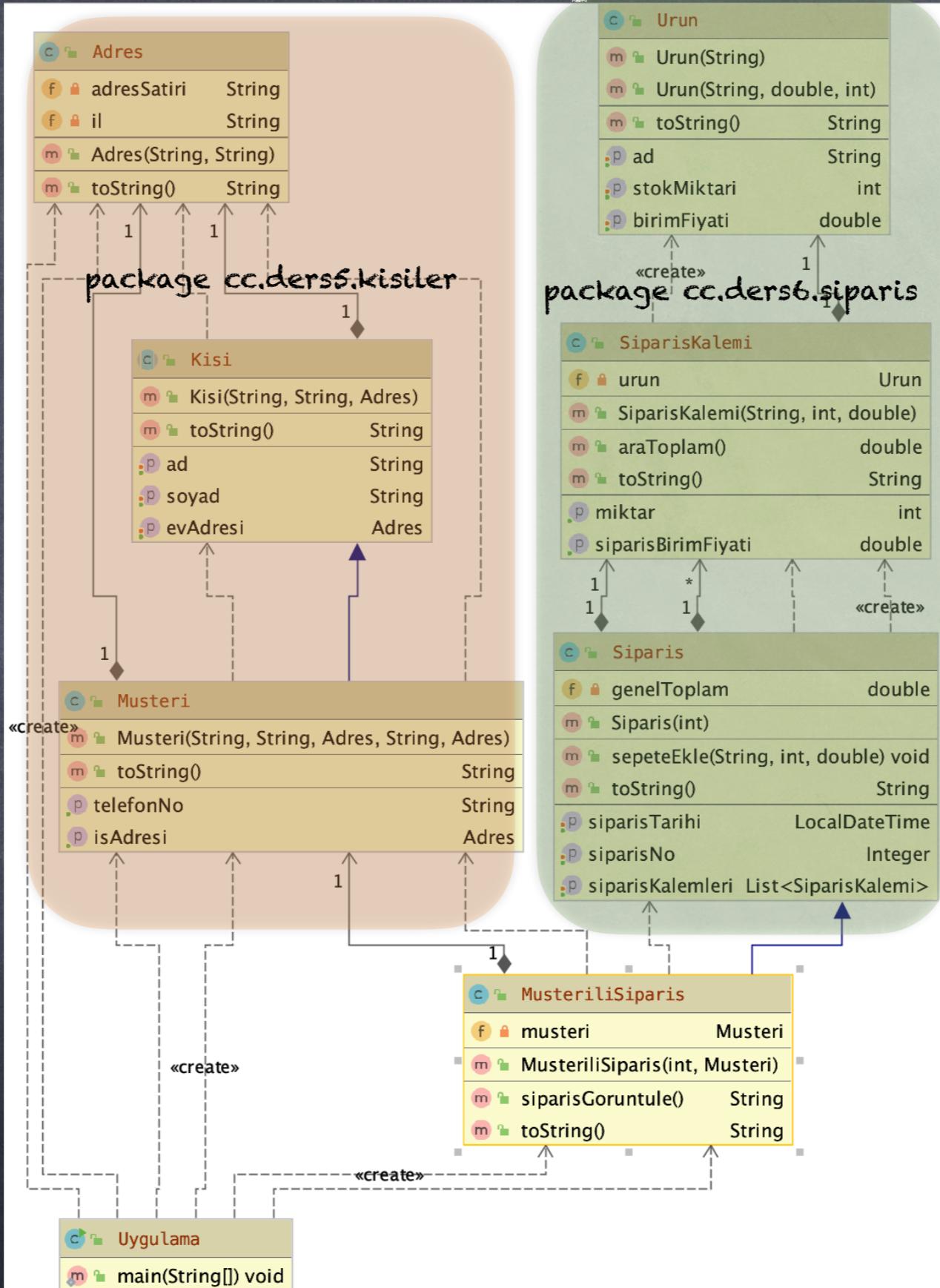


# Nesne Yönelimli Analiz ve Tasarım

Sistem Tasarımı

Sistemin Konuşlandırılması  
Erişim Denetimi (Rol Tabanlı Erişim Denetimi)  
Nesnelerin Depolanması

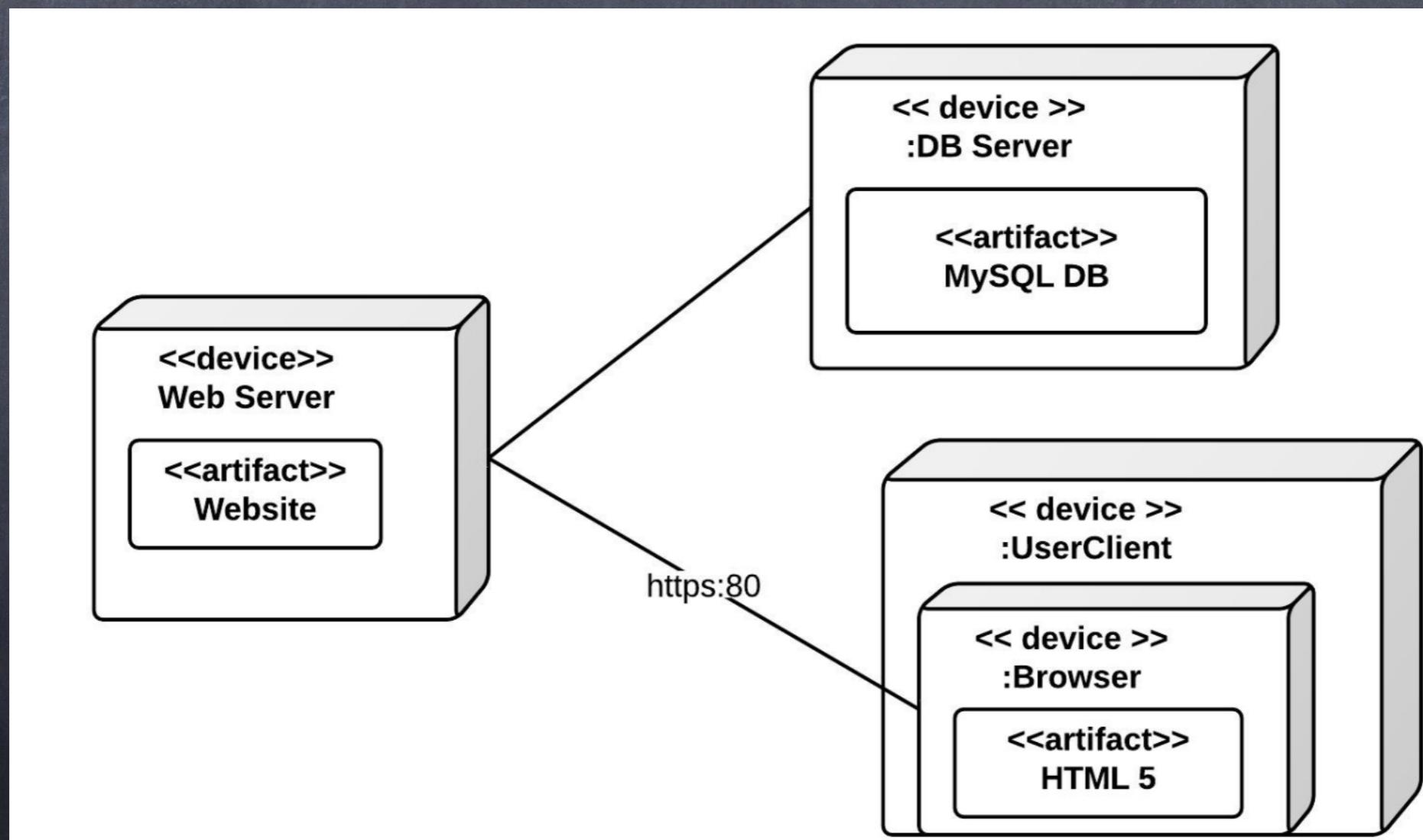
# Laboratuvar Uygulaması



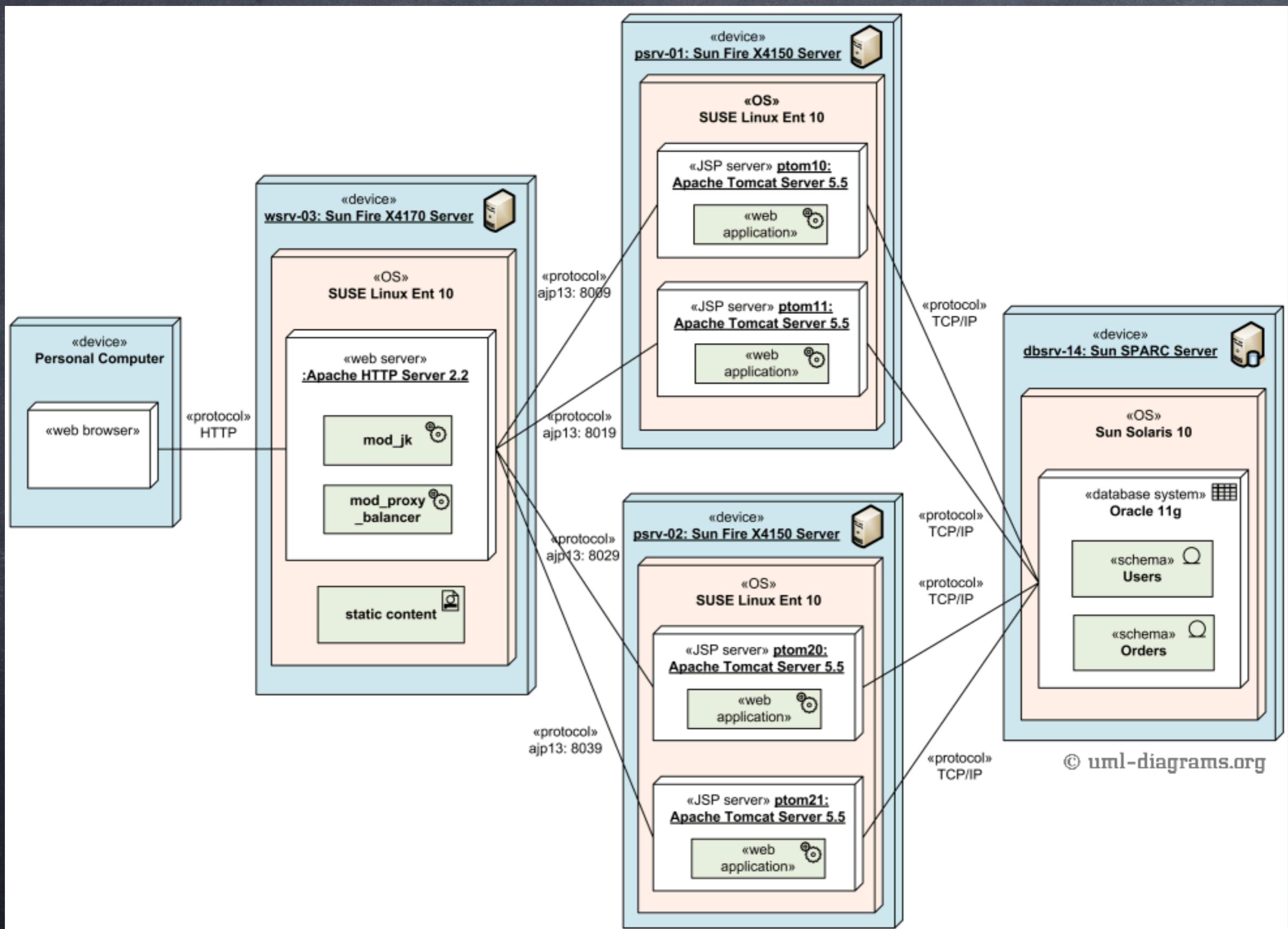
Müşteri Siparişleri

#### 4. Sistemin Konuşlandırılması/ UML "Deployment" Diagramı

- \* Yapısal gösterim şekillerindenidir.
- \* Yazılım ürünlerinin/bileşenlerinin (artifact/component) düğümlere (cihaz, işletim sistemi, sanal makine vb.) konuşlandırmasıyla ilgili bilgi verir.
- \* Yazılımların konuşlandırıldığı fiziksel topolojiyi gösterir.



## 4. Sistemin Konuşlandırılması- Ölçeklenebilir Web Uygulaması



# 5. Erişim Denetimi (Rol Tabanlı Erişim Denetimi)

Kullanıcılar: insan, yazılım, diğer sisteler

Kaynaklar / Değerler / Varlık

İstekim sistemi: Beller, deşyeler, kelesir, part...

Veritabanı yönetim sistemi: Tablolar, satılık yordan, view...

Uygulama yazılımları: Sayfalar/cagrıçalar, modüller (sniff fonksiyonları...)

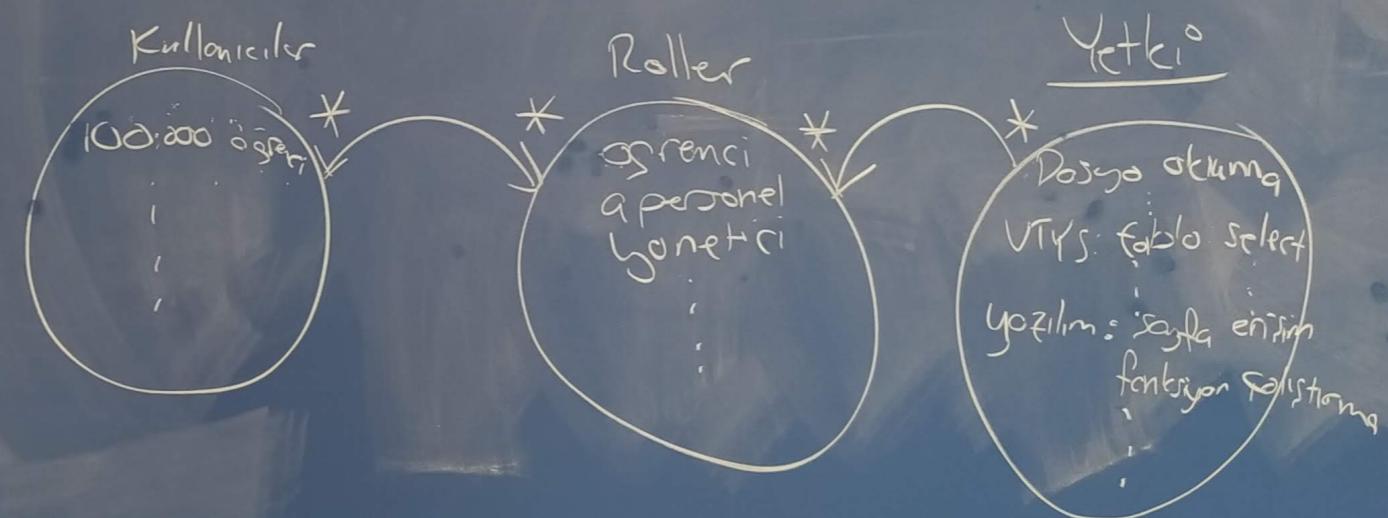
Authorization (yetkilendirme):

Kullanıcıların kaynaklar üzerindeki haklarının belirlenmesi.

Authentication (kimlik doğrulama)

Rol tabanlı erişim denetimi

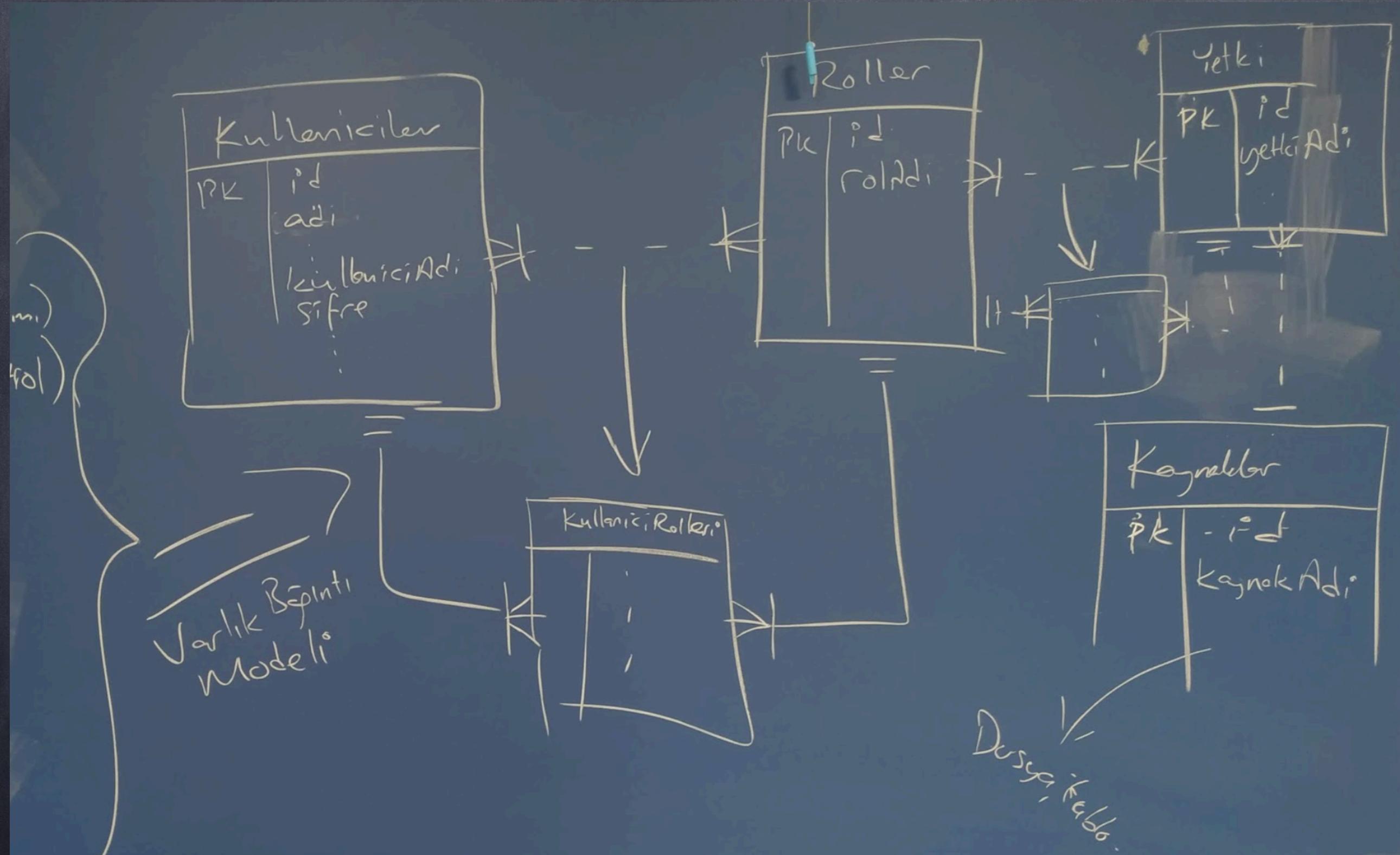
Yetkilendirme → RBAC (Role Based Access Control)  
Erişim Denetimi



Yetki (Permission): Kaynaklar (Sayfa, modül, veritabanı) üzerindeki sahip olunan haklar.

Yetkiler rollerle ilişkilendirilir, kullanıcılar rollere üye yapılır. Böylece yetkilerin yönetimi daha kolay yapılır.

## 5. Erişim Denetimi (Rol Tabanlı Erişim Denetimi)



# 5. Erişim Denetimi (Rol Tabanlı Erişim Denetimi)

## Case Study: Online Shopping Application

ACTIVITY MATRIX				
		External Entities & Roles		
Assets	Action	Guest	Registered User	Admin
Customer Data	Create	Always	Own Data	Always
	Read	Never	Own Data	Always
	Update	Never	Own Data	Always
	Delete	Never	Own Data	Always
Login Page	Access	Always	Always	Always
Customer Profile Page	Access	Never	Always	Never
Login Function	Execute	Always	Always	Always
Add Shopping Chart Function	Execute	Never	Always	Never
Checkout Function	Execute	Never	Always	Never

## 6. Erişim Denetimi (Web Application (Spring Security))

```
@Override  
protected void configure(HttpSecurity http) throws Exception {  
    http  
        .authorizeRequests() ExpressionUrlAuthorizationConfigurer<HttpSecurity>.ExpressionInterceptUrlRegistry  
            .antMatchers("/resources/**", "/").permitAll()  
            .antMatchers("/dashboard").hasRole("ADMIN")  
            .anyRequest().authenticated()  
            .and() HttpSecurity  
        .formLogin() FormLoginConfigurer<HttpSecurity>  
            .defaultSuccessUrl("/dashboard")  
            .and() HttpSecurity  
        .logout() LogoutConfigurer<HttpSecurity>  
            .permitAll();  
}
```

```
@GetMapping("/registeredUser")  
@PreAuthorize("hasAuthority('RegisteredUser')")  
public String moderatorAccess() {  
    return "RegisteredUser Board.";  
}  
  
 @GetMapping("/administrator")  
 @PreAuthorize("hasAuthority('Administrator')")  
 public String adminAccess() { return "Admin Board."; }
```

# 6. Erişim Denetimi (Veritabanı Yönetim Sistemi)

## Yetkilendirme İşlemleri

PUBLIC: Tüm roller / kullanıcılar.

kullaniciAdi: Tek bir kullanıcı.

ALL: Tüm yetkiler.

- rol1 isimli role customers tablosu üzerinde seçim yapma yetkisi ver.

```
GRANT SELECT ON "customers" TO "rol1";
```

- Tüm rollere customers tablosu üzerinde kayıt ekleme yetkisi ver.

```
GRANT INSERT ON "customers" TO PUBLIC;
```

- rol1 isimli kullanıcıya customers tablosu üzerinde tüm yetkileri ver.

```
GRANT ALL ON "customers" TO "rol1";
```

- rol1 isimli rolün customers tablosu üzerindeki güncelleme yetkisini geri al.

```
REVOKE UPDATE ON "customers" FROM "rol1";
```

- rol1 isimli rolün customers tablosu üzerindeki tüm yetkilerini geri al.

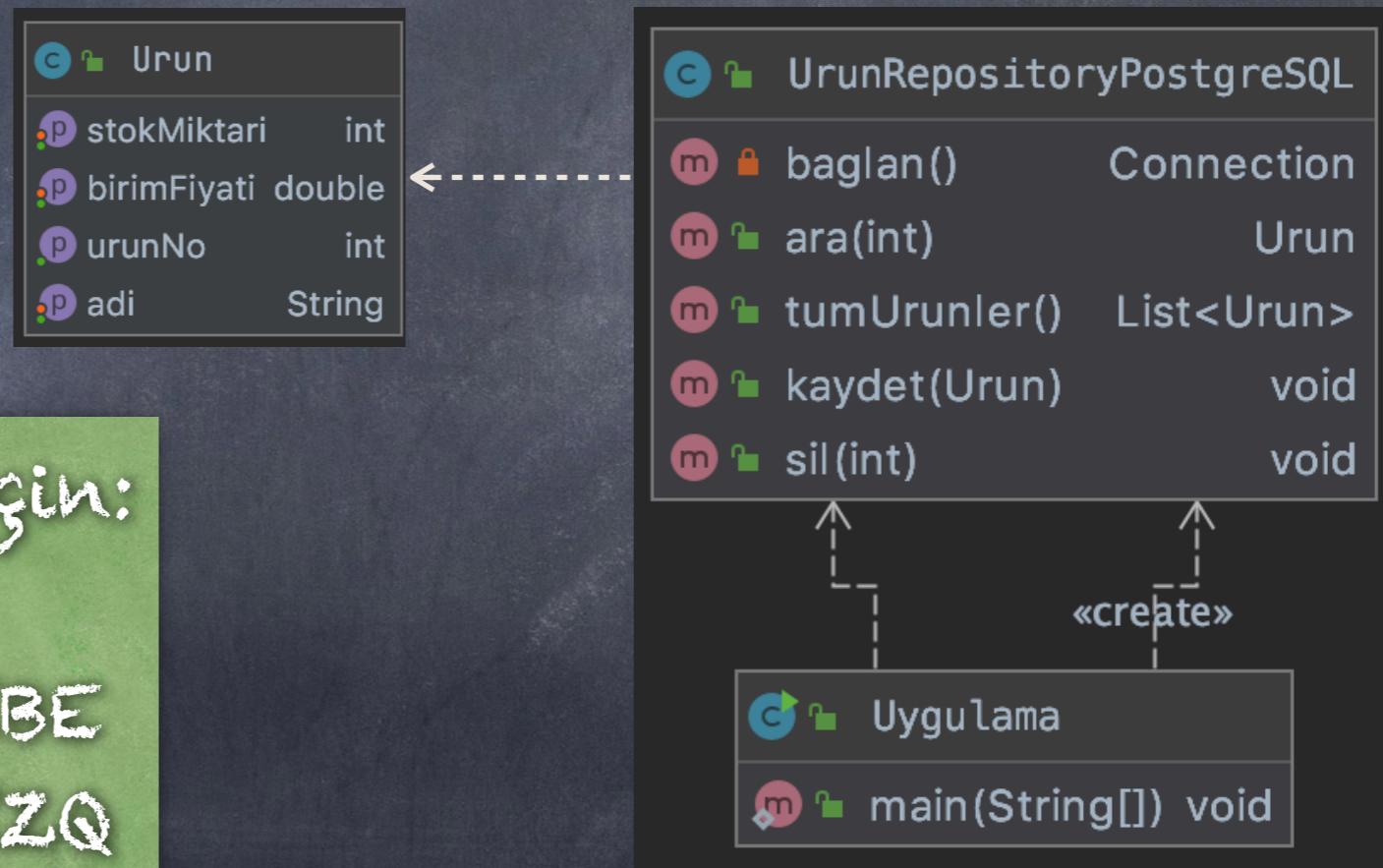
```
REVOKE ALL ON "customers" FROM "rol1";
```

- rol1 kullanıcısının Sema1 içerisindeki nesnelere ait tüm yetkileri geri alınır.

```
REVOKE ALL ON SCHEMA "Sema1" FROM "rol1";
```

## 7. Nesnelerin Depolanması

- \* Bazı nesnelerin (entity object) kalıcı olarak saklanması gereklidir.
- \* Kalıcı depolama için klasik dosyalar ya da veritabanı yönetim sistemleri (ilişkisel VYS, NoSQL vb.) kullanılır.
- \* Veritabanı sürücülerinin yüklenmesi ve uygulamalar içerisinde kullanılması:
  - \* <https://github.com/celalceken/DatabaseManagementSystems/blob/master/VYS06.md>
- \* İlişkisel VYS kullanımı durumunda saklanacak nesneler için tablo oluşturulur.
- \* Tablodaki her satır bir nesneye, tablo ise sınıfa karşılık gelir.



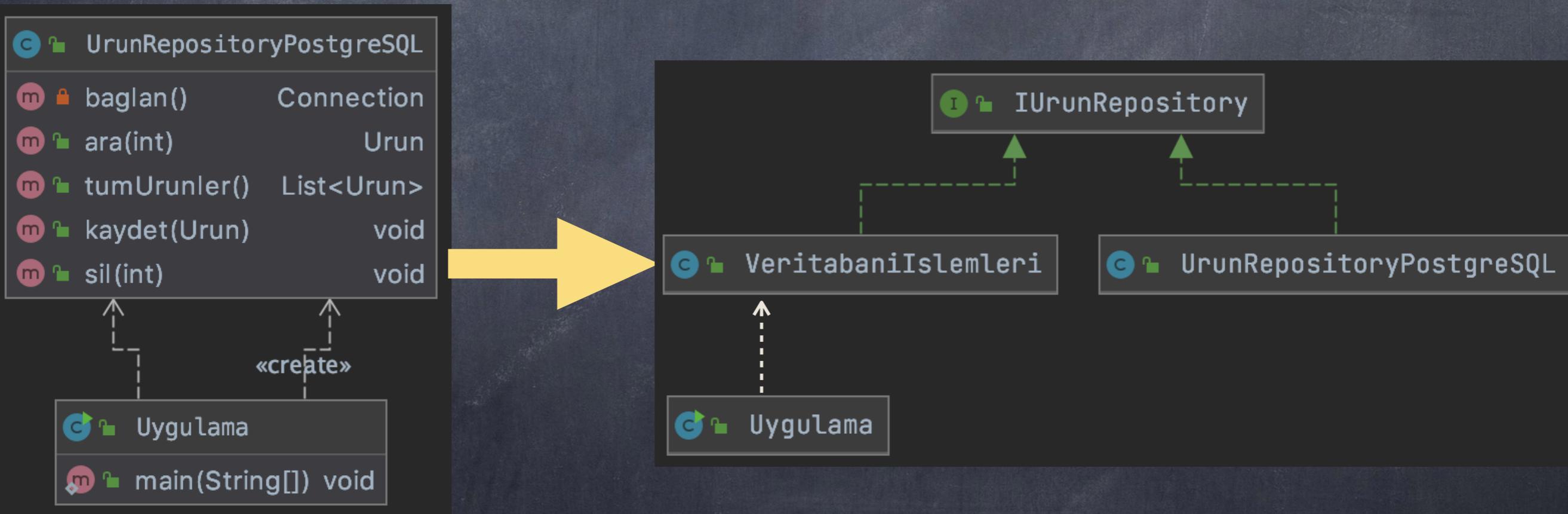
Yandaki örneğin anlatımı için:

<https://youtu.be/SXH3GbaUKBE>

<https://youtu.be/ccWYVUQKXZQ>

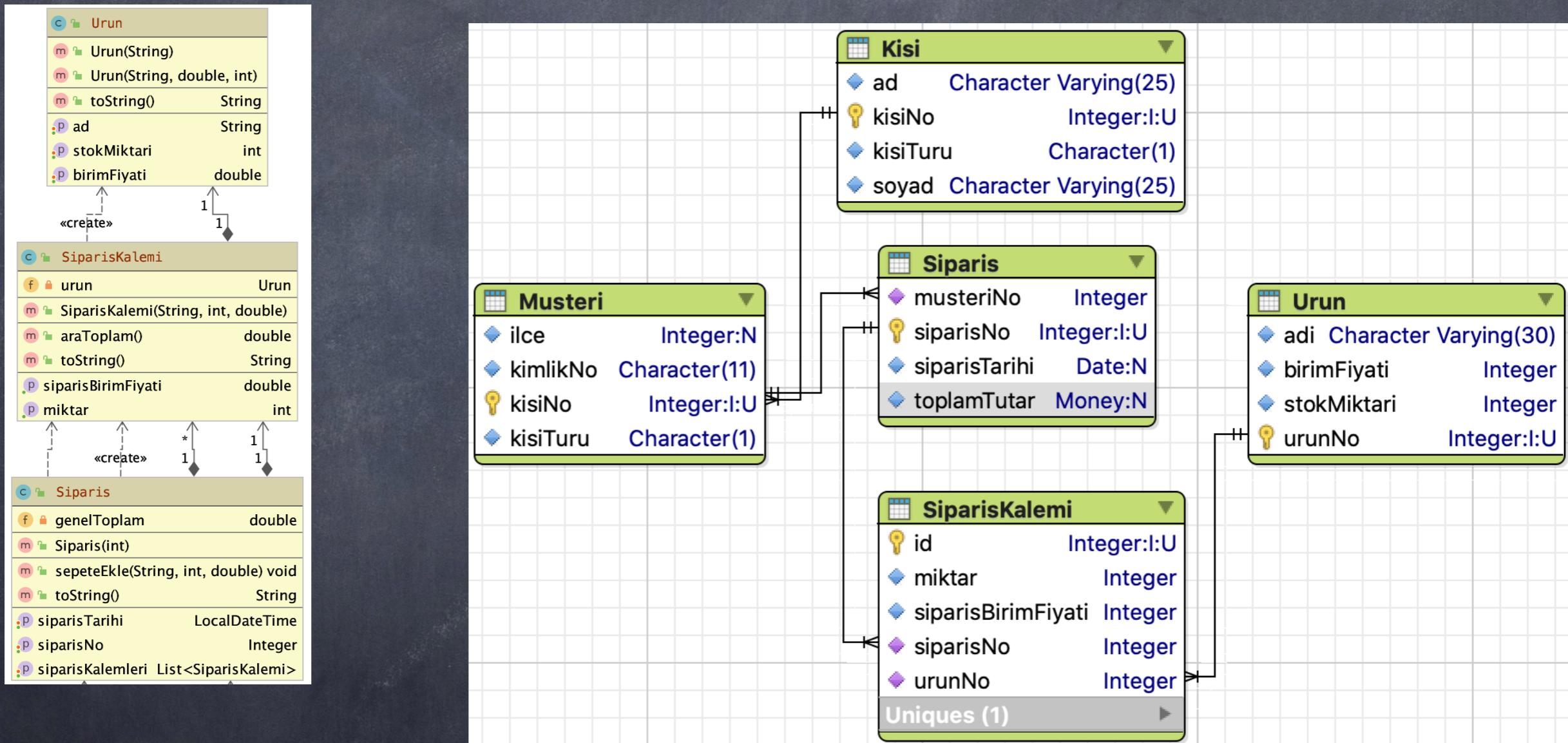
## 7. Nesnelerin Depolanması

- \* Depolama işleminin iş mantığından soyutlanması-Interface ile soyutlama (Repository Pattern)
  - \* istemci kod, gerçekleme kısmındaki değişikliklerden etkilenmez
  - \* gerçeklemeler henüz ortada yokken istemci kod içerisinde kullanılabilir (Yeni özellik eklendiğinde istemci kod değiştirilmez)
  - \* gerçekleme ve istemci modüller aynı anda farklı kodlayıcılar tarafından oluşturulabilir (takım çalışması)
  - \* SOLID Dependency inversion ve open-closed prensipleriyle ilgilidir.



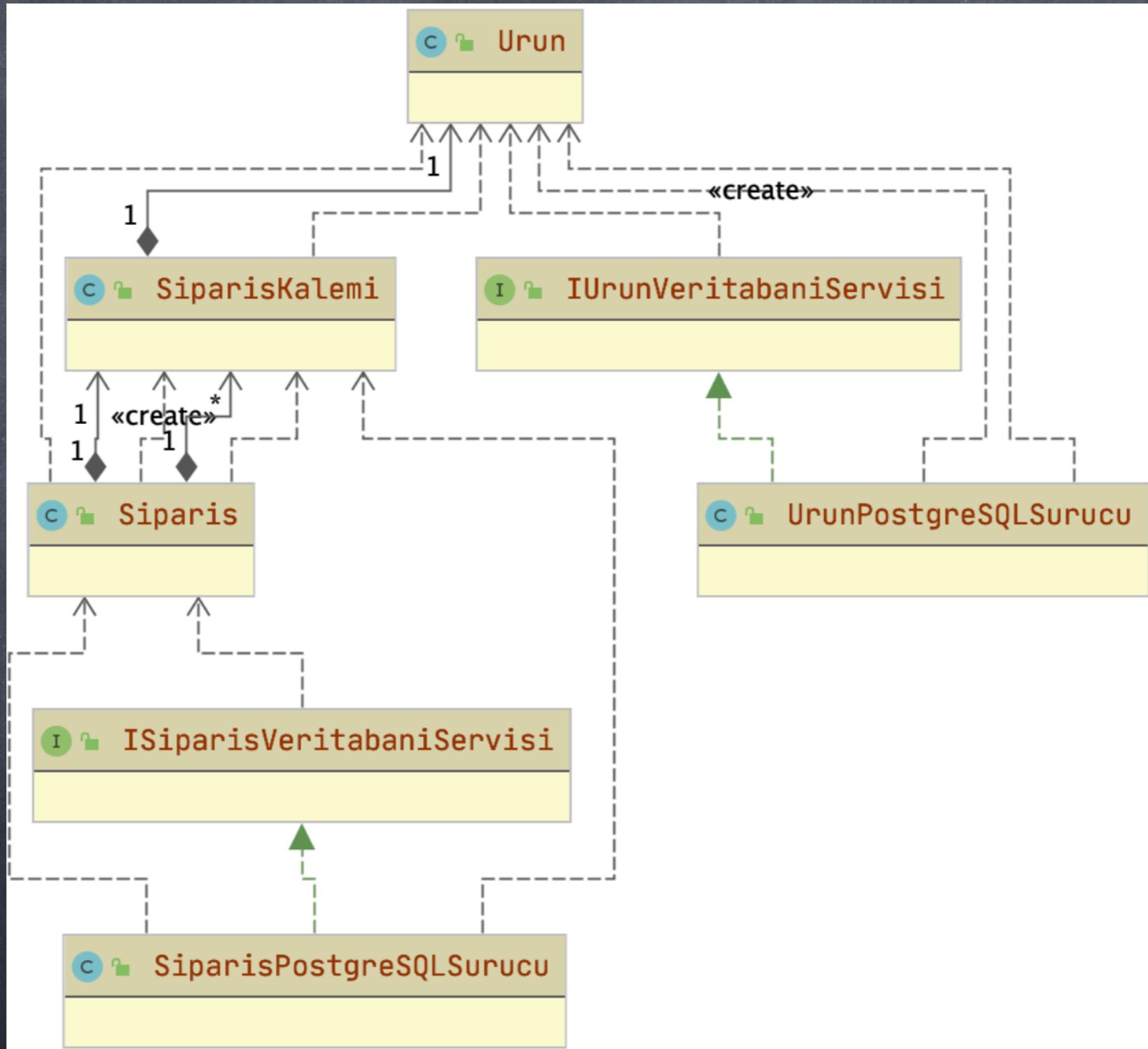
# 7. Nesnelerin Depolanması

- \* SiparisKalemi, Siparis ile Urun arasındaki çok-çok bağıntısının sonucu olan tabloyu ifade eder.

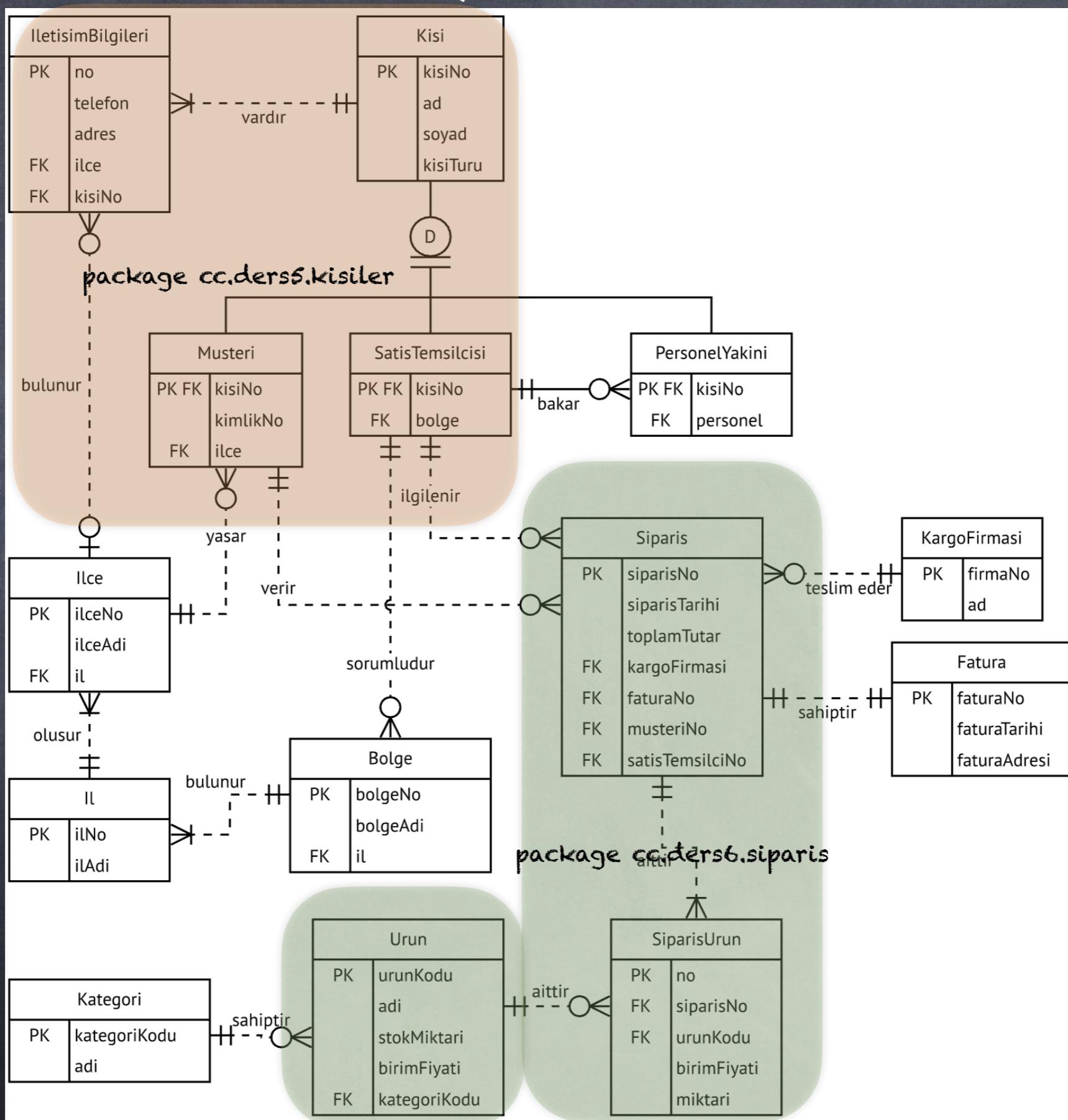


## 7. Nesnelerin Depolanması

- \* SiparisKalemi, Siparis ile Urun arasındaki çok-çok bağıntısının sonucu olan tabloyu ifade eder.



# 6. Nesnelerin Depolanması/VB Modeli



# 6. Nesnelerin Depolanması/ATM Sistemi

```
ders2
ders3
ders5
ders6
ders7.atm
• veritabani
  • PostgreSQLSurucu
  • Uygulama
  • Araclar
  • ATM
  • ATMUygulamasi
  • BakiyeGoruntuleme
  • BankaBilgiSistemi
  • Ekran
  • IBankaBilgiSistemi
  • IEkran
  • IIslem
  • IKartBolmesi
  • IParaBolmesi
  • ISistemTurleri
  • ITusTakimi
  • KartBolmesi
  • KullaniciHesabi
  • Musteri
  • ParaBolmesi
  • ParaCekme
  • ParaYatirma
  • TusTakimi
ders8
ders9.nesnelerindepolanmasi
• crud
• crudinterface
• siparis
ders10
ders12
• abstractfactory
• adapter
• builder

14 ① public KullaniciHesabi kullaniciDogrula(int hesapNumarasi, int sifre) {
15     KullaniciHesabi kullaniciHesabi=null;
16
17     System.out.println("banka bilgi sistemi kullaniciyi doğruluyor..."); 
18     Araclar.bekle(2000);
19     System.out.println("veritabanina baglandi ve kullaniciyi soruluyor..."); 
20     Araclar.bekle(2000);
21     //veritabani.baglan(hesapNumarasi, sifre);
22     try {
23
24         Connection conn = DriverManager.getConnection("jdbc:postgresql://localhost:5432/ATM",
25                                         "postgres", "LecturePassword");
26         if (conn != null)
27             System.out.println("Veritabanina baglandi!");
28         else
29             System.out.println("Baglanti girisimi basarisiz!");
30
31         String sql= "SELECT * FROM \"MusteriHesabi\" WHERE \"hesapNumarasi\"=$hesapNumarasi AND sifre=$sifre";
32
33         Statement stmt = conn.createStatement();
34         ResultSet rs = stmt.executeQuery(sql);
35
36         //***** Baglanti sonlandirma *****
37         conn.close();
38         int hesapNo;
39         double bakiye;
40         String adi;
41         String soyadi;
42         while(rs.next())
43         {
44             hesapNo = rs.getInt("hesapNumarasi");
45             bakiye = rs.getDouble("bakiye");
46             adi = rs.getString("adi");
47             soyadi = rs.getString("soyadi");
48
49             kullaniciHesabi = new KullaniciHesabi(hesapNo, bakiye, adi, soyadi);

```

# 6. Nesnelerin Depolanması/ATM Sistemi

```
ders7.atm
veritabani
PostgreSQLSurucu
Uygulama
Araclar
ATM
ATMUygulamasi
BakiyeGoruntuleme
BankaBilgiSistemi
Ekran
IBankaBilgiSistemi
IEkran
IIslem
IKartBolmesi
IParaBolmesi
IslemTurleri
ITusTakimi
KartBolmesi
KullaniciHesabi
Musteri
ParaBolmesi
ParaCekme
ParaYatirma
TusTakimi
ders8
ders9.nesnelerindepolanmasi
ders10
ders12
ders13
TA-ME

12 public class PostgreSQLSurucu implements IBankaBilgiSistemi {
13
14     public KullaniciHesabi kullaniciDogrula(int hesapNumarasi, int sifre) {...}
15
16     public void hesapGuncelle(KullaniciHesabi kullaniciHesabi) {
17         try {
18             Connection conn = DriverManager.getConnection("jdbc:postgresql://localhost:5432/ATM",
19                     "postgres", "LecturePassword");
20             if (conn != null)
21                 System.out.println("Veritabanına bağlandı!");
22             else
23                 System.out.println("Bağlantı girişimi başarısız!");
24
25
26             String sql= "UPDATE \"KullaniciHesabi\" SET bakiye="+kullaniciHesabi.getBakiye() +
27                         "WHERE \"hesapNumarasi\":"+kullaniciHesabi.getHesapNumarasi();
28
29             Statement stmt = conn.createStatement();
30             stmt.executeUpdate(sql);
31
32             conn.close();
33
34             stmt.close();
35         } catch (Exception e) {
36             e.printStackTrace();
37         }
38
39         System.out.println("banka bilgi sistemi hesabı güncelledi...");
40     }
41 }
```