dream

IMPLEMENTATION

http://dream-implementation.com info@dream-implementation.com

Push notifications

- Firebase Cloud Messaging
- Android and iOS

FCM setup

- Steps required for both platforms:
  1. Create new project in Firebase console
  2. Add Firebase Messaging plugin (https://pub.dev/packages/firebase_messaging) to pubspec.yaml

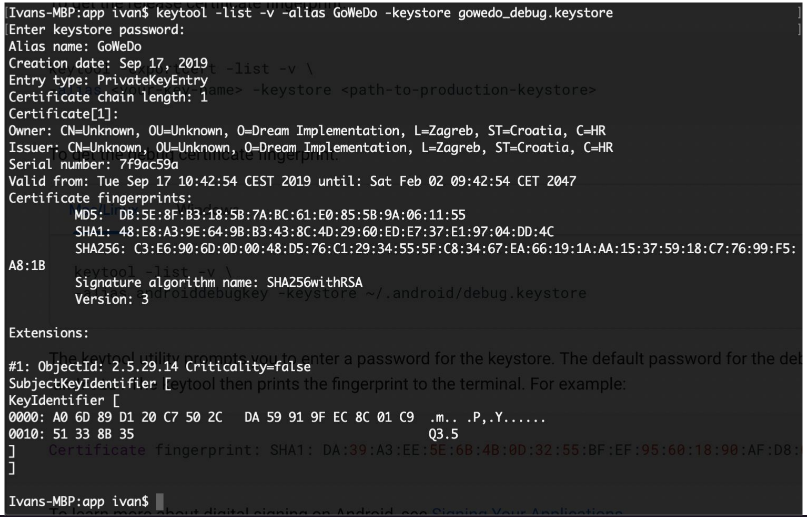
FCM setup Android

Steps required specifically for Android:

1. Create Android app in Firebase console Settings -> General tab
2. Download google-services.json from Settings -> General tab Android app section and put in *android/app/* folder

FCM setup Android

3. Generate SHA-1 for debug and release keystores (https://developers.google.com/android/guides/client-auth)



```
Ivans-MBP:app ivan$ keytool -list -v -alias GoWeDo -keystore gowedo_debug.keystore
Enter keystore password:
Alias name: GoWeDo
Creation date: Sep 17, 2019
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Unknown, OU=Unknown, O=Dream Implementation, L=Zagreb, ST=Croatia, C=HR
Issuer: CN=Unknown, OU=Unknown, O=Dream Implementation, L=Zagreb, ST=Croatia, C=HR
Serial number: 7f9ac59a
Valid from: Tue Sep 17 10:42:54 CEST 2019 until: Sat Feb 02 09:42:54 CET 2047
Certificate fingerprints:
         MD5:  DB:5E:8F:B3:18:5B:7A:BC:61:E0:85:5B:9A:06:11:55
         SHA1: 48:E8:A3:9E:64:9B:B3:43:8C:4D:29:60:ED:E7:37:E1:97:04:DD:4C
         SHA256: C3:E6:90:6D:0D:00:48:D5:76:C1:29:34:55:5F:C8:34:67:EA:66:19:1A:AA:15:37:59:18:C7:76:99:F5:
A8:1B
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: A0 6D 89 D1 20 C7 50 2C   DA 59 91 9F EC 8C 01 C9  .m.. .P,.Y......
0010: 51 33 8B 35                                        Q3.5
]
]

Ivans-MBP:app ivan$
```

FCM setup Android

5. Add to dependencies in *android/build.gradle*:

   classpath 'com.google.gms:google-services:4.3.2'

6. Add to the end of *android/app/build.gradle*:

   apply plugin: 'com.google.gms.google-services'

FCM setup Android

7. Add to AndroidManifest.xml inside activity tag:

```
<intent-filter>
    <action android:name="FLUTTER_NOTIFICATION_CLICK" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

FCM setup iOS

Steps required specifically for iOS

1. At developer.apple.com create app’s bundle id with push notification option on
2. In Keys section create Apple Push Notification service (APNs) token
3. Create provisioning profiles

FCM setup iOS

3. Create iOS app in Firebase console Settings -> General tab

4. Download GoogleService-Info.plist from Settings -> General tab iOS app section, open *ios/Runner.xcworkspace* in Xcode and add the file within Runner folder

FCM setup iOS

5. In Firebase console Settings -> Cloud Messaging under APNs Authentication Key section upload token created in previous step (key ID and Team ID required)

FCM setup iOS

6. In Xcode, select Runner in the Project Navigator, in the Signing & Capabilities Tab turn on Push Notifications and enable Background fetch and Remote notifications under Background Modes.

FCM setup iOS

7. In AppDelegate.swift add:

```
if #available(iOS 10.0, *) {
  UNUserNotificationCenter.current().delegate = self as?
UNUserNotificationCenterDelegate
}
```

FCM Flutter implementation

- add dependency to pubspec.yaml
- **firebase_messaging: 6.0.1**

- init FirebaseMessaging object on app startup

```
@override
void initState() {
 super.initState();
 _firebaseMessaging = FirebaseMessaging();
}
```

FCM (notification) Token

- get notification token from firebase
- store token on server to be associated with user

```
_firebaseMessaging.getToken()
    .then((String token) => _api.sendToken(token))
    .catchError((dynamic error)) => _handleError(error));
```

- handle notification distribution on server

FCM (notification) Token

- token can change in certain conditions
- listen for token changes via stream

```
StreamSubscription<String> tokenSubscription = _firebaseMessaging. onTokenRefresh
    .listen((String token) => _api.sendToken(token);
```

- Good practice: get token once when app starts, listen for changes and update as necessary

iOS notification permission

- iOS requires user permission before receiving notifications in app

```
_firebaseMessaging.requestNotificationPermissions(
      IosNotificationSettings(sound: true, badge: true, alert: true));

_firebaseMessaging.onIosSettingsRegistered.listen((IosNotificationSettings
settings) {
    // settings registered, we can proceed
});
```

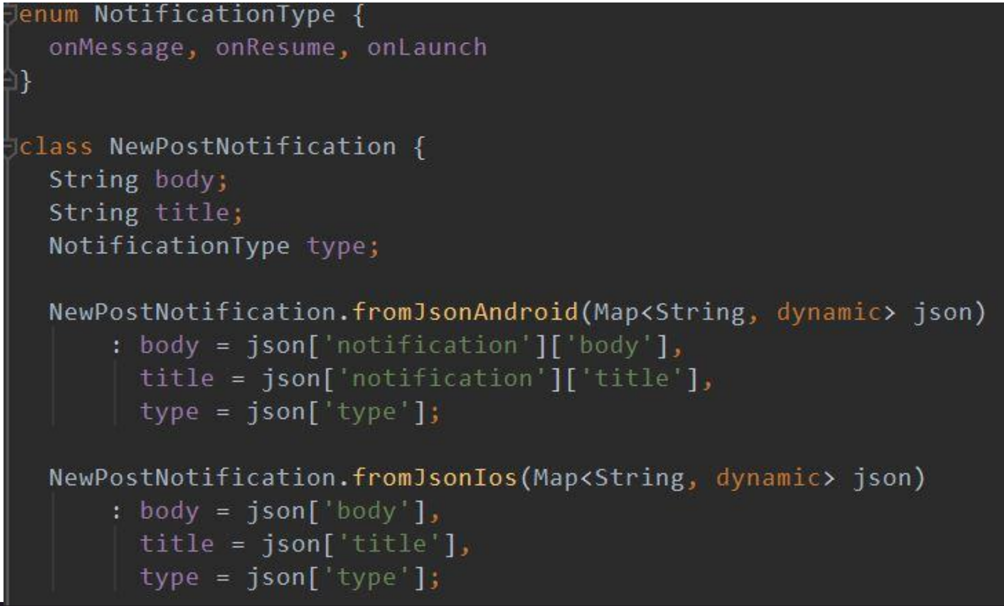
Receive notifications in app

```
_firebaseMessaging.configure(
     onMessage: (Map<String, dynamic> notification) {
       // called if notification is received in-app
     },
     onResume: (Map<String, dynamic> notification) {
       // called if notification is received when app is hidden
     },
     onLaunch: (Map<String, dynamic> notification) {
       // called when app is launched by click on system notification
     },
   );
```

Parse for iOS / Android

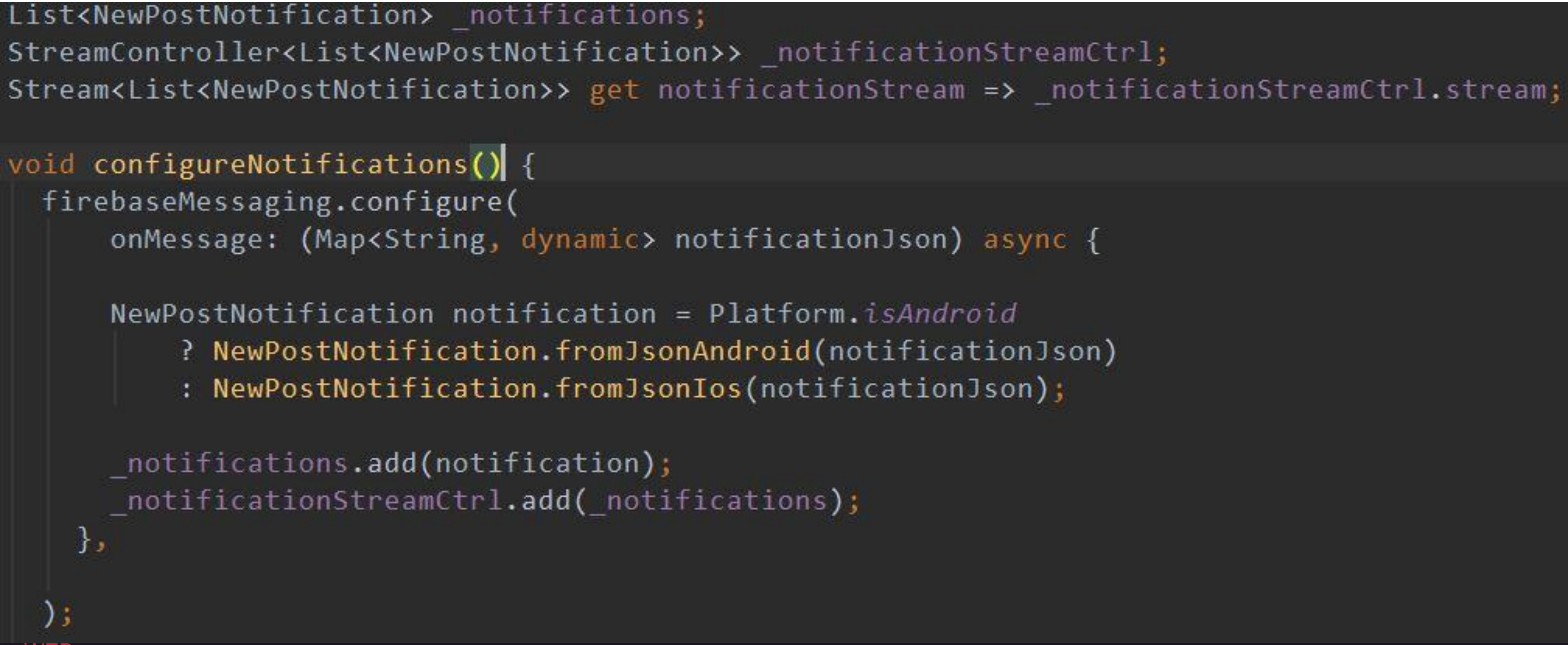
- raw notification in JSON format, Dart handles json as Map<String, dynamic> key value pairs
- add key value pair to differentiate notification type (onMessge, etc.)

```
{
    notification : {
        title : title,
        body : body
    },
    data : {...}
    type : {...}
}
```



```
enum NotificationType {
  onMessage, onResume, onLaunch
}

class NewPostNotification {
  String body;
  String title;
  NotificationType type;

  NewPostNotification.fromJsonAndroid(Map<String, dynamic> json)
      : body = json['notification']['body'],
        title = json['notification']['title'],
        type = json['type'];

  NewPostNotification.fromJsonIos(Map<String, dynamic> json)
      : body = json['body'],
        title = json['title'],
        type = json['type'];
```

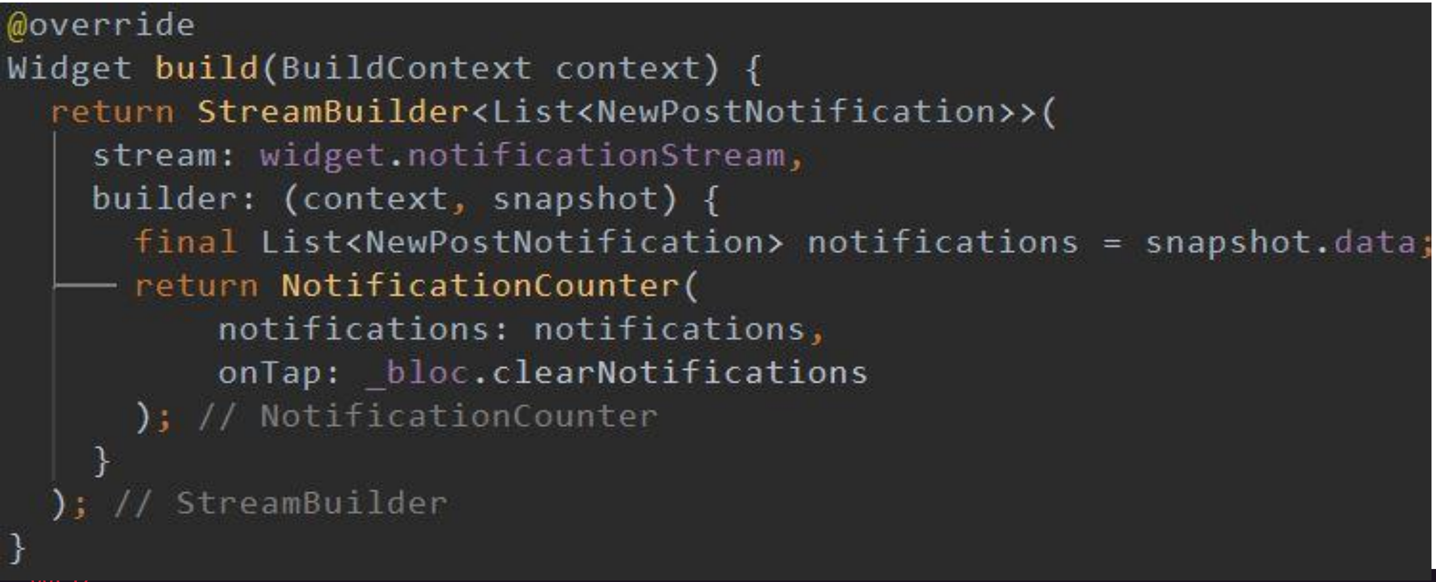
BLoC implementation



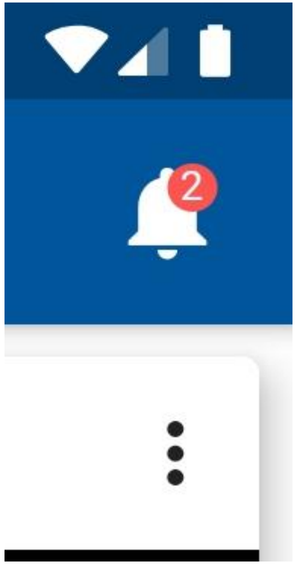
```
List<NewPostNotification> _notifications;
StreamController<List<NewPostNotification>> _notificationStreamCtrl;
Stream<List<NewPostNotification>> get notificationStream => _notificationStreamCtrl.stream;

void configureNotifications() {
  firebaseMessaging.configure(
      onMessage: (Map<String, dynamic> notificationJson) async {

      NewPostNotification notification = Platform.isAndroid
          ? NewPostNotification.fromJsonAndroid(notificationJson)
          : NewPostNotification.fromJsonIos(notificationJson);

      _notifications.add(notification);
      _notificationStreamCtrl.add(_notifications);
    },

  );
```

UI implementation

- reactive notification counter update with StreamBuilder



```
@override
Widget build(BuildContext context) {
  return StreamBuilder<List<NewPostNotification>>(
    stream: widget.notificationStream,
    builder: (context, snapshot) {
      final List<NewPostNotification> notifications = snapshot.data;
      return NotificationCounter(
          notifications: notifications,
          onTap: _bloc.clearNotifications
      ); // NotificationCounter
    }
  ); // StreamBuilder
}
```



Thank you

Ivan Celija: ivan@dream-implementation.com

Goran Kovač: goran@dream-implementation.com