Filmtastic

Filmtastic are ca principal scop crearea unei ”cinemateci” pentru persoanele care se folosesc de această aplicație. Pentru a putea crea această aplicație am utilizat ca platformă de creare Android Studio. De aici am utilizat diferite librării pentru a aduce la viață aplicația.

Ca bază de date am utilizat Firestore. Am creat diferite colecții cu cel puțin un document, iar aici se pot adăuga (deși se face din cod) noi documente care vor fi mai târziu prezente în bază. Tot de la Firebase am folosit si metoda de autentificare cu email și parolă. Putem crea un nou cont care va fi stocat în Firebase Authentication. Cu ajutorul email-ului și al parolei ne vom putea conecta apoi la aplicație.

După conectare vom intra pe pagina principală. În dreapta avem meniul care ne duce la diverse categorii de filme( nouă la număr, fiecare având câte un film prezentat și în cod un altul care așteaptă să fie introdus), actori, directori și la pagina de profil. De asemenea avem butonul de Log out care ne va trimite la pagina Log In.

Pentru a realiza fiecare pagină și pentru a reuși să adaug noi elemente m-am folosit de libraria Picasso( pentru a afișa pozele) și am lucrat în concordanță cu elemente oferite de Android Studio( pagini .xml și Activity în Java).

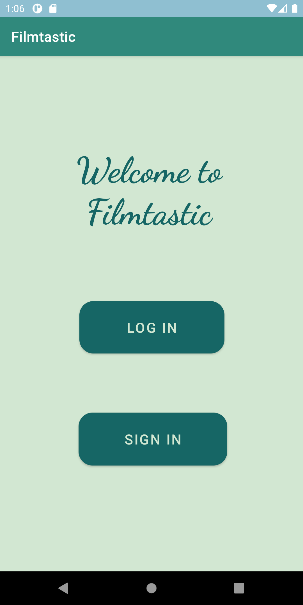
În ceea ce privește codul, pe lângă adăugarea de noi elemente în baza de date, am adăugat funcții care să afișeze elementele colecțiilor în formă de listă și să afișeze datele din Firestore.

Conține 2 butoane care ne trimit la paginile de Log In, respectiv Sign In.

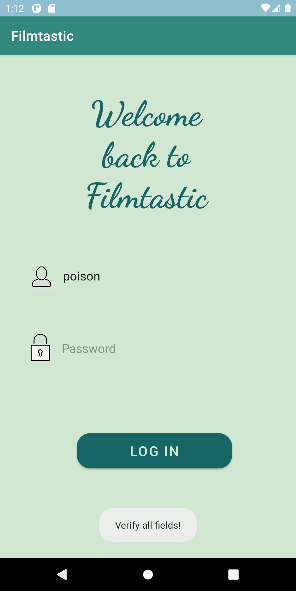
button1= (Button) findViewById(R.id.*button\_log*);  
 button1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 openLoginPage();  
 }  
 });  
 button2= (Button) findViewById(R.id.*button\_sign*);  
 button2.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 openSignInPage();  
 }  
 });  
  
 }  
  
 public void openLoginPage(){  
 Intent intent=new Intent(LandingPage.this,LoginPage.class);  
 startActivity(intent);  
 }  
  
public void openSignInPage(){  
 Intent intent=new Intent(LandingPage.this,SignInPage.class);  
 startActivity(intent);  
 }

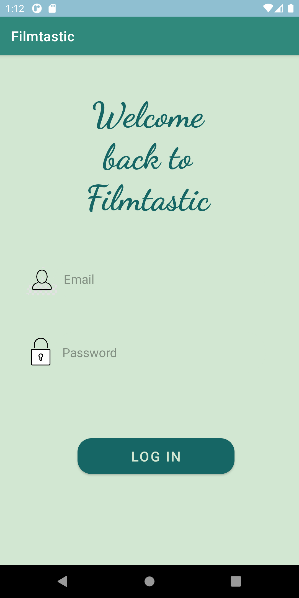
Elemente ale aplicației:

1. Pagina de Landing



1. Pagina de Log In



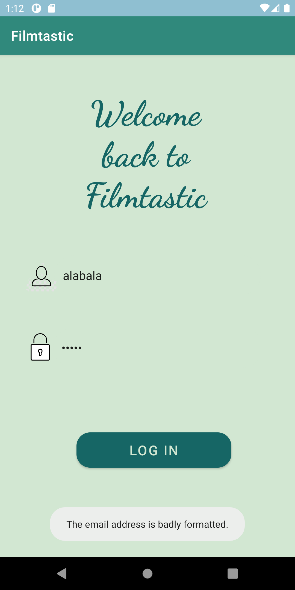


În pagina de Log In avem 2 câmpuri care necesită să fie completate și un buton care ”confirmă” conectarea. Avem 3 cazuri care nu duc la conectare:

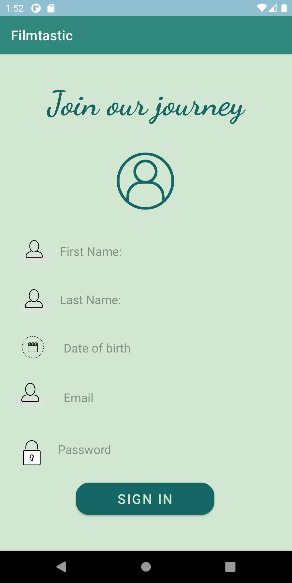
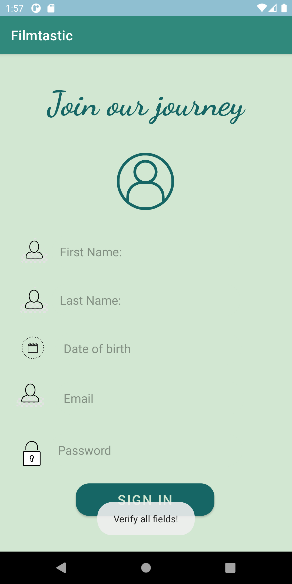
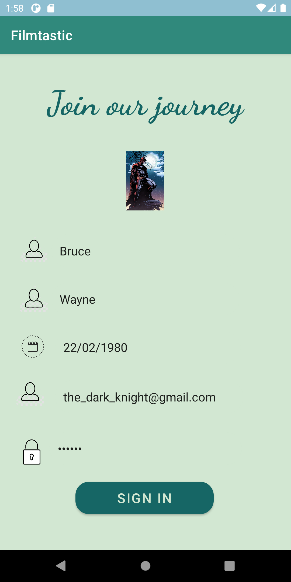
1. Când nu completăm unul dintre câmpuri. Vom primi un mesaj („Verify all fields!”) și nu ne vom putea conecta.
2. Când, deși completăm câmpurile, am introdus o adresă de email greșită. Vom primi mesajul „The email address is badly formatted.” și va trebui să o reintroducem.
3. Când avem email-ul corect, dar parola greșită ni se va afișa „The password is invalid or the user does not have a password”.

Când introducem datele corecte ni se va permite conectarea și vom fi trimiși pe MainPage.

final String mail = user.getText().toString();  
final String pass = password.getText().toString();  
  
if(mail.isEmpty()||pass.isEmpty()){  
 showMessage("Verify all fields!");  
 button.setVisibility(View.*VISIBLE*);  
 loginProgress.setVisibility(View.*INVISIBLE*);  
}  
else{  
 signIn(mail,pass);  
}

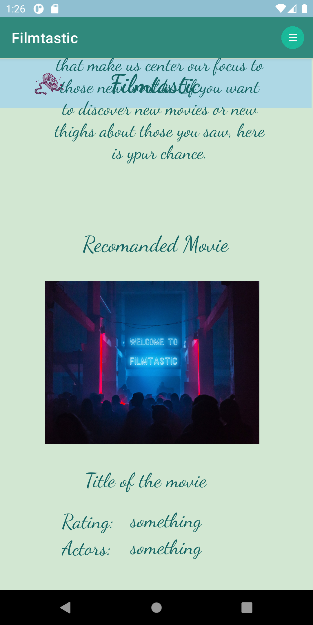
1. Pagina de Sign In

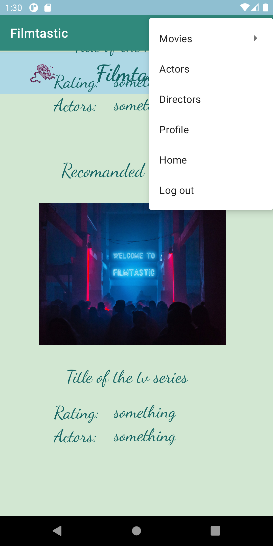
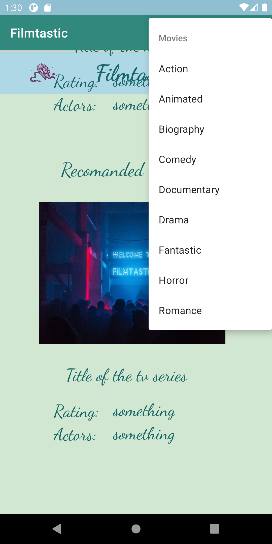
La fel ca la Log In, este necesar, pentru a crea un cont, să completăm toate câmpurile.

Contul pe care l-am creat va fi salvat în Firebase Authentication.

1. Pagina principală

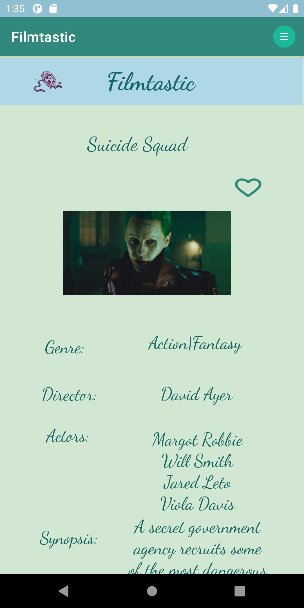
Pe MainPage avem o mini-introducere în ceea ce privește ideea aplicație. Intenția mea era să recomand random un film și un serial(la care am renunțat mai târziu) din Firestore. Ideea există, implementarea mai puțin.

Fiind prima pagină de după logare, aici avem o prezentare a meniului de tip hamburger. Item-ul „Movies” deschide la rândul său o nouă serie de item-uri. De aici ne putem „deplasa” în voie în aplicație.

1. Filme de acțiune

În toate paginile în care avem filme prezentate, avem același model de reprezentare. Avem aceleași câmpuri doar că sunt completare cu filme diferite.

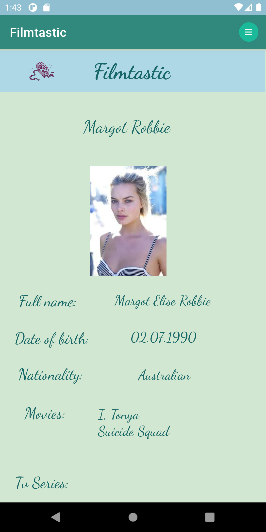
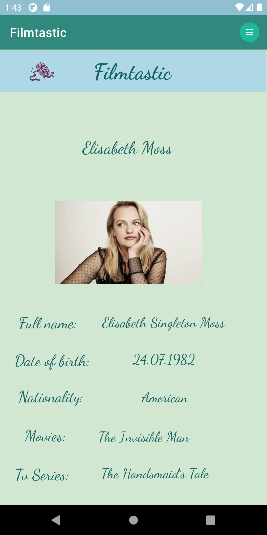
 

În categoria filme (atât aici, cât și în celelalte opt pagini) oferim anumite informații. Ni se oferă detalii despre titlu, genurile filmului, diverși actori ce fac parte din distribuție, numele directorului, o mică descriere a plotului, rating-ul, data apariției, durata filmului și o imagine reprezentativă.

title.setText(titl.toString());  
director.setText(full.toString());  
release.setText(data.toString());  
runtime.setText(nat.toString());  
genre.setText(mov.toString());  
actors.setText(tv.toString());  
rating.setText(rat.toString());  
synopsys.setText(sino.toString());  
  
Picasso.*get*().load(String.*valueOf*(img)).into(imageView);

1. Filme animate
2. Filme biografice
3. Filme de comedie
4. Documentare
5. Filme dramatice
6. Filme fantasy
7. Filme Horror
8. Filme romantice
9. Actori

La fel ca în paginile de filme, prezentăm anumite informații despre actori.

În baza de date găsim diverse informații. Avem posibilitatea de a găsi filme și seriale în care au jucat actorii, data lor de naștere, numele de scena și cel real, dar și naționalitatea. Acestea sunt însoțite de către o poză a acestora.

1. Directori

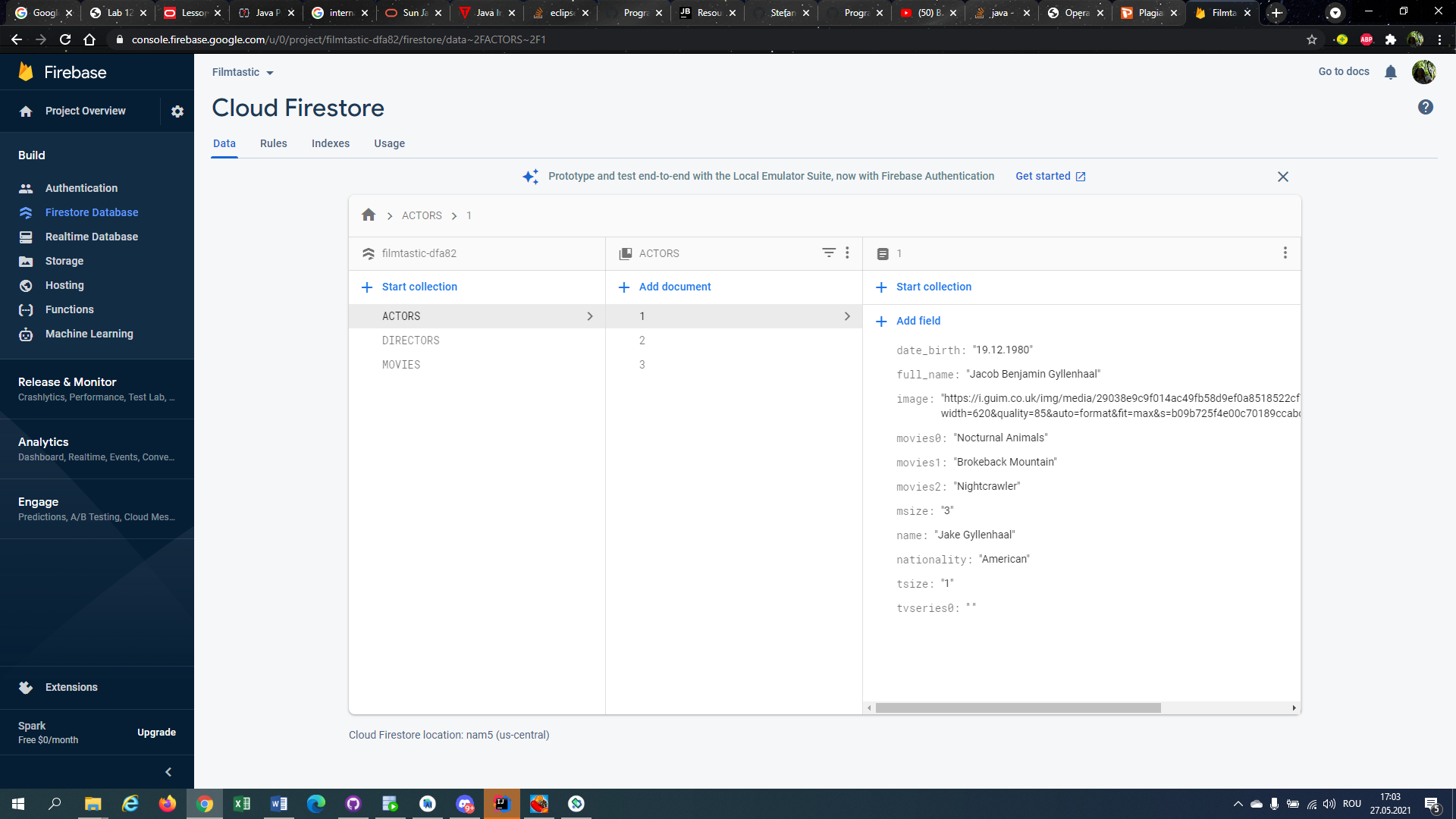
La fel ca în rândul actorilor, vom găsi aceleași informații, singura diferență fiind faptul că sunt prezentate filme/seriale regizate de directori.

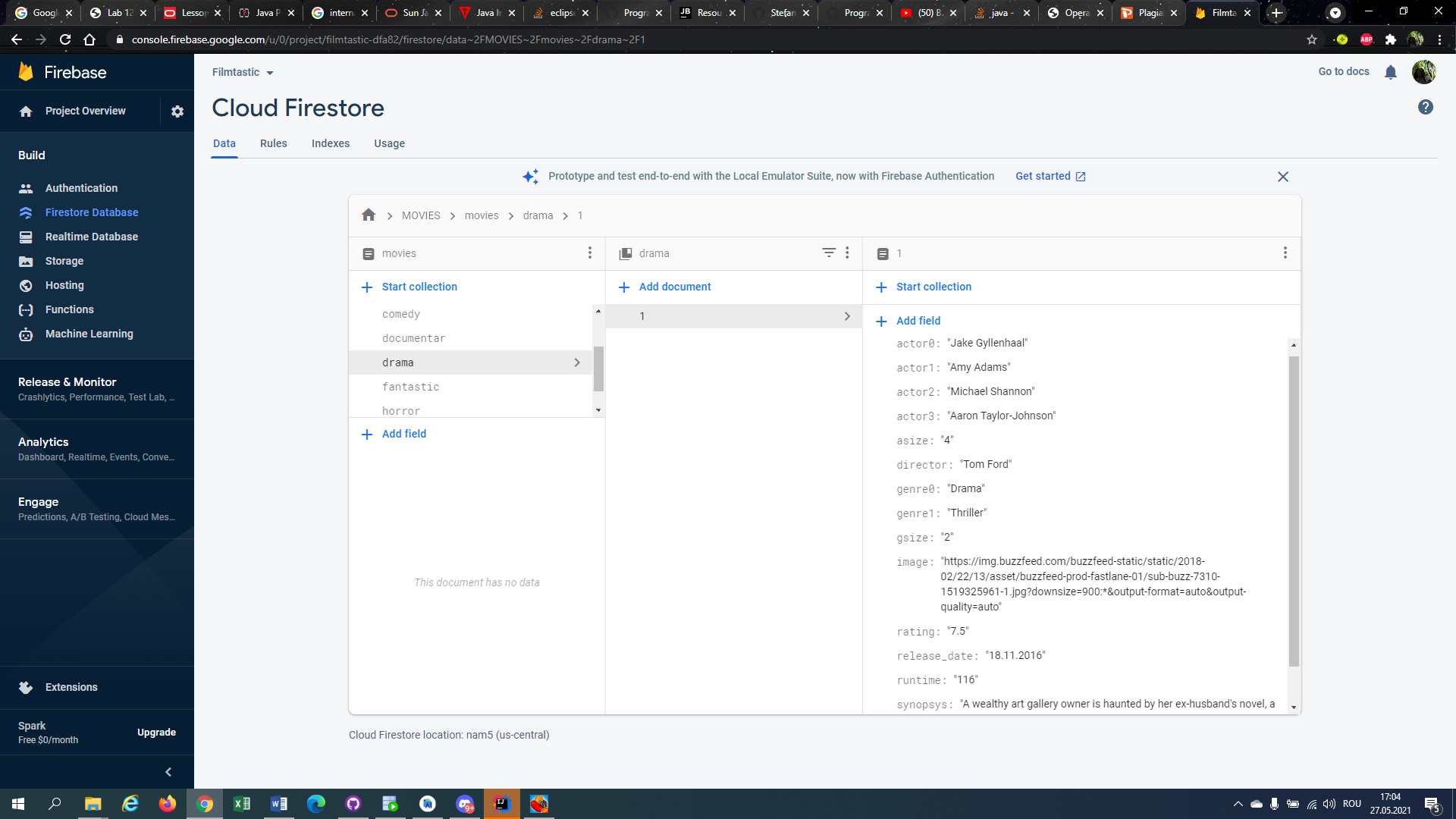
1. Profil

În ceea ce privește profilul, aici ar fi trebuit să existe datele de conectare ale utilizatorului. Poza cu care se loghează, numele, prenumele, email-ul și data nașterii. Pe lângă asta, aici avem implementat un buton care ar fi trebuit să ne prezinte lista de favorite. Din nou, acesta e un concept ce așteaptă să fie îndeplinit.

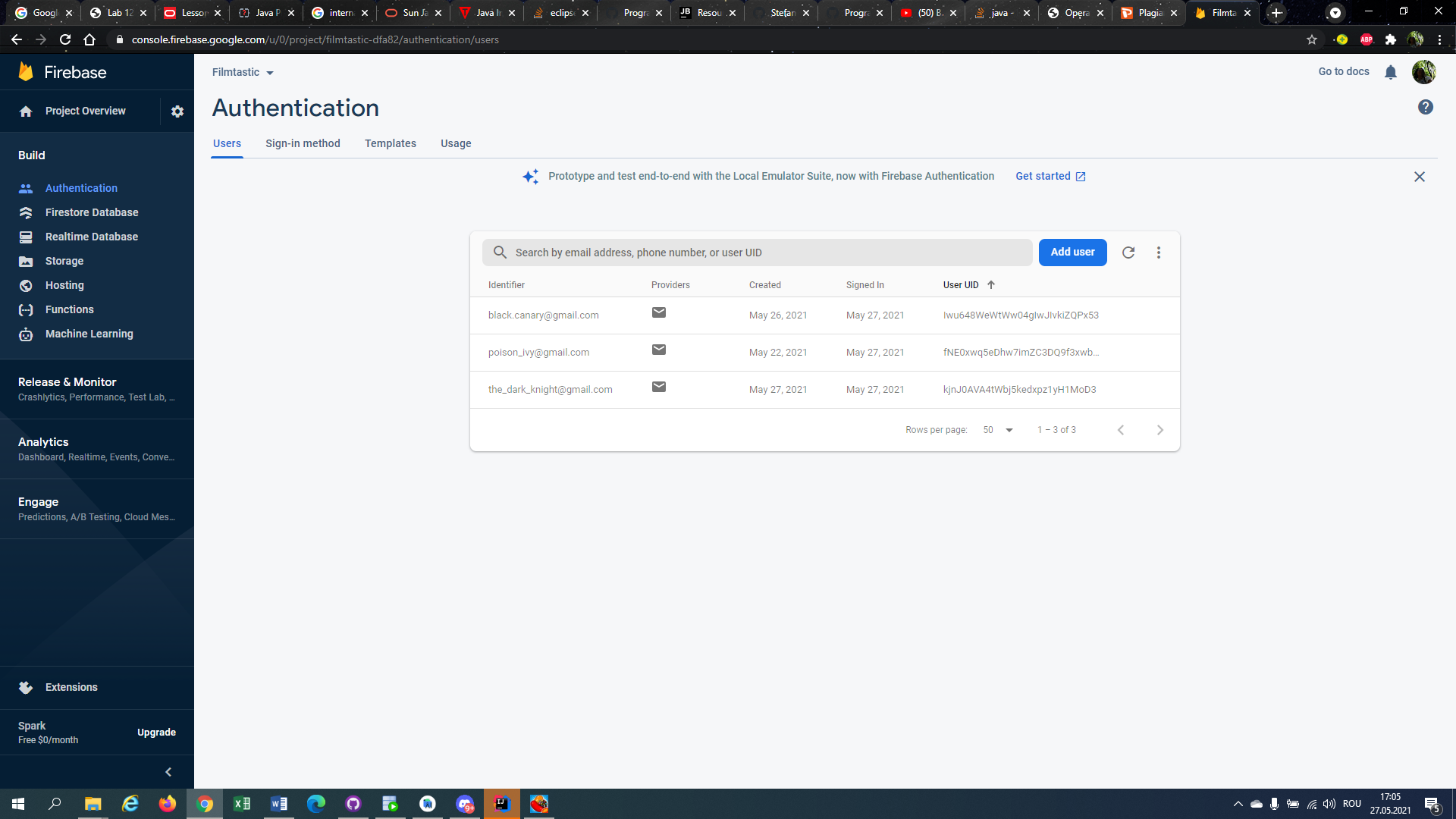
Elementele din spatele aplicației:

1. Firestore





1. Firebase Authentication



Exemple de funcții:

1. Adăugare de noi documente în Firestore

Map<String,Object> action= new HashMap<>();  
action.put("title","I Origins");  
action.put("director","Mark Cahill");  
action.put("genre0","Drama");  
action.put("genre1","Mystery");  
action.put("genre2","Romance");  
action.put("actor0","Michael Pitt");  
action.put("actor1","Steven Yeun");  
action.put("actor2","Brit Marling");  
action.put("gsize","3");  
action.put("asize","3");  
action.put("image","https://ro.pinterest.com/pin/524880531553918582/");  
action.put("runtime","106 min");  
action.put("rating","7.4");  
action.put("release\_date","19.09.2014");  
action.put("synopsys","A molecular biologist and his laboratory partner uncover evidence that may fundamentally change society as we know it.");  
  
db.collection("/MOVIES/movies/drama").document("2").set(action).  
 addOnSuccessListener(new OnSuccessListener<Void>() {  
 @Override  
 public void onSuccess(Void aVoid) {  
 Toast.*makeText*(MoviesDrama.this,"Add new Drama Movie", Toast.*LENGTH\_LONG*).show();  
 }  
 }).addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
 Log.*d*("ERROR",e.getMessage());  
 }  
});

1. Afișare de listă.

List<String> list = new ArrayList<>();  
db.collection("DIRECTORS").get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {  
 @Override  
 public void onComplete(@NonNull Task<QuerySnapshot> task) {  
 if (task.isSuccessful()) {  
  
 for (QueryDocumentSnapshot document : task.getResult()) {  
 list.add(document.getId());  
 }  
 Log.*d*("TAG", list.toString());  
 System.*out*.println(list);  
 } else {  
 Log.*d*("TAG", "Error getting documents: ", task.getException());  
 }  
 }  
});

1. Afișare a tuturor elementelor colecției în consolă.

private void showData() {  
 db.collection("ACTORS").get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {  
 @Override  
 public void onComplete(@NonNull Task<QuerySnapshot> task) {  
 if (task.isSuccessful()) {  
  
 for (QueryDocumentSnapshot document : task.getResult()) {  
 Log.*d*("TAG", document.getId()+"=>"+document.getData());  
 }  
  
 } else {  
 Log.*d*("TAG", "Error getting documents: ", task.getException());  
 }  
 }  
 });  
}