# TP1 : JuliePro

## Consignes (15% de la note finale)[​](https://info.cegepmontpetit.ca/BW5-Web-Transactionelle/tp_Regulier/tp1#consignes-15-de-la-note-finale)

* Lisez toutes les instructions et la grille de correction avant de commencer
* Créez un Repository PRIVÉ et ajoutez votre enseignante comme collaborateur
* Vous **DEVEZ** faire au moins les migrations et les commits demandés mais vous pouvez en faire plus sans problème, tant que vous les documentez correctement

## Étude de cas JuliePro[​](https://info.cegepmontpetit.ca/BW5-Web-Transactionelle/tp_Regulier/tp1" \l "%C3%A9tude-de-cas-juliepro" \o "Lien direct vers Étude de cas JuliePro)

Julie Loiselle est propriétaire de l’entreprise d’entrainement personnel JuliePro. L’entreprise emploi une douzaine d’entraîneurs (**Trainer**) chevronnés et spécialisés (**Speciality** : perte de poids, althérophilie, course, réabilitation, etc). Un **Trainer** a une seule **Speciality**. Chaque client (**Customer**) est assigné à un **Trainer**. Le **Trainer** définit également avec son client un objectif courant (**Objective**) : soit de perte de poids ou de distance. Lorsqu’un **Objective** est atteint, on indique la date.

## Les fonctionnalités à implémenter[​](https://info.cegepmontpetit.ca/BW5-Web-Transactionelle/tp_Regulier/tp1#les-fonctionnalit%C3%A9s-%C3%A0-impl%C3%A9menter)

* Gestion des spécialités (**Speciality**) avec seed🌱
* Gestion des entraîneurs (**Trainer**) et de leur spécialité (**Speciality**) avec seed🌱
* Implémentation des règles d'affaire spécifiques
* Utilisation de **vues partielles**
* Utilisation de **FontAwesome**
* Mise en place du modèle de donnée pour les clients (**Customer**) et de leurs objectifs (**Objective**)

## Création du projet[​](https://info.cegepmontpetit.ca/BW5-Web-Transactionelle/tp_Regulier/tp1#cr%C3%A9ation-du-projet)

1. Créez le Repository PRIVÉ **3W6\_TP\_NOM\_PRENOM** dans GitHub et ajoutez votre enseignant comme collaborateur
2. Créer une branche à partir de **Main** et appelée la **TP1**.

ATTENTION

Ne pas travailler directement dans la branche **Main**. Vous devez ajouter une fonctionnalité à la fois.

1. Créez un nouveau projet MVC qui se nomme **JuliePro**
2. Dans le projet MVC:

* Il faut télécharger ce [fichier zip](https://info.cegepmontpetit.ca/BW5-Web-Transactionelle/assets/files/TP1_FichiersComplementaires-d4fe4d30f5e2e26c28b55bc339146b00.zip) qui contient les images utilisées dans le projet et extraire son contenu dans le répertoire **/wwwroot/**de votre projet ()

ASTUCE

Dites oui pour remplacer **bootstrap.css** et **bootstrap.min.css** (on utilise la version **5.1.1**)

* Remplacez le contenu du fichier de Layout par ce qui suit:

/Views/Shared/\_Layout.cshtml

\_Layout.cshtml

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="utf-8" />  
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
 <title>@ViewData["Title"] - JuliePro</title>  
 <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />  
 <link rel="stylesheet" href="~/css/site.css" />  
 <style>  
 .accordion-item { border: 1px solid rgba(0,0,0,.125) }  
 </style>  
</head>  
<body>  
 <header>  
 <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-dark bg-primary border-bottom box-shadow mb-3">  
 <div class="container">  
 <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index"><img src="~/images/imagesLayout/tile.png" style="width:80%" /></a>  
  
 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls="navbarSupportedContent"  
 aria-expanded="false" aria-label="Toggle navigation">  
 <span class="navbar-toggler-icon"></span>  
 </button>  
 <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">  
 <ul class="navbar-nav flex-grow-1">  
 <li class="nav-item">  
 <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>  
 </li>  
 <li class="nav-item dropdown">  
 <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">  
 Admin  
 </a>  
 <ul class="dropdown-menu" aria-labelledby="navbarDropdown">  
 <li><a class="dropdown-item">Trainers</a></li>  
 <li><a class="dropdown-item">Specialities</a></li>  
 </ul>  
 </li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
 </header>  
 <div class="container">  
 <main role="main" class="pb-3">  
 @RenderBody()  
 </main>  
 </div>  
  
 <footer class="border-top footer text-muted bg-primary">  
 <div class="container">  
 &copy; 2023 - JuliePro - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>  
 </div>  
 </footer>  
 <script src="~/lib/jquery/dist/jquery.min.js"></script>  
 <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>  
 <script src="~/js/site.js" asp-append-version="true"></script>  
 @await RenderSectionAsync("Scripts", required: false)  
</body>  
</html>

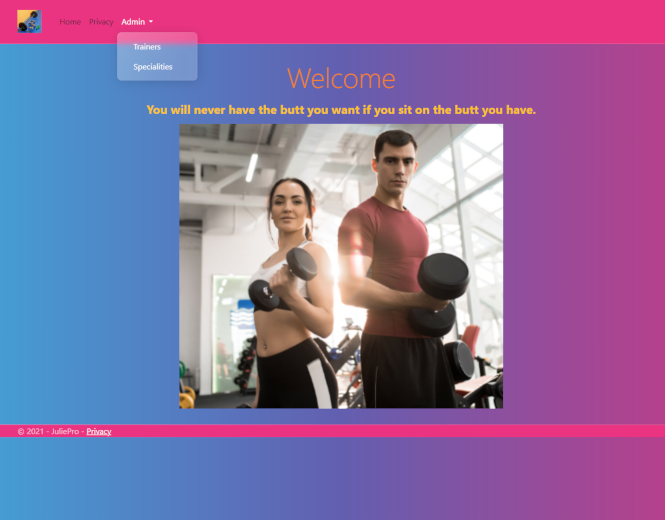
* Remplacez le contenu du fichier d'Index par ce qui suit:

/Views/Home/Index.cshtml

Index.cshtml

@{  
 ViewData["Title"] = "Home Page";  
}  
  
<div class="text-center">  
 <h1 class="display-4 text-danger">Welcome</h1>  
 <h4 class="text-warning">You will never have the butt you want if you sit on the butt you have.</h4>  
  
 <a asp-area="" asp-controller="Home" asp-action="Index"><img src="~/images/imagesLayout/JulioProImage.png" style="width:50%" /></a>  
</div>

* Assurez-vous que la page d'accueil de votre application est identique à celle-ci:



ATTENTION

C'est le moment de **tester**, ajouter des **commentaires** et faire votre **commit** et votre **push**

## Gestion des spécialités[​](https://info.cegepmontpetit.ca/BW5-Web-Transactionelle/tp_Regulier/tp1#gestion-des-sp%C3%A9cialit%C3%A9s)

1. Ajout de la classe **Speciality** qui sera gérer par Entity Framework Core(**EFC**) dans le répertoire **/Models/ (A VOIR DANS LE COURS 3 : créer dbcontext , liaison et plus.)**



Speciality

public string Name { get; set; }

INFO

Il faut ajouter les champs nécessaires pour Entity Framework Core (**EFC**) et les annotations. Référez-vous au diagramme plus haut qui vous montre les champs de la classe **Speciality**.

1. Génération du contrôleur MVC pour la classe **Speciality**
2. Ajouter un seed🌱 pour les spécialités

Seed des spécialités

builder.Entity<Speciality>().HasData(new Speciality() { Id = 1, Name = "Perte de poids" });  
builder.Entity<Speciality>().HasData(new Speciality() { Id = 2, Name = "Course" });  
builder.Entity<Speciality>().HasData(new Speciality() { Id = 3, Name = "Halthérophilie" });  
builder.Entity<Speciality>().HasData(new Speciality() { Id = 4, Name = "Réhabilitation" });

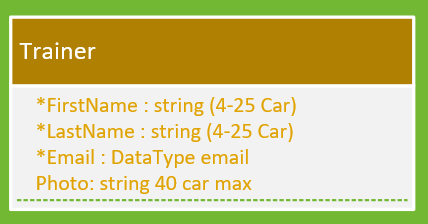
1. Ajoutez une migration et mettez à jour votre base de données
2. Le menu **Admin/Specialities** de la page principale doit nous diriger vers la gestion des spécialités
3. Prenez un moment pour retirer la vue **Details** et retirer le lien dans la vue **Index**
4. À ce point, vous devez pouvoir faire les actions **CRUD** de **Speciality**

ATTENTION

C'est le moment de **tester**, ajouter des **commentaires** et faire votre **commit** et votre **push**

## Gestion des entraîneurs[​](https://info.cegepmontpetit.ca/BW5-Web-Transactionelle/tp_Regulier/tp1#gestion-des-entra%C3%AEneurs)

1. Ajout de la classe **Trainer** qui sera gérer par Entity Framework Core dans le répertoire **/Models/**



Trainer

public string FirstName { get; set; }  
public string LastName { get; set; }  
public string Email { get; set; }  
public string Photo { get; set; }

INFO

Il faut ajouter les champs nécessaires pour les **relations** et les **annotations**.

1. Génération du contrôleur MVC pour la classe **Trainer**
2. Ajouter un seed🌱 pour les entraîneurs

Seed des entraîneurs

builder.Entity<Trainer>().HasData(new Trainer() { Id = 1, FirstName = "Chrystal", LastName = "Lapierre", Email = "Chrystal.lapierre@juliepro.ca", SpecialityId= 1, Photo = "Chrystal.png"});  
builder.Entity<Trainer>().HasData(new Trainer() { Id = 2, FirstName = "Félix", LastName = "Trudeau", Email = "Felix.trudeau@juliePro.ca", SpecialityId = 2, Photo = "Felix.png" });  
builder.Entity<Trainer>().HasData(new Trainer() { Id = 3, FirstName = "François", LastName = "Saint-John", Email = "Frank.StJohn@juliepro.ca", SpecialityId = 1, Photo = "Francois.png" });  
builder.Entity<Trainer>().HasData(new Trainer() { Id = 4, FirstName = "Jean-Claude", LastName = "Bastien", Email = "JC.Bastien@juliepro.ca", SpecialityId = 4, Photo = "JeanClaude.png" });  
builder.Entity<Trainer>().HasData(new Trainer() { Id = 5, FirstName = "Jin Lee", LastName = "Godette", Email = "JinLee.godette@juliepro.ca", SpecialityId = 3, Photo = "Jin Lee.png" });  
builder.Entity<Trainer>().HasData(new Trainer() { Id = 6, FirstName = "Karine", LastName = "Lachance", Email = "Karine.Lachance@juliepro.ca", SpecialityId = 2, Photo = "Karine.png" });  
builder.Entity<Trainer>().HasData(new Trainer() { Id = 7, FirstName = "Ramone", LastName = "Esteban", Email = "Ramone.Esteban@juliepro.ca", SpecialityId = 3, Photo = "Ramone.png" });

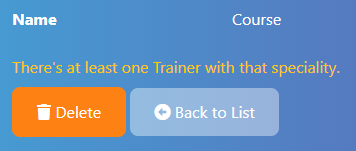
1. Ajoutez une migration et mettez à jour votre base de données
2. Le menu **Admin/Trainers** de la page principale doit nous diriger vers la gestion des entraîneurs
3. Modification de la vue et ajout des entraîneurs
   * Dans la vue Index:
     + Triez les entraîneurs par FirstName en premier et par LastName ensuite
     + Affichez l'image de la photo de l'entraîneur si elle est présente et limitez sa largeur à 200 pixels
     + Affichez le nom de la spécialité, pas son id
   * Dans la vue Details:
     + Affichez l'image de la photo de l'entraîneur en pleine résolution si elle est présente
     + Affichez le nom de la spécialité, pas son id
   * Dans la vue Delete:
     + Affichez le nom de la spécialité, pas son id
   * Dans les vues Create et Edit:
     + Votre select de spécialité est-il vraiment laid?🤮 Rappelez-vous que l'on utilise bootstrap et que les différents éléments doivent utiliser les classes bootstraps!
     + Permettre de sélectionner une spécialité et affichez le nom des spécialités et non pas leurs ids
     + Pour la photo, on garde ça simple pour l'instant et on doit taper le nom de l'image de l'entraîneur

ATTENTION

C'est le moment de **tester**, ajouter des **commentaires** et faire votre **commit** et votre **push**

## Implémentation de la logique d'affaire[​](https://info.cegepmontpetit.ca/BW5-Web-Transactionelle/tp_Regulier/tp1#impl%C3%A9mentation-de-la-logique-daffaire)

1. Empêchez d’effacer une **Speciality** si elle est associé à au moins un **Trainer** et affichez un message approprié à l'utilisateur dans ce cas



1. Assurez-vous que le delete d'une **spécialité** fonctionne correctement si elle n'est pas utilisée
2. Assurez-vous que le delete d'un **entraîneur** fonctionne correctement

ATTENTION

C'est le moment de **tester**, ajouter des **commentaires** et faire votre **commit** et votre **push**

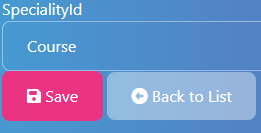
## Utilisation des vues partielles et de FontAwesome[​](https://info.cegepmontpetit.ca/BW5-Web-Transactionelle/tp_Regulier/tp1" \l "utilisation-des-vues-partielles-et-de-fontawesome" \o "Lien direct vers Utilisation des vues partielles et de FontAwesome)

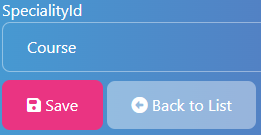
1. Utilisez une vue partielle pour regrouper le bouton d'**action** et le bouton de **retour vers la liste**. Utilisez une vue partielle pour les boutons d'actions des vues suivantes:
   * Trainers/Create
   * Trainers/Edit
   * Trainers/Delete
   * Specialities/Create
   * Specialities/Edit
   * Specialities/Delete

ATTENTION

Chaque action peut avoir sa propre vue partielle et c'est aussi possible que 2 actions partagent une même vue partielle. Au total, vous ne devrez donc pas avoir plus de 3 vue partielles différentes pour cette section, car il existe 3 actions (Create, Edit et Delete) et vous **DEVEZ** réutiliser vos vues partielles entre **Trainer** et **Speciality**

1. Maintenant que vos boutons d'actions sont réutilisés dans vos différentes vues, utilisez FontAwesome pour les boutons suivants:
   * pour Create
   * pour Save
   * pour Delete
   * pour Back to List
2. Si vos boutons sont trop près des autres champs, donnez leur un peu d'espace!

 ❌

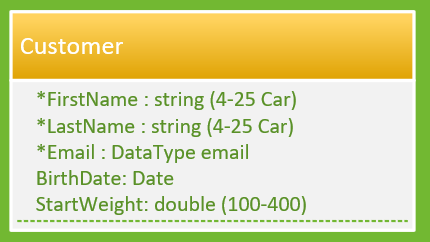
 👌

ATTENTION

C'est le moment de **tester**, ajouter des **commentaires** et faire votre **commit** et votre **push**

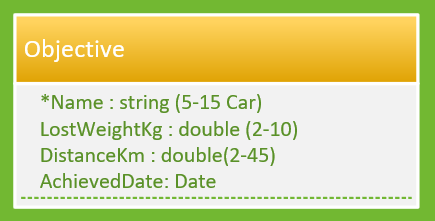
## Ajout de clients et de leurs objectifs[​](https://info.cegepmontpetit.ca/BW5-Web-Transactionelle/tp_Regulier/tp1#ajout-de-clients-et-de-leurs-objectifs)

1. Mettez en place le modèle de donnée pour les clients (**Customer**) et leurs objectifs (**Objective**)



Customer

public string FirstName { get; set; }  
public string LastName { get; set; }  
public string Email { get; set; }  
public DateTime BirthDate { get; set; }  
public double StartWeight { get; set; }



Objective

public string Name { get; set; }  
public double LostWeightKg { get; set; }  
public double DistanceKm { get; set; }  
public DateTime AchievedDate { get; set; }

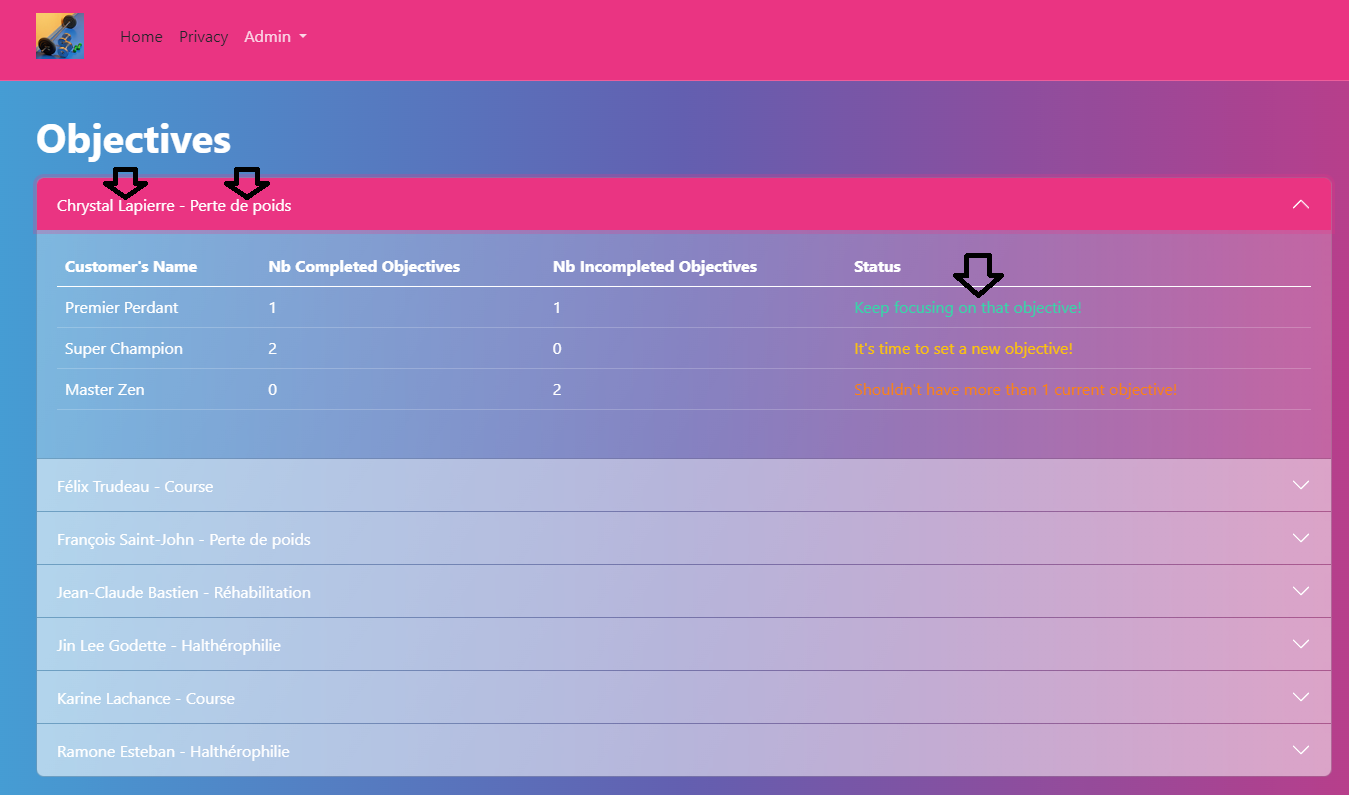
1. Un **Customer** a donc une relation **un à plusieurs** avec ses **Objective**. La seule chose qui différencie l'objectif courant, c'est qu'il n'a pas d'**AchievedDate**.
2. Ajouter un seed 🌱 avec 3 clients (**Customer**) et leurs objectifs (**Objective**).
   * Les 3 clients doivent etre associé au même entraîneur: **Chrystal Lapierre**
   * Un objectif est considéré comme **courant** si il n'a pas d'AchievedDate et comme **complété** si il en a un.
   * Le premier client doit avoir un objectif **courant** et un objectif **complété**.
   * Le deuxième client doit avoir deux objectifs **courants**.
   * Le troisième client doit avoir deux objectifs **complétés**.
   * Le contenu exacte des objectif n'est pas important, mais ils doivent tous être différents.
3. Une fois que c'est fait, vérifiez les données dans votre base de données!

ATTENTION

C'est le moment de **tester**, ajouter des **commentaires** et faire votre **commit** et votre **push**

## Affichage d'un menu d'objectifs[​](https://info.cegepmontpetit.ca/BW5-Web-Transactionelle/tp_Regulier/tp1#affichage-dun-menu-dobjectifs)

1. C'est maintenant le moment de travailler sur la dernière fonctionnalitée, l'affichage des objectifs.
2. Voici ce que l'on doit voir dans ce menu:



1. Pour le menu précédent vous devrez utiliser un **accordion** de bootstrap pour afficher chacun des entraîneurs.
2. Une fois que vous affichez le contenu pour un entraîneur, utilisez simplement un tableau.
3. Pour ce menu, vous **devez** utiliser au une vue partielle pour affichez le tableau et vous **devez** également utiliser une vue partielle pour afficher le contenu de chaque rangé du tableau.
4. Vous **devez** également utiliser un ViewModel qui doit se nommer **TrainerObjectivesVM**.
5. Vous **devez** également créer un ViewModel qui doit se nommer **CustomerObjectivesVM**.
6. Le ViewModel **TrainerObjectivesVM** **doit**, en plus de ses autres propriétés, contenir un IEnumerable\<CustomerObjectivesVM>.
7. Vous pouvez nommer vos vues comme vous le voulez, mais voici une suggestion:
   * Vue principale avec l'accordéon (**accordion**) (AllObjectives.cshtml)
     + Vue partielle avec le tableau (TrainerObjectives.cshtml)
       - Vue partielle avec une rangé du tableau (CustomerObjectives.cshtml)
8. Remarquez que pour chaque titre de l'**accordion** on affiche le nom de l'entraîneur **ET sa spécialité**.
9. Pour le status, il faut afficher un de ces 3 messages qui dépend directement du nombre d'objectif incomplet qui est affiché dans la colonne précédente. Utilisez une couleur différente pour chaque message (color-success, color-warning, color-danger).
10. Finalement, si ce n'est pas déjà fait, il ne faut pas oublier d'ajouter une nouvelle option qui doit se nommer **Objectives** sous **Admin**, dans le menu de navigation pour pouvoir accéder à la page d'objectifs.

ATTENTION

C'est le moment de **tester**, ajouter des **commentaires** et faire votre **commit** et votre **push**

DANGER

Une fois que vous avez terminé votre TP, il est temps de merger votre branche **TP1** dans votre branche **Main**.

## Grille de correction[​](https://info.cegepmontpetit.ca/BW5-Web-Transactionelle/tp_Regulier/tp1#grille-de-correction)

| **Tâche** | **Nb Points** |
| --- | --- |
| Mise en place du projet initial | 5 |
| Gestion des spécialités | 10 |
| Gestion des entraîneurs | 10 |
| Logique d'affaire | 5 |
| Utilisation de vues partielles pour les boutons | 3 |
| Utilisation de FontAwesome | 2 |
| Ajout des clients et de leurs objectifs | 5 |
| Affichage des objectifs | 5 |
| Consignes Git (branches/commits/push) | 5 |
| **Total** | **/50** |