

**PRÁCTICA DE LABORATORIO 6 – DESARROLLO DE APLICACIÓN
MÓVIL HÍBRIDA CON FLUTTER USANDO CONEXIÓN A BASES DE DATOS
EXTERNAS**

Profesora: Msig. Adriana Collaguazo

Paralelo: 1

Grupo: 4

- Debbie Donoso
- Guillermo Merizalde
- Jefferson Vega
- Cesar Vera

Enlace del Repositorio: https://github.com/Daniella252018/lab6_amst

Fecha de inicio: 5 de agosto 2022

Fecha de finalización: 18 de agosto 2022

CLASE MAIN

```
8
9  class MyApp extends StatefulWidget {
10  _AppState createState() => _AppState();
11
12  }
13
14  class _AppState extends State<MyApp>{
15      final GlobalKey<NavigatorState> navigatorKey = new GlobalKey<NavigatorState>();
16
17      @override
18      void initState() {
19          super.initState();
20      }
21
22      @override
23      Widget build(BuildContext context) {
24          return MaterialApp(
25              title: 'Vacheck',
26              navigatorKey: navigatorKey,
27              initialRoute: 'nav',
28              routes: {
29                  'nav': (context) => Nav(),
30              },
31              theme: ThemeData(
32                  primaryColor: Colors.red[400],
33              ), // ThemeData
34          ); // MaterialApp
35      }
36  }
37
```

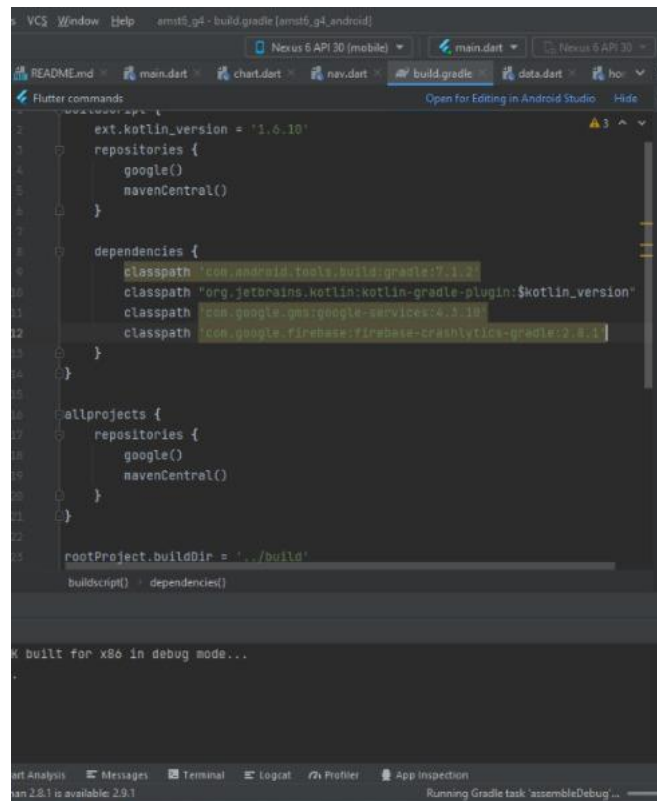
Figure 1 En esta parte del código hacemos el cambio para que extienda de `StatefulWidget` y posteriormente creamos la clase `_AppState` donde haremos uso de `MaterialApp` en donde podemos hacer uso routes para colocar los iconos de navegación en las diferentes páginas.

CLASE HOMEPAGE

```
12      @override
13      Widget build(BuildContext context){
14          return Scaffold(
15              appBar: AppBar(
16                  title: Text('Vacheck'),
17              ), // AppBar
18              body: _llamarPagina(_currentIndex),
19              bottomNavigationBar: _crearNavigationBar(),
20          ); // Scaffold
21      }
22
23      Widget _crearNavigationBar(){
24          return BottomNavigationBar(
25              unselectedItemColor: Color.fromARGB(168, 97, 93, 4),
26              iconSize: 30.0,
27              unselectedIconTheme: IconThemeData(size: 25.0),
28              currentIndex: _currentIndex,
29              onTap: (index){
30                  setState(() {
31                      _currentIndex = index;
32                  });
33              },
34              items: [
35                  BottomNavigationBarItem(
36                      icon: Icon(Icons.data_usage), label: 'Data'), // BottomNavigationBarItem
37                  BottomNavigationBarItem(
38                      icon: Icon(Icons.home), label: 'Home'), // BottomNavigationBarItem
39                  BottomNavigationBarItem(
40                      icon: Icon(Icons.bar_chart), label: 'Chart') // BottomNavigationBarItem
41              ]; // BottomNavigationBar
42      }
```

Figure 2 Se hace la creación de los botones de navegación en el Widget _crearNavigationBar.

DEPENDENCIAS NECESARIAS PARA EJECUTAR APLICACIÓN



```
ext.kotlin_version = '1.6.10'

repositories {
    google()
    mavenCentral()
}

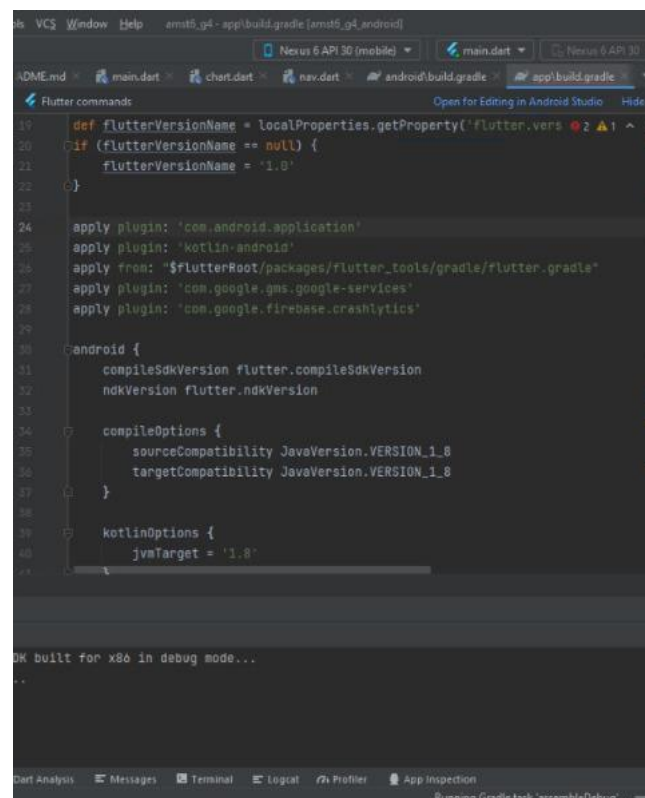
dependencies {
    classpath 'com.android.tools.build:gradle:7.1.2'
    classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    classpath 'com.google.gms:google-services:4.3.10'
    classpath 'com.google.firebase:firebase-crashlytics-gradle:2.8.1'
}

allprojects {
    repositories {
        google()
        mavenCentral()
    }
}

rootProject.buildDir = '../build'
buildscript {
    dependencies {

```

PLUGINS NECESARIOS PARA EJECUTAR APLICACIÓN



```
def flutterVersionName = localProperties.getProperty('flutter.versionName')
if (flutterVersionName == null) {
    flutterVersionName = '1.0'
}

apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply from: "$flutterRoot/packages/flutter_tools/gradle/flutter.gradle"
apply plugin: 'com.google.gms.google-services'
apply plugin: 'com.google.firebase.crashlytics'

android {
    compileSdkVersion flutter.compileSdkVersion
    ndkVersion flutter.ndkVersion

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }

    kotlinOptions {
        jvmTarget = '1.8'
    }
}
```

CLASE CHART

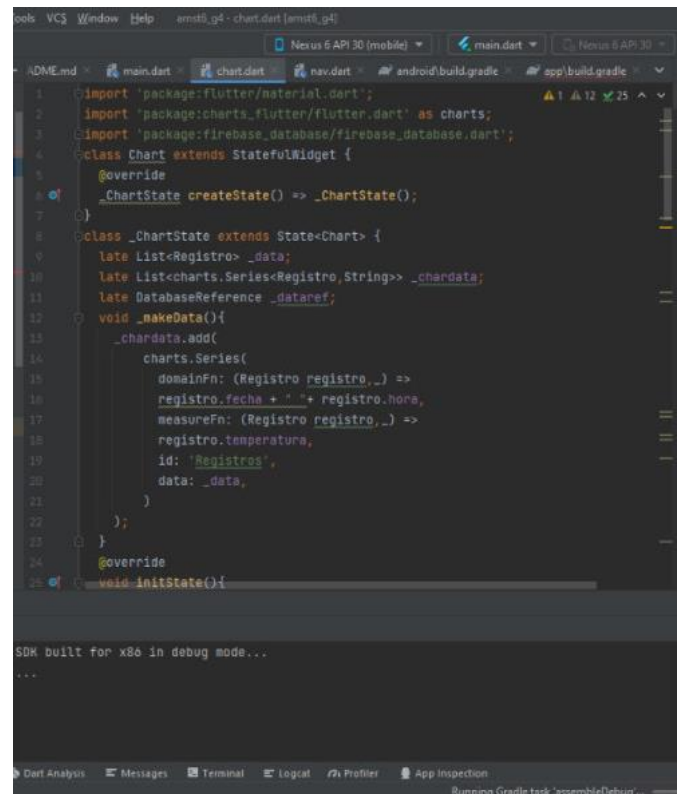


Ilustración 1. Permite crear las barras con las temperaturas añadidas a la base

CLASE DATA

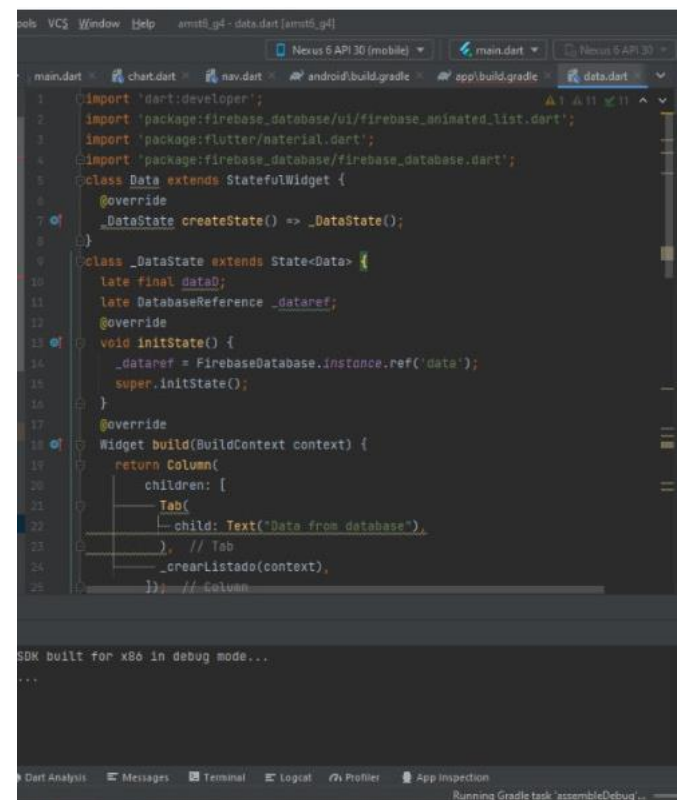


Ilustración 2. Muestra todos los datos de las temperaturas añadidas a la base, junto con la fecha en donde fueron agregadas

CLASE NAT

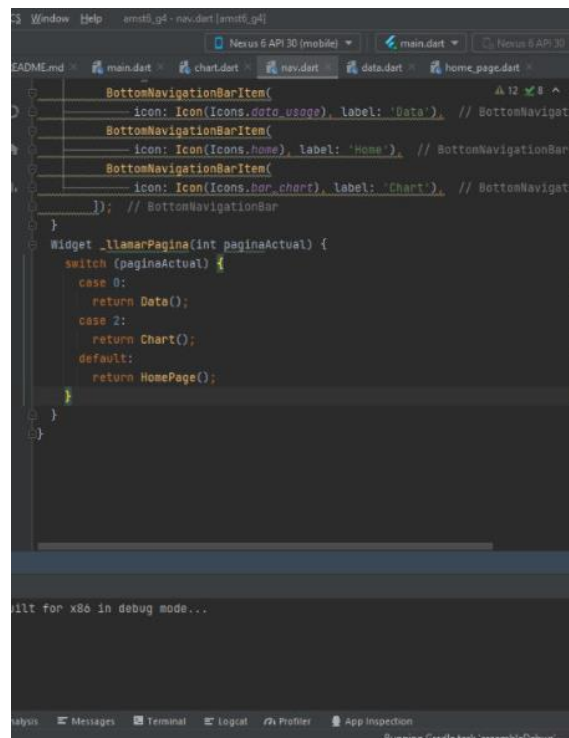


Ilustración 3. Permite que se muestren los datos correspondientes en cada ventana de la aplicación

RESULTADOS FINALES

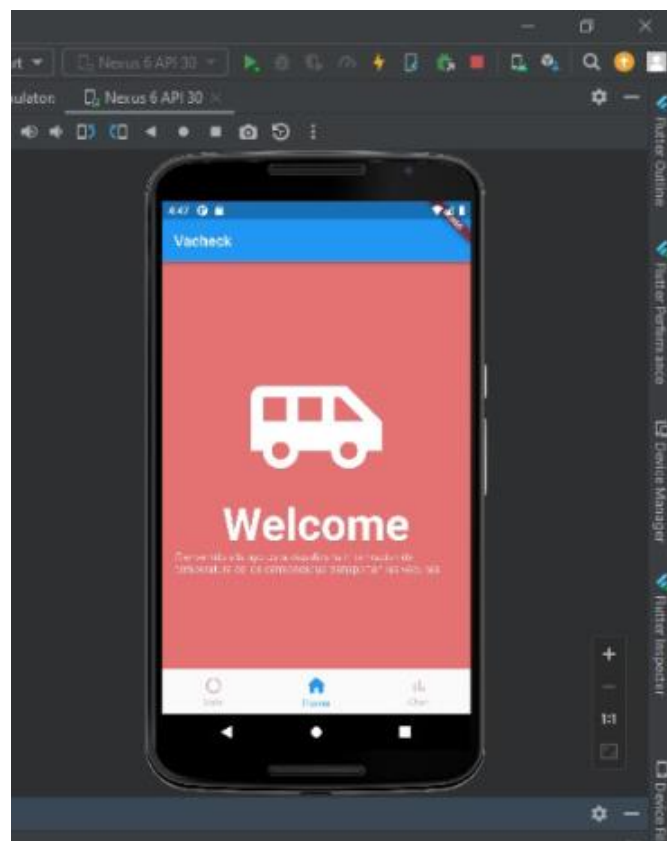


Ilustración 4. HOME

TAREA DESAFIO 1.

Investigar

a. ¿Qué otros frameworks tienen características similares a flutter?

- Ionic.
- React Native.
- Xamarin.
- PhoneGap.
- Native Script.
- Appcelerator Titanium.
- jQuery Móvil.

b. ¿De qué otras maneras pueden mostrar datos de forma gráfica?

Flutter nos permite trabajar con distintos tipos de graficos para representar datos, entre esto tenemos los graficos lineales, graficos de barras horizontales y verticales, graficos circulares, y graficos de dispersión.

CONCLUSIONES / RECOMENDACIONES

Conclusiones

- Se realizo la programación de una app que muestra información de la base de datos firebase de manera fácil mediante la utilización de los distintos Widget que nos ofrece Flutter.
- Se pudo observar que la integración de Firebase con un proyecto creado en flutter se la hace de una forma sencilla al ejecutar los comandos `flutterfire configure` y `flutter pub add firebase_core`.
- Al utilizar el comando `flutterfire configure` se pudo observar cómo realizaba la instalación para los dos sistemas operativos, Android y IOS, teniendo como ventaja que el comando permita realizar las instalaciones en ambos sistemas.

Recomendaciones

- Se recomienda tener un equipo con las características mínimas de hardware para que pueda soportar y trabajar de forma correcta con flutter.
- Se recomienda colocar el path de `firebase-tools` para poder hacer uso de `flutterfire_cli`.

APK:

https://github.com/Daniella252018/lab6_amst