Proiect – Baze de Date

Administrarea activităților Asociației Exploratori pentru Viitor

Gheoace Mircea 333AA

# Descrierea cerințelor

Aplicatie cu baze de date relationale avand interfata vizuala. Această aplicație are rolul de a asigura administrarea activităților și persoanelor din cadrul Asociației Exploratori pentru viitor, care este similară în multe aspecte Cercetașilor. Aceasta este destinată pentru utilizarea de către secretariatul național.

Lista funcționalităților și componentelor: Salvarea informațiilor despre exploratori(participanții care sunt în clasa a V-a până în clasa a VII-a), a părinților acestora(exploratorii fiind minori este nevoie de acordul părinților pentru participarea la activitățile asociației), a proiectelor asociației și a structurii organizatorice a acesteia la nivel național. De asemenea informațiile pot ulterior modificate și se pot face prelucrări în vederea cuantificării performanțelor exploratorilor, a unităților organizatorice(cluburile și conferințele) precum și gradul de succes al proiectelor pe baza numărului de participanți și a numărului de ore petrecute voluntari în acestea.

Organizarea asociației este de formă piramidală. Mai mulți exploratori formează un club(acesta reprezentând forma de organizare la nivelul orașului), mai multe cluburi formează o conferință(fiind forma de organizare la nivelul unei zone). O conferință reprezintă toate cluburile dintr-o zonă geografică(ex: Muntenia, Oltenia, etc.). Exemple de cluburi: Ezra, Neemia, etc.

# Tabele

Exploratori

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IDExplorator | Nume | Prenume | CNP | NrSpecializari | Grad | Instructor | IDParinte | IDClub |
| Integer | Text | Text | Text | Integer | Text | Integer | Integer | Integer |

|  |  |
| --- | --- |
| DataStart | DataFinal |
| Text | Text |

\*

Parinti

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IDParinte | Nume | Prenume | Gen | NrDeTelefon |
| Integer | Text | Text | Text | Text |

Proiecte

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IDProiect | Nume | DataStart | DataFinal | DescriereScurtă |
| Integer | Text | Text | Text | Text |

Cluburi

|  |  |  |  |
| --- | --- | --- | --- |
| IDClub | Nume | IDConferință | IDConducător |
| Integer | Text | Integer | Integer |

Conferințe

|  |  |  |
| --- | --- | --- |
| IDConferință | Nume | IDDirector |
| Integer | Text | Integer |

Tabel de legătură:

ExploratoriInProiecte

|  |  |  |  |
| --- | --- | --- | --- |
| ID | IDProiect | IDExplorator | NrOre |
| Integer | Integer | Integer | Integer |

# Relații

Conferințe – Are(1:N) – Cluburi.

Cluburi –Are(1:N) – Exploratori.

Cluburi – CondusDe(1:1) – Exploratori.

Conferinte – CondusDe(1:1) – Exploratori.

Proiect – ExploratoriInProiecte(N:N) – Exploratori.

Părinți – Are(1:N) – Exploratori. Se ia în calcul un singur părinte pentru fiecare copil.

# Restrângeri și definire tabele

Mai întâi trebuie introduși părinții deoarece Exploratorii au nevoie de ID-ul acestora. Instructorii nu au nevoie să le fie specificați părinții deci nu depinde de instanțe ale Parinti putând fii introduși înainte și după aceștia.

Primary Key ar trebuie să implice NOT NULL conform standardului SQL dar pentru a permite compatibilitatea cu versiuni mai vechi se permit valori NULL pentru cheia primară (SQLite Primary Key, 2019). Ca urmare alături de Primary Key am adăugat și NOT NULL.

***Exploratori***

CREATE TABLE Exploratori(IDExplorator INTEGER NOT NULL primary key AUTOINCREMENT, Nume TEXT NOT NULL, Prenume TEXT NOT NULL, CNP TEXT CHECK(length(CNP) = 13) UNIQUE NOT NULL, NrSpecializari INTEGER NOT NULL, Grad TEXT CHECK(Grad IN ('ucenic','calator','navigator','mesager','ghid-asistent','ghid','master-ghid')), Instructor INTEGER NOT NULL CHECK(Instructor IN (0,1)), IDParinte INTEGER, IDClub INTEGER, DataStart TEXT NOT NULL, DataFinal TEXT NOT NULL,

CONSTRAINT fk\_exploratori\_parinte

FOREIGN KEY (IDParinte)

REFERENCES Parinti(IDParinte)

ON DELETE SET NULL,

CONSTRAINT fk\_exploratori\_club

FOREIGN KEY (IDClub)

REFERENCES Cluburi(IDClub)

ON DELETE CASCADE

)

***Părinti***

CREATE TABLE Parinti( IDParinte INTEGER NOT NULL primary key AUTOINCREMENT, Nume TEXT NOT NULL, Prenume TEXT NOT NULL, Gen TEXT NOT NULL CHECK(Gen IN ('m','f')), NrTelefon TEXT CHECK(LENGTH(NrTelefon) = 10))

***Proiecte***

CREATE TABLE Proiecte(IDProiecte INTEGER NOT NULL primary key AUTOINCREMENT, Nume TEXT NOT NULL, DataStart TEXT NOT NULL, DataFinal TEXT NOT NULL, DescriereScurta TEXT NOT NULL);

***Cluburi***

CREATE TABLE Cluburi3(IDClub INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, Nume TEXT NOT NULL,

IDConferinta INTEGER, IDConducator INTEGER,

CONSTRAINT fk\_cluburi\_conferinte

FOREIGN KEY (IDConferinta)

REFERENCES Conferinte(IDConferinta)

ON DELETE CASCADE,

CONSTRAINT fk\_cluburi\_exploratori

FOREIGN KEY (IDConducator)

REFERENCES Exploratori(IDExplorator)

ON DELETE SET NULL

);

***Conferinte***

CREATE TABLE Conferinte( IDConferinta INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, Nume TEXT NOT NULL, IDDirector INTEGER NOT NULL,

CONSTRAINT fk\_conferinte\_exploratori

FOREIGN KEY (IDDirector)

REFERENCES Exploratori(IDExplorator)

ON DELETE SET NULL);

***ExploratoriInProiecte***

CREATE TABLE ExploratoriInProiect( ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

IDProiect INTEGER NOT NULL, IDExplorator INTEGER NOT NULL, NrOre INTEGER NOT NULL,

CONSTRAINT fk\_exploinproiect\_proiecte

FOREIGN KEY (IDProiect)

REFERENCES Proiecte(IDProiecte)

ON DELETE CASCADE,

CONSTRAINT fk\_exploinproiect\_exploratori

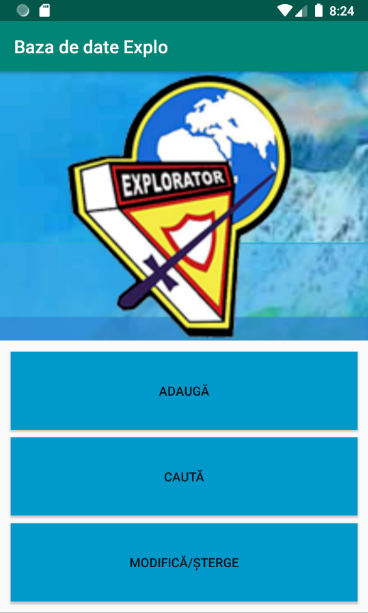
FOREIGN KEY (IDExplorator)

REFERENCES Exploratori(IDExplorator)

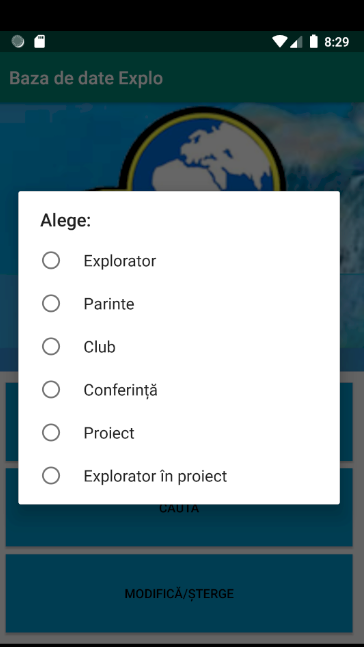
ON DELETE CASCADE

);

# Funcționarea aplicației

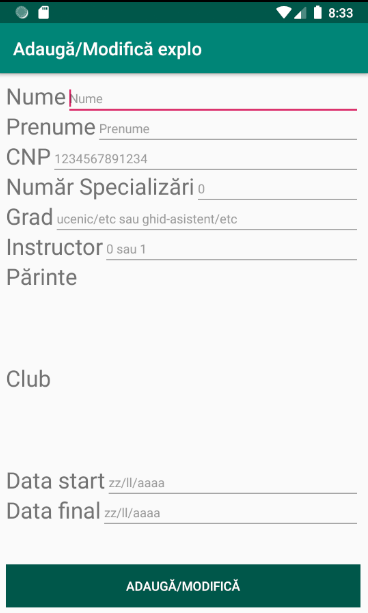


Prima pagină are 3 butoane care permit alegerea funției dorite: de adăugare înregistrări, căutare și modificare a înregistrărilor sau ștergere. Interogările simple și cele complexe pot fi găsite dând click pe butonul căutare, iar la Modifică/Șterge pot fi doar vizionate informațiile introdu-se din toate tabelele, modificate sau șterse.

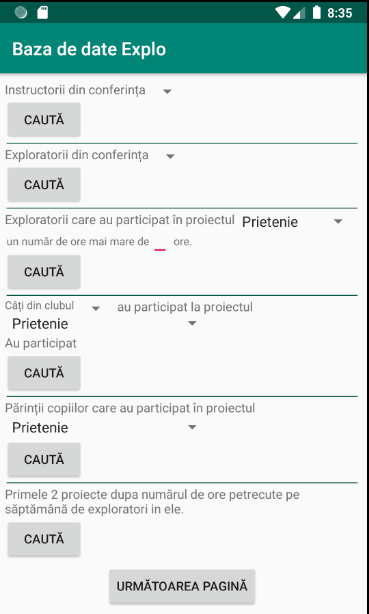


Atunci când se dă click pe una primul sau ultimul buton apare o pagină care permite alegerea tabelului pe care dorim

să-l modificăm.



Aceasta este o pagină de introducere informații despre explorator.



Pagina cu interogări simple este ușor de utilizat, alegerea parametrilor pentru interogării putând fi făcută numai cu valori deja existente în baza de date.

# Interogări Simple

Interogările pot fi găsite în Cautator.java numele tabelelor fiind date sub forma de constante pentru a putea facilita o modificare ușoară a numelor ulterior.

## Interogarea 1

SELECT E.Nume, E.Prenume, E.IDExplorator FROM Exploratori E INNER JOIN Cluburi C ON E.IDClub == C.IDClub INNER JOIN Conferinte Co ON Co.IDConferinta == C.IDConferinta WHERE Co.Nume=’%s’ AND E.Instructor=1;

În loc de %s se află numele Conferinței selectat de utilizator. Ex de valori: Muntenia, Moldova.

## Interogarea 2

SELECT E.Nume, E.Prenume, E.IDExplorator FROM Exploratori E INNER JOIN Cluburi C ON E.IDClub == C.IDClub INNER JOIN Conferinte Co ON Co.IDConferinta == C.IDConferinta WHERE Co.Nume=’%s’ AND E.Instructor=0;

În loc de %s se află numele Conferinței selectat de utilizator. Ex de valori: Muntenia, Moldova.

## Interogarea 3

SELECT E.Nume, E.Prenume, E.IDExplorator FROM Exploratori E INNER JOIN ExploratoriInProiecte EP ON E.IDExplorator == EP.IDExplorator INNER JOIN Proiecte P ON EP.IDProiect == P.IDProiecte WHERE EP.NrOre > nrOre AND P.Nume =’nume’;

nrOre este un număr introdus de utilizator

nume este numele Proiectului selectat de utilizator

## Interogarea 4

SELECT COUNT(E.IDExplorator) AS NrExploratori FROM Exploratori E INNER JOIN ExploratoriInProiecte EP ON E.IDExplorator = EP.IDExplorator INNER JOIN Cluburi C ON E.IDClub = C.IDClub INNER JOIN Proiecte P ON EP.IDProiect = P.IDProiecte WHERE P.Nume = ‘numeProiect’ AND C.Nume = ‘numeClub’;

numeProiect poate fi orice nume de proiect introdus

numeClub poate fi orice nume de club introdus. Ex:

SELECT COUNT(E.IDExplorator) AS NrExploratori FROM Exploratori E INNER JOIN ExploratoriInProiecte EP ON E.IDExplorator = EP.IDExplorator INNER JOIN Cluburi C ON E.IDClub = C.IDClub INNER JOIN Proiecte P ON EP.IDProiect = P.IDProiecte WHERE P.Nume = 'Prietenie' AND C.Nume = 'Moise' ;

## Interogarea 5

SELECT P.Nume, P.Prenume FROM Exploratori E INNER JOIN Parinti P ON E.IDParinte = P.IDParinte INNER JOIN ExploratoriInProiecte EP ON E.IDExplorator = EP.IDExplorator INNER JOIN Proiecte Pr ON Pr.IDProiecte = EP.IDProiect

WHERE Pr.Nume = ‘ numeProiect ‘

numeProiect este numele Proiectului ales de utilizator din lista celor disponibile.

## Interogarea 6

SELECT P.Nume FROM Proiecte P INNER JOIN ExploratoriInProiecte EP ON P.IDProiecte = EP.IDProiect INNER JOIN Exploratori E ON EP.IDExplorator = E.IDExplorator GROUP BY P.IDProiecte, P.Nume HAVING SUM(EP.NrOre) > 0 LIMIT 3;

# Interogări Complexe

## Interogarea 1

SELECT E.Nume, E.Prenume, E.IDExplorator FROM Exploratori E INNER JOIN ExploratoriInProiecte EP ON E.IDExplorator = EP.IDExplorator GROUP BY E.Nume, E.Prenume, E.IDExplorator HAVING EP.NrOre > (SELECT AVG(sumaOre) FROM (SELECT Sum(EP.NrOre) AS sumaOre FROM Exploratori E INNER JOIN ExploratoriInProiecte EP ON E.IDExplorator = EP.IDExplorator INNER JOIN Cluburi C ON E.IDClub = C.IDClub INNER JOIN Conferinte CO ON C.IDConferinta = CO.IDConferinta WHERE CO.Nume = 'numeConferinta' GROUP BY E.Nume, E.Prenume, E.IDExplorator));

„numeConferinta” reprezintă un nume din lista numelor Conferințelor disponibile și este selectat de utilizator.

## Interogarea 2

SELECT C.Nume FROM Cluburi C INNER JOIN Exploratori E ON E.IDClub = C.IDClub GROUP BY C.IDClub HAVING COUNT(E.IDExplorator) > (SELECT AVG(nrExplo) FROM (SELECT COUNT(IDExplorator) As nrExplo FROM Exploratori E1 INNER JOIN Cluburi C1 ON E1.IDClub = C1.IDClub GROUP BY C1.IDClub)) LIMIT 2;

## Interogarea 3

SELECT P.Nume, SUM(EP.NrOre) FROM Proiecte P INNER JOIN ExploratoriInProiecte EP ON P.IDProiecte = EP.IDProiect

WHERE EP.IDExplorator NOT IN (SELECT E1.IDExplorator FROM Exploratori E1 INNER JOIN Cluburi C1 ON E1.IDClub = C1.IDClub

INNER JOIN Conferinte Cf1 ON C1.IDConferinta = Cf1.IDConferinta WHERE C1.Nume = 'Muntenia') AND

EP.IDExplorator IN (SELECT E2.IDExplorator FROM Exploratori E2 INNER JOIN Cluburi C2 ON E2.IDClub = C2.IDClub

INNER JOIN Conferinte Cf2 ON C2.IDConferinta = Cf2.IDConferinta WHERE C2.Nume = 'Moldova' AND C2.IDConducator = E2.IDExplorator)

GROUP BY P.Nume

HAVING SUM(EP.NrOre) > 10

„numeConferinta” reprezintă un nume din lista numelor Conferințelor disponibile și este selectat de utilizator.

n este orice număr natural.

## Interogarea 4

SELECT \* FROM Conferinte Cf INNER JOIN Cluburi C ON Cf.IDConferinta = C.IDConferinta INNER JOIN Exploratori E ON E.IDClub = C.IDClub

INNER JOIN ExploratoriInProiecte EP

GROUP BY CF.Nume

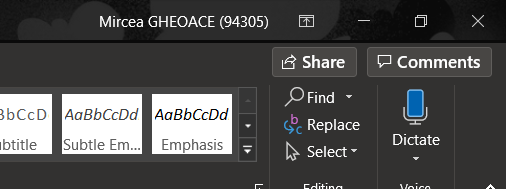
HAVING SUM(EP.NrOre > 1) AND Cf.Nume IN

(SELECT Cf.Nume FROM Conferinte Cf1 INNER JOIN Cluburi C1 ON Cf1.IDConferinta = C1.IDConferinta INNER JOIN Exploratori E1 ON E1.IDClub = C1.IDClub

GROUP BY CF1.Nume

HAVING E1.IDExplorator IN (SELECT E2.IDExplorator FROM Exploratori E2 INNER JOIN ExploratoriInProiecte EP2 ON E2.IDExplorator = EP2.IDExplorator))

# Note

Numele în România nu sunt unice, de aceea numele persoanelor sunt acompaniate de un număr astfel încât combinația nume plus număr să fie unică. O metodă similară este folosită de Microsoft pentru licențe.

Metodă similară folosită de Microsoft



Implementarea mea similară

# Bibliography

*SQLite Primary Key*. (2019). Preluat de pe SQLiteTutorial: https://www.sqlitetutorial.net/sqlite-primary-key/