Table des matières

Suivi des documents :

Version	Action	Personne	Date
La version 1.0	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 pièce : Formulaire – Étape 5	Louis TEMPELAERE	17/02/2021
V1.6	Modification du document pour la mise à jour du projet en . Net6	Nicolas TRAN	

Objet du document

Ce document décrit un travail pratique dédié aux salariés formés à la double compétence et destinés notamment à évoluer sur les sites d'IE du CM-CIC. Il résulte du constat que la partie LOCAL(C#) de la formation à double compétence délivrée jusque-là par HN Institut n'est pas en adéquation avec les travaux demandés ultérieurement au client.

Principe

Le TP vise à simuler une application permettant l'évaluation d'une formation. Cette application sera structurée comme suit :

- Une page d'accueil (Annexe 1)
- Une page d'information sur le formulaire (annexe 2)
- Une page de validation des informations saisies sur le formulaire (Annexe 3)
- Une page montrant la liste des avis restants (Annexe 4)

Besoins

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

- Ce document de spécification expliquant les travaux à effectuer.
- 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 noms, prénom et indicateur O/N afin de constituer la liste figurant sur la « Page Liste » : « DataAvis.xml »
- Une solution Visual Studio entièrement codée et commentée. Il sera consulté en cas de blocage.

Spécifications

Page d'accueil

Afin de commencer sereinement, nous vous proposons de vous accompagner pas à pas dans l'élaboration de cette première page (Annexe 1).

Sur cette page, vous pratiquerez :

- Pour mettre en forme une page Web HTML
- Pour créer un menu
- Pour créer des liens de navigation entre 2 pages

Dans un premier temps, il est nécessaire de créer la solution qui servira de base à tous les développements. Récupérez la solution fournie avec le TP. Il s'agit d'une solution MVC (Vue Controller Model) APS.NET, dans laquelle le contrôleur et le routage sont déjà créés pour vous faire gagner du temps. En effet, vous n'aurez pas à coder cette partie au sein du CIC. Ouvrez TPLOCAL1.sln pour ouvrir cette solution. Le choix technique de la VMC 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 qui est vu sur l'écran du navigateur Web. Il sera composé de pages cshtml, ce qui permet à la fois le code html et le code c#.
- Enfin, le responsable de traitement définit la logique du site web, par exemple les liens entre les pages, les contrôles des données saisies par l'utilisateur, les appels aux bases de données (non présentes dans ce TP), les lectures de fichiers...

Étape 1 : Affichage du logo et du titre

Le but de cette partie est d'aligner le logo et le titre (Annexe 6).

Nous allons maintenant créer la première page d'affichage cshtml. Pour ce faire, dans la fenêtre de l'Explorateur de solutions (Annexe 5), cliquez avec le bouton droit de la souris sur le dossier Affichage/Accueil, puis Ajouter et Afficher. Cliquez sur Ajouter. Nommez la vue Index, puis ajoutez-la.

Cet ajout crée une page acshtml, 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...
- Les corps <body> : partie principale de la page.

DANS L'EN-TÊTE : Tout d'abord, nous allons nommer la page avec la balise <title> : <title>HN - TP LOCAL</title>Cette balise vous permet d'avoir le nom de la page qui apparaît dans l'onglet internet.

DANS LA PARTIE PRINCIPALE DE LA PAGE :

Pour mettre un titre sur la page, vous devez utiliser la balise <header> qui gère l'en-tête d'une page html :

```
<header>
  <h1>Bienvenue chez TP HN</h1>
</en-tête>
```

La balise <h1> permet de définir un titre de niveau 1 et sera utilisée plus tard dans la page. CSS.

Maintenant que le titre est renseigné, vous devez mettre le logo. Pour ce faire, dans le dossier solution, dans le sous-dossier « resources » , mettez le fichier contenant l'image, en vous assurant que le nom du fichier est GroupHN.png. Si le fichier se trouve dans le dossier mais n'est pas visible dans la solution, ajoutez un élément existant dans le dossier « ressources » et choisissez le fichier dans l'Explorateur Windows.
Sur la page d'index. cshtml, vous devez ajouter une balise <img> :

```
<img src="~/resources/GroupeHN.png » class="image » alt="GroupeHN » />
```

L'attribut src indique l'emplacement du fichier et alt indique ce que l'image contient. Le ~ indique qu'il s'agit d'un chemin, et l'autocomplétion du code propose normalement le chemin (Annexe 11). Pour utiliser l'autocomplétion, lorsqu'elle apparaît, utilisez les flèches du clavier et la touche Entrée ou la souris.

Exécutez le débogage en tapant F5 (sur la page cshtml). Si le code ne se compile pas, commentez les parties qui boguent (par exemple, la méthode ValidationForm du contrôleur).

Une ligne est commentée lorsqu'elle a « // » 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, vous devez créer une page. CSS. Le. CSS permet de formater les pages HTML CS.

Pour créer a. CSS, ainsi que pour la création d'une page CSHTML, cliquez sur le dossier « Ressources », puis ajoutez un nouvel élément, choisissez Web/feuille de style. Elle sera nommée Index.css Tout d'abord,

il faut s'occuper de l'image, en ajoutant le code suivant dans la page CSS :

```
.image
{
   Largeur : 150px ;
   hauteur : 150px ;
}
```

Nous venons de créer la classe image. Les attributs width et height sont utilisés pour redimensionner l'image en largeur et en hauteur.

Pour le titre, il faut mettre dans la page CSS :

```
h1
{
   marge-droite : 170px ;
   marge-gauche : 170px ;
   text-align : centre ;
}
```

margin-right et margin-left sont utilisés pour définir une marge à droite et à gauche du texte. text-align est utilisé pour définir l'alignement du texte. Pour lier la page css à la page html, vous devez créer un lien entre ces 2 pages dans la page cshtml. Pour ce faire, utilisez une balise <link> dans la section <head> :

```
<link rel="feuille de style »  href="~/ressources/Index.css"/>
```

À ce stade, le titre et le logo sont formatés, mais lorsque vous exécutez le débogage, ils ne sont toujours pas alignés.

Pour aligner le logo et le titre, ajoutez 2 éléments dans les classes image et h1 : display :inline-block ;

```
   alignement vertical :milieu ;
```

La première ligne sert à faire un bloc des 2 éléments et la seconde ligne sert à aligner le bloc.

Étape 2 : Création du menu

Le but de cette partie est de vous montrer comment créer un menu (Annexe 7).

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

Le code d'un menu est le suivant :

```
< >
   <ul>
      <li><a href="#">Remplissez le formulaire</a></li>
      <li><a href="#">Avis list</a></li>
   </UL>
</NAM>
```

Explication des balises :

- <nav> : lien de navigation principal, utilisé entre autres pour le menu principal,
- : Liste à puces,
- : Élément dans une liste à puces,
- <a> : Balise pour les liens hypertextes, elle doit être associée à l' attribut href pour indiquer vers quelle page le lien doit mener. Ici, il y a un « # » car les pages ne sont pas encore créées, il sera remplacé plus tard.

On peut voir grâce au debug que le menu est apparu sur la page, mais ce n'est pas très joli. Pour améliorer le visuel, vous devez modifier la page CSS :

```
Nav
{
  Largeur : 150px ;
  bordure : 2px solide #8c014c ;
  couleur d'arrière-plan : #8c014c ;
  couleur : blanc ;
}
Li
{
  marge-haut : 10px ;
  marge-bas : 15px ;
} a

{
  couleur : chocolat ;
}
```

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

Explication de l'attribut :

- width : largeur du bloc en pixels (possibilité de le faire en pourcentage)
- Border : Définition de la bordure du bloc, ici la structure est l'épaisseur de la bordure (2 px) le type de bordure (solide) la couleur (#8c014c)
- background-color : couleur de l'arrière-plan du bloc, soit couleur (jaune, bleu ...) ou référence (#8c014c)
- color : couleur du texte
- margin-top : marge au-dessus de la ligne
- margin-bottom : marge en dessous de la ligne

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

Étape 3 : Création du corps de la page

Le but de cette étape est de voir comment remplir et mettre un lien dans le corps de la page (Annexe 8).

Comme précédemment, les modifications sont apportées dans la partie <corps> de la page cshtml.

Pour créer le corps de la page, vous devez ajouter le code suivant :

```
<section>
  < >
    <a href="#">Remplissez le formulaire </a>
  </p>
```

```
</section>
```

Explication des balises :

- <section> : section de page, utilisée pour regrouper le contenu,
- <p> : paragraphe.

Regardez à quoi ressemble votre page maintenant que tous les éléments sont 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, vous devez modifier la page CSS. Dans la classe nav, il faut ajouter :

```
float : left ;
```

La propriété float sert à faire flotter un objet, elle peut être utilisée pour un menu ou une image et peut prendre 2 valeurs qui sont à gauche ou à droite.

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

```
section
{
  marge-gauche : 170px ;
}
```

Étape 4 : Etablissement des liens

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

Tout d'abord, remplacez simplement « # » par « /Home/Index/Form » dans les balises Remplissez le formulaire . Ce chemin d'accès signifie que le contrôleur s'appelle HomeController, et plus précisément sa fonction Index, à laquelle le paramètre Form est fourni. Le contrôleur n'a pas besoin d'être modifié.

Ensuite, vous devez créer la page Form.cshtml. Attention, vous devez créer cette vue dans le dossier Views/Shared pour que le routage fonctionne correctement (si cela vous intéresse, vous trouverez un lien explicatif à la fin de l'affirmation cependant vous n'aurez pas besoin d'en connaître la raison une fois sur place).

Le code View(id) de retour ; liens vers la page Formation. La méthode View vous permet de vous référer à la vue souhaitée, ici sans autres paramètres. Par la suite, nous utiliserons également la forme surchargée de la méthode View avec la page à appeler et un modèle de données.

Lorsque vous cliquez sur le lien nouvellement défini, vous arrivez sur une page blanche, ce qui est normal car elle n'est pas encore encodée.

Lors de la création de la page AvisList.cshtml, vous pourrez configurer son lien dans le menu.

Pour voir le code de la page Index. voir l'annexe 9 et pour la page d'index code.css voir l' annexe 1-0.

Forme

À partir de cette partie, vous serez moins guidé. Si vous rencontrez des difficultés, cherchez la solution sur internet, puis si vous n'êtes pas en mesure de le faire, consultez la solution fournie par HN.

Il s'agit de créer une page qui permettra à l'utilisateur de renseigner ses données personnelles ainsi que son avis sur une formation (Annexe 2).

Sur cette page, vous pratiquerez :

- Pour créer une liste déroulante
- De paramétrer différents contrôles en fonction des informations attendues dans chacun des champs
- Pour gérer les messages d'erreur

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

Étape 1 : Affichage du logo et du titre

Voir précédemment (page d'accueil de la 1ère étape). Attention, le titre est différent de celui de la page d'accueil.

Étape 2 : Création du menu

Voir précédemment (page d'accueil de la 2ème étape). Attention, les rubriques sont différentes de celles de la page d'accueil.

3ème étape : Création de la table de saisie « Informations personnelles »

Comme son nom l'indique, l'utilisateur doit être capable de saisir dans cette table plusieurs données qui lui sont propres.

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

Voici les différents champs que la table doit avoir avec ses restrictions/contrôles :

Colonne de gauche (titres)	**Colonnedu titre (type de champs, contrôles possibles)**
Informations personnelles	
Nom	Champ de saisie
Prénom	Champ de saisie
Genre	- Liste déroulante avec 4 choix : • Sélectionnez un sexe • Homme • Femme • Autre - Par défaut, le champ doit être rempli avec « Sélectionner un sexe »
Adresse	Champ de saisie
Code postal	- Champ de saisie - Contrôle : doit être sur 5 caractères numériques grâce à une expression régulière
Ville	Champ de saisie
Adresse courriel	- Champ de saisie - Contrôle du format à mettre en œuvre

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

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

Les données doivent être stockées dans une classe dans le dossier Models.

4ème étape : Création de la table de saisie « Informations de formation suivie »

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

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

Voici les différents champs que la table doit avoir avec ses restrictions/contrôles :

Colonne de gauche (titres)	**Colonne de droite (type de champs, contrôles éventuels)**
Informations sur la formation	
Date de début de la formation	- Champ de saisie de date - La date doit être inférieure au 01/01/2021
Type de formation	- Champ de saisie - Liste déroulante avec 4 choix : • Sélectionnez un cours • Formation Cobol • Formation par objet • Formation à double compétence - Par défaut, le champ doit être rempli avec « Sélectionner une formation »

Etape 5 : Création du tableau de saisie « Notice de formation »

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

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

Voici les différents champs que la table doit avoir avec ses restrictions/contrôles :

Colonne de gauche (titres)	**Colonne de droite**
Examens de la formation	
Cobol Formation	- Champ de saisie
C# Formation	- Champ de saisie

Pour les contrôles, il est possible de les faire en HTML ou dans la déclaration de variables dans une classe (voir Required et _StringLength Attribute dans MVC - Dot Net Tutorials et Regular Expression Attribute dans MVC - Dot Net Tutorials) ou enfin dans le contrôleur en cliquant sur le bouton Valider.

Quelques explications :

- En HTML, dans la balise d'entrée, il est possible de définir dans le champ « type » le type des données. Selon le type, le visuel est variable. Par exemple, <input type="date"/> donne un calendrier. Ce type de contrôle est très pratique mais pas très flexible, car il existe un nombre limité de types.
- Dans la déclaration de variable, Required peut être ajouté au-dessus de la variable d'attribut pour vérifier que les données sont renseignées :

```
[Obligatoire]
public string Formation { get ;  ensemble ; }
```

Il est également possible d'effectuer des vérifications plus poussées de cette manière, par exemple sur la taille des données à saisir par l'utilisateur, avec des expressions régulières, ou la plage de données de la date.

- Enfin, il est également possible d'effectuer des contrôles dans la fonction appelée par un bouton. Par exemple, en supposant que le modèle où se trouvent les données appelle modelv, nous pouvons renvoyer une erreur si les données d'adresse sont vides ou pas assez longues :

```
Si (modelv. Adresse == null || modelv. Adresse.Longueur < 5)
{
        ModelState.AddModelError(«  », « adresse trop courte ») ;
}
```

Et pour afficher les erreurs, ajoutez à la page cshtml au niveau du champs concerné, une balise de texte (p, span ou autres…) avec l'attribut **asp-validation-for**. Nous avons ensuite le visuel suivant :

Welcome to TP HN

- Home
- Opinions List

Form

Personal information

Name	your name	The Name field is required.
Forename	your firstname	
Gender	Select your gender	
Address		
Zip code	example : 94700	
Town		
Email address	yourmail@server.domain	

Étape 6 : Créer un bouton

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

- Doit générer au moins un message d'erreur si au moins un des contrôles définis sur la page est KO.
- Doit afficher la page de validité correctement remplie au cas où tous les contrôles de la page seraient OK.

Pour information, nous pouvons utiliser le champ action de la balise form pour appeler la méthode ValidationForm du HomeController pour ce bouton. Dans cette méthode, nous allons vérifier les données et envoyer les données en tant que paramètres à la page de droite avec le modèle.

Page de validation

Il s'agit de créer une page qui permettra à l'utilisateur de consulter les informations qu'il vient de renseigner sur la page du formulaire (Annexe 3).

Sur cette page, vous pratiquerez :

- Pour transmettre des données d'une page à une autre

Cette page doit comporter les mêmes éléments que la page du formulaire avec les différences suivantes :

- Dans la première ligne du corps de la page doit figurer la mention « Votre formulaire a été pris en compte », centrée sur la page.
- Les colonnes à droite des tableaux doivent être remplies avec les informations saisies par l'utilisateur sur la page du formulaire. Pour information, pour afficher les données d'une classe C#, dans la page cshtml, nous définissons la classe en haut de la page comme suit :
  `@model TPLOCAL1. Models.FormModel` (avec le chemin d'accès relatif TPLOCAL1. Models et FormModel le nom de la classe de modèle).
  On note avec le signe at les données C# dans les pages cshtml. Ensuite, pour utiliser les données de ce modèle dans la page cshtml, nous écrivons @Model.DataName.
  Le même modèle sera utilisé pour la page de validation que pour la page de formulaire.
- Aucune information ne doit être modifiable par l'utilisateur.
- La date sera affichée au format français (Jour/Mois/Année).

Page de liste

Il s'agit de créer une page qui permettra à l'utilisateur de consulter les informations présentes dans un fichier (Annexe 4).

Sur cette page, vous pratiquerez :

- Pour créer une liste à partir de données présentes dans un fichier
- Pour conditionner la mise en forme d'un champ en fonction de sa valeur

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

Étape 1 : Affichage du logo et du titre

Voir précédemment (page d'accueil de la 1ère étape). Attention, le titre est différent de celui de la page d'accueil.

Étape 2 : Création du menu

Voir précédemment (page d'accueil de la 2ème étape). Attention, les rubriques sont différentes de celles de la page d'accueil.

Étape 3ème : Création de la liste des avis

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

Cette liste sera présentée sous la 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 de la table et la ligne de titre de la colonne. Dans la première colonne apparaîtront les noms des personnes dans le fichier, dans la seconde leur prénom et dans la dernière colonne un haut 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 la table doit contenir :

Colonne 1	**Colonne 2**	**Colonne 3**
Table des étiquettes de titre « Liste »		
Colonne « Nom » de l'étiquette de titre	Libellé du titre : colonne « Prénom »	Libellé : Titre : colonne « Avis donné »
Nom 1	Prénom 1	- Avis donné dans l'information de : • « Oui » si le haut présent dans le fichier source est « Y » • « Non » sinon - Afficher le champ par : • vert si le haut présent dans le fichier source est « O » • rouge sinon
Nom 2	Prénom 2	- Avis donné dans l'information de : • « Oui » si le haut présent dans le fichier source est « Y » • « Non » sinon - Afficher le champ par : • vert si le haut présent dans le fichier source est « O » • rouge sinon
Nom 3	Prénom 3	- Avis donné dans l'information de : • « Oui » si le haut présent dans le fichier source est « Y » • « Non » sinon - Afficher le champ par : • vert si le haut présent dans le fichier source est « O » • rouge sinon

...	...	...

Les données seront extraites du DataNotice file.xml l'aide des classes présentes à l' annexe 11. Le fichier est à placer dans le dossier XML File de la solution.

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

Bonus

Affichage simultané de plusieurs messages d'erreur

L'objectif est le suivant : sur la page du formulaire, en cas de plusieurs erreurs, afficher tous les messages d'erreur correspondants (au lieu d'afficher uniquement le message d'erreur de la première erreur rencontrée).

Dynamisez le menu et l'en-tête

L'objectif serait de :

- Codez une fois le menu et l'en-tête et appelez le code sur chaque page.
- Pour conditionner les informations du menu et de l'en-tête à la page par laquelle ils sont appelés.

Pour plus d'informations sur MVC

Vous pouvez lire le cours d'openclassroom sur Learn ASP.NET MVC - OpenClassrooms pour voir comment fonctionnent le contrôleur et le routage. Cependant, cela n'est pas nécessaire pour travailler au CIC et ne sert qu'à votre connaissance générale.

Annexes

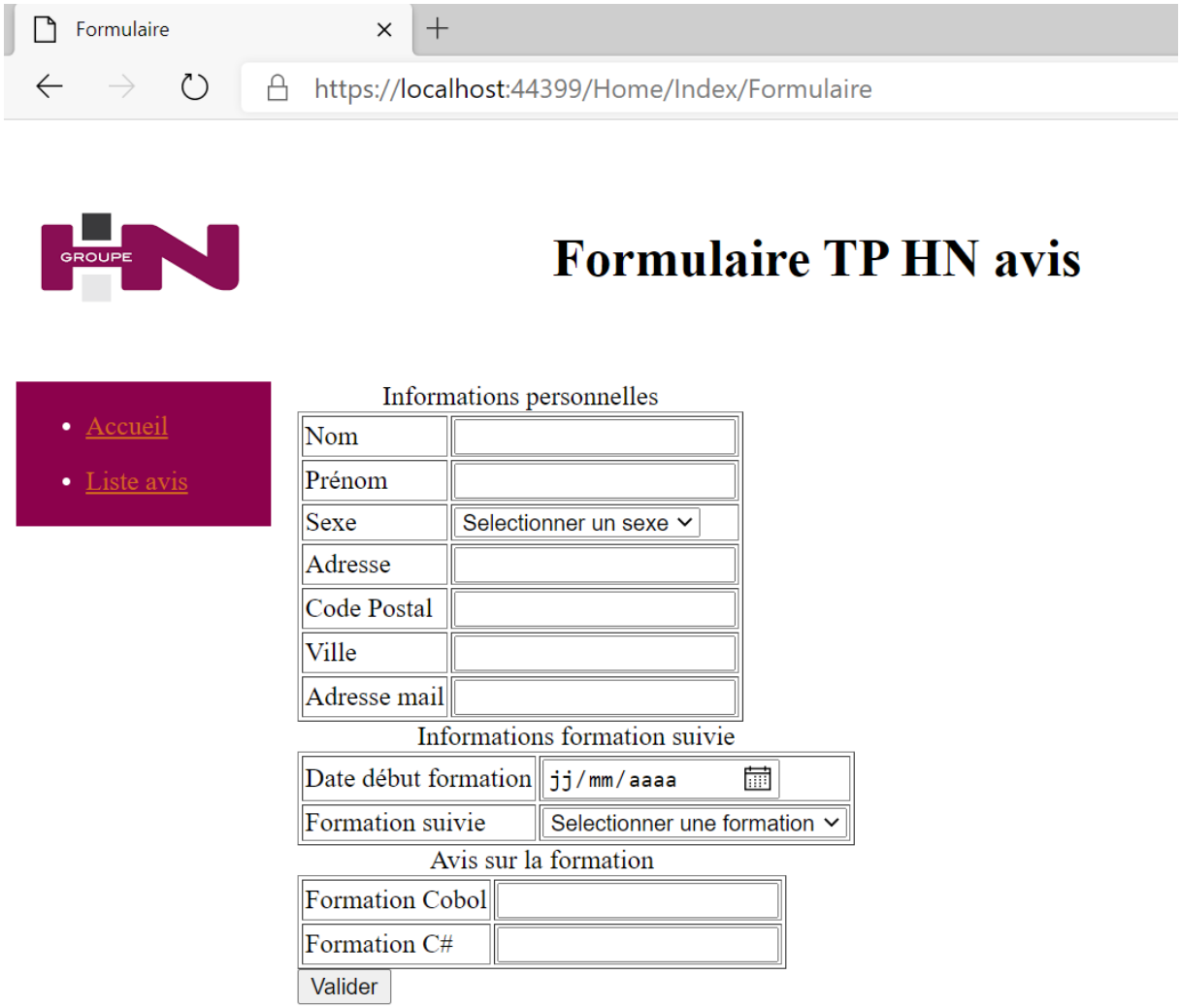
Annexe 1 : Page d'accueil visuelle

(Retour au paragraphe)

Annexe 2 : Forme visuelle

(Retour au paragraphe « Principe »)

(Retour au paragraphe « Forme »)



Annexe 3 : Page de validation visuelle

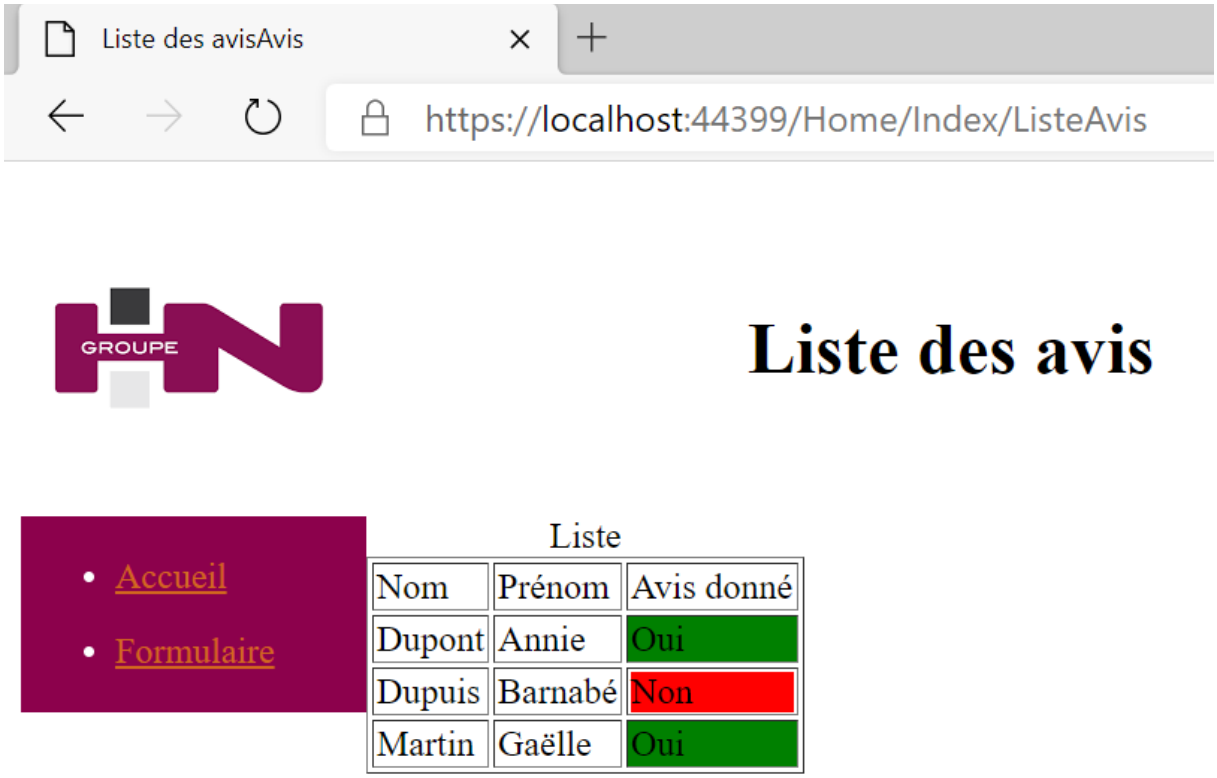
(Retour au paragraphe « Principe »)

(Retour au paragraphe « Page de validation »)

Annexe 4 : Liste visuelle

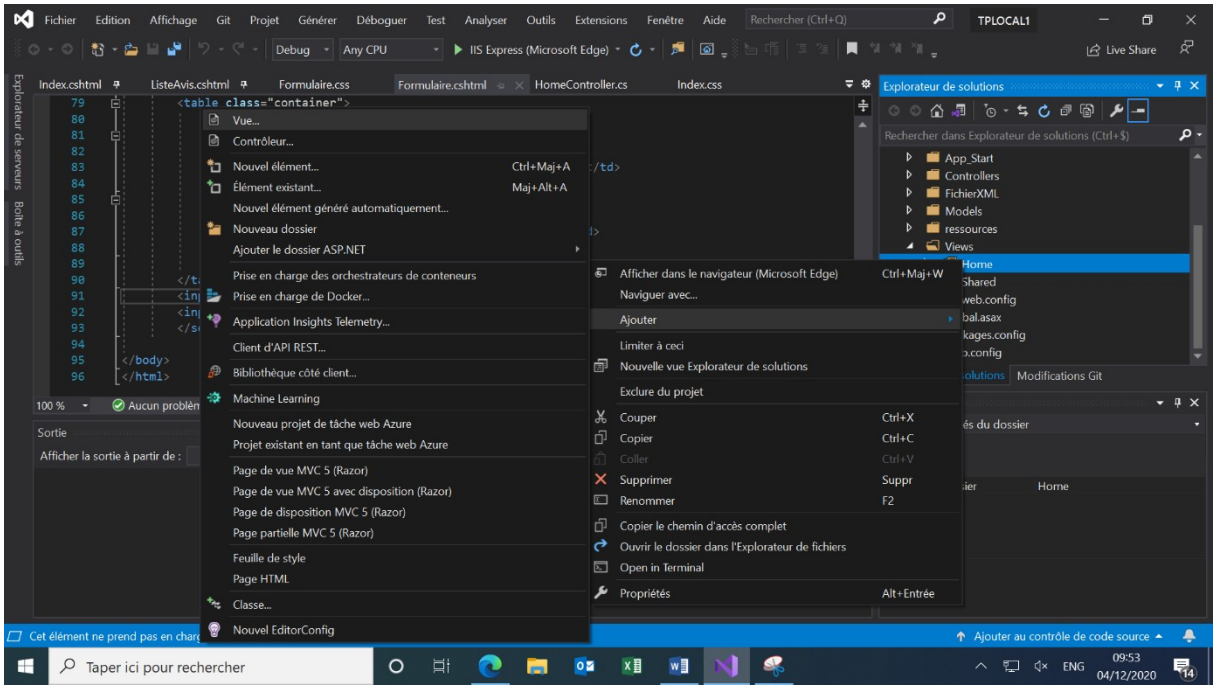
(Retour au paragraphe « Principe »)

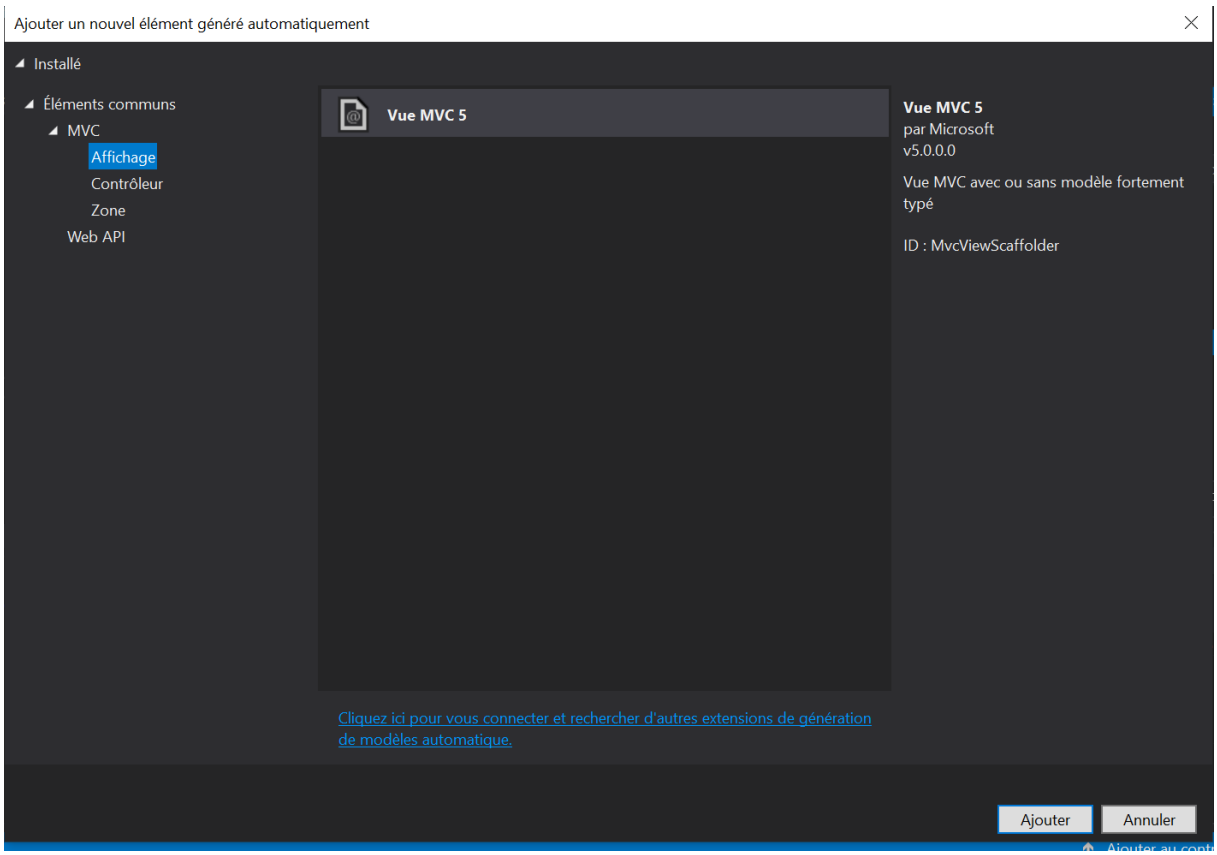
(Retour au paragraphe « Page de liste »)

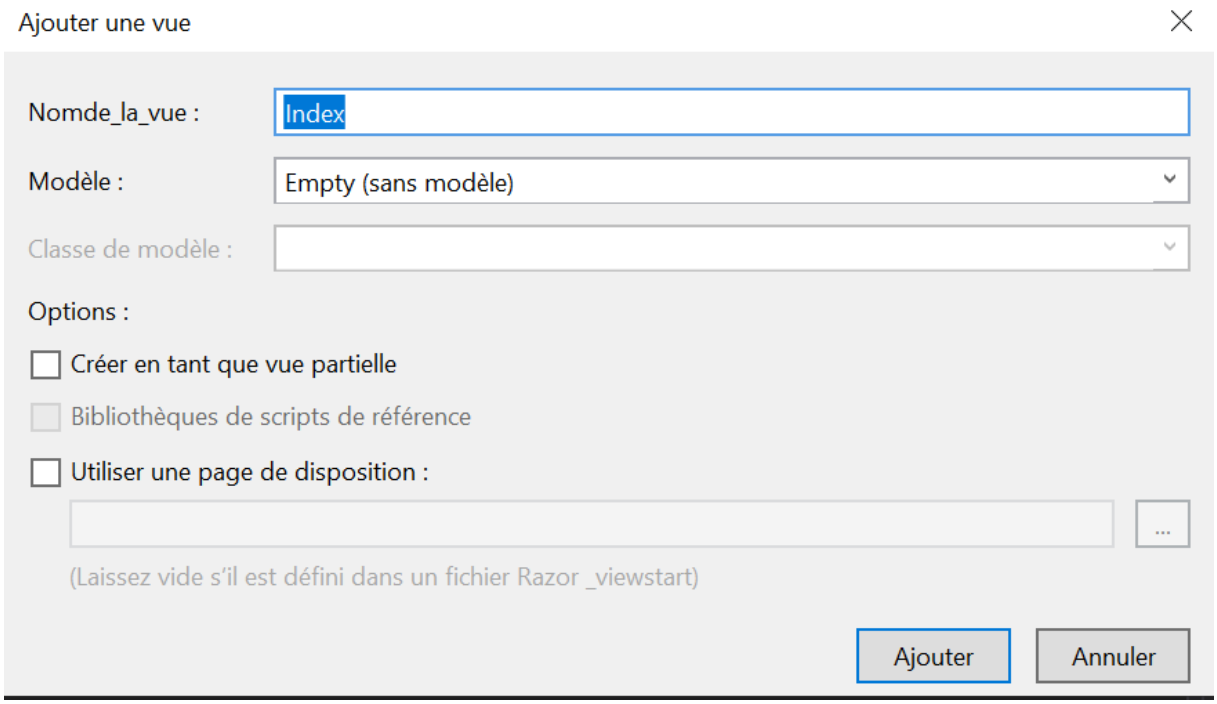


Annexe 5 : Explorateur de solutions

(Retour au paragraphe)





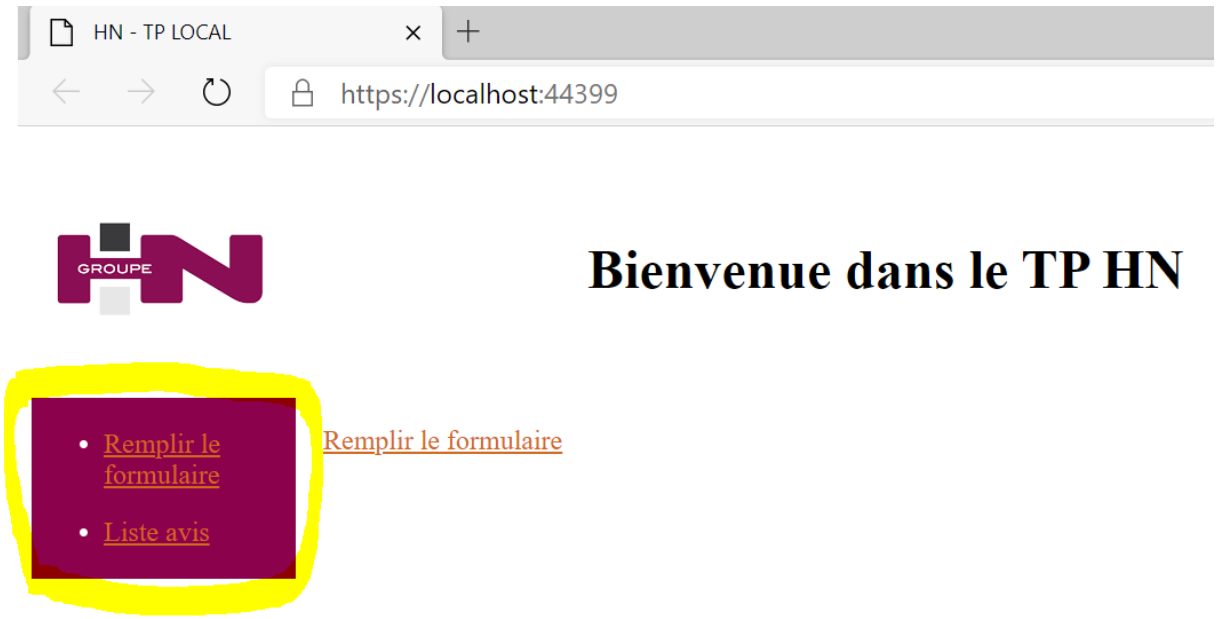

Ajouter une vue
Nomde_la_vue :
Index
Modèle :
Empty (sans modèle)
Classe de modèle :
Options :
Créer en tant que vue partielle
Bibliothèques de scripts de référence
Utiliser une page de disposition :
...
(Laissez vide s'il est défini dans un fichier Razor _viewstart)
Ajouter
Annuler

Annexe 6 : Affichage du logo et du titre

(Retour au paragraphe)

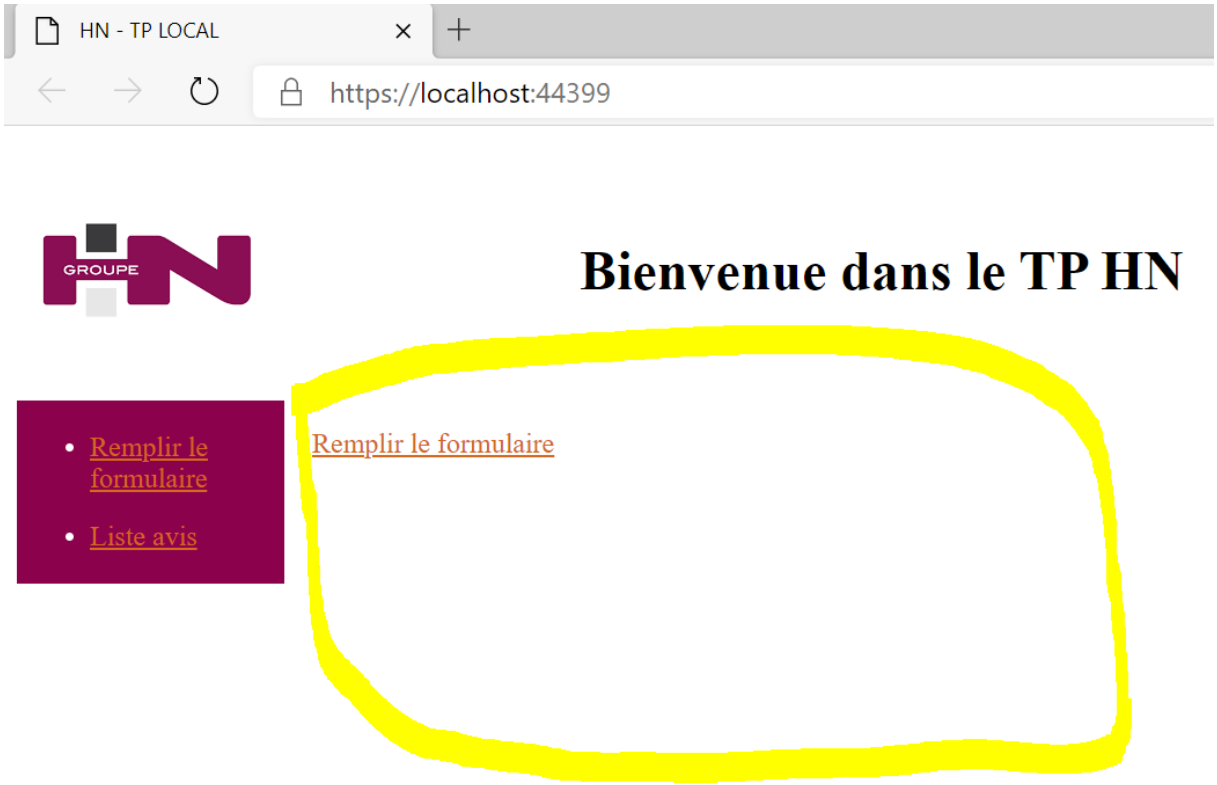
Annexe 7 : Création du menu

(Retour au paragraphe)



Annexe 8 : Création du corps de la page

(Retour au paragraphe)



Annexe 9 : Index. CSHTML

(Retour au paragraphe)

```
< ! DOCTYPE html>
<html>
        <tête>
                <titre>HN - TP LOCAL</titre>
                <link rel="feuille de style » href="~/resources/Index.css"/>
        </tête>
        <corps>
                <en-tête>
                        <img src="~/resources/GroupHN.png » class="image » alt="GroupHN » />
                        <h1>Bienvenue chez TP HN</h1>
                </en-tête>

                < >
                        <ul>
                                <li><a href="/Accueil/Index/Formulaire">Remplir le formulaire</a></li>
                                <li><a href="/Home/Index/AvisList">Avis List</a></li>
                        </UL>
                </NAM>

                <section>
                        < >
                                <a href="/Accueil/Index/Formulaire">Remplissez le formulaire </a>
                        </p>
                /section>
        </corps>
</html>
```

Annexe 10 : Index.css

(Retour au paragraphe)

```
.image {
 Largeur : 150px ;
   hauteur : 150px ;
   affichage : bloc en ligne ;
 alignement vertical : milieu ;
}

h1 {
 marge-droite : 170px ;
 marge-gauche : 170px ;
   text-align : centre ;
   affichage : bloc en ligne ;
 alignement vertical : milieu ;
}

nav {
 flotteur : gauche ;
 Largeur : 150px ;
   bordure : 2px solide #8c014c ;
   couleur d'arrière-plan : #8c014c ;
   couleur : blanc ;
}

li {
 marge-haut : 10px ;
 marge-bas : 15px ;
}

a {
   couleur : chocolat ;
}

section {
marge-gauche : 170px ;
}
```

Annexe 11 : Récupération de la liste des avis

(Retour au paragraphe)

```
Utilisant Système;
en utilisant System.Collections.Generic ;
l'utilisation de System.IO ;
en utilisant System.Linq ;
en utilisant System.Web ;
à l'aide de System.Xml ;

espace de nommage  TPLOCAL
{
        classe publique ListReviews
        {
                <Résumé>
                Fonction permettant de récupérer la liste des avis contenus dans un fichier XML
                </Résumé>
                <param name="file">chemin du fichier</param>
              public List<Reviews> GetAvis(fichier chaîne)
              {
                        Instancier la liste comme vide
                      Liste<Avis> ListReviews = nouveau List<Reviews>() ;
                        Création d'un objet XMLDocument  pour récupérer des données à partir du fichier
              physique
                      XmlDocument xmlDoc = nouveau XmlDocument() ;
                        Lecture du fichier à partir d'un objet StreamReader
                      StreamReader streamDoc = nouveau StreamReader(fichier) ;
                      chaîne dataXml = streamDoc.ReadToEnd() ;
                        Chargement de données dans le document XmlDocument
                      xmlDoc.LoadXml(dataXml) ;

                        Récupération des nœuds pour  les passer en tant qu'objet Notice, puis ajout de ceux-ci
                      à la liste 'ListNotices'
                        On boucle sur chaque nœud de  type XmlNode ayant pour chemin « racine/ligne » (cf
                      structure du fichier xml)
                        La méthode SelectNodes  récupère tous les nœuds avec le chemin d'accès spécifié
                      foreach  (noeud  XmlNode dans xmlDoc.SelectNodes(« root/row »))
                      {
                                Récupération de données dans les nœuds enfants
                              string name = node["Nom"]. InnerText ;
                              string prénom = node["Prénom"]. InnerText ;
                              string avisdonne = node["Avis"]. InnerText ;

                                Création de l'objet de notification à ajouter à la liste des résultats
                              Avis avis = nouvel Avis
                              {
                                 Nom = nom,
                                 Prénom = prénom,
                                 AvisDonne = avisdonne
                              };

                              Ajout de l'objet à la liste
                              listNotice.Add(avis) ;
                      }

                        La liste formée par le traitement est renvoyée à la méthode d'appel
                      liste de retourAvis  ;
              }
```

```
        }

        // . :Info:.
         Cette classe peut être extraite dans une nouvelle page C# , mais dans le cadre du TP, elle peut être laissée
        dans la même page.
        Il faut éviter autant que possible d'avoir la même page avec plusieurs classes à l'intérieur.
        Même si cela fonctionne, cela peut compliquer la lisibilité du code et, en fin de compte, la maintenance.
          <Résumé>
          Regroupement d'objets : données relatives aux avis.
          \n Peut être modifié
          </Résumé>
        public class Avis
        {
          <Résumé>
          Nom de famille
          </Résumé>
        public string Name { get ;  ensemble ; }
          <Résumé>
          Prénom
          </Résumé>
        public string Prenom { get ;  ensemble ; }
          <Résumé>
          Avis donné (Valeurs possibles : O ou N)
          </Résumé>
        public string AvisDonne { get ;  ensemble ; }
        }
}
```

Annexe 12 : autocomplétion du chemin d'un SCR :