Bu kodlar, Swift dilinde bir ağ katmanı (network layer) oluşturmak için yazılmıştır. Ağ katmanı, uygulamanın sunucuyla iletişim kurmasını sağlar. İşte adım adım açıklamalar:

**1. Endpoint.swift**

Bu dosya, API endpoint'lerini ve HTTP metodlarını yönetir.

**HTTP Metodları (httpMethod Enum'ı)**

* GET, POST, PUT, DELETE gibi HTTP metodlarını temsil eder.
* Her bir case, ilgili metodun string karşılığını döndürür (örneğin, .get → "GET").

**Endpoint Enum'ı**

* Uygulamada kullanılan endpoint'leri gruplar (örneğin, GetUsers, comments).
* Yeni endpoint'ler eklenebilir.

**endpointProtocol Protokolü**

* Tüm endpoint'lerin uyması gereken kuralları belirler:
  + baseUrl: API'nin kök URL'i (https://jsonplaceholder.typicode.com).
  + path: Endpoint'in yolu (örneğin, /users).
  + method: İstek için kullanılacak HTTP metodu.
  + header: İstek başlıkları (token vb. için kullanılabilir, şu an nil).
  + request(): URLRequest nesnesi oluşturur.

**Endpoint Extension'ı**

* endpointProtocol'ü uygulayarak her endpoint için değerleri belirler:
  + **baseUrl:** Tüm endpoint'lerin temel URL'i.
  + **path:** Her endpoint için yol (örneğin, GetUsers → /users).
  + **method:** İstek metodu (GET, POST vb.).
  + **header:** İstek başlıkları (bu örnekte kullanılmamış).
  + **request():**
    - URLComponents ile URL oluşturur.
    - URLRequest nesnesine HTTP metodu ve header'ları ekler.
    - Örnek: GetUsers için https://jsonplaceholder.typicode.com/users URL'ini oluşturur.

**2. NetworkManager.swift**

Ağ isteklerini yöneten sınıf. Singleton pattern kullanılarak tek bir örnek üzerinden işlem yapılır.

**shared (Singleton)**

* static let shared = NetworkManager() ile tek bir örnek oluşturulur.
* private init() ile dışarıdan yeni örnek oluşturulması engellenir.

**request() Metodu**

* Generic bir fonksiyon: T: Decodable ile JSON'dan decode edilebilir bir tip bekler.
* Parametreler:
  + endpoint: İstek yapılacak endpoint.
  + completion: Sonucu döndüren closure (başarılı/başarısız).
* İşlem Adımları:
  1. URLSession ile data task oluşturulur.
  2. Gelen veri (data), yanıt (response) ve hata (error) kontrol edilir.
  3. HTTP durum kodu (status code) kontrol edilir (200-299 arası başarılı).
  4. Veri decode edilerek modele dönüştürülür.
  5. Sonuç completion ile iletilir.

**getUser() ve getcomments()**

* Özel metodlar: GetUsers ve comments endpoint'lerini çağırır.
* request() metodunu kullanır ve sonucu işler.

**3. Model Dosyaları (comment.swift ve model.swift)**

API'den gelen JSON verilerini parse etmek için Codable protokolünü uygulayan struct'lar.

**user Struct'ı**

* Kullanıcı bilgilerini temsil eder: id, name, email, address, phone, vb.
* İç içe struct'lar (Address, Geo, Company) ile karmaşık verileri yönetir.

**CommitElement Struct'ı**

* Yorum bilgilerini temsil eder: postID, id, name, email, body.
* CodingKeys ile JSON anahtarlarını özelleştirir (örneğin, postId → postID).

**4. ContentView.swift**

Kullanıcı arayüzü. onAppear içinde ağ isteği yapılır.

**onAppear()**

* NetworkManager.shared.getcomments() ile yorumlar çekilir.
* Result tipi ile başarılı/başarısız durumlar ayrılır:
  + Başarılı: Yorumlar konsola yazdırılır.
  + Başarısız: Hata mesajı gösterilir.

**Önemli Kavramlar**

* **Codable:** JSON ↔ Swift struct dönüşümü için kullanılır.
* **Generic Fonksiyon:** request<T: Decodable> ile farklı modeller desteklenir.
* **Singleton Pattern:** NetworkManager tek bir örnekle yönetilir.
* **Result Tipi:** Başarılı/başarısız durumları temsil eder.

**Geliştirilebilir Alanlar**

* **Parametre Desteği:** Query parametreleri eklenebilir.
* **Header Yönetimi:** Token ekleme mekanizması geliştirilebilir.
* **Hata Yönetimi:** Özelleştirilmiş hata tipleri oluşturulabilir.
* **POST/PUT/DELETE:** Diğer HTTP metodları için body eklenebilir.