

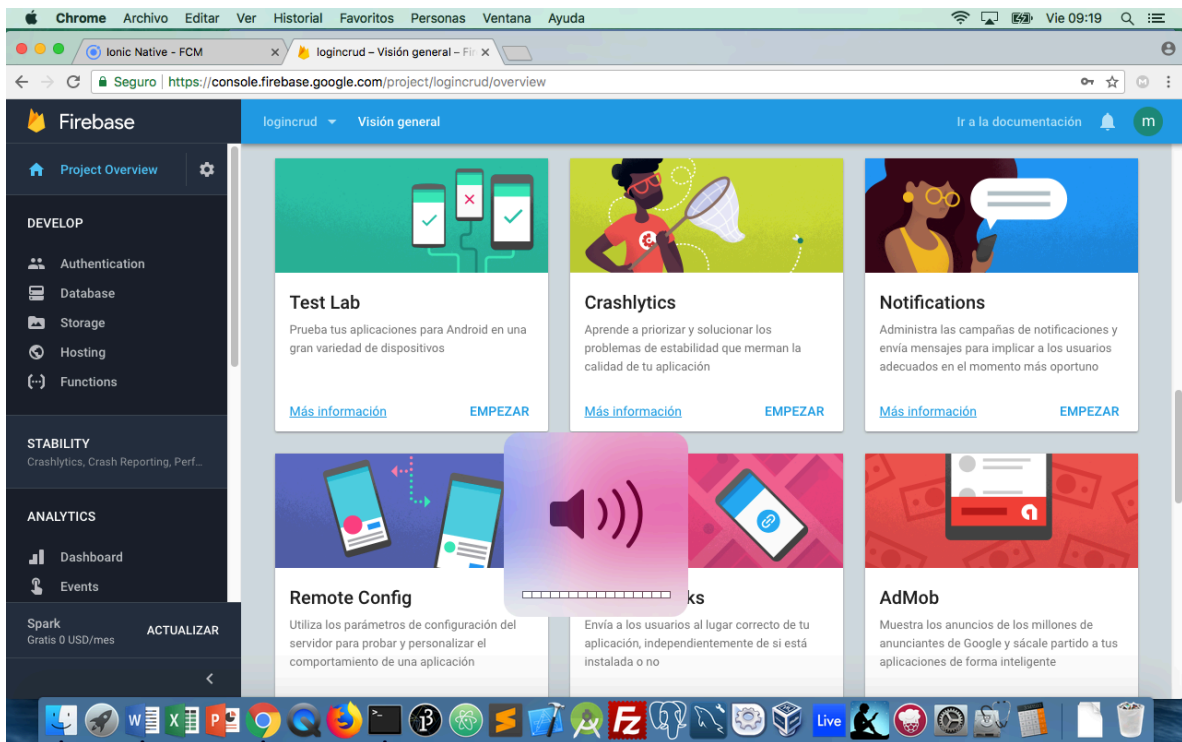
Notificaciones Ionic

José Santorcuato T

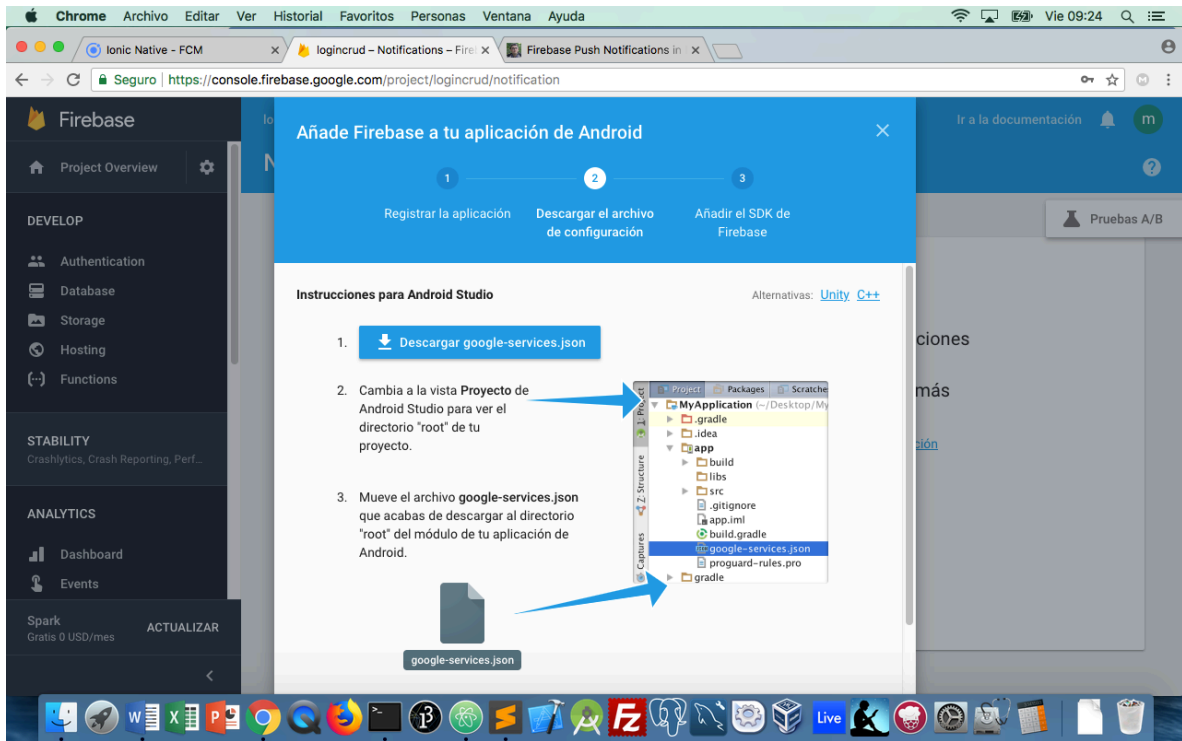
Firestore
Native
FCM

Firestore

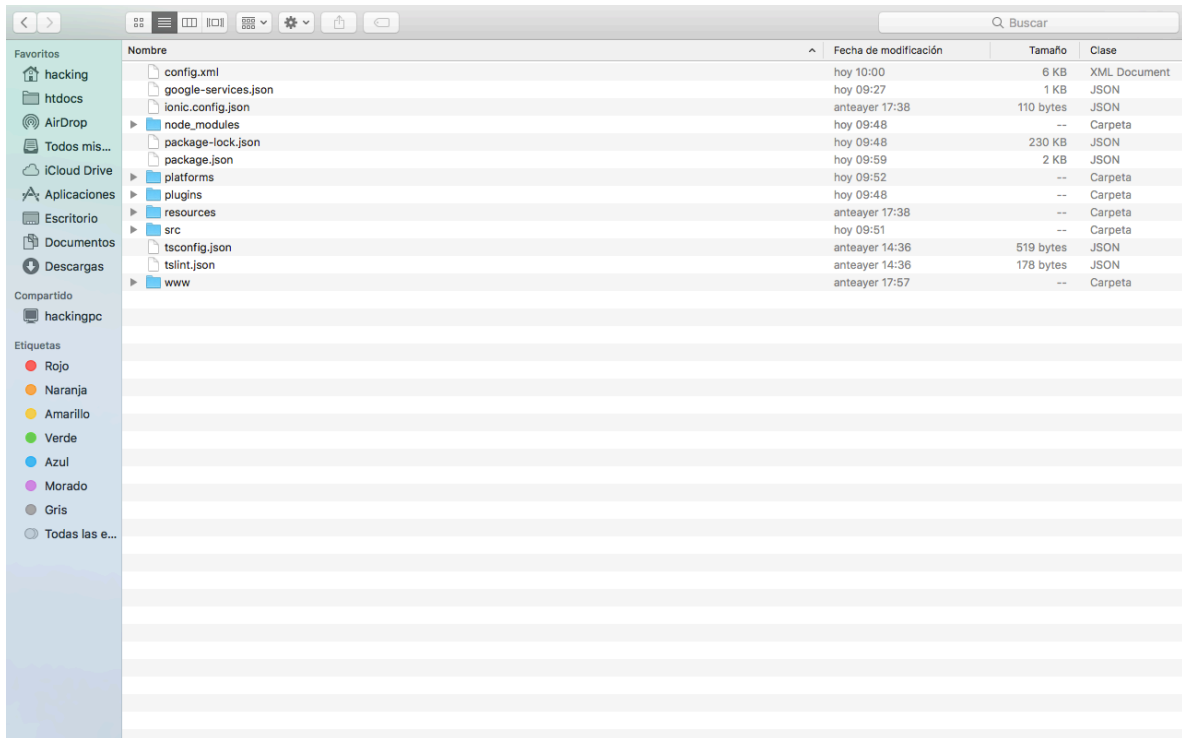
Es importante señalar que el orden de este proyecto es importante, no dará lo mismo en que orden sigamos los pasos.



En notificaciones vamos a agregar nuestra aplicación para ello pinchar sobre icono Android, luego agregar el id que ya hemos configurado en otros proyectos (config.xml) el SHA es opcional.



Guarda el archivo de configuración el google-services .json lo debemos agregar a la carpeta raíz



Configuración proyecto

```
MacBook-Pro-de-hacking:notificaciones hacking$ ionic cordova plugin add cordova-plugin-fcm
```

```
✓ Creating ./www directory for you - done!  
> cordova plugin add cordova-plugin-fcm --save  
Adding cordova-plugin-fcm to package.json
```

```
Saved plugin info for "cordova-plugin-fcm" to config.xml
```

```
npm install --save @ionic-native/fcm
```

Código

app.components.ts

```
import { Component } from '@angular/core';
import { Platform } from 'ionic-angular';
import { StatusBar } from '@ionic-native/status-bar';
import { SplashScreen } from '@ionic-native/splash-screen';
import { FCM } from '@ionic-native/fcm';

import { HomePage } from '../pages/home/home';
@Component({
  templateUrl: 'app.html'
})
export class MyApp {
  rootPage:any = HomePage;

  constructor(platform: Platform, statusBar: StatusBar, splashScreen: SplashScreen,
    private fcm: FCM) {
    platform.ready().then(() => {

      this.fcm.subscribeToTopic('all');
      this.fcm.getToken().then(token => {
        // backend.registerToken(token);
      });

      this.fcm.onNotification().subscribe(data => {
        alert('message received')
        if(data.wasTapped) {
          console.info("Received in background");
        } else {
          console.info("Received in foreground");
        }
      });
      this.fcm.onTokenRefresh().subscribe(token => {
        // backend.registerToken(token);
      });

      // Okay, so the platform is ready and our plugins are available.
      // Here you can do any higher level native things you might need.
      statusBar.styleDefault();
      splashScreen.hide();
    });
  }
}
```

app.modules.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { ErrorHandler, NgModule } from '@angular/core';
import { IonicApp, IonicErrorHandler, IonicModule } from 'ionic-angular';
import { SplashScreen } from '@ionic-native/splash-screen';
import { StatusBar } from '@ionic-native/status-bar';

import { MyApp } from './app.component';
import { HomePage } from '../pages/home/home';

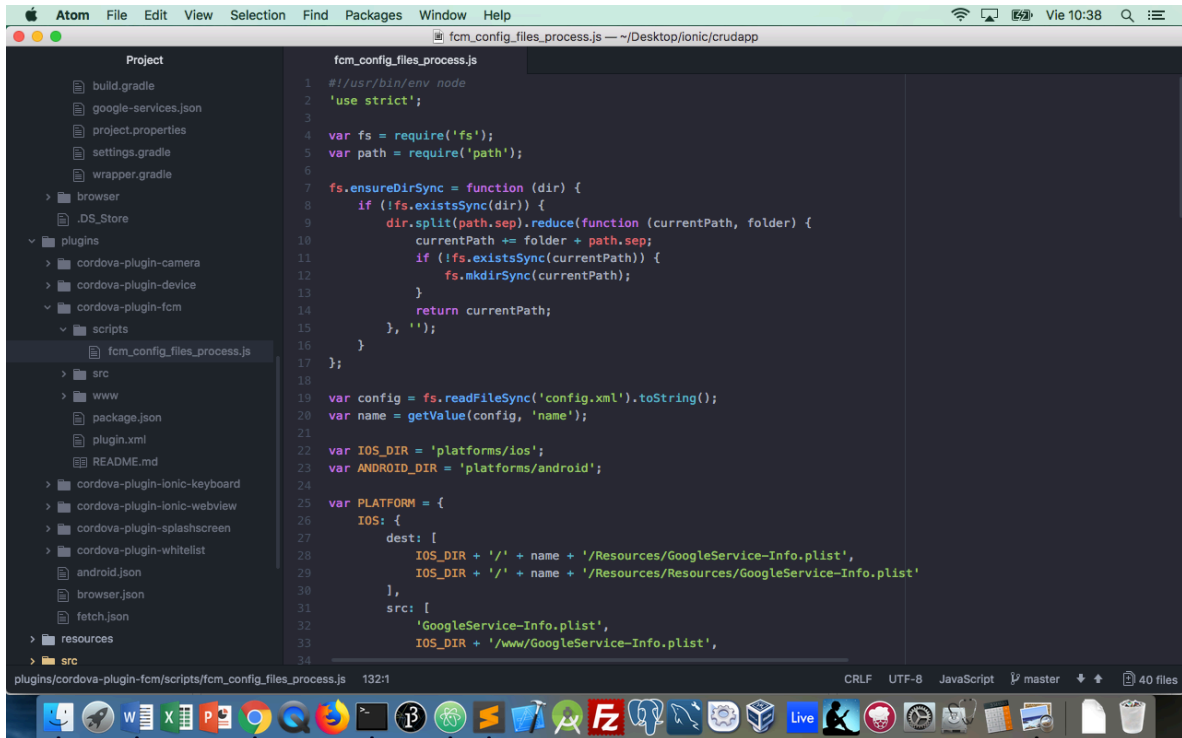
import { FCM } from '@ionic-native/fcm';
```

```
@NgModule({
  declarations: [
    MyApp,
    HomePage
  ],
  imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp)
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    HomePage
  ],
  providers: [
    StatusBar,
    SplashScreen,
    FCM,
    {provide: ErrorHandler, useClass: IonicErrorHandler}
  ]
})
export class AppModule {}
```

Tenemos un error...véalo en el próximo capítulo

Proximo capítulo

Plugins/cordova-plugin/scripts/fcm_config_files_process.js



```
fcm_config_files_process.js
1  #!/usr/bin/env node
2  'use strict';
3
4  var fs = require('fs');
5  var path = require('path');
6
7  fs.ensureDirSync = function (dir) {
8    if (!fs.existsSync(dir)) {
9      dir.split(path.sep).reduce(function (currentPath, folder) {
10        currentPath += folder + path.sep;
11        if (!fs.existsSync(currentPath)) {
12          fs.mkdirSync(currentPath);
13        }
14        return currentPath;
15      }, '');
16    }
17  };
18
19  var config = fs.readFileSync('config.xml').toString();
20  var name = getValue(config, 'name');
21
22  var IOS_DIR = 'platforms/ios';
23  var ANDROID_DIR = 'platforms/android';
24
25  var PLATFORM = {
26    IOS: {
27      dest: [
28        IOS_DIR + '/' + name + '/Resources/GoogleService-Info.plist',
29        IOS_DIR + '/' + name + '/Resources/Resources/GoogleService-Info.plist'
30      ],
31      src: [
32        'GoogleService-Info.plist',
33        IOS_DIR + '/www/GoogleService-Info.plist',
34      ]
35    }
36  };
37
38  function getValue(config, name) {
39    var start = config.indexOf(name + '=');
40    if (start < 0) return '';
41    start += name.length + 1;
42    var end = config.indexOf('<\/pre>
```

```

#!/usr/bin/env node
'use strict';

var fs = require('fs');
var path = require('path');

fs.ensureDirSync = function (dir) {
  if (!fs.existsSync(dir)) {
    dir.split(path.sep).reduce(function (currentPath, folder) {
      currentPath += folder + path.sep;
      if (!fs.existsSync(currentPath)) {
        fs.mkdirSync(currentPath);
      }
      return currentPath;
    }, '');
  }
};

var config = fs.readFileSync('config.xml').toString();
var name = getValue(config, 'name');

var IOS_DIR = 'platforms/ios';
var ANDROID_DIR = 'platforms/android';

var PLATFORM = {
  IOS: {
    dest: [
      IOS_DIR + '/' + name + '/Resources/GoogleService-Info.plist',
      IOS_DIR + '/' + name + '/Resources/Resources/GoogleService-Info.plist'
    ],
    src: [
      'GoogleService-Info.plist',
      IOS_DIR + '/www/GoogleService-Info.plist',
      'www/GoogleService-Info.plist'
    ]
  },
  ANDROID: {
    dest: [
      ANDROID_DIR + '/google-services.json',
      ANDROID_DIR + '/app/google-services.json',
    ],
    src: [
      'google-services.json',
      ANDROID_DIR + '/assets/www/google-services.json',
      'www/google-services.json'
    ],
    stringsXml: ANDROID_DIR + '/app/src/main/res/values/strings.xml'
  }
};

// Copy key files to their platform specific folders
if (directoryExists(IOS_DIR)) {
  copyKey(PLATFORM.IOS);
}
if (directoryExists(ANDROID_DIR)) {
  copyKey(PLATFORM.ANDROID, updateStringsXml)
}

function updateStringsXml(contents) {
  var json = JSON.parse(contents);
  var strings = fs.readFileSync(PLATFORM.ANDROID.stringsXml).toString();

  // strip non-default value
  strings = strings.replace(new RegExp('<string name="google_app_id">([^\@<]+?)</string>', 'i'),
  '');

  // strip non-default value
  strings = strings.replace(new RegExp('<string name="google_api_key">([^\@<]+?)</string>', 'i'),
  '');
}

```



```

// strip empty lines
strings = strings.replace(new RegExp('(\\r\\n|\\n|\\r)[ \\t]*(\\r\\n|\\n|\\r)', 'gm'), '$1');

// replace the default value
strings = strings.replace(new RegExp('<string name="google_app_id">([<]+?)</string>', 'i'),
'<string name="google_app_id">' + json.client[0].client_info.mobilesdk_app_id + '</string>');

// replace the default value
strings = strings.replace(new RegExp('<string name="google_api_key">([<]+?)</string>', 'i'),
'<string name="google_api_key">' + json.client[0].api_key[0].current_key + '</string>');

fs.writeFileSync(PLATFORM.ANDROID.stringsXml, strings);
}

function copyKey(platform, callback) {
  for (var i = 0; i < platform.src.length; i++) {
    var file = platform.src[i];
    if (fileExists(file)) {
      try {
        var contents = fs.readFileSync(file).toString();

        try {
          platform.dest.forEach(function (destinationPath) {
            var folder = destinationPath.substring(0, destinationPath.lastIndexOf('/'));
            fs.ensureDirSync(folder);
            fs.writeFileSync(destinationPath, contents);
          });
        } catch (e) {
          // skip
        }

        callback && callback(contents);
      } catch (err) {
        console.log(err)
      }

      break;
    }
  }
}

function getValue(config, name) {
  var value = config.match(new RegExp('<' + name + '>(.*?)</' + name + '>', 'i'));
  if (value && value[1]) {
    return value[1]
  } else {
    return null
  }
}

function fileExists(path) {
  try {
    return fs.statSync(path).isFile();
  } catch (e) {
    return false;
  }
}

function directoryExists(path) {
  try {
    return fs.statSync(path).isDirectory();
  } catch (e) {
    return false;
  }
}

```

