Diplomado en Desarrollo de Software

Parte V

Programación Móvil Avanzada

Ruthford Jay / rjay@uninorte.edu.co

Programación Móvil Avanzada

Notificaciones y Tareas en Segundo Plano

Objetivos de la Sesión

Al finalizar esta sesión estarás en capacidad de:

- Entender la infraestructura necesaria para el manejo de notificaciones
- Desarrollo de una aplicación apoyada en notificaciones
- Entender las diferentes implementaciones de tareas en segundo plano

Notificaciones

Las notificaciones son una excelente manera de involucrar a sus usuarios para que regresen a su aplicación o para que presten atención a algo mientras están en la aplicación. Hay dos tipos de notificaciones:

- Notificaciones push
- Notificaciones locales

Las notificaciones locales se originan en la propia aplicación, a diferencia de las notificaciones push que se activan desde un servidor remoto.

Notificaciones Locales

Usar Notification en Android es relativamente simple, si quieres usar Notification en Flutter, se puede usar **Flutter Local Notifications Plugin**. Para esto, usamos el siguiente comando para instalarlo:

```
$ flutter pub add flutter_local_notifications
```

Notificaciones Locales: Android

Para mostrar notificaciones en Android el único requerimiento es contar con el icono de la aplicación en la carpeta drawable, posteriormente se pasará el nombre del archivo como argumento para hacer la inicialización. La ruta al archivo debe ser:

android\app\src\main\res\drawable\app_icon.png

Notificaciones Locales: Android

Para mostrar notificaciones en Android el único requerimiento es contar con el icono de la aplicación en la carpeta drawable, posteriormente se pasará el nombre del archivo como argumento para hacer la inicialización. La ruta al archivo debe ser:

android\app\src\main\res\drawable\app_icon.png

Notificaciones Locales: Android

```
import 'package:flutter_local_notifications/flutter_local_notifications.dart’;

late FlutterLocalNotificationsPlugin fnp;

var initializationSettingsAndroid = const AndroidInitializationSettings('notify_icon');
var initializationSettingsIOS = const IOSInitializationSettings();
var initializationSettings =
InitializationSettings(android: initializationSettingsAndroid, iOS:
initializationSettingsIOS);
fnp = FlutterLocalNotificationsPlugin();
fnp.initialize(initializationSettings, onSelectNotification: onSelectNotification);
```

Notificaciones Locales: Android

```
var androidPlatformChannelSpecifics = const AndroidNotificationDetails(
      'Notification-Channel-ID',
      'Channel Name',
      channelDescription: 'Description',
      playSound: false,
      importance: Importance.max,
      priority: Priority.high,
);
var iOSPlatformChannelSpecifics = const IOSNotificationDetails(presentSound: false);
var platformChannelSpecifics = NotificationDetails(android: androidPlatformChannelSpecifics,
iOS: iOSPlatformChannelSpecifics);
fnp.show(0, 'New Alert', 'How to show Local Notification', platformChannelSpecifics,
payload: 'Message Here!');
```

Tareas en Segundo Plano

Una tarea en segundo plano representa líneas de código que serán ejecutadas por fuera del flujo normal de la aplicación, en Flutter estas tareas no comparten recursos directamente con la ejecución de la aplicación.

Para establecer tareas en segundo plano de manera relativamente sencilla dentro de Flutter se usa la dependencia **WorkManager**. WorkManager permite establecer las tareas independiente de la plataforma, adaptándolas de manera automatizada a la plataforma en la que se esté ejecutando.

```
flutter pub add workmanager
```

WorkManager

Con WorkManager solo es necesario seguir los siguientes pasos:

- Definir un callbackDispatcher()
- Inicializar WorkManager()
- Registrar la tarea

WorkManager

Un CallbackDispatcher es un método de “nivel superior” o estático que se encarga de ejecutar la tarea. Un método de nivel superior simplemente es aquel que está definido fuera del entorno de la aplicación (al mismo nivel que **main**()):

Para inicializar el WorkManager solo es necesario pasar el **CallbackDispatcher**, preferiblemente antes de **runApp**():

isInDebugMode: true permite que cada vez que se ejecute la tarea se vea una notificación, la cual es útil para probar esta funcionalidad.

WorkManager

Para registrar tareas se define un id y un nombre, el cual permiten identificar la tarea, se pueden definir tareas de una sola ejecución o recurrentes, así como también se pueden definir el intervalo y el atraso de la primera ejecución.

Cuando se registra una tarea, el código de ejecución es el definido en el callbackDispatcher, si se desea programar más de una tarea se pueden usar los identificadores para modificar la ejecución en el callback.

WorkManager

```
import 'package:workmanager/workmanager.dart';

void callbackInBackground() async {
  Workmanager().executeTask((task, inputData) async {
    // Some work ...
    return Future.value(true);
  });
}

await Workmanager().initialize(
  callbackInBackground,
);

await Workmanager().registerPeriodicTask("task-id-1", "periodicTask", );
```

Componente práctico

Aplicación demo de Notificaciones y Tareas en Segundo Plano