
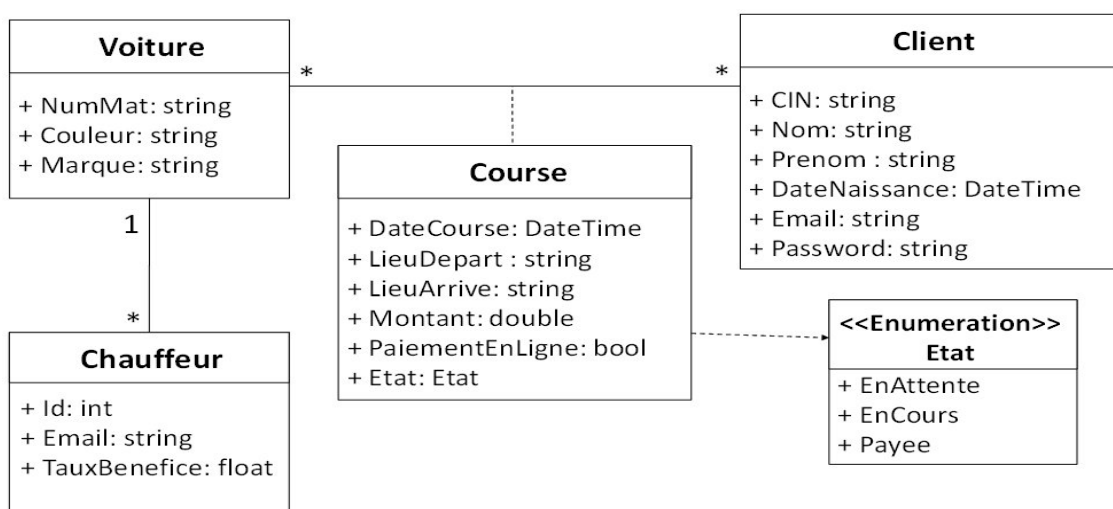


|   |   |   |
|---|---|---|
| <h1 style="margin: 0;">EXAMEN</h1>  |   | <br><small>Ecole Supérieure Privée<br/>d'Ingénierie et de Technologies</small> |
| <b>Semestre :</b> <input checked="" type="checkbox"/> 1 <input type="checkbox"/> 2<br><br><b>Session :</b> <input checked="" type="checkbox"/> Principale <input type="checkbox"/> Rattrapage |   |   |
| <b>Module :</b> Architecture des systèmes d'information I (.Net)<br><b>Enseignants :</b> Équipe .Net<br><b>Classes :</b> 4 SAE  |   |   |
| <b>Documents autorisés :</b> <input checked="" type="checkbox"/> OUI <input type="checkbox"/> NON   |   | <b>Nombre de pages :</b> 3 pages  |
| <b>Date :</b> 12/01/2022  | <b>Heure :</b> 9H   | <b>Durée :</b> 1h30m  |
| Classe : .....<br>Salle : .....   | <b>ETUDIANT(e)</b><br>N° Carte : .....<br>Nom et Prénom : ..... |   |

**NB :** -Toute carte Wifi active est une tentative de fraude  
 -Respectez l'architecture vue en cours et les patrons de conception étudiés.

## Enoncé du problème :

On désire implémenter une application Web de gestion d'Uber, à l'aide du Framework .Net Core. Pour cela, on a analysé le besoin des classes présentées dans le diagramme UML suivant :



## Partie I: Entity-Framework (10 pts)

1. Créer les entités en respectant le diagramme de classes. (3 pts)
2. En utilisant les annotations :
  - a. Configurer la propriété **CIN** de la classe **Client** en tant que clé primaire et de même pour la propriété **NumMat** de la classe **Voiture**. (1 pt)
  - b. Configurer la propriété **Email** de la classe **Client** en tant que propriété obligatoire de type adresse email. (1 pt)
3. En utilisant Fluent API :
  - a. Créer une classe de configuration afin de configurer la classe **Course** comme porteuse de données entre **Voiture** et **Client** avec une clé primaire composée par trois propriétés **VoitureId**, **ClientId** et **DateCourse**. (2 pts)
  - b. Configurer toutes les propriétés de type **string** de sorte qu'elles ne dépassent pas 150 caractères. (1 pt)
4. En utilisant la migration, créer la base de données nommée sous la forme NomPrenomDB. (2 pts)

## Partie II: Services (4 pts)

Dans les classes de services spécifiques, créer:

5. Une méthode qui retourne la voiture la plus demandée. (1pt)
6. Une méthode qui retourne le bénéfice total d'un chauffeur des courses payées à une date donnée, en utilisant la propriété **Date** du type **DateTime**. (1pt)
7. Une méthode qui retourne la liste des courses payées d'un chauffeur à une date donnée, en utilisant la propriété **Date** du type **DateTime**. (2 pts)

## Partie III: ASP MVC (6 pts)

8. Alimenter manuellement la base de données par 2 voitures, 2 clients et 2 chauffeurs. (0.5 pt)
9. Créer et tester la vue "Create" qui permet d'ajouter une course dans la base de données. (3.5 pts)

## Create

### Course

DateCourse

LieuDepart

LieuArrive

Montant

☒ PaiementEnLigne

Etat

VoitureId

ClientId

Create

10. Créer et tester une vue qui permet d'afficher les courses payées d'un chauffeur à la date d'aujourd'hui, en utilisant les services déjà implémentés dans la partie **II**. (2 pts). **NB.** L'identifiant du chauffeur est passé en paramètre dans l'URL (*localhost:port/Course/ListByChauffeur/2*).

| Liste des courses   |            |            |         |                                     |       |                 |          |
|---------------------|------------|------------|---------|-------------------------------------|-------|-----------------|----------|
| DateCourse          | LieuDepart | LieuArrive | Montant | PaiementEnLigne                     | Etat  | VoitureId       | ClientId |
| 20/12/2021 13:00:00 | Tunis      | Sousse     | 20      | <input checked="" type="checkbox"/> | Payee | 455890 Tunis 90 | 78787878 |