Building Places & Location Search with Map View Using Flutter

alfianlosari.com/posts/building-places-location-search-mapview-flutter/

Published at Dec 11, 2018

Flutter has just achieved its latest milestone with the release of stable 1.0 version. It has many amazing features that provide a foundation for amazing future ahead as it matures. Here are several of the great features:

1. PlatformView. Widget to render native platform view such as iOS UIView and Android View
2. MapView plugin. Widget to display Google Map View built on top of PlatformView API
3. Add to App: Tools to convert existing iOS and Android apps to Flutter in stages.
4. Flutter Desktop Embedding & Hummingbird: An initiative to bring Flutter to Web & Desktop.

To celebrate the release of Flutter 1.0, we are going to build an app to display and search nearby places inside Map View using Google Map and Google Places API.

You can download the source code of the project in the GitHub repository at alfianlosari/flutter_places

What we will build

Here are the things that we will perform for this project:

1. Project Setup & Import dependencies.
2. Building Main List that displays popular places around user location using GPS access.
3. Building Detail Screen that displays the place location inside the Map with associated information (name, address, photos, etc).
4. Search Places using Google Places Autocomplete.

Project Setup & Import Dependencies

First make sure you have upgraded your Flutter SDK to 1.0.0 using flutter upgrade command from your shell. Create a new project using flutter create create command, open the project using Visual Studio Code or Android Studio. Next, we will add dependencies inside the pubspec.yaml file. There are 3 dependencies we will use:

1. `google_maps_flutter` : Official Google Map View plugin from the Flutter team that uses PlatformView under the hood. It is still in Developer Preview status, so breaking changes and bugs can be expected. google_maps_flutter.
2. `flutter_google_places` : Plugin from Hadrien Lejard that provides interface to us to communicate with the Google Places Autocomplete API. flutter_google_places.
3. `location` : Plugin from Guillaume Bernos that uses Platform Channel API to access GPS on Android & iOS devices used to get user current location. location pub.

```
name: Placez
description: Display & Search your nearby places

version: 1.0.0+1

environment:
  sdk: ">=2.0.0-dev.68.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter

  google_maps_flutter: ^0.0.3
  flutter_google_places: ^0.1.4
  location: ^1.4.1

dev_dependencies:
  flutter_test:
    sdk: flutter

flutter:
  uses-material-design: true
```

Our app uses Google Maps & Google Places services under the hood to display map view and fetch nearby places, so we need Google API Key for Map & Places. Make sure to create the API key from the site at Places | Google Maps Platform | Google Cloud

There are several configuration steps we need to perform before we can begin to build the app:

1. iOS platform need to opt-in for Flutter embedded preview by adding a boolean property to the app's Info.plist file, with the key io.flutter.embedded*views*preview and the value YES. Make sure to also add NSLocationWhenInUseUsageDescription & NSLocationAlwaysUsageDescription with the description text for GPS access permission.
2. iOS platform need to specify your API key in the application delegate `ios/Runner/AppDelegate.m`

3. Android platform need to specify your API key and add permissions in the application manifest `android/app/src/main/AndroidManifest.xml`.

iOS AppDelegate.m

```
#include "AppDelegate.h"
#include "GeneratedPluginRegistrant.h"
#import "GoogleMaps/GoogleMaps.h"

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
  [GMSServices provideAPIKey:@"YOUR KEY HERE"];
  [GeneratedPluginRegistrant registerWithRegistry:self];
  return [super application:application didFinishLaunchingWithOptions:launchOptions];
}
@end
```

Android Application Manfiest

```
<manifest ...
<uses-permission android:name="android.permission.INTERNET"/>      <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />  <application ...
  <meta-data android:name="com.google.android.geo.API_KEY"
             android:value="YOUR KEY HERE"/>
```

Building Trending Around Places List Screen

In our main screen, we will ask user to grant GPS access and access their location, then we will ask the Google Places API to get nearby places to visit around. Make sure to paste your Google API key in the YOUR*API*KEY placeholder. Here are the main components of this widget:

1. This widget consists of Scaffold with AppBar containing 2 actions item, refresh and search . The body uses Column with MapView as the first item, the second item wraps the ListView in an Expanded widget so it can flex all the remaining vertical space of the screen.
2. When the MapView has been created, a method is invoked to assign the MapControlller as the widget property, then it calls the refresh method that will asks user their current location using location plugin, then using the location it will invoke getNearbyPlaces to retrieve the nearby places using flutter*google*places plugin, then assigned the places as the state of the widget.
3. The places will be displayed in a Card inside ListView, each row shows the place name, address, and type. The inline MapView on the top displays all the pin marker of the places. When user taps on the row, the app will navigate to the place detail screen passing the placeId.

```
import 'dart:async';
import 'package:google_maps_webservice/places.dart';
import 'package:flutter_google_places/flutter_google_places.dart';
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:location/location.dart' as LocationManager;
import 'place_detail.dart';

const kGoogleApiKey = "TOUR_API_KEY";
GoogleMapsPlaces _places = GoogleMapsPlaces(apiKey: kGoogleApiKey);

void main() {
  runApp(MaterialApp(
    title: "PlaceZ",
    home: Home(),
    debugShowCheckedModeBanner: false,
  ));
}

class Home extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return HomeState();
  }
}

class HomeState extends State<Home> {
  final homeScaffoldKey = GlobalKey<ScaffoldState>();
  GoogleMapController mapController;
  List<PlacesSearchResult> places = [];
  bool isLoading = false;
  String errorMessage;

  @override
  Widget build(BuildContext context) {
    Widget expandedChild;
    if (isLoading) {
      expandedChild = Center(child: CircularProgressIndicator(value: null));
    } else if (errorMessage != null) {
      expandedChild = Center(
        child: Text(errorMessage),
      );
    } else {
      expandedChild = buildPlacesList();
    }

    return Scaffold(
        key: homeScaffoldKey,
        appBar: AppBar(
          title: const Text("PlaceZ"),
          actions: <Widget>[
            isLoading
                ? IconButton(
                    icon: Icon(Icons.timer),
```

```
              onPressed: () {},
            )
          : IconButton(
              icon: Icon(Icons.refresh),
              onPressed: () {
                refresh();
              },
            ),
        IconButton(
          icon: Icon(Icons.search),
          onPressed: () {
            _handlePressButton();
          },
        ),
      ],
    ),
    body: Column(
      children: <Widget>[
        Container(
          child: SizedBox(
              height: 200.0,
              child: GoogleMap(
                  onMapCreated: _onMapCreated,
                  options: GoogleMapOptions(
                      myLocationEnabled: true,
                      cameraPosition:
                          const CameraPosition(target: LatLng(0.0, 0.0))))),
        ),
        Expanded(child: expandedChild)
      ],
    ));
}

void refresh() async {
  final center = await getUserLocation();

  mapController.animateCamera(CameraUpdate.newCameraPosition(CameraPosition(
      target: center == null ? LatLng(0, 0) : center, zoom: 15.0)));
  getNearbyPlaces(center);
}

void _onMapCreated(GoogleMapController controller) async {
  mapController = controller;
  refresh();
}

Future<LatLng> getUserLocation() async {
  var currentLocation = <String, double>{};
  final location = LocationManager.Location();
  try {
    currentLocation = await location.getLocation();
    final lat = currentLocation["latitude"];
    final lng = currentLocation["longitude"];
    final center = LatLng(lat, lng);
```

```
    return center;
  } on Exception {
    currentLocation = null;
    return null;
  }
}

void getNearbyPlaces(LatLng center) async {
  setState(() {
    this.isLoading = true;
    this.errorMessage = null;
  });

  final location = Location(center.latitude, center.longitude);
  final result = await _places.searchNearbyWithRadius(location, 2500);
  setState(() {
    this.isLoading = false;
    if (result.status == "OK") {
      this.places = result.results;
      result.results.forEach((f) {
        final markerOptions = MarkerOptions(
            position:
                LatLng(f.geometry.location.lat, f.geometry.location.lng),
            infoWindowText: InfoWindowText("${f.name}", "${f.types?.first}"));
        mapController.addMarker(markerOptions);
      });
    } else {
      this.errorMessage = result.errorMessage;
    }
  });
}

void onError(PlacesAutocompleteResponse response) {
  homeScaffoldKey.currentState.showSnackBar(
    SnackBar(content: Text(response.errorMessage)),
  );
}

Future<void> _handlePressButton() async {
  try {
    final center = await getUserLocation();
    Prediction p = await PlacesAutocomplete.show(
        context: context,
        strictbounds: center == null ? false : true,
        apiKey: kGoogleApiKey,
        onError: onError,
        mode: Mode.fullscreen,
        language: "en",
        location: center == null
            ? null
            : Location(center.latitude, center.longitude),
        radius: center == null ? null : 10000);

    showDetailPlace(p.placeId);
```

```
  } catch (e) {
    return;
  }
}

Future<Null> showDetailPlace(String placeId) async {
  if (placeId != null) {
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => PlaceDetailWidget(placeId)),
    );
  }
}

ListView buildPlacesList() {
  final placesWidget = places.map((f) {
    List<Widget> list = [
      Padding(
        padding: EdgeInsets.only(bottom: 4.0),
        child: Text(
          f.name,
          style: Theme.of(context).textTheme.subtitle,
        ),
      )
    ];
    if (f.formattedAddress != null) {
      list.add(Padding(
        padding: EdgeInsets.only(bottom: 2.0),
        child: Text(
          f.formattedAddress,
          style: Theme.of(context).textTheme.subtitle,
        ),
      ));
    }

    if (f.vicinity != null) {
      list.add(Padding(
        padding: EdgeInsets.only(bottom: 2.0),
        child: Text(
          f.vicinity,
          style: Theme.of(context).textTheme.body1,
        ),
      ));
    }

    if (f.types?.first != null) {
      list.add(Padding(
        padding: EdgeInsets.only(bottom: 2.0),
        child: Text(
          f.types.first,
          style: Theme.of(context).textTheme.caption,
        ),
      ));
    }
```

```
      return Padding(
        padding: EdgeInsets.only(top: 4.0, bottom: 4.0, left: 8.0, right: 8.0),
        child: Card(
          child: InkWell(
            onTap: () {
              showDetailPlace(f.placeId);
            },
            highlightColor: Colors.lightBlueAccent,
            splashColor: Colors.red,
            child: Padding(
              padding: EdgeInsets.all(8.0),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.start,
                crossAxisAlignment: CrossAxisAlignment.start,
                children: list,
              ),
            ),
          ),
        ),
      );
    }).toList();

    return ListView(shrinkWrap: true, children: placesWidget);
  }
}
```

Building Place Detail Screen

This screen displays the detail of the Places when user taps on the row of the main list screen with a MapView and the pin marker. Make sure to paste your Google API key in the YOUR*API*KEY placeholder. Here are the main components of this widget:

1. The widget consists of Scaffold , with AppBar that displays the name of the place inside the Title . The main body used the approach from the main widget that uses Column to stack widget vertically, MapView and ListView .
2. ListView displays the places information such as the name, address, phone number for each of its item . It also displays the photos of the place in a horizontal ListView that acts like a carousel in the first item.
3. When the widget initializes, the fetchPlaceDetail method is invoked to retrieve the detail of the place using the placeId from the Google Places API. It will then assigned the details as the state of the widget.

```
import 'package:google_maps_webservice/places.dart';
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';

const kGoogleApiKey = "YOUR_API_KEY";
GoogleMapsPlaces _places = GoogleMapsPlaces(apiKey: kGoogleApiKey);

class PlaceDetailWidget extends StatefulWidget {
```

```
  String placeId;

  PlaceDetailWidget(String placeId) {
    this.placeId = placeId;
  }

  @override
  State<StatefulWidget> createState() {
    return PlaceDetailState();
  }
}

class PlaceDetailState extends State<PlaceDetailWidget> {
  GoogleMapController mapController;
  PlacesDetailsResponse place;
  bool isLoading = false;
  String errorLoading;

  @override
  void initState() {
    fetchPlaceDetail();
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    Widget bodyChild;
    String title;
    if (isLoading) {
      title = "Loading";
      bodyChild = Center(
        child: CircularProgressIndicator(
          value: null,
        ),
      );
    } else if (errorLoading != null) {
      title = "";
      bodyChild = Center(
        child: Text(errorLoading),
      );
    } else {
      final placeDetail = place.result;
      final location = place.result.geometry.location;
      final lat = location.lat;
      final lng = location.lng;
      final center = LatLng(lat, lng);

      title = placeDetail.name;
      bodyChild = Column(
        mainAxisAlignment: MainAxisAlignment.start,
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: <Widget>[
          Container(
              child: SizedBox(
```

```
          height: 200.0,
          child: GoogleMap(
            onMapCreated: _onMapCreated,
            options: GoogleMapOptions(
                myLocationEnabled: true,
                cameraPosition: CameraPosition(target: center, zoom: 15.0)),
          ),
        )),
        Expanded(
          child: buildPlaceDetailList(placeDetail),
        )
      ],
    );
  }

  return Scaffold(
      appBar: AppBar(
        title: Text(title),
      ),
      body: bodyChild);
}

void fetchPlaceDetail() async {
  setState(() {
    this.isLoading = true;
    this.errorLoading = null;
  });

  PlacesDetailsResponse place =
      await _places.getDetailsByPlaceId(widget.placeId);

  if (mounted) {
    setState(() {
      this.isLoading = false;
      if (place.status == "OK") {
        this.place = place;
      } else {
        this.errorLoading = place.errorMessage;
      }
    });
  }
}

void _onMapCreated(GoogleMapController controller) {
  mapController = controller;
  final placeDetail = place.result;
  final location = place.result.geometry.location;
  final lat = location.lat;
  final lng = location.lng;
  final center = LatLng(lat, lng);
  var markerOptions = MarkerOptions(
      position: center,
      infoWindowText: InfoWindowText(
          "${placeDetail.name}", "${placeDetail.formattedAddress}"));
```

```
    mapController.addMarker(markerOptions);
    mapController.animateCamera(CameraUpdate.newCameraPosition(
        CameraPosition(target: center, zoom: 15.0)));
  }

  String buildPhotoURL(String photoReference) {
    return "https://maps.googleapis.com/maps/api/place/photo?
maxwidth=400&photoreference=${photoReference}&key=${kGoogleApiKey}";
  }

  ListView buildPlaceDetailList(PlaceDetails placeDetail) {
    List<Widget> list = [];
    if (placeDetail.photos != null) {
      final photos = placeDetail.photos;
      list.add(SizedBox(
          height: 100.0,
          child: ListView.builder(
              scrollDirection: Axis.horizontal,
              itemCount: photos.length,
              itemBuilder: (context, index) {
                return Padding(
                    padding: EdgeInsets.only(right: 1.0),
                    child: SizedBox(
                      height: 100,
                      child: Image.network(
                          buildPhotoURL(photos[index].photoReference)),
                    ));
              })));
    }

    list.add(
      Padding(
          padding:
              EdgeInsets.only(top: 4.0, left: 8.0, right: 8.0, bottom: 4.0),
          child: Text(
            placeDetail.name,
            style: Theme.of(context).textTheme.subtitle,
          )),
    );

    if (placeDetail.formattedAddress != null) {
      list.add(
        Padding(
            padding:
                EdgeInsets.only(top: 4.0, left: 8.0, right: 8.0, bottom: 4.0),
            child: Text(
              placeDetail.formattedAddress,
              style: Theme.of(context).textTheme.body1,
            )),
      );
    }

    if (placeDetail.types?.first != null) {
      list.add(
```

```
    Padding(
        padding:
            EdgeInsets.only(top: 4.0, left: 8.0, right: 8.0, bottom: 0.0),
        child: Text(
          placeDetail.types.first.toUpperCase(),
          style: Theme.of(context).textTheme.caption,
        )),
  );
}

if (placeDetail.formattedPhoneNumber != null) {
  list.add(
    Padding(
        padding:
            EdgeInsets.only(top: 4.0, left: 8.0, right: 8.0, bottom: 4.0),
        child: Text(
          placeDetail.formattedPhoneNumber,
          style: Theme.of(context).textTheme.button,
        )),
  );
}

if (placeDetail.openingHours != null) {
  final openingHour = placeDetail.openingHours;
  var text = '';
  if (openingHour.openNow) {
    text = 'Opening Now';
  } else {
    text = 'Closed';
  }
  list.add(
    Padding(
        padding:
            EdgeInsets.only(top: 0.0, left: 8.0, right: 8.0, bottom: 4.0),
        child: Text(
          text,
          style: Theme.of(context).textTheme.caption,
        )),
  );
}

if (placeDetail.website != null) {
  list.add(
    Padding(
        padding:
            EdgeInsets.only(top: 0.0, left: 8.0, right: 8.0, bottom: 4.0),
        child: Text(
          placeDetail.website,
          style: Theme.of(context).textTheme.caption,
        )),
  );
}

if (placeDetail.rating != null) {
```

```
      list.add(
        Padding(
            padding:
                EdgeInsets.only(top: 0.0, left: 8.0, right: 8.0, bottom: 4.0),
            child: Text(
              "Rating: ${placeDetail.rating}",
              style: Theme.of(context).textTheme.caption,
            )),
      );
    }

    return ListView(
      shrinkWrap: true,
      children: list,
    );
  }
}
```

Building Search with Google Places Autocomplete

When user taps on the search ActionBar un the main list AppBar. We will display a search bar where user can enter the name of the place they want to search. Then it will provide Autocomplete suggestions using Google Places autocomplete API for user to choose using a dropdown.

When they tap on the place, we just navigate to the place detail widget passing the placeId just like the nearby places item in the main widgetListView.

Conclusion

That's a wrap!. In such a short time we have been able to build cross platform native mobile application with high performance and rapid development that runs on Android & iOS. Flutter provides unlimited potential to us mobile developers to create the most delightful app to our users. The sky is the limit! 😊.

Tags:

- flutter
- mapview