Capolavoro

Sommario

Cambio delle valute 3
Model e Query 4
AppDbContext 5
Cambio 7
Appsetting.json 9
Controllers 10
Programs 12
Index 14
Risultato finale 19

Cambio delle valute

Per realizzare questo cambio di valute ho utilizzato visual studio code utilizzando C# come linguaggio di programmazione e HTML per la parte grafica della pagina web.

Come prima cosa ho creato il progetto creando un progetto di tipo API Web ASP.NET Core, ho creato le cartelle per il Model, il Controllers e Data.

1. ASP.NET Core
   ASP.NET Core è un framework open-source di sviluppo web, versatile e multipiattaforma, utilizzato per la creazione di applicazioni web moderne. Offre un'architettura modulare e scalabile per la costruzione di applicazioni web robuste e performanti.
2. Entity Framework Core
   Entity Framework Core è un ORM (Object-Relational Mapping) che semplifica l'interazione con il database tramite oggetti .NET. È ampiamente utilizzato per la gestione della persistenza dei dati nelle applicazioni ASP.NET Core.
3. Newtonsoft.Json
   Newtonsoft.Json è una libreria popolare per la serializzazione e la de serializzazione di dati in formato JSON in ambiente .NET. È utilizzato in questo progetto per manipolare i dati JSON ricevuti dalle chiamate API.
4. Chart.js
   Chart.js è una libreria JavaScript per la creazione di grafici interattivi e personalizzabili. È utilizzato per visualizzare il grafico del cambio di valuta nella pagina web del tuo progetto.
5. HttpClient
   HttpClient è una classe .NET utilizzata per effettuare richieste HTTP verso risorse remote. È ampiamente utilizzato in ASP.NET Core per la comunicazione con API esterne, come nel tuo caso per ottenere i tassi di cambio delle valute.

Queste librerie svolgono un ruolo cruciale nel tuo progetto, consentendo la gestione delle richieste HTTP, la manipolazione dei dati JSON, la creazione di interfacce utente e la comunicazione con il database.

Model e Query

```
using System.ComponentModel.DataAnnotations;

namespace cavolavoro.Models
{
    public class Valuta
    {
        [Key]
        public int Id { get; set; }
        public string Codice { get; set; }
        public string Nome { get; set; }
        public decimal TassoDiCambio { get; set; }
    }
}
```

Nel Model definiamo la struttura di una valuta con le proprietà dove la chiave primaria è l'Id, il Codice, il Nome e il TassoDiCambio

```
create table Categorie(
Id integer primary key,
Codice nvarchar(50),
Nome nvarchar(50),
TassoDiCambio DECIMAL(20, 2)
);
```

Nelle query andiamo a creare una tabella di nome Categorie nel database 'cavolavoro' dove:

create table Categorie: si sta creando una nuova tabella nel database chiamata Categorie.

`Id integer primary key, =` Questo definisce una colonna chiamata Id di tipo integer (intero) e la contrassegna come chiave primaria della tabella. Una chiave primaria è univoca per ogni riga nella tabella e viene utilizzata per identificare univocamente ogni riga.

`Codice nvarchar(50), =` Questa istruzione definisce una colonna chiamata Codice di tipo nvarchar(50). nvarchar è un tipo di dati che può contenere una stringa di caratteri Unicode di lunghezza variabile, e (50) specifica la lunghezza massima della stringa (50 caratteri in questo caso).

`Nome nvarchar(50), =` Questa istruzione definisce una colonna chiamata Nome di tipo nvarchar(50),

`TassoDiCambio DECIMAL(20, 2) =` DECIMAL è un tipo di dati numerici che può contenere numeri decimali, e (20, 2) specifica che il numero può avere un massimo di 20 cifre totali, di cui 2 sono cifre decimali. Questo è utile per rappresentare i tassi di cambio con precisione.

AppDbContext

```
using cavolavoro.Models;
using Microsoft.EntityFrameworkCore;

namespace cavolavoro.Data
{
    public class AppDbContext : DbContext
    {
        public DbSet<Valuta> Valute { get; set; }

        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) {}

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {

                    optionsBuilder.UseSqlServer("Server=(localdb)\\MSSQLLocalDB;Databas
                    e=NomeDelTuoDatabase;Trusted_Connection=True;MultipleActiveResultSe
                    ts=true");
            }
        }
    }
}
```

`using cavolavoro.Models; =` Importa il namespace cavolavoro.Models, che presumibilmente contiene la definizione della classe Valuta.

`using Microsoft.EntityFrameworkCore; =` Importa il namespace di Entity Framework Core, necessario per lavorare con DbContext e altre classi correlate.

`public DbSet<Valuta> Valute { get; set; } =` Questa proprietà rappresenta un set di entità di tipo Valuta. In Entity Framework, un DbSet<T> corrisponde a una tabella del database e Valuta è il tipo di entità che sarà mappato a quella tabella.

`public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) {} =` Questo è il costruttore della classe. Accetta un parametro di tipo DbContextOptions<AppDbContext>, che contiene la configurazione per il contesto. Il costruttore chiama il costruttore della classe base (DbContext) passando options.

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
       if (!optionsBuilder.IsConfigured)
       {
       optionsBuilder.UseSqlServer("Server=(localdb)\\MSSQLLocalDB;Database=NomeDelTuoD
       atabase;Trusted_Connection=True;MultipleActiveResultSets=true");
        }
}
```

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder): Questo metodo viene utilizzato per configurare il contesto se non è già configurato.

Viene chiamato da Entity Framework quando crea l'istanza del contesto.

`if (!optionsBuilder.IsConfigured)` = Verifica se optionsBuilder non è già configurato.

`optionsBuilder.UseSqlServer` = Configura il contesto per utilizzare un database SQL Server. La stringa di connessione specifica che il server è (localdb)\MSSQLLocalDB, che è un'istanza locale di SQL Server Express. Specifica anche il nome del database, l'autenticazione trusted e abilita MultipleActiveResultSets.

Cambio

```
using System.Net.Http.Json;

public class ServizioConversioneValuta
{
    private readonly HttpClient _httpClient;

    public ServizioConversioneValuta(HttpClient httpClient)
    {
        _httpClient = httpClient;
    }

    public class RispostaApiTassiDiCambio
    {
        public Dictionary<string, decimal> Tassi { get; set; }
        // Altri membri se presenti nella risposta JSON
    }

    public async Task<decimal> ConvertiValutaAsync(string valutaDa, string valutaA,
decimal importo)
    {
        try
        {
            var response = await
_httpClient.GetFromJsonAsync<RispostaApiTassiDiCambio>($"https://api.exchangeratesapi.
io/latest?base={valutaDa}&symbols={valutaA}");

            if (response != null)
            {
                decimal tassoDiCambio = response.Tassi[valutaA];
                return importo * tassoDiCambio;
            }
            else
            {
                throw new Exception("Errore durante la richiesta all'API di cambio
valuta.");
            }
        }
        catch (Exception ex)
        {
            throw new Exception("Errore durante la conversione della valuta.", ex);
        }
    }
}
```

`using System.Net.Http.Json; =` memorizza un'istanza di HttpClient, che viene utilizzata per inviare richieste HTTP.

```
public class RispostaApiTassiDiCambio
    {
        public Dictionary<string, decimal> Tassi { get; set; }
        // Altri membri se presenti nella risposta JSON
    }
```

Viene rappresentata come arriva la risposta JSON che si prevede di ricevere dall'API di tassi di cambio.
Tassi: Un dizionario che mappa le valute ai rispettivi tassi di cambio rispetto alla valuta di base.

```
var response = await
_httpClient.GetFromJsonAsync<RispostaApiTassiDiCambio>($"https://api.exchangeratesapi.
io/latest?base={valutaDa}&symbols={valutaA}");
```

Viene inviata una richiesta HTTP GET all'endpoint dell'API per ottenere i tassi di cambio.

La risposta viene deserializzata in un oggetto RispostaApiTassiDiCambio.

Se la risposta non è null, viene estratto il tasso di cambio per la valuta di destinazione.

L'importo viene moltiplicato per il tasso di cambio per ottenere il risultato della conversione.

Se la risposta è null, viene sollevata un'eccezione.

Se si verifica un'eccezione durante la richiesta o il calcolo, viene sollevata un'altra eccezione con un messaggio descrittivo.

Appsetting.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection":
"Server=(localdb)\\MSSQLLocalDB;Database=NomeDelTuoDatabase;Trusted_Connection=True;Mu
ltipleActiveResultSets=true"
  }
}
```

`"Logging"` = Questa sezione configura il sistema di log dell'applicazione.

`"LogLevel"` = Specifica i livelli di log per differenti categorie.

`"Default"` = Imposta il livello di log predefinito per l'applicazione.

Spesso viene impostato su "Information", il che significa che verranno registrati i log a livello di informazione e superiore (Warning, Error, Critical).

`"Microsoft.AspNetCore"` = Imposta il livello di log specifico per i componenti di ASP.NET Core. In questo caso è impostato "Warning", quindi verranno registrati solo i log di livello Warning e superiore per questi componenti.

`"AllowedHosts": "*",` = Specifica gli host consentiti per l'applicazione. Un valore di "*" significa che sono consentiti tutti gli host. Questa configurazione è utile per la sicurezza e il controllo del dominio di origine.

`"ConnectionStrings":` = Questa sezione contiene le stringhe di connessione al database che l'applicazione utilizza.

`"DefaultConnection"` = È la stringa di connessione predefinita per il database. In questo caso, è configurata per usare un'istanza locale di SQL Server LocalDB con il nome del database "NomeDelTuoDatabase".

- Server=(localdb)\MSSQLLocalDB: Specifica che viene utilizzato SQL Server LocalDB.
- Database=NomeDelTuoDatabase: Il nome del database a cui connettersi.
- Trusted_Connection=True: Usa l'autenticazione integrata di Windows per connettersi al database.
- MultipleActiveResultSets=true: Permette di avere più set di risultati attivi contemporaneamente per una singola connessione.

Controllers

```
using Microsoft.AspNetCore.Mvc;
using System.Net.Http;
using System.Threading.Tasks;
using Newtonsoft.Json.Linq;

namespace cavolavoro.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class CurrencyController : ControllerBase
    {
        private readonly IHttpClientFactory _httpClientFactory;
        private readonly string _apiKey = "49a2400c4607981bdbef8f7e";

        public CurrencyController(IHttpClientFactory httpClientFactory)
        {
            _httpClientFactory = httpClientFactory;
        }

        [HttpGet("convert")]
        public async Task<IActionResult> ConvertCurrency([FromQuery] string from,
[FromQuery] string to, [FromQuery] decimal amount)
        {
            var client = _httpClientFactory.CreateClient();
            var response = await client.GetStringAsync($"https://v6.exchangerate-
api.com/v6/{_apiKey}/latest/{from}");

            var rates = JObject.Parse(response)["conversion_rates"];
            var rate = rates[to].Value<decimal>();

            var convertedAmount = amount * rate;
            return Ok(new { amount = convertedAmount });
        }
    }
}
```

`private readonly IHttpClientFactory _httpClientFactory;` = Un factory per creare istanze di HttpClient, utilizzato per fare richieste HTTP.

`private readonly string_apiKey = "49a2400c4607981bdbef8f7e";` = La chiave API per accedere al servizio di conversione delle valute.

`CurrencyController(IHttpClientFactory httpClientFactory)` = Costruttore del controller, che accetta un IHttpClientFactory come dipendenza.

Questo metodo ConvertCurrency è un endpoint API che converte un importo da una valuta a un'altra. Ecco un riassunto delle sue operazioni principali:

Route e parametri:

- Mappato a GET /api/currency/convert.
- Accetta tre parametri di query: from (valuta di origine), to (valuta di destinazione) e amount (importo da convertire).

Richiesta HTTP:

- Crea un client HTTP utilizzando IHttpClientFactory.
- Effettua una richiesta GET a ExchangeRate-API per ottenere i tassi di cambio per la valuta di origine (from).

Elaborazione della risposta:

- Parsea la risposta JSON per ottenere i tassi di cambio.
- Estrae il tasso di cambio per la valuta di destinazione (to).

Conversione e risposta:

- Calcola l'importo convertito moltiplicando amount per il tasso di cambio.
- Restituisce l'importo convertito come oggetto JSON.

Programs

```
using Microsoft.AspNetCore.Builder;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.EntityFrameworkCore;
using cavolavoro.Data;
using Microsoft.Extensions.Configuration;

var builder = WebApplication.CreateBuilder(args);

// Configura i servizi
builder.Services.AddControllers();
builder.Services.AddHttpClient();
builder.Services.AddDbContext<AppDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

var app = builder.Build();

// Configura la pipeline HTTP
if (app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
}

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllers();
app.MapFallbackToFile("index.html");

app.Run();
```

`var builder = WebApplication.CreateBuilder(args);` = Crea un builder per configurare i servizi e la pipeline di elaborazione delle richieste dell'applicazione.

`builder.Services.AddControllers();` = Aggiunge i servizi necessari per utilizzare i controller MVC (Model-View-Controller).
`builder.Services.AddHttpClient();` = Aggiunge il servizio HTTP client, che consente di effettuare richieste HTTP.

`builder.Services.AddDbContext<AppDbContext>` = Configura il contesto del database (AppDbContext) per utilizzare SQL Server come provider di database. La stringa di connessione viene recuperata dalle configurazioni dell'applicazione

`var app = builder.Build();` = Costruisce l'applicazione con tutti i servizi configurati.

```
if (app.Environment.IsDevelopment())
{
        app.UseDeveloperExceptionPage();
}
```
= mostra una pagina di eccezione dettagliata quando si verifica un errore non gestito.

`app.UseStaticFiles();` = Abilita la gestione dei file statici (come HTML, CSS, immagini, ecc.).

`app.UseRouting();` = Aggiunge il middleware per il routing delle richieste

`app.UseAuthorization();` = Aggiunge il middleware per la gestione delle autorizzazioni.

`app.MapControllers();` = consente di rispondere alle richieste HTTP che corrispondono ai percorsi definiti nei controller.

`app.MapFallbackToFile("index.html");` = Mappa tutte le richieste non corrispondenti a percorsi definiti a un file specifico.

`app.Run();` = Esegue l'applicazione.

Index

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Currency Converter</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/chart.js@3.7.0"></script>
    <style>
        body {
            font-family: Arial, sans-serif;
            overflow: hidden;
            background-image: url('sfondo.jpg');
            background-size: cover;
            background-repeat: no-repeat;
            background-attachment: fixed;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }

        .container {
            padding: 80px;
            background-color: rgba(255, 255, 255, 0.9);
            border-radius: 24px;
            box-shadow: 0 0 50px rgba(0, 0, 0, 0.1);
            min-height: 80vh;
            width: 80vw;
        }

        .currency-title {
            color: red;
            text-align: center;
            font-size: 60px;
            font-weight: bold;
            animation: colorChange 5s infinite alternate;
            margin-bottom: 40px;
        }

        .currency-form {
            margin-top: 40px;
        }

        .currency-result {
            margin-top: 40px;
            font-size: 24px;
            text-align: center;
        }

        .convert-btn {
            margin-top: 20px;
        }

        .reset-btn {
            margin-top: 20px;
            margin-left: 20px;
        }
```

```
        .form-label {
            font-size: 28px;
        }

        .chart-title {
            font-size: 36px;
            text-align: center;
        }

        .chart-container {
            position: relative;
            width: 80%;
            margin: auto;
        }

        canvas {
            -moz-user-select: none;
            -webkit-user-select: none;
            -ms-user-select: none;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1 class="currency-title">Cambio di Valute</h1>
        <div class="currency-form">
            <div class="form-group">
                <label for="amount" class="form-label">Quantità di denaro a
disposizione:</label>
                <input type="number" id="amount" class="form-control" value="1">
            </div>
            <div class="row mt-3">
                <div class="col-md-6">
                    <div class="form-group">
                        <label for="from" class="form-label">Da:</label>
                        <select id="from" class="form-select">
                            <option value="EUR">EUR</option>
                            <option value="USD">USD</option>
                            <option value="GBP">GBP</option>
                            <option value="JPY">JPY</option>
                            <option value="AUD">AUD</option>
                            <option value="CAD">CAD</option>
                            <option value="CHF">CHF</option>
                            <option value="CNY">CNY</option>
                            <option value="SEK">SEK</option>
                            <option value="NZD">NZD</option>
                            <!-- Aggiungi altre valute se necessario -->
                        </select>
                    </div>
                </div>
                <div class="col-md-6">
                    <div class="form-group">
                        <label for="to" class="form-label">A:</label>
                        <select id="to" class="form-select">
                            <option value="USD">USD</option>
                            <option value="EUR">EUR</option>
                            <option value="GBP">GBP</option>
                            <option value="JPY">JPY</option>
                            <option value="AUD">AUD</option>
                            <option value="CAD">CAD</option>
                            <option value="CHF">CHF</option>
                            <option value="CNY">CNY</option>
                            <option value="SEK">SEK</option>
```

```
                            <option value="NZD">NZD</option>
                            <!-- Aggiungi altre valute se necessario -->
                        </select>
                    </div>
                </div>
            </div>
            <div class="text-center">
                <button onclick="convertCurrency()" class="btn btn-primary convert-
btn">Converti</button>
                <button onclick="resetForm()" class="btn btn-secondary reset-
btn">Reset</button>
            </div>
        </div>
        <div class="currency-result" id="currencyResult"></div>
        <div class="currency-chart">
            <h2 class="chart-title">Grafico del Cambio</h2>
            <div class="chart-container">
                <canvas id="currencyChart"></canvas>
            </div>
        </div>
    </div>

    <script>
        let currencyChart;
        const initialLabels = ['01 May', '03 May', '05 May', '07 May', '09 May', '11
May', '13 May', '15 May', '17 May', '19 May', '21 May', '23 May'];
        const initialData = [1.07, 1.068, 1.069, 1.071, 1.072, 1.074, 1.075, 1.078,
1.079, 1.081, 1.084, 1.086];

        function createLineChart(labels, data) {
            const ctx = document.getElementById('currencyChart').getContext('2d');
            if (currencyChart) {
                currencyChart.destroy();
            }
            currencyChart = new Chart(ctx, {
                type: 'line',
                data: {
                    labels: labels,
                    datasets: [{
                        label: 'Cambio di valuta',
                        data: data,
                        borderColor: 'rgba(75, 192, 192, 1)',
                        backgroundColor: 'rgba(75, 192, 192, 0.2)',
                        borderWidth: 2,
                        fill: true,
                        tension: 0.4 // Smooth curves
                    }]
                },
                options: {
                    scales: {
                        y: {
                            beginAtZero: false
                        }
                    },
                    plugins: {
                        legend: {
                            display: false
                        }
                    },
                    interaction: {
                        intersect: false,
                        mode: 'index'
                    },
                    elements: {
```

```
                        point: {
                            radius: 0
                        }
                    }
                }
            });
        }

        async function convertCurrency() {
            const amount = document.getElementById('amount').value;
            const from = document.getElementById('from').value;
            const to = document.getElementById('to').value;

            if (from === to) {
                document.getElementById('currencyResult').innerHTML = `<p>${amount}
${from} = ${amount} ${to}</p>`;
                return;
            }

            const response = await
fetch(`/api/currency/convert?from=${from}&to=${to}&amount=${amount}`);
            const data = await response.json();

            // Mostra il risultato del cambio
            document.getElementById('currencyResult').innerHTML = `<p>${amount}
${from} = ${data.amount} ${to}</p>`;
        }

        function resetForm() {
            document.getElementById('amount').value = 1;
            document.getElementById('from').value = 'EUR';
            document.getElementById('to').value = 'USD';
            document.getElementById('currencyResult').innerHTML = '';

            // Ripristina il grafico ai valori iniziali
            createLineChart(initialLabels, initialData);
        }

        document.addEventListener('DOMContentLoaded', function () {
            // Inizializza il grafico con i valori iniziali
            createLineChart(initialLabels, initialData);
        });
    </script>
</body>
</html>
```

Struttura e Stile

- Inclusioni: Utilizza Bootstrap per lo stile e Chart.js per i grafici.
- Stile Personalizzato: Definisce uno sfondo con un'immagine, centra i contenuti, e aggiunge stili per pulsanti, form, e grafici.

Contenuto

- Titolo: "Cambio di Valute" con animazione del colore.
- Form di Conversione: Include campi per l'importo, la valuta di origine e destinazione, e pulsanti per convertire e resettare.
- Risultato della Conversione: Un'area per mostrare i risultati.
- Grafico: Visualizza un grafico a linee dei tassi di cambio.

Funzionalità

- Creazione del Grafico: Utilizza Chart.js per creare un grafico a linee con dati iniziali.
- Conversione Valuta: Effettua una richiesta a un'API per ottenere il tasso di cambio e mostra il risultato.
- Reset: Ripristina i campi del form e reimposta il grafico ai dati iniziali.
- Inizializzazione: Al caricamento della pagina, viene creato un grafico con i dati iniziali.

Comportamento

- Interazione Dinamica: La pagina si aggiorna dinamicamente mostrando i risultati della conversione e aggiornando il grafico.

Risultato finale

In questa immagine viene mostrato la schermata del cambio delle valute:

Viene raffigurato in basso un grafico del mese di maggio dove viene mostrato quanto vale al momento l'euro.

L'utente inserirà da tastiera quanto vuole cambiare e selezionerà la valuta che dispone in quel momento per poi scegliere la valuta con la quale desidera cambiare.

Cliccando il tasto reset verrà riportato tutto come viene mostrato nell'immagine