

# 27.10.2025 Ders Notları

---

## Meta Android Developer Professional Certificate - HTTP, REST API & Güvenlik Notları

---

**Tarih:** 27.10.2025

**İçerdiği Konular:** HTTP/HTTPS temelleri, HTTP metodları, request/response yapıları, status kodları, REST API temelleri, REST kısıtlamaları, kaynak yönetimi, endpoint tasarımı, API test araçları, API geliştirme best practicesi, API güvenliği, erişim kontrolü ve Kotlin'de Ktor kullanımı.

---

### HTTP & HTTPS Refresh

HTTP istemci-sunucu arasında veri transferi için kullanılıyor. **HTTPS** ise bunun şifrelenmiş hali - güvenlik gerektiren yerlerde kesinlikle kullanılmalı (kredi kartı bilgileri gibi).

HTTPS'te:

- İstemci veriyi şifreler → sunucuya gönderir
- Sunucu şifreyi çözer → işler → cevabı şifreler → geri gönderir
- İstemci şifreyi çözer → gösterir

**Bu şifreleme işini daha detaylı anlamam lazım, nasıl çalışıyor tam olarak?**

---

### HTTP Metodları (HTTP Verbs)

5 temel metod var:

- GET:** veri almak için
- POST:** veri gönderip yeni kayıt oluşturmak için
- PUT:** tüm kaynağı güncellemek için
- PATCH:** kısmi güncelleme için
- DELETE:** silmek için

Örnek kullanım:

```
GET /users/123
```

```
POST /users
```

```
PUT /users/123
```

---

### HTTP Request & Response

## Request'te neler var:

- HTTP versiyonu (1.1, 2.0)
- URL/path
- HTTP metodu
- Headers (cookie, user agent gibi)
- Body (opsiyonel - form data veya JSON)




## Response'da neler var:

- İstenen kaynak
- Content length, type
- Headers
- ETags, last modified
- **HTTP status codes** - bunlar önemli!

---

## HTTP Status Codes

Kod aralıkları ve anlamları:

- **100-199**: Bilgilendirme mesajları
- **200-299**: Başarılı responses 
- **300-399**: Yönlendirme bilgisi
- **400-499**: Client hataları 
- **500-599**: Server hataları 

En çok kullanılanlar: 200, 201, 303, 304, 400, 401, 403, 404, 500

**İlginç nokta:** Aynı status code farklı metodlarda farklı anlama gelebiliyor. Mesela:

- GET için 200 = içerik bulundu
- PUT için 200 = güncelleme başarılı
- DELETE için 200 = silme başarılı

---

## REST API Nedir?

REST bir API tasarım mimarisi. Diğerlerinden daha kolay geliştirilebildiği için popüler.

### RESTful olması için 6 kısıtlama var:

1. **Client-server architecture**: Sunucu kaynakları servis eder, istemci tüketir
2. **Stateless**: Sunucu client'ın state'ini tutmaz! Bu çok önemli, her request kendi kendine yetmeli.

3. **Cacheable:** Response'lar cache'lenebilmeli
4. **Layered:** Sistem katmanlara ayrılabilirmeli (firewall, load balancer, web server, database gibi)
5. **Uniform interface:** Standart iletişim sistemi olmalı
6. **Code on demand** (opsiyonel): API client'ın çalıştırabileceği kod dönebilmeli

**Stateless kısmını iyi anlamam lazım** - sunucu beni hatırlamıyor, her seferinde kim olduğumu söylemek zorundayım.

---

## Resources (Kaynaklar)

REST API'de her şey resource etrafında dönüyor. Little Lemon örneğinden:

- `/orders` → order object listesi
- `/orders/16` → spesifik order object
- `/orders/16/customer` → o order'ın customer'ı
- `/orders/16/menu-items` → o order'daki menu item'ları

**Stateless örneği:** Önce `/orders/16` sonra sadece `/menu-items` dersem, sunucu hangi order'dan bahsettiğimi unutmuş olur! Açıkça `/orders/16/menu-items` demem lazım.

---

## API Endpoint (URL) Tasarımı

**Best practiceler:**

- **Küçük harf** kullan: `/orders` ✓, `/Orders` ✗
- **Kelime ayırıcı olarak tire** kullan: `/menu-items` ✓, `/menu_items` ✗
- **Değişkenlerde camelCase:** `/users/{userId}` ✓
- **Hiyerarşik ilişkiler için slash:** `/library/books/{bookId}/author`
- **Resource'lar için isim** kullan, fiil değil: `/books` ✓, `/getAllBooks` ✗
- **File extension kullanma:** Bunun yerine query param: `?format=json`
- **Filter için query string:** `/menu-items?category=appetizers`
- **Trailing slash olmasın:** `/orders` ✓, `/orders/` ✗

Query string'ler filtering için süper çözüm - diğer developer'lar da nasıl kullanacaklarını anlayabiliyor.

---

## API Test Araçları

**curl:** Command line tool, basit GET request'ler için ideal

```
curl https://httpbin.org/get
```

**Postman:** Grafiksel arayüz, debugging için harika

**Insomnia:** Benim favorim - kullanımı kolay, ücretsiz, cross-platform

Insomnia'da request collection oluşturup farklı HTTP metodlarını test edebiliyorsun. JSON body vs. kolayca eklenebiliyor.

**Bu araçlarla pratik yapmam lazım,** özellikle Insomnia'yı iyice öğrenmeliyim.

---

## API Geliştirme Best Practiceleri

**KISS (Keep It Simple, Stupid):** Bir API çok fazla iş yapmamalı. Little Lemon'da "günün yemeği" örneği iyiydi - tek API ile hem eski yemeği kaldırıp hem yenisini seçmek yerine, iki ayrı API call yapılıyor.

**Filtering & Pagination:** Büyük sonuç setleri için filter ve sayfalama şart:

```
/menu-items?category=appetizers&page=2&size=20
```

**Versioning:** API'de büyük değişiklik yapınca version kullan. En fazla 2 version aktif tutmak mantıklı.

**Caching:** Database load'u azaltmak için caching şart. Aynı data tekrar tekrar sorgulanmasın.

**Rate Limiting & Monitoring:** API abuse'u önlemek için rate limiting kullan (dakika/saat başına maksimum call). Monitoring ile 4xx/5xx hatalarını ve latency'i takip et.

---

## API Güvenliği

**SSL/HTTPS:** Temel şifreleme - tüm endpoint'ler HTTPS olmalı

**Signed URLs:** HMAC signing ile URL'lerin gerçek kaynaktan geldiğini doğrulama

**Authentication:**

- Basic auth (username/password her request'te) → güvensiz
- **Token-based auth** → daha iyi: login'de token al, sonraki request'lerde header'da gönder

**Status codes:**

- **401 Unauthorized:** Kimlik doğrulama başarısız (yanlış şifre)
- **403 Forbidden:** Kimlik doğru ama yetki yok

**CORS & Firewalls:** Hangi domain'lerin API'yi çağırabileceğini kısıtlayabilirsin. Firewall ile IP restriction da yapılabilir.

---

## Erişim Kontrolü (Authorization)

Authentication = kimlik doğrulama, Authorization = yetkilendirme

### Roller ve privileges:

- **Customer:** menüyü gör, sepete ekle, sipariş ver
- **Manager:** menü item ekle/düzenle/sil, tüm siparişleri gör
- **Delivery crew:** kendi siparişlerini gör, durum güncelle

Bir kullanıcı **birden fazla role** sahip olabilir. General manager = manager + accountant + HR rolleri bir arada.

Access control sistemi iyi planlanmazsa sonradan debugging çok zor olabilir.

---

## Kotlin'de Ktor ile HTTP Calls

Android'de networking için **Ktor** veya **Retrofit** kullanılıyor.

### Ktor avantajları:

- Open-source
- Type-safe requests
- Asynchronous & synchronous call support
- Multi-platform

Temel kullanım:

```
val client = HttpClient(Android)
val response: HttpResponse = client.get("https://api.example.com/data")
```

**Response handling:** HTTP status code, body, timing bilgileri alınabiliyor.

**Asynchronous calls** için `async` ve `await` kullanılıyor - aynı anda birden fazla call yapılabilir.

**Önemli:** Android'de **HTTPS kullanmak zorunlu** - yeni versiyonlar HTTP'yi blokluyor!

Little Lemon app'inde menu fetch etme örneğinde `suspend` function ve coroutine kullanımını gördüm - **bunu iyice pekiştirmem lazım.**

---