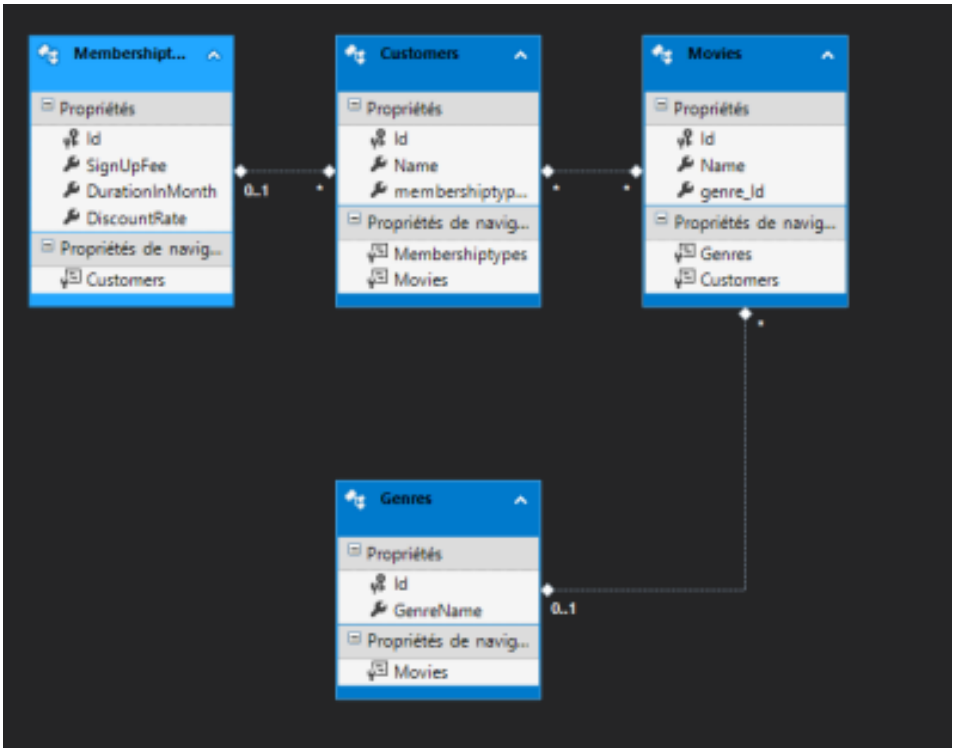
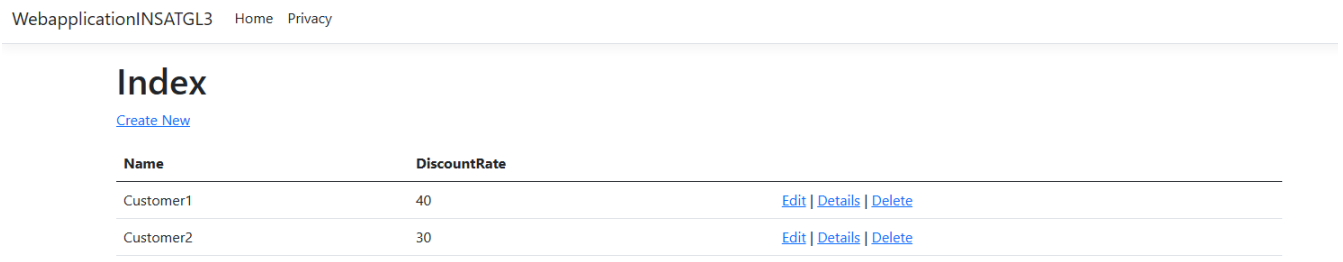
TP3 : EF Core Relations, Seed Data

A. EF Core relations

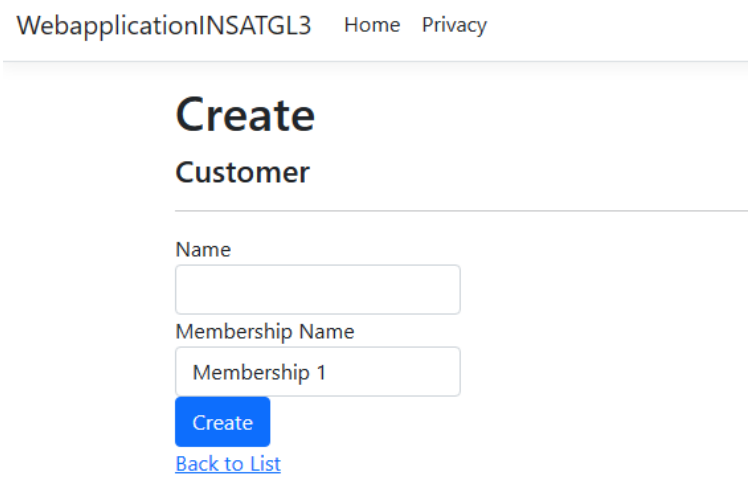
1. Considérer le modèle suivant pour définir votre schéma relationnel



2. Ecrire le code nécessaire permettant d'afficher la vue suivante



3. Ajouter un nouveau client en se référant au formulaire suivant.



//Commentez et expliquez le code suivant

```
public IActionResult Create()
        {
            var members = _db.membershiptypes.ToList();
            ViewBag.member = members.Select(members => new SelectListItem()
            {
                Text = members.Name,
                Value = members.Id.ToString()
            });
            return View();


        }
        [HttpPost]
        public IActionResult Create(Customer c)
        {
            if(!ModelState.IsValid)
            {
                var members = _db.membershiptypes.ToList();
                ViewBag.member = members.Select(members => new SelectListItem()
                {
                    Text = members.Name,
                    Value = members.Id.ToString()
                });
                return View();
            }
            c.CustomerId = new Guid();
            _db.customers.Add(c);
            _db.SaveChanges();
            return RedirectToAction(nameof(Index));
        }
```

```
//Pour la Vue
<div class="form-group">
            <label asp-for="MembershiptypeID" class="control-label">Membership Name</label>
            <select asp-for="MembershiptypeID" class ="form-control" asp-
items="@ViewBag.member"></select>
            <span  asp-validation-for="MembershiptypeID"  class="text-red"></span>
            </div>
```

4. Ajout du contrôle des erreurs du ModelState via ViewBag

ModelState encapsule les erreurs qui peuvent venir à partir de la liaison du modèle et de la validation du Modèle.

ModelState. IsValid renvoie true si le modèle reçu en paramètre de l'action est valide à savoir que les règles de validations appliquées en utilisant les attributs d'annotations de données.

Ajouter ce code au niveau du contrôleur et vue. Expliquez.

```
ViewBag.Errors = ModelState.Values
        .SelectMany(v => v.Errors).Select(e => e.ErrorMessage).ToList();
//View, s'il y a des erreurs, on va les afficher
@if (ViewBag.Errors != null)
{
<div class="text-red ml">
<ul>
@foreach (string error in ViewBag.Errors)
{
<li class="ml">@error</li>
}
</ul>
</div>
    }
```

B. Seed DATA

//Seed DATA au niveau du OnModelCreating

1. Ajouter un fichier JSOn contenant les enregistrements relatifs à la Table Genre de la base de données

```
[
  {
    "Id": "84ca0bcd-082c-49cb-aa77-ea2f1f5f8285",
    "Name": "GenreFromJsonFile1"

  },
  {
    "Id": "79e6f638-d7e7-4f63-8365-f172cb925381",
    "Name": "GenreFromJsonFile2"

  }
]
```

2. Faire les modifications du OnModelCreating pour ajouter les enregistrements à la table Genre de la base de données

```
protected override void OnModelCreating(ModelBuilder model)
        {
            base.OnModelCreating(model);
            string GenreJSon = System.IO.File.ReadAllText("Genres.Json");
            List<Genre>? genres = System.Text.Json.
            JsonSerializer.Deserialize<List<Genre>>(GenreJSon);
            //Seed to categorie
            foreach (Genre c in genres)
                model.Entity<Genre>()
                 .HasData(c);

        }
```

→ **JsonSerializer.Deserialize** Fournit les fonctionnalités permettant de sérialiser des objets ou des types valeur en JSON et de désérialiser JSON en objets ou types valeur.

Deserialize(JsonDocument, Type, JsonSerializerContext)	Convertit la JsonDocument valeur JSON représentant une seule valeur JSON en un `returnType`.

C’est à vous :

- Ecrire le code permettant d’ajouter un Film selon le comportement suivant
  Pensez à ajouter des attributs à la classe d’entité « Movie »