Curse Taxi

Se dorește o aplicație cu o interfață grafică intuitiva, care sa permită gestionarea curselor unei companii de taxi.

Cerinte functionale - 7p:

1) **2p** La pornirea aplicației, se va deschide câte **o fereastră pentru fiecare Șofer**. Fiecare fereastră afișează:
   - **numele** șoferului
   - o lista cu **cursele sale active** (care aparțin șoferului și au statusul IN_PROGRESS); filtrarea curselor se va face la nivel de SQL query, în caz contrar se va aplica o depunctare de 0.5p.

Se vor defini clasele:

- **Driver**: { **id**: int, **name**: String }
- **Order**: { **id**: int, **driverId**: int, **status**: Enum{PENDING, IN_PROGRESS, FINISHED}, **startDate**: LocalDateTime, **endDate**: LocalDateTime, **pickupAddress**: String, **destinationAddress**: String, **clientName:** String

1) **1p** Șoferii își pot **marca cursele ca finalizate**.
   În dreptul fiecarei curse va apărea un buton "*Finished*". Atunci când șoferul acționează butonul, statusul cursei va deveni FINISHED, endDate-ul cursei va fi setat la data curentă, iar cursa va dispărea din lista de curse active.

2) **1p** Tot la pornirea aplicației, se va deschide **o fereastra pentru dispeceratul companiei**. Din acesta fereastra se pot **adauga curse**. Pentru adaugarea unei curse se vor introduce:
   - adresa de plecare
   - adresa destinație
   - numele clientului

   Aplicația va seta automat statusul noii curse la PENDING, si startDate-ul la data curentă.

3) **1.5p** Atunci cand se adauga o cursă, un șofer este **notificat**.
   Șoferul selectat este acela care **nu are nicio cursă activă** (IN_PROGRESS) și **are cel mai mult timp de cand a finalizat ultima sa cursă** (diferenta dintre endDate-ul maxim al curselor sale și data curentă este maxim).
   Șoferului îi apare un dialog cu mesajul "*New order: [pickupAddress] -> [destinationAddress]*" si un buton "*Accept*".
   Dacă șoferul acționează butonul "**Accept**", statusul cursei va deveni IN_PROGRESS, se va seta driverId-ul la id-ul șoferului în cauză, iar cursa va apărea în lista de curse active ale șoferului.

4) **1.5p** Dacă șoferul notificat nu accepta cursa in 5 secunde, **cursa este redistribuita** următorului șofer.
   Notificarea va dispărea din fereastra primului șofer și va apărea în fereastra următorului șofer. Dacă nici acesta nu acceptă notificarea în 5 secunde, cursa este distribuita mai departe etc.
   Se menține ordinea de alegere a șoferilor, ei fiind selectați dintre soferii fără curse active și ordonați descrescător după intervalul de timp de la ultima lor comanda finalizată.

Cerinte non-funcționale: (3p)

- Datele se citesc din baza de date **2 p** (se acordă punctaje intermediare conform cu cerințele funcționale corespunzătoare).
- Procesarea va avea loc numai la nivel de service sau de controller; interacțiunea cu sursa de date se va face numai prin intermediul repository-ului. **0.5p**
- Se cere eliminarea codului care nu este folosit precum și a funcționalităților care nu s-au cerut. **0.25p**
- Clasele, atributele și metodele lor vor avea numele cerute în problema sau nume sugestiv (dacă nu s-a specificat). **0.25p**

Important!!!

- Se punctează doar cerințele funcționale care rulează.
- Orice cod care nu poate fi explicat atrage după sine depunctarea cerințelor din care face parte.

Timp de lucru: 150 minute