Ecole Supérieure Privée
d'Ingénierie et de Technologies

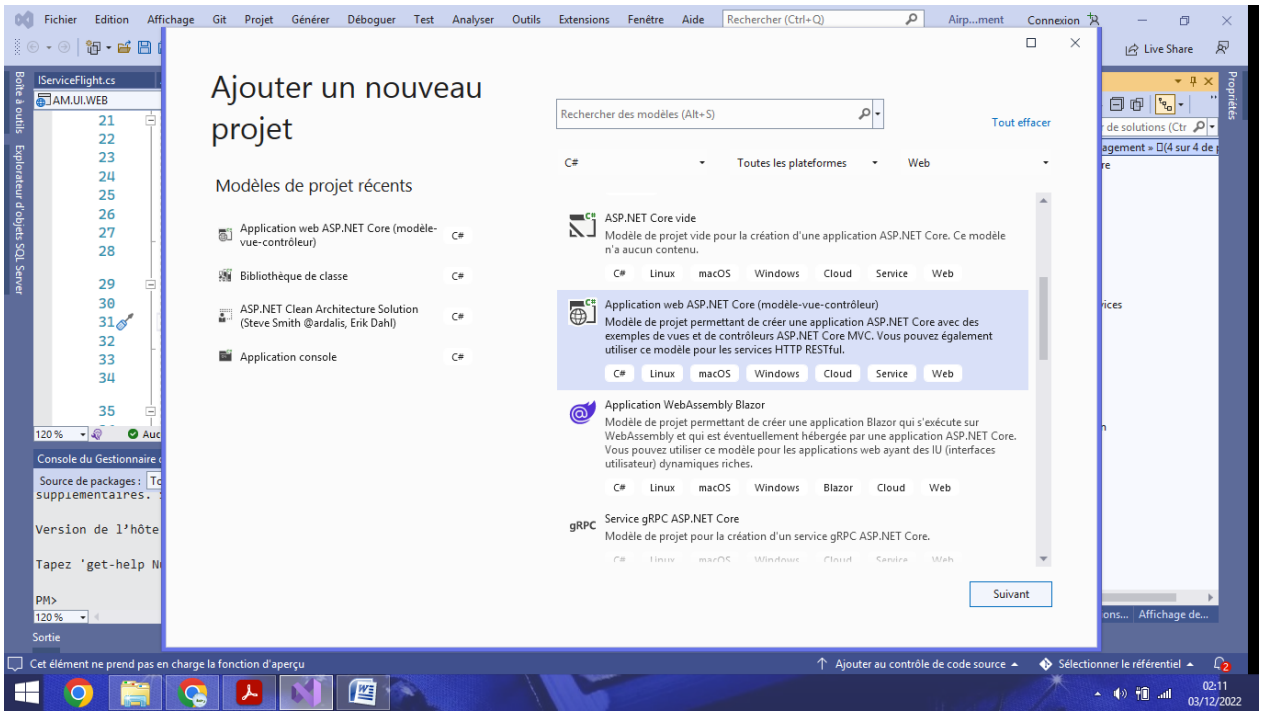
TP Airport Management

Partie 7: ASP.NET MVC

(Correction)

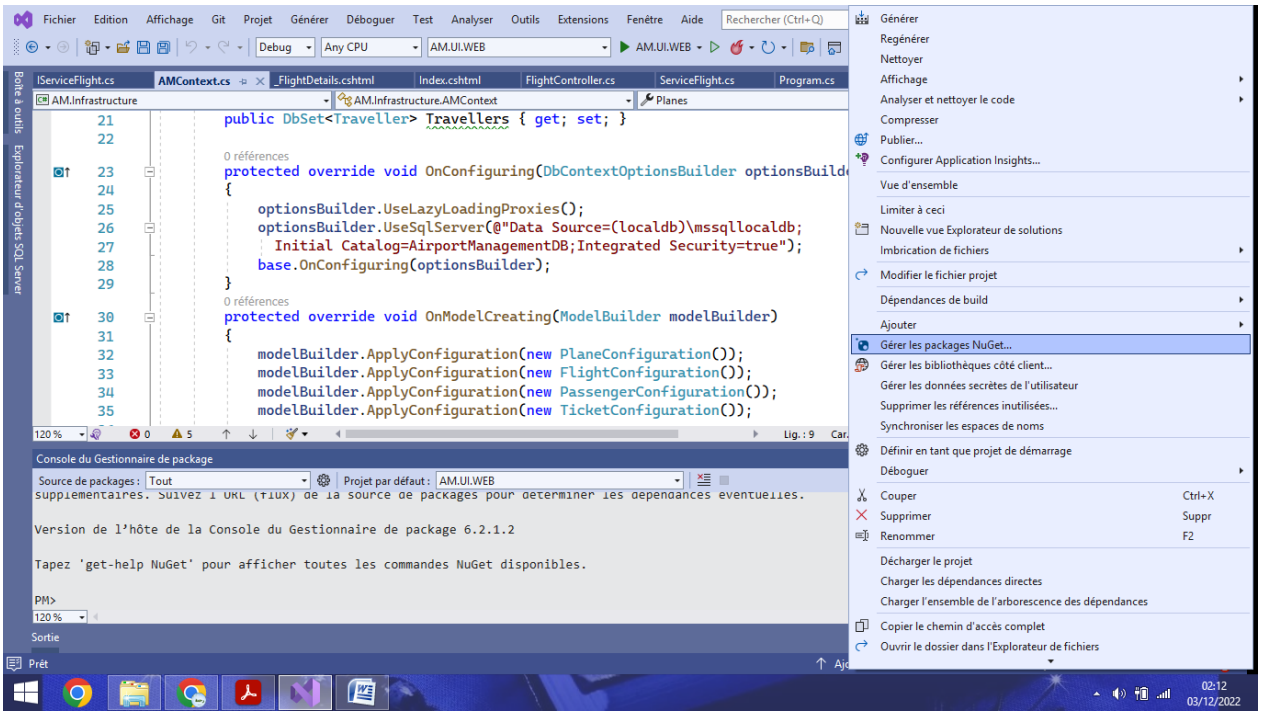
I Création du projet web

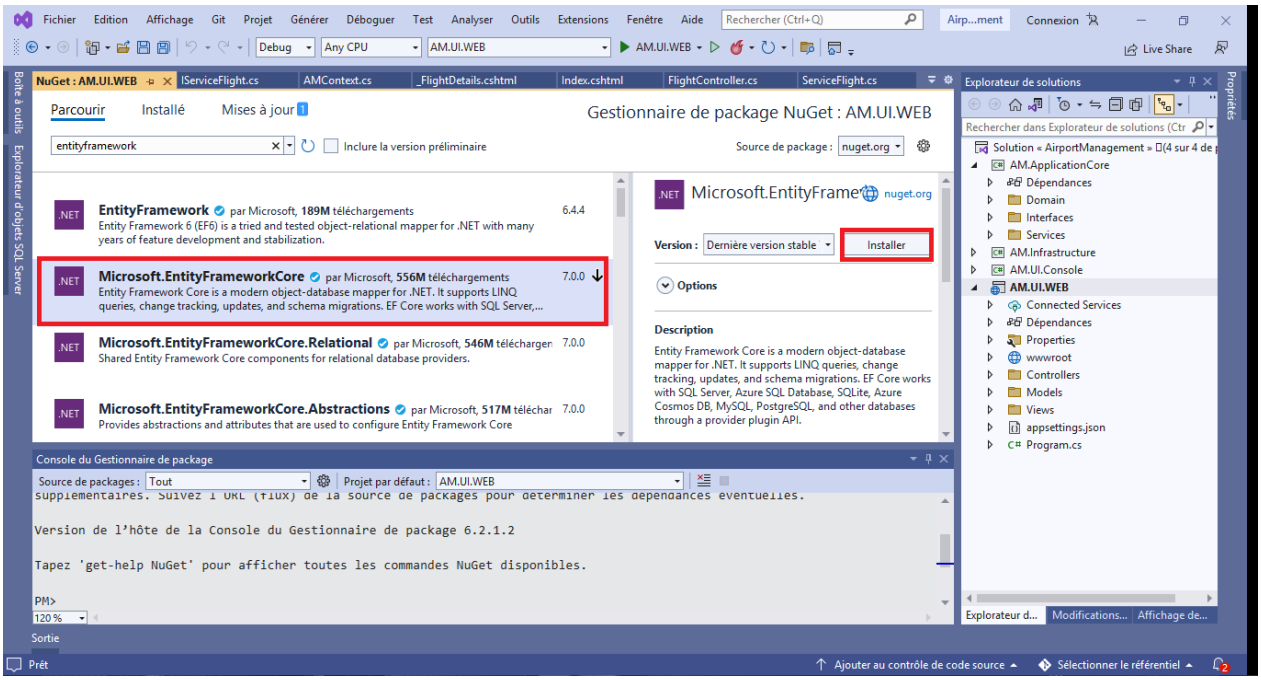
1. Ajouter le projet **AM.UI.WEB** de type Application web **ASP.NET MVC (6.0)**.



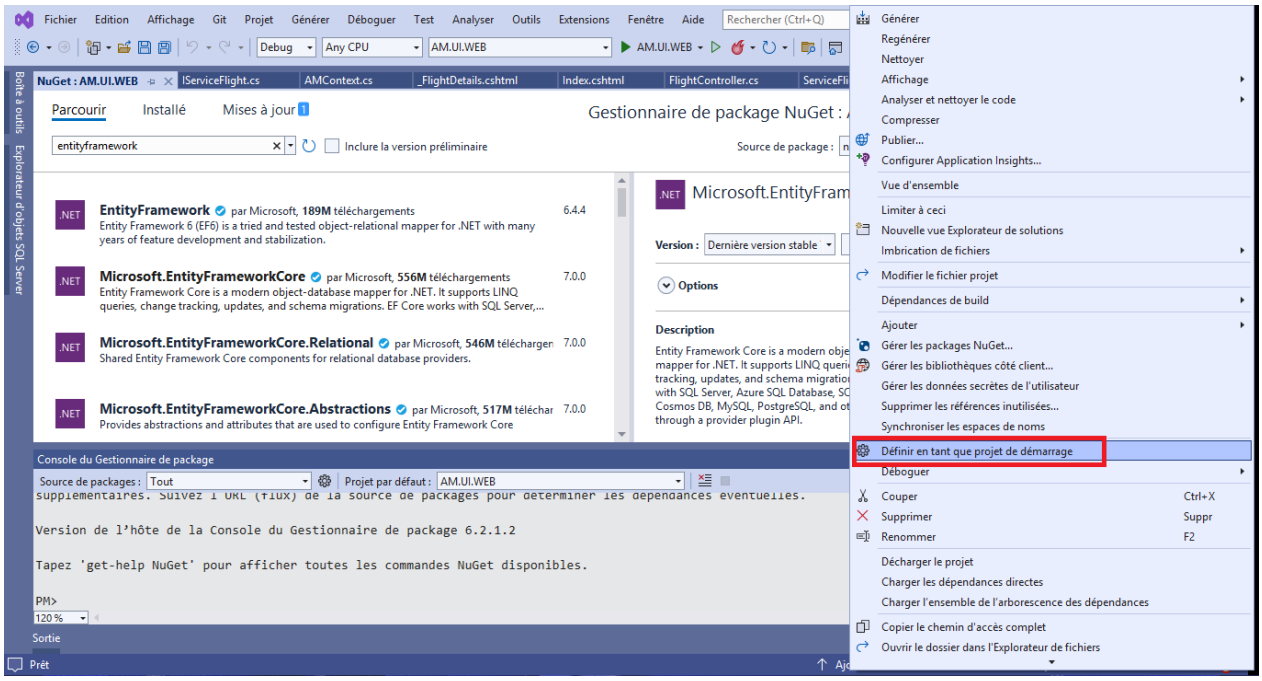
2. Dans le projet Web installer les composants suivants :

   - **EntityFramework Core**
   - **EntityFramework.SqlServer**
   - **EntityFramework.Proxies**





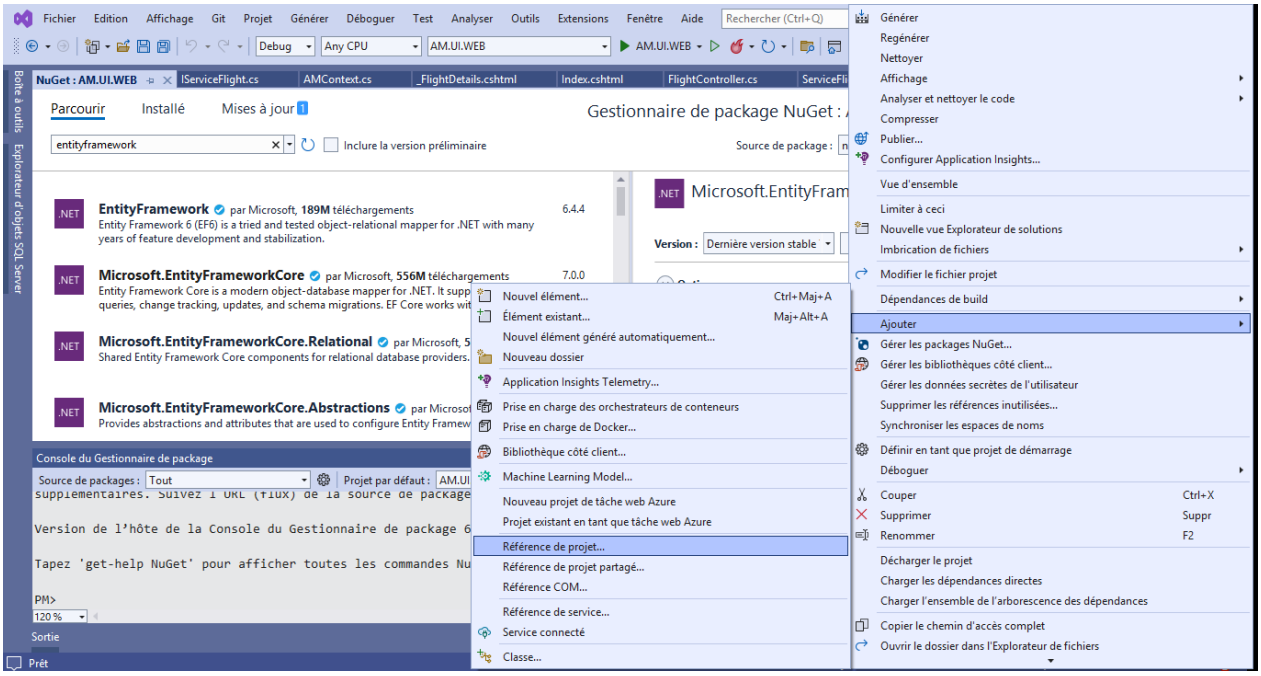
3. Définir le projet **AM.UI.WEB** comme projet de démarrage.

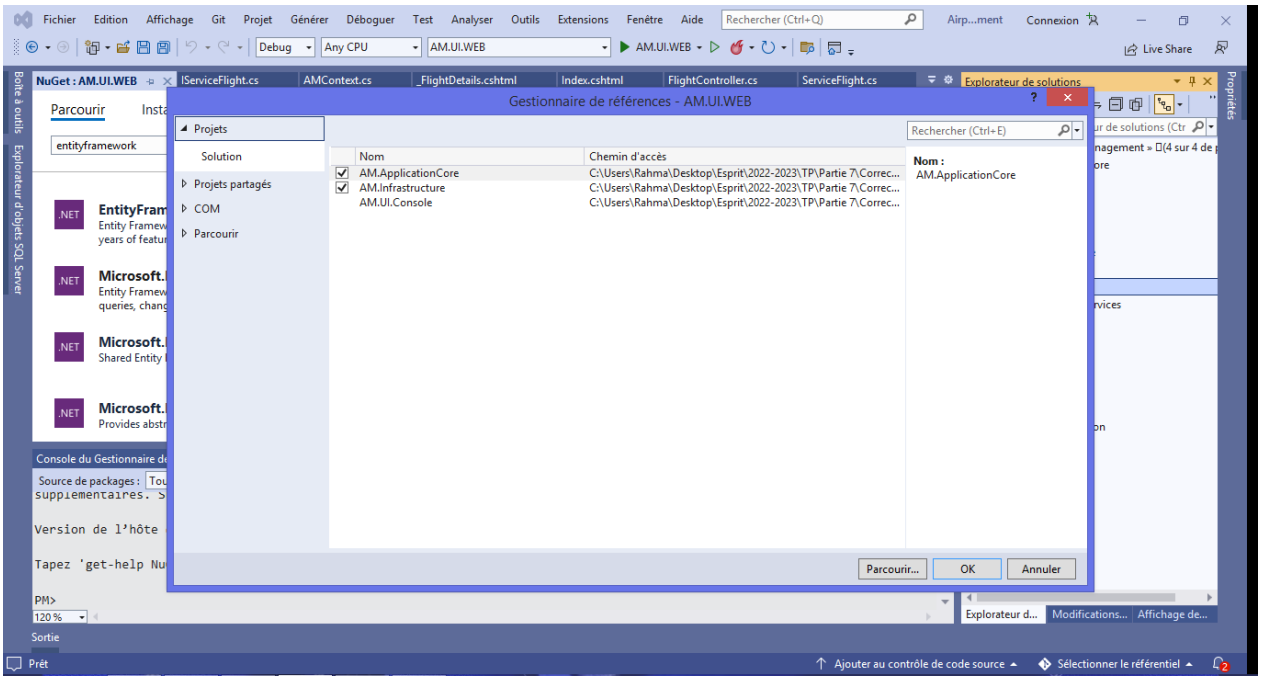


4. Lancer l'application et analyser le résultat.

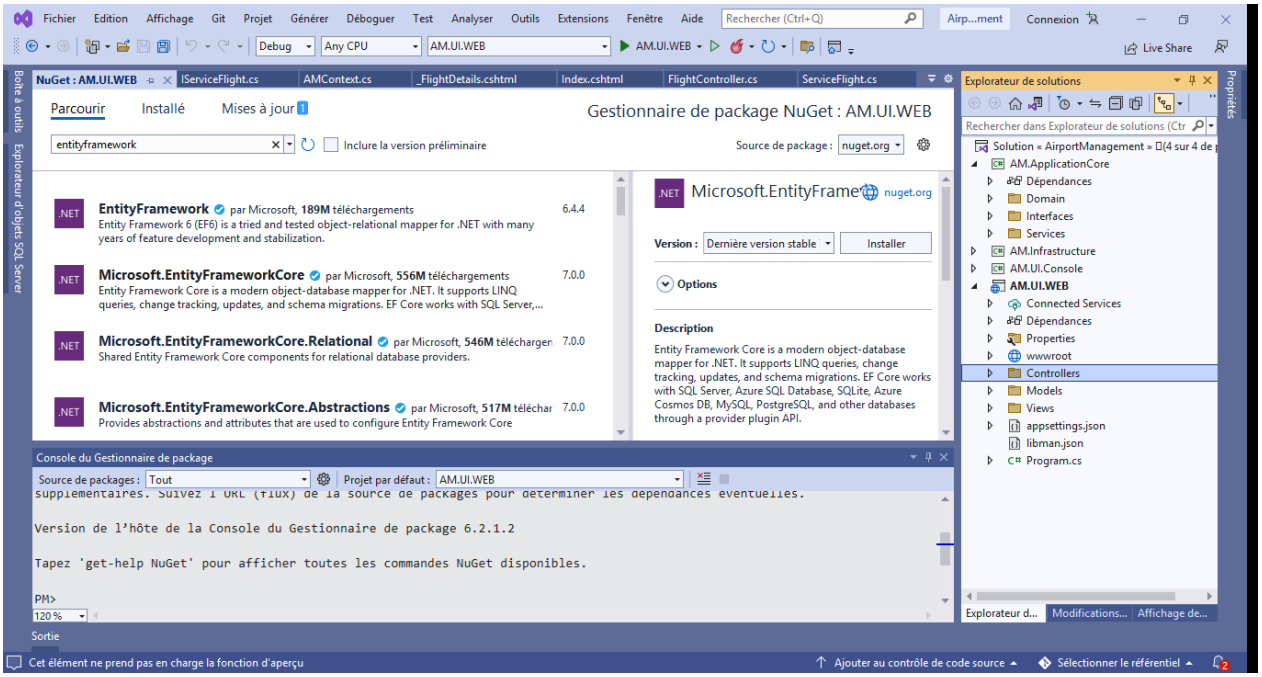
II Création d'un contrôleur

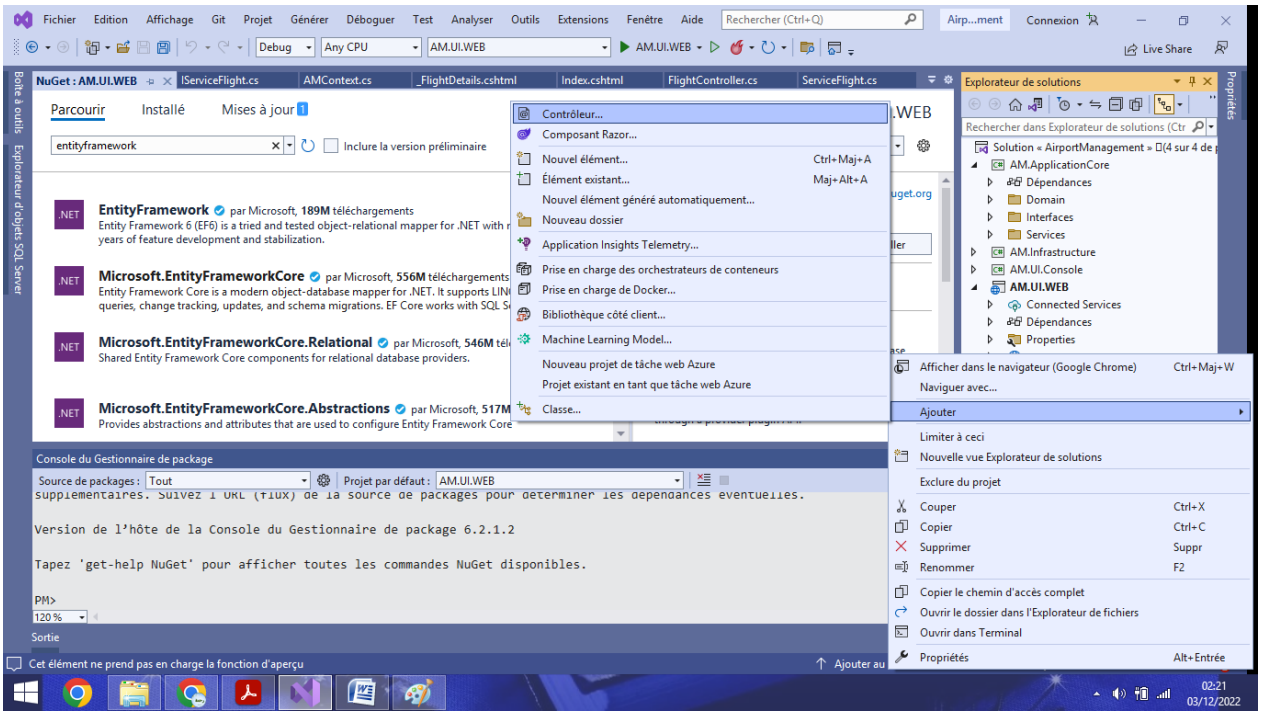
5. Dans le projet **Web**, référencer les projets **Core** et **Infrastructure**.

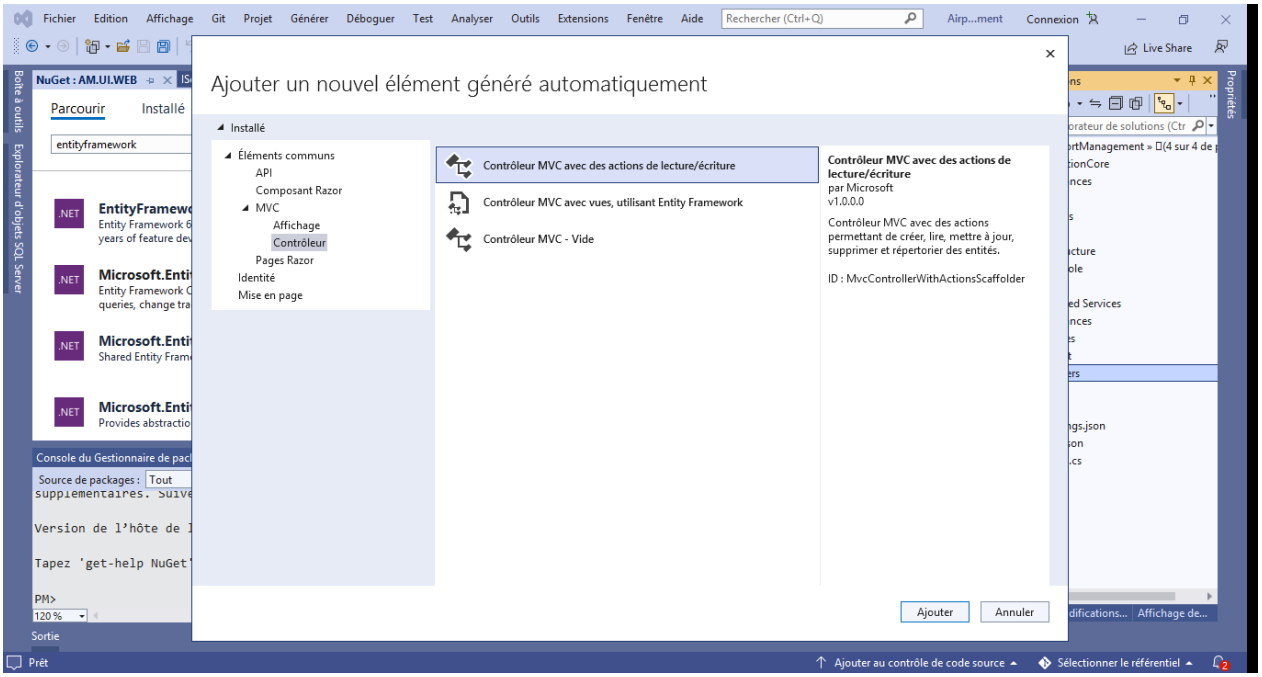


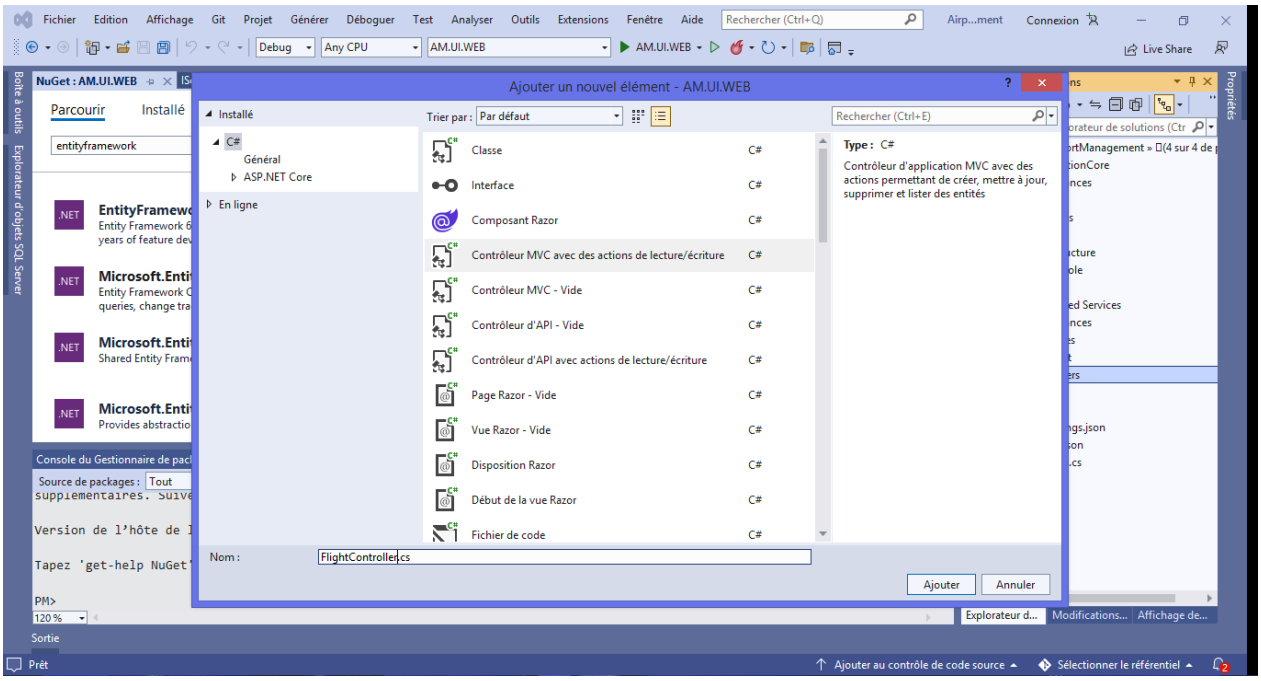


6. Au niveau du dossier **Controllers**, ajouter un contrôleur nommé **FlightController** contenant par défaut les actions de lecture/écriture.




Contrôleur...
Composant Razor...
Nouvel élément... Ctrl+Maj+A
Élément existant... Maj+Alt+A
Nouvel élément généré automatiquement...
Nouveau dossier
Application Insights Telemetry...
Prise en charge des orchestrateurs de conteneurs
Prise en charge de Docker...
Bibliothèque côté client...
Machine Learning Model...
Nouveau projet de tâche web Azure
Projet existant en tant que tâche web Azure
Classe...
Afficher dans le navigateur (Google Chrome) Ctrl+Maj+W
Naviguer avec...
Ajouter
Limiter à ceci
Nouvelle vue Explorateur de solutions
Exclure du projet
Couper Ctrl+X
Copier Ctrl+C
Supprimer Suppr
Renommer F2
Copier le chemin d'accès complet
Ouvrir le dossier dans l'Explorateur de fichiers
Ouvrir dans Terminal
Propriétés Alt+Entrée


Ajouter un nouvel élément généré automatiquement
Installé
Éléments communs
API
Composant Razor
MVC
Affichage
Contrôleur
Pages Razor
Identité
Mise en page
Contrôleur MVC avec des actions de lecture/écriture
Contrôleur MVC avec vues, utilisant Entity Framework
Contrôleur MVC - Vide
Contrôleur MVC avec des actions de lecture/écriture
par Microsoft
v1.0.0.0
Contrôleur MVC avec des actions permettant de créer, lire, mettre à jour, supprimer et répertorier des entités.
ID : MvcControllerWithActionsScaffolder
Ajouter
Annuler



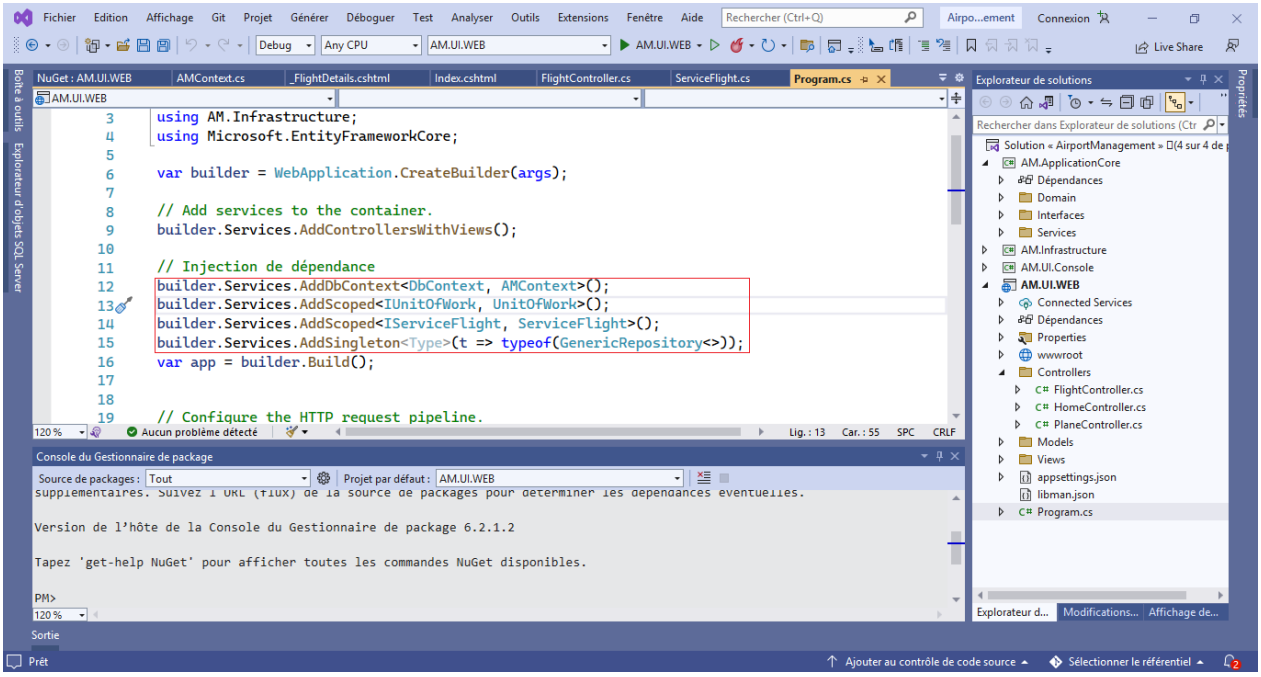
7. Dans la classe **FlightController**, injecter le service **FlightService**.

```
public class FlightController : Controller
    {
        private readonly IServiceFlight _flightService;

        public FlightController(IServiceFlight flightService)
        {
              _flightService = flightService;
        }
```

……

8. Dans la classe **program.cs**, injecter les dépendances en utilisant le conteneur d'injection .Net.



III Création d'une vue Index

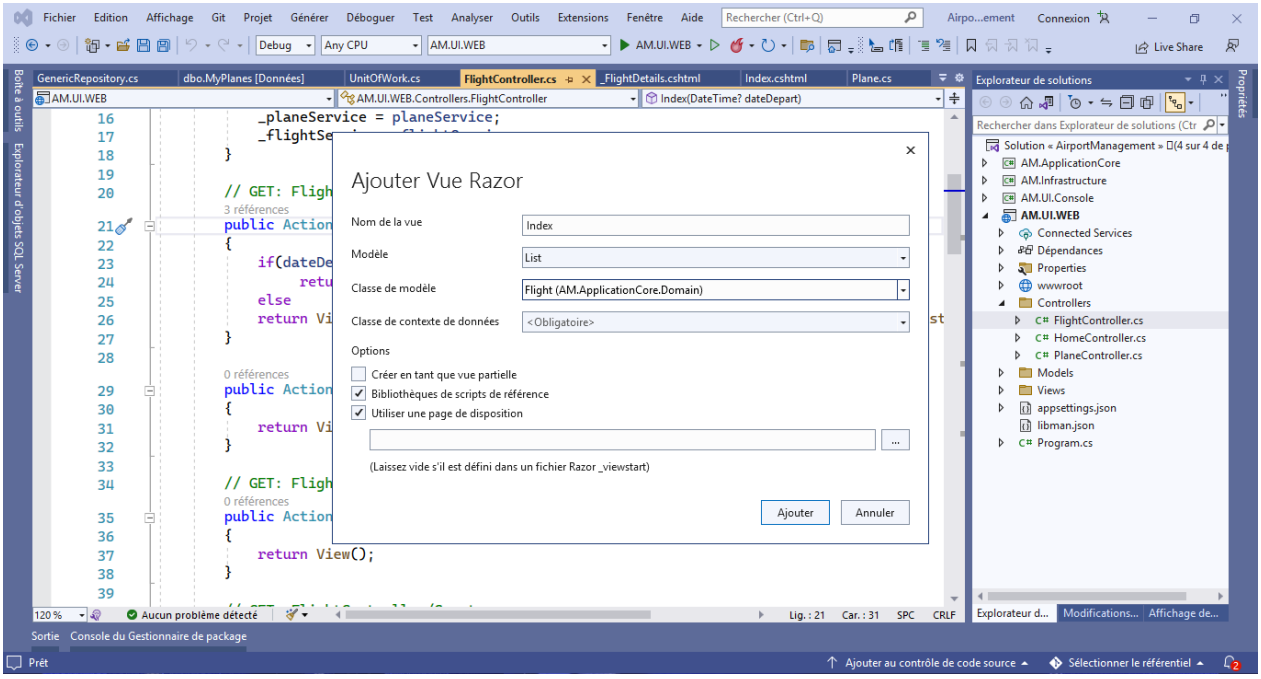
L'objectif est de retourner une vue affichant la liste des vols.

9. Dans **FlightController**, modifier l'action **Index** afin de récupérer la liste de vols et la passer en paramètre à la vue.

```
// GET: FlightController
        public ActionResult Index()
        {

return View(_flightService.GetAll().ToList());
        }
```

10. Créer la vue **Index** : faites un clic droit sur le nom de l'action **Index,** cliquez sur **AddView** et remplir la fenêtre.



11. Exécuter l'application et afficher la vue **Index**.

IV Ajout d'une barre de recherche

On se propose d'ajouter une barre de recherche qui permet de filtrer les vols par date de départ.

12. Insérer le code html nécessaire dans la vue **Index**.

```
<form asp-action="index">
<fieldset>
    <legend> Recherche</legend>
    Saisir une date de départ : <input type="date" name="dateDepart" />
    <input type="submit" value="Serach" />
</fieldset>
</form>
```

13. Modifier l'action **Index** afin de filtrer les vols affichés en fonction du filtre saisi.

```
// GET: FlightController
        public ActionResult Index(DateTime? dateDepart)
        {
            if(dateDepart == null)
                return View(_flightService.GetAll().ToList());
            else
            return
View(_flightService.GetMany(f=>f.FlightDate.Date.Equals(dateDepart)).ToList());
        }
```

14. Lancer l'application et tester la recherche

V Appel de service spécifique

15. Ajouter un bouton qui permet d'ordonner les vols en utilisant l'appel au service **SortFlights()**.

Dans ServiceFlight :

```
public IEnumerable<Flight> SortFlights()
        {
            return GetAll().OrderByDescending(f => f.FlightDate);
           }
```

Dans la vue Index :

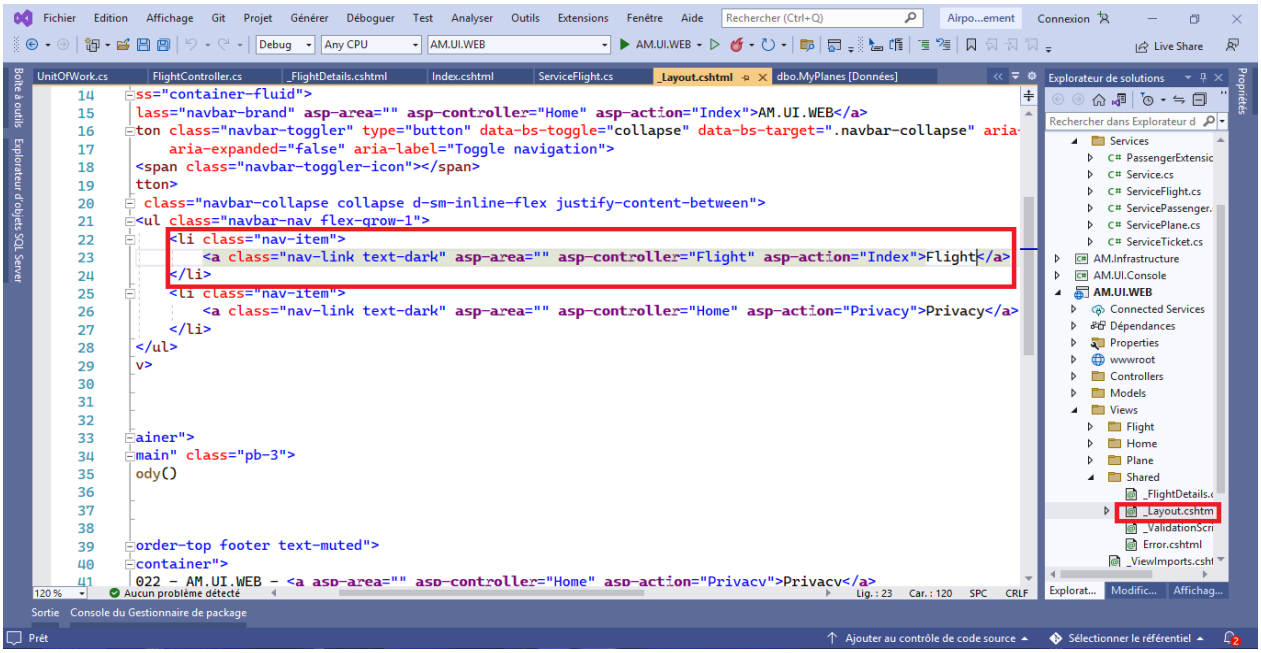
```
<form asp-action="Sort">
<fieldset>
    <input type="submit" value="Sort" />
</fieldset>
   </form>
```

Dans FlightController :

```
public ActionResult Sort()
        {
            return View("Index", _flightService.SortFlights());
           }
```

VI Utilisation d'un fichier layout

16. Modifier le fichier **_Layout.cshtml** afin d'ajouter un lien pour l'action **Index** et tester.



VII Création de la vue Create

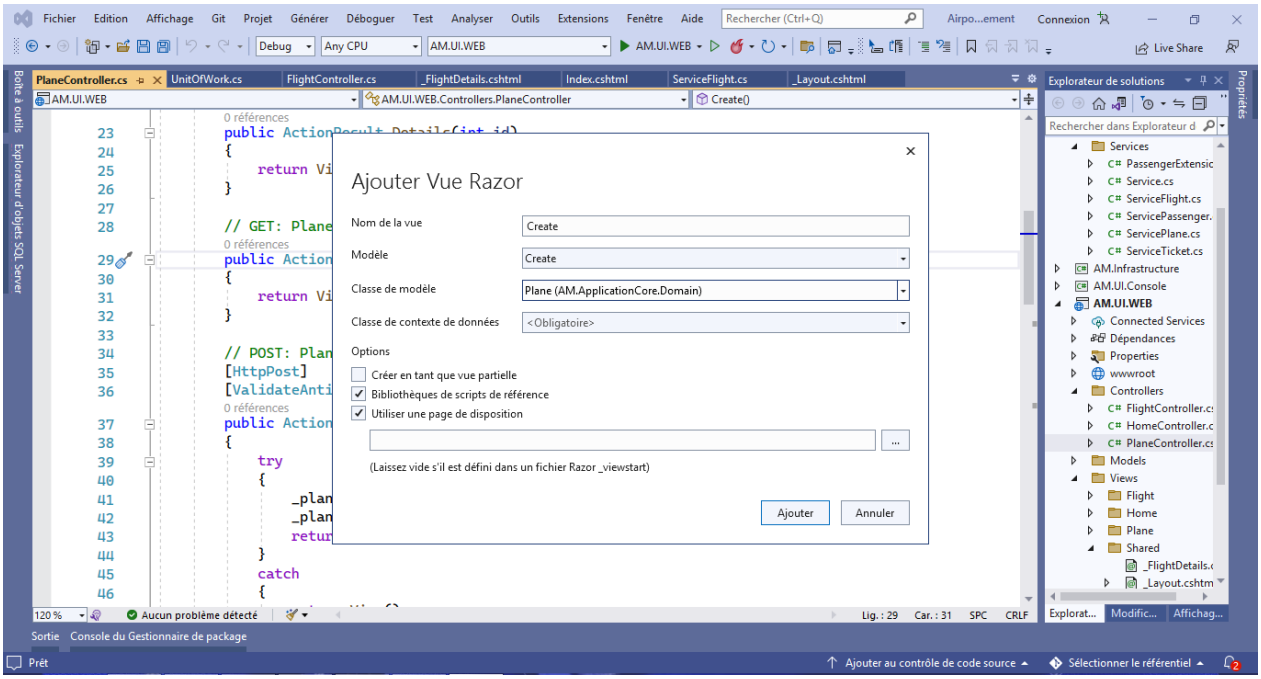
L'objectif est de créer un formulaire d'ajout d'un avion.

17. Ajouter un contrôleur nommé **PlaneController**.
18. Faire les injections nécessaires.

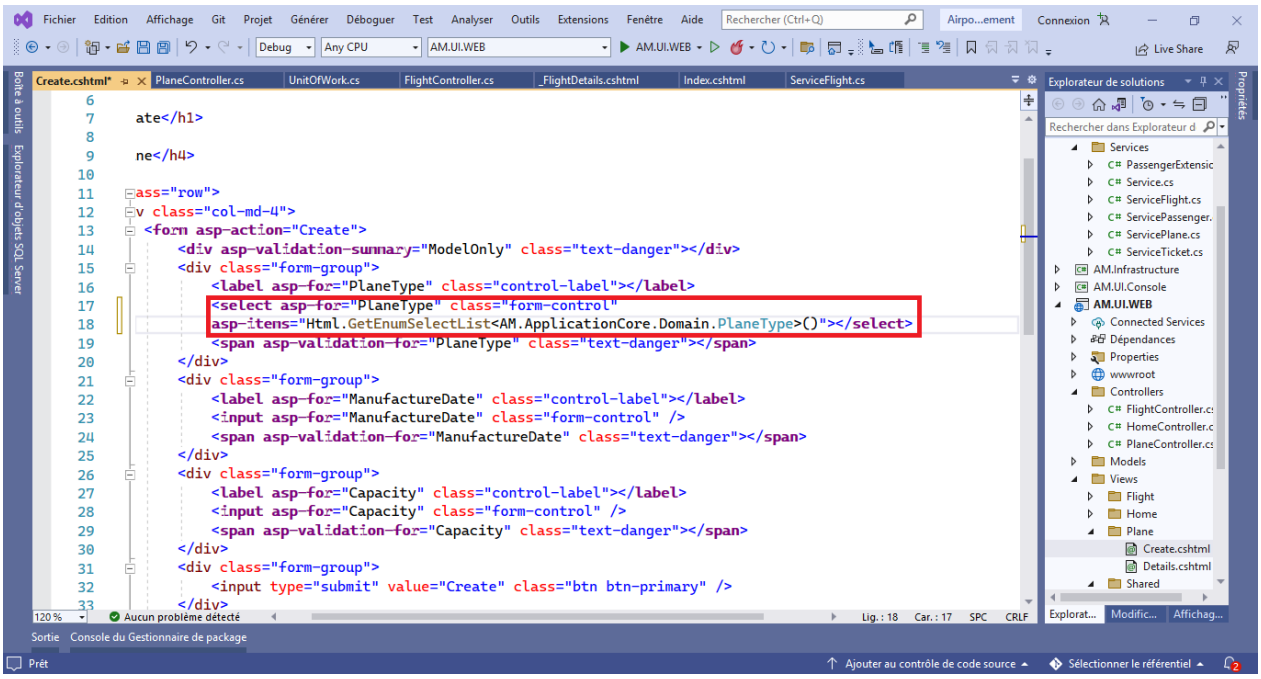
```
private readonly IServicePlane _planeService;

        public PlaneController(IServicePlane planeService)
        {
            _planeService = planeService;
        }
```

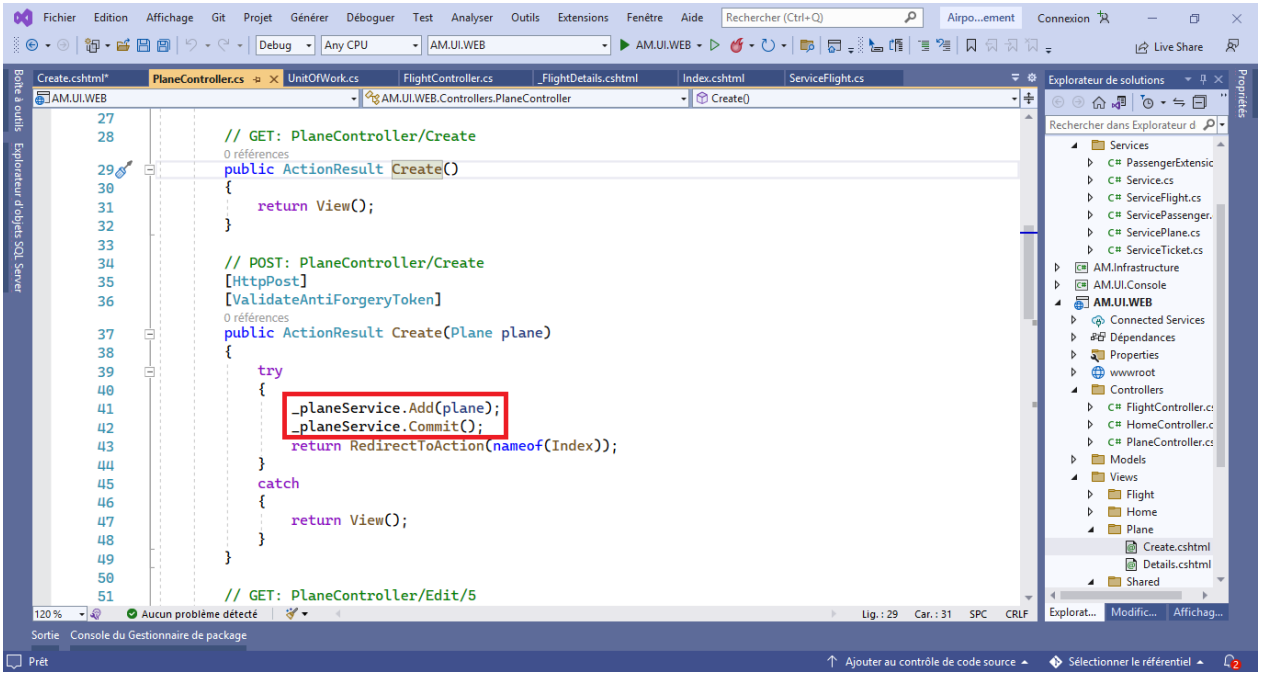
19. Faire un clic droit sur le nom de l'action **Create,** choisir **AddView** et remplir la fenêtre.



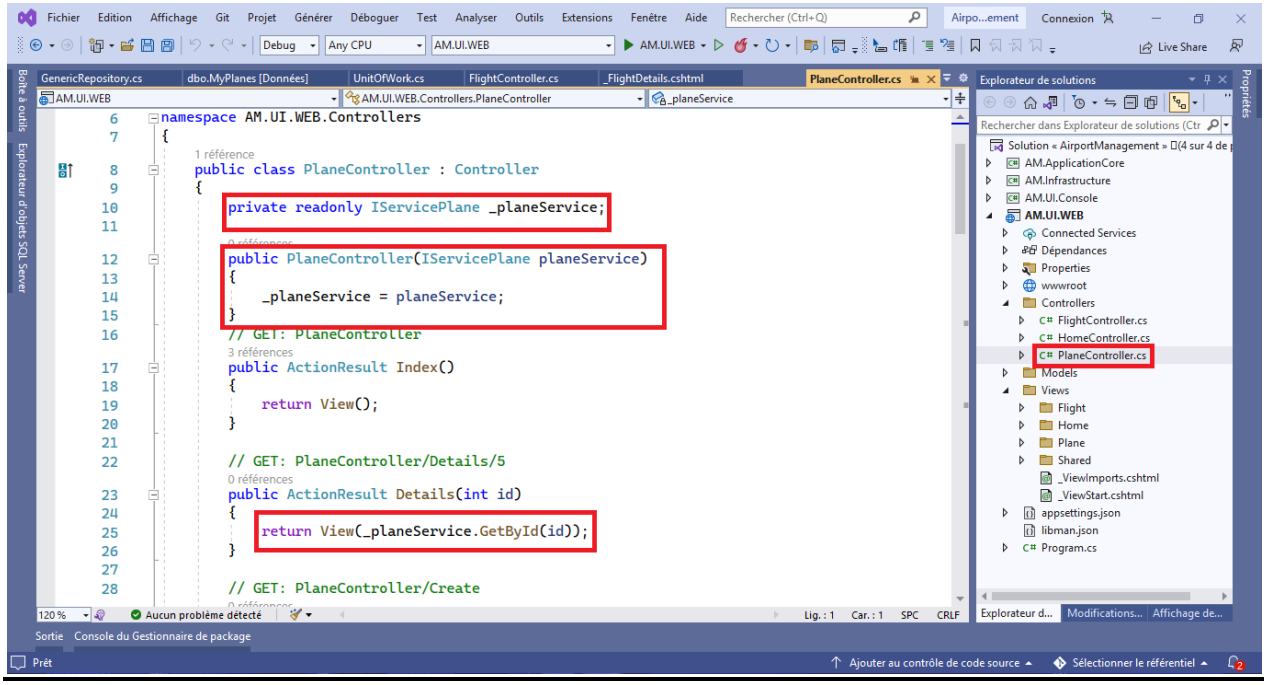
20. Adapter le code de la vue pour sélectionner le type de l'avion.

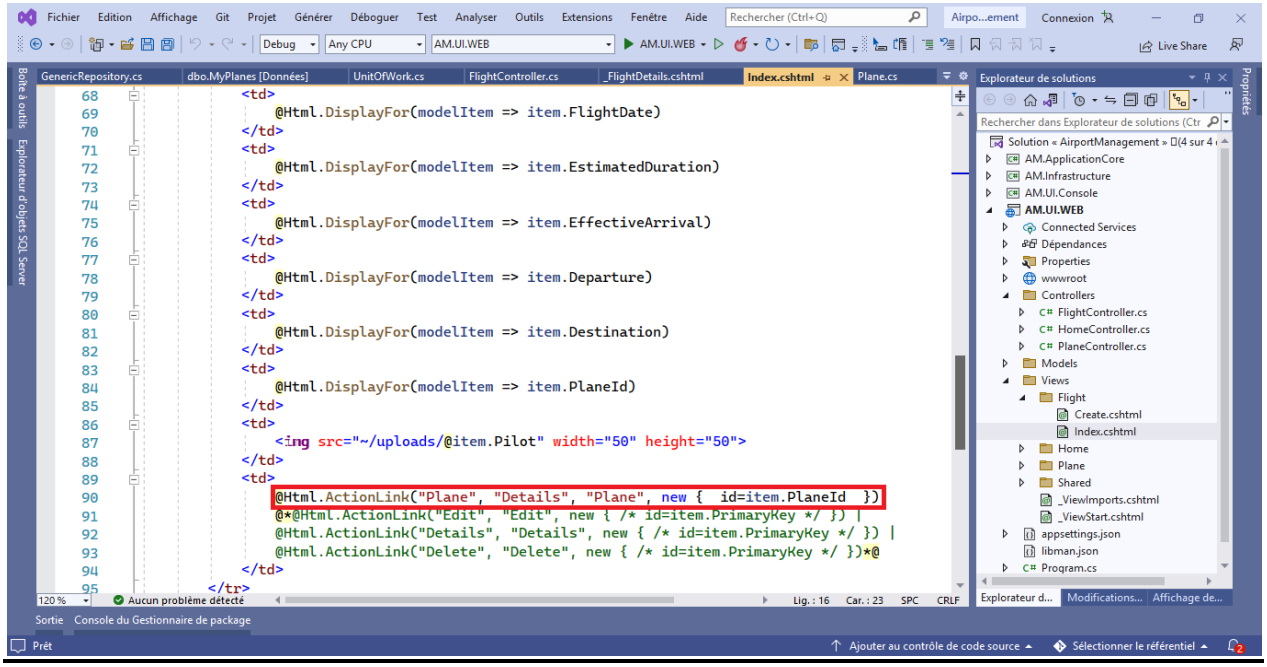


21. Modifier l'action **Create POST** afin d'insérer un avion dans la BDD.



22. Ajouter un lien pour afficher les détails de l'avion affecté à un vol.



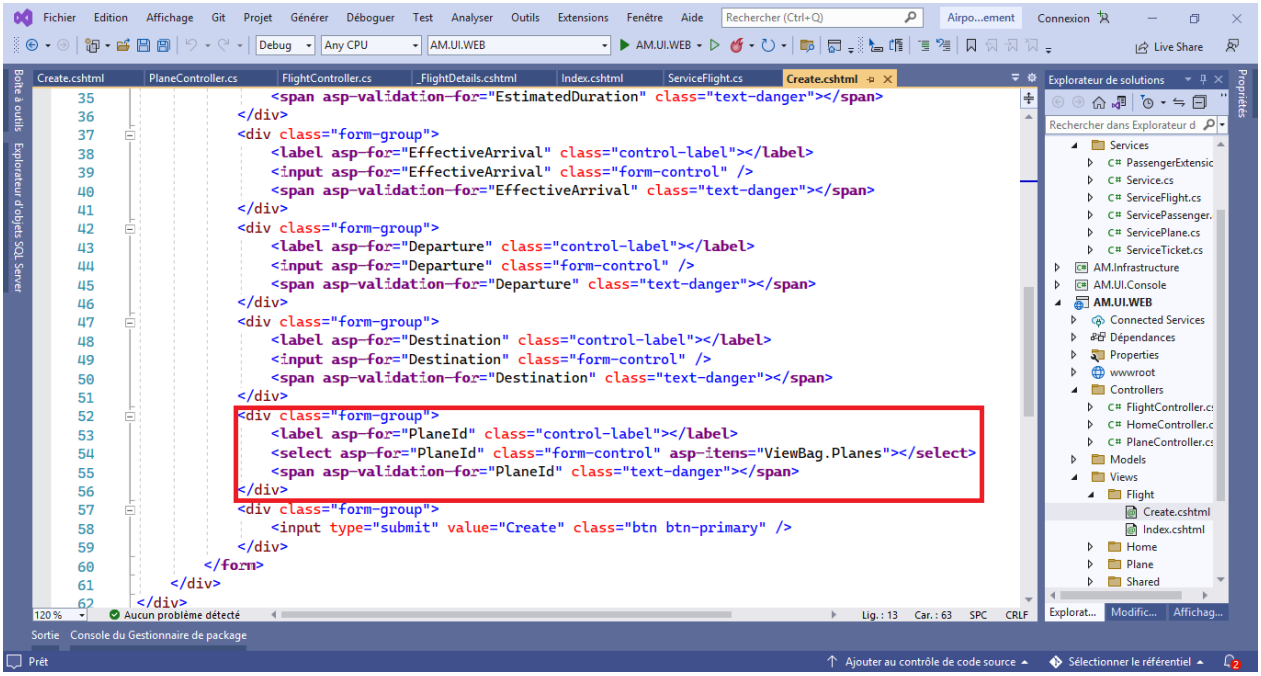


L'objectif est de créer un formulaire d'ajout d'un vol.

23. Ajouter et configurer la vue **Create**.
24. Sélection de l'avion affecté au vol:
    a. Dans la classe FlightContoller, ajouter le code nécessaire qui permet de récupérer la liste des avions.

```
// GET: FlightController/Create
        public ActionResult Create() {
            ViewBag.Planes = new SelectList(_planeService.GetAll().ToList(),
                "PlaneId", "Information");
            return View();
                    }
```

    b. Adapter la vue **Create** pour afficher la liste des avions.



25. Modifier l'action **Create POST** afin d'insérer le vol dans la BDD.

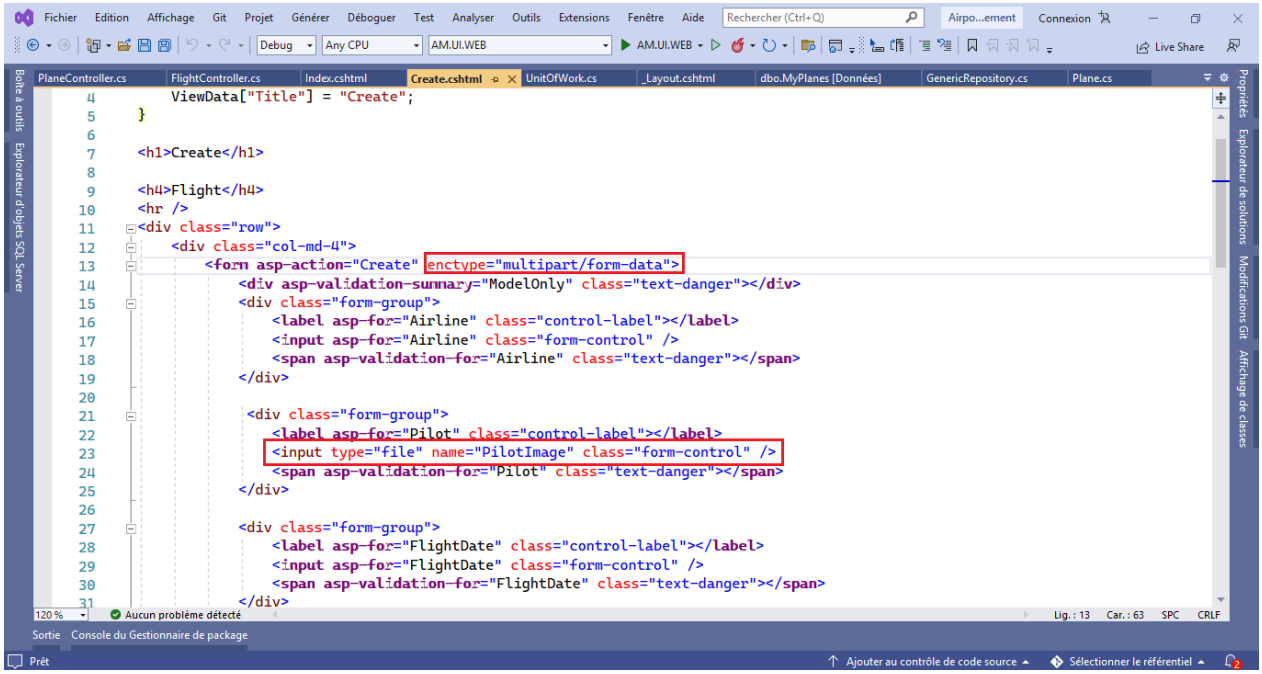
```
// POST: FlightController/Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Flight flight)
        {

            try
            {
                _flightService.Add(flight);
                _flightService.Commit();
                return RedirectToAction(nameof(Index));
            }
```

```
            catch
            {
                return View();
            }
        }
```

VIII Importation d'une image

26. On se propose d'utiliser l'image du pilote comme information supplémentaire pour identifier le vol.
    a. Dans la classe **Flight**, ajouter la propriété **Pilot** de type string.
    b. Mettre à jour la BDD.
    c. Adapter la vue **Create** pour l'upload de l'image.



    d. Dans le projet **AM.UI.WEB**, sous le dossier **wwwroot** ajouter un nouveau dossier nommé **uploads**.

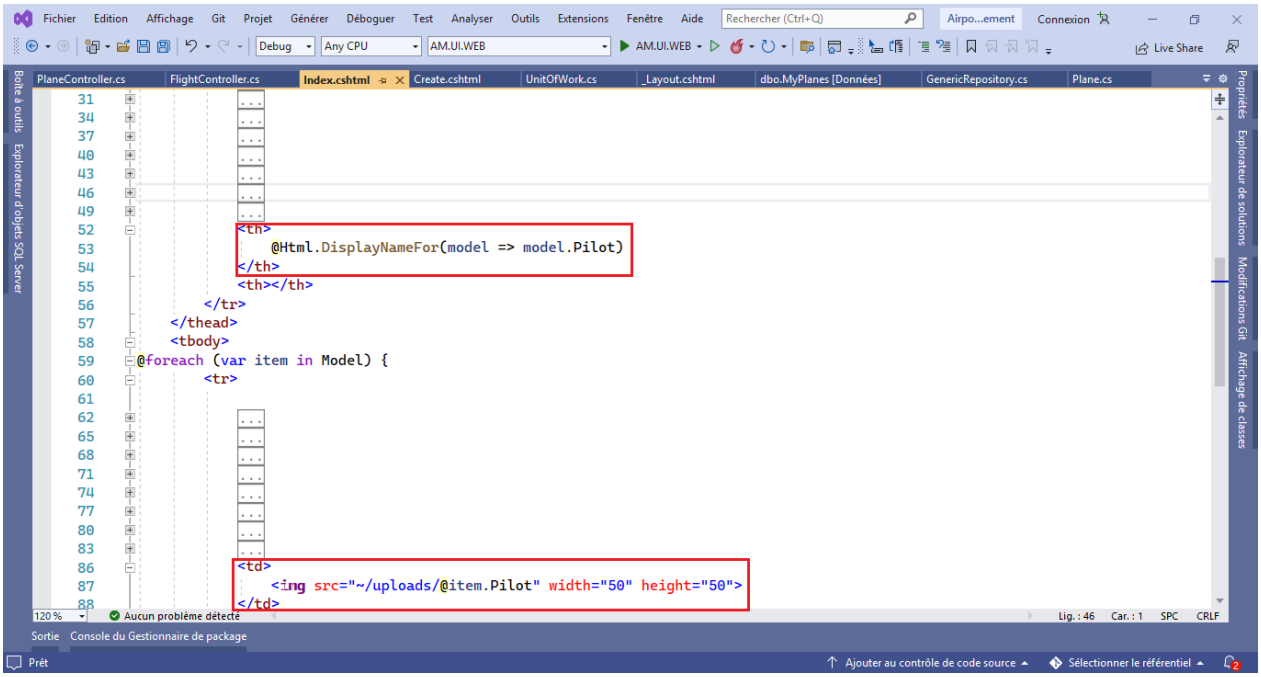
27. Modifier l'action **Create POST** afin de télécharger l'image du pilote et l'insérer sous le répertoire **uploads** en plus de l'ajout du vol.

```
[HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Flight flight, IFormFile PilotImage)
        {

            try
            {
                if (PilotImage != null)
                {
                    var path = Path.Combine(Directory.GetCurrentDirectory(),
"wwwroot", "uploads", PilotImage.FileName);
                    Stream stream = new FileStream(path, FileMode.Create);
                    PilotImage.CopyTo(stream);
```

```
            flight.Pilot = PilotImage.FileName;
        }
        _flightService.Add(flight);
        _flightService.Commit();
        return RedirectToAction(nameof(Index));
    }
    catch
    {
        return View();
    }
}
```

28. Adapter la vue **Index** pour afficher l'image du pilote.

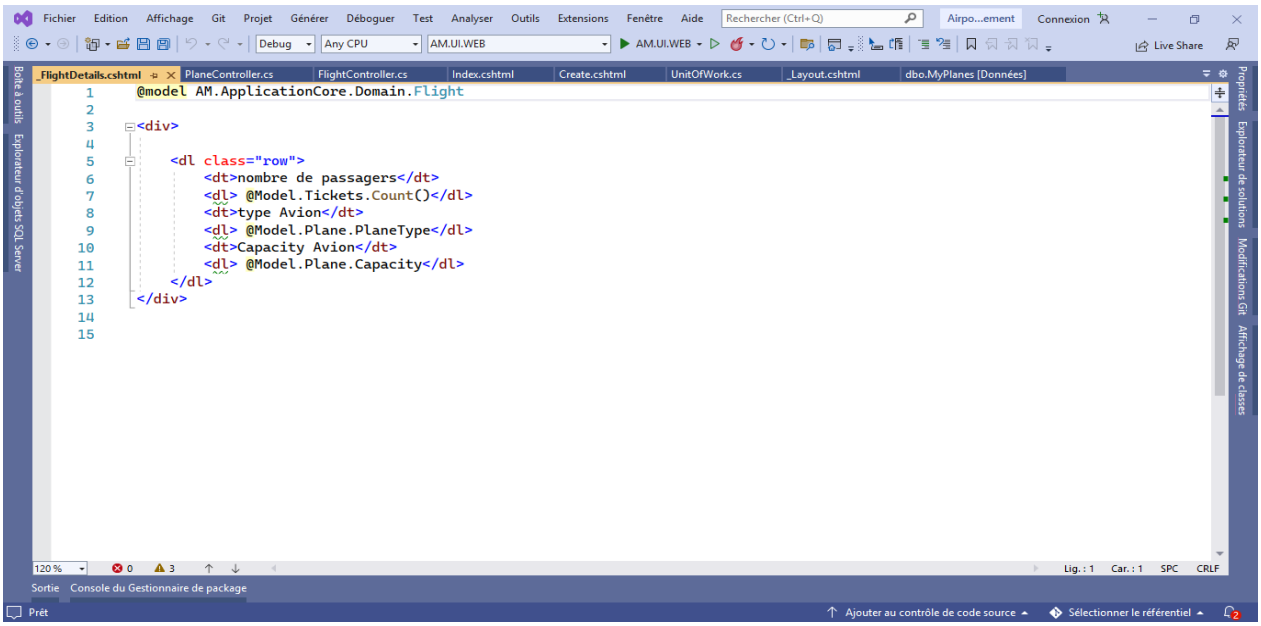


29. Tester l'ajout d'un vol et le téléchargement des images.
30. Créer la vue **Delete** qui permet de supprimer un vol.
31. Créer de la vue **Edit** qui permet de modifier les informations d'un vol.

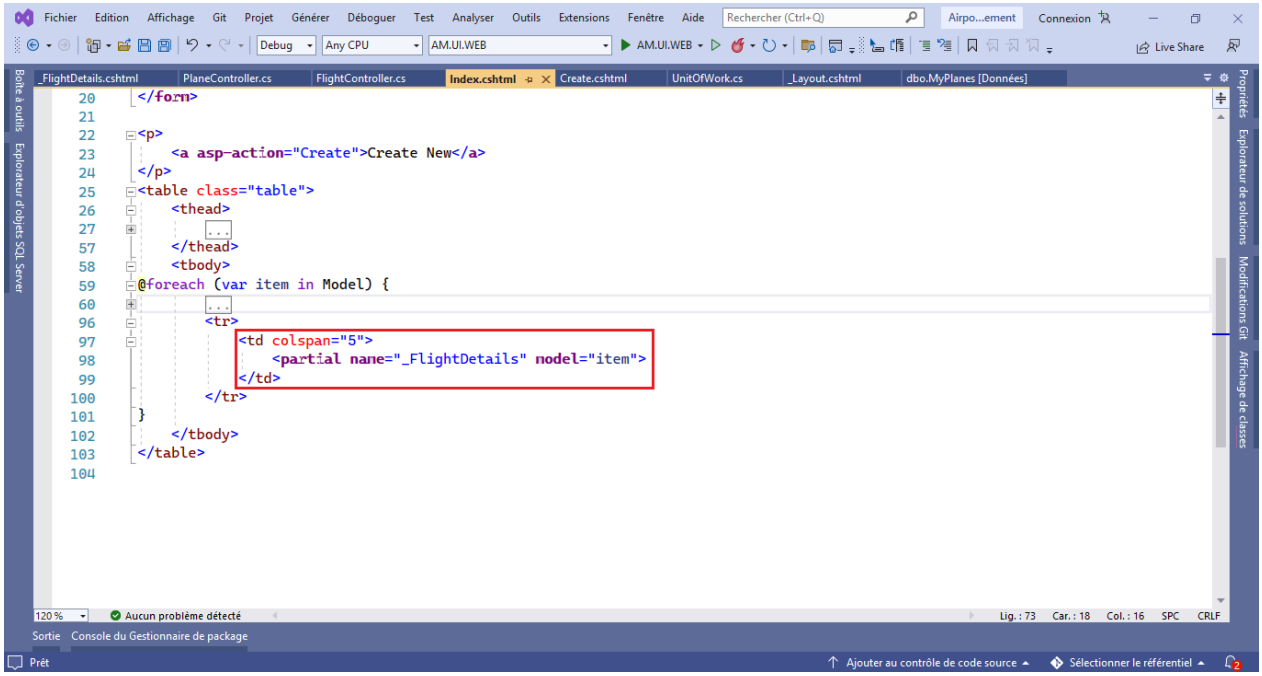
IX Utilisation d'une vue partielle

On se propose d'insérer dans une ligne séparée des informations supplémentaires sur un vol (nombre de passagers, type et capacité de l'avion).

32. Sous le répertoire **Views\Flight** créer et configurer la vue partielle **_FlightDetails**.



33. Adapter la vue **Index** pour afficher la vue partielle dans une deuxième ligne séparée pour chaque vol.



34. Tester.