Frameworkler: Java ve Kotlin için özellikle Android geliştirme tarafında çok kullanılan dependency injection kütüphanesi Dagger. Diğer popüler dependency injection kütüphanelerinin bazıları şunlardır: Guice, CDI, Weld, HK2, PicoContainer…

@SpringBootApplication: @SpringBootApplication SpringBootApplication anotasyonunun kapsadığı anotasyonlar

@Target(ElementType.TYPE) – anotasyonu bir anotasyonun nerelere uygulanabileceğini belirler.

@Retention(RetentionPolicy.RUNTIME) – anotasyonu bir anotasyonun ne kadar tutulacağını belirler. RetentionPolicy.RUNTIME ise SpringBootApplication anotasyonunun kodumuz JMV'de çalıştığı süre boyunca tutulacağını gösteriyor.

@Documented – bir anotasyon bu anotasyonla işaretlenirse o anotasyon kullanıldığı yerlerde, Javadoc aracıyla oluşturulmuş dökümanlarda gösterilir.

@Inherited – anotayonu bir anotasyona uygulandığında o anotasyonun alt sınıflarada uygulanmasını sağlar.

@SpringBootConfiguration – bu anotasyon bir classa uygulandığında o classın Spring konfigürasyonları içerdiğini belirtir ve Spring bu konfigürasyonları otomatik olarak bulur.

@EnableAutoConfiguration – anotsayonu Spring projesinde yazdığımız kodlara göre ihtiyaç olacak ayarların otomatik olarak yapılmasına izin verir.

@ComponentScan – anotasyonu Spring componentlerinin nasıl taranacağını, hangi bileşenlerin Spring projesine dahil olacağını ayarlamaya yarar.

@Primary ve @Qualifier: @Primary , Aynı türden birden fazla bean varsa primary anotasyonlu olan varsayılan olarak gelecektir. @Component anotasyonu olan classlarda ve @Bean anotasyonlu methodlarda kullanılabilir.

@Qualifier, @Autowire ile dependency injection yaparken birden fazla uygun aday varsa hangisinin tercih edileceğini belirtmek için kullanılır. Primary anotasyonundan farkı spesifik bir implementasyonu çağırmaya yarar. Qualifier anotasyonu varsa Primary anotasyonu yok sayılır.

Convention over configuration: Yazılımcıların vermesi gereken kararların sayısını azaltmaya çalışan, aynı kodların tekrar tekrar yazılmasını varsayımlara dayanarak azaltan yazılım modelidir.

Aspect Oriented: Kodun modülerliğini arttırmaya yarayan yazılım yaklaşımıdır. Kodu ufak parçalara ayrılmasıdır. Aynı kodun farklı yerlerde kullanılabilmesini sağlar. Avantajları: -Modülerite salar. -Tekrar eden kodlar yazmayı engeller. -Bakımı kolaydır sürdürülebilirdir. Dezavantajları: -Debug yapmak biraz daha zordur.

SOLID: Single Responsibility: Her bir class yanlızca bir işi yapmalı. Tam yapmalı. Open-Closed Principle: Kod genişletilebilir ama değiştirilemez. Liskov Substitution Principle: Subclasslar Superclasslarının özelliklerine ve davranışlarına sahip olmalı, yerine kullanılabilmeli. Interface Segregation Principle: Classlar ihtiyacı olmayan metodları içeren interfaceleri implement etmemelidir. Dependency Inversion Principle: Üst seviye classlar alt seviye classlara bağlı olmamalıdır. Aradaki ilişki abstractionla sağlanmalıdır.

Swagger: Swagger endpointlerimizi otomatik olarak dökümente etmemize yarayan bir kütüphanedir.

Richardson Maturity Model: APImizi REST kurallarına göre derecelendirmenin bir yoludur. APImiz bu kısıtlamalara ne kadar iyi uyarsa, puanı o kadar yüksek olur. Richardson Maturity Modelin 0-3 arasında 4 seviyesi vardır ve 3. Seviye RESTful API'yi belirtir. Level 0: Swamp of POX – Servisimize tek bir POST methodu üzerinden erişiyorsak

Level 1: Resources: Kaynakların birbirinden ayrıldığı seviyedir. Spesifik bir kaynağa URI değiştirerek ulaşabiliyorsak API'miz bu seviyenin üstündedir.

Level 2: HTTP verbs: GET, POST, DELETE gibi http methodlarının kullanıldığı karşılığında farklı anlamlara gelen http status codeların alındığı seviyedir.

Level 3: Hypermedia controls İki bölümde inceleyebiliri: Content Negotiation: Attığımız sorgunun cevabının gösterilme şeklini kontrol edebilmektir. Json, CSV, XML gibi. HATEOAS: Hypermedia as transfer engine of aplication state

URL, URI, URN: Uniform Resource Identifier: internet üzerindeki bir kaynağın tanımlamak için kullanılır. Hem isim hem de yer belirtebilir. http://www.example.com/home#about

Uniform Resource Name: URI'nin özelleştirilmiş bir halidir. Kaynağın adresini veya ulaşılabilir olmadığını belirtmez. Kaynağın benzersiz bir adla tanımlanmasıdır. urn:isbn:0971450556

Uniform Resource Locator: URI'nin bir diğer özelleştirilmiş halidir. Kaynağın internetteki yerini ve ne şekilde ulaşılabileceğini belirtir. (http, ftp gibi) http://www.example.com/home

Idempotency: Bir request birden fazla kez atıldığında her seferinde aynı etkiyi yapıyorsa idempotentdir. GET, HEAD, PUT, DELETE HTTP metodları idempotentdir.

RFC (Request For Comment): RFC internetle ilgili standartların ve teknik yayınların yapıldığı bir yayın serisidir. Yorum isteği olarak adlandırılmasının sebebi yapılan yayınların resmi bir kural olmadığı herkesin konu hakkında yorum yapabileceğini belirtmek içindir. HTTP 1999 yılında rfc2616 da yayınlanmıştır. URL, URI gibi kavramları. HTTP'nin amacı, kuralları, methodları, status codelarını içerir.