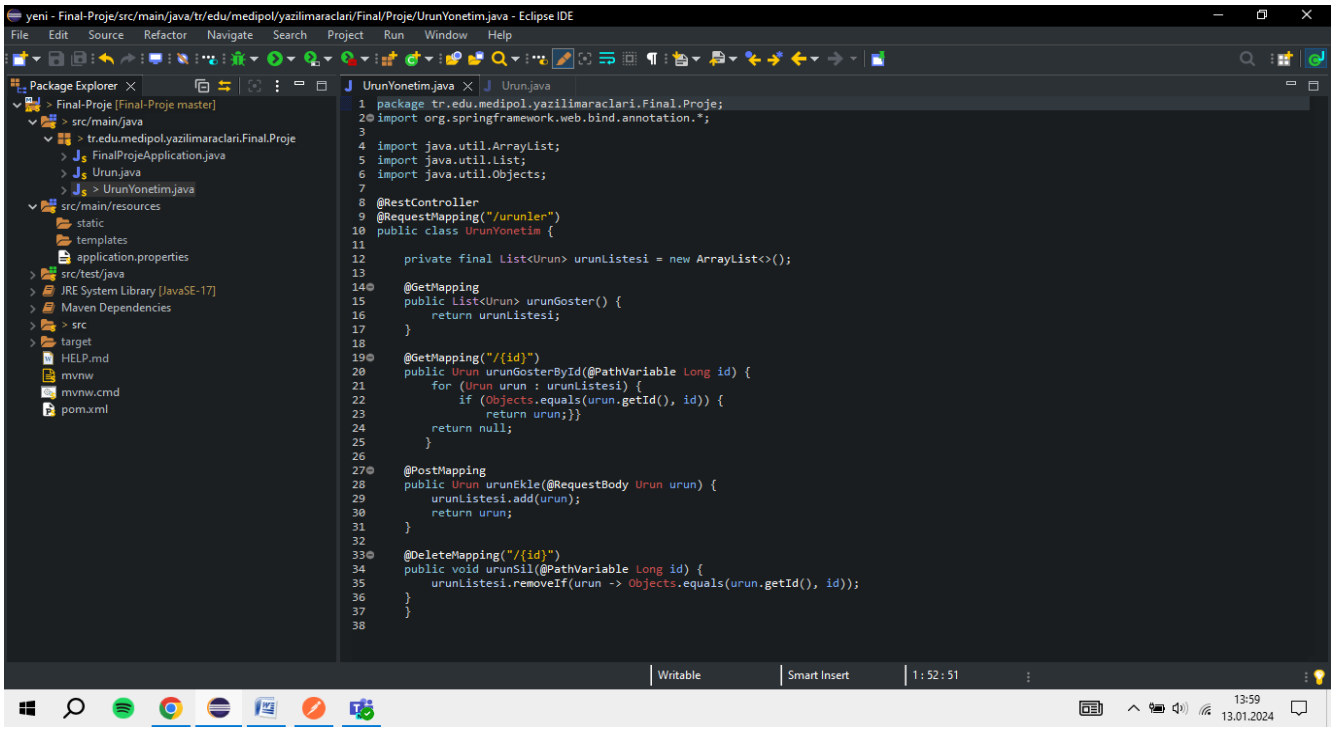
YAZILIM GELİŞTİRME ORTAM VE ARAÇLARI FİNAL PROJE ÖDEVİ

PDF RAPORU

HAPPY CENTER MARKET WEB SERVİS GELİŞTİRMESİ

İlk önce bir spring boot projesi oluşturmak için https://start.spring.io/ adresine gittim. Burda maven ve java seçeneklerini seçtim projemin group ve artifact idlerini yazdım ve bir restful web servis oluşturacağım için spring web dependency projeme ekledim. Ve sonra bu dosyayı zip olarak indirip zipten çıkardım ve bu klasorü Eclipse üzerinde çalıştığım workspace'e import ettim.



İlk Önce UrunYonetim diye bir class oluşturdum ve kullanacağım kütüphaneleri en üstte import ettim. RestController ve RequestMapping anotasyonlarıyla hangi endpointi (/urunler) kontrol edeceğimi belirttim.

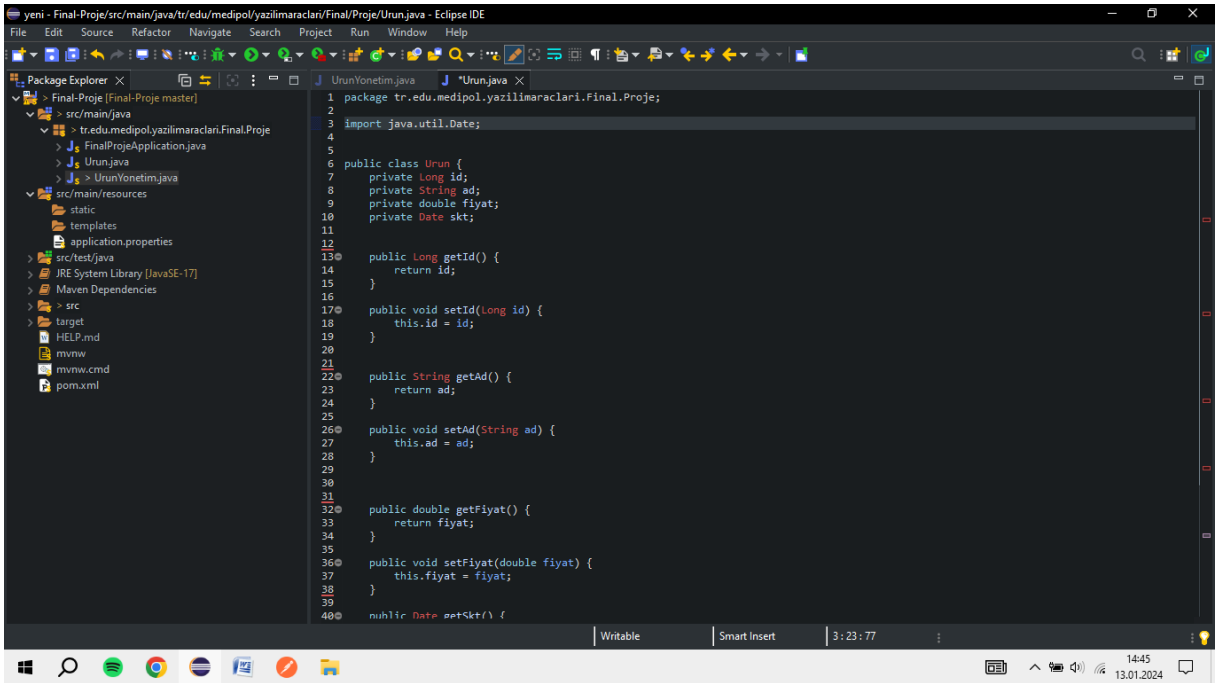
Eklenen ürünleri saklamak için bir ArrayList kullandım.

Ve ilk metodumla yani @GetMappingle Javada bir Restful Api yaratmaya başladım. Bu metod bana tüm ürün listesini (array listin içine postla eklenen ürünleri) döndürüyordu.

İkinci metodumda ise bu sefer get mappingi yani get isteğini spesifik bir id üzerinden çağırıyorum.Burda getId adlı getter metoduyla belirtilen iddeki ürün, ürün listesinde varsa dönüyor yoksa null dönüyor. Aynı zamanda Object yardımcı metoduyla da iki idnin eşit olup olmadığını kontrol ettim.

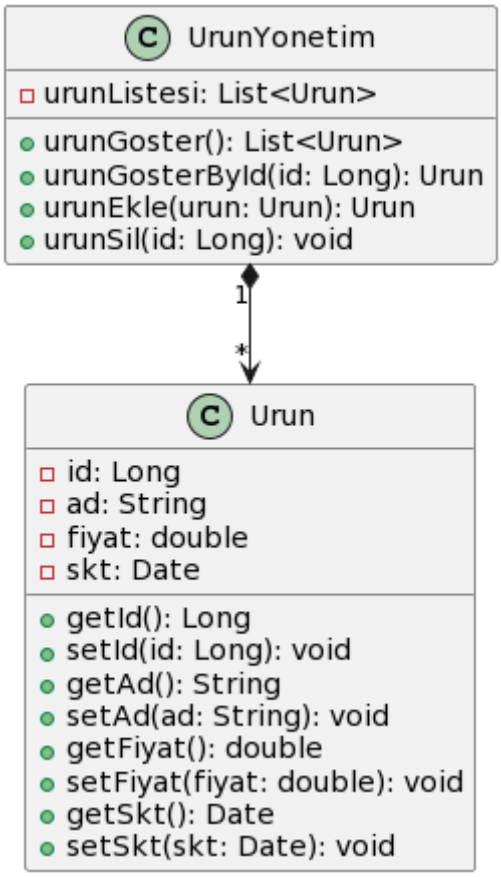
Sıradaki post metodumda @RequestBody anotasyonuyla post isteğine gelen json verilerini bir urun nesnesine (Urun.java sınıfındaki tanımlı alanlara) dönüştürdüm.Post edilen urunleri , urunlistesine kaydettim ve kaydedilen urunu geri döndürdüm.

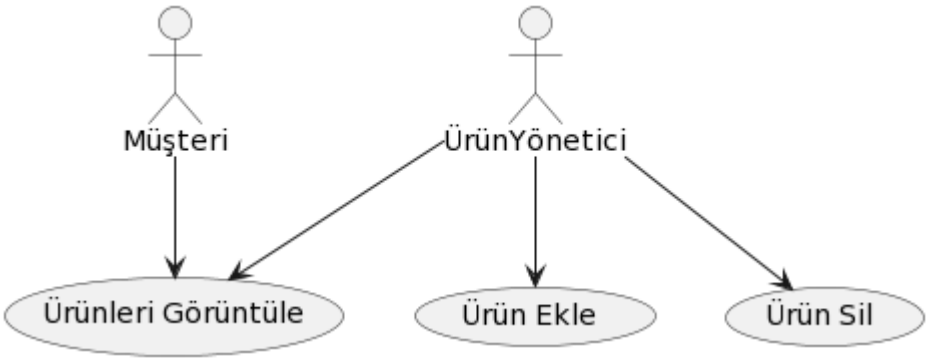
Son metodumda ise belirli bir idye gelen delete isteği üzerinden o idye sahip ürünü listeden sildim. removeIf metodu ile idye eşitlik şartı koydum.



Projemin içine ayrı bir sınıf daha açtım ve ismini Urun.java koydum. Bu sayfada Urun sınıfını tanımladım ve sınıfın özelliklerini belirledim. Aynı zamanda bu 4 özellik için erişim ve değişiklik yapmak için ayrı ayrı getter & setter metodlarını yazdım. Bu metodlar özellikle JUnit testleri için çok gerekli.

UML SINIFLARI VE USECASE DİYAGRAMI



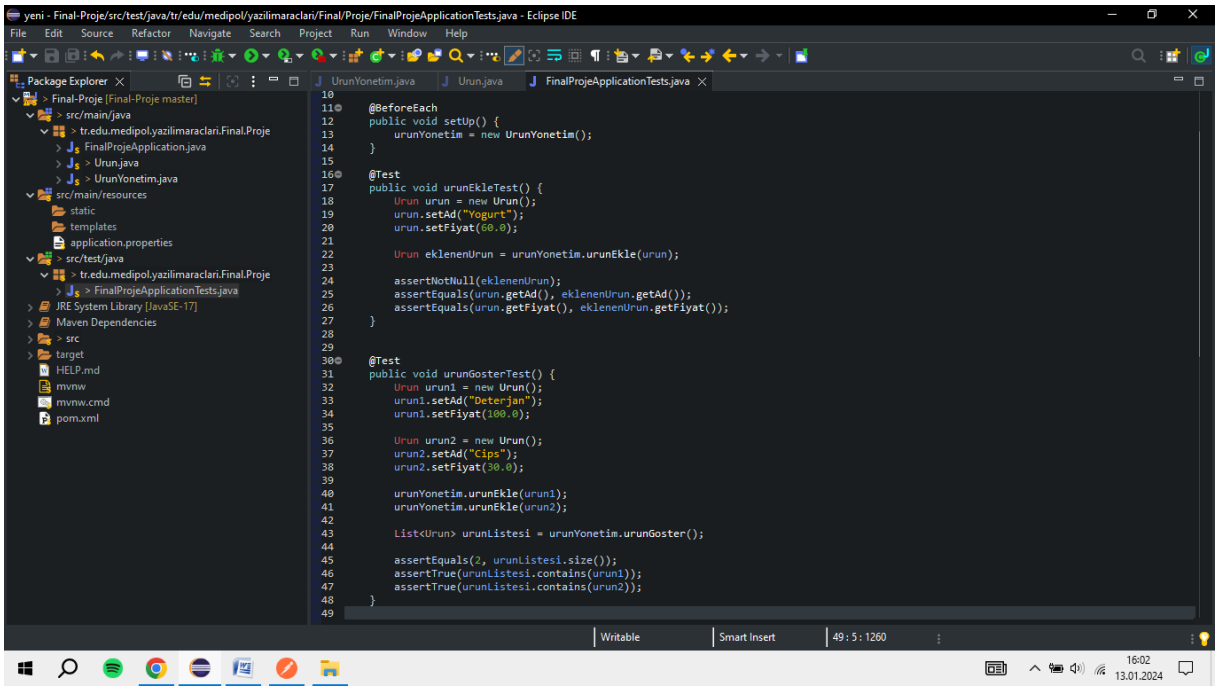


Bu uml diyagramında UrunYonetim sınıfından Urun sınıfını türettim. Sınıfların özelliklerini ve metodlarını tanımladım Ve iki sınıf arasında birçok ilişkisi kullandım UrunYonetim sınıfı , Urun sınıfından birden fazla urun nesnesini içerebilir ve yönetebilir .

Kullanım diyagramındaysa ürünleri yöneten kişi tüm metodlara erişim sağlayabiliyor fakat müşteri sadece ürünleri görüntüleyebiliyor.

BİRİM TESTLER

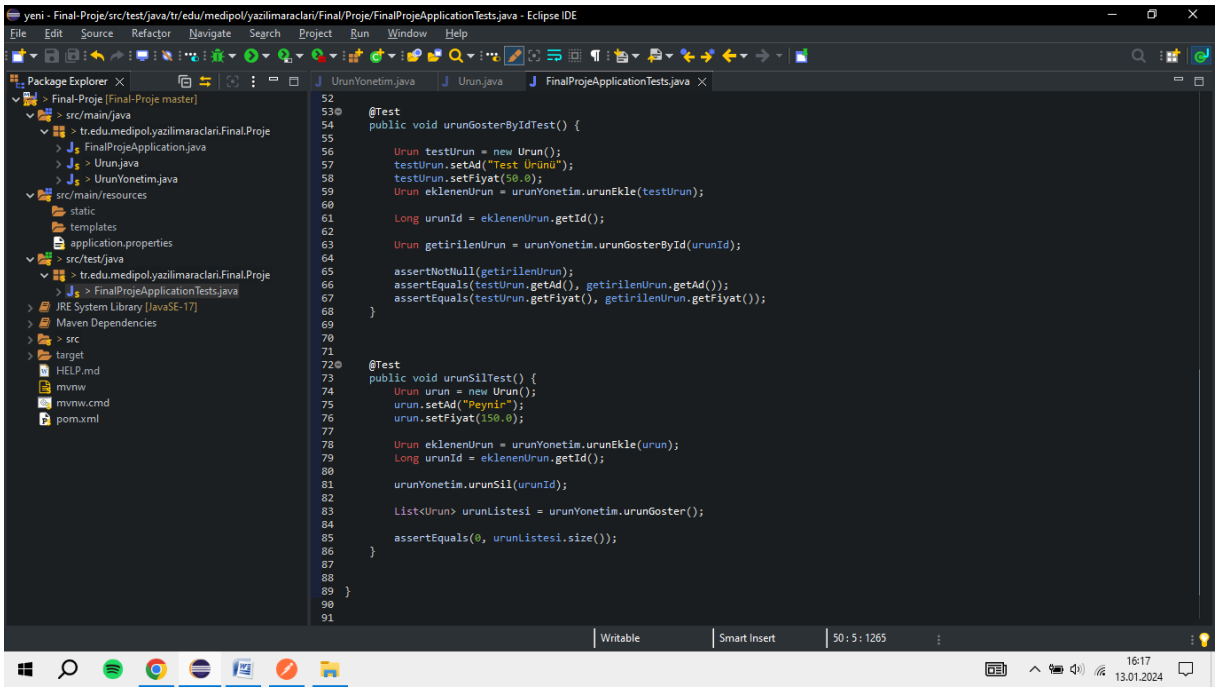
JUnit testler için projemi kurduğumda halihazırda gelen src/test/java altındaki test dosyasını kullandım. Metodlarımın doğru çalışıp çalışmadığını görebilmek için 4 tane metodum için 4 ayrı junit test yazdım.



Test metodlarını çalıştırmadan önce @BeforeEach anotasyonu ve setUp metodunu kullanarak herbir test başlamadan öncesinde UrunYonetim sınıfından bir nesne oluşturularak urunYonetim adlı değişkene atanmasını sağladım . Bu sayede her test metodu bağımsız olarak çalışacak ve birbirlerini etkilemeyecek.

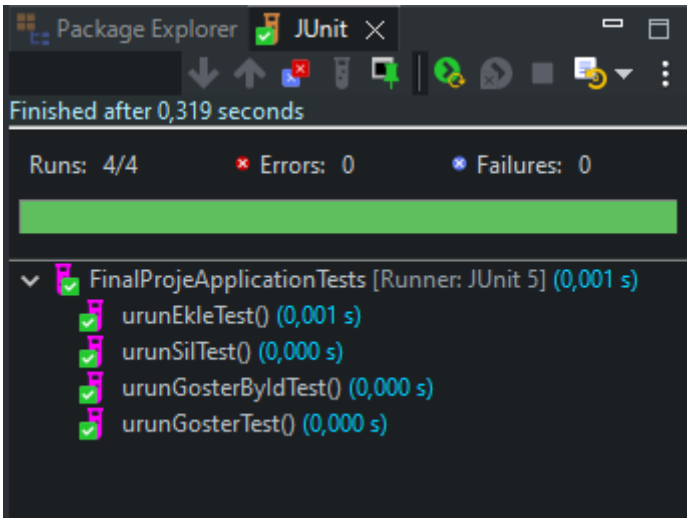
İlk testte Yogurt adında , 60 tl fiyatlı bir urun ekleyip ürünün başarıyla eklenip eklenmediğini test ettim. (post)

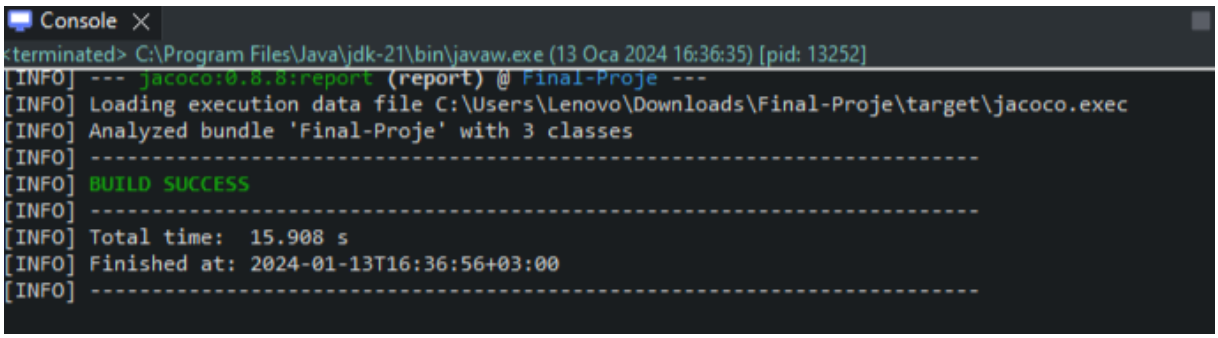
İkinci testte urun listesine iki ayrı ürün ekledim ve ürün listesi boyutunun 2ye eşit olup olmadığını ve ürünleri içerip içermediğini test ettim. (get)



3. testte belirli bir ID'ye sahip ürünü doğru bir şekilde getirip getirilmediğini test ettim. (getById)

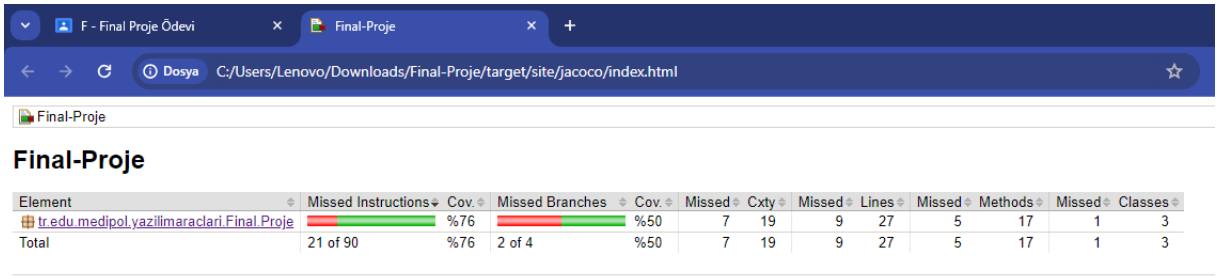
4. testte ise ilk önce listeye bir ürün ekleyip sonra o ürünü sildikten sonra ürün listesinin boyutunun 0 olup olmadığını test ettim. (delete)





Testler başarılı oldu ve kodumun doğruluğu ispatlandı.

Yazdığım birim testlerin kodun ne kadarını kapsadığını görmek için ilk önce pom.xml dosyasına jacoco pluginini ekledim ve projemde maven install yaparak jacoco kütüphanesinin projemin içine eklenmesini sağladım. Sonrasında target/site/ altına jacoconun eklendiğini gördüm ve jacoco klasorünün içindeki index.html dokumanına tıklayarak code coverage oranını gördüm.

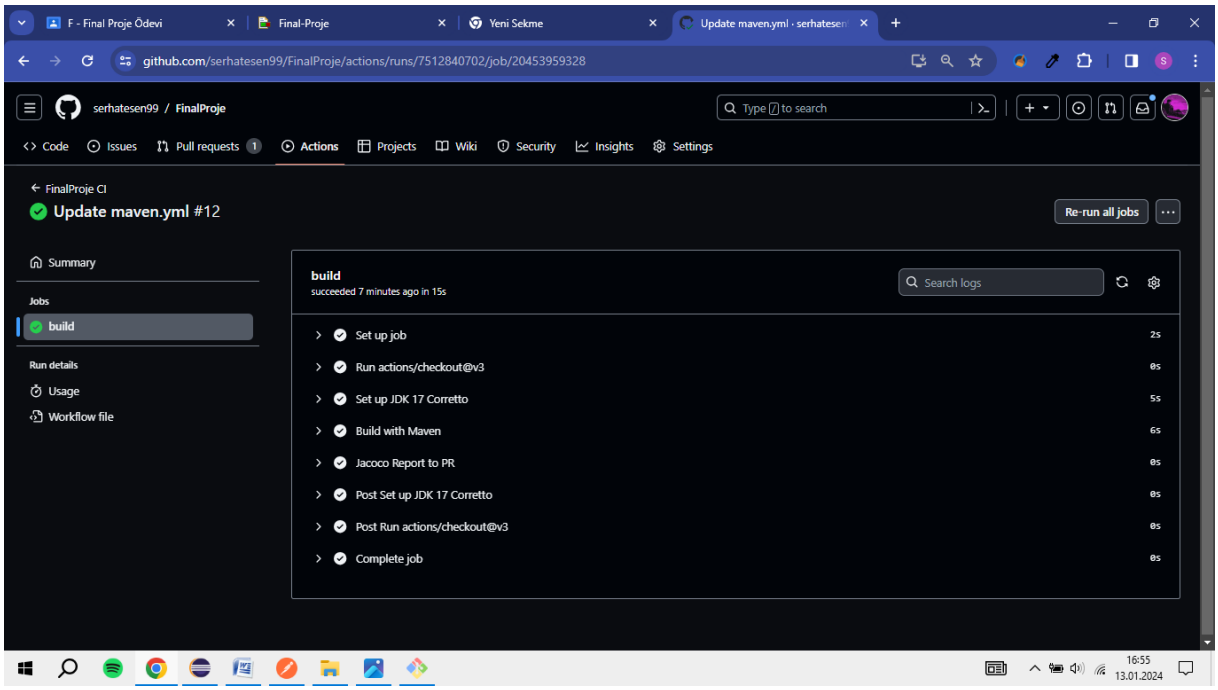


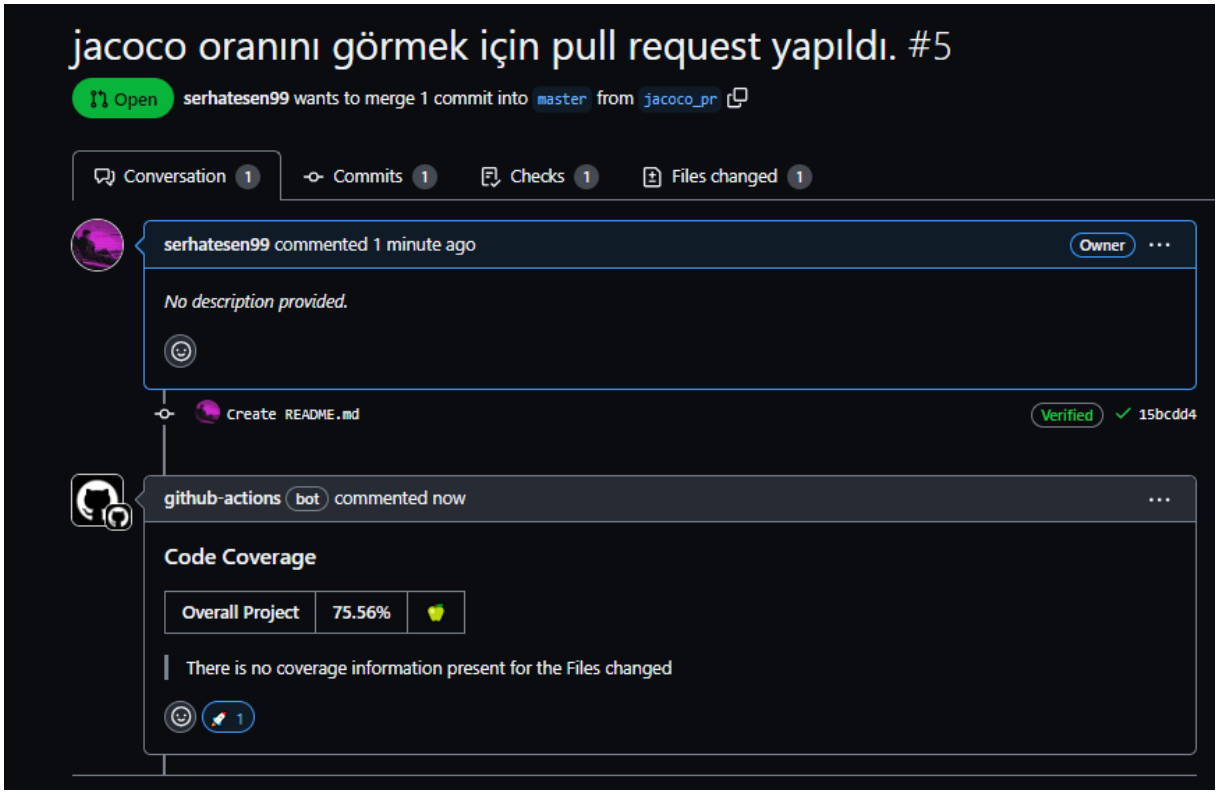
Sürekli Entegrasyon (Github Actions)

Githubdaki actions kısmında java with maven seçerek projemde CI adımını sağlamış oldum bundan sonra her pushlamada veya her pull requestte github actions benim yerime kodu tarayacak ve benim koyduğum şartlara göre derlemeyi başarılı veya başarısız kılabilecek böylece sürekli entegrasyon sağlanmış olacak.

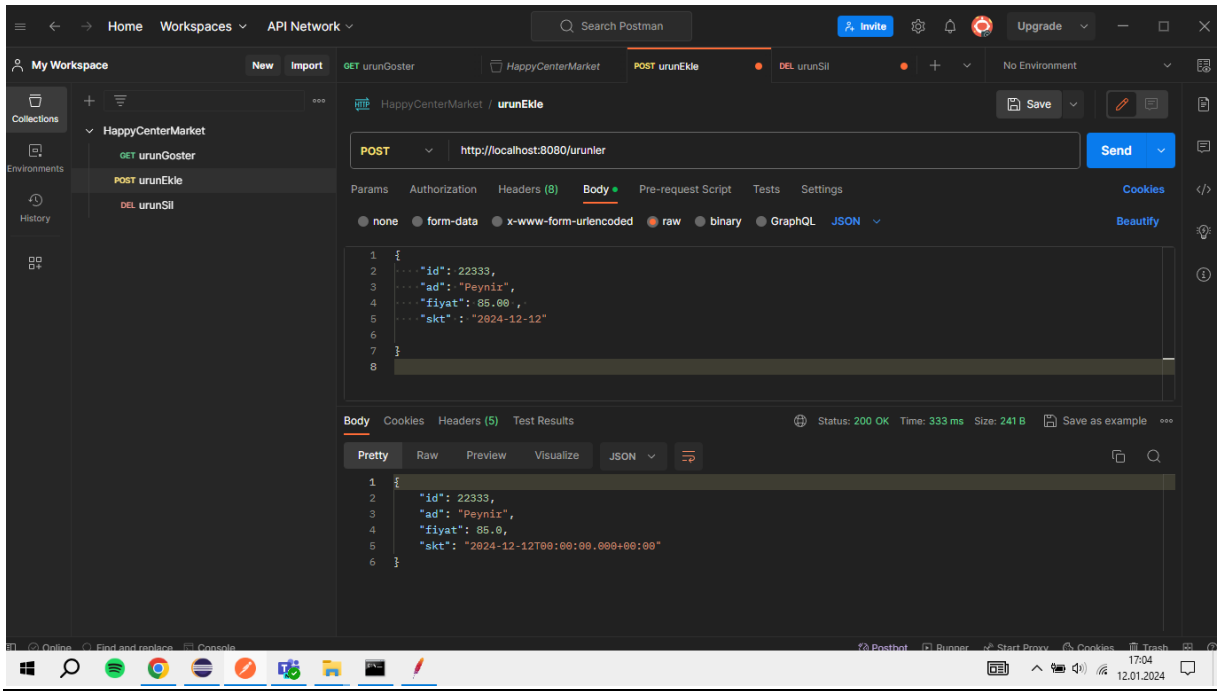
Github workflows klasoru içerisindeki maven.yml dosyasındaki kodları güncelledim ve jacocoyu ekledim .

Jacoco oranını github projemde görmek içinse bir pull request oluşturup read.me dosyası güncelledim raporumda jacoco oranını görmüş oldum.

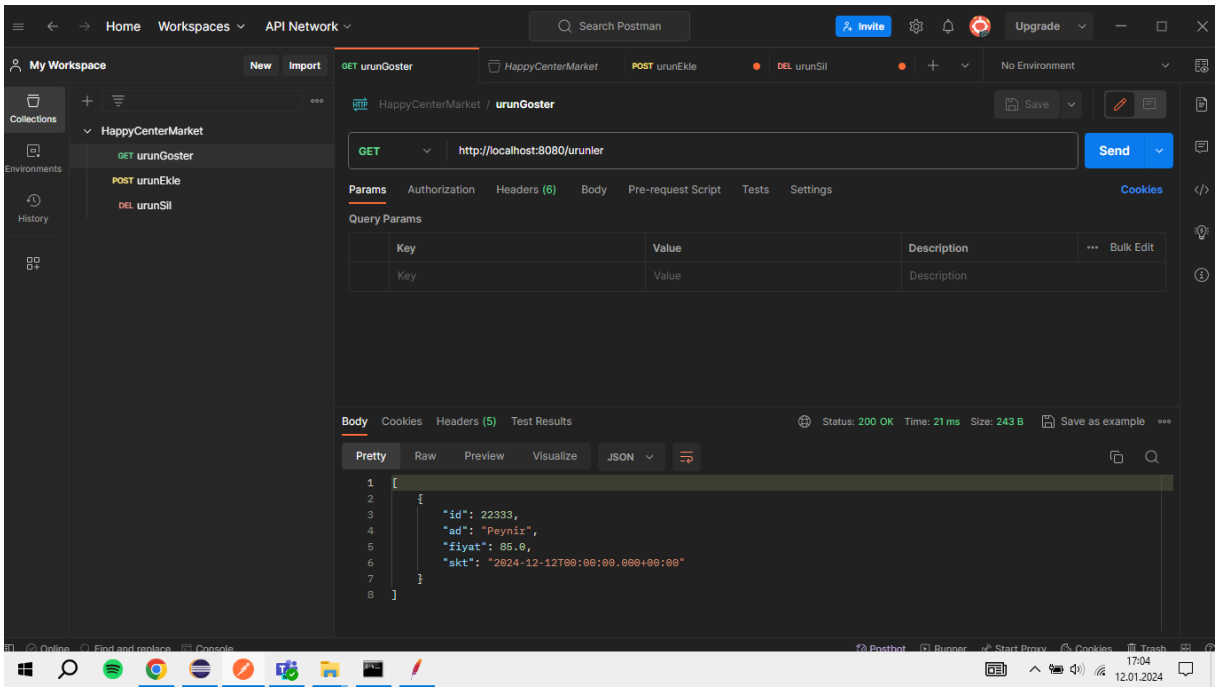




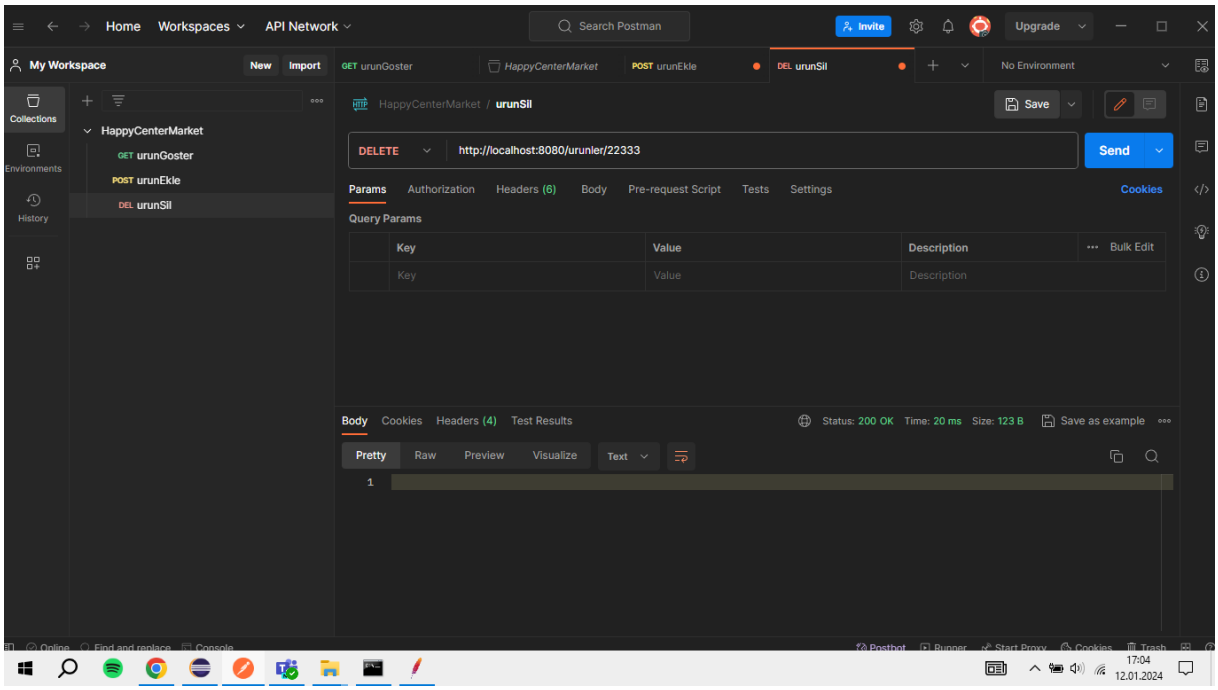
Postman & JMeter Testleri



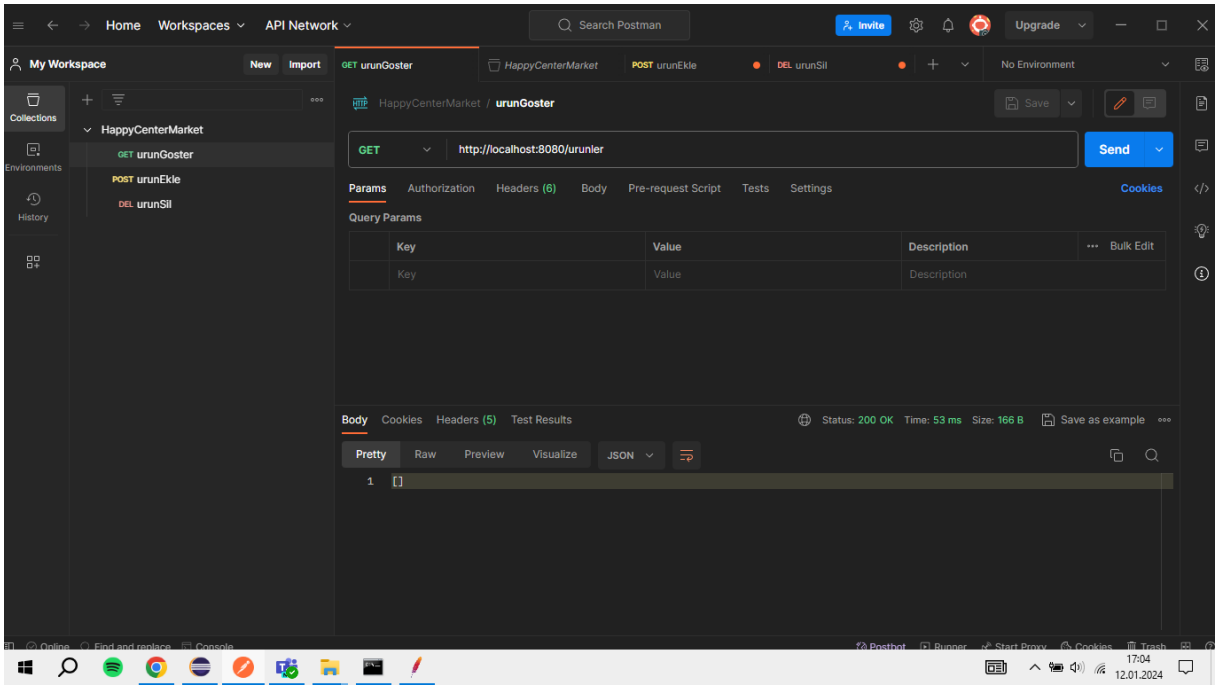
POST İSTEĞİ İLE ÜRÜN EKLENDİ.



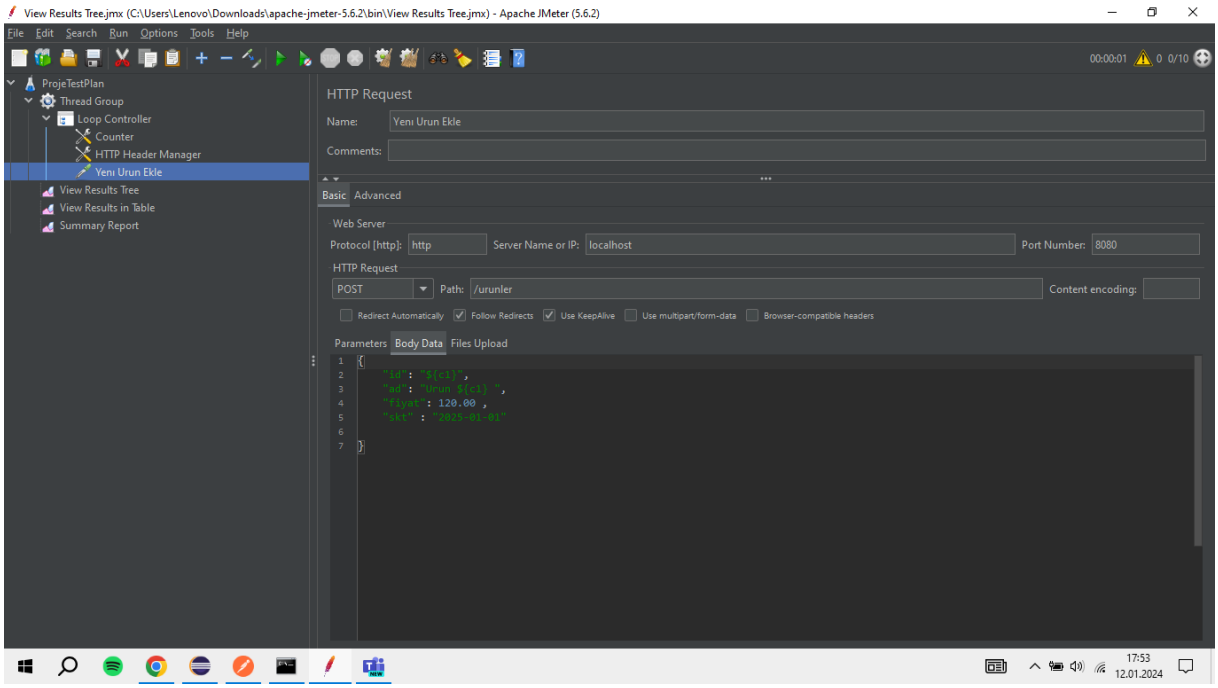
GET İSTEĞİYLE EKLENEN ÜRÜN DOĞRULANDI.



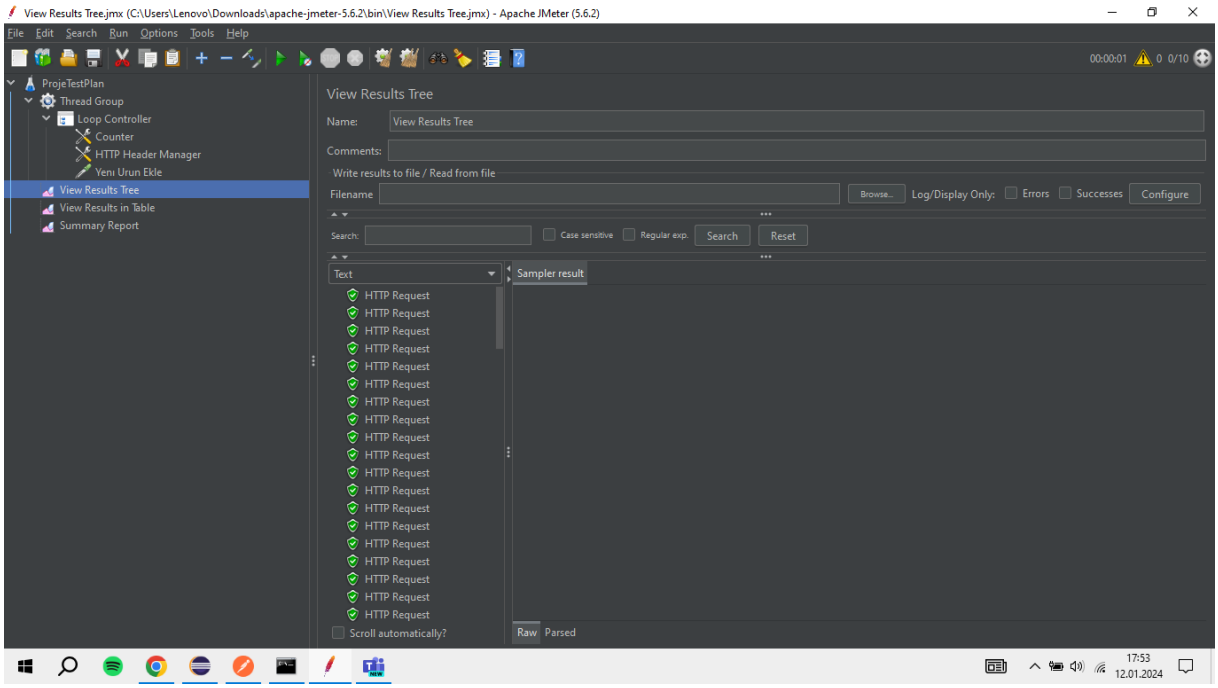
DELETE İSTEĞİYLE ÜRÜN SİLİNDİ.



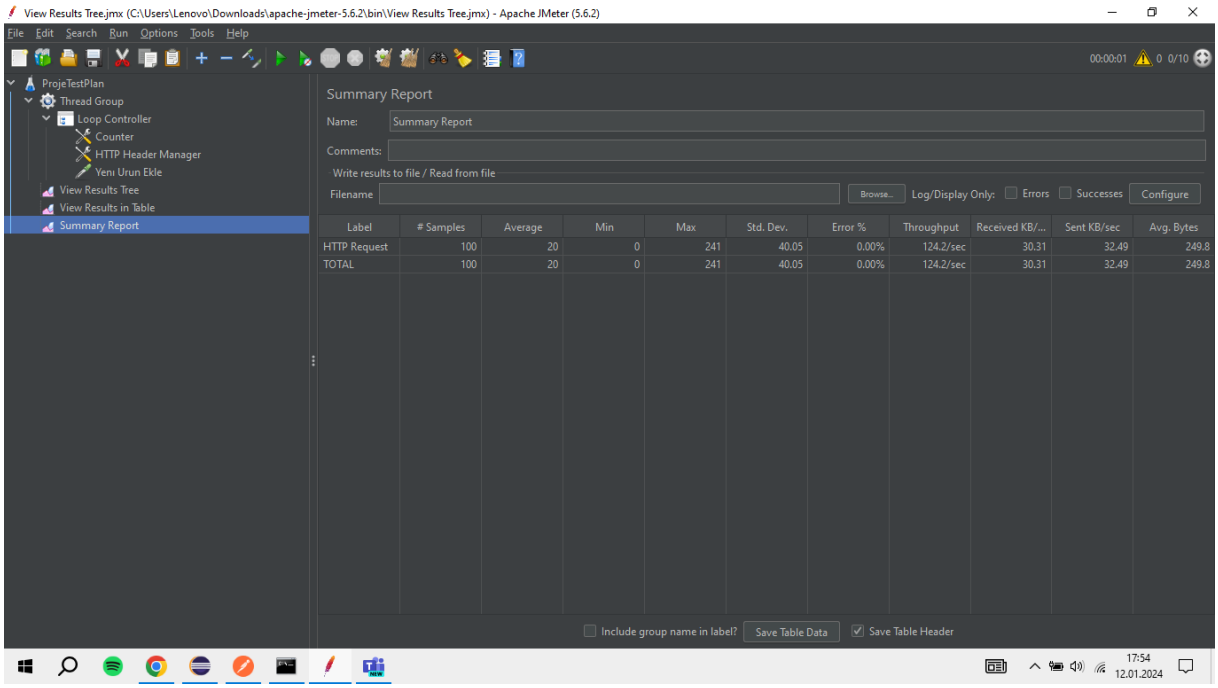
TEKRAR GET İSTEĞİ ATILARAK DELETE İSTEĞİNİN DOĞRULUĞU KONTROL EDİLDİ.



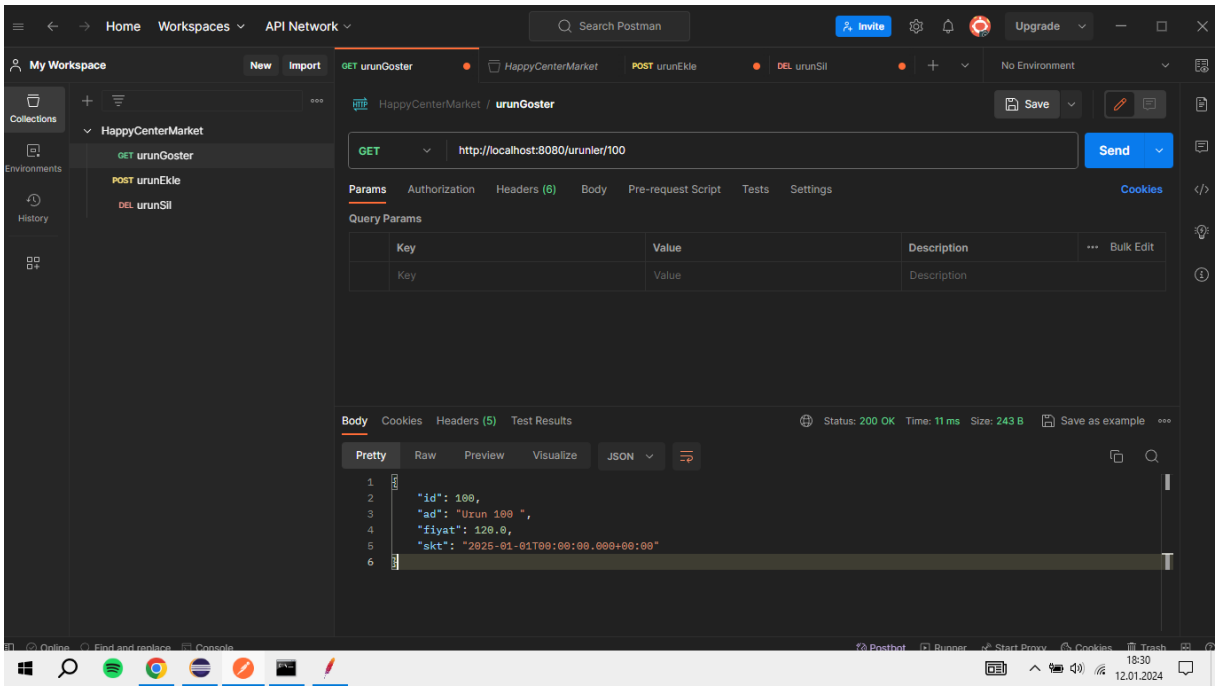
JMETER HTTP POST İSTEĞİ OLUŞTURMA ADIMI



GİDEN İSTEKLERİN DOĞRULUĞU



TOTALDE 100 TANE POST İSTEĞİYLE 100 TANE ÜRÜNÜ BAŞARIYLA OLUŞTURDUK VE BÖYLECE WEB SERVİSİMİZ YAPTIĞIMIZ YÜK TESTİNİ BAŞARIYLA GEÇTİ.



Bu adımdaysa hem urunGosterById metodumuzu hem de JMeter ile yaptığımız yük testi sonucu gerçekten de 100. Ürün oluşmuş mu diye test etmiş olduk.

GIT KULLANIMI

Tüm projemi kodlarımı ve rapor pdf'imi git bash terminali üzerinden komutlarla yolladım sırasıyla kullandıpım komutlar da şöyleydi :

Git init

Git add .

Git commit –m "commit mesajım"

Git remote add origin https://github.com/serhatesen99/FinalProje.git

Git push –u origin master