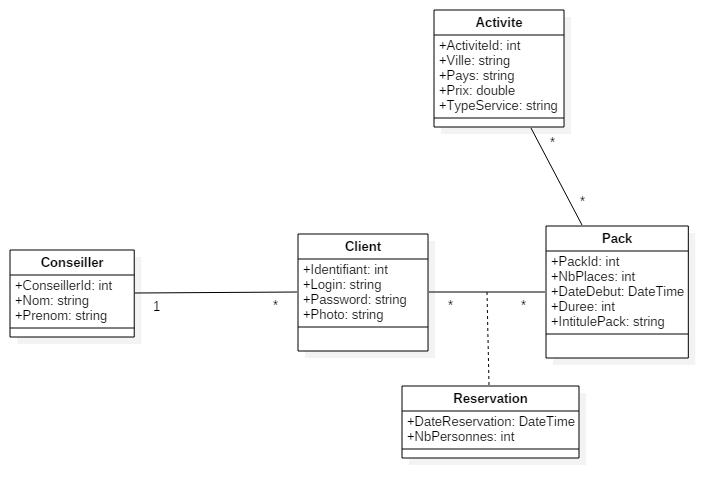
| EXAMEN  **Semestre** : 🗹 1 ⃞ 2  **Session** : 🗹 Principale ⃞ Rattrapage | |  |
| --- | --- | --- |
| **Module** : Architecture des systèmes d’information I (.Net)  **Enseignants** : Équipe .Net  **Classes** : 4ERP-BI | | |
| **Documents** **autorisés**  : 🗹 OUI ⃞ NON **Nombre** **de** **pages** : 4 pages | | |
| **Date** : 06/01/2022 **Heure** : 9h **Durée** : 1h30m | | |
| Classe : ………………  Salle :……………… | **ETUDIANT(e)**  N° Carte : …………………………………………………………...  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 :**

Une agence de voyage souhaite réaliser une application web pour la gestion de ses activités. Soit le diagramme de classes suivant :

****

Il est à noter que :

**NbPersonnes** : Le nombre de personnes pour lesquelles le client a fait la réservation.

**NbPlaces**: Le nombre de places maximum du pack

**Durée** : La durée en jours du pack.

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

1. Implémenter le diagramme de classes ci-dessus. **(2.5 pts)**

2. En utilisant les **annotations:**

Dans la classe **Reservation (1pt)**:

1. La propriété **DateReservation** doit être une date valide.
2. La propriété **NbPersonnes** doit appartenir à l’intervalle [1,4].

Dans la classe **Client (1.5pts)**:

1. La propriété **Identifiant** doit être la clé primaire de la classe.
2. La propriété **Login** doit être obligatoire avec affichage d’un message d’erreur si elle n’est pas saisie par l’utilisateur.
3. La propriété **Password** doit être obligatoire et la valeur saisie par l’utilisateur doit être cachée.

En utilisant les configurations **FluentAPI :**

Sans classe de configuration

3. Définir le type détenu **Zone**qui regroupe les propriétés **Ville** et **Pays** de la classe **Activité (0.5 pt)**

4. Faites le nécessaire pour représenter la relation entre **Client**, **Pack** et **Reservation**. **(1 pt)**

Avec classe de configuration

5.Configurer la relation 1..\* entre **Client** et **Conseiller** en précisant **ConseillerFK** comme clé étrangère et en activant la Cascade On Delete. **(1pt)**

**6.** Mapper toutes les propriétés de type String dans des colonnes de longueur maximale de 15 caractères. **(1pts)**

7. Utiliser les migrations pour générer la base de données nommée sous la forme NomPrenomBD. **(1.5pts)**

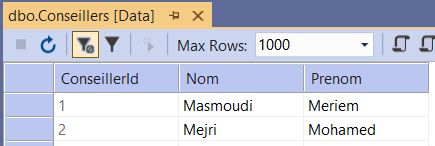
**Partie II: Services (4pts)**

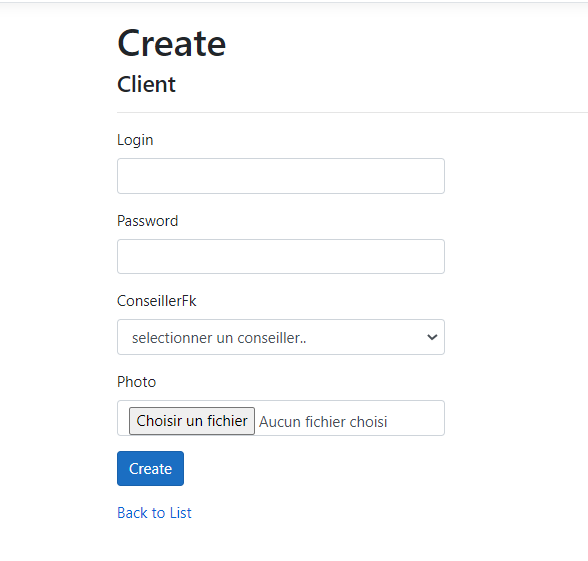
Dans la couche **Service** créer les interfaces et les classes nécessaires pour implémenter les méthodes suivantes :

1. Retourner le montant total des paiements d’un client passé en paramètre. **(1 pt)**
2. Retourner le nombre de réservations d’un client passé en paramètre pendant l’année en cours. **(1 pt)**
3. Calculer le prix total d’un pack. **(1 pt)**
4. Retourner le pourcentage des packs d’une longue durée (durée supérieure à une semaine). **(1 pt)**

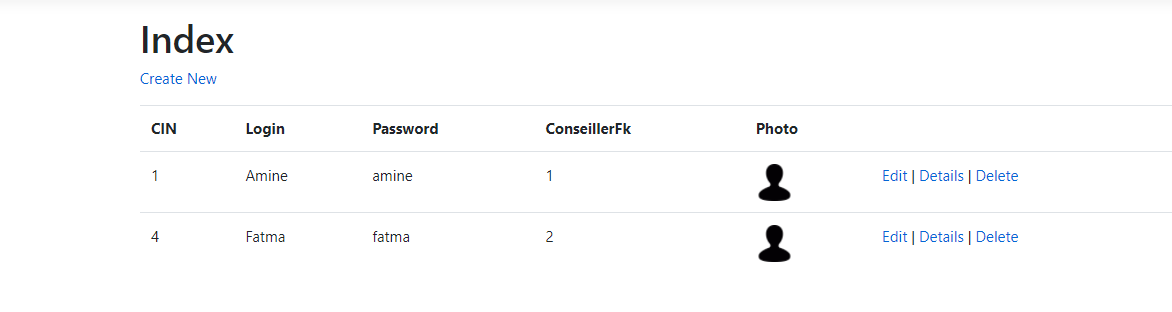
**Partie III: ASP MVC (6pts)**

1-Alimenter la BDD comme montre la figure ci-dessous: **(0.5 pt)**

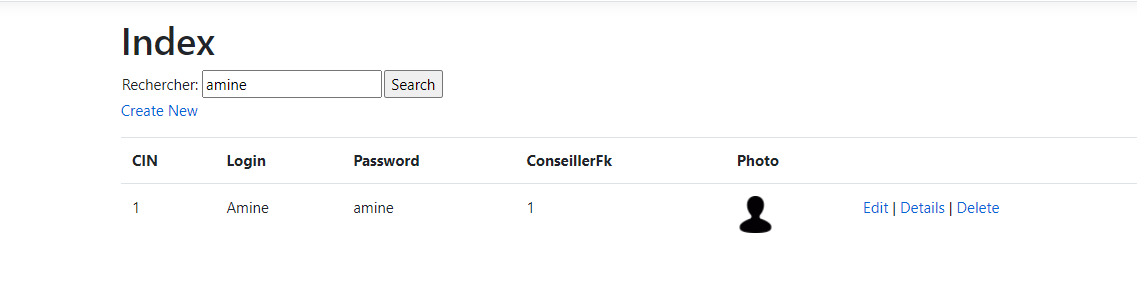
* Deux Conseillers 

2- Réaliser une vue qui permet de créer un **Client** comme le montre la figure suivante, tout en choisissant le **Conseiller** à partir d’une liste déroulante .**(2 pts)**

3- Rediriger vers une vue **Index** qui liste l’ensemble des clients.**(1 pt)**



4- Ajouter un bloc de recherche à la vue **Index**, qui permet de filtrer les clients par Login.**(1pt)**



Bon Travail ☺

5- Ajouter un lien “**Conseiller Client”** à la vue **Index**, qui doit diriger vers une vue **Details** qui affiche les détails relatifs à chaque Conseiller affecté au client. **(1.5pts)**

