

Yazılım Lab 2

Proje 1

Gökhan Çobanoğlu
200201015
Bilgisayar Mühendisliği

Bilal Cihangir Erdem
200201076
Bilgisayar Mühendisliği



Fig. 1: Web Sayfası

I. ÖZET

Bu projede bizden kitaplardan veya farklı yazılı metinlerden alınabilecek cümleler arasındaki benzerliklerin bulunması ve bulunan benzerliklere göre verilen metinlerin birleştirilmesi ve metinlerin bir web arayüzünde gösterilmesi beklenmektedir.

Projede metinlerin birleştirilmesi için kullanılacak programlama dili olarak java istenmiştir.

II. GİRİŞ

Spring Framework java için geliştirilmiş olan açık kaynak kodlu bir uygulama geliştirme arayüzüdür. Genel özellikleri herhangi bir java uygulaması tarafından kullanılabilir olsa bile içerdiği çok sayıda farklı eklentilerle beraber kurumsal Java platformu üzerinde web uygulamaları tasarlamak için de kullanılabilir.

Biz projemizin içerdiği backend ve frontend kısımlarını birleştirmek için Spring Framework kullanışlı olduğu görüldü ve kullanıldı.

Projedeki web arayüzünü tasarlamak için HTML kullanıldı. Kullanıcı istediği sayıda metni bu html sayfasında girebilir bu metinleri birleştirebilir veya kaydedebilir. Metinler birleştirilmesi için Html sayfasından javaya aktarılır javadaki birleştirme ve geçen süre hesaplanması işlemlerinden sonra kullandığımız veritabanına aktarılır. Aktarıldıktan sonra yapılan birleştirme işlemi sonucu oluşan metin ve birleştirildiği süre html sayfasında kullanıcıya gösterilir.

Projede veritabanı olarak NoSQL olan MongoDB veritabanı kullanılmıştır.

Metin Birleştirme Uygulaması

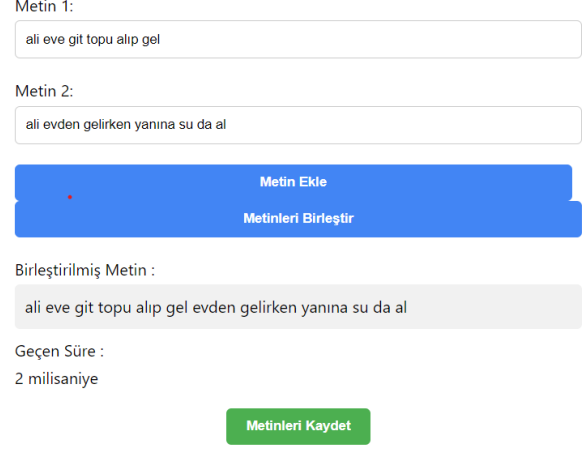


Fig. 2: Web Sayfası

III. YÖNTEM

Öncelikle Java uygulamamızın içindeki classların neler yaptığından bahsedelim.

MergedTextContoller: Bu Controller class kendi içinde bir başka class olan MergedTextService classını da bir değişken olarak çağırıp onun methodlarını kullanmaktadır. Kendisi ise web sayfasındaki değişiklikleri ve butonlara basılmasını algılayıp buna göre farklı methodlar çağırır, daha önce birleştirdiğiniz metinleri json olarak görebilmenizi sağlayacak bir method da içermektedir. Bu classın işi web sitesindeki dataları javanın içinde işlemlerde kullanabilmemiz için aktarmaktır.

MergedTextRepository: MongoRepository adındaki sabit bir kütüphanenin methodlarını override yapan bir "interface"dir. Böylece mongo methodları için sadece bu interface'i kullanacağımız classa dahil etmemiz yeterlidir. Web sitesinden getirdiğimiz dataları kaydetmemiz için gerekli olan methodları bize sağlayacaktır.

MergedTextDto: Web sayfamızdan Liste olarak çekmemizi ve böylece methodlara bu listeyi göndererek işlem kalabalığını azaltmamızı sağlayan bir data classıdır. Ayrıca kullanıcı gireceği metin sayısını kendisi belirleyecek bu yüzden dinamikliği sağlamak için metinleri bir liste ile almamız gereklidir.

```

public MergedText printMergedText(MergedTextDto mergedTextDto) {
    long startTime = System.currentTimeMillis();

    MergedText mergedText = new MergedText();
    mergedText.setText(birlestir(mergedTextDto.getText()));
    mergedText.setId(nextSequenceService.getNextSequence("customSequences"));

    long endTime = System.currentTimeMillis();
    long duration = endTime - startTime;
    mergedText.setTime(duration);

    return mergedText;
}

```

Fig. 3: Kod Örneği

```

@RestController
@RequestMapping("merged-text")
@RequiredArgsConstructor
@CrossOrigin
public class MergedTextController {

    3 usages
    private final MergedTextService mergedTextService;

    no usages
    @GetMapping
    public ResponseEntity<List<MergedText>> getAll() { return ResponseEntity.ok(mergedTextService.getAll()); }

    no usages
    @PostMapping
    public ResponseEntity<MergedText> printMergedText(@RequestBody MergedTextDto mergedTextDto) {
        return ResponseEntity.ok(mergedTextService.printMergedText(mergedTextDto));
    }

    no usages
    @PostMapping("/save")
    public ResponseEntity<String> saveMergedText(@RequestBody MergedText mergedText) {
        return ResponseEntity.ok(mergedTextService.saveMergedText(mergedText));
    }
}

```

Fig. 4: Kod Örneği 2

MergedText: Datalarımızı veritabanına kaydetmek için bu classın biçiminde ayarlamamız gereklidir. MongoDB bir NoSQL veritabanı olduğundan class içindeki özellikleri koleksiyonundaki bir özellik olarak algılayıp kaydedecektir. Bu class yapısı birleştirilmiş metni, birleştirildiği zamanı ve kaydedilen datanın idsini tutar.

MergedTextService: Uygulamanın asıl projemizde bizden istenen işlemini yapan Service classıdır. Bu class içinde MergedTextRepository interface'ini de bir değişken olarak çağırıp MongoDB methodlarını kullanabiliyor. Bu class içindeki printMergedText methodu yeni bir class çağırıp girdi olarak aldığı metinleri birleştirme fonksiyonunu çağırır. Çağırmanın sonrasındaki zaman değerini öncesindeki zaman değerinden çıkartarak birleştirme işleminin ne kadar sürdüğünü hesaplar. Birleştirme algoritması cümleleri küçük harflere çevirir ve kelimelere böler. Ardından kelimeler kendi aralarında kıyaslanır. Örnek olarak girilen 2. metnin ilk kelimesi hem 2. metin ile hem de diğer metinlerin kelimeleri ile tek tek kıyaslanır kıyaslandığı diğer kelime kendisinin sahip olduğu harfleri aynı sırayla içeriyorsa birleştirilen metne eklenmez. Bu classın saveMergedText adlı diğer fonksiyonu ise bahsedilen fonksiyonlarda hesaplanan birleştirilmiş metni ve hesaplanma süresini MongoDB veritabanında merged-text adlı bir koleksiyona kaydeder eğer metin girilmemişse hata döndürür.

```

@Document(collection = "merged_texts")
@Data
public class MergedText {

    no usages
    @Transient
    public static final String SEQUENCE_NAME = "merged_texts_sequence";

    no usages
    @Id
    private int id;

    no usages
    private String text;

    no usages
    private long time;
}

```

Fig. 5: Kod Örneği 3

```

1 usage
public static String birlestir(List<String> cumleler) {
    Map<String, Integer> kelimeSayilari = new HashMap<>();
    for (String cumle : cumleler) {
        String[] kelimeler = cumle.toLowerCase().split(" ");
        for (String kelime : kelimeler) {
            kelimeSayilari.put(kelime, kelimeSayilari.getOrDefault(kelime, 0) + 1);
        }
    }

    StringBuilder birlesmisCumle = new StringBuilder();
    Set<String> birlestirilmisKelimeler = new HashSet<>();
    for (String cumle : cumleler) {
        String[] kelimeler = cumle.toLowerCase().split(" ");
        for (String kelime : kelimeler) {
            if (kelimeSayilari.get(kelime) > 1) {
                if (birlestirilmisKelimeler.contains(kelime)) {
                    continue;
                }
                birlestirilmisKelimeler.add(kelime);
            }
            birlesmisCumle.append(kelime).append(" ");
        }
    }

    return birlesmisCumle.toString().trim();
}

```

Fig. 6: Kod Örneği 4

IV. DENEYSEL SONUÇLAR

REFERENCES

- [1] <https://www.geeksforgeeks.org/count-of-words-that-are-present-in-all-the-given-sentences/>
- [2] <https://www.javatpoint.com/java-program-to-find-the-most-repeated-word-in-a-text-file>
- [3] <https://www.java67.com/2016/09/3-ways-to-count-words-in-java-string.html>
- [4] <https://www.quora.com/How-do-you-check-for-common-words-from-two-strings-Java-string-development>
- [5] <https://stackoverflow.com/questions/53555210/what-would-be-an-efficient-way-to-find-the-common-words-in-3-files-in-java>
- [6] <https://www.youtube.com/watch?v=a3tFsL6vSJA>
- [7] <https://stackoverflow.com/questions/19049785/merge-two-text-files-line-by-line-using-java>
- [8] <https://www.youtube.com/watch?v=v9-3u5Hd8Q&list=WLIindex=24>



Fig. 7: MongoDB

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-boot-starter</artifactId>
    <version>3.0.0</version>
  </dependency>
</dependencies>
```

Fig. 8: Spring Eklentileri

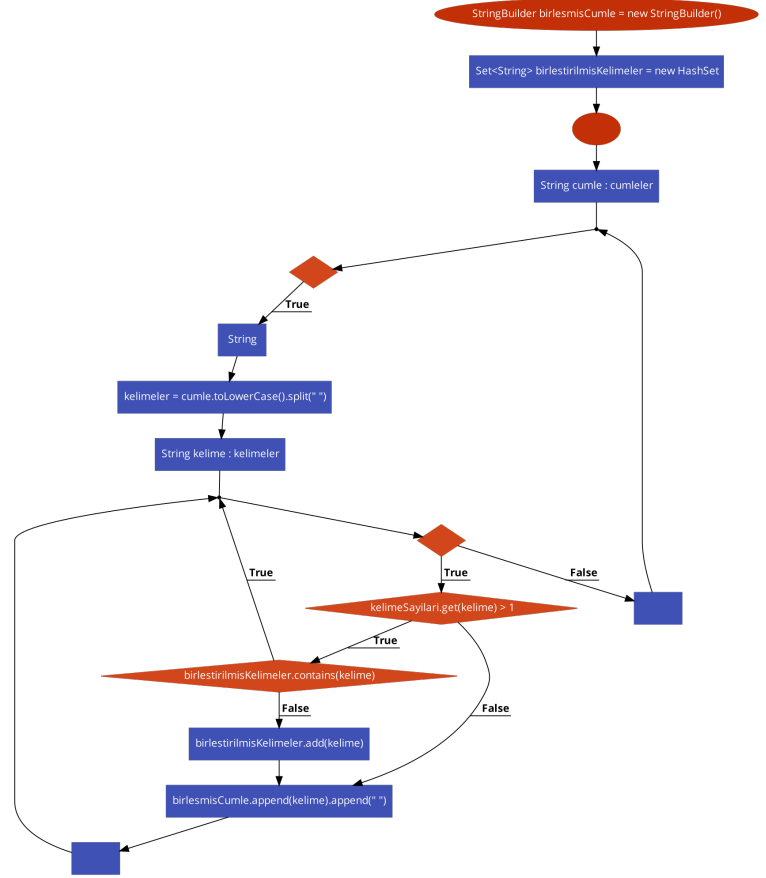


Fig. 9: Akış Şeması

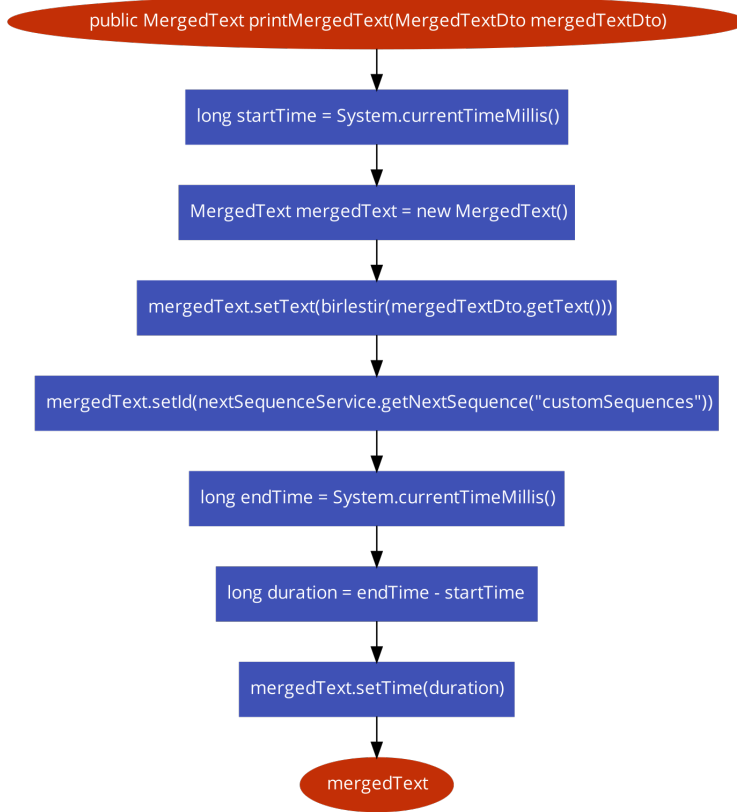


Fig. 10: Akış Şeması 2

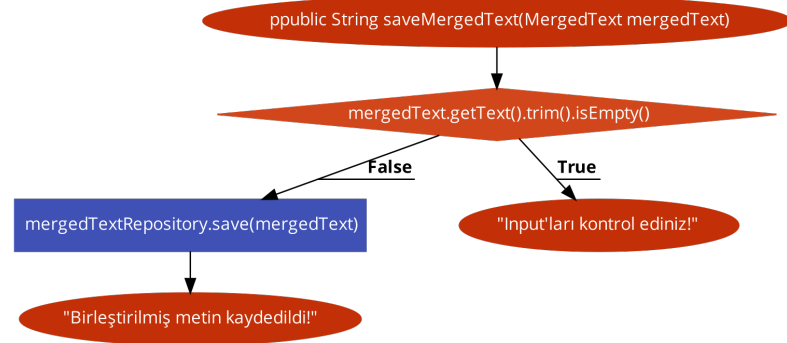


Fig. 12: Akış Şeması 4

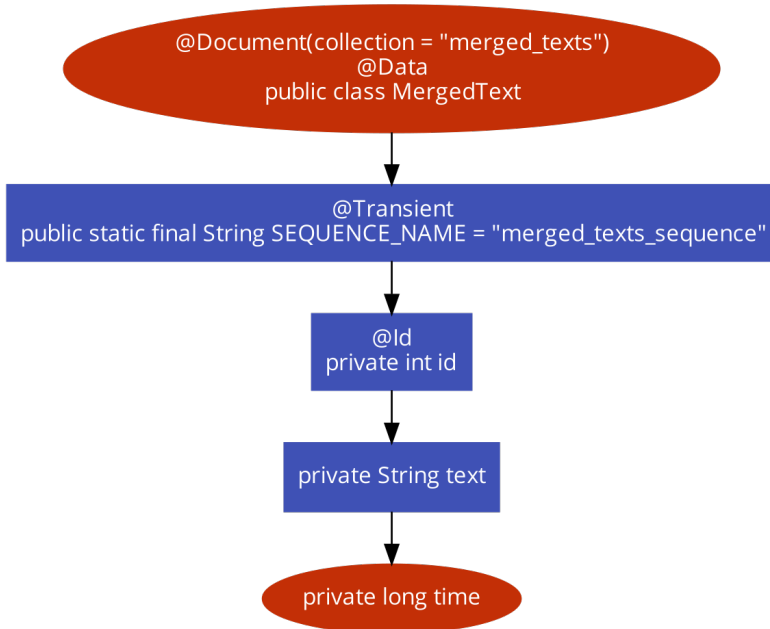


Fig. 11: Akış Şeması 3