

## REQUERIMIENTOS OBLIGATORIOS

**Nota: Todas las modificaciones en la base de datos deben hacerse desde el modelo, nunca directamente en la base de datos:**

1. Modificar el contexto de la base de datos para que en la clase Employee el atributo EmployeeNumber sea único, que no permita duplicados

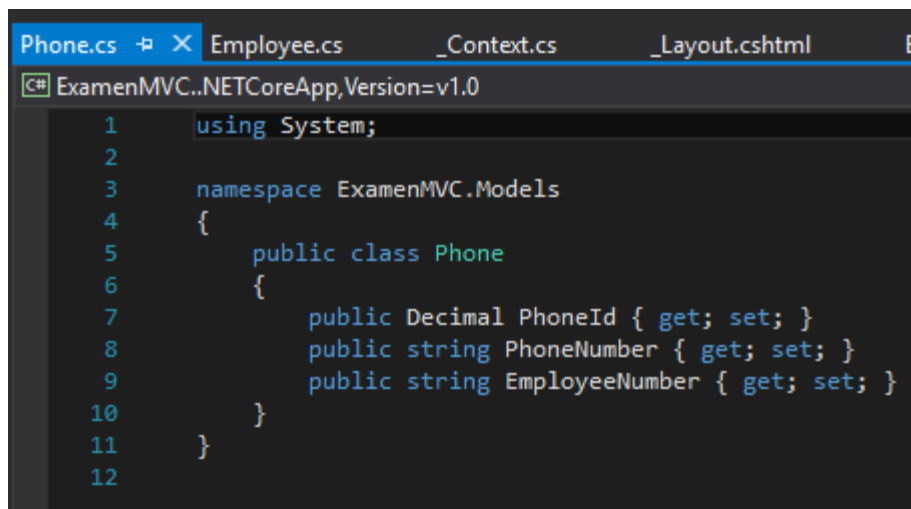
```
modelBuilder.Entity<Employee>(entity =>
{
    entity.ToTable("Employees");
    entity.Property(e => e.EmployeeId).HasColumnName("EmployeeId").HasColumnType("numeric").ValueGeneratedOnAdd();
    entity.Property(e => e.EmployeeNumber).HasColumnName("EmployeeNumber").HasColumnType("varchar(8)");
    entity.Property(e => e.EmployeeName).HasColumnName("EmployeeName").HasColumnType("varchar(200)");
    entity.HasIndex(e => e.EmployeeNumber).IsUnique();
});
```

2. Modificar el contexto de la base de datos para que en la clase Task el atributo TaskName sea único, que no permita duplicados

```
modelBuilder.Entity<Task>(entity =>
{
    entity.ToTable("Taskss");
    entity.Property(e => e.TaskId).HasColumnName("TaskId").HasColumnType("numeric").ValueGeneratedOnAdd();
    entity.Property(e => e.EmployeeId).HasColumnName("EmployeeId").HasColumnType("numeric");
    entity.Property(e => e.TaskName).HasColumnName("TaskName").HasColumnType("varchar(200)");
    entity.HasIndex(e => e.TaskName).IsUnique();
});
```

3. Crear la clase Phone con los siguientes atributos:

- a. Id (identity)
- b. PhoneNumber (string)
- c. EmployeeNumber (string)



```
Phone.cs Employee.cs _Context.cs _Layout.cshtml Er
C# ExamenMVC..NETCoreApp,Version=v1.0
1 using System;
2
3 namespace ExamenMVC.Models
4 {
5     public class Phone
6     {
7         public Decimal PhoneId { get; set; }
8         public string PhoneNumber { get; set; }
9         public string EmployeeNumber { get; set; }
10    }
11 }
12
```

#### 4. Modificar el contexto para crear la definición de la clase Phone

```
namespace ExamenMVC.Models
{
    public partial class _Context : DbContext
    {
        public _Context(DbContextOptions<_Context> options) : base(options) { }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Employee>(entity =>
            {
                entity.ToTable("Employees");
                entity.Property(e => e.EmployeeId).HasColumnName("EmployeeId").HasColumnType("numeric").ValueGeneratedOnAdd();
                entity.Property(e => e.EmployeeNumber).HasColumnName("EmployeeNumber").HasColumnType("varchar(8)");
                entity.Property(e => e.EmployeeName).HasColumnName("EmployeeName").HasColumnType("varchar(200)");
                entity.HasIndex(e => e.EmployeeNumber).IsUnique();
            });

            modelBuilder.Entity<Phone>(entity =>
            {
                entity.ToTable("Phones");
                entity.Property(e => e.PhoneId).HasColumnName("PhoneId").HasColumnType("numeric").ValueGeneratedOnAdd();
                entity.Property(e => e.PhoneNumber).HasColumnName("PhoneNumber").HasColumnType("varchar(20)");
                entity.Property(e => e.EmployeeNumber).HasColumnName("EmployeeNumber").HasColumnType("varchar(8)");
            });

            modelBuilder.Entity<Task>(entity =>
            {
                entity.ToTable("Taskss");
                entity.Property(e => e.TaskId).HasColumnName("TaskId").HasColumnType("numeric").ValueGeneratedOnAdd();
                entity.Property(e => e.EmployeeId).HasColumnName("EmployeeId").HasColumnType("numeric");
                entity.Property(e => e.TaskName).HasColumnName("TaskName").HasColumnType("varchar(200)");
                entity.HasIndex(e => e.TaskName).IsUnique();
            });
        }
    }
}
```

#### 5. Usar el comando **dotnet** (ver puntos 13 y 14 anteriores) para modificar la base de datos

```
Administrador: Símbolo del sistema

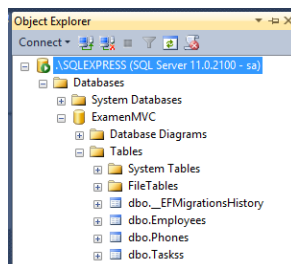
C:\ExamenMVC\src\ExamenMVC>dotnet ef migrations add initial
Project ExamenMVC (.NETCoreApp,Version=v1.0) will be compiled because Input items removed from last build
Compiling ExamenMVC for .NETCoreApp,Version=v1.0
Compilation succeeded.
    0 Warning(s)
    0 Error(s)
Time elapsed 00:00:08.5980342

Done. To undo this action, use 'dotnet ef migrations remove'

C:\ExamenMVC\src\ExamenMVC>dotnet ef database update
Project ExamenMVC (.NETCoreApp,Version=v1.0) will be compiled because Input items added from last build
Compiling ExamenMVC for .NETCoreApp,Version=v1.0
Compilation succeeded.
    0 Warning(s)
    0 Error(s)
Time elapsed 00:00:08.8884564

Done.

C:\ExamenMVC\src\ExamenMVC>
```



6. Modificar la vista **Views/Home/Index.cshtml** y la acción **Index()** el controlador **HomeController.cs** para crear una forma de captura de los datos del empleado, asignarle tareas y teléfonos

ExamenMVC   Home   Employees

## Add employee, phone numbers and tasks

**Employee Name:**

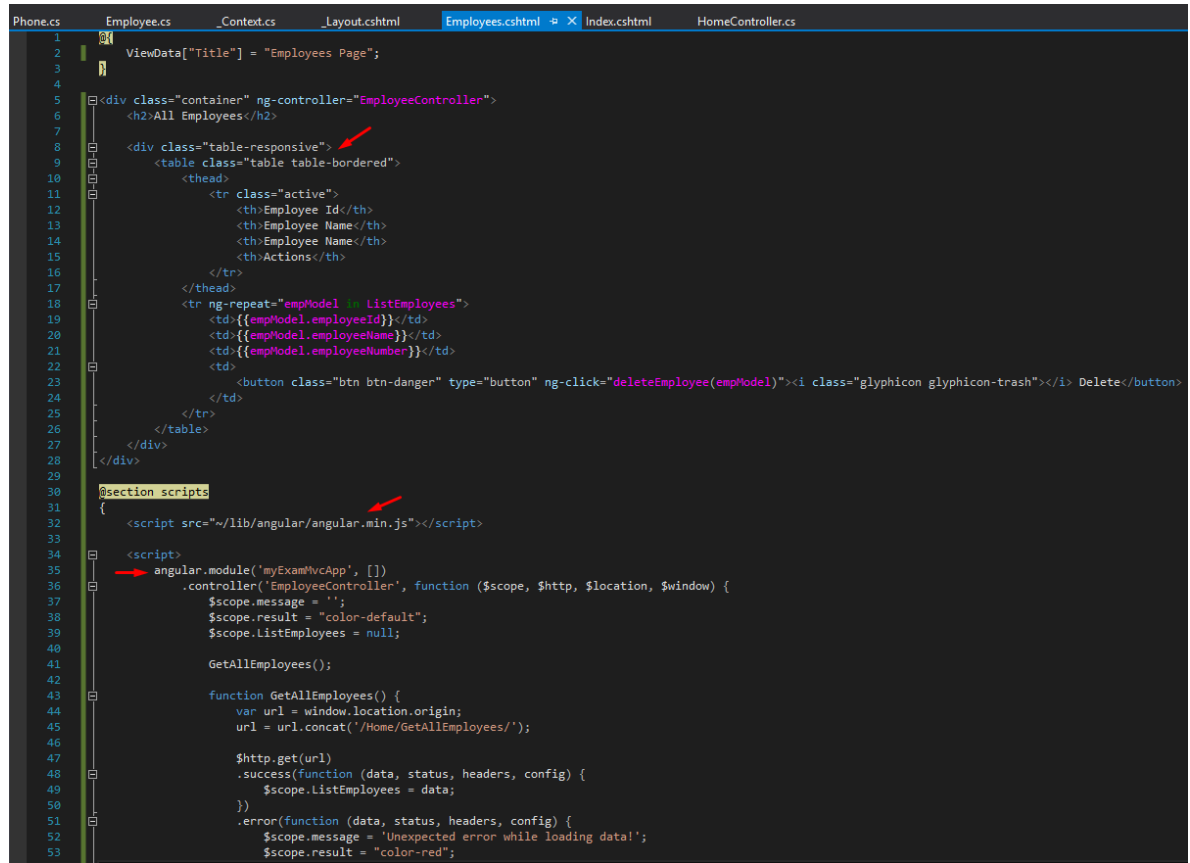
**Employee Number:**

**Phone Numbers:**

**Task Names:**

© 2017 - ExamenMVC - [Antonio Acosta Murillo](#)

7. Crear una vista para consulta y eliminación de empleados con las siguientes características:
- Diseño con Bootstrap
  - Validaciones con AngularJS

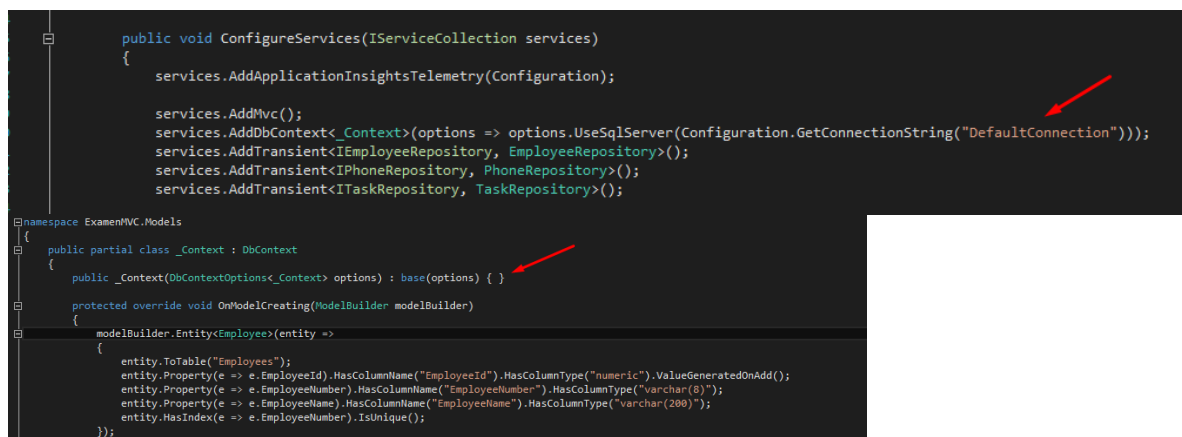


```
1 ViewData["Title"] = "Employees Page";
2
3
4
5 <div class="container" ng-controller="EmployeeController">
6   <h2>All Employees</h2>
7
8   <div class="table-responsive">
9     <table class="table table-bordered">
10      <thead>
11        <tr class="active">
12          <th>Employee Id</th>
13          <th>Employee Name</th>
14          <th>Employee Name</th>
15          <th>Actions</th>
16        </tr>
17      </thead>
18      <tr ng-repeat="empModel in ListEmployees">
19        <td>{{empModel.employeeId}}</td>
20        <td>{{empModel.employeeName}}</td>
21        <td>{{empModel.employeeNumber}}</td>
22        <td>
23          <button class="btn btn-danger" type="button" ng-click="deleteEmployee(empModel)"><i class="glyphicon glyphicon-trash"></i> Delete</button>
24        </td>
25      </tr>
26    </table>
27  </div>
28 </div>
29
30 <section scripts>
31 {
32   <script src="~/lib/angular/angular.min.js"></script>
33
34   <script>
35     angular.module('myExamMvcApp', [])
36       .controller('EmployeeController', function ($scope, $http, $location, $window) {
37         $scope.message = '';
38         $scope.result = "color-default";
39         $scope.ListEmployees = null;
40
41         GetAllEmployees();
42
43         function GetAllEmployees() {
44           var url = window.location.origin;
45           url = url.concat('/Home/GetAllEmployees/');
46
47           $http.get(url)
48             .success(function (data, status, headers, config) {
49               $scope.ListEmployees = data;
50             })
51             .error(function (data, status, headers, config) {
52               $scope.message = 'Unexpected error while loading data!';
53               $scope.result = "color-red";
```

## REQUERIMIENTOS OPTATIVOS

Nota. Aún aunque los siguientes requerimientos son optativos se tomarán en cuenta como discriminador para la selección del candidato.

- Configurar la cadena de conexión usando Dependency Injection



```
public void ConfigureServices(IServiceCollection services)
{
    services.AddApplicationInsightsTelemetry(Configuration);

    services.AddMvc();
    services.AddDbContext<_Context>(options => options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));
    services.AddTransient<IEmployeeRepository, EmployeeRepository>();
    services.AddTransient<IPhoneRepository, PhoneRepository>();
    services.AddTransient<ITaskRepository, TaskRepository>();
}

namespace ExamMvc.Models
{
    public partial class _Context : DbContext
    {
        public _Context(DbContextOptions<_Context> options) : base(options) { }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Employee>(entity =>
            {
                entity.ToTable("Employees");
                entity.Property(e => e.EmployeeId).HasColumnName("EmployeeId").HasColumnType("numeric").ValueGeneratedOnAdd();
                entity.Property(e => e.EmployeeNumber).HasColumnName("EmployeeNumber").HasColumnType("varchar(8)");
                entity.Property(e => e.EmployeeName).HasColumnName("EmployeeName").HasColumnType("varchar(200)");
                entity.HasIndex(e => e.EmployeeNumber).IsUnique();
            });
        }
    }
}
```

## 2. Crear una vista para ver la lista de todos los empleados usando una API REST

```
50
51  → [HttpGet]
52  public JsonResult GetAllEmployees()
53  {
54      var employees = employeeRepository.GetAllEmployees().ToList();
55      return Json(employees);
56  }
57
58  → [HttpDelete]
59  public JsonResult DeleteEmployee(decimal id)
60  {
61      var status = false;
62      var employee = employeeRepository.FindEmployee(id);
63
64      if(employee != null )
65      {
66          taskRepository.deleteTask(employee.EmployeeId);
67          phoneRepository.deletePhone(employee.EmployeeNumber);
68          employeeRepository.DeleteEmployee(employee.EmployeeId);
69          status = true;
70      }
71
72      return Json(new { success = status });
73  }
```