HTTP METOTLARI

1- GET:

- Bu metodu sunucudan veri almak için kullanırız.
- Hızlıdır fakat güvenli değildir.
- Get metodu ile bir istekte bulunduğumuzda bu isteği tarayıcı üzerinde açık bir şekilde görüntüleyebiliriz, bu durum büyük güvenlik sorunlarına yol açmaktadır.
- GET requestleri yalnızca veri almak için kullanılır ve sunucudaki veri üzerinde değişiklik yapmaz. Örneğin, bir API'dan bir kullanıcının bilgilerini almak için GET kullanılabilir.

```
interface ApiService {
    @GET("/users/{id}")
    suspend fun getUser(@Path("id") userId: Int): Response<User>
}

// Kullanıcı bilgilerini almak için
val apiService = retrofit.create(ApiService::class.java)
val response = apiService.getUser(1)
if (response.isSuccessful) {
    val user = response.body()
    println("Kullanıcı Adı: ${user?.name}")
} else {
    println("Hata: ${response.message()}")
}
```

2- POST:

- Sunucudan veri almak için kullanılır ancak mevcut olan bir veriyi güncellemek için de kullanılır.

- Doğrudan sayfaya veri gönderilir ve veriler adres çubuğunda görünmez. Bu yüzden daha güvenilir bir yöntemdir. İstekler tarayıcı üzerinden gizli bir şekilde gönderilir.

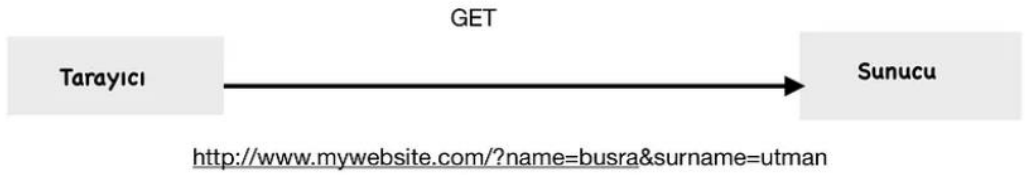
- Bu metotla istek parametreleri hem URL içinde hem de mesaj gövdesinde gönderebiliriz.

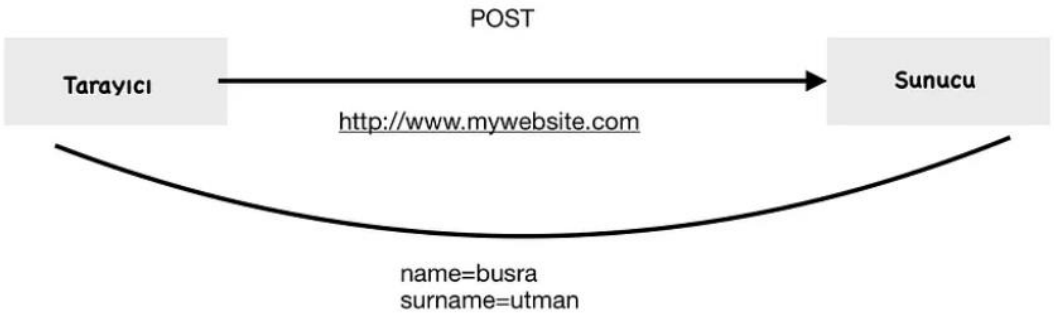
```
interface ApiService {
    @POST("/users")
    suspend fun createUser(@Body user: User): Response<Void>
}

// Yeni kullanıcı oluşturmak için
val newUser = User(name = "Ahmet", email = "ahmet@example.com")
val response = apiService.createUser(newUser)
if (response.isSuccessful) {
    println("Yeni kullanıcı oluşturuldu")
} else {
    println("Hata: ${response.message()}")
}
```

GET-POST ARASINDAKİ FARKLILIKLAR

- Verileri sunucuya göndermek için GET ve POST yöntemi kullanılır ve aralarındaki temel fark, GET yönteminin verileri formun işlem özelliğinde tanımlanan URL'ye eklemesidir. Tersine, POST yöntemi verileri istenen kuruluşa ekler. GET yönteminin kullanılması, hassas bilgilerin formda doldurulması gerektiğinde uygun değildir. POST yöntemi, kullanıcının şifreleri veya diğer gizli bilgileri doldurması gerektiğinde yararlıdır.





3- PUT

- PUT metodu ile sunuculara veri (genellikle de dosya) atılmak için kullanılır.
- Veri güncellemek için kullanılır. PUT ‘un POST’dan farkı idempotent olmasıdır. Yani bir request birden fazla kez tekrarlansa da sonucunun aynı olmasıdır.
- PUT requestleri genellikle bir kaynağın tüm bilgilerini güncellemek için kullanılır. Örneğin, bir kullanıcının profil bilgilerini güncellemek için PUT kullanılabilir.

```
interface ApiService {
    @PUT("/users/{id}")
    suspend fun updateUser(@Path("id") userId: Int, @Body user: User): Response<Voi
}

// Kullanıcı adını güncellemek için
val updatedUser = User(name = "Mehmet", email = "mehmet@example.com")
val response = apiService.updateUser(1, updatedUser)
if (response.isSuccessful) {
    println("Kullanıcı güncellendi")
} else {
    println("Hata: ${response.message()}")
}
```

4- PATCH

- Verinin sadece bir parçasını güncellemek için kullanılır.

5- HEAD

- GET metoduyla benzer işleve sahiptir ancak geri dönen yanıtta mesaj gövdesi bulunmaz (yani başlıklar ve içerikleri GET metoduyla aynıdır.)
- Bu nedenle GET mesajı gönderilmeden önce bir kaynağın var olup olmadığını kontrol etmek için kullanılabilir.

6- DELETE

- Bu metot ile sunucu üzerinde bir veriyi silebilirsiniz. Bu metot ile yaptığınız bir istekte, tanımlanmış olan veri id değerini gönderip, o id değerine karşılık gelen veriyi sunucu üzerinden silebilirsiniz.
- DELETE requestleri genellikle bir kaynağın verilerini kalıcı olarak silmek için kullanılır. Örneğin, bir kullanıcının hesabını silmek için DELETE kullanılabilir.

```
interface ApiService {
    @DELETE("/users/{id}")
    suspend fun deleteUser(@Path("id") userId: Int): Response<Void>
}

// Kullanıcıyı silmek için
val response = apiService.deleteUser(1)
if (response.isSuccessful) {
    println("Kullanıcı silindi")
} else {
    println("Hata: ${response.message()}")
}
```

HTTP Status Kodları

1xx: Sunucunun, tarayıcının isteğini aldığını ve işlemin başladığını belirten bilgi durum kodları.

2xx: Sunucunun, tarayıcının isteğini alıp anladığını ve isteğin başarılı olduğunu belirten durum kodları.

3xx: Talep edilen kaynağın başka bir konuma taşındığını ve yönlendirileceğini belirten durum kodları.

4xx: Talebin yerine getirilemediğini ve talep edilen web sayfası veya web sitesinin mevcut olmadığını belirten durum kodları.

5xx: Sunucunun, tarayıcının isteğini başarıyla aldığını ancak sunucu tarafındaki sorunlar nedeniyle isteğin yerine getirilemediğini belirten durum kodları.

200 Durum Kodu (Tamam)

Özetle ideal durum kodu budur. Bir web sayfası sorunsuz bir şekilde yüklenirse sunucu, tarayıcıya 200 durum koduyla yanıt verir. Sunucu tarayıcıya 200 durum koduyla yanıt veriyorsa ziyaretçi ve web sitesi için her şeyin olması gerektiği gibi olduğunu söyleyebiliriz.

201 (Created)

Yeni bir kaynak başarıyla oluşturuldu.

204 (No Content)

İstek başarılı olsa da, içerik gönderilmedi.

301 Durum Kodu (Kalıcı Olarak Taşındı)

Bu, bir web sayfası kalıcı olarak başka bir web sayfasına taşındığında ziyaretçiyi otomatik olarak yönlendiren bir durum kodudur. 301 durum kodu kullanarak web sayfalarına yönlendirme yaparsanız ve bu sayfalar içerik açısından oldukça benzerse güç kaybı en aza indirilebilir. Bu nedenle web sitesi geçişi gibi işlemler için önerilen en önemli durum kodlarından biridir.

302 Durum Kodu (Bulundu)

Bu durum kodu, bir web sayfasının geçici olarak başka bir web sayfasına taşındığını gösterir. İstenilen sayfanın yeniden etkinleştirileceği belirli durumlarda kullanılması nedeniyle 301 durum kodundan farklılık gösterir. Bu gibi durumlar, web sayfasının test ediliyor olması, bakımda olması veya belirli bir ürünün, eğer bir e-ticaret sitesi ise, geçici olarak stokta kalmamış olması olabilir. Ancak kullanıcılar 301 yönlendirmesi ile 302 yönlendirmesi arasındaki farkı anlayamayacaktır. Web sayfasına erişen kullanıcılar otomatik olarak diğer sayfaya yönlendirilecektir.

400 (Bad Request)

İstek hatalı olduğu için işlenemedi.

401 (Unauthorized)

Kimlik doğrulaması gerekiyor.

403 Durum Kodu (Yasak)

Kullanıcının sunucuya gönderdiği isteğe yanıt olarak gelen bu durum kodu, kullanıcının istenen web sayfasına erişmesine izin verilmediğini veya web sayfasının yasaklandığını belirtir.

404 Durum Kodu (Bulunamadı)

Bu durum kodu istenen web sayfasının sunucuda bulunamadığını gösterir. Web sayfası silinmiş veya URL'si değiştirilmiş olabilir. Ancak 404 durum kodu bu durumun geçici mi yoksa kalıcı mı olduğunu ortaya koymaz. Kullanıcılar talep ettikleri bir web sayfasında 404 durum koduyla karşılaştıklarında siteden ayrılarak farklı sitelere yönelmektedirler. Bu, özellikle çok fazla trafik alıyorsa veya URL'si kullanıcılar tarafından iyi biliniyorsa, web sitesini olumsuz yönde etkileyecektir.