

Nesne Yönelimli Analiz ve Tasarım

Sistem Tasarımı

Tasarım Hedeflerinin Belirlenmesi
Sistemin alt bileşenlerine ayrıştırılması
Yazılım mimarileri

Nesne Yönelimli Tasarım- Sistem Tasarımı

- * Analiz aşaması uygulama alanına (application domain) odaklanır.
- * Uygulama alanı gerçekleştirecek yazılımın çalışacağı ortamı ifade eder.
- * Tasarım aşaması çözüm alanına (solution domain) odaklanır.
- * Tasarım aşaması, sistem tasarımını ve nesne tasarımını olarak iki bölüme ayrılabilir.
- * Sistem tasarımında sistem alt bileşenlerine ayrılır. Sistemle ilgili tasarım hedefleri belirlenir ve kısıtlar ihlal edilmeden bu hedeflere ulaşmak için gerekli mimari oluşturulur.

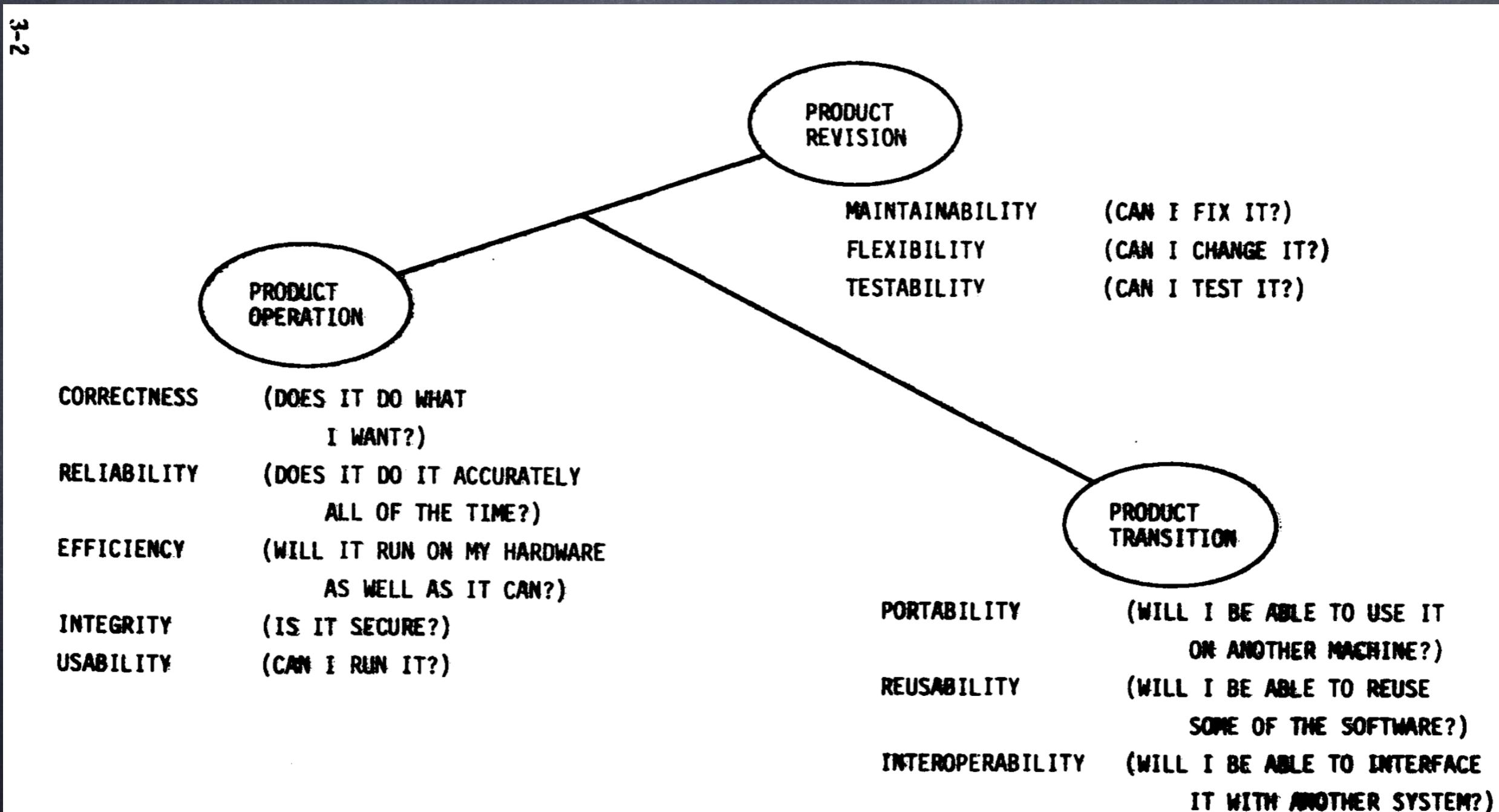
Nesne Yönelimli Tasarım- Sistem Tasarımı

* Analiz aşaması:

- * gereksinimler, kısıtlar belirlenir
 - * kullanım durumları tanılanır
 - * analiz sınıf şeması/ nesne modeli, dinamik modeller (sıralama şeması, etkinlik şeması vb.) oluşturulur
- ## * Sistem Tasarım Aşaması:
- * tasarım hedeflerinin (kalite ölçütleri) belirlenmesi,
 - * sistemin alt bileşenlerine ayrıştırılması
 - * tasarım hedeflerine ulaşmayı sağlayacak yazılım mimarisi (mimari stiller), veri modelleri, erişim denetimi yöntemleri vb. belirlenir.

1. Tasarım Hedeflerinin Belirlenmesi / Yazılımların Kalitesi (McCall Kalite Üçgeni)

Bir yazılımin kalitesine etki eden faktörler üç sınıfa ayrılır.



McCall, J.A., Richards, P.K. and Walters, G.F. (1977) Factors in Software Quality, RADC TR-77-369, Rome Air Development Center, Rome.

Kalite Ölçütlerinin Birbirlerine Olumsuz Etkileri

* Integrity <-> Efficiency

- * Güvenlik denetimi ek kontrol ve fazladan yer almamina gelir
- * bankacılık uygulaması

* Portability <-> Efficiency

- * tüm platformları desteklemesi için eklenen denetimler fazadan kaynak kullanımı ve gecikme almamina gelir

* Reusability (Flexibility, Maintainability) <-> Efficiency

- * program daha küçük parçalar halinde yazılsrsa hız azalır.
- * gömülü sistemler, gerçek zamanlı sistemler...
- * Diğerleri neler olabilir?

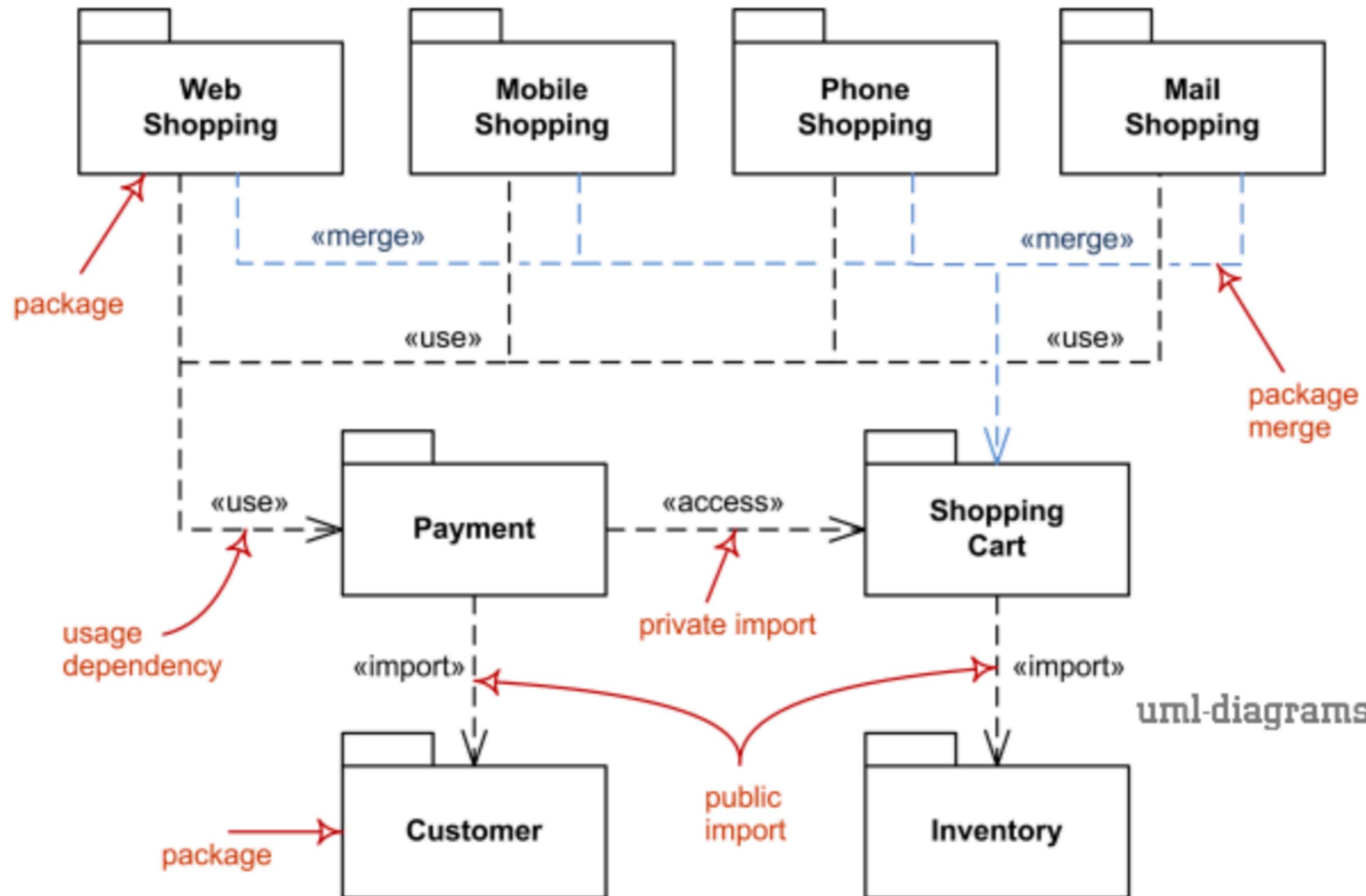
2. Sistemin alt bileşenlerine ayrıştırılması

- * Karmasıklığı azaltmak için yazılım sistemi alt bileşenlerine (alt sistem) ayrılır.
- * Alt Sistemler, birbirleriyle yakından ilişkili; sınıflar, ilişkiler ve kısıtlardan oluşur.
- * Alt sistemler sayesinde;
 - * problem çözümü kolaylaşır,
 - * çok sayıda kişinin aynı projede aynı anda çalışabilmesi sağlanır,
 - * sistemin anlaşılması, anlatılması kolaylaşır
 - * bakım kolaylaşır
 - * modülerlik, kod tekrar kullanımı artar...
- * Analiz aşamasındaki alt bileşenler
 - * sınıf
 - * paket (UML paket seması)
- * Tasarım aşamasındaki alt bileşenler için "UML Component" seması kullanılabilir.

2. Sistemin alt bileşenlerine ayrıştırılması- UML Paket Diagramı

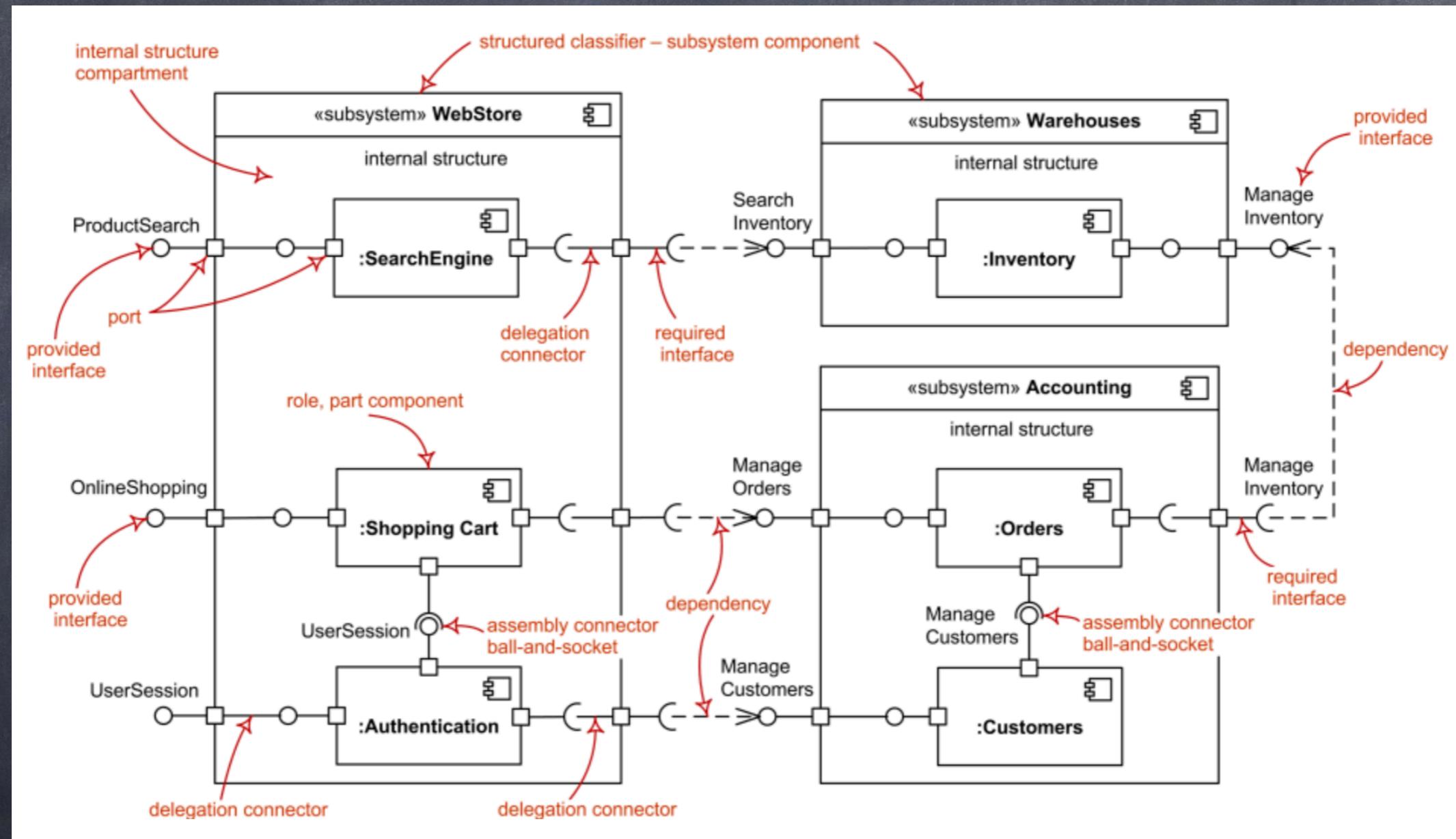
- * Yapısal gösterim şekillerindendir.
- * Paketleri ve paketler arası ilişkileri göstermek için kullanılır.
- * Paketler içerisinde; sınıf, paket, bileşen (component), kısıtlar ve bağımlılıklar olabilir.
- * Java platformunda; **package** ifadesi paket oluşturmak için, **import** ifadesi ise kullanılacak paketleri içe aktarmak için kullanılır.
 - * import java.net.Socket;
 - * import java.io.PrintWriter;
 - * MusteriliSiparis.java
 - * package cc.ders7.lab;
 - * import cc.ders5.Musteri;
 - * import cc.ders6.siparis.Siparis;

2. Sistemin alt bileşenlerine ayrıştırılması- UML Paket Diagramı



2. Sistemin alt bileşenlerine ayrıştırılması- UML "Component" Diagramı

- * Yapısal gösterim şekillерindendir. "Component" (Bileşen), sistemin diğer bölümleriyle haberleşmeyi sağlayan arayzlere (API-Application Programming Interface) sahip alt sistemler ya da sistemlerdir. Ortak amaca hizmet eden sınıfların bir araya getirilmesiyle oluşur.



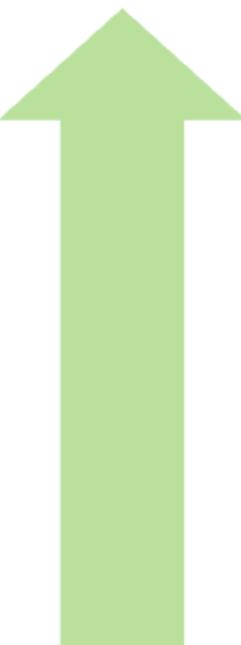
- * Şekilde bağıntılı üç alt sistem (WebStore(Web mağazası), Warehouse(depo, ambar), Accounting(Muhasebe)) bulunmaktadır. Alt sistemler içerisinde başka alt sistemlerde bulunmaktadır.
- * "SearchEngine" sağladığı "ProductSearch" arayüzüyle ürünlerin aranmasını sağlar. Bu işlem için "Inventory" bileşeninin sağladığı "SearchInventory" arayüzüünü kullanır.
- * "Shopping Cart" bileşeni gevrimiçi alış veriş arayüzü sağlar. Alış veriş için kullanıcının oturum açması gereklidir. Siparişlerin yönetimi için "Orders" bileşeninin "Manage Orders" arayüzüünü kullanır.
- * ...

2. Sistemin alt bileşenlerine ayrıştırılması

İyi bir tasarım için

High Coherence

Modüller tek ve özel bir işi, mükemmel bir şekilde yapmalı. Alt sistemler içerisindeki sınıflar benzer işi yapmalı ve ilgili olmalı. "Cohesion" Bunun ölçüsüdür .



Low Coupling

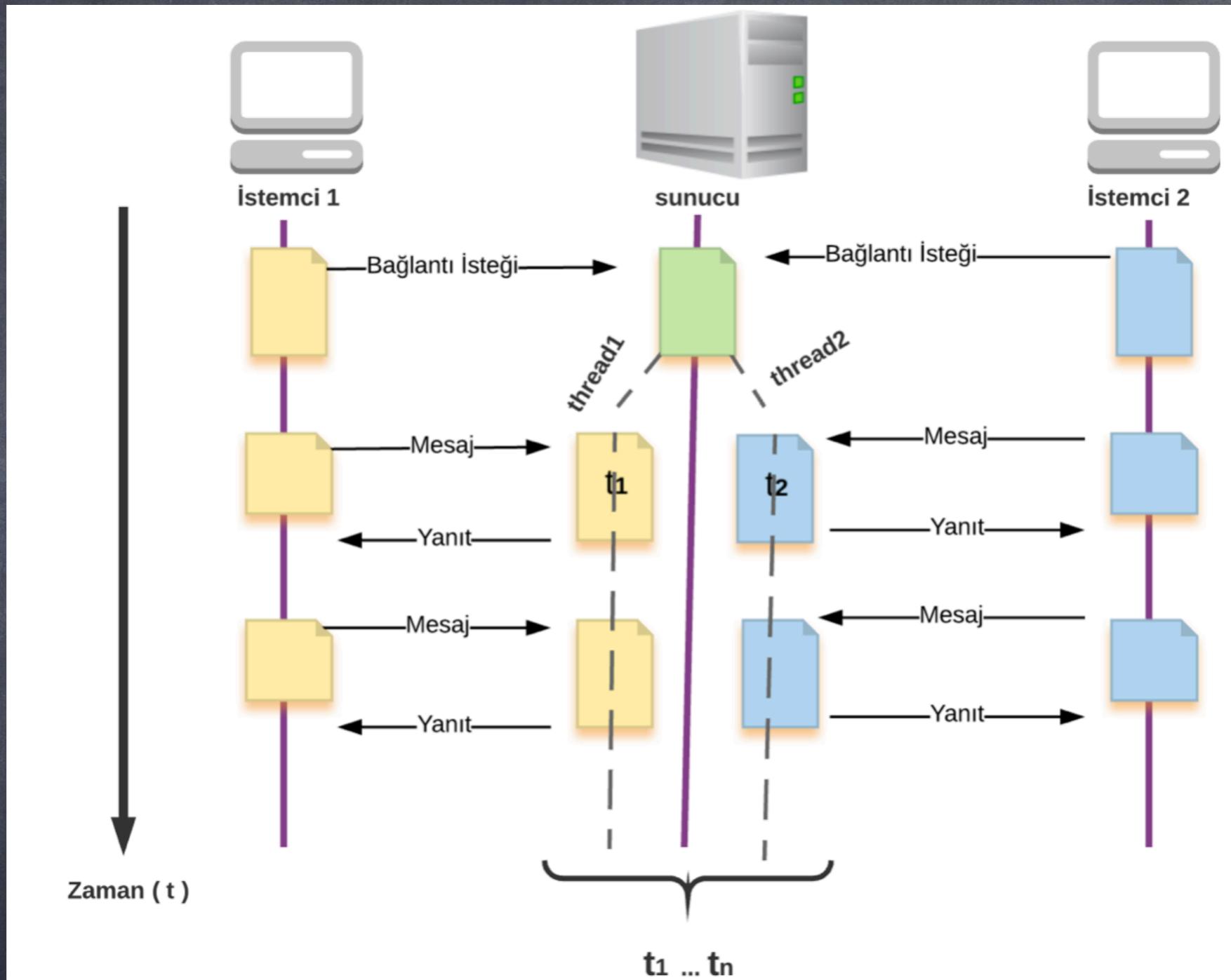
Aradaki bağıntılar ne kadar fazla olursa nesne içerisinde yapılacak köklü değişiklik ona bağlı olan modülleride etkileyecektir.



3. Yazılım mimarileri (mimari stiller)

- * İstemci/Sunucu (Client/Server)
- * Üç Katmanlı Web Mimarisi (Three-tier Web Architecture)
- * Servis Yönelimli Mimari (Service-Oriented Architecture (SOA))
- * Model/Görünüm/Denetleyici (Model/View/Controller(MVC))
- * Peer-To-Peer

3. Yazılım mimarileri (İstemci/Sunucu)



İstemci/Sunucu Örneği

<https://github.com/celalceken/NesneYonelimiAnalizVeTasarimDersiUygulamalari/tree/master/Ders8/IstemciSunucuMimarisi>

3. Yazılım mimarileri (Üç Katmanlı Web Mimarisi (Three-tier Web Architecture))

- * Sunum, iş mantığı ve veri yönetimi fonksiyonlarının fiziksel olarak ayrıldığı İstemci/Sunucu mimarisidir.
- * Modülerlik, başaram, bölümler birbirlerinden bağımsız geliştirilebilir

Sunum Katmanı

Kullanıcı Arayüzleri
(HTML5, JavaScript, CSS)

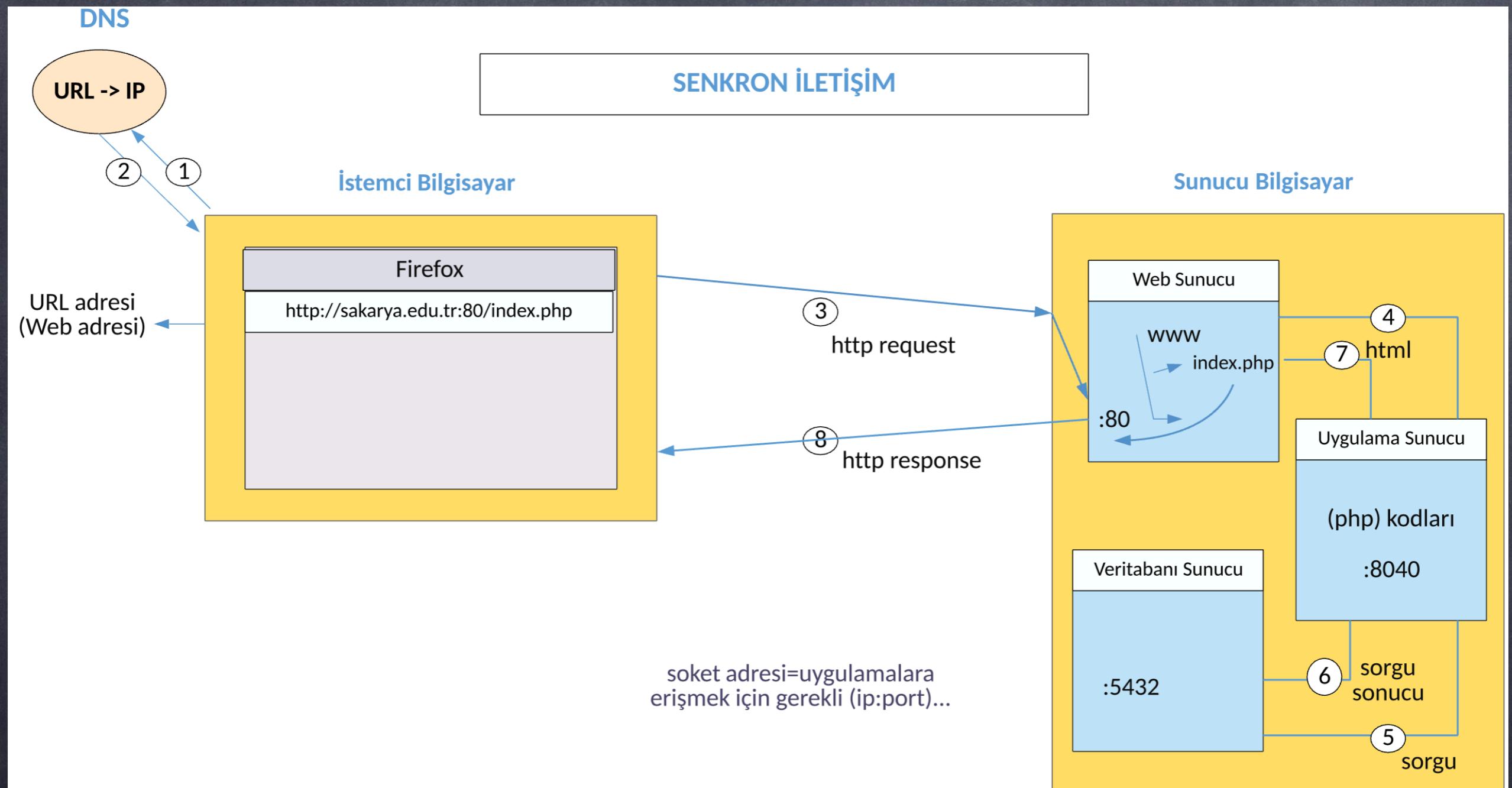
Uygulama Katmanı

İş Mantığı
(PHP, Java, Spring, NodeJS...)

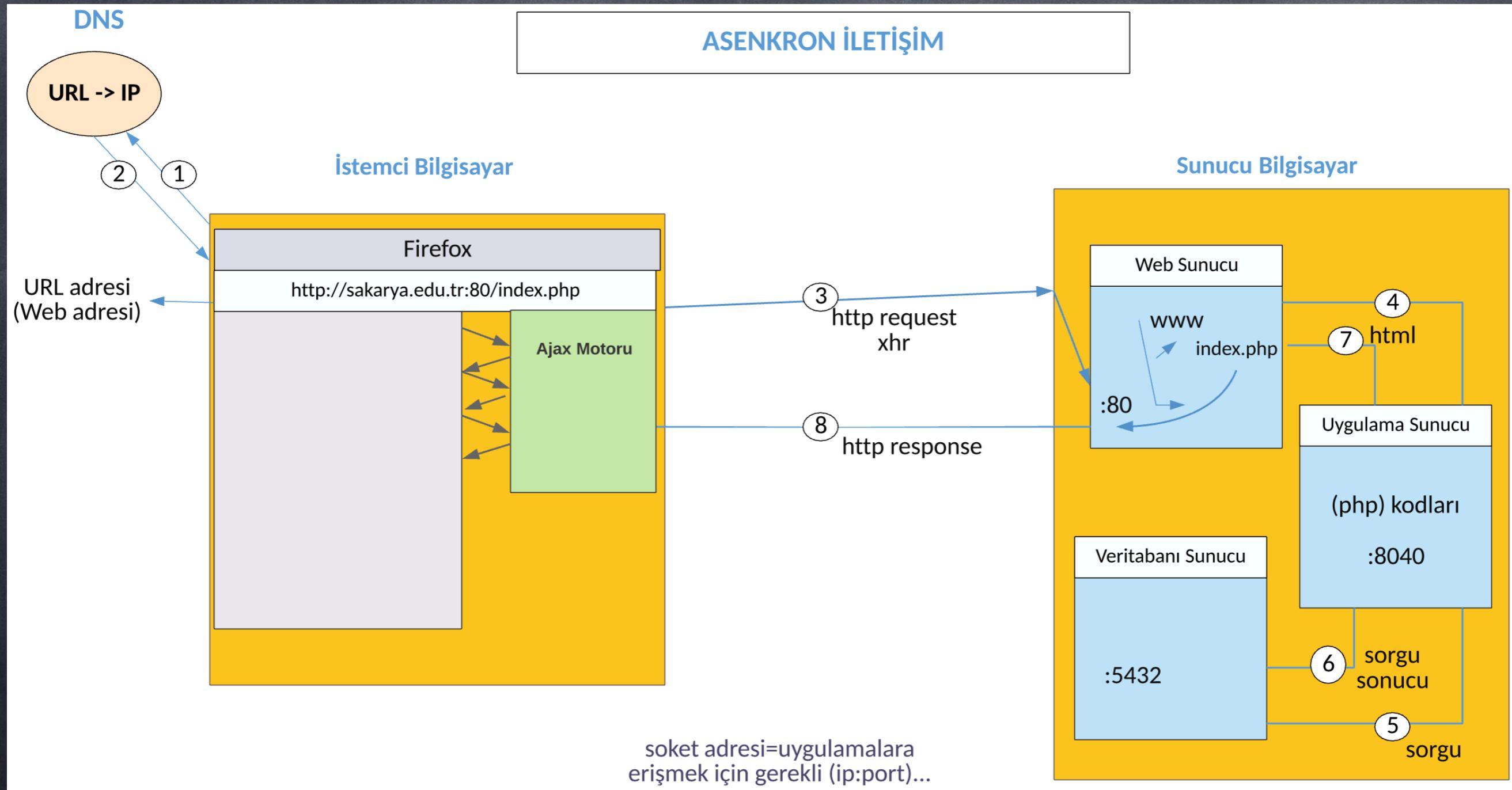
Veri Katmanı

Veri Yönetimi
(PostgreSQL, MongoDB, MySQL, Oracle, MSSQL, Cassandra...)

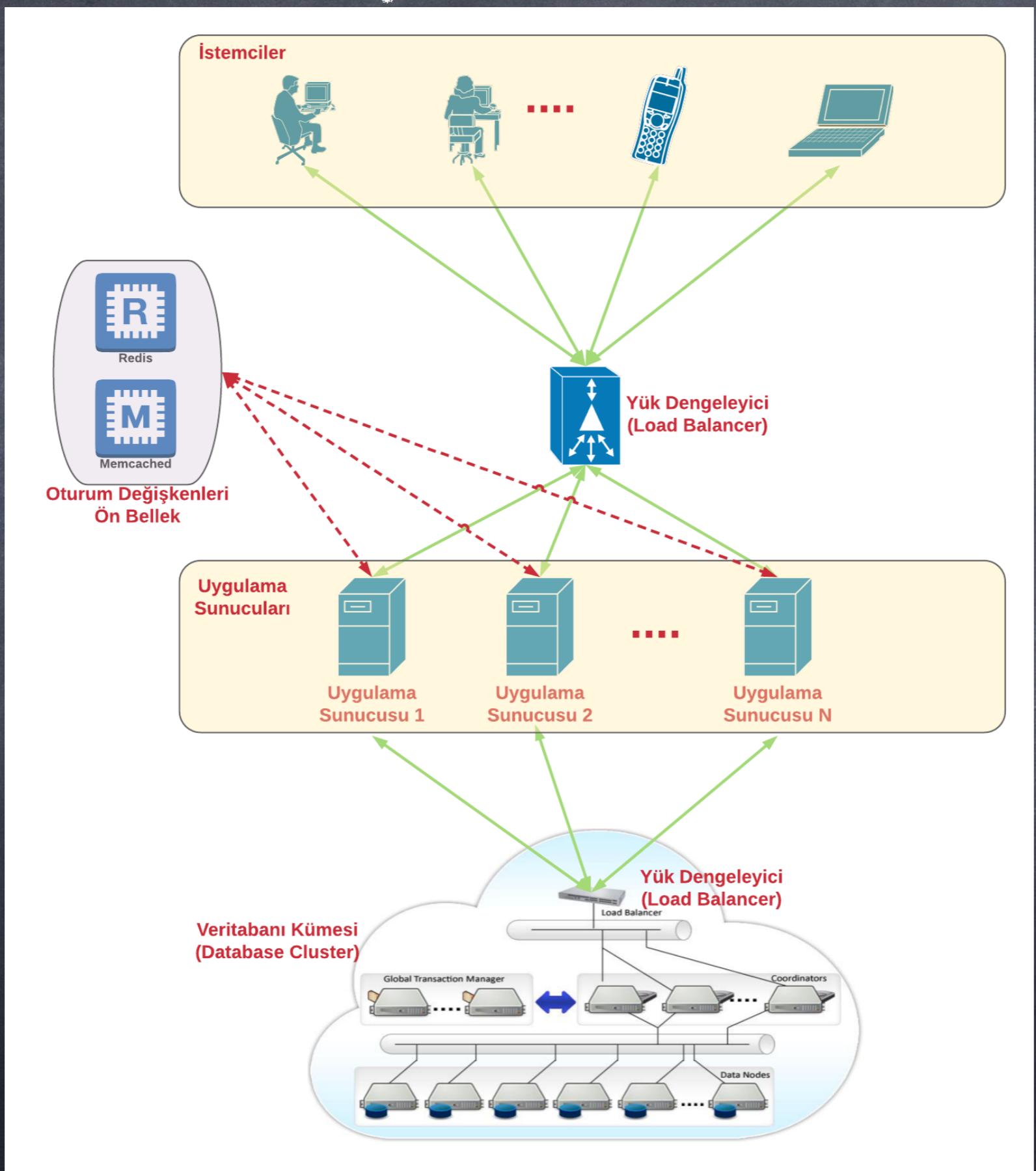
3. Yazılım mimarileri (İstemci/Sunucu (Web Uygulama Mimarisi))



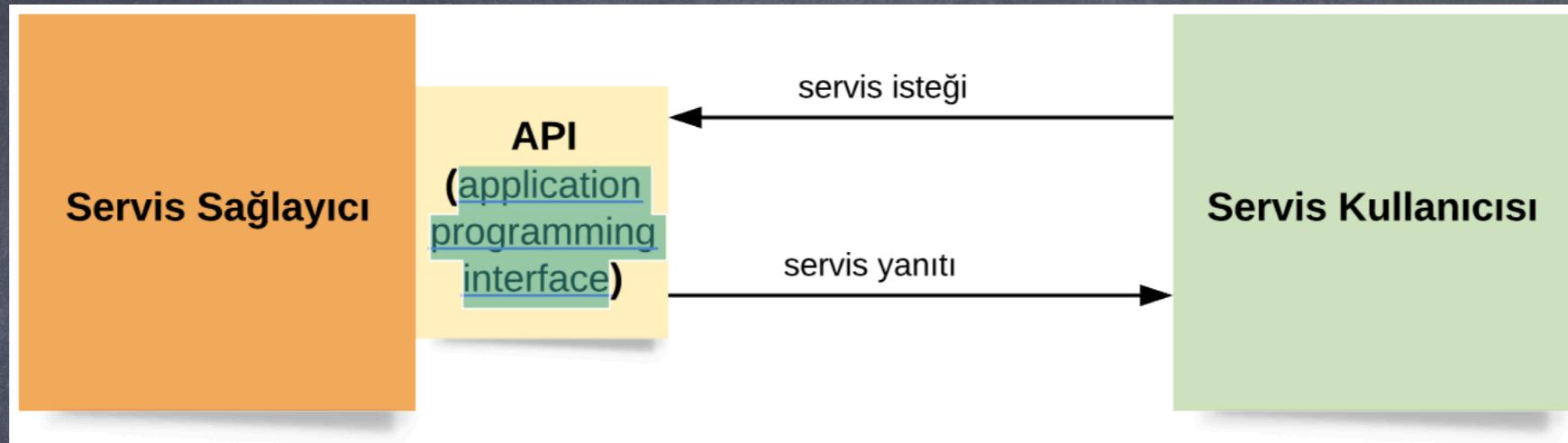
3. Yazılım mimarileri (İstemci/Sunucu (Web Uygulama Mimarisi))



3. Yazılım mimarileri (İstemci/Sunucu(Ölçeklenebilir Web Uygulama Mimarisi))



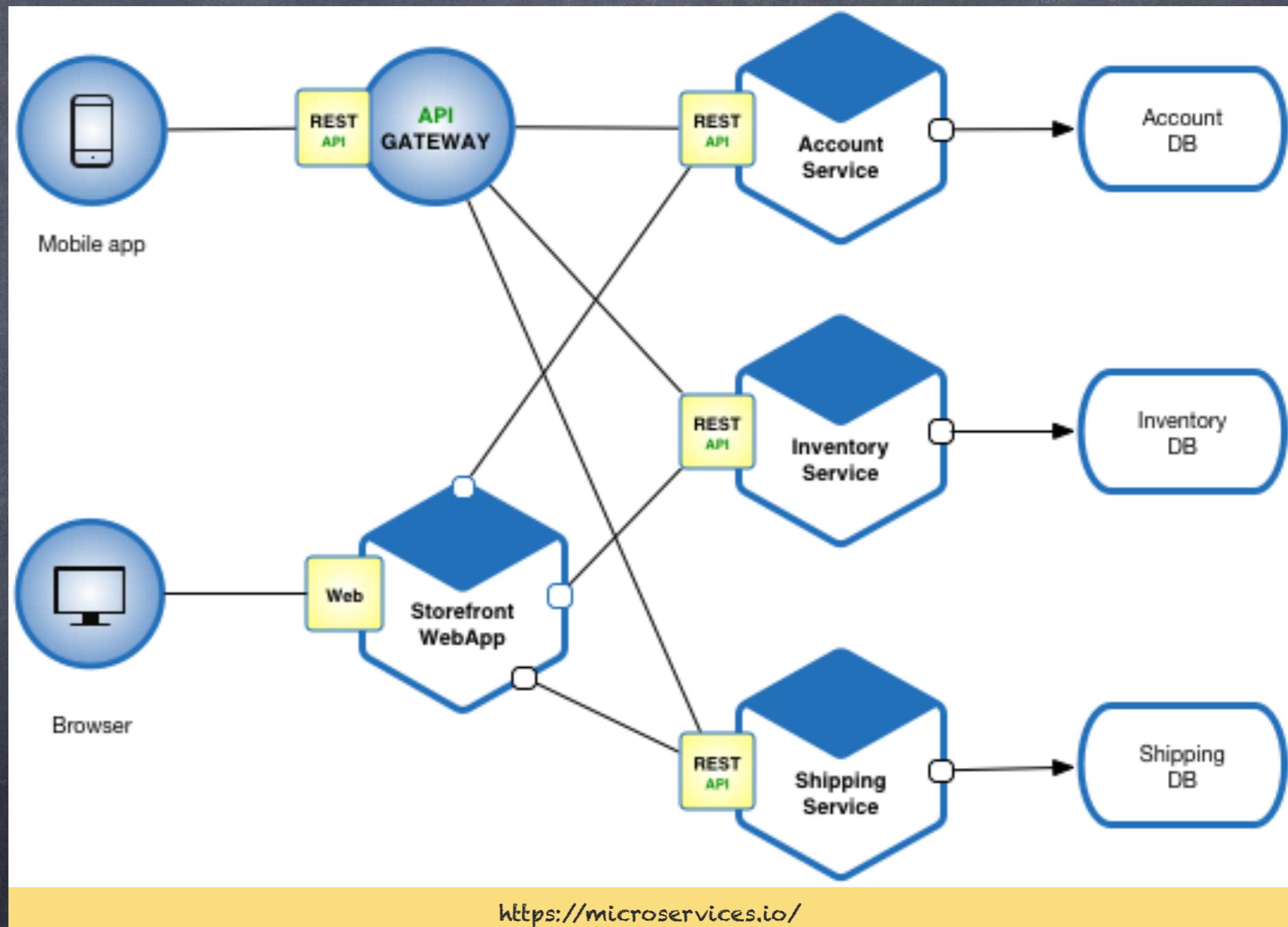
Servis Yönelimli Yazılım Mimarisi (Service-Oriented Architecture (SOA))



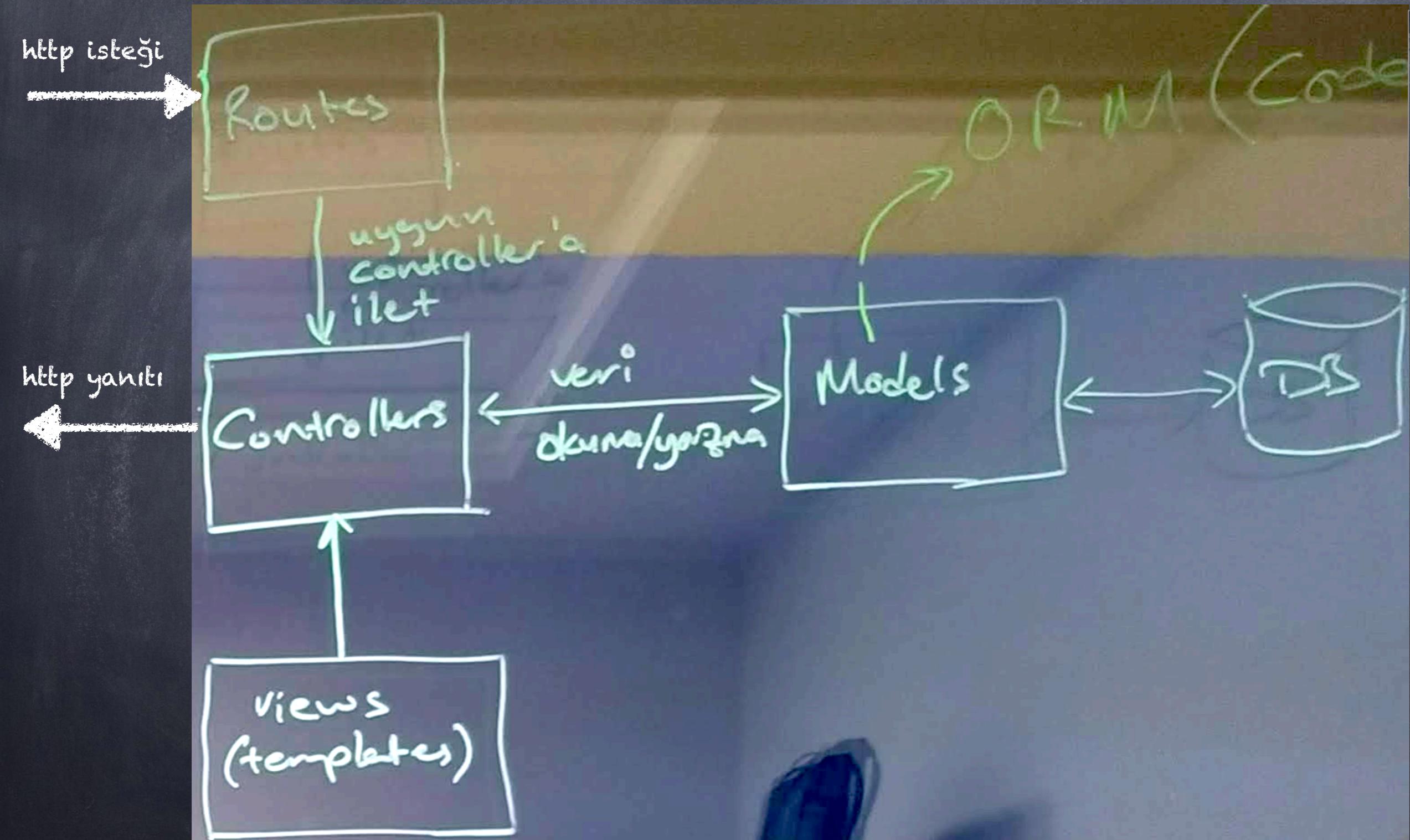
- * Sistemlerin birlikte çalışmasını sağlar
- * Servisler birbirlerinden bağımsızdır.
- * RESTful, gRPC, SOAP, XML-RPC, Jini, COBRA...
- * RESTful ile json desteği
- * Nesnelerin Interneti (IoT) sistemleri de yoğun olarak kullanmaktadır

Mikro Servis Mimarisi

- * servis yönelimli mimarinin modern bir türevidir.
- * uygulamalar çok sayıda servisin bir araya getirilmesiyle oluşturulur.
- * servislerin; bakımı ve testi kolaydır, bağımsız olarak konuşlandırılar/geliştirilirler.



3. Yazılım mimarileri (MVC)



3. Yazılım mimarileri (MVC)

The screenshot shows a Java code editor with the file `OgrenciController.java` open. The code implements the Model-View-Controller (MVC) pattern. It includes annotations for Request Mapping and Autowiring, and methods for displaying student lists and adding new students.

```
17  @RequestMapping("/ogrenciview")
18  public class OgrenciController {
19      private final OgrenciService ogrenciService;
20
21      @Autowired
22      public OgrenciController(OgrenciService ogrenciService) { this.
23
24          /* @GetMapping("ogrenciEkle")
25          public String showSignUpForm(Ogrenci student) {
26              return "add-student";
27          }*/
28
29
30      }
31
32      @GetMapping("ogrenciler")
33      @
34      public String ogrencileriListele(Model model) {
35          model.addAttribute("ogrenciler", ogrenciService.findAll());
36          model.addAttribute("toplam", ogrenciService.count());
37          //System.out.println(ogrenciService.count());
38          return "Ogrenciler/listele.html";
39      }
40  }
```

The code editor's sidebar shows the project structure with packages like controller, model, repository, service, util, and SpringapplicationApplication, along with sources, static assets, and css files.