

CHAPITRE 6 :

Le Framework SPA BLAZOR

Qu'est ce BLAZOR

- C'est un Framework Web SPA (Single Page Application) côté client basé sur le Framework .NET, le langage C# et les standards du web comme HTML5 et CSS3,
- D'habitude on utilise les Framework basés sur le langage javascript comme Angular, React, Vue.



Qu'est ce BLAZOR

- Pouvons-nous utiliser C# à la fois pour le développement côté serveur et côté client?



Qu'est ce BLAZOR

- Comment un navigateur peut-il exécuter du code C# ?
- Les navigateurs ne comprennent et n'exécutent que JavaScript. Comment pouvons-nous exécuter du code c# dans le navigateur client? Eh bien, la réponse est **WebAssembly**.



Qu'est ce BLAZOR

- Blazor peut exécuter du code C# directement dans le navigateur, à l'aide de WebAssembly. Il fonctionne dans le même niveau de sécurité que les frameworks JavaScript comme Angular, React, Vue, etc.
- Pas seulement C#, en fait, nous pouvons exécuter n'importe quel type de code dans le navigateur en utilisant WebAssembly.
- WebAssembly est basé sur des standards Web ouverts. C'est donc une partie native de tous les navigateurs modernes, y compris les navigateurs mobiles. Cela signifie que pour que l'application Blazor fonctionne, il n'est pas nécessaire d'installer un plugin spécial.

Qu'est ce BLAZOR

- Il existe deux modèles d'hébergement pour les applications Blazor.

Blazor Hosting Models

A green rounded rectangle containing the text "Blazor WebAssembly".

**Blazor
WebAssembly**

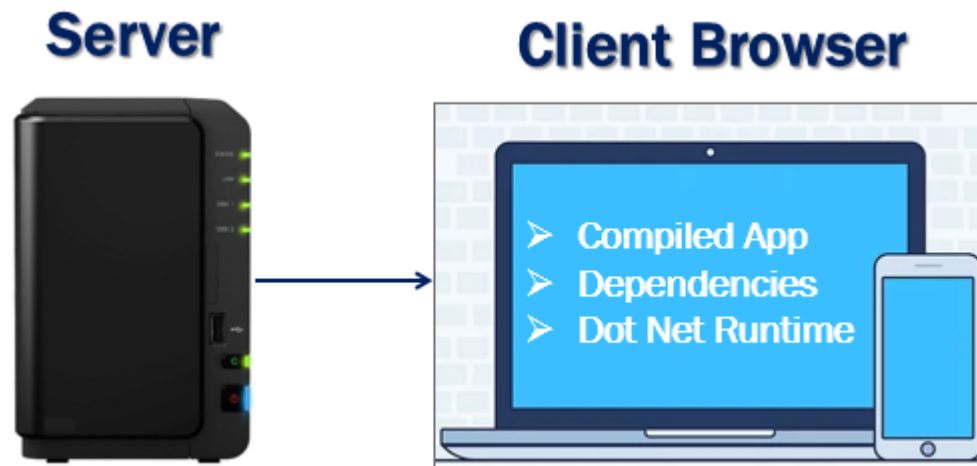
A blue rounded rectangle containing the text "Blazor Server".

**Blazor
Server**

Qu'est ce BLAZOR

- **Modèle d'hébergement Blazor WebAssembly:**

Avec ce modèle d'hébergement, l'application s'exécute directement dans le navigateur sur WebAssembly. Ainsi, tout ce dont l'application a besoin, c'est-à-dire l'application compilée, ses dépendances et le runtime .NET sont téléchargés sur le navigateur client à partir du serveur.

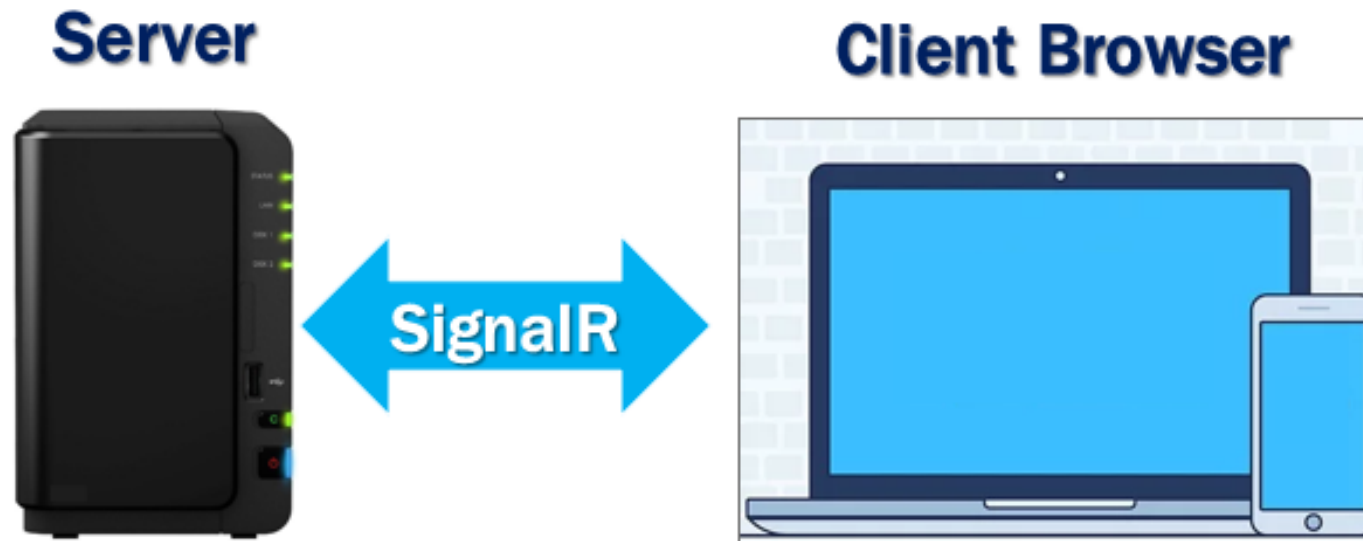


Une application Blazor WebAssembly peut s'exécuter entièrement sur le client sans connexion au serveur ou nous pouvons éventuellement la configurer pour interagir avec le serveur à l'aide d'appels d'API Web ou de SignalR.

Qu'est ce BLAZOR

- **Modèle d'hébergement Blazor Server:**

Avec ce modèle d'hébergement, l'application est exécutée sur le serveur. Entre le client et le serveur, une connexion SignalR est établie. Lorsqu'un événement se produit sur le client tel qu'un clic de bouton par exemple, les informations sur l'événement sont envoyées au serveur via la connexion SignalR








Créer un projet Web BLAZOR


- Ajoutez un nouveau projet serveur Blazor à la solution. Nommez-le EmployeeManagement.Web .
- Ce projet a besoin des classes de modèle que nous avons créées ci-dessus. Il faut donc ajouter une référence à EmployeeManagement.Models.
- Cliquez avec le bouton droit de la souris et définissez EmployeeManagement.Web comme projet de démarrage s'il ne s'agit pas déjà du projet de démarrage.

Créer un projet Web Blazor

Ajouter un nouveau projet


Modèles de projet récents


	Bibliothèque de classes (.NET Standard)	C#
	Application Blazor	C#
	Application web ASP.NET Core	C#
	Application Windows Forms (.NET Framework)	C#
	Application console (.NET Framework)	C#

Rechercher des modèles (Alt+S) 

Tout effacer

C# Toutes les plateformes Web

**Application web ASP.NET Core**
Modèles de projet permettant de créer des applications web ASP.NET Core et des API web pour Windows, Linux et macOS à l'aide du .NET Core ou du .NET Framework. Créez des applications web avec Razor Pages, MVC ou des SPA (applications monopages) via Angular, React ou React + Redux.
C# Linux macOS Windows Cloud Service Web

**Application Blazor**
Modèles de projet pour la création d'applications Blazor qui s'exécutent sur le serveur dans une application ASP.NET Core ou dans le navigateur de WebAssembly (wasm). Vous pouvez utiliser ces modèles pour générer des applications web ayant des IU (interfaces utilisateur) dynamiques riches.
C# Linux macOS Windows Cloud Web

Service gRPC

Créer un projet Web Blazor

Créer une application Blazor



Application serveur Blazor

Modèle de projet permettant de créer une application serveur Blazor qui s'exécute côté serveur dans une application ASP.NET Core, et qui gère les interactions utilisateur via une connexion SignalR. Vous pouvez utiliser ce modèle pour les applications web ayant des IU (interfaces utilisateur) dynamiques riches.



Blazor WebAssembly App

Modèle de projet permettant de créer une application Blazor qui s'exécute sur WebAssembly. Vous pouvez utiliser ce modèle pour les applications web ayant des IU (interfaces utilisateur) dynamiques riches.

Authentification

Aucune authentification

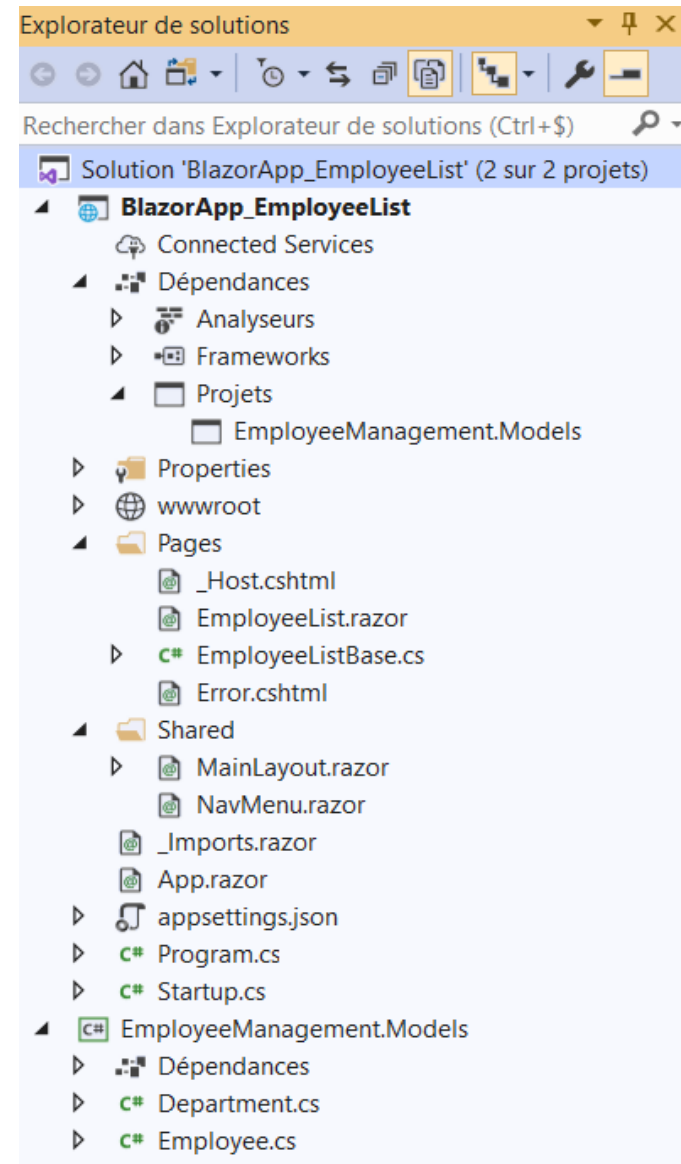
[Changer](#)

Avancé

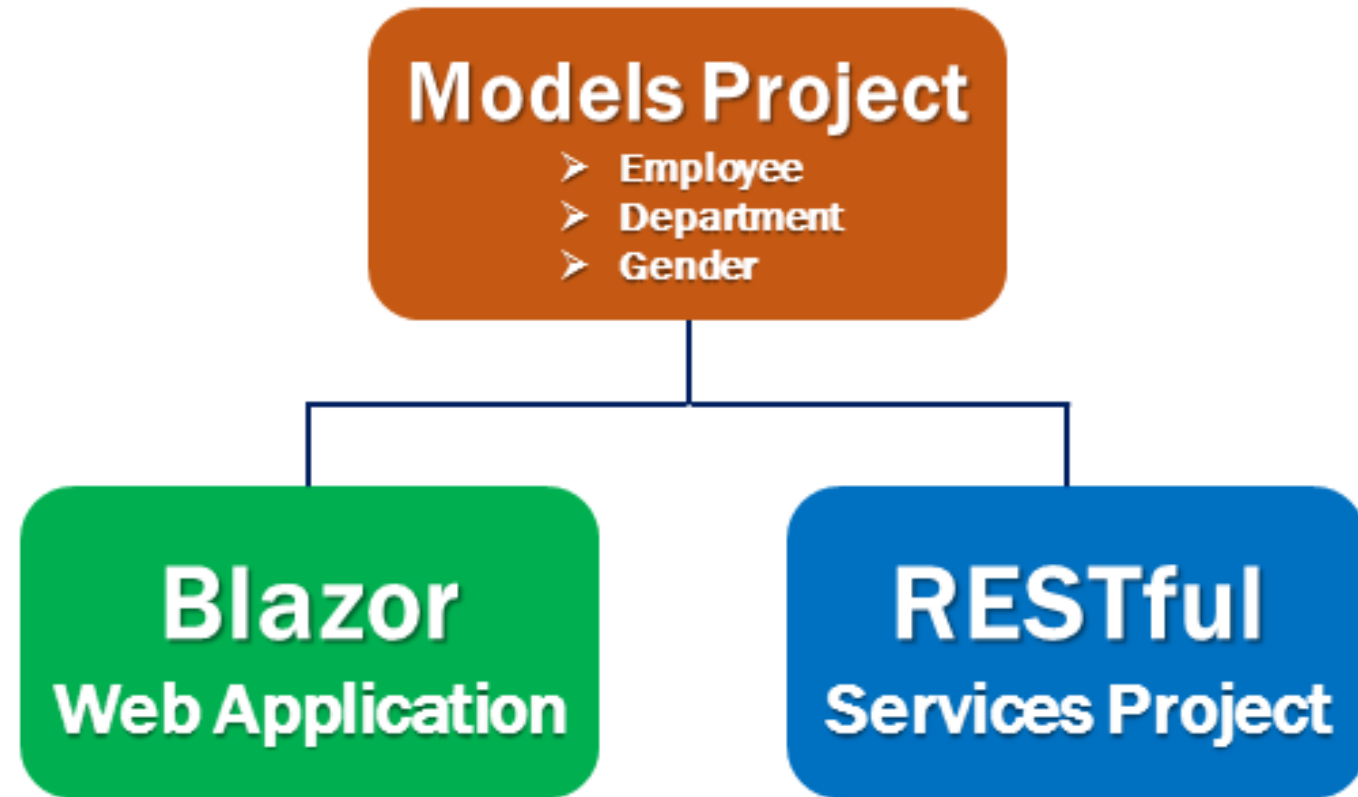
☒ Configurer pour HTTPS

☐ Activer le support de Docker
(Nécessite [Docker Desktop](#))

Linux

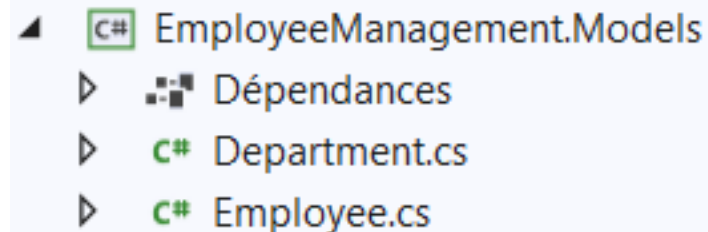


Model classes



Model classes

- L'objectif de ce chapitre est de créer une application cliente Blazor et consommer le service HTTP RESTAPI déjà créé dans la chapitre précédent afin de gérer les opérations d'ajout, recherche, modification et suppression sur les employés.
- On commence par créer les classes de modèle de l'application cliente dans un projet de type bibliothèque de classes standard. Puis on crée notre application web blazor.
- Créez un nouveau projet de bibliothèque de classes standard .NET. Nommez le projet EmployeeManagement.Models . Nommez la solution EmployeesManagment



Model classes

```
namespace EmployeeManagement.Models
```

```
{
```

```
10 références
```

```
public class Employee
```

```
{
```

```
4 références
```

```
public int EmployeeId { get; set; }
```

```
5 références
```

```
public string FirstName { get; set; }
```

```
5 références
```

```
public string LastName { get; set; }
```

```
4 références
```

```
public string Email { get; set; }
```

```
4 références
```

```
public DateTime DateOfBrith { get; set; }
```

```
4 références
```

```
public Gender Gender { get; set; }
```

```
4 références
```

```
public Department Department { get; set; }
```

```
5 références
```

```
public string PhotoPath { get; set; }
```

```
}
```

```
5 références
```

```
public enum Gender
```

```
{
```

```
Male,
```

```
Female,
```

```
Other
```

```
}
```

```
namespace EmployeeManagement.Models
```

```
{
```

```
5 références
```

```
public class Department
```

```
{
```

```
4 références
```

```
public int DepartmentId { get; set; }
```

```
4 références
```

```
public string DepartmentName { get; set; }
```

```
}
```

```
}
```

Ajout de référence

The image shows a Visual Studio interface with two main components: a 'Gestionnaire de références' (Reference Manager) dialog box and the 'Explorateur de solutions' (Solution Explorer).

Gestionnaire de références - BlazorApp_EmployeeList

This dialog box is used to manage project references. It shows a table of references and a search bar.

Projets	
Solution	
Projets partagés	
COM	
Parcourir	

Nom	Chemin d'accès
<input checked="" type="checkbox"/> EmployeeManagement.Models	C:\Users\USER\Desкто...

Rechercher (Ctrl+E)

Nom : EmployeeManagement.Models

Buttons: Parcourir..., OK, Annuler

Explorateur de solutions

The Solution Explorer shows the project structure for 'BlazorApp_EmployeeList' (2 sur 2 projets). The project is expanded, showing the following files and folders:

- Solution 'BlazorApp_EmployeeList' (2 sur 2 projets)
 - BlazorApp_EmployeeList
 - Connected Services
 - Dépendances
 - Analyseurs
 - Frameworks
 - Projets
 - EmployeeManagement.Models
 - Properties
 - wwwroot
 - Pages
 - _Host.cshtml
 - EmployeeList.razor
 - EmployeeListBase.cs
 - Error.cshtml
 - Shared
 - MainLayout.razor
 - NavMenu.razor
 - _Imports.razor
 - App.razor
 - appsettings.json
 - Program.cs
 - Startup.cs
 - EmployeeManagement.Models
 - Dépendances
 - Department.cs
 - Employee.cs

Supprimer les fichiers par défaut du projet

- Mettre EmployeeManagement.Web comme projet de démarrage et supprimer les composants et dossiers suivants du projet :
 - ✓ Le dossier Data
 - ✓ Pages/Counter.razor
 - ✓ Pages/FetchData.razor
 - ✓ Pages/Index.razor
 - ✓ Shared/SurveyPrompt.razor
- Supprimer de la classe startup: `using EmployeeManagement.Web.Data;`
- Supprimez de la méthode ConfigureServices() :
`services.AddSingleton<WeatherForecastService>();`

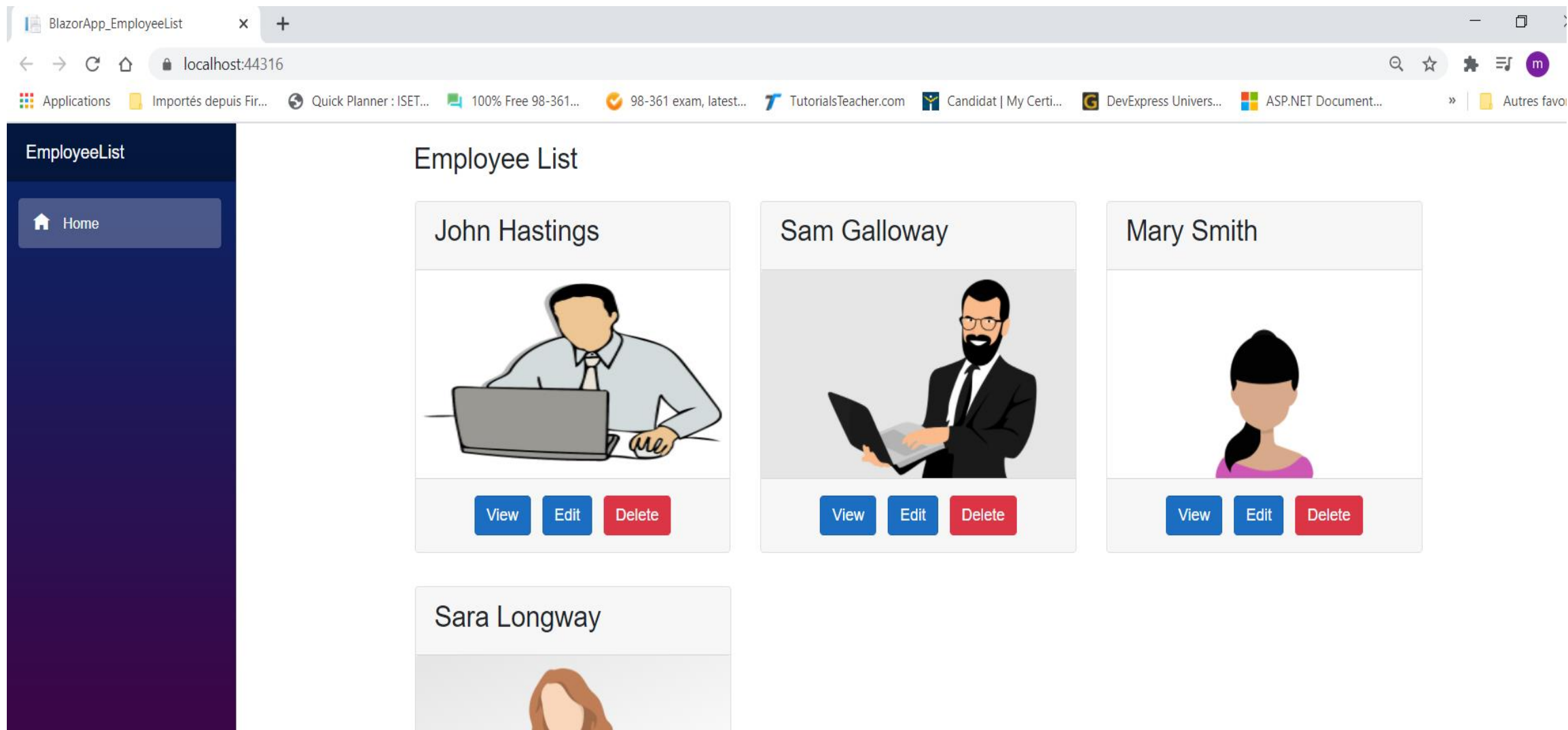
Créer un premier composant (EmployeeList)

- Dans le projet EmployeeManagement.Web , cliquez avec le bouton droit sur le dossier Pages et ajoutez un nouveau composant razor.
- Nommez-le EmployeeList.razor, C'est ce composant que nous utiliserons pour afficher la liste des employés.
- Inclure la directive `@page "/"` dans le fichier EmployeeList.razor. Cela indique à Blazor de rendre ce composant lorsque nous naviguons vers l'URL racine de l'application.
- Changements dans NavMenu.razor: Supprimez les 2 éléments suivants du menu de navigation.

Créer un premier composant (EmployeeList)

```
<li class="nav-item px-3">
  <NavLink class="nav-link" href="counter">
    <span class="oi oi-plus" aria-hidden="true"></span> Counter
  </NavLink>
</li>
<li class="nav-item px-3">
  <NavLink class="nav-link" href="fetchdata">
    <span class="oi oi-list-rich" aria-hidden="true"></span> Fetch data
  </NavLink>
</li>
```

Créer un premier composant (EmployeeList)



Code de EmployeeList.razor

```
1  @page "/EmployeeList"
2  @inherits EmployeeListBase
3
4  <h3>Employee List</h3>
5
6  <div class="card-group">
7      @foreach (var employee in Employees)
8      {
9          <div class="card m-3" style="min-width: 18rem; max-width:30.5%;">
10             <div class="card-header">
11                 <h3>@employee.FirstName @employee.LastName </h3>
12             </div>
13             
14             <div class="card-footer text-center">
15                 <a href="@($"EmployeeDetails/{employee.EmployeeId}")" class="btn btn-primary m-1">View</a>
16
17                 <a href="@($"editemployee/{employee.EmployeeId}")" class="btn btn-primary m-1">Edit</a>
18
19                 <a href="@($"deleteemployee/{employee.EmployeeId}")" class="btn btn-primary m-1">Delete</a>
20             </div>
21         </div>
22     }
23 </div>
```

Code de EmployeeListBase.cs

```
using EmployeeManagement.Models;
using Microsoft.AspNetCore.Components;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace BlazorApp_EmployeeList.Pages
{
    public class EmployeeListBase : ComponentBase
    {
        public IEnumerable<Employee> Employees { get; set; }

        protected override Task OnInitializedAsync()
        {
            LoadEmployees();
            return base.OnInitializedAsync();
        }

        private void LoadEmployees()
        {
            Employee e1 = new Employee
            {
                EmployeeId = 1,
                FirstName = "John",
                LastName = "Hastings",
                Email = "David@pragimtech.com",
                DateOfBrith = new DateTime(1980, 10, 5),
                Gender = Gender.Male,
                Department = new Department { DepartmentId = 1, DepartmentName = "IT" },
                PhotoPath = "images/john.png"
            };
        }
    }
}
```

Code de EmployeeList.razor

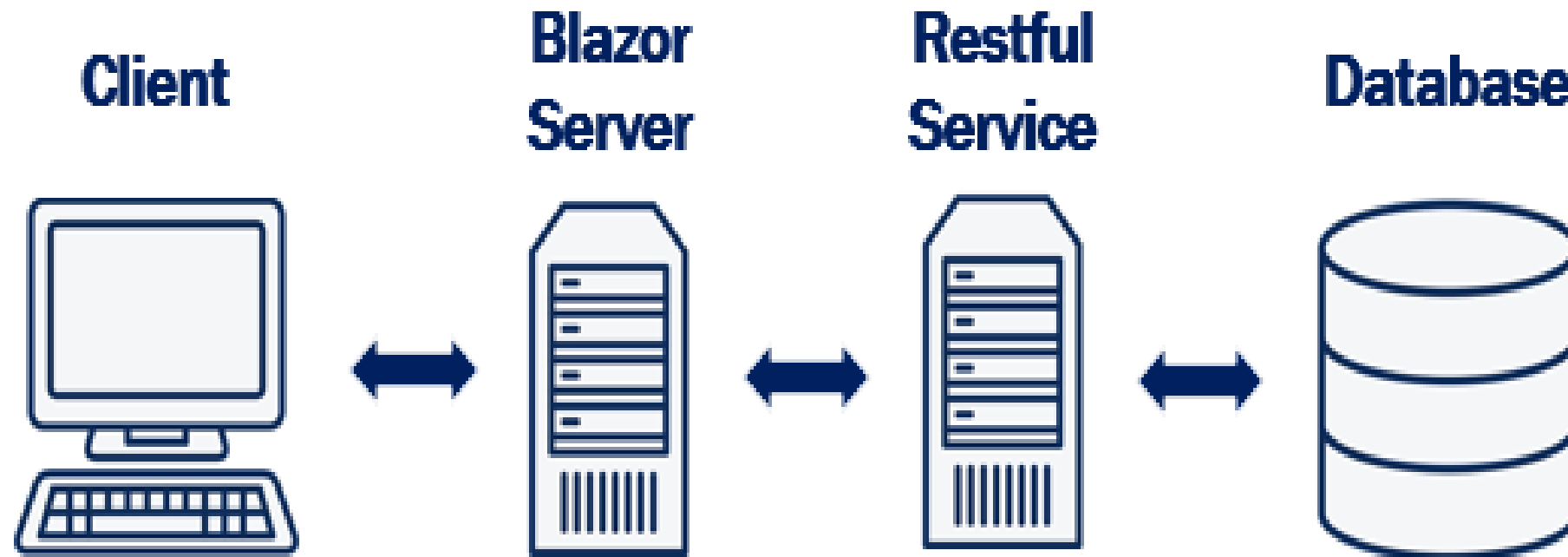
```
Employee e2 = new Employee
{
    EmployeeId = 2,
    FirstName = "Sam",
    LastName = "Galloway",
    Email = "Sam@pragimtech.com",
    DateOfBrith = new DateTime(1981, 12, 22),
    Gender = Gender.Male,
    Department = new Department
    { DepartmentId = 2, DepartmentName = "HR" },
    PhotoPath = "images/sam.jpg"
};

Employee e3 = new Employee
{
    EmployeeId = 3,
    FirstName = "Mary",
    LastName = "Smith",
    Email = "mary@pragimtech.com",
    DateOfBrith = new DateTime(1979, 11, 11),
    Gender = Gender.Female,
    Department = new Department
    { DepartmentId = 1, DepartmentName = "IT" },
    PhotoPath = "images/mary.png"
};

Employee e4 = new Employee
{
    EmployeeId = 3,
    FirstName = "Sara",
    LastName = "Longway",
    Email = "sara@pragimtech.com",
    DateOfBrith = new DateTime(1982, 9, 23),
    Gender = Gender.Female,
    Department = new Department
    { DepartmentId = 3, DepartmentName = "Payroll" },
    PhotoPath = "images/sara.png"
};

Employees = new List<Employee> { e1, e2, e3, e4 };
}
```

Consommer un service Rest API dans une app BLAZOR



Consommer un service Rest API dans une app BLAZOR

- Un composant Blazor peut consommer directement un service Restfull WebApi mais pour des raisons de séparation des couches et une meilleure architecture de notre application il faut plutôt créer un service qui va interagir avec le Rest API.
- Ajoutez un dossier avec le nom Services au projet d'application Web Blazor. Ajouter les 2 fichiers de classe suivants à ce dossier : IEmployeeService.cs et EmployeeService.cs

```
public interface IEmployeeService
{
    Task<IEnumerable<Employee>> GetEmployees();
}
```

- Créer la classe EmployeeService.cs

Consommer un service Rest API dans une app BLAZOR

```
using EmployeeManagement.Models;
using System.Collections.Generic;
using System.Net.Http;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Components;
using System.Net.Http.Json;

namespace BlazorApp_EmployeeList.services
{
    public class EmployeeService : IEmployeeService
    {
        private readonly HttpClient httpClient;

        public EmployeeService(HttpClient httpClient)
        {
            this.httpClient = httpClient;
        }

        public async Task<IEnumerable<Employee>> GetEmployees()
        {
            return await httpClient.GetFromJsonAsync<Employee[]>("api/employees");
        }
    }
}
```

Consommer un service Rest API dans une app BLAZOR

- Nous utilisons la classe **HttpClient** pour appeler le service API REST. Cette classe se trouve dans l' espace de noms **System.Net.Http** .
- HttpClient est injecté dans EmployeeService à l' aide de l'injection de dépendances.
- Enregistrer le service **HttpClient** avec le conteneur d'injection de dépendances.
- Nous **utilisons** **httpClient.GetFromJsonAsync** pour appeler l'API REST. Cette méthode se trouve dans le Namespace **System.Net.Http.Json**, N'oubliez pas d'inclure l'espace de noms **Microsoft.AspNetCore.Components** dans la classe EmployeeListBase.
- Transmettez le point de terminaison de l'API REST (api/employees) à la méthode **httpClient.GetFromJsonAsync** .

```
httpClient.GetFromJsonAsync<Employee[]>("api/employees")
```

Consommer un service Rest API dans une app BLAZOR

- Enregistrer les services HttpClient dans program.cs

```
builder.Services.AddHttpClient<IEmployeeService, EmployeeService>(client =>  
    {  
        client.BaseAddress = new Uri("https://localhost:7147/");  
    });
```

```
var app = builder.Build();
```

```
// Configure the HTTP request pipeline.
```

```
if (!app.Environment.IsDevelopment())  
{
```

Consommer un service Rest API dans une app BLAZOR

- Enfin, **appelez le service `IEmployeeService` à partir du composant blazor `EmployeeList`** .
- Nous utilisons l'attribut **[Inject]** pour injecter un service dans un composant Blazor. Nous ne pouvons pas utiliser de constructeur pour cela.
- Dans la méthode **`OnInitializedAsync`** du composant , nous appelons la méthode `EmployeeService.GetEmployees` .
- Les données (liste d'employés) renvoyées par cette méthode sont ensuite utilisées pour initialiser la propriété `Employees` .
- Le composant blazor `EmployeeList` se lie à la propriété `Employees` pour afficher la liste des employés.

Consommer un service Rest API dans une app BLAZOR

```
using BlazorApp_EmployeeList.services;
using EmployeeManagement.Models;
using Microsoft.AspNetCore.Components;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace BlazorApp_EmployeeList.Pages
{
    public class EmployeeListBase : ComponentBase
    {
        [Inject]
        public IEmployeeService EmployeeService { get; set; }
        public IEnumerable<Employee> Employees { get; set; }
        protected override async Task OnInitializedAsync()
        {
            Employees = (await EmployeeService.GetEmployees()).ToList();
        }
    }
}
```

Employee List

ZRIBI MALEK



View

Edit

Delete

TRIKI MOHAMED ALI



View

Edit

Delete

Abedlli Sleh



View

Edit

Delete

Abdou Salah



View

Edit

Delete

oussema ali



View

Edit

Delete

malek ben Sleh



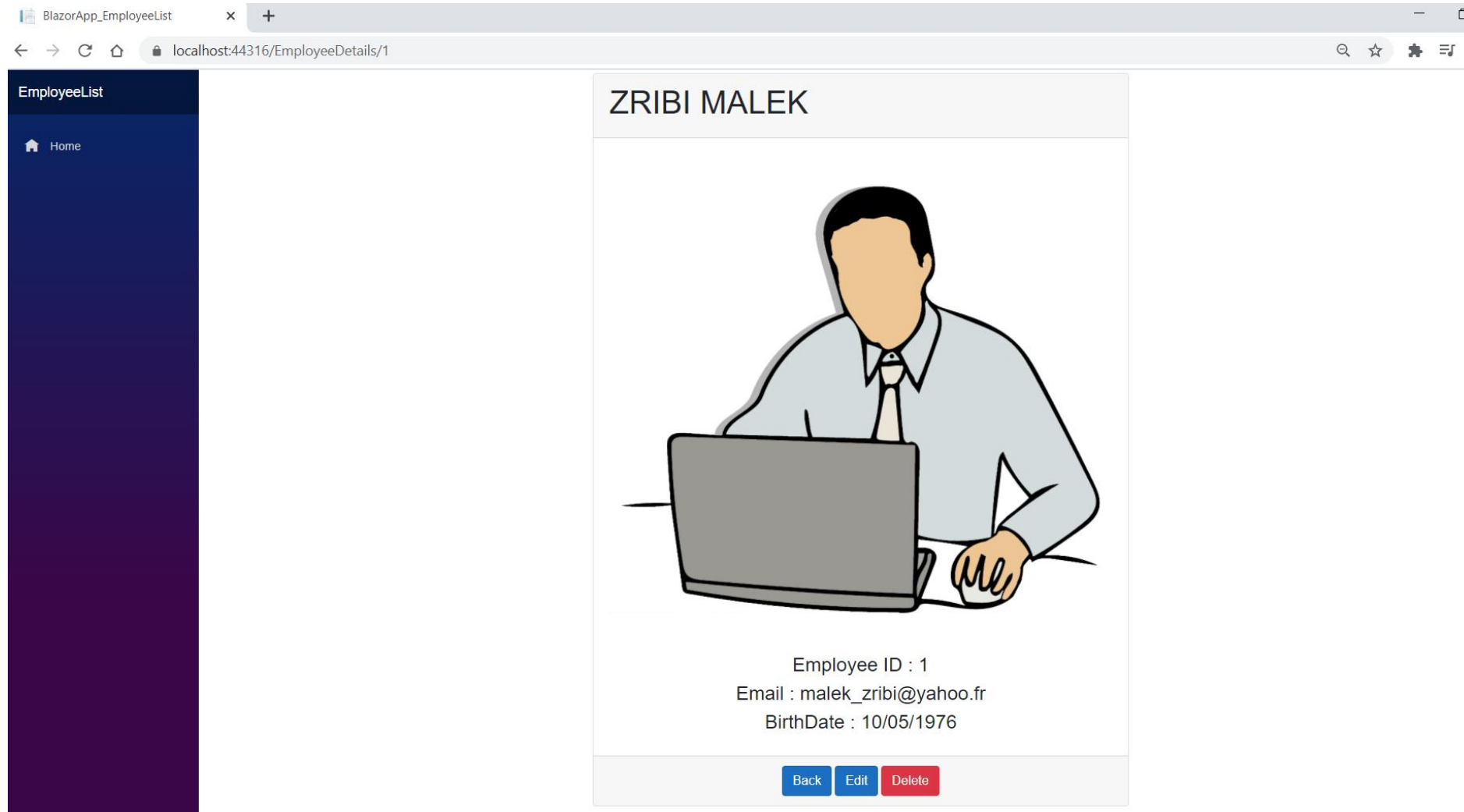
View

Edit

Delete

Afficher Détails d'un Employé

- On veut maintenant afficher les détails d'un employé en cliquant sur le bouton View



Afficher Détails d'un Employé

- Ajouter un nouveau composant razor dans le dossier Pages nommé EmployeeDetails.razor

```
EmployeeDetailsBase.cs EmployeeDetails.razor X
1  @page "/EmployeeDetails/{id}"
2  @inherits EmployeeDetailsBase
3
4  <div class="row justify-content-center m-3">
5      <div class="col-sm-8">
6          <div class="card">
7              <div class="card-header">
8                  <h1>@Employee.FirstName @Employee.LastName</h1>
9              </div>
10
11              <div class="card-body text-center">
12                  
13
14                  <h4>Employee ID : @Employee.EmployeeId</h4>
15                  <h4>Email : @Employee.Email</h4>
16                  <h4>BirthDate : @Employee.DateOfBirth.ToShortDateString()</h4>
17              </div>
18              <div class="card-footer text-center">
19                  <a href="/" class="btn btn-primary">Back</a>
20                  <a href="#" class="btn btn-primary">Edit</a>
21                  <a href="#" class="btn btn-danger">Delete</a>
22              </div>
23          </div>
24      </div>
25  </div>
```


Afficher Détails d'un Employé

- Ajouter une classe pour le code behind du composant nommée EmployeeDetailsBase.cs

```
using BlazorApp_EmployeeList.services;
using EmployeeManagement.Models;
using Microsoft.AspNetCore.Components;
using System.Threading.Tasks;

namespace BlazorApp_EmployeeList.Pages
{
    public class EmployeeDetailsBase : ComponentBase
    {
        public Employee Employee { get; set; } = new Employee();
        [Inject]
        public IEmployeeService EmployeeService { get; set; }
        [Parameter]
        public string Id { get; set; }
        protected async override Task OnInitializedAsync()
        {
            Id = Id ?? "1";
            Employee = await EmployeeService.GetEmployee(int.Parse(Id));
        }
    }
}
```

Afficher Détails d'un Employé

- Ajouter la méthode nécessaire dans le service `IEmployeeService` et la classe `EmployeeService`

```
public interface IEmployeeService
{
    Task<IEnumerable<Employee>> GetEmployees();
    Task<Employee> GetEmployee(int id);
}

public class EmployeeService : IEmployeeService
{
    private readonly HttpClient httpClient;
    public EmployeeService(HttpClient httpClient)
    {
        this.httpClient = httpClient;
    }
    public async Task<IEnumerable<Employee>> GetEmployees()
    {
        return await httpClient.GetFromJsonAsync<Employee[]>("api/employees");
    }
    public async Task<Employee> GetEmployee(int id)
    {
        return await httpClient.GetFromJsonAsync<Employee>($"api/employees/{id}");
    }
}
```

Afficher les Détails d'un Employé

- Dans le composant EmployeeList ajouter le lien View suivant qui va lancer l'appel du composant EmployeeDetails avec spécification de l'Id :

```
<a href="@($"EmployeeDetails/{employee.EmployeeId}")" class="btn btn-primary m-1">View</a>
```

- Si on ne spécifie pas l'ID de l'employé dans l'url, on peut afficher une valeur par défaut, pour cela il faut ajouter une directive page dans le fichier razor

```
@page "/EmployeeDetails"
```

Afficher des données de deux tables

- Si on veut afficher pour chaque employé le nom de son département, il faut ajouter une propriété de navigation Department dans la classe Employee dans le projet webapi.

```
public class Employee
{
    public int EmployeeId { get; set; }
    [Required]
    [StringLength(100, MinimumLength = 2)]
    public string FirstName { get; set; }
    [Required]
    public string LastName { get; set; }
    [Required]
    public string Email { get; set; }
    public DateTime DateOfBirth { get; set; }
    public Gender Gender { get; set; }
    public int DepartmentId { get; set; }
    public string PhotoPath { get; set; }
    public Department Department { get; set; }
}
```

Afficher des données de deux tables

- Ajouter la contrainte Foreign key entre les deux tables dans votre BD pour cela procéder à une migration.
- changer la méthode GetEmployee(int id) dans EmployeeRepository en ajoutant la méthode Include pour inclure le département de chaque employé :

```
public async Task<Employee> GetEmployee(int employeeId)
{
    return await appDbContext.Employees
        .Include(e => e.Department)
        .FirstOrDefaultAsync(e => e.EmployeeId == employeeId);
}
```

Afficher des données de deux tables

```
EmployeeDetails.razor* X
1  @page "/EmployeeDetails/{id}"
2  @page "/EmployeeDetails"
3  @inherits EmployeeDetailsBase
4
5  @if (Employee == null || Employee.Department == null)
6  {
7      <div class="spinner"></div>
8  }
9  else
10 {
11     <div class="row justify-content-center m-3">
12     <div class="col-sm-8">
13         <div class="card">
14             <div class="card-header">
15                 <h1>@Employee.FirstName @Employee.LastName</h1>
16             </div>
17             <div class="card-body text-center">
18                 
19
20                 <h4>Employee ID : @Employee.EmployeeId</h4>
21                 <h4>Email : @Employee.Email</h4>
22                 <h4>BirthDate : @Employee.DateOfBrith.ToShortDateString()</h4>
23                 <h4>Departement : @Employee.Department.DepartmentName</h4>
24             </div>
25             <div class="card-footer text-center">
26                 <a href="/" class="btn btn-primary">Back</a>
27                 <a href="#" class="btn btn-primary">Edit</a>
28                 <a href="#" class="btn btn-danger">Delete</a>
29             </div>

```

A stylized illustration of a man with dark hair, wearing a light blue long-sleeved shirt and a light-colored tie. He is seated at a desk, facing forward, with his hands on a laptop. His right hand is on the mouse. The illustration uses simple black outlines and flat colors.

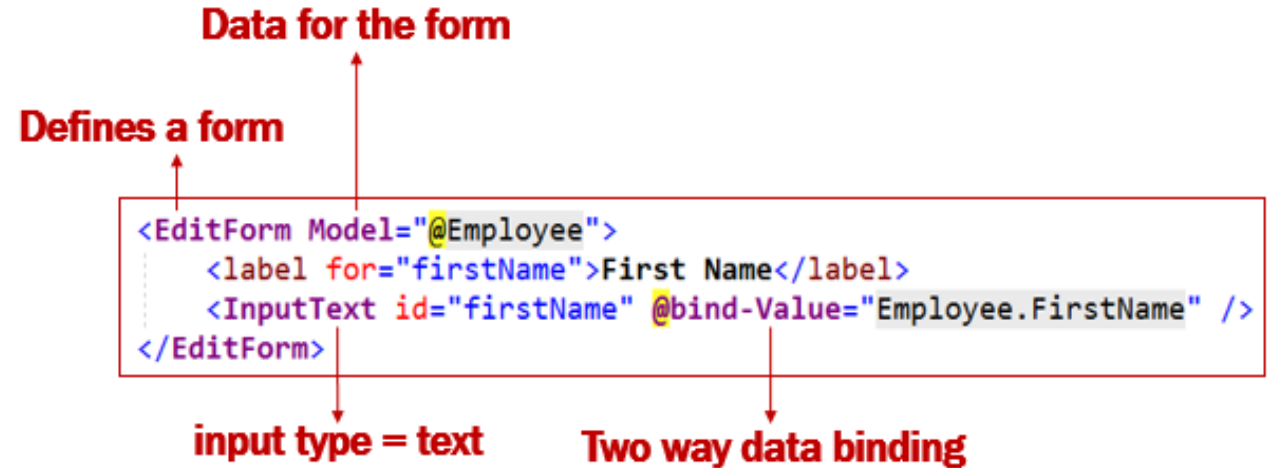
Departement : INFO

Ajout du composant Edit Employee

- Pour implémenter la fonctionnalité Edit pour un employé, on ajoute un nouveau composant razor nommé EditEmployee.
- Changer le lien Edit dans les composants EmployeeList et EmployeeDetails

```
<a href="@($"editemployee/{employee.EmployeeId}")" class="btn btn-primary m-1">Edit</a>
```

Input text element	InputText
Multiline textbox (textarea)	InputTextArea
Daterangepicker	InputDate
Checkbox	InputCheckbox
Input element for numbers	InputNumber
Dropdownlist	InputSelect



EditEmployee.razor

```
1  @page "/editemployee/{id}"
2  @inherits EditEmployeeBase
3
4  <EditForm Model="@Employee">
5      <h3>Edit Employee</h3>
6      <hr />
7      <div class="form-group row">
8          <label for="firstName" class="col-sm-2 col-form-label">
9              First Name
10          </label>
11          <div class="col-sm-10">
12              <InputText id="firstName" class="form-control" placeholder="First Name"
13                  @bind-Value="Employee.FirstName" />
14          </div>
15      </div>
16      <div class="form-group row">
17          <label for="lastName" class="col-sm-2 col-form-label">
18              Last Name
19          </label>
20          <div class="col-sm-10">
21              <InputText id="lastName" class="form-control" placeholder="Last Name"
22                  @bind-Value="Employee.LastName" />
23          </div>
24      </div>
25      <div class="form-group row">
26          <label for="email" class="col-sm-2 col-form-label">
27              Email
28          </label>
29          <div class="col-sm-10">
30              <InputText id="email" class="form-control" placeholder="Email"
31                  @bind-Value="Employee.Email" />
32          </div>
33      </div>
34  </EditForm>
```

```
1  using BlazorApp_EmployeeList.services;
2  using EmployeeManagement.Models;
3  using Microsoft.AspNetCore.Components;
4  using System.Threading.Tasks;
5
6  namespace BlazorApp_EmployeeList.Pages
7  {
8      2 références
9      public class EditEmployeeBase : ComponentBase
10     {
11         14 références
12         public Employee Employee { get; set; } = new Employee();
13
14         [Inject]
15         1 référence
16         public IEmployeeService EmployeeService { get; set; }
17
18         [Parameter]
19         1 référence
20         public string Id { get; set; }
21
22         2 références
23         protected async override Task OnInitializedAsync()
24         {
25             Employee = await EmployeeService.GetEmployee(int.Parse(Id));
26         }
27     }
28 }
```

Edit Employee

BlazorApp_EmployeeList x +

localhost:44316/editemployee/1

EmployeeList

Home

Edit Employee

First Name	<input type="text" value="ZRIBI"/>
Last Name	<input type="text" value="MALEK"/>
Email	<input type="text" value="malek_zribi@yahoo.fr"/>

Binding select element with database data

- On vous demande d'ajouter une liste de choix pour les départements dans le composant EditEmployee.

Edit Employee

First Name

Sara

Last Name

Longway

Email

sara@pragimtech.com

Department

Payroll

IT

HR

Payroll

Admin

Binding select element with database data

- Dans EditEmployee.razor ajouter le code suivant pour injecter la propriété Department:

```
<div class="form-group">
  <div class="form-control">
    <InputText id="email" class="form-control" placeholder="Email"
      @bind-Value="Employee.Email" />
  </div>
</div>
<div class="form-group row">
  <div class="col-sm-2 col-form-label">
    Department
  </div>
  <div class="col-sm-10">
    <InputSelect id="department" @bind-Value="DepartmentId" class="form-control">
      @foreach (var dept in Departments)
      {
        <option value="@dept.DepartmentId">@dept.DepartmentName</option>
      }
    </InputSelect>
  </div>
</div>
</EditForm>
```

Binding select element with database data

- Edit Employee Component Class
(EditEmployeeBase.cs):

```
public class EditEmployeeBase : ComponentBase
{
    [Inject]
    1 référence
    public IEmployeeService EmployeeService { get; set; }

    15 références
    public Employee Employee { get; set; } = new Employee();

    [Inject]
    1 référence
    public IDepartmentService DepartmentService { get; set; }

    2 références
    public List<Department> Departments { get; set; } = new List<Department>();

    5 références
    public string DepartmentId { get; set; }

    [Parameter]
    1 référence
    public string Id { get; set; }

    2 références
    protected async override Task OnInitializedAsync()
    {
        Employee = await EmployeeService.GetEmployee(int.Parse(Id));
        Departments = (await DepartmentService.GetDepartments()).ToList();
        DepartmentId = Employee.Department.DepartmentId.ToString();
    }
}
```

Binding select element with database data

WebApi Project

```
public interface IDepartmentRepository
{
    2 références
    Task<IEnumerable<Department>> GetDepartments();
    2 références
    Task<Department> GetDepartment(int departmentId);
}
```

```
public class DepartmentRepository : IDepartmentRepository
{
    private readonly AppDbContext appDbContext;

    0 références
    public DepartmentRepository(AppDbContext appDbContext)
    {
        this.appDbContext = appDbContext;
    }

    2 références
    public async Task<Department> GetDepartment(int departmentId)
    {
        return await appDbContext.Departments
            .FirstOrDefaultAsync(d => d.DepartmentId == departmentId);
    }

    2 références
    public async Task<IEnumerable<Department>> GetDepartments()
    {
        return await appDbContext.Departments.ToListAsync();
    }
}
```

Binding select element with database data

- Dans le projet webapi, ajouter le contrôleur api DepartmentController.

```
[Route("api/[controller]")]
[ApiController]
public class DepartmentsController : ControllerBase
{
    private readonly IDepartmentRepository departmentRepository;

    public DepartmentsController(IDepartmentRepository departmentRepository)
    {
        this.departmentRepository = departmentRepository;
    }

    [HttpGet]
    public async Task<ActionResult> GetDepartments()
    {
        try
        {
            return Ok(await departmentRepository.GetDepartments());
        }
        catch (Exception)
        {
            return StatusCode(StatusCode.Status500InternalServerError,
                "Error retrieving data from the database");
        }
    }
}
```


Binding select element with database data

```
[HttpGet("{id:int}")]
public async Task<ActionResult<Department>> GetDepartment(int id)
{
    try
    {
        var result = await departmentRepository.GetDepartment(id);

        if (result == null)
        {
            return NotFound();
        }

        return result;
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError,
            "Error retrieving data from the database");
    }
}
```

Binding select element with database data

- Dans le projet Blazor, ajouter l'interface IDepartmentService dans le dossier Services

```
public interface IDepartmentService
{
    Task<IEnumerable<Department>> GetDepartments();
    Task<Department> GetDepartment(int id);
}

public class DepartmentService : IDepartmentService
{
    private readonly HttpClient httpClient;

    public DepartmentService(HttpClient httpClient)
    {
        this.httpClient = httpClient;
    }

    public async Task<Department> GetDepartment(int id)
    {
        return await httpClient.GetFromJsonAsync<Department>($"api/departments/{id}");
    }

    public async Task<IEnumerable<Department>> GetDepartments()
    {
        return await httpClient.GetFromJsonAsync<Department[]>("api/departments");
    }
}
```

Binding select element with database data

- Dans la classe startup du projet Blazor, n'oubliez pas d'enregistrer le service HttpClient IDepartmentService:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddRazorPages();
    services.AddServerSideBlazor();
    services.AddHttpClient<IEmployeeService, EmployeeService>(client =>
    {
        client.BaseAddress = new Uri("https://localhost:44369/");
    });
    services.AddHttpClient<IDepartmentService, DepartmentService>(client =>
    {
        client.BaseAddress = new Uri("https://localhost:44369/");
    });
}
```