



**HE2B-ISIB**

MA1

**DBA**

**Ibrahim Mboula  
(52676)**

Professeur : Ing. S. Rodrigues

Année académique 2022-2023

---

## Table des matières

▼ 1. Installation de SQL server .....	2
▼ 1.1. Création d'une base de donnée, commandes de base .....	2
▼ 2. Choix d'un projet .....	3
3. Création des tables et respect des notions de gestion d'une base de données .....	3
3.1 Liste de tables .....	3
▼ 3.2 Création des tables sur SQL server.....	3
▼ 4. Création de types de données définis par l'utilisateur (TDU) .....	9
▼ 4.1 Modification de la table Employes.....	9
▼ 5. Définir des contraintes .....	10
▼ 5.1. Les Vues .....	11
▼ 6. Procédures stockées (traitement sur les données) / déclencheurs (Triggers).....	12
▼ 6.1. Procédures stockées .....	12
▼ 7. Fonctions utilisateur .....	14
▼ 8. Sauvegarde (manuel / automatique) / restauration .....	15
▼ ..... .....	16
8.2. Sauvegarde automatique.....	16
▼ 8.3. Restauration.....	17
▼ 9. Gestion des données à partir d'une page web (ASP) .....	20
▼ 9.1. Création du projet et de la base de données.....	20
▼ 9.2. Création des pages ASP.....	23

## ▼ 1. Installation de SQL server

L'ensemble de ce travail sera réalisé avec SQL server Developer



### Developer

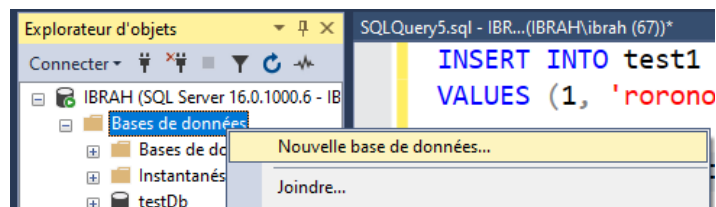
SQL Server 2022 Developer is a full-featured free edition, licensed for use as a development and test database in a non-production environment.

[Download now](#)

Après l'installation, je crée une nouvelle base de données appelée testDb. Dans cette base de données, je crée une table nommée test1 et j'y insère des données.

### ▼ 1.1. Creation d'une base de donnée, commandes de base

Pour créer une base de donnée.



Script SQL pour créer la table et lui donner des collones:

```
CREATE TABLE test1 (  
  id INT PRIMARY KEY,  
  nom VARCHAR(50),  
  age INT,  
  email VARCHAR(100)  
);
```

Script SQL pour insérer des données :

```
INSERT INTO test1 (id, nom, age, email)  
VALUES (1, 'roronoa', 30, 'roro.op@re.com');
```

Pour visualiser toutes les données de la table :

```
SELECT * FROM test1;
```

Résultats		Messages		
	id	nom	age	email
1	1	roronoa	30	roro.op@re.com

Script SQL pour supprimer les données d'une table :

```
DELETE FROM test1;
```

Script SQL pour supprimer une table et consulter les tables de la database:

```
DROP TABLE test1 ;
```

Visualisation de toutes les tables de la base de donnée :

```
SELECT name
FROM sys.tables;
```

## ▼ 2. Choix d'un projet

Le projet que je vais réaliser est un système de gestion des ressources humaines d'une entreprise. Il s'agit d'un outil qui va permettre aux départements des ressources humaine d'une entreprise de pouvoir gérer plus facilement les informations des employés, les congés, les salaires, les formations, le recrutement de candidats , les postes vacants et d'autres éléments qui seront abordé au cours du développement du projet.

## ▼ 3. Création des tables et respect des notions de gestion d'une base de données

### ▼ 3.1 Liste de tables

Pour commencer, je vais utiliser les tables suivante :

1. **Employés** : qui va contenir toutes les données relatif aux employés à savoir son id, nom, genre, date d'embauche, poste, salaire, avantages sociaux...
2. **Départements** : qui sera composé de l'id du departement, son nom, l'employé qui sera responsable de ce departement
3. **Postes** : qui va stocker l'id du poste, nom du poste...
4. **Candidats** : qui va contenir l'id du candidat, le nom du candidat, son cv
5. **Promotion**: qui stocke l'id de la promotion et de l'employé, son poste précédent et son nouveau poste
6. **Absences** : constitué de l'Id\_absence, Id\_employé, les dates de debut et fin et les types d'absences
7. **Formations** : qui va contenir les informations sur les formations que suivent les employés, le nom des formations, date de debut, fin, et le cout de la formation

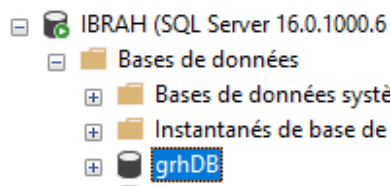
8. **Postes\_vacants** : qui va stocker le nom des poste libre, les compétence nécessaire,
9. **Salaires** : qui regroupe les informations sur le salaire des employés, promotion...

## ▼ 3.2 Création des tables sur SQL server

On dispose de toutes les tables necessaire pour notre projet, nous allons donc procéder à leur création dans la base de donnée.

Tout dabord je cree ma base de donné :

```
CREATE DATABASE grhDB;
```



Lors de la création des tables je vais utiliser certaines notion à savoir :

- **PRIMARY KEY**, designe le fait que chaque enregistrement dans cette colonne est unique et pourra être identifié de manière unique
- **FOREIGN KEY**, peut etre defini comme une **cointrainte** qui permet d'établir une relation entre 2 tables, en gros elle lie la colonne d'une table source à la clés primaires d'une autre table
- **NVARCHAR** pour pouvoir stocker des caratères provenant de n'importe quel langue, y compris les accents, à la différence de **CHAR** qui est limité aux caractères de la langue anglaise.
- **NOT NULL** veut dire que la colonne ne peut pas etre vide lorsqu'on enregistre une nouvelle donnée et **NULL** l'accepte.
- **UNIQUE**, comme son nom l'indique tous les enrégistrement de cette colonne doivent etre complètement différent.
- **DECIMAL(10, 2)** qui veut dire que les enregistrements de cette colonne peuvent aller jusque **9999999999.99**
- **CHECK (Salaire > 0)**, est une condition pour que le salaire qui sera enregistré doit etre supérieur à **0**.
- **REFERENCES** , ici elle est utilisé pour spécifier la colonne et la table a la quel fait référence la clé étrangère

J'ai crée plus de table qu'il n'en faut au cas ou je veux tester d'autre fonctionnalité.

### ▼ 3.2.1 Table Employés

```
--creation de la table Employes
CREATE TABLE Employes (
    ID INT PRIMARY KEY,

    Nom NVARCHAR(50) NOT NULL,
    Prenom NVARCHAR(50) NOT NULL,
    DateNaissance DATE NOT NULL,
    Sexe CHAR(1) NOT NULL,
    Adresse NVARCHAR(100) NOT NULL,
    Telephone NVARCHAR(15),
    Email NVARCHAR(100) UNIQUE,
    DateEmbauche DATE NOT NULL,
    ID_Departement INT,
    ID_Poste INT,
    Salaire DECIMAL(10, 2) CHECK (Salaire > 0),
```

```

    AvantageSante NVARCHAR(50),
    AvantageRetraite NVARCHAR(50),

);

-- Affichage de la table Employes
SELECT * FROM Employes;

```

Résultats	Messages
ID	Nom
Prenom	DateNaissance
Sexe	Adresse
Telephone	Email
DateEmbauche	ID_Departement
ID_Poste	Salare
AvantageSante	AvantageRetraite

Après avoir crée la table Departements je reviens ajoute des clés étrangères à cette table :

```

ALTER TABLE Employes
ADD CONSTRAINT FK_Departement_Employes
FOREIGN KEY (ID_Departement) REFERENCES Departements(ID);

ALTER TABLE Employes
ADD CONSTRAINT FK_Poste_Employes
FOREIGN KEY (ID_Poste) REFERENCES Postes(ID);

```

### ▼ 3.2.2 Table Départements

```

--creation de la table Departements
CREATE TABLE Departements (
    ID INT PRIMARY KEY,
    Nom NVARCHAR(50) NOT NULL,
    Responsable NVARCHAR(100)
);

-- Affichage de la table Departements
SELECT * FROM Departements;

```

Résultats	Messages
ID	Nom
Responsable	

### ▼ 3.2.3 Table Postes

```

--creation de la table Postes
CREATE TABLE Postes (
    ID INT PRIMARY KEY,
    Nom NVARCHAR(50) NOT NULL,
    Description NVARCHAR(255),
    Exigences NVARCHAR(255)
);

-- Affichage de la table Postes
SELECT * FROM Postes;

```

	ID	Nom	Description
		name	
1		Employes	
2		Departements	
3		Postes	

### ▼ 3.2.4 Table Candidats

```
--creation de la table Candidats
CREATE TABLE Candidats (
  ID INT PRIMARY KEY,
  Nom NVARCHAR(50) NOT NULL,
  Prenom NVARCHAR(50) NOT NULL,
  Telephone NVARCHAR(15),
  Email NVARCHAR(100) UNIQUE,
  CV NVARCHAR(MAX),
  Statut NVARCHAR(50) NOT NULL
);

-- Affichage de la table Candidats
SELECT * FROM Candidats;
```

	ID	Nom	Prenom	Telephone	Email	CV	Statut
		name					
1		Employes					
2		Departements					
3		Postes					
4		Candidats					

### ▼ 3.2.5 Table Promotion

```
CREATE TABLE Promotion (
  ID INT PRIMARY KEY,
  ID_Employe INT NOT NULL,
  DatePromotion DATE NOT NULL,
  AncienPoste INT,
  NouveauPoste INT,
  FOREIGN KEY (ID_Employe) REFERENCES Employes (ID),
  FOREIGN KEY (AncienPoste) REFERENCES Postes (ID),
  FOREIGN KEY (NouveauPoste) REFERENCES Postes (ID)
);

-- Affichage de la table Promotion
SELECT * FROM Promotion ;
```

	ID	ID_Employe	DatePromotion	AncienPoste	NouveauPoste
	name				
1	Employes				
2	Departements				
3	Postes				
4	Candidats				
5	Promotion				

### ▼ 3.2.6 Table Absences

```
--creation de la table Absences
CREATE TABLE Absences (
  ID INT PRIMARY KEY,
  ID_Employe INT NOT NULL,
  DateDebut DATE NOT NULL,
  DateFin DATE NOT NULL,
  TypeAbsence NVARCHAR(50) NOT NULL,
  Motif NVARCHAR(255),
  FOREIGN KEY (ID_Employe) REFERENCES Employes (ID)
);

-- Affichage de la table Absences
SELECT * FROM Absences ;
```

	ID	ID_Employe	DateDebut	DateFin	TypeAbsence	Motif
	name					
1	Employes					
2	Departements					
3	Postes					
4	Candidats					
5	Promotion					
6	Absences					

### ▼ 3.2.7 Table Formations

```
--creation de la table Formations
CREATE TABLE Formations (
  ID INT PRIMARY KEY,
  ID_Employe INT NOT NULL,
  Nom NVARCHAR(100) NOT NULL,
  DateDebut DATE NOT NULL,
  DateFin DATE NOT NULL,
  DureeHeures INT CHECK (DureeHeures > 0),
  FOREIGN KEY (ID_Employe) REFERENCES Employes (ID)
);

-- Affichage de la table Formations
SELECT * FROM Formations ;
```



	ID	ID_Employe	Nom	DateDebut	DateFin	DureeHeures
	name					
1	Employes					
2	Departements					
3	Postes					
4	Candidats					
5	Promotion					
6	Absences					
7	Formations					

### ▼ 3.2.8 Table Postes\_vacants

```
--creation de la table Postes_vacants
CREATE TABLE Postes_vacants (
    ID INT PRIMARY KEY,
    ID_Poste INT NOT NULL,
    DatePublication DATE NOT NULL,
    DateExpiration DATE NOT NULL,
    FOREIGN KEY (ID_Poste) REFERENCES Postes (ID)
);

-- Affichage de la table Postes_vacants
SELECT * FROM Postes_vacants ;
```

	ID	ID_Poste	DatePublication	DateExpiration
	name			
1	Employes			
2	Departements			
3	Postes			
4	Candidats			
5	Promotion			
6	Absences			
7	Formations			
8	Postes_vacants			

### ▼ 3.2.9 Table Salaires

```
--creation de la table Salaires
CREATE TABLE Salaires (
    ID INT PRIMARY KEY,
    ID_Employe INT NOT NULL,
    SalaireBase DECIMAL(10, 2) CHECK (SalaireBase > 0),
    Prime DECIMAL(10, 2) CHECK (Prime >= 0),
    DateVersement DATE NOT NULL,
    FOREIGN KEY (ID_Employe) REFERENCES Employes (ID)
);

-- Affichage de la table Salaires
SELECT * FROM Salaires ;
```



Si l'on desire en supprimer un TDU on peut utiliser :

```
DROP TYPE Genre
```

A présent on peut modifier la Table Employes en adaptant les nouveau type.

Pour que ce soit plus simple je vais supprimer colonne de la table associé au type que je veux modifier, puis rajouter cette table avec le nouveau type.

suppression de la colonne "Telephone"

```
-- suppression de la colonne "Telephone"
ALTER TABLE Employes DROP COLUMN Telephone;
```

Ajout de la colonne "Telephone"

```
-- ajout de la nouvelle colonne "Telephone"
ALTER TABLE Employes ADD Telephone NumeroTelephone NOT NULL;
```

A présent on peut consulter la nouvelle table Employe en mentionnant les types :

```
-- j'affiche les noms et types de data de chaque colonne dans la table "Employes"
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'Employes';
```

	COLUMN_NAME	DATA_TYPE
1	ID	int
2	Nom	nvarchar
3	Prenom	nvarchar
4	DateNaissance	date
5	Adresse	nvarchar
6	Email	nvarchar
7	DateEmbauche	date
8	ID_Departement	int
9	ID_Poste	int
10	Salaire	decimal
11	AvantageSante	nvarchar
12	AvantageRetraite	nvarchar
13	Sexe	nvarchar
14	NumeroSecuSociale	nvarchar
15	Telephone	nvarchar

La nouvelle table Employe est la suivante

```
CREATE TABLE Employes (
  ID INT PRIMARY KEY,
  Nom NVARCHAR(50) NOT NULL,
  Prenom NVARCHAR(50) NOT NULL,
  DateNaissance DATE NOT NULL,
  Adresse NVARCHAR(100) NOT NULL,
  Email NVARCHAR(100) UNIQUE,
  DateEmbauche DATE NOT NULL,
  ID_Departement INT,
```

```

ID_Poste INT,
Salaire DECIMAL(10, 2) CHECK (Salaire > 0),
AvantageSante NVARCHAR(50),
AvantageRetraite NVARCHAR(50),

Sexe Genre NOT NULL, --TUD
NumeroSecuSociale NumeroSecuSociale, --TUD
Telephone NumeroTelephone --TUD

FOREIGN KEY (ID_Departement) REFERENCES Departements (ID),
FOREIGN KEY (ID_Poste) REFERENCES Postes (ID)

);

```

## ▼ 5. Définir des contraintes

Des contraintes ont été prédéfinies lors de la création de la table **Employes** et d'autres tables

Contrainte utilisée :

```

-- Table Employes
Salaire DECIMAL(10, 2) CHECK (Salaire > 0)

```

```

-- Table Formations
DureeHeures INT CHECK (DureeHeures > 0),

```

```

--Table Salaires

SalaireBase DECIMAL(10, 2) CHECK (SalaireBase > 0),
Prime DECIMAL(10, 2) CHECK (Prime >= 0),

```

### ▼ 5.1. Les Vues

Avant de commencer les procédures, on va d'abord aborder la notion de **VU**, une vue permet de lier plusieurs tables entre elles. On va donc pouvoir créer une nouvelle table qui regroupe des informations de plusieurs tables à la fois.

Les notions de clés étrangères utilisées précédemment auront leur importance ici, si elles ont mal été définies on va se retrouver avec des erreurs de **jointure**.

Le script suivant permet de créer la vue **VueEmployes** qui sera une combinaison des tables **Employes**, **Departements**, **postes**.

```

-- je crée la vue VueEmployes qui va regrouper des infos sur les Tables
-- Employes , Departements , Postes

CREATE VIEW VueEmployes
AS
SELECT e.ID, e.Nom, e.Prenom, e.DateNaissance, e.Sexe, e.Adresse,
       e.Email, e.Telephone, e.NumeroSecuSociale, e.DateEmbauche,

       d.Nom AS Departement,
       p.Nom AS Poste,

       e.Salaire, e.AvantageSante, e.AvantageRetraite

```

```

FROM Employes e
JOIN Departements d ON e.ID_Departement = d.ID
JOIN Postes p ON e.ID_Poste = p.ID

--affichage de la table crée
select * from VueEmployes

```

ID	Nom	Prenom	DateNaissance	Sexe	Adresse	Email	Telephone	NumeroSecuSociale	DateEmbauche	Departement	Poste	Salaire	AvantageSante	AvantageRetraite
1	luffy	D	2000-01-02	M	20 rue origashima	d.luffy@wano.com	77778888	123456345	2020-01-10	RH	Chef de projet	2000.00	santé	retraite
2	zororo	zoro	2001-01-05	M	20 rue impeldown	rzoro@wano.com	7712588	234563456	2020-01-10	compta	dev	3000.00	santé	retraite

## ▼ 6. Procédures stockées (traitement sur les données) / déclencheurs (Triggers)

### ▼ 6.1. Procédures stockées

Les **Procédures stockées** sont utilisé pour effectuer des traitement sur les données des tables, c'est une requete assez particulière car elle est précompilé et optimisée pour accélérer son exécution.

Le script suivant est une procédure pour ajouter un nouvelle employé

```

-- Procédures stockées (traitement sur les données)

CREATE PROCEDURE AjoutEmploye

    @ID int,
    @Nom NVARCHAR(50),
    @Prenom NVARCHAR(50),
    @DateNaissance DATE,
    @Adresse NVARCHAR(100),
    @Email NVARCHAR(100),
    @DateEmbauche DATE,
    @ID_Departement INT,
    @ID_Poste INT,
    @Salaire DECIMAL(10, 2),
    @AvantageSante NVARCHAR(50),
    @AvantageRetraite NVARCHAR(50),
    @Sexe Genre,
    @NumeroSecuSociale NumeroSecuSociale,
    @Telephone NumeroTelephone

AS
BEGIN
    INSERT INTO Employes (ID,Nom, Prenom, DateNaissance, Adresse, Email, DateEmbauche, ID_Departement, ID_Poste, Salaire, Avant
    VALUES (@ID,@Nom, @Prenom, @DateNaissance, @Adresse, @Email, @DateEmbauche, @ID_Departement, @ID_Poste, @Salaire, @Avantage
END;

```

Pour verifier si la procédure à bien été crée je vais utiliser le script suivant :

```

-- Affichage des procédures

SELECT * FROM INFORMATION_SCHEMA.ROUTINES
WHERE ROUTINE_TYPE = 'PROCEDURE'
AND ROUTINE_SCHEMA = 'dbo';

```

Résultat du script :

	SPECIFIC_CATALOG	SPECIFIC_SCHEMA	SPECIFIC_NAME	ROUTINE_CATALOG	ROUTINE_SCHEMA
1	grhDB	dbo	AjoutEmploye	grhDB	dbo

On sait que la procédure a bien été créée à présent on va l'exécuter pour ajouter une nouvelle employé :

```
-- Exécution de la procédure

EXEC ajoutEmploye

@ID = 3 ,
@Nom = 'tobi',
@Prenom = 'akats',
@DateNaissance = '1999-02-15',
@Adresse = '21 rue konoha',
@Email = 'tobi@el.com',
@DateEmbauche = '2002-01-01',
@ID_Departement = 1,
@ID_Poste = 2,
@Salaire = 4050,
@AvantageSante = 'Santé Plus',
@AvantageRetraite = 'Retraite Plus',
@Sexe = 'M',
@NumeroSecuSociale = '123',
@Telephone = '0486529';
```

On va à présent consulter la **vueEmployes**, pour avoir des informations sur lui, son département et son poste.

```
--affichage vu
select * from VueEmployes
```

Output après l'exécution de la **vueEmployes**, on voit bien que la ligne 3 a bien été ajoutée :

ID	Nom	Prenom	DateNaissance	Sexe	Adresse	Email	Telephone	NumeroSecuSociale	DateEmbauche	Departement	Poste	Salaire	AvantageSante	AvantageRetraite
1	kufy	D	2000-01-02	M	20 rue origachina	d.kufy@wano.com	77778888	123456345	2020-01-10	RH	Chef de projet	2000.00	santé	retraite
2	zorona	zoro	2001-01-05	M	20 rue impeldown	zoro@wano.com	7712588	234563456	2020-01-10	compta	dev	3000.00	santé	retraite
3	tobi	akats	1999-02-15	M	21 rue konoha	tobi@el.com	0486529	123	2002-01-01	RH	dev	4050.00	Santé Plus	Retraite Plus

A titre informatif si l'on désire supprimer cette ligne ou une ligne quelconque on peut utiliser la commande (en adaptant l'ID):

```
-- Supprimer une ligne de la table
DELETE FROM Employes
WHERE ID = 3;
```

## ▼ 6.2. Triggers

Lien utiles pour cette partie :

### SQL Server CREATE TRIGGER

In this tutorial, you will learn how to use the SQL Server CREATE TRIGGER statement to create a new trigger in the database.

<https://www.sqlservertutorial.net/sql-server-triggers/sql-server-create-trigger/>

```
* product_id
product_name
brand_id
category_id
model_year
```

## Triggers in SQL Server

In this article, we will review triggers in SQL Server, different types of trigger events, trigger order and NOT FOR REPLICATION in triggers.

<https://www.sqlshack.com/triggers-in-sql-server/>

```
ALTER DATABASE [Adventureworks] SET RECURSIVE_TRIGGERS ON WITH NO_WAIT
GO

--use Adventureworks
update Locations set LocationName = 'Richmond Cross' where LocationID =1
```

Dans ce point nous allons aborder la notion de **trigger**, ce sont des déclencheurs qui s'exécute après un event du genre traitement de données (Insert, Update, Delete) dans une table.

Il en existe 2 à savoir : **After, INSTEAD**

Pour commencer je vais creer une nouvelle table qui va contenir la date de la derniere modification ou la date de création d'un nouvelle employé dans la table d'employé

```
-- Ajout d'une new table
ALTER TABLE Employes
ADD Derniere_Modif DATETIME NULL;
```

A présent on va creer un trigger qui se lance lorsque je cree un nouvelle employé.

**INNER JOIN inserted i ON e.ID = i.ID;** sert à faire une liaison entre la la table virtuel **inserted (qui enregistre les nouvelles lignes ajoutées)** et la table employé, elle va servir notement à mettre metre à jour la dernière ligne ajouté. Donc si j'ajoute un nouvelle employé, la colonne **Derniere\_Modif** va recevoir la date avec **GETDATE()**

```
CREATE TRIGGER trg_employe_insert
ON Employes
AFTER INSERT
AS
BEGIN
    UPDATE Employes
    SET Derniere_Modif = GETDATE()
    FROM Employes e
    INNER JOIN inserted i ON e.ID = i.ID;
END;
```

Le script suivant va servir à verifer si le trigger à bien été crée :

```
SELECT
    name,
    is_instead_of_trigger
FROM
    sys.triggers
WHERE
    type = 'TR';
```

	Résultats	Messages
	name	is_instead_of_trigger
1	trg_employe_insert	0

On affiche la table employé pour voir si le trigger se declanche lorsqu'on ajoute un nouvelle employé.

ID	Nom	DateEmbauche	Adresse	Email	DateEntree	ID_Departement	ID_Poste	Salaire	AverageCarte	AveragePensee	Sexe	NumeroCarteCivile	Telephone	Devise_Mail
1	luffy	2000-01-01	20 rue impulsion	luffy@marco.com	2020-01-10	1	1	2000.00	anime	marime	M	123456789	77778888	NALL
2	zorona	2001-01-05	20 rue impulsion	zorona@marco.com	2020-01-10	2	2	3000.00	anime	marime	M	23456789	77123456	NALL
3	tobi	1999-02-15	27 rue lamotte	tobi@marco.com	2020-01-01	2	2	4050.00	Serveur Pils	Rayon Pils	M	123	6468529	NALL
4	madara	1990-03-12	50 rue lamotte	madara@marco.com	2020-01-01	2	1	10000.00	RAS	RAS	M	130123	99629	2020-03-27 04:22:45.475

## ▼ 7. Fonctions utilisateur

Les fonctions sont utilisé pour compléter les vues et les procédures stockées abordé plus haut. Elles prennent des paramètres et retournent un certain type de valeur qu'il faudra prédéfinir dans ca déclaration.

On va utiliser cette fonction pour calculer le salaire annuel d'un employé

```
-- fonctions utilisateur :

-- Cacul du salaire annuel

CREATE FUNCTION CalculerSalaireAnnuel (@SalaireMensuel DECIMAL(10, 2))
RETURNS DECIMAL(10, 2)
AS
BEGIN
    RETURN @SalaireMensuel * 12;
END;

-- j'appelle la fonction :

SELECT ID, Nom, Salaire, dbo.CalculerSalaireAnnuel(Salaire) AS SalaireAnnuel
FROM Employes;
```

Le résultat est :

	ID	Nom	Salaire	SalaireAnnuel
1	1	luffy	2000.00	24000.00
2	2	zorona	3000.00	36000.00
3	3	tobi	4050.00	48600.00
4	4	madara	10000.00	120000.00

## ▼ 8. Sauvegarde (manuel / automatique) / restauration

Liens utiles :

### Créer une sauvegarde complète de base de données - SQL Server

Cet article explique comment créer une sauvegarde complète de base de données dans SQL Server à l'aide de SQL Server Management Studio, de Transact-SQL ou de PowerShell.

<https://learn.microsoft.com/fr-fr/sql/relational-databases/backups-restore/create-a-full-database-backup-sql-server?view=sql-server-ver16>

Le dossier de sauvegarde par défaut est :

C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Backup

### ▼ 8.1. Sauvegarde manuel



Liens utiles :

#### Sauvegarde de la base de données MS SQL Server

Vous pouvez faire une copie de votre base de données en l'enregistrant dans un fichier de sauvegarde. Si vous le souhaitez, vous pouvez ensuite déplacer la sauvegarde sur un autre ordinateur et la restaurer dans une autre instance License Metric Tool .

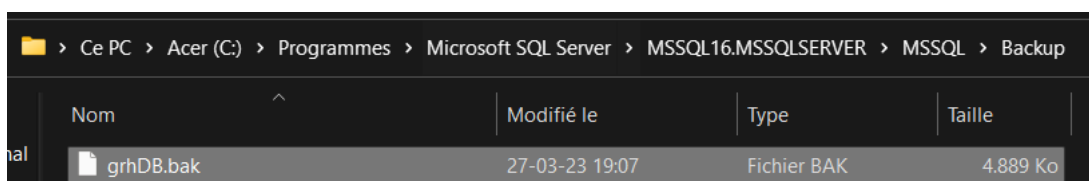
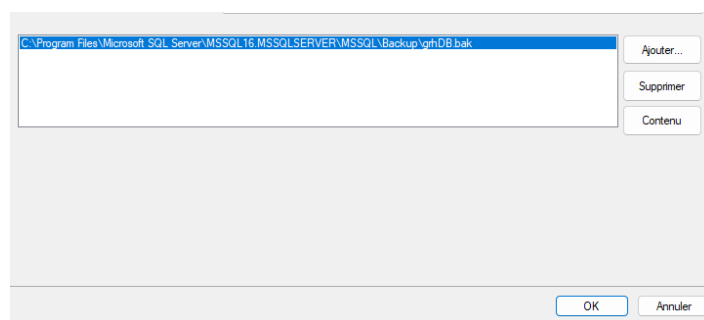
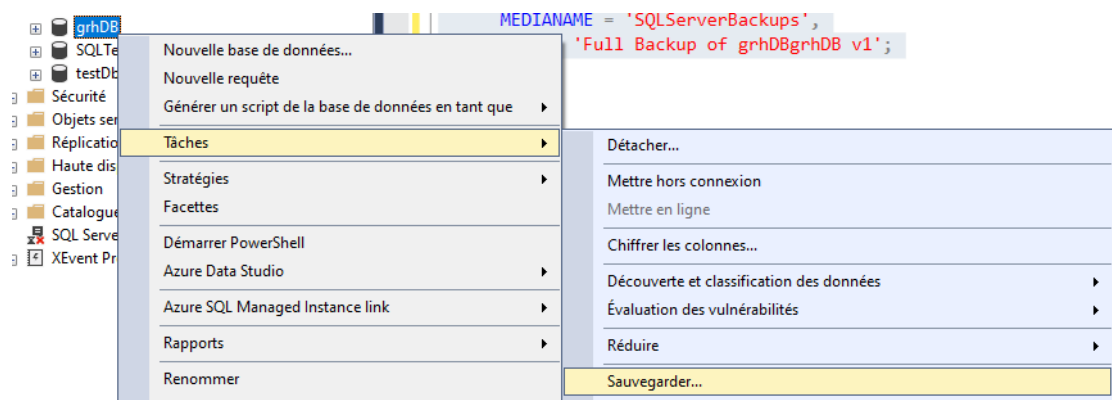
🔗 <https://www.ibm.com/docs/fr/license-metric-tool?topic=database-backing-up-ms-sql-server>

Ce type de sauvegarde peut être fait de 2 façons :

- Au moyen d'un script, il sera sauvegardé sous le nom **grhDB.bak**

```
BACKUP DATABASE grhDB
TO DISK = 'C:\Users\ibrah\Desktop\MA1\Q2\DBA\grhDB.bak'
WITH FORMAT,
    MEDIANAME = 'SQLServerBackups',
    NAME = 'Full Backup of grhDBgrhDB v1';
GO
```

- Sauvegarde grâce à l'interface SQL server



## ▼ 8.2. Sauvegarde automatique

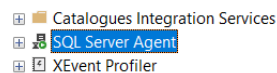
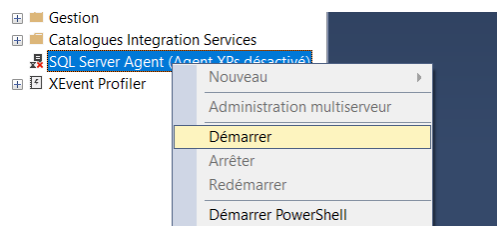
Liens utiles :

### SQL Server Sauvegarde Automatique - Astuces informatiques

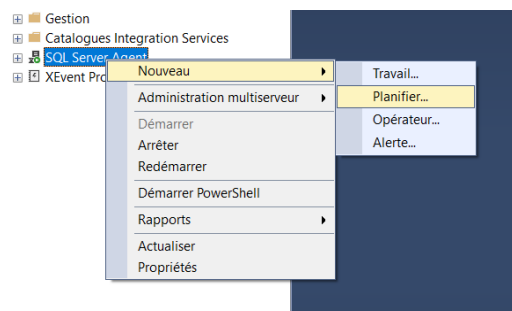
Avez-vous déjà tenté de configurer une sauvegarde automatisée de votre serveur de base de données SQL Server Express, ? Pour vous rendre compte que les options de planification du job ne sont pas disponibles avec l'édition express? Après une brève recherche sur Google, vous avez peut-être remarqué que d'autres ont réussi à mettre en œuvre

<https://blog.itgs-solutions.ch/sql-server-sauvegarde-automatique/>

Pour activer l'agent il faut tout d'abord exécuter sql server en mode administrateur



Nouveau > planifier



C'est sur cette page qu'on définit comment on veut que la sauvegarde se passe

**Nouvelle planification du travail**

Nom :  Travaux planifiés

Type de planification : Périodique ☒ Actif

Une seule occurrence

Date : 25-05-23 Heure : 22:01:08

Fréquence

Périodicité : Hebdomadaire

Répéter toutes les : 1 semaine(s) le

☐ Lundi ☐ Mercredi ☐ Vendredi ☐ Samedi ☒ Dimanche

☐ Mardi ☐ Jeudi

Fréquence quotidienne

☒ Une fois à : 00:00:00

☐ Toutes les : 1 heure(s)

Début à : 00:00:00 Fin : 23:59:59

Durée

Date de début : 25-05-23

☐ Date de fin : 25-05-23

☒ Aucune date de fin :

Résumé

Description : A lieu toutes les semaines le Dimanche à 00:00:00. La planification sera utilisée à partir du 25-05-23.

OK Annuler Aide

### ▼ 8.3. Restauration

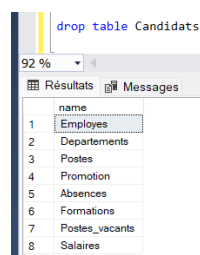
Liens utiles :

#### Sauvegarde et restauration de la base de données

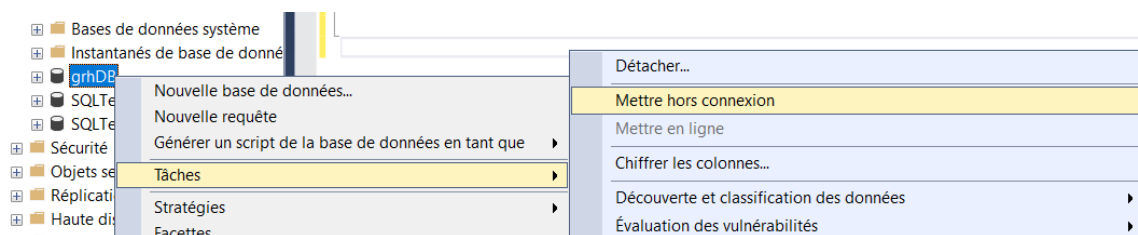
Effectuez des sauvegardes régulières des données stockées dans la base de données, puis restaurez-les au besoin pour éviter la perte de données.

🔗 <https://www.ibm.com/docs/fr/license-metric-tool?topic=database-backing-up-restoring>

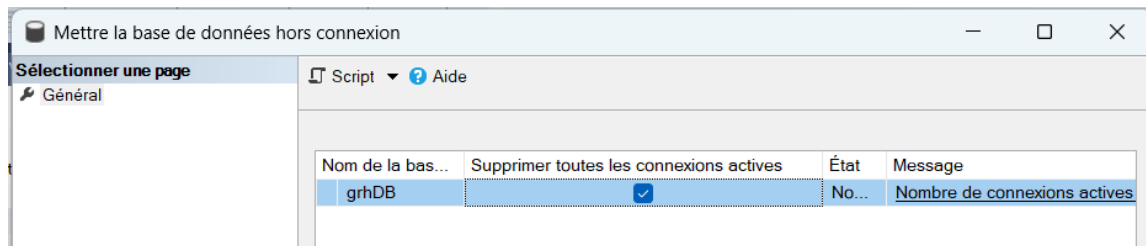
Pour tester la restauration nous allons par exemple supprimer la table candidat



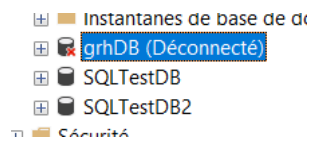
La première étape est de se deconnecter à la base de donnée



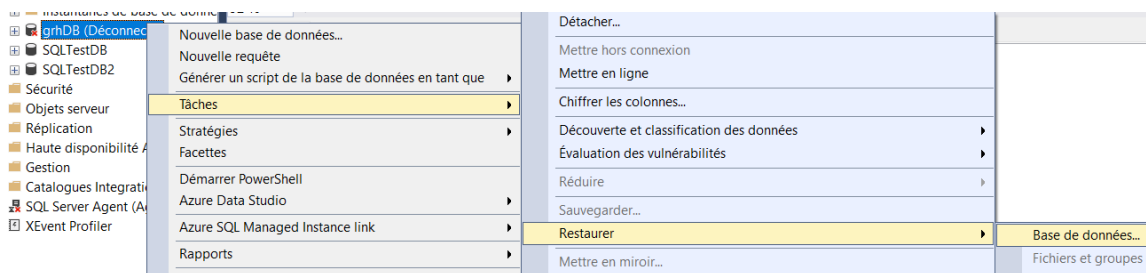
Puis cocher > supprimer toutes les connexions actives > faire "ok"



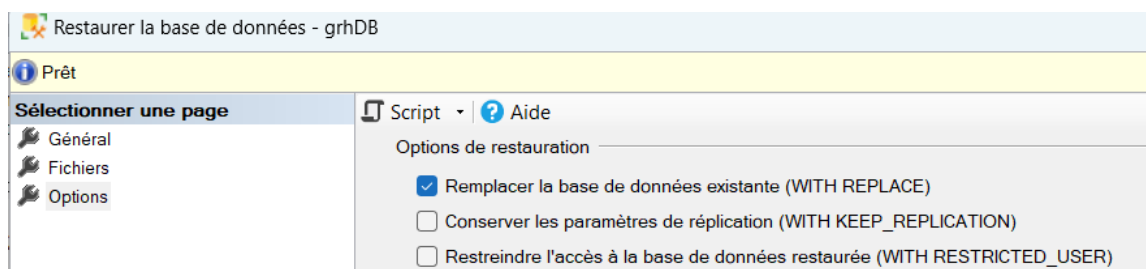
La base de donnée devient :



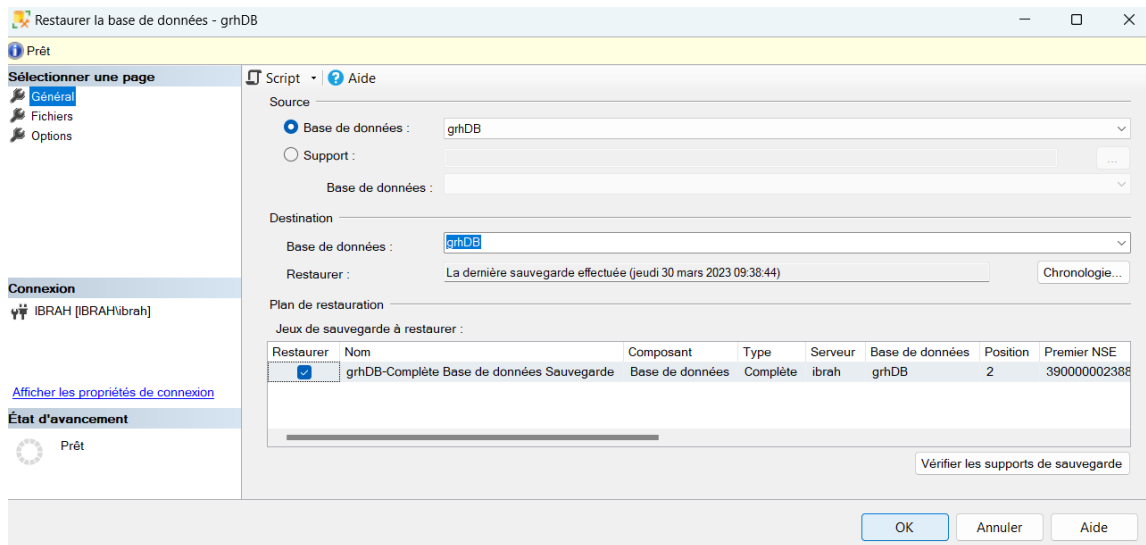
Faire :



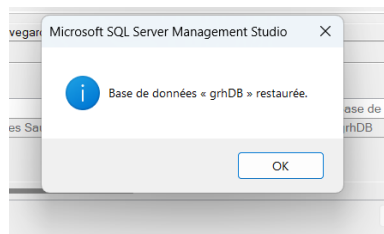
Aller dans option et cocher > " WITH REPLACE "



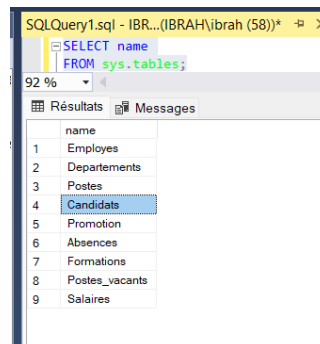
Puis faire "ok" pour valider la restauration :



Si la restauration est correctement faite :



La table à bien été resauré



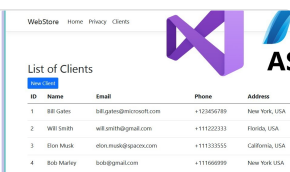
## ▼ 9. Gestion des données à partir d'une page web (ASP)

Liens utiles :

Fill HTML Table From SQL Server Database Using ASP.NET Core and Razor Pages | Visual Studio 2022

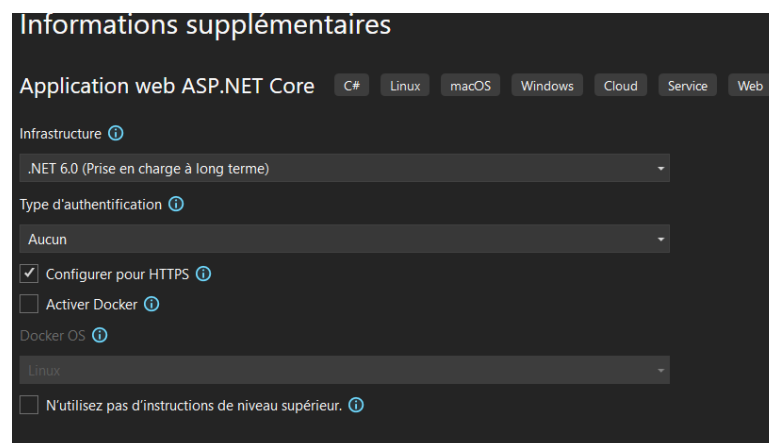
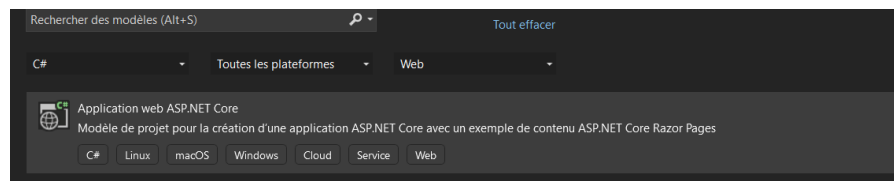
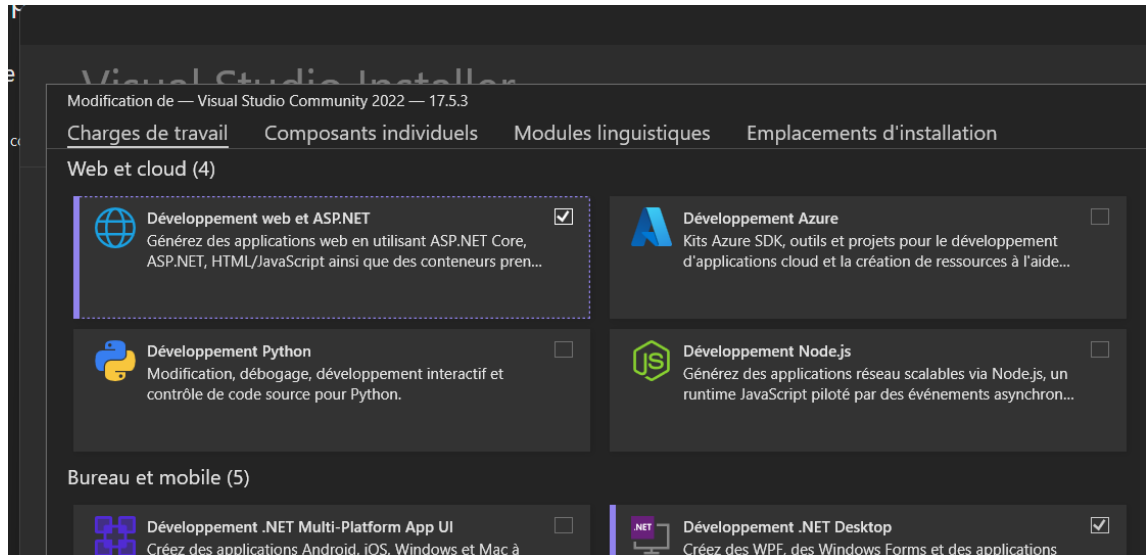
Full course on Udemy : <https://www.udemy.com/course/aspnet-core-web-application-using-razor-pages/?couponCode=28AD7FDE47AA636B1148>

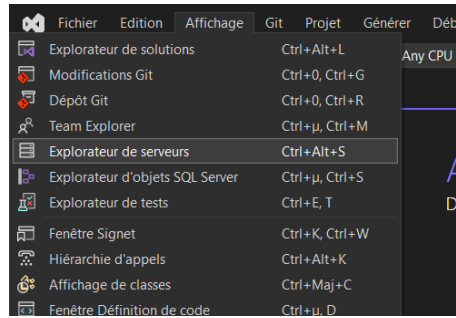
[https://www.youtube.com/watch?v=YUPg41kG\\_kw&t=353s](https://www.youtube.com/watch?v=YUPg41kG_kw&t=353s)



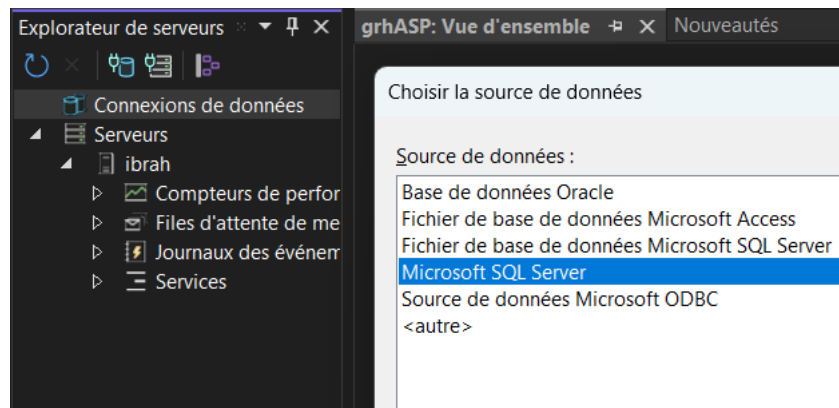
## ▼ 9.1. Création du projet et de la base de données

Pour commencer j'installe l'extention **Développement web et ASP.NET**

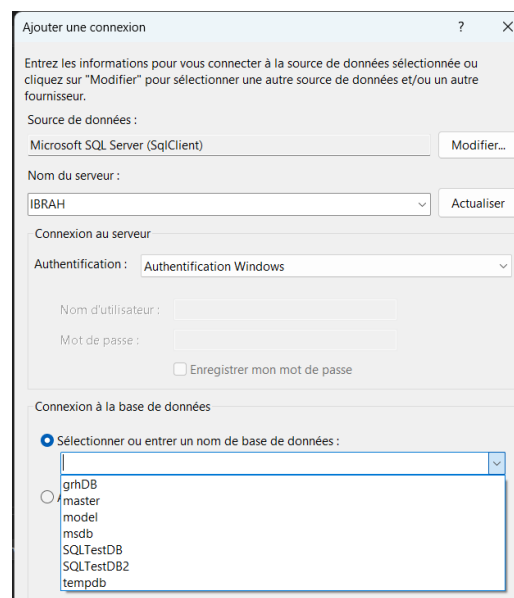




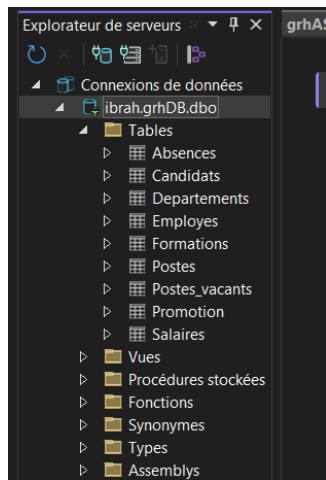
On clique sur l'icone database-prise pour selectionner la base de donnée



En suite il faut entrer le nom du server et selectionner la base de donnée à lier



La connexion à bien été établit



En consultant la table Employes on s'aperçoit qu'il y'a un script SQL qui est généré automatiquement, et on à une interface graphique avec la liste des paramètres.

Tout ca pour dire que c'est possible de mettre à jour la base de donnée assez rapidement par interface graphique ou par script SQL.

**dbo.Employes [Design] - Nouveautés**

Mettre à jour | Fichier de script: dbo.Employes.sql

Nom	Type de données	Autoriser les valeurs Null	Par
ID	int	<input type="checkbox"/>	
Nom	nvarchar(50)	<input type="checkbox"/>	
Prenom	nvarchar(50)	<input type="checkbox"/>	
DateNaissance	date	<input type="checkbox"/>	
Adresse	nvarchar(100)	<input type="checkbox"/>	
Email	nvarchar(100)	<input checked="" type="checkbox"/>	
DateEmbauche	date	<input type="checkbox"/>	
ID_Departement	int	<input checked="" type="checkbox"/>	
ID_Poste	int	<input checked="" type="checkbox"/>	
Salaire	decimal(10,2)	<input checked="" type="checkbox"/>	
AvantageSante	nvarchar(50)	<input checked="" type="checkbox"/>	
AvantageRetraite	nvarchar(50)	<input checked="" type="checkbox"/>	
Sexe	dbo.Genre	<input type="checkbox"/>	
NumeroSecuSociale	dbo.NumeroSecuSociale	<input type="checkbox"/>	
Telephone	dbo.NumeroTelephone	<input type="checkbox"/>	
Derniere_Modif	datetime	<input checked="" type="checkbox"/>	
		<input type="checkbox"/>	

**clés (2)**  
 <sans nom> (Clé primaire, Clustered: ID)  
 <sans nom> (Email)

**vérifier les contraintes (1)**  
 <sans nom> (Salaire)

**index (0)**

**clés étrangères (2)**  
 FK\_Departement\_Employes (ID)  
 FK\_Poste\_Employes (ID)

**déclencheurs (1)**  
 trg\_employe\_insert (Insert)

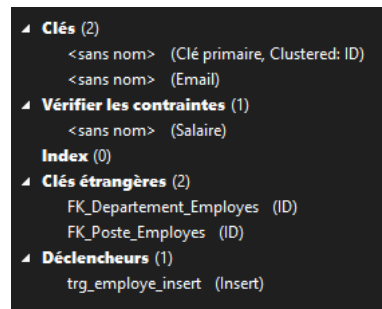
**Conception | T-SQL**

```

CREATE TABLE [dbo] [Employes] (
  [ID] INT NOT NULL,
  [Nom] NVARCHAR (50) NOT NULL,
  [Prenom] NVARCHAR (50) NOT NULL,
  [DateNaissance] DATE NOT NULL,
  [Adresse] NVARCHAR (100) NOT NULL,
  [Email] NVARCHAR (100) NULL,
  [DateEmbauche] DATE NOT NULL,
  [ID_Departement] INT NULL,
  [ID_Poste] INT NULL,
  [Salaire] DECIMAL (10, 2) NULL,
  [AvantageSante] NVARCHAR (50) NULL,
  [AvantageRetraite] NVARCHAR (50) NULL,
  [Sexe] [dbo] [Genre] NOT NULL,
  [NumeroSecuSociale] [dbo] [NumeroSecuSociale] NOT NULL,
  [Telephone] [dbo] [NumeroTelephone] NOT NULL,
  [Derniere_Modif] DATETIME NULL,
  PRIMARY KEY CLUSTERED ([ID] ASC),
  UNIQUE NONCLUSTERED ([Email] ASC),
  CONSTRAINT [FK_Departement_Employes] FOREIGN KEY ([ID_Departement]) REFERENCES [dbo] [Departements] ([ID]),
  CONSTRAINT [FK_Poste_Employes] FOREIGN KEY ([ID_Poste]) REFERENCES [dbo] [Postes] ([ID]),
  CHECK ([Salaire]>=0))
  
```



On à également une vue sur les Clés de la table, contraintes, déclencheur.

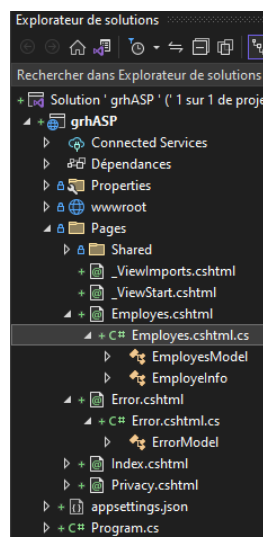


Il est également possible de consulté directement les data d'une table

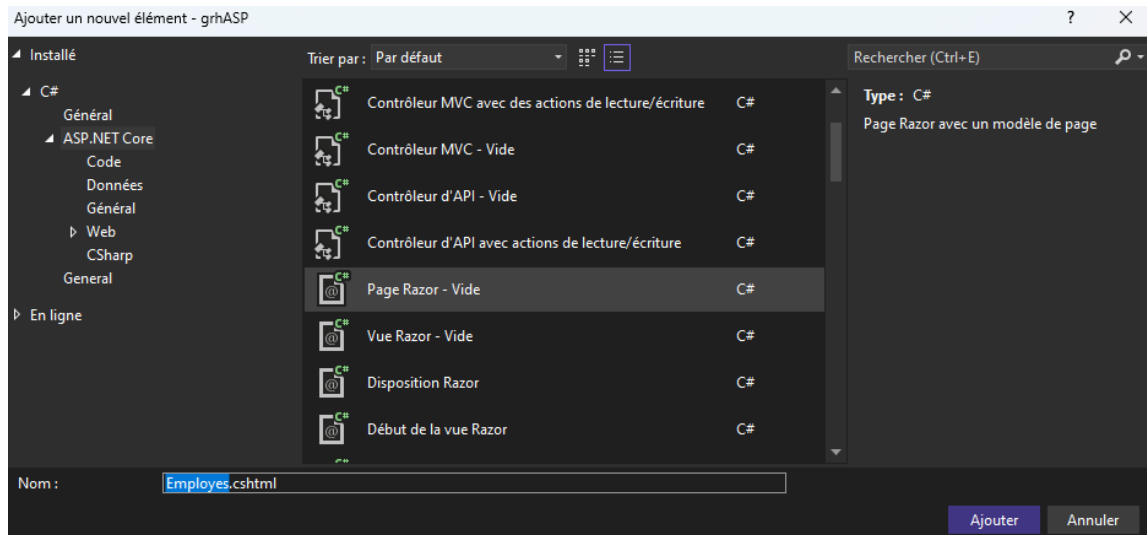
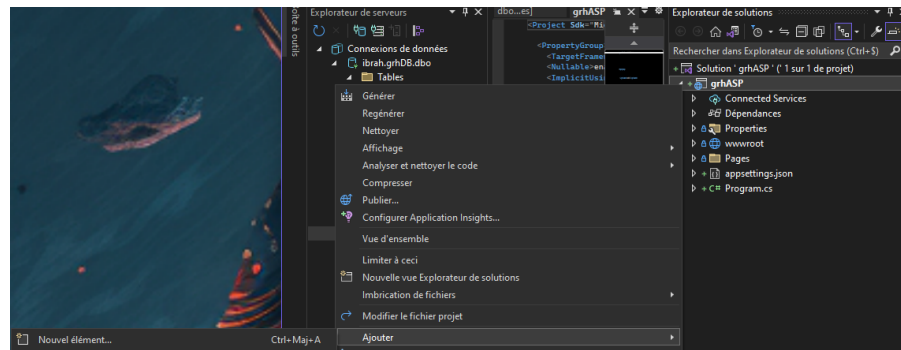
ID	Nom	Prenom	DateNaissance	Adresse	Email	DateEmbauche	ID_Departement	ID_Poste
1	luffy	D	02-01-00	20 rue onigashi...	d.luffy@wano.c...	10-01-20	1	1
2	roronoa	zoro	05-01-01	20 rue impeldo...	rzoro@wano.co...	10-01-20	2	2
3	tobi	akats	15-02-99	21 rue konoha	tobi@el.com	01-01-02	1	2
4	madara	uchiwa	12-03-90	50 rue konoha	madara@el.com	01-01-00	2	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

## ▼ 9.2. Création des pages ASP

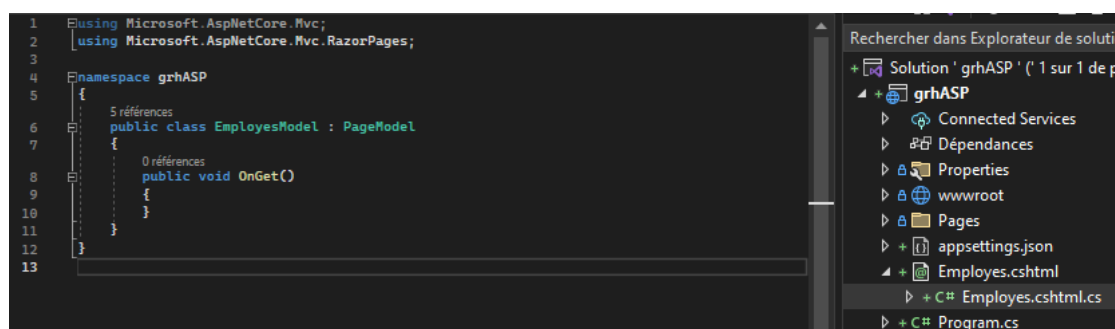
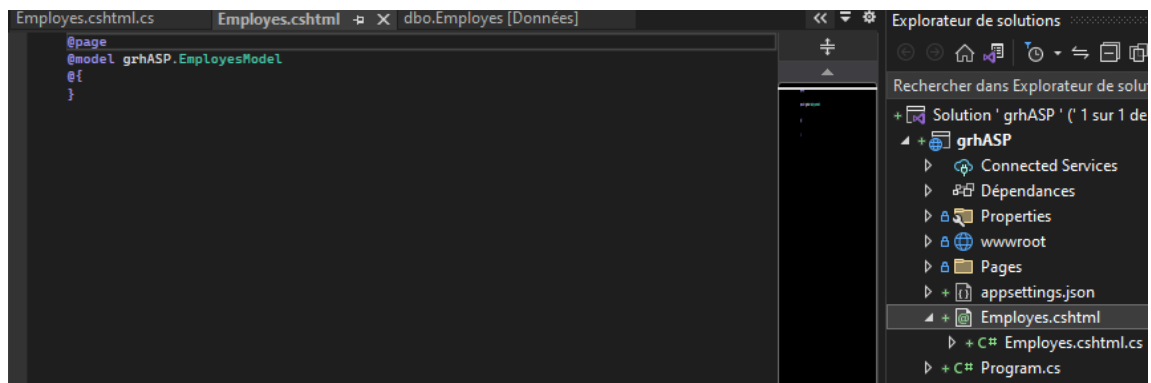
Architecture du projet :



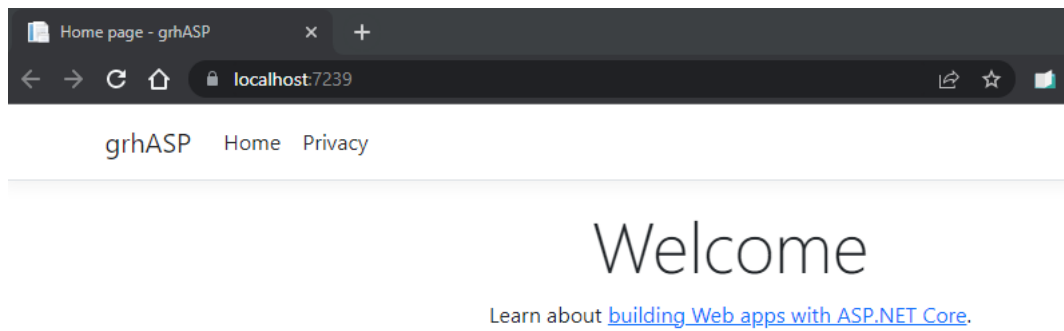
Je crée un nouveau fichier .cshtml



Les pages générées sont les suivantes :

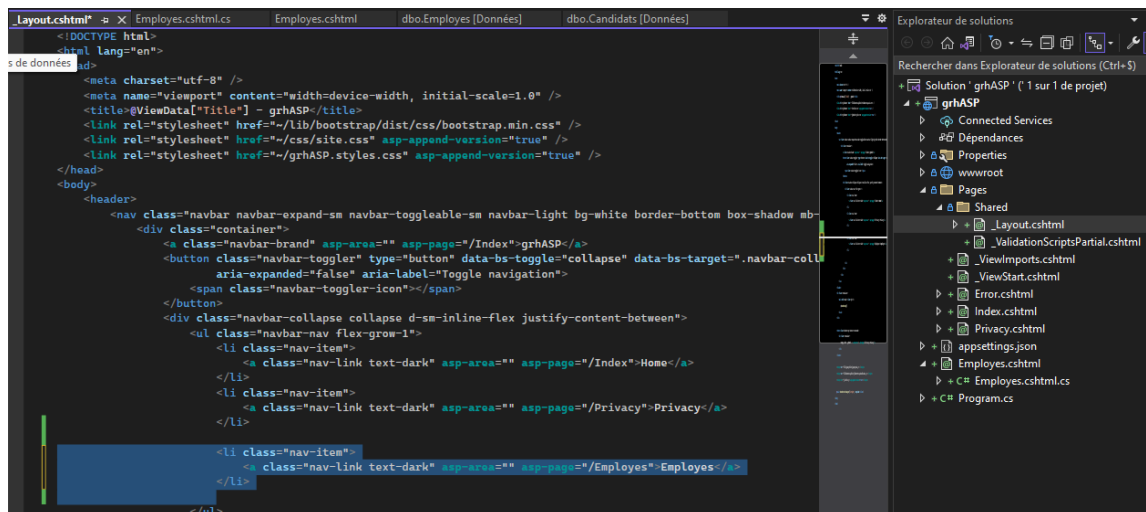


Visualisation de la page :

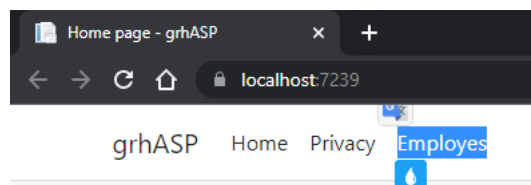


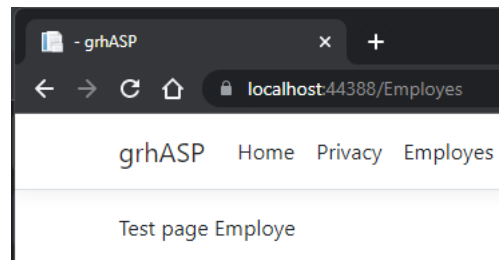
Pour commencer on va ajouter Employé à la nav bar :

```
<li class="nav-item">
  <a class="nav-link text-dark" asp-area="" asp-page="/Employes">Employes</a>
</li>
```

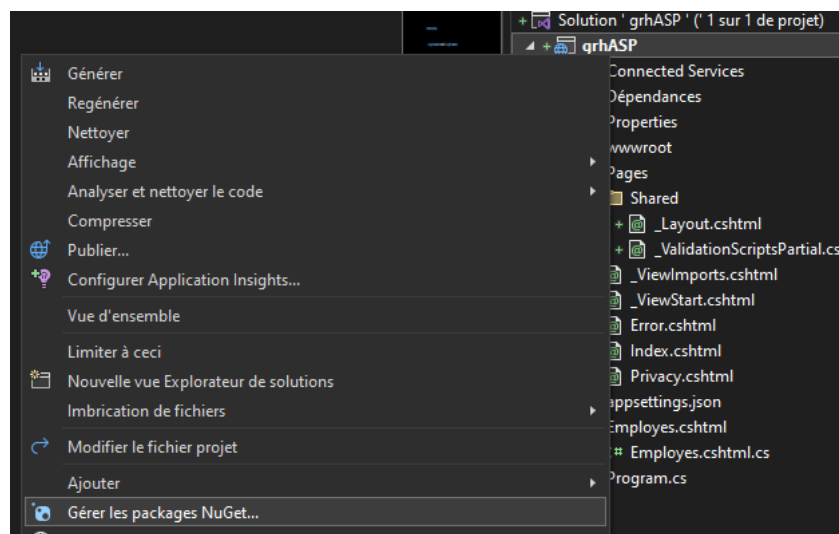


Visualisation de la page :

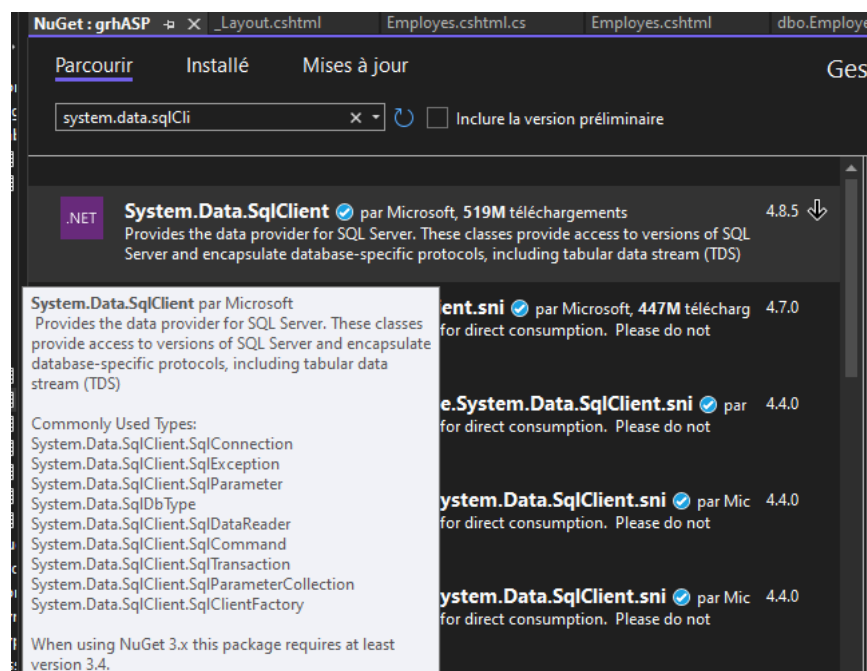




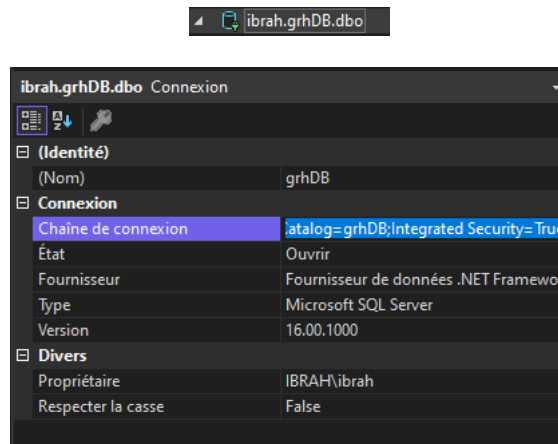
Ensuite je vais dans la gestion des packages Nuget, pour installer la bibliothèque nécessaire pour gerer la data base depuis une app .NET



J'installe : **System.Data.SqlClient**



Je retourne sur mon `EmployesModel` et je récupère également la chaîne de connexion :



J'établis la connexion à la database et je récupère les informations d'un employé depuis la table Employé grâce à **"SELECT \* FROM Employes"** que je stocke dans une liste

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using System.Data.SqlClient;

using System.Threading.Tasks;

namespace grhASP
{
    public class EmployesModel : PageModel
    {
        public List<EmployeeInfo> listEmployee = new List<EmployeeInfo>();

        public void OnGet()
        {
            string connectionString = "Data Source=IBRAH;Initial Catalog=grhDB;Integrated Security=True";
            using (SqlConnection connect = new SqlConnection(connectionString))
            {
                connect.Open();
                string sql = "SELECT * FROM Employes";
                using (SqlCommand command = new SqlCommand(sql, connect))
                {
                    using (SqlDataReader reader = command.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            EmployeeInfo employee = new EmployeeInfo();
                            employee.ID = reader.GetInt32(0);
                            employee.Nom = reader.GetString(1);
                            employee.Prenom = reader.GetString(2);
                            employee.DateNaissance = reader.GetDateTime(3).ToString("yyyy-MM-dd");
                            employee.Adresse = reader.GetString(4);
                            employee.Email = reader.GetString(5);
                            employee.DateEmbauche = reader.GetDateTime(6).ToString("yyyy-MM-dd");
                            employee.ID_Departement = reader.GetInt32(7);
                            employee.ID_Poste = reader.GetInt32(8);

                            decimal salaireDecimal = reader.GetDecimal(9);
                            employee.Salaire = Convert.ToDouble(salaireDecimal);

                            employee.AvantageSante = reader.GetString(10);
                            employee.AvantageRetraite = reader.GetString(11);
                            employee.Sexe = reader.GetString(12);
                            employee.NumeroSecuSociale = reader.GetString(13);
                            employee.Telephone = reader.GetString(14);
                            employee.Derniere_Modif = reader.GetDateTime(15).ToString("yyyy-MM-dd HH:mm:ss");
                            listEmployee.Add(employee);
                        }
                    }
                }
            }
        }
    }
}
```

```

// Pour delete un employé
public IActionResult OnPostDelete(int id)
{
    string connectionString = "Data Source=IBRAH;Initial Catalog=grhDB;Integrated Security=True";
    using var connection = new SqlConnection(connectionString);
    connection.Open();

    string sql = "DELETE FROM Employes WHERE ID = @ID";
    using var command = new SqlCommand(sql, connection);
    command.Parameters.AddWithValue("@ID", id);
    command.ExecuteNonQuery();

    return RedirectToPage();
}

public class EmployeeInfo
{
    public int ID;
    public string Nom;
    public string Prenom;
    public string DateNaissance;
    public string Adresse;
    public string Email;
    public string DateEmbauche;
    public int ID_Departement;
    public int ID_Poste;
    public double Salaire;
    public string AvantageSante;
    public string AvantageRetraite;
    public string Sexe;
    public string NumeroSecuSociale;
    public string Telephone;
    public string Derniere_Modif;
}
}
}

```

J'affiche à présent les infos de la table dans ma page /Employe

```

@page
@model grhASP.EmployesModel

@{
}

<!-- j'affiche la list des employés' -->
<h2>Liste des employés</h2>
<table class="table">
    <thead>
        <tr>
            <th>ID</th>
            <th>Nom</th>
            <th>Prénom</th>
            <th>Date de naissance</th>
            <th>Adresse</th>
            <th>Email</th>
            <th>Date d'embauche</th>
            <th>ID Département</th>
            <th>ID Poste</th>
            <th>Salaire</th>
            <th>Avantage santé</th>
            <th>Avantage retraite</th>
            <th>Sexe</th>
            <th>Numéro de sécurité sociale</th>
            <th>Téléphone</th>
            <th>Dernière modification</th>
            <th>Actions</th>
        </tr>
    </thead>

    <tbody>
        @foreach (var employe in Model.listEmploye)
        {

```

```

<tr>
  <td>@employee.ID</td>
  <td>@employee.Nom</td>
  <td>@employee.Prenom</td>
  <td>@employee.DateNaissance</td>
  <td>@employee.Adresse</td>
  <td>@employee.Email</td>
  <td>@employee.DateEmbauche</td>
  <td>@employee.ID_Departement</td>
  <td>@employee.ID_Poste</td>
  <td>@employee.Salaire</td>
  <td>@employee.AvantageSante</td>
  <td>@employee.AvantageRetraite</td>
  <td>@employee.Sexe</td>
  <td>@employee.NumeroSecuSocial</td>
  <td>@employee.Telephone</td>
  <td>@employee.Derniere_Modif</td>
</td>

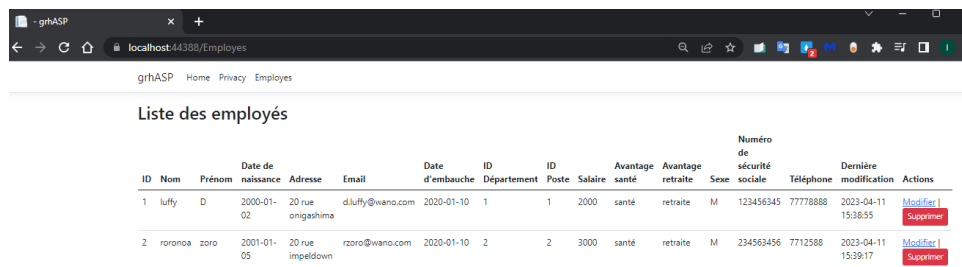
  <a asp-page="./Edit" asp-route-id="@employee.ID">Modifier</a> |

  <!-- Pour delete un employé -->
  <form method="post" asp-page-handler="Delete" asp-route-id="@employee.ID" onsubmit="return confirm('Voulez-vous supprimer cet employé?');">
    <button type="submit" class="btn btn-danger btn-sm">Supprimer</button>
  </form>

</td>
</tr>
}
</tbody>
</table>

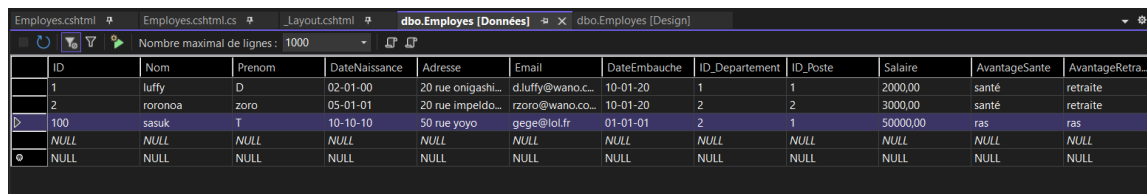
```

J'affiche à présent la liste des employés dans ma page ASP, on voit que tout s'affiche bien



ID	Nom	Prénom	Date de naissance	Adresse	Email	Date d'embauche	ID Département	ID Poste	Salaire	Avantage santé	Avantage retraite	Sexe	Numéro de sécurité sociale	Téléphone	Dernière modification	Actions
1	luffy	D	2000-01-02	20 rue onigashima	d.luffy@wano.com	2020-01-10	1	1	2000	santé	retraite	M	123456345	77778888	2023-04-11 15:38:55	Modifier   Supprimer
2	roronoa	zoro	2001-01-05	20 rue impeldown	rzoro@wano.com	2020-01-10	2	2	3000	santé	retraite	M	234563456	7712588	2023-04-11 15:39:17	Modifier   Supprimer

J'ajoute un nouvelle employé depuis visual studio



ID	Nom	Prenom	DateNaissance	Adresse	Email	DateEmbauche	ID_Departement	ID_Poste	Salaire	AvantageSante	AvantageRetra
1	luffy	D	02-01-00	20 rue onigashi...	d.luffy@wano.co...	10-01-20	1	1	2000,00	santé	retraite
2	roronoa	zoro	05-01-01	20 rue impeldo...	rzoro@wano.co...	10-01-20	2	2	3000,00	santé	retraite
100	sasuk	T	10-10-10	50 rue yoyo	gege@lol.fr	01-01-01	2	1	50000,00	ras	ras
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

On voit qu'il apparait bien dans la page ASP, je vais à présent le delete

## Liste des employés

ID	Nom	Prénom	Date de naissance	Adresse	Email	Date d'embauche	ID Département	ID Poste	Salaire	Avantage santé	Avantage retraite	Sexe	Numéro de sécurité sociale	Téléphone	Dernière modification	Actions
1	luffy	D	2000-01-02	20 rue onigashima	d.luffy@wano.com	2020-01-10	1	1	2000	santé	retraite	M	123456345	77778888	2023-04-11 15:38:55	<a href="#">Modifier</a>   <a href="#">Supprimer</a>
2	roronoa	zoro	2001-01-05	20 rue impeldown	rzoro@wano.com	2020-01-10	2	2	3000	santé	retraite	M	234563456	7712588	2023-04-11 15:39:17	<a href="#">Modifier</a>   <a href="#">Supprimer</a>
100	sasuke	T	2010-10-10	50 rue yoyo	gege@lol.fr	2001-01-01	2	1	50000	ras	ras	M	6161685	89494154	2023-05-26 00:50:54	<a href="#">Modifier</a>   <a href="#">Supprimer</a>

Voici la methode qui gère cela

```
<!-- Pour delete un employé -->
<form method="post" asp-page-handler="Delete" asp-route-id="@employee.ID" onsubmit="return confirm(' supprimer cet employé ?');">
  <button type="submit" class="btn btn-danger btn-sm">Supprimer</button>
</form>
```

On confirme la suppression

localhost:44388 indique supprimer cet employé ?

OK Annuler

ID	Nom	Prénom	Date de naissance	Adresse	Email	Date d'embauche	ID Département	ID Poste	Salaire	Avantage santé	Avantage retraite	Sexe	Numéro de sécurité sociale	Téléphone	Dernière modification	Actions
1	luffy	D	2000-01-02	20 rue onigashima	d.luffy@wano.com	2020-01-10	1	1	2000	santé	retraite	M	123456345	77778888	2023-04-11 15:38:55	<a href="#">Modifier</a>   <a href="#">Supprimer</a>
2	roronoa	zoro	2001-01-05	20 rue impeldown	rzoro@wano.com	2020-01-10	2	2	3000	santé	retraite	M	234563456	7712588	2023-04-11 15:39:17	<a href="#">Modifier</a>   <a href="#">Supprimer</a>
100	sasuke	T	2010-10-10	50 rue yoyo	gege@lol.fr	2001-01-01	2	1	50000	ras	ras	M	6161685	89494154	2023-05-26 00:50:54	<a href="#">Modifier</a>   <a href="#">Supprimer</a>

On voit bien que la suppression à bien été faite

## Liste des employés

ID	Nom	Prénom	Date de naissance	Adresse	Email	Date d'embauche	ID Département	ID Poste	Salaire	Avantage santé	Avantage retraite	Sexe	Numéro de sécurité sociale	Téléphone	Dernière modification	Actions
1	luffy	D	2000-01-02	20 rue onigashima	d.luffy@wano.com	2020-01-10	1	1	2000	santé	retraite	M	123456345	77778888	2023-04-11 15:38:55	<a href="#">Modifier</a>   <a href="#">Supprimer</a>
2	roronoa	zoro	2001-01-05	20 rue impeldown	rzoro@wano.com	2020-01-10	2	2	3000	santé	retraite	M	234563456	7712588	2023-04-11 15:39:17	<a href="#">Modifier</a>   <a href="#">Supprimer</a>



