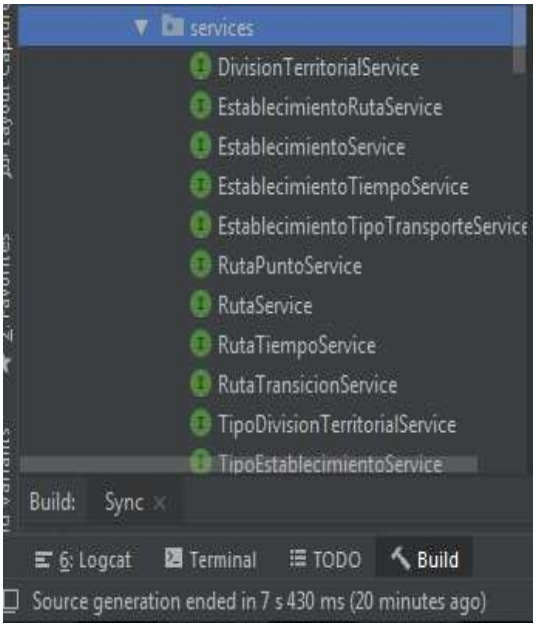
IV.6 Interface ApiModel:

```
package ni.edu.uni.rutanic.api.models;

public interface ApiModel {
    int getId();

    String getDescription();

    String getDescriptionDetail();

    int getIconResource();
```

```
}
```

IV.7 Clase Ruta:

```
package ni.edu.uni.rutanic.api.models;

import java.util.Date;

import lombok.Data;
import ni.edu.uni.rutanic.R;

@Data
public class Ruta implements ApiModel {
    private int id;
    private String nombre;
    private int tipo_transporte_id;
    private int estado;
    private Date modificado;

    @Override
    public String getDescription() {
        return nombre;
    }

    @Override
    public String getDescriptionDetail() {
        return null;
    }

    @Override
    public int getIconResource() {
        return R.drawable.ruta_urbano_icon;
    }
}
```



El paquete servicio tiene la finalidad de mandar a consultar toda la información a cada servicio que brinda la ApiRest de rutanic se utilizo la librería de retrofit para consumir servicios.

Todas son de tipo interface ya que posteriormente las implementaremos en otras clases.

IV.8 Interface RutaService:

```
package ni.edu.uni.rutanic.api.services;

import java.util.List;

import ni.edu.uni.rutanic.api.models.Ruta;
import ni.edu.uni.rutanic.api.models.RutaEstablecimiento;
import retrofit2.Call;
import retrofit2.http.GET;
import retrofit2.http.Path;
import retrofit2.http.Query;

public interface RutaService {

    //cargar datos de todas las rutas de managua
    @GET("rutas")
    public Call<List<Ruta>> obtenerTodos();

    //cargar rutas por departamento
    @GET("rutas/0/departamento/{dptoId}")
    public Call<List<Ruta>> obtenerPorDepartamento(@Path("dptoId") int
departamentoId);

    //aqui se extrae la informacion para poder tener las rutas por terminal
    @GET("rutas/0/terminal/{terminalId}")
```

```
    public Call<List<Ruta>> obtenerPorTerminal(@Path("terminalId") int
terminalId);

    //Con esta funcion se hace el como llegar
    @GET("rutas/0/calcular")
    public Call<List<RutaEstablecimiento>> obtenerPorUbicacionOrigenDestino(
            @Query("lttOrg") String lttOrg,
            @Query("lngOrg") String lngOrg,
            @Query("lttDst") String lttDst,
            @Query("lngDst") String lngDst
    );
}
```

IV.9 Interface RutaService:

```
package ni.edu.uni.rutanic.api.services;

import java.util.List;

import ni.edu.uni.rutanic.api.models.RutaPunto;
import retrofit2.Call;
import retrofit2.http.GET;
import retrofit2.http.Path;
import retrofit2.http.Query;

public interface RutaPuntoService {
    @GET("ruta-puntos/{id}")
    public Call<RutaPunto> obtenerPorId(@Path("id") int id);

    @GET("ruta-puntos/0")
    public Call<List<RutaPunto>> obtenerPorRutaId(@Query("ruta") int rutaId);
}
```

IV.10 Clase RutaNicWS:

```
package ni.edu.uni.rutanic.api;

import android.content.Context;
import android.util.Base64;
import android.widget.Toast;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
```

```
import java.util.Locale;
import java.util.concurrent.TimeUnit;

import ni.edu.uni.rutanic.RutaNicApp;
import ni.edu.uni.rutanic.api.models.DivisionTerritorial;
import ni.edu.uni.rutanic.api.models.Establecimiento;
import ni.edu.uni.rutanic.api.models.EstablecimientoRuta;
import ni.edu.uni.rutanic.api.models.Ruta;
import ni.edu.uni.rutanic.api.models.RutaEstablecimiento;
import ni.edu.uni.rutanic.api.models.RutaPunto;
import ni.edu.uni.rutanic.api.models.RutaTiempo;
import ni.edu.uni.rutanic.api.models.RutaTransicion;
import ni.edu.uni.rutanic.api.models.TipoDivisionTerritorial;
import ni.edu.uni.rutanic.api.models.TipoEstablecimiento;
import ni.edu.uni.rutanic.api.models.TipoTransporte;
import ni.edu.uni.rutanic.api.services.DivisionTerritorialService;
import ni.edu.uni.rutanic.api.services.EstablecimientoRutaService;
import ni.edu.uni.rutanic.api.services.EstablecimientoService;
import ni.edu.uni.rutanic.api.services.EstablecimientoTiempoService;
import ni.edu.uni.rutanic.api.services.EstablecimientoTipoTransporteService;
import ni.edu.uni.rutanic.api.services.RutaPuntoService;
import ni.edu.uni.rutanic.api.services.RutaService;
import ni.edu.uni.rutanic.api.services.RutaTiempoService;
import ni.edu.uni.rutanic.api.services.RutaTransicionService;
import ni.edu.uni.rutanic.api.services.TipoDivisionTerritorialService;
import ni.edu.uni.rutanic.api.services.TipoEstablecimientoService;
import ni.edu.uni.rutanic.api.services.TipoTransporteService;
import ni.edu.uni.rutanic.fragments.FragmentInteractionListener;
import okhttp3.Interceptor;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class RutaNicWs {
    private Context context;

    public List<TipoTransporte> tipoTransportes = new ArrayList<>();
    public List<TipoDivisionTerritorial> tipoDivisionTerritorials = new ArrayList<>();
    public List<TipoEstablecimiento> tipoEstablecimientos = new ArrayList<>();
    public List divisionTerritorials = new ArrayList<>();
    public List terminales = new ArrayList<>();
```

```
    public List rutas = new ArrayList<>();
    public List rutasDpto = new ArrayList<>();
    public List<RutaEstablecimiento> rutasCalculadas = new ArrayList<>();
    public List<RutaPunto> rutaPuntos = new ArrayList<>();
    public List<RutaTiempo> rutaTiempos = new ArrayList<>();
    public List<RutaTransicion> rutaTransicions = new ArrayList<>();

    private TipoTransporteService srvTiposTransportes;
    private TipoEstablecimientoService srvTiposEstablecimientos;
    private TipoDivisionTerritorialService srvTiposDivisionesTerritoriales;
    private DivisionTerritorialService srvDivisionesTerritoriales;
    private EstablecimientoService srvEstablecimientos;
    private RutaService srvRutas;
    private EstablecimientoRutaService srvEstablecRutas;
    private EstablecimientoTiempoService srvEstablecTiempos;
    private EstablecimientoTipoTransporteService srvEstablecTiposTransportes;
    private RutaPuntoService srvRutaPuntos;
    private RutaTiempoService srvRutaTiempos;
    private RutaTransicionService srvRutaTransiciones;

    private RutaNicApp.OnWsInteractionListener listener;
    private int currentProgress = 0;

    public RutaNicWs(Context context) {
        this.context = context;

        final OkHttpClient okHttpClient = new OkHttpClient.Builder()
                .readTimeout(10, TimeUnit.SECONDS)
                .connectTimeout(10, TimeUnit.SECONDS)
                .addInterceptor(new Interceptor() {
                    @Override
                    public okhttp3.Response intercept(Chain chain) throws IOException
{
                        Request original = chain.request();
                        String credentials = Config.API_HTTP_USR + ":" +
Config.API_HTTP_PSW;
                        String basic = "Basic " +
Base64.encodeToString(credentials.getBytes(), Base64.NO_WRAP);
                        Request.Builder builder = original.newBuilder()
                                .header("Authorization", basic);
                        Request request = builder.build();
                        okhttp3.Response response = chain.proceed(request);
                        if (response.code() == 400) {
                            return response;
                        }
                        return response;
```

```
                }
            }).build();

        GsonConverterFactory gsonConverterFactory =
GsonConverterFactory.create();
        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl(Config.API_BASE_URL)
                .addConverterFactory(gsonConverterFactory)
                .client(okHttpClient)
                .build();

        srvTiposTransportes = retrofit.create(TipoTransporteService.class);
        srvTiposEstablecimientos =
retrofit.create(TipoEstablecimientoService.class);
        srvTiposDivisionesTerritoriales =
retrofit.create(TipoDivisionTerritorialService.class);
        srvDivisionesTerritoriales = retrofit.create(DivisionTerritorialService.class);
        srvEstablecimientos = retrofit.create(EstablecimientoService.class);
        srvRutas = retrofit.create(RutaService.class);
        srvEstablecRutas = retrofit.create(EstablecimientoRutaService.class);
        srvEstablecTiempos = retrofit.create(EstablecimientoTiempoService.class);
        srvEstablecTiposTransportes =
retrofit.create(EstablecimientoTipoTransporteService.class);
        srvRutaPuntos = retrofit.create(RutaPuntoService.class);
        srvRutaTiempos = retrofit.create(RutaTiempoService.class);
        srvRutaTransiciones = retrofit.create(RutaTransicionService.class);
    }

    public void setInteractionListener(RutaNicApp.OnWsInteractionListener
listener) {
        this.listener = listener;
    }

    public void cargaInicial() {
        currentProgress = 0;
        Call<List<TipoTransporte>> call_1 = srvTiposTransportes.obtenerTodos();
        call_1.enqueue(new Callback<List<TipoTransporte>>() {
            @Override
            public void onResponse(Call<List<TipoTransporte>> call,
Response<List<TipoTransporte>> response) {
                if (response.body() == null) {
                    return;
                }
                tipoTransportes = response.body();
                publishProgress(20);
            }
```

```
            @Override
            public void onFailure(Call<List<TipoTransporte>> call, Throwable t) {
                Toast.makeText(context, t.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });
        Call<List<TipoDivisionTerritorial>> call_2 =
srvTiposDivisionesTerritoriales.obtenerTodos();
        call_2.enqueue(new Callback<List<TipoDivisionTerritorial>>() {
            @Override
            public void onResponse(Call<List<TipoDivisionTerritorial>> call,
Response<List<TipoDivisionTerritorial>> response) {
                if (response.body() == null) {
                    return;
                }
                tipoDivisionTerritorials = response.body();
                publishProgress(20);
            }

            @Override
            public void onFailure(Call<List<TipoDivisionTerritorial>> call, Throwable
t) {
                Toast.makeText(context, t.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });
        Call<List<TipoEstablecimiento>> call_3 =
srvTiposEstablecimientos.obtenerTodos();
        call_3.enqueue(new Callback<List<TipoEstablecimiento>>() {
            @Override
            public void onResponse(Call<List<TipoEstablecimiento>> call,
Response<List<TipoEstablecimiento>> response) {
                if (response.body() == null) {
                    return;
                }
                tipoEstablecimientos = response.body();
                publishProgress(20);
            }

            @Override
            public void onFailure(Call<List<TipoEstablecimiento>> call, Throwable t) {
                Toast.makeText(context, t.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });
```

```
        Call<List<DivisionTerritorial>> call_4 =
srvDivisionesTerritoriales.obtenerTodos();
        call_4.enqueue(new Callback<List<DivisionTerritorial>>() {
            @Override
            public void onResponse(Call<List<DivisionTerritorial>> call,
Response<List<DivisionTerritorial>> response) {
                if (response.body() == null) {
                    return;
                }
                divisionTerritorials = response.body();
                publishProgress(20);
            }

            @Override
            public void onFailure(Call<List<DivisionTerritorial>> call, Throwable t) {
                Toast.makeText(context, t.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });
        Call<List<Ruta>> call_5 = srvRutas.obtenerTodos();
        call_5.enqueue(new Callback<List<Ruta>>() {
            @Override
            public void onResponse(Call<List<Ruta>> call, Response<List<Ruta>>
response) {
                if (response.body() == null) {
                    return;
                }
                rutas = response.body();
                publishProgress(10);
            }

            @Override
            public void onFailure(Call<List<Ruta>> call, Throwable t) {
                Toast.makeText(context, t.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });
        Call<List<Establecimiento>> call_6 =
srvEstablecimientos.obtenerTerminales();
        call_6.enqueue(new Callback<List<Establecimiento>>() {
            @Override
            public void onResponse(Call<List<Establecimiento>> call,
Response<List<Establecimiento>> response) {
                if (response.body() == null) {
                    return;
                }
```

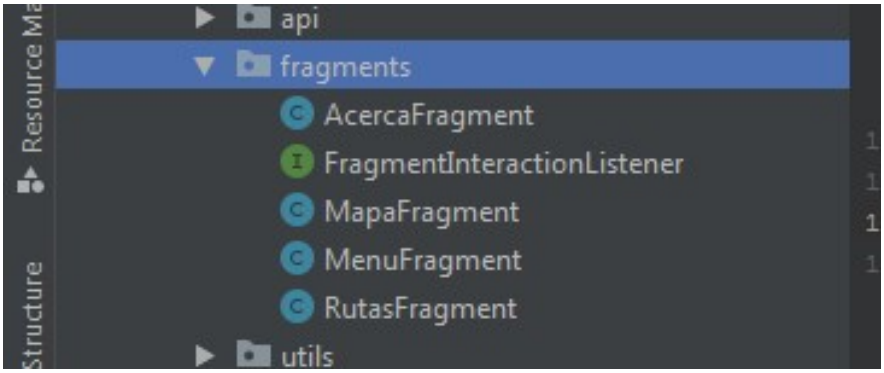
```
                terminales = response.body();
                publishProgress(10);
            }

            @Override
            public void onFailure(Call<List<Establecimiento>> call, Throwable t) {
                Toast.makeText(context, t.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });
    }

    private void publishProgress(int progress) {
        currentProgress += progress;
        if (currentProgress > 100) {
            currentProgress = 100;
        }
        listener.onWsInteraction(0, currentProgress);
    }

    public void loadEstablecimientoRutas() {
        Call<List<EstablecimientoRuta>> call =
srvEstablecRutas.obtenerPorEstablecimientoId(0);
        call.enqueue(new Callback<List<EstablecimientoRuta>>() {
            @Override
            public void onResponse(Call<List<EstablecimientoRuta>> call,
Response<List<EstablecimientoRuta>> response) {
                Toast.makeText(context, String.valueOf(response.code()),
Toast.LENGTH_SHORT).show();
                Toast.makeText(context, response.message(),
Toast.LENGTH_SHORT).show();
                if (response.body() == null) {
                    return;
                }
                for (EstablecimientoRuta item : response.body()) {
                    Toast.makeText(context, item.getCreado().toString(),
Toast.LENGTH_SHORT).show();
                }
            }

            @Override
            public void onFailure(Call<List<EstablecimientoRuta>> call, Throwable t)
{
                Toast.makeText(context, t.getMessage(),
Toast.LENGTH_LONG).show();
            }
```

```
        });
    }

    public void obtenerEstablecimientoRuta() {
        Call<EstablecimientoRuta> call = srvEstablecRutas.obtenerPorId(0);
        call.enqueue(new Callback<EstablecimientoRuta>() {
            @Override
            public void onResponse(Call<EstablecimientoRuta> call,
Response<EstablecimientoRuta> response) {
                Toast.makeText(context, String.valueOf(response.code()),
Toast.LENGTH_SHORT).show();
                Toast.makeText(context, response.message(),
Toast.LENGTH_SHORT).show();
                if (response.body() == null) {
                    return;
                }
                Toast.makeText(context, String.valueOf(response.body()),
Toast.LENGTH_LONG).show();
            }

            @Override
            public void onFailure(Call<EstablecimientoRuta> call, Throwable t) {
                Toast.makeText(context, t.getMessage(),
Toast.LENGTH_LONG).show();
            }
        });
    }

    private FragmentInteractionListener lstListener;

    public void obtenerRutasPorDepartamento(int dptoId,
FragmentInteractionListener listener) {
        lstListener = listener;
        Call<List<Ruta>> call = srvRutas.obtenerPorDepartamento(dptoId);
        call.enqueue(new Callback<List<Ruta>>() {
            @Override
            public void onResponse(Call<List<Ruta>> call, Response<List<Ruta>>
response) {
                if (response.body() != null) {
                    rutasDpto = response.body();
                    lstListener.onFragmentInteraction(6, 1);
                }
            }

            @Override
            public void onFailure(Call<List<Ruta>> call, Throwable t) {
```

```
                Toast.makeText(context, "obtenerRutasPorDepartamento: " +
t.getMessage(), Toast.LENGTH_LONG).show();
            }
        });
    }

    public void obtenerRutasPorTerminal(int terminalId,
FragmentInteractionListener listener) {
        lstListener = listener;
        Call<List<Ruta>> call = srvRutas.obtenerPorTerminal(terminalId);
        call.enqueue(new Callback<List<Ruta>>() {
            @Override
            public void onResponse(Call<List<Ruta>> call, Response<List<Ruta>>
response) {
                if (response.body() != null) {
                    rutasDpto = response.body();
                    lstListener.onFragmentInteraction(6, 1);
                }
            }

            @Override
            public void onFailure(Call<List<Ruta>> call, Throwable t) {
                Toast.makeText(context, "obtenerRutasPorTerminal: " +
t.getMessage(), Toast.LENGTH_LONG).show();
            }
        });
    }

    public void calcularRuta(Double lttOrg, Double lngOrg, Double lttDst, Double
lngDst, FragmentInteractionListener listener) {
        lstListener = listener;
        Call<List<RutaEstablecimiento>> call =
srvRutas.obtenerPorUbicacionOrigenDestino(
                String.format(Locale.US, "%.6f", lttOrg), String.format(Locale.US,
"%.6f", lngOrg),
                String.format(Locale.US, "%.6f", lttDst), String.format(Locale.US,
"%.6f", lngDst)
        );
        call.enqueue(new Callback<List<RutaEstablecimiento>>() {
            @Override
            public void onResponse(Call<List<RutaEstablecimiento>> call,
Response<List<RutaEstablecimiento>> response) {
                if (response.body() != null) {
                    rutasCalculadas = response.body();
                    lstListener.onFragmentInteraction(11, 1);
                }
```

```
            }

            @Override
            public void onFailure(Call<List<RutaEstablecimiento>> call, Throwable t)
{
                Toast.makeText(context, "obtenerRutasPorDepartamento: " +
t.getMessage(), Toast.LENGTH_LONG).show();
            }
        });
    }

    public void obtenerDatosRuta(final int id, FragmentInteractionListener listener)
{
        lstListener = listener;
        Call<List<RutaPunto>> call = srvRutaPuntos.obtenerPorRutaId(id);
        call.enqueue(new Callback<List<RutaPunto>>() {
            @Override
            public void onResponse(Call<List<RutaPunto>> call,
Response<List<RutaPunto>> response) {
                if (response.body() != null) {
                    rutaPuntos = response.body();
                    obtenerTiemposRuta(id);
                }
            }

            @Override
            public void onFailure(Call<List<RutaPunto>> call, Throwable t) {
                Toast.makeText(context, "obtenerDatosRuta: " + t.getMessage(),
Toast.LENGTH_LONG).show();
            }
        });
    }

    public void obtenerTiemposRuta(final int id) {
        Call<List<RutaTiempo>> call = srvRutaTiempos.obtenerPorRutaId(id);
        call.enqueue(new Callback<List<RutaTiempo>>() {
            @Override
            public void onResponse(Call<List<RutaTiempo>> call,
Response<List<RutaTiempo>> response) {
                if (response.body() != null) {
                    rutaTiempos = response.body();
                    obtenerTransicionesRuta(id);
                }
            }

            @Override
```

```
            public void onFailure(Call<List<RutaTiempo>> call, Throwable t) {
                Toast.makeText(context, "obtenerTiemposRuta: " + t.getMessage(),
Toast.LENGTH_LONG).show();
            }
        });
    }

    public void obtenerTransicionesRuta(int id) {
        Call<List<RutaTransicion>> call =
srvRutaTransiciones.obtenerPorRutaId(id);
        call.enqueue(new Callback<List<RutaTransicion>>() {
            @Override
            public void onResponse(Call<List<RutaTransicion>> call,
Response<List<RutaTransicion>> response) {
                if (response.body() != null) {
                    rutaTransicions = response.body();
                    lstListener.onFragmentInteraction(4, 1);
                }
            }

            @Override
            public void onFailure(Call<List<RutaTransicion>> call, Throwable t) {
                Toast.makeText(context, "obtenerTransicionesRuta: " +
t.getMessage(), Toast.LENGTH_LONG).show();
            }
        });
    }
}
```



El paquete fragment contiene a los fragmentos que se usan en la app dentro de los distintos activities.

IV. 11 AcercaFragment:

```
package ni.edu.uni.rutanic.fragments;

import android.content.Context;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.appcompat.widget.AppCompatTextView;
import androidx.fragment.app.Fragment;

import com.google.android.material.floatingactionbutton.FloatingActionButton;

import ni.edu.uni.rutanic.MainActivity;
import ni.edu.uni.rutanic.R;
import ni.edu.uni.rutanic.utils.AndroidUtils;

public class AcercaFragment extends Fragment implements
        FragmentInteractionListener {
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    private FragmentInteractionListener mListener;
    private String mParam1;
    private String mParam2;

    public AcercaFragment() {
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup
```

```
container, Bundle savedInstanceState) {
        ((MainActivity) inflater.getContext()).setTitle(R.string.menu_acerca);
        ((MainActivity)
inflater.getContext()).selectNavigationDrawerItem(R.id.nav_acerca);
        ((MainActivity) inflater.getContext()).onFragmentInteraction(99, null);
        View view = inflater.inflate(R.layout.fragment_acerca, container, false);
        AppCompatTextView tvDetalle = view.findViewById(R.id.textView1);
        tvDetalle.setText(inflater.getContext().getString(R.string.app_acerca,
                AndroidUtils.getAppName(inflater.getContext()),
                AndroidUtils.getAppVersion(inflater.getContext()),
                "contacto@correo.com"));
        return view;
    }

    @Override
    public void onAttach(@NonNull Context context) {
        super.onAttach(context);
        if (context instanceof FragmentInteractionListener) {
            mListener = (FragmentInteractionListener) context;
        } else {
            throw new RuntimeException(context.toString()
                    + " must implement FragmentInteractionListener");
        }
        mListener.setChildFragmentInteraction(this);
        mListener.getFloatingActionButton().setVisibility(View.GONE);
        mListener.onFragmentInteraction(99, null);
    }

    @Override
    public void onDetach() {
        super.onDetach();
    }

    @Override
    public void onFragmentInteraction(int type, Object value) {
    }

    @Override
    public void setChildFragmentInteraction(FragmentInteractionListener listener) {
    }

    @Override
    public FloatingActionButton getFloatingActionButton() {
        return null;
    }
}
```

IV.12. MapaFragment:

```
package ni.edu.uni.rutanic.fragments;

import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.content.res.ColorStateList;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.EditorInfo;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.widget.AppCompatEditText;
import androidx.core.app.ActivityCompat;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationCallback;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationResult;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
```

```
import com.google.android.material.snackbar.Snackbar;

import java.io.IOException;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

import ni.edu.uni.rutanic.MainActivity;
import ni.edu.uni.rutanic.R;
import ni.edu.uni.rutanic.RutaNicApp;
import ni.edu.uni.rutanic.adapters.CardListAdapter;
import ni.edu.uni.rutanic.api.models.ApiModel;
import ni.edu.uni.rutanic.api.models.RutaEstablecimiento;
import ni.edu.uni.rutanic.utils.AndroidUtils;
import ni.edu.uni.rutanic.utils.GeoLocationUtils;
import ni.edu.uni.rutanic.utils.GoogleMapUtils;

public class MapaFragment extends Fragment implements
        FragmentInteractionListener,
        OnMapReadyCallback,
        GoogleApiClient.OnConnectionFailedListener,
        GoogleApiClient.ConnectionCallbacks {
    private RutaNicApp mApp;
    private FragmentInteractionListener mListener;
    private boolean isCentered = false;
    private boolean isDefaultNic = false;
    private GoogleMap mMap;

    private ProgressBar pbSearch;
    private View vwDestino;
    private AppCompatEditText etOrigen;
    private AppCompatEditText etDestino;

    private View vwGoogleMap;
    private View cntEntidad;
    private ProgressBar progressBar;
    private RecyclerView rvRutas;
    private RecyclerView.Adapter adapterRutas;
    private RecyclerView.LayoutManager layoutManager;

    public MapaFragment() {
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
        ((MainActivity) inflater.getContext()).setTitle(R.string.menu_como_llegar);
        ((MainActivity)
inflater.getContext()).selectNavigationDrawerItem(R.id.nav_como_llegar);
        ((MainActivity) inflater.getContext()).onFragmentInteraction(98, null);

        mApp = (RutaNicApp) inflater.getContext().getApplicationContext();

        View view = inflater.inflate(R.layout.fragment_mapa, container, false);
        SupportMapFragment fragment = (SupportMapFragment)
getChildFragmentManager().findFragmentById(R.id.frg);
        if (fragment != null) {
            fragment.getMapAsync(this);
        }

        TextView.OnEditorActionListener exampleListener = new
TextView.OnEditorActionListener() {
            @Override
            public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
                if (!TextUtils.isEmpty(etDestino.getText()) && actionId ==
EditorInfo.IME_NULL
                        && event.getAction() == KeyEvent.ACTION_DOWN) {
                    //example_confirm();//match this behavior to your 'Send' (or Confirm)
button
                    calcularRutas();
                }
                return true;
            }
        };
        vwDestino = view.findViewById(R.id.vwDestino);
        etOrigen = view.findViewById(R.id.etOrigen);
        etDestino = view.findViewById(R.id.etDestino);
        etOrigen.setOnEditorActionListener(exampleListener);
        etDestino.setOnEditorActionListener(exampleListener);
        pbSearch = vwDestino.findViewById(R.id.progressBar1);

        pbSearch.setVisibility(View.GONE);
        vwDestino.setVisibility(View.GONE);

        progressBar = view.findViewById(R.id.progressBarGbl);
        rvRutas = view.findViewById(R.id.my_recycler_view);
        vwGoogleMap = view.findViewById(R.id.vwGoogleMap);
```

```
        cntEntidad = view.findViewById(R.id.cntEntidad);

        progressBar.setVisibility(View.GONE);
        rvRutas.setVisibility(View.GONE);
        vwGoogleMap.setVisibility(View.VISIBLE);
        progressBar.setVisibility(View.GONE);
        cntEntidad.setVisibility(View.GONE);

        // use this setting to improve performance if you know that changes
        // in content do not change the layout size of the RecyclerView
        rvRutas.setHasFixedSize(true);

        // use a linear layout manager
        layoutManager = new LinearLayoutManager(inflater.getContext());
        rvRutas.setLayoutManager(layoutManager);

        return view;
    }

    public LatLng getLocationFromAddress(Context context, String strAddress) {
        Geocoder coder = new Geocoder(context);
        List<Address> address;
        LatLng p1 = null;
        try {
            // May throw an IOException
            address = coder.getFromLocationName(strAddress + ", Nicaragua", 3);
            if (address == null || address.size() < 1) {
                return null;
            }
            Address location = address.get(0);
            p1 = new LatLng(location.getLatitude(), location.getLongitude());
        } catch (IOException ex) {
            // ex.printStackTrace();
        }
        return p1;
    }

    @Override
    public void onAttach(@NonNull Context context) {
        super.onAttach(context);
        if (context instanceof FragmentInteractionListener) {
            mListener = (FragmentInteractionListener) context;
        } else {
            throw new RuntimeException(context.toString()
                    + " must implement FragmentInteractionListener");
        }
```

```
        mListener.setChildFragmentInteraction(this);

mListener.getFloatingActionButton().setImageResource(R.drawable.map_marker
_icon);

mListener.getFloatingActionButton().setBackgroundColor(getResources().getCol
or(R.color.colorWhite));

mListener.getFloatingActionButton().setBackgroundTintList(ColorStateList.value
Of(0xFFFFFFFF));
        mListener.getFloatingActionButton().setVisibility(View.GONE);

mListener.getFloatingActionButton().setOnClickListener(mFloatingActionButtonLi
stener);

        googleApiClient = new GoogleApiClient.Builder(context)
                .addApi(LocationServices.API)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .build();

        locationRequest = new LocationRequest();
        //locationRequest.setSmallestDisplacement(MIN_DISPLACEMENT);
        locationRequest.setInterval(10000); // every 10 seconds
        locationRequest.setFastestInterval(10000); // every 10 seconds

locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

        mLocationCallback = new LocationCallback() {
            @Override
            public void onLocationResult(LocationResult locationResult) {
                for (Location location : locationResult.getLocations()) {
                    centerMap(location);
                }
            }
        };

        mFusedLocationClient =
LocationServices.getFusedLocationProviderClient(context);
        /*if (SystemPermissionUtils.checkAllLocationPermission(context)) {

LocationServices.FusedLocationApi.requestLocationUpdates(googleApiClient,
locationRequest, locationListener);
        }*/
    }
```

```
    private LocationCallback mLocationCallback;
    private FusedLocationProviderClient mFusedLocationClient;
    private static final int MIN_DISPLACEMENT = 10;
    private static final int LOCATION_REQUEST_CODE = 101;

    public void onStart() {
        super.onStart();
        googleApiClient.connect();
    }

    public void onStop() {
        super.onStop();
        mFusedLocationClient.removeLocationUpdates(mLocationCallback);
        googleApiClient.disconnect();
    }

    @Override
    public void onDetach() {
        mListener.getFloatingActionButton().setOnClickListener(null);
        super.onDetach();
    }

    private final View.OnClickListener mFloatingActionButtonListener = new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Snackbar.make(view, "Centrar mapa", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
        }
    };

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        //mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);

        Location location = GeoLocationUtils.getLastLocation(getActivity());
        centerMap(location);
        if (location.getExtras() != null && location.getExtras().get("nic") != null) {
            isDefaultNic = location.getExtras().getBoolean("nic");
        }

        //GoogleMapUtils.drawRoute(getContext(), mMap, null, null);
    }

    private void centerMap(Location location) {
```

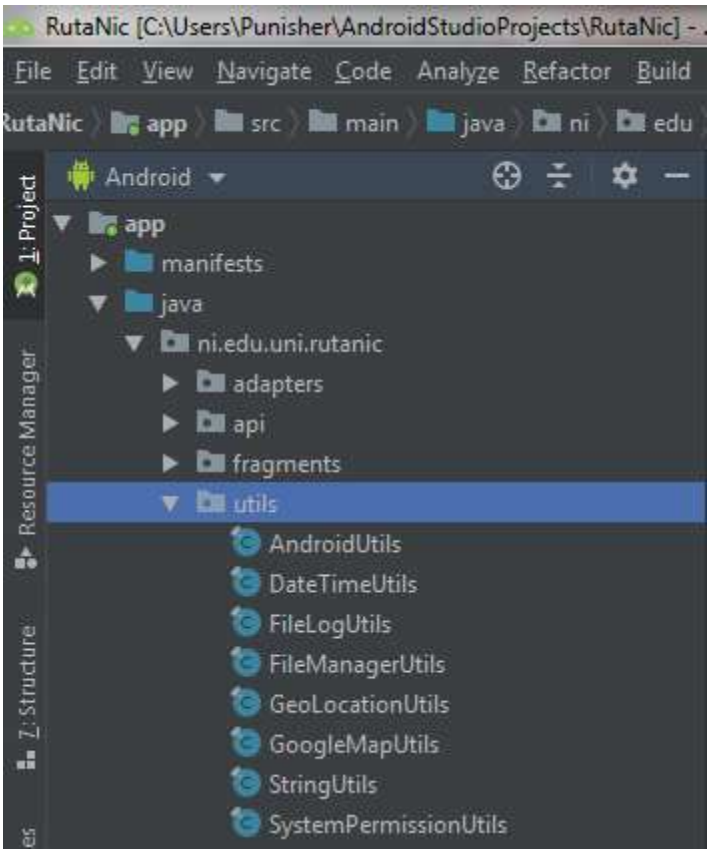
```
        if (location == null) {
            return;
        }
        LatLng pos = new LatLng(location.getLatitude(), location.getLongitude());
        centerMap(pos);
    }

    private void centerMap(LatLng pos) {
        if (isCentered && !isDefaultNic) {
            return;
        }
        isCentered = true;
        isDefaultNic = false;
        mMap.clear();
        mMap.addMarker(new MarkerOptions().position(pos).title("Marker in
Sydney"));
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(pos, 15.0f));
        mMap.getUiSettings().setZoomControlsEnabled(true);
    }

    private GoogleApiClient googleApiClient;
    private boolean isConnected;
    private LocationRequest locationRequest;

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        if (getActivity() == null) {
            return;
        }
        isConnected = true;
        if (ActivityCompat.checkSelfPermission(getActivity(),
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(getActivity(),
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            return;
        }
        mFusedLocationClient.requestLocationUpdates(locationRequest,
mLocationCallback, null);
        mListener.onFragmentInteraction(0, null);
    }

    @Override
    public void onConnectionSuspended(int i) {
        isConnected = false;
```

```
    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult
connectionResult) {
    }

    @Override
    public void onFragmentInteraction(int type, Object value) {
        if (type == 0) {
            etDestino.setText("");
            vwDestino.setVisibility(View.VISIBLE);
            AndroidUtils.focusView(etDestino);
            AndroidUtils.showSoftKeyboard(getActivity(), etDestino);
            return;
        }

        if (type == 1) {
            if ("Cancelar".equals(value)) {
                vwDestino.setVisibility(View.GONE);
            } else {
                cntEntidad.setVisibility(View.GONE);
                calcularRutas();
            }
        }

        if (type == 3) {
            cargarRuta((ApiModel) value);
        }
        if (type == 4) {
            dibujarPuntos();
        }

        if (type == 11) {
            mostrarRutas();
        }
    }

    private ApiModel lastSelected;

    public void cargarRuta(ApiModel pModel) {
        lastSelected = pModel;
        RutaEstablecimiento model = (RutaEstablecimiento) pModel;

        rvRutas.setVisibility(View.GONE);
```

```
        progressBar.setVisibility(View.VISIBLE);
        cntEntidad.setVisibility(View.VISIBLE);
        cntEntidad.findViewById(R.id.textView2).setVisibility(View.VISIBLE);
        cntEntidad.findViewById(R.id.tvDistance).setVisibility(View.VISIBLE);
        cntEntidad.findViewById(R.id.imgWlkCn).setVisibility(View.VISIBLE);
        ((TextView)
cntEntidad.findViewById(R.id.textView1)).setText(model.getDescription());
        ((TextView)
cntEntidad.findViewById(R.id.textView2)).setText(model.getDescriptionDetail());
        ((TextView) cntEntidad.findViewById(R.id.tvDistance)).setText(
                mApp.getString(R.string.app_distancia_metros,
model.getDistanciaPieTotal())
        );
        ((ImageView)
cntEntidad.findViewById(R.id.imageView1)).setImageDrawable(getResources().g
etDrawable(model.getIconResource()));
        ((ImageView)
cntEntidad.findViewById(R.id.imageButton1)).setImageDrawable(getResources()
.getDrawable(R.drawable.go_back_icon));
        cntEntidad.findViewById(R.id.imageButton1).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                cargarListaRutas();
            }
        });

        if (mMap != null) {
            obtenerPuntos(lastSelected);
            return;
        }
        SupportMapFragment fragment = (SupportMapFragment)
getChildFragmentManager().findFragmentById(R.id.frg);
        if (fragment != null) {
            fragment.getMapAsync(this);
        }
    }

    public void obtenerPuntos(ApiModel model) {
        mApp.getWs().obtenerDatosRuta(model.getId(), this);
    }

    public void dibujarPuntos() {
        progressBar.setVisibility(View.GONE);
        vwGoogleMap.setVisibility(View.VISIBLE);
        GoogleMapUtils.drawRoute(getContext(), mMap,
```

```
lastSelected.getDescription(),
            mApp.getWs().rutaPuntos, lastRutaEstablecimiento.get(0), "D");
    }

    public void cargarListaRutas() {
        mMap.clear();
        cntEntidad.setVisibility(View.GONE);
        rvRutas.setVisibility(View.VISIBLE);
    }

    private List<RutaEstablecimiento> lastRutaEstablecimiento;

    @SuppressWarnings("unchecked")
    public void mostrarRutas() {
        progressBar.setVisibility(View.GONE);

        if (mApp.getWs().rutasCalculadas.size() < 1) {
            cntEntidad.setVisibility(View.GONE);
            rvRutas.setVisibility(View.GONE);
            vwGoogleMap.setVisibility(View.VISIBLE);
            return;
        }

        vwGoogleMap.setVisibility(View.GONE);
        lastRutaEstablecimiento = mApp.getWs().rutasCalculadas;
        Collections.sort(lastRutaEstablecimiento, new
Comparator<RutaEstablecimiento>() {
            @Override
            public int compare(RutaEstablecimiento o1, RutaEstablecimiento o2) {
                if (o1.getDistanciaPieTotal() == null || o2.getDistanciaPieTotal() == null)
{
                    return 0;
                }
                return o1.getDistanciaPieTotal() < o2.getDistanciaPieTotal() ? -1 : 1;
            }
        });

        adapterRutas = new CardListAdapter(getActivity(), this, (List)
lastRutaEstablecimiento);
        rvRutas.setAdapter(adapterRutas);

        rvRutas.setVisibility(View.VISIBLE);
    }

    public void calcularRutas() {
        pbSearch.setVisibility(View.VISIBLE);
```

```
        LatLng latLng = getLocationFromAddress(getContext(),
String.valueOf(etDestino.getText()));
        if (latLng == null) {
            return;
        }
        LatLng latLngOrg;
        if (!TextUtils.isEmpty(etOrigen.getText())) {
            latLngOrg = getLocationFromAddress(getContext(),
String.valueOf(etOrigen.getText()));
            if (latLngOrg == null) {
                return;
            }
        } else {
            Location loc = GeoLocationUtils.getLastLocation(getContext());
            if (loc != null) {
                latLngOrg = new LatLng(loc.getLatitude(), loc.getLongitude());
            } else {
                return;
            }
        }
        isCentered = false;
        centerMap(latLng);

        pbSearch.setVisibility(View.GONE);
        vwDestino.setVisibility(View.GONE);
        AndroidUtils.hideSoftKeyboard(getActivity());

        progressBar.setVisibility(View.VISIBLE);
        mApp.getWs().calcularRuta(latLngOrg.latitude, latLngOrg.longitude,
latLng.latitude, latLng.longitude, this);
    }

    @Override
    public void setChildFragmentInteraction(FragmentInteractionListener listener) {
    }

    @Override
    public FloatingActionButton getFloatingActionButton() {
        return null;
    }
}
```



En este paquete se agregaron todas las utilidades que facilitan el uso de zona horaria, geolocalizacion, conversión de cadenas, permisos del sistema, entre otras utilidades.

IV.13. AndroidUtils:

```
package ni.edu.uni.rutanic.utils;

import android.app.Activity;
import android.app.ActivityManager;
import android.content.Context;
import android.content.pm.PackageManager;
import android.net.ConnectivityManager;
import android.os.Build;
import android.util.Log;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.EditText;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

import java.util.List;

public final class AndroidUtils {
    private AndroidUtils() {
```

```
        throw new AssertionError();
    }

    private static final String LOG_DEBUG_TAG = "APP_";

    static void debugMessage(@NonNull String tag, @NonNull String message,
@Nullable String method) {
        if (method != null && !method.equals("")) {
            Log.d(LOG_DEBUG_TAG + tag, "[" + method + "] " + message);
        } else {
            Log.d(LOG_DEBUG_TAG + tag, message);
        }
    }

    public static boolean isNetworkConnected(@NonNull Context context) {
        boolean result = false;
        if
(SystemPermissionUtils.checkAllInternetPermission(context.getApplicationConte
xt())) {
            ConnectivityManager cm = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
            result = (cm.getActiveNetworkInfo() != null &&
cm.getActiveNetworkInfo().isAvailable() &&
cm.getActiveNetworkInfo().isConnected());
        }
        return result;
    }

    @NonNull
    public static String getAppVersion(@NonNull Context context) {
        String ret;
        try {
            ret =
context.getPackageManager().getPackageInfo(context.getPackageName(),
0).versionName;
        } catch (PackageManager.NameNotFoundException ex) {
            ret = "0.0";
        }
        return ret;
    }

    public static String getAppName(@NonNull Context context) {
        int pID = android.os.Process.myPid();
        String processName = "";
        ActivityManager am = (ActivityManager)
context.getSystemService(Context.ACTIVITY_SERVICE);
```

```
        List runningProcesses = am.getRunningAppProcesses();
        for (Object item : runningProcesses) {
            ActivityManager.RunningAppProcessInfo info =
(ActivityManager.RunningAppProcessInfo) item;
            if (info.pid == pID) {
                processName = info.processName;
                break;
            }
        }
        return processName;
    }

    public static void showSoftKeyboard(Activity activity, View view) {
        if (focusView(view)) {
            InputMethodManager imm = (InputMethodManager)
activity.getSystemService(Context.INPUT_METHOD_SERVICE);
            imm.showSoftInput(view, InputMethodManager.SHOW_IMPLICIT);
            imm.toggleSoftInput(InputMethodManager.SHOW_IMPLICIT,
InputMethodManager.HIDE_IMPLICIT_ONLY);
        }
    }

    public static void hideSoftKeyboard(Activity activity) {
        if (activity == null) {
            return;
        }
        View view = activity.getCurrentFocus();
        if (view != null) {
            InputMethodManager imm = (InputMethodManager)
activity.getSystemService(Context.INPUT_METHOD_SERVICE);
            imm.hideSoftInputFromWindow(view.getWindowToken(),
InputMethodManager.HIDE_NOT_ALWAYS);
        }
    }

    @NonNull
    public static String getTopActivityName(Context context) {
        ActivityManager am = (ActivityManager)
context.getSystemService(Context.ACTIVITY_SERVICE);
        String result;
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.LOLLIPOP) {
            @SuppressWarnings("deprecation")
            List<ActivityManager.RunningTaskInfo> taskInfo =
am.getRunningTasks(1);
            result = taskInfo.get(0).topActivity.getClassName();
        } else {
```

```
            List<ActivityManager.RunningAppProcessInfo> tasks =
am.getRunningAppProcesses();
            result = tasks.get(0).processName;
        }
        return (result == null ? "" : result);
    }

    public static boolean focusView(View view) {
        view.setFocusableInTouchMode(true);
        view.setFocusable(true);
        return view.requestFocus();
    }

    public static void disableUnFocusedViews(View... views) {
        for (View view : views) {
            if (view instanceof EditText) {
                ((EditText) view).setKeyListener(null);
            } else {
                view.setEnabled(false);
                view.setFocusable(false);
                view.setFocusableInTouchMode(false);
            }
            //view.clearFocus();
        }
    }
}
```

IV.14. DateTimeUtils:

```
package ni.edu.uni.rutanic.utils;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.Locale;

public final class DateTimeUtils {
    private DateTimeUtils() {
        throw new AssertionError();
    }

    private static final String
            DEFAULT_DATE_FORMAT = "yyyy-MM-dd",
            DEFAULT_TIME_FORMAT = "HH:mm:ss",
            DEFAULT_TIME_DURATION_FORMAT = "HH:mm",
            DEFAULT_TIME_ITERATION_FORMAT = "hh:mm a";

    public static String getDurationFromString(String value) {
        try {
            Date rst = new SimpleDateFormat("HHmm", Locale.US).parse(value);
            Calendar calendar = GregorianCalendar.getInstance();
            calendar.setTime(rst);
            if (calendar.get(Calendar.HOUR_OF_DAY) >= 1) {
                return "" + calendar.get(Calendar.HOUR_OF_DAY) + " horas" +
                        (calendar.get(Calendar.MINUTE) > 0 ? " y " +
calendar.get(Calendar.MINUTE) + " minutos" : "");
            } else {
                return calendar.get(Calendar.MINUTE) + " minutos";
            }
            /*DateFormat dateFormat = new
SimpleDateFormat(DEFAULT_TIME_DURATION_FORMAT,
StringUtils.LOCALE_DEFAULT);
            return dateFormat.format(rst);*/

        } catch (Exception e) {
            return value;
        }
    }

    public static String getIterationFromString(String value) {
        try {
```

```
            Date rst = new SimpleDateFormat("HHmm", Locale.US).parse(value);
            DateFormat dateFormat = new
SimpleDateFormat(DEFAULT_TIME_ITERATION_FORMAT,
StringUtils.LOCALE_DEFAULT);
            return dateFormat.format(rst).toUpperCase();

        } catch (Exception e) {
            return value;
        }
    }

    public static String getDate() {
        return getDate(null);
    }

    public static String getDate(Date date) {
        DateFormat dateFormat = new
SimpleDateFormat(DEFAULT_DATE_FORMAT,
StringUtils.LOCALE_DEFAULT);
        Date ldt = (date != null ? date : new Date());
        return dateFormat.format(ldt);
    }

    public static String getTime() {
        return getTime(null);
    }

    private static String getTime(Date date) {
        DateFormat dateFormat = new
SimpleDateFormat(DEFAULT_TIME_FORMAT,
StringUtils.LOCALE_DEFAULT);
        Date ldt = (date != null ? date : new Date());
        return dateFormat.format(ldt);
    }
}
```

IV.15.GoogleMapUtils:

```
package ni.edu.uni.rutanic.utils;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.location.Location;

import androidx.annotation.NonNull;

import com.directions.route.Route;
import com.directions.route.RouteException;
import com.directions.route.Routing;
import com.directions.route.RoutingListener;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.PolylineOptions;
import com.google.maps.android.SphericalUtil;

import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

import ni.edu.uni.rutanic.R;
import ni.edu.uni.rutanic.api.models.RutaEstablecimiento;
import ni.edu.uni.rutanic.api.models.RutaPunto;
import ni.edu.uni.rutanic.api.models.RutaTiempo;
import ni.edu.uni.rutanic.api.models.RutaTransicion;

@SuppressWarnings("unused")
public final class GoogleMapUtils {
    private GoogleMapUtils() {
        throw new AssertionError();
    }

    public static void drawRoute(Context context, GoogleMap mMap, String title,
                                 List<RutaPunto> puntos,
                                 List<RutaTiempo> tiempos,
                                 List<RutaTransicion> transiciones,
                                 String orientacion) {
```

```
        mMap.clear();
        if (puntos.size() < 1) {
            return;
        }
        PolylineOptions rectOptions = new PolylineOptions();
        int index = 0;
        LatLng iniPnt = null;
        LatLng lstPnt = null;
        for (RutaPunto item : puntos) {
            if (!orientacion.equals(item.getTipo_orientacion())) {
                continue;
            }
            LatLng ill = new LatLng(Double.parseDouble(item.getLatitud()),
Double.parseDouble(item.getLongitud()));
            rectOptions.add(ill);
            if (index == 0) {
                iniPnt = ill;
            }
            lstPnt = ill;
            /*mMap.addMarker(new MarkerOptions()
                    .position(ill)
                    .title("Parada")
                    .snippet("Id: " + item.getId())

.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_RE
D)));*/
            if (index > 0 && item.getEstablecimiento_id() > 0) {
                mMap.addMarker(new MarkerOptions()
                        .position(ill)
                        .title("Parada")
                        //.snippet("Id: " + item.getId())

.icon(BitmapDescriptorFactory.fromBitmap(resizeMapIcons(context,
R.drawable.app_img_parada, 36, 36, false))));
            }
            index++;
        }
        if (lstPnt == null) {
            return;
        }

        mMap.addMarker(new MarkerOptions()
                .position(iniPnt)
                .title(title)
                .icon(BitmapDescriptorFactory.fromBitmap(resizeMapIcons(context,
R.drawable.app_ruta_ini, 48, 48))));
```

```
        mMap.addMarker(new MarkerOptions()
                .position(lstPnt)
                .icon(BitmapDescriptorFactory.fromBitmap(resizeMapIcons(context,
R.drawable.app_ruta_fin, 48, 48))));
        mMap.getUiSettings().setCompassEnabled(true);
        mMap.getUiSettings().setZoomControlsEnabled(true);
        mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(iniPnt,
12));

        //Set the rectangle's stroke color to red
        //rectOptions.strokeColor(Color.BLUE);
        //Set the rectangle's fill to blue
        //rectOptions.fillColor(Color.CYAN);
        //rectOptions.strokeWidth(4);
        //Get back the mutable Polygon
        //mMap.addPolygon(rectOptions);

        rectOptions.width(5);
        rectOptions.color(context.getResources().getColor(R.color.colorPrimary));
        mMap.addPolyline(rectOptions);
    }

    @NonNull
    public static String getDistance(@NonNull List<RutaPunto> puntos) {
        double mtrs = 0.0;
        for (int i = 1; i < puntos.size(); i++) {
            Location i1 = new Location("gps");
            Location i2 = new Location("gps");
            i1.setLatitude(Double.parseDouble(puntos.get(i - 1).getLatitud()));
            i1.setLongitude(Double.parseDouble(puntos.get(i - 1).getLongitud()));
            i2.setLatitude(Double.parseDouble(puntos.get(i).getLatitud()));
            i2.setLongitude(Double.parseDouble(puntos.get(i).getLongitud()));
            mtrs += i1.distanceTo(i2);
        }
        return String.format(Locale.US, "%.2f", mtrs / 1000.0);
    }

    @NonNull
    public static String getDuration(@NonNull List<RutaPunto> puntos) {
        double mtrs = 0.0;
        for (int i = 1; i < puntos.size(); i++) {
            Location i1 = new Location("gps");
            Location i2 = new Location("gps");
            i1.setLatitude(Double.parseDouble(puntos.get(i - 1).getLatitud()));
            i1.setLongitude(Double.parseDouble(puntos.get(i - 1).getLongitud()));
            i2.setLatitude(Double.parseDouble(puntos.get(i).getLatitud()));
```

```
            i2.setLongitude(Double.parseDouble(puntos.get(i).getLongitud()));
            mtrs += i1.distanceTo(i2);
        }
        return String.format(Locale.US, "%.2f", (mtrs / 1000.0) / 45.0);
    }

    public static void drawRoute(Context context, GoogleMap mMap, String title,
                                 List<RutaPunto> puntos,
                                 RutaEstablecimiento ruta,
                                 String orientacion) {
        mMap.clear();
        if (puntos.size() < 1) {
            return;
        }

        LatLng grp1 = new LatLng(Double.parseDouble(ruta.getLttOrgRef()),
Double.parseDouble(ruta.getLngOrgRef()));
        LatLng grp2 = new LatLng(Double.parseDouble(ruta.getLttDstRef()),
Double.parseDouble(ruta.getLngDstRef()));
        LatLng stb1 = new
LatLng(Double.parseDouble(ruta.getEstablecOrg().getLatitud()),
Double.parseDouble(ruta.getEstablecOrg().getLongitud()));
        LatLng stb2 = new
LatLng(Double.parseDouble(ruta.getEstablecDst().getLatitud()),
Double.parseDouble(ruta.getEstablecDst().getLongitud()));

        PolylineOptions rectOptions = new PolylineOptions();
        int index = 0;
        double dSI = 5000.0, dSF = 5000.0, dI, dF;
        LatLng iniPnt = null;
        LatLng lstPnt = null;
        for (RutaPunto item : puntos) {
            if (!orientacion.equals(item.getTipo_orientacion())) {
                continue;
            }
            LatLng ill = new LatLng(Double.parseDouble(item.getLatitud()),
Double.parseDouble(item.getLongitud()));

            if (item.getEstablecimiento_id() > 0) {
                dI = SphericalUtil.computeDistanceBetween(grp1, ill);
                dF = SphericalUtil.computeDistanceBetween(grp2, ill);
                if (dI < dSI) {
                    dSI = dI;
                    stb1 = new LatLng(ill.latitude, ill.longitude);
                }
                if (dF < dSF) {
```

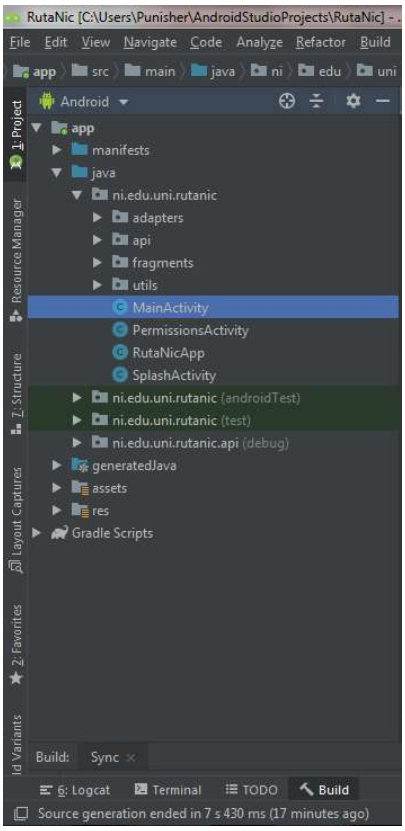
```
                dSF = dF;
                stb2 = new LatLng(ill.latitude, ill.longitude);
            }
        }

        rectOptions.add(ill);
        lstPnt = ill;
        if (index == 0) {
            iniPnt = ill;
        }
        if (index > 0 && item.getEstablecimiento_id() > 0) {
            mMap.addMarker(new MarkerOptions()
                    .position(ill)
                    .title("Parada")
                    //.snippet("Id: " + item.getId())

.icon(BitmapDescriptorFactory.fromBitmap(resizeMapIcons(context,
R.drawable.app_img_parada, 36, 36, false))));
        }
        index++;
    }
    if (iniPnt == null) {
        return;
    }

    mMap.addMarker(new MarkerOptions()
            .position(iniPnt)
            .title(title)

.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZ
URE)));
    mMap.addMarker(new MarkerOptions()
            .position(lstPnt)
            .title(title)

.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZ
URE)));
    mMap.getUiSettings().setCompassEnabled(true);
    mMap.getUiSettings().setZoomControlsEnabled(true);

    rectOptions.width(5);
    rectOptions.color(context.getResources().getColor(R.color.colorPrimary));
    mMap.addPolyline(rectOptions);

    mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(grp1, 15));
    mMap.addMarker(new MarkerOptions()
```

```
                .position(grp1)
                .title("Ubicación Actual")
                .icon(BitmapDescriptorFactory.fromBitmap(resizeMapIcons(context,
R.drawable.app_loc_ini, 32, 32))));
        mMap.addMarker(new MarkerOptions()
                .position(grp2)
                .title("Ubicación Final")
                .icon(BitmapDescriptorFactory.fromBitmap(resizeMapIcons(context,
R.drawable.app_loc_ini, 32, 32))));
        mMap.addMarker(new MarkerOptions()
                .position(stb1)
                .title("Parada Inicial")
                .icon(BitmapDescriptorFactory.fromBitmap(resizeMapIcons(context,
R.drawable.app_ruta_ini, 32, 32))));
        mMap.addMarker(new MarkerOptions()
                .position(stb2)
                .title("Parada Final")
                .icon(BitmapDescriptorFactory.fromBitmap(resizeMapIcons(context,
R.drawable.app_ruta_fin, 32, 32))));
        drawGoogleRoute(context, mMap, grp1, stb1, ruta.getModoViaje());
        drawGoogleRoute(context, mMap, stb2, grp2, ruta.getModoViaje());
    }

    private static Bitmap resizeMapIcons(Context ctx, int drawable, int width, int
height) {
        return resizeMapIcons(ctx, drawable, width, height, true);
    }

    private static Bitmap resizeMapIcons(Context ctx, int drawable, int width, int
height, boolean white) {
        Bitmap imageBitmap = BitmapFactory.decodeResource(ctx.getResources(),
drawable);
        if (!white) {
            return Bitmap.createScaledBitmap(imageBitmap, width, height, false);
        }
        Bitmap newBitmap = Bitmap.createBitmap(imageBitmap.getWidth(),
imageBitmap.getHeight(), imageBitmap.getConfig());
        Canvas canvas = new Canvas(newBitmap);
        canvas.drawColor(Color.WHITE);
        canvas.drawBitmap(imageBitmap, 0, 0, null);
        return Bitmap.createScaledBitmap(newBitmap, width, height, false);
    }

    private static void drawGoogleRoute(final Context context, final GoogleMap
mMap, final LatLng start, final LatLng end, String modoViaje) {
        Routing.TravelMode modo = "vehiculo".equals(modoViaje) ?
```

```
Routing.TravelMode.DRIVING : Routing.TravelMode.WALKING;
        Routing routing = new Routing.Builder()
                .travelMode(modo)
                .key("AIzaSyAuu8lZxFhvb5ORwo2LMXOMhbDAoFDgdx8")
                .withListener(new RoutingListener() {
                    @Override
                    public void onRoutingFailure(RouteException e) {
                    }

                    @Override
                    public void onRoutingStart() {
                    }

                    @Override
                    public void onRoutingSuccess(ArrayList<Route> route, int
shortestRouteIndex) {
                        for (int i = 0; i < route.size(); i++) {
                            PolylineOptions polyOptions = new PolylineOptions();

polyOptions.color(context.getResources().getColor(R.color.colorAccent));
                            polyOptions.width(10 + i * 3);
                            polyOptions.addAll(route.get(i).getPoints());
                            mMap.addPolyline(polyOptions);
                            //Toast.makeText(getContext(),"Route "+ (i+1) +": distance - "+
route.get(i).getDistanceValue()+": duration - "+
route.get(i).getDurationValue(),Toast.LENGTH_SHORT).show();
                        }

                        /*// Start marker
                        MarkerOptions options = new MarkerOptions();
                        options.position(start1);

options.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.H
UE_ORANGE));
                        mMap.addMarker(options);

                        // End marker
                        MarkerOptions options = new MarkerOptions();
                        options.position(end1);

options.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.H
UE_AZURE));
                        mMap.addMarker(options);*/
                    }

                    @Override
```

```
                public void onRoutingCancelled() {
                }
            })
            .waypoints(start, end)
            .build();
        routing.execute();
    }
}
```

IV.16. Modulos principales de la app



Estas son las clases o modulos principales de nuestra app. De de las cuales se llaman a los fragment correspondientes según el menú o la opción seleccionada.

IV.17. Clase RutaNicApp:

```
package ni.edu.uni.rutanic;

import android.app.Application;
import android.content.res.Configuration;

import ni.edu.uni.rutanic.api.Config;
```

```
import ni.edu.uni.rutanic.api.RutaNicWs;
import ni.edu.uni.rutanic.utils.AndroidUtils;
import ni.edu.uni.rutanic.utils.FileLogUtils;

public class RutaNicApp extends Application {
    private RutaNicWs ws;

    @Override
    public void onCreate() {
        super.onCreate();
        ws = new RutaNicWs(getApplicationContext());

        Thread.setDefaultUncaughtExceptionHandler(new
Thread.UncaughtExceptionHandler() {
            public void uncaughtException(Thread thread, Throwable ex) {
                Thread.setDefaultUncaughtExceptionHandler(null);
                if (!Config.DEBUG_MODE) {
                    return;
                }
                try {
                    FileLogUtils.general("globalEx[" +
AndroidUtils.getAppName(getApplicationContext()) + "]", ex);
                } catch (Exception e) {
                    FileLogUtils.general("globalEx", e);
                }
                android.os.Process.killProcess(android.os.Process.myPid());
            }
        });
    }

    public RutaNicWs getWs() {
        return ws;
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
    }

    @Override
    public void onLowMemory() {
        super.onLowMemory();
    }

    public interface OnWsInteractionListener {
        void onWsInteraction(int type, Object value);
```

```
    }
}
```

IV.18. MainActivity:

```
package ni.edu.uni.rutanic;

import android.os.Bundle;
import android.text.TextUtils;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.navigation.NavigationView;

import ni.edu.uni.rutanic.fragments.AcercaFragment;
import ni.edu.uni.rutanic.fragments.FragmentInteractionListener;
import ni.edu.uni.rutanic.fragments.MapaFragment;
import ni.edu.uni.rutanic.fragments.MenuFragment;
import ni.edu.uni.rutanic.fragments.RutasFragment;

public class MainActivity extends AppCompatActivity implements
        NavigationView.OnNavigationItemSelectedListener,
        FragmentInteractionListener {
    private DrawerLayout mDrawer;
    private FloatingActionButton fab;
    private MenuItem search;
    private NavigationView navigationView;
    private FragmentManager fragmentManager;

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

```
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        fab = findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                /*Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();*/
            }
        });
        mDrawer = findViewById(R.id.drawer_layout);
        navigationView = findViewById(R.id.nav_view);
        //setupDrawerContent(navigationView);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                this, mDrawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        mDrawer.addDrawerListener(toggle);
        toggle.syncState();
        navigationView.setNavigationItemSelectedListener(this);

        if (savedInstanceState != null) {
            return;
        }

        fragmentManager = getSupportFragmentManager();
        FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();
        //fragmentTransaction.replace(R.id.flContent, new MapaFragment());
        fragmentTransaction.replace(R.id.flContent, new MenuFragment());
        //fragmentTransaction.addToBackStack(String.valueOf(R.id.nav_inicio));
        fragmentTransaction.commit();
        fragmentManager.addOnBackStackChangedListener(new
FragmentManager.OnBackStackChangedListener() {
            @Override
            public void onBackStackChanged() {
                FragmentManager fm = getSupportFragmentManager();
                if (fm.getBackStackEntryCount() <= 0) {
                    return;
                }
                String name = fm.getBackStackEntryAt(fm.getBackStackEntryCount() -
1).getName();
                if (name != null && !TextUtils.isEmpty(name)) {
                    int rscId = Integer.parseInt(name);
                    selectNavigationDrawerItem(rscId);
```

```
            }
        }
    });
}

public void selectNavigationDrawerItem(int id) {
    navigationView.setCheckedItem(id);
}

@Override
protected void onResume() {
    super.onResume();
}

@Override
public void onBackPressed() {
    DrawerLayout drawer = findViewById(R.id.drawer_layout);
    if (drawer.isDrawerOpen(GravityCompat.START)) {
        drawer.closeDrawer(GravityCompat.START);
        return;
    }
    if (fragmentManager.getBackStackEntryCount() > 0) {
        String name =
fragmentManager.getBackStackEntryAt(fragmentManager.getBackStackEntryCo
unt() - 1).getName();
        if (name != null && !TextUtils.isEmpty(name) && Integer.parseInt(name)
== R.id.nav_inicio) {
            finish();
            return;
        }
    }
    super.onBackPressed();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    search = menu.findItem(R.id.action_search);
    search.setVisible(false);
    return true;
}

@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();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        if (id == R.id.action_search) {
            if (child != null) {
                child.onFragmentInteraction(0, "HOLA");
            }
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @SuppressWarnings("StatementWithEmptyBody")
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        // Handle navigation view item clicks here.
        int id = item.getItemId();
        Fragment fragment;
        Class fragmentClass = null;
        Bundle bundle = new Bundle();

        if (id == R.id.nav_inicio) {
            fragmentClass = MenuFragment.class;
        } else if (id == R.id.nav_rutas_managua) {
            fragmentClass = RutasFragment.class;
        } else if (id == R.id.nav_terminales) {
            fragmentClass = RutasFragment.class;
            bundle.putString("param1", "Terminales");
        } else if (id == R.id.nav_como_llegar) {
            fragmentClass = MapaFragment.class;
        } else if (id == R.id.nav_acerca) {
            fragmentClass = AcercaFragment.class;
        }

        if (fragmentClass != null) {
            try {
                fragment = (Fragment) fragmentClass.newInstance();
                fragment.setArguments(bundle);
            } catch (Exception e) {
                mDrawer.closeDrawer(GravityCompat.START);
```

```
            return true;
        }
    } else {
        mDrawer.closeDrawer(GravityCompat.START);
        return true;
    }

    // Insert the fragment by replacing any existing fragment
    FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();
    fragmentTransaction.replace(R.id.flContent, fragment);
    fragmentTransaction.addToBackStack(String.valueOf(id));

fragmentTransaction.setTransition(FragmentTransaction.TRANSIT_FRAGMENT
_OPEN);
    fragmentTransaction.commit();

    // Highlight the selected item has been done by NavigationView
    item.setChecked(true);
    // Set action bar title
    setTitle(item.getTitle());

    mDrawer.closeDrawer(GravityCompat.START);
    return true;
}

FragmentInteractionListener child;

public void onMenuClick(View view) {
    int id = view.getId();
    Fragment fragment;
    Bundle bundle = new Bundle();
    switch (id) {
        case R.id.btn_acerca:
            setTitle(getString(R.string.menu_acerca));
            fragment = new AcercaFragment();
            break;
        case R.id.btn_como_llegar:
            setTitle(getString(R.string.menu_como_llegar));
            fragment = new MapaFragment();
            break;
        case R.id.btn_rutas_managua:
            setTitle(getString(R.string.menu_rutas_managua));
            fragment = new RutasFragment();
            break;
        case R.id.btn_terminales:
```

```
                setTitle(getString(R.string.menu_terminales));
                fragment = new RutasFragment();
                bundle.putString("param1", "Terminales");
                break;
            default:
                fragment = new AcercaFragment();
        }
        fragment.setArguments(bundle);
        FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.flContent, fragment);
        fragmentTransaction.addToBackStack(String.valueOf(id));

fragmentTransaction.setTransition(FragmentTransaction.TRANSIT_FRAGMENT
_OPEN);
        fragmentTransaction.commit();
    }

    @Override
    public void onFragmentInteraction(int type, Object value) {
        if (search != null) {
            if (type == 99) {
                search.setVisible(false);
            } else {
                search.setVisible(true);
            }
        }
    }

    @Override
    public void setChildFragmentInteraction(FragmentInteractionListener listener) {
        child = listener;
    }

    @Override
    public FloatingActionButton getFloatingActionButton() {
        return fab;
    }

    public void CancelarBusqueda(View view) {
        child.onFragmentInteraction(1, "Cancelar");
    }

    public void EjecutarBusqueda(View view) {
        child.onFragmentInteraction(1, "Buscar");
```

```
    }
}
```

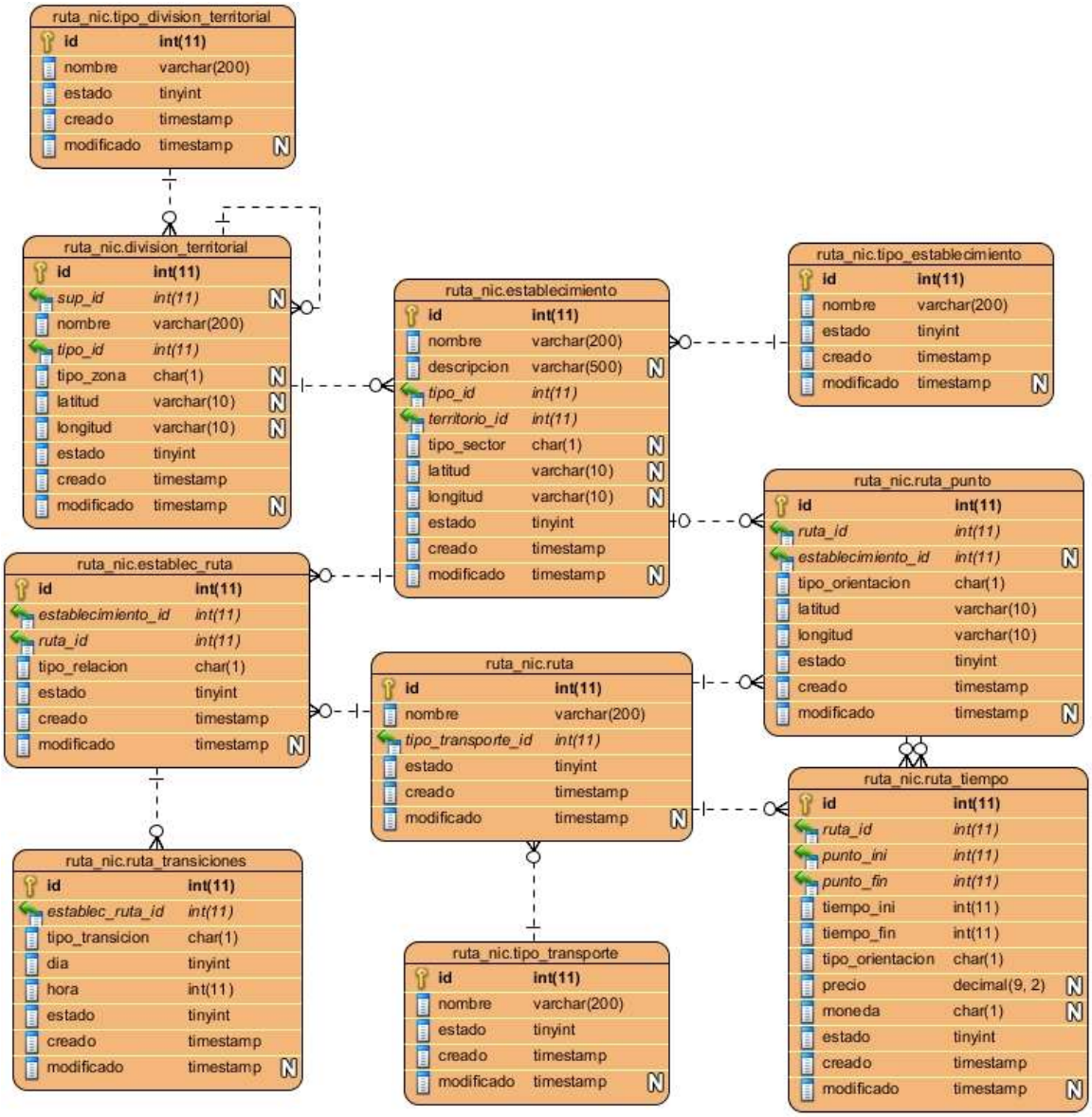
IV.19. Archivo Gradle(Module: app):

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 28

    defaultConfig {
        applicationId "ni.edu.uni.rutanic"
        minSdkVersion 16
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        vectorDrawables.useSupportLibrary = true
    }

    /*compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }*/

    packagingOptions {
        exclude 'META-INF/LICENSE'
        exclude 'META-INF/LICENSE-FIREBASE.txt'
        exclude 'META-INF/NOTICE'

        return void
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
        }
    }

    testOptions {
        unitTests.returnDefaultValues = true
    }

    testBuildType "debug"
```

```
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.1.0-alpha04'
    implementation 'com.google.android.material:material:1.1.0-alpha05'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.0-alpha5'

    implementation 'com.google.android.gms:play-services-location:16.0.0'
    implementation 'com.google.android.gms:play-services-maps:16.1.0'
    implementation 'com.google.maps.android:android-maps-utils:0.5'

    implementation 'com.github.jd-alexander:library:1.1.0'

    implementation 'com.airbnb.android:lottie:2.7.0' // 2.1.0

    //implementation 'com.github.JakeWharton:ViewPagerIndicator:2.4.1'

    /*implementation 'com.google.code.gson:gson:2.8.5'
    implementation 'com.squareup.retrofit2:retrofit:2.5.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.5.0'
    implementation 'com.squareup.okhttp3:okhttp:3.14.1'*/
    implementation 'com.google.code.gson:gson:2.7'
    implementation 'com.squareup.retrofit2:retrofit:2.1.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.1.0'
    implementation 'com.squareup.okhttp3:okhttp:3.5.0'

    implementation 'org.projectlombok:lombok:1.18.6'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation 'androidx.vectordrawable:vectordrawable:1.0.1'
    annotationProcessor 'org.projectlombok:lombok:1.18.6'

    testImplementation 'junit:junit:4.13-beta-2'

    androidTestImplementation 'androidx.test:runner:1.2.0-alpha04'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0-
alpha04'
}
```

V. Estructura de datos

El siguiente modelo relacional muestra de que manera estan relacionadas las entidades en el sistema. Este se creó haciendo uso de VisualParadigma



V.1. Diccionario de datos

division_territorial

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	int(11)			No	*Ninguna*	AUTO_INCREMENT
2	sup_id	int(11)			Sí	*NULL*	
3	nombre	varchar(200)	utf8_unicode_ci		No	*Ninguna*	
4	tipo_id	int(11)			No	*Ninguna*	
5	tipo_zona	char(1)	utf8_unicode_ci		Sí	*NULL*	
6	latitud	varchar(10)	utf8_unicode_ci		Sí	*NULL*	
7	longitud	varchar(10)	utf8_unicode_ci		Sí	*NULL*	
8	estado	tinyint(4)			No	1	
9	creado	timestamp			No	CURRENT_TIMESTAMP	
10	modificado	timestamp			Sí	*NULL*	

establecimiento

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	int(11)			No	*Ninguna*	AUTO_INCREMENT
2	nombre	varchar(200)	utf8_unicode_ci		No	*Ninguna*	
3	descripcion	varchar(500)	utf8_unicode_ci		Sí	*NULL*	
4	tipo_id	int(11)			No	*Ninguna*	
5	territorio_id	int(11)			No	*Ninguna*	
6	tipo_sector	char(1)	utf8_unicode_ci		Sí	*NULL*	
7	latitud	varchar(10)	utf8_unicode_ci		Sí	*NULL*	
8	longitud	varchar(10)	utf8_unicode_ci		Sí	*NULL*	
9	estado	tinyint(4)			No	1	
10	creado	timestamp			No	CURRENT_TIMESTAMP	
11	modificado	timestamp			Sí	*NULL*	

establec_ruta

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	int(11)			No	*Ninguna*	AUTO_INCREMENT
2	establecimiento_id	int(11)			No	*Ninguna*	
3	establec	varchar(1024)	utf8_unicode_ci		Si	*Ninguna*	
4	ruta_id	int(11)			No	*Ninguna*	
5	ruta	varchar(1024)	utf8_unicode_ci		Si	*Ninguna*	

6	tipo_relacion	char(1)	utf8_unicode_ci		No	*Ninguna*	
7	estado	tinyint(4)			No	1	
8	creado	timestamp			No	CURRENT_TIMESTAMP	
9	modificado	timestamp			Sí	*NULL*	

ruta

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	int(11)			No	*Ninguna*	AUTO_INCREMENT
2	nombre	varchar(200)	utf8_unicode_ci		No	*Ninguna*	
3	tipo_transporte_id	int(11)			No	*Ninguna*	
4	estado	tinyint(4)			No	1	
5	creado	timestamp			No	CURRENT_TIMESTAMP	
6	modificado	timestamp			Sí	*NULL*	

ruta_punto

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	int(11)			No	*Ninguna*	AUTO_INCREMENT
2	ruta_id	int(11)			No	*Ninguna*	
3	establecimiento_id	int(11)			Sí	*NULL*	
4	tipo_orientacion	char(1)	utf8_unicode_ci		No	*Ninguna*	
5	latitud	varchar(10)	utf8_unicode_ci		No	*Ninguna*	
6	longitud	varchar(10)	utf8_unicode_ci		No	*Ninguna*	
7	estado	tinyint(4)			No	1	
8	creado	timestamp			No	CURRENT_TIMESTAMP	
9	modificado	timestamp			Sí	*NULL*	

ruta_tiempo

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	int(11)			No	*Ninguna*	AUTO_INCREMENT
2	ruta_id	int(11)			No	*Ninguna*	
3	punto_ini	int(11)			Sí	*NULL*	
4	punto_fin	int(11)			Sí	*NULL*	
5	tiempo_ini	int(11)			No	*Ninguna*	
6	tiempo_fin	int(11)			No	*Ninguna*	
7	tipo_orientacion	char(1)	utf8_unicode_ci		No	*Ninguna*	
8	precio	decimal(9,2)			Sí	*NULL*	
9	moneda	char(1)	utf8_unicode_ci		Sí	*NULL*	
10	estado	tinyint(4)			No	1	
11	creado	timestamp			No	CURRENT_TIMESTAMP	

12	modificado	timestamp			Sí	*NULL*	

ruta_transicion

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	int(11)			No	*Ninguna*	AUTO_INCREMENT
2	establec_ruta_id	int(11)			No	*Ninguna*	
3	tipo_transicion	char(1)	utf8_unicode_ci		No	*Ninguna*	
4	dia	tinyint(4)			No	1	
5	hora	int(11)			No	*Ninguna*	
6	estado	tinyint(4)			No	1	
7	creado	timestamp			No	CURRENT_TIMESTAMP	
8	modificado	timestamp			Sí	*NULL*	

tipo_division_territorial

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	int(11)			No	*Ninguna*	AUTO_INCREMENT
2	nombre	varchar(200)	utf8_unicode_ci		No	*Ninguna*	
3	estado	tinyint(4)			No	1	
4	creado	timestamp			No	CURRENT_TIMESTAMP	
5	modificado	timestamp			Sí	*NULL*	

tipo_establecimiento

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	int(11)			No	*Ninguna*	AUTO_INCREMENT
2	nombre	varchar(200)	utf8_unicode_ci		No	*Ninguna*	
3	estado	tinyint(4)			No	1	
4	creado	timestamp			No	CURRENT_TIMESTAMP	
5	modificado	timestamp			Sí	*NULL*	

tipo_transporte

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	int(11)			No	*Ninguna*	AUTO_INCREMENT
2	nombre	varchar(200)	utf8_unicode_ci		No	*Ninguna*	
3	estado	tinyint(4)			No	1	
4	creado	timestamp			No	CURRENT_TIMESTAMP	
5	modificado	timestamp			Sí	*NULL*	

usuario

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	int(4)			No	*Ninguna*	AUTO_INCREMENT

2	nombre	varchar(256)	latin1_swedish_ci	No	*Ninguna*
3	apellido	varchar(256)	latin1_swedish_ci	No	*Ninguna*
4	user	varchar(256)	latin1_swedish_ci	No	*Ninguna*
5	pass	varchar(1024)	latin1_swedish_ci	No	*Ninguna*
6	creado	datetime		No	CURRENT_TIMESTAMP
7	modificado	timestamp		No	CURRENT_TIMESTAMP
8	estado	int(1)		No	1

Tabla	Descripción
tipo_transporte	Tipos de transportes: urbano, rural, colectivo, privado, otros.
tipo_establecimiento	Tipos de establecimientos: paradas de bus, terminales, nercados, otros puntos de referencia.
tipo_division_territorial	Clasificación de Departamentos, Municipios y Comunidades para el geoposicionamiento.
division_territorial	Contiene todos los registros de la division territorial política de Nicaragua
establecimiento	Agrupa todos los establecimientos y/o puntos de referencia de Nicaragua
ruta	Registro de todas las rutas urbanas colectiva, interdepartamental u otra que se requiera.
establec_ruta	Se configuran las rutas que tienen relación geoposicional con los establecimientos.
ruta_punto	Agrupa todos los puntos gereferenciados de las rutas
ruta_transiciones	Para almacenar las entradas/salidas por los respectivos establecimientos (por día y hora) del transporte que siguen las rutas configuradas.
ruta_tiempo	Precios y tiempo estimado de un punto respecto a otro de las rutas.

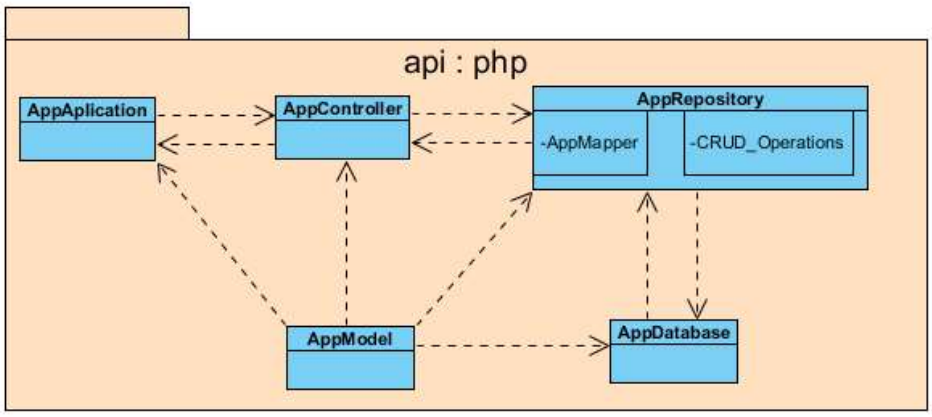
VI. Servicio Web

- Información General de Desarrollo

Servidor Http	**Apache 2.4.38**
Lenguaje	PHP 7.1.27
Distribución MySQL	MariaDB 10.1.38
Arquitectura	MVC – Patrón Controladores y Repositorios
Formato de intercambio de datos	JSON
Métodos permitidos	GET, POST, PUT y DELETE
IDE desarrollo	Visual Code, Visual Paradigm 10

- Detalle de componentes del servicio web

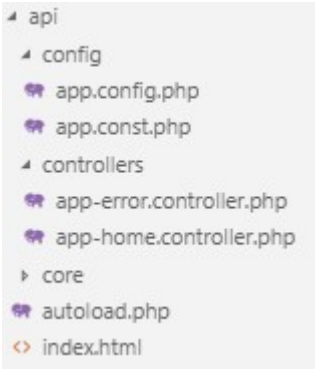
El servicio web está estructurado seguimiento la arquitectura MVC y bajo el patrón de diseño de Controladores y Repositorios, el diagrama general es:



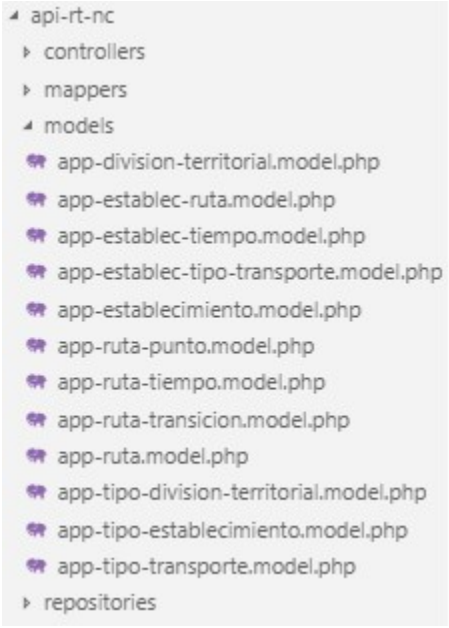
Componente	Descripción
AppAplication	Script principal de ejecución del servicio web, desde cual se hace la carga de los demás componentes según la URL (controlador) consumido por el cliente (navegador, aplicación android, otros.)
AppController	Determina los propiedades básicas de la URL consumida, así como los métodos permitidos GET, POST, PUT y DELETE
AppRepository	Contempla las operaciónes básicas CRUD para cada

	tabla diseñada en la base de datos del servidor.
AppMapper	Sirve para mapear las columnas de las tablas de la base de datos con los modelos (AppModel) del servicio web.
AppDatabase	Manejador y conexión de la base de datos
AppModel	Estructura de modelos utilizadas en el servicio web

Este tipo de arquitectura y patron de diseño ayuda a mantener organizado el código de forma óptima para posterior escalamiento de funcionalidades.

Proyecto api

En el directorio core se encuentran las clases de los componentes del paquete api : php. autoload.php es el script que se importa desde api-rt-nc para el uso de todos los componentes basicos del servicio para su implementación conforme el diseño de tablas de la base de datos. app.config y app.const contienen variables globales de configuración. app-error.controllller y app-home.controller son controladores de uso en común para mostrar errores o url base respectivamente del servicio web.

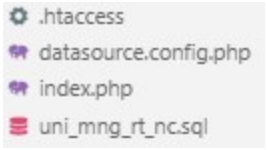
Proyecto api-rt-nc



Contiene la implementación de los componentes de api : php por cada tabla del diseño de la base de datos, seguiendo la siguiente nomenclatura:

- Archivos: app-nombre-tabla(s).[model|mapper|repository|controller].php

- Clases: AppNombreTabla(s)[Model|Mapper|Repository|Controller]
  (s): pluralizado para los componentes Controller y Repository.



.htaccess: para la transformación dinámica de url amigables.
index.php: script inicial de la aplicación.
datasource.config.php: contiene la configuración de conexión a la base de datos.
uni_mng_rt_nc.sql: toda la definición de tablas de la base de datos.

- Detalle de métodos permitidos y estructura de cuerpo solicitudes (http request) y respuestas (http response).

Método	Descripción	Cuerpo Solicitud	Cuerpo Respuesta
GET	Obtener	No aplica	[{REGISTROS OBTENIDOS }]
POST	Registrar	{ data: [NUEVO REGISTRO] }	[{ID INSERTADO }]
PUT	Actualizar	{ data: [REGISTRO A ACTUALIZAR] }	[{ID ACTUALIZADO }]
DELETE	Eliminar	{ data: [REGISTRO A ELIMINAR] }	[{ID ELIMINADO }]

En el caso que se genere un error, el servicio web genera una estructura de respuesta con el detalle del error generado similar al siguiente:

{ "result": 1, "error": { "id": -2, "message": "No existe registro con id 9" } }	result 1: error generado, 0: ejecución exitosa; error.id: id del error (-1 general, -2 base de datos); error.message: detalle del error generado

VI.1. Controlador ruta

```
<?php

class AppRutasController extends AppController {

  public function getCalcular($id, $sid, $qry) {
    header('Content-Type: application/json; charset=utf-8');
    echo json_encode($this->repository->calcular($qry));
  }

  public function getDepartamento($id, $sid, $qry) {
    header('Content-Type: application/json; charset=utf-8');
    echo json_encode($this->repository->selectDepartamento($id, $sid, $qry));
  }
```

```
  public function getTerminal($id, $sid, $qry) {
    header('Content-Type: application/json; charset=utf-8');
    echo json_encode($this->repository->selectTerminal($id, $sid, $qry));
  }

  public function getRuta($id, $sid, $qry) {
    header('Content-Type: application/json; charset=utf-8');
    echo json_encode($this->repository->selectRuta($id, $sid, $qry));
  }

}
```

VI.2. Mapeador de ruta

```
<?php

class AppRutaMapper extends AppMapper {
  public function mapRow($rs, $rn) {
    return $rs;
  }
  public function mapRowRef(&$rs, $rn) {
  }
}
```

Modelo ruta

```
<?php

class AppRutaModel extends AppModel {
  public $id;
  public $nombre;
  public $tipo_transporte_id;
  public $estado;
  public $creado;
  public $modificado;
}
```

VI.3. Repositorio ruta

```
<?php

class AppRutasRepository extends AppRepository {
  public function __construct($db) {
    parent::__construct($db, 'Ruta');
  }

  public function calcular($qry) {
    $sql = "SELECT DISTINCT tbl.*
    FROM ( SELECT e.*, rp.ruta_id as rutas,
get_distance_in_miles_between_geo_locations(e.latitud, e.longitud, :ltt, :lng) *
1609.34 as distancia
          FROM establecimiento as e
          INNER JOIN ruta_punto as rp on rp.establecimiento_id = e.id
        ) as tbl ".
    " WHERE tbl.distancia < 1000.0 ORDER BY tbl.distancia ASC";

    $query = $this->dbc->database->prepare($sql);
    $query->execute(array( ':ltt' => $qry['lttOrg'], ':lng' => $qry['lngOrg'] ));
    $establcOrg = $query->fetchAll(PDO::FETCH_ASSOC); unset($query);

    $query = $this->dbc->database->prepare($sql);
    $query->execute(array( ':ltt' => $qry['lttDst'], ':lng' => $qry['lngDst'] ));
    $establcDst = $query->fetchAll(PDO::FETCH_ASSOC); unset($query);

    $rst = $this->combinarDstOrg($establcOrg, $establcDst, $qry);
    if ($rst == null || count($rst) < 1) {
      return $this->calcularTop2($qry);
    }
    return $rst;
  }

  public function calcularTop2($qry) {
    $sql = "SELECT DISTINCT tbl.*
    FROM ( SELECT e.*, rp.ruta_id as rutas,
get_distance_in_miles_between_geo_locations(e.latitud, e.longitud, :ltt, :lng) *
1609.34 as distancia
          FROM establecimiento as e
          INNER JOIN ruta_punto as rp on rp.establecimiento_id = e.id
        ) as tbl ".
```

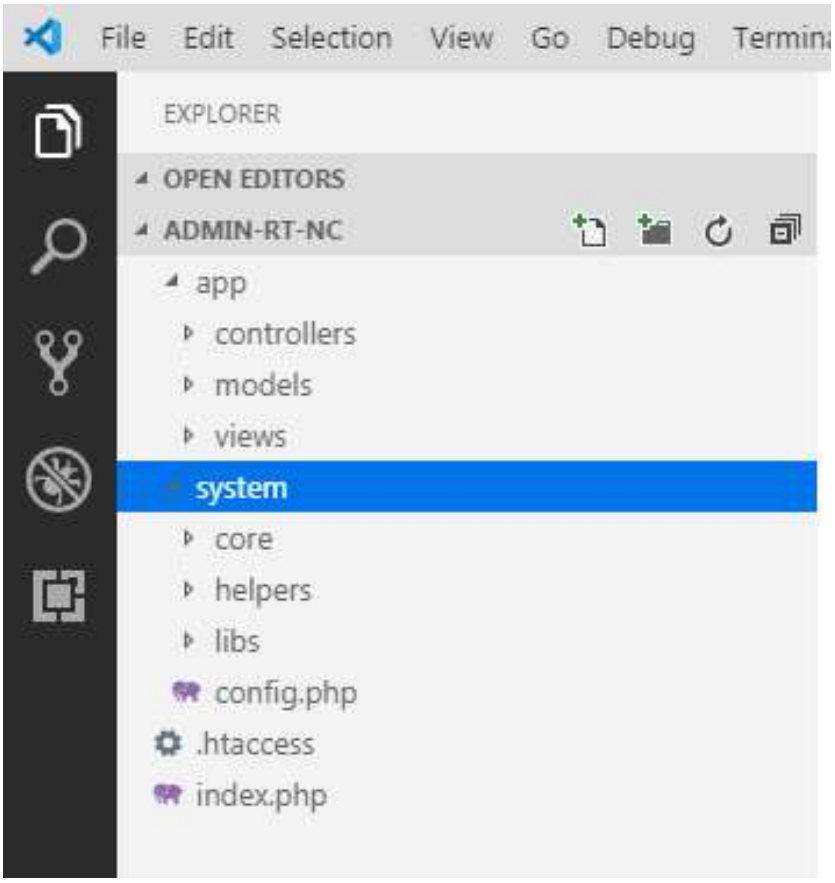
```
    " ORDER BY tbl.distancia ASC";

    $query = $this->dbc->database->prepare($sql);
    $query->execute(array( ':ltt' => $qry['lttOrg'], ':lng' => $qry['lngOrg'] ));
    $establcOrg = $query->fetchAll(PDO::FETCH_ASSOC); unset($query);

    $query = $this->dbc->database->prepare($sql);
    $query->execute(array( ':ltt' => $qry['lttDst'], ':lng' => $qry['lngDst'] ));
    $establcDst = $query->fetchAll(PDO::FETCH_ASSOC); unset($query);

    return $this->combinarDstOrg($establcOrg, $establcDst, $qry, true);
  }

  private function combinarDstOrg($org, $dst, $qry, $top2 = false) {
    $rst = array();
    foreach ($org as $v1) {
      foreach ($dst as $v2) {
        if ($v1['rutas'] == $v2['rutas']) {
          if (!isset($rst[$v2['rutas']])) {
            $item = new stdClass();
            $item->modoViaje = $top2 ? "vehiculo" : "caminando";
            $item->establecOrg = $v1;
            $item->establecDst = $v2;
            $item->lttOrgRef = $qry['lttOrg'];
            $item->lngOrgRef = $qry['lngOrg'];
            $item->distanciaOrgMts = number_format((float)$v1['distancia'], 2, '.', '') .
" metros";
            $item->lttDstRef = $qry['lttDst'];
            $item->lngDstRef = $qry['lngDst'];
            $item->distanciaDstMts = number_format((float)$v2['distancia'], 2, '.', '') . "
metros";
            $item->distanciaPieTotal = ((float)$v1['distancia']) +
((float)$v2['distancia']);
            $rst[$v2['rutas']] = $item;
          } else {
            continue;
          }
          $item = $rst[$v2['rutas']];
          $sql = "SELECT a.* FROM ruta as a WHERE a.id IN (".$v2['rutas'].")";
          $query = $this->dbc->database->prepare($sql); $query->execute();
          $item->ruta = $query->fetch(PDO::FETCH_ASSOC); unset($query);
        }
      }
    }
```

```
    if ($top2) {
      usort($rst, array($this, "cmpRutas"));
      return array_slice($rst, 0, 2);
    }
    return array_values($rst);
  }

  private function cmpRutas($a, $b) {
    return $a->distanciaPieTotal < $b->distanciaPieTotal ? -1 : 1;
  }

  //cargar todas las rutas
  public function select($id, $sid, $qry) {
    $lid = ($id == null || $id < 1 ? null : $id);
    $sql = "SELECT * FROM " . $this->table_name . " WHERE (:id IS NULL OR id
= :id) AND (tipo_transporte_id = 4) and estado=1 ";
    $query = $this->dbc->database->prepare($sql);
    $query->execute(array(':id' => $lid));
    $data = $query->fetchAll(PDO::FETCH_ASSOC);
    if ($lid !== null && count($data) < 1) { throw new Exception('No existe registro
con id ' . $lid); }
    $rst = array();
    foreach ($data as $clave => $valor) {
      $valor['creado'] = new AppDateTimeUTC($valor['creado']);
      $valor['modificado'] = new AppDateTimeUTC($valor['modificado']);
      $lmdl = new $this->model_name();
      $lmdl->loadFromArray($valor);
      $this->mapper->mapRowRef($lmdl, 0);
      $rst[$clave] = $lmdl;
    }
    return $rst;
  }

  public function selectDepartamento($id, $sid, $qry) {
    $sql = "SELECT a.* "
        . "FROM ruta as a "
        . "inner join establec_ruta as b on b.ruta_id = a.id "
        . "inner join establecimiento as c on c.id = b.establecimiento_id "
        . "inner join division_territorial as d on d.id = c.territorio_id "
        . "left join division_territorial as e on e.id = d.sup_id "
        . "WHERE a.tipo_transporte_id = 2 AND b.tipo_relacion = 'T' AND "
        . "(   (d.id = :dpto_id and d.tipo_id = 1)"
         . "OR (d.sup_id = :dpto_id and d.tipo_id = 2)"
         . "OR (e.sup_id = :dpto_id)"
```

```
        . ") ";
    $query = $this->dbc->database->prepare($sql);
    $query->execute(array(':dpto_id' => $sid));
    $data = $query->fetchAll(PDO::FETCH_ASSOC);
    $rst = array();
    foreach ($data as $clave => $valor) {
      $valor['creado'] = new AppDateTimeUTC($valor['creado']);
      $valor['modificado'] = new AppDateTimeUTC($valor['modificado']);
      $lmdl = new $this->model_name();
      $lmdl->loadFromArray($valor);
      $this->mapper->mapRowRef($lmdl, 0);
      $rst[$clave] = $lmdl;
    }
    return $rst;
  }

  //Rutas por terminal
  public function selectTerminal($id, $sid, $qry) {
    $sql = "SELECT a.* "
        . "FROM ruta as a "
        . "INNER JOIN establec_ruta as b on b.ruta_id = a.id "
        . "WHERE b.establecimiento_id = :terminal_id AND b.tipo_relacion = 'T';";
    $query = $this->dbc->database->prepare($sql);
    $query->execute(array(':terminal_id' => $sid));
    $data = $query->fetchAll(PDO::FETCH_ASSOC);
    $rst = array();
    foreach ($data as $clave => $valor) {
      $valor['creado'] = new AppDateTimeUTC($valor['creado']);
      $valor['modificado'] = new AppDateTimeUTC($valor['modificado']);
      $lmdl = new $this->model_name();
      $lmdl->loadFromArray($valor);
      $this->mapper->mapRowRef($lmdl, 0);
      $rst[$clave] = $lmdl;
    }
    return $rst;
  }

  //Select para todas la rutas
  public function selectRuta($id, $sid, $qry) {
    $lid = ($id == null || $id < 1 ? null : $id);
    $sql = "SELECT * FROM " . $this->table_name . " WHERE (:id IS NULL OR id
= :id)";
    $query = $this->dbc->database->prepare($sql);
    $query->execute(array(':id' => $lid));
```

```
    $data = $query->fetchAll(PDO::FETCH_ASSOC);
    if ($lid !== null && count($data) < 1) { throw new Exception('No existe registro
con id ' . $lid); }
    $rst = array();
    foreach ($data as $clave => $valor) {
      $valor['creado'] = new AppDateTimeUTC($valor['creado']);
      $valor['modificado'] = new AppDateTimeUTC($valor['modificado']);
      $lmdl = new $this->model_name();
      $lmdl->loadFromArray($valor);
      $this->mapper->mapRowRef($lmdl, 0);
      $rst[$clave] = $lmdl;
    }
    return $rst;
  }

}
```

VII. Sitio administrativo

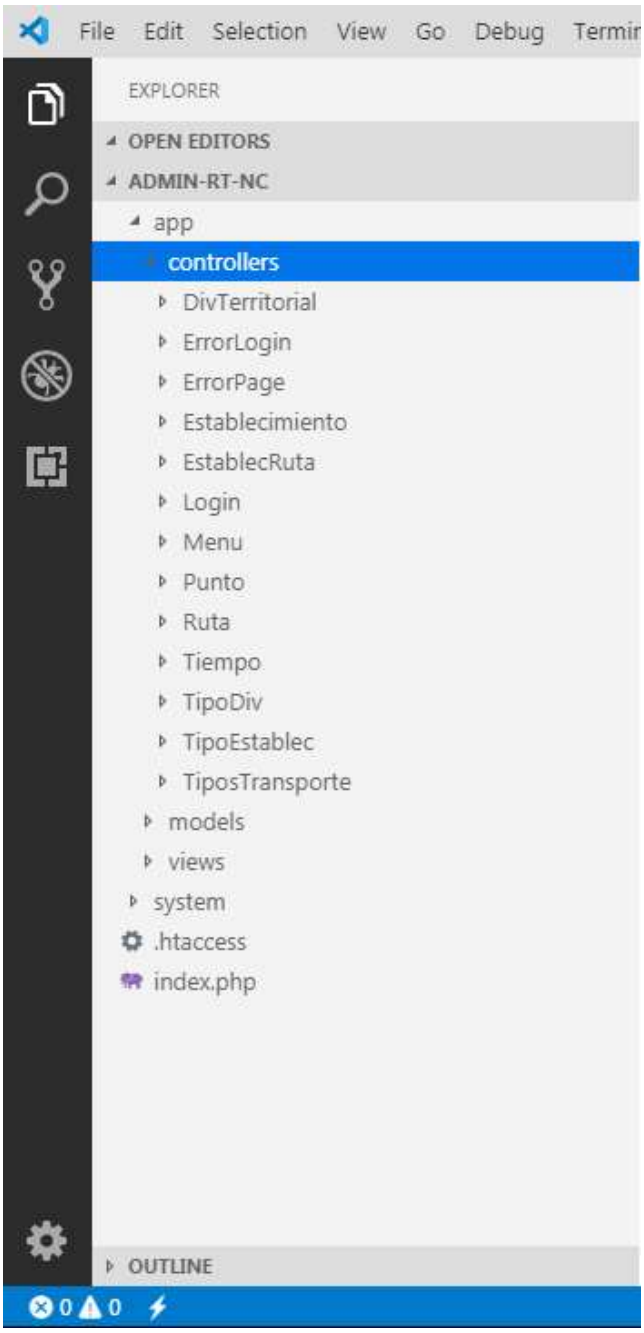


El proyecto consta de dos partes

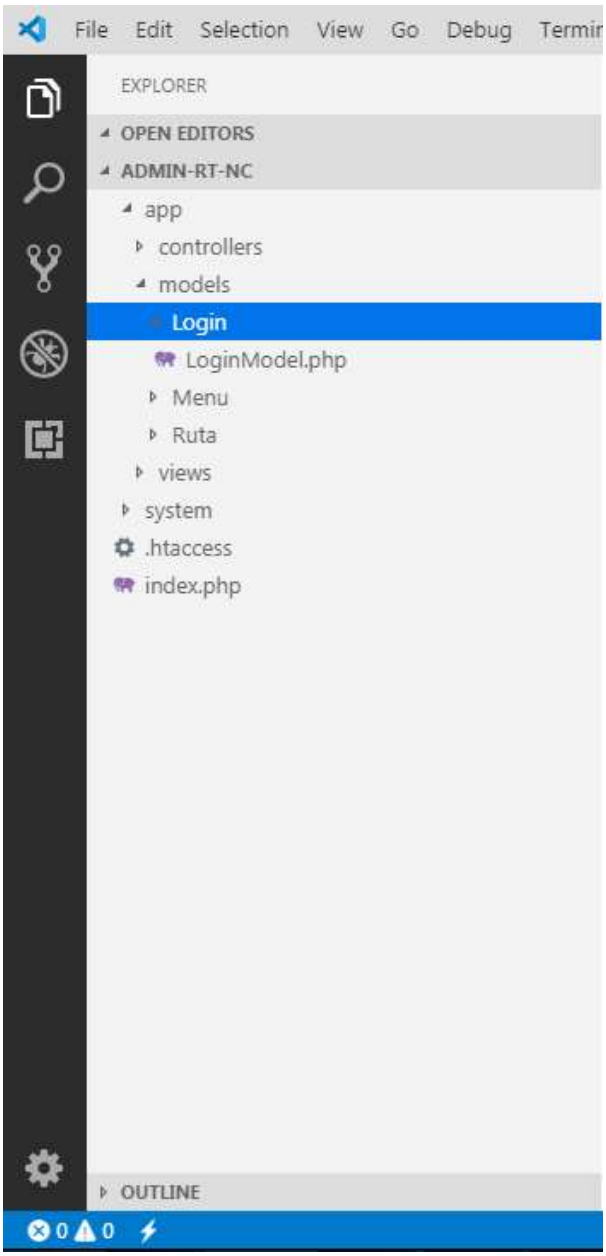
app: nos alberga la capa controllers, models y views esta es la parte donde se agregan nuevas funcionalidades a nuestro sitio.

system:

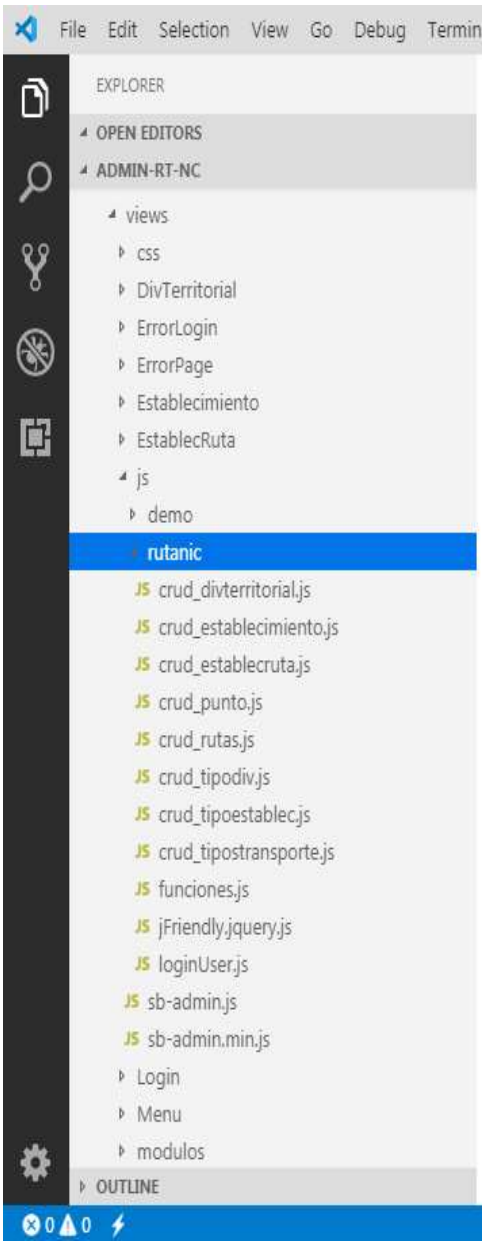
Contiene el nucleo del proyecto en este esta las clases del autoload, controladores generales para que sean de uso global por la app



Controladores del sitio hace referencia a cada pantalla del mismo de igual manera que cada una de ellas es una tabla de la base de datos



El modelo login nos hace referencia a la tabla usuarios.



Aquí almacenamos todo el diseño y programación javascript de la página adminsitrativa. Se hizo uso de la plantilla bootstrap mencionada en el link https://startbootstrap.com/templates/sb-admin/

Se consume el servicio a partir de peticiones Ajax.

VII.1. MenuController:

```
<?php
defined('BASEPATH') or exit('No se permite acceso directo');
require_once ROOT . FOLDER_PATH . '/app/models/Login/LoginModel.php';
require_once LIBS_ROUTE . 'Session.php';

class MenuController extends Controller
{
    private $session;

    public function __construct()
    {
        $this->session = new Session();
        $this->session->init();
        if($this->session->getStatus() === 1 || empty($this->session->get('usuario')))
            header('location: /admin-rt-nc/ErrorLogin/ErrorLogin');
    }

    public function exec()
    {
        $params = array('usuario' => $this->session->get('usuario'));
        $this->render(__CLASS__, $params);
    }

    public function logout()
    {
        $this->session->close();
        header('location: /admin-rt-nc/login/signin');
    }

}
```

VII.2. Programacion de la pantalla rutas:

```
$(document).ready(function() {
    $("#Alert1").hide();
    $("#Alert2").hide();
    var table = $('#dataTableruta').DataTable({
        "destroy": true,
        "ajax": {
            "url": "http://www.rutnicapp.com/api-rt-nc/rutas/0/ruta/",
            "dataSrc": "",
            headers: {
```

```
            "Authorization": "Basic " +
btoa("admin:Hjh/Y/(SDYuh(/DSTfdgs6fsd5f6sdhbdc6%RS%&ADSA")
          }
       },
       "columns": [
          { "data": "id" },
          { "data": "nombre" },
          { "data": "tipo_transporte_id" },
          { "data": "estado" },
       ],
       rowId: 'id',
       "columnDefs": [{
            "targets": 4,
            "data": null,
            "defaultContent": "<button title='Editar' type='button' id='btnEruta'
class='btn btn-warning fas fa-edit' data-toggle='modal' data-
target='#upModalruta'></button>"
          },
          {
            "targets": 5,
            "data": null,
            "defaultContent": "<button title='Eliminar' type='button' id='btndruta'
class='btn btn-danger fas fa-trash-alt' data-toggle='modal' data-
target='#delmodalruta'></button>"
          },
          {
            "targets": 0,
            "visible": false,
            "searchable": false
          }
       ],
       responsive: true
    });

    //**Boton para abrir formulario nuevo */
    $("#addruta").click(function() {
       loadTipoTrans(1);
    });

    //funcion para guardar nuevo
    $("#btnsaveruta").click(function() {

       //obtener datos
       var ruta = $('#inputruta').val();
```

```
        var id_tipo = $('#selTipoTrans').val();

        if (ruta.length <= 0 || id_tipo == "Seleccione el tipo de transporte") {
            $("#Alert1").show("slow");
        } else {


            var obj = {
                "data": [{
                    "nombre": ruta,
                    "tipo_transporte_id": id_tipo,
                    "estado": "1"
                }]
            };
            var data = obj;

            $.ajax({

                url: 'http://www.rutnicapp.com/api-rt-nc/rutas/',
                headers: {
                    "Authorization": "Basic " +
btoa("admin:Hjh/Y/(SDYuh(/DSTfdgs6fsd5f6sdhbdc6%RS%&ADSA")
                },
                type: 'POST',
                data: JSON.stringify(data),
                contentType: 'application/json',
                processData: false,
                dataType: 'json',
                cache: false,
                timeout: 5000,
                success: function(result) {
                    table.ajax.reload();
                    table.draw();
                    window.location.reload();
                    $('#inputruta').val('');
                    $('#addModalruta').modal('hide');
                },
                error: function(xhr, resp, text) {
                    // show error to console
                    var r = jQuery.parseJSON(resp.responseText);
                    alert("Message: " + r.Message);
                    alert("StackTrace: " + r.StackTrace);
                    alert("ExceptionType: " + r.ExceptionType);
```

```
            }

        }).fail(function($xhr) {
            var data = $xhr.responseJSON;
            alert(data);
        });
    }
});

//Actualizar
$("#btnsaveupd").click(function() {

    //obtener datos
    var ruta = $('#inputrutaupd').val();
    var id_tipo = $('#selTipoTransupd').val();
    var idruta = $('#updruta').val();
    var estado = $('#estado').val();
    // console.log(idruta)
    // var estado=$()
    var obj = {
        "data": [{
            "id": idruta,
            "nombre": ruta,
            "tipo_transporte_id": id_tipo,
            "estado": estado
        }]
    };
    var data = obj;

    $.ajax({

        url: 'http://www.rutnicapp.com/api-rt-nc/rutas/' + idruta,
        headers: {
            "Authorization": "Basic " +
btoa("admin:Hjh/Y/(SDYuh(/DSTfdgs6fsd5f6sdhbdc6%RS%&ADSA")
        },
        type: 'PUT',
        data: JSON.stringify(data),
        contentType: 'application/json',
        processData: false,
        dataType: 'json',
        cache: false,
        timeout: 5000,
        success: function(result) {
```

```
            $('#inputrutaupd').val('');
            $('#upModalruta').modal('hide');
            table.ajax.reload();
            table.draw();
            location.reload();
        },
        error: function(xhr, resp, text) {
            // show error to console
            var r = jQuery.parseJSON(resp.responseText);
            console.log("Message: " + r.Message);
            console.log("StackTrace: " + r.StackTrace);
            console.log("ExceptionType: " + r.ExceptionType);

        }

    }).fail(function($xhr) {
        var data = $xhr.responseJSON;
        console.log(data);
    });

});

$("#btndel").click(function() {

    //obtener datos
    var idruta = $('#delruta').val();
    var obj = {
        "data": [
            { "id": idruta }
        ]
    };
    var data = obj;
    console.log(data);

    $.ajax({

        url: 'http://www.rutnicapp.com/api-rt-nc/rutas/' + idruta,
        headers: {
            "Authorization": "Basic " +
btoa("admin:Hjh/Y/(SDYuh(/DSTfdgs6fsd5f6sdhbdc6%RS%&ADSA")
        },
        type: 'DELETE',
        data: JSON.stringify(data),
        contentType: 'application/json',
```

```
        processData: false,
        dataType: 'json',
        cache: false,
        timeout: 5000,
        success: function(result) {
            if (result.result == 1) {
                $("#Alert2").show("slow");
            } else {
                table.ajax.reload();
                table.draw();
                window.location.reload();
                $('#delruta').val('');
                $('#delmodalruta').modal('hide');
            }
        },
        error: function(xhr, resp, text) {
            // show error to console
            var r = jQuery.parseJSON(resp.responseText);
            alert("Message: " + r.Message);
            alert("StackTrace: " + r.StackTrace);
            alert("ExceptionType: " + r.ExceptionType);

        }

    }).fail(function($xhr) {
        var data = $xhr.responseJSON;
        alert(data);
    });

});

//evento  al cerrar el modal de registro
$("#addModalruta").on("hidden.bs.modal", function() {
    $('#inputruta').val('');
    $("#Alert1").hide();
});

//evento  al cerrar el modal de registro
$("#delmodalruta").on("hidden.bs.modal", function() {
    $('#Alert2').hide();
}); //Obtener registros por cada fila del datatable
$('body').on('click', '#btnEruta', function() {
    var row = $(this).parents('tr')[0];
    var estado;
```

```
        var tipotransporte;
        var idruta;
        var ruta;
        idruta = table.row(row).data().id;
        ruta = table.row(row).data().nombre;
        estado = table.row(row).data().estado;
        tipotransporte = table.row(row).data().tipo_transporte_id;
        loadTipoTrans(2, tipotransporte);
        $('#updruta').val(idruta);
        $('#inputrutaupd').val(ruta);
        if (estado == 1) {
            $('#rdo_pick').click();
        } else {
            $('#rdo_drop').click();
        }

    });

    //Obtener registros por cada fila del datatable
    $('body').on('click', '#btndruta', function() {
        var row = $(this).parents('tr')[0];
        var idruta;
        idruta = table.row(row).data().id;
        $('#delruta').val(idruta);

    });

    $('input:radio[name="optradio"]').change(function() {
        if (this.checked && this.value == 'Activo') {
            $('#estado').val(1);
        } else {
            $('#estado').val(0);
        }
    });

    $("#inputruta").focus(function() {
        $("#Alert1").hide();
    });

    $("#selTipoTrans").focus(function() {
        $("#Alert1").hide();
    });
});
```

La estructura de programación es similar en casi todos los modulos.