
Rapport de Développement d'une Application de Gestion de Contacts avec ASP.NET MVC

Réalisé par :

EL MECHRAFI AMINE

ELHAMOUSSI ABDELKABIR

Encadré par :

Pr . ELBANNAY Omar

Introduction :

Ce rapport présente le développement d'une application de gestion de contacts en utilisant ASP.NET MVC (C#). L'application permet de créer, éditer, et supprimer des contacts, avec une interface utilisateur simple et intuitive. Ce projet fait partie du TP 8 du cycle ingénieur, 2ème Année GI.

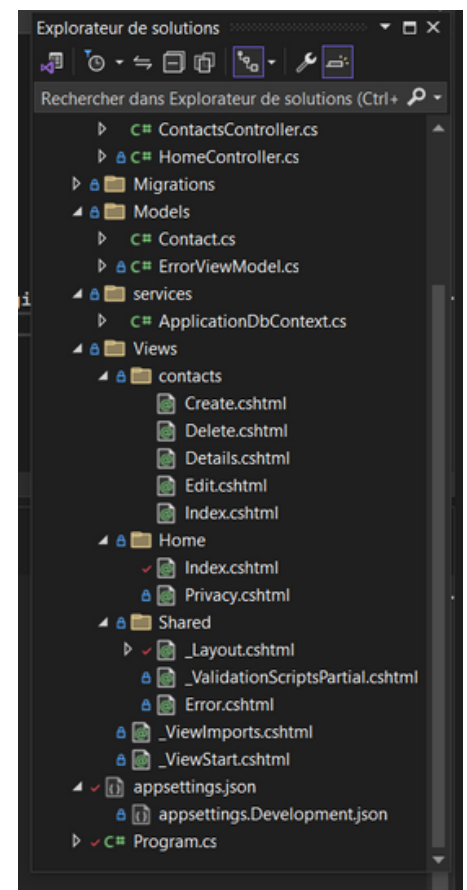
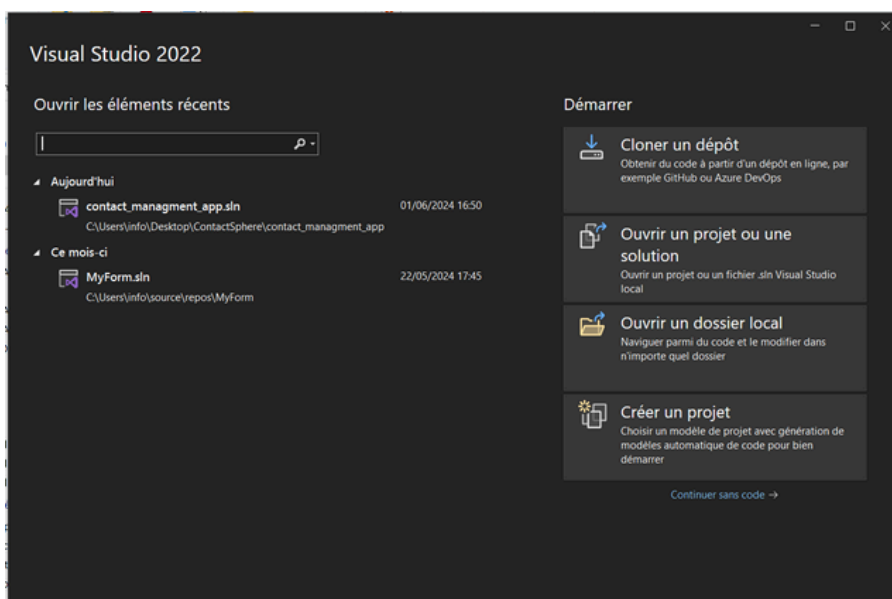
2. Étapes du Développement

1. Création de l'Application ASP.NET MVC

Nous avons commencé par créer une nouvelle application ASP.NET MVC en utilisant Visual Studio. Le modèle choisi est "ASP.NET Web Application" avec le template MVC.

l'architecture du projet

creation d'un projet "contact_managment_app"



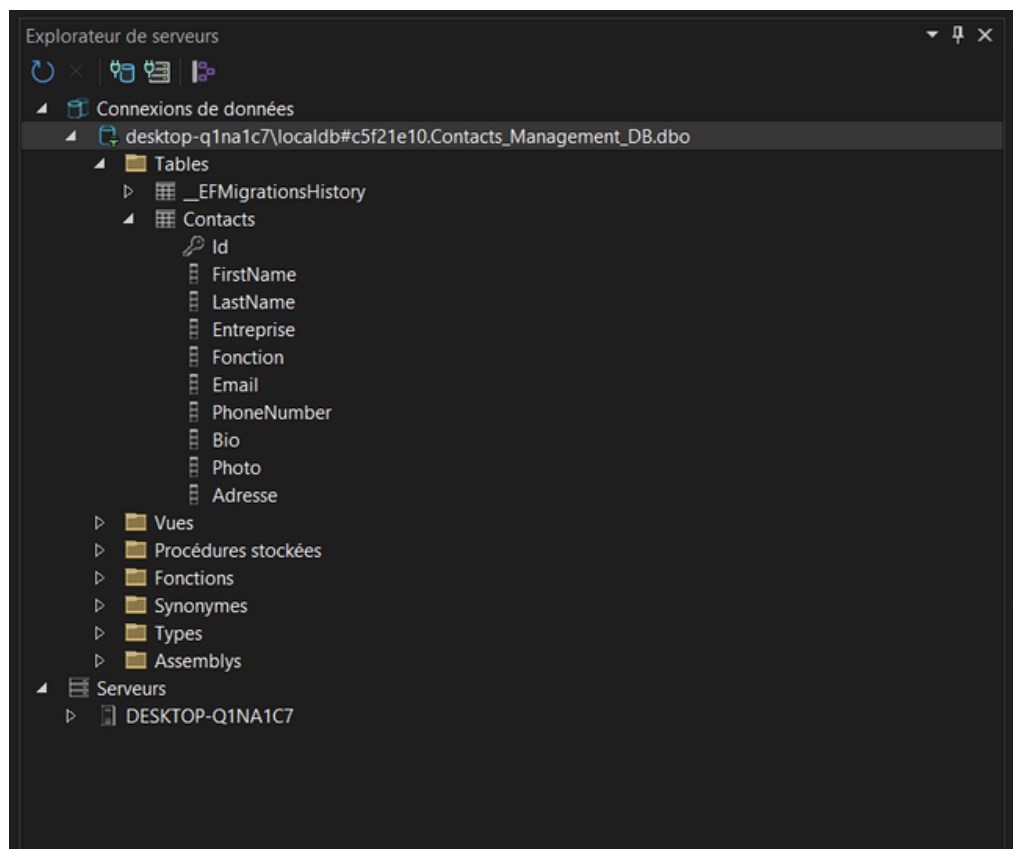
2. Création de la Base de Données

2. Création de la Base de Données

La base de données a été créée en utilisant SQL Server. La table principale, Contacts, contient les colonnes suivantes :

- Id (int, Primary Key)
- FirstName (nvarchar)
- LastName (nvarchar)
- Entreprise (nvarchar)
- Fonction (nvarchar)
- Email (nvarchar)
- PhoneNumber (nvarchar)
- Bio (nvarchar)
- Photo (nvarchar)
- Adresse (nvarchar)

la base de Données du projet “contact_management_app”



Modèle Contact

```

1 using System.ComponentModel.DataAnnotations;
2
3 namespace contact_managment_app.Models
4 {
5     16 références
6     public class Contact
7     {
8         [Key]
9         8 références
10        public int Id { get; set; }
11
12        [Required]
13        [MaxLength(100)]
14        9 références
15        public string FirstName { get; set; }
16
17        [Required]
18        [MaxLength(100)]
19        9 références
20        public string LastName { get; set; }
21
22        [MaxLength(100)]
23        9 références
24        public string? Entreprise { get; set; }
25
26        [MaxLength(100)]
27        9 références
28        public string? Fonction { get; set; }
29
30        [MaxLength(100)]
31        9 références
32        public string? Email { get; set; }
33    }
34 }

```

```

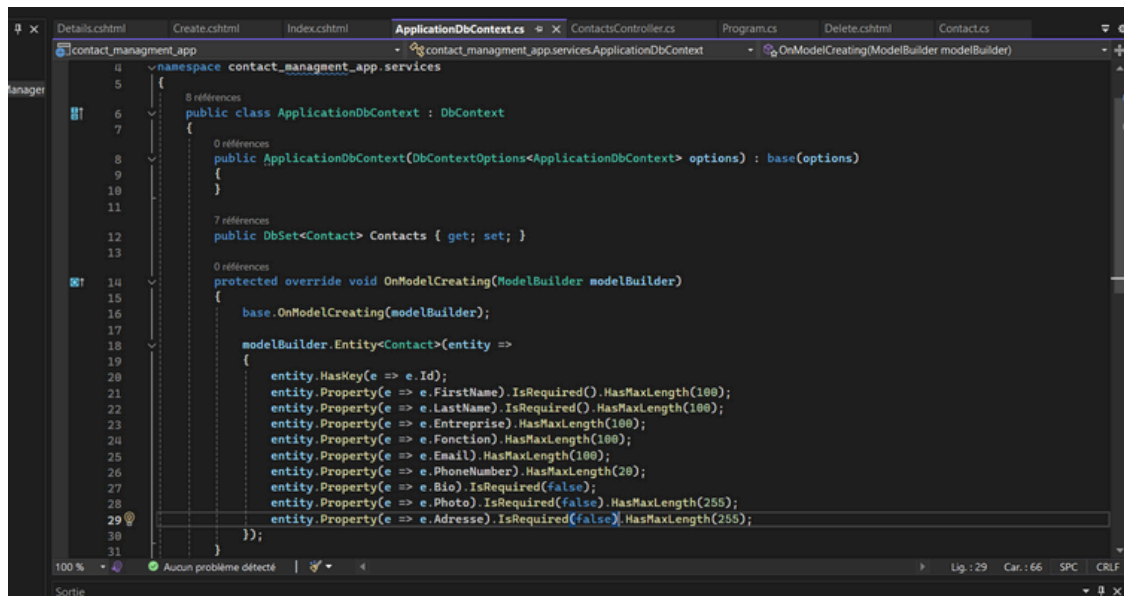
16 public string LastName { get; set; }
17
18 [MaxLength(100)]
19 9 références
20 public string? Entreprise { get; set; }
21
22 [MaxLength(100)]
23 9 références
24 public string? Fonction { get; set; }
25
26 [MaxLength(100)]
27 9 références
28 public string? Email { get; set; }
29
30 [MaxLength(20)]
31 9 références
32 public string? PhoneNumber { get; set; }
33
34 5 références
35 public string? Bio { get; set; }
36
37 [MaxLength(255)]
38 5 références
39 public string? Photo { get; set; }
40
41 [MaxLength(255)]
42 5 références
43 public string? Adresse { get; set; }
44 }

```

3. Génération du Modèle de Classes avec Entity Framework

Nous avons utilisé Entity Framework pour générer le modèle de classes depuis la base de données. La commande Scaffold-DbContext a été utilisée pour créer les classes de modèle.

Configurer le contexte de la base de données : Créer une classe
ApplicationDbContext qui hérite de DbContext.



```

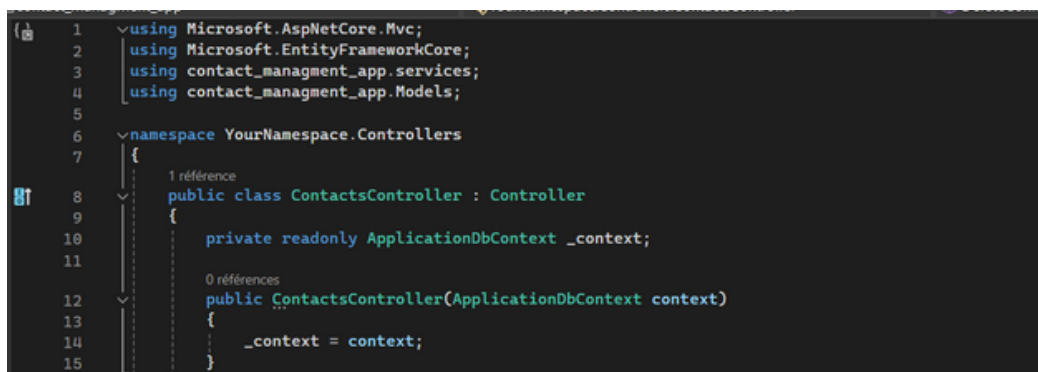
1 namespace contact_management_app.services
2 {
3     8 références
4     public class ApplicationDbContext : DbContext
5     {
6         0 références
7         public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
8         {
9         }
10
11     7 références
12     public DbSet<Contact> Contacts { get; set; }
13
14     0 références
15     protected override void OnModelCreating(ModelBuilder modelBuilder)
16     {
17         base.OnModelCreating(modelBuilder);
18
19         modelBuilder.Entity<Contact>(entity =>
20         {
21             entity.HasKey(e => e.Id);
22             entity.Property(e => e.FirstName).IsRequired().HasMaxLength(100);
23             entity.Property(e => e.LastName).IsRequired().HasMaxLength(100);
24             entity.Property(e => e.Entreprise).HasMaxLength(100);
25             entity.Property(e => e.Fonction).HasMaxLength(100);
26             entity.Property(e => e.Email).HasMaxLength(100);
27             entity.Property(e => e.PhoneNumber).HasMaxLength(20);
28             entity.Property(e => e.Bio).IsRequired(false);
29             entity.Property(e => e.Photo).IsRequired(false).HasMaxLength(255);
30             entity.Property(e => e.Adresse).IsRequired(false).HasMaxLength(255);
31         });
32     }
33 }

```

4. Création de l'Action Contrôleur et de la Vue pour Lister les Contacts

Une action contrôleur Index a été créée dans le ContactsController pour récupérer et afficher tous les contacts. La vue associée utilise le modèle List<Contact> pour afficher les contacts dans un tableau HTML.

Création du Contrôleur ContactController



```

1 using Microsoft.AspNetCore.Mvc;
2 using Microsoft.EntityFrameworkCore;
3 using contact_management_app.services;
4 using contact_management_app.Models;
5
6 namespace YourNamespace.Controllers
7 {
8     1 référence
9     public class ContactsController : Controller
10     {
11         private readonly ApplicationDbContext _context;
12
13         0 références
14         public ContactsController(ApplicationDbContext context)
15         {
16             _context = context;
17         }
18     }
19 }

```

Index.cshtml
Affiche la liste des contacts.

la methde pour afficher la liste des contacts
et les details de chaque contact.

```

4 | <a asp-action="Create">Créer un nouveau contact</a>
5 | </p>
6 |
7 | <table class="table">
8 |   <thead>
9 |     <tr>
10 |       <th>Prénom</th>
11 |       <th>Nom</th>
12 |       <th>Entreprise</th>
13 |       <th>Fonction</th>
14 |       <th>Email</th>
15 |       <th>Téléphone</th>
16 |       <th>Actions</th>
17 |     </tr>
18 |   </thead>
19 |   <tbody>
20 |     @foreach (var contact in Model)
21 |     {
22 |       <tr>
23 |         <td>@contact.FirstName</td>
24 |         <td>@contact.LastName</td>
25 |         <td>@contact.Entreprise</td>
26 |         <td>@contact.Fonction</td>
27 |         <td>@contact.Email</td>
28 |         <td>@contact.PhoneNumber</td>
29 |         <td>
30 |           <a asp-action="Details" asp-route-id="@contact.Id">Détails</a> |
31 |           <a asp-action="Edit" asp-route-id="@contact.Id">Modifier</a> |
32 |           <a asp-action="Delete" asp-route-id="@contact.Id">Supprimer</a>
33 |         </td>
34 |       </tr>
35 |     }
36 |   </tbody>
37 | </table>

```

```

// GET: Contacts
3 références
public async Task<IActionResult> Index()
{
    return View(await _context.Contacts.ToListAsync());
}

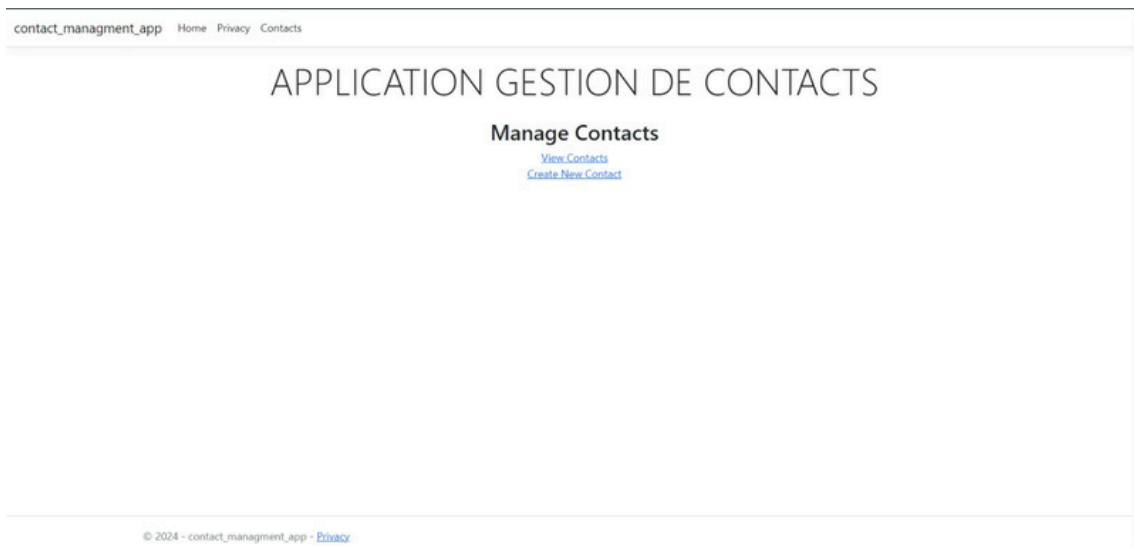
// GET: Contacts/Details/5
0 références
public async Task<IActionResult> Details(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var contact = await _context.Contacts
        .FirstOrDefaultAsync(m => m.Id == id);
    if (contact == null)
    {
        return NotFound();
    }

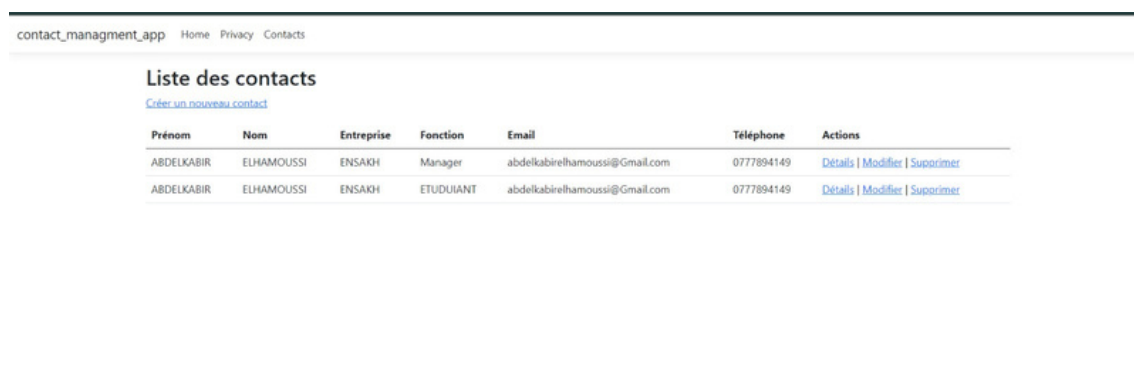
    return View(contact);
}

```

Contact page



liste des Contact page



5. Création des Actions Contrôleur et de la Vue pour Ajouter un Nouveau Contact

L'action Create permet d'ajouter un nouveau contact. Deux méthodes Create ont été définies : une pour afficher le formulaire et une pour traiter les données soumises.

les methdes pourajouter un
nouveau contact



```
40
41 // GET: Contacts/Create
42 0 références
43 public IActionResult Create()
44 {
45     return View();
46 }
```

```
} return View();

// POST: Contacts/Create
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult>
Create([Bind("Id,FirstName,LastName,Entreprise,Fonction,Email,PhoneNumber,Bio,Photo,Adresse")] Contact contact)
{
    if (ModelState.IsValid)
    {
        _context.Add(contact);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(contact);
}
```

Create.cshtml
Formulaire de création de contact.



```
1 @model contact_management_app.Models.Contact
2
3 <h2>Créer un nouveau contact</h2>
4
5 <form asp-action="Create">
6     <div class="form-group">
7         <label asp-for="FirstName" class="control-label"></label>
8         <input asp-for="FirstName" class="form-control" />
9         <span asp-validation-for="FirstName" class="text-danger"></span>
10    </div>
11
12    <div class="form-group">
13        <label asp-for="LastName" class="control-label"></label>
14        <input asp-for="LastName" class="form-control" />
15        <span asp-validation-for="LastName" class="text-danger"></span>
16    </div>
17
18    <div class="form-group">
19        <label asp-for="Entreprise" class="control-label"></label>
20        <input asp-for="Entreprise" class="form-control" />
21        <span asp-validation-for="Entreprise" class="text-danger"></span>
22    </div>
23
24    <div class="form-group">
25        <label asp-for="Fonction" class="control-label"></label>
26        <input asp-for="Fonction" class="form-control" />
27        <span asp-validation-for="Fonction" class="text-danger"></span>
28    </div>
29
30    <div class="form-group">
31        <label asp-for="Email" class="control-label"></label>
32        <input asp-for="Email" class="form-control" />
33        <span asp-validation-for="Email" class="text-danger"></span>
34    </div>
35
36    <div class="form-group">
37        <label asp-for="PhoneNumber" class="control-label"></label>
38        <input asp-for="PhoneNumber" class="form-control" />
39        <span asp-validation-for="PhoneNumber" class="text-danger"></span>
40    </div>
41
42    <input type="submit" value="Créer" />
43 </form>
```

Formulaire de création
de contact.



6. Création des Actions Contrôleur et de la Vue pour Éditer un Contact

Pour éditer un contact existant, deux méthodes Edit ont été définies dans le contrôleur ContactsController.

```

60 }
61
62 // GET: Contacts/Edit/5
63 // 0 références
64 public async Task<ActionResult> Edit(int? id)
65 {
66     if (id == null)
67     {
68         return NotFound();
69     }
70
71     var contact = await _context.Contacts.FindAsync(id);
72     if (contact == null)
73     {
74         return NotFound();
75     }
76     return View(contact);
77 }
78
79 // POST: Contacts/Edit/5
80 [HttpPost]
81 [ValidateAntiForgeryToken]
82 public async Task<ActionResult> Edit(int id, [Bind("Id,FirstName,LastName,Entreprise,Fonction,Email,PhoneNumber,Bio,Photo,Adresse")]

```

les methdes éditer un contact
existant



```

management_app
Create.cshtml Index.cshtml ApplicationDbContext.cs ContactsController.cs Program.cs Delete.cshtml Contacts
YourNamespace.Controllers.ContactsController Create(Contact contact)
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Edit(int id, [Bind("Id,FirstName,LastName,Entreprise,Fonction,Email,PhoneNumber,Bio,Photo,Adresse")]
{
    if (id != contact.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(contact);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!ContactExists(contact.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }

        return RedirectToAction(nameof(Index));
    }

    return View(contact);
}
Aucun problème détecté

```

Edit.cshtml



Formulaire d'édition de contact.

```

Edit.cshtml Details.cshtml Create.cshtml Index.cshtml ApplicationDbContext.cs ContactsController.cs Program.cs Delete.cshtml
contact_management_app
1 @model contact_management_app.Models.Contact
2
3 <h2>Modifier le contact</h2>
4
5 <form asp-action="Edit">
6     <div class="form-group">
7         <label asp-for="FirstName" class="control-label"></label>
8         <input asp-for="FirstName" class="form-control" />
9         <span asp-validation-for="FirstName" class="text-danger"></span>
10    </div>
11
12    <div class="form-group">
13        <label asp-for="LastName" class="control-label"></label>
14        <input asp-for="LastName" class="form-control" />
15        <span asp-validation-for="LastName" class="text-danger"></span>
16    </div>
17
18    <div class="form-group">
19        <label asp-for="Entreprise" class="control-label"></label>
20        <input asp-for="Entreprise" class="form-control" />
21        <span asp-validation-for="Entreprise" class="text-danger"></span>
22    </div>
23
24    <div class="form-group">
25        <label asp-for="Fonction" class="control-label"></label>
26        <input asp-for="Fonction" class="form-control" />
27        <span asp-validation-for="Fonction" class="text-danger"></span>
28    </div>
29
30    <div class="form-group">
31        <label asp-for="Email" class="control-label"></label>
32        <input asp-for="Email" class="form-control" />
33        <span asp-validation-for="Email" class="text-danger"></span>
34    </div>
35
36    <div class="form-group">
37        <label asp-for="PhoneNumber" class="control-label"></label>
38        <input asp-for="PhoneNumber" class="form-control" />
39        <span asp-validation-for="PhoneNumber" class="text-danger"></span>
40    </div>
41
42    <div class="form-group">
43        <label asp-for="Bio" class="control-label"></label>
44        <input asp-for="Bio" class="form-control" />
45        <span asp-validation-for="Bio" class="text-danger"></span>
46    </div>
47
48    <div class="form-group">
49        <label asp-for="Photo" class="control-label"></label>
50        <input asp-for="Photo" class="form-control" />
51        <span asp-validation-for="Photo" class="text-danger"></span>
52    </div>
53
54    <div class="form-group">
55        <label asp-for="Adresse" class="control-label"></label>
56        <input asp-for="Adresse" class="form-control" />
57        <span asp-validation-for="Adresse" class="text-danger"></span>
58    </div>
59
60    <button type="submit" class="btn btn-primary">Modifier</button>
61
62 </form>
Aucun problème détecté

```


7. Création des Actions Contrôleur et de la Vue pour Supprimer un Contact

Enfin, pour supprimer un contact, nous avons défini deux méthodes Delete dans le contrôleur ContactsController.

```
107     }
108     return View(contact);
109 }
110
111 // GET: Contacts/Delete/5
112 public async Task<IActionResult> Delete(int? id)
113 {
114     if (id == null)
115     {
116         return NotFound();
117     }
118
119     var contact = await _context.Contacts
120         .FirstOrDefaultAsync(m => m.Id == id);
121     if (contact == null)
122     {
123         return NotFound();
124     }
125
126     return View(contact);
127 }
```

les methdes pour supprimer un
contact



```
26     return View(contact);
27 }
28
29 // POST: Contacts/DeleteConfirmed
30 [HttpPost]
31 [ValidateAntiForgeryToken]
32 public async Task<IActionResult> DeleteConfirmed(int id)
33 {
34     var contact = await _context.Contacts.FindAsync(id);
35     if (contact == null)
36     {
37         return NotFound();
38     }
39
40     _context.Contacts.Remove(contact);
41     await _context.SaveChangesAsync();
42
43     // Redirect to the index action after successful deletion
44     return RedirectToAction(nameof(Index));
45 }
46
47 1 référence
```

Delete.cshtml

Affiche la page de confirmation
de suppression de contact.



```
1 @model contact_management_app.Models.Contact
2
3 <h2>Supprimer le contact</h2>
4
5 <h3>Êtes-vous sûr de vouloir supprimer ce contact ?</h3>
6 <dl class="row">
7     <dt class="col-sm-2">Prénom</dt>
8     <dd class="col-sm-10">@Model.FirstName</dd>
9
10    <dt class="col-sm-2">Nom</dt>
11    <dd class="col-sm-10">@Model.LastName</dd>
12
13    <dt class="col-sm-2">Entreprise</dt>
14    <dd class="col-sm-10">@Model.Entreprise</dd>
15
16    <dt class="col-sm-2">Fonction</dt>
17    <dd class="col-sm-10">@Model.Fonction</dd>
18
19    <dt class="col-sm-2">Email</dt>
20    <dd class="col-sm-10">@Model.Email</dd>
21
22    <dt class="col-sm-2">Téléphone</dt>
23    <dd class="col-sm-10">@Model.PhoneNumber</dd>
24
25    <!-- Add more properties as needed -->
26 </dl>
27
28 <form asp-action="DeleteConfirmed">
29     <input type="hidden" asp-for="Id" />
30     <div class="form-group">
31         <input type="submit" value="Confirmer la suppression" class="btn btn-danger" />
32         <a asp-action="Index">Annuler</a>
33     </div>
34 </form>
```

Supprimer un Contact

Supprimer le contact

Êtes-vous sûr de vouloir supprimer ce contact ?

Prénom	ABDELKABIR
Nom	ELHAMOUSSI
Entreprise	ENSAKH
Fonction	Manager
Email	abdelkabirelhamoussi@Gmail.com
Téléphone	0777894149

[Confirmer la suppression](#) [Annuler](#)

Détails du contact

Détails du contact

Prénom	ABDELKABIR
Nom	ELHAMOUSSI
Entreprise	ENSAKH
Fonction	Manager
Email	abdelkabirelhamoussi@Gmail.com
Téléphone	0777894149
Adresse	
Bio	
Photo	

[Modifier](#) | [Retour à la liste des contacts](#)

Conclusion

Le développement de cette application de gestion de contacts avec ASP.NET MVC a permis de mettre en pratique les concepts de base de la programmation web et du framework MVC. Les fonctionnalités de création, édition, et suppression de contacts ont été implémentées avec succès, fournissant une base solide pour des extensions futures.