Helping Hands

1. Descrierea temei

Aplicația Helping Hands are ca scop facilitarea găsirii si contactarii rapide a meseriasilor de catre clienti. Utilizatorii au acces la profilele meseriasilor, domeniile de activitate, locațiile acestora, precum și recenziile oferite de alți clienți în urma lucrărilor finalizate.

Platforma permite analiza calității serviciilor oferite prin intermediul unor rapoarte generate pe baza cererilor și ratingurilor, oferind un real suport atat pentru clienti, cât și pentru administratori.

Accentul proiectului este pus pe proiectarea corecta a bazei de date și pe separarea logicii de business la nivelul acesteia.

2. Descrierea bazei de date

Baza de date este relațională și a fost implementată utilizand Microsoft SQL Server.

2.1 Diagrama bazei de date

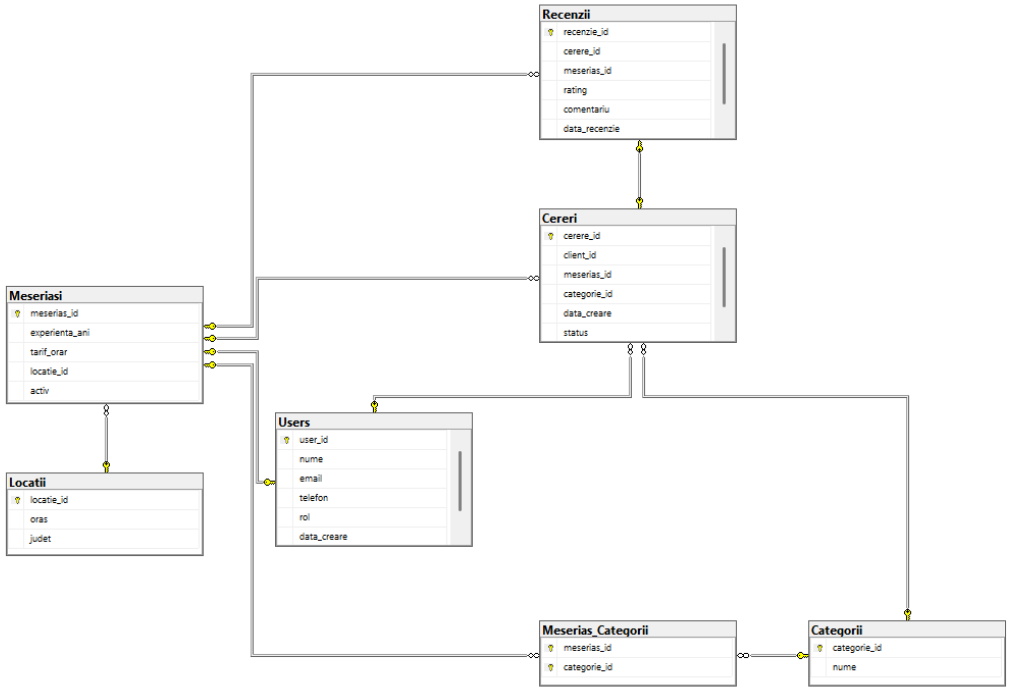


Diagrama evidențiază relațiile dintre utilizatori, meseriași, categorii, cereri și recenzii, precum și legăturile de tip one-to-many și many-to-many dintre entități.

2.2 Structura tabelelor

Users: Stochează informații despre utilizatorii aplicației. Un utilizator poate avea un rol de clienți sau de meseriaș.

Meseriaș: Conține informații specifice utilizatorilor care au rol de meseriași, precum experienta profesionala, tariful pe ora și locația.

Locații: Stochează date geografice(orar și oraș) utilizate pentru filtrarea meseriasilor.

Categorii: Reprezinta domeniile de activitate (ex. instalator, electrician).

Meserias_Categorii: Tabel de legatura care modelează relația de tip many-to-many dintre meseriași și categorii.

Cereri: Reprezinta solicitarile create de clienti catre meseriasi si permite urmarirea statusului lucrărilor.

Recenzii: Stochează evaluările oferite de clienți pentru cererile finalizate, incluzand rating și comentarii.

2.3 Descrierea constrângerilor de integritate

Integritatea datelor este asigurata prin utilizarea **cheilor primare** pentru identificarea unică a fiecărei înregistrări și a **cheilor străine** pentru definirea relațiilor dintre tabele. Acestea previn existența referințelor invalide între entități, cum ar fi asocierea unei cereri cu un meseriaș inexistent.

Sunt utilizate constrângeri de tip **CHECK** pentru validarea valorilor introduse, precum ratingul recenziilor (valori între 1 și 5) și statusul cererilor, care poate avea valori prestabilite (nouă, acceptata, finalizată).

De asemenea, sunt definite constrângeri de tip **UNIQUE** pentru campuri precum adresa de e-mail a utilizatorilor, prevenind introducerea inregistrarilor duplicate.

Relațiile de tip many-to-many sunt implementate prin **tabele de legatura**, cum ar fi cea dintre meseriași și categorii, deoarece un meseriaș poate fi activ în mai multe domenii, iar o categorie poate avea mai mulți meseriași. Relație este implementată printr-un tabel de legatura cu cheie primara compusa.

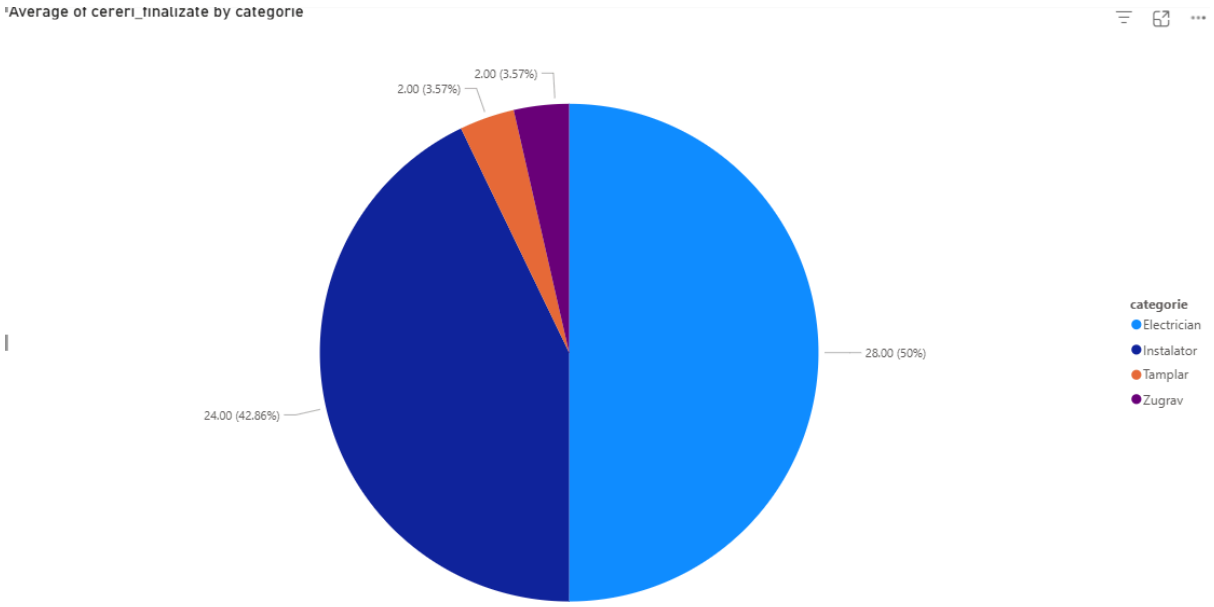
2.4 Descrierea procedurilor și funcțiilor

Pentru generarea rapoartelor și implementarea logicii de business au fost utilizate proceduri stocate, evitand accesul direct la tabele din aplicația de afișare.

sp_raport_cereri_pe_categorii

Această procedură oferă un raport privind numărul de cereri finalizate pentru fiecare categorie de servicii. Raportul este utilizat pentru a analiza cererea pieței pe tipuri de servicii.

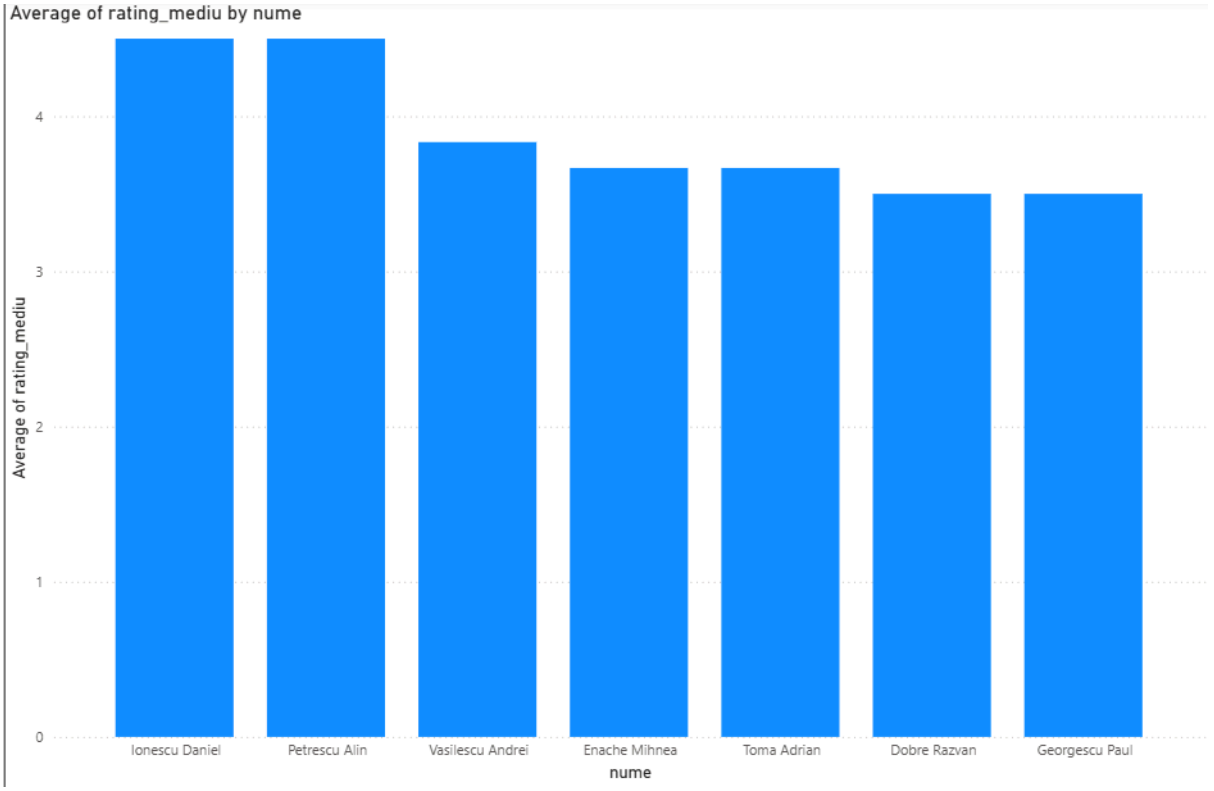
```
-- apelare procedura
EXEC sp_raport_cereri_pe_categorii;
```



sp_top_meseriasi_recomandati

Această procedură returnează lista meseriașilor recomandați, pe baza numărului de cereri finalizate, experienței și ratingului mediu.

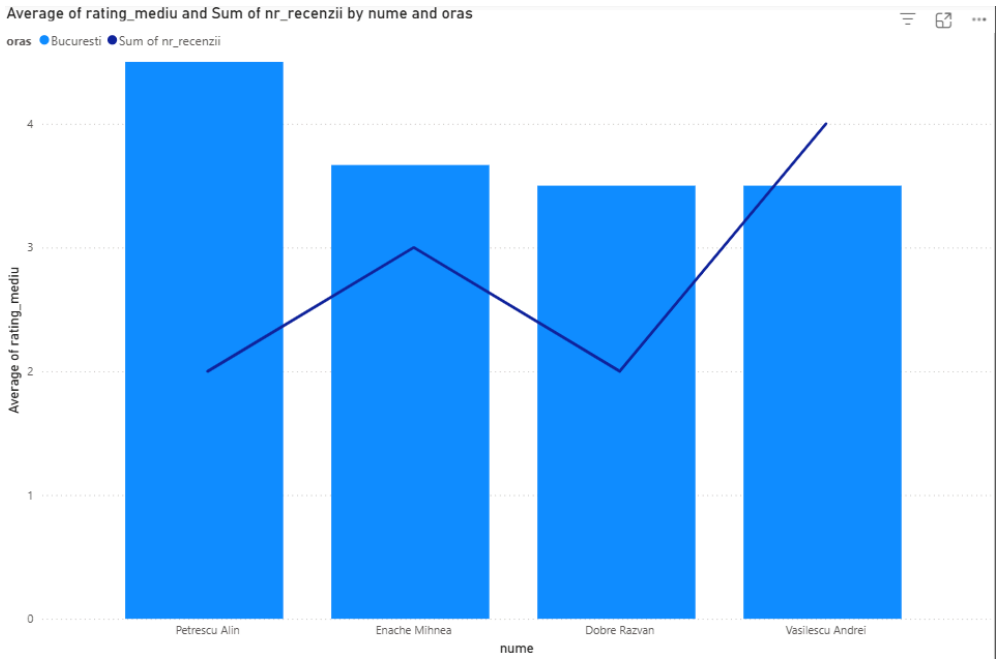
```sql
-- apelare procedura
EXEC sp_top_meseriasi_recomandati
    @min_cereri = 2,
    @rating_min = 3.5,
    @min_experienta = 3;
```



sp_meseriasi_pe_oras_categorie_rating

Această procedură permite identificarea meseriașilor activi dintr-un anumit oraș și categorie, care depășesc un prag minim de rating.Raportul este utilizat pentru recomandări avansate către clienți.

```sql
-- apelare procedura
EXEC sp_meseriasi_pe_oras_categorie_rating
    @oras = 'Bucuresti',
    @categorie = 'Electrician',
    @rating_min = 3.5;
```

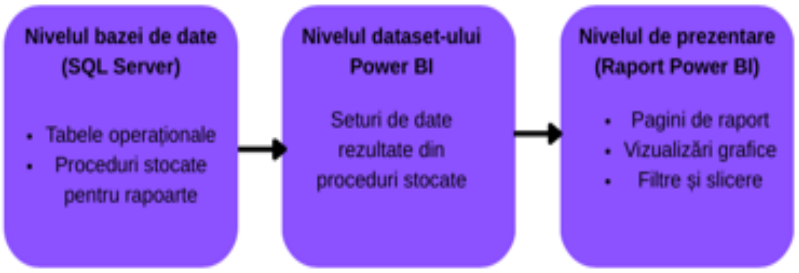


3. Descrierea aplicației

Aplicația are rolul de a afișa datele și rapoartele generate de bază de date, utilizând Power BI ca instrument de vizualizare. Logica de business este implementată exclusiv la nivelul bazei de date, prin proceduri stocate, iar aplicația este utilizata doar pentru prezentarea informațiilor.

3.1 Diagrama de clase

Diagrama de clase este adaptată la arhitectura unei aplicații de tip Business Intelligence, unde clasele sunt înlocuite de componente funcționale ale soluției Power BI.



3.2 Structura claselor

Structura claselor este reprezentată de structura dataset-ului Power BI.Dataset-ul conține următoarele tabele de raport, rezultate din proceduri stocate:

Tabele de raport:

- **RaportCereriCategorii**
  - categorie
  - cereri_finalizate

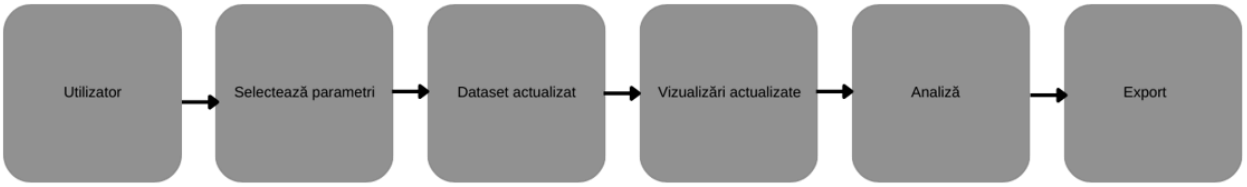
- **RaportTopMeseriasi**
  - nume
  - cereri_finalizate
  - rating_mediu

- **RaportMeseriasiOrasCategorie**
  - nume
  - oras
  - categorie
  - nr_recenzii
  - rating_mediu

3.3 Diagrama de stări și fluxul de lucru (workflow)

Fluxul de lucru al aplicației este următorul:

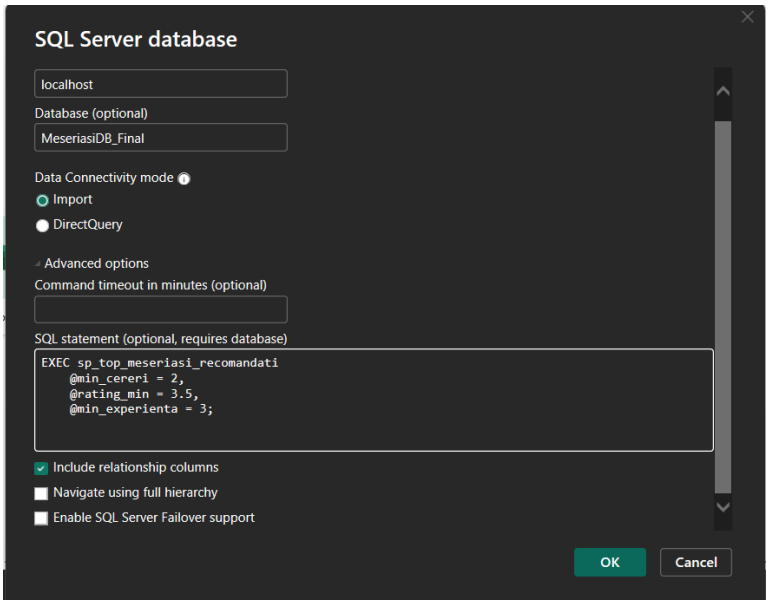
1. Utilizatorul deschide raportul Power BI.
2. Aplicația stabilește conexiunea cu baza de date SQL Server.
3. Procedurile stocate sunt executate pentru obținerea datelor.
4. Dataset-ul Power BI este actualizat.
5. Vizualizările sunt afișate pe ecran.
6. Utilizatorul aplică filtre și slicere.
7. Vizualizările sunt recalculte automat.
8. Utilizatorul analizează rezultatele.
9. Utilizatorul poate exporta datele sau rapoartele.



3.4 Prezentarea modului în care se face conexiunea cu baza de date

Aplicatia Power BI se conecteaza la baza de dare SQL Server, utilizand autentificare de tip Windows. Conectarea se realizeaza la baza de date, fără a expune tabelele către utilizator.

Datele sunt obținute exclusiv prin apelarea procedurilor stocate din baza de date, folosind opțiunea SQL statement din Power BI pentru execuția acestora. Astfel, Power BI este utilizat doar pentru afișarea datelor, fara a contine logica SQL sau procesare suplimentară.



4. Capturi de ecran pentru rapoarte

```sql
-- procedura de complexitate minim 4
CREATE PROCEDURE sp_raport_cereri_pe_categorii
AS
BEGIN
    SELECT
        c.nume AS categorie,
        COUNT(cer.cerere_id) AS cereri_finalizate
    FROM Cereri cer
    JOIN Categorii c ON cer.categorie_id = c.categorie_id
    JOIN Meseriasi m ON cer.meserias_id = m.meserias_id
    WHERE cer.status = 'finalizata'
    GROUP BY c.nume;
END;
```

```sql
-- procedura de complexitate minim 6
CREATE PROCEDURE sp_top_meseriasi_recomandati
    @min_cereri INT,
    @rating_min FLOAT,
    @min_experienta INT
AS
BEGIN
    SELECT u.nume,
           COUNT(c.cerere_id) cereri_finalizate,
           AVG(CAST(r.rating AS FLOAT)) rating_mediu
    FROM Meseriasi m
    JOIN Users u ON m.meserias_id = u.user_id
    JOIN Cereri c ON m.meserias_id = c.meserias_id AND c.status = 'finalizata'
    JOIN Recenzii r ON c.cerere_id = r.cerere_id
    WHERE m.experienta_ani >= @min_experienta
    GROUP BY u.nume
    HAVING COUNT(c.cerere_id) >= @min_cereri
       AND AVG(CAST(r.rating AS FLOAT)) >= @rating_min;
END;
```

```sql
-- procedura de complexitate minim 7
CREATE PROCEDURE sp_meseriasi_pe_oras_categorie_rating
    @oras NVARCHAR(100),
    @categorie NVARCHAR(100),
    @rating_min FLOAT
AS
BEGIN
    SELECT
        u.nume,
        l.oras,
        c.nume AS categorie,
        COUNT(r.recenzie_id) AS nr_recenzii,
        AVG(CAST(r.rating AS FLOAT)) AS rating_mediu
    FROM Meseriasi m
    JOIN Users u ON m.meserias_id = u.user_id
    JOIN Locatii l ON m.locatie_id = l.locatie_id
    JOIN Meserias_Categorii mc ON m.meserias_id = mc.meserias_id
    JOIN Categorii c ON mc.categorie_id = c.categorie_id
    JOIN Recenzii r ON m.meserias_id = r.meserias_id
    WHERE
        l.oras = @oras
        AND c.nume = @categorie
        AND m.activ = 1
    GROUP BY
        u.nume, l.oras, c.nume
    HAVING
        AVG(CAST(r.rating AS FLOAT)) >= @rating_min;
END;
```