**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](#_Toc107412457)

[**2.** **Authentification** 2](#_Toc107412458)

[**2.1.** **Working method** 2](#_Toc107412459)

[**2.2.** **Register** 3](#_Toc107412460)

[**2.3.** **Login** 5](#_Toc107412461)

[**2.4.** **Logout** 7](#_Toc107412462)

[**3.** **Profil personal** 8](#_Toc107412463)

[**3.1.** **User table** 8](#_Toc107412464)

[**3.2.** **Create account** 8](#_Toc107412465)

[**3.3.** **See account** 9](#_Toc107412466)

[**3.4.** **Upate account** 11](#_Toc107412467)

[**3.5.** **Delete account** 11](#_Toc107412468)

[**4.** **Tasks** 11](#_Toc107412469)

[**4.1.** **Task table** 11](#_Toc107412470)

[**4.2.** **Create task** 11](#_Toc107412471)

[**4.3.** **Edit task** 11](#_Toc107412472)

[**4.4.** **Delete task** 11](#_Toc107412473)

[**4.5.** **Mark task as finished** 11](#_Toc107412474)

[**5.** **Design** 12](#_Toc107412475)

[**5.1.** **Register** 12](#_Toc107412476)

[**5.2.** **LogIn** 12](#_Toc107412477)

[**5.3.** **MainPage** 12](#_Toc107412478)

[**5.4.** **PersonalProfile** 12](#_Toc107412479)

[**5.5.** **Edit Profile** 12](#_Toc107412480)

[**5.6.** **Tranzitie intre activitati** 12](#_Toc107412481)

[**6.** **Bibliografie** 12](#_Toc107412482)

# **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) .

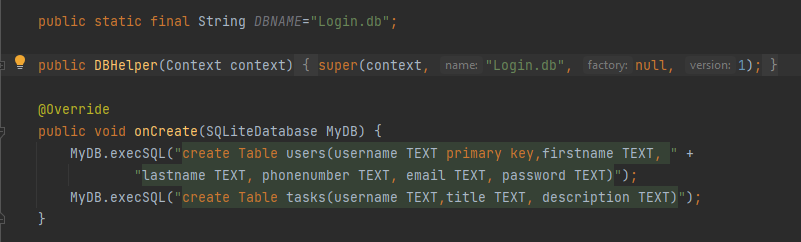


Fig.1

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

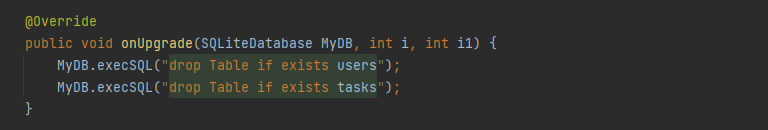


Fig. 2

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)

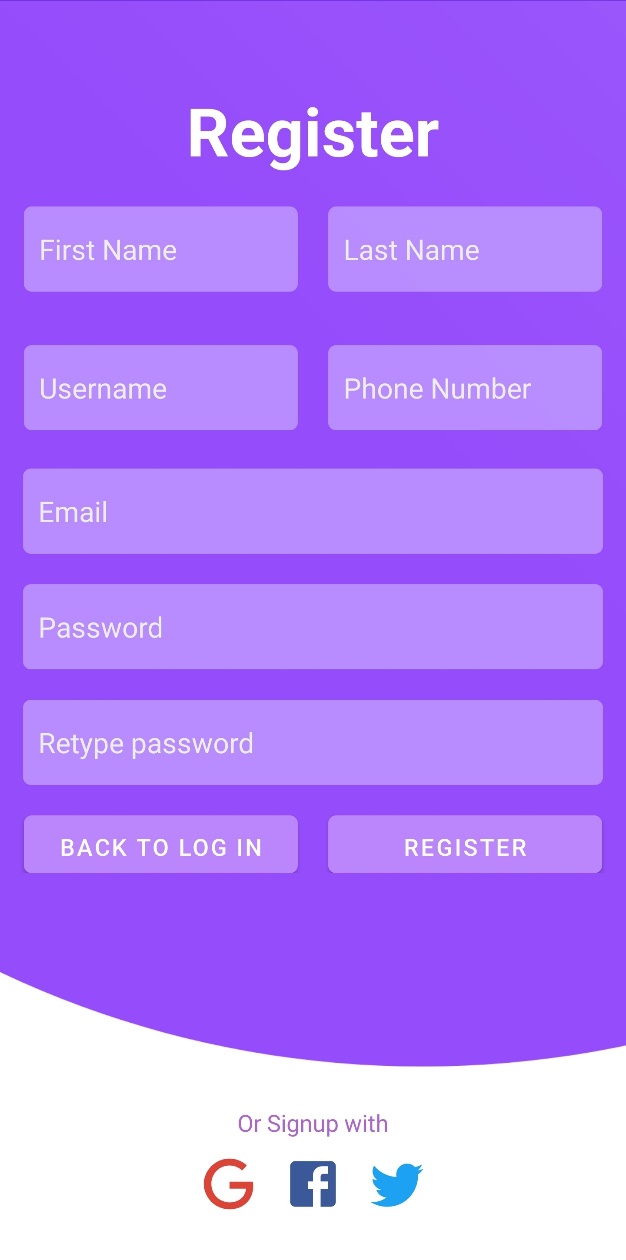
## **Register**

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.

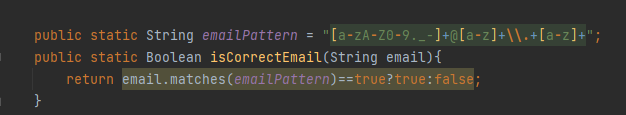


Fig.4



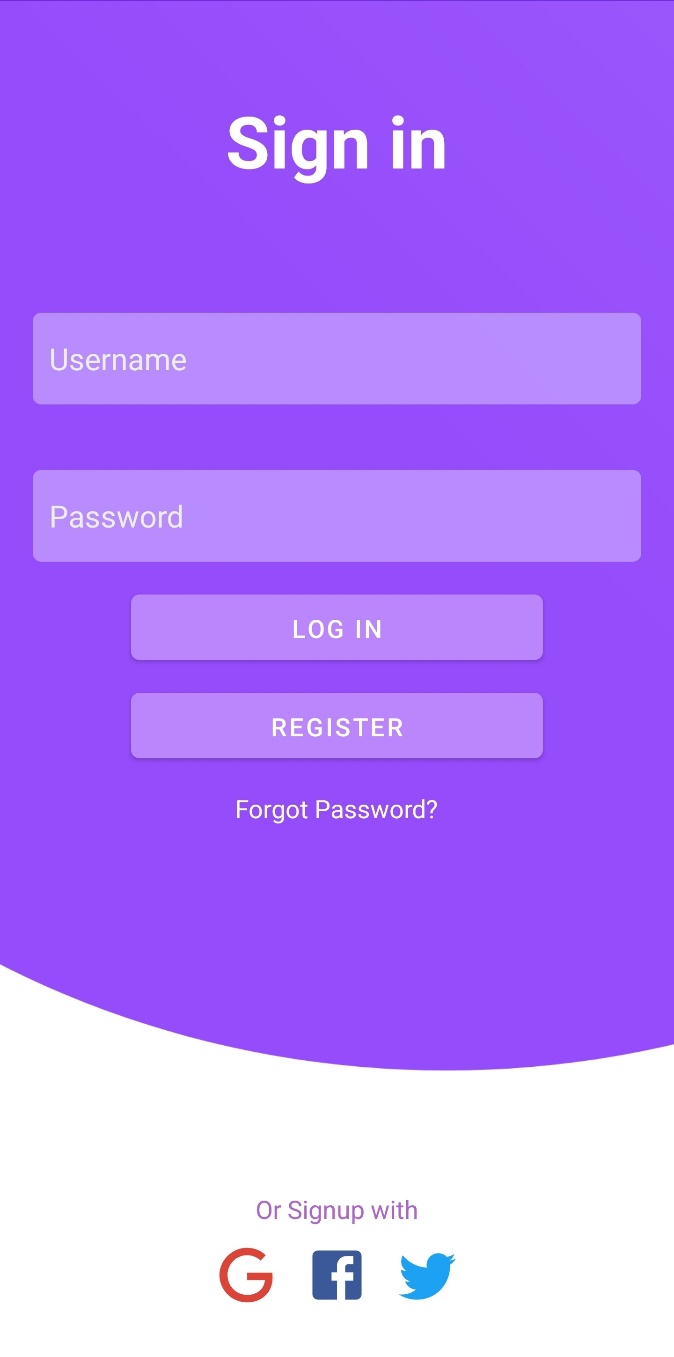
Fig.5

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).

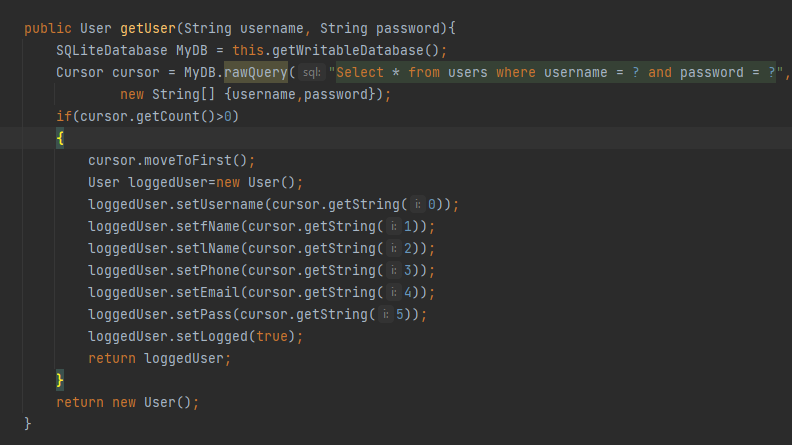
Dupa validarea datelor se insereaza in table **users** nou user. Mai multe despre creare de cont vom discuta la sectiunea **3.2 Create account**.

## **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).

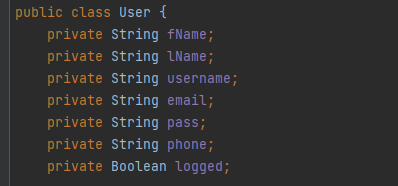


(Fig.6)



(Fig.7)

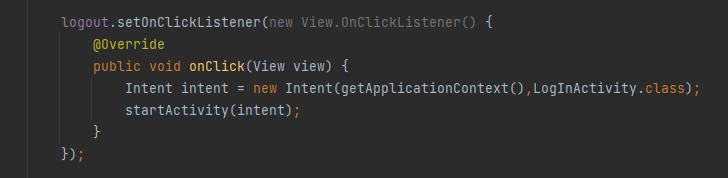
Dupa procesul de autentificare datele user-ului conentat sunt stocate in clasa User(Fig.8), obiect ce va fii pasat intre activitati pentru a se putea utiliza informatiile contului.



(Fig.8)

## **Logout**

Pentru operatie de log este apelata tranzitia catre pagina log in, despre care vom disctuta in sectiunea **5.6 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

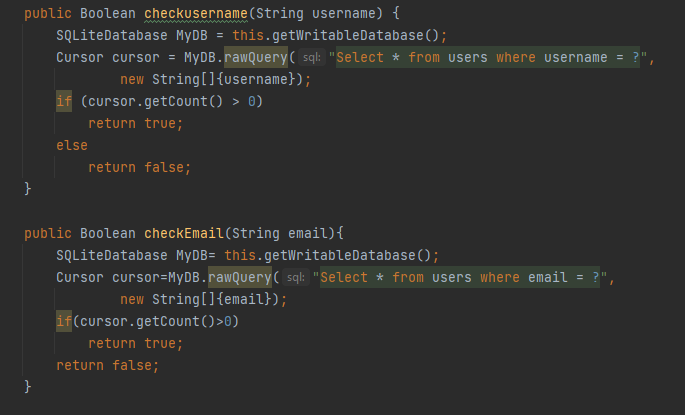
Toate campurile sunt de tip TEXT si retine datele utilizatorului. Cheia primara este redate de campul username, cand ce este verificat sa fie unicat la fiecare creare de cont. Asemena se intampla si cu email-ul, chiar daca acesta nu este chieie primara.

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

## **Create account**

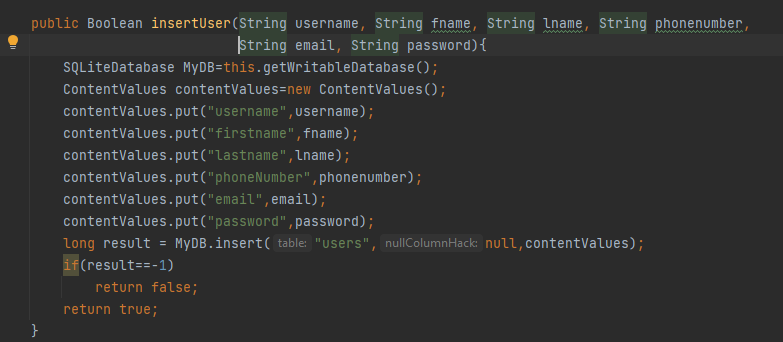
La operatie de creare a contului datele introduse de utilizator trebuie sa treaca de un numar de validatori precum:

* 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**.

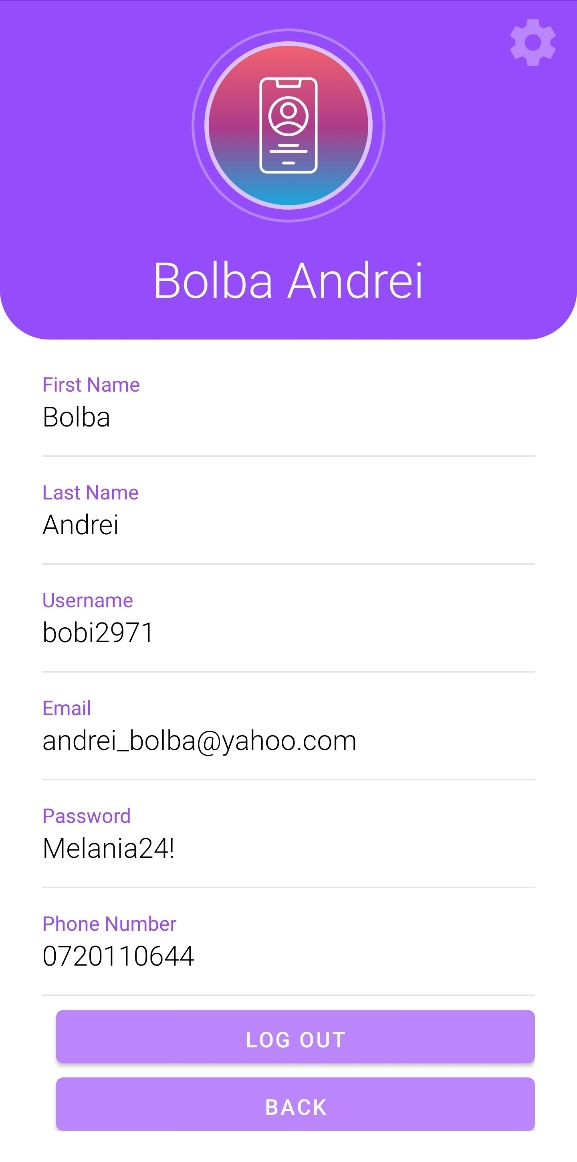
Acesta primeste obiectul **loggedUser**, ce contine date utilizatorului conentat. Apoi sunt populate campurile din pagina profilului cu datele utilizatorului(Fig.13).

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)



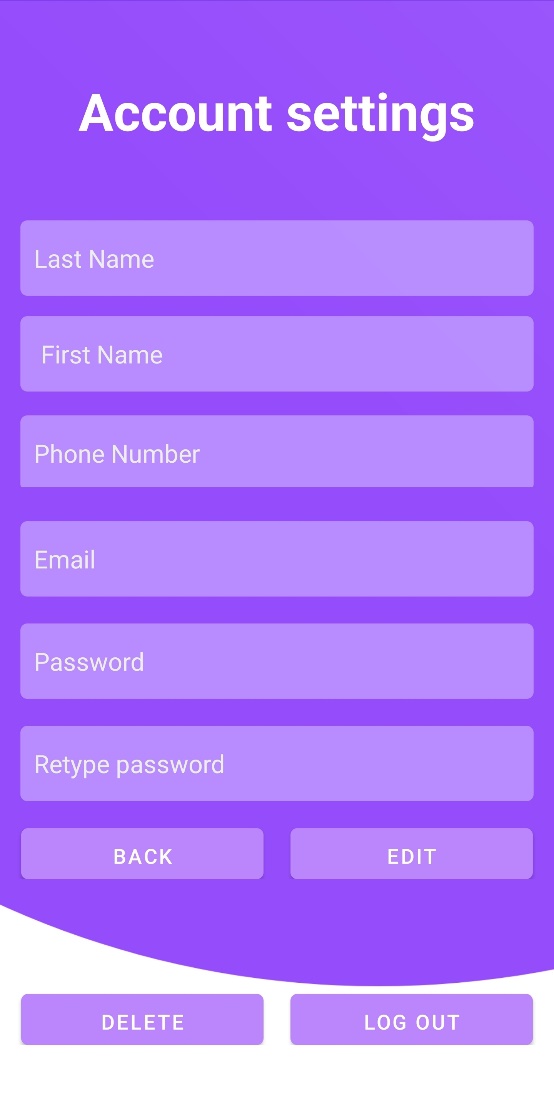
(Fig.13)

## **Upate account**

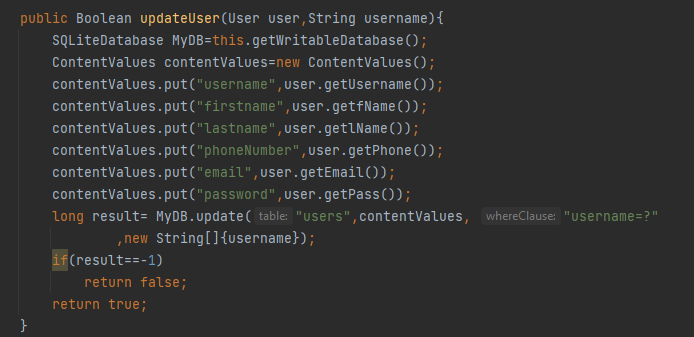
Editarea contuluise efectueaza in activitatea **AccountSettings**,(Fig.14) care are cam acceasi functionalitate cu activitatea de **Register** (Sectiunea **2.2 Register**). Sunt diferite urmatoarele aspect:

* 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).



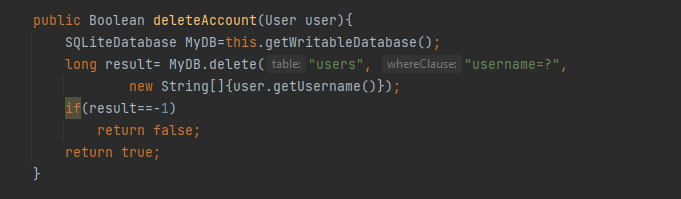
(Fig.14)



(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**

## **Edit task**

## **Delete task**

## **Mark task as finished**

# **Design**

## **Register**

## **LogIn**

## **MainPage**

## **PersonalProfile**

## **Edit Profile**

## **Tranzitie intre activitati**

# **Bibliografie**