Belediye Çalışan Takibi ve Yönetim Sistemi

1. Giriş

Bu proje, Şile Belediyesi çalışanlarının görevlerini takip etmek ve yönetmek amacıyla geliştirilmiştir. Proje kapsamında, Spring Boot kullanarak RESTful servisler oluşturulmuş, birim testler yazılmış, test kapsamı raporları oluşturulmuş ve sürekli entegrasyon süreçleri uygulanmıştır. Ayrıca, Postman ve JMeter ile testler gerçekleştirilmiş ve sonuçları raporlanmıştır.

2. Proje Yapısı ve Kullanılan Teknolojiler

Kullanılan Teknolojiler

- **Java 17**: Proje geliştirme dili.
- **Spring Boot**: RESTful servislerin oluşturulması için kullanılan framework.
- **Maven**: Proje yönetim ve bağımlılık yönetimi.
- **JUnit**: Birim testlerin yazılması için kullanılan test frameworkü.
- **JaCoCo**: Test kapsamı ölçümü için kullanılan araç.
- **GitHub Actions**: Sürekli entegrasyon ve sürekli teslimat (CI/CD) süreçleri için kullanılan platform.
- **Postman**: API testleri için kullanılan araç.
- **JMeter**: Performans ve yük testleri için kullanılan araç.

Proje Yapısı

Proje, Maven kullanılarak oluşturulmuştur ve aşağıdaki ana bileşenlerden oluşmaktadır:

- **src/main/java**: Uygulama kaynak kodları.
- **src/test/java**: Birim testler.
- **pom.xml**: Maven yapılandırma dosyası.

3. REST Servisleri

Görev Ekleme (POST)

Görev eklemek için kullanılan REST endpoint'i aşağıdaki gibidir:

```
@RestController
@RequestMapping("/tasks")
public class TaskController {
    @Autowired
```

```
    private TaskService taskService;

    @PostMapping
    public ResponseEntity<Task> addTask(@RequestBody Task task) {
        Task newTask = taskService.saveTask(task);
        return new ResponseEntity<>(newTask, HttpStatus.CREATED);
    }
}
```

Service Sınıfı

```
@Service
public class TaskService {
    @Autowired
    private TaskRepository taskRepository;

    public Task saveTask(Task task) {
        return taskRepository.save(task);
    }
}
```

Repository Sınıfı

```
@Repository
public interface TaskRepository extends JpaRepository<Task, Long> {
}
```

Görev Listeleme (GET)

Görevleri listelemek için kullanılan REST endpoint'i aşağıdaki gibidir:

Controller Sınıfı

```
@RestController
@RequestMapping("/tasks")
public class TaskController {
    @Autowired
    private TaskService taskService;

    @GetMapping
    public ResponseEntity<List<Task>> getAllTasks() {
        List<Task> tasks = taskService.getAllTasks();
        return new ResponseEntity<>(tasks, HttpStatus.OK);
    }
}
```

Service Sınıfı

```
@Service
public class TaskService {
    @Autowired
    private TaskRepository taskRepository;

    public List<Task> getAllTasks() {
        return taskRepository.findAll();
    }
}
```

4. Birim Testler

JUnit Testleri

Birim testler JUnit frameworkü kullanılarak yazılmıştır. Örnek birim test aşağıdaki gibidir:

```
@SpringBootTest
public class TaskServiceTest {
    @Autowired
    private TaskService taskService;

    @MockBean
    private TaskRepository taskRepository;

    @Test
    public void testSaveTask() {
        Task task = new Task("New Task", "Task Description");
        Mockito.when(taskRepository.save(task)).thenReturn(task);

        Task savedTask = taskService.saveTask(task);
        assertEquals(task.getName(), savedTask.getName());
    }

    @Test
    public void testGetAllTasks() {
        Task task1 = new Task("Task 1", "Description 1");
        Task task2 = new Task("Task 2", "Description 2");
        List<Task> tasks = Arrays.asList(task1, task2);

        Mockito.when(taskRepository.findAll()).thenReturn(tasks);

        List<Task> retrievedTasks = taskService.getAllTasks();
        assertEquals(2, retrievedTasks.size());
    }
}
```

Test Kapsama Raporu

Test kapsama raporu, JaCoCo kullanılarak oluşturulmuştur. Maven yapılandırma dosyasına (pom.xml) eklenen JaCoCo plugin'i ile test kapsamı raporu oluşturulmuştur.

pom.xml Dosyasına Eklenen Plugin

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.22.2</version>
    </plugin>
    <plugin>
      <groupId>org.jacoco</groupId>
      <artifactId>jacoco-maven-plugin</artifactId>
      <version>0.8.7</version>
      <executions>
        <execution>
          <goals>
            <goal>prepare-agent</goal>
          </goals>
        </execution>
        <execution>
          <id>report</id>
          <phase>test</phase>
          <goals>
            <goal>report</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

Maven ile Test Kapsama Raporu Oluşturma

```
mvn clean test
```

Bu komut çalıştırıldıktan sonra, target/site/jacoco/index.html dosyasında test kapsamı raporu oluşturulur. Aşağıda, test kapsamı raporunun ekran görüntüsü bulunmaktadır:

5. Sürekli Entegrasyon (CI)

GitHub Actions kullanılarak sürekli entegrasyon süreci yapılandırılmıştır. Bu süreç, kodun her push işleminde otomatik olarak test edilmesini ve test kapsamı raporlarının oluşturulmasını sağlar.

GitHub Actions Workflow Dosyası

.github/workflows/maven.yml dosyası oluşturularak aşağıdaki yapılandırma eklenmiştir:

```
name: Java CI with Maven



on:

  push:
```

```
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up JDK 17
      uses: actions/setup-java@v2
      with:
        java-version: '17'
        distribution: 'adopt'

    - name: Build with Maven
      run: mvn clean install

    - name: Run tests with Maven
      run: mvn test

    - name: Generate JaCoCo report
      run: mvn jacoco:report
```

6. Git ve GitHub Kullanımı

Proje, Git ile versiyonlanmış ve GitHub repository'sine gönderilmiştir. Visual Studio Code üzerinden aşağıdaki adımlar izlenerek GitHub'a gönderilmiştir:

Git İle Versiyonlama

```
git init
git add .
git commit -m "Initial commit"
git remote add origin https://github.com/BerkeALTINKAYNAK/SileFinal.git
git push -u origin main
```

GitHub Repository'sine Erişim

Proje dosyaları GitHub repository'sinde şu adreste bulunmaktadır: GitHub Repository

7. Postman & JMeter Testleri

Postman Testleri

Postman kullanılarak API testleri gerçekleştirilmiştir. Görev ekleme ve listeleme işlemleri Postman ile test edilmiştir.

- Görev Ekleme (POST) İsteği:
  - URL: `http://localhost:8080/tasks`
  - Metod: POST
  - Body:
  - {
  - "name": "Test Task",
  - "description": "This is a test task"
  - }
  -

- Görev Listeleme (GET) İsteği:
  - URL: `http://localhost:8080/tasks`
  - Metod: GET

JMeter Testleri

JMeter kullanılarak performans ve yük testleri gerçekleştirilmiştir. JMeter ile çeşitli yük senaryoları oluşturularak API'nin performansı test edilmiştir.

8. Sonuç ve Değerlendirme

Bu proje kapsamında, Şile Belediyesi için bir çalışan görev takip ve yönetim sistemi geliştirilmiştir. Spring Boot kullanarak RESTful servisler oluşturulmuş, birim testler yazılmış ve test kapsamı raporları oluşturulmuştur. GitHub Actions kullanılarak sürekli entegrasyon süreci yapılandırılmış ve Postman ve JMeter ile API testleri gerçekleştirilmiştir.