**Mise en place TP Formation double compétence - Spécifications** 07/03/2019

Table des matières

[Objectif du document 3](#_Toc58247790)

[Principe 3](#_Toc58247791)

[Besoins 3](#_Toc58247792)

[Spécifications 3](#_Toc58247793)

[Page d’accueil 3](#_Toc58247794)

[1ère Etape : Affichage du logo et du titre 4](#_Toc58247795)

[2ème Etape : Création du menu 5](#_Toc58247796)

[3ème Etape : Création du corps de page 6](#_Toc58247797)

[4ème Etape : Mise en place des liens 7](#_Toc58247798)

[Formulaire 7](#_Toc58247799)

[1ère Etape : Affichage du logo et du titre 8](#_Toc58247800)

[2ème Etape : Création du menu 8](#_Toc58247801)

[3ème Etape : Création du tableau de saisie « Informations personnelles » 8](#_Toc58247802)

[4ème Etape : Création du tableau de saisie « Informations formation suivie » 8](#_Toc58247803)

[5ème Etape : Création du tableau de saisie « Avis sur la formation » 9](#_Toc58247804)

[6ème Etape : Création d’un bouton 9](#_Toc58247805)

[Page de validation 10](#_Toc58247806)

[Page Liste 10](#_Toc58247807)

[1ère Etape : Affichage du logo et du titre 10](#_Toc58247808)

[2ème Etape : Création du menu 10](#_Toc58247809)

[3ème Etape : Création de la liste des avis 10](#_Toc58247810)

[Bonus 11](#_Toc58247811)

[Afficher plusieurs messages d’erreurs simultanément 11](#_Toc58247812)

[Rendre le menu et le header dynamique 11](#_Toc58247813)

[Pour plus d’informations sur le MVC 12](#_Toc58247814)

[Annexes 13](#_Toc58247815)

[Annexe 1 : Visuel page d’accueil 13](#_Toc58247816)

[Annexe 2 : Visuel formulaire 14](#_Toc58247817)

[Annexe 3 : Visuel page de validation 15](#_Toc58247818)

[Annexe 4 : Visuel liste 16](#_Toc58247819)

[Annexe 5 : Solution Explorer 17](#_Toc58247820)

[Annexe 6 : Affichage du logo et du titre 19](#_Toc58247821)

[Annexe 7 : Création du menu 20](#_Toc58247822)

[Annexe 8 : Création du corps de page 21](#_Toc58247823)

[Annexe 9 : Index.cshtml 21](#_Toc58247824)

[Annexe 10 : Index.css 23](#_Toc58247825)

[Annexe 11 : Récupération de la liste des avis 24](#_Toc58247826)

[Annexe 12 : autocomplétion du chemin d’un src : 25](#_Toc58247827)

**Suivi du document :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Action** | **Personne** | **Date** |
| V1.0 | Création du document | Joffrey LEFRERE | 07/03/2019 |
| V1.1 | Modification du document | Aurélie BROS | 12/03/2019 |
| V1.2 | Modification du document | Daphnée HOT | 08/04/2019 |
| V1.3 | Modification du document | Aurélie BROS | 25/04/2019 |
| V1.4 | Modification du document | Louis TEMPELAERE | 04/12/2020 |
| V1.5 | Ajout d’exemples de différents types de contrôles dans la partie : Formulaire – 5ème étape | Louis TEMPELAERE | 17/02/2021 |

# Objectif du document

Ce document décrit un TP dédié aux collaborateurs formés en double-compétence et voués notamment à évoluer sur les sites EI du CM-CIC. Il résulte du constat que la partie LOCAL(C#) de la formation double compétence délivrée jusqu’alors par HN Institut n’est pas en adéquation avec le travail demandé par la suite chez le client.

# Principe

Le TP vise à simuler une application permettant l’évaluation d’une formation. Cette application s’articulera comme suit :

* Une page d’accueil ([Annexe 1](#_Annexe_1))
* Une page de renseignement de formulaire ([Annexe 2](#_Annexe_2_:))
* Une page de validation des informations renseignées sur le formulaire ([Annexe 3](#_Annexe_3_:))
* Une page faisant apparaître la liste des avis laissés ([Annexe 4](#_Annexe_4_:))

# Besoins

Afin de réaliser ce TP, il faudra mettre à disposition des futurs collaborateurs :

* Ce document de spécifications expliquant le travail à réaliser
* Une solution Visual Studio de départ qui contient le routage et les dossiers appropriés.
* Un fichier avec le logo HN
* Un fichier avec une liste de nom, prénom et indicateur O/N afin de constituer la liste figurant sur la « Page liste » ([Page liste](#_Page_Liste)) : « DataAvis.xml »
* Une solution Visual Studio complètement codée et commentée. Elle sera à consulter en cas de blocage.

# Spécifications

## Page d’accueil

Afin de débuter tranquillement, nous vous proposons de vous accompagner pas à pas dans le développement de cette première page ([Annexe 1](#_Annexe_1)).

Sur cette page vous vous exercerez :

* A mettre en forme une page internet HTML
* A créer un menu
* A créer des liens de navigation entre 2 pages

Dans un premier temps, il faut créer la solution qui servira de base à tous les développements. Récupérer la solution fournie avec le TP. Il s’agit d’une solution MVC (Modèle Vue Controller) APS.NET, dans laquelle le controller et le routage est déjà créés pour vous faire gagner du temps. En effet, vous n’aurez pas à coder cette partie au sein du CIC. Ouvrir le TPLOCAL1.sln pour ouvrir cette solution. Le choix technique du MVC est similaire à celui que vous trouverez au CIC. Le fonctionnement est le suivant :

* Le modèle contient les données utilisées par le site. En C#, toutes les données sont dans des classes, donc le modèle sera sous forme de classes.
* La vue contient ce que l’on voit à l’écran du navigateur web. Elle sera constituée de pages cshtml, qui permet à la fois du code html et du code c#.
* Enfin le controller définit la logique du site web, par exemple les liens entre les pages, les contrôles des données rentrées par l’utilisateur, les appels aux bases de données (non présents dans ce TP), les lectures de fichiers…

### 1ère Etape : Affichage du logo et du titre

Le but de cette partie est d’aligner le logo et le titre([Annexe 6](#_Annexe_8_:)).

Maintenant nous allons créer la première page cshtml de vue. Pour cela, dans la fenêtre Solution Explorer ([Annexe 5](#_Annexe_6_:_1)), faire un clic droit sur le dossier View/Home, puis Ajouter et Vue. Cliquer sur ajouter. Nommer la vue Index puis ajouter.

Cet ajout crée une page cshtml, dans laquelle on va mettre du code html, et qui est reconnue par le moteur MVC. Cette page contient 2 parties :

* L’en-tête avec la balise <head> : cette section permet de donner le titre de la page, l’encodage…
* Le corps <body> : partie principale de la page.

DANS L’EN-TÊTE : Tout d’abord nous allons nommer la page grâce à la balise <title> : <title>HN - TP LOCAL</title>  
Cette balise permet d’avoir le nom de la page qui s’affiche dans l’onglet internet.

DANS LA PARTIE PRINCIPALE DE LA PAGE :

Pour mettre un titre à la page, il faut utiliser la balise <header>, elle gère l’en-tête d’une page html :  
 <header>

<h1>Bienvenue dans le TP HN</h1>

</header>

La balise <h1> permet de définir un titre de niveau 1 et sera utilisé par la suite dans la page .css.

Maintenant que le titre est renseigné, il faut mettre le logo. Pour cela, dans le dossier de la solution, dans le sous-dossier « ressources » mettre le fichier contenant l’image en s’assurant que le nom du fichier est bien GroupeHN.png. Si le fichier est dans le dossier mais n’est pas visible dans la solution, faire ajouter/un élément existant sur le dossier « ressources », et choisir le fichier dans l’explorateur windows.

Dans la page Index.cshtml, il faut rajouter une balise <img> :   
<img src="~/ressources/GroupeHN.png" class="image" alt="GroupeHN" />

L’attribut src indique l’emplacement du fichier et alt indique ce que contient l’image. Le ~ indique qu’il s’agit d’un chemin, et l’autocomplétion du code propose normalement le chemin (annexe 11). Pour utiliser l’autocompletion, quand elle apparait, utiliser les flèches du clavier et entrée ou la souris.

Lancer le debug en tapant F5 (sur la page cshtml). Si le code ne compile pas, mettre en commentaires les parties qui bugguent (par exemple la méthode ValidationFormulaire du controller).

Une ligne est en commentaire quand elle à « // » au début pour une page C#, et < !-- -- > pour une page HTML.

Le logo et le titre sont l’un au-dessus de l’autre. Pour aligner le logo et le titre, il faut créer une page .css. Les pages .css permettent la mise en forme des pages cshtml.  
  
Pour créer une page .css, de même que pour la création d’une page cshtml cliquer sur le dossier « ressources », puis ajouter nouvel élément, choisir web/feuille de style. Elle sera nommée Index.css  
  
Tout d’abord, il faut s’occuper de l’image, en rajoutant le code suivant dans la page css :   
.image

{

width : 150px;

height : 150px;

}   
On vient de créer la classe image. Les attributs width et height servent à redimensionner l’image en largeur et en hauteur.  
  
Pour le titre, il faut mettre dans la page css :   
h1

{

margin-right: 170px;

margin-left: 170px;

text-align : center;

}

margin-right et margin-left servent à définir une marge à droite et à gauche du texte.  
text-align sert à définir l’alignement du texte. Pour relier la page css à la page html, il faut créer un lien entre ces 2 pages dans la page cshtml. Pour cela il faut utiliser une balise <link> dans la section <head> :  
<link rel="stylesheet" href="~/ressources/Index.css"/>

A ce stade, le titre et le logo sont mis en forme, mais lorsqu’on lance le débug, ils ne sont toujours pas alignés.

Pour aligner le logo et le titre, il faut rajouter 2 éléments dans les classe .image et h1 :  
 display:inline-block;

vertical-align:middle;

La première ligne permet de faire un bloc des 2 éléments et la deuxième ligne permet d’aligner le bloc.

### 2ème Etape : Création du menu

Le but de cette partie est de vous montrer comment créer un menu([Annexe 7](#_Annexe_9_:)).

Le menu se trouvera dans la partie <body> de la page cshtml.

Le code pour un menu est le suivant :

<nav>

<ul>

<li><a href="#">Remplir le formulaire</a></li>

<li><a href="#">Liste avis</a></li>

</ul>

</nav>

Explication des balises :

* <nav> : lien principaux de navigation, sert entre autres au menu principal,
* <ul> : Liste à puces,
* <li> : Élément d’une liste à puce,
* <a> : Balise pour les liens hypertexte, il faut lui associer l’attribut href pour indiquer vers quelle page le lien doit conduire. Ici il y a "#" car les pages ne sont pas encore créées, ce sera à remplacer plus tard.

On peut voir grâce au débug que le menu est apparu sur la page mais il n’est pas très joli. Pour améliorer le visuel, il faut modifier la page css :

nav

{

width: 150px;

border: 2px solid #8c014c;

background-color: #8c014c;

color: white;

}

li

{

margin-top: 10px;

margin-bottom: 15px;

}   
a

{

color: chocolate;

}

2 classes css ont été créées, nav, qui correspond à la balise <nav> html et li pour la balise <li> html.

Explication des attributs :

* width : largeur du bloc en pixels (possibilité de le faire en pourcentage %)
* border : définition de la bordure du bloc, ici la structure est épaisseur de la bordure (2 px) type de bordure (solid) couleur (#8c014c)
* background-color : couleur du fond du bloc, soit couleur (yellow, blue …) ou référence (#8c014c)
* color : couleur du texte
* margin-top : marge au-dessus de la ligne
* margin-bottom : marge en-dessous de la ligne

Et voilà le menu est créé, il ne restera plus qu’à mettre les liens.

### 3ème Etape : Création du corps de page

Le but de cette étape est de voir comment remplir et mettre un lien dans le corps de page([Annexe 8](#_Annexe_7_:)).

Comme précédemment les modifications se font dans la partie <body> de la page cshtml.

Pour créer le corps de la page, il faut rajouter le code suivant :

<section>

<p>

<a href="#">Remplir le formulaire </a>

</p>

</section>

Explication des balises :

* <section> : section de page, sert à regrouper des contenus,
* <p> : paragraphe.

Regarder à quoi ressemble votre page maintenant que tous les éléments sont mis en place.  
Il y a un problème, le corps de la page n’est pas en face du menu mais en dessous. C’est normal, il faut modifier la page css. Dans la classe nav, il faut rajouter :  
 float: left;

La propriété float permet de rendre un objet flottant, elle peut être utilisée pour un menu ou une image et peut prendre 2 valeurs qui sont left ou right.

Maintenant le corps de la page et le menu sont alignés mais un peu trop collé. Il suffit de créer une nouvelle classe pour définir une marge à gauche pour la balise section et le tour sera joué :   
section

{

margin-left: 170px;

}

### 4ème Etape : Mise en place des liens

Le but de cette partie est de vous montrer comment créer un lien.

Tout d’abord, il suffit de remplacer "#" par "/Home/Index/Formulaire " dans les balises <a href="/Home/Index/Formulaire">Remplir le formulaire</a>. Ce chemin signifie qu’on appelle le controller HomeController, et plus précisément sa fonction Index, à laquelle on fournit le paramètre Formulaire. Le controller n’est pas à modifier.

Il faut ensuite créer la page Formulaire.cshtml. Attention, il faut créer cette vue dans le dossier Views/Shared pour que le routage fonctionne correctement (si ceci vous intéresse, vous trouverez un lien explicatif en fin d’énoncé cependant vous n’aurez pas besoin d’en connaitre la raison une fois sur site).

Le code return View(id); permet de renvoyer vers la page Formation. La méthode View permet de renvoyer vers la vue désirée, ici sans autres paramètres. Par la suite, on utilisera aussi la forme surchargée de la méthode View avec la page à appeler et un modèle de données.

Lorsque vous cliquez sur le lien nouvellement mis en place, vous arrivez sur une page blanche ce qui est normal car elle n’est pas encore codée.

Lorsque la page Listeavis.cshtml sera créée, vous pourrez mettre en place son lien dans le menu.

Pour voir le code de la page Index.cshtml voir [Annexe 9](#_Annexe_11_:) et pour voir le code de la page Index.css voir [Annexe 10](#_Annexe_12_:).

## Formulaire

A partir de cette partie, vous serez moins guidés. Si vous rencontrez des difficultés, cherchez la solution sur internet, puis si vous n’y arrivez pas, consultez la solution fournie par HN.

Il s’agit ici de créer une page qui permettra à l’utilisateur de renseigner ses données personnelles ainsi que son avis sur une formation ([Annexe 2](#_Annexe_2_:)).

Sur cette page vous vous exercerez :

* A créer une liste déroulante
* A mettre en place divers contrôles selon les informations attendues dans chacun des champs
* A gérer les messages d’erreur

Le développement de cette page s’organise en 6 étapes :

### 1ère Etape : Affichage du logo et du titre

Voir précédemment ([1ère étape page d’accueil](#_1ère_Etape_:)). Attention, le titre est différent de celui de la page d’accueil.

### 2ème Etape : Création du menu

Voir précédemment ([2ème étape page d’accueil](#_2ème_Etape_:)). Attention, les rubriques sont différentes de celles de la page d’accueil.

### 3ème Etape : Création du tableau de saisie « Informations personnelles »

Comme son nom l’indique, l’utilisateur devra être en mesure de renseigner dans ce tableau plusieurs données qui lui sont propres.

Le tableau devra comporter 2 colonnes, l’une pour l’intitulé de l’information demandée, l’autre pour son renseignement. Cependant, il y a une exception pour le titre qui devra être centré sur les 2 colonnes.

Voici les différents champs que devra comporter le tableau avec ses restrictions / contrôles :

|  |  |
| --- | --- |
| **Colonne de gauche (intitulés)** | **Colonne de droite (type de champs, contrôles éventuels)** |
| Informations personnelles | |
| Nom | Champ de saisie |
| Prénom | Champ de saisie |
| Sexe | * Liste déroulante comprenant 4 choix : * Sélectionner un sexe * Homme * Femme * Autre * Par défaut le champ doit être renseigné avec « Sélectionner un sexe » |
| Adresse | Champ de saisie |
| Code postal | * Champ de saisie * Contrôle : devra être sur 5 caractères numériques grâce à une expression régulière |
| Ville | Champ de saisie |
| Adresse mail | * Champ de saisie * Contrôle format à mettre en œuvre |

Regex (= expression régulière) possible : ^([\w]+)@([\w]+)\.([\w]+)$

(Source : <https://lgmorand.developpez.com/dotnet/regex/>)

Les données sont à stocker dans une classe dans le dossier Models.

### 4ème Etape : Création du tableau de saisie « Informations formation suivie »

Ce tableau permet à l’utilisateur d’indiquer des informations sur la formation qu’il a suivie.

Le tableau devra comporter 2 colonnes, l’une pour l’intitulé de l’information demandée, l’autre pour son renseignement. Cependant, il y a une exception pour le titre qui devra être centré sur les 2 colonnes.

Voici les différents champs que devra comporter le tableau avec ses restrictions / contrôles :

|  |  |
| --- | --- |
| **Colonne de gauche (intitulés)** | **Colonne de droite (type de champs, contrôles éventuels)** |
| Informations formation suivie | |
| Date début formation | * Champ de saisie de date * La date doit être inférieure au 01/01/2021 |
| Type de formation | * Champ de saisie * Liste déroulante comprenant 4 choix : * Sélectionner une formation * Formation Cobol * Formation objet * Formation double compétence * Par défaut le champ doit être renseigné avec « Sélectionner une formation » |

### 5ème Etape : Création du tableau de saisie « Avis sur la formation »

Ce tableau permet à l’utilisateur de donner son avis sur la formation qu’il a suivie.

Le tableau devra comporter 2 colonnes, l’une pour l’intitulé de l’information demandée, l’autre pour son renseignement. Cependant, il y a une exception pour le titre qui devra être centré sur les 2 colonnes.

Voici les différents champs que devra comporter le tableau avec ses restrictions / contrôles :

|  |  |
| --- | --- |
| **Colonne de gauche (intitulés)** | **Colonne de droite** |
| Avis sur la formation | |
| Formation Cobol | * Champ de saisie |
| Formation Objet | * Champ de saisie |

Pour les contrôles, il est possible de les faire en HTML ou dans la déclaration des variables dans une classe (voir [Required and StringLength Attribute in MVC - Dot Net Tutorials](https://dotnettutorials.net/lesson/required-attribute-asp-dot-net-mvc/) et [Regular Expression Attribute in MVC - Dot Net Tutorials](https://dotnettutorials.net/lesson/regular-expression-attribute-mvc/)) ou enfin dans le controller lors du clic sur le bouton Valider.

Quelques explications :

* En HTML, dans la balise input, il est possible de définir dans le champ « type » le type de la donnée. En fonction du type, le visuel est variable. Par exemple <input type="date"/> donne un calendrier. Ce type de contrôle est très pratique mais n’est pas très flexible, car il existe un nombre limité de types.
* Dans la déclaration des variables, il est possible d’ajouter Required au-dessus de la variable des attributs pour contrôler que les données sont bien renseignées :

[Required]

public string Formation { get; set; }

Il est aussi possible de faire des contrôles plus poussés de cette façon, par exemple sur la taille des données à rentrer par l’utilisateur, avec les expressions régulières, ou encore la plage de donnée de la date.

* Enfin, il est aussi possible de faire des contrôles dans la fonction appelée par un bouton. Par exemple, en supposant que le modèle où sont les données d’appelle modelv, on peut renvoyer une erreur si la donnée adresse est vide ou n’est pas assez longue :

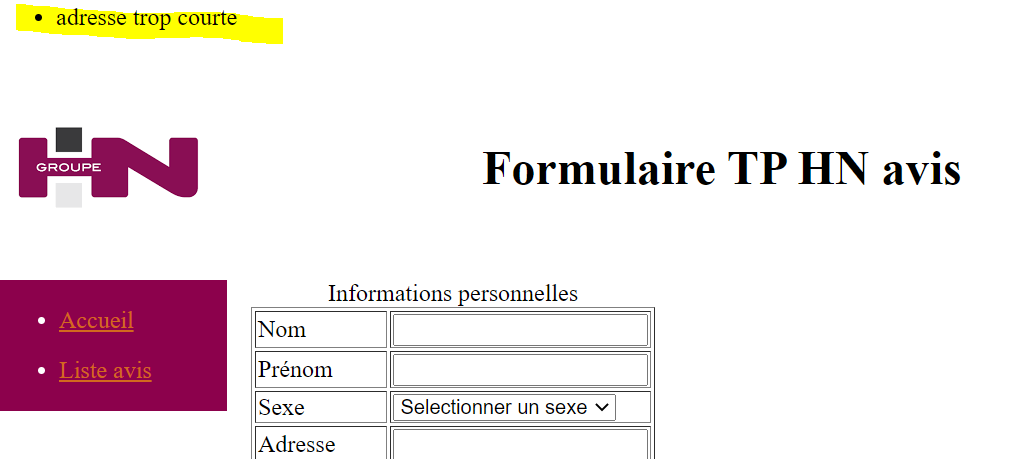
if (modelv.Adresse == null || modelv.Adresse.Length < 5)

{

ModelState.AddModelError("", "adresse trop courte");

}

Et pour afficher les erreurs, on ajoute dans la page cshtml au tout début de la page après le @model TPLOCAL1.Models.FormulaireModel, le code : @Html.ValidationSummary(false,""). On a alors le visuel suivant :



### 6ème Etape : Création d’un bouton

Il s’agit de créer un bouton de « Validation ». Voici ce qui est attendu pour ce bouton :

* Doit générer l’affichage d’au moins un message d’erreur si au moins un des contrôles définis sur la page est KO.
* Doit afficher la page de validation correctement remplie dans le cas où tous les contrôles de la page sont OK.

Pour information, on pourra utiliser le champ action de la balise form pour appeler la méthode ValidationFormulaire du HomeController pour ce bouton. Dans cette méthode, on contrôlera les données et on enverra sur la bonne page avec le modèle des données en paramètres.

## Page de validation

Il s’agit ici de créer une page qui permettra à l’utilisateur de consulter les informations qu’il vient de renseigner sur la page formulaire ([Annexe 3](#_Annexe_3_:)).

Sur cette page vous vous exercerez :

* A véhiculer des données d’une page à une autre

Cette page devra reprendre les mêmes éléments que la page du formulaire aux différences près suivantes :

* En première ligne du corps de page devra figurer la mention « Votre formulaire à bien été pris en compte », centrée sur la page
* Les colonnes de droite des tableaux devront être renseignées avec les informations entrées par l’utilisateur sur la page de formulaire. Pour information, pour afficher une donnée d’une classe C#, dans la page cshtml, on définit la classe en haut de la page de la manière suivante :

@model TPLOCAL1.Models.FormulaireModel (avec le chemin relatif TPLOCAL1.Models et FormulaireModel le nom de la classe du modèle).

On note avec l’arobase les données C# dans les pages cshtml. Ensuite, pour utiliser une donnée de ce modèle dans la page cshtml, on écrit @Model.nomDeLaDonnée.

On utilisera le même modèle pour la page de validation que pour la page de formulaire.

* Aucune information ne devra être modifiable par l’utilisateur.
* La date sera affichée au format français (Jour/mois/année).

## Page Liste

Il s’agit ici de créer une page qui permettra à l’utilisateur de consulter les informations qui sont présentes dans un fichier ([Annexe 4](#_Annexe_4_:)).

Sur cette page vous vous exercerez :

* à constituer une liste à partir de données présentes dans un fichier
* à conditionner la mise en forme d’un champ en fonction de sa valeur

Le développement de cette page s’organise en 3 étapes :

### 1ère Etape : Affichage du logo et du titre

Voir précédemment ([1ère étape page d’accueil](#_1ère_Etape_:)). Attention, le titre est différent de celui de la page d’accueil

### 2ème Etape : Création du menu

Voir précédemment ([2ème étape page d’accueil](#_2ème_Etape_:)). Attention, les rubriques sont différentes de celles de la page d’accueil

### 3ème Etape : Création de la liste des avis

Il s’agit ici d’afficher une liste dont les informations proviennent d’un fichier mis à votre disposition.

Cette liste sera présentée sous forme d’un tableau à 3 colonnes dont les données ne seront pas modifiables. Il comportera autant de lignes qu’il y a de personnes répertoriées dans le fichier source plus la ligne de titre du tableau et celle des titres des colonnes. Dans la première colonne apparaîtront les noms des personnes figurant dans le fichier, dans la seconde leur prénom et dans la dernière colonne un top indiquant si les personnes ont laissé ou non un avis sur la formation. Cependant, il y a une exception pour le titre qui devra être centré sur les 3 colonnes.

Voici les différents champs que devra comporter le tableau :

|  |  |  |
| --- | --- | --- |
| **Colonne 1** | **Colonne 2** | **Colonne 3** |
| Libellé titre tableau « Liste » | | |
| Libellé titre colonne « Nom » | Libellé titre colonne « Prénom » | Libellé titre colonne « Avis donné » |
| Nom 1 | Prénom 1 | * Avis donné renseigné par : * « Oui » si le top présent dans le fichier source est « O » * « Non » sinon * Afficher le champ en : * vert si le top présent dans le fichier source est « O » * rouge sinon |
| Nom 2 | Prénom 2 | * Avis donné renseigné par : * « Oui » si le top présent dans le fichier source est « O » * « Non » sinon * Afficher le champ en : * vert si le top présent dans le fichier source est « O » * rouge sinon |
| Nom 3 | Prénom 3 | * Avis donné renseigné par : * « Oui » si le top présent dans le fichier source est « O » * « Non » sinon * Afficher le champ en : * vert si le top présent dans le fichier source est « O » * rouge sinon |
| … | … | … |

Les données seront récupérées du fichier DataAvis.xml en utilisant les classes présentes dans l’[annexe 11](#_Annexe_13_:). Le fichier est à placer dans le dossier FichierXML de la solution.

Pour parcourir la liste, l’utilisation d’une boucle foreach est imposée.

## Bonus

### Afficher plusieurs messages d’erreurs simultanément

Le but est le suivant : sur la page de formulaire, dans le cas où il y a plusieurs erreurs, afficher tous les messages d’erreurs correspondant (au lieu d’afficher seulement le message d’erreur de la première erreur rencontrée).

### Rendre le menu et le header dynamique

Le but serait :

* De coder le menu et le header une seule fois et d’appeler le code sur chaque page
* De conditionner le renseignement du menu et du header à la page par laquelle ils sont appelés.

### Pour plus d’informations sur le MVC

Vous pouvez lire le cours d’openclassroom sur [Apprenez ASP.NET MVC - OpenClassrooms](https://openclassrooms.com/fr/courses/1730206-apprenez-asp-net-mvc/) pour voir comment le controller et le routage fonctionnent. Cela n’est cependant pas nécessaire pour le travail au CIC et sert uniquement pour votre culture générale.

# Annexes

## Annexe 1 : Visuel page d’accueil

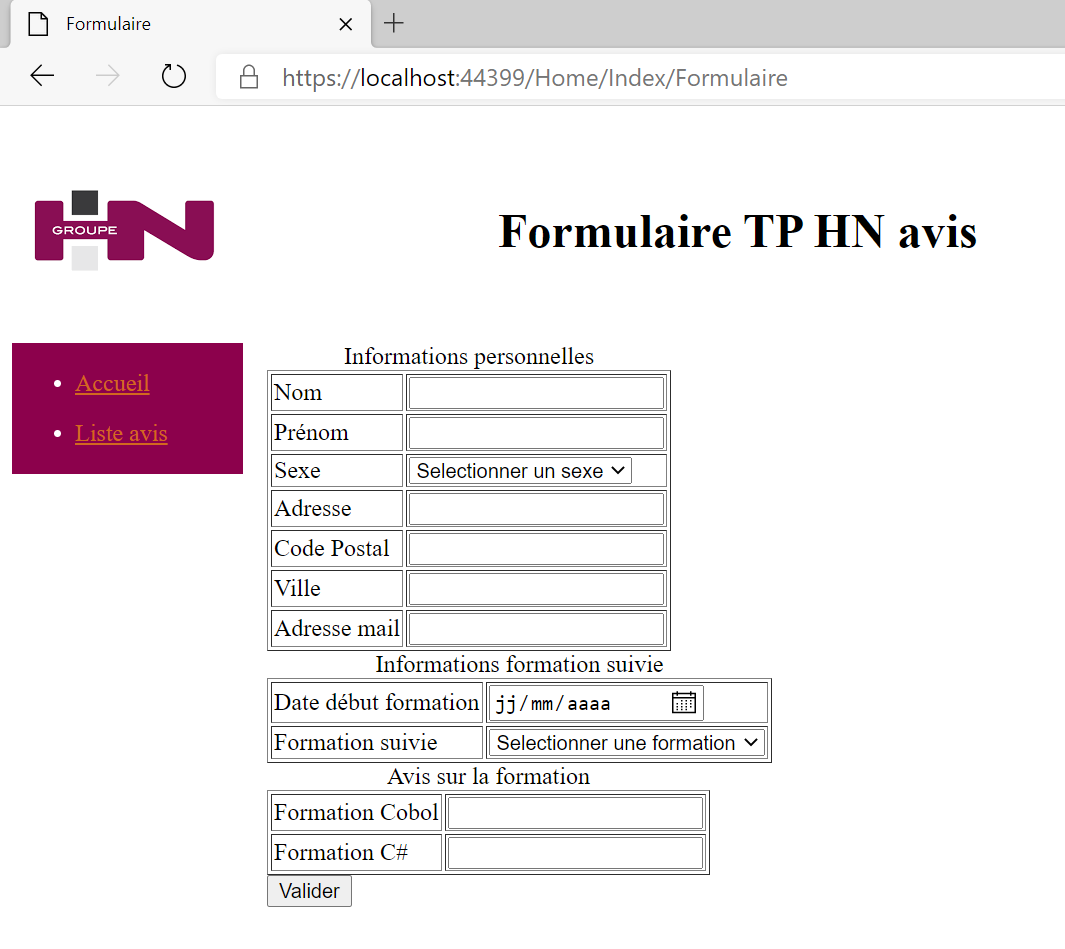
([Retour paragraphe](#_Principe))



## Annexe 2 : Visuel formulaire

([Retour paragraphe « Principe »](#_Principe))

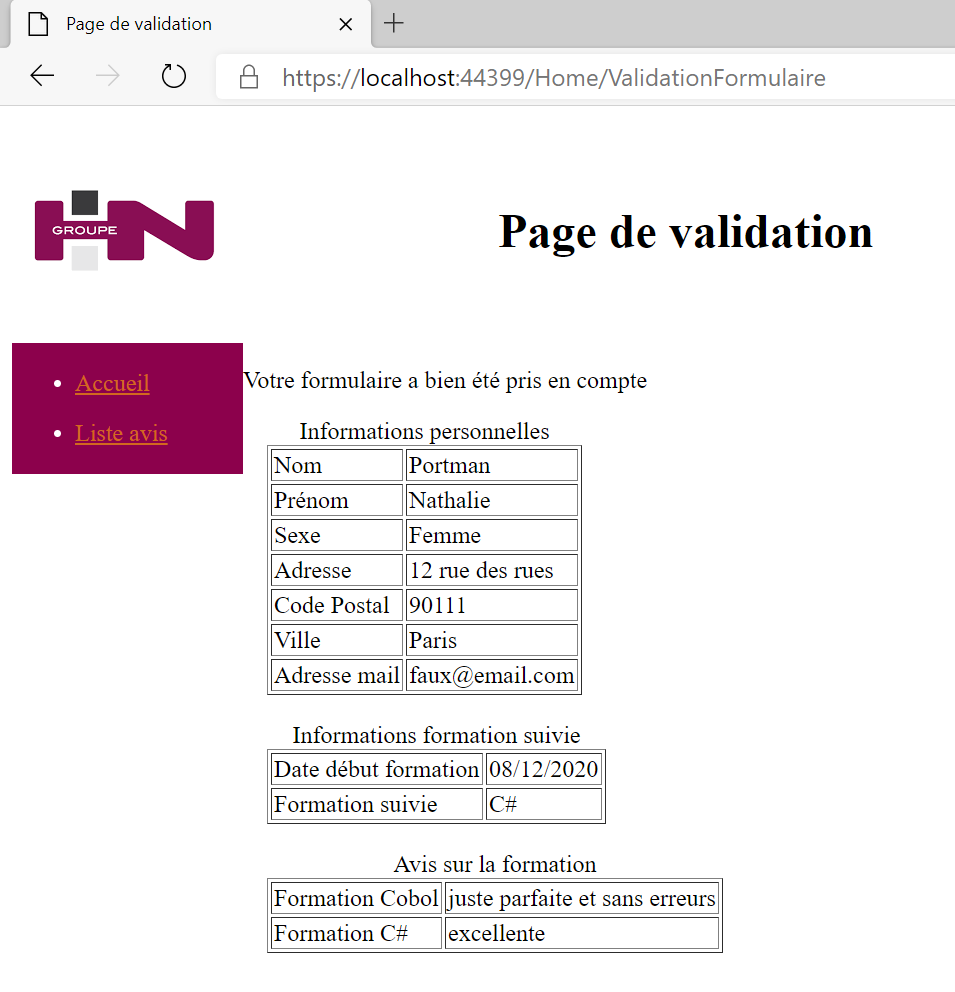
([Retour paragraphe « Formulaire »](#_Formulaire))



## Annexe 3 : Visuel page de validation

([Retour paragraphe « Principe »](#_Principe))

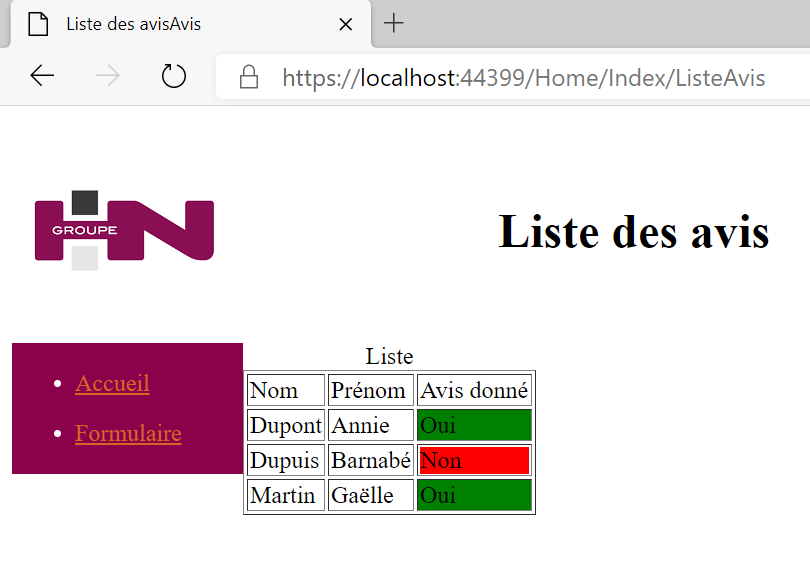
([Retour paragraphe « Page de validation »](#_Page_de_confirmation))



## Annexe 4 : Visuel liste

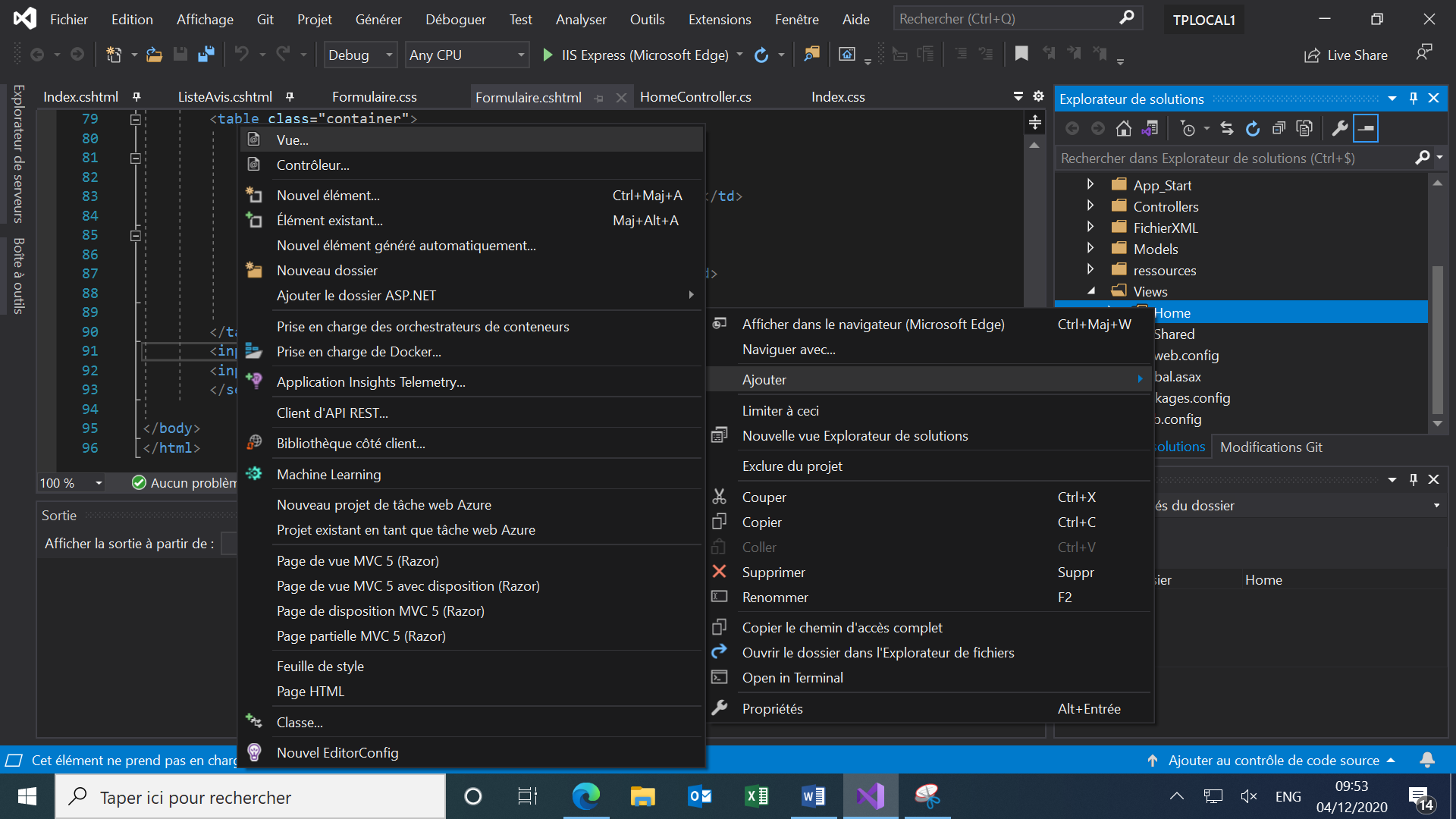
([Retour paragraphe « Principe »](#_Principe))

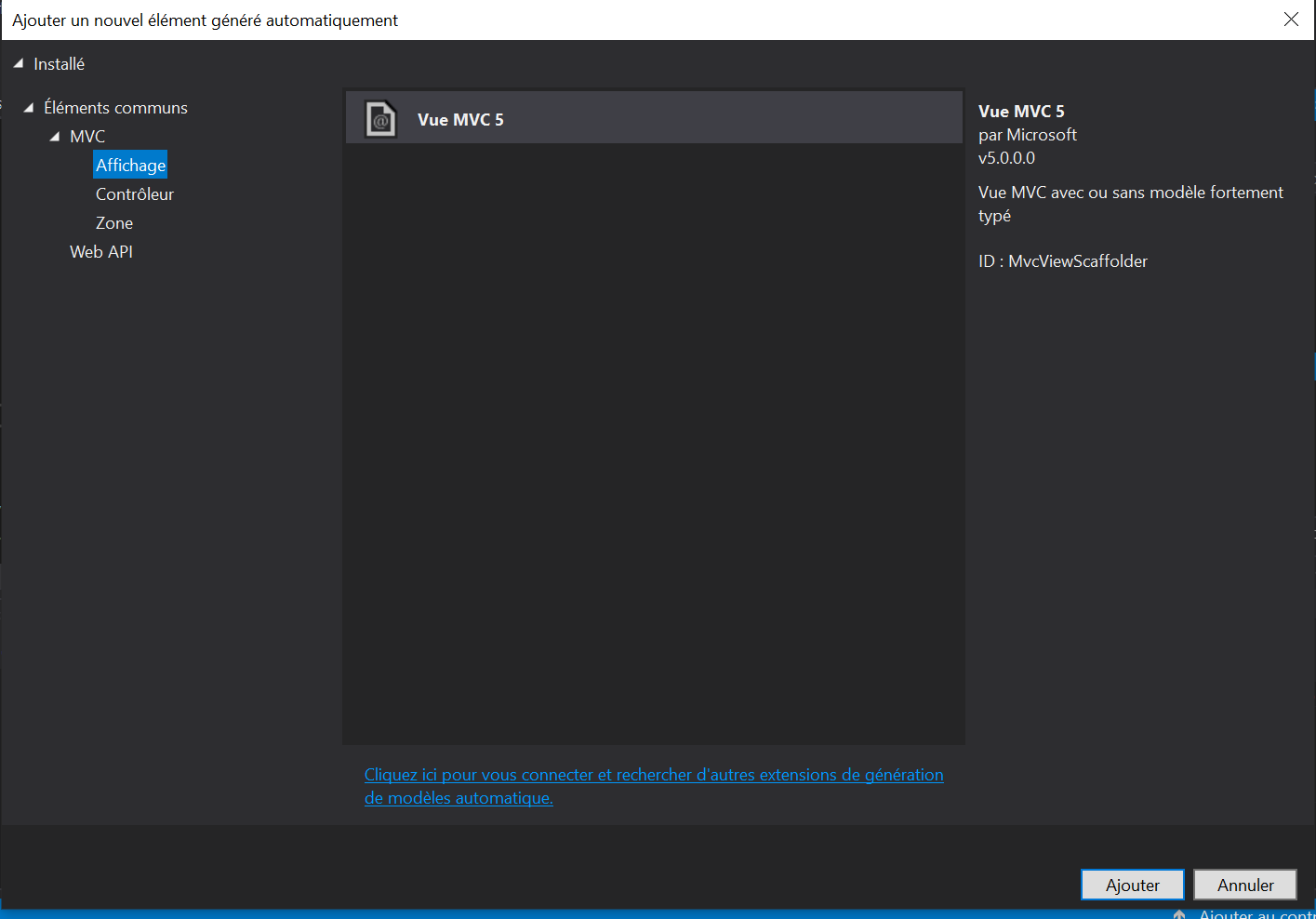
([Retour paragraphe « Page liste »](#_Page_Liste_1))

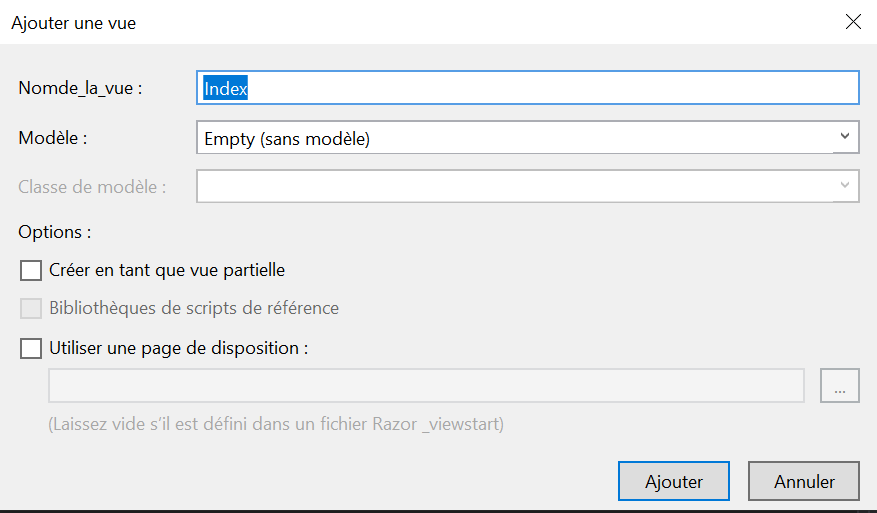


## Annexe 5 : Solution Explorer

([Retour paragraphe](#_1ère_Etape_:))

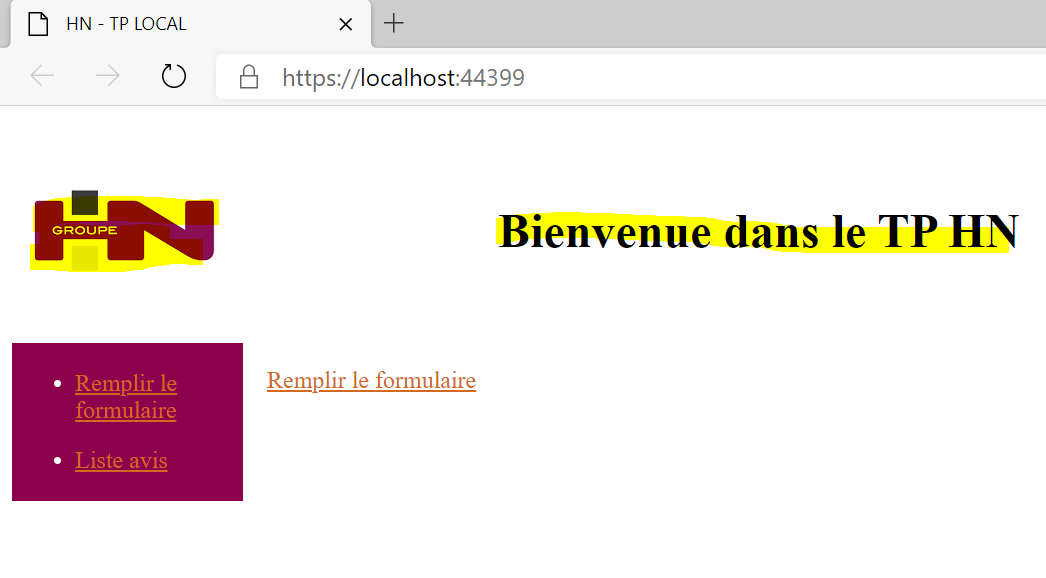






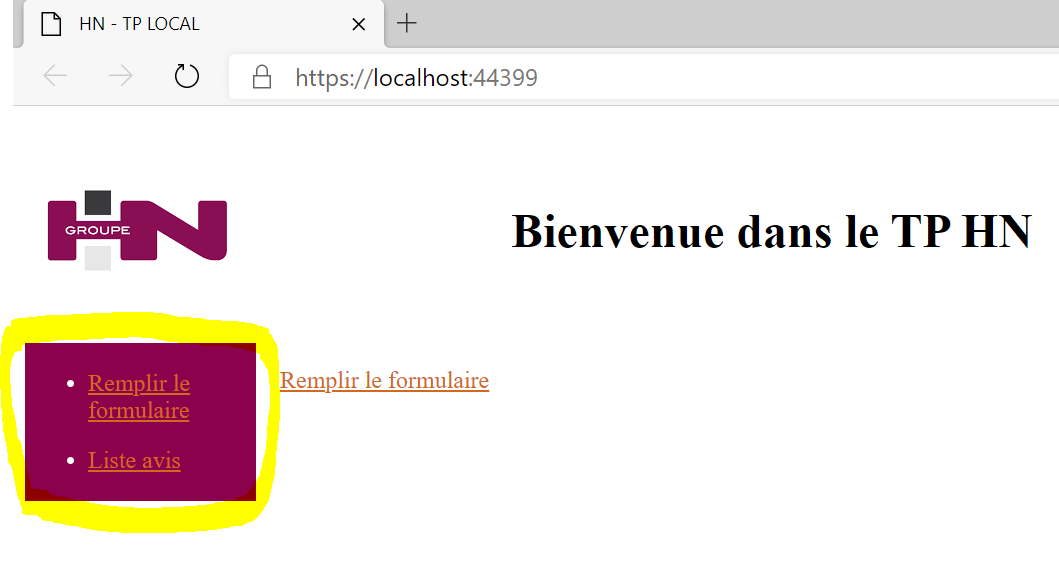
## Annexe 6 : Affichage du logo et du titre

([Retour paragraphe](#_1ère_Etape_:))



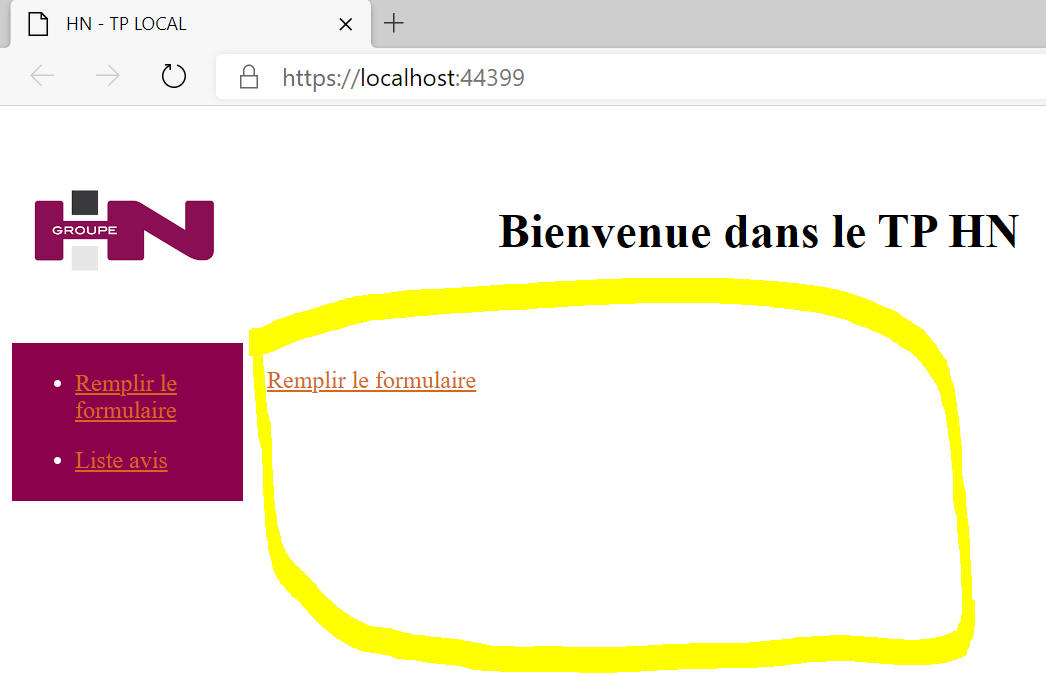
## Annexe 7 : Création du menu

([Retour paragraphe](#_2ème_Etape_:))



## Annexe 8 : Création du corps de page

([Retour paragraphe](#_3ème_Etape_:))



## Annexe 9 : Index.cshtml

([Retour paragraphe](#_1ère_Etape_:))

<!DOCTYPE html>

<html>

<head>

<title>HN - TP LOCAL</title>

<link rel="stylesheet" href="~/ressources/Index.css"/>

</head>

<body>

<header>

<img src="~/ressources/GroupeHN.png" class="image" alt="GroupeHN" />

<h1>Bienvenue dans le TP HN</h1>

</header>

<nav>

<ul>

<li><a href="/Home/Index/Formulaire">Remplir le formulaire</a></li>

<li><a href="/Home/Index/ListeAvis">Liste avis</a></li>

</ul>

</nav>

<section>

<p>

<a href="/Home/Index/Formulaire">Remplir le formulaire </a>

</p>

</section>

</body>

</html>

## Annexe 10 : Index.css

([Retour paragraphe](#_1ère_Etape_:))

.image {

width: 150px;

height: 150px;

display: inline-block;

vertical-align: middle;

}

h1 {

margin-right: 170px;

margin-left: 170px;

text-align: center;

display: inline-block;

vertical-align: middle;

}

nav {

float: left;

width: 150px;

border: 2px solid #8c014c;

background-color: #8c014c;

color: white;

}

li {

margin-top: 10px;

margin-bottom: 15px;

}

a {

color: chocolate;

}

section {

margin-left: 170px;

}

## Annexe 11 : Récupération de la liste des avis

[(Retour paragraphe)](#_3ème_Etape_:_1)

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Web;

using System.Xml;

namespace TPLOCAL

{

public class ListeAvis

{

/// <summary>

/// Fonction permettant de récupérer la liste des avis contenus dans un fichier XML

/// </summary>

/// <param name="fichier">chemin du fichier</param>

public List<Avis> GetAvis(string fichier)

{

// On instancie la liste vide

List<Avis> listeAvis = new List<Avis>();

// Création d'un objet de type XMLDocument permettant de récupérer les données du fichier physique

XmlDocument xmlDoc = new XmlDocument();

// Lecture du fichier à partir d'un objet StreamReader

StreamReader streamDoc = new StreamReader(fichier);

string dataXml = streamDoc.ReadToEnd();

// Chargement des données dans le XmlDocument

xmlDoc.LoadXml(dataXml);

// Récupération des noeuds pour les passer en objet Avis puis ajout à la liste 'listeAvis'

// On boucle sur chaque noeud de type XmlNode ayant pour chemin "root/row" (cf structure du fichier xml)

// La méthode SelectNodes permet de récupérer tous les noeuds ayant le chemin indiqué

foreach (XmlNode node in xmlDoc.SelectNodes("root/row"))

{

// Récupération des données dans les noeuds fils

string nom = node["Nom"].InnerText;

string prenom = node["Prenom"].InnerText;

string avisdonne = node["Avis"].InnerText;

// Création de l'objet Avis à ajouter à la liste des résultats

Avis avis = new Avis

{

Nom = nom,

Prenom = prenom,

AvisDonne = avisdonne

};

// Ajout de l'objet à la liste

listeAvis.Add(avis);

}

// On retourne la liste formée par le traitement à la méthode appelante

return listeAvis;

}

}

// .: Info :.

// Cette classe peut être extraite dans une nouvelle page C# mais dans le cadre du TP elle peut être laissé dans la même page

// Il faut éviter le plus possible d’avoir une même page avec plusieurs classes à l'intérieur.

// Même si cela fonctionne, cela peut compliquer la lisibilité du code et, à terme, la maintenance

/// <summary>

/// Objet regroupant les données liés aux avis

/// \nPeut être modifiée

/// </summary>

public class Avis

{

/// <summary>

/// Nom de famille

/// </summary>

public string Nom { get; set; }

/// <summary>

/// Prénom

/// </summary>

public string Prenom { get; set; }

/// <summary>

/// Avis donné (Valeurs possibles : O ou N)

/// </summary>

public string AvisDonne { get; set; }

}

}

## Annexe 12 : autocomplétion du chemin d’un src :

