**Yazılımcı Adayı Mülakat Akışı**

1. Aday Tanıma
2. Teknik Mülakat

Burada yer alan sorular aşağıdaki kategorilerde sınıflandırılmıştır. Yazılımcının deneyimine göre seçilebilir.

Genel

APP Katmanı

UI Katmanı

Server Katmanı

Veritabanı Katmanı

1. Teknik mülakatın başarılı olduğu düşünülüyorsa akışa devam edilir. Değilse bu aşamada adaydan soru ve cevapları adımına geçilebilir.
2. Bankamız ve Bilgi Teknolojileri hakkında bilgilendirme

* Altyapı Ve Sistem İşletim
* Temel Bankacılık
* Kurumsal Mimari(IB ve Mobil Şube)
* Bilgi Güvenliği

1. Genel outsource işe alım süreçlerinde bilgilendirme
2. Adayın Soru ve Cevapları

***Soru:* Neden Nesne Yönelimli Programlama yapmalıyız ?**

Bu konu biraz derin ancak kısaca daha düzenli ve okunabilir kod yazmak, kod tekrarını engellemek için kullanmalıyız deseniz bile en azından fikir sahibi olduğunuzu belli etmiş olursunuz.

***Soru :*** **SOLID Nedir ? Solid Yazılım Prensipleri Nelerdir ?**

**SOLID** yazılım prensipleri; geliştirilen yazılımın esnek, yeniden kullanılabilir, sürdürülebilir ve anlaşılır olmasını sağlayan, kod tekrarını önleyen ve Robert C. Martin tarafından öne sürülen prensipler bütünüdür.

* Geliştirdiğimiz yazılımın gelecekte gereksinimlere kolayca adapte olması,
* Yeni özellikleri kodda bir değişikliğe gerek kalmadan kolayca ekleyebileceğimiz
* Yeni gereksinimlere karşın kodun üzerinde en az değişimi sağlaması,
* Kod üzerinde sürekli düzeltme hatta yeniden yazma gibi sorunların yol açtığı zaman kaybını da minimuma indirmektir.

[**S — Single-responsibility principle**](https://gokhana.medium.com/single-responsibility-prensibi-nedir-kod-%C3%B6rne%C4%9Fiyle-soli%CC%87d-c8b1602be602)🡺Bir sınıf (nesne) yalnızca bir amaç uğruna değiştirilebilir, o da o sınıfa yüklenen sorumluluktur, yani bir sınıfın(fonksiyona da indirgenebilir) yapması gereken yalnızca bir işi olması gerekir.  
[**O — Open-closed principle**](https://gokhana.medium.com/open-closed-prensibi-nedir-kod-%C3%B6rne%C4%9Fiyle-soli%CC%87d-679619d5376a)🡺Bir sınıf ya da fonksiyon halihazırda var olan özellikleri korumalı ve değişikliğe izin vermemelidir. Yani davranışını değiştirmiyor olmalı ve yeni özellikler kazanabiliyor olmalıdır.  
[**L — Liskov substitution principle**](https://gokhana.medium.com/liskov-substitution-prensibi-nedir-kod-%C3%B6rne%C4%9Fiyle-soli%CC%87d-3cfc1cd63c1a)🡺 Kodlarımızda herhangi bir değişiklik yapmaya gerek duymadan alt sınıfları, türedikleri(üst) sınıfların yerine kullanabilmeliyiz.  
[**I — Interface segregation principle**](https://gokhana.medium.com/interface-segregation-prensibi-nedir-kod-%C3%B6rne%C4%9Fiyle-soli%CC%87d-ac0fd6812ecf)🡺Sorumlulukların hepsini tek bir arayüze toplamak yerine daha özelleştirilmiş birden fazla arayüz oluşturmalıyız.  
[**D — Dependency Inversion Principle**](https://gokhana.medium.com/dependency-inversion-prensibi-nedir-kod-%C3%B6rne%C4%9Fiyle-soli%CC%87d-b61296523565)🡺Sınıflar arası bağımlılıklar olabildiğince az olmalıdır özellikle üst seviye sınıflar alt seviye sınıflara bağımlı olmamalıdır.

***Soru :*** **Tasarım Desenleri hakkında bilgi sahibi misiniz? Neden ihtiyaç duyarız?**

Nedir?

Tasarım kalıpları, yazılım geliştiricilerin yazılım geliştirme sırasında karşılaştıkları genel sorunların çözümüdür. Bu çözümler, uzun bir süre boyunca sayısız yazılım geliştirici tarafından deneme yanılma yoluyla elde edilmiştir. Daha sonra belli problemler için buldukları optimum çözümlere isimler vermişlerdir.

Ancak tasarım kalıplarının amacı tekerleği baştan icat etmemek olduğu için, var olan çözümleri bilip uygun olan yerde kullanmak sizin için en doğrusu olacaktır.

Tasarım Kalıpları Çeşitleri

Yazılım tasarım kalıpları genel olarak 3 ana başlıkta incelenir. Bunlar şunlardır:

* **Creational Patterns (Yaratımsal Kalıplar):**Bu tasarım deseni nesneleri doğrudan new operatörü kullanarak oluşturmak yerine nesne oluşturma mantığını gizleyerek sınıflardan nesne oluşturmaya alternatif çözümler sunar. Bu program akışında hangi nesneye ihtiyaç varsa onu oluşturmada esneklik ve kolaylık sağlar.
* **Structural Patterns (Yapısal Kalıplar):**Bu tasarım deseni nesneler arasındaki ilişkinin yapısını düzenlemek için çözümler sunar.
* **Behavioral Patterns (Davranışsal Kalıplar):**Bu tasarım deseni çalışma zamanında nesneler arasındaki davranışlar için çözümler sunar.

|  |  |  |
| --- | --- | --- |
| **Creational Patterns** | **Structural Patterns** | **Behavioral Patterns** |
| Factory Pattern Abstract Factory Pattern Builder Pattern Prototype Pattern | Adapter Pattern  Bridge Pattern  Filter Pattern  Composite Pattern  Decorator Pattern  Facade Pattern  Flyweight Pattern  Proxy Pattern | Chain of Responsibility Pattern  Command Pattern  Interpreter Pattern  Iterator Pattern  Mediator Pattern  Memento Pattern  Observer Pattern  Null Object Pattern  Strategy Pattern  State Pattern  Visitor Pattern |

**Factory Pattern**Nesne oluşturma işleminin bir iş kuralı dahilinde olduğu durumlarda kullanıcıyı kuralın dışında tutarak nesne oluşturmak için kullanılır. Kurallara uygun bir şekilde soyut tipleri implemente eden  somut nesneler oluşturularak kullanıcılara verilir.

**Abstract Factory Design Pattern**  
Factory nesneleri oluşturmak için kullanılan bir desendir. Yani Abstract Factory deseni ile soyut bir factory nesnesi oluşturulur, bu factory nesnesinden de belirlenen kural dahilinde bir nesne oluşturulur. Küme alt küme mantığına benzer bir yapı söz konusudur.

**Adapter Design Pattern**  
Birbiriyle uyumsuz iki farklı interface arasında anlaşma sağlayan bir köprü vazifesi görür. Bunu gerçek hayatta cep telefonu ile elektrik prizi arasında görev yapan şarj aletine benzetebiliriz.

**Template Pattern**  
Bir algoritma için soyut tipte bir iskelet oluşturarak bu iskeletten somut süreç adımları oluşturmamızı sağlar. Algoritmanın adımlarını somut sınıflarda tanımlarız. İskelet görevi gören tip bir şablon niteliğindedir.

**Singleton Design Pattern**  
Bir sınıfa ait sadece tek bir nesne oluşturmayı ve gerektiğinde ona ulaşabilmeyi garanti eder. Yani nesne oluşturma üzerine oluşturulmuş bir desendir.

**Decorator Pattern**  
Nesnelere kompozisyon yoluyla yeni davranışlar eklememizi sağlar. Bu işlemi aynı temel sınıftan türeterek veya ortak bir interface uygulayıp nesneye enjekte ederek yapabilmek mümkündür.

**State Pattern**  
Bir nesnenin iç durumunda meydana gelen değişikliklerin nesnenin davranışını değiştirmesine izin verir. İç durumdan kasıt interface tipleridir.

**Strategy Pattern**  
Çalışma zamanında dinamik olarak algoritma değişikliğine olanak sağlayan bir desendir. Algoritmalar somut nesnelerde saklanır. Client kod ise algoritma sınıfının türetildiği abstract veya interface türünü tanır.

* **Not:** State ve Strategy pattern diyagram olarak birbirine benzerdir. Ancak çalışma prensipleri açısından birbirlerinden farklıdırlar. State pattern için nesnenin *NE* olduğu önemlidir. Strategy için nesnenin *NASIL* çalıştığı önemlidir.

**Specification Pattern**  
İş nesneleri içerisinde gömülü olan seçim kriterlerini başka nesneler ile paylaşamazsınız veya tekrar tekrar kullanamazsınız. Specification pettern, bu problemi ortadan kaldırır.

**Composite Pattern**  
Nesnelere ağaç yapısında ya da hiyerarşik topluluklar halinde gruplanabilme yeteneği sunar. Örneğin bir kategorinin alt kategorisi olabileceği gibi onun da alt kategorisi olabilir.

**Proxy Design Pattern**  
Bir sınıfın işlevselliği başka bir sınıf ile temsil edilir. Cache mekanizmaları oluşturulurken kaynak kullanımını azaltmak için kullanılabilir. Örneğin veri tabanından veri getiren bir nesne, aynı verilere ihtiyaç duyulduğunda tekrar veri tabanına gitmeye gerek duyulmadan kullanılabilir.

**Builder Design Pattern**  
Kuramsal bir iş akışını işletmek için gerekli olan kompleks bir nesnenin basit nesneler ile adım adım oluşturulmasına olanak sağlar.

***Soru: Mvc Patterni Nedir ? Neden böyle bir yapıya ihtiyaç var ve kullanılmaktadır.***

MVC model-view-controller şeklinde kod yazmamızı sağlayan bir yapıdır.

Geliştirilen web uygulamalarındaki bölümlerin artmasıyla beraber yazılan kodlarda artmaktadır.

Yazılan kodların artması, geliştirilen uygulama için uygun bir mimarinin seçilmemesi kodların karmaşıklaşmasına, kod bakım süresinin uzamasına neden olur.

Microsoft bu sorunu çözmek için web uygulamalarında sıklıkla kullanılan MVC mimarisini ASP.NET içerisine entegre etmiştir.

MVC yani **M**odel-**V**iew-**C**ontroller en basit şekilde geliştirilen uygulamaların parçalara ayrılmasıdır.

Parçalara ayrılmasındaki neden SOC **S**eparation **O**f **C**oncerns yani sorumlulukların ayrılması prensibidir.

Her bir parçanın kendine göre görevleri vardır.

* **Model** – Uygulamada kullanılacak verilerin bulunduğu, veritabanı ile ilgili bağlantının yapıldığı katmandır.
* **View** – Model içerisindeki verilerin görselleştirilmesinden sorumlu katmandır.
* **Controller** – Model ile View arasındaki bağlantıyı sağlayan katmandır. View’dan gelen ekleme, silme, güncelleme vb. isteklere cevap verir.

Örneğin; Kullanıcı View katmanındaki görsel form alanı aracılığıyla ekleme işlemi için veri gönderir.

Gönderilen veriler Controller katmanında kontrol edilir.

Kontrol edilen veriler uygunsa Model katmanına aktarılarak ekleme tamamlanır.

Kontrol başarısızsa View katmanına geri bildirim gönderilir.

Controller katmanındaki kontrol işlemleri zorunlu değildir.

***Soru : Kalıtım Nedir?***

Bir sınıfın başka bir sınıfa ait özellik ve metotları kendine miras almasına kalıtım denir.

***Soru : Framework (Çatı) Nedir ?***

Bir programlama dilinde kodu daha kısa ve düzenli şekilde yazabileceğimiz yapılar olabilir. Bu yapılara Framework adı verilir. Mesela Php için laravel, Javascript için ReactJS bu yapılara örnektir.

***Soru : Api Nedir ?***

Api başka sunucudan veya uygulamadan kendi uygulamamıza ya da kendi uygulamamızdan başka uygulamalara veri aktarım için kullandığımız bir yöntemdir.

***Soru: Git nedir? Neden kullanılır? Alternatifleri nelerdir?***

Team Foundation Server (TFS), bir yazılım projesinde ekip olarak birlikte çalışmayı amaçlayan ve Microsoft Visual Studio ile entegreli olarak çalışan bir Server sistemidir. Bu sistem sayesinde aynı anda projeyi yapan yazılımcılar ve proje yöneticileri projenin durumunu görebilir, geliştirebilir ve gelişmiş raporlama sistemleri oluşturabilirler.

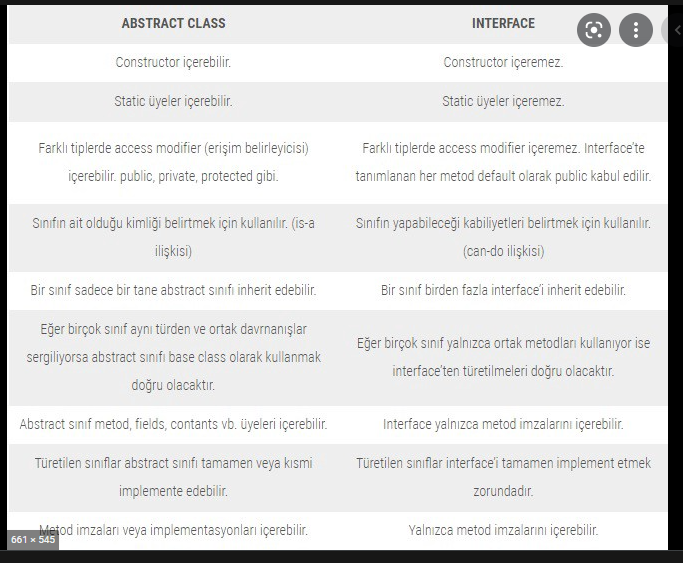
Alternatifleri: TFS, Mercurial SCM, Subversion, VirsualSVN vb.

***Soru(APP Katmanı):*** **Constructor Nedir?**



#### **Soru(APP Katmanı):** Partial class nedir? Neden kullanılır?

Sınıfların karışık yapıda olmasını engellemek için büyük bir sınıfı belirli parçalara bölmek için kullanılan anahtar kelime Partial’dır. Kullanım amacı, Çok sayıda sınıf elemanı içeren bir sınıfın yönetilebilir ve anlaşılabilir olmasını sağlamak içindir. Örneğin partial kullanarak sınıfın özelliklerini bir yerde, metotlarını başka bir yerde tanımlayabiliriz. Her ne kadar sınıfımız parçaya ayrılmış gibi görünse de program çalıştırıldığında sınıfımız tek bir sınıf gibi hareket edecek ve parçaya ayrılmış olan kodlarımız derlenip program tarafından bir bütün olarak çalıştırılacaktır.

***Soru(APP Katmanı):* Interface nedir? Abstract clas nedir? Aralarında bağlantı var mıdır? Farkları nelerdir?** 

**Soru(APP Katmanı):** Generic class nedir?

Normal sınıftan farkı, kullanılacak olan tiplerin sınıf tanımlama aşamasında belirtilmesidir. Yani tasarım anında tipin belli olmamasıdır. Aynı işi farklı veri türlerinde yapmak için birden fazla sınıf tanımlıyorsanız kullanmak gerekir. List nesnesinin generiz olarak aldığı türe göre siz bu listeye belirlediğiniz tipe göre eklemeler yapıyorsunuz eğer generic tip olmadan List olduğunu düşünürsek object olarak geçecektik ve sürekli liste elemanını çekmek isterken cast edecektik.

**Soru(APP Katmanı):** Encapsulation nedir?

Nesne yönelimli programlamanın ilk prensibi **kapsülleme (encapsulation)** olarak adlandırılır. Bu özellik, dilin nesne kullanıcısından gereksiz uygulama ayrıntılarını saklar. Oluşturulan bir **sınıf (class)** içerisinde kullanıcının işlemlerini daha kolay gerçekleştirebilmesi için bazı işlemler birleştirilerek tek bir işlem gibi gösterilir. Bu birleştirme işlemine kapsülleme denir.

**Erişim belirteçleri (access modifier)** sayesinde kapsülleme çok daha kolay yapılmaktadır. Erişim belirteçleri, oluşturulan sınıf veya sınıf içindeki elemanların erişim seviyelerini belirlemek için kullanılan anahtar kelimeler grubuna verilen isimdir. Metotlar ve değişkenler bir anahtar sözcük ile önceden bellirlenen sınırlar dahilinde kullanılabilir. Bu anahtar kelimeler şu şekilde sıralanabilir.

* **public**: Sistemdeki bütün sınıfların erişebilmesini sağlar. Yalnızca aynı proje içinden değil, diğer projelerden de erişim sağlanabilir.
* **private**: Bir **"özellik (property)"in**veya **"metod"un** sadece tanımlandığı sınıftan erişilebilmesini sağlar. Oluşturulan sınıf veya yapıların "public" olması açık bir şekilde belirtilmez ise, derleyici tarafından "private" olarak belirlenir.
* **internal**: Aynı**derleyici (assembly)** içinde bulunan tüm sınıflardan erişim sağlanır.
* **protected**: Sadece tanımlandığı sınıfın içinde ve o sınıftan türetilmiş diğer sınıfların içinde erişilebilir.

Kapsülleme "private" değişkenlerin metotlar gibi kullanılmasına yardımcı olur. **Okuma (Read Only)**işleminin yanısıra **okuma** - **yazma (read - write)** işleminin yapılmasını sağlar.

### **Soru(APP Katmanı): Referance ve Value Type nedir?**

**Value Type**, Bilgileri hafızanın stack bölgesinde direk olarak saklayan nesnelerdir. Null değer alamazlar. Char, Bool örnek verilebilir. Çoğu yerde Value Type için eş anlamlı olduğundan Primitive Type’da denmektedir.

**Referance Type**, Bilgileri hafızanın stack bölgesinde tutar yalnız heap bölgesinde bir referans değeri tutar. Null değer alabilir. String, Class örnek verilebilir

## ***Soru(UI Katmanı):* Javascriptte AJAX kullanımı nedir? Ne gibi faydaları vardır?**

AJAX, “Asynchronous JavaScript and XML” kelimelerinin baş harflerinden oluşan ve türkçe karşılığı “Eşzamansız JavaScript ve XML” olan, web sitelerinde hızlı ve dinamik bir şekilde web sayfaları oluşturmak için kullanılan, web sitelerinde sayfa yenilenmeden sunucu/veritabanı ile bağlantı kurulmasını ve veri alışverişi yapılmasına olanak sağlayan bir web proğramlama tekniğidir.

Google’da arama yapmak için arama kutusuna bir şeyler yazmaya başladığınızda, google aradığınız şeyi otomatik olarak tahmin eder ve bulmak istediğiniz veri ile alakalı terimleri anlık olarak karşınıza çıkarır. Bu süreç arka planda AJAX’ın çalışıyor olduğunun bir göstergesi. Web sayfası yeniden yüklenmeden sunucu ile GET yöntemi ile anlık olarak iletişim kuruluyor, sunucudan sonuçlar getiriliyor ve arama sayfasında gerekli bölgelere bu bilgiler yazdırılıyor.

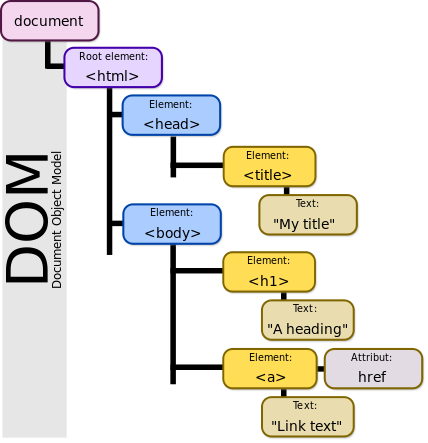
AJAX kullanımı denildiğinde akla ilk gelen yöntem jquery kütüphanesi oluyor. Tabii [**jquery ajax kullanımı**](https://muhammedyaman.com/jquery-ajax-post-kullanimi/) daha kolay bir yöntem ancak herhangi bir kütüphane kullanmadan da sade javascript ile AJAX kullanılabilir, GET veya POST yöntemleriyle veri alışverişi yapılabilir.

#### **Soru(UI Katmanı) : Jquery ve Ajax arasındaki başlıca farklar Nelerdir?**

* Jquery bir ekip tarafından geliştirilmiş açık kaynaklı javascript kodlama kütüphanesidir.
* Ajax JavaScript ve XMLHttpRequest kullanımı ile web sayfalarında etkileşimli uygulama yapmanızı sağlayan script dilidir.
* Ajax, web kodlamada bir uygulanış şekli, Jquery ise bir kütüphanedir.
* Ajax’ın amacı one-page(tek sayfa) üzerinde uygulama yazmak için yardımcı olmaktır.
* Ajax ile kodlama yaparak sayfayı yenilemeden yani client tarafından sunucuya arka planda istekler gönderip alabilirsiniz.
* Ajax genelde web 2.0 ı geliştirip 3.0 yapmış gibi kabul edilir.
* Jquery dünyada en çok bilinen javascript kütüphanesidir.
* Jqueryde yoğun bir şekilde animasyon kullanabilirsiniz.
* İkisi birbiriyle kıyaslanamaz.

## ***Soru(UI Katmanı):* DOM nedir? Nasıl oluşur? HTML rendering, HTML parsering nedir?**

Bir web sayfası yüklendiğinde, tarayıcı sayfanın **D**ocument **O**bject **M**odel'ini oluşturur. **HTML DOM** modeli bir Nesneler ağacı olarak **oluşturulur**:**DOM D**ocument **O**bject **M**odel kelimesinin kısaltılmasından gelir ve Türkçe karşılığı Belge Nesne Modelidir. DOM,**HTML, XML** gibi belgelerin diğer programlama dilleri veya script dilleriyle iletişim kurabilmesini sağlamak için geliştirilmiş bir arabirimdir.Programlama dillerinde sıkça karşımıza çıkan**Object Oriented**yaklaşımı da bu yapıya benzerdir. Object Oriented yaklaşımında kullandığımız nesneler yada sınıflar, DOM yapısına göre bir HTML sayfası içerisinde bulunan **etiketleri (tag)** kapsar.



***Soru(Servis Katmanı):* SOAP ile Rest servisler arasındaki farklar nelerdir?**

|  |
| --- |
| * SOAP XML veri tipini desteklerken REST istenen veri türüyle işlem yapabilir. JSON veri tipi ile XML’den çok daha düşük boyutlarla veri tutulabildiği için REST ile daha hızlı işlem yapılabilir. |
| * Ayrıca SOAP için WSDL ile tanımlama yapmak gerekirken REST için böyle bir zorunluluk yoktur. (WADL REST için kullanılan WSDL’e benzer bir yapıdır fakat kullanma zorunluluğu yoktur.) Bir dile ihtiyaç duymadan HTTP metodlarıyla tasarlanabildiği için REST’i kullanması ve tasarlaması daha kolaydır. |
| * SOAP için birçok geliştirme aracı mevcuttur, REST için geliştirme araçlarına ihtiyaç duyulmaz, tasarlaması kolaydır. |
| * SOAP; XML-Scheme kullanırken REST; URI-scheme kullanır yani metotlar için URI’ler tanımlanır. |
| * Her ikisi de HTTP protokolünü kullanırlar. Fakat REST için HTTP zorunluluğu varken SOAP; TCP, SMTP gibi başka protokollerle de çalışabilir. |
| * Test ve hata ayıklama aşaması REST için daha kolaydır. Çünkü HTTP hatalarını döndürür ve bunlar bir toola ihtiyaç duyulmadan görülebilir. SOAP için hata ayıklama araçları gerekebilir. |
| * REST basit HTTP GET metodunu kullandığı için cacheleme işlemi daha kolaydır. SOAP ile cacheleme yapabilmek için karmaşık XML requestleri yapılmalıdır. |
| * İkisi de HTTPS destekler, SOAP için WS-SECURITY adlı bir eklenti mevcuttur. |
| * Güvenlik açısından SOAP daha gelişmiştir çünkü hazır yapılar bulunmaktadır. |
| * Dokümantasyon bakımından SOAP daha gelişmiştir ve daha fazla kaynak bulunmaktadır. |

#### **Soru(Servis Katmanı): Web Api nedir?**

HTTP, üzerinden REST servisler geliştirmek için kullanılan bir framework’tür.

#### **Soru(Servis Katmanı): Web Api ve Web Service arasındaki fark nedir?**

* Her web servisi bir Web Api dir, ancak Web Api ler web servisi değildir. Yani Web Api web servislerini kapsar.
* Web servisleri internete bağlı cihazlarda çalışan uygulamaların konuşmaları şeklinde özetleyebiliriz.
* Web Api ler için durum daha gelişmiştir.
* Web servisler internet üzerinden haberleşirler. Web Api lerin çalışma mantığı daha farklıdır. Bir uygulamanın apisini kendi uygulamanıza entegre ederek o uygulamadan faydalanabilirsiniz.
* Web servisler sadece üç hizmeti kullanabilirler SOAP, REST ve XML-RPC. Web api ler için herhangi bir kısıt yoktur.
* Yukarıdaki bilgiler temel düzeyde bir özettir. Detaylı açıklamaları aşağıda bulabilirsiniz.

##### **Soru(Servis Katmanı) :Başlıca Web Api Özellikleri**

* Http Get, Post, Put ve Delete metodlarıyla çalışabildiğinden CRUD işlemelrini destekler,
* Response’larda HttpStatusCode ve Accept Header parametreleri bulunur,
* Response’lar kullanıcının istediği türde MediaTypeFormatter tarafından formatlanabilir,
* OData desteği bulunmaktadır ve Query yazması oldukça kolaydır,
* Bir uygulama içerisinde veya IIS üzerinde host edilebilir,
* MVC’nin bazı özelliklerini taşır (routing, controllers, action results, filter, model binders)

##### **Web Servisler**

* Http protokolü ile XML üzerinden iletişimi gerçekleştiren yöntemdir.
* XML olduğu için farklı platformlar birbiriyle iletişim halindedir.
* Web servislerinde veriler XML olarak tutulur ve SOAP kullanılarak iletişim gerçekleştir.
* Örnek bir web servis olarak sitenize eklediğiniz güncel Hava Durumunu düşünebilirsiniz.

### **Soru(Servis Katmanı) : JSON (JavaScript Object Notation) Nedir, Nasıl ve Nerede Kullanılır?**

Json, Javascript uygulamaları için oluşturulmuş bir veri formatıdır. Javascript Object Notation’ın kısaltmasıdır. Json’ın çıkış amacı veri transferlerinde verilerin XML’den daha az yer kaplamasını sağlamaktır. Şu an sadece Javascript uygulamalarında değil, yazılım geliştirmede kullanılan bir çok teknolojide Json formatındaki veriler tercih edilmektedir.

**Soru(Veritabanı Katmanı):** **SQL'de bulunan Index, Trigger ve Stored Procedure kavramları nelerdir?**

Index; İlişkili tablonun indexleme alanı olarak kullanılan kolondaki verilere göre sıralanmış biçimde işleme konulmasını sağlar. Sorguların geç cevap vermesi durumunda kullanılır. Trigger; Tetikleyici anlamındadır. Veritabanında bir işlem yapıldığında başka bir işleminde otomatik olarak yapılması için kullanılır. Stored Procedure; Belli bir işi yapan kod parçasıdır. İlk derlemeden sonra bir daha derlenmeye ihtiyaç duymaz.

***Soru(Veritabanı Katmanı):* Clustered vs Non-Clustered arasındaki farklar nelerdir?**

SQL Server açısından index kullanımının en önemli amacı, istenen bilginin daha az veri okunarak daha kısa zamanda getirilmesini sağlamaktır.

Sql server'da 2 farkı indexleme yöntemi vardır.

Clustered Index (kümelenmiş, sıralanmış)

Non-Clustered Index (kümelenmemiş, sıralanmamış)

|  |  |
| --- | --- |
| Clustered Index | Non Clustered Index |
| Veriler(tablo) diskte clustered index’e göre mantıksal olarak sıralı bir şekilde tutulur. | Non clustered index koyulan kolonlar tablodan bağımsız olarak diskte ayrı bir şekilde tutulur. Bu yüzden ekstra yer kaplarlar ve tabloda çok fazla non clustered index varsa, her insert,update ve delete işlemi tablonun haricinde bu tablodaki bütün non clustered index’lere de uygulanacağı için insert,update ve delete performansı yavaşlayacaktır. Bu yüzden çok fazla non clustered index oluşturmak her zaman iyi bir şey değildir. |
| Arama yaparken Leaf Level’e ulaştığımızda verinin kendisine ulaşırız. Leaf Level için -> “SQL Server’da Index Kavramı ve Performansa Etkisi” | Non Clustered Index’te arama yaparken Leaf Level’e ulaştığımızda verinin kendisi yerine Row Locator bulunur. |
|  |
| Eğer tablo heap ise(tabloda clustered index yoksa) row locator olarak pointer /Row ID(RID) bulunur. Arama yaparken Leaf Level’a gelindiğinde row id kullanılarak ilgili kayda ulaşılır. Bu işlem RID Lookup olarak geçer. İyi bir şey değildir. Tablonuza mutlaka bir clustered index koyun. |
|  |
| Eğer tablo heap değilse(clustered index varsa) row locator olarak Clustered Index Key vardır. Arama yaparken Leaf Level’a gelindiğinde clustered index key kullanılarak ilgili kayda ulaşılır. Bu işlem Key Lookup olarak geçer. |
| bir tabloda bir tane olabilir(Çünkü bütün tablo clustered index’e göre diskte mantıksal olarak sıralı tutuluyor. Sadece bir tane clustered index’e göre sıralı olarak tutulabilir.) | Tablo non clustered index’e göre sıralı olmadığı için, non clustered index tablonun dışından ekstra bir alan kullanılarak oluşturulduğu için bir tabloda birden fazla sayıda olabilir. |
| Included Kolon olmaz. Çünkü Leaf Level’da verinin kendisi tutulur. | [Included Kolon olabilir. “SQL Server’da Index Kavramı ve Performansa Etkisi](http://www.veritabani.gen.tr/2016/11/17/sql-serverda-index-kavrami-performansa-etkisi/) |
| ” isimli makalemde indexler’de included kolon’u detaylı olarak örnekli bir şekilde açıkladım. |
| Unique te olabilir non unique’te olabilir. Non unique olursa(ki hiç tavsiye etmem) clustered index’e sahip tabloda oluşturacağımız non clustered index’lerin leaf level’indeki row locator’da ki clustered index key’leri unique hale getirmek için uniqueifier isminde 4 byte’lık bir belirleyici koyar. Bu şekilde non clustered index kullanılarak yapılan bir aramada ihtiyacı olan veriyi bulmasını sağlar. Extra maliyet, extra büyüklük. | Unique te olabilir non unique’te olabilir. Unique olursa aramalar daha hızlı sonuç döner. Örneğin tablonuzda 100 kayıt var. Index koyacağınız kolondaki 100 kaydın 100’üde birbirinden farklı. İçinden bir tane kaydı aradığınızda direk o kaydı bulabilirsiniz. Bu işleme index seek denir. Ama kolondaki 100 kaydın 70’inin değeri birbiri ile aynıysa bu kolona koyduğunuz index’in hepsi taranır ve ihtiyacınız olan kayıtlar bulunur. Bu işleme de index scan denir. Kötü gibi görünsede bazı durumlarda tüm tablonun taranmasından daha iyidir.(table scan) Ama bu tip unique’liği sağlamayan kolonları filtre olarak kullanmak yerine unique’liğe yakın kolonları ya da direk unique kolonları filtreleyerek sorgu yazılması daha performanslı olacaktır. |

#### **Soru(Veritabanı Katmanı): Sql Stored Procedure ve Function Nedir? Arasındaki farklar nelerdir?**

Database server’ında tutulan ve ilk derlenmeden sonra bir daha derlenmeye ihtiyaç duyulmayan SQL ifadeleridir. DB Server’ında tutulduğu için çok hızlı çalışırlar, hızlı çalışmasının sebebi ise bir kez derlendikten sonra bir daha derlenmeye ihtiyaç duymamalarıdır. Halbuki uygulama katmanında yazmış olduğumuz SQL kodu her çağırmada tekrar tekrar derlenmek zorundadır.

### **Soru(Veritabanı Katmanı):DDL, DML, DCL, TCL Nedir? DDL (Data Definition Language) Nedir?**

Veri Tanımlama Dili (DDL) deyimleri tabloları, veritabanı yapısı veya şema tanımlamak için kullanılır.

Bunlardan bazıları;

**CREATE** – Veritabanındaki nesneleri oluşturmak için kullanılır.  
**ALTER** – Veritabanı nesnelerinin yapısını değiştirmek için kullanılır.  
**DROP** – Veritabanındaki nesneleri silmek ya da başka bir ifadeyle ilgili nesneleri bütünüyle kaldırmak için kullanılır.  
**TRUNCATE** – Kayıtlar için ayrılan tüm boşluklar dahil, bir tablodaki tüm kayıtları kaldırılır  
**COMMENT** – Veri sözlüğüne yorum eklemek için kullanılır.  
**RENAME** – Bir nesneyi yeniden adlandırmak için kullanılır.

#### **DML (Data Manipulation Language) Nedir?**

Veri İşleme Dili (DML) deyimleri tablo ya da şema nesneleri içindeki verileri yönetmek için kullanılır.

##### **Bunlardan bazıları;**

**SELECT** – Veritabanından kayıt çekmek için kullanılır.  
**INSERT** – Tabloya kayıt ya da kayıtları eklemek için kullanılır.  
**UPDATE** – Tablodaki kayıt ya da kayıtları güncellemek için kullanılır.  
**DELETE** – Tablodan kayıt ya da kayıtları silmek için kullanılır (Veriler silinse de ancak kapladığı alan kalır)  
**MERGE** – UPSERT işlemi (ekleme veya güncelleme), başka bir ifadeyle birleştirme yapar.  
**CALL** – PL/SQL veya Java alt programını çalıştırır.  
**EXPLAIN PLAN** – Verilere erişim yolunun detaylarını açıklamak için kullanılır.  
**LOCK TABLE** – Kontrolü eş zamanlılığı sağlamak için kullanılır.

#### **DCL (Data Control Language) Nedir?**

Veri Kontrol Dili (DCL) deyimleri yetkilendirme ya da ayrıcalıkları belirlemek için kullanılır.

##### **Bunlardan bazıları;**

**GRANT** – Belirli bir kullanıcı ya da gruba veritabanının belirtilen nesnelerine erişim ayrıcalıklarını verir.  
**REVOKE** – GRANT komutu ile verilen ayrıcalıkların bir kısmını ya da tümünü geri almak için kullanılır.

#### **TCL (Transaction Control) Nedir?**

İşlem Kontrol (TCL) deyimleri DML ifadeleri tarafından yapılan değişiklikleri yönetmek için kullanılır.

##### **Bunlardan bazıları;**

**COMMIT** – Yapılanları kalıcı hale getirir. İşin tamamlanmasını sağlar.  
**SAVEPOINT** – Daha sonra geri dönülecek bir dönüş noktası belirler.  
**ROLLBACK** – Son COMMIT’e kadar olan yeri geri alır.

### **Soru(Veritabanı Katmanı): SQL’de deadlock nedir? Nasıl Çözülebilir?**

Veritabanına gönderilen iki ayrı işlemin birbiri ile bağlantılı olup ikisininde görevinin yapma koşulunun diğer işlemin sonucuna bağlı olması, diğer işlemin bitmesini beklemesi gerektiği yani sonsuz bekleyişe girilen ölümcül kilitlenme durumudur. Bu işlemler önceden tespit edilip deadlock durumunda daha önemli görülen işleme öncelik atanarak çözümlenebilir.  
**Soru(Veritabanı Katmanı): TSQL de GROUP BY,HAVING ve ORDER BY saklı komutları ne işe yarar?**

**Soru(Veritabanı Katmanı): İnner, Left,Right, Outer Join nedir ?**

***SON SORU:* Bakım sorumluluğunuzda olan uygulamaya ait bir sayfada yavaşlama ve buna bağlı hata yaşandığı bilgisi verildi. Uygulamanızın büyük bir organizasyonda kullandığını düşünürsek sizin tarafınız dan alınabilecek aksiyonlar nelerdir?**

Katmanlı mimari olduğu ipuçları görüşürken verilmeli. Katmanlarda tek tek performans kontrolleri yapılmalı. Soru detaylandırılmalı.