

CHATAPP EN FLUTTER



Desarrollado por: José Francisco Vásquez Carrillo

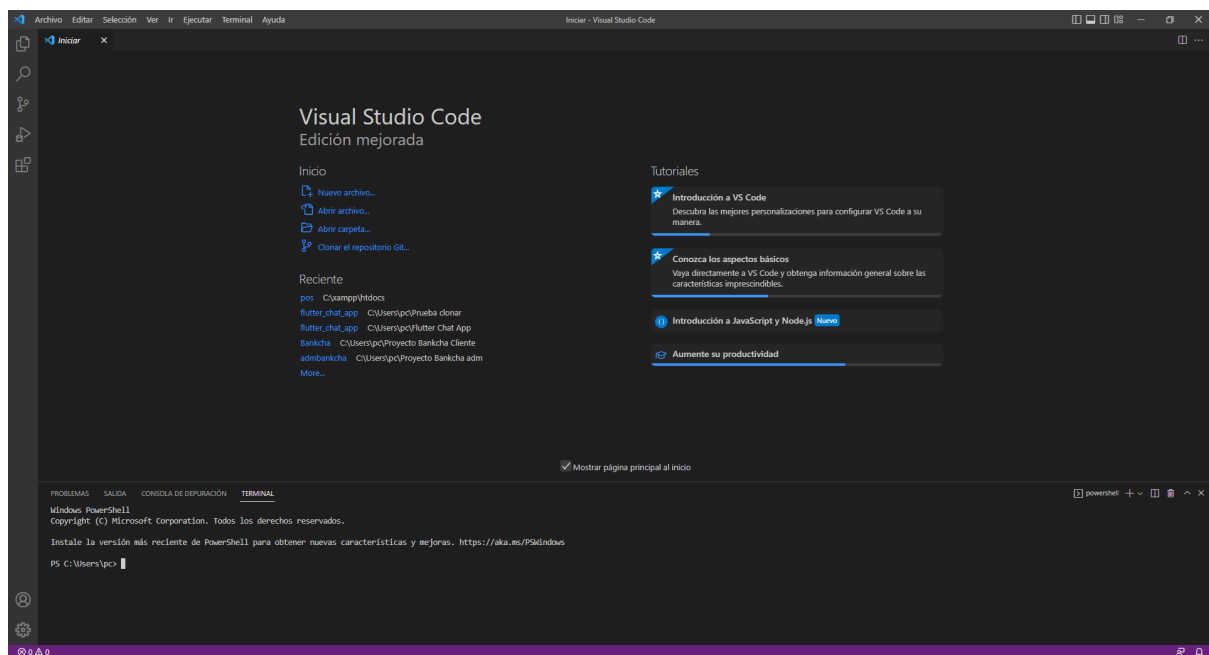
Repositorio en GitHub:

https://github.com/JoseFrancisco93/flutter_chat_app

DESPLIEGUE DE LA APLICACIÓN EN LA NUBE

1. Creación de la aplicación bajo Flutter en VSCode:

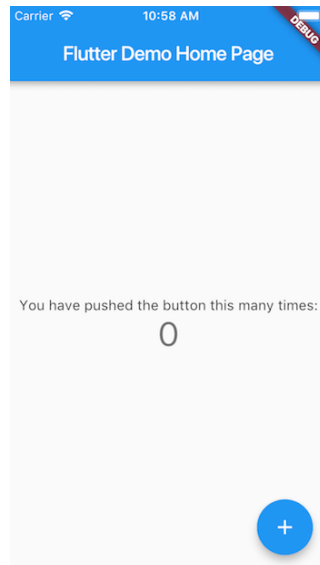
1.1 Crear un nuevo proyecto usando el terminal de VSCODE, con el nombre del proyecto ***flutter_chat_app***



```
$ flutter create flutter_chat_app  
$ cd flutter_chat_app
```

1.2 Ejecutar el siguiente comando ***flutter run*** compilaremos la app inicial en el dispositivo que seleccionemos para emular.

```
$ flutter run
```



1.3 Reemplazar el contenido de `lib/main.dart` para empezar a desarrollar nuestra UI.

¡Comenzar a Crear el diseño de la Interfaz de la APP!

2. Despliegue de Firebase Authentication en Flutter:

2.1 Importar los siguientes paquetes al proyecto y lo agregamos al `pubspec.yaml` en dependencias, guardar y ejecutar el posterior comando.

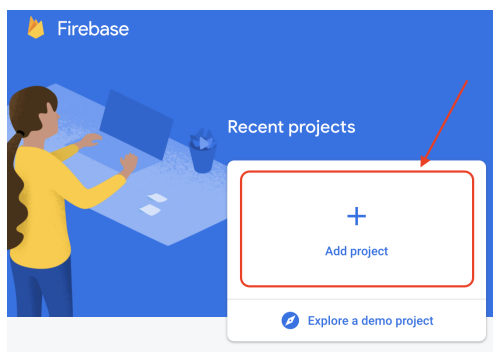
Dependencies:

```
- firebase_core:^1.24.0  
- firebase_auth:^3.11.1
```

```
$ flutter packages get
```

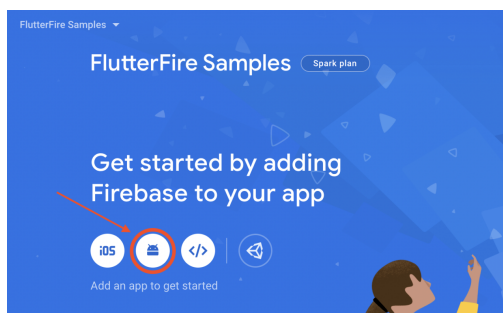
2.2 Configurar proyecto en Firebase Console

2.2.1 Agregar proyecto con el nombre de **ChatApp**, y esperar que el proyecto sea creado.



2.2.2 Configuración para Android.

- Seleccionar el icono de android



- Ingresar el nombre del paquete de Android, un apodo de la aplicación y el SHA-1 y registrar la aplicación.

× Add Firebase to your Android app

1 Register app

com.souvikbiswas.flutterfire_samples

App nickname (optional) ⓘ
FlutterFire Samples

Debug signing certificate SHA-1 (optional) ⓘ
1B:2D:30:CC:A9:0A:49:75:48:39:12:29:73:62:A4:85:D/

Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Register app

2 Download config file

3 Add Firebase SDK

- Descargar el archivo **google-services.json** y agregarlo al directorio **android/app**, y luego hacer click en siguiente.

× Add Firebase to your Android app

✓ Register app
Android package name: com.souvikbiswas.flutterfire_samples, App nickname: FlutterFire Samples

2 Download config file
Instructions for Android Studio below | [Unity](#) [C++](#)

Download google-services.json

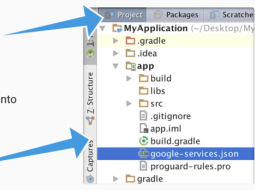
Switch to the Project view in Android Studio to see your project root directory.

Move the google-services.json file you just downloaded into your Android app module root directory.

google-services.json

Next

3 Add Firebase SDK



- Siga las instrucciones y agregue los fragmentos de código al proyecto, y luego hacer click en siguiente.

3 Add Firebase SDK Instructions for Gradle | [Unity](#) [C++](#)

The Google services plugin for [Gradle](#) loads the google-services.json file you just downloaded. Modify your build.gradle files to use the plugin.

Project-level build.gradle (<project>/build.gradle):

```
buildscript {
    repositories {
        // Check that you have the following line (if not, add it):
        google() // Google's Maven repository
    }
    dependencies {
        ...
        // Add this line
        classpath 'com.google.gms:google-services:4.3.5'
    }
}

allprojects {
    ...
    repositories {
        // Check that you have the following line (if not, add it):
        google() // Google's Maven repository
    }
    ...
}
```

☒ Java ☐ Kotlin

App-level build.gradle (<project>/<app-module>/build.gradle):

```
apply plugin: 'com.android.application'
// Add this line
apply plugin: 'com.google.gms.google-services'

dependencies {
    // Import the Firebase BoM
    implementation platform('com.google.firebase:firebase-bom:26.6.0')

    // Add the dependency for the Firebase SDK for Google Analytics
    // When using the BoM, don't specify versions in Firebase dependencies
    implementation 'com.google.firebase:firebase-analytics'

    // Add the dependencies for any other desired Firebase products
    // https://firebase.google.com/docs/android/setup#available-libraries
}
```

By using the Firebase Android BoM, your app will always use compatible Firebase library versions. [Learn more](#)

Finally, press "Sync now" in the bar that appears in the IDE:

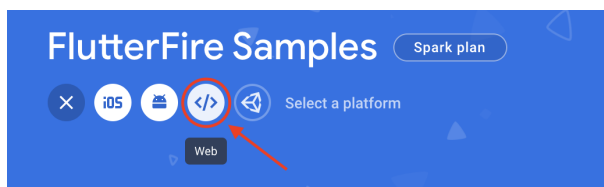
Gradle files have changed sit [Sync now](#)

[Previous](#) [Next](#)

¡Ha configurado correctamente Firebase para Android!

2.2.3 Configuración para Web

- Seleccionar el icono de android



- Ingrese el apodo de la aplicación y haga clic en Registrar aplicación, y luego guardar el siguiente script .

2 Add Firebase SDK

Copy and paste these scripts into the bottom of your <body> tag, but before you use any Firebase services:

```

<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/8.2.10/firebase-app.js"></scrip

<!-- TODO: Add SDKs for Firebase products that you want to use
https://firebase.google.com/docs/web/setup#available-libraries -->
<script src="https://www.gstatic.com/firebasejs/8.2.10/firebase-analytics.js"><

<script>
  // Your web app's Firebase configuration
  // For Firebase JS SDK v7.20.0 and later, measurementId is optional
  var firebaseConfig = {
    apiKey: "AIzaSyBIXxQgUPYDU2srK_UpgjiTATON5rrdMd0",
    authDomain: "flutterfire-samples.firebaseio.com",
    projectId: "flutterfire-samples",
    storageBucket: "flutterfire-samples.appspot.com",
    messagingSenderId: "132381301913",
    appId: "1:132381301913:web:f0d0a643bc90bed91eec08",
    measurementId: "G-ZK5194H8VH"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
  firebase.analytics();
</script>

```

Learn more about Firebase for web: [Get Started](#), [Web SDK API Reference](#), [Samples](#)

[Continue to console](#)

¡Ha configurado correctamente Firebase para Web!

2.2.3 Habilitar Proveedor de acceso para **Correo Electrónico/Contraseña**

Authentication

Users Sign-in method Templates Usage Settings

Proveedores de acceso

[Agregar proveedor nuevo](#)

Proveedor	Estado
Correo electrónico/contraseña	Habilitado

Opciones avanzadas

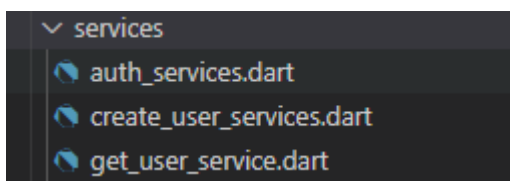
Autenticación de varios factores mediante SMS

Permite que tus usuarios agreguen un nivel de seguridad adicional a sus cuentas. Una vez que se habilite, integre y configure, los usuarios podrán acceder a sus cuentas a través dos pasos, mediante SMS. [Más información](#)

★ La MFA y otras funciones avanzadas están disponibles con Identity Platform, la solución completa de identidad de clientes de Google Cloud diseñada en asociación con Firebase. Esta actualización está disponible en los planes Spark y Blaze.

[Actualizar para habilitar](#)

2.2.3 Crear archivo **auth_services.dart** dentro de la carpeta **lib/services**.



- Agregar script de funciones en el archivo **auth_services.dart**

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart' as auth;
import 'package:flutter_chat_app/models/user.dart' as local;

// Auth Service Firebase
class AuthService {
  final auth.FirebaseAuth firebaseAuth = auth.FirebaseAuth.instance;

  final FirebaseFirestore firestore = FirebaseFirestore.instance;

  // Create user obj based on Firebaseuser
  local.User? _userFromFirebaseUser(auth.User user) {
    return local.User(idUser: user.uid);
  }

  // Auth change user stream
  Stream<auth.User?> get authState {
    return firebaseAuth.idTokenChanges();
  }

  // Function SignIn
  Future signInUserWithEmailAndPass(String userMail, String password) async {
    try {
      auth.User user = (await firebaseAuth.signInWithEmailAndPassword(
        email: userMail, password: password))
        .user!;
      // ignore: unnecessary_null_comparison
      if (user != null) {
        // Save data provider
        _userFromFirebaseUser(user);
        return true;
      }
    } on auth.FirebaseAuthException catch (e) {
      return e.code;
    }
  }

  // Function SignUp
  Future signUpUserWithEmailAndPass(
    String userMail, String password, String fullName) async {
    try {
      auth.User user = (await firebaseAuth.createUserWithEmailAndPassword(
        email: userMail, password: password))
        .user!;

      // ignore: unnecessary_null_comparison
      if (user != null) {
        // Save data user in Database Firestore
        firestore.collection('users').doc(user.uid).set({
          "userMail": userMail,
```

```

        "fullName": fullName,
        "idUser": user.uid,
    });
    // Save data provider
    _userFromFirebaseUser(user);
    return true;
}
} on auth.FirebaseAuthException catch (e) {
    return e.code;
}
}

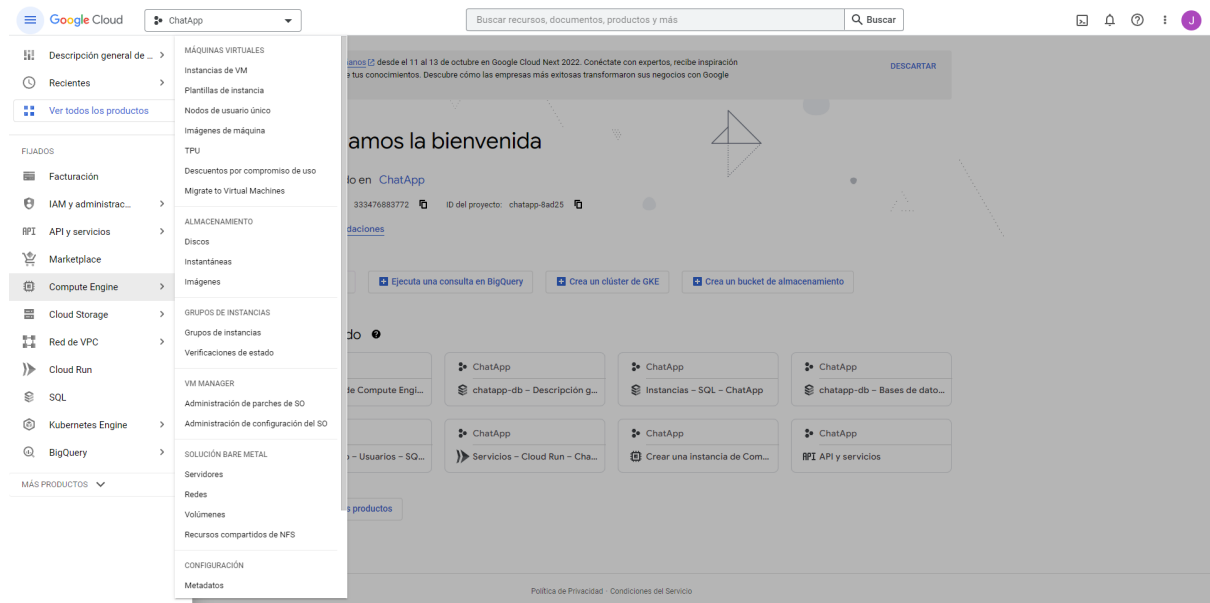
// Function Recovery Password
Future resetPassword(String email) async {
    try {
        await firebaseAuth.sendPasswordResetEmail(email: email);
    } catch (e) {
        return null;
    }
}

// Function SignOut
Future signOut() async {
    try {
        return await firebaseAuth.signOut();
        // ignore: unused_catch_clause
    } on auth.FirebaseAuthException catch (e) {
        return null;
    }
}
}

```


3. Despliegue de Base de Datos Relacional (MySQL) en Máquina Virtual de GCP:

3.1 En el panel izquierdo del menú de Google Cloud Console, ingresar en **Compute Engine -> Instancias de VM**



3.2 Crear nueva instancia con las siguientes características

- Nombre y Región

Nombre *
chatapp

Etiquetas ?
+ AGREGAR ETIQUETAS

Región *
us-central1 (Iowa)

Zona *
us-central1-a

La región es permanente

La zona es permanente

- Disco de Arranque

Disco de arranque ?

Nombre	chatapp
Tipo	Disco persistente balanceado nuevo
Tamaño	10 GB
Tipo de licencia ?	Gratis
Imagen	 Ubuntu 18.04 LTS

CAMBIAR

- Firewall


Firewall ?

Agrega etiquetas y reglas de firewall para permitir determinados tipos de tráfico de red desde Internet

- ☒ Permitir tráfico HTTP
- ☒ Permitir tráfico HTTPS



Posterior dar click en **Crear**, y esperamos hasta que quede creada nuestra instancia.

[INSTANCIAS](#) [PROGRAMAS DE LAS INSTANCIAS](#)


 Más información sobre el agente de operaciones [Más información](#) [DESCARTAR](#)


Las instancias de VM son máquinas virtuales altamente configurables para ejecutar cargas de trabajo en la infraestructura de Google. [Más información](#)


Filtro


<input type="checkbox"/>	Estado	Nombre ↑	Zona	Recomendaciones	En uso por	IP interna	IP externa	Conectar
<input type="checkbox"/>		mv-chatapp	us-central1-a	 Ahorrar \$13/mes		10.128.0.3 (nic0)	104.197.246.220 (nic0)	SSH ▾ ⋮


Acciones relacionadas [HIDE](#)


 **Explora Actifio GO**
Crea una copia de seguridad de tus VMs y configura la recuperación ante desastres.

 **Consulta el informe de facturación**
Visualiza y administra tu facturación de Compute Engine.

 **Supervisa VMs**
Visualiza los valores atípicos de VMs en métricas como CPU y red.

 **Explora los registros de VM**
Visualiza, busca, analiza y descarga los registros de instancias de VM.

 **Configura reglas de firewall**
Controla el tráfico hacia y desde una instancia de VM.

 **Administración de parches**
Programa actualizaciones de parches y verifica su cumplimiento en las instancias de VM.

3.3 Una vez creada nuestra Instancia, dar click en conectar **SSH** para agregar y configurar nuestra máquina con **MySQL**.

IP externa	Conectar
104.197.246.220 (nic0)	SSH ▾ ⋮

- Actualizar el índice de paquetes en el servidor:

```
$ sudo apt update
```

- Instalar el paquete de **mysql-server**. Puede verificar la guía de instalación en <https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-18-04>

```
$ sudo apt install mysql-server
```

Una vez instalado el paquete de **mysql-server**, se crean los usuarios con los siguientes comandos, teniendo en cuenta **myuser(Como el usuario) y 123456789(Como la contraseña)**.

```
$ CREATE USER 'myuser'@'localhost' IDENTIFIED BY '123456789';  
$ CREATE USER 'myuser'@'%' IDENTIFIED BY '123456789';
```

Luego,

```
$ GRANT ALL ON *.* TO 'myuser'@'localhost';  
$ GRANT ALL ON *.* TO 'myuser'@'%'; FLUSH PRIVILEGES;
```

Una vez culminado, reiniciar MySQL con el siguiente comando:

```
$ sudo service mysql restart
```

Finalizado conectarse a MySQL con el comando e ingresar las credenciales, y crear una tabla con las siguientes declaraciones:

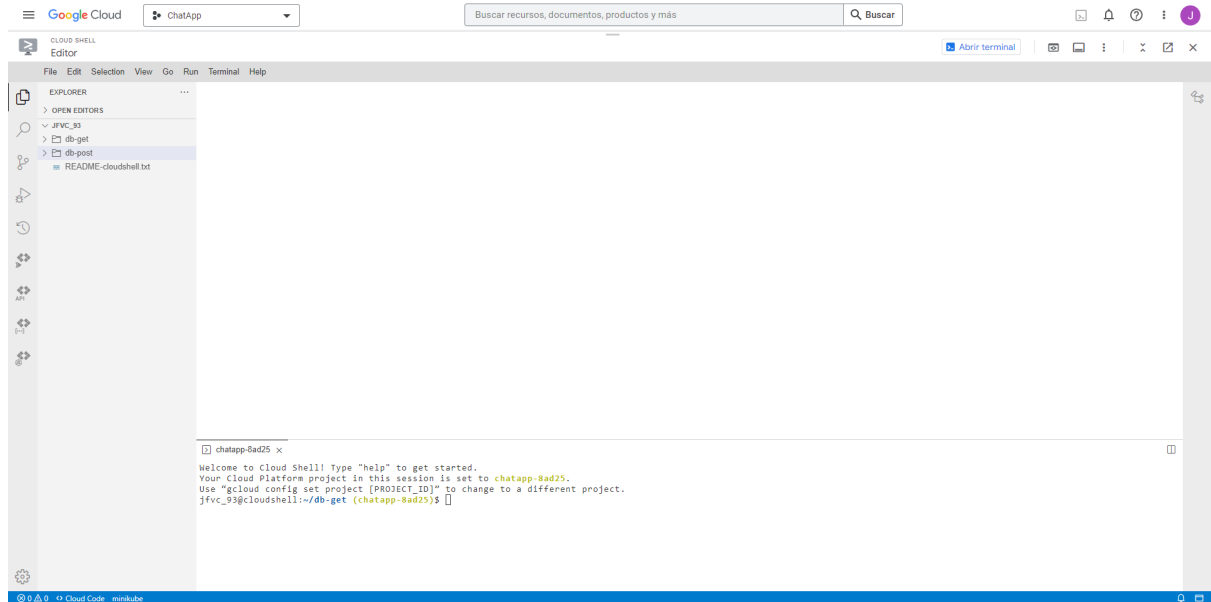
```
$ sudo mysql -u root -p
```

```
CREATE TABLE IF NOT EXISTS dataUsers (  
  id_user VARCHAR(50),  
  full_name VARCHAR(30),  
  user_age INT(3),  
  user_mail VARCHAR(50)  
)
```

Finalizado.

4. Despliegue de Endpoints usando Cloud Run:

4.1 En el panel izquierdo del menú de Google Cloud Console, ingresar en **Cloud Run**, y abrir el **Editor de CLOUD SHELL**.



4.2 Creación de **ENDPOINT-POST CON PHP**

4.2.1 Crear un directorio nuevo llamado **db-post** y cambiar el directorio al nuevo:

```
$ mkdir db-post  
$ cd db-post
```

4.2.2 Crear un archivo llamado **index.php** con el siguiente código:

```
<?php  
header("Access-Control-Allow-Origin: *");  
  
// Input  
$response = file_get_contents('php://input');  
$resp = json_decode($response, TRUE);  
  
// Database  
$servername = "104.197.246.220";  
$username = "myuser";  
$password = "123456789";
```

```

$db = "chatapp";
// Create connection
$conn = mysqli_connect($servername, $username, $password, $db);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Table
$table = 'dataUsers';

// Data
$id_user = $resp["id_user"];
$full_name = $resp["full_name"];
$user_age = $resp["user_age"];
$user_mail = $resp["user_mail"];
$sql = "INSERT INTO $table (id_user, full_name, user_age, user_mail)
VALUES ('$id_user', '$full_name', '$user_age', '$user_mail')";
$result = $conn->query($sql);
echo "success";
$conn->close();
return;

```

4.2.3 Crear un archivo nuevo llamado **Dockerfile** en el mismo directorio que los archivos fuente, y añadir la extensión de MySQL.

```

# Use the official PHP image.
# https://hub.docker.com/_/php
FROM php:8.0-apache

RUN docker-php-ext-install mysqli

# Configure PHP for Cloud Run.
# Precompile PHP code with opcache.
RUN docker-php-ext-install -j "$(nproc)" opcache
RUN set -ex; \
{ \
    echo "; Cloud Run enforces memory & timeouts"; \
    echo "memory_limit = -1"; \
    echo "max_execution_time = 0"; \
    echo "; File upload at Cloud Run network limit"; \
}

```

```

    echo "upload_max_filesize = 32M"; \
    echo "post_max_size = 32M"; \
    echo "; Configure OpCache for Containers"; \
    echo "opcache.enable = On"; \
    echo "opcache.validate_timestamps = Off"; \
    echo "; Configure OpCache Memory (Application-specific)"; \
    echo "opcache.memory_consumption = 32"; \
} > "$PHP_INI_DIR/conf.d/cloud-run.ini"

# Copy in custom code from the host machine.
WORKDIR /var/www/html
COPY . ./

# Use the PORT environment variable in Apache configuration files.
# https://cloud.google.com/run/docs/reference/container-contract#port
RUN sed -i 's/80/${PORT}/g'
/etc/apache2/sites-available/000-default.conf /etc/apache2/ports.conf

# Configure PHP for development.
# Switch to the production php.ini for production operations.
# RUN mv "$PHP_INI_DIR/php.ini-production" "$PHP_INI_DIR/php.ini"
#
https://github.com/docker-library/docs/blob/master/php/README.md#configuration
RUN mv "$PHP_INI_DIR/php.ini-development" "$PHP_INI_DIR/php.ini"

```

4.2.4 Agregar un archivo `.dockerignore` para excluir archivos en la imagen de contenedor.

```

# The .dockerignore file excludes files from the container build
process.
#
# https://docs.docker.com/engine/reference/builder/#dockerignore-file

# Exclude locally vendored dependencies.
vendor/

# Exclude "build-time" ignore files.
.dockerignore
.gcloudignore

```

```
# Exclude git history and configuration.  
.gitignore
```

4.2.5 En el directorio del código fuente, implementar desde la fuente con el siguiente comando:

```
$ gcloud config set project chatapp-8ad25  
$ gcloud builds submit --tag gcr.io/chatapp-8ad25/db-post  
$ gcloud run deploy --image gcr.io/chatapp-8ad25/db-post --platform managed  
  
// Name: db-post  
// Server: 28
```

Finalizado, tenemos una URL de servicio para visitar el servicio implementado.

4.3 Creación de **ENDPOINT-GET CON PHP**

4.3.1 Crear un directorio nuevo llamado **db-get** y cambiar el directorio al nuevo:

```
$ mkdir db-get  
$ cd db-get
```

4.2.2 Crear un archivo llamado **index.php** con el siguiente código:

```
<?php  
header("Access-Control-Allow-Origin: *");  
  
// Get ID USER  
$id = $_GET['id'];  
  
// Database  
$servername = "104.197.246.220";  
$username = "myuser";  
$password = "123456789";  
$db = "chatapp";  
// Create connection  
$conn = mysqli_connect($servername, $username, $password, $db);  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli_connect_error());  
}
```

```

}

// Table
$table = 'dataUsers';

// Data
$data_user = [];
$sql = "SELECT id_user, full_name, user_age, user_mail FROM $table
WHERE id_user = '$id'";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        $data_user[] = $row;
    }
    // Send back the complete records as a json
    echo json_encode($data_user);
} else {
    echo "error";
}
$conn->close();
return;

```

4.2.3 Crear un archivo nuevo llamado **Dockerfile** en el mismo directorio que los archivos fuente, y añadir la extensión de MySQL.

```

# Use the official PHP image.
# https://hub.docker.com/_/php
FROM php:8.0-apache

RUN docker-php-ext-install mysqli

# Configure PHP for Cloud Run.
# Precompile PHP code with opcache.
RUN docker-php-ext-install -j "$(nproc)" opcache
RUN set -ex; \
{ \
    echo "; Cloud Run enforces memory & timeouts"; \
    echo "memory_limit = -1"; \
    echo "max_execution_time = 0"; \
    echo "; File upload at Cloud Run network limit"; \
    echo "upload_max_filesize = 32M"; \

```



```

    echo "post_max_size = 32M"; \
    echo "; Configure Opcache for Containers"; \
    echo "opcache.enable = On"; \
    echo "opcache.validate_timestamps = Off"; \
    echo "; Configure Opcache Memory (Application-specific)"; \
    echo "opcache.memory_consumption = 32"; \
} > "$PHP_INI_DIR/conf.d/cloud-run.ini"

# Copy in custom code from the host machine.
WORKDIR /var/www/html
COPY . ./

# Use the PORT environment variable in Apache configuration files.
# https://cloud.google.com/run/docs/reference/container-contract#port
RUN sed -i 's/80/${PORT}/g'
/etc/apache2/sites-available/000-default.conf /etc/apache2/ports.conf

# Configure PHP for development.
# Switch to the production php.ini for production operations.
# RUN mv "$PHP_INI_DIR/php.ini-production" "$PHP_INI_DIR/php.ini"
#
https://github.com/docker-library/docs/blob/master/php/README.md#conf
iguration
RUN mv "$PHP_INI_DIR/php.ini-development" "$PHP_INI_DIR/php.ini"

```

4.2.4 Agregar un archivo .dockerignore para excluir archivos en la imagen de contenedor.

```

# The .dockerignore file excludes files from the container build
process.
#
# https://docs.docker.com/engine/reference/builder/#dockerignore-file

# Exclude locally vendored dependencies.
vendor/

# Exclude "build-time" ignore files.
.dockerignore
.gcloudignore

# Exclude git history and configuration.

```

```
.gitignore
```

4.2.5 En el directorio del código fuente, implementar desde la fuente con el siguiente comando:

```
$ gcloud config set project chatapp-8ad25  
$ gcloud builds submit --tag gcr.io/chatapp-8ad25/db-get  
$ gcloud run deploy --image gcr.io/chatapp-8ad25/db-get --platform managed  
  
// Name: db-get  
// Server: 28
```

Finalizado, tenemos una URL de servicio para visitar el servicio implementado.

5. Despliegue de Firebase Firestore para Mensajería Instantánea:

5.1 Importar los siguientes paquetes al proyecto y lo agregamos al `pubspec.yaml` en dependencias, guardar y ejecutar el posterior comando.

```
Dependencies:
```

```
- cloud_firestore:^3.5.0
```

```
$ flutter packages get
```

5.2 Creación de la función de envío de mensajes en Flutter

```
////////////////////////////////////////
//////// Function Send Message //////////
////////////////////////////////////////
onSendMessage() async {
  final user = Provider.of<auth.User?>(context, listen: false);
  var contentMessage = message.text;

  if (message.text.isNotEmpty) {
    Map<String, dynamic> dataMessage = {
      "idSender": user!.uid,
      "message": contentMessage,
      "time": DateTime.now().millisecondsSinceEpoch,
      "timeSystem": FieldValue.serverTimestamp(),
      "idReceiver": widget.idReceiver,
    };
    await firestore
      .collection('chatRoom')
      .doc(widget.chatId)
      .collection('chats')
      .add(dataMessage)
      .then(
        (value) {
          firestore
            .collection('users')
            .doc(user.uid)
            .collection('chats')
            .doc(widget.chatId)
            .set(
              {
```

```

        "fullName": widget.nameReciever,
        "idReciever": widget.idReciever,
        "message": contentMessage,
        "time": DateTime.now().millisecondsSinceEpoch,
    },
);
firestore
    .collection('users')
    .doc(widget.idReciever)
    .collection('chats')
    .doc(widget.chatId)
    .set(
    {
        "fullName": widget.nameSender,
        "idReciever": user.uid,
        "message": contentMessage,
        "time": DateTime.now().millisecondsSinceEpoch,
    },
);
    },
);
}
}

```

5.3 Obteniendo lista de Conversaciones en Flutter

```

////////////////////////////////////
////////// CONVERSATIONS //////////
////////////////////////////////////
Center conversations() {
    final user = Provider.of<auth.User?>(context, listen: false);
    var idUser = user!.uid;
    return Center(
        child: ConstrainedBox(
            constraints: const BoxConstraints(minWidth: 50, maxWidth:
450),
            child: Column(
                children: [
                    Expanded(
                        child: StreamBuilder<QuerySnapshot>(
                            stream: firestore

```

```
.collection('users')
.doc(idUser)
.collection('chats')
.orderBy("time", descending: true)
.snapshots(),
builder: (BuildContext context,
    AsyncSnapshot<QuerySnapshot> snapshot) {
  if (snapshot.hasError) {
    return Text('Error: ${snapshot.error}');
  }
  switch (snapshot.connectionState) {
    case ConnectionState.waiting:
      return const Center(child:
CircularProgressIndicator());
    default:
      return ListView.builder(
        physics: const BouncingScrollPhysics(),
        itemCount: snapshot.data!.docs.length,
        itemBuilder: (context, index) {
          //data
          var data = snapshot.data!.docs[index];

          //Generator ChatRoomId
          String chatId =
            chatRoomId(user.uid,
data['idReciever']);

          //Format Time Message
          var time =
DateTime.fromMillisecondsSinceEpoch(data['time']);
          var hora = DateFormat('hh:mm
a').format(time);

          return ConversationListItem(
            onTap: () {
              nextScreen(
                context,
                ChatScreen(
                  chatId: chatId,
                  nameReciever: data['fullName'],
                  nameSender: userData!.fullName
```

```

        idReciever: data['idReciever'],
      ));
    },
    fullname: data['fullName'],
    message: data['message'],
    time: hora.toString(),
  );
},
);
}
},
),
),
],
),
),
);
}

```

5.4 Obteniendo lista de Descubrir personas (Discover) en Flutter

```

////////////////////////////////////
////////// DISCOVER //////////
////////////////////////////////////
Center discover() {
  final user = Provider.of<auth.User?>(context, listen: false);
  return Center(
    child: ConstrainedBox(
      constraints: const BoxConstraints(minWidth: 50, maxWidth:
450),
      child: Column(
        children: [
          Expanded(
            child: StreamBuilder<QuerySnapshot>(
              stream: firestore
                .collection('users')
                .where("idUser", isNotEqualTo: user!.uid)
                .snapshots(),
              builder: (BuildContext context,
                AsyncSnapshot<QuerySnapshot> snapshot) {
                if (snapshot.hasError) {

```

```

        return Text('Error: ${snapshot.error}');
    }

    switch (snapshot.connectionState) {
        case ConnectionState.waiting:
            return const Center(child:
CircularProgressIndicator());
        default:
            return ListView.builder(
                physics: const BouncingScrollPhysics(),
                itemCount: snapshot.data!.docs.length,
                itemBuilder: (context, index) {
                    //data
                    var data = snapshot.data!.docs[index];

                    //Generator ChatRoomId
                    String chatId = chatRoomId(user.uid,
data['idUser']);

                    return ChatListItem(
                        onTap: () {
                            nextScreen(
                                context,
                                ChatScreen(
                                    chatId: chatId,
                                    nameReciever: data['fullName'],
                                    nameSender: userData!.fullName,
                                    idReciever: data['idUser'],
                                ));
                            setState(() {
                                _selectedIndex = 0;
                            });
                        },
                        fullname: data['fullName']);
                },
            );
    }
},
),
],
),
),

```

```
);
}
```

5.5 Obteniendo lista de Chats en Flutter

```
////////////////////////////////////
// Message Bubble //
////////////////////////////////////
Expanded(
  child: StreamBuilder<QuerySnapshot>(
    stream: firestore
      .collection('chatRoom')
      .doc(widget.chatId)
      .collection('chats')
      .orderBy("timeSystem", descending: true)
      .snapshots(),
    builder: (BuildContext context,
      AsyncSnapshot<QuerySnapshot> snapshot) {
      if (snapshot.data != null) {
        return ListView.builder(
          physics: const BouncingScrollPhysics(),
          reverse: true,
          itemCount: snapshot.data!.docs.length,
          itemBuilder: (context, index) {
            Map<String, dynamic> map =
snapshot.data!.docs[index]
              .data() as Map<String, dynamic>;
            return messageBubble(size, map, user!.uid);
          },
        );
      } else {
        return Container();
      }
    },
  ),
),
```