dream

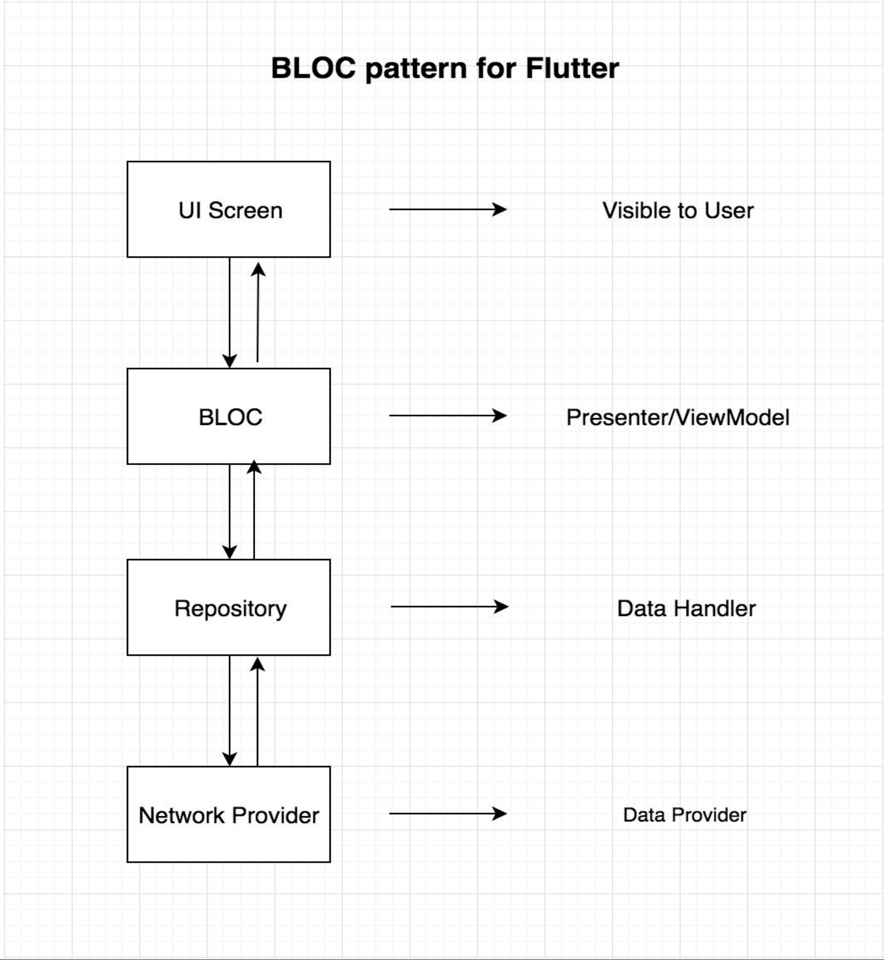
IMPLEMENTATION

http://dream-implementation.com

info@dream-implementation.com

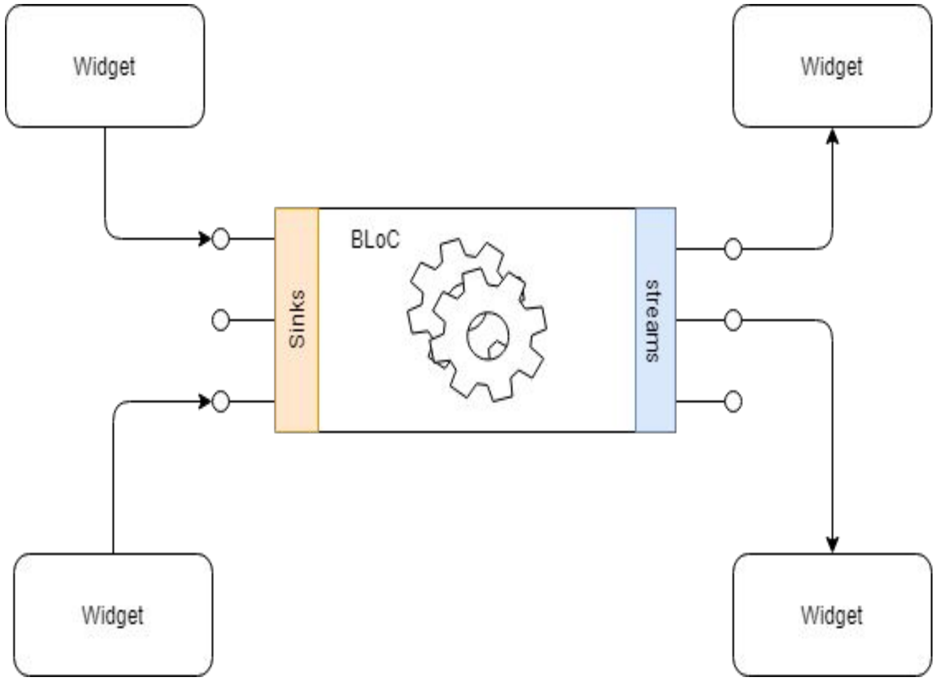
BLoC a.k.a **B**usiness **L**ogic **C**omponents

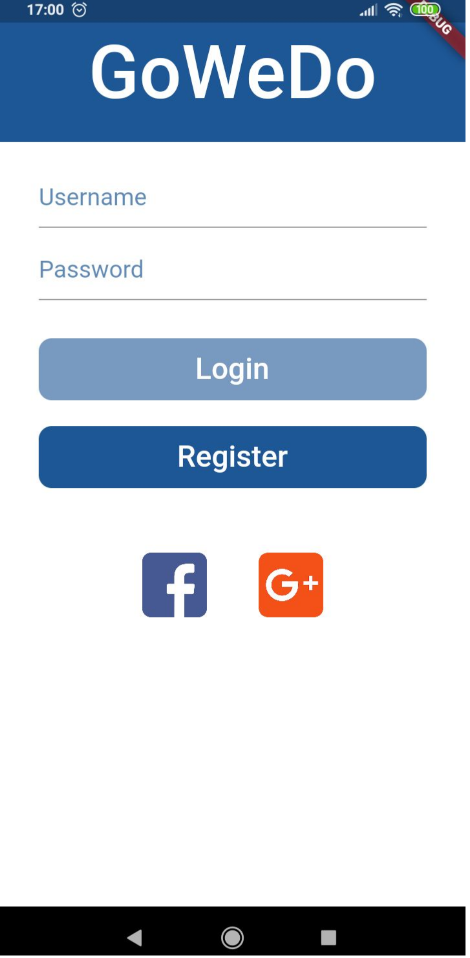
- state management system for Flutter recommended by Google
- no need for external libraries
- separates business logic from the UI
- similar to **MVP** and **MVVM**
- BLoC doesn't have any reference to UI
- UI has reference to BLoC
- aims for reactive programming without the need to call expensive **setState()** method



Streams

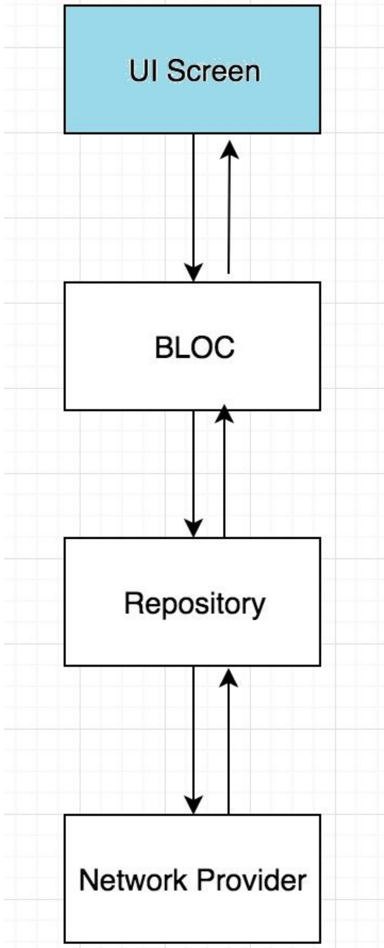
- **StreamController** controls the flow of data with stream and sink properties
- Widgets send events to the BLoC via **sinks**
- BLoC notifies the widgets via **streams**
- **StreamBuilder** rebuilds the widget tree according to the data or error pushed to the stream
- **StreamTransformer** transforms input and output data of a stream
- **RxDart** offers extended stream functionality, compatible with Dart streams





Login Screen

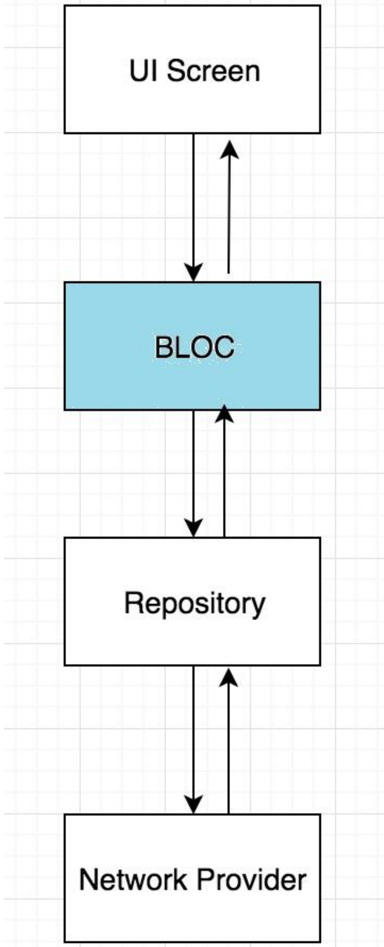
```
Scaffold
  SingleChildScrollView
    StreamBuilder<LoginState>
      Column
        StreamBuilder<String>
          TextField(username)
        StreamBuilder<String>
          TextField(password)
        CupertinoButton(login)
        CupertinoButton(register)
        Row
          CupertinoButton(facebook)
          CupertinoButton(google)
```



Login BLoC

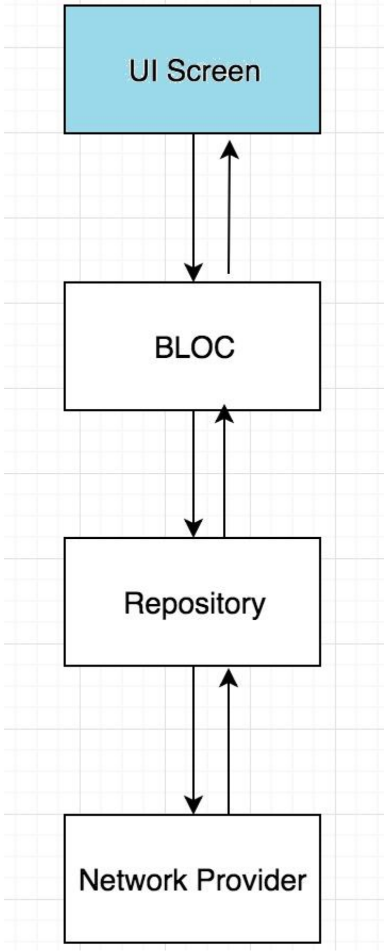
- business logic happens here,
- communication between UI and Repository

```
enum LoginState {idle, inProgress, finished}

StreamController<LoginState> _loginStateController;

Stream<LoginState> get loginStateStream => _loginStateController.stream;

void onLoginTapped() {
 _loginStateController.add(LoginState.inProgress); // show loading dialog
 _loginRepository.login(
      username: _usernameController.value,
      password: _passwordController.value)
   .then(() => _loginStateController.add(LoginState.finished))//go to MainScreen
   .catchError((error) => _loginStateController.addError(error)); //show error
}
```



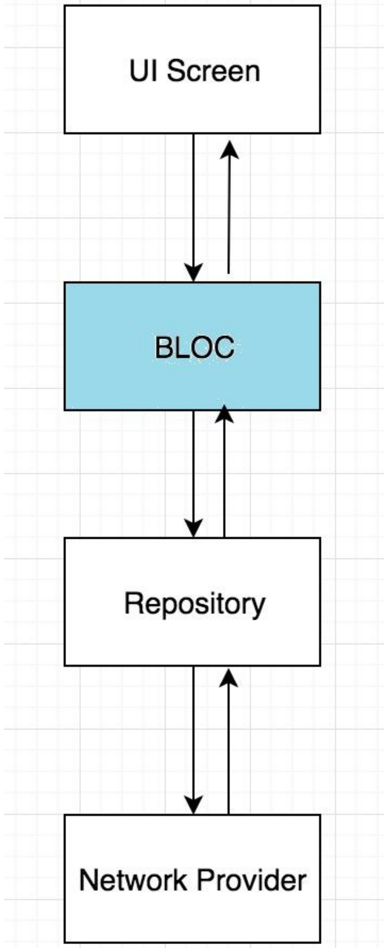
StreamBuilder<LoginState>

```
StreamBuilder<LoginState>(
  stream: _bloc.loginStateStream,
  initialData: LoginState.idle
  builder: (BuildContext context, AsyncSnapshot<LoginState> snapshot) {
    if (snapshot.hasError) {
     _showErrorDialog(snapshot.error);

    } else if (snapshot.hasData) {
     switch (snapshot.data) {
      case LoginState.inProgress: _showProgressDialog();
         break;
      case LoginState.idle: _dismissProgressDialog();
         break;
      case LoginState.finished: _navigateToMainScreen();
         break;
    }
    return _buildLoginForm(snapshot);
  },
```



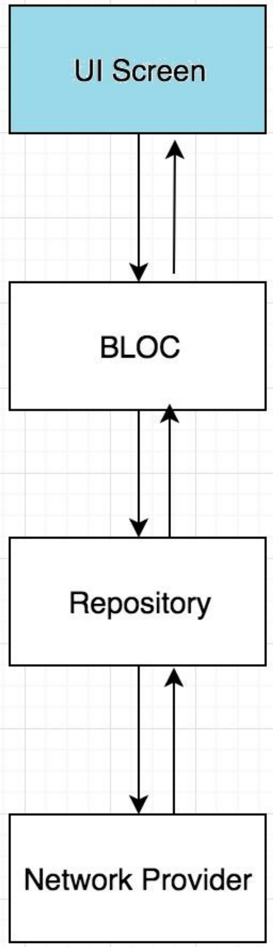
Login BLoC

```
BehaviorSubject<String> _usernameController;  //RxDart StreamController

Function(String) get onUsernameChanged =>  _usernameController.sink.add;

Stream<String> get usernameStream =>
_usernameController.stream.transform(StreamTransformer<String,String>
      .fromHandlers(handleData: (String username, EventSink<String> sink) =>
      username.isNotEmpty
            ? sink.add(username)
            : sink.addError("please enter username")
));
```



StreamBuilder<String>

```
StreamBuilder<String>(
 stream: _loginBloc.usernameStream, // stream from Bloc to Widget
 builder: (BuildContext context, AsyncSnapshot<String> snapshot) {
   return TextField(
     focusNode: _usernameFocusNode,
     keyboardType: TextInputType.emailAddress,
     onChanged: _loginBloc.onUsernameChanged, // sink from Widget to Bloc
     onSubmitted: (_) =>
FocusScope.of(context).requestFocus(_passwordFocusNode),
     decoration: InputDecoration(
       hintText: MyLocalization.of(context).username,
       errorText: snapshot.error
     )
   );
 }
),
```

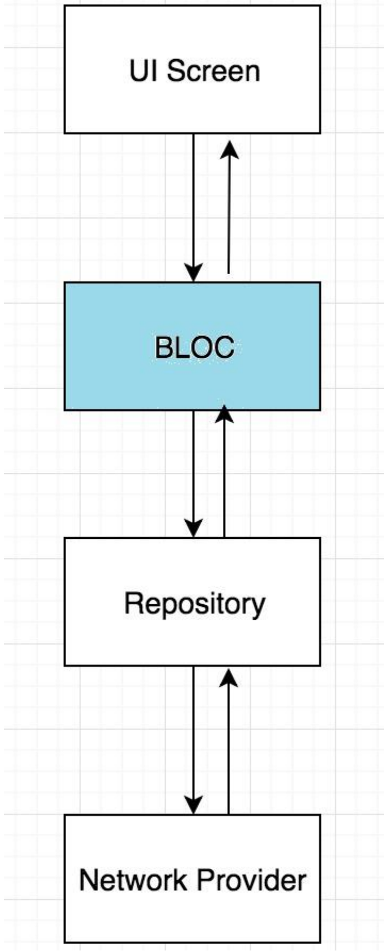


Don’t forget to dispose (use StatefulWidget or BlocProvider):

```
void dispose() {
 _loginStateController?.close();
 _usernameController?.close();
 _passwordController?.close();
}
```

Call navigation / show dialogs after build completes:

```
WidgetsBinding.instance.addPostFrameCallback((_)=> showLoadingDialog());
```



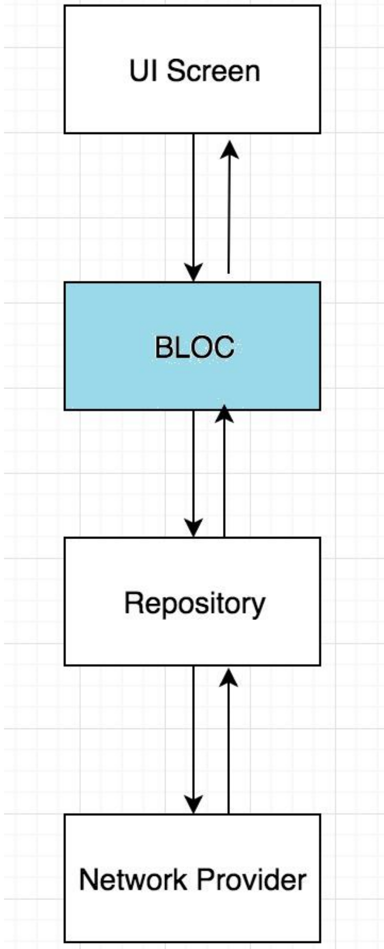
UI		BLOC

Login state stream - listens for login button and updates the screen if necessary, handles navigation to main screen

StreamBuilder<LoginState>	⇐	StreamController<LoginState>
LoginButton	⇒	*.add()* or*.addError()*

User input streams - take user input, validate it and update TextFields

StreamBuilder<String>	⇐	BehaviorSubject<String>
TextField(username)	⇒	*.sink.add()* or *.sink.addError()*
StreamBuilder<String>	⇐	BehaviorSubject<String>
TextField(password)	⇒	*.sink.add()* or *.sink.addError()*

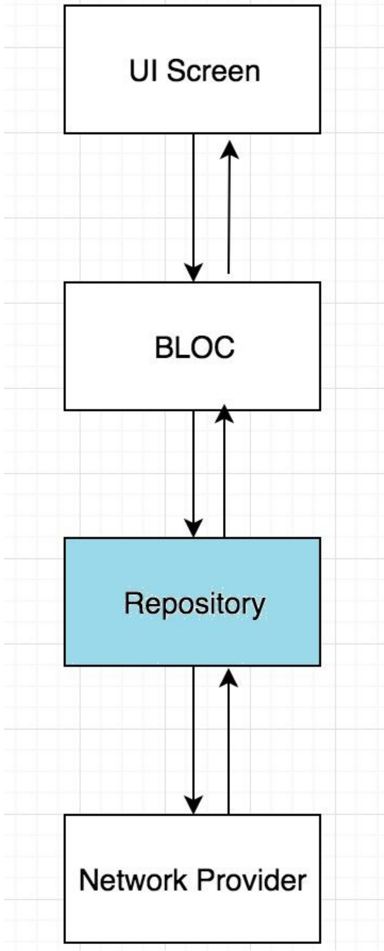


Repository

- handles data from various providers and forwards it to BLoC
- organized by functionality or screens (LoginRepository, etc.)

```
Future<String> login({String username, String password}) {
     return GoWeDo.api.login(username: username, password: password);
}

Future<Null> saveSecurityToken(String securityToken) {
    return GoWeDo.localStorage.setSecurityToken(securityToken);
}
```

- shared_preferences 0.5.3 plugin

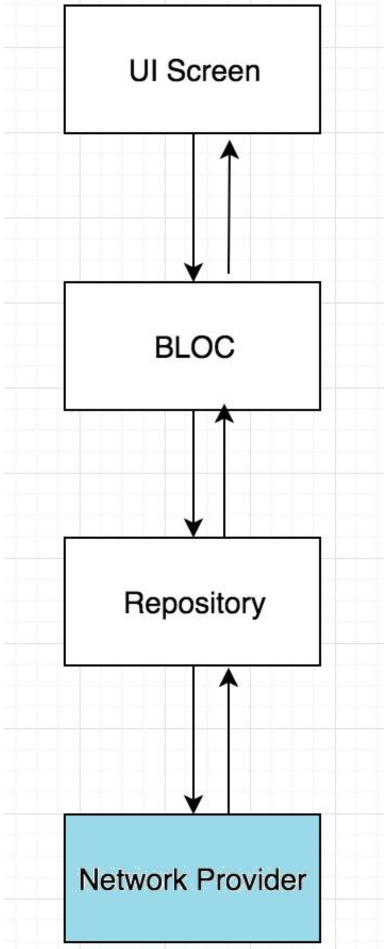


Network Provider

- handles Api calls

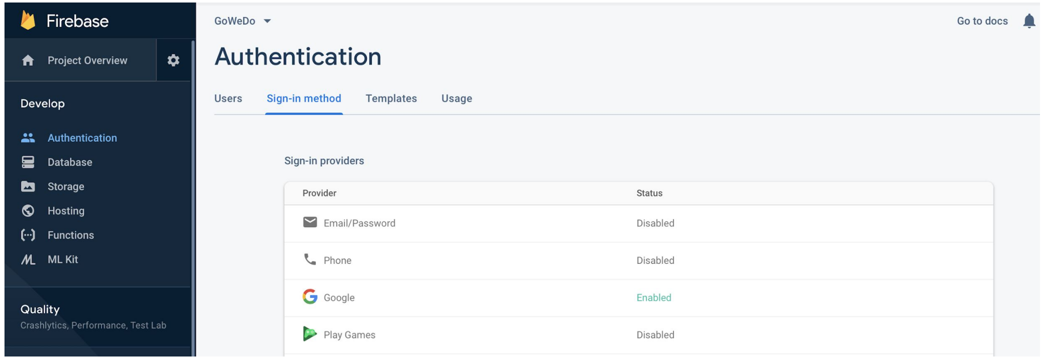
```
Future<String> login({String username, String password}) {
 final Map<String, dynamic> bodyMap =
<String, dynamic>{'username': username, 'password': password};

 return apiClient.request<Map<String, dynamic>>(Config(
   uri: Uri.parse('$server/login/'),
   body: RequestBody.json(bodyMap),
   method: RequestMethod.post,
   responseType: ResponseBody.json()
 )).then((Map<String, dynamic> jsonResponse) => jsonResponse['token']);
}
```

- http 0.12.0 plugin



Google sign-in

- Steps required for both platforms:
  1. Create new project in Firebase console
  2. Enable Google in Authentication -> Sign-in method



Provider	Status
Email/Password	Disabled
Phone	Disabled
Google	Enabled
Play Games	Disabled

Google sign-in

3. Add google_sign_in plugin (https://pub.dev/packages/google_sign_in) to pubspec.yaml

Google sign-in - Android

Steps required specifically for Android:

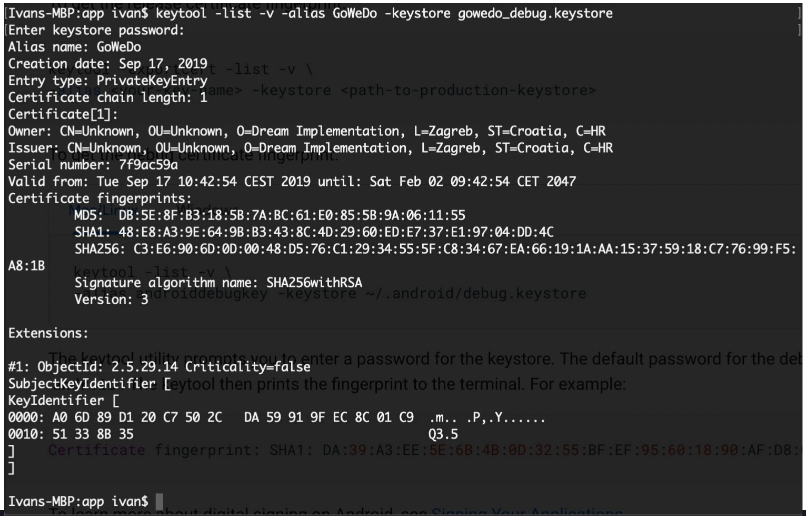
1. Create Android app in Firebase console Settings -> General tab
2. Download google-services.json from Settings -> General tab Android app section and put in *android/app/* folder

Google sign-in - Android

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

Google sign-in - Android

4. 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$ 
```

Google sign-in - 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'

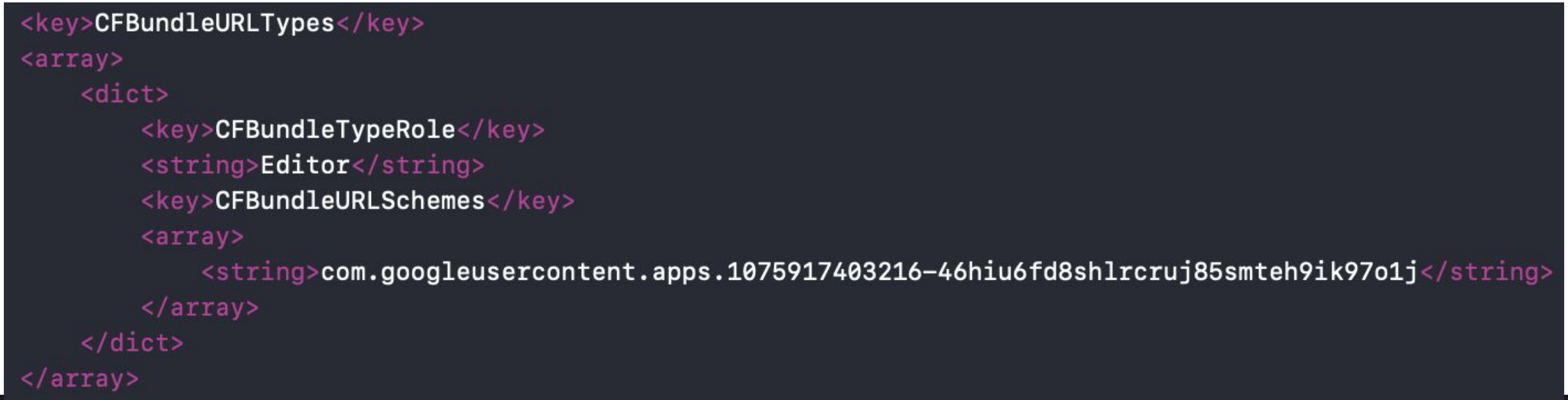
Google sign-in - iOS

Steps required specifically for iOS

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

Google sign-in - iOS

3. Open Info.plist in Xcode as Source Code and add these lines:



```
<key>CFBundleURLTypes</key>
<array>
    <dict>
        <key>CFBundleTypeRole</key>
        <string>Editor</string>
        <key>CFBundleURLSchemes</key>
        <array>
            <string>com.googleusercontent.apps.1075917403216-46hiu6fd8shlrcruj85smteh9ik97o1j</string>
        </array>
    </dict>
</array>
```

Google sign-in

Plugin can be used with this code:

```
final GoogleSignIn _googleSignIn = GoogleSignIn(scopes: ['email']);
try {
        GoogleSignInAccount googleSignInAccount = await _googleSignIn.signIn();
        GoogleSignInAuthentication googleSignInAuthentication = await
googleSignInAccount?.authentication;
        print(googleSignInAuthentication?.idToken);
} catch (error) {
        print(error);
}
```

Facebook sign-in

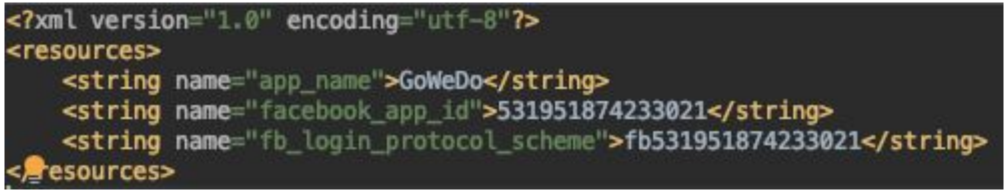
Steps required for both platforms:

1. Create new app at developers.facebook.com
2. Add flutter_facebook_login plugin (https://pub.dev/packages/flutter_facebook_login) to pubspec.yaml

Facebook sign-in - Android

Steps required specifically for Android (https://developers.facebook.com/docs/facebook-login/android):

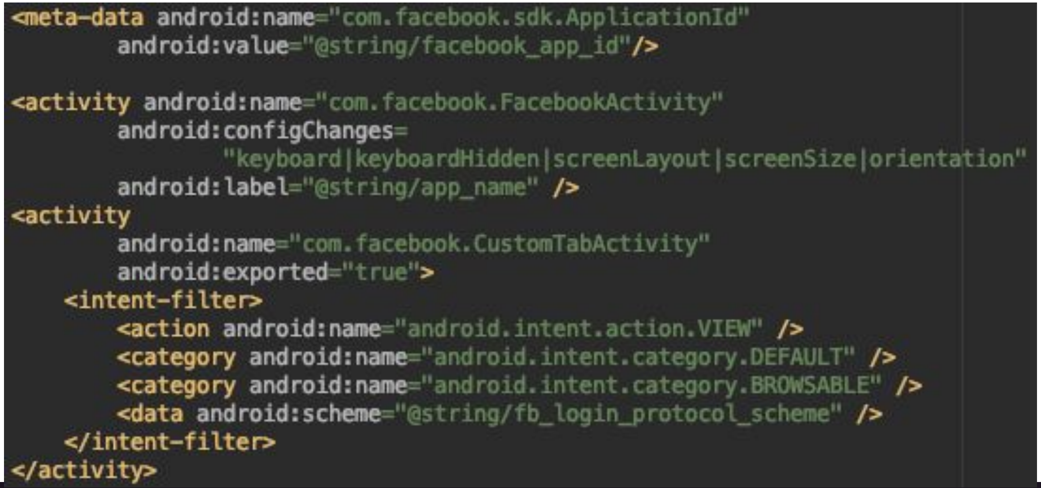
1. Create strings.xml in *android/app/src/main/res/values/* folder and put



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">GoWeDo</string>
    <string name="facebook_app_id">531951874233021</string>
    <string name="fb_login_protocol_scheme">fb531951874233021</string>
</resources>
```

Facebook sign-in - Android

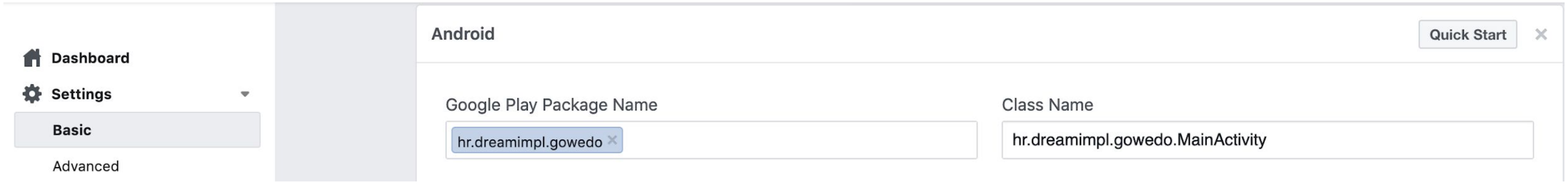
2. In *android/app/src/main/AndroidManifest.xml* inside <*application*> tag put:



```
<meta-data android:name="com.facebook.sdk.ApplicationId"
        android:value="@string/facebook_app_id"/>

<activity android:name="com.facebook.FacebookActivity"
        android:configChanges=
                "keyboard|keyboardHidden|screenLayout|screenSize|orientation"
        android:label="@string/app_name" />
<activity
        android:name="com.facebook.CustomTabActivity"
        android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
        <data android:scheme="@string/fb_login_protocol_scheme" />
    </intent-filter>
</activity>
```

Facebook sign-in - Android

3. Add Package Name and Default Activity Class Name in Settings -> Basic Android section



Google Play Package Name: hr.dreamimpl.gowedo

Class Name: hr.dreamimpl.gowedo.MainActivity

Facebook sign-in - Android

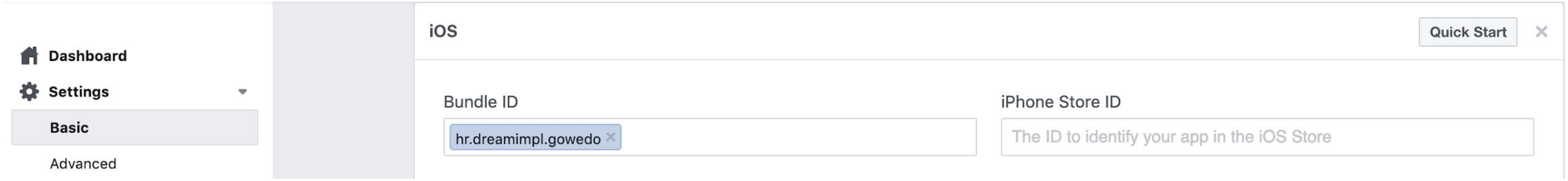
4. Generate debug and release keystore hashes and put to Key Hashes in Settings -> Basic Android section

keytool -exportcert -alias GoWeDo -keystore gowedo_debug.keystore | openssl sha1 -binary | openssl base64

Facebook sign-in - iOS

Steps required specifically for iOS

1. Register iOS Bundle ID in Settings -> Basic iOS section

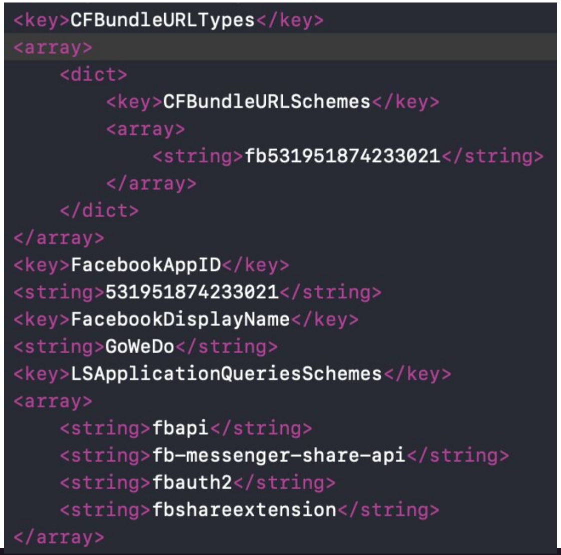


Bundle ID: hr.dreamimpl.gowedo

iPhone Store ID: The ID to identify your app in the iOS Store

Facebook sign-in - iOS

2. Open Info.plist in Xcode as Source Code and add these lines:



```xml
<key>CFBundleURLTypes</key>
<array>
    <dict>
        <key>CFBundleURLSchemes</key>
        <array>
            <string>fb531951874233021</string>
        </array>
    </dict>
</array>
<key>FacebookAppID</key>
<string>531951874233021</string>
<key>FacebookDisplayName</key>
<string>GoWeDo</string>
<key>LSApplicationQueriesSchemes</key>
<array>
    <string>fbapi</string>
    <string>fb-messenger-share-api</string>
    <string>fbauth2</string>
    <string>fbshareextension</string>
</array>
```

Facebook sign-in

Plugin can be used with this code:

```
final FacebookLogin _facebookLogin = FacebookLogin();
final FacebookLoginResult result = await _facebookLogin.logIn(<String>['email']);
switch (result.status) {
  case FacebookLoginStatus.loggedIn:
        print(result.accessToken?.token);
        break;
  case FacebookLoginStatus.cancelledByUser:
        break;
  case FacebookLoginStatus.error:
        print('error: ${result?.errorMessage}');
        break;
}
```

Thank you

Ivan Celija: ivan@dream-implementation.com

Goran Kovač: goran@dream-implementation.com