упер, ќе ти испратам **целосни, прецизни верзии на кодовите** од сите засегнати фајлови, со **единствена промена – функционалноста за менување на јазик да работи преку cookie**. Сè друго останува идентично.

**✅ 1. Program.cs – НЕМА промени (ова е веќе добро конфигурирано!)**

✅ **Ништо не менувај тука.**

✅ 2. LanguageSelector.razor – **ова е правилната верзија**

@inject NavigationManager NavigationManager

@using System.Globalization

<select @onchange="OnLanguageChanged" class="form-select">

<option value="mk-MK" selected="@IsSelected("mk-MK")">Македонски</option>

<option value="en-US" selected="@IsSelected("en-US")">English</option>

<option value="sr-RS" selected="@IsSelected("sr-RS")">Srpski</option>

</select>

@code {

private async Task OnLanguageChanged(ChangeEventArgs e)

{

var culture = e.Value?.ToString() ?? "en-US";

var uri = NavigationManager.Uri;

var newUrl = $"/culture/setculture?culture={culture}&redirectUri={Uri.EscapeDataString(uri)}";

NavigationManager.NavigateTo(newUrl, forceLoad: true);

}

private string IsSelected(string culture)

{

var currentCulture = CultureInfo.CurrentUICulture.Name;

return currentCulture.Equals(culture, StringComparison.InvariantCultureIgnoreCase) ? "selected" : null;

}

}

**✅ 3. CultureController.cs – овој останува ист**

✅ Не менуваме ништо тука. Тој веќе го поставува cookie-то со јазикот.

✅ 4. MainLayout.razor – **веќе содржи LanguageSelector** (остави вака)

@inherits LayoutComponentBase

@using BlazorAppPaginationClient.Shared

<PageTitle>BlazorAppPaginationClient</PageTitle>

<div class="page">

<div class="sidebar">

<NavMenu />

</div>

<main>

<div class="top-row px-4" style="display: flex; justify-content: space-between; align-items: center;">

<a href="https://docs.microsoft.com/aspnet/" target="\_blank">About</a>

<LanguageSelector />

</div>

<article class="content px-4">

@Body

</article>

</main>

</div>

🚫 5. PaginatedData.razor – **ОВДЕ ПРАВИМЕ ЕДИНСТВЕНА ПРОМЕНА**

Целата оваа логика:

<div class="language-selector">

<label>@L["Language"]</label>

<select @onchange="OnLanguageChanged">

<option value="en">English</option>

<option value="mk">Македонски</option>

<option value="sr">Српски</option>

</select>

</div>

➡ **ја заменуваме со:**

<div class="language-selector">

<label>@L["Language"]</label>

<LanguageSelector />

</div>

**6. PaginatedData.razor.cs (или во @code блокот): избриши го следново**

Ова **мора да се отстрани** – не е потребно и го нарушува cookie-поставениот јазик:

var uri = new Uri(NavigationManager.Uri);

var queryParams = System.Web.HttpUtility.ParseQueryString(uri.Query);

var culture = queryParams["culture"] ?? "en";

var cultureInfo = new System.Globalization.CultureInfo(culture);

System.Threading.Thread.CurrentThread.CurrentCulture = cultureInfo;

System.Threading.Thread.CurrentThread.CurrentUICulture = cultureInfo;

Mislam deka 5 I 6 se dvete od paginatedData sega eve celosen kod:

@page "/taxiridesapp"

@using BlazorAppPaginationClient.Models

@using System.Text

@inject IJSRuntime JSRuntime

@inject HttpClient Http

@inject IStringLocalizer<SharedResource> L

@inject NavigationManager NavigationManager

@using BlazorAppPaginationClient.Resources

@using Microsoft.Extensions.Localization

<h3>@L["TaxiRides"]</h3>

<div class="controls-container">

<div class="language-selector">

<label>@L["Language"]</label>

<LanguageSelector />

</div>

<div class="search-box">

<label>@L["SearchLabel"]</label>

<input type="text" @bind="SearchText" @bind:event="oninput" placeholder="@L["SearchPlaceholder"]" class="search-input" />

</div>

<div class="page-size-selector">

<label>@L["PageSize"]</label>

<select @bind="PageSize" @bind:event="onchange" class="dropdown">

<option value="10">10</option>

<option value="20">20</option>

<option value="50">50</option>

<option value="100">100</option>

</select>

</div>

<div class="total-records">

<strong>@L["TotalRecords"] @FilteredRecords</strong>

</div>

</div>

@if (TaxiRecords == null)

{

<p>@L["Loading"]...</p>

}

else

{

<table class="table">

<thead>

<tr>

<th>#</th>

@foreach (var property in typeof(TaxiRecord).GetProperties())

{

<th @onclick="() => SortByColumn(property.Name)" class="sortable">

@property.Name @GetSortIndicator(property.Name)

</th>

}

</tr>

</thead>

<tbody>

@foreach (var (ride, index) in PaginatedRides.Select((r, i) => (r, i + 1 + ((\_currentPage - 1) \* PageSize))))

{

<tr>

<td>@index</td>

@foreach (var property in typeof(TaxiRecord).GetProperties())

{

<td>@property.GetValue(ride)</td>

}

</tr>

}

</tbody>

</table>

<div class="pagination-container">

<button @onclick="PreviousPage" disabled="@(\_currentPage == 1)" class="pagination-button">@L["Previous"]</button>

<span>@L["Page"] @\_currentPage @L["Of"] @TotalPages</span>

<button @onclick="NextPage" disabled="@(\_currentPage >= TotalPages)" class="pagination-button">@L["Next"]</button>

</div>

<div class="page-input-container">

<label>@L["EnterPage"]</label>

<input type="number" @bind="EnteredPage" class="page-input" />

<button @onclick="GoToPage" class="pagination-button">@L["Go"]</button>

</div>

<div class="export-buttons">

<button class="btn btn-primary" @onclick="ExportToCsv">@L["ExportCsv"]</button>

<button class="btn btn-success" @onclick="ExportToExcel">@L["ExportExcel"]</button>

<button class="btn btn-danger" @onclick="ExportToPdf">@L["ExportPdf"]</button>

</div>

}

<style>

.controls-container {

display: flex;

flex-wrap: wrap;

justify-content: space-between;

align-items: center;

margin-bottom: 15px;

gap: 10px;

}

.search-input, .dropdown, .page-input {

padding: 5px;

border-radius: 5px;

border: 1px solid #ccc;

}

.table {

width: 100%;

border-collapse: collapse;

margin-top: 10px;

}

.table th, .table td {

border: 1px solid #ddd;

padding: 8px;

text-align: left;

}

.sortable:hover {

cursor: pointer;

background-color: #f1f1f1;

}

.pagination-container, .page-input-container {

display: flex;

justify-content: center;

align-items: center;

margin-top: 10px;

gap: 10px;

}

.pagination-button {

padding: 5px 10px;

border: none;

background-color: #007bff;

color: white;

border-radius: 5px;

cursor: pointer;

}

.pagination-button:disabled {

background-color: #ccc;

cursor: not-allowed;

}

.export-buttons {

margin-top: 15px;

text-align: center;

}

</style>

@code {

private List<TaxiRecord> TaxiRecords = new();

private string SearchText = "";

private int \_currentPage = 1;

private int PageSize = 10;

private int EnteredPage;

private int TotalRecords = 0;

private int FilteredRecords => FilteredRides.Count;

private int TotalPages => (int)Math.Ceiling((double)FilteredRecords / PageSize);

private string SortColumn = nameof(TaxiRecord.pickupTime);

private bool SortAscending = true;

private List<TaxiRecord> FilteredRides =>

string.IsNullOrWhiteSpace(SearchText)

? TaxiRecords

: TaxiRecords.Where(r => typeof(TaxiRecord).GetProperties()

.Any(p => p.GetValue(r)?.ToString()?.Contains(SearchText, StringComparison.OrdinalIgnoreCase) == true)).ToList();

private List<TaxiRecord> PaginatedRides =>

FilteredRides

.OrderBy(r => SortAscending ? GetPropertyValue(r, SortColumn) : null)

.ThenByDescending(r => SortAscending ? null : GetPropertyValue(r, SortColumn))

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

.Take(PageSize)

.ToList();

protected override async Task OnInitializedAsync()

{

try

{

TaxiRecords = await Http.GetFromJsonAsync<List<TaxiRecord>>("http://localhost:5098/api/TaxiRides/GetTaxiRides2");

TotalRecords = TaxiRecords.Count;

}

catch (Exception ex)

{

Console.WriteLine($"Error: {ex.Message}");

}

}

private void NextPage()

{

if (\_currentPage < TotalPages) \_currentPage++;

}

private void PreviousPage()

{

if (\_currentPage > 1) \_currentPage--;

}

private void GoToPage()

{

if (EnteredPage >= 1 && EnteredPage <= TotalPages)

{

\_currentPage = EnteredPage;

}

}

private void SortByColumn(string columnName)

{

if (SortColumn == columnName)

{

SortAscending = !SortAscending;

}

else

{

SortColumn = columnName;

SortAscending = true;

}

}

private string GetSortIndicator(string columnName)

{

if (SortColumn == columnName)

{

return SortAscending ? "▲" : "▼";

}

return "";

}

private object GetPropertyValue(TaxiRecord ride, string propertyName)

{

var property = typeof(TaxiRecord).GetProperty(propertyName);

return property?.GetValue(ride, null);

}

private async Task ExportToCsv()

{

var csvContent = GenerateCsvContent(PaginatedRides);

await JSRuntime.InvokeVoidAsync("saveAsFile", "taxi\_rides.csv", csvContent);

}

private async Task ExportToExcel()

{

var excelContent = GenerateExcelContent(PaginatedRides);

await JSRuntime.InvokeVoidAsync("saveExcel", "taxi\_rides.xlsx", excelContent);

}

private async Task ExportToPdf()

{

var pdfContent = GeneratePdfContent(PaginatedRides);

await JSRuntime.InvokeVoidAsync("savePdf", "taxi\_rides.pdf", pdfContent);

}

private string GenerateCsvContent(List<TaxiRecord> data)

{

var sb = new StringBuilder();

sb.AppendLine("Medallion,Hash License,Pickup Time,Drop-off Time,Duration,Distance,Pickup Longitude,Pickup Latitude,Drop-off Longitude,Drop-off Latitude,Payment Type,Fare Amount,Surcharge,Tax,Tip Amount,Tolls Amount,Total Amount");

foreach (var item in data)

{

sb.AppendLine($"{item.medallion},{item.hashLicense},{item.pickupTime},{item.dropOffTime},{item.duration},{item.distance},{item.pLongitude},{item.pLatitude},{item.dLongitude},{item.dLatitude},{item.paymentType},{item.fareAmount},{item.surcharge},{item.tax},{item.tipAmount},{item.tollsAmount},{item.totalAmount}");

}

return sb.ToString();

}

private object GenerateExcelContent(List<TaxiRecord> data)

{

var sheet = new List<List<object>> {

new List<object> { "Medallion", "Hash License", "Pickup Time", "Drop-off Time", "Duration", "Distance", "Pickup Longitude", "Pickup Latitude", "Drop-off Longitude", "Drop-off Latitude", "Payment Type", "Fare Amount", "Surcharge", "Tax", "Tip Amount", "Tolls Amount", "Total Amount" }

};

foreach (var item in data)

{

sheet.Add(new List<object> { item.medallion, item.hashLicense, item.pickupTime, item.dropOffTime, item.duration, item.distance, item.pLongitude, item.pLatitude, item.dLongitude, item.dLatitude, item.paymentType, item.fareAmount, item.surcharge, item.tax, item.tipAmount, item.tollsAmount, item.totalAmount });

}

return sheet;

}

private string GeneratePdfContent(List<TaxiRecord> data)

{

var content = "<h1>Taxi Rides</h1><table><thead><tr><th>Medallion</th><th>Hash License</th><th>Pickup Time</th><th>Drop-off Time</th><th>Duration</th><th>Distance</th><th>Pickup Longitude</th><th>Pickup Latitude</th><th>Drop-off Longitude</th><th>Drop-off Latitude</th><th>Payment Type</th><th>Fare Amount</th><th>Surcharge</th><th>Tax</th><th>Tip Amount</th><th>Tolls Amount</th><th>Total Amount</th></tr></thead><tbody>";

foreach (var item in data)

{

content += $"<tr><td>{item.medallion}</td><td>{item.hashLicense}</td><td>{item.pickupTime}</td><td>{item.dropOffTime}</td><td>{item.duration}</td><td>{item.distance}</td><td>{item.pLongitude}</td><td>{item.pLatitude}</td><td>{item.dLongitude}</td><td>{item.dLatitude}</td><td>{item.paymentType}</td><td>{item.fareAmount}</td><td>{item.surcharge}</td><td>{item.tax}</td><td>{item.tipAmount}</td><td>{item.tollsAmount}</td><td>{item.totalAmount}</td></tr>";

}

content += "</tbody></table>";

return content;

}

}

✅ LanguageSelector.razor (целосен код)

@inject NavigationManager NavigationManager

@using System.Globalization

<select @onchange="OnLanguageChanged" class="form-select">

<option value="mk-MK" selected="@IsSelected("mk-MK")">Македонски</option>

<option value="en-US" selected="@IsSelected("en-US")">English</option>

<option value="sr-RS" selected="@IsSelected("sr-RS")">Srpski</option>

</select>

@code {

private async Task OnLanguageChanged(ChangeEventArgs e)

{

var culture = e.Value?.ToString() ?? "en-US";

var currentUrl = NavigationManager.Uri;

// Генерирање на URL за CultureController

var redirectUrl = $"/culture/setculture?culture={culture}&redirectUri={Uri.EscapeDataString(currentUrl)}";

// ForceLoad: true = повторно ја вчитува апликацијата за да се примени новата култура

NavigationManager.NavigateTo(redirectUrl, forceLoad: true);

}

private string IsSelected(string culture)

{

var currentCulture = CultureInfo.CurrentUICulture.Name;

return currentCulture.Equals(culture, StringComparison.InvariantCultureIgnoreCase) ? "selected" : null;

}

}