PRÀCTICA 3

MARIA PEREGRINA USIETO

CURS 2022-2023

JUNY 2023

TecnoCampus
Escola Superior Politècnica

Centre adscrit a la

upf. Universitat Pompeu Fabra Barcelona

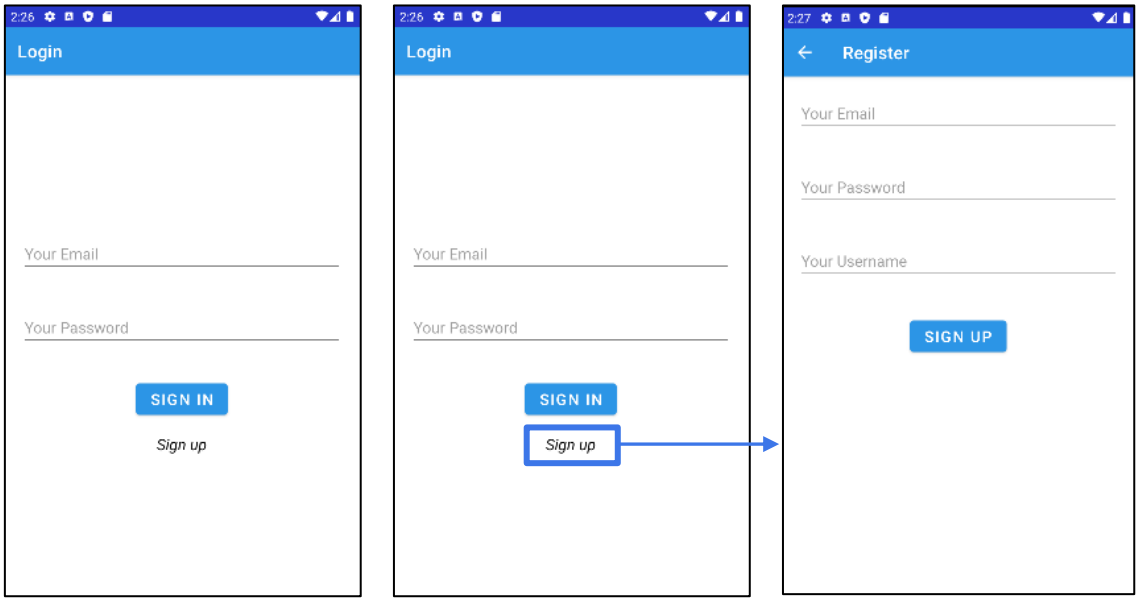
ÍNDEX

Explicació de l'Aplicació

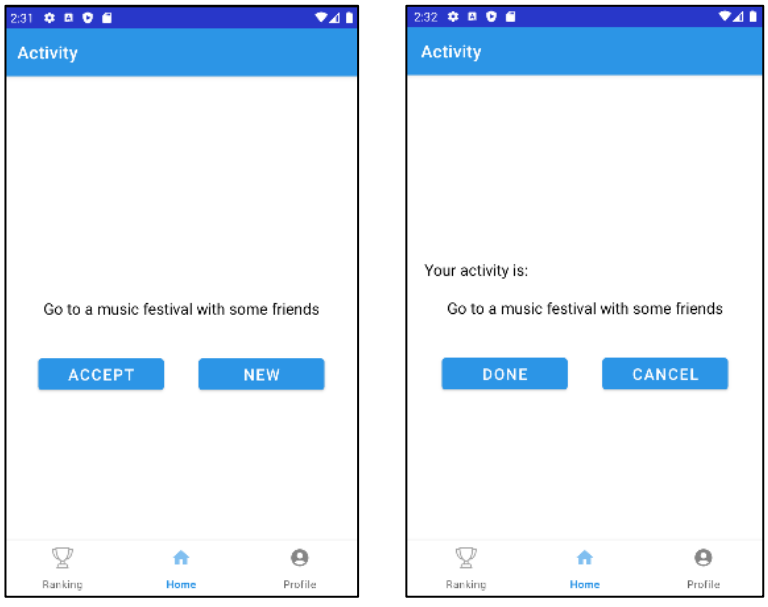
L'aplicació té com a objectiu principal oferir a l'usuari activitat per fer. Utilitzo una API que retorna activitats aleatòries, d'aquestes activitats ens dona diferent informació, en aquesta versió només utilitzem la Key i el Text de l'activitat.

L'aplicació es forma de tres pantalles principals (Ranking, Main i Profile) i una pantalla secundària (Configuration) amb dos fragments. A més, també inclou una pantalla d'inici de sessió i un altre amb un formulari de registre.

Si l'usuari encara no ha iniciat sessió, s'obrirà la pantalla d'iniciar sessió on l'usuari haurà d'introduir el correu i la contrasenya del seu compte. En el cas que no tingui compte, pot accedir al formulari de registre clicant al text "Sing up o Registrar".

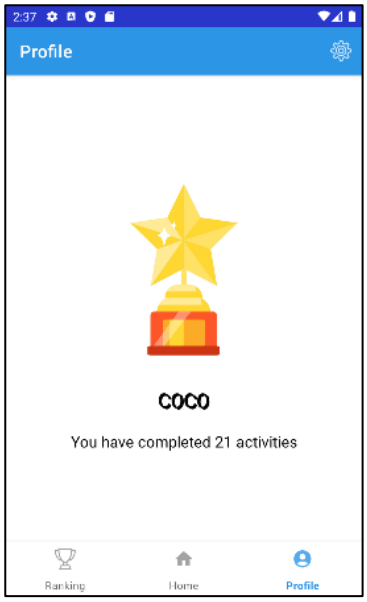


Inicialment, l'aplicació s'obrirà amb la pantalla MainActivity (si l'usuari ja està logejat) on l'usuari podrà escollir una activitat, i en el cas que ja tingui alguna activa, la mostrarà. Per tant, podríem dir que aquesta pantalla té dues versions, en la primera versió l'usuari ha d'escollir activitat i en la segona l'usuari ha de finalitzar o cancel·lar l'activitat activa.

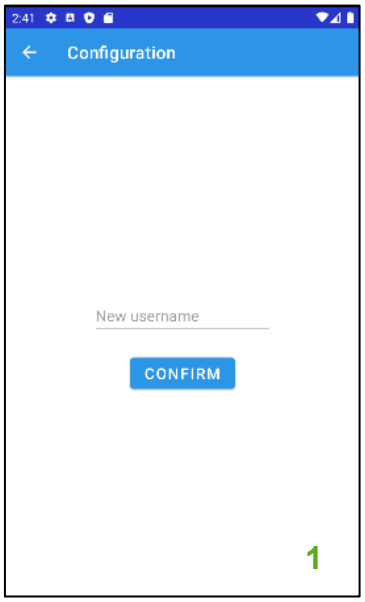


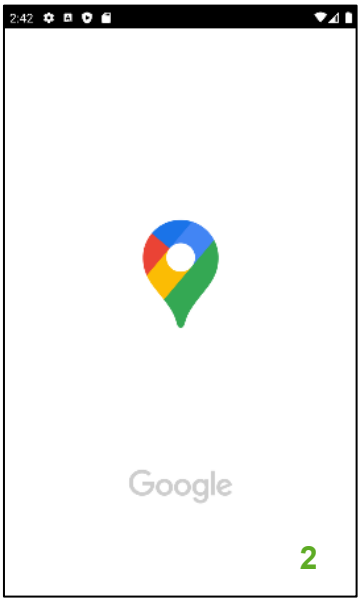
A baix de la pantalla podem veure un menú, on ens podem moure a les altres dues pantalles principals.

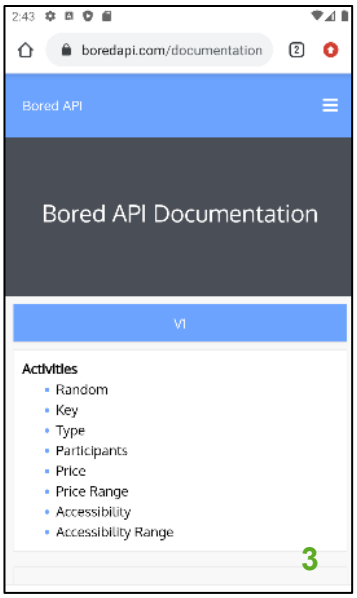
En el Ranking podrem veure un llistat d'usuaris ordenat pel nombre d'activitats que ha completat i en la pantalla Profile podem veure una imatge que variarà depenent del nombre d'activitats dutes a terme, el nom de l'usuari i ens indicarà quantes activitats ha completat.

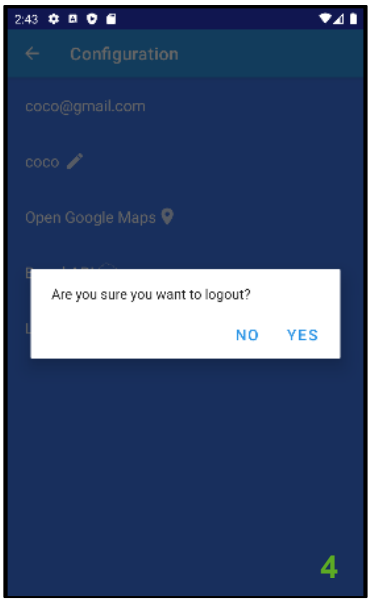


En el Perfil, podem veure en el menú superior la icona de configuració. Si fem clic sobre aquest, podrem veure informació de l'usuari, amb l'opció de modificar el nom d'usuari, també ens dona l'opció d'obrir el Google Maps i d'accedir a la documentació de l'Api que d'utilitzar. En últim lloc, donem l'opció de tancar sessió a l'usuari.









URL GITHUB

https://github.com/TCM2023Android/finalassigment-mperegrina

LoginActivity

Introducció

Pantalla per iniciar sessió. Es permet iniciar sessió amb un compte existent a partir del correu electrònic i la contrasenya. A continuació mostrarem els elements de la pantalla i el codi que creiem més rellevant.

Layout

El Layout es compon de dos EditText, un de tipus email i l'altre de tipus contrasenya. També tenim un botó per iniciar sessió i un TextView per anar a la pantalla de registre.

Codi

signUpClicked

En aquest mètode enviem a l'usuari a la pantalla de Registre.

```
public void signUpClicked(View view) {
    Intent intent = new Intent(getApplicationContext(),
                                        RegisterActivity.class);
    startActivity(intent);
}
```

signInClicked

Quan l'usuari fa clic al botó per iniciar sessió, s'executa aquesta funció. Abans de fer la crida al FirebaseAuth. Recollim les dades introduïdes per l'usuari i les validem. Si la validació és correcte, es cridarà el mètode login. En cas contrari, els mètodes de validació mostraran erros a l'usuari.

```
public void signInClicked(View view) {
    String email = etEmail.getText().toString();
    String password = etPassword.getText().toString();
    if(firstDataValidation(email, password))
        login(email, password);
}
```

Login

Aquest mètode fa la crida a FirebaseAuth per iniciar sessió. Podem veure que per paràmetre es passa el correu i la contrasenya que ha indicat l'usuari. Un cop fem la crida, comprovem el seu resultat. Si s'ha pogut realitzar exitosament l'inici de sessió, enviarem a l'usuari a la pantalla MainActivity, en cas contrari li indicarem a l'usuari per un missatge setError que és el que ha passat.

```
public void login(String email, String password) {
    mAuth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(this,
                        new OnCompleteListener<AuthResult>() {
         @Override
         public void onComplete(@NonNull Task<AuthResult> task) {
            if(task.isSuccessful()){
                Intent intent= new Intent(getApplicationContext(),
                                                    MainActivity.class);
                startActivity(intent);
            }
            else etEmail.setError(task.getException().getMessage());
         }
    });
}
```

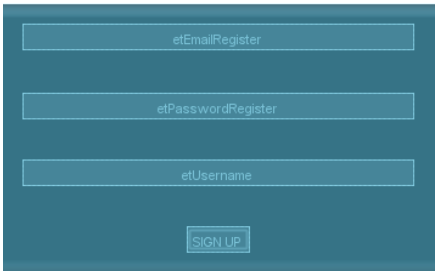
RegisterActivity

Introducció

Pantalla per Registrar un nou usuari. En el cas que l'usuari no tingui compte encara, la pot crear en aquesta pantalla de registre. Per registrar-se només haurà d'iniciar el correu i la contrasenya amb la qual podran iniciar sessió i el nom d'usuari de el compte. A continuació mostrarem els elements de la pantalla i el codi que creiem més rellevant.

Layout

La pantalla es compon de tres EditText, un per cada camp requerit per crear la nova compta i finalment un botó per confirmar el registre.



Codi

signUpClicked

Aquest mètode s'executa en capturar el clic sobre el botó per registrar-se. Podem veure que recull les dades que ens ha indicat l'usuari i les valida. Si són correctes, es crida al mètode registre, en cas contrari, els mètodes de validació avisaran a l'usuari amb missatges d'error (setError).

```
public void signUpClicked(View view) {
    String username = etUsername.getText().toString();
    String password = etPassword.getText().toString();
    String email = etEmail.getText().toString();
    if(firstDataValidation(username, password, email))
        register(username, password, email);
}
```

register

Aquest mètode fa la petició per crear un usuari al FirebaseAuth amb el correu i la contrasenya que ens ha indicat l'usuari. En capturar la resposta de la petició, mirem si s'ha pogut crear, en cas afirmatiu cridem al mètode addUserToDataBase, en cas contrari, mostrem a l'usuari el missatge d'error.

```
public void register(String username, String password, String email) {
    mAuth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(this, new
                        OnCompleteListener<AuthResult>() {
         @Override
         public void onComplete(@NonNull Task<AuthResult> task) {
             if(task.isSuccessful())
                    addUserToDataBase(email, username);
             else
                  etEmail.setError(task.getException().getMessage());
         }
    });
}
```

AddUserToDataBase

En aquest mètode afegim a la base de dades (FirebaseFirestore) un nou registre amb el correu i el nom d'usuari (Aquest registre podria emmagatzemar altres dades de l'usuari si fos necessari). En comprovar el resultat, si s'ha pogut afegir, enviem un Toast a l'usuari per avisar i seguidament l'enviem a la pantalla MainActivity. En cas contrari mostrem un missatge Log amb l'error.

```
private void addUserToDataBase(String email, String username) {
    Map<String, Object> user = new HashMap<>();
    user.put("email", email);
    user.put("username", username);

    db.collection("User")
            .add(user)
            .addOnSuccessListener(
                        new OnSuccessListener<DocumentReference>() {
              @Override
              public void onSuccess(DocumentReference
                                                documentReference) {
                 Toast.makeText(RegisterActivity.this,
                        getString(R.string.toast_register_ok),
                                        Toast.LENGTH_SHORT).show();
                 Intent intent = new Intent(getApplicationContext(),
                                                MainActivity.class);
                 startActivity(intent);
             }})
            .addOnFailureListener(new OnFailureListener() {
              @Override
              public void onFailure(@NonNull Exception e) {
                 Log.v("ERROR-AddUser:", e.getMessage());
                }
     });
}
```

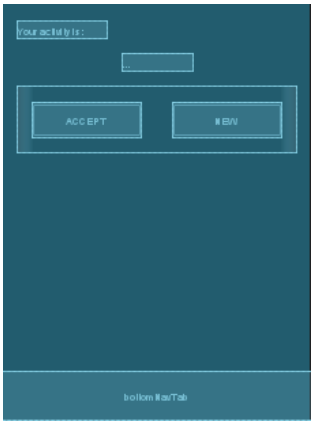
MainActivity

Introducció

Aquesta pantalla, tal com he indicat en l'explicació de l'aplicació, ens permet acceptar una activitat i en cas que l'usuari tingui una activitat activa, li mostra per pantalla l'activitat amb les opcions de finalitzar i cancel·lar. A més, conté un menú inferior on permet accedir a la pantalla del Ranking i del Perfil. A continuació mostrarem els elements de la pantalla i el codi que creiem més rellevant.

Layout

La pantalla es compon de dos TextView, un només es visualitzarà quan l'usuari tingui una activitat activa i l'altre l'utilitzem per visualitzar l'activitat. També conte dos botons que els he anomenat One i Two. El primer farà referència a l'acció d'acceptar o finalitzar una activitat i el botó dos farà referència a l'acció de nova activitat o de cancel·lar. En últim lloc, tenim el BottomNavigationView a la part inferior de la pantalla.



Codi

onCreate

Aquest no és el codi complet del mètode onCreate, aquí només mostro el més important on podrem veure l'ús de SharedPreferences, l'assignació del BottomNavigationView, i la inicialització de l'aplicació.

Podem veure que el primer que fem és mirar si hi ha un usuari identificat. En cas negatiu, s'envia directament l'usuari a la pantalla d'inici de sessió, en cas contrari el que fem és guardar a SharedPreferences el correu de l'usuari i cridar al mètode prepareUserInformation.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    setTitle(R.string.title_activity);
    sharedPreferences = getSharedPreferences("Preferences",
                                            Context.MODE_PRIVATE);
    bottomNavigationView = findViewById(R.id.bottomNavTab);
    bottomNavigationView.setOnNavigationItemSelectedListener(this);
    bottomNavigationView.setSelectedItemId(R.id.it_home);

    FirebaseUser currentUser = firebaseAuth.getCurrentUser();
    if(currentUser == null){
        intent=new Intent(this,LoginActivity.class);
        startActivity(intent);
    }
    else{
        prepareUserInformation(currentUser.getEmail());
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putString("email", currentUser.getEmail());
        editor.commit();
    }
}
```

onNavigationItemSelected

Aquest mètode captura el clic de BottomNavigationView. Podem veure que té un Switch on mirem les diferents possibilitats d'Items. Depenent de l'opció enviem a l'usuari a una pantalla o a un altre.

```
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item) {
    switch(item.getItemId()){
        case R.id.it_home: break;
        case R.id.it_profile:
            intent = new Intent(this, ProfileActivity.class);
            startActivity(intent);
            break;
        case R.id.it_ranking:
            intent = new Intent(this, RankingActivity.class);
            startActivity(intent);
            break;
    }
    return true;
}
```

prepareUserInformation

Aquest mètode modifica el correu a l'objecte usuari i crida d'un mètode més per recopilar les activitats de l'usuari.

```
public void prepareUserInformation(String userEmail){
    user.setEmail(userEmail);
    findUserActivities();
}
```

findUserActivities

En aquest mètode recollim totes les activitats de l'usuari i ho emmagatzemem en l'objecte usuari, a més mirem si l'usuari té alguna activitat pendent de realitzar. Finalment es crida el mètode showScreen. Si la petició té un resultat negatiu, mostrarem per pantalla el missatge d'error.

```
private void findUserActivities() {
    db.collection("Activity")
      .whereEqualTo("userEmail", user.getEmail())
      .get()
      .addOnCompleteListener(
                        new OnCompleteListener<QuerySnapshot>(){
       @Override
       public void onComplete(@NonNull Task<QuerySnapshot> task){
           if(task.isSuccessful()){
               String key = "";
               ArrayList<Activity> aux = new ArrayList<>();
               for(QueryDocumentSnapshot document:
                                                task.getResult()){
                  aux.add(new Activity(document.getId(),
                       document.getData().get("key").toString(),
                       (Boolean) document.getData().get("done")));
                  if(!(Boolean) document.getData().get("done")){
                       user.setActive(true);
                       key = document.getData()
                                              .get("key").toString();
                  }
               }
               user.setActivities(aux);
               showScreen(key);
           }
           else
               Log.v("ERROR-FindUserActivities",
                              task.getException().getMessage());
       }
   });
}
```

showScreen

En aquest mètode escollim quina pantalla mostrarem depenent de si l'usuari té o no alguna activitat pendent. Podem veure que depenent de l'opció es crida a un mètode diferent.

```
private void showScreen(String key){
    if(user.getActive())
        userHaveActivityScreen(key);
    else
        chooseActivityScreen();
}
```

chooseActivityScreen

En el cas que l'usuari encara no tingui cap activitat pendent, es mostrarà per pantalla la possibilitat d'acceptar una activitat. En aquest cas el que fem és amagat el TextView, canviar els noms dels botons i es crida al mètode newActivity.

```
private void chooseActivityScreen(){
    tvInfo.setVisibility(View.INVISIBLE);
    btnOne.setText(R.string.btn_accept);
    btnTwo.setText(R.string.btn_new);
    newActivity();
}
```

newActivity

Aquest mètode fa una crida a l'API per recuperar una activitat aleatòria i aquesta la mostrem per pantalla.

```
public void newActivity(){
    JsonObjectRequest jsonObjectRequest =
       new JsonObjectRequest(Request.Method.GET, urlRandom, null,
          new Response.Listener<JSONObject>() {
             @Override
             public void onResponse(JSONObject response) {
                try {
                  tvActivity.setText(response
                                          .getString("activity"));
                   activityKey = response.getString("key");
                }catch (JSONException e) {
                    Log.v("ERROR-API", e.getMessage());}
             }
            }, new Response.ErrorListener() {
              @Override
              public void onErrorResponse(VolleyError error) {
                 Log.v("ERROR-API", error.getMessage());
              }
    });
    queue.add(jsonObjectRequest);
}
```

userHaveActivityScreen

En el cas que l'usuari tingui una activitat que encara no ha finalitzat, mostrem el TextView, canviem els noms dels botons i, en últim lloc, cridem al mètode getActivityByKey per recuperar el text de l'activitat.

```
private void userHaveActivityScreen(String key) {
    tvInfo.setVisibility(View.VISIBLE);
    btnOne.setText(R.string.btn_done);
    btnTwo.setText(R.string.btn_cancel);
    getActivityByKey(key);
}
```

getActivityByKey

Aquest mètode fa una petició a l'API per recuperar una activitat a partir de la Key d'aquesta.

```
public void getActivityByKey(String key){
    String url = "https://www.boredapi.com/api/activity?key="+key;
    JsonObjectRequest jsonObjectRequest =
          new JsonObjectRequest(Request.Method.GET, url, null,
             new Response.Listener<JSONObject>() {
               @Override
               public void onResponse(JSONObject response) {
                  try {
                     tvActivity.setText(response
                                        .getString("activity"));
                   }catch (JSONException e) {
                      Log.v("ERROR-API", e.getMessage());}
               }
             }, new Response.ErrorListener() {
               @Override
               public void onErrorResponse(VolleyError error) {
                  Log.v("ERROR-API", error.getMessage());
               }
     });
     queue.add(jsonObjectRequest);
}
```

updateData

Aquest mètode modifica una activitat de la base de dades. Per paràmetre ens arriba l'Id del document de la base de dades, i el que fem és modificar l'atribut done de l'activitat.

```
private void updateData(String documentId) {
    db.collection("Activity").document(documentId)
                                        .update("done", true);
}
```

clickedButtonOne

Aquest mètode captura el clic sobre el botó 1 i realitza una cosa depenent de si l'usuari te o no una activitat pendent. Si l'usuari té una activitat pendent, el que fa és modificar l'objecte usuari perquè ja no tingui l'activitat com a pendent, es crida al mètode updateData i finalment modifica la pantalla perquè l'usuari pugui escollir un altra activitat. En cas contrari s'accepta l'activitat.

```
public void clickedButtonOne(View view) {
    if(user.getActive()){
        for(Activity a : user.getActivities()){
            if(!a.getDone()){
                updateData(a.getId());
                a.setDone(true);
                user.setActive(false);
                chooseActivityScreen();
                return;
            }
        }
    }
    else{
        userAcceptActivity();
        user.setActive(true);
        userHaveActivityScreen(activityKey);
    }
}
```

clickedButtonTwo

Aquest mètode captura el clic sobre el botó 2 i realitza una cosa depenent de si l'usuari te o no una activitat pendent. Si l'usuari té una activitat pendent, el que fa és eliminar l'activitat. En cas contrari es crida el mètode per demanar una nova activitat.

```
public void clickedButtonTwo(View view) {
    if(user.getActive()){
        for(Activity a : user.getActivities()){
            if(!a.getDone()){
                removeActivity(a.getId());
                user.deleteActivity(a);
                user.setActive(false);
                chooseActivityScreen();
            }
        }
    }
    else newActivity();
}
```

userAcceptActivity

Aquest mètode afegeix un nou registre a la col·lecció Activity de la base de dades i també l'afegeix a l'objecte usuari. En el cas que la petició falli, aleshores imprimirem per un missatge de l'error.

```
private void userAcceptActivity(){
    Map<String, Object> activity = new HashMap<>();
    activity.put("userEmail", user.getEmail());
    activity.put("key", activityKey);
    activity.put("done", false);

    db.collection("Activity").add(activity)
        .addOnSuccessListener(
           new OnSuccessListener<DocumentReference>() {
             @Override
             public void onSuccess(DocumentReference
                                              documentReference) {
                Toast.makeText(MainActivity.this,
                        getString(R.string.toast_new_activity_ok),
                                    Toast.LENGTH_SHORT).show();
                Activity a = new Activity(
                  documentReference.getId(), activityKey, false);
                user.addActivity(a);
             }
           }).addOnFailureListener(new OnFailureListener(){
              @Override
              public void onFailure(@NonNull Exception e){
                 Log.v("ERROR-UserAcceptActivity",
                                              e.getMessage());
              }
           });
}
```

removeActivity

Aquest mètode fa una petició a la base de dades per eliminar un registre d'activitat a partir de l'id del document passat per pantalla. En el cas que la petició no es pugui realitzar, mostrem l'error.

```
private void removeActivity(String documentId){
    db.collection("Activity").document(documentId).delete()
        .addOnSuccessListener(new OnSuccessListener<Void>() {
           @Override
           public void onSuccess(Void unused) {
               Toast.makeText(MainActivity.this,
                        getString(R.string.toast_delete_ok),
                                 Toast.LENGTH_SHORT).show();
           }
           }) .addOnFailureListener(new OnFailureListener() {
               @Override
               public void onFailure(@NonNull Exception e) {
                   Log.v("ERROR-RemoveActivity", e.getMessage());
               }
           });
}
```

RankingActivity

Introducció

Aquesta pantalla mostra el Ranking dels usuaris ordenats per nombre d'activitats completades. En el llistat es mostra el correu de l'usuari i el nombre d'activitats que ha completat. A més, conté un menú inferior on permet accedir a la pantalla de l'Home i del Perfil. A continuació mostrarem els elements de la pantalla i el codi que creiem més rellevant.

Layout

La pantalla es compon de tres TextView, un RecyclerView i un BottomNavigationView.

Codi

onCreate

Aquest no és el codi complet del mètode onCreate, aquí només mostro el més important on podrem veure l'assignació del BottomNavigationView i la crida del mètode getData.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_ranking);
    setTitle(R.string.title_ranking);
    bottomNavigationView = findViewById(R.id.bottomNavTab2);
    bottomNavigationView.setOnNavigationItemSelectedListener(this);
    bottomNavigationView.setSelectedItemId(R.id.it_ranking);
    db = FirebaseFirestore.getInstance();
    getData();
}
```

onNavigationItemSelected

Aquest mètode captura el clic de BottomNavigationView. Podem veure que té un Switch on mirem les diferents possibilitats d'Items. Depenent de l'opció enviem a l'usuari a una pantalla o a un altre.

```
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item) {
    switch(item.getItemId()){
        case R.id.it_home:
            intent = new Intent(this, MainActivity.class);
            startActivity(intent);
            break;
        case R.id.it_profile:
            intent = new Intent(this, ProfileActivity.class);
            startActivity(intent);
            break;
        case R.id.it_ranking: break;
    }
    return true;
}
```

getData

Aquest mètode el que fa és recollir tots els registres d'Activitats de la base de dades i els emmagatzema en un llistat anomenat users. Un cop s'han recorregut tots els registres ordenem el llistat i enviem les dades a l'adaptar.

```
public void getData(){
    db.collection("Activity").get()
        .addOnCompleteListener(
         new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(Task<QuerySnapshot> task){
              if(task.isSuccessful()){
                 for (QueryDocumentSnapshot document :
                                          task.getResult()){
                    String email = document.getData()
                                .get("userEmail").toString();
                   if((Boolean)document.getData().get("done")){
                       int index = userPosition(email);
                       if(index != -1)
                          users.get(index)
                                    .addActivity(new Activity());
                       else{
                          User u = new User();
                          u.setEmail(email);
                          u.addActivity(new Activity());
                          users.add(u);
                       }
                    }
                  }
                  Collections.sort(users);
                  userAdapter.notifyDataSetChanged();
                }
                else
                  Log.v("ERROR-GetData:",
                            task.getException().getMessage());
              }
            });
}
```

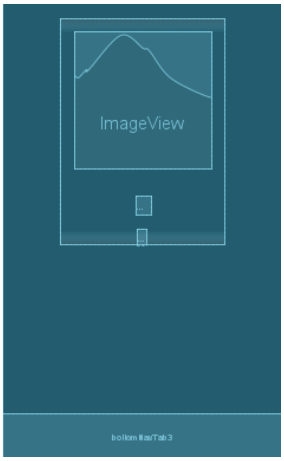
ProfileActivity

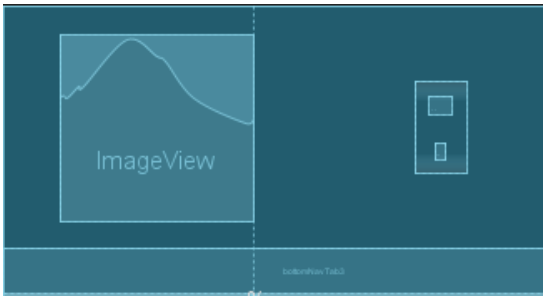
Introducció

Aquesta pantalla mostrarà una imatge diferent depenent del nombre d'activitats de l'usuari, també mostrarà el nom d'usuari i li indicarà quantes activitats ha completat. Des d'aquesta pantalla es pot accedir a la pantalla de configuració i a més, conté un menú inferior on permet accedir a la pantalla de l'Home i del Ranking. A continuació mostrarem els elements de la pantalla i el codi que creiem més rellevant.

Layout

Aquesta pantalla té dues distribucions diferents depenent de l'orientació de la pantalla. Es compon d'un ImageView, dos TextView i un BottomNavigationView.





Codi

onCreate

Aquest no és el codi complet del mètode onCreate, aquí només mostro el més important on podrem veure l'assignació del BottomNavigationView, l'ús del SharedPreference, la crida dels altres mètodes i també captura el retorn de l'intent a Configuració.

Podem veure que recuperem del SharedPreferences el correu de l'usuari i seguidament fem la crida als mètodes findUserActivities i finsUser.

L'activitat configuració ens pot retornar resultat després d'enviar l'intent per tal de modificar el valor del nom d'usuari. Es pot veure que si tot és correcte es crida al mètode updateUserUsername.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_profile);
    setTitle(R.string.title_profile);
    sharedPreferences = getSharedPreferences("Preferences",
                                        Context.MODE_PRIVATE);
    bottomNavigationView = findViewById(R.id.bottomNavTab3);
    bottomNavigationView.setOnNavigationItemSelectedListener(this);
    bottomNavigationView.setSelectedItemId(R.id.it_profile);

    email = sharedPreferences.getString("email", "");
    findUserActivities();
    findUser();

    activityDataResult = registerForActivityResult(
       new ActivityResultContracts.StartActivityForResult(),
       new ActivityResultCallback<ActivityResult>() {
          @Override
          public void onActivityResult(ActivityResult result){
             if(result.getResultCode() == RESULT_OK){
                if(!result.getData().getStringExtra("newUsername")
                                                    .equals(username))
                   updateUserUsername(result.getData()
                                .getStringExtra("newUsername"));
                }
             }
          });
}
```

onNavigationItemSelected

Aquest mètode captura el clic de BottomNavigationView. Podem veure que té un Switch on mirem les diferents possibilitats d'Items. Depenent de l'opció enviem a l'usuari a una pantalla o a un altre.

```
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item){
    switch(item.getItemId()){
        case R.id.it_home:
            intent = new Intent(this, MainActivity.class);
            startActivity(intent);
            break;
        case R.id.it_profile:  break;
        case R.id.it_ranking:
            intent = new Intent(this, RankingActivity.class);
            startActivity(intent);
            break;
    }
    return true;
}
```

onOptionsItemsSelected

Aquest mètode recull el clic de l'Item del menú superior i el que fem és llençar un intent enviant-li dades i esperant resposta d'aquest.

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()){
        case R.id.itConfiguration:
            Intent intent = new Intent(this,
                        ConfigurationActivity.class);
            intent.putExtra("username", username);
            intent.putExtra("email", email);
            activityDataResult.launch(intent);
            break;
    }
    return super.onOptionsItemSelected(item);
}
```

updateUserUsername

Aquest mètode actualitza de la base de dades el nom d'usuari.

```
private void updateUserUsername(String newUsername) {
    username = newUsername;
    tvUsername.setText(username);
    db.collection("User").document(userId)
        .update("username", newUsername);

    Toast.makeText(this, getString(R.string.toast_update_username),
                                        Toast.LENGTH_SHORT).show();
}
```

findUserActivities

Aquesta activitat recull les activitats de l'usuari per saber el nombre d'activitats que ha completat.

```
private void findUserActivities() {
   db.collection("Activity")
      .whereEqualTo("userEmail", email)
      .get()
      .addOnCompleteListener(
         new OnCompleteListener<QuerySnapshot>() {
          @Override
          public void onComplete(@NonNull Task<QuerySnapshot> task){
            if(task.isSuccessful()){
              for(QueryDocumentSnapshot document: task.getResult()){
                 if((Boolean) document.getData().get("done"))
                    numActivities ++;
              }
               tvNum.setText(getString(R.string.tv_num_activities_1)
                     +" "+numActivities+" "+
                           getString(R.string.tv_num_activities_2));
               putImage();
                     }
             else
                Log.v("ERROR-FindUserActivities",
                              task.getException().getMessage());
            }
          });
}
```

putImage

Aquest mètode fa la crida al mètode chooseImage i mostra la imatge per pantalla.

```
private void putImage() {
    int img = chooseImage();
    String url = "gs://practica4mariaperegrina.appspot.com
                                              /imgProfile"+img+".jpg";
    gsReference = storage.getReferenceFromUrl(url);
    Glide.with(this).load(gsReference).into(image);
}
```

chooseImage

Aquest mètode retorna el número de la imatge que s'ha de mostrar depenent del nombre d'activitats de l'usuari.

```
public int chooseImage(){
     int num;
     if(numActivities <= 5) num = 7;
     else if(numActivities > 5 && numActivities <= 10) num=6;
     else if(numActivities>10 && numActivities <= 15) num=5;
     else if(numActivities>15 && numActivities<=20) num=4;
     else if(numActivities>20 && numActivities<=30) num=3;
     else if(numActivities>30 && numActivities<=40) num=2;
     else num=1;
     return num;
}
```

findUser

Aquest mètode recull el nom d'usuari per mostrar-lo per pantalla.

```
private void findUser() {
   db.collection("User")
      .whereEqualTo("email", email)
      .get()
      .addOnCompleteListener(
        new OnCompleteListener<QuerySnapshot>() {
          @Override
          public void onComplete(Task<QuerySnapshot> task){
            if(task.isSuccessful()) {
              if (task.getResult().size() == 1)
                 for (QueryDocumentSnapshot document :
                                          task.getResult()){
                    username = document.getData()
                                       .get("username").toString();
                    tvUsername.setText(username);
                    userId = document.getId();
                 }
             }
             else
               Log.v("ERROR-FindUser",
                          task.getException().getMessage());
            }
         });
}
```

ConfigurationActivity

Introducció

Aquesta activitat conte dos Fragments per la seva visualització. Per tant, aquesta activitat només conté mètodes de control per saber quan hem de visualitzar cada fragment. A continuació mostrarem els elements de la pantalla i el codi que creiem més rellevant.

Layout

Aquesta pantalla té dues distribucions diferents depenent de l'orientació de la pantalla. El layout vertical conte un únic fragment, mentre que el fragment horitzontal conte dos fragments diferents.

Codi

Logout

Aquest mètode fa la crida al FirebaseAuth per tal de tancar la sessió de l'usuari, i per tant llença un intent al LoginActivity.

```
public void logout(){
    if(FirebaseAuth.getInstance().getCurrentUser()!=null){
        FirebaseAuth.getInstance().signOut();
        startActivity(new Intent(this, LoginActivity.class));
    }
}
```

showEditUsernameFragment

Aquest mètode mostra el fragment d'editar el nom d'usuari. Podem veure que depenent de l'orientació ho posarem en un o en un altre element.

```
public void showEditUsernameFragment(String username){
    editUsernameFragment =
        EditUsernameFragment.newInstance(username);
    FragmentTransaction ft = fm.beginTransaction();
    int orientation = getResources().getConfiguration().orientation;
    if(orientation == Configuration.ORIENTATION_LANDSCAPE)
        ft.replace(R.id.fgEditUsername, editUsernameFragment);
    else
        ft.replace(R.id.fgConfiguration, editUsernameFragment);
    ft.commit();
}
```

updateUsername

Aquest mètode retorna a ConfigurationActivity el nou no d'usuari.

```
public void updateUsername(String newUsername){
    Intent result = new Intent();
    result.putExtra("newUsername", newUsername);
    setResult(RESULT_OK, result);
    finish();
}
```

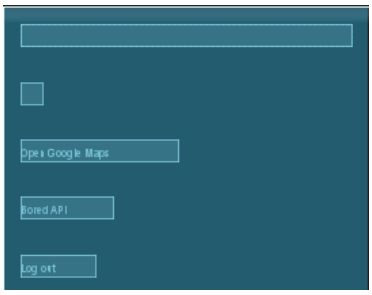
ConfigurationFragment

Introducció

Aquesta pantalla mostra el correu de l'usuari, el nom d'usuari (donant-li l'opció d'editar-lo), dona l'opció d'obrir el Google Maps, d'obrir la documentació de l'API i, en últim lloc, permet tancar la sessió. A continuació mostrarem els elements de la pantalla i el codi que creiem més rellevant.

Layout

Aquesta pantalla conte cinc TextView, un per cada informació que es mostra.



Codi

Captura clic Username

Aquest codi és una part del mètode onCreat on es captura el clic del TextView on es mostra el nom d'usuari. El que fem és cridar el mètode de ConfigurationActivity showEditUsernameFragment.

```
tvUsername.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
      ((ConfigurationActivity)getActivity())
                                .showEditUsernameFragment(username);
    }
});
```

Captura clic Google Maps

Aquest codi és una part del mètode onCreat on es captura el clic del TextView d'obrir el Google Maps. El que fem és llençar un intent al Google Maps.

```
tvMaps.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(Intent.ACTION_VIEW);
        intent.setPackage("com.google.android.apps.maps");
        startActivity(intent);
    }
});
```

Captura clic API

Aquest codi és una part del mètode onCreat on es captura el clic del TextView d'obrir la documentació de l'API. El que fem és llençar un intent a l'URL de la documentació.

```
tvApi.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(Intent.ACTION_VIEW);
        intent.setData(Uri
                .parse("https://www.boredapi.com/documentation"));
        startActivity(intent);
    }
});
```

Captura clic Logout

Aquest codi és una part del mètode onCreat on es captura el clic del TextView per tancar sessió. El que fem és cridar al mètode confirmLogout.

```
tvLogout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        confirmLogout();
    }
});
```

confirmLogout

Aquest mètode llença un AlertDialog per confirmar tancar la sessió. Si la resposta és afirmativa crida el mètode logout de ConfigurationActivity, en cas contrari només es tanca el diàleg.

```
private void confirmLogout() {
    AlertDialog.Builder alertDialogBuilder =
                        new AlertDialog.Builder(getContext());
    alertDialogBuilder.setMessage(getString(
                        R.string.dialog_question_cancel))
       .setCancelable(false)
       .setPositiveButton(R.string.dialog_yes,
        new DialogInterface.OnClickListener() {
         public void onClick(DialogInterface dialog, int id) {
            ((ConfigurationActivity)getActivity()).logout();
         }
       })
       .setNegativeButton(getString(R.string.dialog_no),
         new DialogInterface.OnClickListener() {
          public void onClick(DialogInterface dialog, int id) {
              dialog.cancel();
          }
         });
    AlertDialog alertDialog = alertDialogBuilder.create();
    alertDialog.show();
}
```

EditUsernameFragment

Introducció

Aquest fragment permet a l'usuari modificar el nom d'usuari.

Layout

Es compon de dos elements, un EditText i un botó per confirmar el canvi.

Codi

Captura del clic del botó de confirmar

Aquest codi és una part del mètode onCreat on es captura el clic del Botó per confirmar el nou nom d'usuari. El que es fa és validar el nou nom d'usuari i cridar el mètode updateUsername de la classe ConfigurationActivity.

```
btnConfirm.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String newUsername = etUsername.getText().toString();
        if(validateUsername(newUsername))
            ((ConfigurationActivity)getActivity())
                                .updateUsername(newUsername);
    }
});
```

Themes

Codi

Tema general

En el tema general he modificat el color de l'aplicació, la mida del Text i el color d'aquest.

```
<style name="Theme.MariaPeregrinaUsieto"
    parent="Theme.MaterialComponents.DayNight.DarkActionBar">

    <item name="colorPrimary">@color/light_blue</item>
    <item name="colorPrimaryVariant">@color/dark_blue</item>
    <item name="colorOnPrimary">@color/white</item>

    <item name="android:textSize">18sp</item>
    <item name="android:textColor">@color/black</item>
</style>
```

Tema ConfigurationActivity

En el tema per la ConfigurationActivity només modifiquem el color de fons i el color del text.

```
<style name="ConfigurationTheme"
                        parent="Theme.MariaPeregrinaUsieto">
    <item name="android:textColor">@color/white</item>
    <item name="android:windowBackground">@color/blue</item>
</style>
```

Tema per els títols

En el tema per als títols modifiquem la mida, el color, indiquem que volem la lletra en negreta i, en últim lloc, he modificat la font pels títols.

```
<style name="TitleTheme"  parent="android:Widget.TextView">
    <item name="android:textSize">25dp</item>
    <item name="android:textColor">@color/black</item>
    <item name="android:textStyle">bold</item>
    <item name="android:fontFamily">@font/paradox_mosaic</item>
</style>
```