**Universitatea Transilvania Brasov**

**Facultatea de Matematica si Informatica**

****

Lucrare sementriala

**Aplicatie Android**

**TaskTracker**

**Studenti:**

**Bolba-Mateescu Andrei-Ioan**

**Acatrinei Stefan**

**Cuprins**

[1. Prezentare generala a aplicatiei 2](#_Toc107441911)

[2. Authentification 3](#_Toc107441912)

[**2.1.** **Working method** 3](#_Toc107441913)

[**2.2.** **Register** 4](#_Toc107441914)

[**2.3.** **Login** 6](#_Toc107441915)

[**2.4.** **Logout** 8](#_Toc107441916)

[3. Profil personal 9](#_Toc107441917)

[**3.1.** **User table** 9](#_Toc107441918)

[**3.2.** **Create account** 9](#_Toc107441919)

[**3.3.** **See account** 11](#_Toc107441920)

[**3.4.** **Upate account** 12](#_Toc107441921)

[**3.5.** **Delete account** 14](#_Toc107441922)

[4. Tasks 14](#_Toc107441923)

[**4.1.** **Task table** 14](#_Toc107441924)

[**4.2.** **Create task** 14](#_Toc107441925)

[**4.3.** **See tasks** 15](#_Toc107441926)

[**4.4.** **Delete task** 16](#_Toc107441927)

[**4.5.** **Mark task as finished** 17](#_Toc107441928)

[5. Design 17](#_Toc107441929)

[**5.1.** **Tranzitie intre activitati** 17](#_Toc107441930)

[**5.2.** **Drawable** 18](#_Toc107441931)

[**5.3.** **Mipmap** 19](#_Toc107441932)

[**5.4.** **Theme** 19](#_Toc107441933)

[**5.5.** **Layout** 19](#_Toc107441934)

[**5.5.1.** **Button** 19](#_Toc107441935)

[**5.5.2.** **ImageButtons** 20](#_Toc107441936)

[**5.5.3.** **EditText** 21](#_Toc107441937)

[**5.5.4.** **TextView** 21](#_Toc107441938)

[**5.5.5.** **LinearLayout** 22](#_Toc107441939)

[**5.5.6.** **RelativeLayout** 22](#_Toc107441940)

[**5.5.7.** **RecyclerView** 23](#_Toc107441941)

[**5.6.** **Values** 23](#_Toc107441942)

[**5.6.1.** **Strings** 23](#_Toc107441943)

[**5.6.2.** **Colors** 24](#_Toc107441944)

[6. Posibile dezvoltari 25](#_Toc107441945)

[7. Bibliografie 25](#_Toc107441946)

# **Prezentare generala a aplicatiei**

TaskTrackerApp este o aplicatie ce isi doreste sa ajute user isa fie mai productivi de-a lungul zilei prin tinerea evidentei task-urilor ce le au de facut in ziua respective. Astefl ei vor putea adauga task-uri, si o posibila descrie a task-ului pentru a stii ce au de facut in ziua respectiva.

De asemenea acestia vor putea sa modifice datele contului daca isi doresc acest lucru, avand o pagina unde isi pot vedea datele personale.

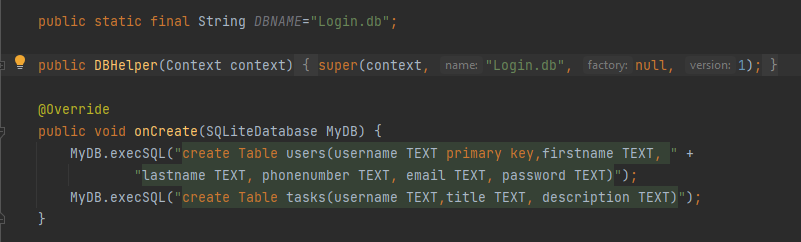
Cand un task este finalizat acesta va trece automat la finalul listei. Lista de task-uri este reprezentata de un **RecycleView** ce preia date din baza de date din tabele **tasks**, depre care vom discuta la sectiunea **4.1 Task table.**

# **Authentification**

## **Working method**

Pentru partea de autentificare am folosit SQLite pentru baza de date. Acesta este o baza de date locala, iar acesta se genereaza in momentul instalarii aplicatiei pe dispozitiv.

In momentul instalarii aplicatiei se creeaza atat baza de date, cand si tabele din aceasta (Fig.1) .



1. Fig.1

In momentul dezinstalarii aplicatiei se sterg tabele din baza de date, concomitent cu baza de date(Fig.2) .

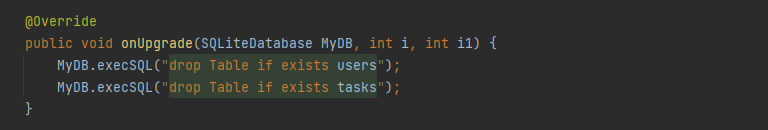


Fig. 2

Astfel, iar o reinstalare ulterioara a aplicatiei baza de date se va crea din nou acesta va fii goala, iar tabele nu vor avea nici o inserare in ele. Procesul urmat in cazul reinstalarii aplicatiei este (Fig.2) apoi (Fig.1).

Aceste metode se gasesc in clasa **DBHelper** ce extinde **SQLiteOpenHelper** (Fig.3).



(Fig.3)

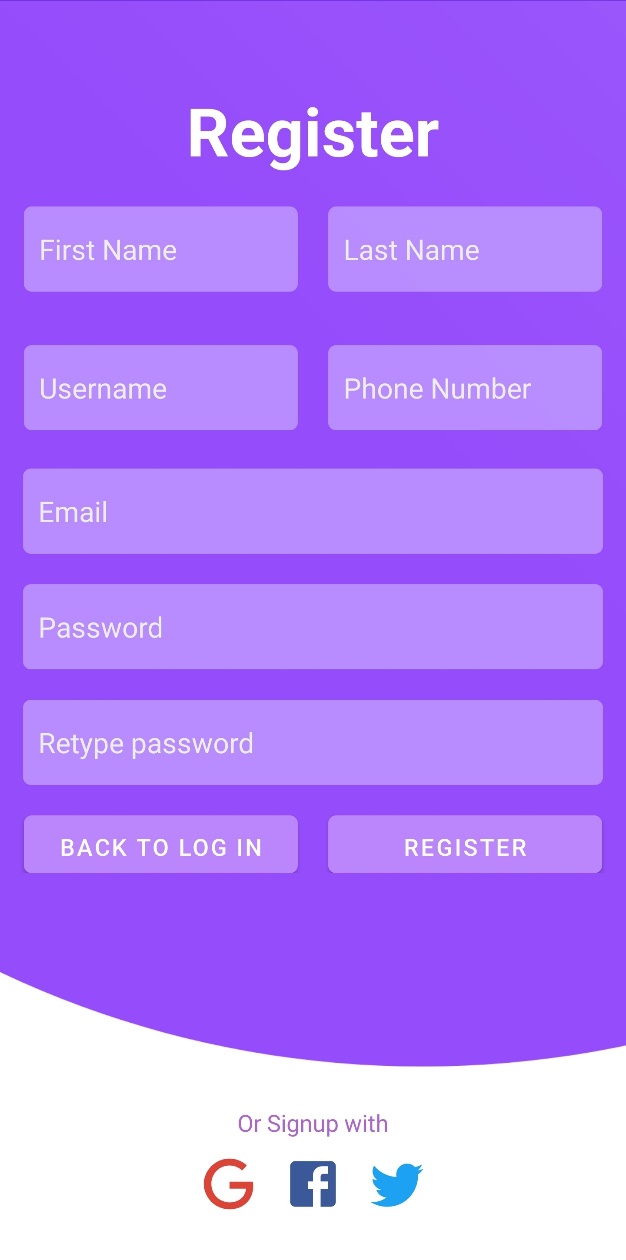
## **Register**

Pentru creea de contul de utilizator de folosim de tabela users, iar datele de la utilizator le luam din **activity\_register.xml** (Fig.3), ce este conentata la clasa **RegisterActivity** ce extinde **AppCompatActivity**.

Username-ul si email-ul sunt verificate sa fie unice pentru a putea facilta conetarea pe baza de username. Mai multe de spre acesr subiect la **2.3 Login.**

Email-ul este validat de regex, pentru a avea forma corecta de email(Fig.4), iar parola trebuie sa indeplineasca elemente de securitate precum:

* Minim 8 caractere
* Minim o litera mare
* Minim o litera mica
* Minim o litera
* Minim un caracter special



(Fig.3)

Aceste de securitate de verifica in functia **checkPassword** din clasa **Utils** (Fig.5). Acesta returmeaza un cod ce este tratat la apasarea butonului de **Register**. Fiecare cod are mesajul sau.

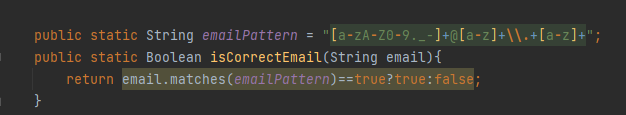


Fig.4



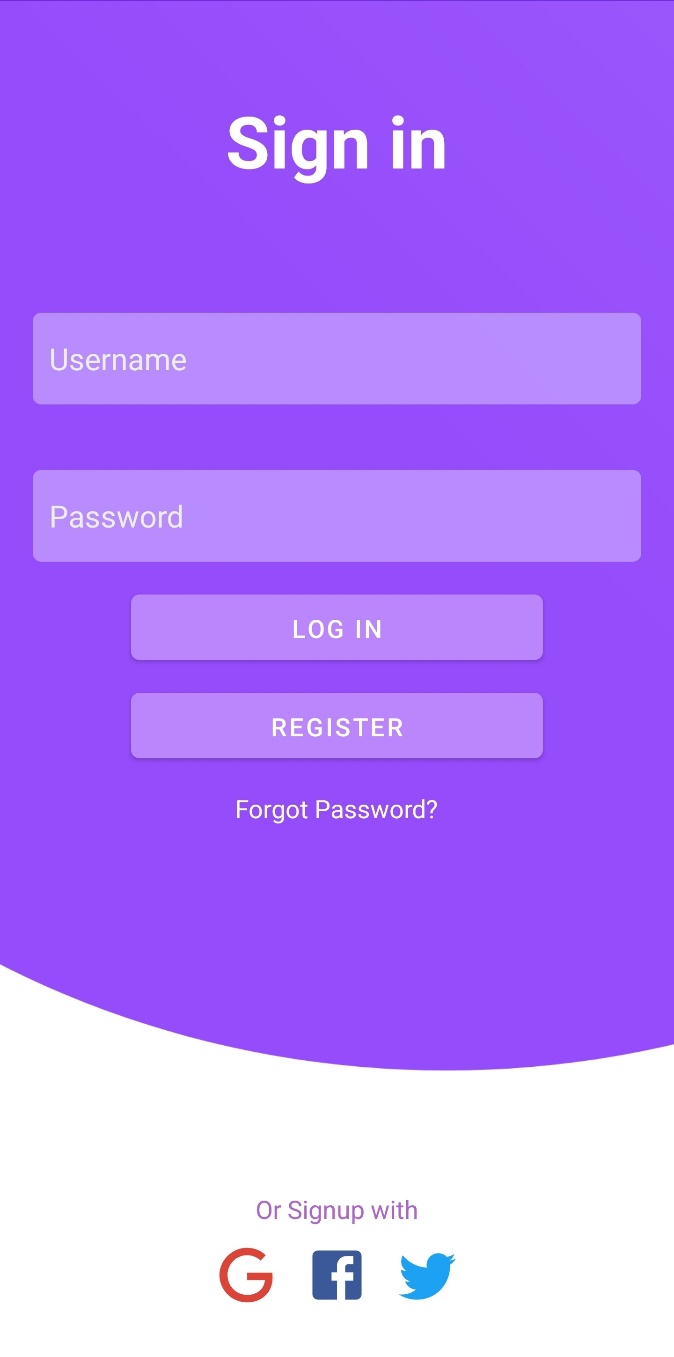
Fig.5

String-ul regex pentru email este definit de **emailPattern** din (Fig.4), iar lista caracterelor speciale este descrisa de string-ul **passPatternSpecial** din (Fig.5).

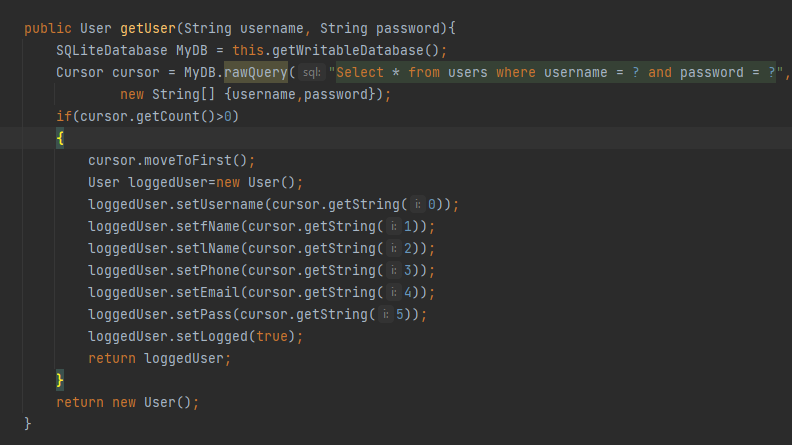
Dupa validarea datelor se insereaza in table **users** nou user. Mai multe despre creare de cont vom discuta la sectiunea **3.2 Create account**.

## **Login**

Acum ca avem cont trebuie sa ne conectam pentru a putea utiliza apicatia. Conetarea la cont se va face pe baza username-ului care este unic si a unei parole(Fig.6). Acesta sunt valiadate la apsarea butonului **LOG IN**, iar daca exista un user care indeplineste aceste conditii este returnat user-ul ce s-a logat. Aceste operatii se ragasesc in **HelperDB** la functie **getUser**(Fig.7).

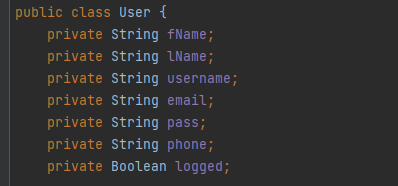


(Fig.6)



(Fig.7)

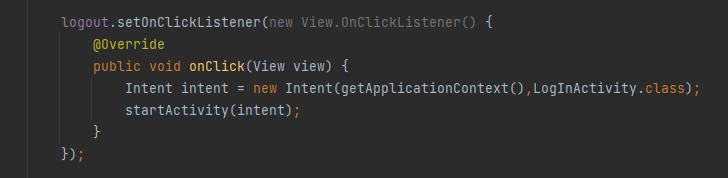
Dupa procesul de autentificare datele user-ului conentat sunt stocate in clasa User(Fig.8), obiect ce va fii pasat intre activitati pentru a se putea utiliza informatiile contului.



(Fig.8)

## **Logout**

Pentru operatie de log este apelata tranzitia catre pagina log in, despre care vom disctuta in sectiunea **5.1 Tranzitie intre activitati** pentru a permite o noua logare(Fig.9). Acesta operatie se poate executa din doua parti, ori de la **Personal Profile**, ori de la **Profile Edit**, dar despre asta mai multe la sectiunea **5. Design**.



(Fig.9)

# **Profil personal**

## **User table**

Tabele de user este format din 6 campuri:

1. First Name
2. Last Name
3. Phone Number
4. Username
5. Email
6. Parola

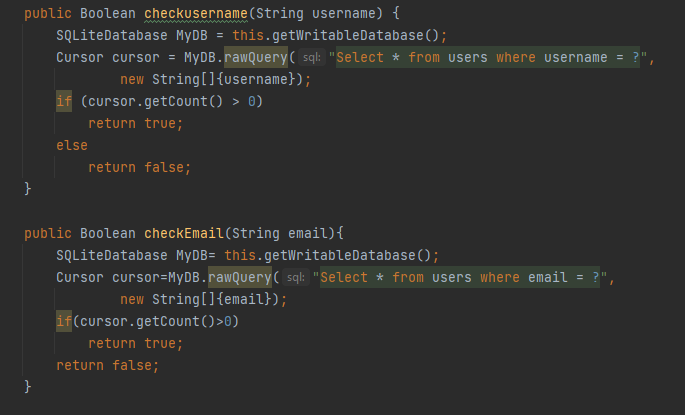
Toate campurile sunt de tip TEXT si retine datele utilizatorului. Cheia primara este redate de campul username, cand ce este verificat sa fie unicat la fiecare creare de cont. Asemena se intampla si cu email-ul, chiar daca acesta nu este chieie primara.

Opratiile **CRUD** effectuate pe tabela users sunt efectuate in clasa **DBHelper,**  dar despre aceste operatii vom discuta in sectiunile ce urmeaza.

## **Create account**

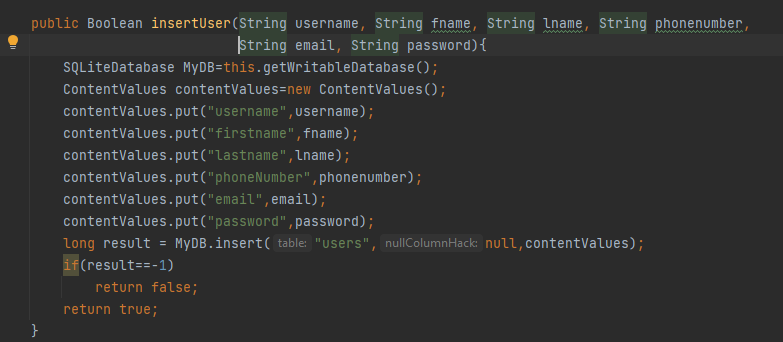
La operatie de creare a contului datele introduse de utilizator trebuie sa treaca de un numar de validatori precum:

* Username-ul trebuie sa fie unic(Fig.10)
* Email-ul trebuie sa fie unic(Fig.10)
* Email-ul trebuie adecvat
* Sectiuna de parola si reintroducerea parolei trebuie sa fie identice
* Parola trebuie sa fie conforma(Vezi Sectiunea **2.2 Register**, pag 3)
* De asemena nu trebuie sa fie campuri nule



(Fig.10)

Dupa ce toate aceste cerinte sunt indeplinite, se creeaza conutul nou, se executa operatie de inserare in baza de date(Fig.11), apoi facandu-se tranzitia catre pagina de log in.



(Fig.11)

## **See account**

Opreatia de vizualizare a contului se executa in metoda **onCreate**(Fig.12) a clasei responsabile de **PersonalProfile**.

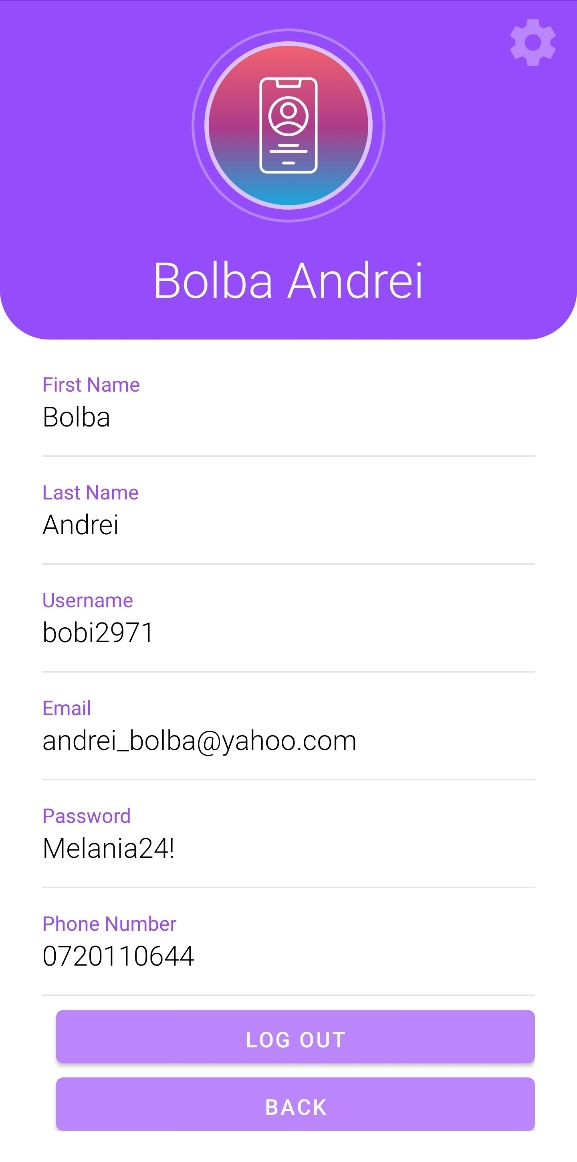
Acesta primeste obiectul **loggedUser**, ce contine date utilizatorului conentat. Apoi sunt populate campurile din pagina profilului cu datele utilizatorului(Fig.13).

De aici utilizatorul poate opta pentru a se intoarce la pagina principala, se poate deloga sau poate sa isi editeze datele contului, despre care vom vorbi in sectiunea urmatoare, operatie ce este reprezentate de iconita **“Settings”** din partea sus dreapta a paginii.

Despre parte de design a acestei pagini vom dezvolta mai mult in sectiunea **5.Design**, unde se vor przenta elementele si culorile folosite.



(Fig.12)



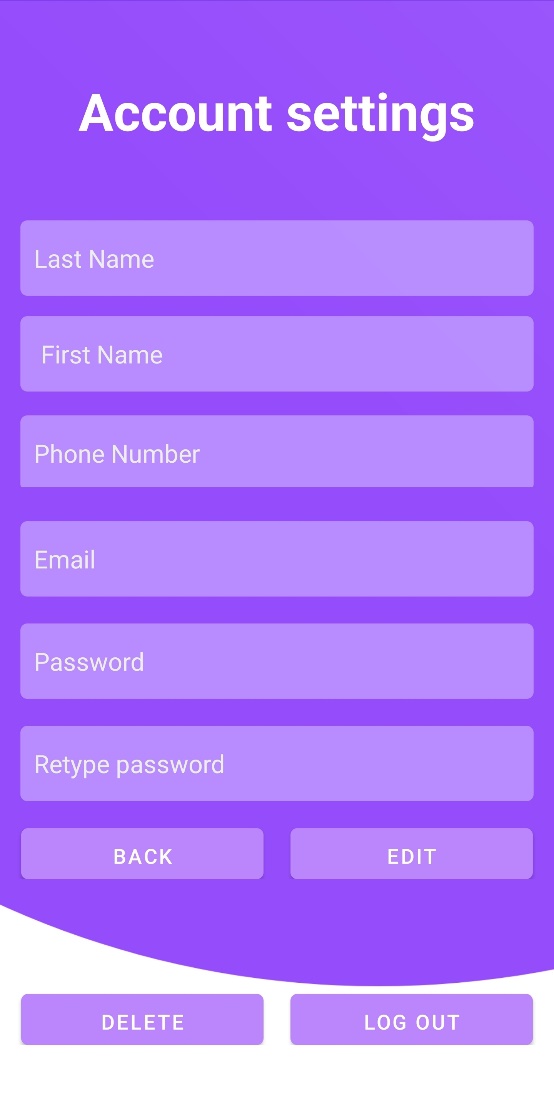
(Fig.13)

## **Upate account**

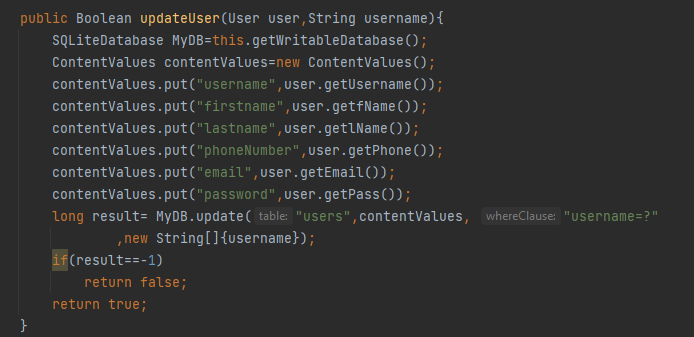
Editarea contuluise efectueaza in activitatea **AccountSettings**,(Fig.14) care are cam acceasi functionalitate cu activitatea de **Register** (Sectiunea **2.2 Register**). Sunt diferite urmatoarele aspect:

* 1. Trebuie sa fie macar un camp completat la **AccountSettigns**(exceptia este la parola unde trebuie sa fie doua), fata de **Register** unde trebuie sa fie toate complete.
  2. Username-ul nu apare la AccountSettigns deoarece acesta nu se va putea schimba fiind chieie primara.

Astfel al apasarea butonul de edit se verifica email-ul si parolele, daca acestea exista urmand sistemul prezentat in Sectiunea **2.2 Register.** Apoi dupa validare se va actualiza si table **users**, apelandu-se functie **updateUser** din clasa **DBHelper**(Fig.15).



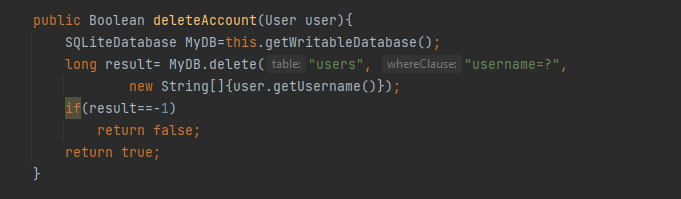
(Fig.14)



(Fig.15)

## **Delete account**

Operatiea de stergere a contului este posibila din activitatea **AccountSettigns** (Fig.14). Acesta apeleaza functia de **deleteAccount** din clasa **DBHelper**(Fig.16).



(Fig.16)

La apasarea butonului **DELETE,** se sterge contul si aplicatia trece la activitatea **LogIn**.

# **Tasks**

## **Task table**

Tabele de tasks este format din 3 campuri:

1. Username
2. Title
3. Description

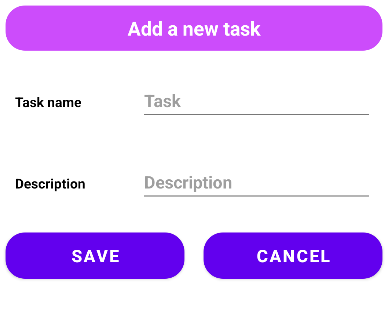
Toate campurile sunt de tip TEXT si retine datele task-ului introdus de utilizator.

Opratiile **CRUD** effectuate pe tabela tasks sunt efectuate in clasa **DBHelper,**  dar despre aceste operatii vom discuta in sectiunile ce urmeaza.

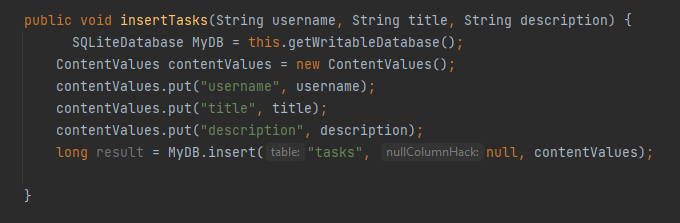
## **Create task**

Cand utilizatorul doreste sa adauge un nou task, va apase pe semnul de plus, ce este reprezentat de un floating button. La apasarea acestuia se va deschide un pop-up (Fig.17) unde userul sa poata introduce infromatii despre task-ul respectiv.

Apoi dupa ce sunt validate datele se executa inserarea in tabela, prin functia **insertTasks** din **DBHelper**(Fig.18)**.**

****

(Fig.17)

****

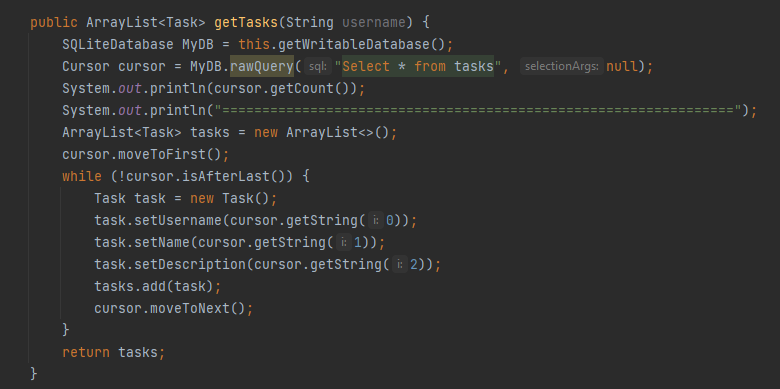
(Fig.18)

## **See tasks**

Dupa log in se afiseaza toate task-urile introduse de utilizator. Acestea se selecteaza din baza de date din tabelul **tasks** si se afiseaza intr-un **RecycleView,** avand desing-ul specific creat pentru iteme(Fig.19). Selectarea se va face cu ajutorul fucntiei **gettTasks** din **DBHelper**(Fig.20)**.**

****

(Fig.19)



(Fig.20)

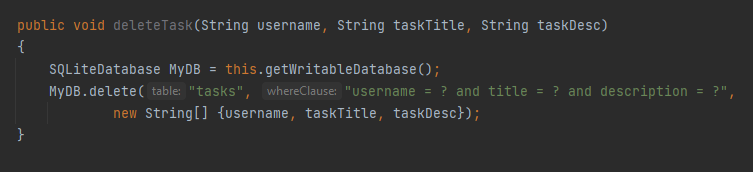
## **Delete task**

Cand utilizatorul doreste sa stearga un nou task, va apase pe butonul de pa task(Fig.21). La apasarea acestuia se va sterge respectivul task.

Apoi dupa ce sunt validate datele se executa inserarea in tabela, prin functia **deleteTask** din **DBHelper**(Fig.22)**.**

****

(Fig.21)

****

(Fig.22)

## **Mark task as finished**

O data ce task-ul este setat ca fiind finalizat acesta se sterge. Despre operatia de stergere am vorbit mai sus. Deci, cu alte cuvinte marcarea ca finalizat a task-ului este operatie similara cu cea a stergerii uni task. Aceasta actiune se declanseaza la apasarea butonului **Done**  de pe task(Fig.23).



(Fig.23)

# **Design**

## **Tranzitie intre activitati**

Fiind o aplicatie ce foloseste mai multe activitati, este necesara tranzitia intre ele. Acesta se realizeaza la nivel de UI prin apasarea de buton. Cat despre partea de cod aplicatie face tranzitia prin intermediul lui **Intent**

Astfel un exemplu de cod ce face tranzitia intre activitati folosind **Intent** (Fig.24)



(Fig.24)

Astfel la executarea lui **startActivity(intent)** se efectueaza efectiv tranzitia intre activitati.

## **Drawable**

O resursă desenabilă este un concept general pentru un grafic care poate fi desenat pe ecran și pe care îl puteți prelua cu API-uri precum **getDrawable(int)** sau îl puteți aplica la o altă resursă XML cu atribute precum android:drawable și **android:icon**.

Astfel in aplicatia TaskTracker aceste resure ne-au ajutat sa avem iconite, iar aplicatie sa fie mai user friendly.

Un exemplu de cod penteu o resursa desenabila(Fig.25), utilizarea ei in codul din fisierul XML(Fig.26) si afisarea ei in aplicatie(Fig.27).



(Fig.25)



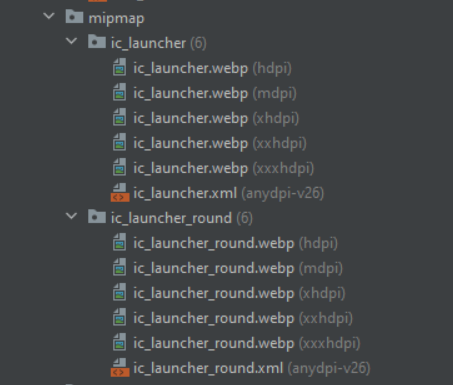
(Fig.26)



(Fig.27)

## **Mipmap**

Acesta subcategorie a dosarului **res** a fost utilizata pentru a crea si utiliza iconita aplicatiei. Astfel continutul acestul fisier este compus din mai doua dosare ce contin fiecare mai multe fisiere cu extensia **.webp** pe diferite marimi(Fig.28).



(Fig.28)

## **Theme**

Acesta subcategorie faciliteaza implementarea themei dark pentru aplicatie. Momentan TaskTracker dispune doar de light mode.

## **Layout**

Aplicatia este impertita pe mai multe activitati, fiecare avand cel putin una din urmatoarele elemente specifice Android Studio.

## **Button**

Button-ul este utilizat pentru a face tranzitia intre activitati si pentru a executa operatiile CRUD atat pentru **users** cat si pentu **tasks** si pentru a trimite datele preluate din EditTexr inspre a fii validate.

Un exemplu de cod pentru button(Fig.29) si un button in aplicatie(Fig.30).



(Fig.29)

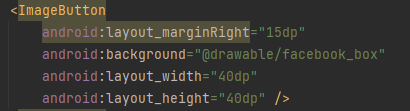


(Fig.30)

## **ImageButtons**

ImageButton sutn practice butoane care in loc de text si o culoare de background au doar o imagine.

Un exemplu de cod pentru button(Fig.31) si un button in aplicatie(Fig.32).



(Fig.31)



(Fig.32)

## **EditText**

EditText-ul este utilizat pentru a prelua datele de la utilizator, mai apoi trimitand datele mai departe sprevalidare la apasarea buttoanelor**.**

Un exemplu de cod pentru EditText(Fig.33) si un EditText in aplicatie (Fig.34).



(Fig.33)



(Fig.34)

## **TextView**

EditText-ul este utilizat pentru a afisate date ce nu dorim sa fie modficare si care ajuta la intelegea functionalitatii aplicatiei.

Un exemplu de cod pentru TextView(Fig.35) si un TextView in aplicatie (Fig.36).

****

(Fig.35)

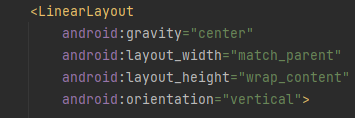
****

(Fig.36)

## **LinearLayout**

LinearLayout este un grup de vizualizare care aliniază toți copiii într-o singură direcție, vertical sau orizontal. Puteți specifica direcția aspectului cu atributul **android:orientation**.

Un exemplu de cod cu LinearLayout(Fig.37).

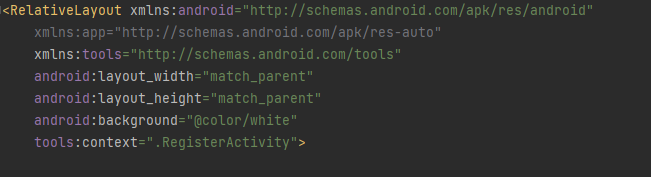


(Fig.37)

## **RelativeLayout**

RelativeLayout este un grup de vizualizări care afișează vizualizările copil în poziții relative. Poziția fiecărei vizualizări poate fi specificată ca fiind relativă la elementele frate (cum ar fi la stânga sau dedesubtul unei alte vizualizări) sau în poziții relativ la zona parinte RelativeLayout (cum ar fi aliniată la partea de jos, la stânga sau la centru).

Un exemplu de cod cu RelativeLayout (Fig.38).

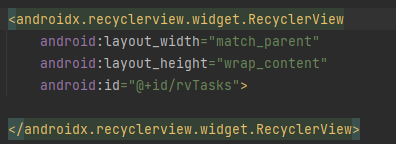


(Fig.38)

## **RecyclerView**

RecyclerView este ViewGroup care conține vizualizările corespunzătoare datelor dvs. Este o vizualizare în sine, așa că adăugați RecyclerView în aspectul dvs. așa cum ați adăuga orice alt element UI.

Un exemplu de cod cu RecyclerView (Fig.39).



(Fig.39)

## **Values**

In dosarul de values gasim doua subcategorii si anume strings si colors.

## **Strings**

String ne ajuta sa definim siruri de caractere pe care le putem folosi in mai multe locuri, iar la o modificare ulterioara a unui string nu va trebui sa modificam peste tot unde a fost folosit, ci se va schimba automat. Un exemplu de string este prezentata in (Fig.40).



(Fig.40)

Astfel la schimbarea textul cu numele **app\_name** se va schimba peste tot unde a fost utilizat. Acest lucru ajuta la organizarea codului si usurarea modificarii ulterioare a acestuia.

## **Colors**

Subcategoria Colors se bazeaza pe acelasi sistem ca si strings, dar ca acestea reprezinta culori.(Fig.41).

De asemena la schimbarea culorii cu numele **circleBorder** se va schimba peste tot unde a fost utilizat. Acest lucru ajuta, de asemenea, la organizarea codului si usurarea modificarii ulterioare a acestuia.



(Fig.41)

# **Posibile dezvoltari**

Aplicatie TaskTracker nu este intr-o varianta finala, asa ca se mai pot dezvolta feature-uri. Unele la care ne-am gandit ar fii:

* Sa se prioritizeze task-urile, astfel utilizatorul sa poate vedea improtanta task-urilor si sa poate organiza mia eficient ziua
* Sa se implementeze un sistem de reminder pentru a avertiza user-ul ca mai are task-uri de facut. Asta evident daca task-urile exista.
* Sa se implementeze un sistem prin care mai multi useri sa poate imparti un task
* Sa se creeze organizarea task-urilor pe zile, saptamani,luni etc

Pe langa aceste idei evident mai sunt multe ce se pot implementa. Aici sunt doar catev aidei.

# **Bibliografie**

* <https://www.youtube.com/watch?v=8obgNNlj3Eo>
* <https://developer.android.com/courses>
* <https://elearning.unitbv.ro/course/view.php?id=9403>
* <https://www.youtube.com/watch?v=8obgNNlj3Eo&t=444s>
* <https://www.youtube.com/watch?v=312RhjfetP8>
* <https://www.youtube.com/c/Foxandroid01>
* <https://www.youtube.com/watch?v=5lNQLR53UtY>
* <https://stackoverflow.com>
* <https://www.youtube.com/watch?v=onFs0GvylsU&list=PLlkSO32XQLGpF9HzRulWLpMbU3mWZYlJS&index=9>