//Aqui abaixo temos a implementação de uma classe interna que gerenciará cada item interno a esta classe  
//private class UserItem extends Item<ViewHolder> {  
private class UserItem extends Item {  
  
 //Aqui abaixo é criado um objeto "user" da classe "User"  
 private final User user;  
 //Aqui abaixo é criado um construtor para poder pegar o usuário  
 private UserItem(User user) {  
 this.user = user;  
 }  
  
 @Override  
 public void bind(@NonNull GroupieViewHolder viewHolder, int position) {  
 //  
 TextView txtNomeUsuario = viewHolder.itemView.findViewById(R.id.*idTextViewUsuario*);  
 ImageView imagemFoto = viewHolder.itemView.findViewById(R.id.*idImagemAvatar*);  
  
 //Aqui abaixo temos o objeto "txtNomeUsuario" setando o método "setText()" trazer o nome do usuário em questão  
 txtNomeUsuario.setText(user.getUsername());  
  
 //Aqui abaixo pegamos a classe "Picasso" que setará o método "get()" de buscar ou trazer  
 Picasso.*get*()  
 //Aqui abaixo temos o carregamento da foto pelo objeto "user" e o método "getProfileUrl()"  
 .load(user.getProfileUrl())  
 .into(imagemFoto);  
 }  
  
 @Override  
 public int getLayout() {  
 return R.layout.*item\_user*;  
 }  
}

//Aqui abaixo temos a implementação de uma classe interna que gerenciará cada item interno a esta classe  
//private class UserItem extends Item<ViewHolder> {  
private class UserItem extends Item<ViewHolder> {  
  
 //Aqui abaixo é criado um objeto "user" da classe "User"  
 private final User user;  
 //Aqui abaixo é criado um construtor para poder pegar o usuário  
 private UserItem(User user) {  
 this.user = user;  
 }  
  
 @Override  
 public void bind(@NonNull ViewHolder viewHolder, int position) {  
 //  
 TextView txtNomeUsuario = viewHolder.itemView.findViewById(R.id.*idTextViewUsuario*);  
 ImageView imagemFoto = viewHolder.itemView.findViewById(R.id.*idImagemAvatar*);  
  
 //  
 txtNomeUsuario.setText(user.getUsername());  
  
 //Aqui abaixo pegamos a classe "Picasso" que setará o método "get()" de buscar ou trazer  
 Picasso.*get*()  
 //Aqui abaixo temos o carregamento da foto pelo objeto "user" e o método "getProfileUrl()"  
 .load(user.getProfileUrl())  
 .into(imagemFoto);  
 }   
  
 @Override  
 public int getLayout() {  
 return R.layout.*item\_user*;  
 }  
}

Arquivo RegistrarActivity.java

package com.ramerips\_cursos;  
  
//##Observação importante: Com dois cliques na tecla "SHIFT" temos um atalho para todos os arquivos da aplicação ##//  
  
//<!--###- TECLAS DE ATALHOS PARA LOCALIZAR CLASSES OU ARQUIVOS.-###-->  
//<!--##- Obs: CTRL+N -> Localiza uma "Classe", CTRL+SHIFT+N -> Localiza um "Arquivo" e CTRL+SHIFT+F -> Localiza um "Caminho" -##-->  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.graphics.Bitmap;  
import android.graphics.drawable.BitmapDrawable;  
import android.net.Uri;  
import android.os.Bundle;  
import android.provider.MediaStore;  
import android.util.Log;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ImageView;  
import android.widget.Toast;  
  
import com.google.android.gms.tasks.OnCompleteListener;  
import com.google.android.gms.tasks.OnFailureListener;  
import com.google.android.gms.tasks.OnSuccessListener;  
import com.google.android.gms.tasks.Task;  
import com.google.firebase.auth.AuthResult;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.firestore.DocumentReference;  
import com.google.firebase.firestore.FirebaseFirestore;  
import com.google.firebase.storage.FirebaseStorage;  
import com.google.firebase.storage.StorageReference;  
import com.google.firebase.storage.UploadTask;  
  
import java.io.IOException;  
import java.util.UUID;  
  
import static android.content.Intent.*ACTION\_PICK*;  
  
public class RegistrarActivity extends AppCompatActivity {  
  
 //Aqui serão criados os atributos relacionados aos campos do layout "act\_login"  
 private EditText mEditText\_Nome;  
 private EditText mEditText\_Email;  
 private EditText mEditText\_Senha;  
 private Button mBtn\_Cadastrar;  
 private Button mBtn\_FotoUsuario;  
 private ImageView mImg\_Foto;  
 private Uri mSelectedUri;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*act\_registrar*);  
  
 //Aqui serão recuperados os dados inseridos nos campos do layout "act\_registrar.xml"  
 mEditText\_Nome = findViewById(R.id.*editText\_Nome*);  
 mEditText\_Email = findViewById(R.id.*editText\_Email*);  
 mEditText\_Senha = findViewById(R.id.*editText\_Senha*);  
 mBtn\_Cadastrar = findViewById(R.id.*btn\_Cadastrar*);  
 mBtn\_FotoUsuario = findViewById(R.id.*idBtnFotoUsuario*);  
 mImg\_Foto = findViewById(R.id.*idImg\_Foto*);  
  
  
 //Aqui será criado um evento de clique acionado pelo botão "mBtn\_FotoUsuario" do layout "act\_registrar.xml" para seleção da  
 //foto desejada no ambiente do sistema ou na galeria de imagens  
 mBtn\_FotoUsuario.setOnClickListener(new View.OnClickListener() {  
 @Override  
 //Aqui temos a função ou método "onClick()" para acionar o evento de click  
 public void onClick(View v) {  
 //Aqui estamos chamando o método ou função "selecionarFoto()", que seleciona a foto desejada no ambiente  
 //do sistema ou galeria de imagens por meio do objeto "mBtn\_FotoUsuario"  
 selecionarFoto();  
 }  
 });  
  
  
 //Aqui será criado um evento de clique acionado pelo botão "btn\_Cadastrar" do layout "act\_registrar.xml" para cadastrar um  
 //usuário desejado no ambiente do banco de dados "Firebase"  
 mBtn\_Cadastrar.setOnClickListener(new View.OnClickListener() {  
 @Override  
 //Aqui temos a função ou método "onClick()" para acionar o evento de click  
 public void onClick(View v) {  
 //Aqui abaixo é apenas para teste do desenvolvedor, onde será verificados no console do "6:Logcat ou no 4:Run"  
 Log.*i*("Teste", "Criar Usuário Funciona");  
 //Aqui chamamos o método ou função criar usuário "createUser()", para efetuar a sua execução junto ao Firebase  
 createUser();  
 }  
 });  
 }  
  
  
 //Aqui será implementado uma codificação que retornará um resultado em código, no caso, da imagem selecionada, de uma outra  
 //activity que foi a da galeria que selecionamos a imagem ou foto  
 @Override  
 protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
 //Aqui será verificado se o "requestCode" é igual a "0" para poder executar o conteúdo do "if" que na verdade este código  
 //representa o caminho onde esta imagem ou foto está armazenada  
 if(requestCode == 0){  
 //Aqui abaixo a variável "mSelectedUri" receberá os dados "data" por meio do método "getData()"  
 mSelectedUri = data.getData();  
  
 //Aqui abaixo será criado um objeto "objBitmap" do tipo "Bitmap" e que inicialmente receberá um valor nulo "null"  
 Bitmap objBitmap = null;  
  
 //Aqui abaixo temos a implementação do "try e catch", onde será capturada as excessões  
 try {  
 //Aqui o objeto "objBitmap" receberá todos os parâmetros necessários para captura da imagem selecionada, ou seja,  
 //receberá a imagem que foi selecionada propriamente dito  
 objBitmap = MediaStore.Images.Media.*getBitmap*(getContentResolver(), mSelectedUri);  
 //Aqui o objeto "mImg\_Foto" setará o método "setImageDrawable()" onde está a imagem, pegando-a para ser utilizada  
 mImg\_Foto.setImageDrawable(new BitmapDrawable(objBitmap));  
 //Aqui o objeto "mBtn\_FotoUsuario" setará o método "setAlpha(0)" de valor "0", ou seja, a imagem "mBtn\_FotoUsuario"  
 //irá desaparecer e surgirá a imagem "mImg\_Foto", pois na verdade ele está alí para poder ser pego a ação de click  
 mBtn\_FotoUsuario.setAlpha(0);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
  
 //## Selecionando uma imagem ou foto na galeria ou ambiente do sistema ##//  
 //Aqui será implementado o método que seleciona uma foto na galeria do sistema ou máquina  
 private void selecionarFoto(){  
 //Aqui será criada uma nova "Intent(Intenção)" de "ACTION\_PICK(Ação de se obter algo)", no caso seria uma imagem na galeria  
 Intent objIntent = new Intent(Intent.*ACTION\_PICK*);  
 //Aqui o objeto "objIntent" setará o método "setType()" para selecionar ou buscar o tipo de dados, no caso, buscar as imagens "image"  
 //e enfatizando que serão todos os tipos de imagens -> "\*"  
 objIntent.setType("image/\*");  
 //Aqui será passado como parâmetro o objeto intent "objIntent" criado e o código recebido pelo método "onActivityResult()"  
 startActivityForResult(objIntent, 0);  
 //Obs: Após selecionar a foto com o método "selecionarFoto()", é preciso resgatar ou pegar esta foto com o método "onActivityResult()"  
 //logo acima.  
 }  
  
  
 //### Aqui abaixo temos o início da função ou método criar usuário "createUser()" ###//  
 //Aqui temos a implementação de um método "createUser()", ou seja, uma função de criar um usuário novo  
 private void createUser() {  
 //Aqui abaixo as variáveis do tipo string(nome, email e senha), receberão os valores contidos nos dados recuperados  
 //por meio do método "getText()" e transformados em string por meio do método "toString()"  
 String nome = mEditText\_Nome.getText().toString();  
 String email = mEditText\_Email.getText().toString();  
 String senha = mEditText\_Senha.getText().toString();  
  
 //Aqui será verificado se os campos do formulário da senha e email estão inválidos ou nulos(Vazios), onde a mensagem do "Toast"  
 //só será impressa para o usuário caso estejam com problemas "Inválidos ou vazios".  
 if (nome == null || nome.isEmpty() || email == null || email.isEmpty() || senha == null || senha.isEmpty()){  
 //Aqui será implementado um estilo de mensagem volátil do tipo "Toast" que será exibida ao usuário, onde existe a possibilidade de  
 //ser do tipo "Toast.LENGTH\_LONG(Longa duração) e Toast.LENGTH\_SHORT(Rápida duração)", onde desaparece automaticamente.  
 Toast.*makeText*(this, "Nome, senha e email devem ser preenchidos corretamente", Toast.*LENGTH\_LONG*).show();  
 //Aqui temos o retorno de toda a contextualização desta função  
 return;  
 }  
  
 //## Temos que inserir a biblioteca do "FirebaseAuth" no arquivo "build.gradle(Module:app)" se ainda não existir ##//  
 //Aqui será implementada a criação de um "Id" de usuário criptografados e que receberá um email e uma senha por meio do "FirebaseAuth"  
 FirebaseAuth.*getInstance*().createUserWithEmailAndPassword(email, senha)  
 //Aqui será inserido no "Firebase" na aba "Authentication", um "email" e uma "senha" e também criado um "Id" de usuário criptografado  
 .addOnCompleteListener(new OnCompleteListener<AuthResult>(){  
 @Override  
 public void onComplete(@NonNull Task<AuthResult> task) {  
 //Aqui será verificado um possível sucesso na executada da "task(Tarefa)" por meio da autenticação junto ao "Firebase"  
 //caso sucesso, será impresso o resultado desta tarefa executada por meio dos métodos "getResult(), getUser() e getUid()".  
 if(task.isSuccessful()) {  
 //Aqui temos uma mensagem em modo "Log.i" com a impressão da tarefa que é mostada ao desenvolvedor no painel do  
 //"6: Logcat" com o conteúdo logo abaixo que cria um "id" criptografado para o usuário  
 Log.*i*("Teste", task.getResult().getUser().getUid()); //Aqui é apenas para teste do desenvolvedor  
  
 //Aqui será chamado o método "salvarUsuarioNoFirebase()"  
 salvarUsuarioNoFirebase();  
 //Aqui será implementado um estilo de mensagem volátil do tipo "Toast" que será exibida ao usuário, onde existe a possibilidade de  
 //ser do tipo "Toast.LENGTH\_LONG(Longa duração) e Toast.LENGTH\_SHORT(Rápida duração)", onde desaparece automaticamente.  
 Toast.*makeText*(RegistrarActivity.this, "Usuário Criado com Sucesso!", Toast.*LENGTH\_SHORT*).show();  
 }  
 //Obs:Verificar junto ao "Firebase" na aba "Authentication" a inclusão do "Email" e a criação do "Id" do usuário  
 //criptografado.  
  
 //Obs: Temos abaixo uma tradução de Inglês para Português  
 //1->The email address is badly formatted. -> O endereço de email está mal formatado.  
 //2->The given password is invalid. [ Password should be at least 6 characters ] -> A senha fornecida  
 //é inválida. [A senha deve ter pelo menos 6 caracteres]  
 //3->A network error (such as timeout, interrupted connection or unreachable host) has occurred. -> Ocorreu  
 //um erro de rede (como tempo limite, conexão interrompida ou host inacessível).  
 //4->The email address is already in use by another account. -> O endereço de email já está sendo usado por outra conta.  
 //5->PERMISSION\_DENIED: Missing or insufficient permissions. -> PERMISSÃO NEGADA: Permissões ausentes ou insuficientes.  
 }  
 })  
  
 //Aqui será adicionada ou capturada uma falha ou erro de "Exception" na execução desta autenticação ou "task(Tarefa)"  
 .addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
 //Aqui temos uma mensagem de erro em modo "Log.i" que é mostada ao desenvolvedor no painel do "6: Logcat"  
 //com o conteúdo logo abaixo ou mensagem de erro específica  
 Log.*i*("Teste", e.getMessage()); //Aqui é apenas para teste do desenvolvedor  
  
 //Obs: Tradução de Inglês para Português  
 //1->The email address is badly formatted. -> O endereço de email está mal formatado.  
 //2->The given password is invalid. [ Password should be at least 6 characters ] -> A senha fornecida  
 //é inválida. [A senha deve ter pelo menos 6 caracteres]  
 //3->A network error (such as timeout, interrupted connection or unreachable host) has occurred. -> Ocorreu  
 //um erro de rede (como tempo limite, conexão interrompida ou host inacessível).  
 //4->The email address is already in use by another account. -> O endereço de email já está sendo usado por outra conta.  
 //5->PERMISSION\_DENIED: Missing or insufficient permissions. -> PERMISSÃO NEGADA: Permissões ausentes ou insuficientes.  
 }  
 });  
 }  
 //### Aqui abaixo temos o fim da função ou método criar usuário "createUser()" ###//  
  
  
  
 //### Aqui abaixo temos o início da função ou método salvar usuário no Firebase "salvarUsuarioNoFirebase()" ###//  
 //Aqui será implementado um método que salvará os dados de um usuário cadastrado no "Firebase" mais a imagem de referência  
 private void salvarUsuarioNoFirebase() {  
 //Aqui abaixo temos uma variável "nomeArquivo" que receberá um código randômico(Aleatório) em formato de string, porém  
 //um "Hash(Confusão, misturado)", ou seja, com os caracteres(Alfanuméricos), com letras e números todos eles misturados  
 //que identificará unicamente cada um usuário distintamente  
 String nomeArquivo = UUID.*randomUUID*().toString();  
 //Aqui abaixo o objeto "objReference" do tipo "StorageReference" e que não muda(final), receberá uma imagem e nome do arquivo  
 final StorageReference objReference = FirebaseStorage.*getInstance*().getReference("/images/" + nomeArquivo);  
 //Com o objeto "objReference" pronto, a imagem será enviada para o "Storage" do Firebase por meio do método "putFile()" e seu  
 //parâmetro "mSelectedUri" do tipo "Uri", pois essa imagem será identificada por uma "Hash" aleatória alfanumerica  
 objReference.putFile(mSelectedUri)  
 //Aqui com o método "addOnSuccessListener()" verifica se realmente houve a transferência da cópia da imagem selecionada  
 //na galeria para o "Storage" do Firebase com sucesso  
 .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {  
 @Override  
 //Esse método abaixo entra automaticamente ao digitar o "new OnSuccessListener... " logo acima.  
 public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {  
 //Aqui o objeto "objReference" setará o método "getDownloadUrl()" para baixar uma cópia também da imagem que foi enviada  
 //ao "Firebase", por meio da "URL" criada, pois com essa referência você pode até visualizar esta imagem que está no "Storage"  
 //do Firebase em qualquer navegador.  
 objReference.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {  
 @Override  
 //Esse método abaixo "onSuccess()" é criado automaticamente, trazendo a referência da "URL" por meio dos parâmetros abaixo  
 //"URI uri" que detem esta "URL" da imagem recuperada em formato de string.  
 public void onSuccess(Uri uri) {  
 //Aqui abaixo temos uma mensagem do tipo string "Teste" em modo "Log.i" com a impressão da "URL -> uri.toString()" criada  
 //no "Storage do Firebase" que é mostada ao desenvolvedor no painel do "6: Logcat" com o link de acesso da mesma  
 Log.*i*("Teste", uri.toString());  
  
 //## Aqui abaixo regataremos um usuário cadastrado no Firebase do Google ##//  
 //Aqui temos a variável "uid"(usuário-id) que receberá toda a autenticação da instância do usuário  
 //criado no "Firebase do Google" em sua coleção já arquivada ou cadastrada  
 String uid = FirebaseAuth.*getInstance*().getUid();  
 //Aqui abaixo a variável "username" receberá o valor do "mEditText\_Nome" em formato de string  
 String username = mEditText\_Nome.getText().toString();  
 //Aqui abaixo a variável "profileUrl" receberá o "uri" em formato de string, ou seja, a "URL" da imagem cadastrada no Firebase  
 String profileUrl = uri.toString();  
  
 //Aqui será criado um novo objeto "user" com todos os seus atributos cadastrados no banco de dados "Firebase database"  
 User user = new User(uid, username, profileUrl);  
  
 //Aqui abaixo é o evento que escutará o sucesso da execução da tarefa caso insira um novo usuário  
 FirebaseFirestore.*getInstance*().collection("users")  
 .add(user)  
 .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {  
 @Override  
 public void onSuccess(DocumentReference documentReference) {  
 //Aqui abaixo temos uma mensagem "Teste" em modo "Log.i" com a impressão do "Id" do "usuário" em  
 //formato de string que é mostada ao desenvolvedor no painel do "6: Logcat" com o link de acesso da mesma  
 Log.*i*("Teste", documentReference.getId());  
  
 //Aqui é criado um novo objeto "objIntent" do tipo "Intent" que receberá uma nova página ou layout  
 //"MessagesActivity" e determinada a sua abertura pela "RegistrarActivity"  
 Intent objIntent = new Intent(RegistrarActivity.this, MessagesActivity.class);  
 //Aqui logo abaixo o objeto "objIntent" setará o método "setFlags()" que determina que a página ou layout  
 //a ser aberto seja fique em primeiro plano que é a "MessagesActivity" e a outra seja escondida ou ocultada  
 objIntent.setFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK* | Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  
 //Aqui temos a inicialização da activity "MessagesActivity" por meio do método "startActivity()"  
 startActivity(objIntent);  
 }  
 })  
 //Aqui abaixo é o evento que escutará o erro da execução da tarefa, caso não insira um novo usuário  
 .addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
 //Aqui abaixo temos uma mensagem "Teste" em modo "Log.i" com a impressão da "erro" em formato  
 //de string que é mostada ao desenvolvedor no painel do "6: Logcat" com o link de acesso da mesma  
 Log.*i*("Teste", e.getMessage());  
 }  
 });  
 }  
 });  
 }  
 })  
 //Aqui será adicionada ou capturada uma falha ou erro de "Exception" na execução desta autenticação ou "task(Tarefa)"  
 .addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
 //Aqui temos uma mensagem de erro em modo "Log.i" que é mostada ao desenvolvedor no painel do "6: Logcat"  
 //com o conteúdo logo abaixo ou mensagem de erro específica  
 Log.*i*("Teste", e.getMessage(), e);  
 }  
 });  
 }  
 //### Aqui abaixo temos o final da função ou método salvar usuário no Firebase "salvarUsuarioNoFirebase()" ###//  
}