



Universitatea
Transilvania
din Brașov



Universitatea
Transilvania
din Brașov
FACULTATEA DE MATEMATICĂ
ȘI INFORMATICĂ

Aplicație Recunoaștere Optică a Caracterelor

IAG 10LF481

Student: Szekely Zoltan-Ioan

Supervizor: Dr. Răzvan Bocu

Cuprins

1. Introducere.....	3
2. Cerințe preliminare.....	3
3. Cerințe imagini.....	4
4. Crearea resursei Azure Cognitive Service.....	5
5. Crearea aplicației Blazor pe partea de server.....	9
6. Configurarea serviciului.....	18
7. Crearea componentei UI Blazor.....	18
8. Executarea demo-ului.....	25
9. Rezumat.....	27

1. Introducere

Această documentație prezintă crearea în detaliu a unei aplicații de recunoaștere optică a caracterelor (optical character recognition – OCR) folosind Blazor și serviciul de Computer Vision Cognitive Service de pe platforma de cloud Azure.

Computer Vision este un serviciu AI care analizează conținutul unei imagini. OCR-ul este un feature al Computer Vision-ului, folosit pentru a detecta textul tipărit într-o imagine.

Aplicația extrage textul din imagine și detectează limba textului. În prezent, serviciul suportă 25 de limbi.

2. Cerințe preliminare

- Instalarea celui mai recent .NET Core 3.1 SDK

<https://dotnet.microsoft.com/download/dotnet-core/3.1>

- Instalarea celei mai noi versiuni Visual Studio

<https://visualstudio.microsoft.com/downloads/>

- Crearea unei subscripții gratuite pe platforma Azure

<https://azure.microsoft.com/en-in/free/>

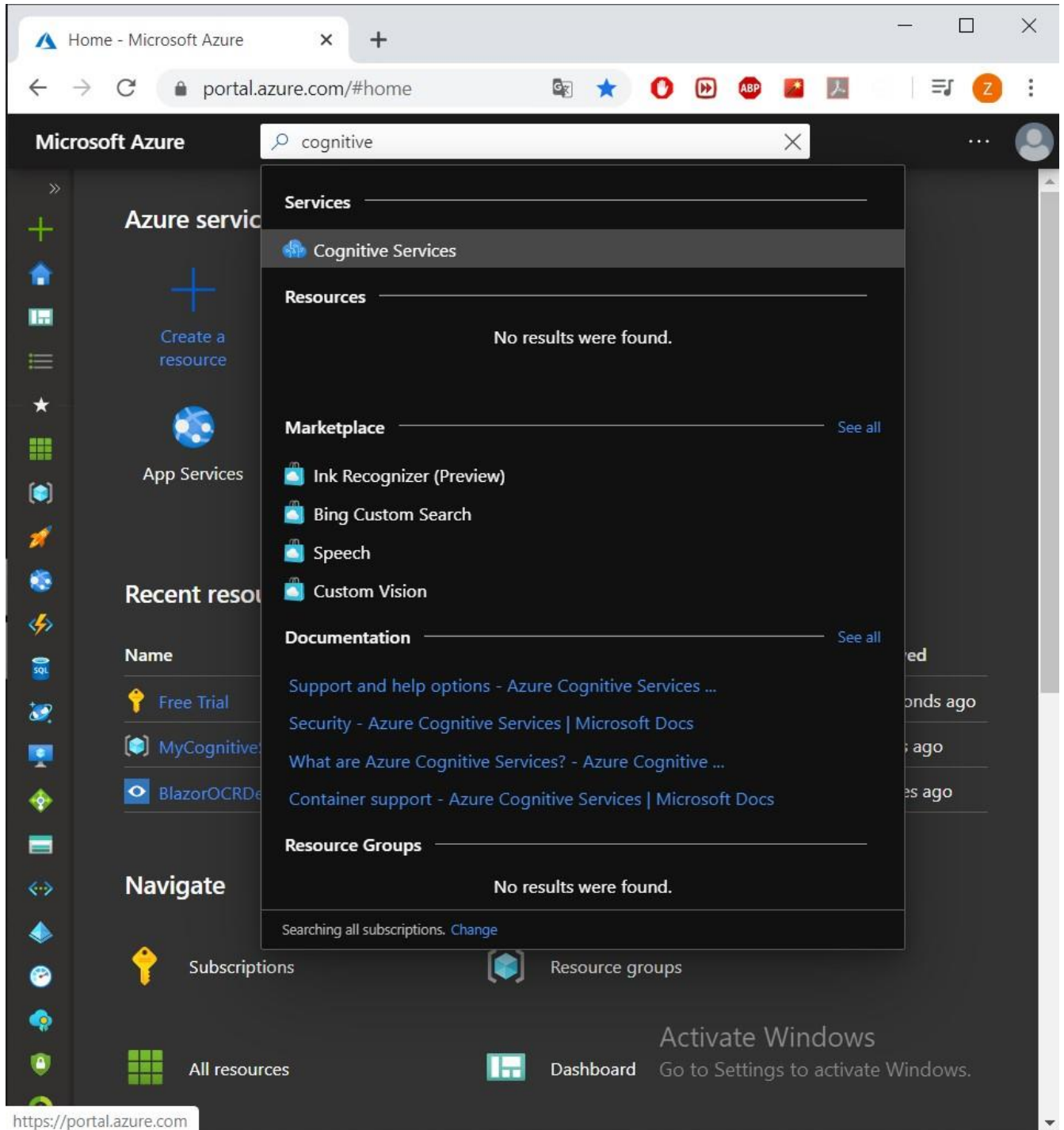
3. Cerințe imagini

API-ul de recunoaștere optică a caracterelor va funcționa doar pe imagini care îndeplinesc cerințele următoare:

- Formatul imaginii trebuie să fie JPEG, PNG, GIF sau BMP.
- Mărimea imaginii trebuie să fie între 50 x 50 și 4200 x 4200 pixeli.
- Dimensiunea imaginii să fie mai mică de 4 MB.
- Textul din imagine să poată fi rotit cu orice multiplu de 90 de grade plus un unghi de până la 40 de grade.

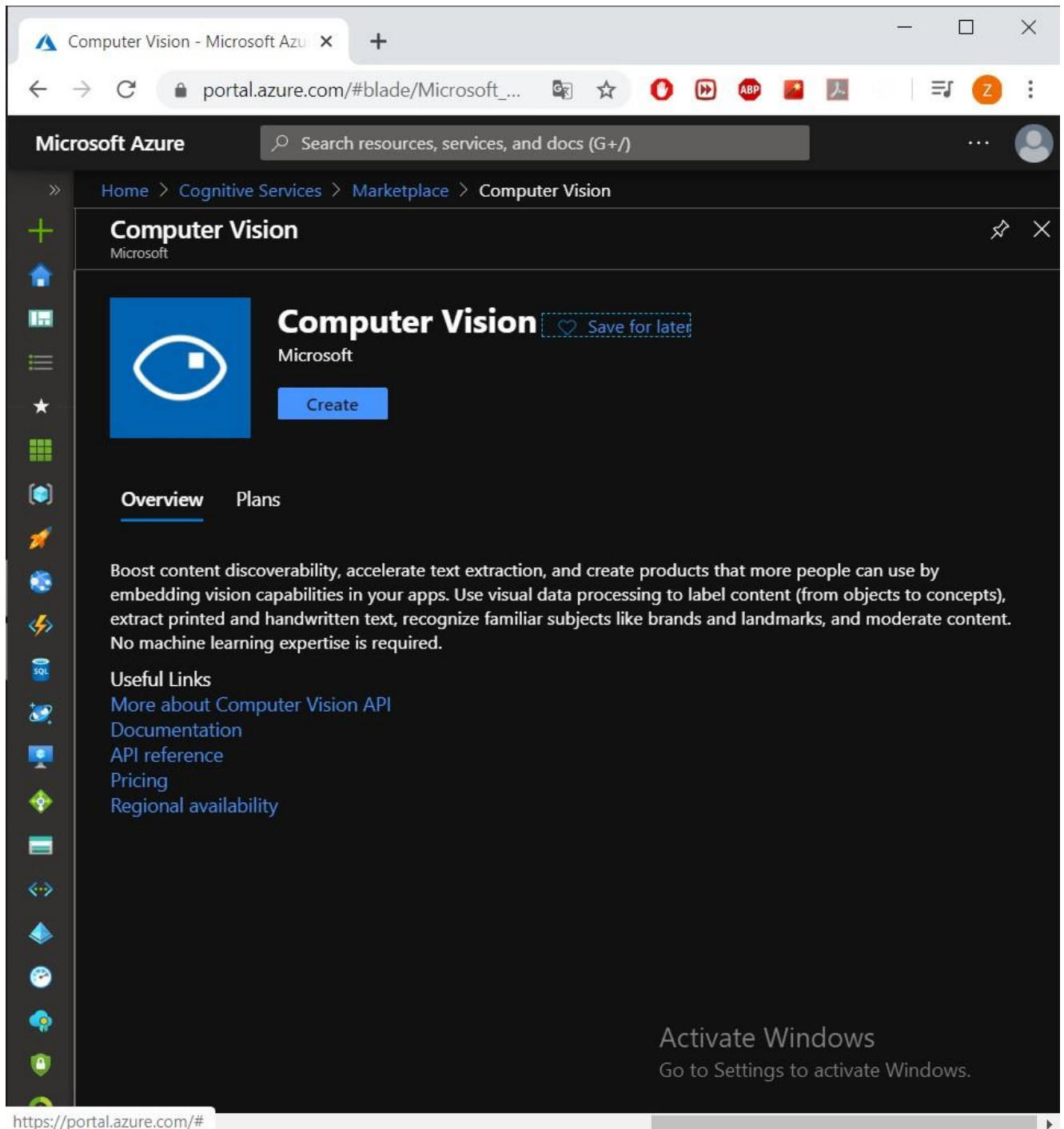
4. Crearea resursei Azure Computer Vision Cognitive Service

Se conectează la platforma Azure și în bara de căutare, introducem Cognitive Services, pe care o și selectăm.



Pe pagina următoare, facem click pe butonul “Add”. Acesta va deschide pagina marketplace a serviciilor cognitive.

Căutăm Computer Vision în bara de căutare și selectăm rezultatul. Se va deschide pagina API Computer Vision. Selectăm butonul “Create” pentru a crea o resursă Computer Vision nouă.



În pagina Create, completăm detaliile indicate mai jos.

- Name: Dăm un nume unic resursei noastre.
- Subscription: Selectăm tipul de subscripție.
- Pricing tier: Selectăm nivelul de preț după preferințe.
- Resource Group: Selectăm un grup de resurse existent sau creăm unul nou.

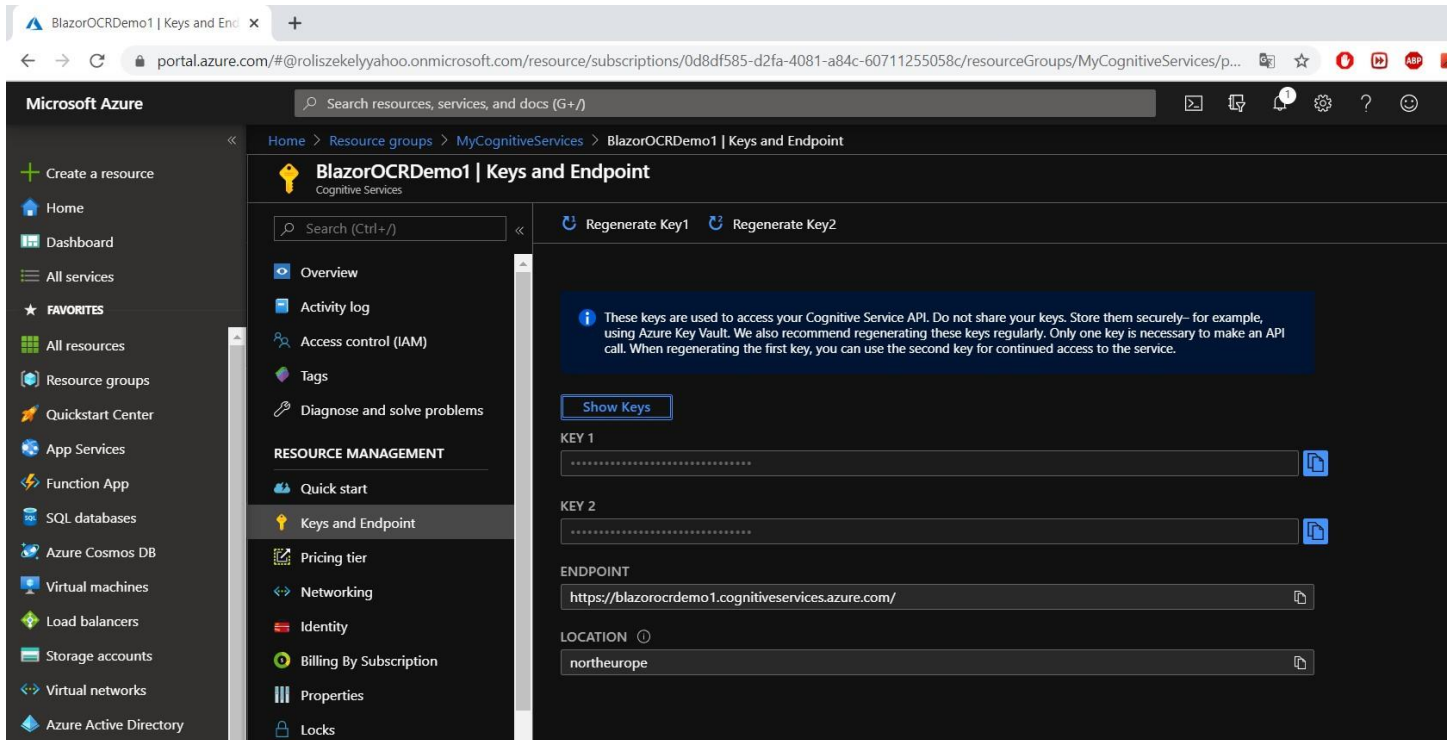
Selectăm butonul „Create”.

The screenshot shows the Microsoft Azure portal interface for creating a new resource. The breadcrumb navigation at the top indicates the path: Home > Cognitive Services > Marketplace > Computer Vision > Create. The main form is titled 'Create' and 'Computer Vision'. It contains the following fields:

- Name ***: A text input field containing 'BlazorOCRDemo1' with a green checkmark on the right.
- Subscription ***: A dropdown menu showing 'Free Trial'.
- Location ***: A dropdown menu showing '(Europe) North Europe'.
- Pricing tier (View full pricing details) ***: A dropdown menu showing 'S1 (10 Calls per second)'.
- Resource group ***: A dropdown menu showing '(New) MyCognitiveServices'. Below this dropdown is a link labeled 'Create new'.

At the bottom of the form, there is a blue 'Create' button and a link for 'Automation options'. An 'Activate Windows' watermark is visible in the bottom right corner of the page.

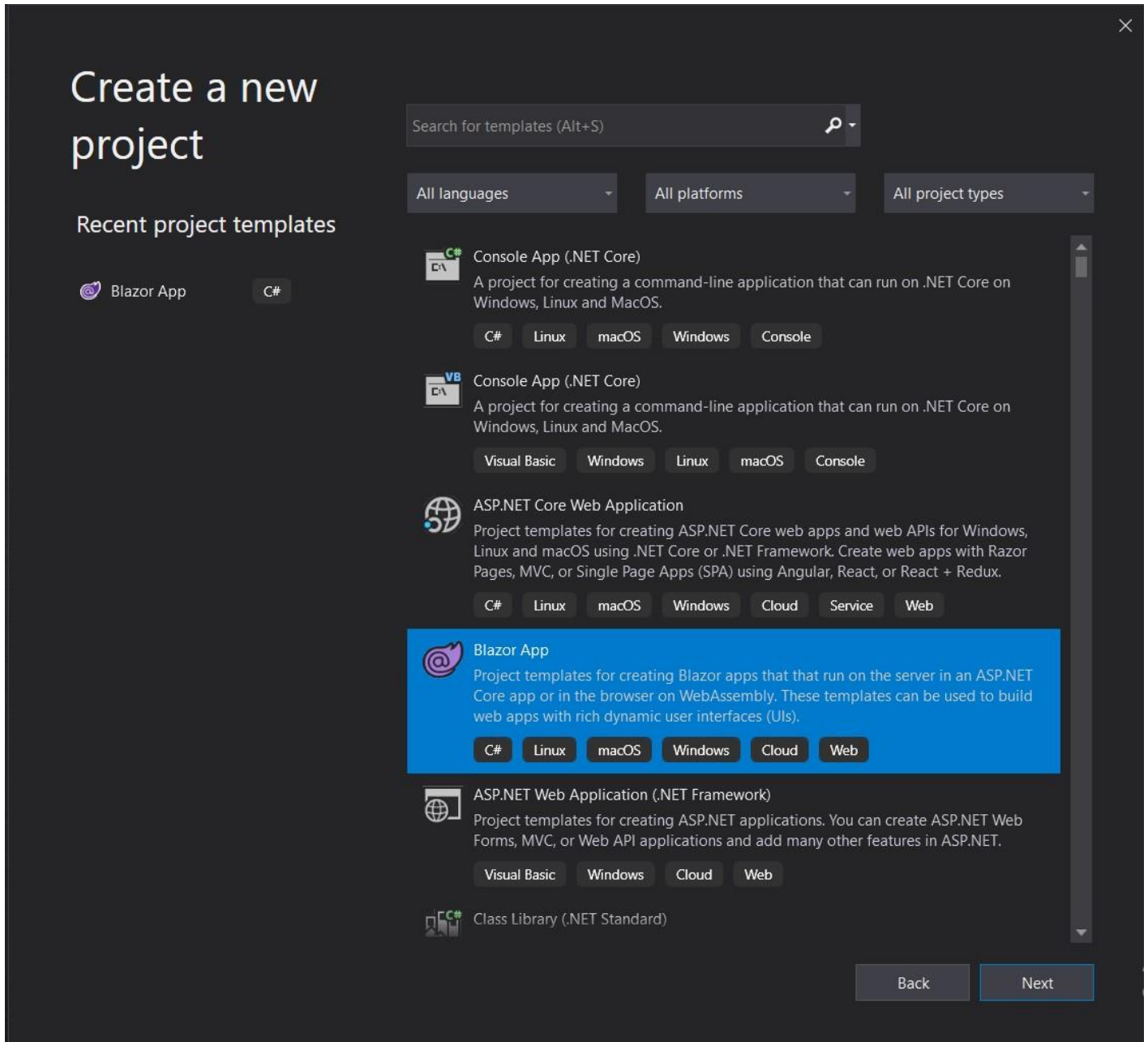
După ce resurse este implementată cu succes, selectăm butonul “Go to resource”. Aici vom vedea Cheia și endpoint-ul noii resurse Computer Vision.



Notăți-vă Cheia și endpoint-ul. Acestea se vor folosi într-o parte mai avansată a creării aplicației pentru a invoca API-ul OCR de Computer Vision din codul .NET. Valorile sunt mascate pentru confidențialitate.

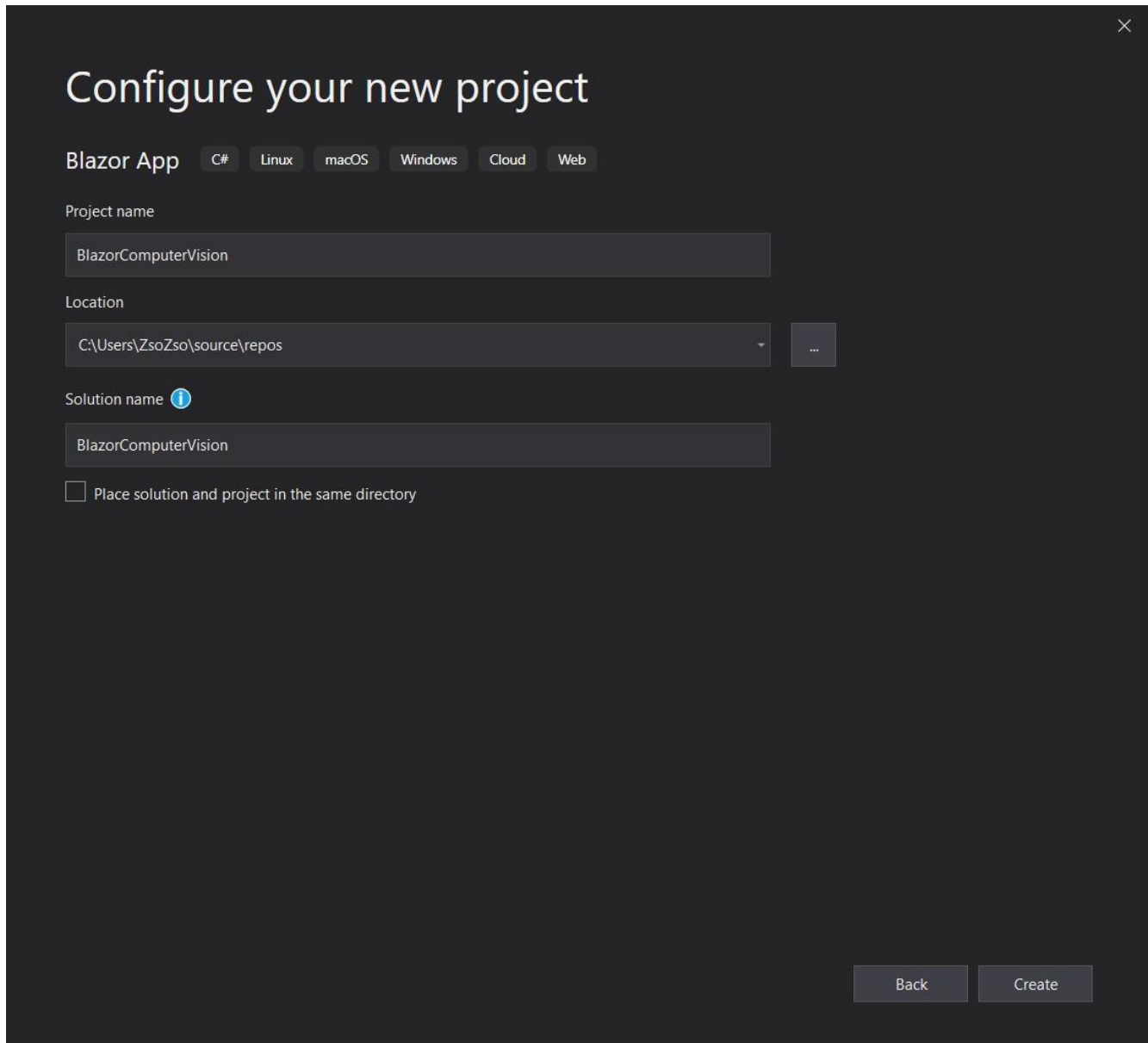
5. Crearea aplicației Blazor pe partea de server

Vom deschide Visual Studio și selectăm “Create a new project”. Selectăm “Blazor App” după selectăm butonul “Next”.



În următoarea fereastră, vom numi proiectul „BlazorComputerVision” și vom selecta butonul “Create”.

Fereastra următoare ne va cere să selectăm tipul de aplicație Blazor. Selectăm “Blazor Server App”, după care selectăm “Create” pentru a crea o aplicație Blazor nouă pe partea de server.



Configure your new project

Blazor App C# Linux macOS Windows Cloud Web

Project name

BlazorComputerVision

Location

C:\Users\ZsoZso\source\repos

Solution name ⓘ

BlazorComputerVision

☐ Place solution and project in the same directory

Back Create

Instalarea librăriei API Computer Vision

Vom instala librăria API Azure Computer Vision ce ne va oferi modelele prelucrate pentru a putea gestiona răspunsul API-ului REST Computer Vision.

Pentru a instala pachetul, vom naviga către Tools > NuGet Package Manager > Package Manager Console. Ne va deschide consola Package Manager.

Vom rula următoarea comandă:

```
Install-Package Microsoft.Azure.CognitiveServices.Vision.ComputerVision -  
Version 5.0.0
```

Crearea modelelor

Click-dreapta pe proiectul “BlazorComputerVision” și selectăm Add > New Folder. Numim folderul “Models”. Click-dreapta din nou pe fișierul “Models” și selectăm Add > Class pentru a adăuga un fișier clasă nou. Vom numi clasa “LanguageDetails.cs” și selectăm Add.

Deschidem “LanguageDetails.cs” și adăugăm următorul cod:

```
namespace BlazorComputerVision.Models  
{  
    public class LanguageDetails  
    {  
        public string Name { get; set; }  
        public string NativeName { get; set; }  
        public string Dir { get; set; }  
    }  
}
```

În mod similar, adăugăm un fișier clasă “AvailableLanguage.cs” în care adăugăm următorul cod:

```
using System.Collections.Generic;

namespace BlazorComputerVision.Models
{
    public class AvailableLanguage
    {
        public Dictionary<string, LanguageDetails> Translation { get; set; }
    }
}
```

În final, vom adăuga o clasă DTO (Data Transfer Object) pentru a trimite date înapoi client-ului.

Adăugăm o clasă nouă “OcrResultDTO.cs” și inserăm codul:

```
namespace BlazorComputerVision.Models
{
    public class OcrResultDTO
    {
        public string Language { get; set; }

        public string DetectedText { get; set; }
    }
}
```

Crearea serviciului Computer Vision

Click-dreapta pe folderul “BlazorComputerVision/Data” și selectăm Add > Class pentru a adăuga o clasă nouă. Vom numi fișierul “ComputerVisionService.cs” și selectăm „Add”.

Deschidem fișierul “ComputerVisionService.cs” și inserăm următorul cod:

```
using BlazorComputerVision.Models;
using Microsoft.Azure.CognitiveServices.Vision.ComputerVision.Models;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;

namespace BlazorComputerVision.Data
{
    public class ComputerVisionService
    {
        static string subscriptionKey;
        static string endpoint;
        static string uriBase;

        public ComputerVisionService()
        {
            subscriptionKey = "b993f3afb4e04119bd8ed37171d4ec71";
            endpoint = "https://blazorocrdemo.cognitiveservices.azure.com/";
            uriBase = endpoint + "vision/v2.1/ocr";
        }

        public async Task<OcrResultDTO> GetTextFromImage(byte[] imageFileBytes)
        {
            StringBuilder sb = new StringBuilder();
            OcrResultDTO ocrResultDTO = new OcrResultDTO();
```

```

try
{
    string JsonResult = await ReadTextFromStream(imageFileBytes);

    OcrResult ocrResult =
JsonConvert.DeserializeObject<OcrResult>(JsonResult);

    if (!ocrResult.Language.Equals("unk"))
    {
        foreach (OcrLine ocrLine in ocrResult.Regions[0].Lines)
        {
            foreach (OcrWord ocrWord in ocrLine.Words)
            {
                sb.Append(ocrWord.Text);
                sb.Append(' ');
            }
            sb.AppendLine();
        }
    }
    else
    {
        sb.Append("This language is not supported.");
    }
    ocrResultDTO.DetectedText = sb.ToString();
    ocrResultDTO.Language = ocrResult.Language;
    return ocrResultDTO;
}
catch
{
    ocrResultDTO.DetectedText = "Error occurred. Try again";
    ocrResultDTO.Language = "unk";
    return ocrResultDTO;
}
}

```

```

static async Task<string> ReadTextFromStream(byte[] byteData)

```

```

    {
        try
        {
            HttpClient client = new HttpClient();
            client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",
subscriptionKey);

            string requestParameters = "language=unk&detectOrientation=true";
            string uri = uriBase + "?" + requestParameters;
            HttpResponseMessage response;

            using (ByteArrayContent content = new ByteArrayContent(byteData))
            {
                content.Headers.ContentType = new
MediaTypeHeaderValue("application/octet-stream");
                response = await client.PostAsync(uri, content);
            }

            string contentString = await response.Content.ReadAsStringAsync();
            string result = JToken.Parse(contentString).ToString();
            return result;
        }
        catch (Exception e)
        {
            return e.Message;
        }
    }

    public async Task<AvailableLanguage> GetAvailableLanguages()
    {
        string endpoint = "https://api.cognitive.microsofttranslator.com/languages?api-
version=3.0&scope=translation";
        var client = new HttpClient();
        using (var request = new HttpRequestMessage())
        {
            request.Method = HttpMethod.Get;
            request.RequestUri = new Uri(endpoint);

```

```

        var response = await client.SendAsync(request).ConfigureAwait(false);
        string result = await response.Content.ReadAsStringAsync();

        AvailableLanguage deserializedOutput =
        JsonConvert.DeserializeObject<AvailableLanguage>(result);

        return deserializedOutput;
    }
}
}
}

```

În constructorul clasei am inițializat cheia și endpoint-ul URL pentru API-ul OCR.

În metoda “ReadTextFromStream” vom crea un request new “HttpRequestMessage”. Acest request Http este un request Post. Vom trece cheia subscripției Azure în header-ul requestului. API-ul va returna un obiect JSON, acesta conținând atât fiecare cuvânt din imagine cât și limba textului.

Metoda “GetTextFromImage” va accepta datele imaginii ca un șir de biți și va reuturna un obiect de tip „OcrResultDTO”. Vom apela metoda “ReadTextFromStream” ce va deserializa răspunsul într-un obiect de tip „OcrResult”. Apoi vom forma propoziția prin iterarea obiectului “OcrWord”.

Metoda „GetAvailableLanguages” va returna o listă cu toate limbile suportate (25) de către API-ul de traducere a textului. Setăm requestul URI și creăm requestul “HttpRequestMessage”, care va fi un request Get. Acest request URL va returna un obiect JSON care va fi deserializat într-un obiect de tipul “AvailableLanguage”.

De ce trebuie să invocăm lista limbilor suportate?

API-ul OCR returnează codul limbii detectate (ex: en pentru Engleză, de pentru Germană, etc.). Însă, codul limbii nu poate fi prezentat în UI pentru că nu este user-friendly. Așadar, vom avea nevoie de un dicționar care va căuta numele limbii corespunzătoare codului limbii.

API-ul Azure Computer Vision OCR suportă 25 de limbi, acestea fiind un subset al API-ului Azure Translate Text. Cum nu există un endpoint API dedicat cererii listei limbilor suportate, folosim endpointul API-ului de Translate Text.

Vom crea un dicționar de căutare al limbii folosind răspunsul JSON din apelul API și vom filtra rezultatul pe baza codului de limbă returnat de către OCR API.

Instalarea pachetului BlazorInputFile NuGet

BlazorInputFile este o componentă de input pentru aplicații Blazor ce ne dă abilitatea de încărcare una sau mai multe fișiere într-o aplicație Blazor.

Deschidem fișierul “BlazorComputerVision.csproj” și adăugăm o dependență pentru pachetul “BlazorInputFile”:

```
<ItemGroup>
  <PackageReference Include="BlazorInputFile" Version="0.1.0-preview-00002" />
</ItemGroup>
```

Deschidem fișierul “BlazorComputerVision/Pages/_Host.cshtml” și adăugăm o referință fișierului JavaScript din pachet, adăugând următoarea linie de cod în secțiunea “<head>”:

```
<script src="_content/BlazorInputFile/inputfile.js"></script>
```

Adăugăm următoarea linie de cod în fișierul “_Imports.razor” :

```
@using BlazorInputFile
```

6. Configurarea serviciului

Pentru a face serviciul valabil tuturor componentelor, trebuie să îl configurăm pe aplicația pe partea de server.

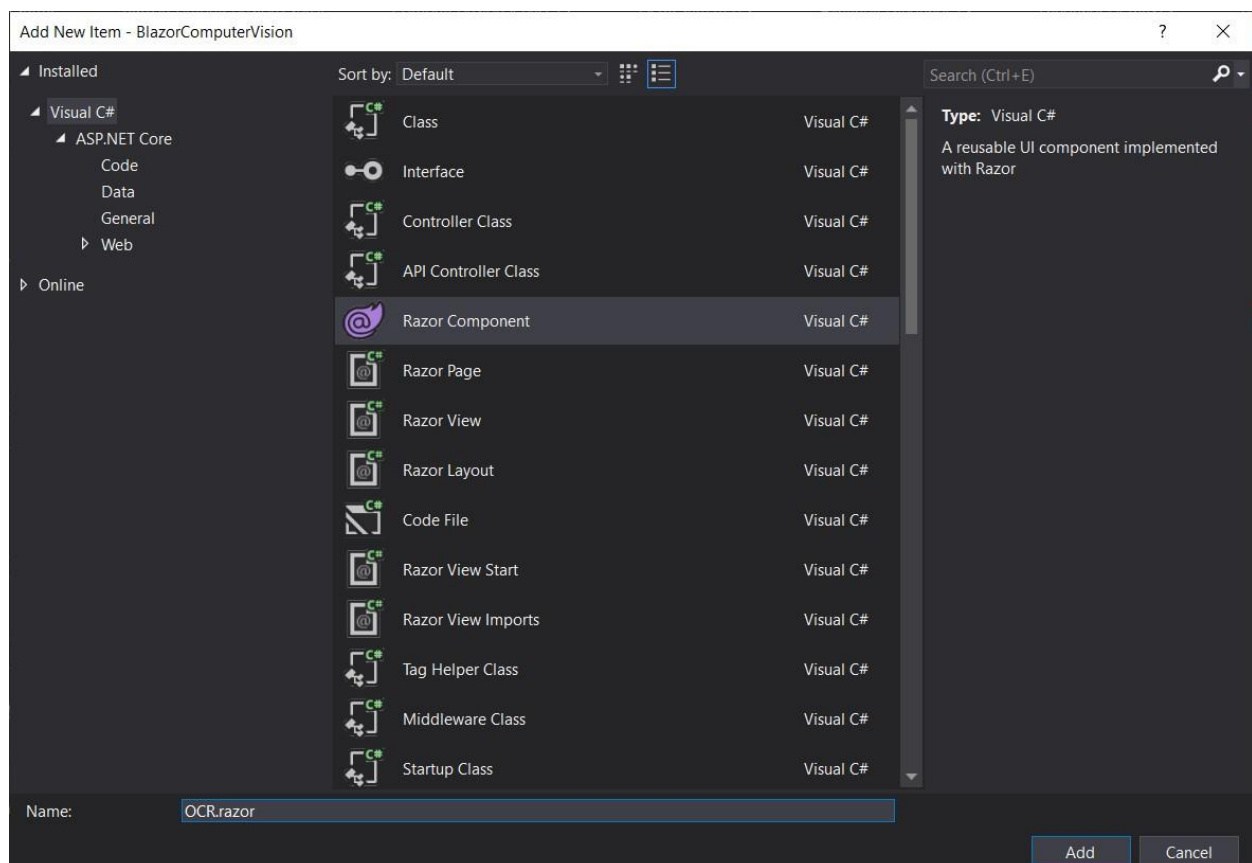
Deschidem fișierul “Startup.cs” și adăugăm următoarea linie de cod în metoda “ConfigureServices” din clasa Startup:

```
services.AddSingleton<ComputerVisionService>();
```

7. Crearea componentei UI Blazor

Vom adăuga o pagină Razor în folderul “BlazorComputerVision/Pages”. Ca și default, aplicația ne dă paginile “Counter” și “Fetch Data”. Aceste pagini default nu afectează aplicația dar pentru a simplifica procesul creării acestei aplicații, le vom șterge din folderul “BlazorComputerVision/Pages”.

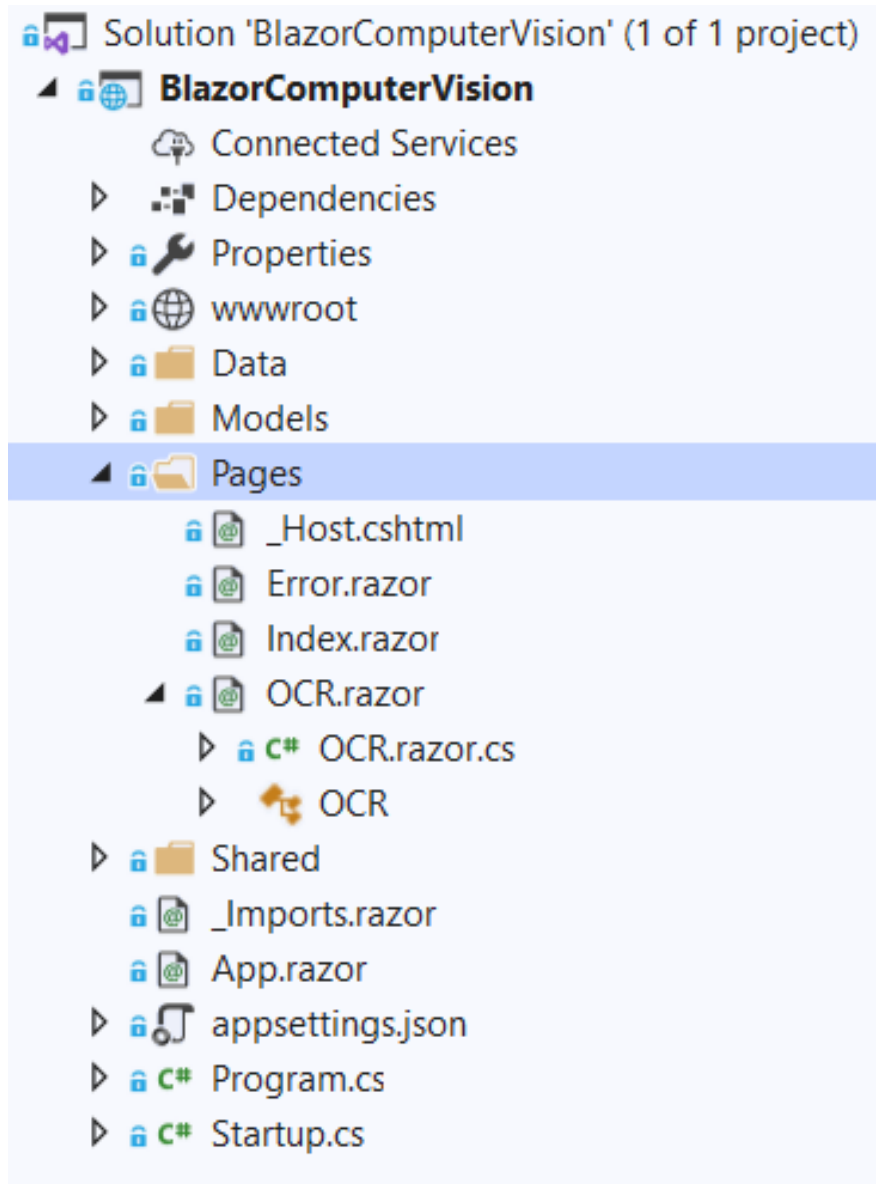
Click-dreapta pe folderul “BlazorComputerVision/Pages” și selectăm Add > New Item. Un dialog box “Add New Item” va apărea în care selectăm “Visual C#” din panoul din stânga, apoi selectăm “Razor Component” din panoul templates, numind componenta “OCR.razor” după care selectăm “Add”.



În „OCR.razor”, vom adăuga un fișier de tip code-behind care va ține codul și prezentarea separat. Acest aspect ne va permite o mentenanță mai ușoară a aplicației.

Click-dreapta pe folderul “BlazorComputerVision/Pages” după care selectăm Add > Class.

Numim clasa “OCR.razor.cs”. Framework-ul Blazor este suficient de inteligent pentru a eticheta acest fișier clasă fișierului razor.



Blazor UI component code behind

Deschidem fișierul „OCR.razor.cs” și inserăm codul următor:

```
using Microsoft.AspNetCore.Components;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.IO;
using BlazorComputerVision.Models;
using BlazorInputFile;
using BlazorComputerVision.Data;

namespace BlazorComputerVision.Pages
{
    public class OCRModel : ComponentBase
    {
        [Inject]
        protected ComputerVisionService computerVisionService { get; set; }

        protected string DetectedTextLanguage;
        protected string imagePreview;
        protected bool loading = false;
        byte[] imageFileBytes;

        const string DefaultStatus = "Maximum size allowed for the image is 4 MB";
        protected string status = DefaultStatus;

        protected OcrResultDTO Result = new OcrResultDTO();
        private AvailableLanguage availableLanguages;
        private Dictionary<string, LanguageDetails> LanguageList = new
Dictionary<string, LanguageDetails>();
        const int MaxFileSize = 4 * 1024 * 1024; // 4MB
```

```

protected override async Task OnInitializedAsync()
{
    availableLanguages = await computerVisionService.GetAvailableLanguages();
    LanguageList = availableLanguages.Translation;
}

protected async Task ViewImage(IFileListEntry[] files)
{
    var file = files.FirstOrDefault();
    if (file == null)
    {
        return;
    }
    else if (file.Size > MaxFileSize)
    {
        status = $"The file size is {file.Size} bytes, this is more than the allowed limit of {MaxFileSize} bytes.";
        return;
    }
    else if (!file.Type.Contains("image"))
    {
        status = "Please upload a valid image file";
        return;
    }
    else
    {
        var memoryStream = new MemoryStream();
        await file.Data.CopyToAsync(memoryStream);
        imageFileBytes = memoryStream.ToArray();
        string base64String = Convert.ToBase64String(imageFileBytes, 0,
imageFileBytes.Length);

        imagePreview = string.Concat("data:image/png;base64,", base64String);
        memoryStream.Flush();
        status = DefaultStatus;
    }
}

```

```

    }

    protected private async Task GetText()
    {
        if (imageFileBytes != null)
        {
            loading = true;
            Result = await computerVisionService.GetTextFromImage(imageFileBytes);
            if (LanguageList.ContainsKey(Result.Language))
            {
                DetectedTextLanguage = LanguageList[Result.Language].Name;
            }
            else
            {
                DetectedTextLanguage = "Unknown";
            }
            loading = false;
        }
    }
}

```

În această clasă injectăm serviciul “ComputerVisionService”.

Metoda “OnInitializedAsync” este o metodă de lifecycle în Blazor, invocată când o componentă este inițializată. Invocăm metoda “GetAvailableLanguages” din serviciu, în interiorul metodei “OnInitializedAsync” după care, inițializăm variabila “LanguageList” care este un dicționar ce reține detaliile limbilor disponibile.

În metoda “ViewImage”, vom verifica dacă fișierul încărcat este doar o imagine și dacă mărimea este mai mică de 4 MB. Vom transfera imaginea încărcată în memory stream apoi o vom converti într-un șir de biți.

Pentru a seta preview-ul imaginii, vom converti imaginea dintr-un șir de biți într-un string encodat în bază 64. Metoda “GetText” va invoca metoda “GetTextFromImage” din serviciu și ne va da șirul de biți ca și un argument. Căutăm numele limbii din dicționar bazându-ne pe codul

limbii returnat din serviciu. Dacă nici un cod de limbă nu este valabil, limba se va seta ca fiind unknown.

Template-ul componentei UI Blazor

Deschidem fișierul “OCR.razor” și adăugăm următorul cod:

```
<h2>Optical Character Recognition (OCR) Using Blazor and Azure Computer Vision  
Cognitive Service</h2>

<div class="row">
    <div class="col-md-5">
        <textarea disabled class="form-control" rows="10"
cols="15">@Result.DetectedText</textarea>
        <hr />
        <div class="row">
            <div class="col-sm-5">
                <label><strong> Detected Language :</strong></label>
            </div>
            <div class="col-sm-6">
                <input disabled type="text" class="form-control" value="@DetectedTextLanguage" />
            </div>
        </div>
    </div>
    <div class="col-md-5">
        <div class="image-container">
            <img class="preview-image" src=@imagePreview>
        </div>
        <InputFile OnChange="ViewImage" />
        <p>@status</p>
        <hr />
        <button disabled="@loading" class="btn btn-primary btn-lg" @onclick="GetText">
            @if (loading)
            {
```

```

        <span class="spinner-border spinner-border-sm mr-1"></span>
    }
    Extract Text
</button>
</div>
</div>

```

Aici am definit ruta componentei. Am moștenit clasa “OCRModel” care ne permite să accesăm proprietățile și metoda clasei din template.

Folosim Bootstrap pentru designul UI. Avem o zonă pentru text ce afișează textul detectat și un text box ce afișează limba detectată. Eticheta de imagine este utilizată pentru a previzualiza imaginea după încărcarea ei. Componenta “<InputFile>” ne va permite încărcarea unei imagini și invocarea metodei “ViewImage” în timp ce încărcăm imaginea.

Adăugarea stilului componentei

În fișierul “BlazorComputerVision\wwwroot\css\site.css” adăugăm următoarea definiție de stil:

```

.preview-image {
    max-height: 300px;
    max-width: 300px;
}
.image-container {
    display: flex;
    padding: 15px;
    align-content: center;
    align-items: center;
    justify-content: center;
    border: 2px dashed skyblue;
}

```


Adăugare link în meniul Navigation

Ultimul pas este să adăugăm link-ul componentei OCR în meniul de navigație.

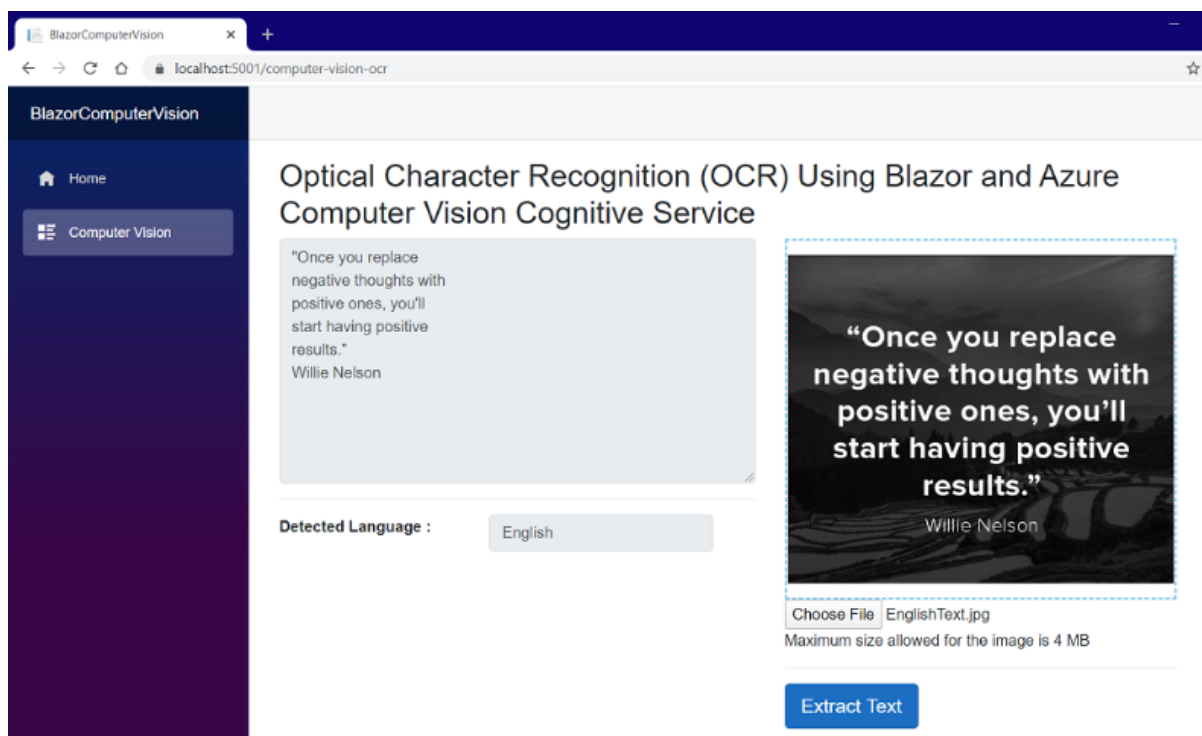
Deschidem fișierul “BlazorComputerVision/Shared/NavMenu.razor” și adăugăm următorul cod:

```
<li class="nav-item px-3">
    <NavLink class="nav-link" href="computer-vision-ocr">
        <span class="oi oi-list-rich" aria-hidden="true"></span> Computer Vision
    </NavLink>
</li>
```

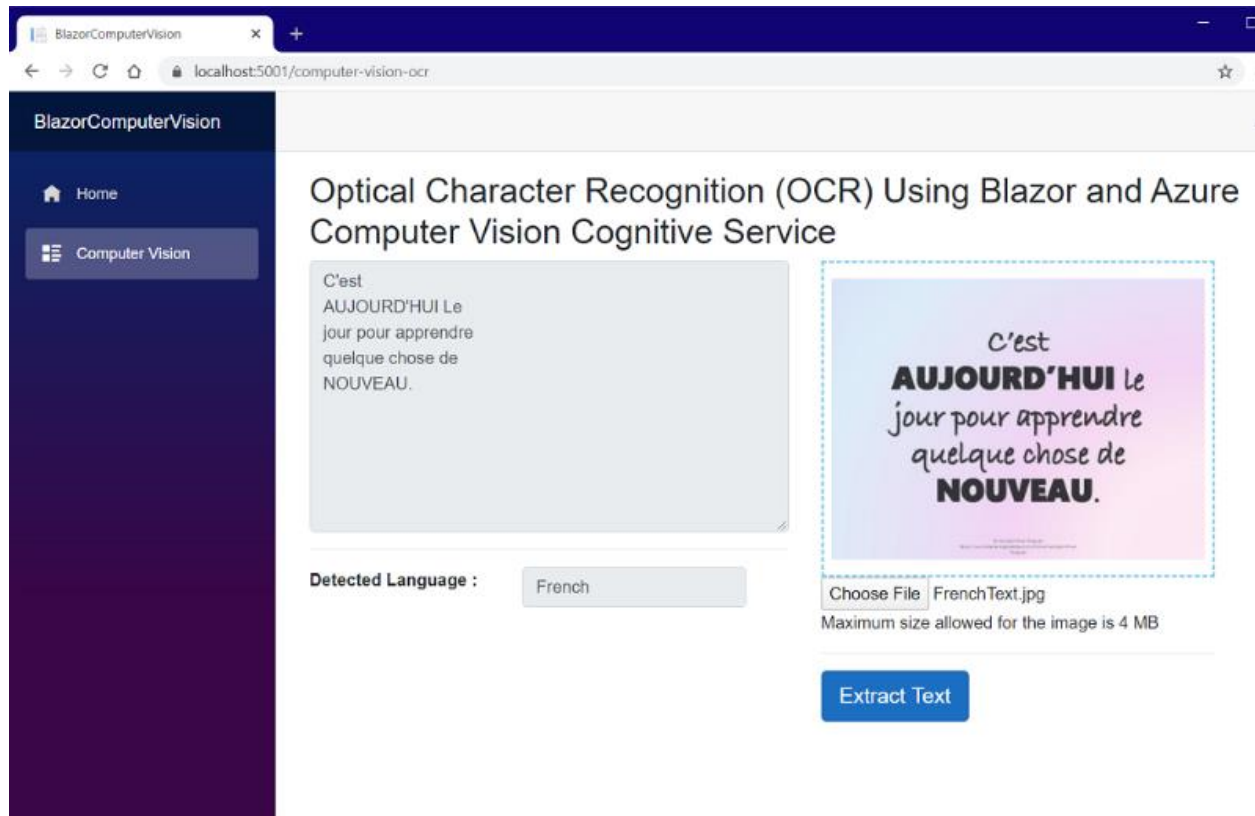
Ștergem linkurile de navigare pentru componentele “Counter” și “Fetch-data” într-ucât nu vor fi folosite în această aplicație.

8. Executarea demo-ului

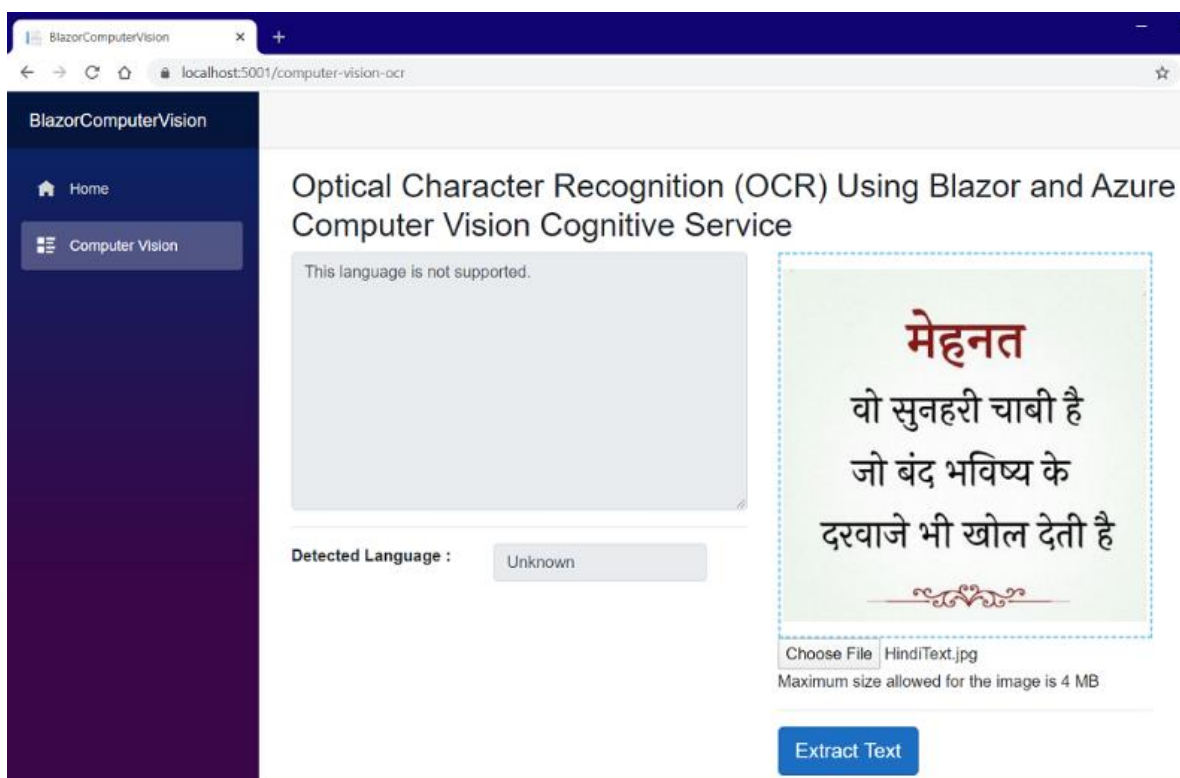
Apăsăm F5 pentru a porni aplicația. Selectăm butonul de “Computer Vision” în meniul de navigare din partea stângă. Pe pagina următoare, încărcăm o imagine cu text și selectăm butonul “Extract Text”. Vom vedea textul extras în zona pentru text extras în partea stângă împreună cu limba textului detectată.



Vom încerca să încărcăm o imagine cu text francez. Putem observa textul extras și limba detectată.



Dacă încercăm să încărcăm o imagine cu o limbă nesuportată, vom primi o eroare.



9. Rezumat

În concluzie, am creat o aplicație de recunoaștere optică a caracterelor (OCR) folosind Blazor și serviciul de Computer Vision din cadrul Azure Cognitive Services. Am adăugat un feature de încărcare unei imagini folosind componenta “BlazorInputFile”. Aplicația poate extrage textul printat din imaginea încărcată și poate recunoaște limba textului. API-ul OCR al Computer Vision-ului este folosit, ce poate recunoaște 25 de limbi.