yazı tipi, logo, simge, sembol, daire içeren bir resim

Açıklama otomatik olarak oluşturuldu

**GAZİ ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**BMT 308**

**WEB PROGRAMLAMA**

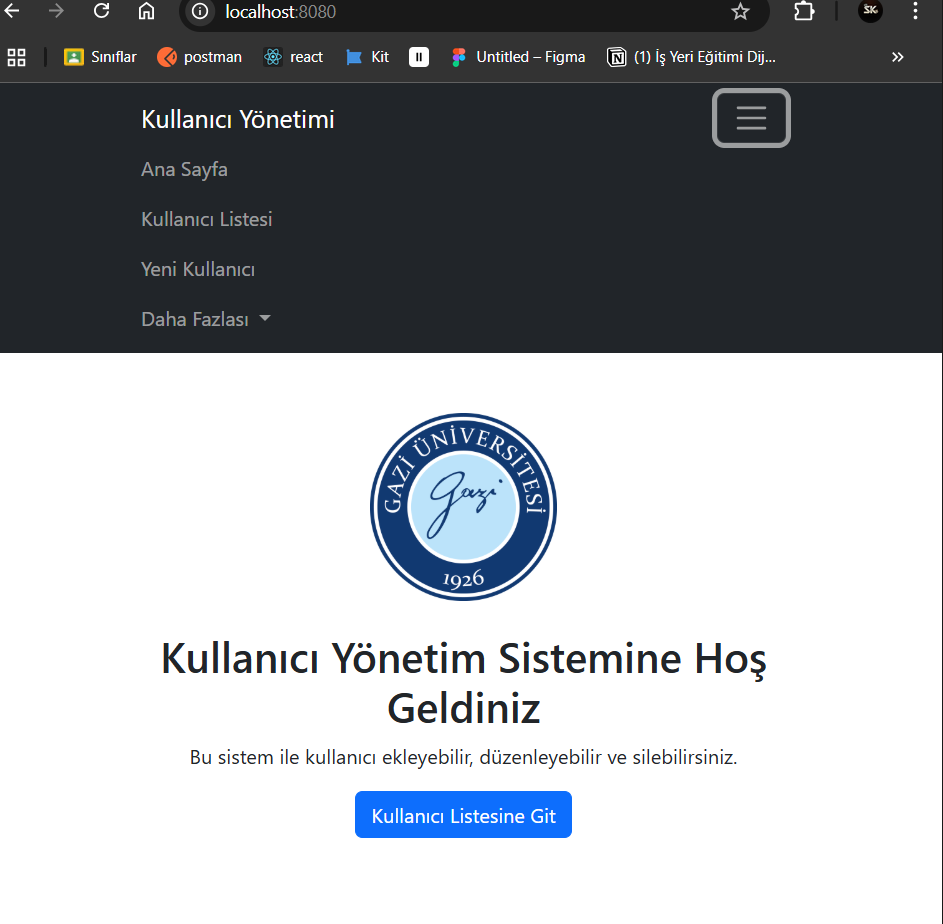
**LAB 5 RAPORU**

**2024-2025 BAHAR YARIYILI**

**Salih Karakaya**

**191816775**

**Doç. Dr. SİNAN TOKLU**

****

**metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**metin, yazılım, multimedya yazılımı, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**metin, yazı tipi, sayı, numara, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**metin, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

1. **Thymeleaf’te kullanılan action, object, each, field özellikleri kullanımı**

**action Kullanımı**  
metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**th:action**, bir HTML formunun veriyi hangi URL’ye göndereceğini belirtmek için kullanılır.

**object Kullanımı**

metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**th:object**, form içindeki nesnenin bağlanacağı model nesnesini belirtir.

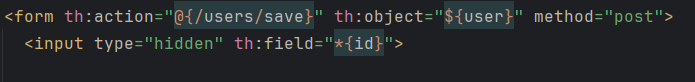
**each Kullanımı**

**metin, ekran görüntüsü, yazı tipi, multimedya yazılımı içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**th:each,** bir liste veya koleksiyonu döngüyle dönerek içindeki her öğeyi tek tek işler.

**field Kullanımı**

****

**th:field,** bir form alanını bir nesnenin belirli bir özelliğine bağlar.

**For exp. ->** th:field="\*{id}" → Kullanıcının **id** özelliğine bağlanır.

Spring Model içinde UserDTO nesnesi varsa, form alanları otomatik olarak bu nesneye bağlanır.

1. **Thymeleaf sayfa yapısı için Controller’da ilgili metodların ayrı ayrı oluşturulması ve Model yapısı kullanımı**

Thymeleaf, Spring MVC ile dinamik HTML sayfaları oluşturmak için kullanılan bir şablon motorudur.  
Bu yapıda, Controller içinde her sayfa için ayrı bir metod oluşturmalı ve Model nesnesini kullanarak Thymeleaf’e veri göndermeliyiz.

**Controller**

metin, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.  
  
Service katmanı, iş mantığını içeren ve Repository katmanına erişimi yöneten katmandır.

Spring Model nesnesi, Controller’dan Thymeleaf’e **veri göndermek** için kullanılır.

**Örnek Kullanım:**

model.addAttribute("key", value);

**key → Thymeleaf sayfasında kullanılacak değişken adı.**  
**value → Controller’dan gelen değer.**

1. **Yeni kayıt sayfası hem Controller üzerinden hem Thymeleaf üzerinden gerçekleştirilmesi**  
     
   **Thymeleaf**

* **templates/form.html** (Yeni Kullanıcı Ekleme / Düzenleme Sayfası)

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

ekran görüntüsü, metin içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Kullanıcı ekleme ve düzenleme işlemi için th:object ve th:field kullanıldı.  
Veri kaydetmek için @{/users/save} ile POST isteği gönderildi.  
Eğer edit metodu çağrılırsa, form düzenleme için de kullanılabilir.

* **templates/list.html** (Kullanıcı Listeleme)

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Thymeleaf içinde Model nesnesinden gelen users listesini th:each ile döndük.  
CRUD işlemleri için Düzenle ve Sil butonları eklendi.

1. **Thymeleaf için $,#,@,~ yapıları uygulamlarda ayrı ayrı kullanımı**

**$ (Değişken Erişimi)**

<p th:text="${user.name}"></p> <!-- Model içindeki `user` nesnesinin `name` alanı gösterilir -->

<p th:text="${user.email}"></p> <!-- Kullanıcının email adresi gösterilir -->

$ işareti, Model nesnesindeki değişkenlere erişmek için kullanılır.

**# (Özel Nesneler ve Fonksiyonlar)**

<p th:text="#{app.title}"></p> <!-- `messages.properties` içinde `app.title=Uygulama Başlığı` ise ekrana yazar -->

# işareti, Thymeleaf'in özel nesnelerine ve fonksiyonlarına erişmek için kullanılır.

**@ (Bağlantı ve URL İşlemleri)**

<a th:href="@{/users/edit/{id}(id=${user.id})}">Düzenle</a>

<a th:href="@{/users/delete/{id}(id=${user.id})}">Sil</a>

@ işareti, URL ve yönlendirmeleri Thymeleaf içinde oluşturmak için kullanılır.

Dinamik URL oluşturma işlemleri için @{} formatında kullanılır.

Menüde veya sayfalar arası geçişte @{} kullanılarak linkler oluşturulabilir;

<nav>

<a th:href="@{/}">Ana Sayfa</a>

<a th:href="@{/users}">Kullanıcı Listesi</a>

<a th:href="@{/users/new}">Yeni Kullanıcı</a>

</nav>

**~ (Statik Dosyalara Erişim)**

<link rel="stylesheet" th:href="@{/css/style.css}">

Bu kullanım /static/css/style.css dosyasına bağlanır.

<script th:src="@{/js/script.js}"></script>

Bu kullanım /static/js/script.js dosyasına bağlanır.

<img th:src="@{/images/logo.png}" alt="Logo">

Bu kullanım /static/images/logo.png dosyasını gösterir.

~ işareti, statik dosyaları (CSS, JS, Resimler) Thymeleaf içinde çağırmak için kullanılır.

1. **Bootstrap’tan NavBar çekilme ve gösterimleri bu NavBar konseptine uygun gerçekleştirilmesi**

**Bootstrap ile NavBar Kullanımı**

Bootstrap 5 kullanılarak duyarlı (responsive) bir NavBar oluşturulmuştur.

Mobil uyum için navbar-toggler ve collapse sınıfları kullanılmıştır.  
Ana sayfa, kullanıcı listesi ve yeni kullanıcı ekleme gibi menü öğeleri eklenmiştir.

metin, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

NavBar, tüm sayfalarda ortak olarak kullanılması için fragments/navbar.html dosyasında tanımlanmıştır.

**Thymeleaf ile NavBar’ı Sayfalara Dahil Etme**

NavBar tüm Thymeleaf şablonlarına th:replace ile eklenmiştir.  
Bu sayede sayfa başına tek tek NavBar kodu yazılmamış, tekrar eden kodlar önlenmiştir.

<div th:replace="~{fragments/navbar :: nav}"></div>

**Dropdown Menü ile Ekstra Sayfalar (Hakkımızda, İletişim)**

"Daha Fazlası" adlı bir Dropdown menü eklenerek, ekstra sayfalara yönlendirme sağlanmıştır.  
Dropdown içinde "Hakkımızda" ve "İletişim" sayfaları bulunmaktadır.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

1. **Mysql Veri Tabanı kullanımı**

**metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**metin, yazılım, bilgisayar simgesi, web sayfası içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

1. **Bütün işlemleri DTO(Data Transfer Object Sınıfı ve Mapper Sınıfı) kullanılarak yapılması**

**UserMapper**

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**metin, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**DepartmentMapper**

**metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

1. **Validation İşlemleri**

Validation işlemleri, kullanıcıdan alınan verilerin doğruluğunu ve geçerliliğini sağlamak için kullanılan önemli bir adımdır. Bu işlemler, uygulamanın güvenliğini artırmak ve hatalı veri girişlerini engellemek için kritik bir rol oynar. Proje kapsamında, kullanıcı girişlerini doğrulamak için Spring Framework'ün sağladığı bean validation özelliklerinden yararlanılmıştır.

**a. Spring Validation Kullanımı**

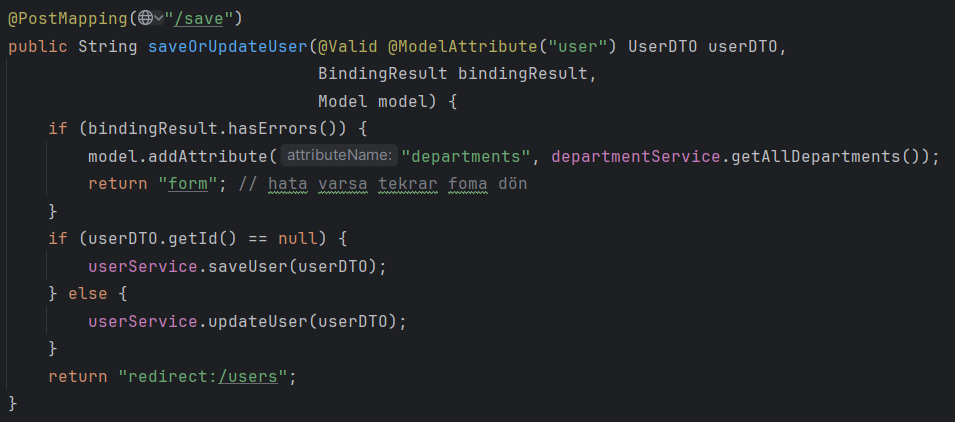
Spring, javax.validation kütüphanesi üzerinden sağlanan @Valid ve @NotNull, @Email, @Size gibi annotasyonlarla veri doğrulama işlemleri yapılmasını sağlar. Bu, uygulamaya entegre edilen modellerin doğruluğunu garanti eder.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**b. BindingResult ile Hata Yönetimi**

Spring, model verisi üzerinde yapılan doğrulama işlemleri sonrasında, BindingResult objesi ile doğrulama hatalarını yönetmemize imkan tanır. Eğer doğrulama hatası varsa, form üzerinde kullanıcıya hata mesajları gösterilebilir. Aşağıda kullanıcı formunda doğrulama hatası olursa, tekrar kullanıcı formuna dönülmesi örneği verilmiştir:



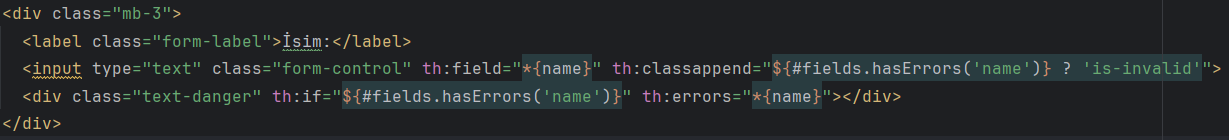
**c. Kullanıcı Girişi ve Adres Validation**

Kullanıcı girişindeki alanlar ve adres bilgileri için aşağıdaki gibi doğrulama işlemleri gerçekleştirilmiştir:

* **İsim** alanı için **@NotNull** ve **@Size** annotasyonları kullanılarak boş ya da çok kısa isimlerin girilmesi engellenmiştir.
* **Email** alanı için **@Email** annotasyonu ile geçerli bir e-posta adresi formatı sağlanmıştır.
* **Adres bilgileri** için, şehir, sokak ve posta kodu gibi alanlar için benzer şekilde doğrulamalar yapılmıştır.

#### **d. Hata Mesajları ve Kullanıcı Dostu Bildirimler**

Doğrulama hataları kullanıcıya anlaşılır bir biçimde iletilir. Kullanıcı formunda hata oluştuğunda, hata mesajları ilgili alanın altında dinamik olarak görüntülenir. Örnek:



Bu sayede kullanıcı, formu hatalı doldurduğunda neyi düzeltmesi gerektiği hakkında bilgilendirilir.

**e. Global Validation**

Bazı durumlarda, sadece belirli bir alanın değil, tüm formun doğruluğunu kontrol etmek gerekebilir. Bunun için **@Valid** anotasyonu ile modelin tüm alanları kontrol edilir. Bu, doğrulama hatalarını sistematik olarak yönetmeyi sağlar.

1. **Embeddable ve Foreign Key kullanımı**

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**metin, yazı tipi, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**