

UNIVERSIDAD POLITÉCNICA DE VICTORIA

Aplicación de Android para Taxistas Profesionales Construida en Flutter de Meta

REPORTE DE ESTANCIA I

Que presenta

Jorge Luis Charles Torres

Alumno de la carrera de

Ingeniería en Tecnologías de la Información

Asesor institucional

M.S.I. José Fidencio López Luna

Asesor empresarial

Dr. Rubén Machucho Cadena

Organismo receptor

Universidad Politécnica de Victoria

Cd. Victoria, Tamaulipas, mayo de 2019

RESUMEN

Para la materia de estancia I, se emprendió la creación de una aplicación móvil para dispositivos Android, destinada a los taxistas profesionales de Ciudad Victoria, Tamaulipas. Específicamente, se desarrollaron los formularios para que las flotillas de taxis o taxis independientes puedan registrar sus servicios, así como también el formulario para que los pasajeros puedan solicitar un taxi.

Todo lo mencionado anteriormente, se diseñó con ayuda del framework Flutter que emplea Dart como lenguaje de programación. Además, para el manejo de la información ingresada por los usuarios, se implementó una base de datos MySQL montada en un servidor local el cual se enlaza con la aplicación.

El resultado de este proyecto es una aplicación que cuenta con una interfaz gráfica muy amigable con cualquier usuario, lo cual significa que es fácil entender cada una de las funciones que la compone.

ABSTRACT

For the subject of estancia I (stay I), was run the creation of a mobile application for Android devices that was designed for professional taxi drivers in Ciudad Victoria, Tamaulipas. Specifically, the forms were developed so that taxi fleets or independent taxis can register their services, as well as the form so that passengers can request a taxi.

Everything mentioned above was designed with the help of the Flutter framework that uses Dart as a programming language. In addition, to manage the information entered by users, a MySQL database mounted on a local server was implemented, which is linked to the application.

The result of this project is an application that has a graphical interface that is very friendly to any user, which means that it is easy to understand each of the functions that make it up.

Palabras claves: Taxistas, transporte, flotillas, sitios, pasajeros, software, aplicación, servicios, móvil, formularios.

ÍNDICE TEMÁTICO

I.	INTRODUCCIÓN	4
II.	MARCO TEÓRICO	5
II.1	ANTECEDENTE SIMILAR DENOMINADO “Mi Taxi”	6
II.2	ANTECEDENTE SIMILAR DENOMINADO “Taxibeat”	7
III.	JUSTIFICACIÓN	8
IV.	OBJETIVOS	8
V.	DESARROLLO DEL PROYECTO	9
V.1	FORMULARIOS PARA EL REGISTRO DE SERVICIOS DE LOS TAXISTAS	10
V.2	FORMULARIO PARA LA SOLICITUD DE UN TAXI	23
VI.	RESULTADOS	25
VII.	CONCLUSIONES	26
	BIBLIOGRAFÍA	27
	ANEXOS	28

I. INTRODUCCIÓN

Con el desarrollo de este proyecto nombrado “Aplicación de Android para Taxistas Profesionales Construida en Flutter de Meta”, se pretende generar más oportunidades de trabajo (viajes) para todos los taxistas profesionales que se registren en la aplicación y que radiquen en Ciudad Victoria, Tamaulipas. Así como también, por el lado de los usuarios, se pretende que retomen la confianza en los taxistas profesionales de nuestra ciudad al poder contar con la información de contacto sobre el conductor que se les sea asignado.

Esto último va de la mano con la problemática que se desea resolver, ya que a la fecha que se realiza este reporte, los taxistas de Ciudad Victoria presentan una remarcada disminución en la carga de trabajo debido a la llegada de plataformas como DiDi o Uber, las cuales generan más “seguridad” al usuario debido a que conoce a su conductor. Y lo anterior, combinado con la inseguridad que se vive en la actualidad, generó un mayor decremento en la carga de trabajo de los taxistas profesionales.

Para abordar esta investigación, se emplearon varias técnicas a lo largo de la elaboración del proyecto, de las cuales las que más resaltan son la observación hacia el flujo de trabajo de los taxistas profesionales y su comportamiento hacia los conductores de diferentes aplicaciones y entrevistas dirigidas a dichos taxistas para conocer el porqué de la situación.

Los resultados de la investigación se presentan en este reporte con el cual se da inicio al desarrollo del proyecto “Aplicación de Android para Taxistas Profesionales Construida en Flutter de Meta” formulado por la empresa Universidad Politécnica de Victoria (UPV). Esta empresa es una institución destinada a la enseñanza que esta constituida por diferentes facultades, las cuales tienen como objetivo formar investigadores altamente capacitados, con reconocida capacidad científico-tecnológica, que participen en la solución de problemas de diferentes áreas que contribuyan al progreso de México con un sentido humanista y de desarrollo sustentable.

II. MARCO TEÓRICO

Como se mencionó anteriormente, en la actualidad existe un conflicto en Ciudad Victoria entre los taxistas profesionales y los conductores de aplicaciones como DiDi o Uber, esto se debe a diferentes factores de los cuales el que más resalta es que los taxistas exigen que los conductores de las diversas plataformas también paguen impuestos tal como lo indica la ley de transporte [1].

También otro factor que destaca, el cual es la razón principal por el que se desarrolla este proyecto, es que los taxistas profesionales (en su mayoría los que se ubican en la Central Camionera de Ciudad Victoria) presentan un decremento en su carga de trabajo debido al arribo de conductores de diferentes plataformas.

El delegado del sitio Liborio Tovar indicó que por cada siete vehículos de alguna plataforma que acuden a recoger o dejar a un pasajero a la Central de Autobuses, solamente un taxista obtiene pasaje (Figura 1).

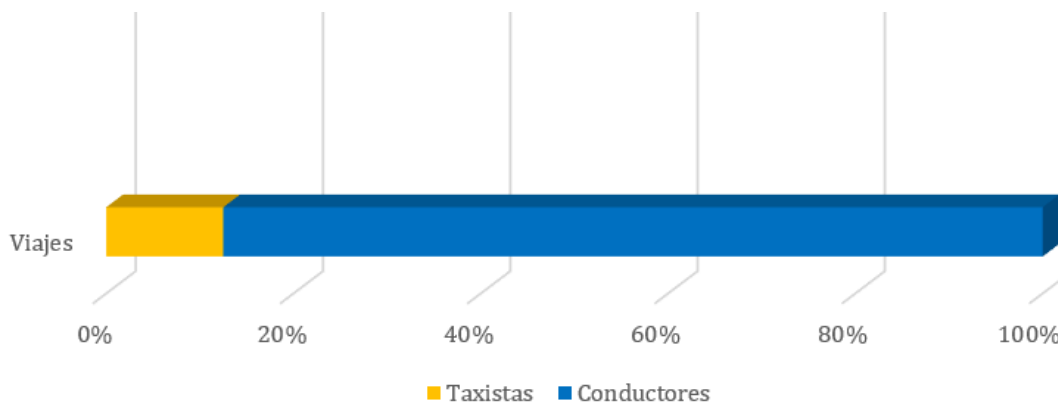


Figura 1. Gráfica de pasajes

Se investigó la razón que genera esta situación y se encontró un estudio realizado por la Asociación de Internet.MX, el cual menciona que el 57% de los encuestados aseguran que prefieren este tipo de aplicaciones por seguridad y es que es muy conocido que los taxis no lo son del todo al no existir la certeza de quién es la persona que conduce, algo que sí pasa en las aplicaciones.

Otro criterio que también influye en la elección de los encuestados es la conveniencia de no tener que discutir con el taxista por el “no voy para allá”, “no me da tiempo”, “le cobro doble porque está lejos”. Estos y más aspectos se pueden encontrar expuestos en [2].

II.1 ANTECEDENTE SIMILAR DENOMINADO “Mi Taxi”

Durante la búsqueda de información se encontró una aplicación similar a la desarrollada en este proyecto, esta es llamada “App CDMX” la cual cuenta con diferentes módulos, pero el que importa en este caso es el apartado “Mi Taxi” (Figura 2) lanzado en septiembre de 2019, el cual funciona como herramienta digital para pedir desde cualquier punto de la Ciudad de México un servicio de transporte.

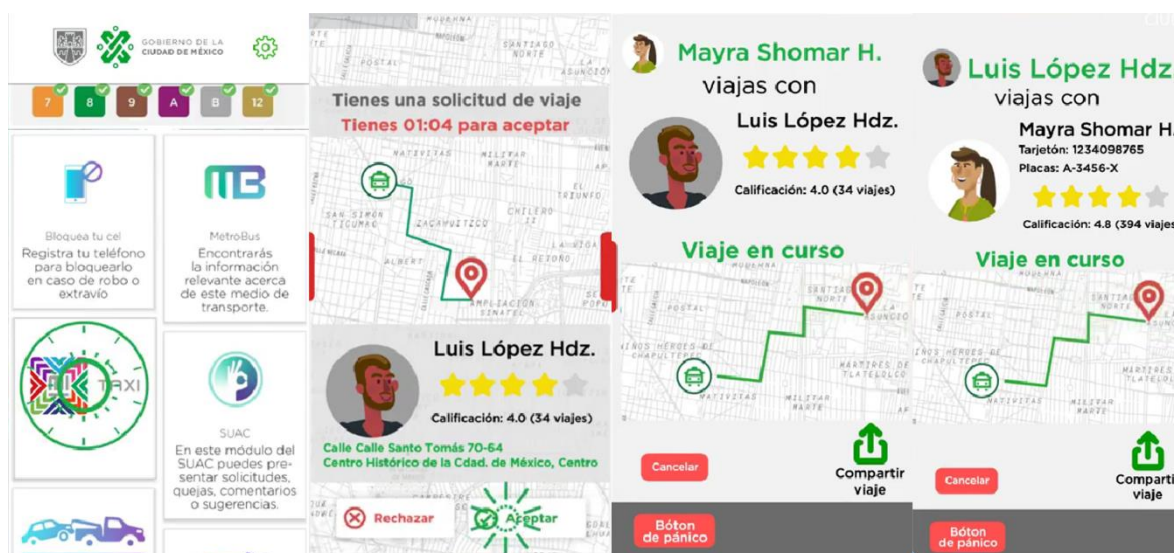


Figura 2. Pantallas del módulo “Mi Taxi”

Al igual que este proyecto, la herramienta “Mi Taxi” fue creada bajo la premisa de que con ella los viajes en taxi serían más seguros para los pasajeros puesto que se puede conocer al conductor que se les sea asignado. Además, pretende darle todo el potencial que se requiere a este método de transporte y que sea realmente una aplicación que compita en las mejores condiciones para fortalecer más este modo de transporte.

También, así como el proyecto, este módulo se desarrolló debido al incremento de la inseguridad en el transporte público puesto que, durante la primera mitad de 2019, se registraron 268 carpetas de investigación de la Procuraduría General de Justicia (PGJ) por el delito de “robo a pasajero a bordo de taxi con violencia”, de acuerdo con Datos Abiertos de la Ciudad de México. Un incremento significativo comparado con 2018 (212) y 2017 (160).

II.2 ANTECEDENTE SIMILAR DENOMINADO “Taxibeat”

Igual que como se comentó en la sección precedente, durante la búsqueda de información se encontró otra aplicación con el mismo objetivo por el cual se desarrolla este proyecto, esta es llamada “Taxibeat”. A diferencia de la anterior, esta tiene cobertura en más países y no solo está enfocada en México, aunque es importante mencionar que en nuestro país solo se encuentra disponible para la Ciudad de México.

“Taxibeat” es otra alternativa para solicitar taxi desde un celular (Figura 3). Se debe descargar la app y registrarse antes de solicitar el servicio. Desde la aplicación se puede ver el modelo y año del auto, el tipo de unidad (si es taxi de sitio o radio), los servicios que ofrece el taxista como por ejemplo hablar otros idiomas, y ver las reseñas de otros usuarios.

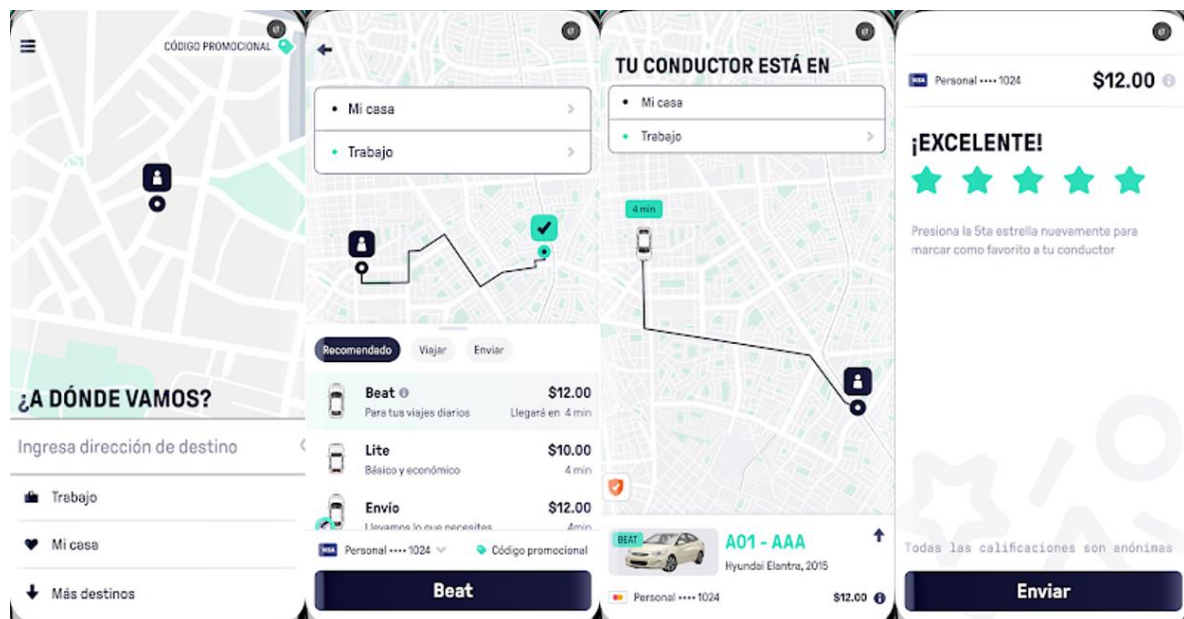


Figura 3. Pantallas de la app “Taxibeat”

Esta aplicación pretende eliminar muchos de los principales problemas que se pueden presentar al tomar un taxi, el principal de ellos, la seguridad (como se ha mencionado a lo largo de este documento). Con esta app se puede saber exactamente quién está conduciendo y así sentirse un poco más cómodo.

Una vez expuesto el contexto de la situación actual y realizada la comparativa de estas dos aplicaciones que se encuentran vigentes en el mercado, se puede identificar que existe cierta similitud entre las necesidades de todos los usuarios y también las opciones que ofrecen este tipo de apps, incluso comparten características con aplicaciones de transporte privado como DiDi o Uber. Con esto identificado, se pudo generar una base (idea) para empezar con el desarrollo de este proyecto.

III. JUSTIFICACIÓN

Como se ha estado mencionado a lo largo de este documento, en la actualidad existe una “guerra” entre los taxistas profesionales de Ciudad Victoria, Tamaulipas y los conductores de plataformas como DiDi o Uber, debido a diferentes situaciones, pero la que más sobresale es que el pasaje (viajes) de los taxistas a disminuido considerablemente por la llegada de las plataformas antes mencionadas.

Los pasajeros prefieren el transporte privado que otorga este tipo de plataformas en lugar de los taxis convencionales, ya que se sienten más seguros al poder conocer quien es el conductor que los esta llevando a su destino y también cual es la unidad correcta que se les ha asignado.

Debido a lo mencionado anteriormente, es que se decidió generar este apoyo hacia los taxistas locales de Ciudad Victoria, Tamaulipas desarrollando el proyecto “Aplicación de Android para Taxistas Profesionales Construida en Flutter de Meta”.

IV. OBJETIVOS

Este proyecto pretende el desarrollo de una aplicación que sirva de ayuda para todos los taxistas profesionales de Ciudad Victoria, Tamaulipas que se registren en ella. En esta primera fase del proyecto se realizarán los formularios correspondientes al registro de los servicios de los taxistas locales de forma independiente (taxi libre) o dependiente (taxi de sitio) según sea el caso. Así como también, se desarrollará el formulario con el que los pasajeros puedan solicitar un taxi.

Los primeros formularios se hacen con el fin de conseguir la información personal del taxista, al igual que la de su unidad, para que pueda ser mostrada a los pasajeros en el momento que se les asigne un taxi y así generarles una mayor seguridad, de esta manera se espera cubrir la principal problemática por la cual disminuyo el trabajo de los taxistas y puedan recuperar poco a poco la carga de trabajo que tenían antes de que llegaran las plataformas de transporte privado.

También se espera que todo esto pueda traer beneficios para los pasajeros, puesto que, al existir más competencia justa, cada plataforma tratará de dar el mejor precio (menor costo) para obtener más clientes.

V. DESARROLLO DEL PROYECTO

Para el desarrollo de este proyecto se necesitaron varios programas los cuales tenían que fusionarse para trabajar en conjunto y así poder crear la aplicación. También, se requirió de un buen equipo de cómputo el cual soportara el uso simultáneo de los softwares para trabajar cómodamente.

Primeramente, se hará referencia al Entorno de Desarrollo Integrado empleado en este trabajo el cual fue Android Studio que es el IDE oficial para el desarrollo de apps para Android desarrollado por Google, por esta razón fue elegido sobre su competencia como lo es Visual Studio Code. Otros aspectos por los que se prefirió usar Android Studio es que ya cuenta con un emulador rápido y cargado de funciones, además ofrece las herramientas y servicios con los que los creadores pueden confeccionar aplicaciones sin tener que recurrir a nada más. Este IDE ayudó a implementar la siguiente herramienta, usada para la creación de todo el proyecto, la cual es Flutter.

Flutter es un SDK (Kit de Desarrollo de Software) igualmente desarrollado por Google para crear aplicaciones móviles tanto para el sistema operativo Android como para iOS. Esta es su principal ventaja ya que genera código 100% nativo para cada plataforma, con lo que el rendimiento y la UX (Experiencia de Usuario) es totalmente idéntico a las aplicaciones nativas tradicionales. Este framework utiliza Dart como lenguaje de programación, también desarrollado por Google, algunos expertos en la materia consideran esto como una desventaja ya que Dart no es un lenguaje muy conocido ni usado por la comunidad de desarrolladores móviles tal como lo son Swift, Objective-C, Kotlin, Java, JavaScript, PHP y Ruby. Aunque en realidad es muy parecido a Java y C# (Microsoft), lo que significa que si se tiene experiencia con algunos de estos lenguajes es muy intuitivo empezar con Dart. Otra de las características que más sobresale de Flutter es que cuenta con Hot Reload, esta funcionalidad permite experimentar rápida y fácilmente cambios en el código del programa como por ejemplo añadir funcionalidades o arreglar bugs. Hot Reload trabaja inyectando ficheros de código fuente actualizados en la Máquina Virtual (VM) Dart en ejecución. Después de que VM actualiza clases con la nueva versión de campos y funciones, el framework Flutter automáticamente reconstruye el árbol de widgets, permitiendo ver rápidamente los efectos de los cambios. Esta funcionalidad fue de gran ayuda durante el desarrollo de la aplicación ya que no se perdió tiempo en ejecutar desde cero el código. En términos generales, la herramienta Flutter se encargó del diseño de la aplicación, así como también de las funciones de los widgets empleados, pero toda la

información ingresada por el usuario debía almacenarse en algún lugar y es aquí donde entran las bases de datos en MySQL.

MySQL es el sistema de gestión de bases de datos relacional más extendido en la actualidad al estar basada en código abierto. Con una base de datos relacional, los datos son fragmentados en varias áreas de almacenamiento separadas (llamadas tablas) en lugar de poner todo junto en una gran unidad de almacenamiento. Pero para lograr que los datos ingresados en la aplicación se vean reflejados en las tablas, se necesito montar la base de datos en un servidor (por el momento local) por lo cual se empleo XAMPP. Esta es una herramienta de desarrollo que permite probar los trabajos de páginas web o programación en el mismo ordenador sin necesidad de tener que acceder a internet.

Todo lo anterior se implementó en un equipo de cómputo con las características necesarias para soportar el uso simultaneo de los softwares, algunas de ellas son:

- Procesador AMD Ryzen 3 3300U
- Gráficos AMD Radeon Vega 6
- Memoria RAM de 12 GB
- Almacenamiento SATA de 1 TB y SSD de 128 GB

Para llevar un orden en las actividades, se creó un plan de trabajo el cual consistía primeramente en recopilar toda la información necesaria para entender el funcionamiento y uso de los softwares que se iban a implementar en la creación de la aplicación, posteriormente, se estipulo desarrollar el formulario correspondiente al registro de servicios de los taxistas independientes, después el formulario correspondiente al registro de servicios de los taxistas dependientes y por último el formulario para solicitar un taxi por parte de los pasajeros.

V.1 FORMULARIOS PARA EL REGISTRO DE SERVICIOS DE LOS TAXISTAS

Esta es la primera fase del desarrollo de la aplicación, la cual se divide en dos partes, el formulario para el registro de servicios de los taxistas independientes (taxi libre) y el formulario para el registro de servicios de los taxistas dependientes (taxi de sitio).

Primeramente, en la pantalla de inicio se creó un menú desplegable en el que aparece la opción "Pertener a TAXI APP" (Figura 4), al pulsarla se manda llamar a la pantalla donde se encuentran las diferentes opciones de registro ("/OpcionesDeRegistro").

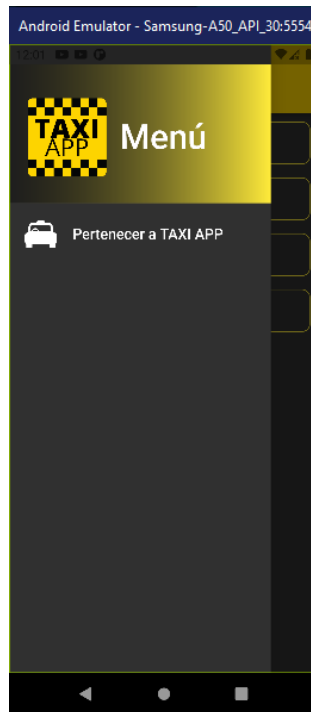


Figura 4. Menú

```
drawer: Drawer( Cajon de Menú
  child: ListView( Lista de opciones
    children: <Widget>[
      DrawerHeader( Encabezado
        decoration: const BoxDecoration(
          gradient: LinearGradient(colors: [Colors.black38,
Colors.yellow])),
      ),
      child: Row( Contenido del encabezado
        children: <Widget>[
          Image.asset("assets/logo.png"),
          const SizedBox( Espacio entre imagen y texto
            width: 10,
          ),
          const Text(
            "Menú",
            style: TextStyle(fontSize: 40, color: Colors.white,)),
        ],
      ),
    ],
  ),
  ListTile( Opción de registro para taxistas
    leading: Image.asset("assets/icono_taxi.png"),
    title: const Text(
      "Pertenece a TAXI APP",
      style: TextStyle(fontSize: 17,)),
```



```

padding: const EdgeInsets.fromLTRB(0, 30, 0, 30), Tamaño
del espacio (Solo se aplicó en Top y Bottom)
child: Column(
  children: const <Widget>[
    Text(
      "Crear Flotilla (Sitio)",
      style: TextStyle(fontSize: 29, fontWeight:
FontWeight.bold,)),
    ),
    Text(
      "Ser líder de una flotilla con taxis propios o de
otras personas",
      style: TextStyle(fontSize: 11,)),
    ),
  ],
),
),
onPressed: () {
Navigator.of(context).pushNamed("/RegistroDependientePrimera"); Manda
llamar a la primera pantalla de Registro Dependiente
},
),

```

En el primer botón se estableció que cuando sea pulsado se mande llamar a la pantalla ("/RegistroDependientePrimera") donde se encuentra el formulario para registrar una flotilla (Figura 6).

Figura 6. Formulario para crear flotilla

Este formulario está compuesto por campos de texto llamados TextFormField, este es el widget que permite validar si el usuario ha ingresado algún dato. También se colocó un botón que al pulsarlo es el que realiza todo el trabajo, primero se valida si todos los campos están llenos, cuando es falso no se realiza nada y se manda un mensaje en el campo que se encuentra vacío y si es verdadero se manda llamar a una función que comprueba la conexión de red mediante un ping a una página de internet.

```
Future<bool> internetConnectivity() async { Función que comprueba la
conexión a internet mediante un ping
  try {
    final result = await InternetAddress.lookup('google.com');
    if (result.isNotEmpty && result[0].rawAddress.isNotEmpty) {
      return true; Conectado
    }
  } on SocketException catch (_) {
    return false; Desconectado
  }
  return false;
}
```

Si la función retorna false significa que no hay conexión a internet y en este caso se estableció que se mande llamar a la pantalla ("/ErrorRegistroDependiente") que notifica al usuario del error (Figura 7).



Figura 7. Notificación de error

Si la función retorna true significa que si hay conexión a internet y en este caso se estableció que se mande llamar a otra función la cual solicita acceso a la base de datos para mandar la información (Esta función requiere de varios archivos php los cuales son los encargados de realizar la conexión entre la aplicación y la base de datos en el servidor local).

```
void adddata_flotilla(){ Función que llama a la base de datos para enviar los datos
    var url="http://192.168.1.69/taxi_app/adddata_flotilla.php";
    http.post(Uri.parse(url), body: { Se envían los datos (ingresados por el usuario) de los controladores a cada atributo de la tabla en la base de datos
        "SITIO": controllerSITIO.text,
        "NOMBRE_LIDER": controllerNOMBRE_LIDER.text,
        "TELEFONO": controllerTELEFONO.text,
        "DOMICILIO": controllerDOMICILIO.text
    });
}
```

Obviamente, para llegar a este punto ya se creó la tabla (Figura 8) correspondiente a este formulario en una base de datos alojada en el servidor local.

taxi_app flotilla	
SITIO	varchar(60)
NOMBRE_LIDER	varchar(60)
TELEFONO	varchar(10)
DOMICILIO	varchar(200)

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/> 1	SITIO 🗝️	varchar(60)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/> 2	NOMBRE_LIDER	varchar(60)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/> 3	TELEFONO	varchar(10)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/> 4	DOMICILIO	varchar(200)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más

Figura 8. Tabla para los datos de las flotillas

Una vez enviada la información a la tabla, se manda llamar a la pantalla ("/ListoRegistroDependiente") que notifica al usuario que todo ha marchado bien y el registro está listo (Figura 9).



Figura 9. Notificación de éxito

Como se mencionó anteriormente, el botón de finalizar es el que realiza todos los procesos y una vez explicado todo lo que lo compone, es importante ver como quedó finalmente estructurado su código.

```

RaisedButton( Botón
  child: const Text("FINALIZAR",
    style: TextStyle(fontSize: 20,)),
  ),
  color: Colors.yellow, Color del botón
  textColor: Colors.black, Color del texto del botón
  onPressed: () async{
    if(formKey.currentState!.validate()){ Se comprueba que los cuadros de
    texto contengan algo
      var ping=await internetConnectivity(); Se manda llamar la función
    que comprueba la conexión a internet para obtener una respuesta
      if(ping==true){ Si hay conexión a internet
        adddata_flotilla(); Se manda llamar la función que llama a la
    base de datos para enviar los datos
        Navigator.of(context).pushReplacementNamed("/ListoRegistroDependien
    te"); Manda a la pantalla de Listo Registro Dependiente con un
    check
      }else{ Si no hay conexión a internet
        Navigator.of(context).pushReplacementNamed("/ErrorRegistroDependien
    te"); Manda a la pantalla de Listo Registro Dependiente con una
    equis
    }
  }
)

```



```

    },
  },
}

```

En el segundo botón de la pantalla de las opciones de registro, se estableció que cuando sea pulsado se mande llamar a la pantalla ("/RegistroDependienteSegunda") donde se encuentra el formulario para registrar taxistas dentro de una flotilla existente (Figura 10).

Figura 10. Formulario para registrar taxistas en una flotilla

Este formulario, al igual que el anterior, está compuesto por campos de texto tipo `TextFormField` y también cuenta con un botón que al pulsarlo es el que realiza todo el trabajo. Lo que lo hace diferente del anterior es que en este se ha agregado una opción (Figura 11) que manda llamar a otra pantalla ("/ListaDeSitios") la cual muestra un listado de todos los sitios que se hayan registrado en el formulario pasado, esto debido a que el taxista que se este registrando debe pertenecer a una flotilla existente y si coloca una que no esta disponible, el registro no se podrá realizar. El ofrecerle esta opción reduce la probabilidad de equivocarse y así se evitan problemas.



Figura 11. Opción para ver la lista de sitios registrados

El código que compone a la pantalla donde se muestra la lista de los sitios registrados (Figura 12) es bastante sencillo, primeramente, se creó una función la cual realiza la petición de los datos ubicados en la tabla de flotillas (sitios) de la base de datos montada en el servidor local (Esta función también requiere de varios archivos php los cuales son los encargados de realizar la conexión entre la aplicación y la base de datos).

```
Future<List> getdata_SITIOS() async{
    final response=await
    http.get(Uri.parse("http://192.168.1.69/taxi_app/getdata_SITIOS.php"))
    );
    return json.decode(response.body);
}
```

Posteriormente, se agregó un widget llamado FutureBuilder que manda llamar la función anterior para obtener un arreglo con los sitios registrados. Si se demora la consulta se mostrará un círculo de carga, cuando ya se obtenga el arreglo se pasa como parámetro a otro widget llamado ListView el cual es el constructor de la lista.

```
FutureBuilder<List>(  
    future: getdata_SITIOS(), Se llama el listado de los sitios  
    builder: (context, snapshot){  
        if(snapshot.hasError) print(snapshot.error); Si existe un error (que  
        este vacío el listado o algo similar) se muestra el error por consola
```

```

    return snapshot.hasData Se retorna la información
    ? ListaSitiosRegistrados( Si no hay un error, se llama la clase que
hace la lista
    list: snapshot.data?[], Pasandole la información
    )
    : const Center( Si se demora la consulta, se muestra una barra
circular de carga
    child: CircularProgressIndicator(),
    );
  },
),

ListView.builder( Lista del contenido
  itemCount: list.length, Se declara la cantidad de elementos en la lista
  itemBuilder: (BuildContext context, int i){
    return Card(
      child: Text(
        list[i]['SITIO'],
        style: const TextStyle(fontSize: 20),
      ),
    );
  },
),

```

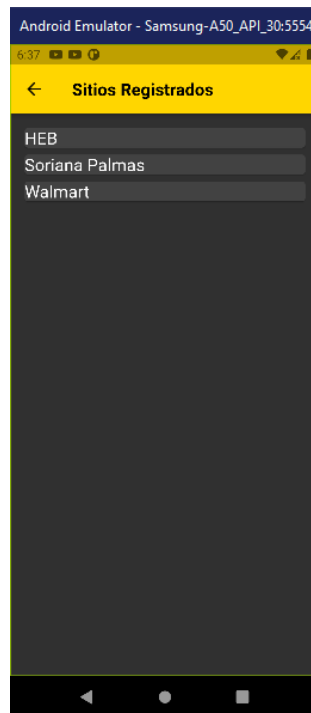


Figura 12. Lista de sitios registrados

Pulsando el icono de la flecha hacia la izquierda ubicado en la parte superior, se regresa hacia la pantalla anterior en la cual para terminar el proceso de registro se pulsa el botón de finalizar, este hace lo mismo que el del formulario pasado (no se explicara de nuevo para evitar redundancia), aunque claramente los datos se dirigen hacia la tabla correspondiente a este formulario (Figura 13), la cual tiene atributos diferentes a la anterior pero que se encuentran relacionadas entre sí mediante el campo SITIO (Figura 14).

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1 SITIO	varchar(60)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	2 NOMBRE_DUEÑO	varchar(60)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	3 NOMBRE	varchar(40)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	4 APELLIDO	varchar(40)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	5 TELEFONO	varchar(10)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	6 DOMICILIO	varchar(200)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	7 CORREO	varchar(100)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	8 CURP	varchar(18)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	9 RFC	varchar(13)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	10 UNIDAD	varchar(10)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	11 MATRICULA	varchar(10)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	12 COLOR	varchar(7)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más

Figura 13. Tabla para los datos de los taxistas dependientes



Figura 14. Relación entre las tablas de los sitios (flotilla) y taxistas (dependientes)

En el tercer botón de la pantalla de las opciones de registro, se estableció que cuando sea pulsado se mande llamar a la pantalla ("/RegistroIndependiente") donde se encuentra el formulario para registrar taxistas independientes es decir taxis libres (Figura 15). Realmente, se desarrolló muy similar al formulario para registrar taxistas en una flotilla (solo se omitió el campo del sitio) con el propósito de que el usuario no se sienta confundido al tener que entender dos formularios completamente diferentes.

Figura 15. Formulario para registrar taxistas independientes

Debido a la similitud entre este formulario y el anterior, no se explicará todo el funcionamiento de nuevo porque prácticamente es el mismo. Lo único en que se diferencia del anterior es que se cambia el destino de la información en los archivos php, puesto que se creó una tabla especialmente para este formulario en la base de datos (Figura 16).

```
<?php

include 'conexion.php';

$NOMBRE_DUENO = $_POST['NOMBRE_DUENO'];
$NOMBRE = $_POST['NOMBRE'];
$APELLIDO = $_POST['APELLIDO'];
$TELEFONO = $_POST['TELEFONO'];
$DOMICILIO = $_POST['DOMICILIO'];
$CORREO = $_POST['CORREO'];
```

```

$CURP = $_POST['CURP'];
$RFC = $_POST['RFC'];
$UNIDAD = $_POST['UNIDAD'];
$MATRICULA = $_POST['MATRICULA'];
$COLOR = $_POST['COLOR'];

$conn->query("INSERT INTO independientes
(NOMBRE_DUENO,NOMBRE,APELLIDO,TELEFONO,DOMICILIO,CORREO,CURP,RFC,UN
IDAD,MATRICULA,COLOR) VALUES
('".$NOMBRE_DUENO."','".$NOMBRE."','".$APELLIDO."','".$TELEFONO."','".$
".$DOMICILIO."','".$CORREO."','".$CURP."','".$RFC."','".$UNIDAD."','".$
".$MATRICULA."','".$COLOR."')")

```

?>

taxi_app independientes	
NOMBRE_DUENO	varchar(60)
NOMBRE	varchar(40)
APELLIDO	varchar(40)
TELEFONO	varchar(10)
DOMICILIO	varchar(200)
CORREO	varchar(100)
CURP	varchar(18)
RFC	varchar(13)
UNIDAD	varchar(10)
MATRICULA	varchar(10)
COLOR	varchar(7)

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1 NOMBRE_DUENO	varchar(60)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	2 NOMBRE	varchar(40)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	3 APELLIDO	varchar(40)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	4 TELEFONO	varchar(10)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	5 DOMICILIO	varchar(200)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	6 CORREO	varchar(100)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	7 CURP	varchar(18)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	8 RFC	varchar(13)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	9 UNIDAD	varchar(10)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	10 MATRICULA	varchar(10)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	11 COLOR	varchar(7)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más

Figura 16. Tabla para los datos de los taxistas independientes

V.2 FORMULARIO PARA LA SOLICITUD DE UN TAXI

Esta es la segunda fase del desarrollo de la aplicación, se decidió crear este formulario en la pantalla principal de la aplicación para mayor comodidad y también que fuera sencillo para que cualquier usuario pueda solicitar un taxi.

Primeramente, se colocaron los campos de textos necesarios para la solicitud de un taxi (Figura 17), esto se hizo con la ayuda de los widgets llamados TextFormField, los cuales también se emplearon en la creación de los formularios de registro. Para esta pantalla solo se emplearon 4 campos, de los cuales 2 son obligatorios (origen y destino) y los otros 2 son opcionales (Referencia del Origen y Referencia del Destino).

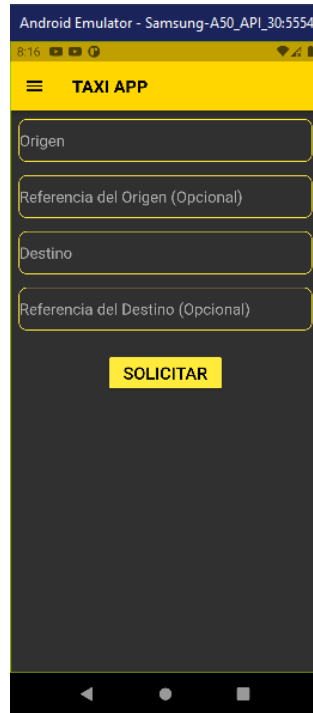


Figura 17. Formulario para solicitar un taxi

Posteriormente, se colocó un botón con el texto “SOLICITAR” que al pulsarlo es el que realiza todo el trabajo, primero se valida si los campos obligatorios están llenos, cuando es falso no se realiza nada y se manda un mensaje en el campo que se encuentra vacío y si es verdadero se manda llamar a la función (usada en los demás formularios) que comprueba la conexión de red mediante un ping a una página de internet. Si la función retorna false significa que no hay conexión a internet y en este caso se estableció que no hay una acción. Si la función retorna true significa que si hay conexión a internet y en este caso se estableció que se mande llamar a la función que solicita acceso a la base de datos para mandar la información.

```
void adddata_viajes(){ Función que llama a la base de datos para enviar los datos
```

```

var url="http://192.168.1.69/taxi_app/adddata_viajes.php";
http.post(Uri.parse(url), body: { Se envían los datos (ingresados por
el usuario) de los controladores a cada atributo de la tabla en la base
de datos
  "ORIGEN": controllerORIGEN.text,
  "REFERENCIA_ORIGEN": controllerREFERENCIA_ORIGEN.text,
  "DESTINO": controllerDESTINO.text,
  "REFERENCIA_DESTINO": controllerREFERENCIA_DESTINO.text
});
}

```

Esta función requiere de varios archivos php los cuales son los encargados de realizar la conexión entre la aplicación y la base de datos en el servidor local.

```

<?php
$connect = new mysqli("localhost","root","", "taxi_app");
if($connect){
}else{
    echo "Fallo, revise ip o firewall";
    exit();
}

<?php

include 'conexion.php';

$ORIGEN = $_POST['ORIGEN'];
$REFERENCIA_ORIGEN = $_POST['REFERENCIA_ORIGEN'];
$DESTINO = $_POST['DESTINO'];
$REFERENCIA_DESTINO = $_POST['REFERENCIA_DESTINO'];

$connect->query("INSERT INTO viajes
(ORIGEN,REFERENCIA_ORIGEN,DESTINO,REFERENCIA_DESTINO) VALUES
('".$ORIGEN."','".$REFERENCIA_ORIGEN."','".$DESTINO."','".$REFERENC
IA_DESTINO."')")

?>

```

Claramente, para llegar a este punto ya se creó la tabla (Figura 18) correspondiente a este formulario en la base de datos alojada en el servidor local.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/> 1	ID	int(5)			No	Ninguna		AUTO_INCREMENT	Cambiar Eliminar Más
<input type="checkbox"/> 2	ORIGEN	varchar(100)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/> 3	REFERENCIA_ORIGEN	varchar(100)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/> 4	DESTINO	varchar(100)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/> 5	REFERENCIA_DESTINO	varchar(100)	utf8_spanish2_ci		No	Ninguna			Cambiar Eliminar Más

Figura 18. Tabla para los datos del viaje

VI. RESULTADOS

Debido a que el proyecto se encuentra en la primera fase de desarrollo, es imposible ponerla a prueba para obtener resultados. Lo que se hizo en su lugar, fue mostrar el progreso que se lleva a diferentes taxistas profesionales de Ciudad Victoria, Tamaulipas con los cuales se mantuvo comunicación a lo largo del desarrollo de la aplicación.

Ellos comentaron que les parecía una buena solución al problema que presentan y sobre todo recalcaron que el diseño era simple pero completo, lo que provoca que cualquier taxista con acceso a un celular inteligente e internet pueda usarla.

Por otro lado, también se le mostró la aplicación a los pasajeros, los cuales mencionaron que se les hacía más simple que otras aplicaciones similares a esta y que si se llegara a lanzar al mercado obteniendo buenos resultados (en temas de seguridad) sería seguro que la usarían en su día a día.

VII. CONCLUSIONES

Como se expuso en la sección anterior, los resultados obtenidos del progreso que se lleva hasta el momento, son buenos y esperanzadores. Dichos resultados hacen pensar que el objetivo principal el cual es generar más trabajo a los taxistas profesionales será cumplido.

Con la realización de este proyecto se aprendió como es el ambiente laboral en una empresa, lo cual sirve para irse formando como profesional desde estos momentos. Verdaderamente es algo increíble el empezar desde cero un proyecto que tiene mucha proyección positiva a futuro y sobre todo que tendrá consecuencias positivas para los taxistas trabajadores, honestos y honrados de Ciudad Victoria.

Además, al haber desarrollado esta aplicación de manera individual incrementa las habilidades de autoaprendizaje que sinceramente para este tipo de proyectos son muy utilizadas y explotadas al máximo. Claramente, esto trae una carga de trabajo muy pesada, pero con organización y ganas de trabajar todo se puede lograr.

BIBLIOGRAFÍA

[1] Congreso del Estado, «LEY DE TRANSPORTE DEL ESTADO DE TAMAULIPAS,» Periódico Oficial, Ciudad Victoria, 19 de diciembre del 2001.

[2] J. Bravo, «5 razones por las que los usuarios prefieren Uber sobre los taxis,» <https://autosblogmexico.com/>, México, 2019.

ANEXOS

Código completo de la pantalla principal

```
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:io';

class HomePedirTaxi extends StatefulWidget {
  const HomePedirTaxi({Key? key}) : super(key: key);

  @override
  _HomePedirTaxiState createState() {
    return _HomePedirTaxiState();
  }
}

class _HomePedirTaxiState extends State<HomePedirTaxi> {

  TextEditingController controllerORIGEN=TextEditingController();//Se
  declaran los controladores que guardan las respuestas(variables)
  TextEditingController
  controllerREFERENCIA_ORIGEN=TextEditingController();
  TextEditingController controllerDESTINO=TextEditingController();
  TextEditingController
  controllerREFERENCIA_DESTINO=TextEditingController();

  final formKey=GlobalKey<FormState>();//Se declara una globalkey (Para
  acceder desde cualquier lugar)

  FocusNode focusREFERENCIA_ORIGEN=FocusNode();//Se declara la variable
  focus que ayudan a pasar al siguiente cuadro de texto
  FocusNode focusDESTINO=FocusNode();
  FocusNode focusREFERENCIA_DESTINO=FocusNode();
  void requestFocus(BuildContext context, FocusNode focusNode){//Funcion
  que pasa al siguiente cuadro de texto
    FocusScope.of(context).requestFocus(focusNode);
  }

  Future<bool> internetConnectivity() async { //Función que comprueba la
  conexión a internet mediante un ping
    try {
      final result = await InternetAddress.lookup('google.com');
      if (result.isNotEmpty && result[0].rawAddress.isNotEmpty) {
        return true;//Conectado
      }
    } on SocketException catch (_) {
      return false;//Desconectado
    }
    return false;
  }
```

```

    void adddata_viajes(){//Función que llama a la base de datos para enviar
    los datos
        var          url="http://192.168.1.69/taxi_app/adddata_viajes.php";//
    IpDeComputadora/NombreDeCarpeta/NombreDelArchivoPHP
        // El archivo es el php que enlaza con la base de datos
        http.post(Uri.parse(url), body: { //Se envían los datos (ingresados por
    el usuario) de los controladores a cada atributo de la tabla en la base de
    datos
            "ORIGEN": controllerORIGEN.text,
            "REFERENCIA_ORIGEN": controllerREFERENCIA_ORIGEN.text,
            "DESTINO": controllerDESTINO.text,
            "REFERENCIA_DESTINO": controllerREFERENCIA_DESTINO.text
        });
    }

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                backgroundColor: Colors.yellowAccent[700], //Color de la barra
    superior
                foregroundColor: Colors.black, //Color principal de la barra
    superior
                title: const Text(
                    "TAXI APP", //Título en la barra superior
                    style: TextStyle(fontWeight: FontWeight.bold, ), //Negritas
                ),
            ),
            drawer: Drawer( //Cajón de Menú
                child: ListView( //lista de opciones
                    children: <Widget>[
                        DrawerHeader( //Encabezado
                            decoration: const BoxDecoration(
                                gradient:      LinearGradient(colors:      [Colors.black38,
    Colors.yellow]),
                            ),
                        child: Row( //Contenido del encabezado
                            children: <Widget>[
                                Image.asset("assets/logo.png"),
                                const SizedBox( //Espacio entre imagen y texto
                                    width: 10,
                                ),
                                const Text(
                                    "Menú",
                                    style: TextStyle(fontSize: 40, color: Colors.white, ),
                                ),
                            ],
                        ),
                    ],
                ),
            ),
        ),
    
```

```

        ListTile( //Opción de registro para taxistas
          leading: Image.asset("assets/icono_taxi.png"),
          title: const Text(
            "Pertener a TAXI APP",
            style: TextStyle(fontSize: 17,)),
        ),
        onTap: () {
          Navigator.pop(context); //Desaparece el drawer
          Navigator.of(context).pushNamed("/OpcionesDeRegistro");
        },
      ),
    ],
  ),
  body: Padding( //Espacio entre los margenes de la pantalla y el
    contenido
    padding: const EdgeInsets.all(10), //Tamaño del espacio
    child: Form( //Widget de Formularios
      key: formKey, //Se usa la globalkey
      child: Column( //Contenido
        children: <Widget>[
          Container( //Encierra el campo de texto para agregar
            decoration: BoxDecoration(
              borderRadius: BorderRadius.circular(10), //Borde
              border: Border.all(color: Colors.yellow) //Color de
            ),
            child: TextFormField(
              controller: controllerORIGEN, //Se indica el controlador
              style: const TextStyle(fontSize: 18), //Tamaño de texto
              decoration: const InputDecoration(hintText: "Origen",
                border: InputBorder.none), //Se le quita el borde al cuadro de texto
              validator: (value){
                if(value == null || value.isEmpty){
                  return "Campo Obligatorio";
                }
                return null;
              },
              onEditingComplete: () => requestFocus(context,
                focusREFERENCIA_ORIGEN), //Se manda llamar la función que cambia de cuadro
                de texto indicándole a cual queremos ir
                textInputAction: TextInputAction.next, //Se cambia la
                tecla listo por siguiente en el teclado
              ),
            ),
          ),
        ],
      ),
    ),
  ),

```

```

const SizedBox(//Espacio
  height: 15,
),
Container( //Encierra el campo de texto para agregar
decoracion
  decoration: BoxDecoration(
    borderRadius: BorderRadius.circular(10), //Borde
circular
    border: Border.all(color: Colors.yellow) //Color de
borde
  ),
  child: TextFormField(
    controller: controllerREFERENCIA_ORIGEN, //Se indica el
controlador correspondiente a este cuadro de texto
    style: const TextStyle(fontSize: 18), //Tamaño de texto
    decoration: const InputDecoration(hintText:"Referencia
del Origen (Opcional)", border: InputBorder.none), //Se le quita el borde
al cuadro de texto
    focusNode: focusREFERENCIA_ORIGEN, //Se indica la
variable focus correspondiente a este cuadro de texto
    onEditingComplete: () => requestFocus(context,
focusDESTINO), //Se manda llamar la función que cambia de cuadro de texto
indicandole a cual queremos ir
    textInputAction: TextInputAction.next, //Se cambia la
tecla listo por siguiente en el teclado
  ),
),
const SizedBox(//Espacio
  height: 15,
),
Container( //Encierra el campo de texto para agregar
decoracion
  decoration: BoxDecoration(
    borderRadius: BorderRadius.circular(10), //Borde
circular
    border: Border.all(color: Colors.yellow) //Color de
borde
  ),
  child: TextFormField(
    controller: controllerDESTINO, //Se indica el
controlador correspondiente a este cuadro de texto
    style: const TextStyle(fontSize: 18), //Tamaño de texto
    decoration: const InputDecoration(hintText:"Destino",
border: InputBorder.none), //Se le quita el borde al cuadro de texto
    validator: (value){
      if(value == null || value.isEmpty){
        return "Campo Obligatorio";
      }
      return null;
    },
  ),

```

```

        focusNode: focusDESTINO, //Se indica la variable focus
correspondiente a este cuadro de texto
        onEditingComplete: () => requestFocus(context,
focusREFERENCIA_DESTINO), //Se manda llamar la función que cambia de cuadro
de texto indicándole a cual queremos ir
        textInputAction: TextInputAction.next, //Se cambia la
tecla listo por siguiente en el teclado
    ),
),
const SizedBox( //Espacio
    height: 15,
),
Container( //Encierra el campo de texto para agregar
decoracion
    decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(10), //Borde
circular
        border: Border.all(color: Colors.yellow) //Color de
borde
    ),
    child: TextFormField(
        controller: controllerREFERENCIA_DESTINO, //Se indica el
controlador correspondiente a este cuadro de texto
        style: const TextStyle(fontSize: 18), //Tamaño de texto
        decoration: const InputDecoration(hintText: "Referencia
del Destino (Opcional)", border: InputBorder.none), //Se le quita el borde
al cuadro de texto
        focusNode: focusREFERENCIA_DESTINO, //Se indica la
variable focus correspondiente a este cuadro de texto
    ),
),
const SizedBox( //Espacio
    height: 25,
),
RaisedButton( //Botón
    child: const Text("SOLICITAR",
        style: TextStyle(fontSize: 20,)),
    ),
    color: Colors.yellow, //Color del botón
    textColor: Colors.black, //Color del texto del botón
    onPressed: () async{
        if(formKey.currentState!.validate()){ //Se comprueba que
los cuadros de texto contengan algo
            var ping=await internetConnectivity(); //Se manda
llamar la función que comprueba la conexión a internet para obtener una
respuesta

            if(ping==true){ //Si hay conexión a internet
                adddata_viajes(); //Se manda llamar la función que
llama a la base de datos para enviar los datos
            }else{ //Si no hay conexión a internet

```



```

}
}
);
),
),
),
],
),
},
}
}

```