

DEPARTAMENTO: Ciencias de la Ingeniería

CARRERA: Sistemas de Información

CURSO: Octavo **PARALELO:** "A"

ASIGNATURA: Plataformas de Desarrollo 2

PROFESOR: Mg. Luis Fernando Aguas B.

ESTUDIANTE: Marco Antonio Ayala Lituma

DESCRIPCIÓN: Tarea Semana 6

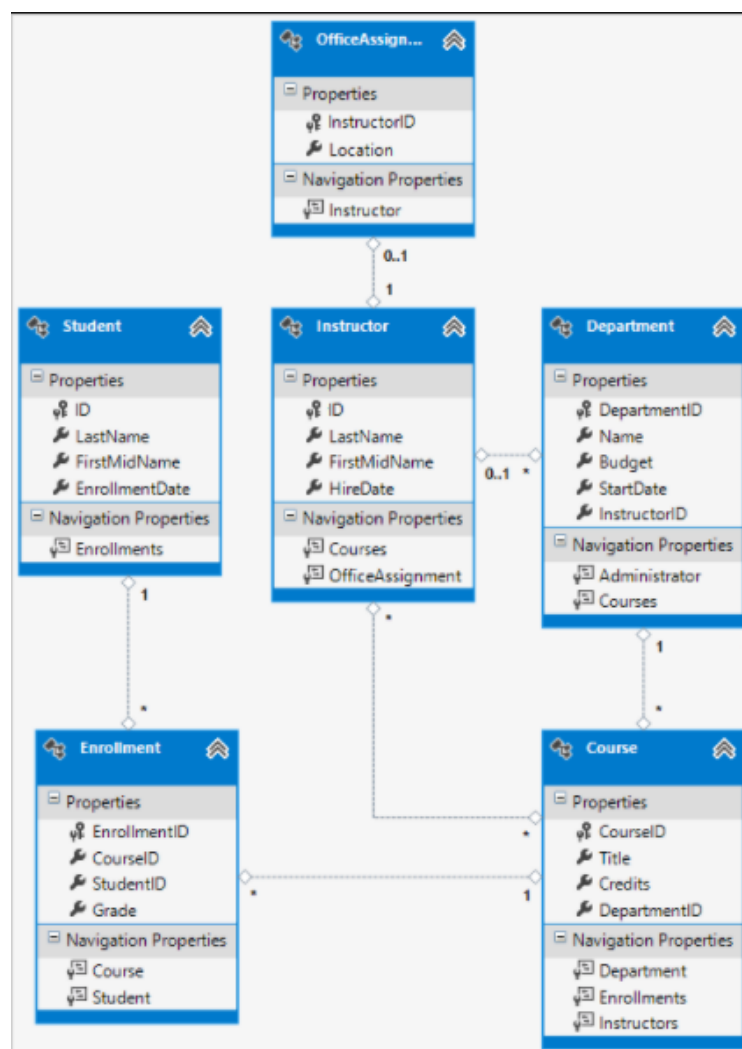
TEMA:

Investigue un código en C#, en el cual se aplique MVC

DESARROLLO:

Vamos a investigar el código fuente del proyecto por ejemplo de ContosoUniversity de la publicación de Microsoft.

Empezamos analizando el diagrama de clases del proyecto.



Aquí podemos visualizar el modelo de la clase Student la misma que contiene algunas etiquetas interesantes como los tipos de dato con DataAnnotations.

El atributo DataType se utiliza para especificar un tipo de datos más específico que el tipo intrínseco de la base de datos. En este caso solo se quiere realizar el seguimiento de la fecha, no de la fecha y la hora. La enumeración DataType proporciona muchos tipos de datos, como Date, Time, PhoneNumber, Currency, EmailAddress, etc. El atributo DataType también puede permitir que la aplicación proporcione automáticamente características específicas del tipo. Por ejemplo, se puede crear un vínculo de mailto: para DataType. EmailAddressy se puede proporcionar un selector de fecha para DataType. Date en exploradores compatibles con HTML5. Los atributos DataType emiten atributos de datos HTML 5 (pronunciados con guiones de datos) que los exploradores HTML 5 pueden comprender. Los atributos DataType no proporcionan ninguna validación.

```
C# Copiar

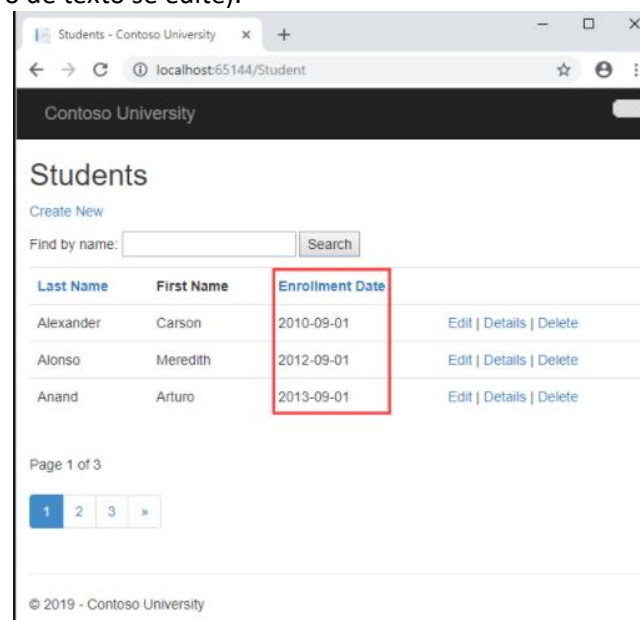
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace ContosoUniversity.Models
{
    public class Student
    {
        public int ID { get; set; }
        public string LastName { get; set; }
        public string FirstMidName { get; set; }
        [DataType(DataType.Date)]
        [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
        public DateTime EnrollmentDate { get; set; }

        public virtual ICollection<Enrollment> Enrollments { get; set; }
    }
}
```

Una visualización rápida del código generado de la vista Student.

El valor ApplyFormatInEditMode especifica que el formato especificado también debe aplicarse cuando el valor se muestra en un cuadro de texto para su edición. (Es posible que no desee que en algunos campos, por ejemplo, para los valores de moneda, no desee que el símbolo de moneda en el cuadro de texto se edite).



Last Name	First Name	Enrollment Date	
Alexander	Carson	2010-09-01	Edit Details Delete
Alonso	Meredith	2012-09-01	Edit Details Delete
Anand	Arturo	2013-09-01	Edit Details Delete

También puede especificar reglas de validación de datos y mensajes de error de validación mediante atributos. El atributo `StringLength` establece la longitud máxima en la base de datos y proporciona la validación del lado cliente y del lado servidor para ASP.NET MVC. En este atributo también se puede especificar la longitud mínima de la cadena, pero el valor mínimo no influye en el esquema de la base de datos.

Imagine que quiere asegurarse de que los usuarios no escriban más de 50 caracteres para un nombre. Para agregar esta limitación, agregue los atributos `StringLength` a las propiedades `LastName` y `FirstMidName`, como se muestra en el ejemplo siguiente:

```
C# Copiar

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace ContosoUniversity.Models
{
    public class Student
    {
        public int ID { get; set; }
        [StringLength(50)]
        public string LastName { get; set; }
        [StringLength(50, ErrorMessage = "First name cannot be longer than 50 characters.")]
        public string FirstMidName { get; set; }
        [DataType(DataType.Date)]
        [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
        public DateTime EnrollmentDate { get; set; }

        public virtual ICollection<Enrollment> Enrollments { get; set; }
    }
}
```

Ahora en la clase `Student` editamos con los siguientes campos y atributos personalizados.

```
C# Copiar

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace ContosoUniversity.Models
{
    public class Student
    {
        public int ID { get; set; }
        [Required]
        [StringLength(50)]
        [DisplayName = "Last Name"]
        public string LastName { get; set; }
        [Required]
        [StringLength(50, ErrorMessage = "First name cannot be longer than 50 characters.")]
        [Column("FirstName")]
        [DisplayName = "First Name"]
        public string FirstMidName { get; set; }
        [DataType(DataType.Date)]
        [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
        [DisplayName = "Enrollment Date"]
        public DateTime EnrollmentDate { get; set; }

        [DisplayName = "Full Name"]
        public string FullName
        {
            get
            {
                return LastName + ", " + FirstMidName;
            }
        }

        public virtual ICollection<Enrollment> Enrollments { get; set; }
    }
}
```



Revisamos el atributo required hace que las propiedades de nombre sean campos obligatorios. El Required attribute no es necesario para los tipos de valor como DateTime, int, Double y Float. A los tipos de valor no se les puede asignar un valor null, por lo que se tratan de forma inherente como campos obligatorios.

```
C# Copiar

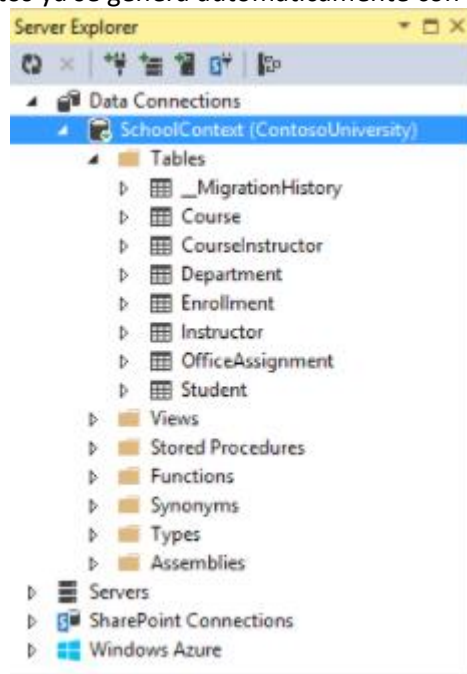
[DisplayName = "Last Name"]
[Required]
[StringLength(50, MinimumLength=2)]
public string LastName { get; set; }
```

Para generar la base de datos se puede actualizar el modelos de la base de datos con la cadena de conexión.

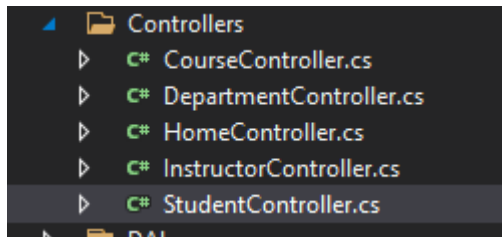
```
XML Copiar

<add name="SchoolContext" connectionString="Data Source=(LocalDb)\v11.0;Initial Catalog=CU_Test;Integrated Se
providerName="System.Data.SqlClient" />
```

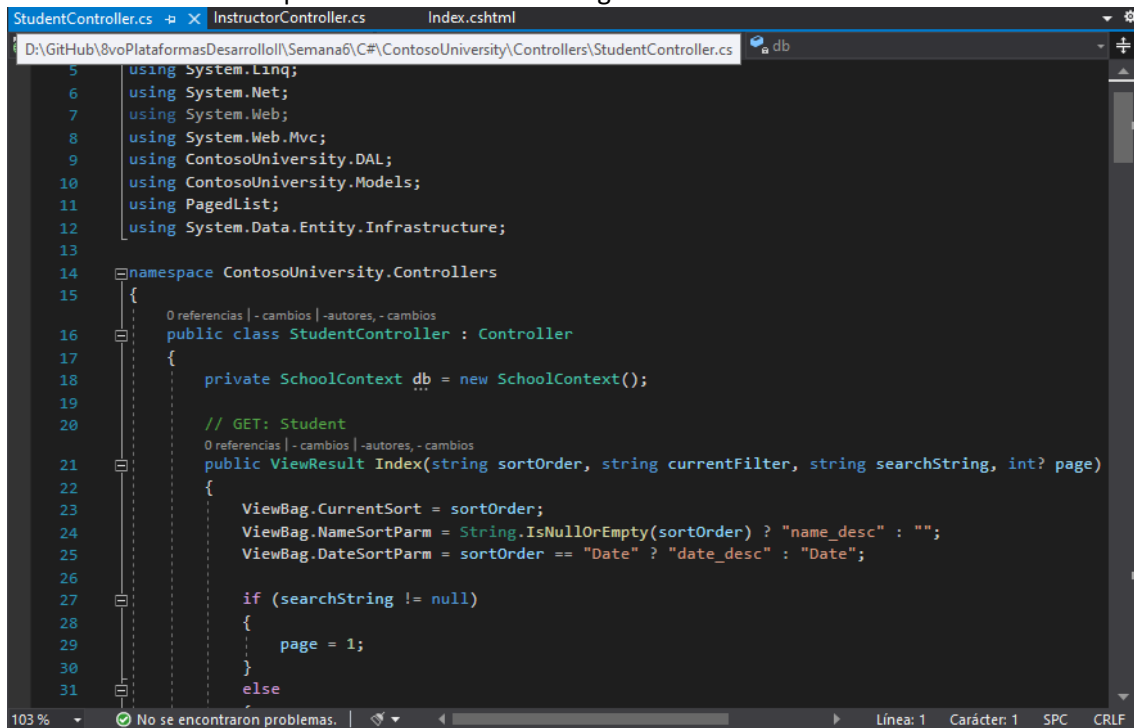
El modelo de la base de datos ya se genera automáticamente con el script.



Ahora revisamos que esta carpeta de Controladores contiene la clase StudentController.

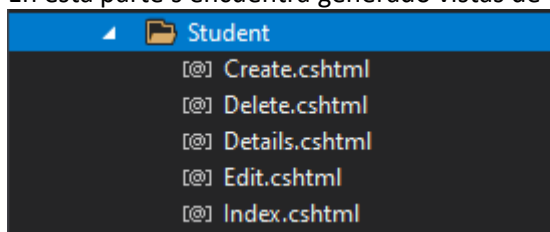


Si revisamos el archivo podemos visualizar el código fuente.



```
5 using System.Linq;
6 using System.Net;
7 using System.Web;
8 using System.Web.Mvc;
9 using ContosoUniversity.DAL;
10 using ContosoUniversity.Models;
11 using PagedList;
12 using System.Data.Entity.Infrastructure;
13
14 namespace ContosoUniversity.Controllers
15 {
16     0 referencias | - cambios | -autores, - cambios
17     public class StudentController : Controller
18     {
19         private SchoolContext db = new SchoolContext();
20
21         // GET: Student
22         0 referencias | - cambios | -autores, - cambios
23         public ActionResult Index(string sortOrder, string currentFilter, string searchString, int? page)
24         {
25             ViewBag.CurrentSort = sortOrder;
26             ViewBag.NameSortParm = String.IsNullOrEmpty(sortOrder) ? "name_desc" : "";
27             ViewBag.DateSortParm = sortOrder == "Date" ? "date_desc" : "Date";
28
29             if (searchString != null)
30             {
31                 page = 1;
32             }
33             else
34             {
35             }
```

En esta parte se encuentra generado vistas de la clase de Student del modelo

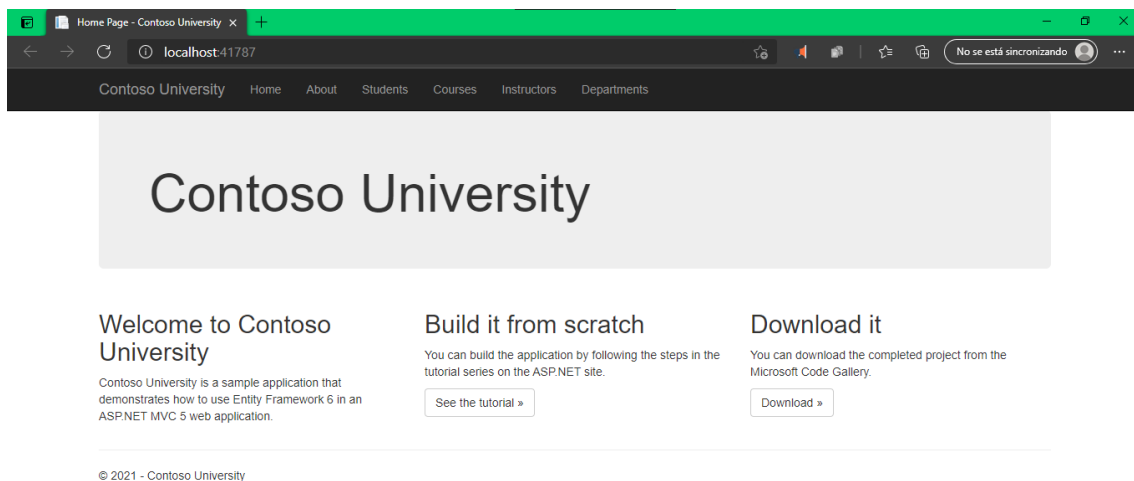


Si abrimos el archivo index de student se visualiza de esta manera, las misma que maneja etiquetas de razor.



```
Index.cshtml
1 @model PagedList.IPagedList<ContosoUniversity.Models.Student>
2 using PagedList.Mvc;
3 <link href="~/Content/PagedList.css" rel="stylesheet" type="text/css" />
4
5
6 ViewBag.Title = "Students";
7
8 <h2>Students</h2>
9
10
11 <p>
12     <a href="#">Create New</a>
13 </p>
14 using (Html.BeginForm("Index", "Student", FormMethod.Get))
15 {
16     <p>
17         Find by name: <input type="text" value="@ViewBag.CurrentFilter as string" />
18         <input type="submit" value="Search" />
19     </p>
20
21     <table class="table">
22     <tr>
23         <th>
24             <a href="#">Last Name</a>
25         </th>
26         <th>
27             First Name
28         </th>
29         <th>
30             <a href="#">Enrollment Date</a>
31         </th>
32     </tr>
33 </table>
34 }
```

El proyecto ejecutándose se visualiza de esta manera



COMENTARIO:

La rama de la ingeniería del software se preocupa por crear procesos que aseguren calidad en los programas que se realizan y esa calidad atiende a diversos parámetros que son deseables para todo desarrollo, como la estructuración de los programas o reutilización del código, lo que debe influir positivamente en la facilidad de desarrollo y el mantenimiento.

Los ingenieros del software se dedican a estudiar de qué manera se pueden mejorar los procesos de creación de software y una de las soluciones a las que han llegado es la arquitectura basada en capas que separan el código en función de sus responsabilidades o conceptos. Por tanto, cuando estudiamos MVC lo primero que tenemos que saber que son buenas practicas.

BIBLIOGRAFÍA:

Creación aplicación ASP.NET MVC, recuperador en.



[Tutorial: creación de un modelo de datos más complejo para una aplicación ASP.NET MVC | Microsoft Docs](#)

