FIREBASE

INICIO SESION

Claro, voy a explicarte el código línea por línea. Este código es parte de una actividad en una aplicación Android que maneja el inicio de sesión de un usuario utilizando Firebase Authentication.

1. Declaración de variables

```
EditText correo, pass;

FirebaseAuth auth;
```

- **correo y pass**: Son variables para referirse a los campos de texto donde el usuario va a ingresar su correo y su contraseña.
- **auth**: Es una instancia de FirebaseAuth, la cual se usa para autenticar a los usuarios con Firebase (en este caso, para iniciar sesión).

2. Método onCreate

```
@Override

protected void onCreate(Bundle savedInstanceState) {

   super.onCreate(savedInstanceState);

   EdgeToEdge.enable(this);

   setContentView(R.layout.activity_inicio_sesion);
```

- **onCreate**: Este método es llamado cuando la actividad es creada. Es donde generalmente se configura la interfaz de usuario y las variables.
- **EdgeToEdge.enable(this)**: Esta línea permite que la interfaz de usuario ocupe todo el espacio de la pantalla, incluyendo las áreas de las "system bars" (barra de estado, barra de navegación).
- **setContentView(R.layout.activity_inicio_sesion)**: Se establece el layout de la actividad (activity_inicio_sesion), que define cómo se verá la pantalla.

3. Ajustes de márgenes para las "system bars"

```
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

   Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

   v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

   return insets;

});
```

- Aquí, se ajustan los márgenes de la interfaz para asegurarse de que el contenido no se superponga con las barras del sistema (como la barra de estado o la barra de navegación). Esto es importante especialmente en dispositivos con pantallas más grandes o "notches" (muescas) en la pantalla.

4. Inicialización de las vistas

```
correo = findViewById(R.id.correoInicio);

pass = findViewById(R.id.passInicio);

auth = FirebaseAuth.getInstance();
```

- **correo = findViewById(R.id.correoInicio)**: Obtiene una referencia al campo de texto para el correo electrónico.
- **pass = findViewById(R.id.passInicio)**: Obtiene una referencia al campo de texto para la contraseña.
- **auth = FirebaseAuth.getInstance()**: Inicializa la instancia de FirebaseAuth que se utilizará para gestionar el inicio de sesión.

5. Método iniciarSesion

```
public void iniciarSesion(View view){
   String correoUsuario = correo.getText().toString().trim();
   String passUsuario = pass.getText().toString().trim();


   if(correoUsuario.isEmpty() || passUsuario.isEmpty()){
      Toast.makeText(this, "Ingrese los datos", Toast.LENGTH_SHORT).show();
   }else{
      inicioUsuario(correoUsuario, passUsuario);
   }
}
```

- **iniciarSesion**: Este es el método que se ejecuta cuando el usuario hace clic en el botón de inicio de sesión.
    - **correoUsuario y passUsuario**: Obtiene los valores que el usuario ha introducido en los campos de correo y contraseña.
    - **Comprobación de campos vacíos**: Si alguno de los campos está vacío, muestra un Toast notificando que debe ingresar los datos.

- **Llamada a inicioUsuario**: Si ambos campos están completos, se llama al método inicioUsuario para proceder con la autenticación.

6. Método inicioUsuario

```
private void inicioUsuario(String correoUsuario, String passUsuario) {
   auth.signInWithEmailAndPassword(correoUsuario,
passUsuario).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
      @Override
      public void onComplete(@NonNull Task<AuthResult> task) {
         if(task.isSuccessful()){
            finish();
            startActivity(new Intent(InicioSesion.this, MainActivity.class));
            Toast.makeText(InicioSesion.this, "Bienvenido/a",
Toast.LENGTH_SHORT).show();
         }else{
            Toast.makeText(InicioSesion.this, "Error", Toast.LENGTH_SHORT).show();
         }
      }
   }).addOnFailureListener(new OnFailureListener() {
      @Override
      public void onFailure(@NonNull Exception e) {
         Toast.makeText(InicioSesion.this, "Error al iniciar sesion",
Toast.LENGTH_SHORT).show();
      }
   });
}
```

- **inicioUsuario**: Este método realiza el intento de inicio de sesión con los datos proporcionados (correo y contraseña).
    - **auth.signInWithEmailAndPassword(correoUsuario, passUsuario)**: Llama al método de Firebase para intentar autenticar al usuario con su correo y contraseña.

- **addOnCompleteListener**: Se ejecuta cuando la operación de inicio de sesión ha terminado. Si la autenticación es exitosa:
  - **finish()**: Cierra la actividad de inicio de sesión.
  - **startActivity(new Intent(...))**: Inicia la actividad principal (MainActivity), redirigiendo al usuario a la pantalla principal de la app.
  - **Toast.makeText(...)**: Muestra un mensaje de bienvenida.
- Si la autenticación falla, muestra un mensaje de error.
- **addOnFailureListener**: En caso de que la operación de inicio de sesión falle por alguna razón, se muestra un mensaje de error adicional.

7. Método registrarse

```
public void registrarse(View view){
   startActivity(new Intent(InicioSesion.this, Registro.class));
}
```

- **registrarse**: Este método se ejecuta cuando el usuario hace clic en el botón para registrarse. Redirige al usuario a la actividad de registro (Registro.class), donde podrá crear una cuenta si aún no tiene una.

Resumen:

Este código es para una pantalla de inicio de sesión en una aplicación Android que usa Firebase Authentication. Los pasos son:

1. El usuario ingresa su correo y contraseña.
2. Si los campos no están vacíos, se intenta iniciar sesión con Firebase.
3. Si el inicio de sesión es exitoso, se redirige al usuario a la pantalla principal de la aplicación (MainActivity).
4. Si el inicio de sesión falla, se muestra un mensaje de error.
5. Si el usuario no tiene cuenta, puede ir a la pantalla de registro.

Además, se realizan ajustes para asegurar que la interfaz no se solape con las barras del sistema en dispositivos con pantallas grandes o con notch.

REGISTRO USUARIO

Este código es para una actividad de registro en una aplicación Android que utiliza Firebase para registrar a un nuevo usuario. Te explicaré línea por línea qué hace cada parte del código.

1. Declaración de variables

EditText nombre, correo, pass;

FirebaseFirestore firestore;

FirebaseAuth auth;

- **nombre, correo, pass**: Son variables para almacenar las referencias a los campos de texto donde el usuario ingresará su nombre, correo electrónico y contraseña, respectivamente.
- **firestore**: Es una instancia de FirebaseFirestore, que se utilizará para almacenar los datos del usuario en la base de datos de Firebase.
- **auth**: Es una instancia de FirebaseAuth, que se usa para manejar la autenticación del usuario (en este caso, el registro).

2. Método onCreate

@Override

protected void onCreate(Bundle savedInstanceState) {

  super.onCreate(savedInstanceState);

  EdgeToEdge.enable(this);

  setContentView(R.layout.activity_registro);

- **onCreate**: Este método es el que se ejecuta cuando la actividad se crea.
- **EdgeToEdge.enable(this)**: Permite que la interfaz de usuario ocupe todo el espacio de la pantalla, incluyendo áreas como las barras de sistema (barra de estado, barra de navegación) si se está utilizando el diseño "edge-to-edge".
- **setContentView(R.layout.activity_registro)**: Establece el archivo de diseño (XML) que se usará para la actividad, en este caso activity_registro.

3. Configuración de los márgenes para los "system bars"

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

  Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

  v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

```
    return insets;
});
```

- Aquí, se ajusta el diseño para que no se solapen con las barras del sistema (barra de estado y barra de navegación).
- ViewCompat.setOnApplyWindowInsetsListener: Se usa para escuchar los cambios en los márgenes de la ventana y ajustar el contenido de la interfaz de usuario.

4. Inicialización de las vistas

```
nombre = findViewById(R.id.nombre);
correo = findViewById(R.id.correo);
pass = findViewById(R.id.pass);
firestore = FirebaseFirestore.getInstance();
auth = FirebaseAuth.getInstance();
```

- Aquí se obtienen las referencias a los elementos de la interfaz (campo de texto para nombre, correo, y contraseña) utilizando findViewById. También se inicializan las instancias de FirebaseFirestore y FirebaseAuth.

5. Método registrarme

```
public void registrarme(View view) {
    String nombreUsuario = nombre.getText().toString().trim();
    String correoUsuario = correo.getText().toString().trim();
    String passUsuario = pass.getText().toString().trim();

    if (nombreUsuario.isEmpty() || correoUsuario.isEmpty() || passUsuario.isEmpty()) {
        Toast.makeText(this, "Complete todos los campos", Toast.LENGTH_SHORT).show();
    } else {
        registroUsuario(nombreUsuario, correoUsuario, passUsuario);
    }
}
```

- **registrarme**: Este es el método que se ejecuta cuando el usuario hace clic en el botón de registro. Toma los valores de los campos de texto y los verifica:

- Si alguno de los campos está vacío, muestra un mensaje (Toast) pidiendo que complete todos los campos.
- Si todos los campos están completos, llama al método registroUsuario.

6. Método registroUsuario

```
private void registroUsuario(String nombreUsuario, String correoUsuario, String passUsuario) {
   auth.createUserWithEmailAndPassword(correoUsuario, passUsuario).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
      @Override
      public void onComplete(@NonNull Task<AuthResult> task) {
         String id = auth.getCurrentUser().getUid();
         Map<String, Object> map = new HashMap<>();
         map.put("id", id);
         map.put("nombre", nombreUsuario);
         map.put("correo", correoUsuario);
         map.put("pass", passUsuario);


firestore.collection("usuario").document(id).set(map).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
               finish();
               startActivity(new Intent(Registro.this, MainActivity.class));
               Toast.makeText(Registro.this, "Usuario registrado", Toast.LENGTH_SHORT).show();
            }
         }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
```

```
                Toast.makeText(Registro.this, "Error al guardar",
Toast.LENGTH_SHORT).show();
            }
        });
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        Toast.makeText(Registro.this, "Error al registrar", Toast.LENGTH_SHORT).show();
    }
});
}
```

- **registroUsuario**: Este método es el que se encarga de registrar al usuario en Firebase.
    - Primero, crea una cuenta de usuario con el correo y la contraseña proporcionados mediante auth.createUserWithEmailAndPassword().
    - Si la creación del usuario es exitosa:
        - Obtiene el ID del usuario (auth.getCurrentUser().getUid()).
        - Crea un Map con los datos del usuario (ID, nombre, correo, y contraseña).
        - Guarda esos datos en una colección de Firestore llamada usuario utilizando firestore.collection("usuario").document(id).set(map).
        - Si la escritura en Firestore es exitosa, muestra un mensaje de éxito y redirige al usuario a la pantalla principal (MainActivity).
        - Si ocurre un error en cualquiera de las operaciones, se muestra un mensaje de error usando Toast.

Resumen

Este código permite que un usuario se registre en la aplicación mediante su correo electrónico y una contraseña. Los datos del usuario se almacenan en Firebase Authentication para la autenticación y en Firebase Firestore para la base de datos.

Además, maneja correctamente los errores y muestra mensajes de retroalimentación al usuario.

¡Claro! Para crear un método listarUsuarios que liste todos los usuarios registrados en Firestore, primero necesitamos asegurarnos de que tienes una colección en Firestore donde se almacenan los usuarios. En el código que proporcionaste, los usuarios se almacenan en la colección "usuario".

Aquí te dejo un ejemplo de cómo podrías implementar el método listarUsuarios en la clase Registro, utilizando Firestore para obtener todos los documentos de la colección usuario y luego mostrarlos, por ejemplo, en un Toast o en un Log para propósitos de depuración:

Código para el método listarUsuarios:

```
private void listarUsuarios() {
   firestore.collection("usuario")  // Accedemos a la colección "usuario"
      .get()  // Obtiene todos los documentos de la colección
      .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
         @Override
         public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
               QuerySnapshot querySnapshot = task.getResult();  // Obtenemos el resultado
               StringBuilder usuariosList = new StringBuilder();

               // Recorremos todos los documentos (usuarios) en la colección
               for (DocumentSnapshot document : querySnapshot.getDocuments()) {
                  // Extraemos los datos de cada usuario (puedes agregar más campos si es necesario)
                  String id = document.getString("id");
                  String nombre = document.getString("nombre");
                  String correo = document.getString("correo");

                  // Añadimos la información de cada usuario a un StringBuilder
```

```
                usuariosList.append("ID: ").append(id)
                        .append(", Nombre: ").append(nombre)
                        .append(", Correo: ").append(correo)
                        .append("\n");
                }

                // Mostramos los usuarios listados en un Toast o en Log (para depuración)
                if (usuariosList.length() > 0) {
                    Toast.makeText(Registro.this, usuariosList.toString(),
Toast.LENGTH_LONG).show();
                } else {
                    Toast.makeText(Registro.this, "No hay usuarios registrados.",
Toast.LENGTH_SHORT).show();
                }
            } else {
                Toast.makeText(Registro.this, "Error al obtener los usuarios.",
Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

Explicación del código:

1. **Acceso a la colección "usuario"**:
    - Usamos firestore.collection("usuario") para acceder a la colección de usuarios en Firestore.
2. **Obtención de los documentos**:
    - Usamos el método .get() para obtener todos los documentos de la colección. Esto devuelve un QuerySnapshot, que contiene todos los documentos encontrados.
3. **Procesamiento de los documentos**:

- Si la operación es exitosa (task.isSuccessful()), recorremos los documentos obtenidos con querySnapshot.getDocuments().
- Para cada documento, usamos document.getString("campo") para obtener los valores de los campos almacenados en Firestore (en este caso, id, nombre y correo).

4. **Mostrar los usuarios**:
   - La información de todos los usuarios se concatena en un StringBuilder, que luego se muestra en un Toast. Si no se encuentran usuarios, se muestra un mensaje indicando que no hay usuarios registrados.
5. **Manejo de errores**:
   - Si la operación falla, mostramos un mensaje de error.

¿Cómo llamar al método listarUsuarios?

Puedes llamar a este método en cualquier parte de la actividad, por ejemplo, después de un botón de "Ver usuarios", o llamarlo directamente desde el método onCreate para que se ejecute cuando la actividad se cargue.

Ejemplo de cómo llamar a listarUsuarios:

```
// Llamada al método desde algún botón, por ejemplo:

Button btnVerUsuarios = findViewById(R.id.btnVerUsuarios);

btnVerUsuarios.setOnClickListener(new View.OnClickListener() {

  @Override

  public void onClick(View v) {

    listarUsuarios();

  }

});
```

Consideraciones:

- Este ejemplo muestra la lista de usuarios en un Toast. Si necesitas mostrar los usuarios en una interfaz de usuario más avanzada (por ejemplo, en una lista), podrías usar un RecyclerView para mostrar los datos de manera más estructurada.
- Asegúrate de manejar correctamente los permisos de acceso a Firestore si estás usando reglas de seguridad.

Este es un enfoque básico para listar los usuarios registrados en Firestore. ¡Puedes modificarlo según lo que necesites!

MAIN

¡Claro! Aquí tienes una explicación línea por línea de este código, que corresponde a una actividad principal (MainActivity) en una aplicación Android que maneja datos de mercados y permite iniciar sesión, cerrar sesión e insertar información en Firestore.

1. Declaración de variables

FirebaseAuth auth;

EditText id, nombre, ubicacion, inicio, fin;

FirebaseFirestore firestore;

- **auth**: Instancia de FirebaseAuth utilizada para gestionar la autenticación de usuarios (en este caso, para cerrar sesión).
- **id, nombre, ubicacion, inicio, fin**: Son referencias a campos de texto (EditText) donde el usuario ingresará la información del mercado.
- **firestore**: Instancia de FirebaseFirestore, que se utiliza para interactuar con la base de datos de Firestore.

2. Método onCreate

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_main);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -
> {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom);
        return insets;
    });
```

- **onCreate**: Este es el método principal que se ejecuta cuando la actividad es creada. Aquí se inicializan las vistas y se configuran elementos de la interfaz.

- **EdgeToEdge.enable(this)**: Permite que el contenido de la aplicación se extienda por toda la pantalla, incluso debajo de las "system bars" (barra de estado y barra de navegación).
- **setContentView(R.layout.activity_main)**: Establece el layout de la actividad, que define el diseño visual de la pantalla de la actividad principal.
- **ViewCompat.setOnApplyWindowInsetsListener(...)**: Ajusta el padding (relleno) del contenido para evitar que se solape con las barras del sistema (como la barra de estado o la barra de navegación).

3. Inicialización de las vistas

```
auth = FirebaseAuth.getInstance();

firestore = FirebaseFirestore.getInstance();

id = findViewById(R.id.idMercado);

nombre = findViewById(R.id.nombreMercado);

ubicacion = findViewById(R.id.ubiMercado);

inicio = findViewById(R.id.inicioMercado);

fin = findViewById(R.id.finMercado);
```

- **auth = FirebaseAuth.getInstance()**: Inicializa la instancia de FirebaseAuth, lo que permite manejar la autenticación del usuario.
- **firestore = FirebaseFirestore.getInstance()**: Inicializa la instancia de FirebaseFirestore, lo que permite interactuar con la base de datos de Firestore.
- **Inicialización de las vistas**: Cada campo de texto (EditText) se inicializa con findViewById, para asociarlo con su respectivo componente en el XML.

4. Método cerrarSesion

```
public void cerrarSesion(View view) {
   auth.signOut();
   finish();
   startActivity(new Intent(this, InicioSesion.class));
}
```

- **cerrarSesion**: Este método se ejecuta cuando el usuario quiere cerrar sesión.
    - **auth.signOut()**: Cierra la sesión del usuario actual utilizando Firebase Authentication.

- **finish()**: Finaliza la actividad actual (en este caso, MainActivity).
- **startActivity(new Intent(this, InicioSesion.class))**: Redirige al usuario a la actividad de inicio de sesión (InicioSesion), lo que permite que el usuario inicie sesión nuevamente si lo desea.

5. Método insertarMercado

```
public void insertarMercado(View view) {
   String idMercado = id.getText().toString().trim();
   String nombreMercado = nombre.getText().toString().trim();
   String ubicacionMercado = ubicacion.getText().toString().trim();
   String inicioMercado = inicio.getText().toString().trim();
   String finMercado = fin.getText().toString().trim();
   agregarMercado(idMercado, nombreMercado, ubicacionMercado, inicioMercado, finMercado);
}
```

- **insertarMercado**: Este método se ejecuta cuando el usuario hace clic en el botón para insertar la información del mercado en Firestore.
  - **Obtención de datos**: Los valores de los campos EditText se obtienen con getText().toString().trim(), lo que asegura que se eliminen los espacios en blanco al inicio y al final.
  - **Llamada a agregarMercado**: Después de obtener los datos, se pasa esa información al método agregarMercado para agregarla a Firestore.

6. Método agregarMercado

```
private void agregarMercado(String idMercado, String nombreMercado, String ubicacionMercado, String inicioMercado, String finMercado) {
   Map<String, Object> map = new HashMap<>();
   map.put("id", idMercado);
   map.put("nombre", nombreMercado);
   map.put("ubicacion", ubicacionMercado);
   map.put("inicio", inicioMercado);
   map.put("fin", finMercado);
```

```
    firestore.collection("mercado").add(map).addOnCompleteListener(new
OnCompleteListener<DocumentReference>() {
        @Override
        public void onComplete(@NonNull Task<DocumentReference> task) {
            Toast.makeText(MainActivity.this, "Mercado Registrado",
Toast.LENGTH_SHORT).show();
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(MainActivity.this, "Error al insertar",
Toast.LENGTH_SHORT).show();
        }
    });
}
```

- **agregarMercado**: Este método agrega la información del mercado a Firestore.
    - **Map<String, Object> map = new HashMap<>()**: Crea un mapa que contiene los datos del mercado, como el id, nombre, ubicacion, inicio, y fin.
    - **firestore.collection("mercado").add(map)**: Agrega los datos del mercado en la colección "mercado" en Firestore. Este es un proceso asincrónico.
    - **addOnCompleteListener**: Se ejecuta cuando la operación de agregar el mercado se completa. Si la inserción es exitosa, muestra un Toast que dice "Mercado Registrado".
    - **addOnFailureListener**: Si la operación falla por alguna razón, muestra un Toast con el mensaje "Error al insertar".

Resumen general del flujo de la aplicación:

1. **Pantalla Principal (MainActivity)**:
    - El usuario puede insertar datos sobre un mercado (como id, nombre, ubicación, horarios de apertura y cierre).

- Al hacer clic en el botón de "Insertar Mercado", los datos se recogen de los campos de texto y se almacenan en Firestore.
- También puede cerrar sesión, lo que terminará su sesión actual y lo llevará a la pantalla de inicio de sesión.

2. **Firebase Authentication y Firestore**:
    - **Autenticación**: Se utiliza para gestionar la sesión del usuario.
    - **Firestore**: Se utiliza para almacenar los datos de los mercados.

Este código gestiona la sesión de un usuario, permite almacenar información sobre mercados en Firestore y manejar algunas interacciones básicas con la interfaz de usuario.