**CREAREA UNEI APLICATII C.R.U.D. CU BAZA DE DATE LOCALA IN FLUTTER**

**Elemente de M-Business**

**Ciont Cristian**

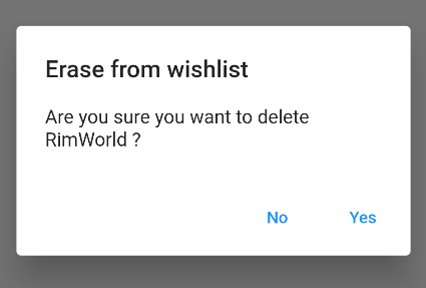
# Prezentare

Aplicatia permite utilizatorilor sa isi gestioneze jocurile video preferate care vor sa le cumpere in viitor, sub forma unui Wishlist. Acestia pot sa adauge titluri noi, sa vizualizeze lista jocurile deja existente, sa editeze si sa stearga orice element din lista precedenta. Toate informatiile despre entitate sunt stocate intr-o baza de date locale SQFLite pentru a se pastra cand aplicatia este redeschisa.

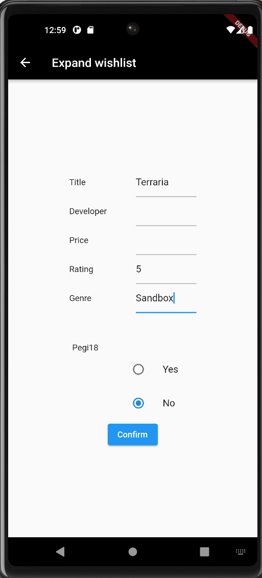
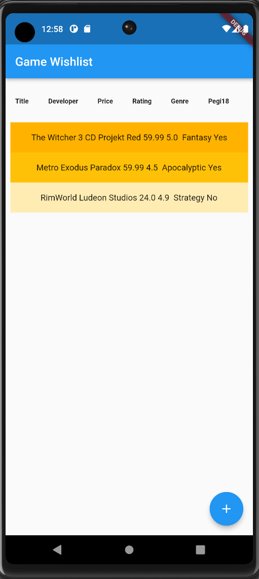
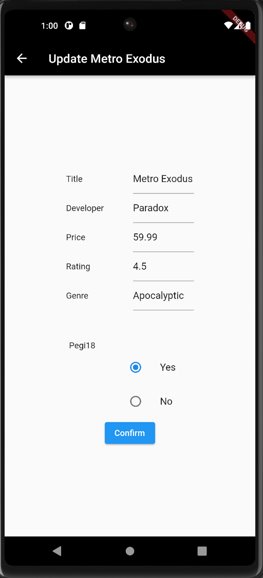
Flutter este un framework open source de la Google care permite crearea de aplicatii Android, compilate nativ, multi-platforma (web, mobile, embedded). Este un framework rapid, care permite recompilarea aplicatiilor in timp ce ruleaza cu ajutorul functiei Hot Reload.

Aplicatia este dezvoltata in Android Studio, folosind limajul Flutter si este formata din 3 ecrane principale si un Pop-up:

1. Ecranul principal este folosit pentru majoritatea functionalitatilor. Contine lista cu toate entitatile din baza de date, din care se poate ajunge printr-un click scurt la ecranul de editare a unei entitati, iar printr-un click lung la afisarea unui Pop-up care serveste ca si confirmare pentru stergerea entitatii selectate. De asemenea, in partea de jos a ecranului este prezent si un buton pentru adaugarea unei noi entitati;
2. Ecranul pentru adaugarea unui joc, alaturi de diferite inputs care trebuie introduse;
3. Ecranul pentru editarea unui joc, alaturi de caracteristicile care pot fi schimbate;

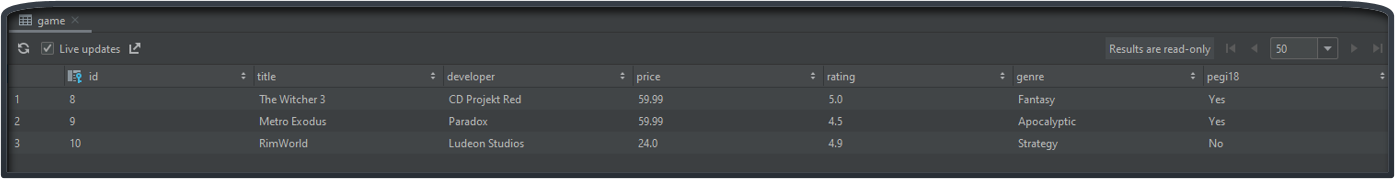


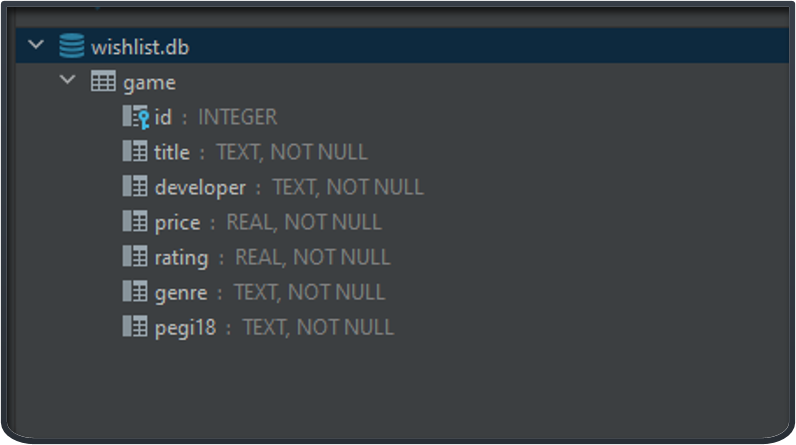
Imaginea 1: Pop-up pentru confirmare stergere element



Imaginile 2,3,4 : Ecranele prezente in aplicatie

Pentru persistarea datelor, a fost creat un tabel pentru a ingloba toate caracteristicile care trebuie salvate despre un joc. Meniul Database Inspector din cadrul Android Studio arata daca tabelele au fost create corect si ce date sunt prezente momentan.

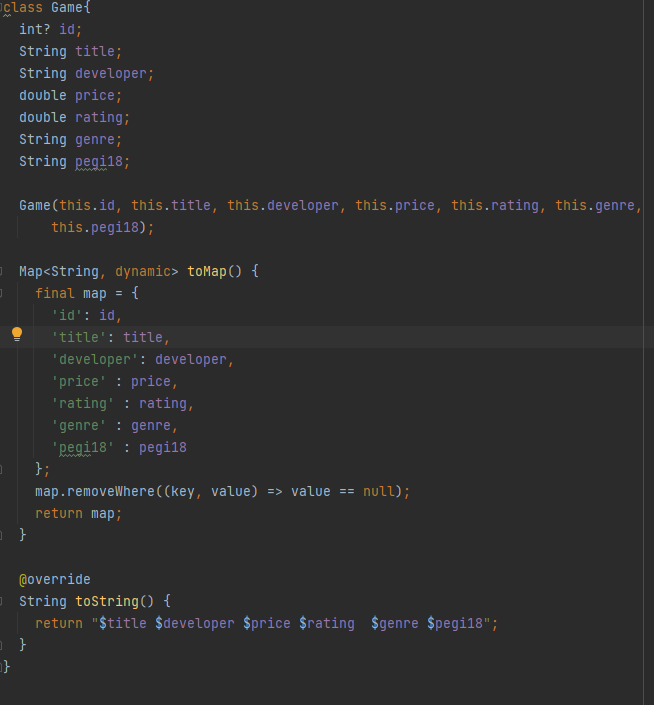
Imaginea 5: Intrarile din tabelul Game din baza de date Wishlist



Imaginea 6: Caracteristicile tabelului Game

# Structura & Implementare

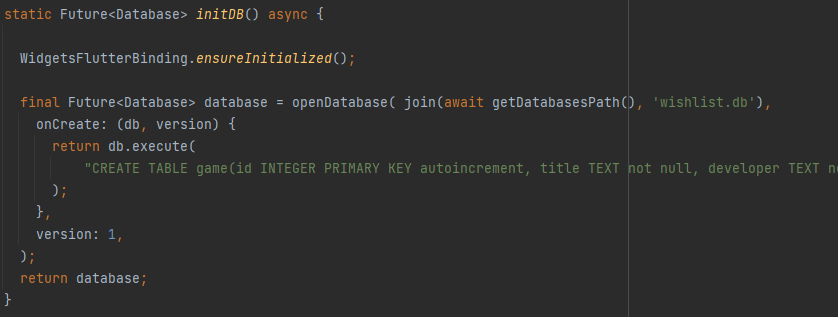
* Game.dart : Clasa model pentru entitatea Game



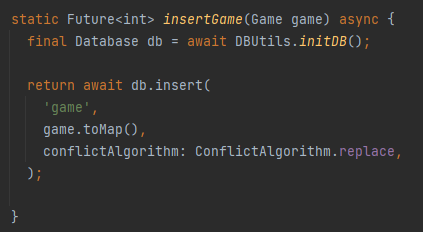
In clasa Game au fost trecute toate caracteristicile unui joc (id: Integer, title: String, developer: String, price: Double, rating: Double, genre: String, pegi18: String). De asemenea avem o metoda toMap() care ne ajuta sa transformam valorile variabilelor din fiecare instanta de obiect intr-un Hashmap, pentru a putea fi scrise cu succes in baza de date. Metoda toString() este folosita pentru a vizualiza obiectele in lista din ecranul principal.

* DBUtils: Clasa care se ocupa cu relationarea cu baza date

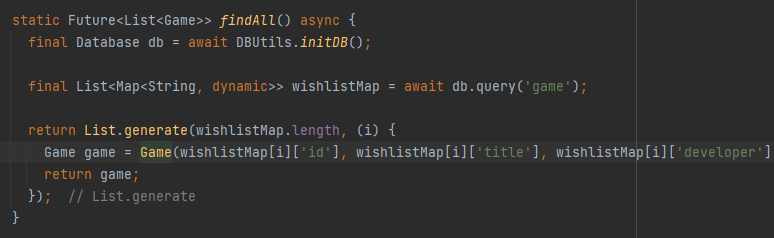
(Metode pentru crearea tabelelor, add, update, delete, find all)



Metoda initDB() creaza o baza de date noua cu denumirea ‘wishlist.db’, daca aceasta nu exista deja, intr-o locatie aleasa de calculator, si o marcheaza ca fiind versiunea 1. Ulterior daca dorim sa modificam sau sa adaugam alte tabele, va trebui sa schimbam versiunea pentru ca metoda sa se execute iarasi.

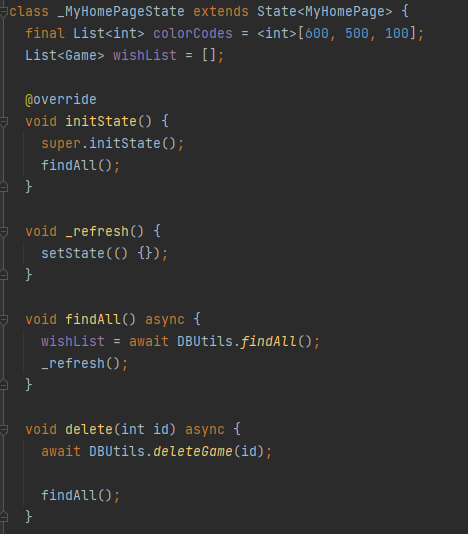


Metoda insertGame adauga obiectul de tip Game primit ca parametru in tabelul cu numele ‘game’. De asemenea este specificat un algoritm de conflict, pentru cazul in care se adauga un obiect cu un id explicit care exista deja in baza de date. Acesta va suprascrie valorile asociate anterior cu acel id.

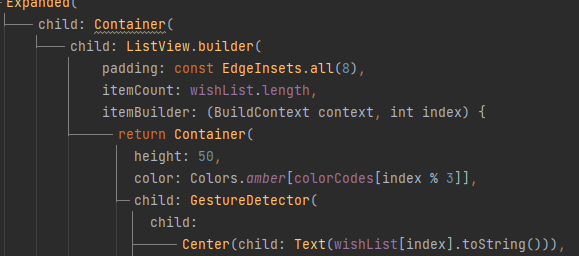


Metoda findAll() returneaza toate obiectele din tabela ‘game’. Acestea vin in modul default intr-un Hashmap, iar noi trebuie sa le convertim intr-o lista obisnuita cu obiecte de tip Game.

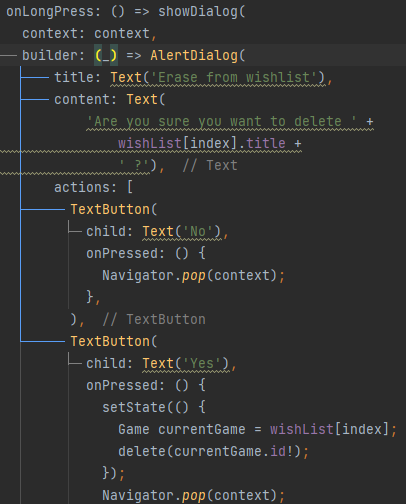
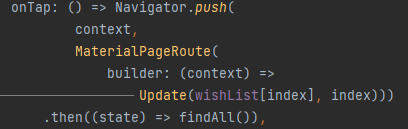
* Main.dart: Clasa principala cu lista de jocuri, contine:



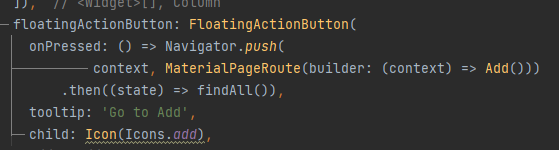
Metoda initState() este apelata tot timpul la initierea ecranului curent. Aici apelam functia ajutatoare findAll() care la randul ei apeleaza functia care se ocupa cu recuperarea entitatilor din baza de date. Metoda \_refresh() alaturi de functia integrata setState() ne ajuta sa facem build la tot ecranul curent fara a ne deplasa in alt ecran.



In metoda build() sunt create si instantiate toate elementele de UI (User Interface), principalul fiind un ListView care este populat cu datele primite de la baza de date.

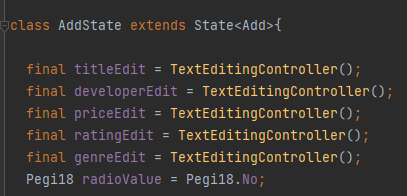


In metodele onTap() si onLongPress() scriem ce comportament ar trebui sa aiba aplicatia in momentul in care interactionam cu un element din ListView. La click scurt pe un element din lista se deschide ecranul pentru editarea elementului respectiv La click lung pe un element din lista apare un pop-up pentru confirmarea stergerii elementului, respectiv apelarea metodei ajutatoare.

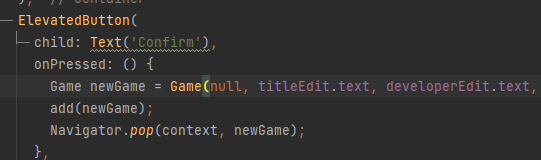


In partea de jos a ecranului avem si un buton care deschide la apasare ecranul aferent introducerii unui joc nou in baza de date.

* Add.dart: Clasa pentru adaugarea unui joc nou



In clasa de AddState incepem prin initializarea tutoror inputurilor pe care userul trebuie sa le completeze pentru a adauga un joc nou. Acestea sunt in principal casete de editare text. In metoda build() atribuim fiecarui text editing controller un container pentru a putea fi vizibil la utilizator.



In momentul in care butonul pentru confirmarea adaugarii este apasat, jocul este trimis catre baza de date si revenim in ecranul principal.

* Update.dart: Clasa pentru editarea unui joc deja existent

Similara ca si comportament cu Add, diferenta fiind la initierea ecranului cand se porneste cu toate datele jocului care a fost selectat din lista.

# Imbunatatiri

* Adaugarea de exceptii pentru cazurile speciale ( Field-uri goale, tipuri incompatibile de variabile);
* Functii de search / filtrari pe diferite proprietati ale entitatilor
* Interfata grafica mai bogata