UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA

UNAN – LEÓN

FACULTAD DE CIENCIAS Y TECNOLOGÍA

INGENIERÍA EN SISTEMAS DE LA INFORMACIÓN

AÑO LECTIVO: 2025

SEMESTRE: II

Componente Curricular: # PROGRAMACION ORIENTADA A LA WEB II

Grupo: 1

Sub-Grupo:

Profesor(a): JUAN CARLOS LEYTON BRIONES

Autor:

Luis Ignacio Hidalgo Molina.

León, Nicaragua, 2025.

“¡A la Libertad por la Universidad!”

PRACTICA 2

CONTROLADOR GX, MODELOS, VISTAS Y JSON

Siendo GX = GPU

Controlador GX

El controlador **GxController** pertenece a la capa de controladores de ASP.NET Core MVC y se encarga de **gestionar información de GPUs** a partir de un archivo JSON. Permite listar todas las GPUs, buscar por modelo y mostrar detalles individuales.

Campos y Constructor

```
public List<Gx> lstgx = null;

public GxController()
{
    var myJsonString = System.IO.File.ReadAllText("Models/gx.json");

    // Deserializar al contenedor
    GxContainer hardware =
JsonConvert.DeserializeObject<GxContainer>(myJsonString);

    // Obtener la lista de GPUs
    lstgx = hardware.GPUs;
}
```

Descripción:

- lstgx: Lista pública que contiene todos los objetos Gx (GPUs).

Constructor:

- Lee el archivo JSON ubicado en Models/gx.json.
- Deserializa el JSON a un objeto GxContainer.
- Obtiene la lista de GPUs y la asigna a lstgx.

- Esto asegura que la lista de GPUs esté disponible para todos los métodos del controlador.

Métodos del controlador

1. Index()

```
public IActionResult Index()
{
   return View(lstgx);
}
```

Descripción:

- No recibe parámetros.
- Retorna la vista por defecto Index.cshtml, pasando la lista completa de GPUs (lstgx).
- Sirve como página principal para mostrar todas las GPUs.

Ejemplo de uso:

- URL → /Gx/Index
- Muestra todas las GPUs del JSON.

2. Find(string Gx)

```
public IActionResult Find(string Gx)
{
    List<Gx> lstResultGx = new List<Gx>();
    foreach (var item in lstgx)
    {
        if (item.Modelo.ToLower().Contains(Gx.ToLower()))
        {
            lstResultGx.Add(item);
        }
    }

    return View("Index", lstResultGx);
}
```

Descripción:

- Recibe un parámetro de búsqueda (Gx), que es una cadena con el modelo de GPU.
- Recorre lstgx y agrega a lstResultGx todos los elementos cuyo modelo contenga el texto buscado (sin distinguir mayúsculas/minúsculas).
- Retorna la vista Index mostrando solo los resultados filtrados.

Ejemplo de uso:

- **Entrada:** Find("RTX") → Retorna todas las GPUs cuyo modelo contenga "RTX".

3. Details(int id)

```
public IActionResult Details(int id)
{
    var item = lstgx.FirstOrDefault(x => x.Id == id);
    if (item == null)
    {
        return NotFound();
    }
    return View(item);
}
```

Descripción:

- Recibe un ID de GPU.
- Busca en la lista lstgx la GPU con ese ID.
- Si no existe → retorna un 404 NotFound.
- Si existe → retorna la vista Details.cshtml mostrando los detalles de la GPU.

Ejemplo de uso:

- **Entrada:** Details(3) → Muestra los detalles de la GPU con ID 3.
- **Entrada:** Details(99) → Retorna error 404 si no existe.

MODELOS

Clase GX y GXContainer

Estos modelos representan tarjetas gráficas (GPUs) y un contenedor para listas de GPUs, usados en GxController.

1.Clase Gx

```
public class Gx
{
    public int Id { get; set; }
    public string? Marca { get; set; }
    public string? Modelo { get; set; }
    public string? VRAM { get; set; }
    public int NucleosCuda { get; set; }
    public bool RayTracing { get; set; }
    public string? Imagen { get; set; }
}
```

Descripción de propiedades:

Propiedad	Tipo	Descripción
Id	int	Identificador único de la GPU. Se usa para buscar detalles específicos.
Marca	string?	Marca de la GPU (ej. NVIDIA, AMD). Puede ser nula si no se especifica.
Modelo	string?	Modelo de la GPU (ej. RTX 3080, RX 6800). Puede ser nulo.
VRAM	string?	Cantidad de memoria de video (ej. "8GB", "16GB").
NucleosCuda	int	Número de núcleos CUDA (para GPUs NVIDIA). Para GPUs AMD puede usarse como referencia de procesamiento paralelo.
RayTracing	bool	Indica si la GPU soporta Ray Tracing. true → sí, false → no.
Imagen	string?	Ruta o URL de la imagen representativa de la GPU.

Notas:

- El signo ? indica que la propiedad puede ser nula (nullable).
- Id se utiliza en el controlador para buscar la GPU específica (Details(int id)).

2. Clase GxContainer

```
public class GxContainer
{
    public List<Gx> GPUs { get; set; } = new List<Gx>();
}
```

Descripción:

- Contenedor que agrupa una lista de objetos Gx.
- Propiedad GPUs:
- Tipo: List<Gx>
- Se inicializa vacía por defecto (= new List<Gx>()).
- Se utiliza principalmente al deserializar JSON, para obtener todas las GPUs del archivo gx.json.

Ejemplo de uso:

```
var myJsonString = System.IO.File.ReadAllText("Models/gx.json");
GxContainer container = JsonConvert.DeserializeObject<GxContainer>(myJsonString);
List<Gx> listaGPUs = container.GPUs;
```

Relación con el controlador GxController:

- GxController lee el JSON y deserializa a GxContainer.
- La lista lstgx se llena con container.GPUs.
- Los métodos Index, Find y Details utilizan los objetos Gx para mostrar información en la vista.

VISTAS

1.Index.cshtml

La vista Index.cshtml está diseñada para mostrar una lista de GPUs y permitir búsquedas por modelo. Se basa en el modelo List<Gx> de la aplicación.

Declaración del modelo

@model List<Proyecto1.Models.Gx>

Esta vista recibe una lista de objetos Gx desde el controlador (lstgx).

Permite acceder a todas las propiedades de cada GPU para renderizar la información en la interfaz.

Formulario de búsqueda:

```
<form class="form-inline" method="POST" action="/Gx/Find">
   <div class="form-group mx-sm-3 mb-2">
      <label class="mr-3">Buscar Grafica</label>
      <input type="text" class="form-qontrol" required name="gx">
   </div>
   <button type="submit" class="btn btn-primary mb-2">Buscar</button>
</form>
```

Descripción:

- Formulario para buscar GPUs por modelo.
- **method="POST"** → envía los datos al método Find(string Gx) del controlador.
- **action="/Gx/Find"** → define la ruta donde se procesa la búsqueda.
- **<input>:** campo obligatorio (required) donde el usuario escribe el modelo a buscar.
- **<button>:** envía el formulario al controlador.

Relación con el controlador:

Cuando el usuario envía el formulario, GxController.Find filtra las GPUs y devuelve la vista Index con los resultados.

Renderizado de la lista de GPUs:

```
<div class="container">
   <div class="row">
      @foreach (var item in Model)
      {
         <div class="col-md-3">
            <div class="card mb-3">
               <img src="@item.Imagen" class="card-img-top" alt="....." />
               <div class="card-body">
                  <h5 class="card-title">@item.Modelo</h5>
                  <p class="card-text">Marca: @item.Marca</p>
                  <p class="card-text"><small class="text-muted">VRAM:
@item.VRAM</small></p>
                  <a href="@Url.Action("Details", "Gx", new { id = item.Id })" class="btn
btn-primary">Details</a>
               </div>
            </div>
         </div>
      }
   </div>
</div>
```

Descripción de elementos:

Elemento	Función
	Contenedor principal de la grilla de GPUs.
	Fila que organiza las tarjetas (cards) de las GPUs.
@foreach (var item in Model)	Recorre la lista de GPUs y genera una tarjeta por cada elemento.
	Cada tarjeta ocupa un cuarto de la fila (4 columnas).

Elemento	Función
	Tarjeta que contiene la información de la GPU.
<img src="@item.Imagen">	Muestra la imagen de la GPU.
<h5 class="card-title">@item.Modelo</h5>	Muestra el modelo de la GPU.
Marca: @item.Marca	Muestra la marca de la GPU.
<small class="text-muted">VRAM: @item.VRAM</small>	Muestra la cantidad de VRAM.
Details	Enlace para ver los detalles individuales de la GPU.

Relación con el controlador:

- Cada tarjeta se genera a partir de los objetos Gx pasados desde Index() o Find().
- El enlace "Details" llama al método GxController.Details(int id) para mostrar información completa de la GPU seleccionada.

JSON

Ejemplo de elemento del arreglo GPUs:


```
{
  "Id": 1,
  "Marca": "NVIDIA",
  "Modelo": "GeForce RTX 4090",
  "VRAM": "24 GB GDDR6X",
  "NucleosCuda": 16384,
  "RayTracing": true,
  "Imagen": "https://m.media-amazon.com/images/I/51c1zFDNVmL._AC_SL1003_.jpg"
}
```

Explicación de cada propiedad:

Propiedad	Tipo	Descripción
Id	int	Identificador único de la GPU. Se utiliza para buscar o mostrar detalles específicos.
Marca	string	Marca del fabricante de la GPU (ej. NVIDIA, AMD).
Modelo	string	Nombre del modelo de la GPU (ej. GeForce RTX 4090).
VRAM	string	Cantidad y tipo de memoria de video (ej. "24 GB GDDR6X").
NucleosCuda	int	Número de núcleos CUDA para GPUs NVIDIA (o equivalente de procesamiento paralelo).
RayTracing	bool	Indica si la GPU soporta Ray Tracing. true → soporta, false → no soporta.
Imagen	string	URL o ruta de la imagen representativa de la GPU. Se usa para mostrar la tarjeta visualmente en la vista.

Relación con la aplicación:

- Este JSON se deserializa en objetos Gx dentro de GxContainer.
- Cada propiedad se usa en la vista Index.cshtml y Details.cshtml:
- Modelo, Marca, VRAM → se muestran como texto.
- Imagen → se muestra como imagen de la tarjeta.
- Id → se usa para identificar y enlazar al método Details(id).
- RayTracing y NucleosCuda → se muestran en detalles de la GPU.

API
CONTROLADOR GPU, MODELO Y VISTA

El controlador GpuController está diseñado para obtener información de GPUs a través de un servicio (GpuService) y mostrarla en la vista Index.cshtml.

Se basa en inyección de dependencias para consumir el servicio de manera asincrónica (async/await).

Campos y Constructor:

```
private readonly GpuService _gpuService;

public GpuController(GpuService gpuService)
{
    _gpuService = gpuService;
}
```

Descripción:

- _gpuService: instancia del servicio GpuService que se encarga de obtener la información de GPUs desde una API o fuente de datos externa.
- El constructor recibe GpuService a través de inyección de dependencias.
- Esto permite desacoplar la lógica de obtención de datos del controlador y facilita pruebas unitarias

Método Index():

```
public async Task<IActionResult> Index()
{
    var gpus = await _gpuService.GetGpusAsync();
    return View(gpus);
}
```

Descripción:

1. async Task<IActionResult>

    - El método es asíncrono para no bloquear el hilo mientras se obtiene la información de la API.

2. var gpus = await _gpuService.GetGpusAsync();

    - Llama al método GetGpusAsync() del servicio GpuService para obtener la lista de GPUs.

    - await asegura que se espere la respuesta de manera asincrónica.

    - return View(gpus);

3. Retorna la vista Index.cshtml pasando la lista de GPUs obtenida.

    - La vista puede usar la lista para mostrar tarjetas, detalles, o filtrado, similar al ejemplo de GxController.

Relación con el Servicio (GpuService)

El controlador no conoce cómo se obtienen los datos, solo llama al servicio.

GpuService podría:

- Llamar a una API REST externa.

- Leer desde una base de datos.

- Leer un archivo JSON local.

Esto mejora la separación de responsabilidades (SRP) y facilita mantenimiento.

MODELO

Modelo GPUinfo.cs

Este modelo representa una tarjeta gráfica con propiedades relevantes para mostrar información técnica y comercial. Se utiliza en la vista Index.cshtml y en los servicios que consumen APIs o fuentes de datos de GPUs.

```
namespace Proyecto.Models
{
    public class Gpu
    {
        public string? Model { get; set; }
        public string? Launch { get; set; }
        public string? CoreClockMHz { get; set; }
        public int MemorySizeGB { get; set; }
        public int TDPWatts { get; set; }
        public int ReleasePriceUSD { get; set; }
    }
}
```

Explicación de cada propiedad

Propiedad	Tipo	Descripción
Model	string?	Nombre o modelo de la GPU (ej. "GeForce RTX 4090"). Puede ser nulo si no se proporciona.
Launch	string?	Fecha de lanzamiento de la GPU (ej. "2022-10-12"). Puede ser nula.
CoreClockMHz	string?	Velocidad del reloj del núcleo en MHz (ej. "2235 MHz"). Se guarda como string para incluir valores con boost o rangos.
MemorySizeGB	int	Tamaño de memoria de la GPU en GB (ej. 24).
TDPWatts	int	Consumo energético típico de la GPU en Watts.
ReleasePriceUSD	int	Precio de lanzamiento en dólares estadounidenses.

Relación con el controlador y la vista

1. GpuController.Index() obtiene una lista de objetos Gpu desde GpuService.
2. La lista se pasa a la vista Index.cshtml para mostrar tarjetas de GPUs.
3. Cada propiedad se puede renderizar en la interfaz:
    - Model → nombre visible.
    - Launch → fecha de lanzamiento.

- CoreClockMHz → velocidad de reloj.
- MemorySizeGB, TDPWatts, ReleasePriceUSD → detalles técnicos y comerciales.

VISTA

index.cshtml

La vista Index.cshtml está diseñada para mostrar una lista de GPUs en forma de tabla. Recibe como modelo un Dictionary<string, Gpu>, donde la clave (Key) puede ser un identificador de GPU y el valor (Value) es el objeto Gpu.

Declaración del modelo

@model Dictionary<string, Proyecto.Models.Gpu>

La vista espera un diccionario con:

- Key → identificador de la GPU.
- Value → objeto Gpu que contiene los datos técnicos y comerciales.
- Permite iterar sobre cada elemento y mostrar la información en filas de tabla.

Título de la vista:

<h2>Lista de GPUs</h2>

Muestra un encabezado indicando que la tabla contiene todas las GPUs obtenidas de la API.

Tabla de GPUs

```
<table class="table">
   <thead>
      <tr>
         <th>ID</th>
         <th>Modelo</th>
         <th>Lanzamiento</th>
```

```
            <th>Core Clock (MHz)</th>
            <th>Memoria (GB)</th>
            <th>TDP (Watts)</th>
            <th>Precio (USD)</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var gpu in Model)
        {
            <tr>
                <td>@gpu.Key</td>
                <td>@gpu.Value.Model</td>
                <td>@gpu.Value.Launch</td>
                <td>@gpu.Value.CoreClockMHz</td>
                <td>@gpu.Value.MemorySizeGB</td>
                <td>@gpu.Value.TDPWatts</td>
                <td>@gpu.Value.ReleasePriceUSD</td>
            </tr>
        }
    </tbody>
</table>
```

Descripción de elementos:

Elemento	Función
<table class="table">	Crea una tabla con estilo Bootstrap.
<thead>	Define la cabecera de la tabla con los nombres de columnas.
<tbody>	Contiene las filas de la tabla generadas dinámicamente.
@foreach (var gpu in Model)	Itera sobre cada par Key/Value del diccionario para mostrar cada GPU.
@gpu.Key	Muestra el identificador de la GPU.
@gpu.Value.Model	Muestra el modelo de la GPU.
@gpu.Value.Launch	Muestra la fecha de lanzamiento.
@gpu.Value.CoreClockMHz	Muestra la velocidad del núcleo en MHz.
@gpu.Value.MemorySizeGB	Muestra la memoria en GB.
@gpu.Value.TDPWatts	Muestra el consumo energético en Watts.
@gpu.Value.ReleasePriceUSD	Muestra el precio de lanzamiento en USD.

Relación con el controlador y modelo:

1. GpuController.Index() llama a GpuService.GetGpusAsync() y obtiene un diccionario de GPUs.
2. La vista recibe ese diccionario como modelo.
3. Cada fila de la tabla representa un objeto Gpu, mostrando sus propiedades más importantes de manera clara**.**