

TAREA: CLASE 11

TAREAS CAPACITACIÓN C#

 **Ezequiel Remus**
ezequielremus@gmail.com

RESUMEN

En esta ocasión la tarea consiste en aprovechar lo que veníamos haciendo con la aplicación de stock y llevar esto a una aplicación web. Para esto utilizaremos **ASP . Net Core (M.V.C)** Los incisos estan reenumerados para hacer un acoplamiento segun los incisos remarcados en el word de la tarea, en particular, en la sección 6 se juntan las tareas desde la 8 a la 13

1. Crear un nuevo Proyecto web

1.1. Enunciado

Crear un nuevo proyecto web, con el nombre WebAppStock. El mismo debe ser del tipo Modelo – Vista Controlador

1.2. Solución

Lo que primero debemos hacer es crear el proyecto indicado. En este caso es un **ASP .Net Core (M.V.C)**

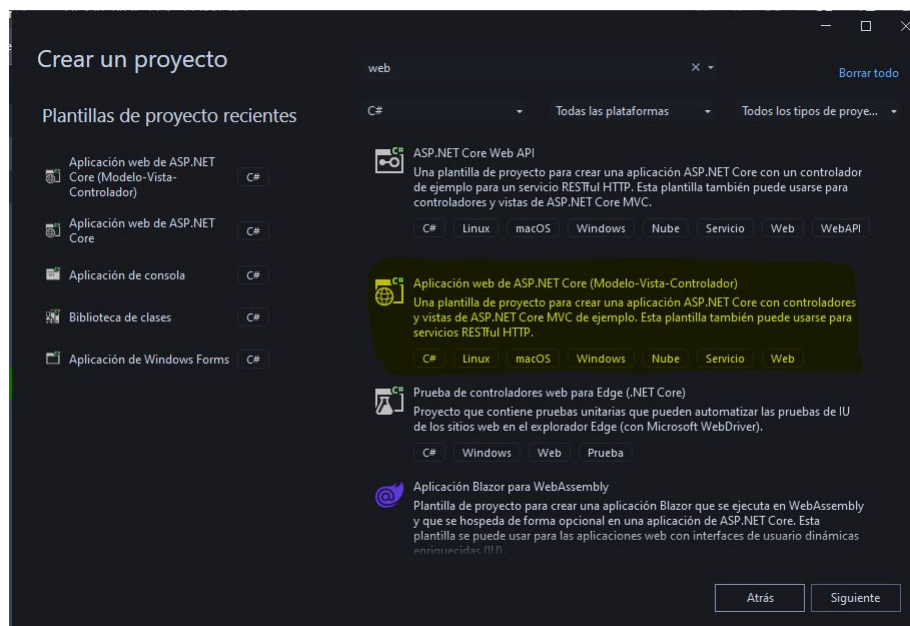


Figura 1: Creacion del Proyecto

Una vez abierto el proyecto vemos que nos crea el siguiente Árbol

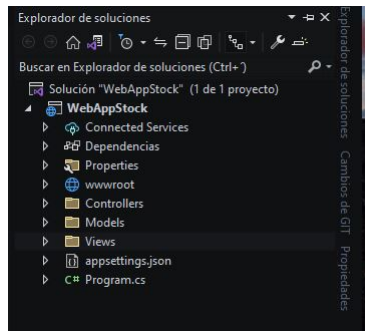


Figura 2: Árbol del Proyecto

En las dependencias del proyecto debemos asegurarnos de incertar el `CodigoComun.dll` y además instalar los paquetes nugget necesarios, en este caso son.



Figura 3: Paquetes de Entity

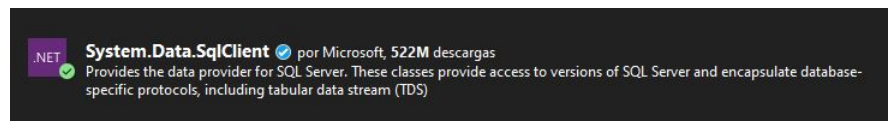


Figura 4: Paquete de SqlClient

2. Primer Prueba

2.1. Enunciado

Una vez creado ejecutar el mismo y verificar que funciones

2.2. Solución

Corremos el programa y verificamos que la pagina se muestra correctamente

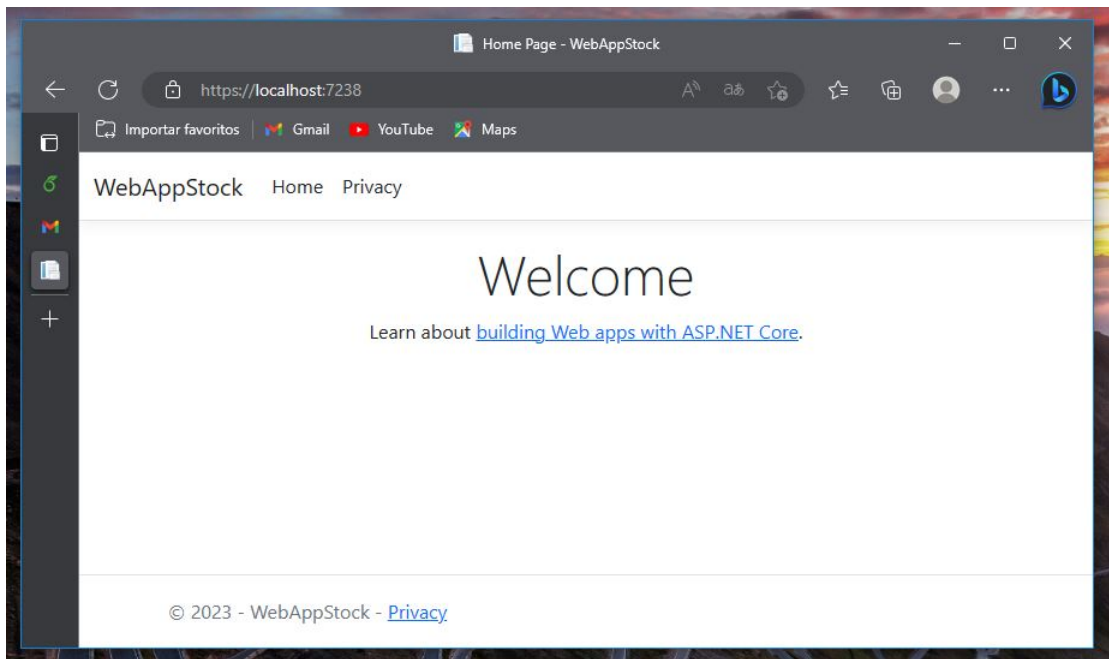


Figura 5: Primera Ejecución del programa

3. Creación de Controladores

3.1. Enunciado

Crear los controladores:

- 3.2 ArticulosController
- 3.2 DepositosController
- 3.2 StockController

3.2. Solución

Para crear los controladores¹ nos paramos en la carpeta **Controller**, hacemos clic derecho, vamos a *Agregar* y luego *Controlador*. Nos aparecera una ventana de como a que se ve en 6 Debemos darle en la opción de *Controlador Vacío*.

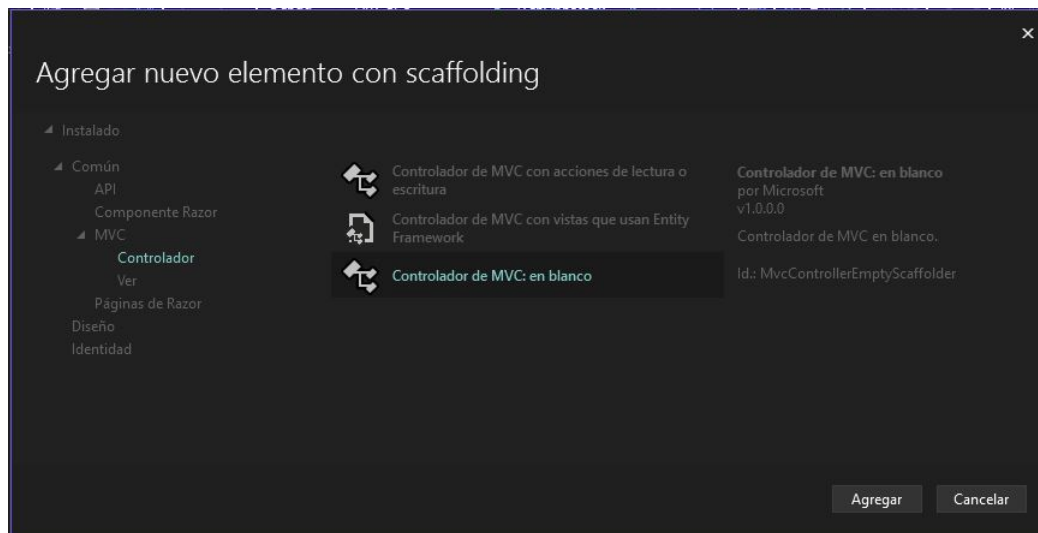


Figura 6: Agregamos un controlador

Debemos crear uno para cada requerimiento. El árbol nos quedara como sigue

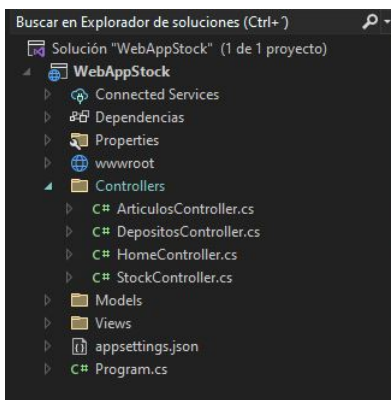


Figura 7: Árbol del Controller

Lo cual nos creara los archivos cuyo código se aprecia en las figuras 3.2, 3.2 y 3.2

¹Primero nos conviene verificar que no estemos corriendo el programa, ya que si lo este esta corriendo no nos permitirá acceder a la opción.

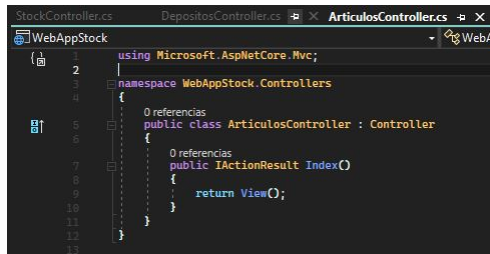


Figura 8: ArticulosController

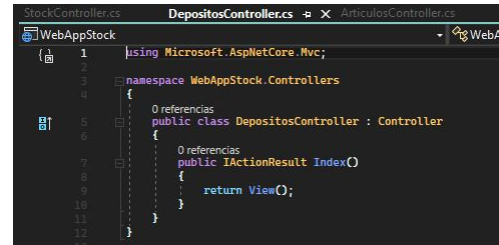


Figura 9: DepositosController

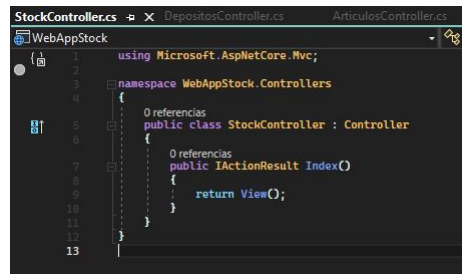


Figura 10: StockController

4. Agregamos las Vistas

4.1. Enunciado

En cada controlador, debemos tener un método index (lo genera por defecto) agregarle una vista que este relacionada con este método y adentro agregarle un header que la identifique. Por ejemplo para el index de Artículos

```

1 <h1>Soy el index de Articulo</h1>
2

```

Verificar que corriendo la aplicación entrando en /NombreControlador/Index se navegue a cada una de las views creadas. Mandar print de pantalla de cada una de las paginas en el navegador donde se vea la URL.

4.2. Solución

Primero lo que vamos a hacer es *¿*, dentro de la carpeta **Views** creamos las carpetas **Articulos**, **Depositos** y **Stock**.

En cada una de estas carpetas agregaremos una vista `index.cshtml`. Para crear una vista, debemos dar clic derecho en la carpeta donde queremos que se cree, vamos a *Agregar*, luego *Vista* y creamos una *Vista vacia* a la cual le daremos el nombre `index.cshtml`.

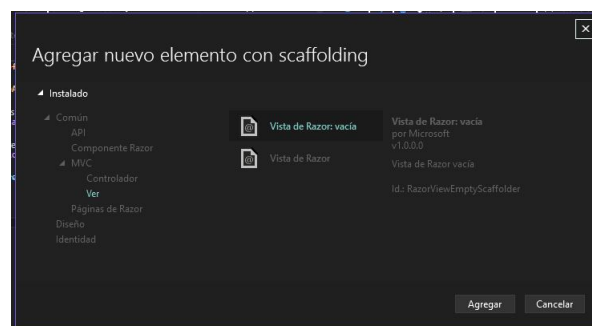


Figura 11: Agregamos una vista

Nos debería quedar el árbol de la figura 4.2 Luego, agregamos en cada vista creada una etiqueta

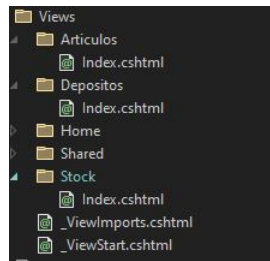


Figura 12: Árbol

```
1 <h1> ... </h1>  
2
```

con un mensaje adecuado.

Ejecutemos el programa y vayamos a la ruta

localhost:puerto/CarpetaVista/Archivo

de esta forma podremos ver las vistas de las figuras 4.2, 4.2 y 4.2

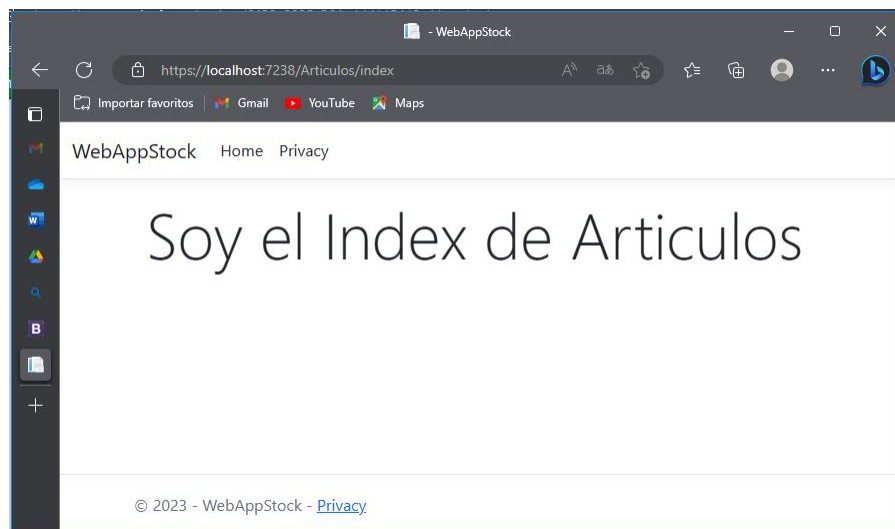


Figura 13: Vista del Articulos

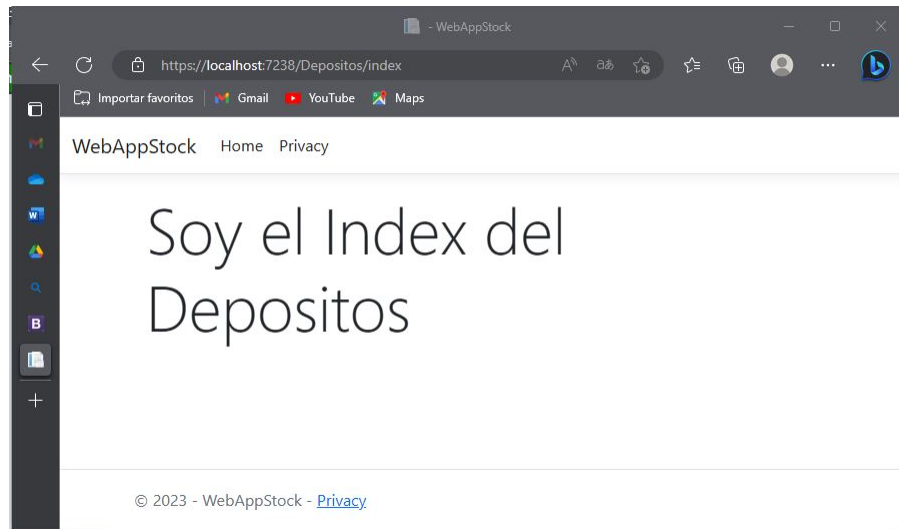


Figura 14: Vista del Depositos

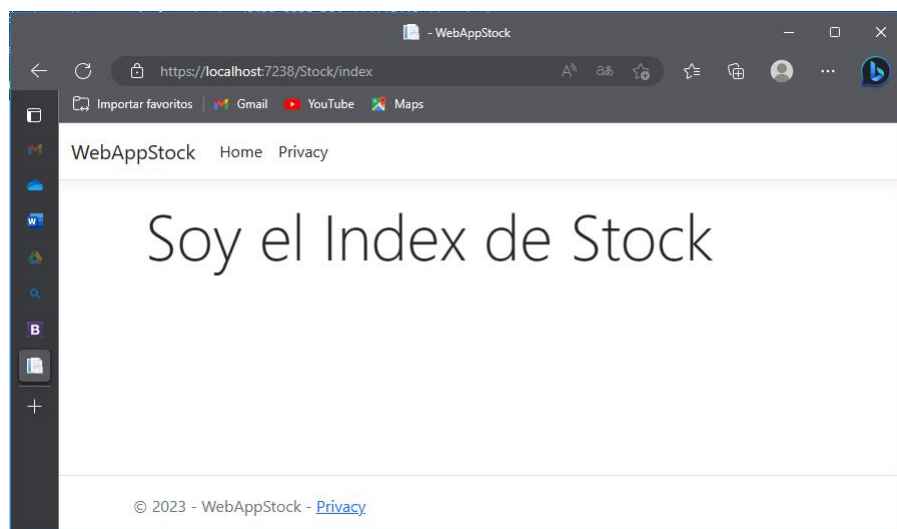


Figura 15: Vista del Stock

5. _Layout

5.1. Enunciado

Agregar en el archivo _layout dentro de la carpeta Shared de las View los accesos directos en el menú superior de la web. Debería quedar el menú con Artículos – Depositos – Stocks y haciendo click allí ingresar en el index de cada uno.

Mandar print donde se vea este menú en la web

5.2. Solución

En el archivo _Layout dentro de **Views/Shared** hay código que comparten todas las vistas. En este caso, nos presenta un header donde se encuentra el nav. Agregaremos las siguientes etiquetas al nav para que aparezcan los botones que nos manden a las vistas Cuando ejecutamos el programa vemos que el nav presenta estos cambios

```
<li class="nav-item">
  <a class="nav-link" asp-area="" asp-controller="Articulos" asp-action="Index">Articulos</a>
</li>
<li class="nav-item">
  <a class="nav-link" asp-area="" asp-controller="Depositos" asp-action="Index">Depositos</a>
</li>
<li class="nav-item">
  <a class="nav-link" asp-area="" asp-controller="Stock" asp-action="Index">Stock</a>
</li>
```

Figura 16: Etiquetas agregadas al Layout

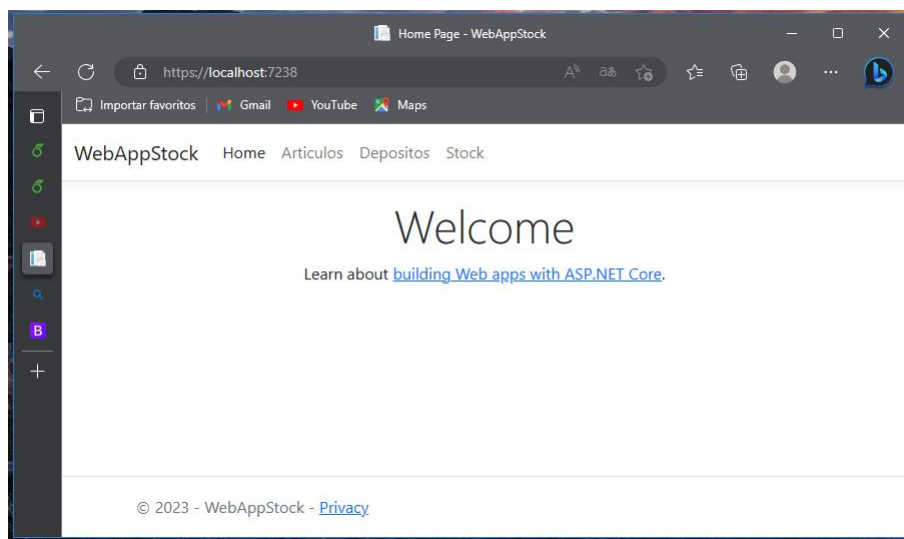


Figura 17: Nuevo nav

6. Trabajamos con DepositosController y su vista

6.1. Enunciado

En el DepositosController en el método Index Instanciar un DepositoServices y llamar al método **GetDepositoPorID()** pasandole un deposito que exista. Y Probar debuguando que se traiga los datos del Deposito. Acordarse de que tienen que agregar las referencias a los paquetes nuget de entity framework.

En la view Index asociada al DepositoController agregar que reciba como modelo, un Deposito, cambiar el método index del depositoController para que le pase el modelo del punto 8 a la Vista.

En la vista modificar que se muestren todos los datos del deposito que envia el controlador.

Enviar código del método Index del depositoController

Enviar PRint de la pantalla Index mostrando datos del deposito

6.2. Solución

Trabajamos con el método Index en DepositosController. Lo modificamos para que devuelva a la vista un deposito que este en la base de datos, en este caso el que tiene $Id = 1$

```

1  using CodiceComun.Negocio;
2  using CodiceComun.Entities;
3
4  using Microsoft.AspNetCore.Mvc;
5
6  namespace WebAppStock.Controllers
7  {
8      0 referencias
9      public class DepositosController : Controller
10     {
11         0 referencias
12         public IActionResult Index()
13         {
14             DepositoServices ds = new DepositoServices();
15             Deposito deposito = ds.GetDeposito(1);
16             return View(deposito);
17         }
18     }

```

Figura 18: Código del Index en DepositosController

Si corremos y hacemos el debug vemos que la variable toma los valores, lo cual es muy buena señal

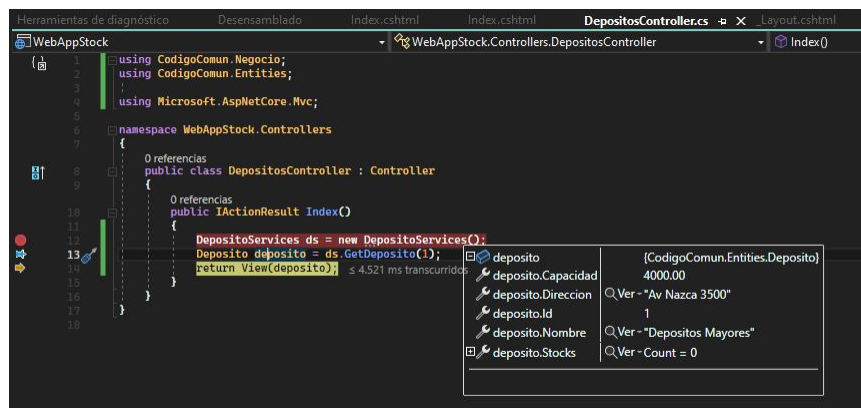


Figura 19: Código del Index en DepositosController con Debug

Luego, en el index correspondiente a la vista de Depositos modificamos el código para que se vean los datos del deposito pasado desde el controller

```

1  @*
2  For more information on enabling MVC for empty projects, visit https://go.microsoft
3
4  *@
5  using CodiceComun.Entities;
6  @model Deposito;
7
8  @{
9
10     <div class="container">
11         <h1 class="display-1 text-dark"> Soy el Index del Depositos </h1>
12         <br/><br/>
13         <div class="row">
14             <div class="col">
15                 <div class="card" style="width: 18rem;">
16                     <div class="card-header">
17                         @Model.Nombre
18                     </div>
19                     <ul class="list-group list-group-flush">
20                         <li class="list-group-item">Dirección: @Model.Direccion</li>
21                         <li class="list-group-item">Capacidad: @Model.Capacidad</li>
22                     </ul>
23                 </div>
24             </div>
25         </div>
26     </div>

```

Figura 20: Código del Index en DepositosController

Corremos la aplicación y verificamos que desde el **nav** nos mande a la URL del Index de Depositos y que la view nos muestre lo que escribimos.

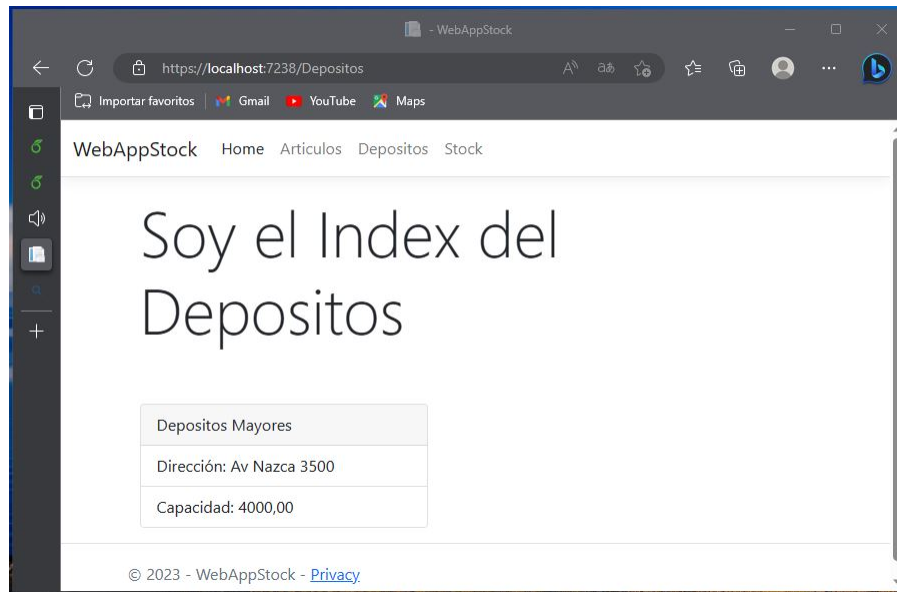


Figura 21: Vista desde el navegador sobre el Index de Depositos

7. Referencias

1. Link al Drive con el Código
2. Problema solucionado en el string de conexión