**MAESTRO DETALLE CON ENTITY FRAMEWORK MVC USANDO JS**

https://www.youtube.com/watch?v=YKOcp\_Y9w4k

Creamos un proyecto mvc con autenticación

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Creamos la base de datos y las tablas:**

USE master ;

GO

CREATE DATABASE MaestroDetalle

ON

( NAME = MaestroDetalle\_dat,

FILENAME = 'C:\Users\Ego\Documents\ARCHIVOS HP SYNC\CURSO ASP.NET C#\CURSO COMPLETO ASP MVC 5 C#\MaestroDetalle\MaestroDetalle.mdf',

SIZE = 10,

MAXSIZE = 50,

FILEGROWTH = 5 )

LOG ON

( NAME = MaestroDetalle\_log,

FILENAME = 'C:\Users\Ego\Documents\ARCHIVOS HP SYNC\CURSO ASP.NET C#\CURSO COMPLETO ASP MVC 5 C#\MaestroDetalle\MaestroDetalle.ldf',

SIZE = 5MB,

MAXSIZE = 25MB,

FILEGROWTH = 5MB ) ;

GO

--CREAR LAS TABLAS

USE MaestroDetalle;

GO

CREATE TABLE Venta(

Id INT PRIMARY KEY IDENTITY(1,1),

Fecha DATETIME,

Cliente VARCHAR(50),

)

CREATE TABLE Concepto(

Id INT PRIMARY KEY IDENTITY(1,1),

IdVenta INT,

Cantidad INT,

Nombre VARCHAR(50),

PrecioUnitario DECIMAL(18,2),

Total DECIMAL(18,2)

)

**Las relacionamos las tablas:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Que no permitan valore null:**

Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

**Conexión a la base de datos:**

Usamos ADO.NET

**Captura de pantalla de computadora

Descripción generada automáticamente**

en nueva conexión

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

ingresamos el nombre del servidor

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Aceptar

Interfaz de usuario gráfica, Aplicación, Correo electrónico

Descripción generada automáticamente

Si incluir datos confidenciales

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Seleccionamos las tablas:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Vemos que reconoce la relación entre las tablas:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Entity framework a creado las clases por nosotros:

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

**En Web.config Indicamos el mismo ruta a la base de datos en el connectionString a DefaultConnection para que use la misma base de datos para las tablas de los registros del login:**

Copiamos en DefaultConnection, el data source hasta **MultipleActiveResultSets=True**

<add name="DefaultConnection" connectionString="**data source=DESKTOP-J1SSK74\SQLEXPRESS;initial catalog=MaestroDetalle;user id=sa;password=Sqlexp;MultipleActiveResultSets=True**" providerName="System.Data.SqlClient" />

**<add name="MaestroDetalleEntities"**

connectionString="metadata=res://\*/Models.DBMaestroDetalle.csdl|res://\*/Models.DBMaestroDetalle.ssdl|res://\*/Models.DBMaestroDetalle.msl;provider=System.Data.SqlClient;provider connection string=&quot;**data source=DESKTOP-J1SSK74\SQLEXPRESS;initial catalog=MaestroDetalle;user id=sa;password=Sqlexp;MultipleActiveResultSets=True**;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />

**corremos la aplicación:**

nos registramos

[eliezer@gmail.com](mailto:eliezer@gmail.com)

Eliezer+1

Interfaz de usuario gráfica, Texto, Sitio web

Descripción generada automáticamente

Vemos que las tablas del login se han creado en nuestra base de datos:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

**MAESTRO:**

**Agregamos un controlador en blanco llamado MaestroDetalle:**

Interfaz de usuario gráfica, Texto, Aplicación

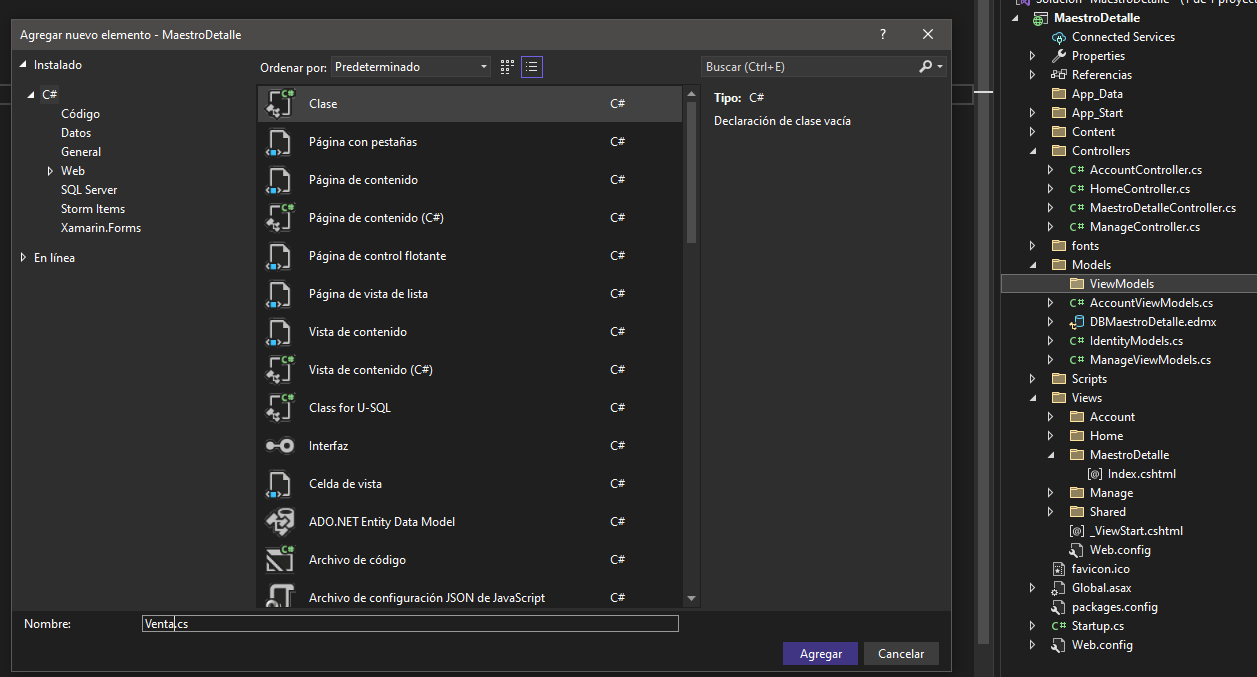
Descripción generada automáticamente

**Agregamos una vista vacía:**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**En Models Creamos una nueva carpeta ViewModel y dentro creamos un nueva clase Ventas.cs que es el modelo de vetas:**



**MODELO VentaViewModel contiene al modelo Concepto:**

Lo llamamos VentaViewModel para que no se confunda con Venta que creo entityframework

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace MaestroDetalle.Models.ViewModels

{

//Modelo de Ventas

//indicamos los campos que usaremos

public class VentaViewModel

{

public string Cliente { get; set; }

public List<Concepto> Conceptos { get; set; }

}

//Modelo de Concepto

//indicamos los campos que usaremos

public class Concepto

{

public int Cantidad { get; set; }

public string Nombre { get; set; }

public decimal PrecioUnitario { get; set; }

}

}

**CONTROLLER:**

//usamos nuestro modelos

using MaestroDetalle.Models;

using MaestroDetalle.Models.ViewModels;

namespace MaestroDetalle.Controllers

{

public class MaestroDetalleController : Controller

{

// GET: MaestroDetalle

[HttpGet]

public ActionResult Add()

{

return View();

}

//Agrgar registro

[HttpPost]

public ActionResult Add(VentaViewModel model)

{

try

{

//conectar a la base de datos usando entity

using (MaestroDetalleEntities db = new MaestroDetalleEntities())

{

//creamos un objeto tipo Ventas para cargar los datos del modelo

Venta venta = new Venta();

venta.Fecha = DateTime.Now;

venta.Cliente = model.Cliente;

//agregamos el objeto ventas a la base de datos y guardamos

db.Venta.Add(venta);

db.SaveChanges();

}

//como tenemos otra vista que se llama Add nos retorna a ella misma sin necesidad de indicar su nombre

return View();

}

catch(Exception ex)

{

return View(model);

}

}

}

}

**VISTA:**

@model MaestroDetalle.Models.ViewModels.VentaViewModel

@{

ViewBag.Title = "Maestro Detalle";

}

<h2>@ViewBag.Title</h2>

<!--usamos el helper para el formulario-->

<!--le indicamos 3 parametris: "Metodo", "Controlador", FormMethod.Post-->

@using (Html.BeginForm("Add", "MaestroDetalle", FormMethod.Post))

{

<div class="row">

<div class="form-group">

@Html.LabelFor(v => v.Cliente)

@Html.TextBoxFor(V => V.Cliente)

</div>

<div class="form-group">

<input type="submit"/>

</div>

</div>

}

**CORREMOS EL PROYECTO:**

Vemos que ya podemos insertar:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Se agrego el registro a la base de datos:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

**DETALLE:**

USAREMOS JAVA SCRIPT PARA HACER EL DETALLE DINAMICO:

**DETALLE ESTATICO**

**CON DATOS QUEMADOS** PARA PROBAR QUE FUNCIONA:

Controller:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

//usamos nuestro modelos

using MaestroDetalle.Models;

using MaestroDetalle.Models.ViewModels;

namespace MaestroDetalle.Controllers

{

public class MaestroDetalleController : Controller

{

// GET: MaestroDetalle

[HttpGet]

public ActionResult Add()

{

return View();

}

//Agrgar registro

[HttpPost]

public ActionResult Add(VentaViewModel model)

{

try

{

//conectar a la base de datos usando entity

using (MaestroDetalleEntities db = new MaestroDetalleEntities())

{

//creamos un objeto tipo Ventas para cargar los datos del modelo

Venta venta = new Venta();

venta.Fecha = DateTime.Now;

venta.Cliente = model.Cliente;

//agregamos el objeto ventas a la base de datos y guardamos

db.Venta.Add(venta);

//al momento de guardar se le genera a ventas un id

db.SaveChanges();

//Agregar los Conceptos a la base de datos

//debemos reccorrer model.Conceptos

foreach (var ConceptoModel in model.Conceptos)

{

//creamos un objeto para llenar los campos que vienen en el model y los campos que se calcular automaticamente

Concepto concepto = new Concepto();

concepto.Cantidad = ConceptoModel.Cantidad;

concepto.Nombre = ConceptoModel.Nombre;

concepto.PrecioUnitario = ConceptoModel.PrecioUnitario;

concepto.Total = ConceptoModel.Cantidad \* ConceptoModel.PrecioUnitario;

//son los id que estan relacionados

concepto.IdVenta = venta.Id;

//agregamos el objeto a la tabla de la db

db.Concepto.Add(concepto);

}

//gurdamos los cambios en la db

db.SaveChanges();

}

ViewBag.Mensaje = "Registro Insertado en la DB";

//como tenemos otra vista que se llama Add nos retorna a ella misma sin necesidad de indicar su nombre

return View();

}

catch(Exception ex)

{

return View(model);

}

}

}

}

**VISTA CON DATOS QUEMADOS PARA LLENAR LA LISTA DEL DETALLE DE CONCEPTOS:**

@model MaestroDetalle.Models.ViewModels.VentaViewModel

@{

ViewBag.Title = "Maestro Detalle";

}

<h2>@ViewBag.Title</h2>

<div style="color: forestgreen">

@ViewBag.Mensaje

</div>

<!--usamos el helper para el formulario-->

<!--le indicamos 3 parametris: "Metodo", "Controlador", FormMethod.Post-->

@using (Html.BeginForm("Add", "MaestroDetalle", FormMethod.Post))

{

<div class="row">

<h3>Maestro:</h3>

<div class="form-group">

@Html.LabelFor(v => v.Cliente)

@Html.TextBoxFor(v => v.Cliente, new { @class = "form-control" })

</div>

<hr />

</div>

<div class="row">

<!--input que agregan los valores a los campos-->

<div class="form-group">

<label for="Cantidad">Cantidad</label>

<input type="number" class="form-control" id="Cantidad" placeholder="Ingres una Cantidad">

</div>

<div class="form-group">

<label for="Nombre">Nombre</label>

<input type="text" class="form-control" id="Nombre" placeholder="Ingres un Nombre">

</div>

<div class="form-group">

<label for="Precio">Precio</label>

<input type="number" class="form-control" id="Precio" placeholder="Ingres un Precio">

</div>

<!--return false, para que no realice el submit-->

<input type="button" value="Agregar" class="btn btn-success" onclick="AgregaConcepto(); return false" />

<h3>Detalle:</h3>

<!--el datalle va a llenar una tabla de forma automatica,

agregamos un registro quemado solo para que se vea un registro

usaremos javaScript para que se llene automaticamente.

-->

<div id="divConceptos">

<table id="tablaConceptos">

<tr>

<th>Cantidad</th>

<th>Nombre</th>

<th>Precio Unitario</th>

<th>Total</th>

</tr>

<tr>

<td>2</td>

<td>Papa Fritas</td>

<td>10</td>

<td>20</td>

</tr>

<tr>

<td>1</td>

<td>Hamburguesa</td>

<td>10</td>

<td>10</td>

</tr>

</table>

<!--Usamos un input type = hidden para que no se muestren los valores

y name = Concpetos.Index, name debe tener el mismo nombre que el campo Conceptos de tipo lista del modelo VentasViewModel

para que MVC lo reconozca y pueda hacer el mach.

el primer input tiene un name=Conceptos.Index y value = "0"

es el input que le indica el indice a la lista Conceptos en el controller.-->

<input type="hidden" name="Conceptos.Index" id="Conceptos.Index" value="0" />

<!--envia los valores que tendra la lista Conceptos en el indice 0

el controller obtiene los valores por medio de [HttpPost]-->

<input type="hidden" name="Conceptos[0].Cantidad" id="Conceptos[0].Cantidad" value="2" />

<input type="hidden" name="Conceptos[0].Nombre" id="Conceptos[0].Nombre" value="Papas Fritas" />

<input type="hidden" name="Conceptos[0].PrecioUnitario" id="Conceptos[0].PrecioUnitario" value="10" />

<!--index el indice de la lista continua en 1, value = 1, es el valor que continua del indice de la lista de conceptos-->

<input type="hidden" name="Conceptos.Index" id="Conceptos.Index" value="1" />

<!--insertar los valores en la lista Conceptos en el indice 1-->

<input type="hidden" name="Conceptos[1].Cantidad" id="Conceptos[1].Cantidad" value="1" />

<input type="hidden" name="Conceptos[1].Nombre" id="Conceptos[1].Nombre" value="Hanburguesa" />

<input type="hidden" name="Conceptos[1].PrecioUnitario" id="Conceptos[1].PrecioUnitario" value="10" />

</div>

<div class="form-group">

<input type="submit" />

</div>

</div>

<!--Funcion JS que agrega conceptos a la tabla cuando se haga clic en el boton Agregar-->

<script>

function AgregaConcepto() {

//obtener los valores de los input:

let Cantidad = document.getElementById("Cantidad").value;

let Nombre = document.getElementById("Nombre").value;

let Precio = document.getElementById("Precio").value;

//Obtener la tabla tablaConceptos

let Tabla = document.getElementById("tablaConceptos");

//Crear en memotia los tr de la tabla

let TR = document.createElement("tr");

//Crear en memoria los td de la tabla

let TDCantidad = document.createElement("td");

let TDNombre = document.createElement("td");

let TDPrecio = document.createElement("td");

let TDTotal = document.createElement("td");

//Agregar nombres a los td, usamos las let de los valores de los input

TDCantidad.innerHTML = Cantidad;

TDNombre.innerHTML = Nombre;

TDPrecio.innerHTML = Precio;

//Total lo vamos a calcular en el back end para evitar errores, lo usaremos aqui de forma ilustrativa

TDTotal.innerHTML = parseFloat(Cantidad) \* parseFloat(Precio);

//Agregar al TR los td

TR.appendChild(TDCantidad);

TR.appendChild(TDNombre);

TR.appendChild(TDPrecio);

TR.appendChild(TDTotal);

//Agregar a la Tabla los TR

Tabla.appendChild(TR);

//Agregamos los hiddems para que se haga mach con el modelo

//Borrar los valores de los input

document.getElementById("Cantidad").value = "";

document.getElementById("Nombre").value = "";

document.getElementById("Precio").value = "";

}

</script>

}

**Corremos el proyecto:**

Enviamos un registro en el maestro

Y dos registros en el detalle:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

El maestro enviado a la base de datos el Luna con id 9

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Podemos ver su detalle con dos registros:

idVenta = 9

Tabla

Descripción generada automáticamente

**VAMOS A CRERA UN REPOSITORIO LOCAL EN GIT PARA IR GURDADNDO LOS CAMBIOS:**

Git—Crear Repositorio-solo local-crear

**Captura de pantalla de un celular

Descripción generada automáticamente**

**Se ha creado la carpeta git dentro de nuestro proyecto:**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Se han colocado candados los directorios de nuestro proycto indique que estamos usando git:**

**Pantalla de computadora con letras

Descripción generada automáticamente con confianza media**

**Usamos sourceTree para gestionar nuestro proyecto**

[**https://www.youtube.com/watch?v=NTFnC7gF64Q**](https://www.youtube.com/watch?v=NTFnC7gF64Q)

**Lo abrimos en local:**

Le damos en add

**Interfaz de usuario gráfica, Texto, Aplicación, Word, Correo electrónico

Descripción generada automáticamente**

**Listo estamos en la rama master:**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Creamos un commit inicial**

**Creamos una nueva rama:**

[**https://www.youtube.com/watch?v=t-2B9yIlICk**](https://www.youtube.com/watch?v=t-2B9yIlICk)

La llamamos ConceptosJS

**Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente**

**Nos colocamos sobre la nueva rama que es una copia de master:**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

**En visual studio .net:**

Vemos que nos ubica en la nueva rama:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Listo ya podemos trabajar de forma local sobre la nueva rama.

E ir realizando los commit necesarios para guardar los cambios.

**ACTUALIZAR BOOTSTRAP:**

**Actualizamos el \_Layout.cshhtml**

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@ViewBag.Title - Mi aplicación ASP.NET</title>

@Styles.Render("~/Content/css")

@Scripts.Render("~/bundles/modernizr")

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">

@Html.ActionLink("Inicio", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="navbar-collapse collapse" id="navbarSupportedContent">

<ul class="navbar-nav mr-auto">

<li class="nav-item active">

@Html.ActionLink("Home", "Index", "Home", null, new { @class = "nav-link" })

</li>

<li class="nav-item">

@Html.ActionLink("Acerca de", "About", "Home", null, new { @class = "nav-link" })

</li>

<li class="nav-item dropdown">

<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-expanded="false">

Dropdown

</a>

<div class="dropdown-menu" aria-labelledby="navbarDropdown">

@Html.ActionLink("Contacto", "Contact", "Home", null, new { @class = "dropdown-item" })

<a class="dropdown-item" href="#">Another action</a>

<div class="dropdown-divider"></div>

<a class="dropdown-item" href="#">Something else here</a>

</div>

</li>

<li class="nav-item">

<a class="nav-link disabled">Disabled</a>

</li>

</ul>

@Html.Partial("\_LoginPartial")

<form class="form-inline my-2 my-lg-0">

<input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">

<button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>

</form>

</div>

</nav>

<div class="container body-content">

@RenderBody()

<hr />

<footer>

<p>&copy; @DateTime.Now.Year - Mi aplicación ASP.NET</p>

</footer>

</div>

@Scripts.Render("~/bundles/jquery")

@Scripts.Render("~/bundles/bootstrap")

@RenderSection("scripts", required: false)

</body>

</html>

**Actualizar el \_LoginPartial.cshtml**

Le agregamos la clase nav-link

else

{

<ul class="nav navbar-nav navbar-right">

<li>@Html.ActionLink("Registrarse", "Register", "Account", routeValues: null, htmlAttributes: new { id = "registerLink", @class = "nav-link" })</li>

<li>@Html.ActionLink("Iniciar sesión", "Login", "Account", routeValues: null, htmlAttributes: new { id = "loginLink", @class = "nav-link" })</li>

</ul>

}

**DETALLE DINAMICO USANDO JAVA SCRIPT:**

<https://www.youtube.com/watch?v=YKOcp_Y9w4k>

**VIEWMODELOS:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace MaestroDetalle.Models.ViewModels

{

//Modelo de Ventas

//indicamos los campos que usaremos

public class VentaViewModel

{

public string Cliente { get; set; }

public List<ConceptoViewModel> Conceptos { get; set; }

}

//Modelo de Concepto

//indicamos los campos que usaremos

//el resto de campos los calcularemos atomaticamente y estan en el modelo Concepto que crea entity de forma automatica

public class ConceptoViewModel

{

public int Cantidad { get; set; }

public string Nombre { get; set; }

public decimal PrecioUnitario { get; set; }

}

}

**CONTROLLER:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

//usamos nuestro modelos

using MaestroDetalle.Models;

using MaestroDetalle.Models.ViewModels;

namespace MaestroDetalle.Controllers

{

public class MaestroDetalleController : Controller

{

// GET: MaestroDetalle

[HttpGet]

public ActionResult Add()

{

return View();

}

//Agrgar registro

[HttpPost]

public ActionResult Add(VentaViewModel model)

{

try

{

//conectar a la base de datos usando entity

using (MaestroDetalleEntities db = new MaestroDetalleEntities())

{

//creamos un objeto tipo Ventas para cargar los datos del modelo

Venta venta = new Venta();

venta.Fecha = DateTime.Now;

venta.Cliente = model.Cliente;

//agregamos el objeto ventas a la base de datos y guardamos

db.Venta.Add(venta);

//al momento de guardar se le genera a ventas un id

db.SaveChanges();

//VentaViewModel posee un campo tipo lista de Conceptos

//Agregar la lista de Conceptos a la tabla concepto de la base de datos

//debemos reccorrer model.Conceptos que es de tipo VentaViewModel

foreach (var ConceptoModel in model.Conceptos)

{

//creamos un objeto para llenar los campos que vienen en el model.Conceptos y los campos que se calcular automaticamente

Concepto concepto = new Concepto();

concepto.Cantidad = ConceptoModel.Cantidad;

concepto.Nombre = ConceptoModel.Nombre;

concepto.PrecioUnitario = ConceptoModel.PrecioUnitario;

concepto.Total = ConceptoModel.Cantidad \* ConceptoModel.PrecioUnitario;

//son los id que estan relacionados

//se le asigna el mismo venta.Id al listado de conceptos

concepto.IdVenta = venta.Id;

//agregamos el prinmer objeto a la tabla Concepto de la db

db.Concepto.Add(concepto);

}

//gurdamos los cambios en la db

db.SaveChanges();

}

ViewBag.Mensaje = "Registro Insertado en la DB";

//como tenemos otra vista que se llama Add nos retorna a ella misma sin necesidad de indicar su nombre

return View();

}

catch(Exception ex)

{

return View(model);

}

}

}

}

**VISTA CON SCRIPT JS:**

@model MaestroDetalle.Models.ViewModels.VentaViewModel

@{

ViewBag.Title = "Maestro Detalle";

}

<h2>@ViewBag.Title</h2>

<div style="color: forestgreen">

@ViewBag.Mensaje

</div>

<!--usamos el helper para el formulario-->

<!--le indicamos 3 parametris: "Metodo", "Controlador", FormMethod.Post-->

@using (Html.BeginForm("Add", "MaestroDetalle", FormMethod.Post))

{

<div class="row">

<h3>Maestro:</h3>

<div class="form-group">

@Html.LabelFor(v => v.Cliente)

@Html.TextBoxFor(v => v.Cliente, new { @class = "form-control" })

</div>

<hr />

</div>

<div class="row">

<!--input que agregan los valores a los campos-->

<div class="form-group">

<label for="Cantidad">Cantidad</label>

<input type="number" class="form-control" id="Cantidad" placeholder="Ingres una Cantidad">

</div>

<div class="form-group">

<label for="Nombre">Nombre</label>

<input type="text" class="form-control" id="Nombre" placeholder="Ingres un Nombre">

</div>

<div class="form-group">

<label for="Precio">Precio</label>

<input type="number" class="form-control" id="Precio" placeholder="Ingres un Precio">

</div>

<!--return false, para que no realice el submit-->

<!--solo inicia el llenado de la lista Conceptos-->

<input type="button" value="Agregar" class="btn btn-success" onclick="AgregaConcepto(); return false" />

<h3>Detalle:</h3>

<!--el datalle va a llenar una tabla de forma automatica,

usaremos javaScript para que se llene automaticamente.

-->

<div id="divConceptos">

<table class="table" id="tablaConceptos" style="width:75%">

<tr>

<th>Cantidad</th>

<th>Nombre</th>

<th>Precio Unitario</th>

<th>Total</th>

</tr>

</table>

<!--los tr y td con los datos los insertamos con el script de js-->

</div>

<!--en via los datos del formulario al controlador-->

<div class="form-group">

<input type="submit" value="Gurdar Registros" class="btn btn-primary" />

</div>

</div>

<!--Funcion JS que agrega conceptos a la tabla cuando se haga clic en el boton Agregar-->

<script>

//variable globla para indicar el indiece los input hidden

var num = 0;

function AgregaConcepto() {

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* LLENAR LA TABLA \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//input principales:

//obtener los valores de los input:

let Cantidad = document.getElementById("Cantidad").value;

let Nombre = document.getElementById("Nombre").value;

let Precio = document.getElementById("Precio").value;

//Obtener la tabla tablaConceptos

//let Tabla = document.getElementById("tablaConceptos");

//[0] obtenemos el primer tbody

let Tbody = document.getElementsByTagName("tbody")[0];

//Crear en memoria los tr de la tabla

let TR = document.createElement("tr");

//Crear en memoria los td de la tabla

let TDCantidad = document.createElement("td");

let TDNombre = document.createElement("td");

let TDPrecio = document.createElement("td");

let TDTotal = document.createElement("td");

//Agregar los valores a los td, usamos las let de los valores de los input

TDCantidad.innerHTML = Cantidad;

TDNombre.innerHTML = Nombre;

TDPrecio.innerHTML = Precio;

//Total lo vamos a calcular en el back end para evitar errores, lo usaremos aqui de forma ilustrativa

TDTotal.innerHTML = parseFloat(Cantidad) \* parseFloat(Precio);

//Agregar al tr los td

TR.appendChild(TDCantidad);

TR.appendChild(TDNombre);

TR.appendChild(TDPrecio);

TR.appendChild(TDTotal);

//Agregar al tbody de la tabla los TR ya contiene a los td

Tbody.appendChild(TR);

//Agregar a la Tabla los TR ya contiene a los td

// Tabla.appendChild(Tbody);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ENVIAR LOS DATOS AL CONTROLADOR \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//el div que contiene a la tabla

let DivConceptos = document.getElementById("divConceptos");

//Creamos los inputs y los cargamos en una varible, estos input son para que se haga mach con el modelo

//para enviaran los datos al controlador que de momento solo estan cargados en la tabla

//los ocultamos hidden para que nos e vean en la vista

let HiddenIndex = document.createElement("input"); // HiddenIndex sirve para que MVC sepa que numero de indice se esta utilizado

let HiddenCantidad = document.createElement("input");

let HiddenNombre = document.createElement("input");

let HiddenPrecio = document.createElement("input");

//Agregamos las propiedades al input HiddenIndex:

//Conceptos debe ser exactemente el nombre que tiene el campo en el modelo: VentaViewModel Conceptos { get; set; }

//HiddenIndex sirve para que el controlador sepa que numero de indice se esta utilizado

//ya que Conceptos es una lista que se va a ir llenando

//INICIA A LLENAR LA LISTA EN LA POSICION 0:

HiddenIndex.name = "Conceptos.Index"

HiddenIndex.value = num; //num inicia en 0 es la varaible global

HiddenIndex.type = "hidden";

//Agregamos las propiedades del input principal Cantidad al input HiddenCantidad:

//Inicia En la posicion Conceptos[0].Cantidad

HiddenCantidad.name = "Conceptos[" + num + "].Cantidad";

HiddenCantidad.value = Cantidad; //son los valores de los input que estamos obteniendo en las variables del principio

HiddenCantidad.type = "hidden";

//agregamos las propiedades del input principal Nombre al input HiddenNombre

HiddenNombre.name = "Conceptos[" + num + "].Nombre";

HiddenNombre.value = Nombre;

HiddenNombre.type = "hidden";

//agregamos las propiedades de input principal Precio al input HiddenPrecio

HiddenPrecio.name = "Conceptos[" + num + "].PrecioUnitario"; //los nombres deben ser como estan en el modelo: VentaViewModel

HiddenPrecio.value = Precio;

HiddenPrecio.type = "hidden";

//agregamos al DivConceptos los input Hidden

//aunque no se mostraran ya que la tabla los estara mostrando, si no se verian dos veces

//los input Hidden nos sirven unicamente para enviar los datos al controlador y llenar la lista Conceptos

DivConceptos.appendChild(HiddenIndex);

DivConceptos.appendChild(HiddenCantidad);

DivConceptos.appendChild(HiddenNombre);

DivConceptos.appendChild(HiddenPrecio);

//Borrar los valores de los input

document.getElementById("Cantidad").value = "";

document.getElementById("Nombre").value = "";

document.getElementById("Precio").value = "";

//aumentar la variable global, usada comno indice para llenar la lista

num++;

}

</script>

}

**CORREMOS EL PROYECTO:**

Agregamos datos:

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Guardamos:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

**Verificamos la DB**

Vemos que se insertaron los registros:

En ventas se inserto el id 40

Texto, Tabla

Descripción generada automáticamente con confianza media

**En la tabla concepto**

vemos los registros asignados al id 40:

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente