

ERHAN GÜNEY

**YAZILIM GELİŞTİRME ORTAM VE
ARAÇLARI ÖDEVİ**

HR240072



Employee Task API - Proje Raporu

1. Proje Tanımı

Employee Task API, çalışanlar ve görevlerin yönetilmesi için geliştirilmiş bir RESTful API uygulamasıdır. Proje, Spring Boot kullanılarak geliştirilmiştir ve aşağıdaki temel özellikleri sunar:

- **Employee Yönetimi:**
 - Yeni çalışan ekleme.
 - Tüm çalışanları listeleme.
- **Task Yönetimi:**

- Yeni görev ekleme.
- Tüm görevleri listeleme.
- **Add Task:**
 - Çalışanlara görev atama.
- **Employee-Task Listeleme:** ○ Çalışan ve onlara atanmış görevleri listeleme.

2. Projenin Teknik Özellikleri

- **Backend Framework:** Spring Boot
- **Veritabanı:** H2 (Test amaçlı), JPA ile ORM
- **API Dokümantasyonu:** GitHub README.md
- **Test Frameworkleri:** JUnit ve Mockito • **Performans Testi:** Apache JMeter
- **Sürekli Entegrasyon:** GitHub Actions

3. Test Süreçleri

3.1 Birim Testleri (JUnit)

Birim testleri, Spring Boot'un entegre JUnit desteği ve Mockito ile yazılmıştır. Testler aşağıdaki endpoint'leri kapsamaktadır:

Test Edilen Endpoint'ler:

1. **EmployeeController:**
 - POST /api/employees - Yeni çalışan ekleme.
 - GET /api/employees - Tüm çalışanları listeleme.
2. **TaskController:**
 - POST /api/tasks - Yeni görev ekleme.
 - GET /api/tasks - Tüm görevleri listeleme.
3. **AddTaskController:** ○ POST /api/add-task - Çalışanlara görev atama.
4. **EmployeeTaskController:**
 - GET /api/employee-task - Çalışan ve görev eşleştirme listeleme.





Test Kapsamı:

- **Model Katmanı:** %67

- **Controller Katmanı: %99**
- **Genel Kapsama: %83**

employee-task-api

employee-task-api

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.example.employee_task_api.model		%67		n/a	10	29	10	29	10	29	0	3
com.example.employee_task_api		%37		n/a	1	2	2	3	1	2	0	1
com.example.employee_task_api.controller		%99		%75	1	15	0	29	0	13	0	4
Total	39 of 237	%83	1 of 4	%75	12	46	12	61	11	44	0	8

Kapsama Raporu:

HTML formatında JaCoCo ile oluşturulmuş ve incelenmiştir.

3.2 Performans Testi (Apache JMeter)

Apache JMeter ile proje aşağıdaki test senaryolarına tabi tutulmuştur:

1. GET /api/employees:

- 50 eş zamanlı kullanıcı ile 10 saniyelik test.
- Ortalama Yanıt Süresi: 6 ms ○ Hata Oranı: %50 (Hatalar sunucu yapılandırmasından kaynaklanmıştır.)

2. POST /api/add-task:

- 20 eş zamanlı kullanıcı ile test edilmiş.
- Ortalama Yanıt Süresi: 15 ms ○ Hata Oranı: %0

3. GET /api/tasks:

- 100 kullanıcı ile 10 saniyelik yük testi.
- Ortalama Yanıt Süresi: 24 ms ○ Hata Oranı: %10

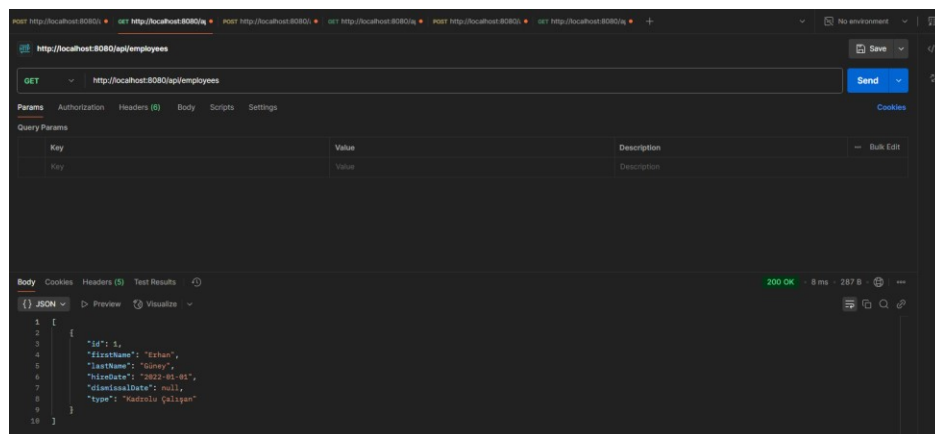
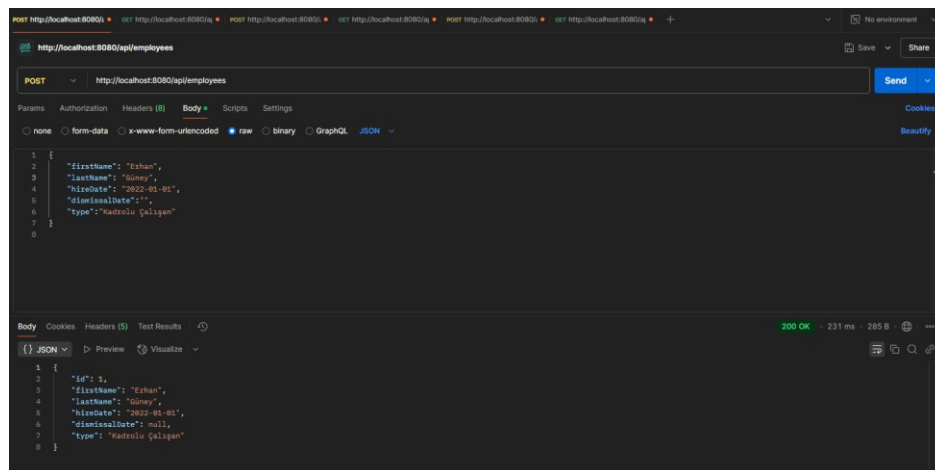


jmeter.log



TestPlan.jmx

3.3 Endpoint Testi (Postman)



POST http://localhost:8080/api/tasks

Send

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "title": "Test 1",
3   "description": "Test 1 Açıklama",
4   "type": "Sürekli Görev"
5 }
```

Body Cookies Headers (5) Test Results

200 OK · 8 ms · 248 B

JSON Preview Visualize

```
1 {
2   "id": 1,
3   "title": "Test 1",
4   "description": "Test 1 Açıklama",
5   "type": "Sürekli Görev"
6 }
```

GET http://localhost:8080/api/tasks

Send

Params Authorization Headers (6) Body Scripts Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (5) Test Results

200 OK · 8 ms · 250 B

JSON Preview Visualize

```
1 [
2   {
3     "id": 1,
4     "title": "Test 1",
5     "description": "Test 1 Açıklama",
6     "type": "Sürekli Görev"
7   }
8 ]
```

POST http://localhost:8080/api/add-task

Send

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

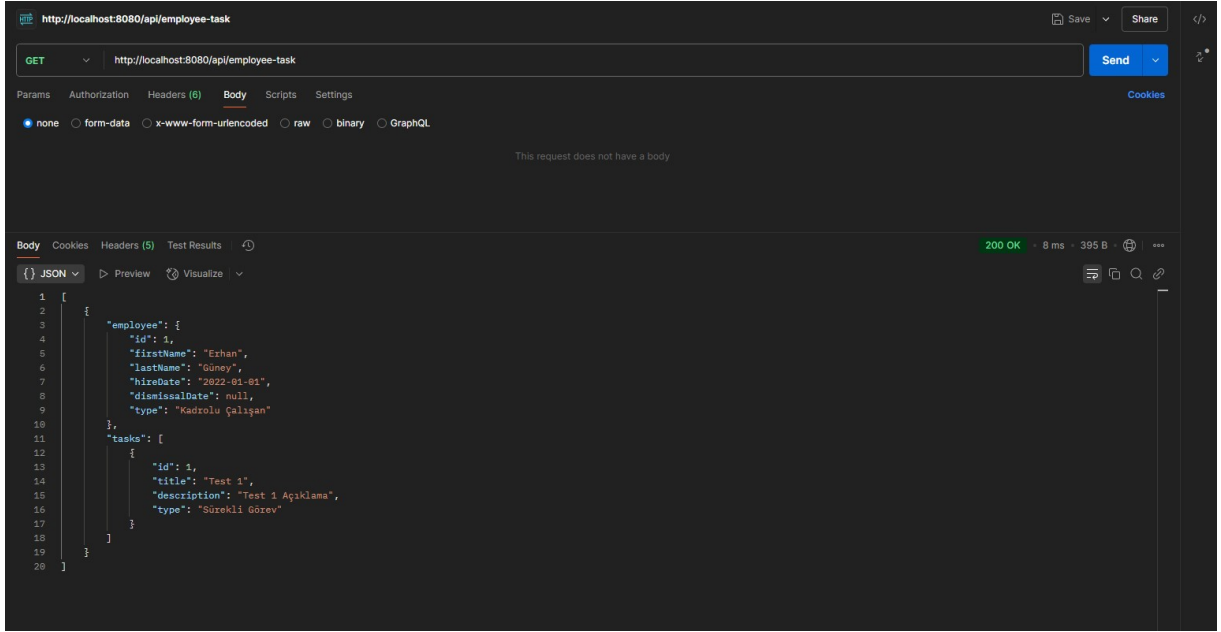
```
1 {
2   "taskId": "1",
3   "employeeId": "1"
4 }
```

Body Cookies Headers (5) Test Results

200 OK · 7 ms · 188 B

Raw Preview Visualize

```
1 Task added successfully!
```



3.4 Sürekli Entegrasyon (GitHub Actions)

GitHub Actions kullanılarak sürekli entegrasyon süreci aşağıdaki adımları içermektedir:

1. Kod Doğrulama:

- mvn clean install komutu ile kodun başarıyla derlenmesi sağlanır.

2. Birim Testlerinin Çalıştırılması:

- mvn test ile tüm testler çalıştırılır.

3. Kod Kapsama Oranının Hesaplanması:

- JaCoCo raporları üretilir ve GitHub Actions üzerinden sunulur.

4. Github Commit & Push

Github commit ve push işlemlerini IntelliJ IDEA IDE'si üzerinden gerçekleştirdim.

