Proyecto Integrador

Android Material Design, CoordinatorLayout, Firebase, Providers

Android Material Design

Es un conjunto de directrices de diseño creado por Google para aplicaciones móviles y web, basado en:

- **Superficies y profundidad**: Usa sombras y capas para simular objetos físicos.
- **Animaciones fluidas**: Transiciones suaves y respuestas táctiles.
- **Colores vivos**: Paletas contrastantes y atractivas.
- **Tipografía clara**: Jerarquía visual con fuentes como Roboto.
- **Componentes reutilizables**: Botones, menús y otros elementos listos para usar.
- **Diseño adaptable**: Optimizado para diferentes tamaños de pantalla.

Material Design crea experiencias consistentes, modernas y fáciles de usar.

Algunos recursos y configuraciones

-Descargar las imágenes del proyecto ejemplo aquí

-Recurso ShapeOfView aquí

-Base de datos en Firebase aquí

- Dependencias en gradle

```
    }
    buildFeatures{
        viewBinding=true
    }
}
dependencies {
    implementation("androidx.appcompat:appcompat:1.7.0")
    implementation("com.google.android.material:material:1.12.0")
    implementation("androidx.activity:activity-ktx:1.7.2")
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")
    implementation("androidx.cardview:cardview:1.0.0") // Usar CardView de AndroidX
    implementation("androidx.recyclerview:recyclerview:1.2.1")
    implementation(libs.firebase.auth) // Usar RecyclerView de AndroidX
    implementation("com.google.firebase:firebase-firestore:25.1.1")
    implementation(libs.activity)

    // Dependencias de testing
    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")

    // Glide para manejar imágenes
    implementation("com.github.bumptech.glide:glide:4.12.0")
    annotationProcessor("com.github.bumptech.glide:compiler:4.12.0")
    implementation ("io.github.florent37:shapeofview:1.4.7")

}
```

CoordinatorLayout

El **CoordinatorLayout** permite que las vistas "se comuniquen" entre sí de manera más dinámica. Por ejemplo, puedes hacer que una vista se desplace o cambie de tamaño en función de la interacción con otra (como cuando un RecyclerView se desplaza y la AppBar se oculta o muestra).

Compatibilidad con diseños de Material Design

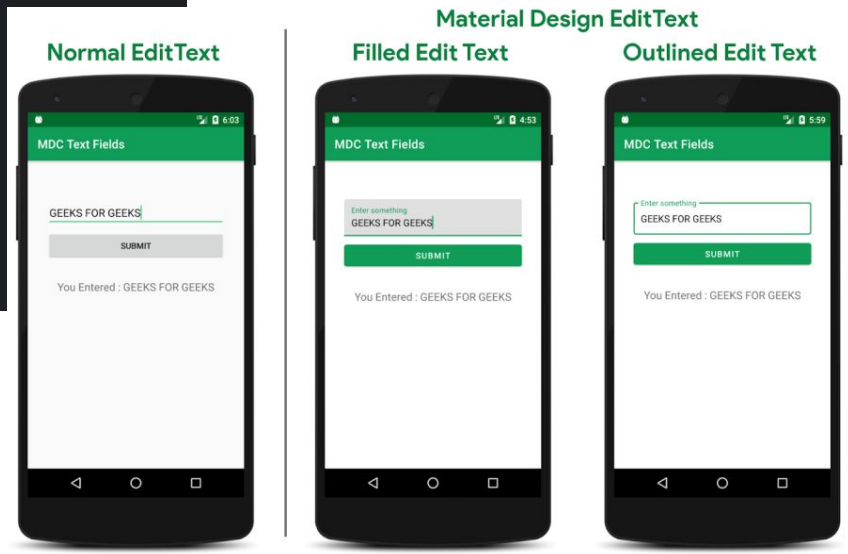
Desplazamiento flexible

En un **CoordinatorLayout**, puedes incluir otros **layouts** o vistas.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <LinearLayout …
    </LinearLayout>
    <LinearLayout …
    </LinearLayout>
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

google.android.material.textfield.TextInputLayout

```
<com.google.android.material.textfield.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
    app:boxBackgroundMode="none"
    >
<com.google.android.material.textfield.TextInputEditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textEmailAddress"
    android:hint="Correo electrónico"
    android:singleLine="true"
   />
```



Se utiliza para mejorar la apariencia y funcionalidad de los campos de texto, proporcionando características como etiquetas flotantes, mensajes de error, y compatibilidad con entradas de tipo contraseña. Mejora la experiencia visual y de usabilidad en formularios.

Seguimos modelo MVVM

Crearemos los paquetes:

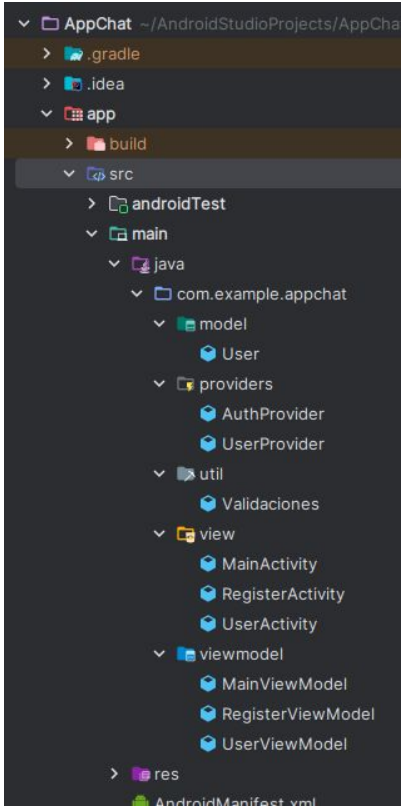
model: Aquí colocas las clases que representan entidades

view: Esta carpeta contiene las actividades o fragmentos que forman la interfaz de usuario

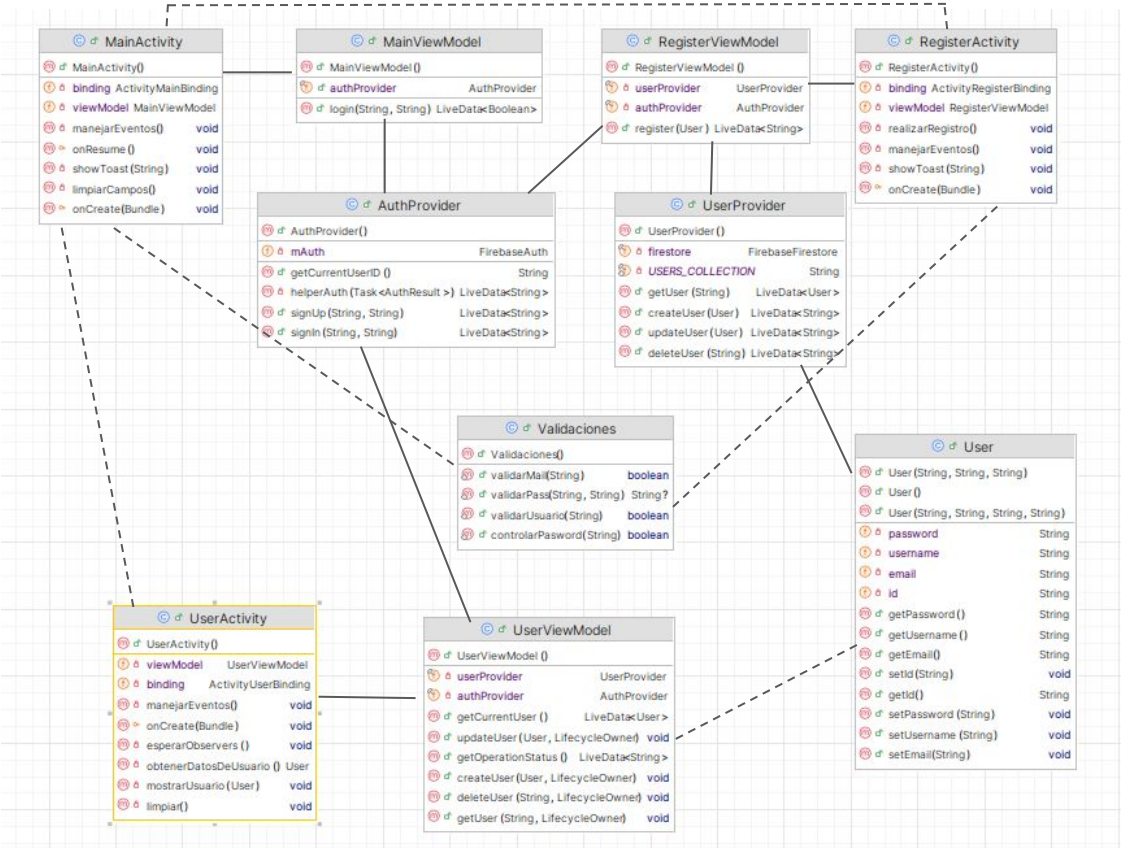
viewmodel: gestionan la lógica de negocio y expone los datos de los repositorios a la UI.

providers: Esta carpeta contiene los repositorios que interactúan directamente con Firebase para realizar operaciones CRUD.

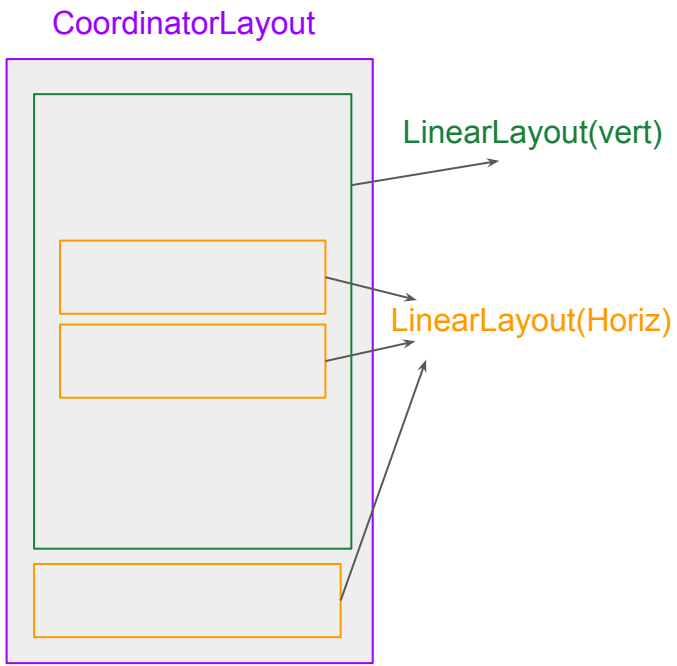
útil: clases de utilidad

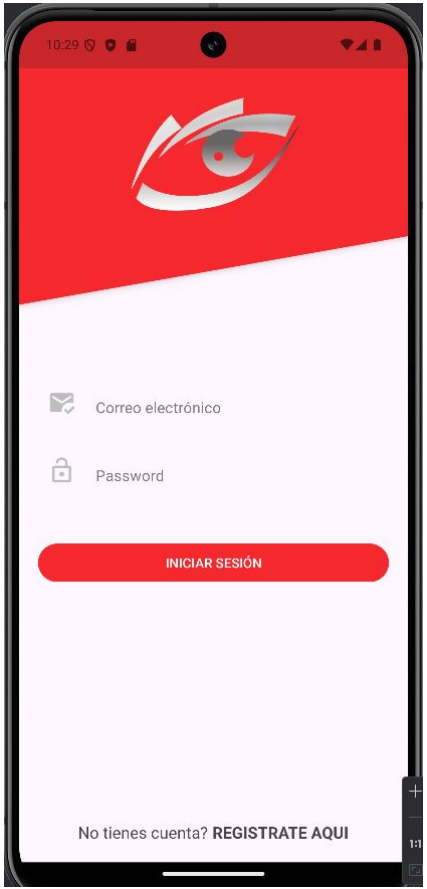


Las clases por ahora…

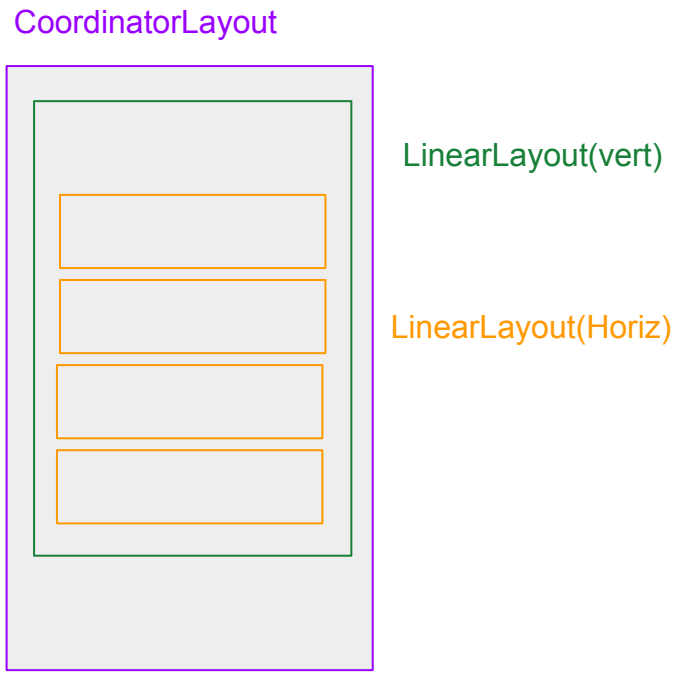


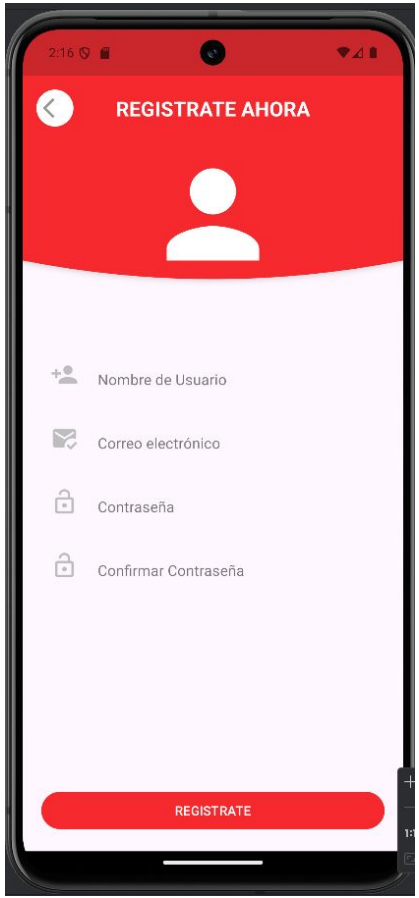
Diseñamos la vista login - Video



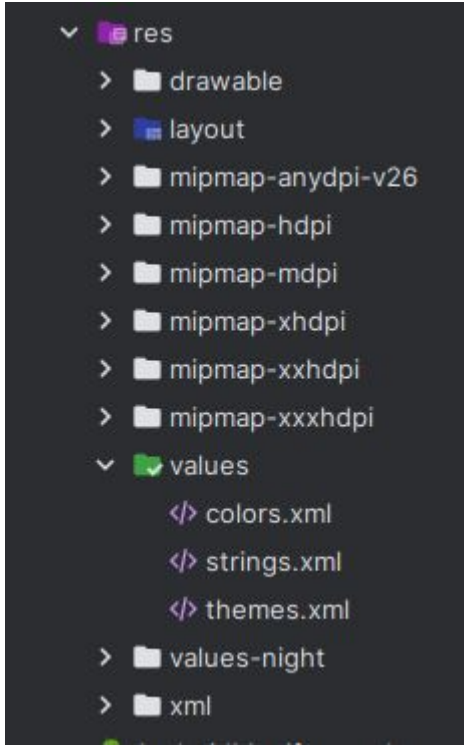


Diseñamos la vista Registrar





Personalizamos el color de la barra de status



themes.xml

```
1  <resources xmlns:tools="http://schemas.android.com/tools">
2      <!-- Base application theme. -->
3      <style name="Base.Theme.AppRedSocial" parent="Theme.Material3.DayNight.NoActionBar">
4          <item name="android:statusBarColor">@color/red_osc</item>
5      </style>
6
7      <style name="Theme.AppRedSocial" parent="Base.Theme.AppRedSocial" />
8  </resources>
```

colors.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <color name="black">#FF000000</color>
4      <color name="primary">#484444</color>
5      <color name="white">#FFFFFFFF</color>
6      <color name="red">#F6292F</color>
7      <color name="red_osc">#C9272C</color>
8  </resources>
```

Clase User en el paquete model

```java
public class User {  29 usages  mariel
    private String id;  3 usages
    private String username;  4 usages
    private String email;  4 usages
    private String password;  4 usages

    public User() {  no usages  mariel
        // Constructor vacío necesario para Firebase
    }
    public User(String username, String email, String password) {...}
    public User(String id, String username, String email, String password) {...
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }
    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }
    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
}
```

Clase Validaciones del paquete util

```java
public class Validaciones {  7 usages  mariel
    public static boolean validarUsuario(String usuario) {  1 usage  mariel
        return usuario != null && !usuario.isEmpty() && usuario.length() > 3;
    }
    public static boolean validarMail(String email) {  2 usages  mariel
        String emailPattern = "^[A-Za-z0-9+_.-]+@[A-Za-z0-9.-]+$";
        return email != null && email.matches(emailPattern);
    }

    public static String validarPass(String pass, String pass1) {  1 usage  mariel
        if (pass == null || pass.isEmpty() || pass1 == null || pass1.isEmpty()) {
            return "La contraseña no puede estar vacía";
        }
        if (pass.length() < 6) {
            return "La contraseña debe tener al menos 6 caracteres";
        }
        if (!pass.equals(pass1)) {
            return "Las contraseñas no coinciden";
        }
        return null; // Contraseña válida
    }
    public static boolean controlarPasword(String pass){  1 usage  mariel

        return  (pass!=null && pass.length()>=6);
    }
}
```

providers

Integrando Firebase a Android

Conexión a Firebase

Video parte 2

```java
public class AuthProvider {  9 usages  mariel *
    private FirebaseAuth mAuth;  7 usages
    public AuthProvider() { mAuth = FirebaseAuth.getInstance(); }
    private LiveData<String> helperAuth(Task<AuthResult> task) {  2 usages  mariel
        MutableLiveData<String> authResult = new MutableLiveData<>();
        task.addOnCompleteListener(new OnCompleteListener<AuthResult>() {  mariel
            @Override  mariel
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful() && mAuth.getCurrentUser() != null) {
                    authResult.setValue(mAuth.getCurrentUser().getUid());
                } else {
                    authResult.setValue(null);
                }
            }
        });
        return authResult;
    }
```

```java
    // Inicio de sesión a Firebase
    public LiveData<String> signIn(String email, String password) {  1 usage  mariel
        return helperAuth(mAuth.signInWithEmailAndPassword(email, password));
    }
    // Registro en Firebase
    public LiveData<String> signUp(String email, String password) {  2 usages  mariel
        return helperAuth(mAuth.createUserWithEmailAndPassword(email, password));
    }
    public String getCurrentUserID() {  no usages  mariel
        return mAuth.getCurrentUser() != null ? mAuth.getCurrentUser().getUid() : null;
    }
}
```

UserProvider -CreateUser

```java
public class UserProvider {  6 usages  mariel *
    private final FirebaseFirestore firestore;  5 usages
    private static final String USERS_COLLECTION = "users";  4 usages
    public UserProvider() {  2 usages  mariel *
        firestore = FirebaseFirestore.getInstance();
    }

    public LiveData<String> createUser(User user) {  2 usages  mariel
        MutableLiveData<String> result = new MutableLiveData<>();
        firestore.collection(USERS_COLLECTION).document(user.getId()).set(user)
                .addOnCompleteListener(task -> {
                    if (task.isSuccessful()) {
                        result.setValue("Usuario creado correctamente");
                    } else {
                        result.setValue("Error al crear usuario");
                    }
                });
        return result;
    }
```

UserProvider - getUser

```java
public LiveData<User> getUser(String mail) {  1 usage  mariel
    MutableLiveData<User> userData = new MutableLiveData<>();
    firestore.collection(USERS_COLLECTION) CollectionReference
            .whereEqualTo( field: "email", mail) Query
            .get() Task<QuerySnapshot>
            .addOnSuccessListener(queryDocumentSnapshots -> {
                if (!queryDocumentSnapshots.isEmpty()) {
                    DocumentSnapshot document = queryDocumentSnapshots.getDocuments().get(0);
                    User user = document.toObject(User.class);
                    if (user != null) {
                        Log.d( tag: "UserProvider",  msg: "Usuario encontrado: " + user.getEmail());
                    } else {
                        Log.d( tag: "UserProvider",  msg: "Error: documento encontrado, pero usuario es null");
                    }
                    userData.setValue(user);
                } else {
                    Log.d( tag: "UserProvider",  msg: "Usuario no encontrado en Firestore con email: " + mail);
                    userData.setValue(null);
                }
            })
            .addOnFailureListener(e -> {
                Log.e( tag: "UserProvider",  msg: "Error en la consulta a Firestore: ", e);
                userData.setValue(null);
            });
    return userData;
}
```

UserProvider - updateUser

```
public LiveData<String> updateUser(User user) {  1 usage  mariel
    MutableLiveData<String> result = new MutableLiveData<>();
    firestore.collection(USERS_COLLECTION).document(user.getId()).set(user)
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    result.setValue("Usuario actualizado correctamente");
                } else {
                    result.setValue("Error al actualizar usuario");
                }
            });
    return result;
}
```

UserProvider - deleteUser

```
public LiveData<String> deleteUser(String userId) {  1 usage  mariel
    MutableLiveData<String> result = new MutableLiveData<>();
    firestore.collection(USERS_COLLECTION).document(userId).delete()
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    result.setValue("Usuario eliminado correctamente");
                } else {
                    result.setValue("Error al eliminar usuario");
                }
            });
    return result;
}
```

activities y sus viewModels

MainActivity

```java
public class MainActivity extends AppCompatActivity {  mariel *
    private ActivityMainBinding binding;
    private MainViewModel viewModel;
    @Override  mariel *
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        viewModel = new ViewModelProvider( owner: this).get(MainViewModel.class);
        manejarEventos();
    }
    private void manejarEventos() {...}

    private void showToast(String message) {  mariel
        Toast.makeText( context: MainActivity.this, message, Toast.LENGTH_LONG).show();
    }
   @Override  new *
   protected void onResume() {
        super.onResume();
        limpiarCampos();
   }
   private void limpiarCampos() {  mariel *
        binding.itUsuario.setText("");
        binding.itPassword.setText("");

    }
}
```

MainActivity - manejarEventos()

```java
private void manejarEventos() { 1 usage  mariel *
    binding.tvRegistro.setOnClickListener(new View.OnClickListener() {  mariel
        @Override  mariel
        public void onClick(View v) {
            Intent intent = new Intent( packageContext: MainActivity.this, RegisterActivity.class);
            startActivity(intent);
        }
    });
    binding.btLogin.setOnClickListener(new View.OnClickListener() {  mariel *
        @Override  mariel *
        public void onClick(View v) {
            String email = binding.itUsuario.getText().toString().trim();
            String pass = binding.itPassword.getText().toString().trim();

            if (!Validaciones.validarMail(email)) {
                showToast("Email incorrecto");
                return;
            }
            if (!Validaciones.controlarPasword(pass)) {
                showToast("Password incorrecto");
                return;
            }
```

```java
            // Observa el resultado del login
            viewModel.login(email, pass).observe( owner: MainActivity.this, loginSuccessful -> {
                if (loginSuccessful) {
                    Intent intent=new Intent( packageContext: MainActivity.this,UserActivity.class);
                    startActivity(intent);
                } else {
                    showToast("Login fallido");
                }
            });
        }
    });
}
```

MainViewModel

```
public class MainViewModel extends ViewModel {  3 usages  mariel *
    public final AuthProvider authProvider;  2 usages

    public MainViewModel(){  no usages  mariel *
        authProvider=new AuthProvider();
    }

    public LiveData<Boolean> login(String email, String password) {  1 usage  mariel
        MutableLiveData<Boolean> loginResult = new MutableLiveData<>();
        authProvider.signIn(email, password).observeForever(userId -> {
            loginResult.setValue(userId != null);
        });
        return loginResult;
    }

}
```

RegisterActivity

```java
public class RegisterActivity extends AppCompatActivity {  mariel
    private ActivityRegisterBinding binding;  8 usages
    private RegisterViewModel viewModel;  2 usages
    @Override  mariel
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityRegisterBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        viewModel = new ViewModelProvider( owner: this).get(RegisterViewModel.class);
        manejarEventos();
    }
    private void manejarEventos() {...}

    private void realizarRegistro() {...}


    private void showToast(String message) {  4 usages  mariel
        Toast.makeText( context: RegisterActivity.this, message, Toast.LENGTH_LONG).show();
        finish();
    }
}
```

RegisterActivity - manejarEventos()

```
private void manejarEventos() {  1 usage  mariel
    // Evento volver a login
    binding.circleImageBack.setOnClickListener(new View.OnClickListener() {  mariel
        @Override  mariel
        public void onClick(View v) {
            finish();
        }
    });

    // Evento de registro
    binding.btRegistrar.setOnClickListener(new View.OnClickListener() {  mariel
        @Override  mariel
        public void onClick(View v) { realizarRegistro(); }
    });
}
```

RegisterActivity - realizarRegistro()

```java
private void realizarRegistro() { 1 usage mariel
    String usuario = binding.itUsuario.getText().toString().trim();
    String email = binding.itEmail.getText().toString().trim();
    String pass = binding.itPassword.getText().toString().trim();
    String pass1 = binding.itPassword1.getText().toString().trim();

    if (!Validaciones.validarUsuario(usuario)) {
        showToast("Usuario incorrecto");
        return;
    }

    if (!Validaciones.validarMail(email)) {
        showToast("El correo no es válido");
        return;
    }

    String passError = Validaciones.validarPass(pass, pass1);
    if (passError != null) {
        showToast(passError);
        return;
    }

    User user = new User(usuario, email, pass);
    viewModel.register(user).observe( owner: this, result -> {
        showToast(result);
    });
}
```

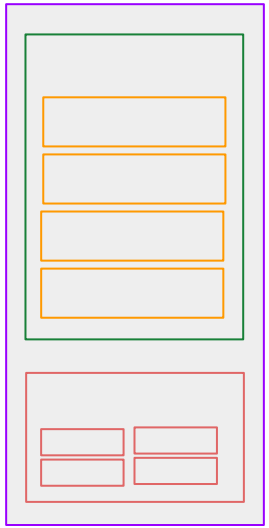
RegisterViewModel

```java
public class RegisterViewModel extends ViewModel {  3 usages  mariel *
    private final AuthProvider authProvider;  2 usages
    private final UserProvider userProvider;  2 usages
    public RegisterViewModel() {  no usages  mariel
        authProvider = new AuthProvider();
        userProvider = new UserProvider();
    }
    public LiveData<String> register(User user) {  1 usage  mariel *
        MutableLiveData<String> registerResult = new MutableLiveData<>();

        // Primero registra al usuario en Firebase Authentication
        authProvider.signUp(user.getEmail(), user.getPassword()).observeForever(new Observer<String>() {
            @Override  mariel
            public void onChanged(String uid) {
                if (uid != null) {
                    user.setId(uid);
                    userProvider.createUser(user).observeForever(new Observer<String>() {  mariel
                        @Override  mariel
                        public void onChanged(String result) { registerResult.setValue(result); }
                    });
                } else {
                    registerResult.setValue("Error en la autenticación");
                }
            }
        });
        return registerResult;
    }
}
```

activity_user.xml

CoordinatorLayout



LinearLayout(vert)

LinearLayout(Horiz)

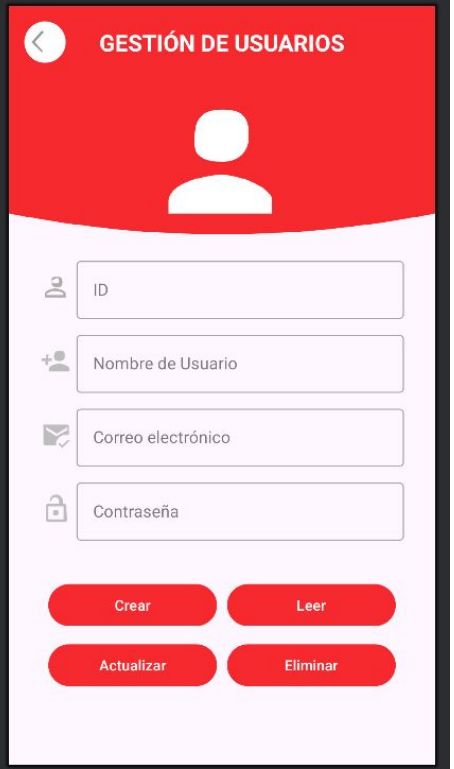
gridLayout

```android:layout_width="match_parent"
<!-- Botones de CRUD -->
<GridLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="30dp"
    android:columnCount="2"
    android:rowCount="2"
    android:layout_gravity="center"
    android:padding="8dp">

    <Button
        android:id="@+id/btnCreateUser"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_columnWeight="1"
        android:text="Crear"
        android:backgroundTint="@color/red"
        android:textColor="@color/white" />

    <Button
        android:id="@+id/btnReadUser"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_columnWeight="1"
        android:text="Leer"
        android:backgroundTint="@color/red"
        android:textColor="@color/white"
        android:layout_marginLeft="10dp" />

    <Button
        android:id="@+id/btnUpdateUser"
        android:layout_width="0dp"
```



UserActivity

Los métodos esperarObservers()

y

manejarEventos()

los mostramos en las otras diapositivas

```java
public class UserActivity extends AppCompatActivity {  mariel *
    private ActivityUserBinding binding;  21 usages
    private UserViewModel viewModel;  7 usages
    @Override  mariel *
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityUserBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        viewModel = new ViewModelProvider( owner: this).get(UserViewModel.class);
        esperarObservers();
        manejarEventos();
    }
    private void esperarObservers() {...}
    private void manejarEventos() {...}
    private User obtenerDatosDeUsuario() {  2 usages  mariel
        String username = binding.itUsuario.getText().toString();
        String email = binding.itEmail.getText().toString().trim();
        String id = binding.itId.getText().toString().trim();
        String password = binding.itPassword.getText().toString().trim();
        return new User(id, username, email, password);
    }
    private void mostrarUsuario(User user) {  1 usage  mariel *
        binding.itUsuario.setText(user.getUsername());
        binding.itEmail.setText(user.getEmail());
        binding.itId.setText(user.getId());
        binding.itPassword.setText(user.getPassword());
        Log.d( tag: "mostrar",  msg: user.getId()+"-"+user.getUsername());
    }
    private void limpiar() {  1 usage  mariel
        binding.itUsuario.setText("");
        binding.itEmail.setText("");
        binding.itId.setText("");
        binding.itPassword.setText("");
    }
}
```

UserActivity - esperarObservers()

```
private void esperarObservers() {  1 usage  mariel *
    viewModel.getOperationStatus().observe( owner: this, new Observer<String>() {  mariel
        @Override  mariel
        public void onChanged(String status) {
            Toast.makeText( context: UserActivity.this, status, Toast.LENGTH_SHORT).show();
            limpiar();
        }
    });
    viewModel.getCurrentUser().observe( owner: this, new Observer<User>() {  mariel *
        @Override  mariel *
        public void onChanged(User user) {
            if (user != null) {
                mostrarUsuario(user);
            }
        }
    });
}
```

UserActivity - manejarEventos()

```java
private void manejarEventos() { 1 usage  mariel *
    binding.btnCreateUser.setOnClickListener(new View.OnClickListener() {  mariel
        @Override  mariel
        public void onClick(View v) {
            User usuario = obtenerDatosDeUsuario();
            viewModel.createUser(usuario, lifecycleOwner: UserActivity.this);
        }
    });
    binding.btnUpdateUser.setOnClickListener(new View.OnClickListener() {
        @Override  mariel
        public void onClick(View v) {
            User usuario = obtenerDatosDeUsuario();
            viewModel.updateUser(usuario, lifecycleOwner: UserActivity.this);
        }
    });
```

```java
    binding.btnDeleteUser.setOnClickListener(new View.OnClickListener() {  mariel
        @Override  mariel
        public void onClick(View v) {
            String id = binding.itId.getText().toString().trim();
            viewModel.deleteUser(id, lifecycleOwner: UserActivity.this);
        }
    });
    binding.btnReadUser.setOnClickListener(new View.OnClickListener() {  mariel *
        @Override  mariel *
        public void onClick(View v) {
            String email = binding.itEmail.getText().toString().trim();
            viewModel.getUser(email, lifecycleOwner: UserActivity.this);
        }
    });
    binding.circleImageBack.setOnClickListener(new View.OnClickListener() {  mariel *
        @Override  new *
        public void onClick(View v) { finish(); }
    });
}
```

UserViewModel

```java
public class UserViewModel extends ViewModel {  3 usages  mariel *
    private final AuthProvider authProvider;  2 usages
    private final UserProvider userProvider;  5 usages
    private final MutableLiveData<User> currentUser;  3 usages
    private final MutableLiveData<String> estado;  8 usages
    public UserViewModel() {  no usages  mariel *
        authProvider = new AuthProvider();
        userProvider = new UserProvider();
        estado=new MutableLiveData<>();
        currentUser=new MutableLiveData<>();
    }
    public LiveData<User> getCurrentUser() {return currentUser;  }  1 usage  new *
    public LiveData<String> getOperationStatus() { return estado; }

    public void createUser(User user, LifecycleOwner lifecycleOwner) {  1 usage  mariel *
        authProvider.signUp(user.getEmail(), user.getPassword()).observe(lifecycleOwner, uid -> {
            if (uid != null) {
                user.setId(uid);
                userProvider.createUser(user).observe(lifecycleOwner, status -> {
                    if (status != null) {
                        estado.setValue(status);

                    } else {
                        estado.setValue("Error al crear usuario en Firestore");
                    }
                });
            } else {
                estado.setValue("Error al registrar usuario en FirebaseAuth");
            }
        });
    }
```

UserViewModel -más métodos

```
    public void updateUser(User user, LifecycleOwner lifecycleOwner) {  1 usage  mariel *
        LiveData<String> result = userProvider.updateUser(user);
        result.observe(lifecycleOwner, new Observer<String>() {  mariel *
            @Override  mariel *
            public void onChanged(String status) { estado.setValue(result.getValue()); }
        });
    }

    public void deleteUser(String userId, LifecycleOwner lifecycleOwner) {  1 usage  mariel *
        LiveData<String> result = userProvider.deleteUser(userId);
        result.observe(lifecycleOwner, new Observer<String>() {  mariel *
            @Override  mariel *
            public void onChanged(String status) { estado.setValue(status); }
        });
    }

    public void getUser(String email, LifecycleOwner lifecycleOwner) {  1 usage  mariel *
        LiveData<User> user = userProvider.getUser(email);
        user.observe(lifecycleOwner, new Observer<User>() {  mariel *
            @Override  mariel *
            public void onChanged(User foundUser) {
                if (foundUser != null) {
                    Log.d( tag: "User Info",  msg: "ID: " + foundUser.getId() + ", Username: " + foundUser.getUsername());
                    currentUser.setValue(foundUser);
                } else {
                    estado.setValue("No encontrado");
                }
            }
        });
    }
}
```