Uber Evaluation

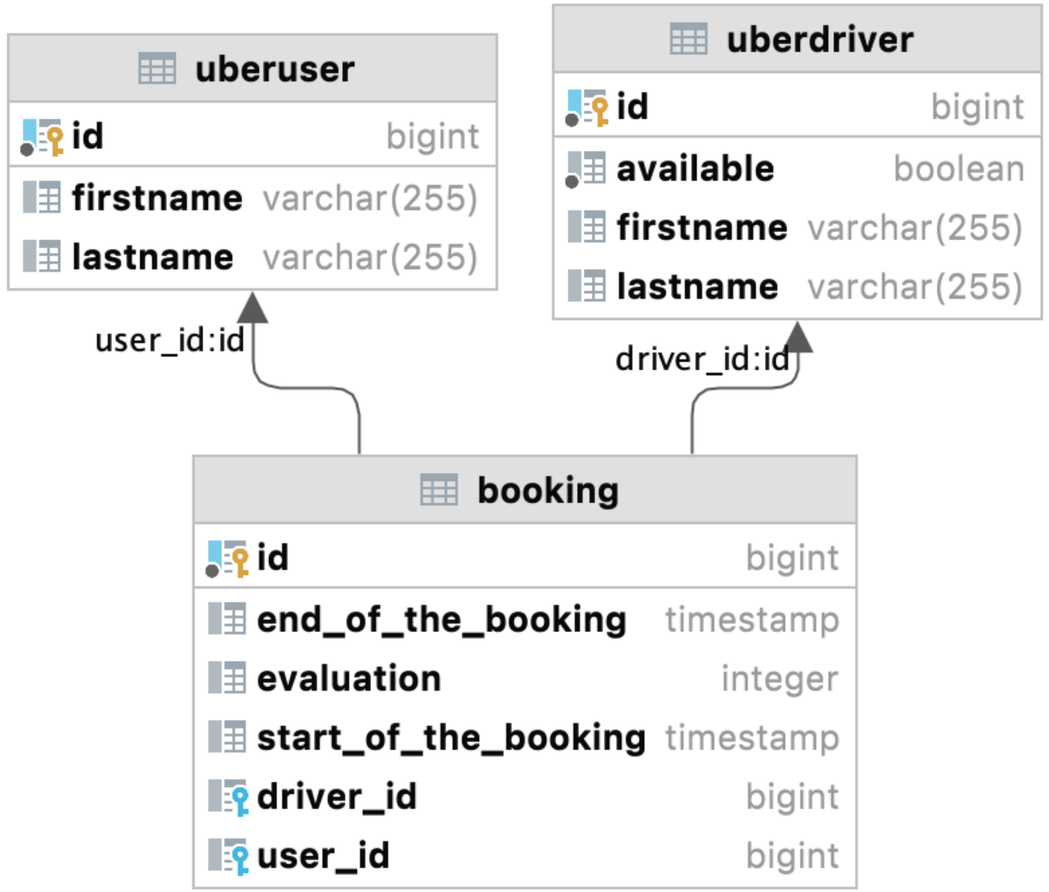
On veut faire une appli Uber-like. Dans laquelle des conducteurs et des clients peuvent s'inscrire.

1. Et donc comme dans Uber, les clients peuvent réserver un driver (un seul à la fois), ce qui rend ce driver inaccessibles aux autres.

2. A la fin d'une course, on doit pouvoir cesser la course, et le client peut alors évaluer le conducteur.

3. On peut aussi lister les courses d'un conducteur et avoir la note moyenne du conducteur.

4. L'api telle qu'exposée dans le `main` ne permet pas de contrôler les droits des utilisateurs (n'importe qui peut appeler n'importe quelle méthode). Notamment la méthode d'évaluation du conducteur et celle pour annoncer la fin d'une course ne devrait être accessible qu'au client de la course. On pourra considérer que seul l'utilisateur à accès à ses propres données personnelles (`uberUser` ou `uberUser2` dans le `main`) et que ces données pourront donc être utilisées comme un contrôle d'accès.



Main.class

```java
package com.freestack.evaluation;

import java.util.List;

public class Main {

    public static void main(String[] args) {

        UberUser uberUser = new UberUser("joe", "bah");
        UberApi.enrollUser(uberUser);

        UberUser uberUser2 = new UberUser("joe", "bee");
        UberApi.enrollUser(uberUser2);

        UberDriver uberDriver = new UberDriver("jane", "dee");
        UberApi.enrollDriver(uberDriver);

        Booking booking1 = UberApi.bookOneDriver(uberUser);
        if (booking1 == null) throw new AssertionError("uberDriver should be found available");

        Booking booking2 = UberApi.bookOneDriver(uberUser2);
        if (booking2 != null) throw new AssertionError("the only one driver is already booked, " +
            "we should not found any free");

        UberApi.finishBooking(booking1);
        int evaluationOfTheUser = 5;
        UberApi.evaluateDriver(booking1, evaluationOfTheUser);

        List<Booking> bookings = UberApi.listDriverBookings(uberDriver);
        if (bookings.size() != 2) throw new AssertionError();
        if (!bookings.get(0).getScore().equals(evaluationOfTheUser)) throw new AssertionError();

        Booking booking3 = UberApi.bookOneDriver(uberUser2);
        if (booking3 == null) throw new AssertionError("uberDriver should be now available");

        List<Booking> unfinishedBookings = UberApi.listUnfinishedBookings();
        if (unfinishedBookings.size() != 1) throw new AssertionError("only booking3 should be
        unfinished");

        float meanScore = UberApi.meanScore(uberDriver);
        if (meanScore != 5) throw new AssertionError("one eval of 5 should give a mean of 5");

    }
}
```