Service за вчитување со страницирање:

using System.Net.Http.Json;

using TestBed.Models;

namespace TestBed.Services

{

public class TaxiRideService

{

private readonly HttpClient \_httpClient;

public TaxiRideService(HttpClient httpClient)

{

\_httpClient = httpClient;

}

public async Task<List<TaxiRide>> GetTaxiRidesAsync(int pageNumber, int pageSize)

{

var response = await \_httpClient.GetFromJsonAsync<List<TaxiRide>>(

$"api/TaxiRides2?pageNumber={pageNumber}&pageSize={pageSize}");

return response ?? new List<TaxiRide>();

}

}

}

**2. Controller во API за страницирање:**

(Во **TaxiRidesController.cs** на твоето Web API)

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using TestBed.Data;

using TestBed.Models;

namespace TestBed.Controllers

{

[ApiController]

[Route("api/[controller]")]

public class TaxiRides2Controller : ControllerBase

{

private readonly ApplicationDbContext \_context;

public TaxiRides2Controller(ApplicationDbContext context)

{

\_context = context;

}

[HttpGet]

public async Task<ActionResult<IEnumerable<TaxiRide>>> GetTaxiRides(int pageNumber = 1, int pageSize = 10)

{

var taxiRides = await \_context.TaxiRides2

.Skip((pageNumber - 1) \* pageSize)

.Take(pageSize)

.ToListAsync();

return Ok(taxiRides);

}

}

}

**3. Во Blazor (страница) — да прикажеме податоците:**

На пример, ќе направиш нова страница TaxiRides.razor:

@page "/taxirides"

@inject TaxiRideService TaxiRideService

<h3>Taxi Rides</h3>

@if (taxiRides == null)

{

<p><em>Loading...</em></p>

}

else

{

<table class="table">

<thead>

<tr>

<th>Medallion</th>

<th>Pickup Time</th>

<th>Drop Off Time</th>

<th>Distance (miles)</th>

<th>Fare Amount (USD)</th>

<th>Total Amount (USD)</th>

</tr>

</thead>

<tbody>

@foreach (var ride in taxiRides)

{

<tr>

<td>@ride.medallion</td>

<td>@ride.pickupTime</td>

<td>@ride.dropOffTime</td>

<td>@ride.distance</td>

<td>@ride.fareAmount</td>

<td>@ride.totalAmount</td>

</tr>

}

</tbody>

</table>

<div class="mt-2">

<button class="btn btn-primary" @onclick="PreviousPage" disabled="@(\_pageNumber == 1)">Previous</button>

<span class="mx-2">Page @\_pageNumber</span>

<button class="btn btn-primary" @onclick="NextPage">Next</button>

</div>

}

@code {

private List<TaxiRide> taxiRides = new();

private int \_pageNumber = 1;

private int \_pageSize = 10;

protected override async Task OnInitializedAsync()

{

await LoadTaxiRides();

}

private async Task LoadTaxiRides()

{

taxiRides = await TaxiRideService.GetTaxiRidesAsync(\_pageNumber, \_pageSize);

}

private async Task NextPage()

{

\_pageNumber++;

await LoadTaxiRides();

}

private async Task PreviousPage()

{

if (\_pageNumber > 1)

{

\_pageNumber--;

await LoadTaxiRides();

}

}

}

**4. Startup:**

Во Program.cs, регистрирај го сервисот:

builder.Services.AddScoped<TaxiRideService>();

Во твојата TaxiRides.razor страница, над табелата:

<div class="mb-3">

<button class="btn btn-success" @onclick="ExportToCsv">Export to CSV</button>

<button class="btn btn-warning mx-2" @onclick="ExportToExcel">Export to Excel</button>

<button class="btn btn-danger" @onclick="ExportToPdf">Export to PDF</button>

</div>

Потоа во @code делот, ги додаваме методите:

private async Task ExportToCsv()

{

var csv = new StringBuilder();

csv.AppendLine("Medallion,PickupTime,DropOffTime,Distance,FareAmount,TotalAmount");

foreach (var ride in taxiRides)

{

csv.AppendLine($"{ride.medallion},{ride.pickupTime},{ride.dropOffTime},{ride.distance},{ride.fareAmount},{ride.totalAmount}");

}

await DownloadFile("taxi\_rides.csv", "text/csv", csv.ToString());

}

private async Task ExportToExcel()

{

var excel = new StringBuilder();

excel.AppendLine("Medallion\tPickupTime\tDropOffTime\tDistance\tFareAmount\tTotalAmount");

foreach (var ride in taxiRides)

{

excel.AppendLine($"{ride.medallion}\t{ride.pickupTime}\t{ride.dropOffTime}\t{ride.distance}\t{ride.fareAmount}\t{ride.totalAmount}");

}

await DownloadFile("taxi\_rides.xls", "application/vnd.ms-excel", excel.ToString());

}

private async Task ExportToPdf()

{

var pdfContent = new StringBuilder();

pdfContent.AppendLine("Taxi Rides Report\n");

foreach (var ride in taxiRides)

{

pdfContent.AppendLine($"Medallion: {ride.medallion}");

pdfContent.AppendLine($"Pickup Time: {ride.pickupTime}");

pdfContent.AppendLine($"Drop Off Time: {ride.dropOffTime}");

pdfContent.AppendLine($"Distance: {ride.distance} miles");

pdfContent.AppendLine($"Fare Amount: {ride.fareAmount} USD");

pdfContent.AppendLine($"Total Amount: {ride.totalAmount} USD");

pdfContent.AppendLine("------------------------------------");

}

await DownloadFile("taxi\_rides.txt", "application/octet-stream", pdfContent.ToString());

}

И помошна функција DownloadFile:

private async Task DownloadFile(string filename, string contentType, string content)

{

var bytes = System.Text.Encoding.UTF8.GetBytes(content);

var base64 = Convert.ToBase64String(bytes);

await JS.InvokeVoidAsync(

"downloadFile",

filename,

contentType,

base64

);

}

**3. Во wwwroot/index.html или wwwroot/\_Host.cshtml (Blazor Server)**

Додај скрипта за download:

<script>

window.downloadFile = (filename, contentType, base64) => {

const link = document.createElement('a');

link.href = `data:${contentType};base64,${base64}`;

link.download = filename;

link.click();

};

</script>

Еве ти **целосниот код** за TaxiRides.razor:

@page "/taxirides"

@inject HttpClient Http

@inject IJSRuntime JS

<h3>Taxi Rides</h3>

<div class="mb-3">

<button class="btn btn-success" @onclick="ExportToCsv">Export to CSV</button>

<button class="btn btn-warning mx-2" @onclick="ExportToExcel">Export to Excel</button>

<button class="btn btn-danger" @onclick="ExportToPdf">Export to PDF</button>

</div>

<table class="table table-striped">

<thead>

<tr>

<th>Medallion</th>

<th>Pickup Time</th>

<th>Drop Off Time</th>

<th>Distance (miles)</th>

<th>Fare Amount (USD)</th>

<th>Total Amount (USD)</th>

</tr>

</thead>

<tbody>

@foreach (var ride in taxiRides)

{

<tr>

<td>@ride.medallion</td>

<td>@ride.pickupTime</td>

<td>@ride.dropOffTime</td>

<td>@ride.distance</td>

<td>@ride.fareAmount</td>

<td>@ride.totalAmount</td>

</tr>

}

</tbody>

</table>

<div class="d-flex justify-content-between mt-3">

<button class="btn btn-primary" @onclick="PreviousPage" disabled="@(\_currentPage == 1)">Previous</button>

<span>Page @\_currentPage of @\_totalPages</span>

<button class="btn btn-primary" @onclick="NextPage" disabled="@(\_currentPage == \_totalPages)">Next</button>

</div>

@code {

private List<TaxiRide> taxiRides = new();

private int \_currentPage = 1;

private int \_pageSize = 10;

private int \_totalPages = 1;

protected override async Task OnInitializedAsync()

{

await LoadTaxiRides();

}

private async Task LoadTaxiRides()

{

var allRides = await Http.GetFromJsonAsync<List<TaxiRide>>("https://localhost:5001/api/TaxiRides"); // Промени URL ако треба

if (allRides != null)

{

\_totalPages = (int)Math.Ceiling(allRides.Count / (double)\_pageSize);

taxiRides = allRides

.Skip((\_currentPage - 1) \* \_pageSize)

.Take(\_pageSize)

.ToList();

}

}

private async Task NextPage()

{

if (\_currentPage < \_totalPages)

{

\_currentPage++;

await LoadTaxiRides();

}

}

private async Task PreviousPage()

{

if (\_currentPage > 1)

{

\_currentPage--;

await LoadTaxiRides();

}

}

private async Task ExportToCsv()

{

var csv = new StringBuilder();

csv.AppendLine("Medallion,PickupTime,DropOffTime,Distance,FareAmount,TotalAmount");

foreach (var ride in taxiRides)

{

csv.AppendLine($"{ride.medallion},{ride.pickupTime},{ride.dropOffTime},{ride.distance},{ride.fareAmount},{ride.totalAmount}");

}

await DownloadFile("taxi\_rides.csv", "text/csv", csv.ToString());

}

private async Task ExportToExcel()

{

var excel = new StringBuilder();

excel.AppendLine("Medallion\tPickupTime\tDropOffTime\tDistance\tFareAmount\tTotalAmount");

foreach (var ride in taxiRides)

{

excel.AppendLine($"{ride.medallion}\t{ride.pickupTime}\t{ride.dropOffTime}\t{ride.distance}\t{ride.fareAmount}\t{ride.totalAmount}");

}

await DownloadFile("taxi\_rides.xls", "application/vnd.ms-excel", excel.ToString());

}

private async Task ExportToPdf()

{

var pdfContent = new StringBuilder();

pdfContent.AppendLine("Taxi Rides Report\n");

foreach (var ride in taxiRides)

{

pdfContent.AppendLine($"Medallion: {ride.medallion}");

pdfContent.AppendLine($"Pickup Time: {ride.pickupTime}");

pdfContent.AppendLine($"Drop Off Time: {ride.dropOffTime}");

pdfContent.AppendLine($"Distance: {ride.distance} miles");

pdfContent.AppendLine($"Fare Amount: {ride.fareAmount} USD");

pdfContent.AppendLine($"Total Amount: {ride.totalAmount} USD");

pdfContent.AppendLine("------------------------------------");

}

await DownloadFile("taxi\_rides.txt", "application/octet-stream", pdfContent.ToString());

}

private async Task DownloadFile(string filename, string contentType, string content)

{

var bytes = System.Text.Encoding.UTF8.GetBytes(content);

var base64 = Convert.ToBase64String(bytes);

await JS.InvokeVoidAsync(

"downloadFile",

filename,

contentType,

base64

);

}

}

Плус — во wwwroot/index.html (ако е Blazor WebAssembly) или wwwroot/\_Host.cshtml (ако е Blazor Server), додади го ова најдолу пред </body>:

<script>

window.downloadFile = (filename, contentType, base64) => {

const link = document.createElement('a');

link.href = `data:${contentType};base64,${base64}`;

link.download = filename;

link.click();

};

</script>

Еве го **целосниот код** за TaxiRides.razor со убаво табеларно PDF генерирање:

@page "/taxirides"

@inject HttpClient Http

@inject IJSRuntime JS

<h3>Taxi Rides</h3>

<div class="mb-3">

<button class="btn btn-success" @onclick="ExportToCsv">Export to CSV</button>

<button class="btn btn-warning mx-2" @onclick="ExportToExcel">Export to Excel</button>

<button class="btn btn-danger" @onclick="ExportToPdf">Export to PDF</button>

</div>

<table class="table table-striped">

<thead>

<tr>

<th>Medallion</th>

<th>Pickup Time</th>

<th>Drop Off Time</th>

<th>Distance (miles)</th>

<th>Fare Amount (USD)</th>

<th>Total Amount (USD)</th>

</tr>

</thead>

<tbody>

@foreach (var ride in taxiRides)

{

<tr>

<td>@ride.medallion</td>

<td>@ride.pickupTime</td>

<td>@ride.dropOffTime</td>

<td>@ride.distance</td>

<td>@ride.fareAmount</td>

<td>@ride.totalAmount</td>

</tr>

}

</tbody>

</table>

<div class="d-flex justify-content-between mt-3">

<button class="btn btn-primary" @onclick="PreviousPage" disabled="@(\_currentPage == 1)">Previous</button>

<span>Page @\_currentPage of @\_totalPages</span>

<button class="btn btn-primary" @onclick="NextPage" disabled="@(\_currentPage == \_totalPages)">Next</button>

</div>

@code {

private List<TaxiRide> taxiRides = new();

private int \_currentPage = 1;

private int \_pageSize = 10;

private int \_totalPages = 1;

protected override async Task OnInitializedAsync()

{

await LoadTaxiRides();

}

private async Task LoadTaxiRides()

{

var allRides = await Http.GetFromJsonAsync<List<TaxiRide>>("https://localhost:5001/api/TaxiRides"); // смени URL ако треба

if (allRides != null)

{

\_totalPages = (int)Math.Ceiling(allRides.Count / (double)\_pageSize);

taxiRides = allRides

.Skip((\_currentPage - 1) \* \_pageSize)

.Take(\_pageSize)

.ToList();

}

}

private async Task NextPage()

{

if (\_currentPage < \_totalPages)

{

\_currentPage++;

await LoadTaxiRides();

}

}

private async Task PreviousPage()

{

if (\_currentPage > 1)

{

\_currentPage--;

await LoadTaxiRides();

}

}

private async Task ExportToCsv()

{

var csv = new StringBuilder();

csv.AppendLine("Medallion,PickupTime,DropOffTime,Distance,FareAmount,TotalAmount");

foreach (var ride in taxiRides)

{

csv.AppendLine($"{ride.medallion},{ride.pickupTime},{ride.dropOffTime},{ride.distance},{ride.fareAmount},{ride.totalAmount}");

}

await DownloadFile("taxi\_rides.csv", "text/csv", csv.ToString());

}

private async Task ExportToExcel()

{

var excel = new StringBuilder();

excel.AppendLine("Medallion\tPickupTime\tDropOffTime\tDistance\tFareAmount\tTotalAmount");

foreach (var ride in taxiRides)

{

excel.AppendLine($"{ride.medallion}\t{ride.pickupTime}\t{ride.dropOffTime}\t{ride.distance}\t{ride.fareAmount}\t{ride.totalAmount}");

}

await DownloadFile("taxi\_rides.xls", "application/vnd.ms-excel", excel.ToString());

}

private async Task ExportToPdf()

{

var headers = new[] { "Medallion", "Pickup Time", "Drop Off Time", "Distance", "Fare Amount", "Total Amount" };

var rows = taxiRides.Select(ride => new object[]

{

ride.medallion,

ride.pickupTime.ToString(),

ride.dropOffTime.ToString(),

ride.distance.ToString(),

ride.fareAmount.ToString(),

ride.totalAmount.ToString()

}).ToArray();

await JS.InvokeVoidAsync("generatePdf", headers, rows);

}

private async Task DownloadFile(string filename, string contentType, string content)

{

var bytes = System.Text.Encoding.UTF8.GetBytes(content);

var base64 = Convert.ToBase64String(bytes);

await JS.InvokeVoidAsync(

"downloadFile",

filename,

contentType,

base64

);

}

}

И сега, во wwwroot/index.html или wwwroot/\_Host.cshtml, пред </body>, додади го ова:

<script src="https://cdnjs.cloudflare.com/ajax/libs/pdfmake/0.1.72/pdfmake.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/pdfmake/0.1.72/vfs\_fonts.js"></script>

<script>

window.downloadFile = (filename, contentType, base64) => {

const link = document.createElement('a');

link.href = `data:${contentType};base64,${base64}`;

link.download = filename;

link.click();

};

window.generatePdf = (headers, rows) => {

var docDefinition = {

content: [

{ text: 'Taxi Rides Report', style: 'header' },

{

table: {

headerRows: 1,

widths: ['\*', '\*', '\*', '\*', '\*', '\*'],

body: [

headers,

...rows

]

}

}

],

styles: {

header: {

fontSize: 18,

bold: true,

marginBottom: 15

}

}

};

pdfMake.createPdf(docDefinition).download('taxi\_rides.pdf');

};

</script>