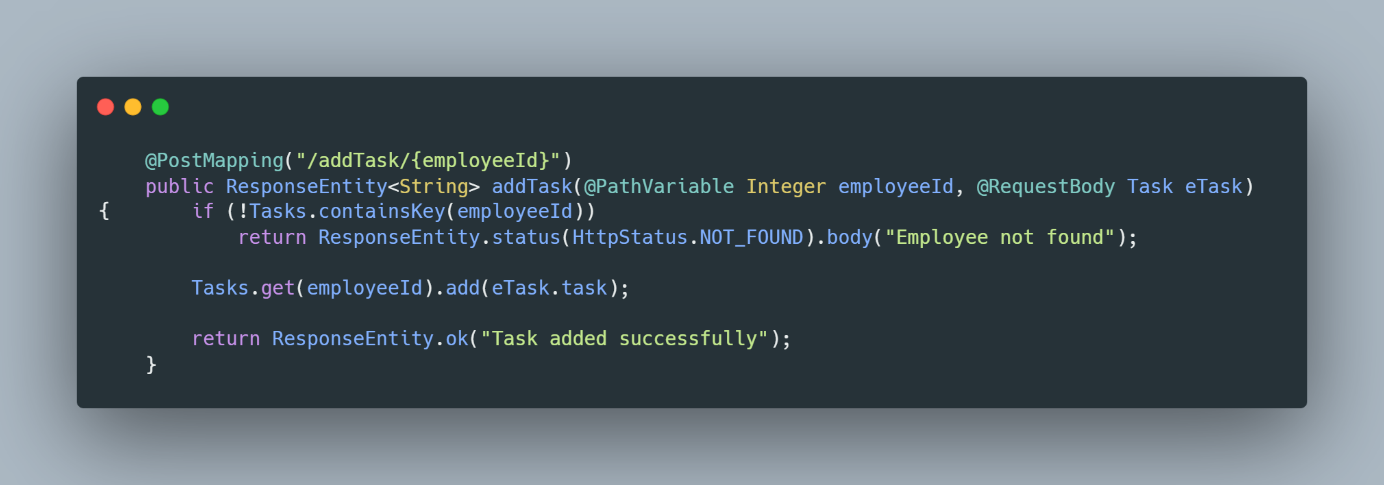
**GÜNGÖREN BELEDİYESİ BELEDİYE ÇALIŞAN TAKİP SİSTEMİ**

**Web Servis Geliştiremesi:** 8080 portu üzerine herhangi bir path eklenmeden rest api servisleri oluşturuldu.

**addTask:** @PathVariable ile URL'den gelen employeeId değerini metot parametresine alır. @RequestBody ile İstek gövdesindeki Task nesnesini alır ve eTask parametresine atar.

Tasks koleksiyonunda employeeId anahtarının olup olmadığını kontrol eder eğer null dönerse verilen hata mesajı NOT\_FOUND döner. employeeId mevcutsa, ilgili çalışanın görev listesine yeni görevi ekler.

Görev başarıyla eklendiğinde, HTTP 200 (OK) durumu ve "Task added successfully" mesajıyla yanıt döner. 

**listAllTasks:** HTTP GET isteğiyle /taskList URL'sine gelen istekleri bu metoda yönlendirir.

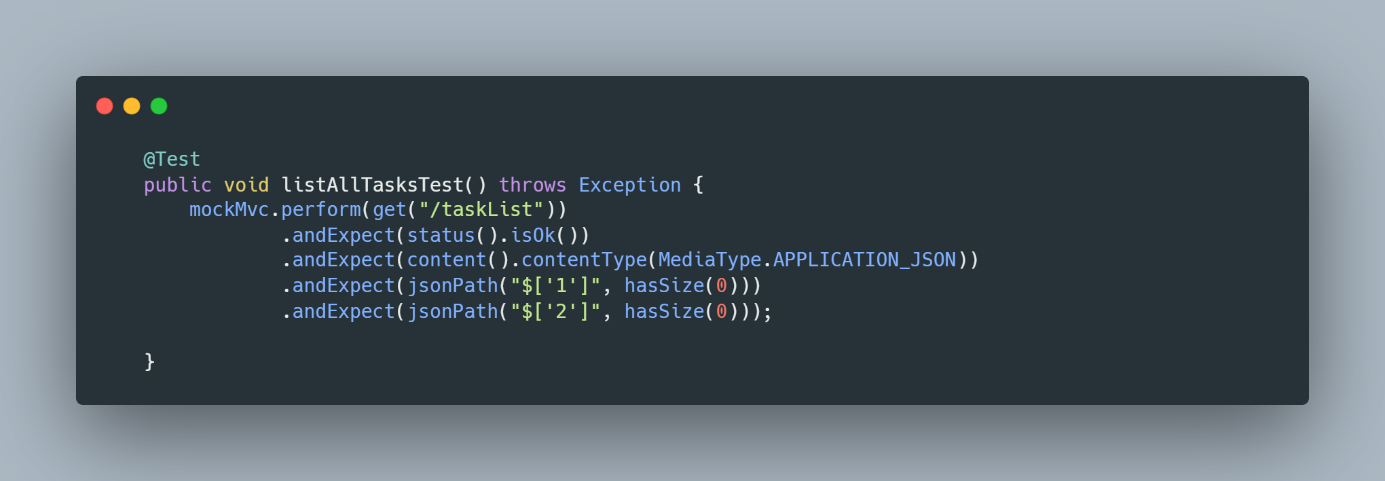
Bu metodun dönüş tipi, Integer türünde anahtarlar ve List<String> türünde değerler içeren bir haritadır.

“return Tasks;” ile Tasks haritasını döner. Bu harita, çalışanların ID'lerini anahtar olarak, bu çalışanlara atanmış görevlerin (String) listelerini değer olarak içerir.



**BİRİM TESTLER:** Programlama dilleri temel olarak giriş-input alarak çıktı-output üretir. Girişlerin ve çıkışların kaynağı farklı olsa da aynı girişlerin aynı sonuç vermesi beklenir. Bu amaç ile JUnit kütüphanesini kullanarak yazılan servislerin testleri gerçekleştirildi.

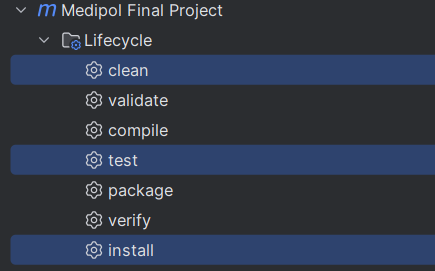
**listAllTasksTest:** MockMvc nesnesi kullanarak /taskList URL'sine bir GET isteği gönderir. Yanıt durumunun HTTP 200 (OK) olmasını bekler. Yanıtın içeriğinin JSON formatında olmasını bekler. JSON yanıtında 1 ve 2 anahtarına karşılık gelen listenin boyutunun 0 olmasını bekler. (öncesinde herhangi bir test taskı eklenmemiş olmasından kaynaklı)

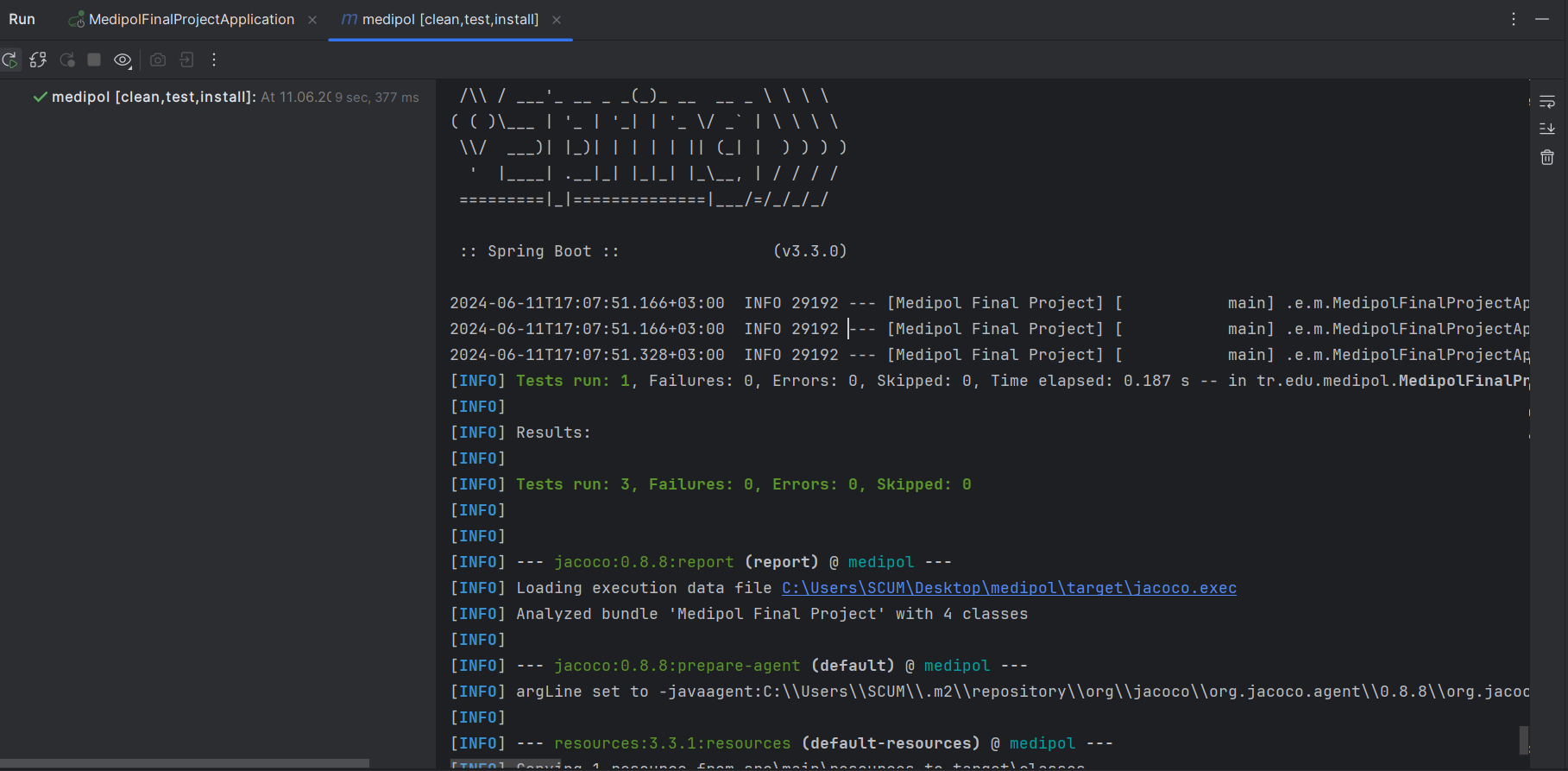


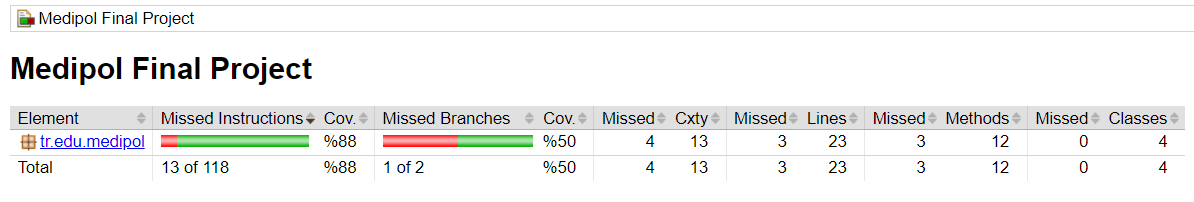
**addTaskTest:** "Test Task" adında yeni bir Task nesnesi oluşturur. MockMvc nesnesi kullanarak /addTask/1 URL'sine bir POST isteği gönderir. 1 burada employeeId'yi temsil eder. Yanıt durumunun HTTP 200 (OK) olmasını ve Yanıtın içeriğinin "Task added successfully" olmasını bekler. Son olarak bir önceki fonksiyondaki list isteğini gönderiyor fakat bu sefer beklenen değer “Test Task” olarak modifiye edildi.



**COVERAGE:** test kapsama oranını hesaplatmak için jacoco plugini “pom.xml” dosyasının en alt kısmındaki “build/plugins” etiketinin altında eklendi. Ardından aşağıdaki aksiyonlar çalıştırıldı.

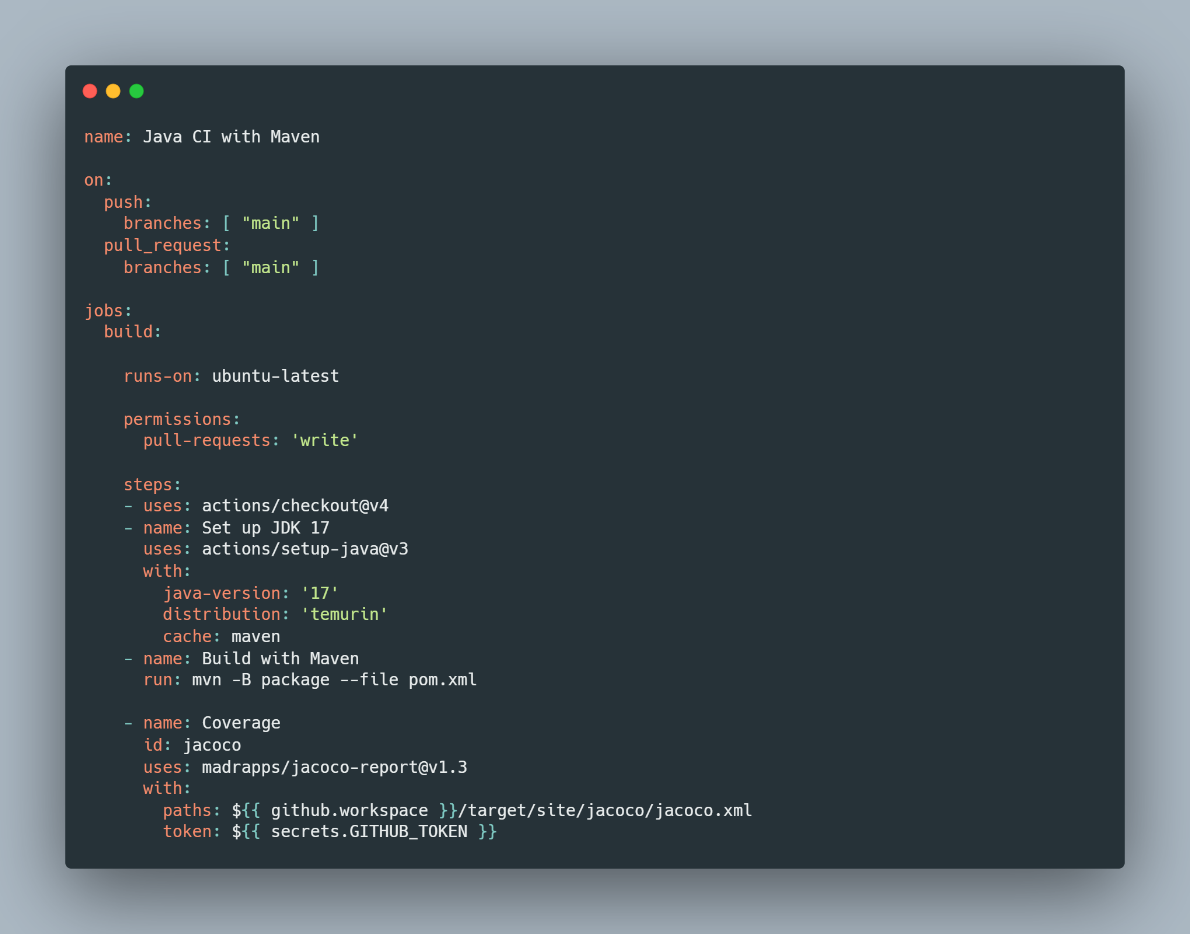






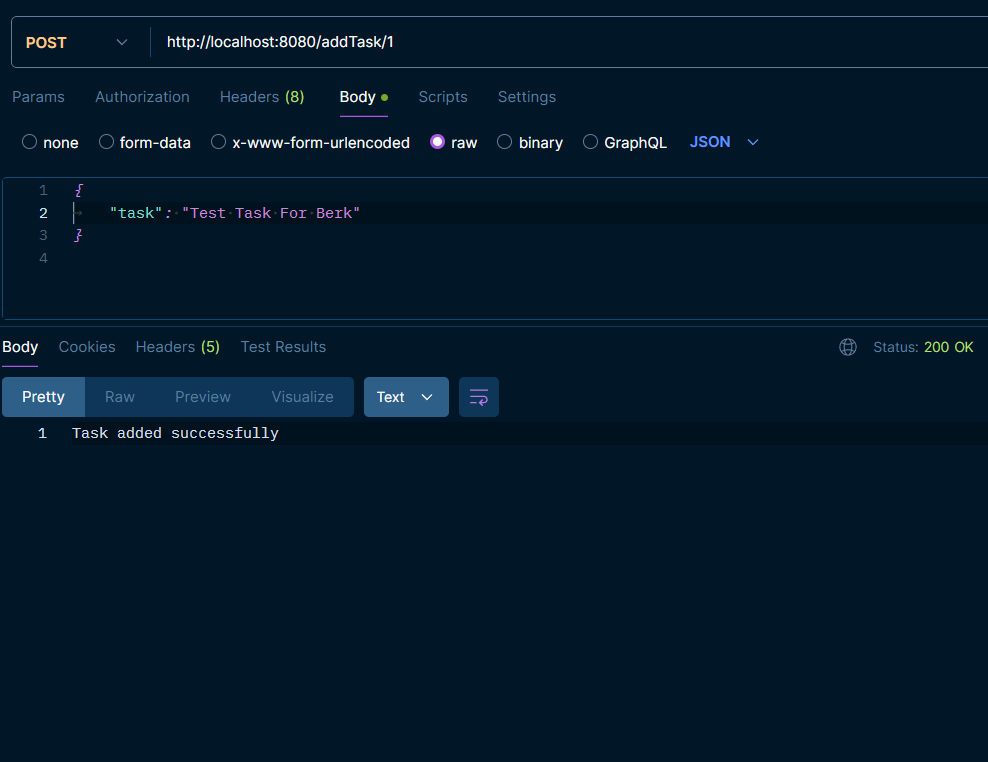
**Sürekli Entegrasyon:** yapılan değişikliklerini sürekli olarak entegre etmek için otomatikleştirilmiş bir süreçtir. Bu nedenle belirtilen kodu eklendi ve gerekli izinler sağlandı.

Bu action kod üzerindeki her güncellemede -pullrequestler dahil olmak üzere- test kapsama oranını hesaplayacak.

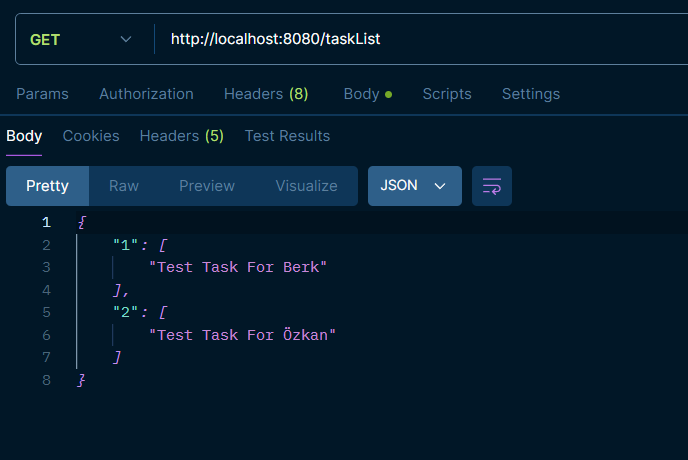


**Postman & JMeter Testleri:** Postman ile servisler, JMeter ile strese dayanıklılık testleri yapıldı.

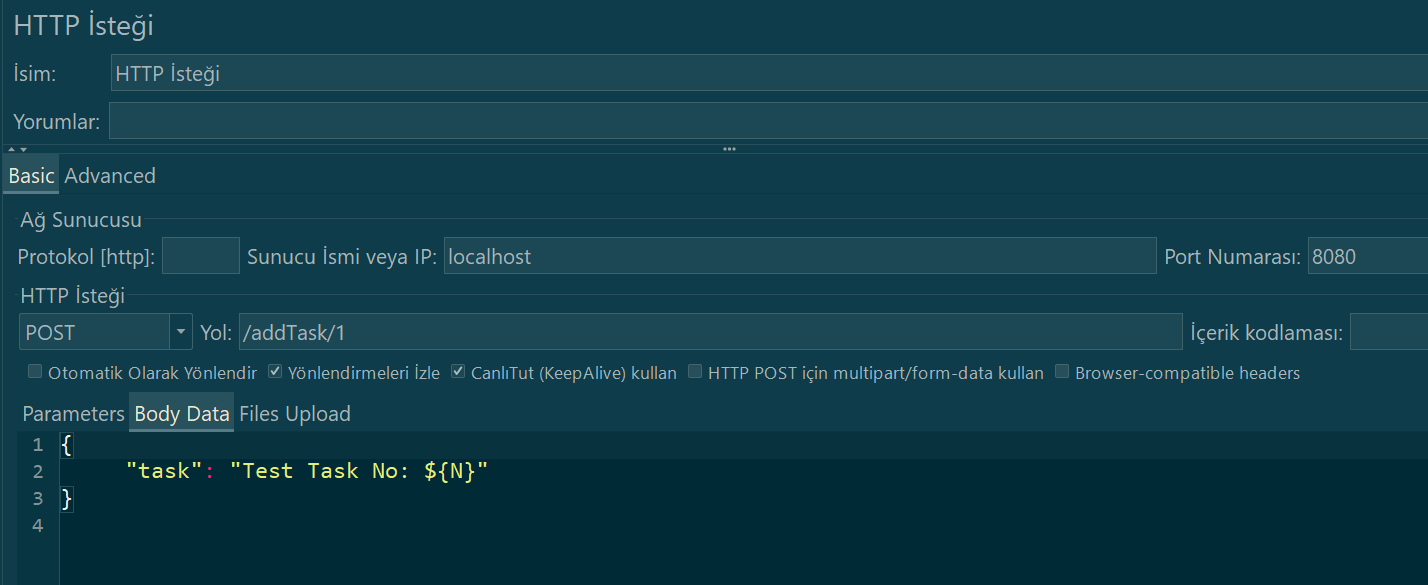
**/addTask/{EmployeeId}:** Verilen URL adresine adrese çalışanın idsi ve “task” bodysi ile POST isteği gönderildi.

****

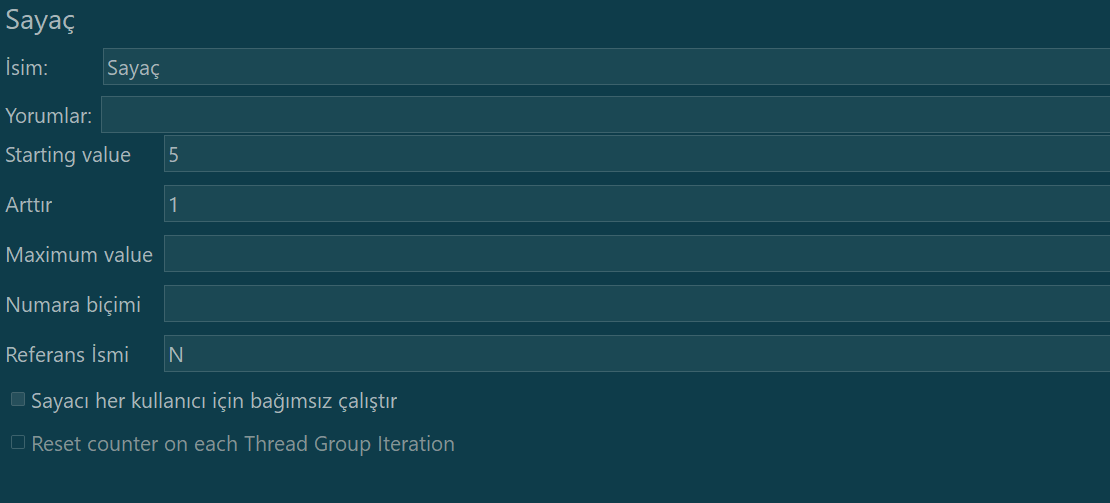
**/taskList:** Verilen URL adresine GET isteği gönderilerek öncesinde oluşturulan taskler listelendi.

****

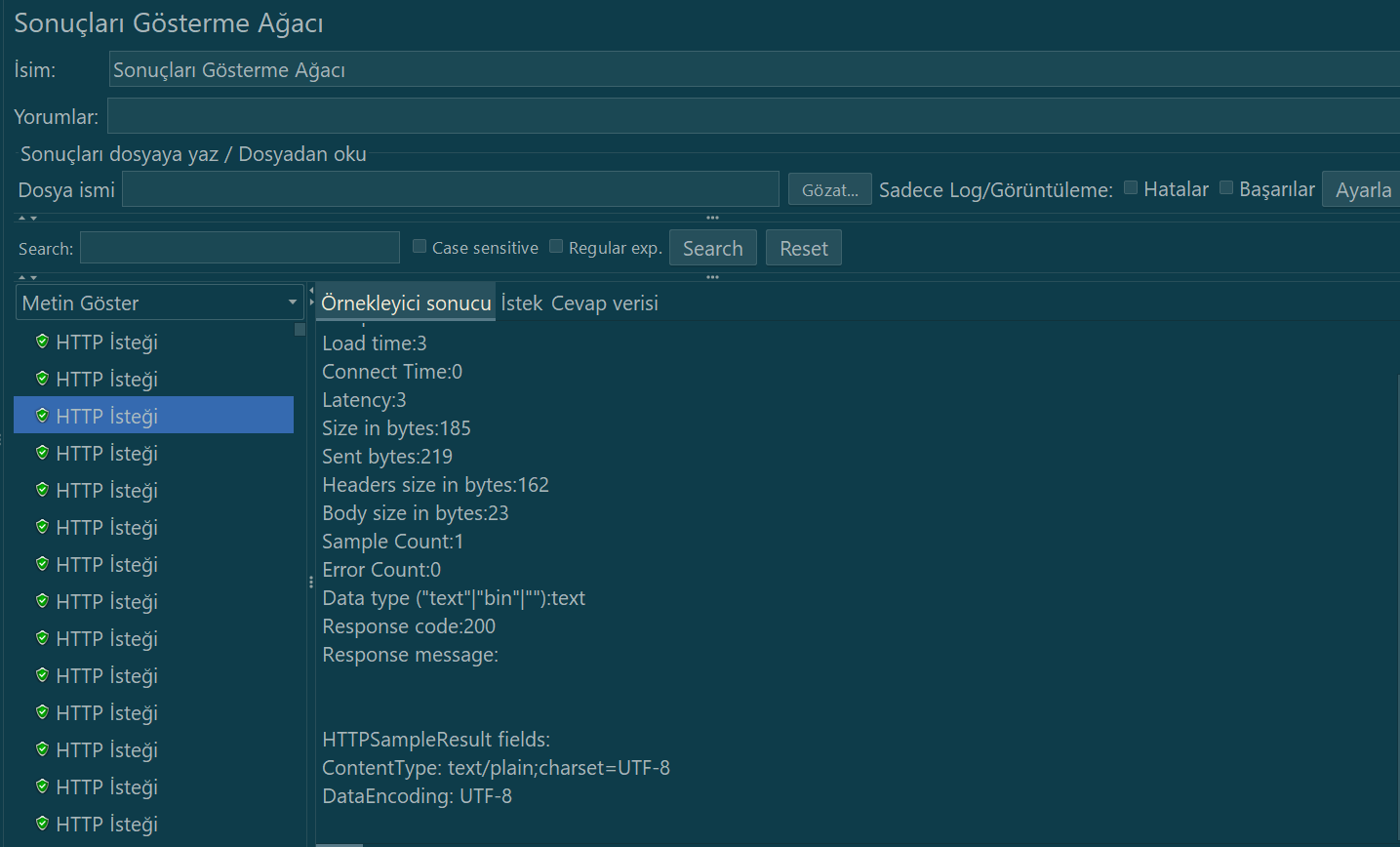
**Http Request:** Göndereceğimiz isteğin türü, bodysi ve yolunu girerek testimizin requestini tanımlandı.

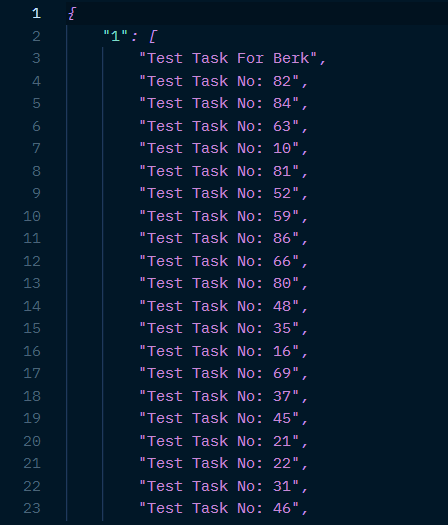
****

**Counter:** Counter ile her gönderilen istekte 1 artan bir değişken oluşturuldu.

****

**Result Tree:** Çalıştırılmanın sonrasında isteklerin başarılı olup olmadığını ve detaylarını görüntülemek için kullanıldı.

****

****

**GİT:** Verilen kodları sırasıyla kullanılan IDEnin terminale girilip repository oluşturuldu ve pushlandı.

****