

Prueba Cristian Arroyo

## Desarrollo del Proyecto

Instalamos el paquete nuget de entity framework para facilitarnos la comunicación con la base de datos



## Creación de tablas en base de datos

Se genera el script con el diseño de la base de datos para la generación de las cartas

```
SQLQuery3.sql - L...LAP-CBARROYO (53))* X
/*SCRIPT PRUEBA CRISTIAN ARROYO
Generacion de la base de datos*/
CREATE DATABASE PruebaCristian
USE PruebaCristian

/* Generacion de tabla de cartas para el guardado de las mismas*/
CREATE TABLE Cartas (
  Id INT IDENTITY(1,1) PRIMARY KEY,
  Nombre VARCHAR (50) NOT NULL,
  Imagen VARBINARY(MAX) NOT NULL,
  IdTabla INT
)

/*Generacion de tabla que contendra las 16 cartas al azar*/
CREATE TABLE Tabla (
  Id INT IDENTITY(1,1) PRIMARY KEY,
  IdImagen INT
)
```

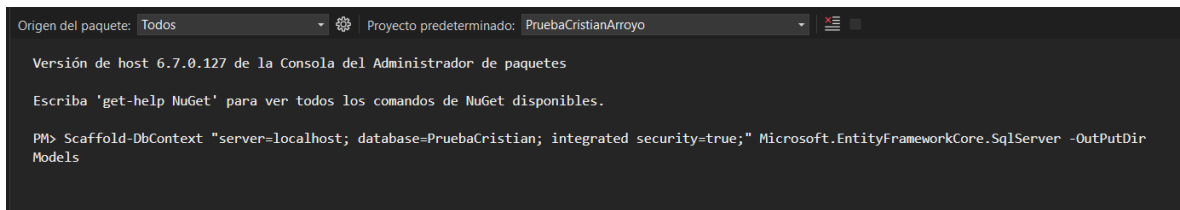
100 %

Messages

Commands completed successfully.

Completion time: 2023-11-02T12:02:57.0610891-06:00

Realizamos un Scaffold para la conexión y modelado de nuestra base de datos ya creada y utilizando los paquetes instalados previamente



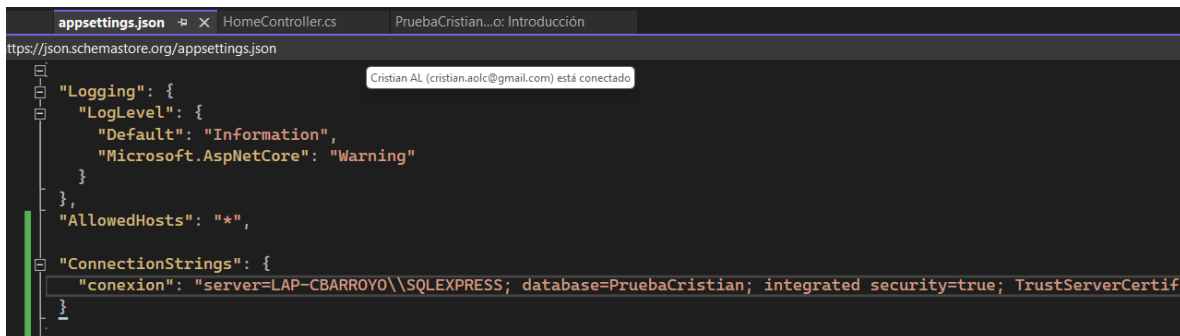
```
Origen del paquete: Todos Proyecto predeterminado: PruebaCristianArroyo

Versión de host 6.7.0.127 de la Consola del Administrador de paquetes

Escriba 'get-help NuGet' para ver todos los comandos de NuGet disponibles.

PM> Scaffold-DbContext "server=localhost; database=PruebaCristian; integrated security=true;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models
```

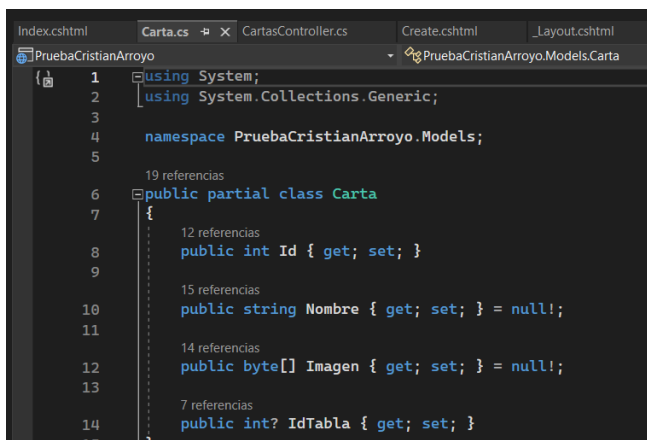
Configuramos nuestra cadena de conexión hacia la base de datos



```
appsettings.json HomeController.cs PruebaCristian...o: Introducción
https://json.schemastore.org/appsettings.json

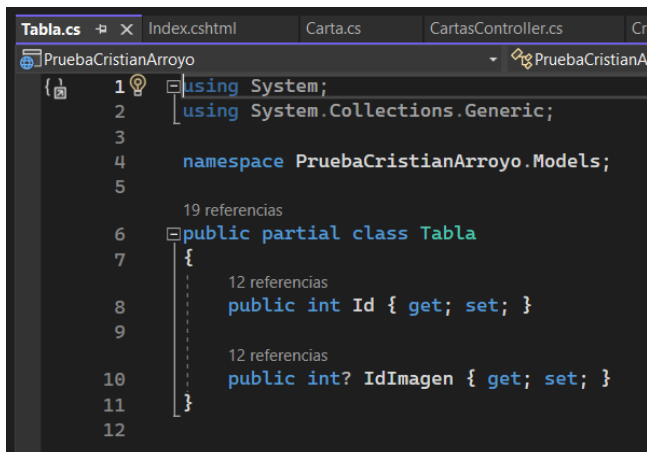
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "conexion": "server=LAP-CBARROYO\\SQLEXPRESS; database=PruebaCristian; integrated security=true; TrustServerCertificate=..."
  }
}
```

Creamos nuestros modelos para los controladores y la información a guardar



```
Index.cshtml Carta.cs CartasController.cs Create.cshtml _Layout.cshtml
PruebaCristianArroyo PruebaCristianArroyo.Models.Carta

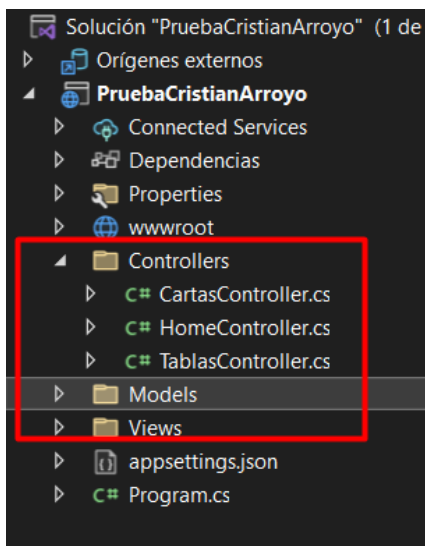
1 using System;
2 using System.Collections.Generic;
3
4 namespace PruebaCristianArroyo.Models;
5
6 public partial class Carta
7 {
8     public int Id { get; set; }
9
10    public string Nombre { get; set; } = null!;
11
12    public byte[] Imagen { get; set; } = null!;
13
14    public int? IdTabla { get; set; }
15 }
```



```
Tabla.cs Index.cshtml Carta.cs CartasController.cs Cr...
PruebaCristianArroyo PruebaCristianArroyo.Models.Tabla

1 using System;
2 using System.Collections.Generic;
3
4 namespace PruebaCristianArroyo.Models;
5
6 public partial class Tabla
7 {
8     public int Id { get; set; }
9
10    public int? IdImagen { get; set; }
11 }
12
```

## Estructura de Proyecto



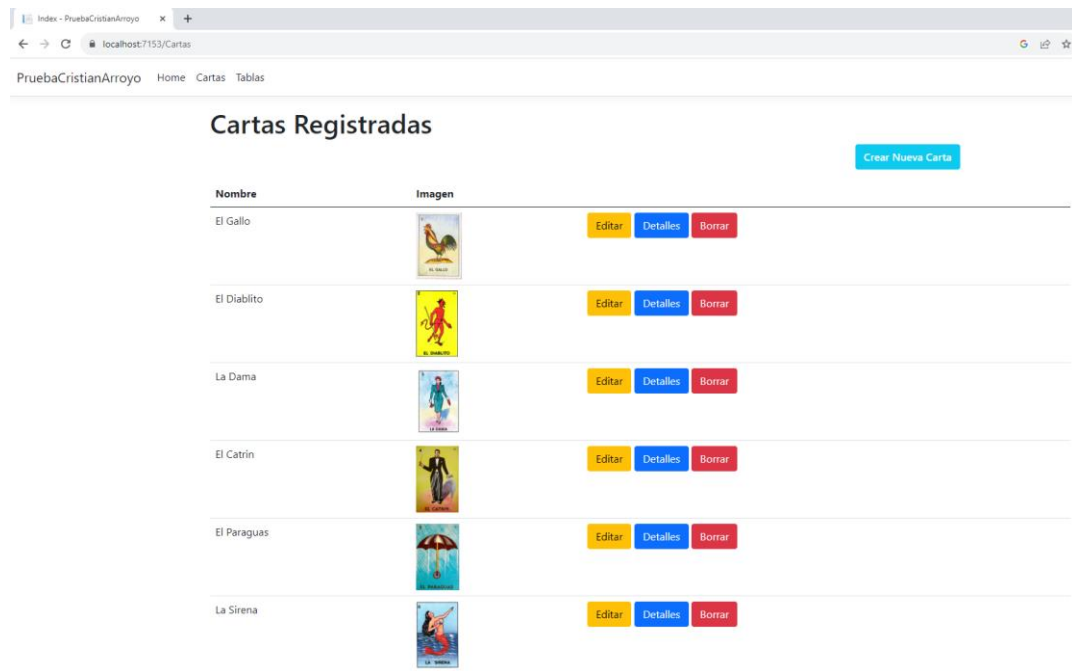
## Generación de Vistas

### Vista General de Cartas

### Pantalla Principal



## Pantalla de Cartas



## Código Fuente MVC

### Controlador

```
0 referencias
public async Task<IActionResult> Create([Bind("Id,Nombre,Imagen,IdTabla")] Carta carta,List<IFormFile> Imagen)
{
    foreach (var item in Imagen)
    {
        if(item.Length>0)
        {
            using (var stream = new MemoryStream())
            {
                await item.CopyToAsync(stream);
                carta.Imagen = stream.ToArray();
            }
        }
    }

    if (carta.Imagen != null && carta.Nombre != null && carta.Nombre != "")
    {
        _context.Add(carta);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(carta);
}

// GET: Cartas/Edit/5
0 referencias
public async Task<IActionResult> Edit(int? id)
{
    if (id == null || _context.Cartas == null)
    {
        return NotFound();
    }

    var carta = await _context.Cartas.FindAsync(id);
    if (carta == null)
    {
        return NotFound();
    }
    return View(carta);
}
```

## Vista

```
<h1>Cartas Registradas</h1>

<p>
  <a asp-action="Create">Crear Nueva Carta</a>
</p>

<table class="table">
  <thead>
    <tr>
      <th>
        @Html.DisplayNameFor(model => model.Nombre)
      </th>
      <th>
        @Html.DisplayNameFor(model => model.Imagen)
      </th>
      <th></th>
    </tr>
  </thead>
  <tbody>
    @foreach (var item in Model) {
      <tr>
        <td>
          @Html.DisplayFor(modelItem => item.Nombre)
        </td>
        <td>
          @{
            var base64 = Convert.ToBase64String(item.Imagen);
            var imgsrc = string.Format("data:image/gif;base64,{0}", base64);
          }
          
        </td>
        <td>
          <a asp-action="Edit" asp-route-id="@item.Id">Editar</a> |
          <a asp-action="Details" asp-route-id="@item.Id">Detalles</a> |
          <a asp-action="Delete" asp-route-id="@item.Id">Borrar</a>
        </td>
      </tr>
    }
  </tbody>
</table>
```

## Creación de Carta

Create - PruebaCristianArroyo x +

localhost:7153/Cartas/Create

PruebaCristianArroyo Home Cartas Tablas

### Crear Carta

Nombre

Imagen

Seleccionar archivo Ninguno archivo selec.

Guardar Regresar a la lista

## Codigo fuente

```
@{
    ViewData["Title"] = "Create";
}

<h1>Crear Carta</h1>

<hr />

<div class="row">
    <div class="col-md-4">
        <form asp-action="Create" method="post" enctype="multipart/form-data">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Nombre" class="control-label"></label>
                <input asp-for="Nombre" class="form-control" />
                <span asp-validation-for="Nombre" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Imagen" class="control-label"></label>
                <input type="file" asp-for="Imagen" class="form-control" />
                <span asp-validation-for="Imagen" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Guardar" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

## Editor de Carta

← → ↻ localhost:7153/Cartas/Edit/2

PruebaCristianArroyo Home Cartas Tablas

### Editar Carta

Nombre

Imagen

Ninguno archivo selec.

## Codigo Fuente

```
@model PruebaCristianArroyo.Models.Carta

@{
    ViewData["Title"] = "Edit";
}

<h1>Editar Carta</h1>

<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit" method="post" enctype="multipart/form-data">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="Id" />
            <div class="form-group">
                <label asp-for="Nombre" class="control-label"></label>
                <input asp-for="Nombre" class="form-control" />
                <span asp-validation-for="Nombre" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Imagen" class="control-label"></label>
                <input type="file" asp-for="Imagen" class="form-control" />
                <span asp-validation-for="Imagen" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Guardar" class="btn btn-primary" />
            </div>
        </form>
    </div>
    <div>
        <a asp-action="Index">Regresar a la lista</a>
    </div>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

## Detalles de Carta

PruebaCristianArroyo   Home   Cartas   Tablas

### Detalles de Carta

Nombre

El Diablito

Imagen



Editar

Regresar a la lista

## Codigo Fuente

```
@model PruebaCristianArroyo.Models.Carta

@{
    ViewData["Title"] = "Details";
}

<h1>Detalles de Carta</h1>

<div>
    <hr />
    <dl class="row">
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Nombre)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Nombre)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Imagen)
        </dt>
        <dd class="col-sm-10">
            @{
                var base64 = Convert.ToBase64String(Model.Imagen);
                var imgsrc = string.Format("data:image/gif;base64,{0}", base64);
            }
            
        </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model?.Id">Editar</a> |
    <a asp-action="Index">Regresar a la lista</a>
</div>
```

## Borrar Carta

PruebaCristianArroyo   Home   Cartas   Tablas

### Borrar Carta

¿Esta seguro de que desea borrar esta carta?

Carta

Nombre

La Dama

Imagen



Borrar

Regresar a la lista



## Generación de Tablas

PruebaCristianArroyo Home Cartas Tablas

Generar

### Muestra de generación de 6 tablas desde SQL

Id	IdCarta	NumCarta
429	2	2
430	16	2
431	14	2
432	13	2
433	15	2
434	4	2
435	7	2
436	5	2
437	10	2
438	9	2
439	8	2
440	11	2
441	17	2
442	12	2
443	2	3
444	11	3
445	8	3
446	3	3
447	18	3
448	14	3
449	17	3
450	5	3
451	12	3
452	15	3
453	4	3
454	16	3
455	9	3
456	13	3
457	10	3
...	-	-

Id	IdCarta	NumCarta
459	18	4
460	5	4
461	2	4
462	11	4
463	12	4
464	17	4
465	9	4
466	3	4
467	15	4
468	13	4
469	8	4
470	14	4
471	4	4
472	10	4
473	7	4
474	6	4
475	4	5
476	16	5
477	11	5
478	12	5
479	10	5
480	14	5
481	2	5
482	9	5
483	5	5
484	7	5
485	17	5
486	18	5
487	3	5

Id	IdCarta	NumCarta
478	12	5
479	10	5
480	14	5
481	2	5
482	9	5
483	5	5
484	7	5
485	17	5
486	18	5
487	3	5
488	8	5
489	15	5
490	13	5
491	13	6
492	18	6
493	4	6
494	15	6
495	7	6
496	11	6
497	5	6
498	17	6
499	3	6
500	16	6
501	10	6
502	9	6
503	2	6
504	12	6
505	14	6
506	6	6

En esta pantalla se muestra el id de la carta y en que numero de tabla se estaría generando, por ejemplo la carta con id = 2 que sería “El gallo” estaría en la carta 2, cumpliendo las 16 cartas por tablero

## Código de generación de Tabla

```
int TablasUsuario = Convert.ToInt32(usr);

//Leemos las cartas guardadas en la base de datos
List<Carta> cartas = _context.Cartas.ToList();
//Instanciamos el metodo random
Random rand = new Random();

//Generacion de Tablas dependiendo del usuario
Tabla tablaList = new Tabla();
var rows = from o in _context.Tablas
select o;
//Limpiamos las tablas para generarlas dependiendo el usuario
foreach (var row in rows)
{
    _context.Tablas.Remove(row);
}
_context.SaveChanges();

//ciclo para generar las thalas dependiendo el usuario
for (int tablaR = 1; tablaR <= TablasUsuario; tablaR++)
{
    //Creamos una nueva lista con el metodo random
    var CartasRandom = cartas.OrderBy(_ => rand.Next()).ToList();

    //se guardan dependiendo el numero de carta
    for (int numCar = 0; numCar < 16; numCar++)
    {
        tablaList = new Tabla();
        tablaList.NumCarta = tablaR;
        tablaList.IdCarta = CartasRandom[numCar].Id;
        _context.Add(tablaList);
        _context.SaveChanges();
    }
}
```

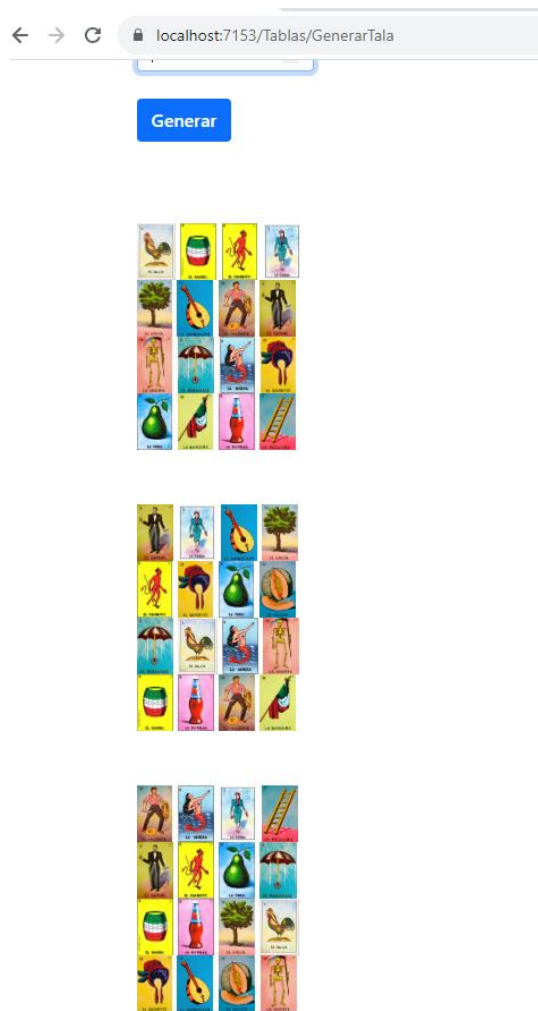
```
@{
    List<PruebaCristianArroyo.Models.ViewTabla> lista = ViewBag.Tabla;
    ViewData["Title"] = "Index";
}

<div class="row">
    <div class="col-md-3 form-horizontal">
        <form asp-action="GenerarTala" method="post">
            <br />
            <div>
                <input type="number" class="form-control" name="usr" placeholder="Numero de Tablas" required autofocus />
            </div>
            <button class="btn btn-primary" type="submit" style="color: white; font-weight: 600;">
                Generar
            </button>
        </form>
        @if (lista != null)
        {
            @for (int i = 0; i < lista.Count; i++)
            {
                if (i % 16 == 0)
                {
                    @:<br> <br> <br>
                    @:<table>
                    @:</table>
                }

                if (i % 4 == 0)
                {
                    @:<tr>
                    @:</tr>
                }

                <td>
                    @{
                        var base64 = Convert.ToBase64String(lista[i].Imagen);
                        var imgsrc = string.Format("data:image/gif;base64,{0}", base64);
                    }
                    
                </td>
            }
        }
    }
</div>
```

## Vista



Como se puede apreciar, se genera una vista con las tablas contenidas por 16 cartas seleccionadas sin que se repitan y generando tablas diferentes para el usuario.

## CONCLUSION

El diseño de esta prueba se puede mejorar en cuanto a estilos dependiendo del objetivo del proyecto, también para completar la funcionalidad y experiencia total del juego como en la vida real. Se pueden añadir Stored Procedures para realizar la funcionalidad original el juego y así completar el mismo.