KONYA TEKNİK ÜNİVERSİTESİ

MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

YAZILIM MÜHEDİSLİĞİ BÖLÜMÜ

WEB TEKNOLOJİLERİ DERSİ
VİZE ÖDEVİ RAPORU

Ad: MEHMET DOĞAN

Soyad: UYANIK

No: 211229050

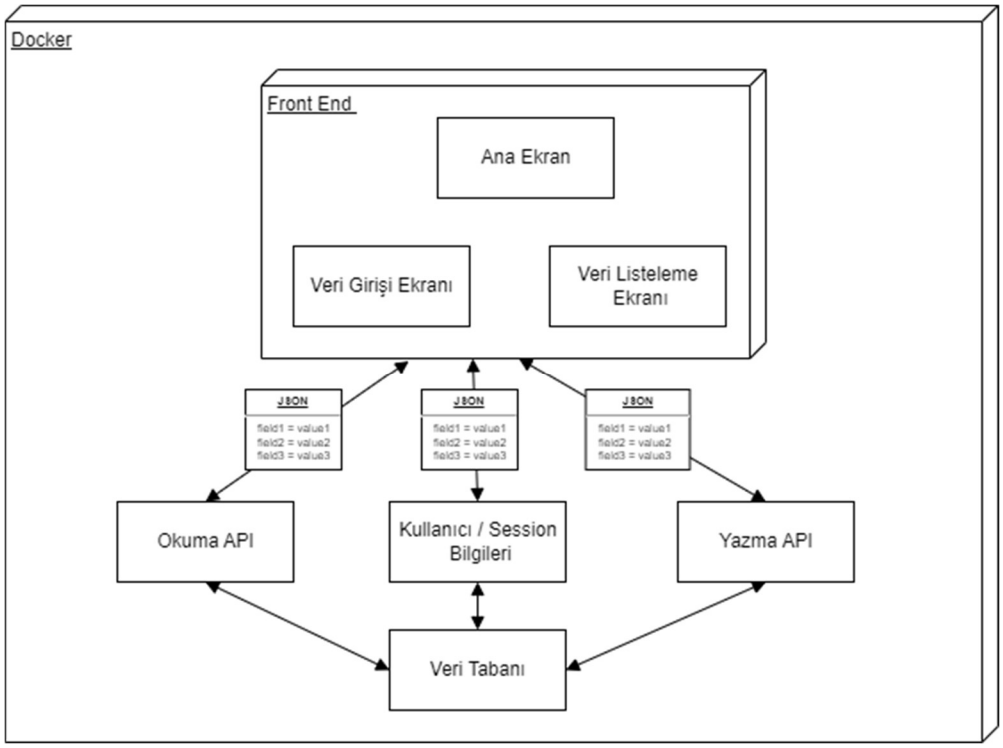
Github Link: https://github.com/mhmtdgnu/WEB-APPLICATION

	KONYA TEKNİK ÜNİVERSİTESİ MÜHENDİSLİK VE DOĞA BİLİMLERİFAKÜLTESİ YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ WEB TEKNOLOJİLERİ DERSİ 1. VİZE ÖDEVİ	

Soru 1

Tasarım size ait olmak üzere, listelenen içeriklerden birine uygun olarak, minimum seviyede Şekil 1'deki özellikleri içerecek bir tasarım yaparak Docker üzerinden yayına almanız istenmektedir.

Ödev tesliminde, ilgili öğretim elemanına, tasarımınızı detaylı bir şekilde anlattıktan sonra, yayına alma sürecini anlatmanız, API çağrıları, veri tabanı ve verilerin transferinde kullandığınız metotlar hakkında bilgi vermeniz beklenmektedir.



Şekil 1. Minimum gereksinimler

İçerik Listesi:

1- **Çocuklar için kitap tavsiye sistemi:**
Kaydolan öğrenciler için yaş aralığına uygun olarak kitap tavsiyelerinde bulunup, popüler kitapları haftalık olarak güncelleyerek yeni gelen kitapları güncel olarak ekrana verecek.

2- **Tıbbi takip sistemi:**
Kullanıcının, kaydolduktan sonra, hastalık bilgisi, düzenli kullandığı ilaç bilgisi, tahlil bilgisi gibi içeriklere ulaşabileceği bir sağlık takip sistemi

3- **Endüstriyel Takip Sistemi**
Bir fabrikada çalışmakta olan üretim bantlarından gelen sensör bilgilerinin sürekli kaydı yapılarak, login olan yöneticinin her üretim hattında hangi üründen ne kadar üretildiğini inceleyebileceği bir raporlama sistemi

Serbest Tasarım
Liste dışında Şekil1'deki minimum gereksinimleri sağlayacak kendi tasarımınız.

Şekil 1: WEB uygulamaları temsili görsel.

WEB Nedir: "Web", genellikle "World Wide Web" (WWW) Dünya Çapında Ağ olarak adlandırılan, internet üzerindeki bilgi ve kaynakların bir ağıdır. Web, dünya çapında milyonlarca insanın bilgi alışverişi yapmasına, iletişim kurmasına ve çeşitli dijital içerikleri (metin, resim, video, müzik vb.) erişmesine olanak tanıyan bir platformdur.

Web'in Temel Bileşenleri

1. **Web Sayfaları:**
   - ❖ Web'in temel yapı taşlarıdır. HTML (Hypertext Markup Language) ile yazılan web sayfaları, metin, görsel ve diğer multimedya içerikleri barındırır.
2. **Web Tarayıcıları:**
   - ❖ Kullanıcıların web sayfalarını görüntülemesini sağlayan yazılımlardır. Örnekler arasında Google Chrome, Mozilla Firefox, Safari ve Microsoft Edge bulunur.
3. **Web Sunucuları:**
   - ❖ Web sayfalarını ve diğer web içeriklerini barındıran ve internet üzerinden erişilebilir hale getiren bilgisayar sistemleridir.
4. **URL'ler (Uniform Resource Locators):**
   - ❖ Web'deki her kaynağın (web sayfası, resim, video dosyası vb.) benzersiz bir

adresidir. Örneğin, **https://www.example.com**.

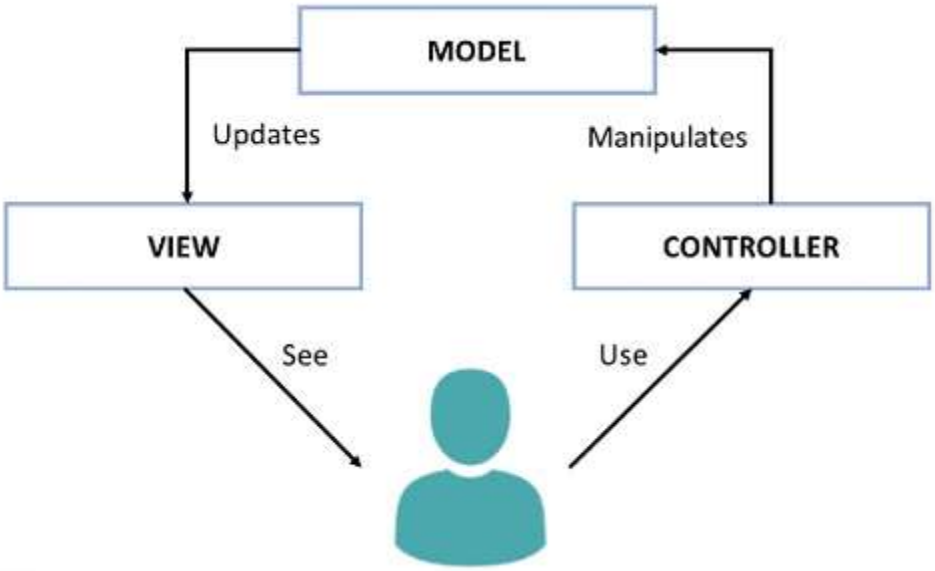
5. **HTTP (Hypertext Transfer Protocol):**
   - ❖ Web tarayıcıları ve web sunucuları arasındaki iletişimi sağlayan protokoldür. HTTPS, güvenli bir iletişim kurmak için HTTP üzerine şifreleme katmanları ekler.

Web'in İşleyişi:

Web'in işleyişi genellikle basit bir döngü izler:

1. Kullanıcı, web tarayıcısında bir URL girer veya bir bağlantıya tıklar.
2. Web tarayıcısı, bu URL'yi temsil eden web sunucusuna bir HTTP isteği gönderir.
3. Web sunucusu, istenen kaynağı (genellikle bir web sayfası) işler ve bir HTTP yanıtı olarak tarayıcıya geri gönderir.
4. Web tarayıcısı, alınan içeriği işler ve kullanıcıya gösterir.

Web'in Önemi ve Kullanım Alanları: Web, bilgiye erişimi kolaylaştırır ve dünya genelindeki insanlar arasında iletişim ve iş birliği için önemli bir araçtır. Eğitimden ticarete, haber yayıncılığından sosyal medyaya kadar pek çok alanda kullanılır. Web sayesinde, insanlar arama motorları üzerinden bilgi arayabilir, çevrimiçi alışveriş yapabilir, sosyal medya platformlarında etkileşimde bulunabilir ve çok daha fazlasını gerçekleştirebilir.



Şekil 2: MVC temsili görseli.

MVC Nedir: Model-View-Controller (MVC) mimarisi, web uygulamalarının geliştirilmesinde yaygın olarak kullanılan bir tasarım desenidir. Bu desen, uygulamanın farklı yönlerini - veri modeli, kullanıcı arayüzü ve kontrol mantığı - üç ayrı bileşene bölerek, geliştirme sürecini daha düzenli, anlaşılır ve yönetilebilir hale getirir. Aşağıda MVC'nin her bir bileşenini gerçek hayattan örneklerle açıklamaya çalışacağım:

1. Model: Model, uygulamanın veri ve iş kurallarını temsil eder. Bir web uygulamasında model, veri tabanı kayıtları, kullanıcı profilleri veya alışveriş sepeti gibi verileri içerebilir.
Örnek: Bir e-ticaret sitesinde, "Ürün" modeli, ürünle ilgili verileri (fiyat, stok durumu, açıklama vb.) ve bu ürünle ilgili iş kurallarını (stoktan düşme, fiyat güncelleme vb.) içerir.

2. View: View, kullanıcının gördüğü arayüzü temsil eder. HTML, CSS ve bazı JavaScript kodlarından oluşur. View, kullanıcının model verilerine nasıl eriştiğini ve nasıl etkileşimde bulunduğunu belirler.
Örnek: Aynı e-ticaret sitesinde, ürün detay sayfası bir "View" örneğidir. Bu sayfa, modelden alınan ürün bilgilerini gösterir ve kullanıcının bu ürüne yorum yapmasına veya sepete eklemesine olanak tanır.

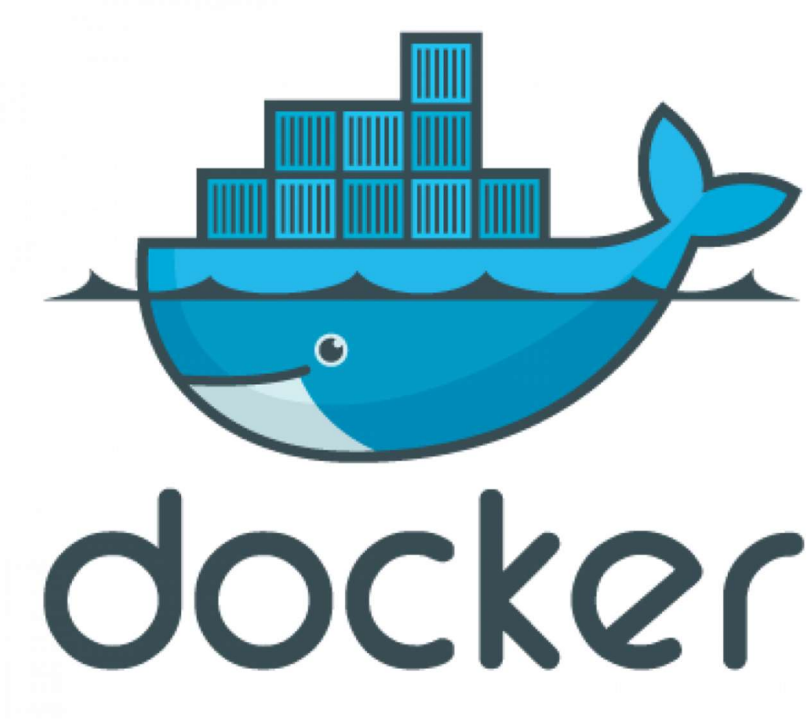
3. Controller (Kontrolcü): Controller, model ve view arasındaki etkileşimi yönetir. Kullanıcıdan gelen istekleri alır, modeli bu isteklere göre günceller ve sonuç olarak bir view seçer ve onu kullanıcıya sunar.
Örnek: Kullanıcı, e-ticaret sitesinde bir ürünü sepete eklemek istediğinde, bu istek bir "Controller"a gider. Controller, "Sepet" modelini günceller ve bu değişikliği gösteren bir "Sepet Görünümü"ne yönlendirir.

MVC'nin Gerçek Hayat Uygulaması:MVC'nin gücü, uygulama bileşenlerini ayrıştırarak daha temiz ve yönetilebilir kod yapısı oluşturmasındadır. Örneğin, bir haber sitesini düşünün:

- **Model**: Haber makaleleri, yazar bilgileri ve yorumlar, veri tabanında saklanan verilerdir.
- **View**: Kullanıcının haber makalelerini ve yorumları gördüğü web sayfasıdır.

- **Controller**: Kullanıcının bir makale okuma isteği geldiğinde, bu makaleyi veri tabanından çeker ve okunmak üzere ilgili görünüme yönlendirir.

- MVC mimarisi sayesinde, haber sitesindeki her bir makalenin görünümü değişse bile, bu değişiklik sadece view katmanında yapılır ve model veya kontrolcü üzerinde hiçbir etkisi olmaz. Benzer şekilde, veri tabanı yapısı değişse dahi, bu değişiklik yalnızca model katmanını etkiler ve view ve controller katmanlarına etki etmez. Bu ayrıştırma, büyük ve karmaşık web uygulamalarının geliştirilmesi ve bakımı süreçlerini kolaylaştırır.



Şekil 3: Docker görseli.

Docker Nedir: Docker, uygulamaların hızlı ve tutarlı bir şekilde dağıtılmasını sağlamak için kullanılan bir açık kaynak konteynerleştirme uygulamaaıdır. Konteynerleştirme, uygulamaları ve onların bağımlılıklarını bir "konteyner" içinde paketlemek anlamına gelir. Bu yaklaşım, uygulamaların farklı ortamlarda (geliştirme, test, üretim vb.) sorunsuz ve tutarlı bir şekilde çalışmasını sağlar.

Docker'ın Çalışma Prensibi

1. **Konteynerler:**

- ❖ Docker, uygulamaları izole edilmiş "konteynerler" içinde çalıştırır. Bir konteyner, uygulamanın çalışması için gerekli olan her şeyi (kod, çalışma zamanı, sistem araçları, kütüphaneler vb.) içerir. Bu, uygulamanın farklı ortamlarda aynı şekilde çalışacağını garanti eder.

2. **Docker İmajları:**
   - ❖ Her konteyner, bir Docker imajından oluşturulur. Docker imajları, uygulamanın ve onun çalışması için gerekli olan tüm bağımlılıkların bir "şablonu" veya "mavi baskısı" olarak düşünülebilir.
3. **Dockerfile:**
   - ❖ Docker imajları, **Dockerfile** adı verilen bir yapılandırma dosyası kullanılarak tanımlanır. Bu dosyada, uygulamanın nasıl çalıştırılacağı belirtilir.
4. **Docker Hub ve Registryler:**
   - ❖ Oluşturulan Docker imajları, Docker Hub gibi kayıt defterlerine (registry) yüklenebilir. Bu, imajların başka yerlerde kolayca indirilip kullanılmasını sağlar.

Gerçek Hayattan Örneklendirmesi

1. **Geliştirme ve Üretim Tutarlılığı**: Bir yazılım geliştiricisi, yerel geliştirme ortamında bir web uygulaması üzerinde çalışıyor. Uygulama, belirli bir Python sürümüne ve çeşitli kütüphanelere ihtiyaç duyuyor. Geliştirici, bu uygulamayı Docker konteyneri içinde geliştirir. Uygulama, Docker imajı sayesinde tüm gerekli bağımlılıkları içerir. Geliştirici, uygulamayı Docker Hub'a yükler. Ardından, uygulama üretim ortamına taşınacak. Üretim ortamı, geliştirme ortamından farklı olabilir (farklı işletim sistemi, farklı ağ yapılandırmaları vb.). Ancak, Docker sayesinde, geliştirici üretim ortamında aynı Docker imajını kullanır. Bu, uygulamanın yerelde nasıl çalıştıysa, üretimde de aynı şekilde çalışacağını garanti eder.

2. **Mikro Hizmetler ve Ölçeklenebilirlik**: Bir şirket, birden çok mikro hizmetten oluşan büyük bir uygulama işletiyor. Her bir mikro hizmet, farklı bir işlevi yerine getirir ve bağımsız olarak ölçeklenebilir. Docker kullanılarak, her bir mikro hizmet ayrı bir konteyner olarak dağıtılır ve yönetilir. Bu sayede, her mikro hizmet bağımsız olarak güncellenebilir, ölçeklenebilir ve hata durumlarında izole edilebilir. Böylece, sistem daha dayanıklı ve yönetilebilir hale gelir.

Şekil 4: Temsili veri tabanı görseli.

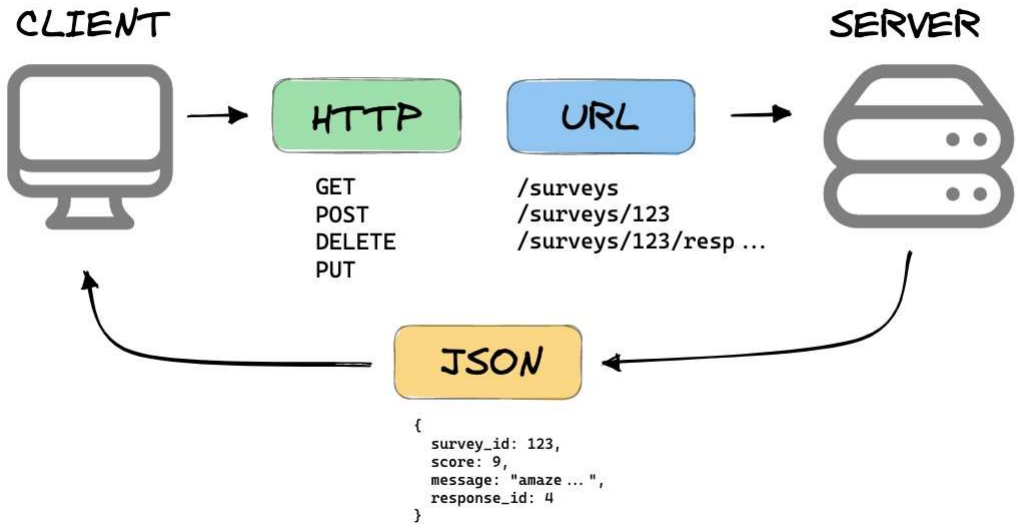
Veri Tabanı Nedir: Veri tabanı, sistemli bir şekilde düzenlenmiş ve saklanmış bilgilerin (verilerin) toplamıdır. Veri tabanları, verileri etkin bir şekilde saklamak, yönetmek ve almak için tasarlanmıştır. Bir veri tabanı, verileri çeşitli biçimlerde (sayısal, metinsel, ikili vb.) içerebilir ve bu verilerin sorgulanmasını, güncellenmesini, ekleme veya silme işlemlerini kolaylaştırmak için çeşitli yöntemler ve araçlar sunar.

Veri Tabanı Temel Bileşenleri

1. **Veri Tabloları ve Kayıtlar:**
    - ❖ Veri tabanları genellikle tablolar halinde düzenlenir. Her tablo, benzer veri öğelerini (örneğin, müşteri bilgileri, ürün detayları) içerir ve bu öğeler satırlar (kayıtlar) ve sütunlar (alanlar) olarak organize edilir.
2. **İlişkiler:**
    - ❖ İlişkisel veri tabanlarında, farklı tablolar arasındaki ilişkiler önemlidir. Bu ilişkiler, verilerin daha anlamlı ve verimli bir şekilde sorgulanmasını sağlar. Örneğin, bir "Müşteriler" tablosu ile bir "Siparişler" tablosu arasında bir ilişki olabilir.
    - ❖
3. **Sorgu Dili (SQL):**
    - ❖ Veri tabanları, genellikle SQL (Structured Query Language) adı verilen

standart bir sorgu dilini kullanarak etkileşime girer. SQL, verileri sorgulamak, güncellemek, eklemek ve silmek için kullanılır.

4. **Veri tabanı Yönetim Sistemi (DBMS):**
   - ❖ Veri tabanlarını oluşturmak, yönetmek ve sürdürmek için kullanılan yazılım sistemleridir. Örnekler arasında MySQL, PostgreSQL, Microsoft SQL Server, Oracle ve MongoDB bulunur.



Şekil 5: Temsili WEB API görseli.

API ve WEB API Nedir: API (Application Programming Interface), yazılımların birbiriyle etkileşim kurmasını sağlayan bir arayüzdür. API'ler, bir yazılımın belirli işlevlerini veya verilerini başka bir yazılımın kullanabilmesi için bir "sözleşme" sunar. Bu, yazılımların birbirlerinin işlevlerini kullanabilmesini, veri alışverişi yapabilmesini ve entegrasyon sağlamasını mümkün kılar.

Örneğin, bir hava durumu uygulaması, bir hava durumu servisinin API'sini kullanarak güncel hava durumu bilgilerini alabilir. Bu API, hava durumu verilerine nasıl erişileceğini, hangi parametrelerin kullanılacağını ve verinin nasıl geri döneceğini tanımlar.

Web API, genellikle HTTP protokolü üzerinden erişilebilen ve genellikle web uygulamaları için tasarlanmış bir API türüdür. Web API'leri, web tabanlı servislerle etkileşim kurmak için kullanılır ve genellikle JSON veya XML formatında veri döndürür.

Örneğin, bir sosyal medya platformunun Web API'si, geliştiricilere bu platformdaki kullanıcı verilerine erişme, mesaj gönderme veya takipçi listesi alma gibi işlevler sağlayabilir.

API ve Web API Arasındaki Farklar

1. **Erişim Yöntemi:**
    - ❖ Genel API'ler, genellikle bir yazılım kütüphanesi veya framework içerisinde tanımlanır ve doğrudan yazılım içinde çağrılır.
    - ❖ Web API'leri ise HTTP protokolü üzerinden erişilir ve genellikle internet üzerinden kullanılır.
2. **Veri Formatı ve Protokol:**
    - ❖ API'ler, çeşitli protokoller ve veri formatları kullanabilir. Web API'leri ise genellikle HTTP/HTTPS protokollerini ve JSON veya XML gibi web dostu formatları kullanır.
3. **Kullanım Alanı:**
    - ❖ API'ler, genellikle belirli bir uygulama veya sistem içindeki işlevleri diğer yazılımlarla paylaşmak için kullanılır.
    - ❖ Web API'leri, genellikle farklı sistemler ve uygulamalar arasında veri alışverişini ve etkileşimi sağlamak için kullanılır.
4. **Platform Bağımsızlık:**
    - ❖ Web API'leri genellikle platform bağımsızdır, yani farklı programlama dilleri ve sistemler tarafından kullanılabilir. Bu, internet üzerindeki farklı uygulamaların ve hizmetlerin birbiriyle etkileşime girmesini kolaylaştırır.
    - ❖ Geleneksel API'ler bazen belirli bir dil veya platforma özgü olabilir.

Şekil 6: json api temsili görseli.

Detaylar

1. **Veri tabanı Bağlantısı:**
   - ❖ Web API, bir veri tabanına bağlanarak istenen verileri sorgular. Bu, SQL sorguları, ORM (Object-Relational Mapping) araçları veya diğer veri tabanı etkileşim teknikleri kullanılarak yapılabilir.
2. **Veri Çekme ve İşleme:**
   - ❖ API, belirli bir iş mantığına göre veri tabanından veri çeker. Bu veriler, genellikle işlenir ve istemci tarafından kullanılmak üzere uygun bir formata dönüştürülür.
3. **JSON Formatında Veri Sunumu:**
   - ❖ Çekilen veriler, JSON (JavaScript Object Notation) formatında serileştirilir. JSON, hafif bir veri alışverişi formatıdır ve insanlar tarafından okunabilirken, makineler tarafından da kolayca ayrıştırılabilir.

Avantajları:

1. **Platform Bağımsızlık:**
   - ❖ JSON, çoğu programlama dilince desteklenir ve platform bağımsızdır. Bu, farklı teknolojiler kullanarak geliştirilen istemci uygulamalarının Web API'den kolayca veri almasını sağlar.
2. **Hafif ve Hızlı:**
   - ❖ JSON, verimli bir biçimde ayrıştırılabilir ve ağ üzerinden hızlı bir şekilde iletilir. Bu, performans açısından avantaj sağlar, özellikle mobil cihazlar ve düşük bant genişliğine sahip ortamlarda.

3. **Kolay Okunabilir ve Geliştirilebilir:**
    - ❖ JSON, insanlar tarafından okunabilir bir yapıya sahiptir, bu da geliştirme ve hata ayıklama süreçlerini kolaylaştırır.
4. **Esneklik:**
    - ❖ JSON formatı, esnek bir veri yapısına sahiptir. Farklı tipte verileri (sayılar, dizeler, booleanlar, nesneler, diziler) destekler ve bu verileri hiyerarşik olarak organize edebilir.

Dezavantajları:

1. **Güvenlik Sorunları:**
    - ❖ API üzerinden veri tabanına erişim, güvenlik açıklarına yol açabilir. Doğru kimlik doğrulama ve yetkilendirme mekanizmalarının uygulanması gerekir.
2. **Veri Serileştirme Maliyeti:**
    - ❖ Verilerin JSON formatına serileştirilmesi ve ayrıştırılması ekstra işlem gücü gerektirir. Büyük veri setleri için bu, performans üzerinde olumsuz bir etki yaratabilir.
3. **Sınırlı Veri Tipi Desteği:**
    - ❖ JSON, tüm veri tiplerini desteklemez (örneğin, tarih ve saat tipleri). Bu, bazı durumlarda veri dönüşümüne ekstra dikkat gerektirebilir.
4. **Ağ Trafiği Yönetimi:**
    - ❖ Eğer API çok miktarda veri döndürüyorsa, bu ağ trafiğini ve yanıt sürelerini olumsuz etkileyebilir. Veri sayfalama ve sıkıştırma gibi teknikler bu sorunu hafifletebilir.

Şekil 7: Responisve sayfa.

Sayfa Responsive Özelliği: Web'de sayfaların "responsive" olması, bir web sayfasının farklı cihazlarda ve ekran boyutlarında düzgün bir şekilde görüntülenebilmesi anlamına gelir. Responsive tasarım, web sayfalarının içeriğinin, kullanıldığı cihazın ekran boyutuna ve çözünürlüğüne uyum sağlamasını ifade eder. Bu yaklaşım, kullanıcı deneyimini iyileştirmek ve cihazlar arası tutarlı bir görünüm sağlamak için kritik öneme sahiptir.

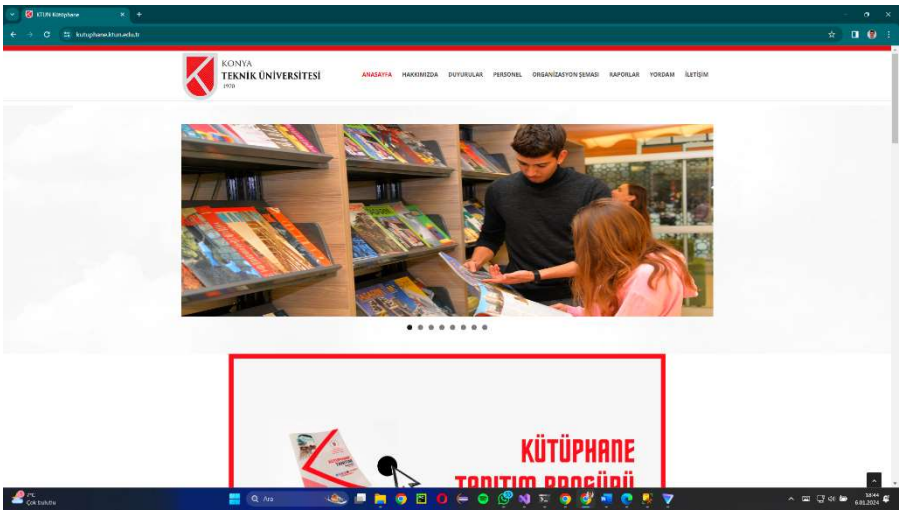
Responsive Tasarımın Temel Özellikleri

1. **Esnek Düzenler:** Web sayfası, sabit piksel genişliklerine dayanmak yerine yüzde gibi orantılı birimleri kullanarak oluşturulur. Bu, sayfanın farklı ekran genişliklerine göre ölçeklenebilmesini sağlar.
2. **Esnek Resimler ve Medya:** Sayfadaki resimler, videolar ve diğer medya öğeleri, içeriklerini çevreleyen elementlere göre boyutlandırılır. Bu, ekran boyutu ne olursa olsun medyanın uygun şekilde görüntülenmesini sağlar.
3. **Medya Sorguları:** CSS medya sorguları, farklı ekran boyutları için farklı stil kuralları uygulanmasına olanak tanır. Bu, tasarımcıların belirli bir ekran genişliğinde veya cihaz türünde spesifik CSS özelliklerini etkinleştirmesini sağlar.
4. **Duyarlı Navigasyon:** Ekran boyutuna bağlı olarak, navigasyon menüleri, farklı cihazlarda daha iyi kullanıcı deneyimi sağlayacak şekilde düzenlenir.

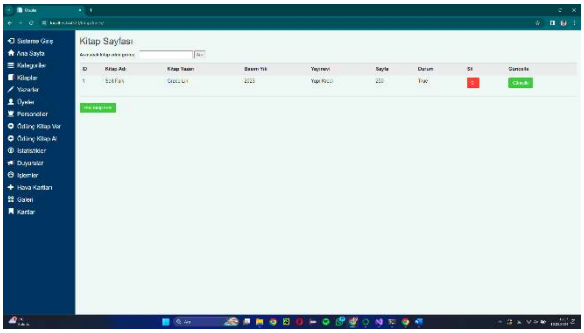
- Geliştirdiğim WEB uygulaması bir kişisel kişisel giriş sayfası olan kütüphane yönetim-bilgi sistemidir.

- Geliştirilen uygulamada hem .Net Core hem de MVC yapısı uygulanmıştur.
- Uygulamayı geliştirmeden önce MVC (Model – View – Controller) Model – Görünüm – Kontrolcü mimarisinde karar kılınmıştır ve proje de bu mimari üzerinden geliştirilmiştir.
- Veri tabanı olarak MSSQL platformundan yararlanılmıştır.
- Projede API (Application Programming Interface) sisteminde ise:
    - ✓ API geliştirmesinde de C# dilindeki Web API projesi rol almıştır.
    - ✓ Projedeki fonksiyonlar Dapper ORM kullanılarak geliştirilmiştir.
    - ✓ Veri tabanındaki bilgilerin WEB uygulamasına alış-verişi .json formatında gerçekleştirilmiştir.
- Front End tasarımı gelişmiş bir kurumsal kuruma yönelik geliştirilmiştir.
- Projede sade, kullanışlı, özgün ve standarda oturtulmuş bir Front End tasarımı geliştirilmiştir.
- Projede sayfaların kullanılabilirliğini, farklı cihazlarda kalitesini ve sürdürülebilirliğini arttırmak için sayfaların responsive olmasına dikkat edilmiştir.
- **WEB uygulamamın özellikleri şunlardır. (Ek özellikleri konusunda görüş öğretim görevlisine kalmışıtr.)**
    - ✓ WEB uygulamasında kütüphane yönetim sistemi yani sistemin businnes tarafında Ana Sayfa sekmesine tıklandığında fidan bağışı, kitap bağışı gibi çeşitli sitelere yönlendirme adımları bulunmaktadır bir nevi web dünyasında yeni pencereler açmaktadır.

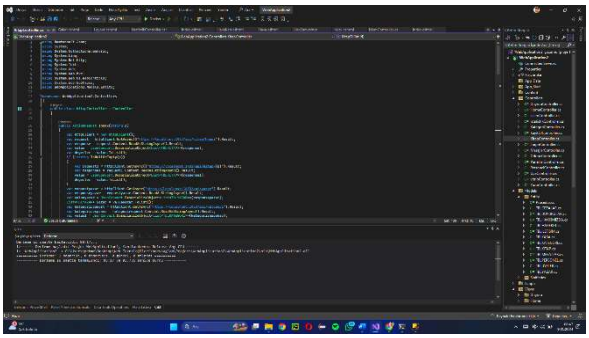
    - ✓ Ana Sayfa sekmesinde Konya Teknik Üniversitesi Kütüphanesinin internet sitesine atmaktadır.



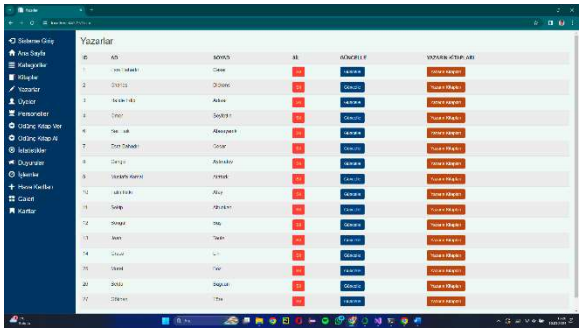
✓ Kategori, Kitap, Yazar, Personel, Ödünç Kitap İşlemleri dinamik olarak silme, güncelleme, ekleme gibi sistem gereksinimleri olarak eklenmiştir.



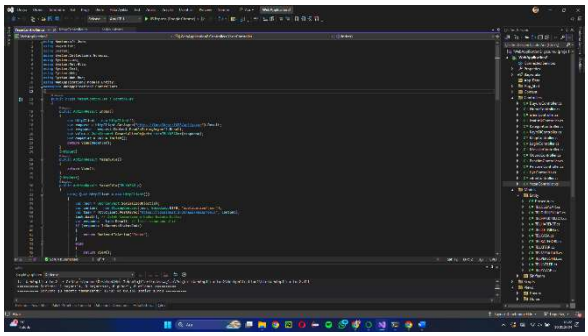
Şekil 11: Kitap Sayfası frontend



Şekil 10: Kitap Sayfası backend

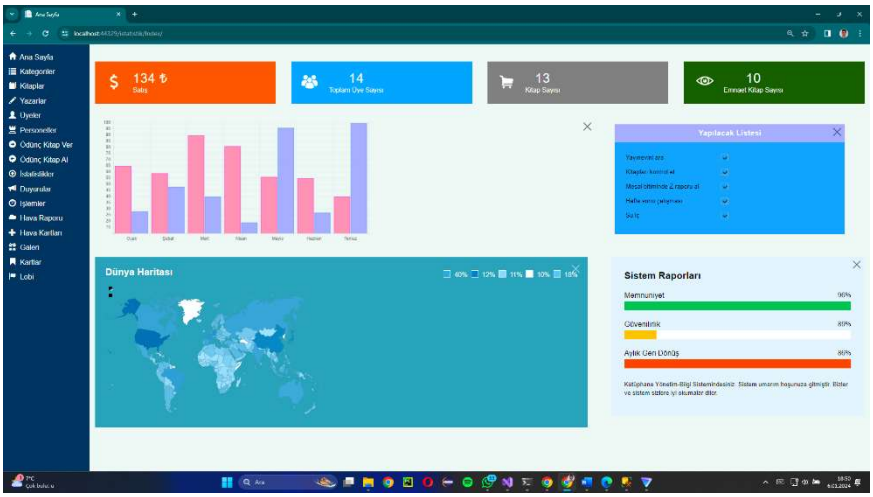


Şekil 9: Yazarlar sayfası görünümü.



Şekil 8: Yazarlar sayfası backend (YazarController)

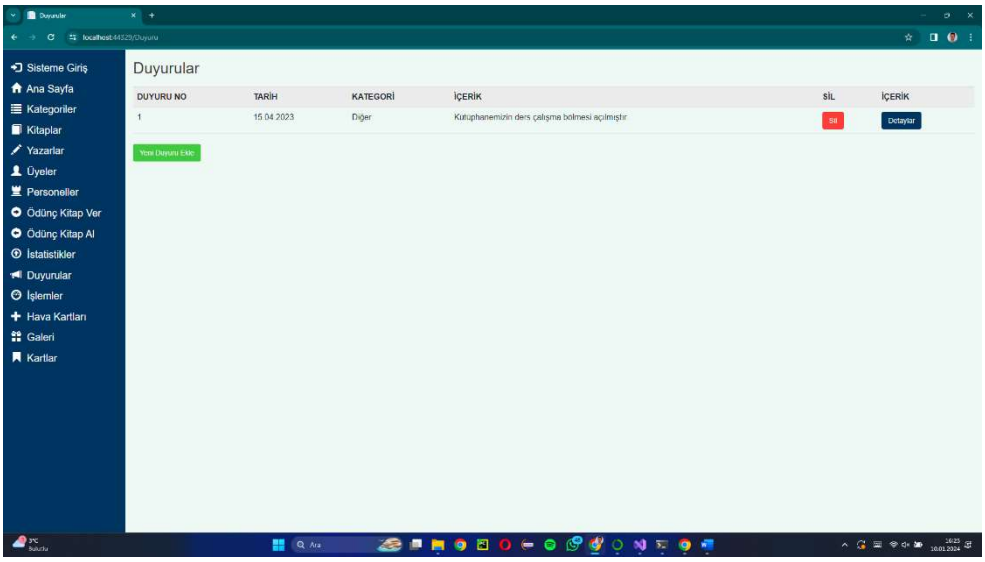
✓ Bazı listeleme alanlarında örneğin Üyeler sekmesinde listeleme yapılırken sayfalama da yapılmıştır.



✓ Ek özellik olarak kütüphane bilgi sistemindeki İstatistikler sayfasında kütüphanenin Gelir miktarı (Geç teslim edilen kitaptan, geç teslim edilen gün başında alınan ücret), toplam üye sayısı, toplam kitap sayısı ve emanet kitap

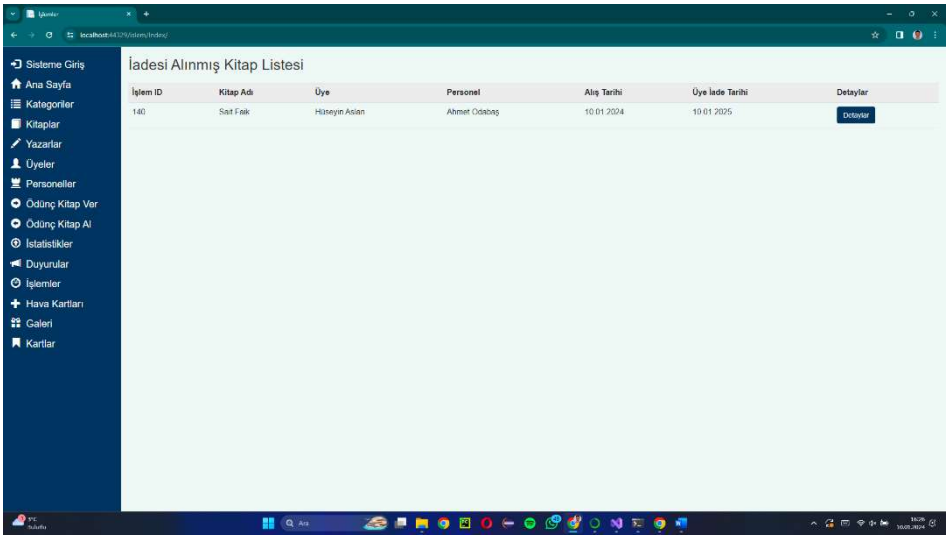
sayıları veri tabanından dinamik olarak API ile alınıp view kısmında gösterilmiştir

- ✓ Yine İstatistik sayfasında sistem raporları, kullanılan ilgili kurumdaki sistem kullanıcısının işlemleri hatırlayabilmesi adına Yapılacaklar Listesi çhartı, alt tarafta Sistem Raporları ve ülkelere göre kitap okuma oranlarını gösteren bir dünya haritası eklenmiştir.
- ✓ Duyurular sayfasında ise kütüphane sisteminin gerekli duyuruları yapma imkanı sağlanmıştır. Bu sayfa içinde yeni duyuru ekleyebilme, Eklenen duyuruyu silme ve güncelleme işlemleri dinamik bir şekilde gerçekleştirilmiştir.



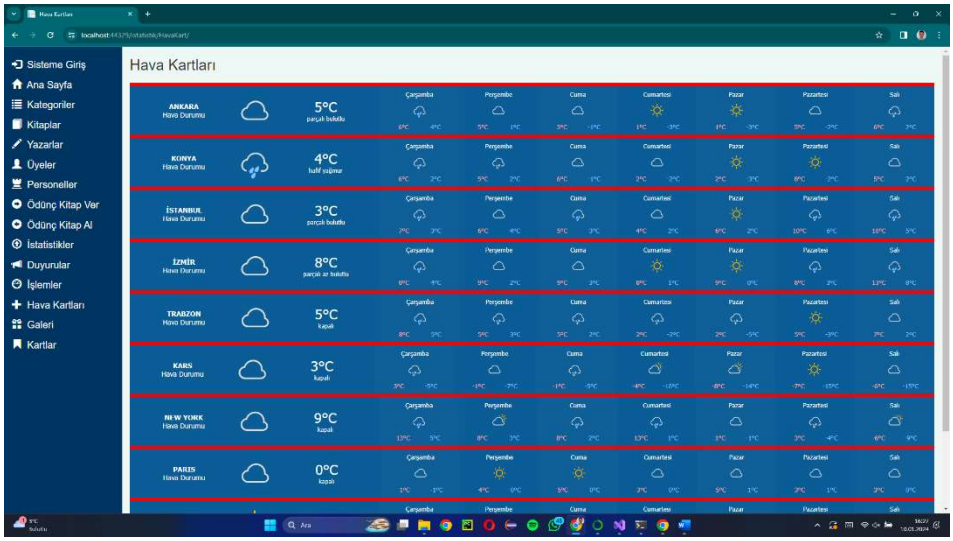
Şekil 12: Yazarlar sayfası görünümü.

✓ İşlemler sekmesinde ise kütüphane web uygulamasının kullanıma açıldığı andan itibaren kitap girdi çıktıları kayıt altına alınmaktadır. Detaylarda ise kayıt detayları bulunmaktadır.



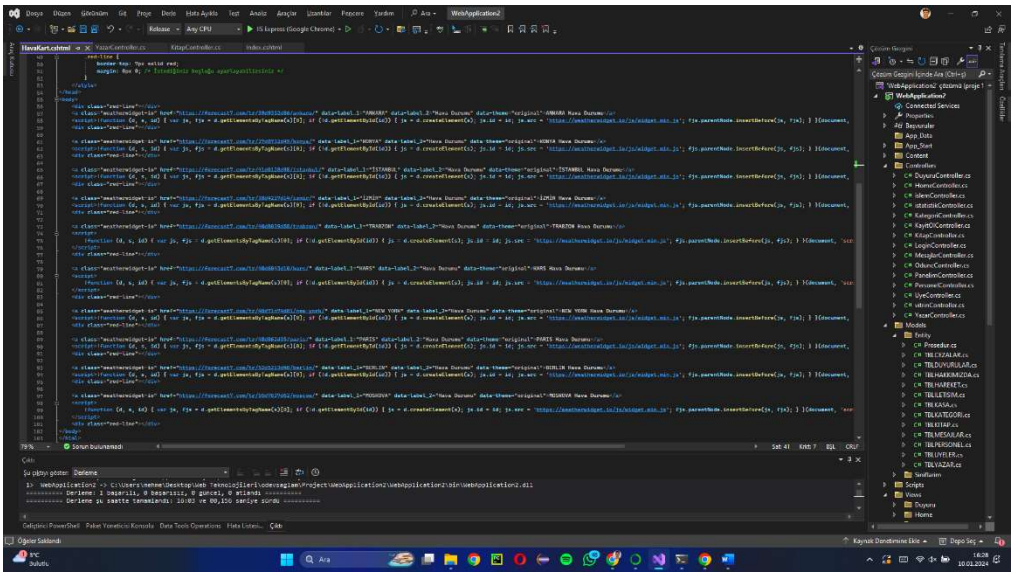
Şekil 13: İadesi alınmış işlemleri gerçekleştirilmiş kitapların listelendiği sayfa.

✓ Hava Raporu ve Hava Kartları sayfalarında ise hazır CSS' lerden faydalanılmıştır. (https://www.w3schools.com/)



Şekil 14: Farklı ülkelerin farklı şehirlerinden alınan hava durumu ve kartları.

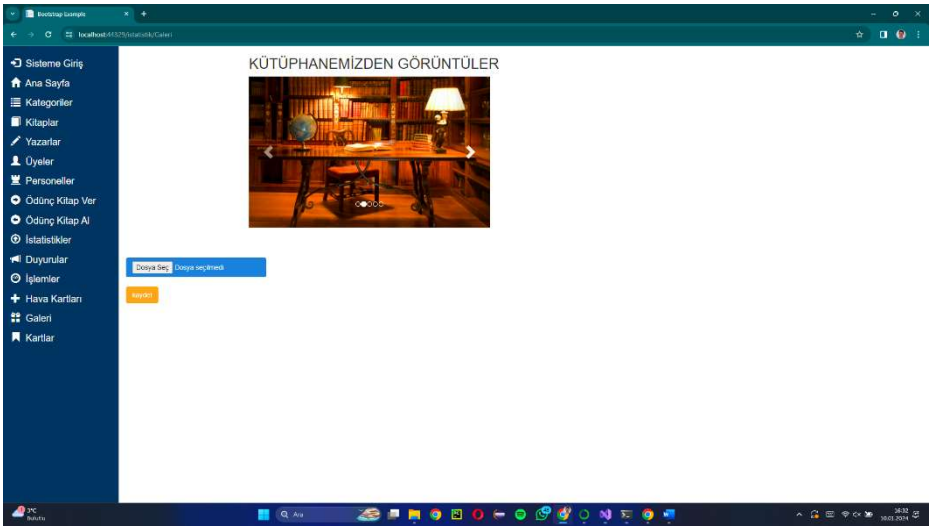
- ✓ Frontend kısmı görünen bu sayfanın backend tarafı ise aşağıdaki gibi görünmektedir ve kod açıklanacaktır.



Şekil 15: Hava kartları HTML-CSS bölümü.

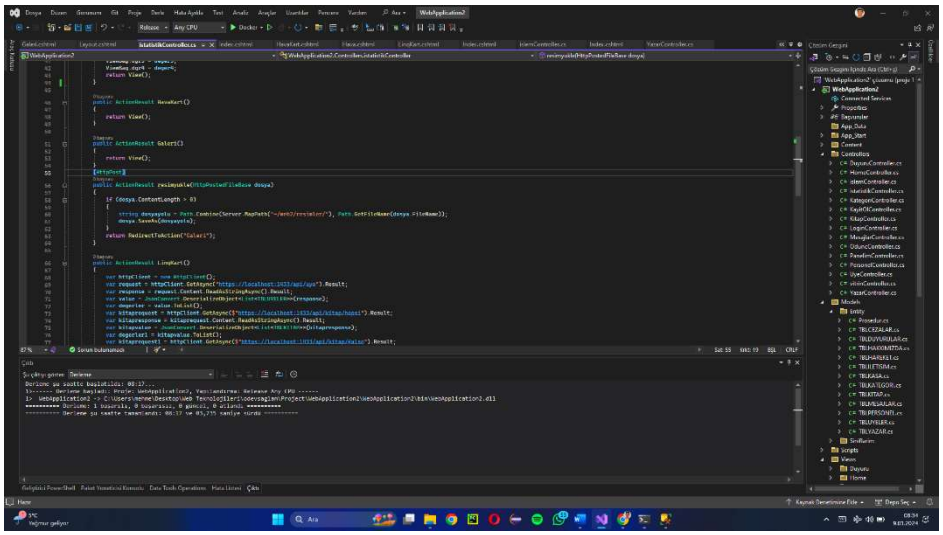
- ✓ Üst tarafta gösterilen kod bir web sayfasında çeşitli şehirler için hava durumu bilgilerini gösteren widget'lar (küçük uygulamalar) eklemek için yazılmış bir HTML ve JavaScript kodu.
- ✓ HTML Bağlantılar (Anchors): Her a etiketi, belirtilen şehirlerin hava durumuna dair bilgileri gösteren bir web sitesine bir bağlantı içerir. Bu bağlantılar, kullanıcıların hava durumu bilgilerine hızlı bir şekilde ulaşmasını sağlar.
- ✓ JavaScript Fonksiyonları: Kod içindeki JavaScript weatherwidget.io tarafından sağlanan hava durumu widget'larını sayfaya entegre etmek için kullanılır. Bu fonksiyonlar, widget'ın çalışması için gerekli JavaScript dosyasını (widget.min.js) sayfaya dinamik olarak ekler.
- ✓ Stil Ayarlamaları (CSS): .red-line sınıfı ile tanımlanan CSS stili, sayfada kırmızı bir çizgi oluşturur. Bu çizgiler, farklı hava durumu widget'ları arasında görsel bir ayrım sağlamak için kullanılıyor.

- ✓ Ardından Galeri sayfası gelmektedir. Bu sayfada ise kütüphaneyi gösteren örnek fotoğraflar vardır. Kullanıcının isteği doğrultusunda kendisi de fotoğraf ekleyebilir.



Şekil 16: Galeri sayfası ve kütüphaneden görüntüler.

- ✓ Galeri sayfasının frontend tarafı üst taraftaki görselde gösterilmiştir. Backend tarafı ise alt tarafta gösterilmiştir.



Şekil 17: Galeri sayfası backend kısmı.

- ✓ Galeri tarafının backend tarafı ise burada gösterilmiştir. Burada ise galeriye fotoğraf yüklemek için gerekli işlemler yapılmıştır
- ✓ HttpPost Attribute: [HttpPost] etiketi, bu metodun yalnızca HTTP POST istekleri için kullanılacağını belirtir. Bu, bir web formundan veya başka bir HTTP POST işleminden gelen istekleri de işleyebilir.

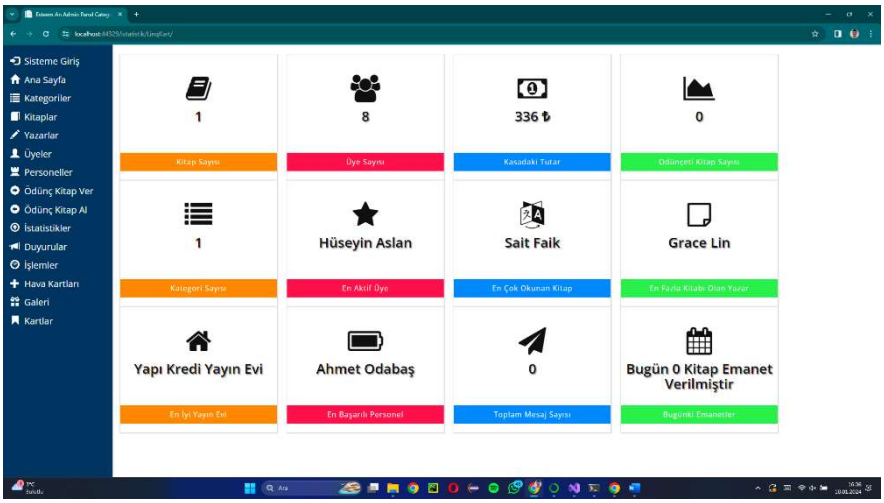
- ✓ ActionResult Metodu: public ActionResult resimyukle(HttpPostedFileBase dosya) ifadesi, resimyukle adında bir metod tanımlar. Bu metod, HttpPostedFileBase tipinde bir parametre alır (dosya), bu da genellikle bir web formundan yüklenen bir dosyayı temsil eder.

- ✓ Dosya Boyut Kontrolü: if (dosya.ContentLength > 0) ifadesi, yüklenen dosyanın boyutunu kontrol eder. Eğer dosya boyutu 0'dan büyükse, kodun devamındaki işlemleri gerçekleştirir.

- ✓ string dosyayolu = Path.Combine(Server.MapPath("~/web2/resimler/"), Path.GetFileName(dosya.FileName));: Bu satır, yüklenen dosyanın sunucuda kaydedileceği dosya yolunu belirler. Server.MapPath metodu, uygulamanın sunucudaki yolunu döndürür ve Path.Combine ile bu yola dosyanın adı eklenerek tam dosya yolu oluşturulur.
  dosya.SaveAs(dosyayolu);: Bu satır, yüklenen dosyayı daha önce belirlenen yola kaydeder.

- ✓ Yönlendirme (Redirect): Son olarak, return RedirectToAction("Galeri"); ifadesi, kullanıcıyı başka bir eyleme ("Galeri" adlı eyleme) yönlendirir. Bu genellikle işlem başarılı olduktan sonra kullanıcıyı başka bir sayfaya (bu örnekte bir galeri sayfasına) götürmek için kullanılır.

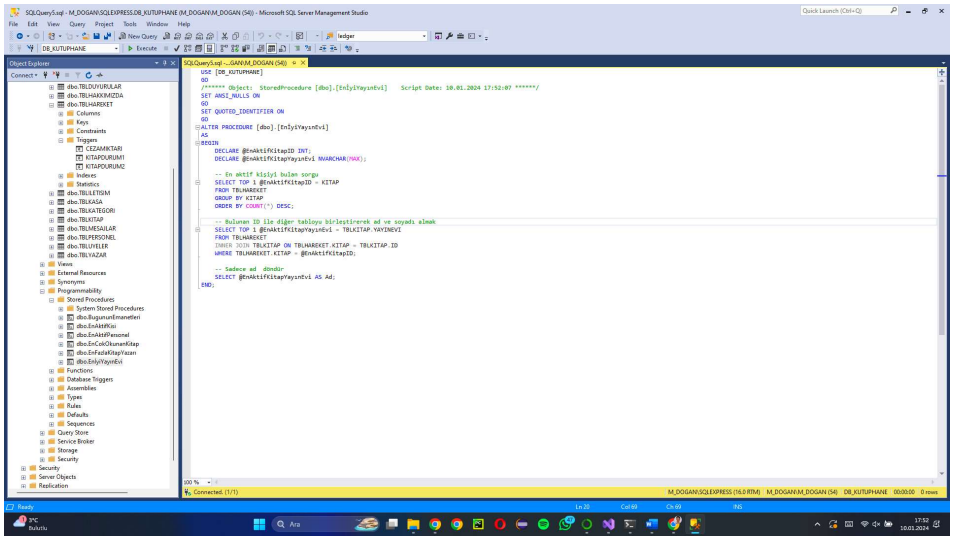
- ✓ Bir sonraki sayfa ise Kartlar sayfasıdır. Bu sayfada ise kütüphanenin istatistiksel sonuçları verilmiştir. Toplam Kitap Sayısı, Üye Sayısı, Toplam Mesaj Sayısı, O günkü toplam emanet verile kitap sayısı, En çok çalışan personel, En aktif üye gibi çeşitli bilgiler vardır. Bir alttaki görselde ise backend tarafı vardır.



Şekil 18: Kütüphane verisi kısa bilgi kartları sayfası.

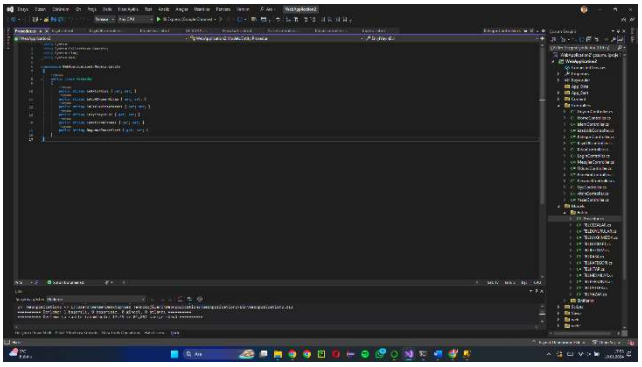
- ✓ Kısa bilgi kartlarına veriler veri tabanında prosedür ve trigerler kullanılarak veriler istelien şekilde filtrelenip WEB API ile web tarafına çekilmiştir.
    - **Veri Tabanı Prosedür:** Veri tabanı prosedürü (database procedure), bir veri tabanı yönetim sistemi (DBMS) içinde tanımlanan ve bir veya daha fazla SQL sorgusunu veya işlemi içeren önceden derlenmiş ve adlandırılmış bir veri tabanı nesnesidir. Veri tabanı prosedürleri, veri tabanı üzerinde tekrar kullanılabilen işlemler, sorgular veya iş mantığı parçalarını saklamak ve yönetmek için kullanılır.
    - **Veri Tabanı Prosedür:** Veri tabanı tetikleyicisi (database trigger), belirli bir veri tabanı olayı (event) gerçekleştiğinde otomatik olarak çalışan ve tanımlanan bir işlemi gerçekleştiren bir veri tabanı nesnesidir. Tetikleyiciler, veri tabanı yönetim sistemi (DBMS) içinde tanımlanır ve genellikle veri bütünlüğünü korumak, iş süreçlerini otomatikleştirmek veya izlemek gibi amaçlarla kullanılırlar.

✓ Aşağıdaki görselde ise prosedür fotoğrafları ve kodları bulunmaktadır.

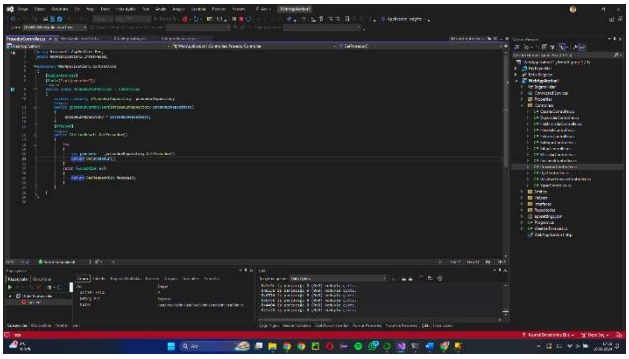


Şekil 19: Veri Tabanı prosedürler örnekleri ve triggerlar.

✓ Konu triggerlara gelmişken API'den triggerları şu şekilde çağrılmaktadır:



Şekil 21: WEB tarafı prosedürlerin işlem sonucunu taşıma model sınıfı



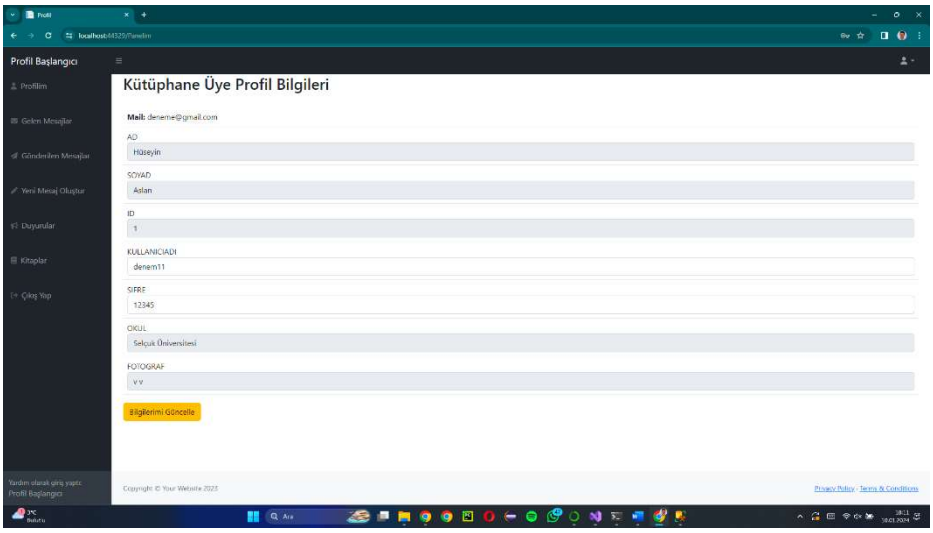
Şekil 20: API tarafı prosedür çağırma merkezi.

✓ Aşağıda ise kartlara gelen bilgilerin frontend'e çekilmeden önceki son durağı Action fonksiyonu verilmiştir. **(Yazının puntosunu arttırarak kodu daha detaylı inceleyebilirsiniz)**

```
public ActionResult LinqKart()
{
    var httpClient = new HttpClient();
    var request = httpClient.GetAsync("https://localhost:1433/api/uye").Result;
    var response = request.Content.ReadAsStringAsync().Result;
    var value = JsonConvert.DeserializeObject<List<TBLUYELER>>(response);
    var degerler = value.ToList();
    var kitaprequest = httpClient.GetAsync($"https://localhost:1433/api/kitap/hepsi").Result;
    var kitapresponse = kitaprequest.Content.ReadAsStringAsync().Result;
    var kitapvalue = JsonConvert.DeserializeObject<List<TBLKITAP>>(kitapresponse);
    var degerler1 = kitapvalue.ToList();
    var kitaprequest1 = httpClient.GetAsync($"https://localhost:1433/api/kitap/false").Result;
    var kitapresponse1 = kitaprequest1.Content.ReadAsStringAsync().Result;
    var kitapvalue1 = JsonConvert.DeserializeObject<List<TBLKITAP>>(kitapresponse1);
    var degerler2 = kitapvalue1.ToList();
    var kitaprequest2 = httpClient.GetAsync($"https://localhost:1433/api/cezalar").Result;
    var kitapresponse2 = kitaprequest2.Content.ReadAsStringAsync().Result;
    var kitapvalue2 = JsonConvert.DeserializeObject<List<TBLCEZALAR>>(kitapresponse2);
    var degerler3 = kitapvalue2.ToList();
    float toplampara = (float)degerler3.Sum(k => k.PARA);
    var kategorirequest = httpClient.GetAsync("https://localhost:1433/api/kategori").Result;
    var kategoriresponse = kategorirequest.Content.ReadAsStringAsync().Result;
    var kategorivalue = JsonConvert.DeserializeObject<List<TBLKATEGORI>>(kategoriresponse).ToList();
    var iletisimrequest = httpClient.GetAsync("https://localhost:1433/api/iletisim").Result;
    var iletisimresponse = iletisimrequest.Content.ReadAsStringAsync().Result;
    var iletisimvalue = JsonConvert.DeserializeObject<List<TBLKATEGORI>>(iletisimresponse).ToList();
    var prosedurrequest = httpClient.GetAsync("https://localhost:1433/api/prosedur").Result;
    var prosedurresponse = prosedurrequest.Content.ReadAsStringAsync().Result;
```

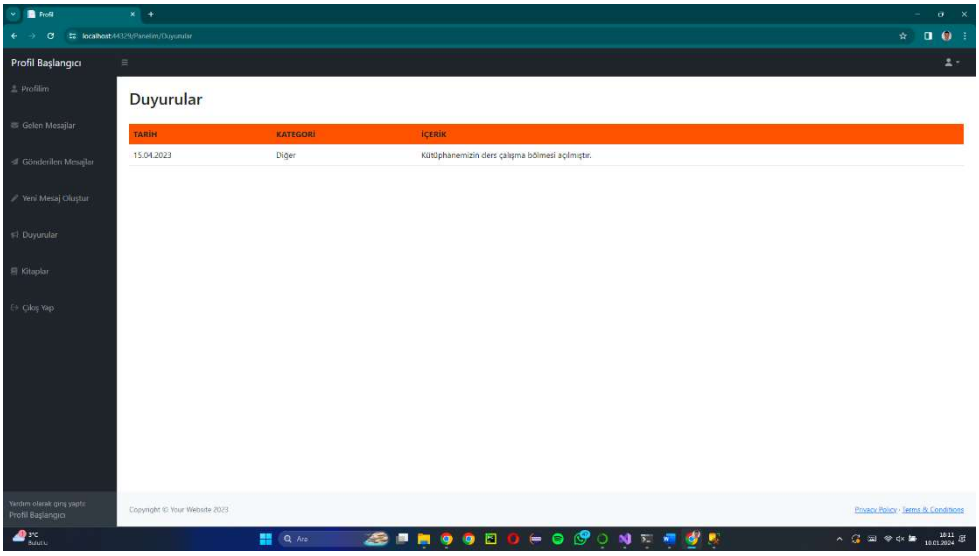
```
        var prosedurvalue = JsonConvert.DeserializeObject<Prosedur>(prosedurresponse);
        var deger1 = degerler1.Count();
        var deger2 = degerler.Count();
        var deger3 = toplampara;
        var deger4 = degerler2.Count();
        var deger5 = kategorivalue.Count();
        var deger6 = prosedurvalue.EnAktifKisi;
        var deger7 = prosedurvalue.EnCokOkunanKitap;
        var deger8 = prosedurvalue.EnFazlaKitapYazarı;
        var deger9 = prosedurvalue.EnİyiYayınEvi;
        var deger10 = prosedurvalue.EnAktifPersonel;
        var deger11 = iletisimvalue.Count();
        var deger12 = prosedurvalue.BugununEmanetleri;
        ViewBag.dgr1 = deger1;
        ViewBag.dgr2 = deger2;
        ViewBag.dgr3 = deger3;
        ViewBag.dgr4 = deger4;
        ViewBag.dgr5 = deger5;
        ViewBag.dgr6 = deger6;
        ViewBag.dgr7 = deger7;
        ViewBag.dgr8 = deger8;
        ViewBag.dgr9 = deger9;
        ViewBag.dgr10 = deger10;
        ViewBag.dgr11 = deger11;
        ViewBag.dgr12 = deger12;
    return View();
}
```

✓ Sisteme girmesi halinde kullanıcıyı karşılayacak ekran kişisel bilgileri düzenleme ekranıdır. Buradan kullanıcı adını ve şifresini düzenleme/değiştirme fırsatı verilmiştir. Ardından kullanıcıya gelen mesajları, kullanıcının gönderdiği mesajları ve yeni mesaj oluştur sayfaları bulunmaktadır. Buradan geçmiş işlemlerini görebilir ve yeni mesaj oluşturup gönderebilmektedir.



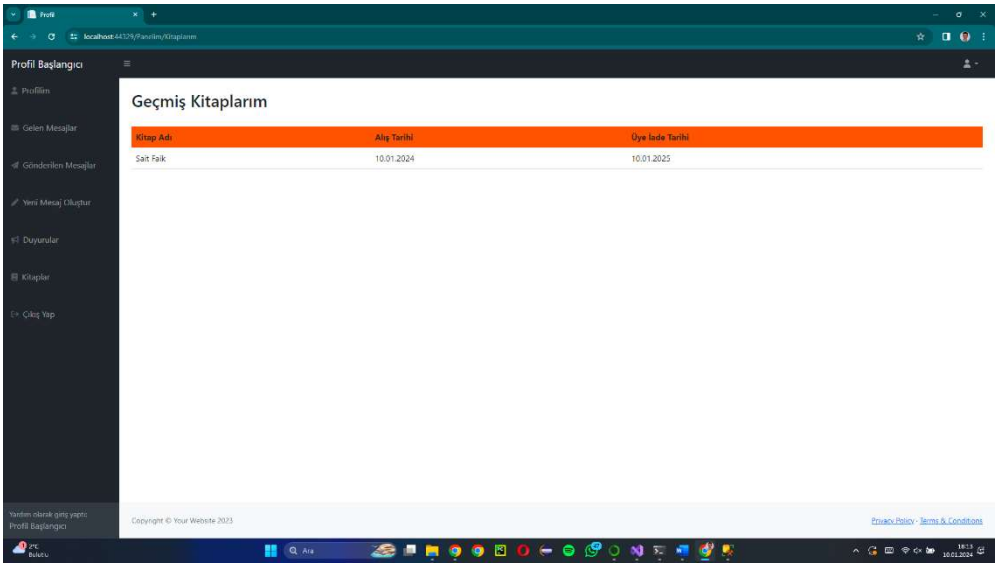
Şekil 22: Kütüphane üye profil bilgileri sayfası.

✓ Ardından duyurular sekmesinden kütüphanenin yaptığı duyuruları görebilmektedir.



Şekil 23: Kullanıcı girişi karşılama sayfası.

✓ Kitaplar sayfasında ise kişinin bugüne kadar yaptığı kitap işlemlerini göstermektedir.

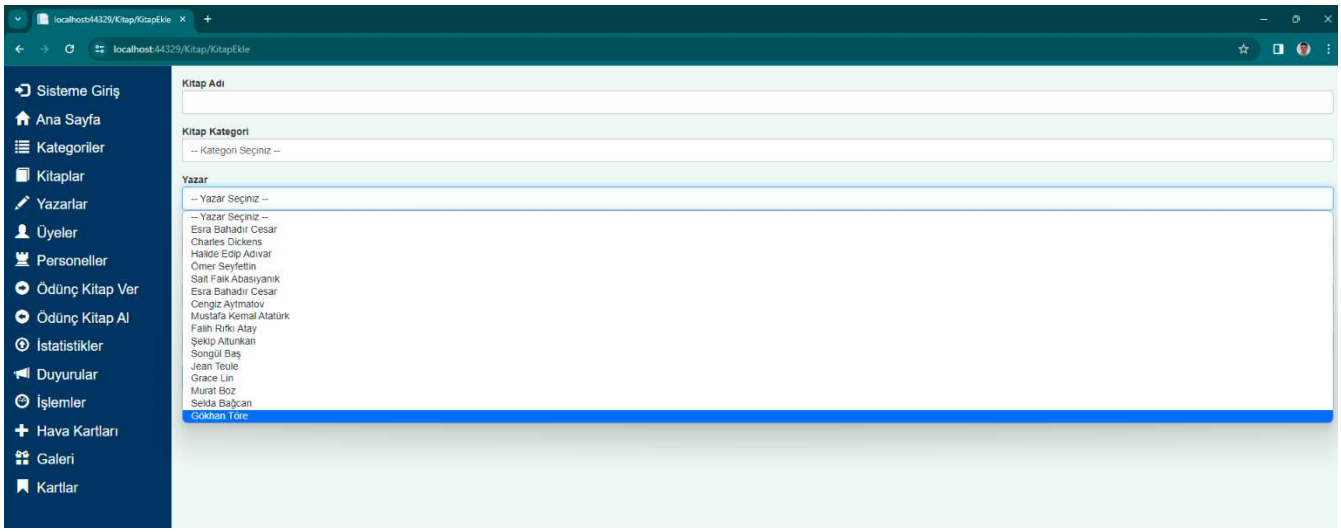


Şekil 24: Kullanıcı kitap hareketleri sayfası.

✓ Ve en son Çıkış Yap sekmesinden ise kişisel kütüphane kullanım alanından çıkabilir.

✓ Hazır CSS lerden yararlanarak, C# ve HTML kullanılarak yazılmış bu kod, kullanıcı panelini ASP.NET Core uygulamasının görünümünü temsil eden bir Razor sayfasını (cshtml dosyasını) içermektedir. Sayfa, web uygulamasının ön yüzünü oluşturur ve kullanıcı arayüzünü tanımlar. Kod ise şu şekildedir: **(Yazının puntosunu arttırarak kodu daha detaylı inceleyebilirsiniz)**

```
@{
    Layout = null;
}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
    <meta name="description" content="" />
    <meta name="author" content="" />
    <title>Profil</title>
    <link href="https://cdn.jsdelivr.net/npm/simple-datatables@7.1.2/dist/style.min.css" rel="stylesheet" />
    <link href="~/webpanel/a/css/styles.css" rel="stylesheet" />
    <script src="https://use.fontawesome.com/releases/v6.3.0/js/all.js" crossorigin="anonymous"></script>
</head>
<body class="sb-nav-fixed">
    <nav  class="sb-topnav navbar navbar-expand navbar-dark bg-dark">
        <!-- Navbar Brand-->
        <a class="navbar-brand ps-3" href="/Panelim/Index/">Profil Başlangıcı</a>
        <!-- Sidebar Toggle-->
        <button class="btn btn-link btn-sm order-1 order-lg-0 me-4 me-lg-0" id="sidebarToggle" href="#!"><i class="fas fa-bars"></i></button>
        <!-- Navbar Search-->
        <form class="d-none d-md-inline-block form-inline ms-auto me-0 me-md-3 my-2 my-md-0">
        </form>
        <!-- Navbar-->
        <ul class="navbar-nav ms-auto ms-md-0 me-3 me-lg-4">
            <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" id="navbarDropdown" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false"><i class="fas fa-
user fa-fw"></i></a>
                <ul class="dropdown-menu dropdown-menu-end" aria-labelledby="navbarDropdown">
                    <li><a class="dropdown-item" href="/Panelim/Index/">Hesabım</a></li>
                    <li><a class="dropdown-item" href="/Vitrin/Index/#about">Hakkımızda</a></li>
                    <li><hr class="dropdown-divider" /></li>
                    <li><a class="dropdown-item" href="/Panelim/LogOut/">Çıkış Yap</a></li>
                </ul>
            </li>
        </ul>
    </nav>
    <div id="layoutSidenav">
        <div id="layoutSidenav_nav">
            <nav class="sb-sidenav accordion sb-sidenav-dark" id="sidenavAccordion">
                <div class="sb-sidenav-menu">
                    <div class="nav">
                        <a class="nav-link" href="/Panelim/Index/">
                            <div class="sb-nav-link-icon"><i class="fas fa-user"></i></div>
                            Profilim
                        </a>
                        <br />
                        <a class="nav-link" href="/Mesajlar/Index/">
                            <div class="sb-nav-link-icon"><i class="fas fa-envelope"></i></div>
                            Gelen Mesajlar
                        </a>
                        <br />
                        <a class="nav-link" href="/Mesajlar/Giden/">
                            <div class="sb-nav-link-icon"><i class="fas fa-paper-plane"></i></div>
                            Gönderilen Mesajlar
                        </a>
                        <br />
                        <a class="nav-link" href="/Mesajlar/YeniMesaj/">
                            <div class="sb-nav-link-icon"><i class="fas fa-pen"></i></div>
                            Yeni Mesaj Oluştur
                        </a>
                        <br />
                        <a class="nav-link" href="/Panelim/Duyurular">
                            <div class="sb-nav-link-icon"><i class="fas fa-bullhorn"></i></div>
                            Duyurular
                        </a>
                        <br />
                        <a class="nav-link" href="/Panelim/Kitaplarım">
                            <div class="sb-nav-link-icon"><i class="fas fa-book"></i></div>
                            Kitaplar
                        </a>
                        <br />
                        <a class="nav-link" href="/Panelim/LogOut/">
                            <div class="sb-nav-link-icon"><i class="fas fa-sign-out"></i></div>
                            Çıkış Yap
                        </a>

                    </div>
                </div>
                <div class="sb-sidenav-footer" >
                    <div class="small">
                        Yardım olarak giriş yaptı:
                    </div>
                    Profil Başlangıcı
                </div>
            </nav>
        </div>
        <div id="layoutSidenav_content">
            <main>
                <div class="container-fluid px-4">
                    @RenderBody()
                </div>
            </main>
            <footer class="py-4 bg-light mt-auto">
                <div class="container-fluid px-4">
                    <div class="d-flex align-items-center justify-content-between small">
                        <div class="text-muted">Copyright &copy; Your Website 2023</div>
                        <div>
                            <a href="#">Privacy Policy</a>
                            &middot;
                            <a href="#">Terms &amp; Conditions</a>
                        </div>
                    </div>
                </div>
            </footer>
        </div>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" crossorigin="anonymous"></script>
    <script src="~/webpanel/a/js/scripts.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.8.0/Chart.min.js" crossorigin="anonymous"></script>
    <script src="~/webpanel/a/assets/demo/chart-area-demo.js"></script>
    <script src="~/webpanel/a/assets/demo/chart-bar-demo.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/simple-datatables@7.1.2/dist/umd/simple-datatables.min.js" crossorigin="anonymous"></script>
    <script src="~/webpanel/a/js/datatables-simple-demo.js"></script>
</body>
</html>
```

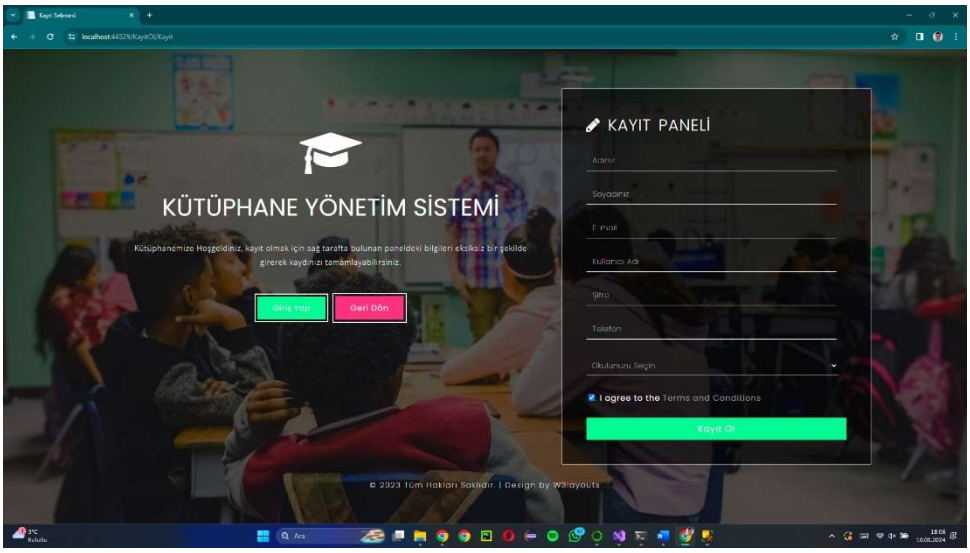


Şekil 25: Kitap ekle sayfası.

✓ Burada kitap eklemek için veri tabanından yazarların isimleri getirilip kitap eklerken yazar seçmek için seçeneklerde listeleme yapılmıştır. Listeleme işlemi ise aşağıdaki kod ile yapılmışıtır.

```
<div style="margin-top:15px">
 <label>Yazar</label>
 @Html.DropDownListFor(k => k.YAZAR,
     new SelectList(yazarListesi.Select(y => new
          {
         ID = y.ID,
         AdSoyad = y.AD + " " + y.SOYAD
     }), "ID", "AdSoyad"),
     "-- Yazar Seçiniz --",
     new { @class = "form-control", id = "YAZAR" })
</div>
```

✓ HTML form sayfasında bir dropdown listesi (seçim kutusu) oluşturur. İşte bu kodun yaptığı işlevlerin açıklamaları:



Şekil 26: Kullanıcı kayıt sayfası.

✓ Yeni kullanıcı kaydı yapmak üzere açılan sayfamızın görüntüsü şekildeki gibidir. Controller kodu aşağıdaki gibidir. **(Yazının puntosunu arttırarak kodu daha detaylı inceleyebilirsiniz)**

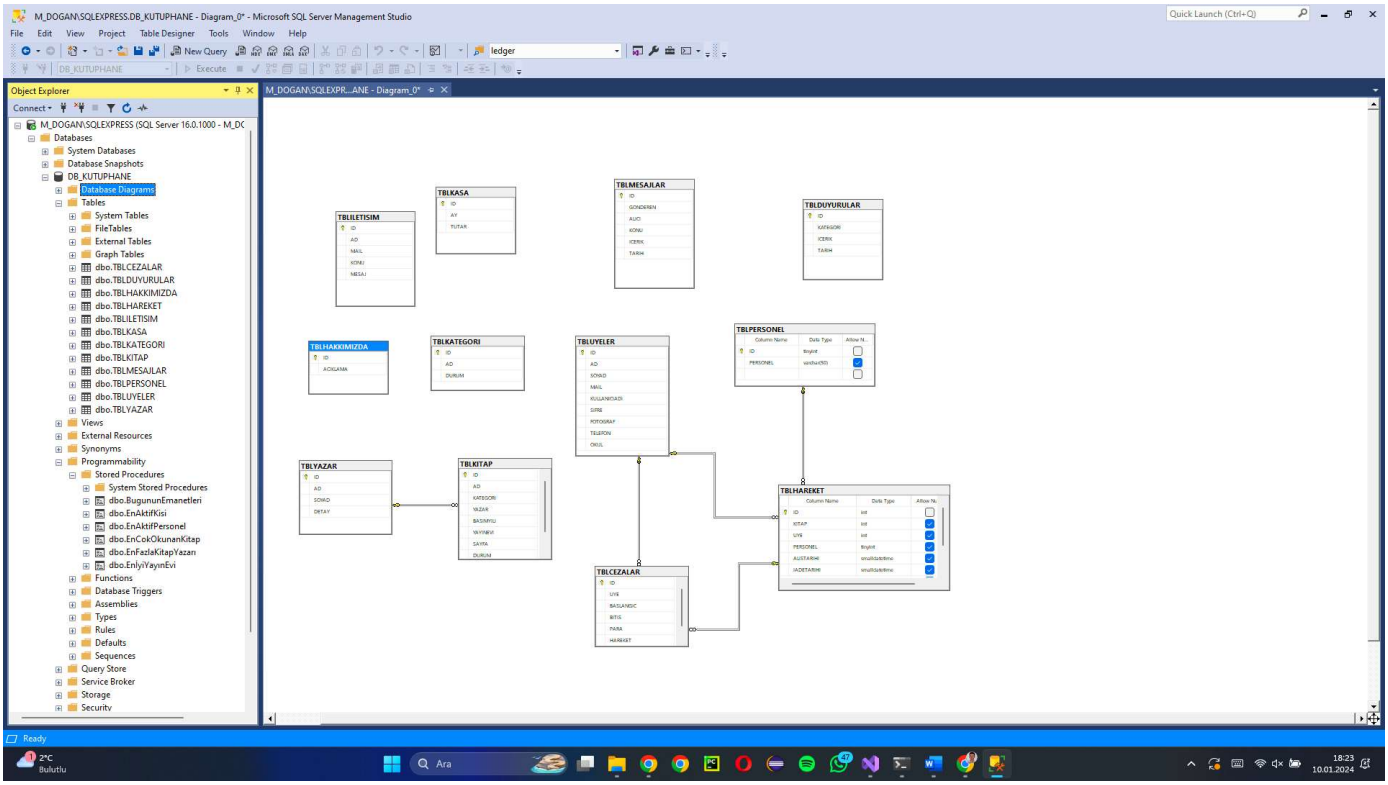
```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Web;
using System.Web.Mvc;
using WebApplication2.Models.Entity;
namespace WebApplication2.Controllers
{
    public class KayitOlController : Controller
    {
        [HttpGet]
        public ActionResult Kayit()
        {
            return View();
        }
        [HttpPost]
        public ActionResult Kayit(TBLUYELER u)
        {
            u.FOTOGRAF = "null";
            using (var httpClient = new HttpClient())
            {
                var json = JsonConvert.SerializeObject(u);
                var content = new StringContent(json, Encoding.UTF8, "application/json");
                var task = httpClient.PostAsync("https://localhost:1433/api/uye/ekle",
content);
                task.Wait();
                var response = task.Result;
                if (response.IsSuccessStatusCode)
                {
                    return View();
                }
                else
                {
                    return View();
                }
            }
        }
    }
}
```

✓ Views kodu ise aşağıdaki gibidir ve seçenek listeleme burada da kullanılmıştır fakat önceki sisteme göre daha farklı şekilde kullanılmıştır. **(Yazının puntosunu arttırarak kodu daha detaylı inceleyebilirsiniz)**

```
@{
    Layout = null;
}
<!--A Design by W3layouts
Author: W3layouts
Author URL: http://w3layouts.com
License: Creative Commons Attribution 3.0 Unported
License URL: http://creativecommons.org/licenses/by/3.0/
-->
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kayıt Sekmesi</title>
    <!-- custom-theme -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <meta name="keywords" content="Course Register Form Responsive web template, Bootstrap Web Templates, Flat Web Templates, Android Compatible web template,
Smartphone Compatible web template, free webdesigns for Nokia, Samsung, LG, SonyEricsson, Motorola web design" />
    <script type="application/x-javascript">
        addEventListener("load", function() { setTimeout(hideURLbar, 0); }, false);
            function hideURLbar(){ window.scrollTo(0,1); } </script>
    <!-- //custom-theme -->
    <!-- css files -->
    <link
href="//fonts.googleapis.com/css?family=Poppins:100,100i,200,200i,300,300i,400,400i,500,500i,600,600i,700,700i,800,800i,900,900i&amp;subset=devanagari,latin-ext"
rel="stylesheet">
    <link href="//fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i,800,800i&amp;subset=cyrillic,cyrillic-ext,greek,greek-ext,latin-
ext,vietnamese" rel="stylesheet">
    <link href="~/webregister/css/style.css" type="text/css" rel="stylesheet" media="all">
    <!-- //css files -->

    <link rel="stylesheet" href="~/webregister/css/font-awesome.css"> <!-- Font-Awesome-Icons-CSS -->
    <script>
        function validateForm() {
            var checkbox = document.querySelector('input[name="checkbox"]');

            if (!checkbox.checked) {
                alert('Şartları kabul etmelisiniz.');
                return false; // Form gönderimini engelle
            }

            return true; // Form gönderimine izin ver
        }
    </script>
</head>
<!-- body starts -->
<body>
    <!-- section -->
    <section class="register">
        <div class="register-full">
            <div class="register-left">
                <div class="register">
                    <div class="logo">
                        <a href="#"><span class="fa fa-graduation-cap" aria-hidden="true"></span></a>
                    </div>
                    <h1>KÜTÜPHANE YÖNETİM SİSTEMİ</h1>
                    <p>
                        Kütüphanemize Hoşgeldiniz, kayıt olmak için sağ tarafta bulunan paneldeki bilgileri eksiksiz bir
                        şekilde girerek kaydınızı tamamlayabilirsiniz.
                    </p>
                    <div class="content3">
                        <ul>
                            <li><a class="" href="/Login/GirisYap/"> Giriş Yap</a></li>
                            <li><a class="read" href="/Vitrin/Index/"> Geri Dön</a></li>
                        </ul>
                    </div>
                </div>
            </div>
            <div class="register-right">
                <div class="register-in">
                    <h2><span class="fa fa-pencil"></span> KAYIT PANELİ</h2>
                    <div class="register-form">
                        <form action="#" method="post">
                            <div class="fields-grid">
                                <div class="styled-input agile-styled-input-top">
                                    <input type="text" name="AD" required="">
                                    <label>Adınız</label>
                                    <span></span>
                                </div>
                                <div class="styled-input agile-styled-input-top">
                                    <input type="text" name="SOYAD" required="">
                                    <label>Soyadınız</label>
                                    <span></span>
                                </div>
                                <div class="styled-input">
                                    <input type="email" name="MAIL" required="">
                                    <label>E-mail</label>
                                    <span></span>
                                </div>
                                <div class="styled-input">
                                    <input type="text" name="KULLANICIADI" required="">
                                    <label>Kullanıcı Adı</label>
                                    <span></span>
                                </div>
                                <div class="styled-input">
                                    <input type="password" name="SIFRE" required="">
                                    <label>Şifre</label>
                                    <span></span>
                                </div>
                                <div class="styled-input">
                                    <input type="tel" name="TELEFON" required="">
                                    <label>Telefon</label>
                                    <span></span>
                                </div>
                                <div class="styled-input agile-styled-input-top">
                                    <select id="OKUL" name="OKUL" required="">
                                        <option value="">Okulunuzu Seçin</option>
                                        <option value="Konya Teknik Üniversitesi">Konya Teknik Üniversitesi</option>
                                        <option value="Selçuk Üniversitesi">Selçuk Üniversitesi</option>
                                        <option value="Necmettin Erbakan Üniversitesi">Necmettin Erbakan Üniversitesi</option>
                                        <option value="KTO Karatay Üniversitesi">KTO Karatay Üniversitesi</option>
                                        <option value="Konya Gıda ve Tarım Üniversitesi">Konya Gıda ve Tarım Üniversitesi</option>
                                    </select>
                                    <span></span>
                                </div>
                                <div class="clear"> </div>
                                <label class="checkbox"><input type="checkbox" name="checkbox" checked="checked"><i></i>I agree to the <a href="#">Terms and
Conditions</a></label>
                            </div>
                            <input type="submit" value="Kayıt Ol">
                        </form>
                    </div>
                </div>
                <div class="clear"> </div>
            </div>
            <div class="clear"> </div>
        </div>
        <!-- copyright -->
        <p class="agile-copyright">© 2023 Tüm Hakları Saklıdır. | Design by <a href="https://w3layouts.com/" target="_blank">W3layouts</a></p>
        <!-- //copyright -->
    </section>
    <!-- //section -->
</body>
<!-- //body ends -->
</html>
```

- ✓ Giriş ekranı frontend kısmındaki kodun amacı ve belirli komutların işlevi ise şu şekildedir:
    - o @{ Layout = null; }: Bu satır, Razor sayfasının layout kullanmadığını belirtir. Yani, sayfa kendi başına bir HTML belgesi oluşturur. Bu, sayfanın başka bir şablona bağlı olmadan görüntülenmesini sağlar.
    - o <!DOCTYPE html>: HTML belgesinin tipini belirtir. Bu satır, sayfanın HTML5 belgesi olduğunu gösterir.
    - o <html lang="en">: HTML belgesinin başladığını ve İngilizce dilini kullanacağını belirtir. "lang" özelliği, sayfanın dilini belirler.
    - o <head>: Sayfanın başlık ve meta bilgilerini içeren başlık bölümünü başlatır. Başlık bölümü, sayfanın tarayıcı sekmesinde görünen metni içerir.
    - o <title>Kayıt Sekmesi</title>: Tarayıcı sekmesinde görünen sayfa başlığını belirtir. Bu başlık, sayfanın kullanıcıya ne hakkında olduğunu anlatır.
    - o <link> ve <script> etiketleri: Bu etiketler, sayfanın stil ve betik (script) kaynaklarını içerir. Örneğin, kullanılan yazı tiplerini ve sayfanın tasarımını düzenleyen stil dosyalarını içerir.

VERİ TABANI TASARIMI:



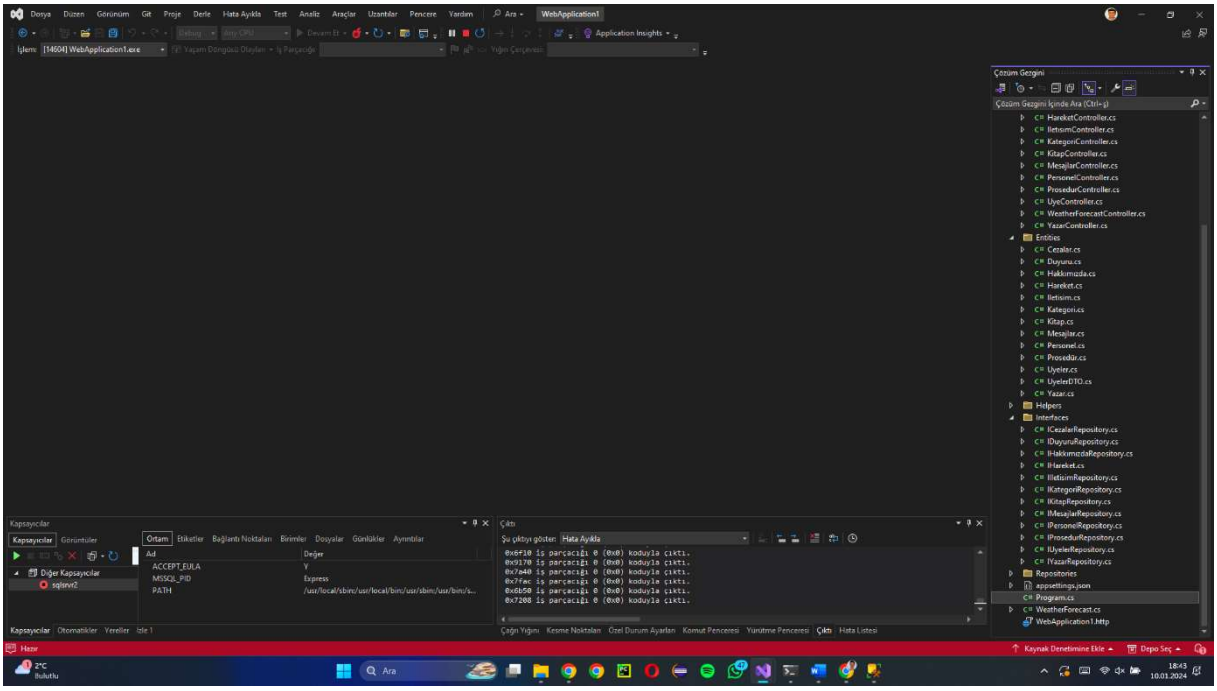
Şekil 27: Veri tabanı tasarımı ve diyagram gösterimi.

- Veri tabanı olarak Microsoft SQL server kullanılmıştır sebepleri ise şu şekildedir:

- ✓ **Veri Saklama ve Yönetim:** Microsoft SQL Server, verilerin saklanması, yönetilmesi ve işlenmesi için kullanılan güçlü bir RDBMS'dir. Veriler, tablolar ve sütunlar şeklinde yapılandırılır. Kolay erişilebilir ve değiştirilebilirdir.
- ✓ **Yüksek Performans:** SQL Server, yüksek performanslı veri tabanı işlemleri için optimize edilmiştir. Verilerin hızlı erişimi ve işlenmesi için çeşitli indeksleme ve sorgu optimizasyon özelliklerine sahiptir. (Prosedür, Trigger)
- ✓ **Veri Güvenliği:** SQL Server, güvenlik önlemleri ve erişim denetimleri ile verileri korur. Şifreleme yapılabilir, kullanıcıların ve rollerin belirlenmesi, güvenlik duvarları ve denetim günlükleri gibi güvenlik özellikleri sunar.
- ✓ **İş Zekası ve Veri Analitiği:** SQL Server, iş zekası ve veri analitiği uygulamaları için mükemmel bir platform sağlar. Özel raporlama araçları, veri madenciliği ve iş zekası için entegrasyon özellikleri sunar.
- ✓ **Uyumluluk:** WEB uygulamasını geliştirirken kullanmış olduğum Visual Studio IDE sine de uygun bir yapısı vardır.

- Veri tabanımda toplamda 6 sı ilişkisel olmak üzere 12 tablodan ve 6 prosedür, 3 triggerdan oluşmaktadır. Genel olarak her tabloda ID değişkeni vardır primary key olarak ve otomatik artan olarak atanmıştır. Daha fazlası UML diyagramında verilmiştir.

API KULLANIMI



- .NET core üzerinde geliştirilen WEB API sisteminde genel olarak 7 bölümden oluşmaktadır:

1. Bölüm Controller' lar:

- ✓ **Örnek CezaController:**

```
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Interfaces;

namespace WebApplication1.Controllers
{
    [ApiController]
    [Route("api/cezalar")]
    public class CezalarController : Controller
    {
        private readonly ICezalarRepository _cezalarRepository;
        public CezalarController(ICezalarRepository cezalarRepository)
        {
            _cezalarRepository = cezalarRepository;
        }
        [HttpGet]
        public IActionResult GetCezalar()
        {
            try
            {
                var cezalar = _cezalarRepository.GetCezalar();
                return Ok(cezalar);
            }
            catch (Exception ex)
            {
                return BadRequest(ex.Message);
            }
        }
    }
}
```

Açıklama: C# ile bir ASP.NET Core Web API uygulamasının denetleyici (controller) sınıfını temsil eden kontrol classı.

1. **using İfadeleri:** Bu bölüm, gerekli namespace'leri içeri aktarır.
2. **CezalarController Sınıfı:** Bu sınıf, bir denetleyici sınıfını temsil eder ve ASP.NET Core Web API projesinin bir parçasıdır. Controller sınıfından türetilir.
3. **Constructor Metodu:** CezalarController sınıfının constructor metodu, bu sınıfın başlatılmasını yönetir. ICezalarRepository arayüzü tipinde bir bağımlılık (_cezalarRepository) bu sınıfa enjekte edilir.
4. **[HttpGet] Niteliği:** Bu nitelik, bu denetleyici yönteminin bir HTTP GET isteğine yanıt vereceğini belirtir.
5. **GetCezalar Metodu:** Bu metot, HTTP GET isteği aldığında çalışır. İşte bu metotun işlevleri:
    - **Try-Catch Bloğu:** Bir hata durumunda istemciden dönen hata mesajıyla birlikte bir HTTP 400 Bad Request yanıtı gönderilir.
    - **_cezalarRepository.GetCezalar() Çağrısı:** _cezalarRepository üzerinden GetCezalar adlı bir metot çağrısı yapılır. Bu metot, veri tabanından ceza bilgilerini çekmelidir.
    - **return Ok(cezalar) Satırı:** Eğer herhangi bir hata oluşmazsa, ceza bilgileri Ok() metodu ile birlikte gönderilir. Bu, bir HTTP 200 OK yanıtı ile birlikte veriye sahip bir yanıtı temsil eder.

2. Entities'ler:

✓ **Örnek Cezalar:**

```
namespace WebApplication1.Entities
{
    public class Cezalar
    {
        public int ID { get; set; }
        public Nullable<int> UYE { get; set; }
        public Nullable<System.DateTime> BASLANGIC { get; set; }
        public Nullable<System.DateTime> BITIS { get; set; }
        public Nullable<decimal> PARA { get; set; }
        public Nullable<int> HAREKET { get; set; }
    }
}
```

Açıklama: Yukarıdaki kodumuz ise C# sınıfı, TBLCEZALAR veri tabanı tablosunun modelini temsil eder. Bu sınıflar veri tabanı işlemlerini (veri okuma, yazma, güncelleme, silme) ve uygulama mantığını kolaylaştırmak için kullanılmıştır.

1. **public int ID { get; set; }:** Bu özellik, ceza kaydının benzersiz bir tanımlayıcısını (ID) temsil etmketedir. Veri tabanı tablosundaki birincil anahtar alanına karşılık gelmektedir.
2. **public Nullable<int> UYE { get; set; }:** Bu özellik, ceza kaydının hangi üye tarafından oluşturulduğunu temsil eder. Üye ID'si, ilişkilendirilmiş bir üye tablosuna bağlanıp ve bu şekilde üyelerin cezaları izlenebilmektedir. Nullable<int> kullanılmasının nedeni, bu alanın null olabileceğini belirtmek içindir.
3. **public Nullable<System.DateTime> BASLANGIC { get; set; }:** Bu özellik, cezanın başlangıç tarihini temsil etmektedir. Bir üye ne zaman cezalandırıldıysa, bu tarih burada saklanır.

4. Helpers:

- ✓ **Bir tane Helrpers vardır ConnectionHelpers:**

```
using Microsoft.Data.SqlClient;
using System.Data;

namespace WebApplication1.Helpers
{
    public class ConnectionHelper
    {
        private readonly IConfiguration _configuration;
        private readonly string _connectionString;
        public ConnectionHelper(IConfiguration configuration)
        {
            _configuration = configuration;
            _connectionString =
_configuration.GetConnectionString("SqlConnection");
        }
        public IDbConnection CreateSqlConnection() => new
SqlConnection(_connectionString);
    }
}
```

Açıklama: ConnectionHelpers C# sınıfı, bir ASP.NET Core uygulamasında veri tabanı bağlantısı oluşturmak için oluşturduğum bir yardımcı sınıftır.

1. **using Microsoft.Data.SqlClient;:** Bu komut, Microsoft.Data.SqlClient adlı namespace'i içeri aktarmak içindir. Bu namespace, SQL Server veri tabanıyla etkileşimde bulunmak için kullanılan sınıfları içerektedir.
2. **using System.Data;:** Bu ifade ise, System.Data namespace'ini içeri aktarmak içindir. Bu namespace, ADO.NET (ActiveX Data Objects .NET) ile ilgili sınıfları içerir ve veri tabanı işlemleri için kullanılır.

3. **private readonly IConfiguration _configuration;:** Bu satır, _configuration adlı bir IConfiguration örneğini sınıf içinde özel olarak saklar. IConfiguration, uygulama yapılandırma verilerini okumak için kullanılır.
4. **private readonly string _connectionString;:** Bu satır, _connectionString adlı bir dizeyi sınıf içinde özel olarak saklar. Bu dize, SQL Server veritabanına bağlanmak için kullanılacak bağlantı dizesini içerir.

5. Interfaces:

- ✓ **ICezalarRepository:**

```
using WebApplication1.Entities;

namespace WebApplication1.Interfaces
{
    public interface ICezalarRepository
    {
        public IEnumerable<Cezalar> GetCezalar();

    }
}
```

Açıklama: Bu C# arayüzü (interface), bir veri tabanı veya veri kaynağından ceza bilgilerini çekmek amacıyla kullanılacak bir repository (veritabanı erişim sınıfı) tanımlar. İşte bu ICezalarRepository arayüzünün işlevi:

1. **using WebApplication1.Entities;:** Bu ifade, WebApplication1.Entities namespace'ini içeri aktarır. Bu namespace, Cezalar sınıfını içerir ve bu sınıf, ceza kayıtlarının modelini temsil eder.
2. **public interface ICezalarRepository:** Bu satırda ICezalarRepository adlı bir C# arayüzü tanımlanır. Arayüz, ceza verilerini çekmek için kullanılacak metotları ve işlevleri içerir.
3. **public IEnumerable<Cezalar> GetCezalar();:** Bu arayüz içinde tanımlanan tek metot, GetCezalar adını taşır ve geriye IEnumerable<Cezalar> türünde bir koleksiyon döndürür. Bu metot, ceza verilerini çekmek için kullanılacak ve bu verileri bir koleksiyon içinde döndürmek içindir.

6. Repositories:

- ✓ **CezalarRepository:**

```
using Dapper;
using WebApplication1.Entities;
using WebApplication1.Helpers;
using WebApplication1.Interfaces;

namespace WebApplication1.Repositories
{
    public class CezalarRepository : ICezalarRepository
    {
        private readonly ConnectionHelper _connectionHelper;
        public CezalarRepository(ConnectionHelper connectionHelper)
        {
            _connectionHelper = connectionHelper;
        }
        public IEnumerable<Cezalar> GetCezalar()
        {
            var query = "SELECT * FROM TBLCEZALAR";
            using var connection = _connectionHelper.CreateSqlConnection();
            var cezalar = connection.Query<Cezalar>(query);
            return cezalar.ToList();
        }
    }
}
```

Açıklama: Yukarıdaki kod C# sınıfı, bir ASP.NET Core uygulamasında ICezalarRepository arayüzünü uygulayan bir repository (veritabanı erişim sınıfı) sınıfını temsil eder. Bu sınıf, ceza verilerini bir SQL veritabanından çekmek için kullanılır.

1. **using Dapper;:** Bu fonksiyon, Dapper adlı bir ORM (Object-Relational Mapping) kütüphanesinin kullanılarak işlem yapar. Dapper, veritabanı işlemlerini kolaylaştıran ve veri nesnelerini veri tabanı kayıtlarıyla eşleştirmek için kullanılan bir kütüphanedir.
2. **using WebApplication1.Entities;:** WebApplication1.Entities namespace'ini içeri aktarır. Bu namespace, Cezalar sınıfını içerir ve bu sınıf, ceza kayıtlarının modelini temsil eder.
3. **using WebApplication1.Helpers;:** WebApplication1.Helpers namespace'ini içeri aktarır. Bu namespace, veritabanı bağlantısı oluşturmak için kullanılan ConnectionHelper sınıfını içerir.
4. **using WebApplication1.Interfaces;:** WebApplication1.Interfaces namespace'ini içeri aktarır. Bu namespace, ICezalarRepository arayüzünü içerir ve bu arayüz, ceza verilerini çekmek için metotları tanımlar.
5. **public class CezalarRepository : ICezalarRepository:** Bu satırda CezalarRepository adlı bir sınıf tanımlanır ve ICezalarRepository arayüzünü uygular. Bu sınıf, ceza verilerine erişmek ve işlemek için kullanılır.
6. **private readonly ConnectionHelper _connectionHelper;:** Bu satırda _connectionHelper adlı bir ConnectionHelper örneği sınıf içinde özel olarak saklanır. Bu, veritabanı bağlantısını oluşturmak için kullanılacak yardımcı bir sınıftır.
7. **public CezalarRepository(ConnectionHelper connectionHelper):** Bu, sınıfın yapıcı (constructor) metodu olup, bağımlılıkları enjekte etmek için kullanılır.

ConnectionHelper örneği enjekte edilir ve bağlantı oluşturmak için kullanılacak sınıfa erişim sağlar.

8. **public IEnumerable<Cezalar> GetCezalar():** Bu metot, ICezalarRepository arayüzünden uygulanan bir metottur. Bu metot, SQL veritabanından ceza verilerini çekmek için kullanılır. İşte metotun işleyişi:
    - Bir SQL sorgusu olan var query = "SELECT * FROM TBLCEZALAR";
    - using bloğu içinde bir SQL bağlantısı oluşturulur ve _connectionHelper.CreateSqlConnection() metodu kullanılarak bağlantı açılır.
    - Dapper kütüphanesi kullanılarak SQL sorgusu çalıştırılır ve sonuç ceza verilerini içeren bir koleksiyon olarak döndürülür.

7.appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "SqlConnection":
"Server=M_DOGAN\\SQLEXPRESS;Database=DB_KUTUPHANE;Trusted_Connection=True;;
TrustServerCertificate=True"
  }
}
```

Açıklama: Yukarıdaki kod JSON yapılandırma dosyasından alınmıştır, ASP.NET Core uygulamasının yapılandırma ayarlarını içermektedir. Bu ayarlar, uygulamanın çalışma zamanında davranışını ve ayarlarını belirlemesi için yazılmıştır.

1. **"Logging":** Bu bölüm, uygulamanın kayıt (log) düzeylerini yapılandırmak için kullanılır. ASP.NET Core uygulamaları, çeşitli olayları ve hataları kaydetmek için loglama kullanabilirler. Bu bölümde Default ve Microsoft.AspNetCore loglama kategorileri belirtilmiştir.
    - **"LogLevel":** Bu özellik, belirli loglama kategorileri için varsayılan log seviyesini belirtir.
2. **"AllowedHosts":** Bu özellik, uygulamanın hangi ana bilgisayar adlarını veya IP adreslerini kabul edeceğini belirtir. "*" işareti, tüm ana bilgisayarlar ve IP adreslerini kabul edeceğini belirtir.
3. **"ConnectionStrings":** Bu bölüm, uygulamanın kullanacağı veri tabanı bağlantı dizesini içerir. Bu veri tabanı bağlantı dizesi, SQL Server veri tabanına bağlanmak için kullanılacak olan bilgileri içerir.

    Özellikle şunları belirtir:
    - **"Server":** Veri tabanı sunucusunun adını veya bağlantı dizesini içerir. Kodumda ise "M_DOGAN\SQLEXPRESS" olarak belirtilmiştir.
    - **"Database":** Bağlanılacak veri tabanının adını içerir. Kodumda ise "DB_KUTUPHANE" olarak belirtilmiştir.

- **"Trusted_Connection":** Windows kimlik doğrulama kullanılıp kullanılmayacağını belirtmektedir. Kodumda "True" olarak ayarlandığı için Windows kimlik doğrulama kullanılır.
- **"TrustServerCertificate":** Güvenli bir sunucu sertifikası kullanılıp kullanılmayacağını belirtir. "True" olarak ayarlandığım için, sistem sunucu sertifikası güvence altına alınır.

API sisteminde veriler JSON formatında alınıp gönderilmektedir. Bu sistem için

```
                var json = JsonConvert.SerializeObject(p);
                var content = new StringContent(json, Encoding.UTF8,
"application/json");
```

kod parçacığında verilen `JsonConvert.SerializeObject` fonksiyonu ile serileştirdikten sonra `StringContent(json, Encoding.UTF8, "application/json");` fonksiyonu ile json formatına çevirerek veri tabanına gönderdim veri tabanından veri okurken de tam tersini uyguladık.