



AD: BEKTAŞ

SOYAD: ERGİN

NO: HR220031

KONU: BekEmlak Projesi

İçindekiler :

1. BekEmlak Projesinin Özeti
2. Veritabanı Diagramı
3. Veritabanı Örnek Datalar
4. Login Page (Fotoğraf)
5. Login Page (Frontend Kodları)
6. Login Page (Backend Kodları)
7. Register Page (Fotoğraf)
8. Register Page (Frontend Kodları)
9. Register Page (Frontend Kodları)
10. Register Page (Backend Kodları)
11. Forgot Page (Fotoğraf)
12. Forgot Error Page (Fotoğraf)
13. Forgot Page (Email Girme Frontend Kodları)
14. Forgot Page (Email Girme Backend Kodları)
15. Forgot Page (Email Gelen Kodu Girme Frontend Kodları)
16. Forgot Page (Email Gelen Kodu Girme Backend Kodları)
17. Forgot Page (Yeni Şifre Girme Frontend Kodları)
18. Forgot Page (Yeni Şifre Girme Backend Kodları)
19. Forgot Error Page (Email hatası Frontend Kodları)
20. Forgot Error Page (Kod Hatası Frontend Kodları)
21. Anasayfa (index.html) ve anaSayfaLoginSonrasi (Fotoğraflar)
22. Anasayfa (index.html) (Frontend Kodu)
23. anaSayfaLoginSonrası Page (Frontend Kodu)
24. ilanEkle Page (Fotoğraf ve Frontend Kodu)
25. ilanEkle Page (Frontend Kodu)
26. ilanEkle Page (Backend Kodu)
27. ilanSilme Page (Fotoğraf ve Frontend Kodu)
28. ilanSilme Page (Backend Kodu)
29. ilanGuncelle Page (Fotoğraf ve Frontend Kodu)
30. ilanGuncelle Page (Backend Kodu)

ÖZET

BekEmlak projesi adındanda öğrenileceği gibi bir emlak projesidir. Kullanıcılarımız web sitemize girdikleri zaman birbirinden güzel ilanlar karşlarına çıkmaktadır ve dilekdikleri ilanı kolaylıkla bulabilirler. İstedikleri ilanla alakalı bilgi almak için açıklama kısmını okuyabilir olmadı kayıtlı telefon numarasından ilan sahibini arayarak bilgi edinebilirler. Projemiz de backend kodları için Spring boot java bilgisayar dili kullanılmıştır frontend kodları için ise html, css ve javascript bilgisayar dili kullanılmıştır. Database dili olarak ise mssql tercih edilmiştir. Apilerin kontrolu için swagger-ui arayüzü ve postman arayüzü kullanılmıştır. Kodlarımızı yazmak için IntelliJ kullanılmıştır. Database işlemleri için ise SQL Management Studio kullanılmıştır.

Veri Tabanı Diagramları :

Users			
	Column Name	Data Type	Allow Nulls
🔑	userid	int	<input type="checkbox"/>
	username	nvarchar(50)	<input checked="" type="checkbox"/>
	password	nvarchar(50)	<input checked="" type="checkbox"/>
	name	nvarchar(50)	<input checked="" type="checkbox"/>
	surname	nvarchar(50)	<input checked="" type="checkbox"/>
	email	nvarchar(50)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Announcements			
	Column Name	Data Type	Allow Nulls
🔑	announcementid	int	<input type="checkbox"/>
	name	nvarchar(200)	<input checked="" type="checkbox"/>
	price	float	<input checked="" type="checkbox"/>
	location	nvarchar(MAX)	<input checked="" type="checkbox"/>
	explanation	nvarchar(MAX)	<input checked="" type="checkbox"/>
	image	varbinary(MAX)	<input checked="" type="checkbox"/>
	phone	nvarchar(MAX)	<input type="checkbox"/>
			<input type="checkbox"/>

Veritabanı Örnek Datalar :

	userid	username	password	name	surname	email
▶	1	bektasergin	Bektas123456	Bektaş	ERGİN	erginbeko34@g...
	2	irem.ergin	İrem12345	İrem	ERGİN	iremergin58@g...
•	NULL	NULL	NULL	NULL	NULL	NULL

[illegible]

E1 – Login Page

Üye Girişi

E-posta:

Şifre:

Giriş Yap

[Şifremi Unuttum](#)

Üye değil misiniz ?

[Hemen Üye Olun](#)

Frontend Kodları ;

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>BekEmlak Login Page</title>
6      <link rel="stylesheet" href="assets/loginPageStyle.css">
7      <script>
8          window.onload = function() {
9              document.getElementById("loginButton").addEventListener("click", function() {
10                  var email = document.getElementById("email").value;
11                  var password = document.getElementById("password").value;
12
13                  var xhr = new XMLHttpRequest();
14                  xhr.open("POST", "/login", true);
15                  xhr.setRequestHeader('Content-Type', 'application/json');
16                  xhr.onreadystatechange = function() {
17                      if (xhr.readyState === XMLHttpRequest.DONE) {
18                          if (xhr.status === 200) {
19                              alert("Giriş başarılı!")
20                              window.location.href = "index.html";
21                          } else {
22
23                              alert(xhr.responseText);
24                          }
25                      }
26                  };
27                  var data = JSON.stringify({
28                      email: email,
29                      password: password
30                  });
31                  xhr.send(data);
32              });
33          };
34      </script>
35  </head>

```

```

36  <body>
37  <div class="ustKutu"></div>
38
39  <div class="loginKutusu">
40      <h2 style="background-color: #f0f0f0;">Üye Girişi</h2>
41      <div>
42          <label class="email" for="email">E-posta: </label>
43          <input type="email" id="email" name="email" required>
44
45          <br>
46          <br>
47
48          <label class="password" for="password">Şifre: </label>
49          <input type="password" id="password" name="password" required>
50
51          <br>
52          <br>
53
54          <a href="#" id="loginButton" class="loginButton" >Giriş Yap</a>
55
56          <div/>
57          <br>
58
59          <a href="forgotPassword.html" class="forgotPassword" >Şifremi Unuttum</a>
60
61          <h2 style="background-color: #f0f0f0;" >Üye değil misiniz ?</h2>
62
63          <a href="registerPage.html" class="register" >Hemen üye olun</a>
64      </div>
65  </div>
66  </body>
67  </html>

```

Bu kod, bir internet sitesine giriş yapmak için kullandığımız o giriş sayfasının kodu. Sayfanın görünümünü (HTML) ve kullanıcı butonlarına tıkladığımızda ne olacağını (JavaScript) belirliyor. HTML bölümünde, sayfanın yapısını görüyoruz. Başlık, e-posta ve şifre giriş alanları gibi temel bileşenler var. JavaScript kısmında ise, giriş butonuna tıklandığında yapılacak işlemler tanımlanmış. E-posta ve şifre bilgileri alınıp sunucuya gönderiliyor ve cevap alındığında kullanıcı bilgilendiriliyor. Stil dosyası da var, ama o daha çok sayfanın görünümünü belirliyor. Fontlar, renkler, düzenlemeler vs. gibi şeylerin olduğu yer. Kısacası, bu kod bir internet sitesine giriş yapmamızı sağlıyor ve kullanıcı adı ve şifreyi doğrulamak için sunucuya gönderiyor. Basit ama önemli bir şey!

Backend Kodları ;

```
@PostMapping("/login")
public ResponseEntity<String> login(@RequestBody LoginDTO loginDto) {
    /*loginDtonun email ve passwordunu
    email ve password diye yeni bir atama yapıyor*/
    String email = loginDto.getEmail();
    String password = loginDto.getPassword();

    /*Girilen emaili araştırıyor kayıtlı mı değil mi diye*/
    Users user = usersRepository.findByEmail(email);

    /*Eğer girilen email ve şifre doğruysa "Giriş başarılı!" uyarısı veriyor,
    yanlışsa "Email ve Şifrenizi kontrol edin!" uyarısı veriyor*/
    if (user != null && user.getPassword().equals(password)) {
        return ResponseEntity.ok(body: "Giriş başarılı!");
    } else {
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Email ve Şifrenizi kontrol edin!");
    }
}
```

Bu Java kodu, bir kullanıcının bir web sitesine giriş yapmasını sağlıyor. Öncelikle, '@PostMapping("/login")' işlevi, '/login' yoluna gelen POST isteklerini yakalıyor. Sonra, 'login' fonksiyonu, gelen isteği işliyor. Bu fonksiyon, kullanıcının girdiği e-posta ve şifreyi alıyor ('LoginDTO' kullanarak). Daha sonra, bu e-postayı veritabanında arayarak, kayıtlı bir kullanıcı olup olmadığını kontrol ediyor. Eğer kullanıcı kayıtlıysa ve şifresi doğruysa, yani veritabanında bir eşleşme bulunmuşsa, kullanıcıya "Giriş başarılı!" mesajını döndürüyor. Ama eğer eşleşme bulunamazsa veya şifre yanlışsa, "Email ve Şifrenizi kontrol edin!" gibi bir hata mesajı döndürüyor. Yani, bu kod, kullanıcının giriş yapmasını kontrol ediyor ve ona uygun bir mesaj veriyor: ya başarılı ya da hatalı giriş.

E2 – Register Page ;

Kayıt Olun

Ad:

Soyad:

Şifre:

E-posta:

Doğum Tarihi:

Telefon No:

Profil Resimi:

Kayıt Ol

Frontend Kodları ;

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="assets/registerPageStyle.css">
7   <title>BekEmlak Register Page</title>
8   <script>
9     1+ usages
10    function addUser() {
11      const name = document.getElementById("name").value;
12      const surname = document.getElementById("surname").value;
13      const password = document.getElementById("password").value;
14      const email = document.getElementById("email").value;
15      const birthday = document.getElementById("birthday").value;
16      const phone = document.getElementById("phone").value;
17      const picture = document.getElementById("picture").value;
18
19      if (!password || !name || !surname || !email || !birthday || !phone || !picture) {
20        alert("Lütfen boş bıraktığınız yerleri doldurun!")
21        return;
22      }
23
24      const parametreler = {
25        name: name,
26        surname: surname,
27        password: password,
28        email: email,
29        birthday: birthday,
30        phone: phone,
31        picture: picture,
32      };
33
34      var request = new XMLHttpRequest();
35      request.open("POST", "http://localhost:8080/addUser", true);
36      request.setRequestHeader("Content-type", "application/json");
37      request.onreadystatechange = function () {
38        if (request.readyState === XMLHttpRequest.DONE) {
39          if (request.status === 200) {
40            alert(request.responseText);
41            window.location.href = "loginPage.html";
42          } else if (request.status === 409) {
43            alert("Bu e-posta adresi zaten kullanılıyor!");
44          }
45        }
46      };
47      request.send(JSON.stringify(parametreler));
48    }
49
50   </script>
51 </head>

```

```

52 <body>
53 <div class="topBox"></div>
54 <div class="registerBox">
55     <h2 style="text-align: center;">Kayıt Olun</h2>
56     <div>
57         <label class="name" for="name">Ad: </label>
58         <input type="text" id="name" name="name" required>
59         <br>
60         <br>
61         <label class="surname" for="surname">Soyad: </label>
62         <input type="text" id="surname" name="surname" required>
63         <br>
64         <br>
65         <label class="password" for="password">Şifre: </label>
66         <input type="password" id="password" name="password" required>
67         <br>
68         <br>
69         <label class="email" for="email">E-posta: </label>
70         <input type="email" id="email" name="email" required>
71         <br>
72         <br>
73         <label class="birthday" for="birthday">Doğum Tarihi: </label>
74         <input type="date" id="birthday" name="birthday" required>
75         <br>
76         <br>
77         <label class="phone" for="phone">Telefon No: </label>
78         <input type="text" id="phone" name="phone" required>
79         <br>
80         <br>
81         <label class="picture" for="picture">Profil Resimi: </label>
82         <input type="text" id="picture" name="picture" required placeholder="URL girin!">
83         <br>
84         <br>
85         <br>
86         <a class="registerButoon" onclick="addUser()" >Kayıt Ol</a>
87     </div>
88 </div>
89
90 </body>
91 </html>

```

Bu kod, bir web sitesinde kayıt olma işlemi için kullanılan bir sayfanın HTML ve JavaScript kısmını içeriyor. HTML kısmında, kullanıcıya gerekli bilgileri girmesi için alanlar sunuluyor. İsim, soyisim, şifre, e-posta, doğum tarihi, telefon numarası ve bir profil resmi eklemek için URL girişi yapabilecekleri kutular var. JavaScript kısmı ise, kullanıcının girdiği bilgileri alıp, bir kullanıcı oluşturmak için sunucuya gönderiyor. Eğer kullanıcı tüm bilgileri eksiksiz girmezse, bir uyarı mesajı gösteriyor ve kayıt işlemini durduruyor. Eğer tüm bilgiler doğruysa, bu bilgileri bir JSON nesnesine dönüştürüp sunucuya gönderiyor. Sunucudan gelen cevaba göre işlem sonuçlarına uygun mesajlar görüntüleniyor. Yani, bu kod, kullanıcıların web sitesine kaydolmalarını sağlayan bir araç. Kullanıcıların girdikleri bilgileri doğrulamak ve sunucuya iletmek için kullanılıyor.

Backend Kodları :

```
@PostMapping("/addUser")
public String addUser(@RequestBody @NotNull UsersDTO usersDTO){

    /*Bu kod kayıt olurken girdiğimiz emailin veritabanında başka
    bir hesapta kayıtlı olup olmadığını kontrol ediyor.*/
    if (userService.isUserExists(usersDTO.getEmail())) {
        return "Bu e-posta adresi zaten kullanılıyor!";
    }

    /*Bu kod Usersdan bir nesne oluşturuyor ve oluşturduğumuz nesnenin
    özellikleriyle dtodaki özellikleri eşliyor.*/
    Users users = new Users();

    users.setPassword(usersDTO.getPassword());
    users.setName(usersDTO.getName());
    users.setSurname(usersDTO.getSurname());
    users.setEmail(usersDTO.getEmail());
    users.setBirthday(usersDTO.getBirthday());
    users.setPhone(usersDTO.getPhone());
    users.setPicture(usersDTO.getPicture());

    /*Kayıt olurken girdiğimiz bilgileri değişkene atıyor ve veri tabanına kayıt ediyor.*/
    Users registeredUser = userService.addUser(users);

    return registeredUser.getName() + " " + registeredUser.getSurname() + " ismiyle yeni bir kullanıcı oluşturuldu!";
}
```

Bu Java kodu, kullanıcıların web sitesine kaydolmalarını sağlayan bir işlevin kodunu içeriyor.Öncelikle, `@PostMapping("/addUser")` işlevi, `/addUser` yoluna gelen POST isteklerini yakalıyor.Sonra, `addUser` fonksiyonu, gelen isteği işliyor. Bu fonksiyon, gelen kullanıcı bilgilerini (`UsersDTO`) alıyor ve önce veritabanında böyle bir e-posta adresinin daha önce kullanılıp kullanılmadığını kontrol ediyor. Eğer kullanılıyorsa, "Bu e-posta adresi zaten kullanılıyor!" şeklinde bir mesaj döndürüyor.Eğer e-posta adresi kullanılmıyorsa, yeni bir `Users` nesnesi oluşturuyor ve bu nesneye gelen bilgileri (`name`, `surname`, `password` vs.) atıyor. Daha sonra bu kullanıcıyı veritabanına kaydediyor.Son olarak, kayıt işlemi başarılı olduğunda, kullanıcının ismini ve soyismini içeren bir mesaj döndürüyor, yeni kullanıcının oluşturulduğunu bildiriyor.Yani, bu kod, kullanıcıların kayıt olmasını sağlıyor ve bunu veritabanına kaydederek başarılı bir şekilde gerçekleştirip gerçekleştirmediğini bildiriyor.

Sifrenizi mi unuttunuz?

- 1. Email adresinizi girin.
- 2. Sistemimiz email adresinize bir OTP gönderecek.
- 3. OTP yi sonraki sayfada dogrulamak için giriniz

Email adresinizi girin.

Kayıtlı email adresinizi giriniz.Adiından sisteminiz OTP gönderecektir.

Veni sifre alın.

Giris ekranina geri donun.

Mailinize gelen kodu giriniz.

Reset Password

Lütfen Yeni Şifrenizi Belirleyiniz!

Şifre:

Şifreyi oluşturun

Email hatalı veya boş
bıraktınız, lütfen tekrar
deneyiniz

[Geri dön](#)

Kodu yanlış girdiniz lütfen
tekrar deneyin!

[Geri dön](#)

Frontend Kodları / Email Girme

```
1 <!doctype html>
2 <html>
3 <head>
4   <meta charset='utf-8'>
5   <meta name='viewport' content='width=device-width, initial-scale=1'>
6   <title>Sifrenizi unuttunuz</title>
7   <link href='https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css' rel='stylesheet'>
8   <link href='' rel='stylesheet'>
9   <link rel='stylesheet' href='assets/forgotPasswordStyle.css'>
10 </head>
11 <body oncontextmenu='return false' class='snippet-body'>
12 <div class='topBox'></div>
13 <div class='container padding-bottom-3x mb-2 mt-5'>
14   <div class='row justify-content-center'>
15     <div class='forgotBox'>
16       <div class='forgot'>
17         <h2>Sifrenizi mi unuttunuz?</h2>
18         <ol class='list-unstyled'>
19           <li><span class='text-primary text-medium'>1. </span>Email adresinizi girin.</li>
20           <li><span class='text-primary text-medium'>2. </span>Sistemimiz email adresinize bir OTP gönderecek.</li>
21           <li><span class='text-primary text-medium'>3. </span>OTP yi sonraki sayfada doğrulamak için giriniz</li>
22         </ol>
23       </div>
24       <form class='card mt-4' action='forgotPassword' method='POST' onsubmit='return validateForm()'>
25         <div class='card-body'>
26           <div class='form-group'>
27             <label for='email-for-pass'>Email adresinizi girin.</label>
28             <input class='form-control' type='text' name='email' id='email-for-pass' required=''>
29             <small class='form-text text-muted'>Kayıtlı email adresinizi giriniz.Ardından sistemimiz OTP göndereceğiz.</small>
30           </div>
31         </div>
32         <div class='card-footer'>
33           <button class='btn btn-success' type='submit'>Yeni şifre alın.</button>
34           <button class='btn btn-danger' type='button' onclick='window.location.href='loginPage.html''>Giriş ekranına geri dönün.</button>
35         </div>
36       </form>
37     </div>
38   </div>
39 </div>
40 <script type='text/javascript' src='https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.bundle.min.js'></script>
41 <script type='text/javascript' src=''></script>
42 <script type='text/javascript' src=''></script>
43 <script type='text/JavaScript'>
44   1+ usages
45   function validateForm() {
46     var email = document.getElementById("email-for-pass").value;
47     var emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
48     if (!emailPattern.test(email)) {
49       window.location.href = 'ErrorPage.html';
50       return false;
51     }
52     return true;
53   }
54 </script>
55 </body>
56 </html>
```

Bu kod, kullanıcıların şifrelerini unuttuklarında kullanabilecekleri bir şifre sıfırlama işlemi için bir sayfanın HTML ve JavaScript kısmını içeriyor. HTML kısmında, kullanıcıya e-posta adresini girmesi ve yeni bir şifre alması için talimatlar veriliyor. Form aracılığıyla kullanıcı e-posta adresini girdikten sonra, "Yeni şifre alın" butonuna basarak şifre sıfırlama işlemi başlatabiliyorlar. Ayrıca, "Giriş ekranına geri dön" butonuyla giriş sayfasına geri dönebiliyorlar. JavaScript kısmı, kullanıcının girdiği e-posta adresinin geçerli olup olmadığını kontrol ediyor. Eğer geçerli değilse, kullanıcıyı bir hata sayfasına yönlendiriyor. Yani, bu kod, kullanıcıların şifrelerini unuttuklarında yeni bir şifre alabilmeleri için gerekli olan adımları içeriyor. Kullanıcıdan e-posta adresi alınarak, bu adresin geçerli olup olmadığı kontrol ediliyor ve buna göre işlem yapılıyor.

Backend Kodları / Email Girme

```
@PostMapping("/forgotPassword")
```

/*Bu kod, bir kullanıcının şifresini unuttuğunu belirttiği bir e-posta adresini alarak, bu e-posta adresine şifre sıfırlama bağlantısı içeren bir e-posta gönderen bir hizmeti çağırıyor ve bu işlem sonucunda bir ModelAndView nesnesi döndürüyor.*/

```
public ModelAndView forgotPassword(@RequestParam String email) throws MessagingException {  
    return forgotPasswordService.sendForgotPasswordEmail(email);  
}
```

```
@Service
```

```
public class ForgotPasswordService {
```

```
@Autowired
```

```
public JavaMailSender javaMailSender;
```

```
@Autowired
```

```
private HttpServletRequest request;
```

```
1 usage
```

```
public ModelAndView sendForgotPasswordEmail(String email) throws MessagingException {  
    ModelAndView modelAndView = new ModelAndView();
```

```
    if (email != null && !email.equals("")) {  
        Random rand = new Random();  
        int otpvalue = rand.nextInt( bound: 1255650);
```

```
        HttpSession session = request.getSession();  
        session.setAttribute( s: "otp", otpvalue);  
        session.setAttribute( s: "email", email);
```

```
        MimeMessage message = javaMailSender.createMimeMessage();  
        MimeMessageHelper helper = new MimeMessageHelper(message);
```

```
        helper.setFrom("erginbeko34@gmail.com");  
        helper.setTo(email);  
        helper.setSubject("Şifremi Unuttum");  
        helper.setText("Şifre sıfırlama için OTP: " + otpvalue);  
        javaMailSender.send(message);  
        System.out.println("Message sent successfully");
```

```
        modelAndView.setViewName("redirect:/EnterOtp.html");  
        modelAndView.addObject( attributeName: "message", attributeValue: "OTP e-posta adresinize gönderildi");  
    } else {  
        modelAndView.setViewName("redirect:/ErrorPage.html");  
        modelAndView.addObject( attributeName: "errorMessage", attributeValue: "E-posta gereklidir!");  
    }  
    return modelAndView;
```

```
}
```

Bu kod, kullanıcıların şifrelerini unuttuklarında şifre sıfırlama işlemini yöneten bir servis sınıfını içeriyor.Öncelikle, `sendForgotPasswordEmail` fonksiyonu, kullanıcıya bir e-posta göndererek şifre sıfırlama işlemini başlatıyor. Kullanıcı e-posta adresini girdiğinde, bu adresin geçerli olup olmadığını kontrol ediyor. Eğer geçerli ise, bir rastgele OTP (One Time Password) oluşturuyor ve bu OTP'yi kullanıcının e-posta adresine gönderiyor. Aynı zamanda, oluşturulan OTP'yi ve kullanıcının e-posta adresini bir oturum (session) değişkenine kaydediyor.Sonra, oluşturulan e-posta gönderildikten sonra, kullanıcıyı bir başka sayfaya yönlendiriyor ve bir mesaj gösteriyor. Eğer kullanıcı geçerli bir e-posta adresi girmezse, hata mesajı göstererek işlemi durduruyor.Yani, bu kod, kullanıcıların şifrelerini unuttuklarında şifre sıfırlama işlemini başlatıyor ve bu işlemi gerçekleştirmek için gerekli adımları atıyor.

Frontend Kodları / Email Kod Girme

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>BekEmlak EnterOTP</title>
  <link rel="stylesheet" href="assets/enterOtpStyle.css">
  <script>
    1+ usages
    function validateForm() {
      var otpValue = document.getElementById("otp").value;

      if (otpValue.trim() === "") {
        alert("Lütfen bir OTP kodu giriniz.");
        return false;
      }

      return true;
    }
  </script>
</head>
<body>

<div class="topBox"></div>
<div class="panel-body">

  <form id="register-form" action="/ValidateOtp" role="form" autocomplete="off" class="form" method="post" onsubmit="return validateForm()">
    <div class="form-group">
      <div class="input-group">
        <span class="input-group-addon"><i
          class="glyphicon glyphicon-envelope color-blue"></i>
        </span>
        <input id="otp" name="otp" placeholder="Mailinize gelen kodu giriniz." class="form-control" type="number" required="required">
      </div>
    </div>

    <input type="hidden" name="email" value="{email}">
    <div class="form-group">
      <input name="recover-submit"
        class="resetButton"
        value="Reset Password" type="submit">
    </div>
  </form>

</div>
</body>
</html>
```

Bu kod, kullanıcıların şifrelerini sıfırlamak için gerekli olan OTP (One Time Password) kodunu girmelerini sağlayan bir sayfanın kodunu içeriyor. HTML kısmında, kullanıcıya bir giriş formu sunuluyor. Kullanıcıdan OTP kodunu girmesi isteniyor ve bu kod, bir input alanı aracılığıyla alınıyor. Form, kullanıcının OTP kodunu girmesini sağlayacak şekilde tasarlanmış. JavaScript kısmında, kullanıcının girdiği OTP kodunun boş olup olmadığı kontrol ediliyor. Eğer kullanıcı boş bir değer girerse, bir uyarı mesajı gösterilerek işlem durduruluyor. Yani, bu kod, kullanıcıların şifrelerini sıfırlamak için gereken OTP kodunu girmelerini sağlayan bir arayüz sağlıyor. Kullanıcıdan gerekli bilgi alındıktan sonra, bu bilgi sunucuya gönderilerek şifre sıfırlama işlemi tamamlanacak.

Backend Kodları / Email Kod Girme

```
@PostMapping("/ValidateOtp")
public ModelAndView validateOtp(@RequestParam("otp") int value, HttpServletRequest request) {
    // Mevcut HTTP oturumunu alır.
    HttpSession session = request.getSession();

    // Oturumdan OTP (One-Time Password) ve e-posta adresini alır.
    Integer otpInteger = (Integer) session.getAttribute(s: "otp");
    String email = (String) session.getAttribute(s: "email");

    // Eğer OTP değeri null değilse,
    // integer değere dönüştürür, aksi halde 0 olarak ayarlar.
    int otp = otpInteger != null ? otpInteger : 0;

    // Yeni bir ModelAndView nesnesi oluşturulur.
    ModelAndView modelAndView = new ModelAndView();

    // Girilen değer OTP ile eşleşirse...
    if (value == otp) {
        // Yönlendirme yapılacak sayfanın görünümünü ayarlar ve
        // başarılı bir durumda yeni bir şifre belirleme sayfasına yönlendirir.
        modelAndView.setViewName("redirect:/newPassword.html");
        // Yeni sayfaya e-posta adresini ve başarılı bir durumun mesajını ekler.
        modelAndView.addObject(attributeName: "email", email);
        modelAndView.addObject(attributeName: "status", attributeValue: "success");
    } else {
        // OTP ile eşleşmezse, hata sayfasına yönlendirir.
        modelAndView.setViewName("redirect:/ErrorPageee.html");
        // Hata mesajını ekler.
        modelAndView.addObject(attributeName: "message", attributeValue: "Girilen kod yanlış, lütfen tekrar deneyiniz!");
    }

    return modelAndView;
}
```

Bu kod, kullanıcının girdiği OTP (One-Time Password) kodunu doğrulayan bir işlemi gerçekleştiriyor. Öncelikle, `validateOtp` fonksiyonu, kullanıcının girdiği OTP kodunu alıyor ve HTTP oturumu içindeki OTP değeriyle karşılaştırıyor. Aynı zamanda, e-posta adresini de oturum içinden alıyor. Eğer kullanıcının girdiği OTP kodu oturumdaki değerle eşleşirse, yeni bir `ModelAndView` nesnesi oluşturuluyor. Bu nesne, kullanıcıyı yeni bir şifre belirleme sayfasına yönlendiriyor ve e-posta adresini ve başarılı bir durum mesajını bu sayfaya iletiyor. Eğer OTP kodu eşleşmezse, kullanıcıyı bir hata sayfasına yönlendiriyor ve hata mesajını gösteriyor. Yani, bu kod, kullanıcının girdiği OTP kodunu doğrulayarak, başarılı veya başarısız duruma göre uygun işlemleri gerçekleştiriyor.

Frontend Kodları / Yeni Şifre Girme

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>BekEmlak New Password</title>
  <link rel="stylesheet" href="assets/newPasswordStyle.css">

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
<body>
<div class="topBox"></div>
<div class="registerBox">
  <div class="newPasswordBox">
    <h2 style="text-align: center;">Lütfen Yeni Şifrenizi Belirleyiniz!</h2>
    <label class="password" for="password">Şifre: </label>
    <input style="width: 100%;" type="password" id="password" name="password" required>
    <br>
    <br>
    <br>
    <br>
    <a class="newPasswordSave" id="createPasswordButton">Şifreyi oluşturun</a>
  </div>
</div>

<script>
  // Wait for the document to be fully loaded
  // Wait for the document to be fully loaded
  $(document).ready(function () {
    // When the button is clicked
    $('#createPasswordButton').click(function () {
      // Get the password from the input field
      var password = $('#password').val();

      // Make an AJAX POST request to your API endpoint
      $.ajax({
        type: 'POST',
        url: '/newPassword',
        data: {
          // Pass the password as a parameter
          password: password,
          // You may need to handle confirmPassword in the JavaScript as well
          confPassword: password
        },
        success: function (data) {
          alert("Şifreniz Başarıyla Değiştirildi!")
          window.location.href = "loginPage.html";
        },
        error: function () {
          // Handle error, if any
          alert('Şifre oluşturulurken bir hata oluştu-');
        }
      });
    });
  });
</script>
</body>
</html>
```

Bu kod, kullanıcının yeni bir şifre belirlemesini sağlayan bir arayüzü içeriyor. HTML kısmında, kullanıcıya yeni bir şifre belirlemesi için bir giriş kutusu sunuluyor. Kullanıcı buraya istediği yeni şifreyi giriyor. JavaScript kısmında, kullanıcının girdiği yeni şifreyi alıp, bir AJAX isteğiyle sunucuya gönderiyor. Sunucudan gelen cevaba göre işlem başarılıysa kullanıcıya bir bildirim gösterilip, giriş sayfasına yönlendiriliyor. Eğer bir hata olursa, kullanıcıya hata bildirimi gösteriliyor. Yani, bu kod, kullanıcının yeni bir şifre belirlemesini ve bu şifreyi sunucuya ileterek kaydetmesini sağlayan bir arayüz sağlıyor. Kullanıcıya işlem sonucunu bildirmek için basit bir bildirim kullanılıyor.

Backend Kodları / Yeni Şifre Girme

```
@PostMapping("/newPassword")
public ModelAndView resetPassword(HttpSession session,
    @RequestParam String password,
    @RequestParam String confPassword) {

    // Yeni bir ModelAndView nesnesi oluşturulur.
    ModelAndView modelAndView = new ModelAndView();

    // Girilen şifre ve şifre doğrulama alanları boş değilse ve şifreler eşleşiyorsa...
    if (password != null && confPassword != null && password.equals(confPassword)) {
        try (Connection connection = dataSource.getConnection()) {
            // Veritabanı bağlantısı alınır ve şifre güncelleme işlemi için bir PreparedStatement hazırlanır.
            PreparedStatement pst = connection.prepareStatement("UPDATE Users SET password = ? WHERE email = ?");
            // PreparedStatement parametreleri ayarlanır.
            pst.setString(1, password);
            pst.setString(2, (String) session.getAttribute("email"));

            // UPDATE sorgusu çalıştırılır ve etkilenen satır sayısı alınır.
            int rowCount = pst.executeUpdate();
            // Eğer en az bir satır etkilenmişse...
            if (rowCount > 0) {
                // Başarılı bir şekilde şifre güncellendiği için giriş sayfasına yönlendirme yapılır.
                modelAndView.setViewName("redirect:/LoginPage.html");
            } else {
                // Etkilenen satır yoksa, hata sayfasına yönlendirme yapılır.
                modelAndView.setViewName("redirect:/ErrorPage.html");
            }
        } catch (SQLException e) {
            // Veritabanı hatası durumunda, hata sayfasına yönlendirme yapılır ve hata ekrana yazdırılır.
            e.printStackTrace();
            modelAndView.setViewName("redirect:/ErrorPage.html");
        }
    } else {
        // Şifreler eşleşmiyorsa veya herhangi bir alan boşsa, hata sayfasına yönlendirme yapılır.
        modelAndView.setViewName("redirect:/ErrorPage.html");
    }

    return modelAndView;
}
```

Bu kod, kullanıcının yeni bir şifre belirlemesini ve bu şifreyi veritabanında güncellemesini sağlayan bir işlemi gerçekleştiriyor. Öncelikle, 'resetPassword' fonksiyonu, kullanıcının girdiği yeni şifreyi ve şifre doğrulama bilgisini alıyor. Eğer bu bilgiler boş değilse ve şifreler eşleşiyorsa, veritabanına bağlanarak kullanıcının şifresini güncelliyor. Veritabanında şifre güncelleme işlemi başarılıysa, kullanıcıyı giriş sayfasına yönlendiriyor. Eğer herhangi bir hata oluşursa, kullanıcıyı bir hata sayfasına yönlendiriyor ve hata bilgisini ekrana yazdırıyor. Yani, bu kod, kullanıcının yeni bir şifre belirlemesini ve bu şifreyi veritabanında güncellemesini sağlayan bir işlemi gerçekleştiriyor. Herhangi bir hata durumunda kullanıcıya uygun bir hata mesajı gösteriliyor ve ilgili işlem gerçekleştirilmiyor.

Frontend Kodları / Email Hatası

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Error Page</title>
  <link rel="stylesheet" href="assets/errorPageStyle.css">
</head>
<body>
  <div class="topBox"></div>
  <div class="errorBox">
    <h1 style="text-align: center;">Email hatalı veya boş bıraktınız, lütfen tekrar deneyiniz</h1>
    <button class="backButton" type="submit" onclick="redirectToForgotPasswordPage()">Geri dön</button>
  </div>
<script>
  1+ usages
  function redirectToForgotPasswordPage() {
    window.location.href = 'forgotPassword.html';
  }
</script>
</body>
</html>
```

Bu kod, kullanıcının karşılaştığı bir hata durumunda gösterilecek bir hata sayfasını oluşturuyor. HTML kısmında, kullanıcıya hata mesajını gösterecek bir div içeriği ve geri dönme butonu bulunuyor. Hata mesajı, kullanıcıya neyin yanlış gittiğini anlatıyor ve geri dönme butonu, kullanıcıyı bir önceki sayfaya yönlendiriyor. JavaScript kısmında, geri dönme butonuna tıklandığında `redirectToForgotPasswordPage` fonksiyonu çağırılıyor. Bu fonksiyon, kullanıcıyı şifre sıfırlama sayfasına yönlendiriyor. Yani, bu kod, kullanıcıya hata durumunda bilgilendirici bir mesaj sunan ve kullanıcının bir önceki sayfaya geri dönmesini sağlayan bir hata sayfası oluşturuyor.

Frontend Kodları / Kod Hatası

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Error Page</title>
  <link rel="stylesheet" href="assets/ErrorPageStyle.css">
</head>
<body>
  <div class="topBox"></div>
  <div class="errorBox">
    <h1 style="text-align: center;">Kodu yanlış girdiniz lütfen tekrar deneyin!</h1>
    <button class="backButton" type="submit" onclick="redirectToEnterOtpPage()">Geri dön</button>
  </div>
  <script>
    1+ usages
    function redirectToEnterOtpPage() {
      window.location.href = 'EnterOtp.html';
    }
  </script>
</body>
</html>
```

Bu kod, kullanıcının OTP (One-Time Password) kodunu yanlış girdiği durumda gösterilecek bir hata sayfasını oluşturuyor. HTML kısmında, kullanıcıya hata mesajını gösterecek bir div içeriği ve geri dönme butonu bulunuyor. Hata mesajı, kullanıcıya kodun yanlış girildiğini ve tekrar denemesi gerektiğini belirtiyor. Geri dönme butonu ise, kullanıcıyı OTP kodunu girmesi gereken sayfaya yönlendiriyor. JavaScript kısmında, geri dönme butonuna tıklandığında `redirectToEnterOtpPage` fonksiyonu çağrılıyor. Bu fonksiyon, kullanıcıyı OTP kodunu girmesi gereken sayfaya yönlendiriyor. Yani, bu kod, kullanıcıya OTP kodunu yanlış girdiği durumda bilgilendirici bir mesaj sunan ve tekrar giriş yapması için gerekli olan sayfaya yönlendiren bir hata sayfası oluşturuyor.

[E4 -index.html \(AnaSayfa\)](#)

BEKMLAK

Giriş Yap Kayıt Ol

İlanlar



İsim:BAŞAKŞEHİR DÜZ GİRİŞ
880 METRE İŞYERİ 55 metre YOL
CEPHESİ
Fiyat: 175000
Konum: İstanbul/Başakşehir
Açıklama: BAŞAKŞEHİR DÜZ
GİRİŞ 880 METRE İŞYERİ 55
metre YOL CEPHESİ 20 araçlık park
yeri
Telefon: 0530 746 17 21

[E5 - anaSayfaLoginSonrasi](#)

BEKMLAK

Hosgeldin, bektasergin !

Çıkış

İlanlar



İlan Ekleİlan Silİlan Güncelle



İsim:BAŞAKŞEHİR DÜZ GİRİŞ
880 METRE İŞYERİ 55 metre YOL
CEPHESİ
Fiyat: 175000
Konum: İstanbul/Başakşehir
Açıklama: BAŞAKŞEHİR DÜZ
GİRİŞ 880 METRE İŞYERİ 55
metre YOL CEPHESİ 20 araçlık park
yeri
Telefon: 0530 746 17 21

Frontend Kodları/Anasayfa

```
1 <!DOCTYPE html>
2 <html lang="tr">
3 <head>
4   <meta charset="UTF-8">
5   <title>BekEmlak - Anasayfa</title>
6   <link rel="stylesheet" href="assets/anaSayfaStyle.css">
7 </head>
8 <body>
9   <div class="topBox">
10     
11
12     <div class="buttons">
13       <a class="anaLoginButton" href="loginPage.html">Giriş Yap</a>
14       <a style="background-color: #007bff; color: white;" class="anaRegisterButton" href="registerPage.html">Kayıt Ol</a>
15     </div>
16   </div>
17
18   <h1 class="homes">İlanlar</h1>
19
20   <div class="homeBox" id="ilanlarContainer">
21
22   </div>
23
24   <script>
25     // usages
26     function ilanGetir() {
27       var request = new XMLHttpRequest();
28       request.open("GET", "http://localhost:8080/ilan", true);
29       request.onreadystatechange = function () {
30         if (request.readyState === XMLHttpRequest.DONE && request.status === 200) {
31           var ilanlar = JSON.parse(request.responseText);
32           var ilanlarHTML = "";
33
34           ilanlar.forEach(function (ilan) {
35             ilanlarHTML += "<div class='bilgiKutu'>";
36             ilanlarHTML += "<img class='ilanFoto' src='data:" + ilan.image + "' alt='İlan Resmi'>";
37             ilanlarHTML += "<div class='ilan'>";
38             ilanlarHTML += "<p style='text-align: center;'><b>İsim:</b> " + ilan.name + "</p>";
39             ilanlarHTML += "<p><b>Fiyat:</b> " + ilan.price + "</p>";
40             ilanlarHTML += "<p><b>Konum:</b> " + ilan.location + "</p>";
41             ilanlarHTML += "<p><b>Açıklama:</b> " + ilan.explanation + "</p>";
42             ilanlarHTML += "<p style='text-align: center;'><b>Telefon:</b> " + ilan.phone + "</p>";
43             ilanlarHTML += "</div>";
44             ilanlarHTML += "</div>";
45           });
46
47           document.getElementById("ilanlarContainer").innerHTML = ilanlarHTML;
48         }
49       };
50       request.send();
51     }
52
53     window.onload = ilanGetir();
54   </script>
55 </body>
56 </html>
```


Bu kod, bir web sayfasının yapı taşlarını oluşturuyor. HTML bölümü, sayfanın içeriğini tanımlar; CSS, bu içeriğe stil ve düzen sağlar; JavaScript ise sayfanın dinamik davranışlarını yönetir. HTML kısmında, sayfanın başlığını, logo ve giriş/kayıt butonlarını içeren bir üst bölüm ve ilanların listeleneceği bir alan bulunuyor. CSS ile, bu elemanlara görünüm ve düzen kazandırılıyor; örneğin, butonların rengi veya ilanların yerleşimi gibi. JavaScript ise sayfanın en dinamik kısmını oluşturuyor. `ilanGetir()` fonksiyonu, sunucudan ilan verilerini almak için bir istek yapar ve gelen verileri işler. Bu veriler, sayfa içeriği olarak düzenlenir ve kullanıcıya sunulur. Sonuç olarak, bu kodlar bir araya gelerek kullanıcıya ilanları gösteren bir web sayfası oluşturur. Kullanıcı sayfayı açtığında, JavaScript tarafından yapılan bir istekle ilanlar sunucudan çekilir ve HTML içeriğine dönüştürülerek kullanıcıya gösterilir.

Frontend Kodları/anasayfaLoginSonrasi

```
1 <!DOCTYPE html>
2 <html lang="en" xmlns:th="http://www.w3.org/1999/xhtml">
3 <head>
4   <meta charset="UTF-8">
5   <title>BekEmlak - HomePage</title>
6   <link rel="stylesheet" href="assets/anaSayfaStyle.css">
7 </head>
8 <body>
9   <div class="topBox">
10     
11     <div id="userName" class="userName"></div>
12     <a class="anaLogoutButton" href="index.html">Çıkış</a>
13   </div>
14
15   <div class="konteynir">
16     <h1 class="homes">İlanlar</h1>
17     
18     <a class="islem" href="ilanEkle.html">İlan Ekle</a>
19     <a class="islem" href="ilanSilme.html">İlan Sil</a>
20     <a href="ilanGuncelle.html" class="islem">İlan Güncelle</a>
21   </div>
22
23   <div class="homeBox" id="ilanlarContainer"></div>
24
25 </body>
26 </html>
27
28 <script>
29   1+ usages
30   function ilanGetir() {
31     var request = new XMLHttpRequest();
32     request.open("GET", "http://localhost:8080/ilan", true);
33     request.onreadystatechange = function () {
34       if (request.readyState === XMLHttpRequest.DONE && request.status === 200) {
35         var ilanlar = JSON.parse(request.responseText);
36         var ilanlarHTML = "";
37
38         ilanlar.forEach(function (ilan) {
39           ilanlarHTML += "<div class='bilgiKutu'>";
40           ilanlarHTML += "<img class='ilanFoto' src='data:image/jpeg;base64,'" + ilan.image + "' alt='İlan Resmi'>";
41           ilanlarHTML += "<div class='ilan'>";
42           ilanlarHTML += "<p style='margin-top: 50px'><b>İsim:</b>" + ilan.name + "</p>";
43           ilanlarHTML += "<p><b>Fiyat: </b>" + ilan.price + "</p>";
44           ilanlarHTML += "<p><b>Konum: </b>" + ilan.location + "</p>";
45           ilanlarHTML += "<p><b>Açıklama: </b>" + ilan.explanation + "</p>";
46           ilanlarHTML += "<p style='margin-bottom: 50px'><b>Telefon: </b>" + ilan.phone + "</p>";
47           ilanlarHTML += "</div>";
48         });
49
50         document.getElementById("ilanlarContainer").innerHTML = ilanlarHTML;
51       }
52     };
53     request.send();
54   }
55
56   // Sayfa yüklendiğinde ilanları otomatik olarak getir
57   window.onload = ilanGetir();
58
59   // Sayfa yüklendiğinde ilanları otomatik olarak getir
60   window.onload = function () {
61     ilanGetir();
62     var userName = sessionStorage.getItem('userName');
63     document.getElementById('userName').innerText = "Hoşgeldin, " + userName + " !";
64   };
65 </script>
66 </body>
67 </html>
```

Bu kod, bir web sayfasının temel unsurlarını içerir: HTML, CSS ve JavaScript. HTML kısmı, sayfanın yapısal iskeletini oluşturur. Başlık, logo ve giriş/kayıt butonları gibi statik içeriklerle birlikte, dinamik içerik için bir alan da sağlar. CSS, HTML içeriğine stil ve düzen kazandırır. Bu sayede, sayfa elemanları arasındaki ilişki ve düzen ayarlanır. Örneğin, butonların görünümü veya ilanların yerleşimi gibi özellikler burada belirlenir. JavaScript, sayfanın dinamik davranışlarını yönetir. Özellikle, `ilanGetir()` fonksiyonu sayesinde sunucudan ilan verileri çekilir, bu veriler HTML içeriğine dönüştürülerek kullanıcıya gösterilir. Bu sayfadan ilanı silme, güncelleme ve ekleme sayfalarına da yönlendirme yapılabilir. Böylece, kullanıcı sayfayı açtığında ilanlar otomatik olarak yüklenir ve görüntülenir. Bu üç bileşen bir araya gelerek, kullanıcıya interaktif bir web deneyimi sunar. Sayfa, hem statik hem de dinamik içerikle zenginleştirilir, böylece kullanıcılar ilanları kolayca görüntüleyebilir ve etkileşime girebilir.

E6 – ilanEkle



İLAN EKLEYİN :)

İlan ismi:

İlanın Ücreti:

İlan Konumu:

İlan açıklaması:

İlan fotoğrafları:

DOSYA SEÇ

Dosya seçmediniz

İlan telefonu:

[İlan Ekle](#)

Frontend Kodları / ilanEkle

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Bekemlak - İlan Ekleme</title>
6      <link rel="stylesheet" href="assets/ilanEkleStyle.css">
7  </head>
8  <body>
9      <div class="topBox">
10         
11     </div>
12
13     <div class="ilanBox">
14         <h1 STYLE="background-color: #f0f0f0;"><b>İLAN EKLEYİN :)</b></h1>
15     </div>
16     <div class="inputs">
17         <label class="name" for="name">İlan ismi: </label>
18         <input type="text" id="name" required>
19         <br>
20         <br>
21         <label class="price" for="price">İlanın Ücreti: </label>
22         <input type="number" id="price" required>
23         <br>
24         <br>
25         <label class="location" for="location">İlan Konumu: </label>
26         <input type="text" id="location" required>
27         <br>
28         <br>
29         <label class="explanation" for="explanation">İlan açıklaması: </label>
30         <input type="text" id="explanation" required>
31         <br>
32         <br>
33         <label class="image" for="image">İlan fotoğrafları: </label>
34         <input type="file" id="image" accept="image/*" required>
35         <br>
36         <br>
37         <label class="phone" for="phone">İlan telefonu: </label>
38         <input type="text" id="phone" required>
39     </div>
40     <br>
41     <br>
42     <a class="announcementRegister" onclick="addAnnouncement()">İlan Ekle</a>
43 </div>
```

```

46 function addAnnouncement() {
47     const name = document.getElementById("name").value;
48     const price = document.getElementById("price").value;
49     const location = document.getElementById("location").value;
50     const explanation = document.getElementById("explanation").value;
51     const image = document.getElementById("image").files[0]; // Get the selected file
52     const phone = document.getElementById("phone").value;
53
54     if (!name || !price || !location || !explanation || !image || !phone) {
55         alert("Lütfen boş bıraktığınız yerleri doldurun!")
56         return;
57     }
58
59     const formData = new FormData();
60     formData.append("name", name);
61     formData.append("price", price);
62     formData.append("location", location);
63     formData.append("explanation", explanation);
64     formData.append("image", image); // Append the file to FormData
65     formData.append("phone", phone);
66
67     var request = new XMLHttpRequest();
68     request.open("POST", "http://localhost:8080/addAnnouncement", true);
69     request.onreadystatechange = function () {
70         if (request.readyState === XMLHttpRequest.DONE) {
71             if (request.status === 200) {
72                 alert(request.responseText);
73                 window.location.href = "anaSayfaLoginSonrasi.html";
74             } else if (request.status === 409) {
75                 alert("Bilgilerinizi kontrol ediniz!");
76             }
77         }
78     };
79     request.send(formData); // Send FormData instead of JSON
80 }
81 </script>
82
83 </body>
84 </html>
85

```

HTML kısmı, sayfanın yapısal bileşenlerini içerir. Bir logo ve bir ilan ekleme formu bulunur. Form, kullanıcının ilan bilgilerini girmesi için bir dizi metin girişi ve bir resim yükleme alanı içerir. Ayrıca, kullanıcıdan alınacak bilgilerin yanında uygun etiketler (label) bulunur. CSS, bu yapısal bileşenlere stil ve düzen kazandırır. Örneğin, form elemanlarının boyutu, arka plan rengi ve metin stilini belirler. Bu CSS dosyası, HTML dosyasıyla bağlantılıdır ve sayfaya eklenmiştir. JavaScript, sayfanın dinamik davranışlarını yönetir. `addAnnouncement()` fonksiyonu, kullanıcının ilan bilgilerini formdan alır ve bir HTTP POST isteği yaparak sunucuya gönderir. İsteğin gövdesi FormData nesnesiyle doldurulur, bu da hem metin bilgilerini hem de seçilen resmi içerir. Sunucudan gelen yanıtı göre, kullanıcıya uygun bir mesaj gösterilir ve gerekirse sayfa yenilenir veya başka bir sayfaya yönlendirilir. Özetle, bu kod bir web sayfası oluşturarak kullanıcılara ilan eklemeyi sağlar. Kullanıcılar formu doldurduktan sonra ilan bilgileri sunucuya gönderilir ve işlenir. Bu sayede, web sitesine yeni ilanlar eklenmiş olur.

```
48 @PostMapping("/addAnnouncement")
49 public String addAnnouncement(@RequestParam("image") MultipartFile image,
50                               @RequestParam("name") String name,
51                               @RequestParam("price") float price,
52                               @RequestParam("location") String location,
53                               @RequestParam("explanation") String explanation,
54                               @RequestParam("phone") String phone) {
55     Announcements announcements = new Announcements();
56
57     announcements.setName(name);
58     announcements.setPrice(price);
59     announcements.setLocation(location);
60     announcements.setExplanation(explanation);
61     announcements.setPhone(phone);
62
63     // Handle image upload
64     try {
65         announcements.setImage(image.getBytes());
66     } catch (IOException e) {
67         e.printStackTrace();
68         // Handle error
69     }
70
71     Announcements registeredAnnouncement = announcementsService.addAnnouncements(announcements);
72
73     return registeredAnnouncement.getName() + " Adında yeni bir ilan oluşturuldu!";
74 }
```

Bu kod bir Spring Boot uygulamasında bir controller sınıfında bulunur ve kullanıcıların ilan eklemesini işler. `@PostMapping("/addAnnouncement")` anotasyonu, HTTP POST isteklerini `/addAnnouncement` URL yoluna yönlendirir. Bu URL'e gelen istekler, bu metodu çalıştırır. Metodun parametreleri, HTTP isteğinden gelen verileri alır. `MultipartFile` türünde olan `image`, kullanıcının yüklediği ilan resmini temsil eder. Diğer parametreler, kullanıcı tarafından doldurulan ilan bilgilerini alır: isim, fiyat, konum, açıklama ve telefon numarası. Bu metod, ilan verilerini bir `Announcements` nesnesine atar. Daha sonra, resmi işlemek için `getBytes()` metodu kullanılarak ilan resmi `Announcements` nesnesine eklenir. `announcementsService.addAnnouncements(announcements)` çağrısı, ilanı veritabanına eklemek için bir servis sınıfına ilan nesnesini iletir. Son olarak, eklenen ilanın ismiyle bir mesaj döndürülür, bu mesajda yeni ilanın başarıyla oluşturulduğu belirtilir. Özetle, bu kod, kullanıcıların ilan eklemesi için bir HTTP POST isteğini işler ve ilan verilerini alarak bir ilan nesnesi oluşturur. Bu nesne daha sonra veritabanına eklenir ve bir mesajla kullanıcıya geri döndürülür.

E7 - ilanSilme



Sileceğiniz ilanın adını yazınız :)

Frontend Kodları / ilanSilme

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Bekemlak - İlan Silme Sayfası</title>
6   <link rel="stylesheet" href="assets/ilanSilmeStyle.css">
7 </head>
8 <body>
9   <div class="topBox">
10    
11  </div>
12  <div class="silmeBox">
13    <h1 style="text-align: center;font-family: 'Comic Sans MS';font-size: 50px">Sileceğiniz ilanın adını yazınız :)</h1>
14    <input type="text" id="name" required placeholder="İlanın ismini giriniz!">
15    <a class="silmeButton" onclick="ilanSilme()">İlanı Sil</a>
16  </div>
17
18  <script>
19    1+ usages
20    function ilanSilme() {
21      var ilanAdi = document.getElementById("name").value;
22
23      if (!ilanAdi){
24        alert("İsim giriniz!")
25      }
26
27      // İstek yapılacak endpoint'in URL'si
28      var endpointURL = "/delete/" + encodeURIComponent(ilanAdi); // İlan adı URL'ye uygun hale getiriliyor
29
30      // HTTP DELETE isteği gönderme
31      fetch(endpointURL, {
32        method: 'DELETE',
33        headers: {
34          'Content-Type': 'application/json'
35        }
36      })
37      .then(response => {
38        if (response.ok) {
39          // Başarılı yeni durumda kullanıcıya bilgi verme
40          alert("İlan başarıyla silindi.");
41          window.location.href = "anaSayfaLoginSonrasi.html";
42        } else {
43          // Yanıt hatalı ise hata mesajını gösterme
44          alert("İlan silinemedi. Hata: " + response.statusText);
45        }
46      })
47      .catch(error => {
48        // İstek gönderilirken ortaya çıkan hataları yakalamak
49        alert("İstek gönderilirken bir hata oluştu: " + error.message);
50      });
51    }
52  </script>
53 </body>
54 </html>
```

Bu kod bir web sayfası oluşturuyor ve kullanıcılara ilan silme işlemi yapma imkanı sunuyor. Şimdi, bu kodu açıklayayım HTML kısmı, sayfanın yapısal bileşenlerini içerir. Bir logo ve bir ilan silme formu bulunur. Form, kullanıcının silecekleri ilanın adını girmesi için bir metin girişi içerir.CSS dosyası, bu yapısal bileşenlere stil ve düzen kazandırır. Örneğin, metin girişinin boyutu, arka plan rengi ve metin stilini belirler. Bu CSS dosyası, HTML dosyasıyla bağlantılıdır ve sayfaya eklenmiştir.JavaScript kısmı, sayfanın dinamik davranışlarını yönetir. `ilanSilme()` fonksiyonu, kullanıcının girdiği ilan adını alır ve bir HTTP DELETE isteği yaparak sunucuya gönderir. Bu isteğin sonucuna göre kullanıcıya bilgi verilir: ilan başarıyla silindiğinde başarılı bir mesaj gösterilir ve sayfa yenilenir, aksi takdirde bir hata mesajı gösterilir.Özetle, bu kod, kullanıcıların belirledikleri bir ilan silmelerine olanak sağlayan bir web sayfası oluşturur. Kullanıcı ilan adını girdikten sonra ilgili butona tıkladığında, ilgili ilan sunucudan silinir ve kullanıcıya uygun bir geri bildirim gösterilir.

Backend Kodları / ilanSilme

```
@DeleteMapping("/{delete}/{name}")
public ResponseEntity<String> deleteAnnouncementByName(@PathVariable String name) {
    try {
        announcementsService.deleteAnnouncementByName(name);
        return new ResponseEntity<>( body: "Announcement with name '" + name + "' deleted successfully.", HttpStatus.OK);
    } catch (Exception e) {
        return new ResponseEntity<>( body: "Failed to delete announcement with name '" + name + "'.", HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

```
@Override
public void deleteAnnouncementByName(String name) {
    Announcements announcement = announcementsRepo.findByName(name);
    announcementsRepo.delete(announcement);
}
```

Bu kod bir Spring Boot uygulamasında bir controller sınıfında bulunur ve kullanıcının belirli bir ilanı silmesini işler. `@DeleteMapping("/{delete}/{name}")` anotasyonu, HTTP DELETE isteklerini `/{delete}/{name}` URL yoluna yönlendirir. `{name}` yeri, silcekleri ilanın adını temsil eder. Bu URL'e gelen DELETE istekleri, bu metodu çalıştırır. Metodun parametresi, `@PathVariable` anotasyonu ile belirtilen `name` değişkenidir. Bu, HTTP isteğinden gelen ilan adını temsil eder. Metod, `announcementsService.deleteAnnouncementByName(name)` çağrısı yaparak ilanı silen bir servis sınıfını kullanır. İlan başarıyla silinirse, `ResponseEntity` kullanılarak HTTP 200 (Başarılı) durum kodu ve bir mesaj döndürülür. Aksi takdirde, bir istisna oluşursa, HTTP 500 (Sunucu İç Hatası) durum kodu ve bir hata mesajı döndürülür. Özetle, bu kod, belirli bir ilanı silme işlemini işler. Kullanıcının belirttiği ilan adıyla ilgili bir ilanı veritabanından siler ve buna uygun bir HTTP yanıtı döndürür.

E8 - ilanGuncelle

BEKEMLAK

Lütfen güncellemek istediğiniz ilanın önce
ismini sonra da yeni bilgilerini giriniz!

İlanın İsmi:
İlanın Yeni İsmi:
İlanın Yeni Ücreti:
İlanın Yeni Konumu:
İlanın Yeni Açıklaması:
İlanın Yeni Resimleri:
İlanın Yeni Numarası:

İlanı Güncelle

Frontend Kodları / ilanGuncelle

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>BekeEnlak - İlan Güncelleme Sayfası</title>
  <link rel="stylesheet" href="assets/ilanGuncelleStyle.css">
</head>
<body>
<div class="topBox">
  
</div>

<div class="guncellemeBox">
  <h1 style="margin-bottom: 50px;text-align:
center;font-family: 'Comic Sans MS'
font-size: 40px">Lütfen güncellemek istediğiniz ilanın önce <br> ismini sonra da yeni bilgilerini giriniz!</h1>
  <form id="updateForm">
    <div class="inputs">
      <div class="input-group">
        <label class="oldName" for="oldName">İlanın İsmi:</label>
        <input type="text" id="oldName" name="oldName">
      </div>
      <br>
      <div class="input-group">
        <label class="newName" for="newName">İlanın Yeni İsmi:</label>
        <input type="text" id="newName" name="newName">
      </div>
      <br>
      <div class="input-group">
        <label class="price" for="price">İlanın Yeni Ücreti:</label>
        <input type="number" id="price" name="price">
      </div>
      <br>
      <div class="input-group">
        <label class="location" for="location">İlanın Yeni Konumu:</label>
        <input type="text" id="location" name="location">
      </div>
      <br>
      <div class="input-group">
        <label class="explanation" for="explanation">İlanın Yeni Açıklaması:</label>
        <input type="text" id="explanation" name="explanation">
      </div>
      <br>
      <div class="input-group">
        <label class="image" for="image">İlanın Yeni Resimleri:</label>
        <input type="file" id="image" name="image">
      </div>
      <br>
      <div class="input-group">
        <label class="phone" for="phone">İlanın Yeni Numarası:</label>
        <input type="text" id="phone" name="phone">
      </div>
      <br>
      <button class="updatebutton" onclick="update()" type="submit">İlanı Güncelle</button>
    </form>
  </div>
```

```
<script>
  // usages
  function update() {
    document.getElementById("updateForm").addEventListener("submit", function(event) {
      event.preventDefault();
      const oldName = document.getElementById("oldName").value;
      const newName = document.getElementById("newName").value;
      const price = document.getElementById("price").value;
      const location = document.getElementById("location").value;
      const explanation = document.getElementById("explanation").value;
      const image = document.getElementById("image").files[0];
      const phone = document.getElementById("phone").value;

      if (!oldName || !newName || !price || !location || !explanation || !image || !phone) {
        alert("Lütfen boş bıraktığınız yerleri doldurunuz!")
        return;
      }

      // FormData kullanarak verileri gönder
      const formData = new FormData();
      formData.append("updatedName", newName);
      formData.append("updatedPrice", price);
      formData.append("updatedLocation", location);
      formData.append("updatedExplanation", explanation);
      formData.append("updatedImage", image);
      formData.append("updatedPhone", phone);

      fetch(`/update/${oldName}`, {
        method: 'PUT',
        body: formData
      })
        .then(response => {
          if (response.ok) {
            return response.json();
          }
          throw new Error('Network response was not ok. ');
        })
        .then(data => {
          alert("İlanınız başarıyla güncellendi!")
          window.location.href = "anaSayfaLoginSonrasi.html"
        })
        .catch(error => {
          alert("İlanınız güncellenirken bir hatayla karşılaşıldı. Lütfen tekrar deneyiniz!")
          console.error('Error updating announcement:', error);
        });
    });
  }
</script>

</body>
</html>
```

Bu kod bir web sayfası oluşturuyor ve kullanıcılara bir ilanı güncelleme imkanı sunuyor. Şimdi, bu kodu adım adım açıklayayım. HTML kısmı, sayfanın yapısal bileşenlerini içerir. Bir logo ve bir ilan güncelleme formu bulunur. Form, kullanıcının güncellemek istediği ilanın mevcut adını ve yeni bilgilerini girmesi için bir dizi metin girişi ve bir resim yükleme alanı içerir. CSS dosyası, bu yapısal bileşenlere stil ve düzen kazandırır. Örneğin, form elemanlarının boyutu, arka plan rengi ve metin stilini belirler. Bu CSS dosyası, HTML dosyasıyla bağlantılıdır ve sayfaya eklenmiştir. JavaScript kısmı, sayfanın dinamik davranışlarını yönetir. `update()` fonksiyonu, formun gönderilmesini engelleyerek, JavaScript ile formun dinamik olarak işlenmesini sağlar. Kullanıcı formu doldurduktan sonra, formun submit olayını dinleyerek, ilanın güncellenmesi için bir HTTP PUT isteği yapar. Bu istek, ilgili ilanın adını ve güncellenmiş bilgilerini içeren bir FormData nesnesi ile gönderilir. Sunucudan gelen yanıtla göre, kullanıcıya uygun bir geri bildirim gösterilir: ilan başarıyla güncellendiğinde başarılı bir mesaj gösterilir ve sayfa yenilenir, aksi takdirde bir hata mesajı gösterilir. Özetle, bu kod, kullanıcıların belirli bir ilanı güncellemesine olanak sağlayan bir web sayfası oluşturur. Kullanıcı ilanın mevcut adını ve yeni bilgilerini girdikten sonra ilgili butona tıkladığında, ilgili ilanın sunucuda güncellenmesi işlemi gerçekleşir ve kullanıcıya uygun bir geri bildirim gösterilir.

Backend Kodları / ilanGuncelle

```
@PostMapping("/{name}/{name}")
public ResponseEntity<Announcements> updateAnnouncementByName(@PathVariable String name,
                                                                @RequestParam("updatedName") String updatedName,
                                                                @RequestParam("updatedPrice") float updatedPrice,
                                                                @RequestParam("updatedLocation") String updatedLocation,
                                                                @RequestParam("updatedExplanation") String updatedExplanation,
                                                                @RequestParam("updatedImage") MultipartFile updatedImage,
                                                                @RequestParam("updatedPhone") String updatedPhone) {
    Announcements announcement = announcementsService.updateAnnouncementByName(name, updatedName, updatedPrice, updatedLocation, updatedExplanation, updatedImage, updatedPhone);
    if (announcement != null) {
        return ResponseEntity.ok(announcement);
    } else {
        return ResponseEntity.notFound().build();
    }
}
```

```
@Override
public Announcements updateAnnouncementByName(@RequestParam("name") String name,
                                                @RequestParam("updatedName") String updatedName,
                                                @RequestParam("price") float updatedPrice,
                                                @RequestParam("location") String updatedLocation,
                                                @RequestParam("explanation") String updatedExplanation,
                                                @RequestParam("image") MultipartFile updatedImage,
                                                @RequestParam("phone") String updatedPhone) {
    Announcements announcement = announcementsRepo.findById(name);
    if (announcement != null) {
        // Güncelleme yapılacaksa DTO'dan gelen verilerle güncelle
        announcement.setName(updatedName);
        announcement.setPrice(updatedPrice);
        announcement.setLocation(updatedLocation);
        announcement.setExplanation(updatedExplanation);

        // Handle image update
        try {
            announcement.setImage(updatedImage.getBytes());
        } catch (IOException e) {
            e.printStackTrace();
            // Handle error
        }

        announcement.setPhone(updatedPhone);
        // Güncellenmiş ilanı kaydet
        return announcementsRepo.save(announcement);
    }
    return null;
}
```

Bu iki kod bir Spring Boot uygulamasında bir servis sınıfı ve bir controller sınıfında bulunur. Bir ilanın güncellenmesini işleyen HTTP PUT isteği için bir endpoint ve ilan güncelleme işlemini gerçekleştiren bir servis metodunu içerir. İlk olarak, `updateAnnouncementByName` adlı metod, ilanın güncellenmesini gerçekleştirir. Bu metod, ilanın mevcut adını temel alır ve yeni bilgilerle günceller. İlgili ilanın veritabanından alınması, yeni bilgilerin atanması ve gerekirse resmin güncellenmesi işlemleri yapılır. Son olarak, güncellenmiş ilan veritabanına kaydedilir. İkinci olarak, `updateAnnouncementByName` adlı metod, HTTP PUT isteklerini `/update/{name}` URL yoluna yönlendirir. Bu metod, `@PathVariable` ile belirtilen `{name}` parametresi aracılığıyla güncellenecek ilanın adını alır. Bu metod, gelen güncellenmiş bilgileri içeren bir DTO (Data Transfer Object) kullanır ve ilgili servis metodu aracılığıyla ilanın güncellenmesini gerçekleştirir. Güncelleme işlemi başarılı ise güncellenmiş ilan veritabanından alınarak HTTP 200 (Başarılı) durum kodu ile yanıt verilir. Eğer güncellenen ilan bulunamazsa, HTTP 404 (Bulunamadı) durum kodu ile yanıt verilir. Özetle, bu iki kod, bir ilanın güncellenmesi için HTTP PUT isteklerini işler. Kullanıcı tarafından gönderilen güncellenmiş ilan bilgileri, ilgili metodlar aracılığıyla alınır ve veritabanında güncellenir. Bu sayede, kullanıcıların ilanlarını güncelleyebilmesine olanak sağlanır.

