Contenido

[Manual Técnico App Freelancers 2](#_Toc84540217)

[Software a instalar 2](#_Toc84540218)

[Android Studio 2](#_Toc84540219)

[Visual Studio Code 3](#_Toc84540220)

[Extensión dart para visual studio code 3.12.0 3](#_Toc84540221)

[Extensión flutter para visual studio code 3.12.0 5](#_Toc84540222)

[Flutter Sdk 7](#_Toc84540223)

[Vysor 8](#_Toc84540224)

[Código fuente en flutter 9](#_Toc84540225)

[Carpetas Importantes 10](#_Toc84540226)

[Base de datos y servidor desde Firebase Console 15](#_Toc84540227)

[Configuración de Firebase 22](#_Toc84540228)

[Autenticación 22](#_Toc84540229)

[Storage 23](#_Toc84540230)

[Firestore Database 24](#_Toc84540231)

[Pruebas de usuario 28](#_Toc84540232)

# Manual Técnico App Freelancers

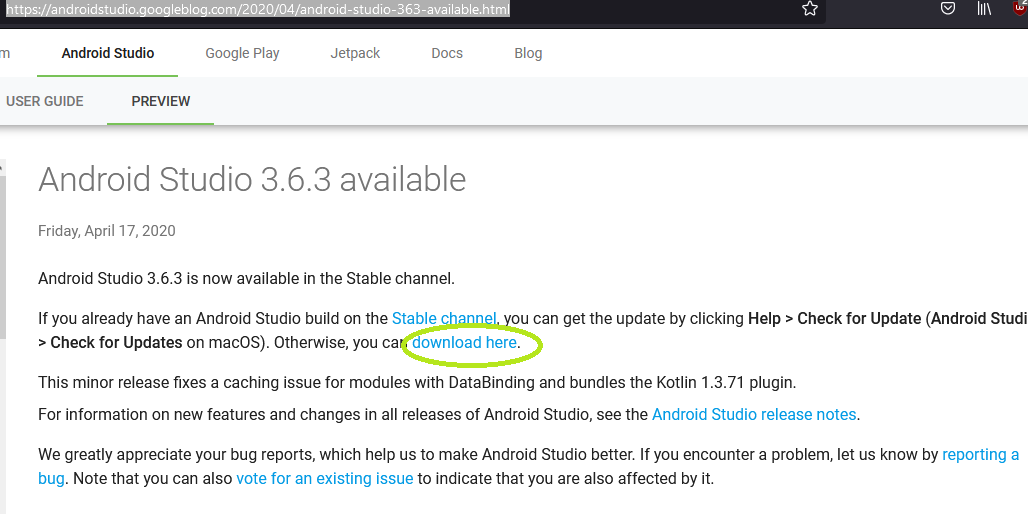
La aplicación móvil freelancers fue desarrollada con el framework flutter, utilizando el lenguaje dart, ambas tecnologías pertenecen a google. La base de datos utilizada es cloudfirestore que es una base de datos no sql de firebase, firebase también es una tecnología de google.

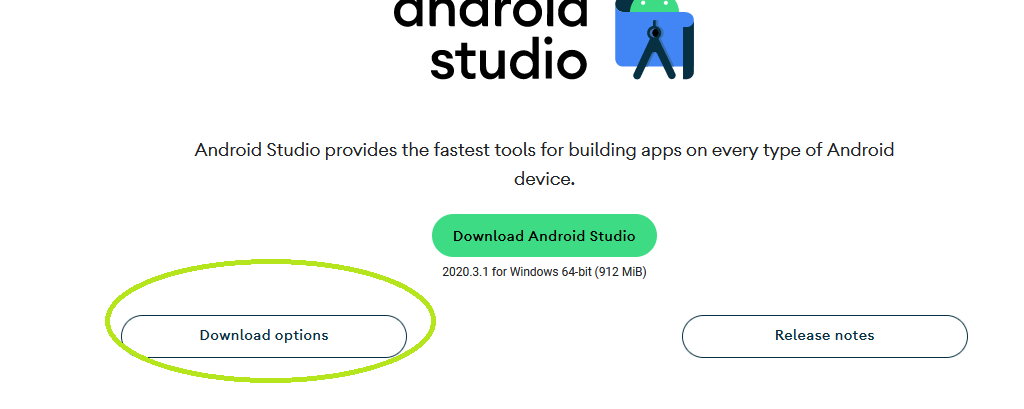
## Software a instalar

1. Android studio versión 3.6.3
2. Visual Studio code
3. Extensión dart para visual studio code 3.12.0
4. Extensión flutter para visual studio code 3.12.0
5. Flutter Sdk (flutter\_windows\_v1.12.13+hotfix.5-stable)
6. Vysor (cualquier versión)

### Android Studio

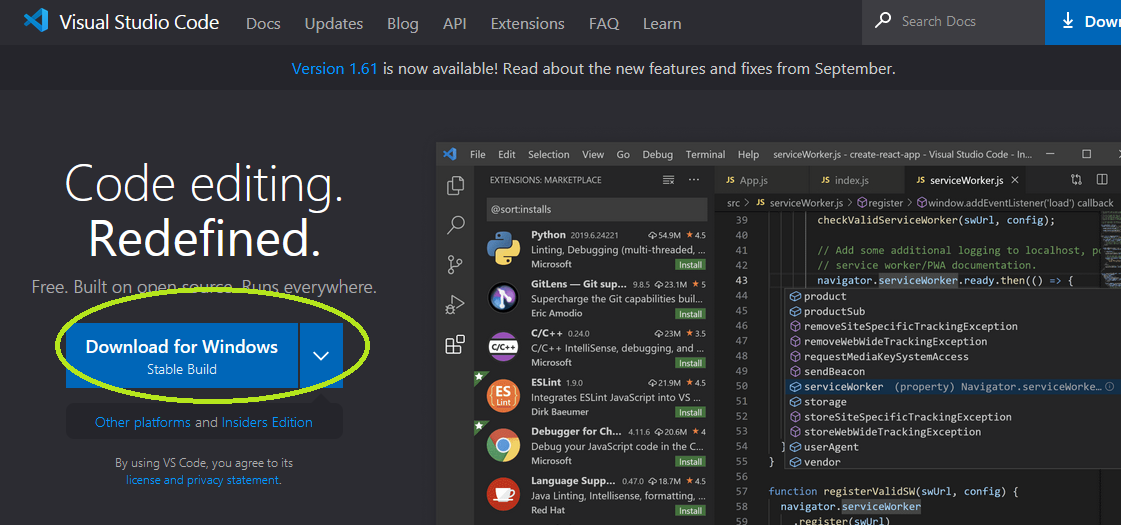
Necesitaremos instalar Android Studio 3.6.3, para eso vamos al link <https://androidstudio.googleblog.com/2020/04/android-studio-363-available.html>





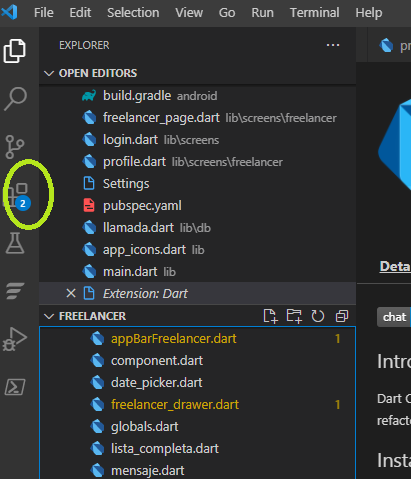
### Visual Studio Code

Visual Studio Code lo podemos descargar en la siguiente pagina <https://code.visualstudio.com/>

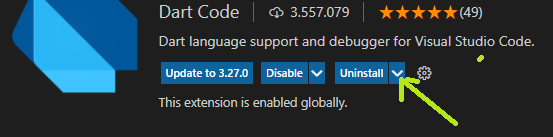


### Extensión dart para visual studio code 3.12.0

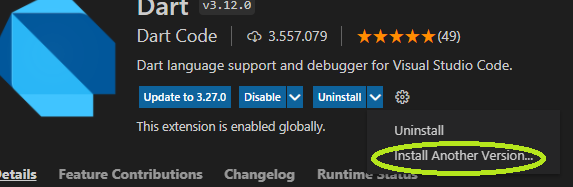
Para instalar la extención de dart para visual studio code, se da clic en el icono de extensiones.



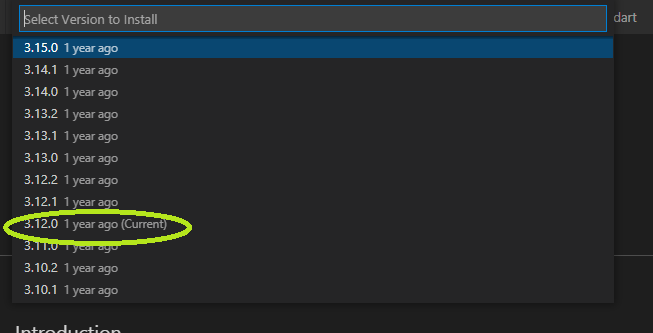
Luego instalamos el plugin dart que se nos muestra, dando clic en install, después que se ha instalado, bajaremos a la versión que necesitamos, damos clic en la flechita unistall.



Luego seleccionamos la opción install another versión.



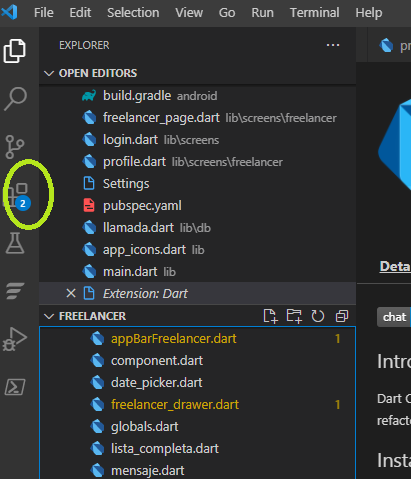
Luego seleccionamos la versión que deseamos, en este caso la versión 3.12.0



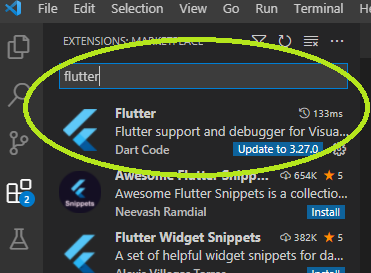
Despues de dar clic en la versión 3.12.0, se habrá instalado el plugin y podemos proceder a instalar flutter.

### Extensión flutter para visual studio code 3.12.0

Para instalar la extensión de flutter en visual studio code damos clic en el icono de extensiones.



Luego buscamos el plugin flutter.

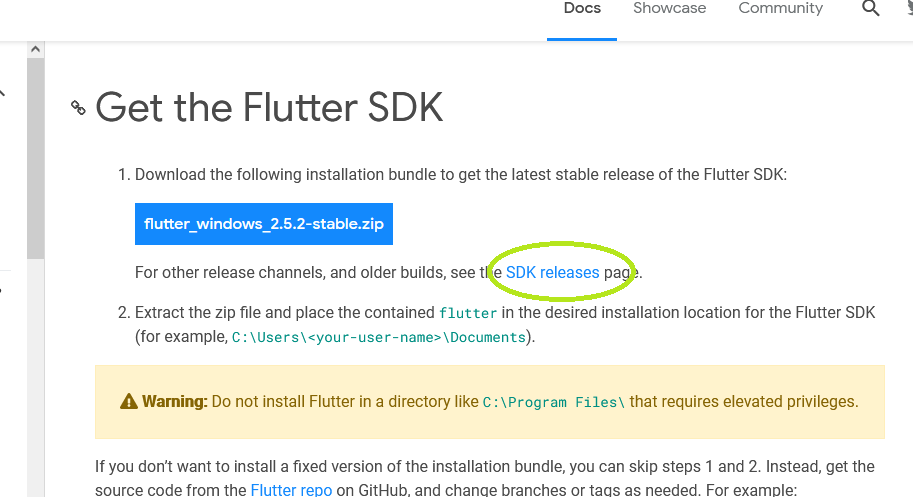


Luego de instalada, esta habrá actualizado a la versión 3.12.0 automáticamente porque esta es la versión de flutter que hace pareja con dart, entonces visual studio code detecta esto y selecciona la versión.

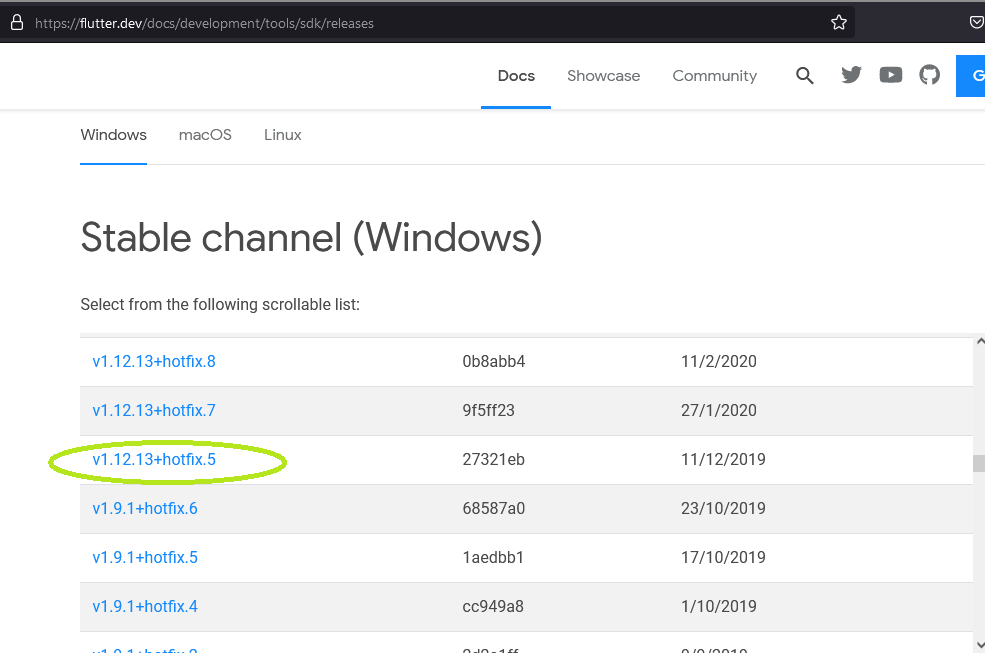
### Flutter Sdk

Para que flutter funcione tenemos que instalar el sdk de flutter, este lo podemos encontrar en el siguiente link <https://flutter.dev/docs/get-started/install/windows>

Alli encontraremos la siguiente información, daremos clic en el botón SDK releases para instalar la versión que queremos.



Luego elegimos la versión v1.12.13+hotfix5



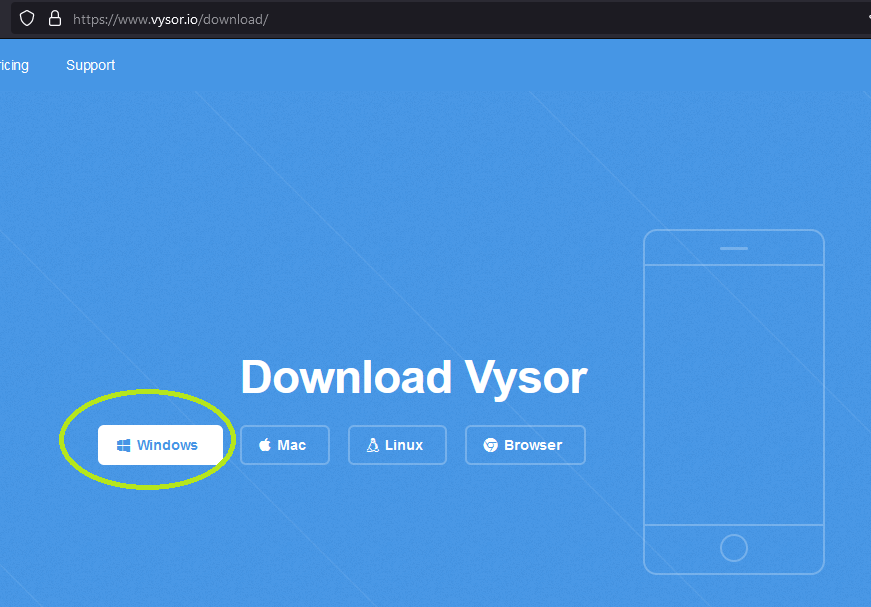
Luego la descargamos como zip, y cuando abramos nuestro proyecto flutter nos saldrá una advertencia que no se ha encontrado flutter sdk, y nos preguntara si queremos localizarlo o descargarlo, en este caso como ya lo tenemos descargado, solo lo ubicamos y con esto estará enlazado al proyecto.

### Vysor

Este software es opcional, consiste en un software que proyecta a nuestro computador algún teléfono que tengamos conectado mediante usb. Para que funcione basta con haber activado el modo desarrollador en nuestro teléfono.

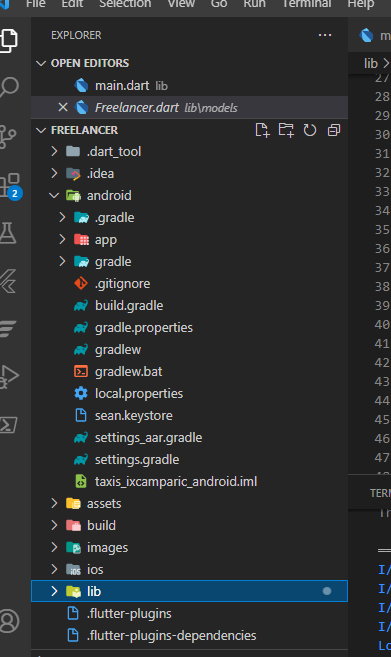
Podemos descargarlo en este link:

<https://www.vysor.io/download/>



## Código fuente en flutter

Para crear un proyecto flutter en visual studio code presionamos control + shift + p y nos saldrá la opción de crear un proyecto flutter, escribimos el nombre y automáticamente nos generara todas las carpetas necesarias para poder empezar codificar.



### Carpetas Importantes

#### Android

En esta carpeta se crea un proyecto nativo Android, a partir del código que se desarrolle en el proyecto.

#### Android/app

En esta carpeta se genera el codigo Android que se menciona en el inciso anterior, lo mas importante a resaltar son los archivos /android/app/build.gradle y /Android/build.gradle,que es donde se importan librerías de firebase que se utilizara como base de datos.

#### Ios

En esta carpeta se genera codigo nativo de xcode para poder generar el app en la app store, en este tutorial nos enfocamos en andoroid.

#### Lib

Esta es la carpeta mas importante, y es que aca esta el codigo fuente del app. Por convención todo inicia en un archivo llamado main.dart donde se suele llamar al login o pantallas iniciales.

#### Lib/db

En esta carpeta se almacena codigo fuente que conecta a la base de datos de firebase, estos son los llamados servicios.

#### Lib/models

En esta carpeta se almacenan los modelos, que son clases tipo “plantilla” para mostrar los datos obtenidos de la base de datos en los widgets del frontend.

#### Lib/screens

En esta carpeta se almacenan los archivos de codigo fuente que hacen posible la interface grafica. La mayoría de este codigo cuenta con los distintos widgets que provee flutter para crear aplicaciones.

#### Lib/util

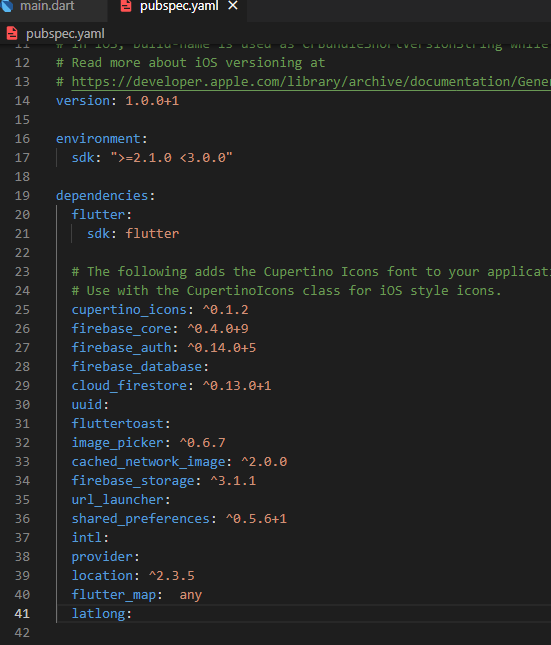
En esta carpeta se almacena codigo fuente que se reutiliza en toda la aplicación.

#### Archivo puspect.yaml

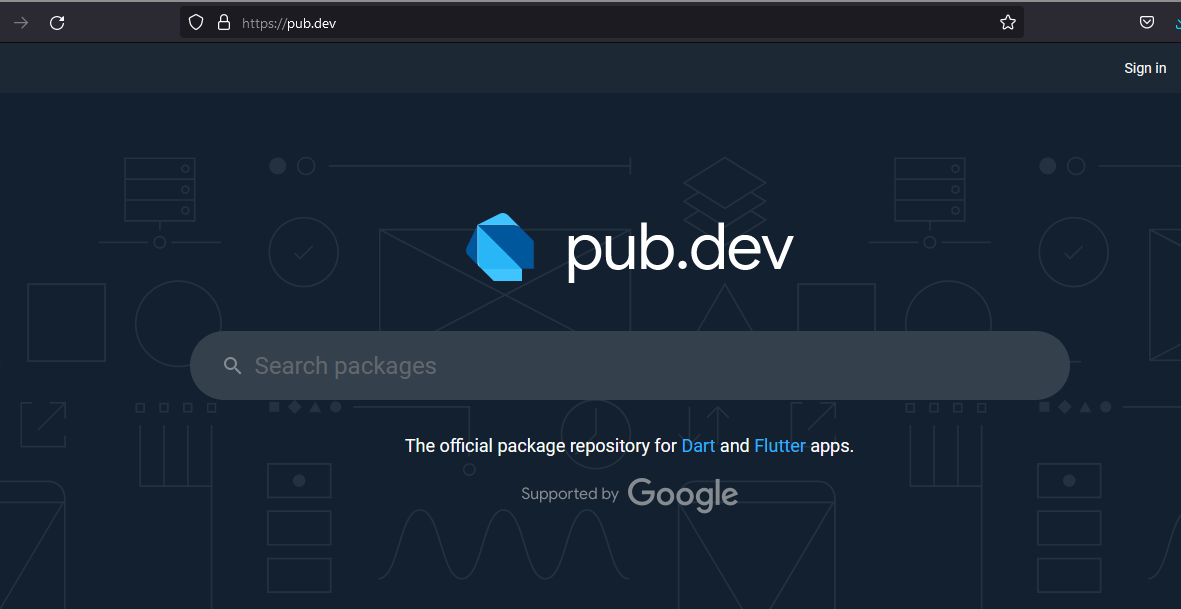
Este es uno de los archivos de configuración mas importantes, cuenta con la info de las librerías que se implementan.Para implementar un “plugin” se busca alguno que deseemos en la pagina

<https://pub.dev/>

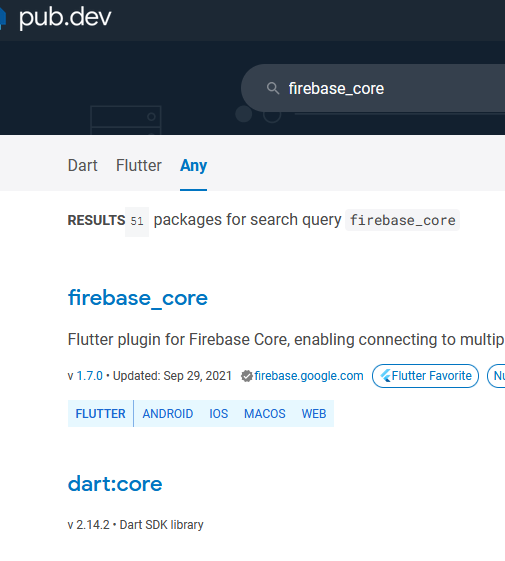
Vista del archivo pubspect.yaml



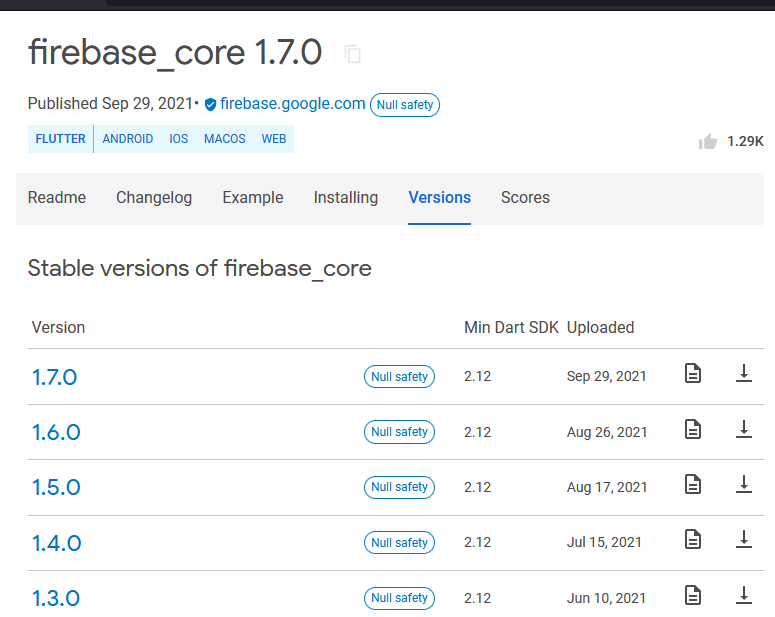
Vista de la pagina



Por ejemplo para buscar firebase\_core



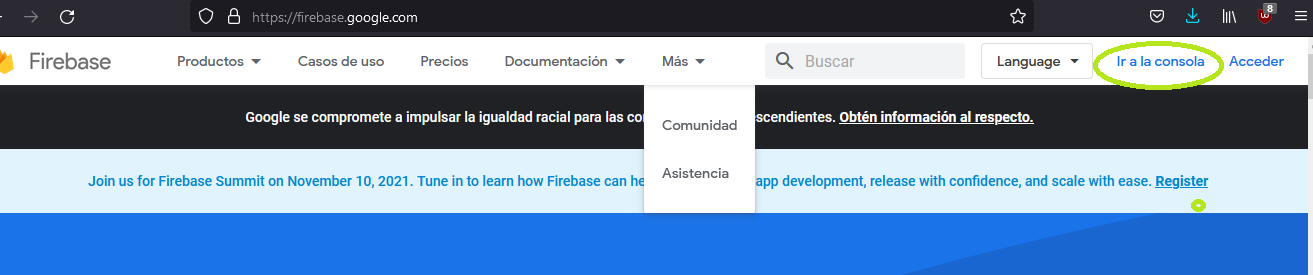
Se selecciona el plugin y se se selecciona la versión a utilizar



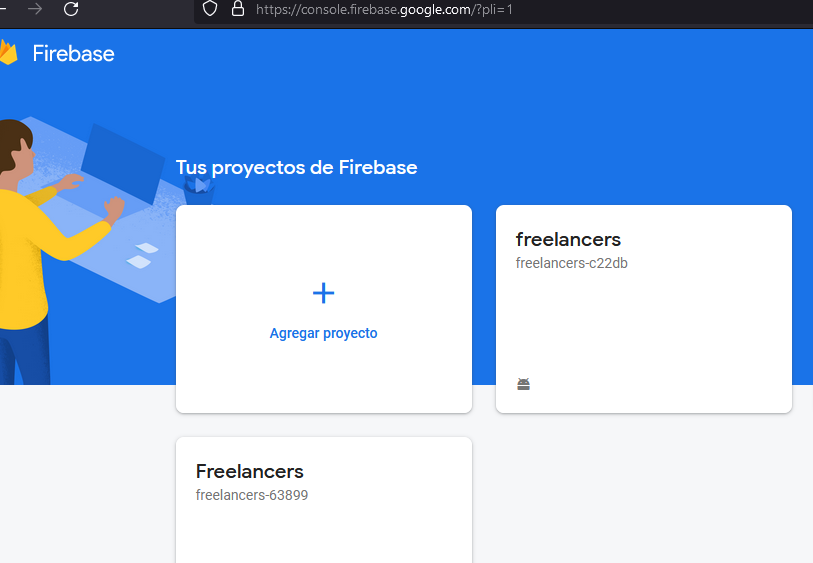
## Base de datos y servidor desde Firebase Console

Firebase console nos sirve como un servidor y como base de datos con cloud firebase store.

Para crear una tenemos que tener un usuario de google y crear una cuenta en firebase. En el siguiente link. <https://firebase.google.com/>



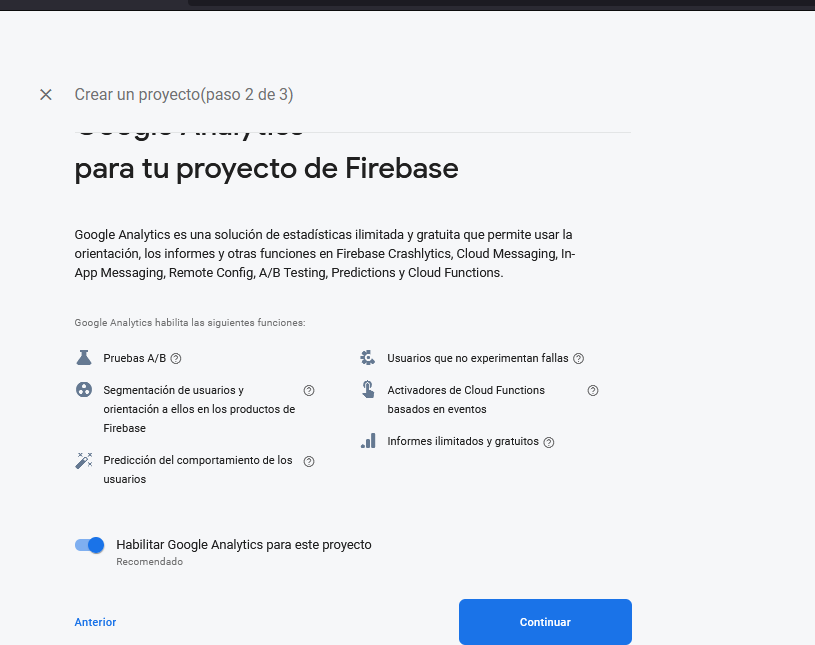
Luego nos logueamos con una cuenta de google o Gmail, si no hemos creado un proyecto nos aparecera la opción.



El siguiente paso es dar clic en agregar proyecto, se mostrara la siguiente pantalla



Luego nos pregunta si deseamos instalar google analytics, le damos clic en habilitar y en continuar

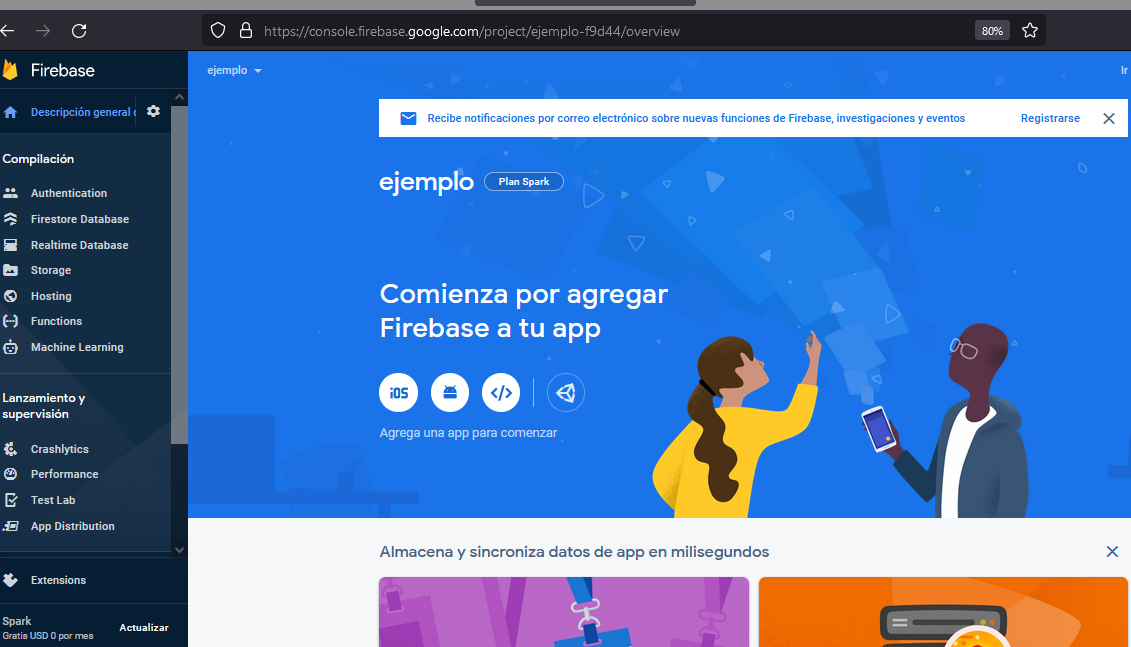


Luego seleccionamos la cuenta default de firebase



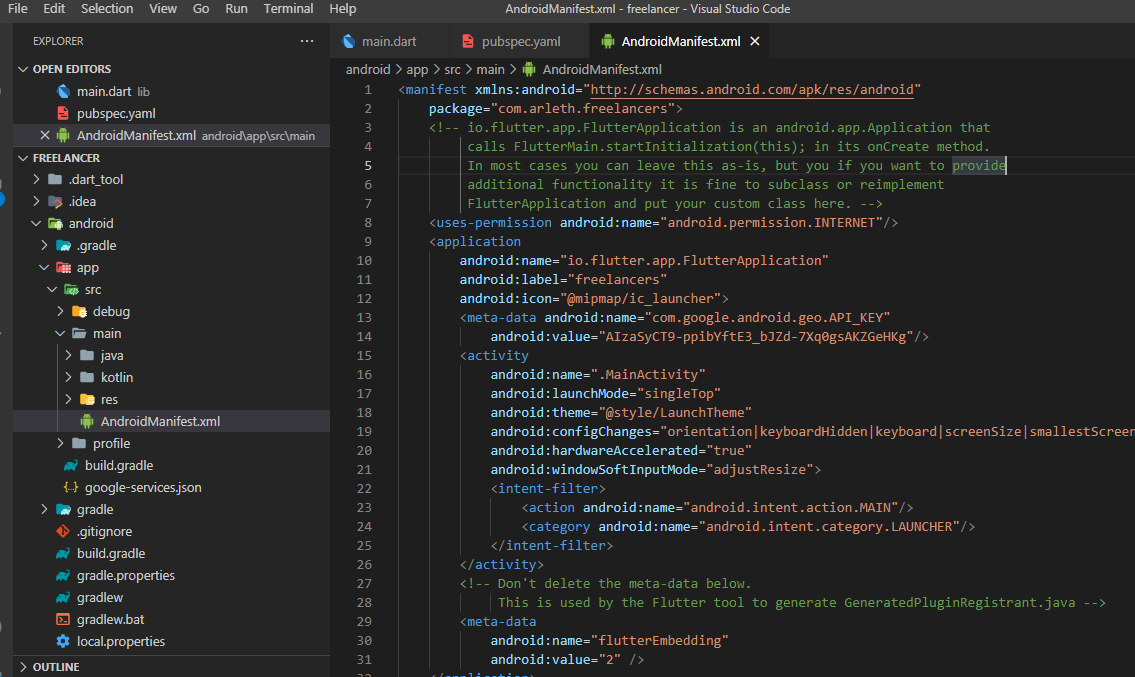
Por ultimo en crear proyecto, y aparecerá el menú principal de nuestro proyecto.

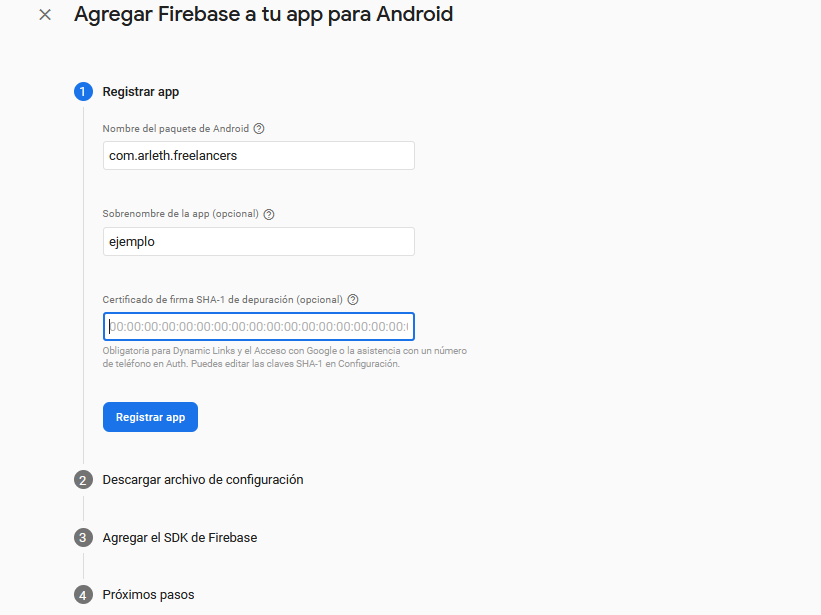
Luego damos clic en el icono de Android para crear un proyecto Android.



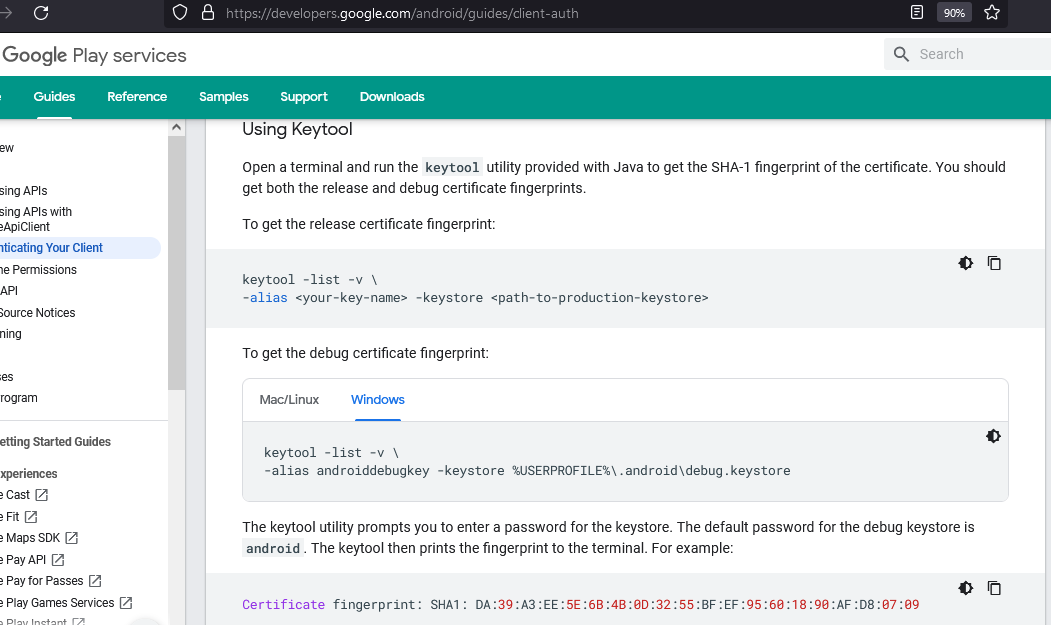
Luego damos clic en el icono de Android para crear un proyecto firebase enlazado a un proyecto Android, por ejemplo al proyecto flutter que habíamos creado antes.

Luego nos preguntara que nombre de paquete tiene nuestra app, por ejemplo la de este proyecto es com.arleth.freelancers, este esta ubicado en el archivo /Android/app/src/main/AndroidManifest.xml



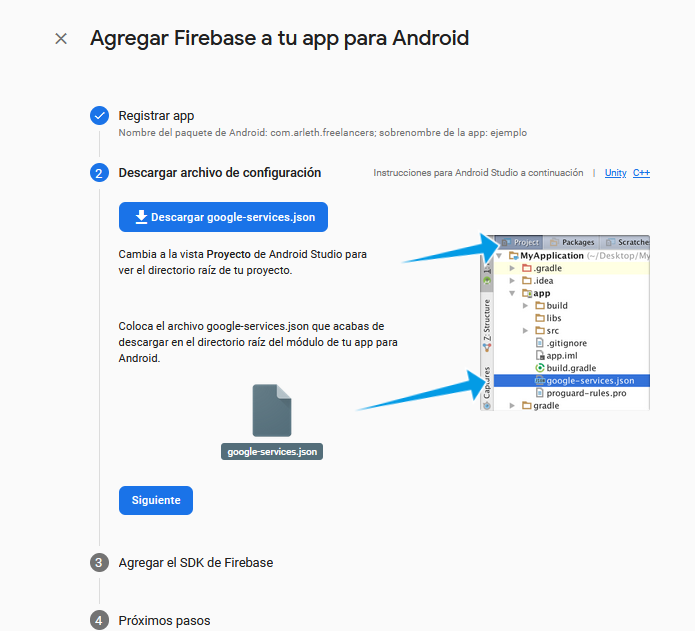


Luego nos pregunta por un certificado sha1 que es opcional, como esta app se loguea por numero de teléfono entonces necesitamos uno, este lo generamos con el siguiente comando desde una termminal.

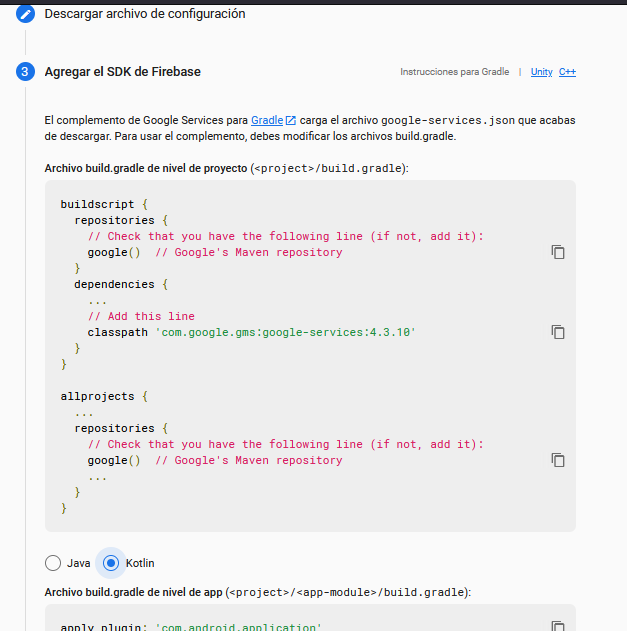


Luego de generado este sha1, lo pegamos en el registro de firebase y pasamos al siguiente paso.

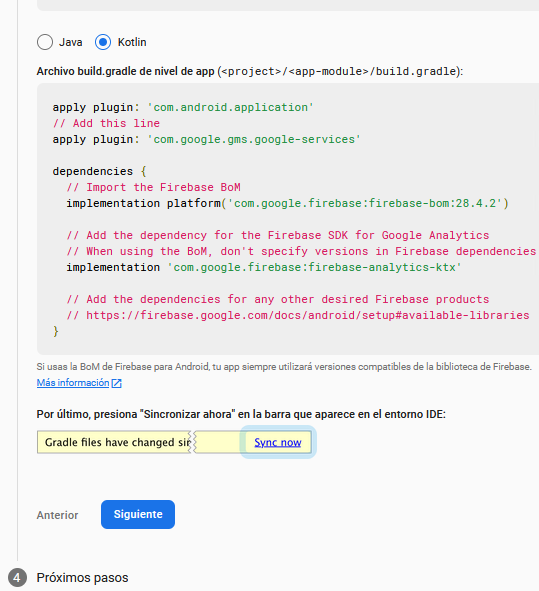
Despues descargamos el archivo google-services.json que viene siendo como nuestra conexión a nuestra futura base de datos firebase. Este archivo se debe guardar en el path /Android/app/google-services.json



Luego incluimos los servicios de google en el archivo Android/build.gradle



Despues agregamos este codigo al archivo /Android/app/build.gradle

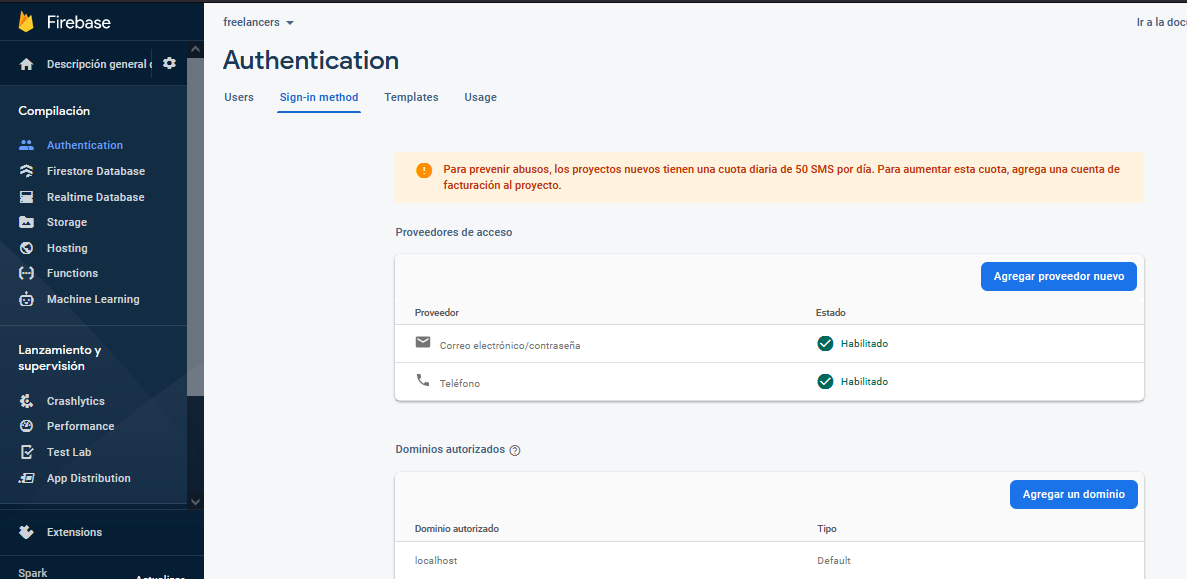


Habiendo hecho esto, tenemos conectada nuestra app flutter a firebase, sin embargo tenemos que ver como configuramos la autenticación de usuarios, la base de datos y habilitar el storage para subir imágenes u otros.

## Configuración de Firebase

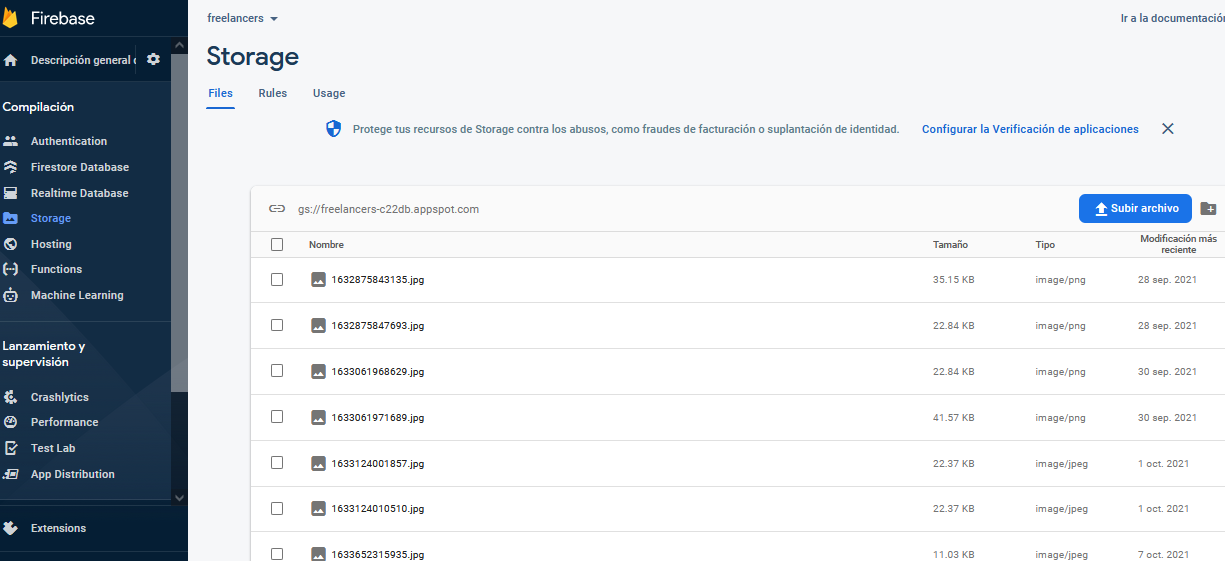
### Autenticación

Lo primero que haremos es ir a la pestaña de autenticación y habilitar el inicio de sesión con numero telefónico.

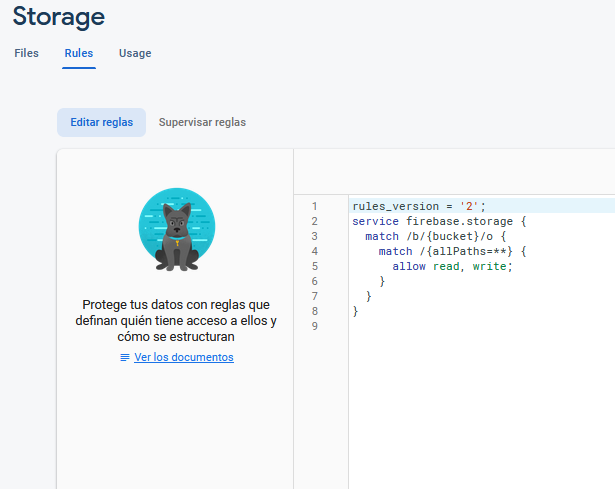


### Storage

El storage nos sirve para tener una nube en nuestro firebase, allí se pueden subir archivos que se usaran en el app.



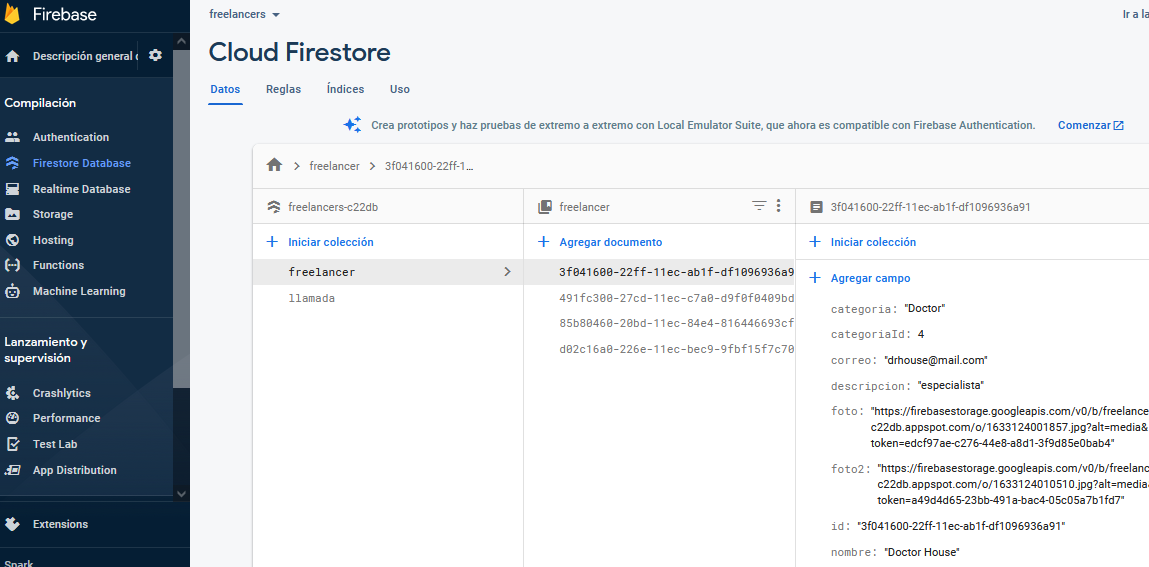
Para habilitar la subida de archivos nos vamos a la pestaña rules y escribimos el siguiente codigo.



### Firestore Database

Finalmente habilitamos nuestra base de datos, esta es una base de datos no sql lo que significa que no se utilizan tablas, sino utilizamos “colecciones” una colección puede tener varios “documentos” un documento viene siendo en un tradicional sql un “row” o fila, un documento puede tener varios campos que en un sql tradicional le llamamos columnas.

Para crear nuestra base de datos no es necesario crearla desde firebase console, basta con crearla desde nuestro codigo flutter que estaba alojado en la carpeta “db”.



Ejemplo de codigo en flutter para un crud en firestore database.

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:freelancers/models/SeanLista.dart';

import 'package:freelancers/models/Freelancer.dart';

import 'package:uuid/uuid.dart';

import 'package:freelancers/util/globals.dart' as globals;

class FreelancerService {

   Firestore \_firestore = Firestore.instance;

  String ref = 'freelancer';

  String create({String nombre, String profesion,

  String foto,String foto2, String descripcion, String telefono, int categoriaId, String categoria,

   String correo}){

    var id = Uuid();

    String freelanceId = id.v1();

    \_firestore.collection(ref).document(freelanceId).setData({

              'id': freelanceId,

              'nombre' : nombre,

              'profesion' : profesion,

              'foto' : foto,

              'foto2': foto2,

              'descripcion' : descripcion,

              'telefono': telefono,

              'categoriaId': categoriaId,

              'categoria': categoria,

              'correo': correo

      });

      return freelanceId;

  }

 Future<void> update({String freelancerId,String nombre, String profesion,

  String foto, String foto2, String descripcion, String telefono, int categoriaId,

  String correo, String categoria

   }) async {

    \_firestore.collection(ref).document(freelancerId).updateData({

              'id': freelancerId,

              'nombre' : nombre,

              'profesion' : profesion,

              'foto' : foto,

              'foto2': foto2,

              'descripcion' : descripcion,

              'telefono': telefono,

              'categoriaId': categoriaId,

              'categoria': categoria,

              'correo': correo

      });

  }

   void delete({String freelancerId

   }){

    \_firestore.collection(ref).document(freelancerId).delete();

  }

 Future<List<DocumentSnapshot>> getAll() =>

      \_firestore.collection(ref).getDocuments().then((snap){

        return snap.documents;

      });

Future<List<DocumentSnapshot>> getById(String id) =>

      \_firestore.collection(ref).where('id', isEqualTo: id).getDocuments().then((snap){

        return snap.documents;

      });

 Future<List<Freelancer>> getFreelancers() async {

   List<Freelancer> lista=[];

    List<DocumentSnapshot> dataArray = await getAll();

    for(DocumentSnapshot d in dataArray) {

        Freelancer freelancer = Freelancer(

              d.data['id'],

          d.data['nombre'],

          d.data['descripcion'],

          d.data['foto'],

          d.data['foto2'],

          d.data['categoriaId'],

          d.data['categoria'],

           d.data['telefono'],

          d.data['correo'],

          d.data['profesion']

        );

        lista.add(freelancer);

    }

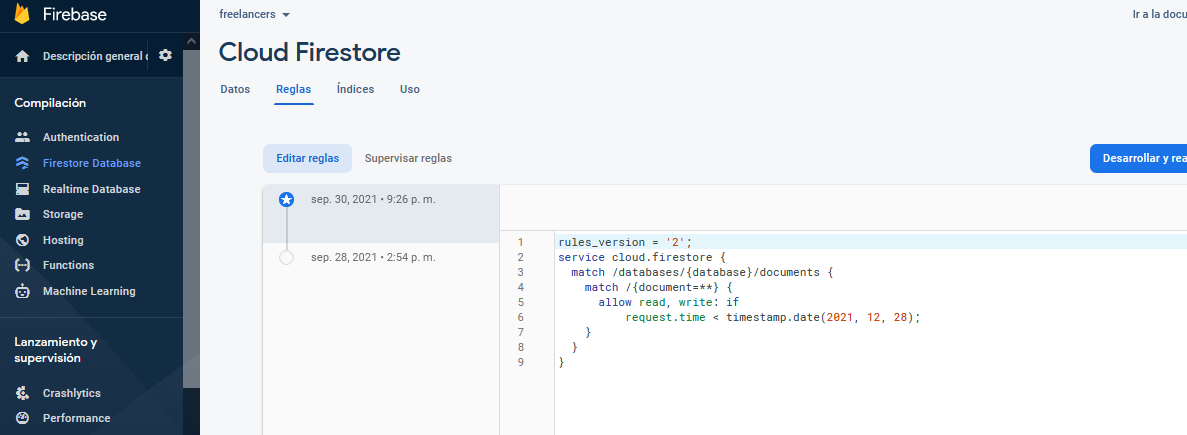
    return lista;

 }

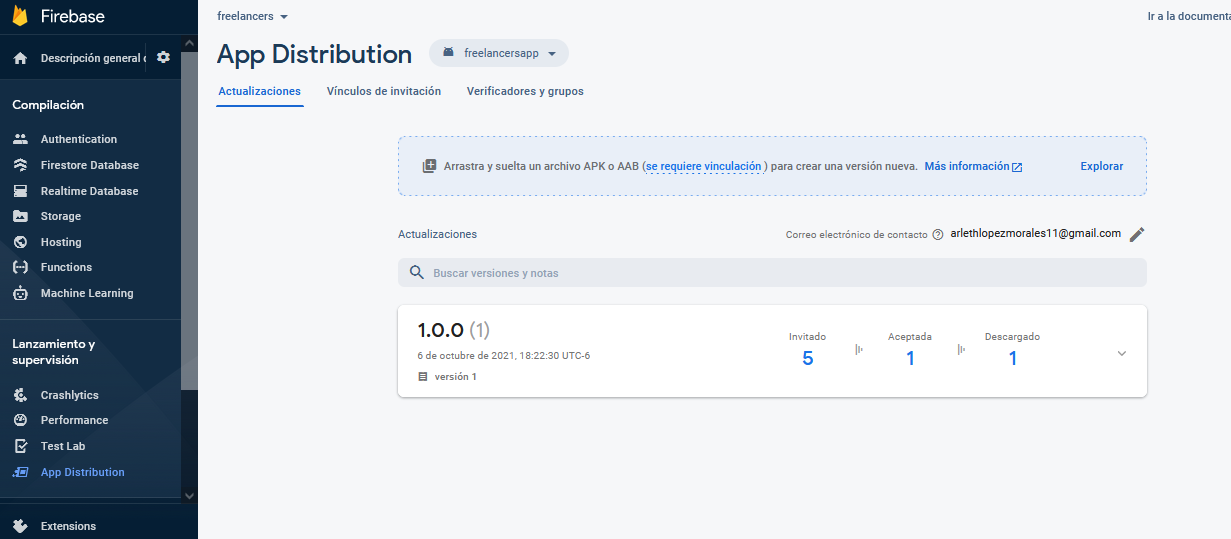
}

Este codigo es un fragmento de /lib/db/freelancer.dart que es el servicio principal para nuestra colección freelancer de nuestra base de datos de firebase.

Para configurar acceso a la base de datos también vamos a la pestaña rules, y escribimos el siguiente codigo.



## Pruebas de usuario

Para probar nuestra aplicación podemos ir a la opción App distribution

En esta opción podemos agregar testers a nuestra aplicación, con agregar su correo electrónico.

