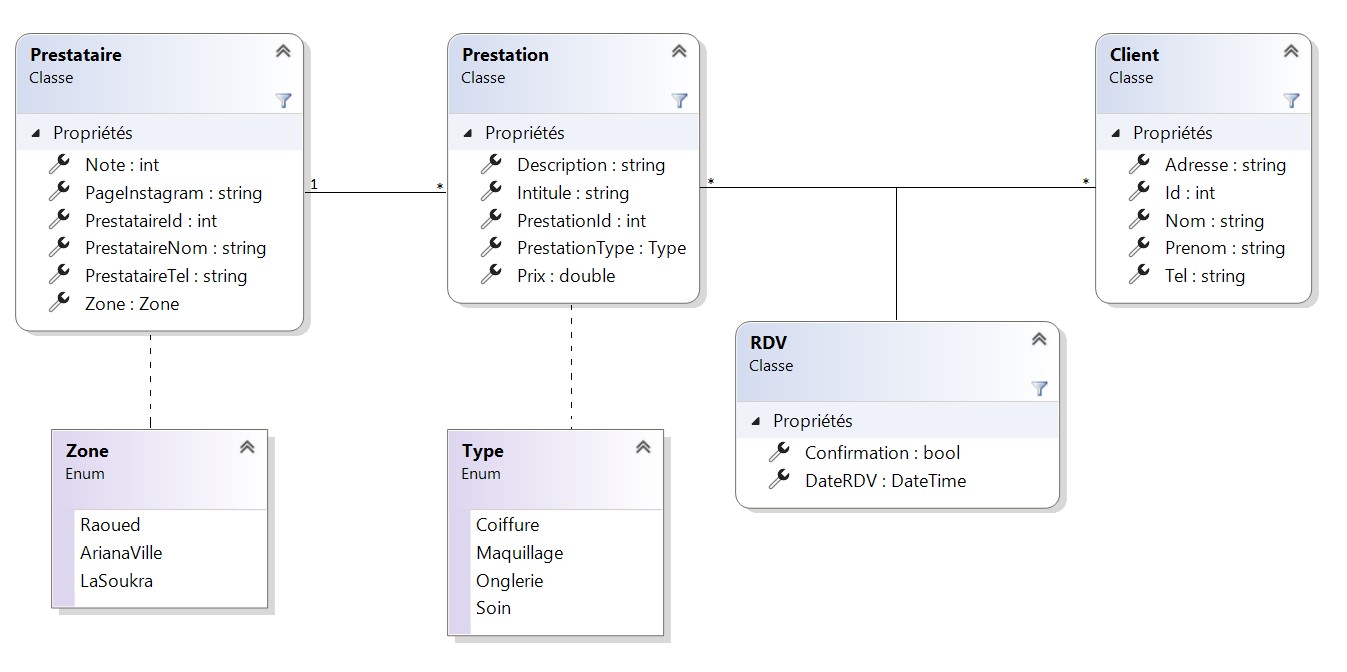
|  |  |  |
| --- | --- | --- |
|  | EXAMEN  **Semestre** : ⃞ 1  2  **Session** :  Principale ⃞ Rattrapage | |
| **Module** : Architecture des systèmes d’information I (.Net)  **Enseignants** : Équipe .Net **Classes** : 4 TWIN  **Documents autorisés** :  OUI ⃞ NON **Nombre de pages** : 4 pages  **Date** : 18/05/2023 **Heure** : 9h **Durée** : 1h30m | | |
| **ETUDIANT(e)**  N° Carte : …………………………………………………………...  Nom et Prénom :……………………………………………… .. | | Classe : ……………… Salle :……………… |

**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 simplifiée de mise en relation entre les clients et les prestataires de services de beauté à domicile, à l'aide du Framework .Net 6 en suivant le diagramme de classes ci-dessous :



# Partie I: Entity-Framework (10 pts)

1. Créer les entités et les relations entre elles en respectant le diagramme de classes. **(3**

## pts)

2. En utilisant les **annotations** :

1. Configurer la propriété **Note** de la classe **Prestataire** afin qu’elle ait une valeur entre 0 et 5. **(0.5 pt)**

[Range(0, 5)]

public int Note { get; set; }

1. Configurer la propriété **Description** de la classe **Prestation** afin qu’elle soit une zone de texte sur plusieurs lignes. **(0.5 pt)**

[DataType(DataType.MultilineText)]

public string Desciption { get; set; }

1. Configurer la propriété **Prix** de la classe **Prestation** afin qu’elle soit une valeur

monétaire. **(0.5 pt)**

[DataType(DataType.Currency)]

public double prix { get; set; }

1. Configurer la relation entre **Prestation** et **Prestataire** afin que la clé étrangère de cette relation soit mappée vers une colonne nommée **PrestataireFK**. **(0.5**

## pt)

[ForeignKey("Prestataire")]

public int PrestataireFK { get; set; }

1. En utilisant **les pré-conventions** :
   1. Configurer toutes les propriétés de type **string** pour qu’elles soient mappées vers des colonnes de longueur maximale égale à 150 caractères. **(0.5 pt)**

protected override void ConfigureConventions(ModelConfigurationBuilder configurationBuilder)

{

configurationBuilder.Properties<string>().HaveMaxLength(150);

base.ConfigureConventions(configurationBuilder);

}

1. Créer une classe de **configuration Fluent API** dans laquelle vous devez : **(0.5 pt)**
   1. Configurer la relation entre **Prestation**, **RDV** et **Client** afin que les clés étrangères de cette relation soient mappées vers des colonnes nommées

## PrestationFK et ClientFK. (1 pt)

Créer Dossier Configuration et la classe de configuration suivante :

public class RDVConfiguration : IEntityTypeConfiguration<RDV>

{

public void Configure(EntityTypeBuilder<RDV> builder)

{

builder

.HasOne(r => r.Client)

.WithMany(p => p.RDV)

.HasForeignKey(r => r.ClientFK);

builder

.HasOne(r => r.Prestation)

.WithMany(f => f.RDV)

.HasForeignKey(r => r.PrestationFK);

b. Configurer la clé primaire de la classe **RDV** composée par les trois propriétés **PrestationFK**, **ClientFK** et **DateRDV**. **(1 pt)**

//Pour la configuration complete de la classe Expertise il faut ajouter la clé primaire

builder.HasKey(r => new

{

r.ClientFK,

r.PrestationFK,

r.DateRDV

});

5. Faire le nécessaire pour générer la base de données dont le nom doit être sous la forme **BeautyNomPrenom**. **(2 pts)**

***1-Ajouter les Db :***

public DbSet<Client> Client { get; set; }

public DbSet<Prestation> Prestation { get; set; }

public DbSet<Prestataire> Prestataire { get; set; }

public DbSet<RDV> RDV { get; set; }

***2-Le nom de la base:***

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer(@"Data Source=(localdb)\mssqllocaldb;

Initial Catalog=Beautydina;Integrated Security=true;

MultipleActiveResultSets=true");

optionsBuilder.UseLazyLoadingProxies();

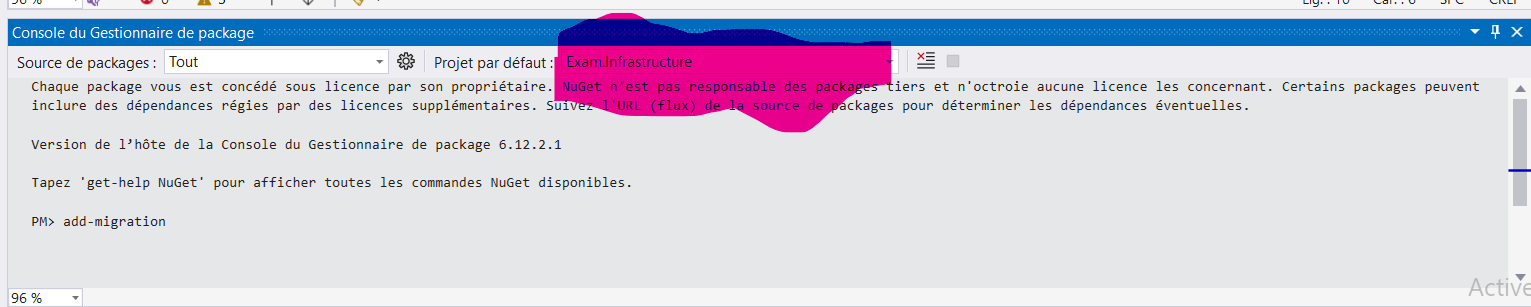
base.OnConfiguring(optionsBuilder);

}

***3-Ajouter les configuration :***

modelBuilder.ApplyConfiguration(new RDVConfiguration());

*4-Migration :*

*5-Update-database*

# Partie II: Services (4 pts)

Créer les interfaces et les classes nécessaires qui comportent des méthodes permettant de :

1. Retourner les 3 prestataires les mieux notés dans la zone de Raoued. **(1.25 pt)**

**1-Créer l’interface et le service de prestataires :**

public IEnumerable<Prestataire> GetTop3PrestatairesInRaoued()

{

return GetAll()

.Where(p => p.PrestataireNom != null && p.PrestataireId != 0) // Filtrer pour exclure les prestataires vides ou non initialisés

.Where(p => p.PrestataireNom.Contains("Raoued")) // Filtrer par zone "Raoued"

.OrderByDescending(p => p.Note) // Trier par note décroissante

.Take(3) // Prendre les 3 meilleurs

.ToList();

}

1. Retourner le prix moyen de la prestation de type Coiffure. **(1.25 pt)**

**1-Créer l’interface et le service de Prestation :**

public decimal GetAveragePriceForCoiffure()

{

return GetAll()

.Where(p => p.PrestationType == TypeP.Coiffure) // Filtrer par type "Coiffure"

.Average(p => p.Prix); // Calculer la moyenne des prix

}

1. Retourner la liste des rendez-vous confirmés d’un client donné, groupés par date. (**1.5**

**pt)**

**dans le service de rdv :**

public IEnumerable<IGrouping<DateTime, RDV>> GetConfirmedRendezvousByClient(int clientId)

{

return GetAll()

.Where(r => r.ClientId == clientId && r.Status == "Confirmé") // Filtrer par client et statut confirmé

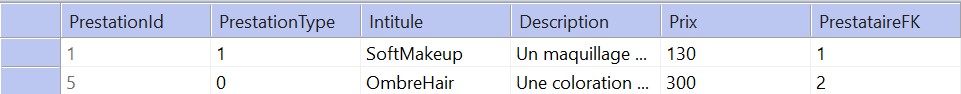
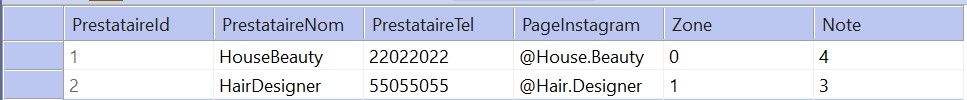
.GroupBy(r => r.DateRendezvous.Date) // Grouper par date

.ToList();

}

# Partie III: ASP MVC (6 pts)

1. Alimenter la base de données par 2 prestataires et 2 prestations. **(1 pt)**



1. Implémenter une page Index qui permet d’afficher la liste des prestataires ordonnés par nom. **(1 pt)**

**Dans le controller de prestataires  (voir « doc\_aut/Creation\_Controller » pour créer le controller) :**

***1-Ajouter :***

readonly IService<Prestataire> PrestataireService;

public PrestataireController(IService<Prestataire> PrestataireService)

{

this.PrestataireService = PrestataireService;

}

***2-Créer la page index:***

Clique droite sur la fonction index ajouter->créer une vueet choisir list puis prestataire

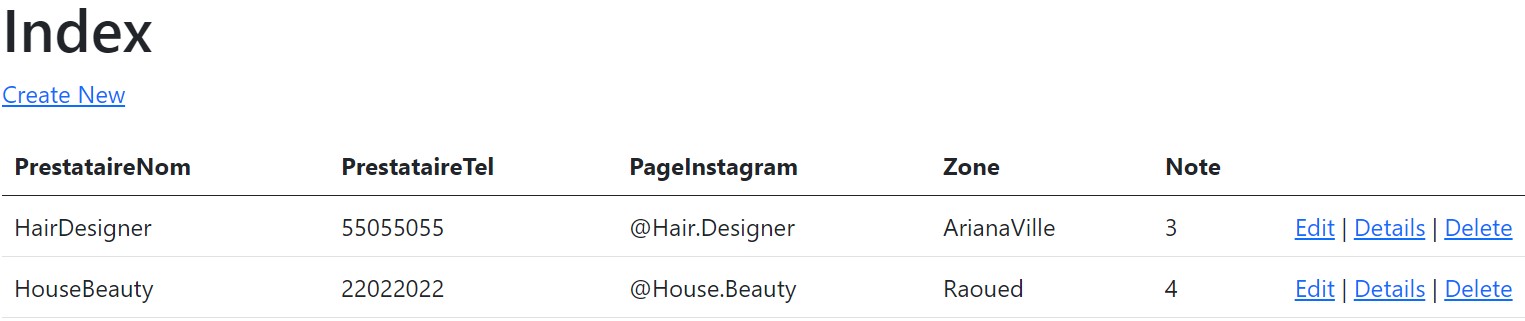
***3-Ajouter :***

public ActionResult Index()

{

return View(PrestataireService.GetAll().OrderBy(h => h.PrestataireNom));

}



1. Implémenter une page Create qui permet d'insérer dans la base de données une prestation comme le montre la figure ci-dessous . **(2 pts)**



1-Créer le controller de praistataires puis :

readonly IService<Prestation> PrestationService;

public PrestationController(IService<Prestation> PrestationService)

{

this.PrestationService = PrestationService;

}

***2- Clique droite sur la fonction Create ajouter->créer une vueet choisir Create puis prestation***

***3-Ajouter dans create partie post :***

public ActionResult Create(IFormCollection collection, Prestation p)

{

try

{

PrestationService.Add(p);

PrestationService.Commit();

return RedirectToAction(nameof(Index));

}

catch

{

return View();

}

}

***4-N’oublie pas l’injection dans programme.cs :***

builder.Services.AddScoped<IService<Prestation>, PrestationService>();

***5-Dans le html :***

***Supprimer l’id et remplacer l’enumeration par:***

<div class="form-group">

<**label** **asp-for**="PrestationType" class="control-label"></**label**>

<**select** **asp-for**="PrestationType" class="form-control"

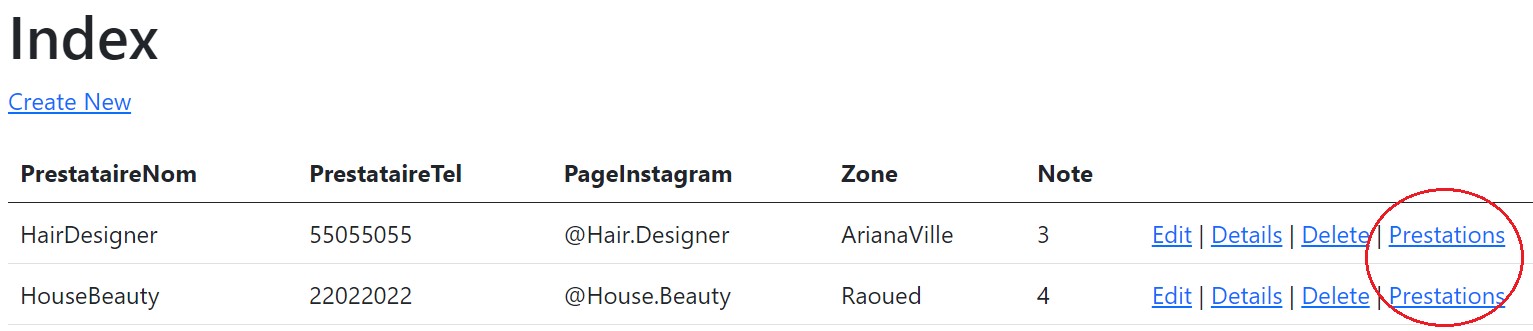
**asp-items**="Html.GetEnumSelectList<ApplicationCore.Domain.TypeP>()"></**select**>

<**span** **asp-validation-for**="PrestationType" class="text-danger"></**span**>

</div>

1. Dans la page Index des prestataires, ajouter un lien Prestations qui permet la redirection vers la liste des prestations associée à chaque prestataire. **(2 pts)**

1-Dans PrestataireController :



readonly IService<Prestataire> PrestataireService;

readonly IService<Prestation> PrestationService;

public PrestataireController(IService<Prestataire> PrestataireService, IService<Prestation> PrestationService)

{

this.PrestataireService = PrestataireService;

this.PrestationService = PrestationService ; }

2-Ajouter la fonction :

public ActionResult Prestations(int prestataireId)

{

// Récupération des prestations associées au prestataire

var prestations = PrestationService.GetAll().Where(p => p.PrestataireFK == prestataireId).ToList();

// Passer les données à la vue

return View(prestations);

}

3-Clique droite sur cette fonction est créer une vue

4-Ajouter ce lien dans View/Prestataires/Index. cshtml:

@Html.ActionLink("Prestations", "Prestations", new { prestataireId = item.PrestataireId })

