Projecte de llenguatge de marques - CocktailDB

Nil Jimeno, Bernat Brucet

23 de maig de 2024

1. Justificació del projecte

Figura 1: Captura de la pàgina

Per fer l'aplicació de l'api, hem utilitzat CocktailDB, una api que dóna informació sobre cocktails.

CocktailDB és una api de lliure ús que dóna informació sobre cocktails. Té varies trucades útils per aconseguir informació de la base de dades, que és enviada en forma de llista, a on cada atribut és un cocktail diferent.

Una altra raó per la qual hem decidit utilitzar aquesta api en concret és que és de lliure ús. Això vol dir que no té un rate limit aparent, per la qual cosa no ens hem de precupar pel límit de trucades a l'api, i tampoc ens hem de registrar per utilitzar-la. CocktailDB tampoc necessita registrar-se per utilitazar-la, la qual cosa facilita el treball.

L'estil de la web està basat en l'ambientació de Va-11 Hall-a, una novel·la visual on la principal mecànica és preparar cocktails. Els colors de la pàgina i la lletra estan fets de tal manera que encaixin amb l'estètica del fons, que està tret de la novela visual.

Per fer l'estil de la pàgina no s'ha utilitzat cap template d'html, hem preferit fer el treball nosaltres en comptes de copiar i enganxar d'internet.

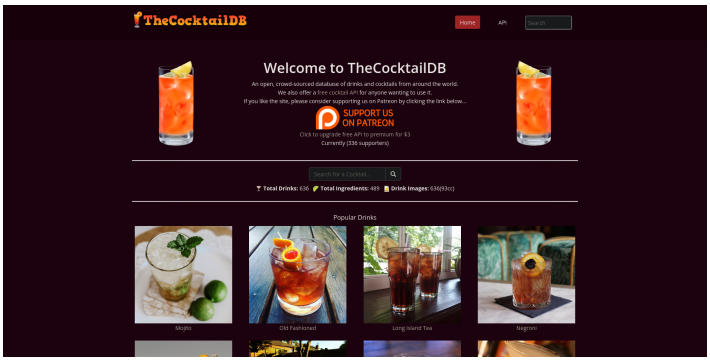

Figura 2: Pàgina principal de cocktaildb

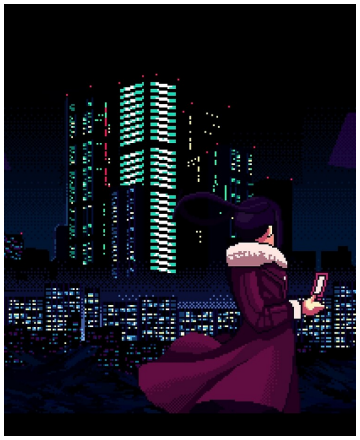

Figura 3: Portada de Va-11 Hall-a

2. Arquitectura del programa

Models A la carpeta Models hi ha les classes a on es guarden els atributs que es reben des de l'api. Cada un dels atributs rebuts en json passa a una variable de la classe.

```
Models
- Cocktail.cs
```

Controllers Conté l'arxiu principal de a web (HomeController.cs), que té les funcions per cada una de les vistes. Aquest arxiu també manega les trucades de les apis, guarda la informació rebuda a una instància de la classe corresponent i passa aquesta classe com a model a la vista.

```
Controllers/
- HomeController.cs
```

Views És la carpeta a on es guarden les pàgines en format cshtml. Es divideix en dues subcarpetes: home, a on es guarden les pàgines, i shared, a on es guarda el layout compartit (que es repeteix a totes les pàgines). Els arxius _ViewStart.cs i _ViewImport.cs serveixen, respectivament, per executar els cshtml corresponents (incluint el layout i la pàgina) i importar els moduls necessaris.

```
Views/
- Home
    - Cocktails.cshtml
    - CocktailsByIngredient.cshtml
    - Index.cshtml
    - One.cshtml
- Shared
    - _Layout.cshtml
- _ViewImports.cshtml
- _ViewStart.cshtml
```

wwwroot És a on es guarden els assets de la web, incluint el css, el ttf i la imatge del fons.

```
wwwroot/
- Darker_Glitch_City.png
- VT323-Regular.ttf
- site.css
```

3. Explicació del codi

3.1 HomeController

El fitxer HomeController.cs és l'encarregat de manegar la web. Té una funció per cada pàgina', que prepara la vista d'aquesta i mostra l'html resultant al client. Depèn de la funció executada, carrega una vista o una altra, que té el mateix nom que la funció.

3.2 Trucades a la api

Per aconseguir els valors per mostrar a la pàgina, primer es fa una trucada a l'api des de HomeController.cs (aquesta depèn de la pàgina que està carregant i l'argument que ha rebut).

```
public ActionResult CocktailsByIngredient(string ingredient)
{
    string apiUrl = $"https://www.thecocktaildb.com/api/json/v1/1/filter.php?i={ingredient}

    Console.WriteLine(apiUrl);

    var client = new HttpClient();
    var response = client.GetAsync(apiUrl).Result;
    var content = response.Content.ReadAsStringAsync().Result;

    Simplified model = JsonConvert.DeserializeObject<Simplified>(content);
    return View(model);
}
```

Figura 4: Exemple de funció a HomeController.cs

Els resultats de la trucada són rebuts en format JSON, que es transforma utilitzant Newtonsoft per guardar-se en la variable model, que és una instància d'una classe amb els paràmetres que es volen mostrar i s'utilitza com a argument al carregar la vista. En el nostre cas, les classes són llistes amb informació sobre les begudes que es volen mostrar.

```
public class SimplifiedCocktail
{
    public string? idDrink { get; set; }
    public string? strDrink { get; set; }
    public string? strDrinkThumb { get; set; }
}
```

Figura 5: Exemple de classe a Cocktail.cs

3.3 Vistes

Per carregar la vista, primer s'executen també els arxius _ViewStart.cs i _ViewImport.cs per carregar els layouts corresponents i importar els mòduls necessaris.

Les vistes són l'html rebut per part del controlador, que és el que es mostra a la pantalla del client. Aquests es fan a partir dels fitxers cshtml, que ajunten html amb funcions de C# i ASP.

Algunes d'aquestes funcions utilitzades són el foreach, que repeteix un codi html per cada beguda que es vulgui mostrar, o un if, per comprovar condicions. També s'utilitzen variables per mostrar

paràmetres a l'html, com ara la informació de les begudes mostrades (incluint títol, descripció, imatge...).

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

@if (@Model.drinks != null) {

<div class="items">

@foreach (var drink in Model.drinks) {
<div class="item normal-item smol">
    <div class="title">
    <h2>@drink.strDrink</h2>
    </div>

    <div class="image">
    <img src=@drink.strDrinkThumb alt="">
    </div>

    <div class="description">
    <p>
    @drink.strCategory,
    @if (drink.strIBA != null) {
    <span>
        @drink.strIBA,
    </span>
    }
    @drink.strAlcoholic
    @{string[] ingredients = [drink.strIngredient1,
    drink.strIngredient2,
    drink.strIngredient3,
```

Figura 6: Exemple de cshtml a Cocktails.cshtml

Els fitxers de la carpeta home utilitzen comparteixen el codi del layout, que s'afegeix automàticament a totes les vistes. Aquest conté configuracions del head, el títol, el navbar i la barra de busca. És un fitxer compartit per no haver de repetir aquella porció de codi a cada cshtml.

3.4 Links

Per anar de pàgina en pàgina, s'utilitzen links amb paràmetres especials (de cshtml) que truquen a altres funcions de HomeController. Per cada funció, es carrega la vista amb aquest mateix nom. En cas de que requereixin més informació, com ara els valors que es volen buscar (com a argument), s'utilitza asp-target per canviar-los.

D'una manera similar funciona el formulari, al qual se l'ha d'especificar a quina funció està trucant, i aquest retornará els valors del formulari en forma d'argument. Els formularis són utilitzats per fer funcionar la barra de busca.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - mvc</title>
    <link rel="stylesheet" href="~/site.css" asp-append-version="true" />
</head>
<body>


<h1 class="page-title">CocktailDB</h1>
<p class="marquee">
<br>

@using(Html.BeginForm("Cocktails", "Home", FormMethod.Post))
{
<div class="sdiv">
<input class="search" type="text" name="search" id="search" placeholder="Search drink h
</div>

<input type="submit" value="Search" style="width:0px; height:0px; font-size:0; border:
}
<div class="navbar">
<a class="nav-link" asp-area="" asp-controller="Home" asp-action="Index">Home
</a><a class="nav-link" asp-area="" asp-controller="Home" asp-action="One">One
</a><a class="nav-link" asp-area="" asp-controller="Home" asp-action="Cocktails">Many
</a>
</div>


@RenderBody()

</body>
</html>
```

Figura 7: _Layout.cshtml (tallat)

4. Possibles millores

Millorar l'espaiat Quan hi ha dos resultats en una fila, s'espaien de manera estranya (cada un a un cantó). Estaria bé solucionar aquest problema per aconseguir una millor cohesió estètica.

Millorar la funcionalitat si es pogués. Al no estar registrats i no haver fet una donació, no tenim permís per accedir a certs aspectes de la api. Això és el millor que podem fer de moment.

Canviar de llenguatge Es podria millorar el programa si utilitzessim un altre llenguatge de programació, com ara Rust o Zig. Mentre utilitzem C#, MVC i Razor, el programa no pot millorar gaire més, al menys fins on arriben les nostres capacitats.