

## KOTLİN ÖDEV 6 – BEYZA PARLAK

### RETROFİT

Retrofit, Android uygulamalarında HTTP tabanlı API isteklerini yönetmek için kullanılan bir kütüphanedir. Bu kütüphane, RESTful API'lerle iletişim kurmayı kolaylaştırır. Kod tekrarını azaltır, hata ayıklamayı kolaylaştırır ve uygulamanızın performansını artırır. API'ler, sunucudan veri almak veya sunucuya veri göndermek için kullanılır. Retrofit, HTTP isteklerini gerçekleştirmek için güçlü bir araçtır.

Retrofit' in başlıca avantajları:

- Kolay Kullanımı: Retrofit, RESTful API isteklerini oluşturmayı, göndermeyi ve yanıtları işlemeyi oldukça kolaylaştırır. Böylece daha az kod yazarız ve daha az hata yaparız.
- Modülerlik: Retrofit, uygulamalarımızda kullanmak için modüler bir yapı sağlar. Böylece uygulamamızın gelişimi ve bakımı kolaylaşır.
- Tip Güvenliği: Retrofit, Java veya Kotlin gibi tip güvenli dillerle uyumludur. API isteklerinin ve yanıtlarının türleri belirtilir. Böylece derleme zamanında hataları tespit etmek mümkündür.
- Entegrasyon Kolaylığı: Retrofit, diğer popüler kütüphaneler (GSON, MOSHI gibi JSON kütüphaneleri) ile kolayca entegre olabilir. Böylece geliştiricilerin veri dönüştürme işlemleri kolaylaşır.

### Retrofit HTTP Metodları

#### @GET

GET, belirli bir kaynaktan veri istemek için kullanılan bir metoddur. GET istekleri, hassas veriler ile uğraşılırken asla kullanılmamalıdır.

Kullanım Alanları: Kullanıcının profil bilgilerinin çekilmesi, medium yazısının içeriğinin çekilmesi ya da e-ticaret sitesindeki ürünlerin listelenmesi olabilir.



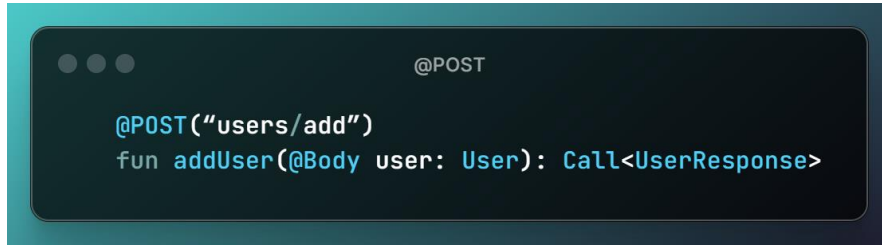
```
@GET("/users/{userId}")
fun getUser(@Path("userId")userId: String) : Call<UserResponse>
```

image 1

#### @POST

POST, bir kaynağı oluşturmak ya da bir kaynaktaki veriyi güncellenmek amacıyla sunucuya veri göndermek için kullanılan bir metoddur. Gönderilen veriler genellikle form verileri ya da JSON verilerinden oluşur. POST, GET metoduna kıyasla daha güvenlidir çünkü gönderilen veriler, URL' nin bir parçası değildir.

Kullanım Alanları: Yeni kullanıcı oluşturma, yeni veri girişi, e-ticaret sitesine yeni bir ürünün eklenmesi olabilir.

A screenshot of a code editor with a dark background. At the top, there are three small circles and the text '@POST'. Below this, the code is: 

```
@POST("users/add")
fun addUser(@Body user: User): Call<UserResponse>
```

image 2

## @PUT

PUT, var olan kaynağı güncelleme işlemlerinde kullanılan bir metoddur. Genellikle köklü değişiklikler yapılır.

Kullanım Alanları: Kullanıcı verilerini güncellenmesi, bir belgenin içeriği güncellenmesi ya da e-ticaret sitesindeki bir ürünün isminin güncellenmesi olabilir.

A screenshot of a code editor with a dark background. At the top, there are three small circles and the text '@PUT'. Below this, the code is: 

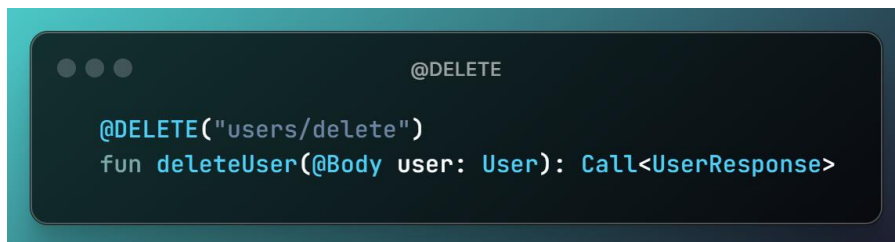
```
@PUT("users/update")
fun updateUser(@Body user: User): Call<UserResponse>
```

image 3

## @DELETE

DELETE, var olan kaynağı silme işlemlerinde kullanılan bir metoddur. Sunucu, kaynağı kalıcı olarak siler.

Kullanım Alanları: Kullanıcının hesabını silmesi, bir belgenin silinmesi, e-ticaret sitesi veri tabanından bir ürünün silinmesi olabilir.

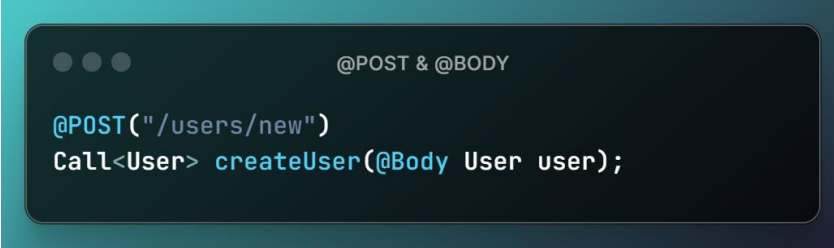
A screenshot of a code editor with a dark background. At the top, there are three small circles and the text '@DELETE'. Below this, the code is: 

```
@DELETE("users/delete")
fun deleteUser(@Body user: User): Call<UserResponse>
```

image 4

## Retrofit İçerisinde En Çok Kullanılan Annotations

### @POST & @BODY



```

@POST & @BODY

@POST("/users/new")
Call<User> createUser(@Body User user);

```

image 5

'/users/new' adresine bir HTTP POST isteği gönderiliyor ve yeni bir kullanıcı oluşturuluyor. @Body annotation, isteğin gövdesinde (Body'de) gönderilen kullanıcı nesnesini belirtir.

### @PUT & @PATH



```

@PUT & @PATH

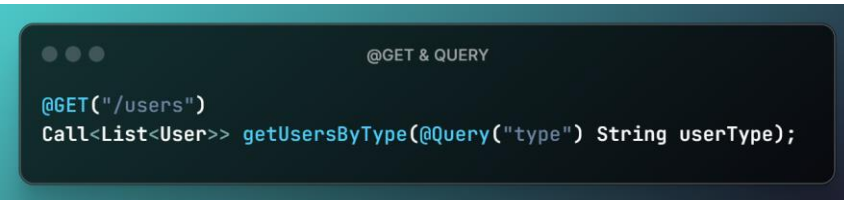
@PUT("/users/{id}")
Call<User> updateUser(@Path("id") int userId, @Body User user);

```

image 6

'/users/{id}' adresine bir HTTP PUT isteği gönderiliyor ve belirli bir kullanıcıyı güncelleniyor. @Path annotation, URL'de {id} yerine geçecek olan kullanıcı kimliğini belirtir.

### @GET & @QUERY



```

@GET & @QUERY


@GET("/users")
Call<List<User>> getUsersByType(@Query("type") String userType);

```

image 7

'/users' adresine bir HTTP GET isteği gönderiliyor ve type @query parametresi ile filtrelenmiş kullanıcıları alıyor.

### @DELETE



```

@DELETE

@DELETE("/users/{id}")
Call<Void> deleteUser(@Path("id") int userId);

```

image 8

'/users/{id}' adresine bir HTTP DELETE isteği gönderiliyor. Bu, belirli bir kullanıcıyı silmek için kullanılır. @Path annotation, silinecek kullanıcıyı belirtir.

## @FIELD & @FORMURLENCODED

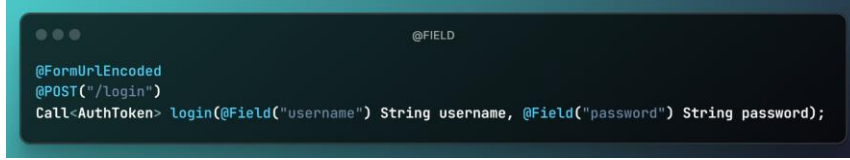


image 9

'/login' adresine bir HTTP POST isteği gönderiliyor. @Field annotation, @FormUrlEncoded veri göndermek için kullanılır. Kullanıcı adı ve şifre gibi alanlar formda gönderilir.

@FormUrlEncoded annotations, bir HTTP POST isteğinin form-urlencoded biçiminde gönderileceğini belirtir. Form-urlencoded, verilerin anahtar-değer çiftleri olarak gönderildiği ve URL kodlaması kullanılarak kodlandığı bir veri gönderme biçimidir.

## En Çok Kullanılan HTTP Status Kodları

HTTP (Hyper Text Transfer Protocol) harflerden oluşmuş bir kısaltmadır [1]. Türkçe çevirisi "Üstün Metin Transferi Protokolü" anlamına gelmektedir [1]. HTTP durum kodları, bir HTTP isteğinin sonucunu belirten üç haneli sayısal kodlardır ve isteğin başarıyla tamamlandığını, yönlendirildiğini, hatalı olduğunu veya başka bir durumda olduğunu belirtmek için kullanılır.

HTTP durum kodlarının amacı, bir HTTP isteğinin sonucunu belirlemek ve iletişim sırasında oluşan durumları anlamak için bir standart sağlamaktır. Bu kodlar, isteğin işlendiği şekli hakkında bilgi verir. Bu kodlar, istemciler ve sunucular arasındaki iletişimin daha tutarlı, güvenilir ve anlaşılır olmasını sağlar.



image 10 [2]

## 100 Continue

Sunucunun, istemcinin bir isteği almaya ve işlemeye hazır olduğunu bildirdiği ön bilgi kodudur.

## 200 OK

Sunucunun istemcinin isteğini başarıyla işlediğini ve istemcinin talebinin karşılandığını belirtir. Bu, isteğin başarılı bir şekilde tamamlandığı anlamına gelir.



image 11 [3]

## 201 Created

Sunucunun bir HTTP POST isteğini başarıyla işlediğini ve yeni bir kaynak oluşturduğunu anlamına gelir. Bu kod, genellikle yeni bir kayıt veya belge oluşturulduğunda kullanılır.

## 202 Accepted

Sunucunun isteği kabul ettiğini ancak henüz işleme koymadığını anlamına gelir. İstek daha sonra işlenecek veya başka bir iş parçasına yönlendirilecek demektir.

## 302 Found

Sunucunun istemcinin yaptığı bir talebi başka bir URL'ye yönlendirdiğini anlamına gelir. Bu, istemcinin otomatik olarak yeni bir konuma yönlendirilmesi gerektiğini ifade eder.

## 304 Not Modified

Sunucunun, istemcinin bir kaynağın güncellendiği son tarihi kontrol etmesinin ardından, istemcinin isteğinin karşılık geldiği kaynağın güncelliğinin değişmediğini anlamına gelir.

## 400 Bad Request

Sunucunun bir isteği anlamadığını veya işleyemediğini belirtir, çünkü istemcinin gönderdiği istek formatı geçersiz veya anlaşılmazdır [6].

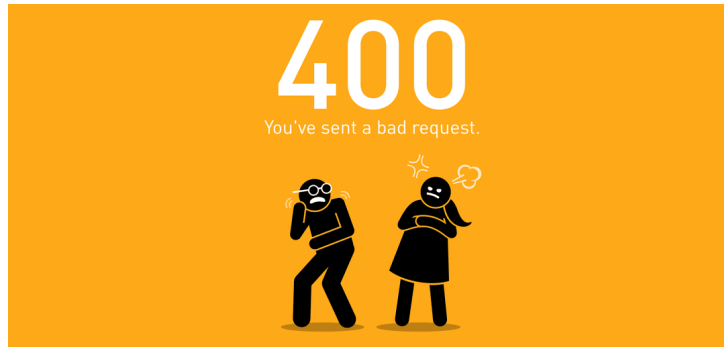


image 12 [4]

#### 401 Unauthorized

Sunucunun istemcinin bir kaynağa erişim izni olmadığını anlamına gelir. Bu genellikle kullanıcının kimlik doğrulaması gerektiği durumlarda ortaya çıkar.

#### 403 Forbidden

Sunucunun istemcinin bir kaynağa erişimini reddettiğini anlamına gelir. Bu, istemcinin kaynağa erişim yetkisinin olmadığı veya kaynağın erişimine izin verilmediği durumlarda ortaya çıkar.

#### 404 Not Found

Sunucunun istemcinin istediği kaynağı bulamadığını anlamına gelir. Bu durumda, istemcinin isteği karşılayacak bir kaynak sunucuda bulunmamaktadır.

#### 500 Internal Server Error

Sunucunun isteği yerine getirmesini engelleyen, beklenmeyen bir durumla karşılaştığı anlamına gelir [7].

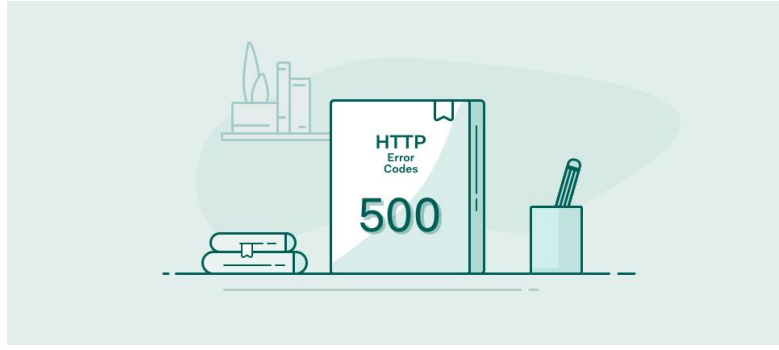


image 13 [5]

#### 502 Bad Gateway

Sunucunun bir ara sunucu hatası aldığını ve bu nedenle istemcinin isteğini yerine getiremediğini belirtir. Bu durum genellikle, sunucunun arkasında başka bir sunucuyla yapılan bir iletişim hatası veya hizmet dışı kalması durumunda ortaya çıkar.

#### 504 Gateway Timeout

Sunucunun zaman aşımına uğradığını ve istemcinin beklenen yanıtı zamanında alamadığını belirtir. Bu durum genellikle, sunucunun istemcinin isteğini yerine getirmek için arka planda başka bir sunucu veya servis ile iletişim kurarken zaman aşımına uğraması durumunda ortaya çıkar.

#### 505 HTTP Version Not Supported

Sunucunun istemcinin kullandığı HTTP protokolünün desteklenmediğini belirtir. Bu durumda, sunucu istemcinin kullandığı HTTP protokol sürümünü tanımaz veya desteklemez. Dolayısıyla istemciyle de iletişim kuramaz.

## KAYNAKLAR

1. <https://blog.kobisi.com/en-cok-karsilasilan-durum-ve-hata-kodlari/>
2. <https://www.ionos.com/digitalguide/hosting/technical-matters/the-most-important-http-status-codes-at-a-glance/>
3. <https://sitechecker.pro/what-is-200-status-code/>
4. <https://www.elegantthemes.com/blog/wordpress/how-to-fix-the-http-error-400-for-wordpress-websites>
5. <https://www.elegantthemes.com/blog/wordpress/how-to-fix-the-http-error-400-for-wordpress-websites>
6. <https://www.hosting.com.tr/bilgi-bankasi/400-durum-kodu-bad-request/>
7. <https://tls.tc/lh0lh>