

WEB PROGRAMLAMA RAPOR

Ad: Metin

Soyad: Aydın

Numara: G221210383

Grup: 2B

Github: <https://github.com/metinaydinn/Web-Programlama-Proje-2024-2025>



1. Proje Tanıtımı

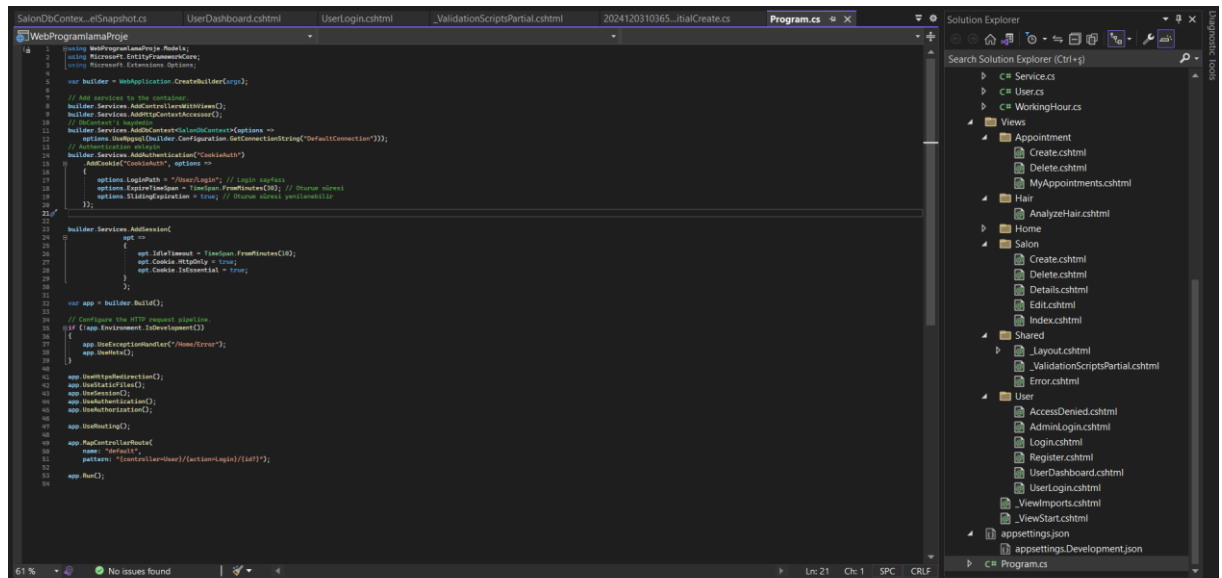
Proje, Web Programlama dersi kapsamında geliştirilmiştir. Bu proje, kullanıcı dostu bir arayüz ve etkili bir veritabanı yapısı ile, web tabanlı bir uygulama olarak tasarlanmıştır. Projenin temel amacı, kullanıcılara kolay ve etkili bir çevrimiçi deneyim sunmaktır.

2.Proje Amaçları:

- Kullanıcı dostu bir web arayüzü oluşturmak.
- Etkin bir veritabanı yapısı ile veri yönetimini sağlamak.
- Modern web teknolojileri kullanarak güvenli ve performanslı bir uygulama geliştirmek.

PROGRAM.CS

Program.cs, uygulamanın nasıl başlatılacağını, hangi servislerin yükleneceğini ve gelen/giden HTTP isteklerinin nasıl işleneceğini tanımlar.



```
1 using Microsoft.AspNetCore.Builder;
2 using Microsoft.EntityFrameworkCore;
3 using Microsoft.Extensions.Options;
4
5 var builder = WebApplication.CreateBuilder(args);
6
7 // Add services to the container
8 builder.Services.AddControllersWithViews();
9 builder.Services.AddHttpContextAccessor();
10 // Microsoft.AspNetCore.Authentication.Cookies
11 builder.Services.AddCookieAuthentication(options =>
12 {
13     options.LoginPath = "/User/Login/"; // login sayfası
14     options.ExpireTimeSpan = TimeSpan.FromMinutes(30); // 30 dakika
15     options.SlidingExpiration = true; // Otomatik yenileme
16 });
17
18 builder.Services.AddSession(options =>
19 {
20     options.Cookie.IsEssential = true;
21 });
22
23 var app = builder.Build();
24
25 // Configure the HTTP request pipeline
26 if (app.Environment.IsDevelopment())
27 {
28     app.UseDeveloperExceptionPage();
29 }
30 else
31 {
32     app.UseExceptionHandler("/Home/Error");
33     app.UseHsts();
34 }
35
36 app.UseHttpsRedirection();
37 app.UseStaticFiles();
38 app.UseRouting();
39 app.UseCookieAuthentication();
40 app.UseSession();
41
42 app.MapControllerRoute(
43     name: "default",
44     pattern: "{controller=Home}/{action=Index}/{id?}");
45
46 app.Run();
```

The screenshot shows the Visual Studio IDE with the Program.cs file open. The code defines the web application's startup, including service registration, cookie authentication, session management, and the HTTP request pipeline. The Solution Explorer on the right shows the project structure, including Views, Shared, and User folders.

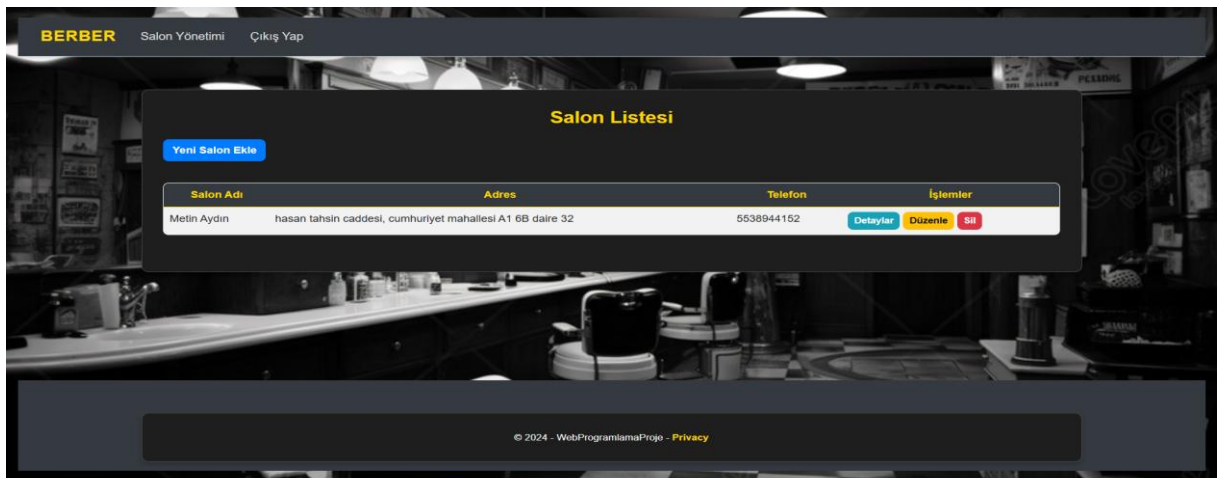
LINQ Kullanımı Örneği:

Projenin bir çok kısmında özellikle controller kısımlarında LINQ kullanımıyla projede yer alan veri tabanındaki uygun verilerden sorgulamalar yapıldı.

```
WebProgramlamaProje WebProgramlamaProje.Controllers.SalonController Edit(Salon salon)
174      salon.Services = new List<Service>();
175  }
176  }
177  return View(salon);
178  }
179  }
180  }
181  // Salon düzenleme (POST)
182  [HttpPost]
183  public IActionResult Edit(Salon salon)
184  {
185      if (!IsAdmin())
186      {
187          TempData["Error"] = "Bu sayfaya erişim yetkiniz yok!";
188          return RedirectToAction("Login", "User"); // Giriş ekranına yönlendirme
189      }
190      if (ModelState.IsValid)
191      {
192          // Veritabanından mevcut salon ve ilişkili hizmetleri yükle
193          var existingSalon = _context.Salons
194              .Include(s => s.Services)
195              .Include(s => s.Employees) // Çalışanları yükle
196              .FirstOrDefault(s => s.Id == salon.Id);
197      }
198      if (existingSalon == null)
199      {
200          return NotFound($"Salon ID {salon.Id} bulunamadı.");
201      }
202      // Salon temel bilgilerini güncelle
203      existingSalon.Name = salon.Name;
204      existingSalon.Address = salon.Address;
205      existingSalon.Phone = salon.Phone;
206      // Mevcut hizmetleri güncelle veya yeni hizmetleri ekle
207      foreach (var service in salon.Services)
208      {
209          if (service.Id == 0)
210          {
211              // Yeni hizmet ekle
212              existingSalon.Services.Add(new Service
213              {
214                  Name = service.Name,
215                  Price = service.Price,
216                  Duration = service.Duration,
217                  SalonId = salon.Id
218              });
219          }
220          else
221          {
222              // Mevcut hizmeti bul ve güncelle
223              var existingService = existingSalon.Services.FirstOrDefault(s => s.Id == service.Id);
224              if (existingService != null)
225              {
226                  existingService.Name = service.Name;
227                  existingService.Price = service.Price;
228                  existingService.Duration = service.Duration;
229              }
230          }
231      }
232      return RedirectToAction("Index");
233  }
```

Admin Paneli

Uygulamada bulunan admin paneli sayesinde admin berberin hizmetlerini, çalışanlarını ve ücretlerini gibi bölümleri düzenleyebilir, görüntüleyebilir veya salonu silebilir.



Kayıt Ol Sayfası

Yeni kayıt olmak isteyen kullanıcıların kayıt olma sayfası. Önceden kayıtlı E-postaların kontrol edildiği veritabanından kayıtlı e-posta varsa bu e-posta zaten kayıtlı kontrollerinin yapıldığı sayfa.

BERBER

Giriş Yap

Kayıt Ol

Ad Soyad:

E-posta:

Şifre:

Telefon:

Kayıt Ol

Kayıtlı Kullanıcı Sayfası

Sisteme kayıtlı kullanıcıların randevu alabildiği, önceden aldığı randevuları sorgulayabildiği ve yapay zekâ desteğiyle saç önerisi alabildiği sayfa.

BERBER

Kullanıcı Paneli

Çıkış Yap

Kullanıcı Paneli

Randevu Al

Randevularımı Görüntüle

Saç Önerisi al

Çıkış Yap

© 2024 - WebProgramlamaProje - Privacy

RESTAPI Kullanımı

- REST API'de POST metodunun bir örneği olarak, aşağıdaki metodlar RESTful bir API davranışı gösteriyor:
 - DeleteService
 - DeleteEmployee
 - DeleteAppointment

Bu metodlar, istemci tarafında (örneğin, JavaScript/AJAX çağrıları) gönderilen JSON verilerini almak için FromBody kullanıyor.

Bu, bir REST API'nin DELETE veya POST çağrısına benzer bir yapıdır.

```
//Meden FromBody kullanıyoruz Bu metod, bir HttpPost isteği olarak tanımlandığı için, genellikle istemci tarafında (örneğin bir AJAX çağrısı) gönderilen JSON verilerini alır.
[HttpPost]
OrdersController.DeleteEmployee([FromBody] int employeeId)
{
    if (!IsAdmin())
    {
        TempData["Error"] = "Bu sayfaya erişim yetkiniz yok!";
        return RedirectToAction("Login", "User"); // Giriş ekranına yönlendirme
    }
    var employee = _context.Employees.FirstOrDefault(e => e.Id == employeeId);

    if (employee != null)
    {
        _context.Employees.Remove(employee);
        _context.SaveChanges();

        return Json(new { success = true, message = "Çalışan başarıyla silindi." });
    }

    return Json(new { success = false, message = "Çalışan bulunamadı." });
}

[HttpPost]
OrdersController.DeleteAppointment([FromBody] int appointmentId)
{
    // Admin kontrolü
    if (!IsAdmin())
    {
        TempData["Error"] = "Bu sayfaya erişim yetkiniz yok!";
        return RedirectToAction("Login", "User");
    }

    // appointmentId'yi kontrol et
    Console.WriteLine("AppointmentId from frontend: " + appointmentId); // Logla

    // Veritabanından randevuyu bul
    var appointment = _context.Appointments.FirstOrDefault(a => a.Id == appointmentId);

    if (appointment != null)
    {
        _context.Appointments.Remove(appointment);
        _context.SaveChanges();
        return Json(new { success = true, message = "Randevu başarıyla silindi." });
    }

    // Eğer randevu bulunamadıysa
    Console.WriteLine("Appointment not found with ID: " + appointmentId); // Logla
    return Json(new { success = false, message = "Randevu bulunamadı." });
}
```

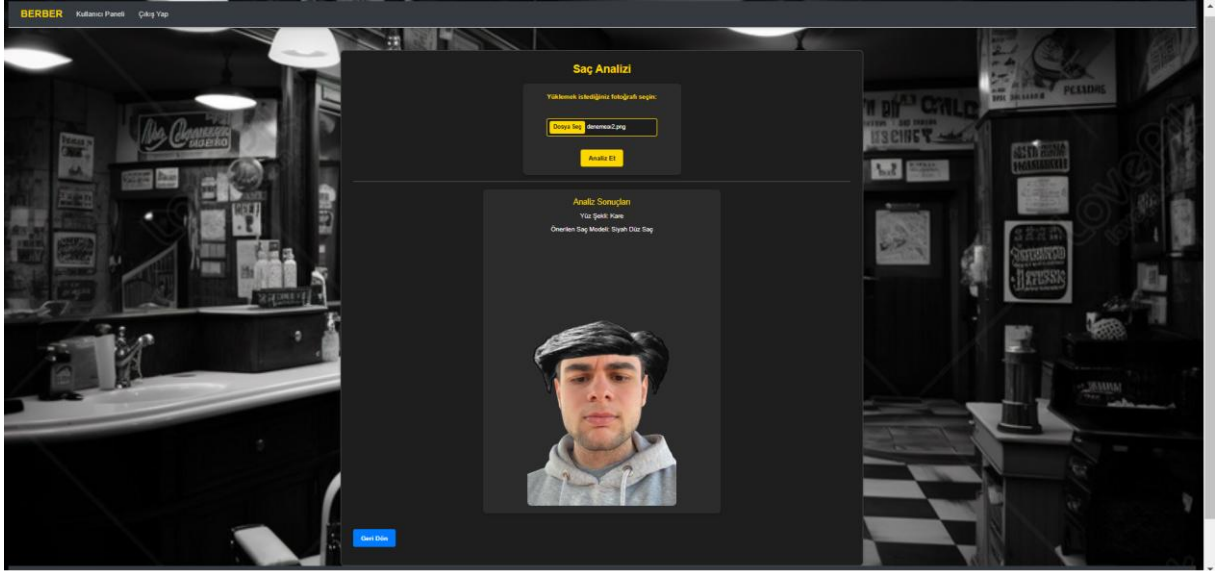
Yapay Zekâ Entegrasyonu

Bu Flask uygulaması, yüz şekline uygun saç modeli öneren ve önerilen modeli yüz üzerine bindiren bir yapay zekâ sistemi olarak çalışmaktadır.

- ❓ Fotoğrafı işler ve yüz şekli analiz eder.
- ❓ Önerilen saç modelini seçer.
- ❓ Seçilen modeli fotoğrafa bindirir.
- ❓ Çıktıyı bir dosyada saklar ve kullanıcıya sonucu JSON olarak döner

Kullanılan Teknolojiler

- Flask: Web uygulamasını çalıştırır ve API endpointlerini sağlar.
- Mediapipe: Yüz analizi ve yüz özelliklerini belirleme işlemleri için kullanılır.
- OpenCV: Görsel işleme (saç modelini yüz üzerine bindirme) işlemleri için kullanılır.



3.VERİTABANI MODELİ:

Veritabanı Yapısı:

Proje, aşağıdaki veritabanı tablolarından oluşmaktadır Ve Postgresql kullanılmıştır:

1. Kullanıcılar Tablosu:

- Kullanıcı ID (Primary Key)
- Kullanıcı Adı
- Email
- Şifre
- Telefon
- Rol

2. Salonlar Tablosu:

- Salon ID (Primary Key)

- Adres
- Telefon
- Ad

3. Çalışanlar Tablosu:

- Çalışan ID (Primary Key)
- Ad
- Uzmanlık (Liste)
- Beceriler (Liste)
- Bağlı Olduğu Salon ID (Foreign Key)

4. Hizmetler Tablosu:

- Hizmet ID (Primary Key)
- Ad
- Fiyat
- Süre (Dakika)
- Bağlı Olduğu Salon ID (Foreign Key)

5. Çalışma Saatleri Tablosu:

- Çalışma Saati ID (Primary Key)
- Gün (Metin)
- Başlangıç Saati (Metin)
- Bitiş Saati (Metin)
- Çalışan ID (Foreign Key)
- Salon ID (Foreign Key)

6. Randevular Tablosu:

- Randevu ID (Primary Key)
- Tarih (Zaman Damgası)
- Çalışan ID (Foreign Key)
- Hizmet ID (Foreign Key)
- Salon ID (Foreign Key)
- Müşteri ID (Foreign Key)

Migrations Yapısı:

Veritabanı yapısının oluşturulmasında Entity Framework Core kullanılmıştır. Aşağıda "InitialCreate" migration işlemi ile oluşturulan tabloların detayları verilmiştir:

```
4 using Microsoft.EntityFrameworkCore.Metadata;
5
6 #nullable disable
7
8 namespace WebProgramlamaProje.Migrations
9 {
10     public partial class InitialCreate : Migration
11     {
12         protected override void Up(MigrationBuilder migrationBuilder)
13         {
14             migrationBuilder.CreateTable(
15                 name: "Salons",
16                 columns: table => new
17                 {
18                     Id = table.Column<int>(type: "integer", nullable: false),
19                     Annotation("SqlServerValueGenerationStrategy.IdentityByDefaultColumn"),
20                     Name = table.Column<string>(type: "character varying(100)", maxLength: 100, nullable: false),
21                     Address = table.Column<string>(type: "character varying(200)", maxLength: 200, nullable: false),
22                     Phone = table.Column<string>(type: "text", nullable: false)
23                 },
24                 constraints: table =>
25                 {
26                     table.PrimaryKey("PK_Salons", x => x.Id);
27                 });
28
29             migrationBuilder.CreateTable(
30                 name: "Users",
31                 columns: table => new
32                 {
33                     Id = table.Column<int>(type: "integer", nullable: false),
34                     Annotation("SqlServerValueGenerationStrategy.IdentityByDefaultColumn"),
35                     Name = table.Column<string>(type: "text", nullable: false),
36                     Email = table.Column<string>(type: "text", nullable: false),
37                     Password = table.Column<string>(type: "text", nullable: false),
38                     Role = table.Column<string>(type: "text", nullable: false)
39                 },
40                 constraints: table =>
41                 {
42                     table.PrimaryKey("PK_Users", x => x.Id);
43                 });
44
45             migrationBuilder.CreateTable(
46                 name: "Employees",
47                 columns: table => new
48                 {
49                     Id = table.Column<int>(type: "integer", nullable: false),
50                     Name = table.Column<string>(type: "text", nullable: false),
51                     Email = table.Column<string>(type: "text", nullable: false)
52                 });
53         }
54     }
55 }
```


1. CreateTable: Salonlar, Kullanıcılar, Çalışanlar, Hizmetler, Çalışma Saatleri ve Randevular tabloları oluşturulmuştur.

2. Foreign Key İlişkileri:

- Salonlar, Çalışanlar ve Hizmetler tablolarına bağlanmıştır.
- Çalışma Saatleri, Çalışanlar ve Salonlar tablolarına bağlanmıştır.
- Randevular, Çalışanlar, Hizmetler, Salonlar ve Kullanıcılar tablolarına bağlanmıştır.

3. Veritabanı Dizini (Indexes):

- Her bir tabloda hızlı erişim için gerekli olan dizinler tanımlanmıştır.

POSTGRESQL VERİTABANI ŞEMA GÖRÜNTÜSÜ



Örnek veritabanı sorguları:

1

SELECT *

2

FROM "Employees"

3

WHERE "SalonId" = 12345;

Data Output

Notifications

Messages

	<div><div>Id</div><div>[PK] integer</div><div></div></div>	<div><div>Name</div><div>text</div><div></div></div>	<div><div>Specialty</div><div>text[]</div><div></div></div>	<div><div>Skills</div><div>text[]</div><div></div></div>	<div><div>SalonId</div><div>integer</div><div></div></div>
1	<div>5</div>	<div>Mehmet Akif</div>	<div>{"Saç Kesme"}</div>	<div>{"Saç Kesme","Sakal"}</div>	<div>12345</div>
2	<div>7</div>	<div>Cengizhan Tezer</div>	<div>{"Kaş Alma, Sakal"}</div>	<div>{"Saç Kesme","Kaş Alma","Sakal"}</div>	<div>12345</div>

Query

Query History

1

SELECT *

2

FROM "Services"

3

ORDER BY "Price" DESC

Data Output

Notifications

Messages

<

SONUÇ

Bu proje, Web Programlama dersi kapsamında geliştirilerek hem teknik becerilerin geliştirilmesi hem de modern web teknolojilerinin pratikte uygulanması amacıyla tasarlanmıştır, dokümanda istenilen bütün gereksinimler karşılanmıştır ve hepsi doğru bir şekilde çalışmaktadır. Projenin tamamlanmasıyla kullanışı kolay ve sade bir web uygulaması ortaya çıkarılmıştır. Veritabanı yapısının etkin bir şekilde kullanılması ve uygulamanın ekran tasarımları, projeyi bir bütün haline getirmiştir.