BAB IV
IMPLEMENTASI dan PEMBAHASAN

4.1. Model Data

Tujuan utama dari *scraping* adalah mendapatkan data terstruktur dari sumber data yang tidak terstruktur. Model data dibutuhkan dalam menentukan struktur data yang akan di ekstrasi. Data yang ada pada struktur data mengikuti data yang dibutuhkan pada analisa kebutuhan data. Data yang dibutuhkan dan yang akan diekstraksi dari website dapat dilihat pada *Tabel 4- 1*.

Tabel 4- 1. Tabel data ekstraksi

No	Data	Tipe Data
1.	Nama Hotel	Text
2.	Bintang Hotel	Decimal
3.	Alamat Hotel	Text
4.	Rating Hotel	Decimal
5.	Rating Review	Decimal
6.	Tanggal Review	Datetime
7.	Name Review	Text
8.	Tema Review	Text
9.	Teks Review	Text

Model data ini diimplementasikan dalam *class HotelReview*. *Class HotelReview* merupakan turunan dari *class scrapy.Item* dan ditempatkan pada *file items.py*. Di dalam *items.py* terdapat *class HotelReview* yang berfungsi untuk memetakan data yang telah di *scraping* untuk disalurkan ke *pipelines* dan disimpan kedalam *elasticsearch*.

```
class HotelReview(scrapy.Item):
    hotelName = scrapy.Field()
    hotelStar = scrapy.Field()
    hotelAddress = scrapy.Field()
    hotelRating = scrapy.Field()
    reviewRating = scrapy.Field()
    reviewDate = scrapy.Field()
    reviewName = scrapy.Field()
    reviewTheme = scrapy.Field()
    reviewText = scrapy.Field()
```

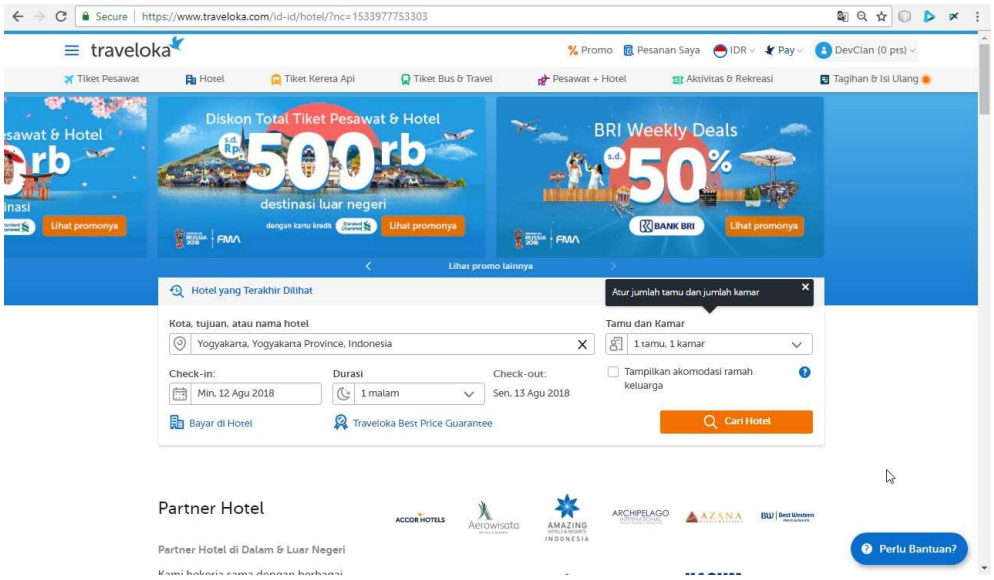
4.2. Analisa Web Traveloka

Website Traveloka perlu dianalisa untuk mengetahui hubungan atau *link* antar halaman dan letak data yang akan diekstrasi. Hubungan *link* antara halaman diperlukan untuk melakukan proses *crawling*. Sedangkan letak data yang akan dikestrasi diperlukan untuk melakukan proses *scraping*.

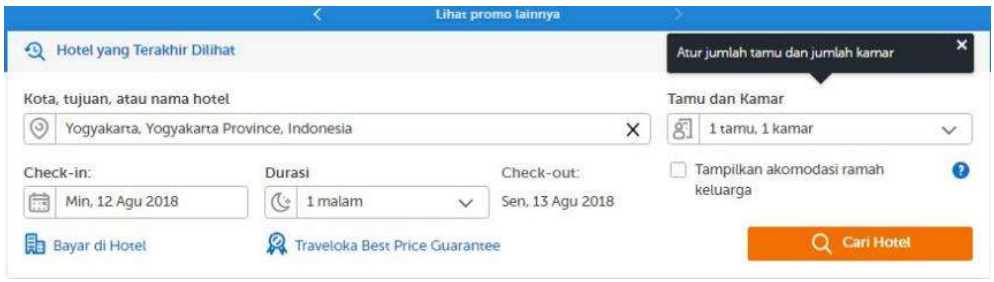
Ada 3 halaman utama yang menjadi perhatian dan letak sumber data. Pertama halaman *home* sebagai awal akses ke web Traveloka, kedua halaman hotel *list* dimana terdapat daftar semua *link* hotel, dan ketiga adalah halaman hotel dimana terdapat *review* hotel. Hubungan antar 3 halaman tersebut dapat dilihat pada Gambar 4-1.

Gambar 4-1. Struktur web Traveloka

Ketika membuka *website* Traveloka dengan *url* https://www.traveloka.com/id-id/hotel , halaman yang tampil adalah halaman pencarian hotel seperti paga Gambar 4-2.

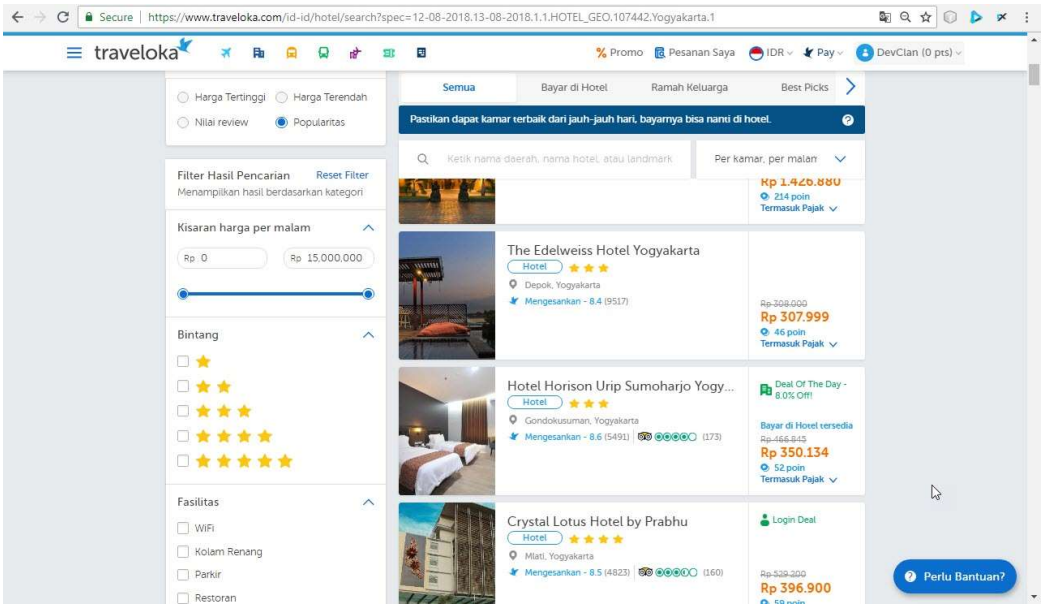


Gambar 4-2. Halaman pencarian hotel



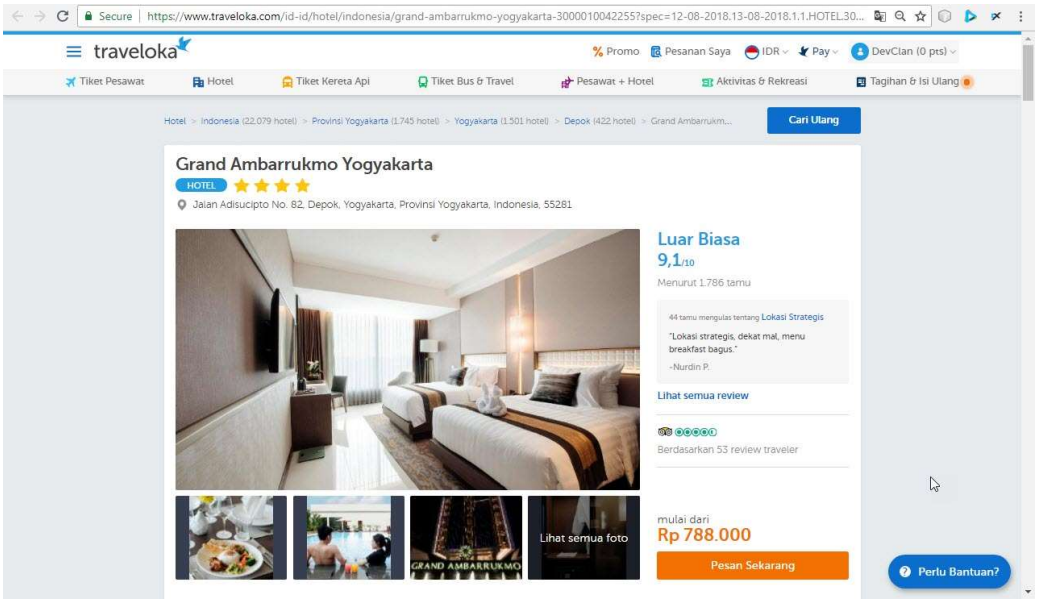
Gambar 4-3. *Form* pencarian hotel

Pada halaman pencarian hotel, *form* pencarian harus diisi untuk mendapatkan daftar hotel yang sesuai dengan yang diinginkan. *Form* pencarian hotel dapat dilihat pada Gambar 4-3. Isi *form* pencarian dengan mengganti nama lokasi menjadi Yogyakarta dan tekan *button* Cari Hotel, maka akan menampilkan daftar hotel yang ada di Yogyakarta seperti pada Gambar 4-4.



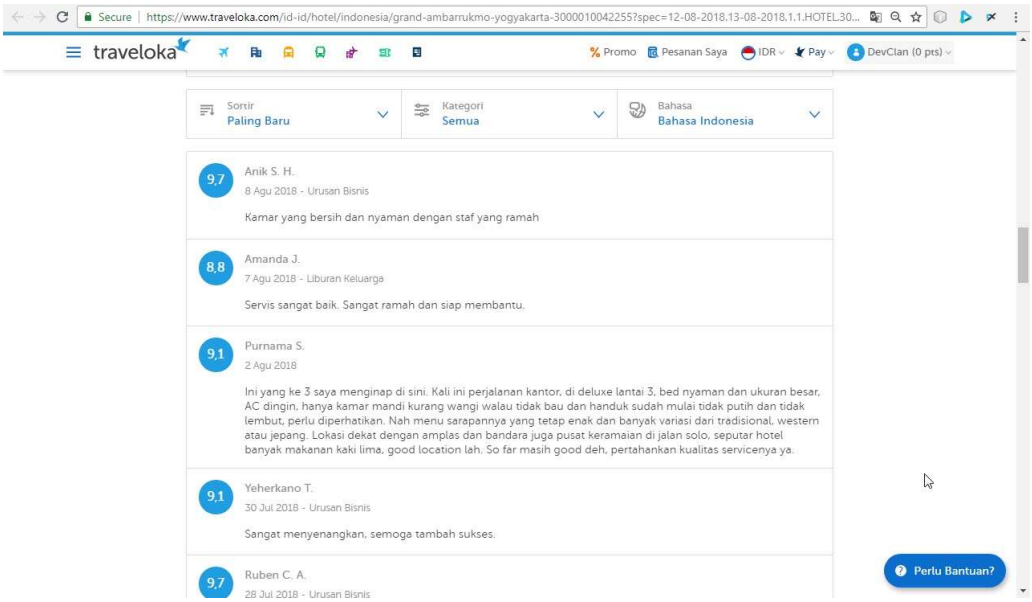
Gambar 4-4. Halaman daftar hotel

Setelah itu untuk dapat melihat *review* masing-masing hotel, harus masuk kedalam halaman hotel tertentu. Klik bagian salah satu hotel dalam daftar hotel dan akan muncul halaman hotel seperti pada Gambar 4-5.



Gambar 4-5. Halaman hotel Traveloka

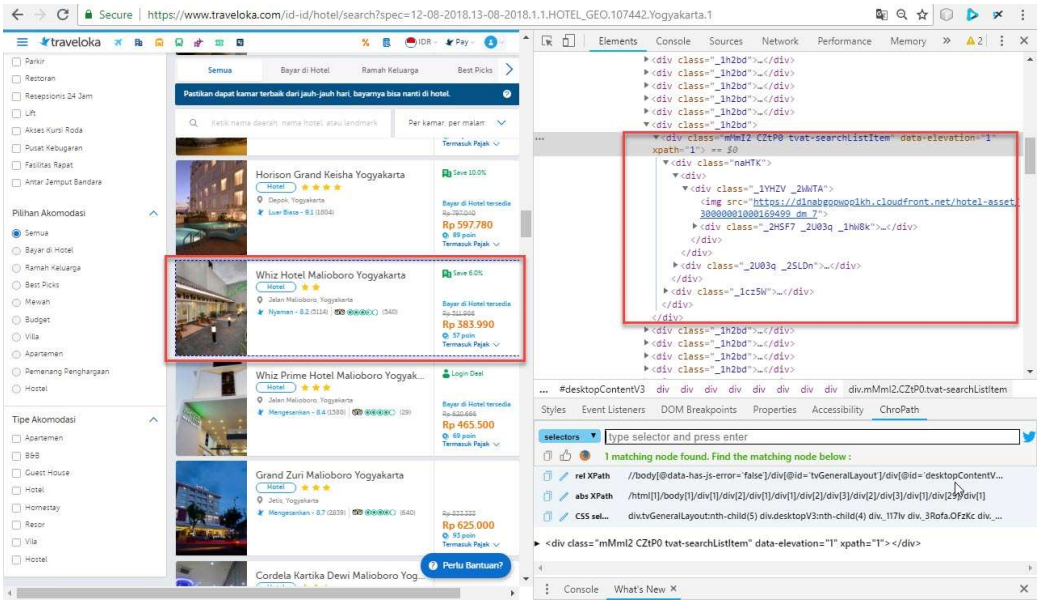
Review dari sebuah hotel terdapat pada bagian bawah pada setiap halaman hotel seperti Gambar 4-6.



Gambar 4-6. *Review* hotel Traveloka

Pada halaman web Traveloka bagian daftar hotel dan daftar *review* pada tiap hotel merupakan data *Ajax*. Sebelum melakukan ekstraksi harus ditentukan elemen mana saja yang didalamnya terdapat data yang ingin diekstraksi. Untuk memilih elemen-elemen pada teks *html*, diperlukan sebuah *selector*. *Selector* yang akan digunakan adalah *Xpath selector*.

Untuk mendapatkan *review* dari masing-masing hotel, maka halaman dari tiap-tiap hotel harus dibuka terlebih dahulu. Untuk membuka halaman hotel pada daftar hotel melalui *selenium* diperlukan elemen dari masing-masing hotel seperti pada Gambar 4-7.

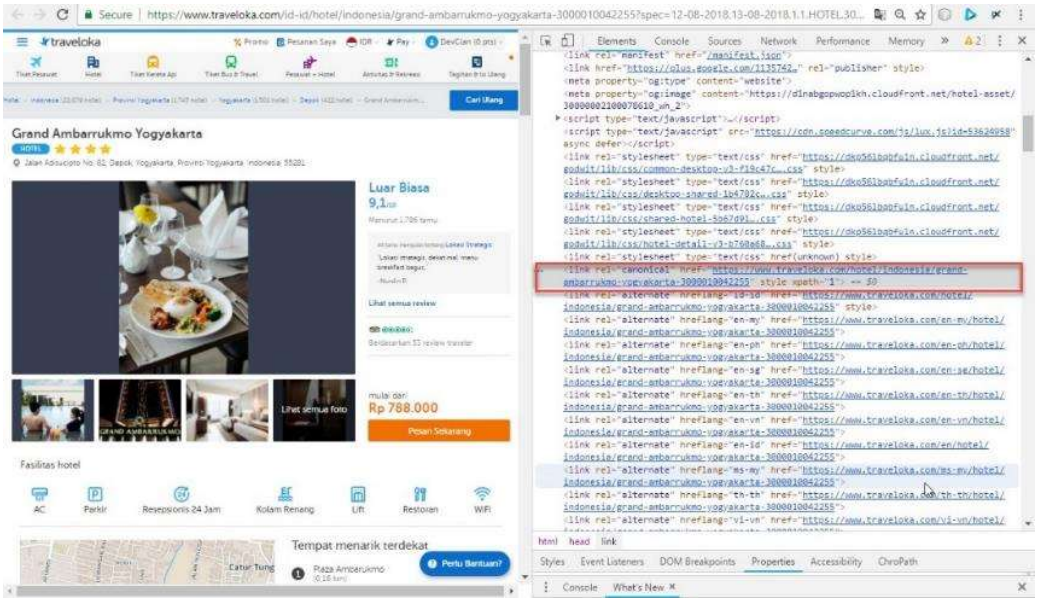


Gambar 4-7. Elemen *url* hotel

Gambar 4-8 merupakan elemen yang digunakan untuk membuka halaman pada masing-masing hotel.

Gambar 4-8. Elemen membuka halaman hotel

Setelah masuk pada halaman hotel, maka yang diperlukan adalah mengambil *link url* dari hotel tersebut tanpa menggunakan tanggal pencarian hotel. Untuk mengambil link url tersebut elemen yang digunakan dapat dilihat pada Gambar 4-9.

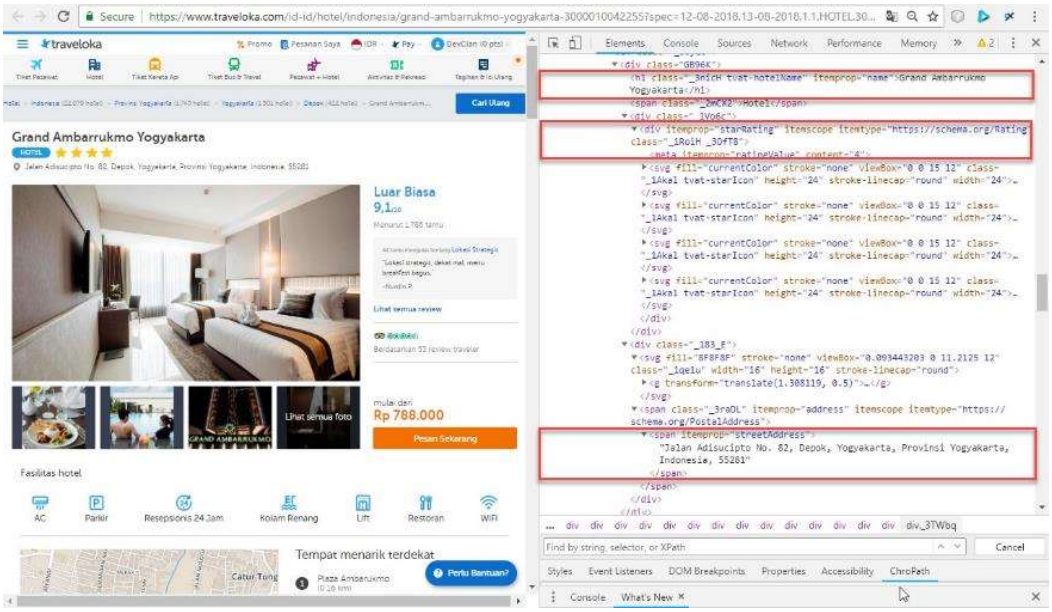


Gambar 4-9. Elemen *url* hotel

```
<link rel="canonical" href="https://www.traveloka.com/hotel/indonesia/grand-ambarrukmo-yogyakarta-3000010042255" style xpath="1"> == $0
```

Gambar 4-10. Elemen *canonical*

Elemen *Canonical* pada Gambar 4-10. Merupakan elemen yang terdapat *link url* hotel tanpa kata kunci pencarian atau tanggal pencarian. Selanjutnya, setelah link hotel yang telah di dapat itu dibuka, muncul halaman hotel yang akan diekstraksi datanya. Untuk mengekstraksi halaman hotel juga perlu dipilih elemen-elemen yang akan digunakan seperti pada Gambar 4-11.



Gambar 4-11. Elemen data hotel

```
▼<div class="GB96K">
   <h1 class="_3nicH tvat-hotelName" itemprop="name">Grand Ambarrukmo
   Yogyakarta</h1>
```

Gambar 4-12. Elemen nama hotel

```
  <div class="_3TWbq" style>9,1</div>
</div>
```

Gambar 4-13 Elemen *rating* hotel

Gambar 4-12 merupakan elemen untuk nama hotel, sedangkan Gambar 4-13 merupakan elemen yang digunakan untuk menggambil *rating* hotel.

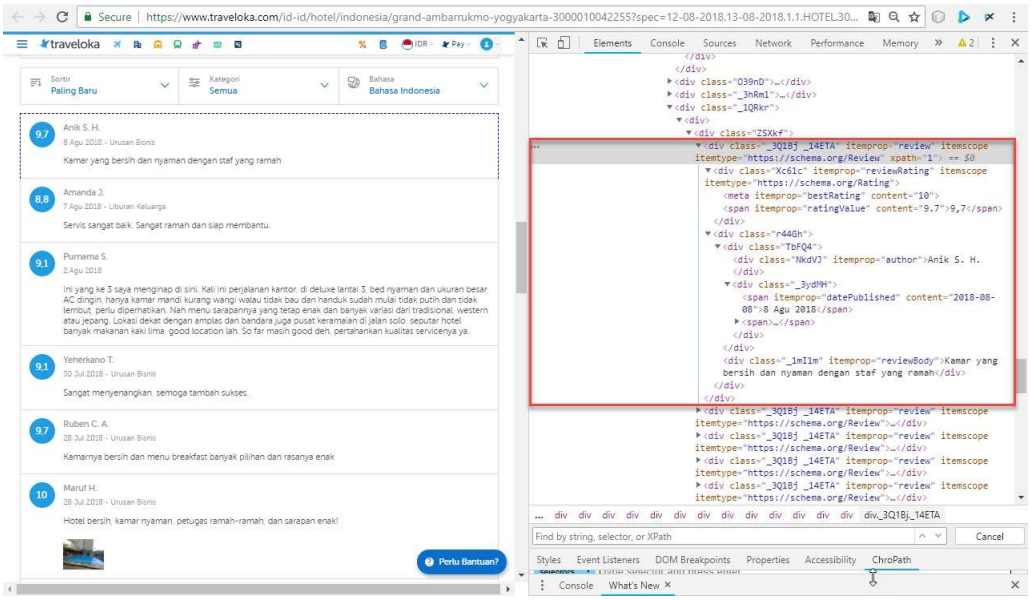
```
▼<div itemprop="starRating" itemscope itemtype="https://schema.org/Rating"
class="_1RoiH _3OfT8">
   <meta itemprop="ratingValue" content="4">
```

Gambar 4-14. Elemen bintang hotel

```
▼<span itemprop="streetAddress">
   "Jalan Adisucipto No. 82, Depok, Yogyakarta, Provinsi Yogyakarta,
   Indonesia, 55281"
 </span>
```

Gambar 4-15. Elemen alamat hotel

Pada Gambar 4-14 menunjukan elemen untuk bintang hotel dan Gambar 4-15 adalah elemen untuk mengambil alamat hotel. Selanjutnya *scroll* kebawah pada halaman hotel dan akan menampilkan daftar *review* hotel seperti Gambar 4-16.

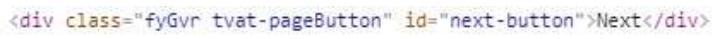


Gambar 4-16. Elemen *review* hotel

Semua data yang terdapat pada review hotel ada di dalam elemen *review* seperti pada Gambar 4-17. Di dalam elemen *review* terdapat data berupa nama *review*, rating *review*, tanggal *review*, tema *review* dan text *review*.

Gambar 4-17. Elemen *review*

Pada bagian *ajax* yang terledak di daftar *list* hotel dan daftar *review* pada tiap hotel, terdapat *button next* yang akan di *crawling* oleh *scrapy* untuk membuka list berikutnya. Elemen pada *button next* dapat dilihat pada Gambar 4-18.



Gambar 4-18. Elemen *button next*

Semua elemen-elemen tersebut akan digunakan dalam proses *scraping* dan *crawling* untuk membuka halaman hotel, memilih data-data yang akan diekstraksi dan untuk melakukan melakukan *crawling* pada *list* data berikutnya yang akan diekstraksi.

X*path* untuk membuka halaman hotel, elemen canonical dan proses crawling dapat dilihat pada Tabel 4- 2. Sedangkan xpath untuk data yang akan diekstraksi dapat dilihat pada Tabel 4- 3.

Tabel 4- 2. Tabel *XPath* 1

No	Nama	XPath
1.	Untuk membuka halaman hotel	//div[@class='mMmI2 CZtP0 tvat-searchListItem']
2.	Elemen Canonical	//link[@rel='canonical']

3.	Button-next	//div[@id='next-button']

Tabel 4- 3. Tabel *XPath* 2

No	Data	(XPath)
1.	Nama Hotel	//h1[@itemprop='name']
2.	Bintang Hotel	//meta[@itemprop='ratingValue']
3.	Alamat Hotel	//span[@itemprop='streetAddress']
4.	Rating Hotel	//div[@class='_3TWbq']
5.	Rating Review	//div[@itemprop='review']
6.	Tanggal Review	//div[@itemprop='review']
7.	Name Review	//div[@itemprop='review']
8.	Tema Review	//div[@itemprop='review']
9.	Teks Review	//div[@itemprop='review']

4.3. Instalasi Scrapy Framework

Setelah mengetahui struktur dari *website* Traveloka, proses instalasi semua modul dilakukan ditahap ini. Implementasi menggunakna bahasa python, sehingga semua modul di-*install* di dalam *PIP* (*Python Package Management*). Sebelum melakukan instalasi pada *scrapy*, *install extention python* terlebih dahulu pada *Visual Studio Code*. *Python* yang digunakan adalah *python 3.6*. Scrapy di *install* pada terminal *Visual Studio Code* dengan perintah :

```
Pip install Scrapy
```

Selain melakukan instalasi pada *scrapy*, instalasi *selenium* dan *elasticsearch* juga dilakukan pada tahap ini dengan perintah :

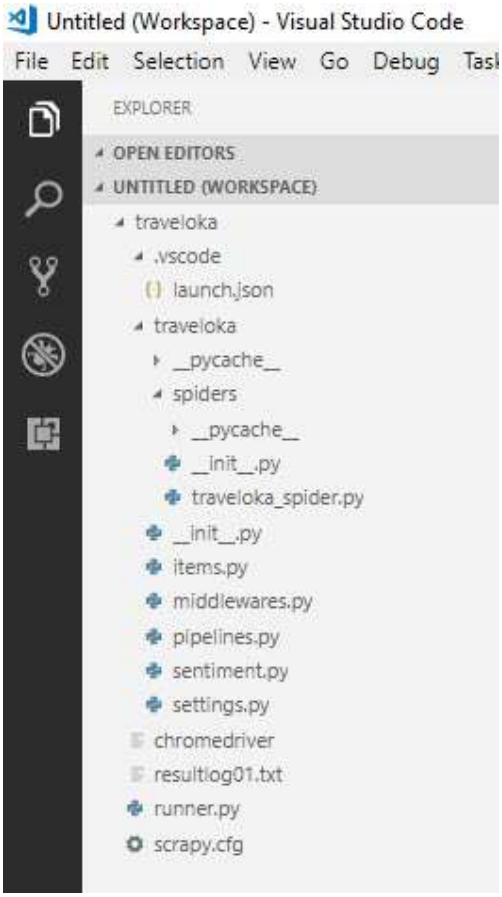
```
Pip install Scrapy-Selenium
Pip install ScrapyElasticSearch
```

Setelah semua modul di *install*, maka yang selanjutnya yang dilakukan adalah membuat *Scrapy Project* pada *Workspace Visual Studio Code* dengan menuliskan perintah :

```
scrapy startproject traveloka
```

File Directory project akan muncul ketika proses *startproject* berhasil dilakukan. *File directory project* terletak di dalam *workspace* pada bagian *explorer Visual Studio Code* seperti pada Gambar 4-19.

Setelah membuat *Scrapy Project*, selanjutnya adalah membuat satu *file spider* pada *folder spider*. *Spider* yang dibuat adalah *traveloka_spider.py.*



Gambar 4-19. *File directory scrapy project*

Gambar 4-19 merupakan tampilan direktori dari *project scrapy* yang dibuat. Direktori *spiders* merupakan tempat untuk *file source code spider*. Pengembangan teknik *scraping* dan *crawling* dilakukan dengan membuat *class spider* pada *file traveloka_spider.py* yg diletakkan pada direktori *spiders*. *File item.py* berisi satu class HotelReview yang merupakan implementasi dari model data hasil *scraping*.

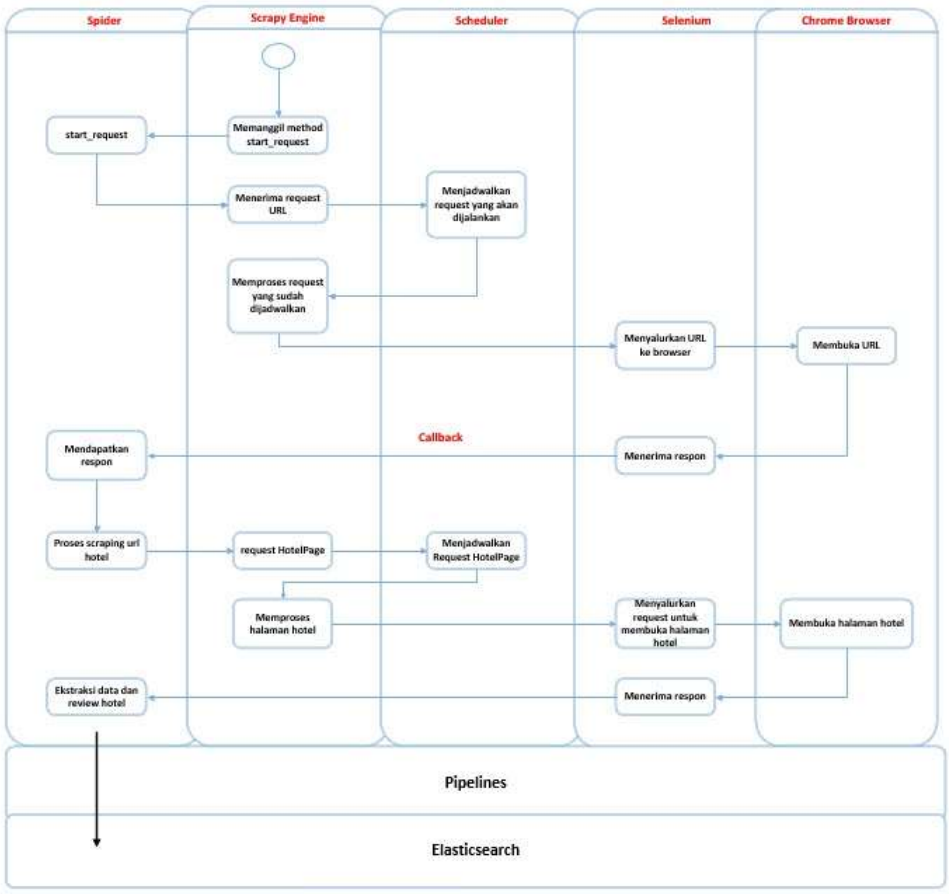
Selain itu juga terdapat *file setting.py. Setting.py* merupakan *file* untuk konfigurasi *framework scrapy*.

4.4. Pengembangan scrapy

Pengembangan *scrapy* dimulai dari proses pada *activity diagram*, melakukan konfigurasi pada *scrapy*, membuat model data pada *items*, membuat *code scraping* dan *crawling* pada *spider* dan konfigurasi pada *elasticserach* sebagai tempat penyimpanan data yang telah diekstraksi.

4.4.1. Activity Diagram

Pengembagan teknik *scraping* dan *crawling* diawali dengan merencanakan proses *scraping* dan *crawling*. Perencanaan proses dituangkan dalam diagram aktifitas. Pada diagram aktifitas dijabarkan proses yang terjadi pada semua modul modul yang digunakan.



Gambar 4-20. *Activity diagram*

Gambar 4-20 merupaka *Activity Diagram* proses *scraping* dan *crawling*. Proses ekstraksi data dilakukan di dalam *spider*. Spider merupakan sebuah *class* yang dibuat dalam *scrapy project. Code* yang terdapat dalam *spider* dimulai dari sebuah request sampai respon. Proses pertama adalah *engine scrapy* memanggil *spider* melalui *method strat_request*. *Method start_request* dalam *spider* berisi *link url* dari daftar *list* hotel. *Scheduler* menjadwalkan *request* yang masuk melalui *scrapy engine*. Setelah menerima *http request* yang telah dijadwalkan oleh *scheduler*, *scrapy engine* akan mengirim *request* tersebut ke *selenium* dan dilanjutkan ke *chrome driver* untuk diproses oleh *browser chrome*. Setelah itu *spider* akan menerima *http* respon *website* yang akan diakses dari *chrome*. *Spider*

akan mendapatkan respon melalui *method callback* yang telah didefinisikan pada *request scrapy request*. *Method callback* berikutnya adalah *method hotelList*.

Method selanjutnya adalah *method hotelList*. *Method hotelList* digunakan untuk mengambil *link url* (*canonical*) dari semua hotel yang ada. Setiap hotel dari daftar hotel dibuka terlebih dahulu menggunakan fitur *click* dari *selenium*. Proses klik dilakukan menggunakan *javascript* untuk *action click 'arguments[0].click().* Klik dengan *javascript* digunakan karena respon aksi klik diproses dengan *javascript* untuk membuka halaman hotel. Setelah halaman hotel dibuka, lalu *link url* hotel dapat diambil dengan menggunakan *xpath selector*. Proses itu terjadi berulang sampai sistem mendapatkan semua *link url* dari daftar hotel.

Jika semua *link url* hotel sudah masuk ke dalam *listURL*, maka *spider* akan mengirim *list url* ke *scheduler* melalu *scrapy engine* untuk dijadwalkan oleh *scheduler*. Setelah itu, *list* yang sudah dijadwalkan oleh *scheduler* akan dikirim ke *selenium* melalui *scrapy engine* untuk selanjutnya di proses di dalam *chrome driver*. Halaman hotel akan dibuka berdasarkan *list url* yang diterima oleh *chrome* dan akan diekstraksi data yang dibutuhkan sesuai yang elemen yang di tulis pada *xpath*. Lalu proses ekstraksi dilakukan oleh sistem secara berulang sampai semua *review* hotel di ekstraksi.

Pada saat ekstraksi data review hotel berlangsung, saat satu data *review* berhasil di ekstraksi data tersebut akan masuk ke dalam *item pipelines* dan oleh *item pipelines* akan diteruskan ke *elasticsearch*. *Elasticseach* adalah tempat akhir dimana data hasil *scraping* akan disimpan atau diproses lebih lanjut.

4.4.2. Konfigurasi Scrapy

Konfigurasi *scrapy* di dilakukan di dalam *setting.py.* Konfigurasi pada *scrapy* dilakukan untuk mengubah pengaturan pada *browser*.

```
BOT_NAME = 'traveloka'
SPIDER_MODULES = ['traveloka.spiders']
NEWSPIDER_MODULE = 'traveloka.spiders'

ROBOTSTXT_OBEY = False
CONCURRENT_REQUESTS = 8
DOWNLOAD_DELAY = 10
```

```
CONCURRENT_REQUESTS_PER_DOMAIN = 16
CONCURRENT_REQUESTS_PER_IP = 16

COOKIES_ENABLED = False

DEFAULT_REQUEST_HEADERS = {
   'Accept-Language': 'id',
}
```

Pada *setting.py* konfigurasi yang dilakukan berupa *disable robotstxt, concurrent request, download delay, disable cookies* dan bahasa yang digunakan *browser*. Disable *robotstxt* digunakan agar *scrapy* dapat melakukan *crawling* halaman-halaman tanpa mengukuti *setting* akses di *web server*. *Concurrent request* digunakan untuk menentukan jumlah akses *web browser* secara bersamaan. *Download delay* digunakan untuk memberikan jeda antar akses web. Dan *disable cookies* digunakan supaya akses web tanpa melibatkan *cookies* sehingga akses web jadi lebih cepat. Untuk web Traveloka dapat diakses tanpa menggunakan *cookies*.

4.4.3. Spider Url Hotel

Proses ekstraksi *url* hotel dilakukan terlebih dahulu. Terdapat dua *method* yang akan diproses sebelum ekstraksi pada *review* hotel dilakukan, yaitu *method start_request* dan *method hotelList.*

```
class TravelokaSpider(scrapy.Spider):
    name = "traveloka"

    def start_requests(self):
        urls = [
        'https://www.traveloka.com/id-id/hotel/search?spec=15-08-
         2018.16-08-
         2018.1.1.HOTEL_GEO.100154.Lampung%20Selatan.1'

        ]
        for url in urls:
            yield scrapy.Request(url,self.parse_hotelList)
```

Method start_request digunakan untuk melakukan *start_request. Start_request* dimulai dengan memasukan *url* yang akan dibuka pada *browser. Url* yang digunakan pada *start_request* adalah *url* daftar *list* hotel yang ada di Yogyakarta untuk kemudian di proses pada *method* selanjutnya untuk mengambil *url* dari masing-masing hotel.

```
def parse_hotelList(self, response):
print('==================================== start get hotel list
data ====================================')
```

```
    options = Options()
    options.add_argument('--headless')
    options.add_argument('--lang=id')
    options.add_argument('--disable-gpu')
    dv = webdriver.Chrome('E:/TOKOPEDIA/chromedriver.exe',
      chrome_options=options)
    dv.get(response.url)
    wait = WebDriverWait(dv,20)
    dv.implicitly_wait(7)
    listURL = []

wait.until(lambda dv:
dv.find_element_by_xpath("//div[@class='mMmI2 CZtP0 tvat-
searchListItem']"))
    nextPage = True
    while nextPage:
hotels = dv.find_elements_by_xpath("//div[@class='mMmI2 CZtP0
tvat-searchListItem']")
            try:
                for hotel in hotels:
                    hotel.click()
                    mainwindow = dv.window_handles[0]
                    hotelwindow = dv.window_handles[1]
                    dv.switch_to_window(hotelwindow)
wait.until(lambda dv:
dv.find_elements_by_xpath("//link[@rel='canonical']"))
hotelLink =
dv.find_element_by_xpath("//link[@rel='canonical']").get_attribute
('href')
                    listURL.append(hotelLink)
                    dv.close()
                    dv.switch_to_window(mainwindow)

nextButton = dv.find_element_by_xpath("//div[@id='next-button']")
dv.execute_script('arguments[0].click();', nextButton)
wait.until(lambda dv:
dv.find_element_by_xpath("//div[@class='mMmI2 CZtP0 tvat-
searchListItem']"))

          except Exception as e:
       print(e)
       nextPage = False

print('============================== finish get data LIST
HOTEL   ==========================')
      dv.quit()
      del dv

      for url in listURL:
yield scrapy.Request(url, self.parse_hotelPage)

 del listURL
```

Method diatas merupakan *method hotelList* yang digunakan untuk mengambil *list url* hotel (*canonical*). Pada bagian *options* diatur *options* untuk *webdriver*. *Browser* yang digunakan adalah *headless browser*. Pada *method* ini, sistem akan membuka satu *browser* untuk membuka daftar hotel. Pada saat meng-klik hotel di dalam daftar hotel, halaman hotel akan di tampilkan pada *tab* baru. Setelah itu, proses ekstrkasi pada elemen *canonical* akan dilakukan. Setelah mendapat *url* hotel, *url* tersebut akan disimpan di dalam *listURL* dan sistem akan menutup *tab* halaman hotel tersebut.

Setelah daftar hotel pada bagian *ajax* pertama selesai, sistem akan membaca *button next* dan otomatis melakukan *crawling* untuk mendapatkan *url* hotel pada halaman selanjutnya. Jika masih terdapat *next-page*, maka proses ekstraksi *url* hotel akan diteruskan sampai sistem mendapatkan semua *url* hotel yang terdapat pada daftar hotel. Setelah semua *url* hotel telah masuk ke dalam *listURL*, maka sistem akan mengirim semua *url* hotel yang terdapat pada *listURL* ke *scheduler*. Lalu sistem akan mengahpus *listURL*.

4.4.4. Spider Review Hotel

Setelah semua *url* hotel di dapatkan, sistem akan menjalankan *method hotelPage* di dalam *traveloka_spider* untuk melakukan ekstraksi pada *review* di masing-masing hotel.

```
def parse_hotelPage(self, response):
    print('========= start hotel page ============')
    options = Options()
    options.add_argument('--headless')
    options.add_argument('--lang=id')
    options.add_argument('--disable-gpu')
         page =webdriver.Chrome('E:/TOKOPEDIA/chromedriver.exe',
         chrome_options=options)
    page.get(response.url)
    wait = WebDriverWait(page,20)
    page.implicitly_wait(7)

         hotellName=page.find_element_by_xpath("//h1[@itemprop='n
         ame']").text

          hotelRatingElemen =
         page.find_elements_by_xpath("//div[@class='_3TWbq']")
    if (len(hotelRatingElemen) > 0):
        hotelRating =
         float(hotelRatingElemen[0].text.replace(',','.'))
```

```
else:
    hotelRating = 0

     hotelStarElemen =
     page.find_elements_by_xpath("//meta[@itemprop='ratingVal
     ue']")
if (len(hotelStarElemen) > 0):
    hotelStar =
     float(hotelStarElemen[0].get_attribute('content'))
else:
    hotelStar = 0

     hotelAddress =
     page.find_element_by_xpath("//span[@itemprop='streetAddr
     ess']").text

try:
    wait.until(lambda page:
     page.find_elements_by_xpath("//div[@itemprop='review']")
     )
     getReview = True

except:
          getReview = False

counter = 0
nextPage = True
while nextPage and getReview:
    reviews
     =page.find_elements_by_xpath("//div[@itemprop='review']"
     )
     hotelReviews = []
     for review in reviews:
         counter += 1
         reviewContent = review.text.split("\n")
         dtContent = reviewContent[2].split(' - ')
         hotelReview = HotelReview()
         hotelReview['hotelName'] = hotellName
         hotelReview['hotelStar'] = hotelStar
         hotelReview['hotelAddress'] hotelAddress
         hotelReview['hotelRating'] = hotelRating
         hotelReview['reviewRating'] = reviewContent[0]
         hotelReview['reviewName'] = reviewContent[1]
     hotelReview['reviewDate'] =
     self.getReviewDate(dtContent[0])
         if(len(dtContent)>1):
             hotelReview['reviewTheme'] = dtContent[1]
         hotelReview['reviewText'] = reviewContent[3]
         hotelReviews.append(hotelReview)

     for hotelReview in hotelReviews:
         yield hotelReview

     nextButtonElemen =
     page.find_elements_by_xpath("//div[@id='next-button']")
     if(len(nextButtonElemen)>0):
```

```
            page.execute_script('arguments[0].click();',
            nextButtonElemen[0])
                page.implicitly_wait(5)
            wait.until(lambda page:
            page.find_elements_by_xpath("//div[@itemprop='review']")
            )
             else:
                nextPage = False
                del hotelReviews

    print('========= end hotel page ============')
    page.quit()
    del page
```

Method ini berfungsi untuk mengekstraksi data hotel dan *review* hotel yang dibutuhkan. Pada bagian *options* diatur *options* untuk *webdriver*. *Browser* yang digunakan adalah *headless browser*.Setelah semua *url* hotel masuk di *scheduler*, *scheduler* akan mengatur jadwal untuk masing-masing *url* hotel yang akan dibuka oleh *browser*. Dalam *setting.py* diatur *concurrent request* untuk *browser*, yaitu *browser* dapat membuka 8 *url* hotel dari *scheduler* untuk dibuka secara bersamaan dalam 8 *browser*.

Setelah *url* hotel dibuka oleh *browser*, sistem akan memulai ekstraksi pada data hotel berupa nama hotel, alamat hotel, *rating* hotel, dan bintang hotel. Lalu ekstraksi selanjutnya dilakukan pada *review* hotel. Semua data *review* hotel yang berupa teks akan diekstraksi seperti nama, *rating*, tanggal, tema dan teks *review*. Setelah selesai mengekstraksi daftar *review* yang pertama, sistem akan melakukan *crawling* untuk membuka daftar *review* selanjutnya dan mengekstraksi *review* selanjutnya sampai pada *review* terakhir di daftar *review* hotel tersebut. Proses itu terjadi pada masing-masing halaman hotel yang akan di ekstraksi.

Ketika sistem telah mendapatkan hasil ekstraksi dari halaman hotel, data hasil ekstraksi akan dikirim ke *items pipelines* untuk selanjutnya di simpan dalam *elasticsearch*. Dan saat semua halaman hotel sudah selesai di esktraksi, sistem akan menutup dan menghapus semua halaman hotel.

```
def getReviewDate(self, dt):
  tanggal = dt.split(" ")

  if(tanggal[1] == "Jan"):
      bl = 1
  elif(tanggal[1] == "Feb"):
      bl = 2
  elif(tanggal[1] == "Mar"):
```

```
        bl = 3
    elif(tanggal[1] == "Apr"):
        bl = 4
    elif(tanggal[1] == "Mei"):
        bl = 5
    elif(tanggal[1] == "Jun"):
        bl = 6
    elif(tanggal[1] == "Jul"):
        bl = 7
    elif(tanggal[1] == "Agu"):
        bl = 8
    elif(tanggal[1] == "Sep"):
        bl = 9
    elif(tanggal[1] == "Okt"):
        bl = 10
    elif(tanggal[1] == "Nov"):
        bl = 11
    elif(tanggal[1] == "Des"):
        bl = 12

    tgl = tanggal[0] + " " + str(bl) + " " + tanggal[2]
        hasil = datetime.strptime(tgl,'%d %m %Y').strftime('%Y-%m-
        %dT%H:%M:%S%z')
        return hasil
```

Method getReviewDate merupakan *method* yang berfungsi untuk mengubah format tanggal pada hasil ekstraksi menjadi format tanggal yang bisa diterima oleh elasticsearch. Pada *method hotelPage*, data yang di ekstraksi semuanya adalah berupa data tesk termasuk tanggal, jadi dibutuhkan *method getReviewDate* untuk mengubah format tanggal yang awalnya teks menjadi *datetime*.

4.4.5. Konfigurasi Modul Elasticsearch

Konfigurasi modul *elasticsearch* dilakukan di dalam *setting.py.* Didalam *pipelines.py* terdapat satu *class* yang digunakan untuk menerima *object* berupa *items* dari *object* yang telah dibuat pada *items.py*. Pada *setting.py* dilakukan konfigurasi *pipelines* untuk *elasticseacrh* agar data yang bisa disimpan di dalam *elasticsearch*. Angka 500 pada *ScrapyElasticsearch* menunjukan prioritas untuk proses di *pipelines*. Selanjutnya terdapat konfigurasi *elasticsearch* untuk *server*, *index*, *format index date*, *username*, *password*, *type* dan *buffer length*.

```
class TravelokaPipeline(object):
def process_item(self, item, spider):
      return item


ITEM_PIPELINES = {
```

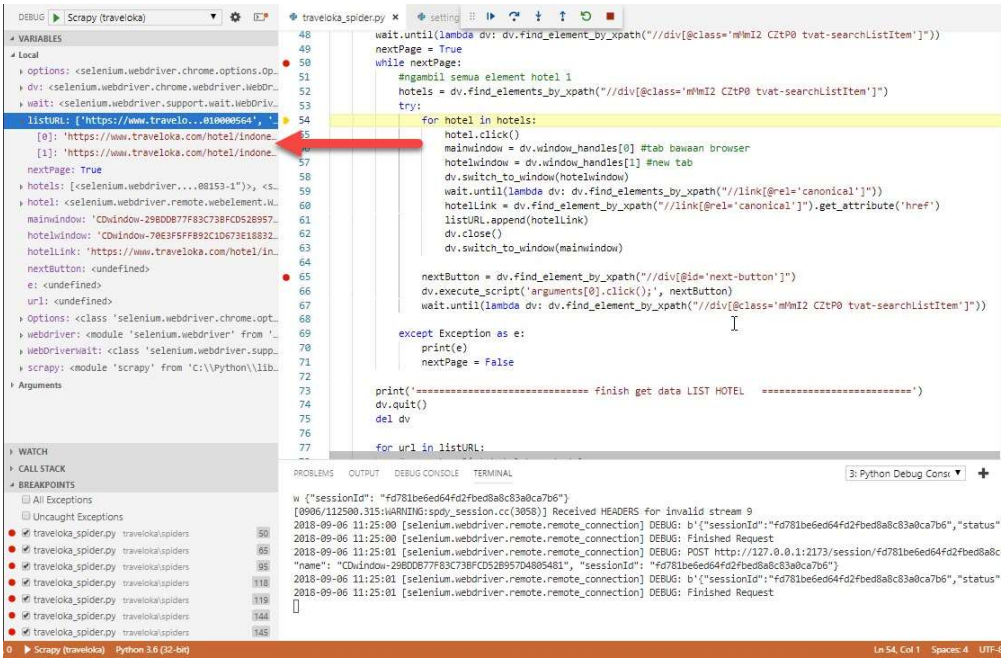
```
    'traveloka.sentiment.ReviewSentimentPolarity':300,

        'scrapyelasticsearch.scrapyelasticsearch.ElasticSearchPi
        peline': 500
}

ELASTICSEARCH_SERVERS = ['139.162.60.42']
ELASTICSEARCH_INDEX = 'reviewHotel'
ELASTICSEARCH_INDEX_DATE_FORMAT = '%Y-%m'
ELASTICSEARCH_USERNAME = 'elastic'
ELASTICSEARCH_PASSWORD = 'TeloGosong'
ELASTICSEARCH_TYPE = 'items'
ELASTICSEARCH_BUFFER_LENGTH = 50
```

4.5. Pengujian

Pengujian sistem scraping dan crawling dibagi menjadi dua tahap yaitu pengujian fungsional dan pengujian proses. Pengujian fungsional bertujuan untuk dapat mengetahui apakah *method* yang dibuat pada *spider* dapat berfungsi dengan baik untuk melakukan *scraping* dan *crawling*. Sedangkan pengujian proses adalah untuk menguji keseluruhan proses hingga mendapatkan hasil data terstruktur sesuai yang diharapkan.

4.5.1. Pengujian Fungsional Scraping dan Crawling

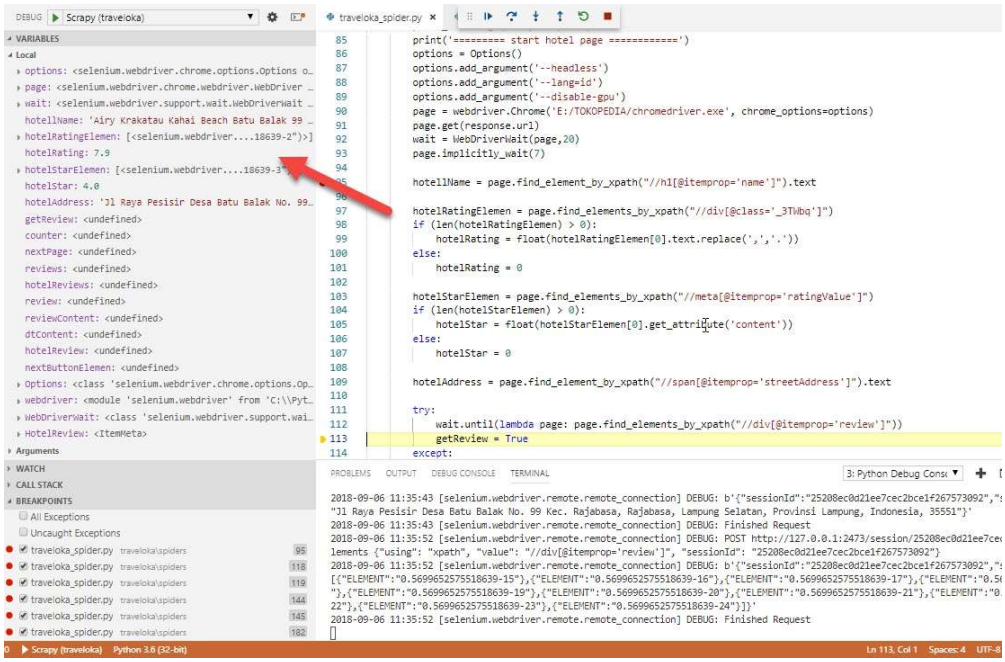
Pengujian Sistem *Scraping* dan *Crawling* dilakukan dengan cara *Debugging*. Proses *Debugging* dilakukan dengan cara memberikan *break point* pada *code* yang ada di dalam *traveloka_spider* untuk melihat apakah data berhasil di ekstraksi dan proses *crawling* berjalan dengan baik. Pada proses *debugging* dilakukan pengamatan pada variabel-variabel yang digunakan untuk menampung data hasil *scraping*. Selain itu juga dilakukan pengamatan pada proses pemanggilan *method callback* yang dibuat di *class spider*.



Gambar 4-21. Pengujian ekstraksi *list url* hotel

Pada Gambar 4-21 menunjukan hasil *list url* hotel yang sudah berhasil diekstraksi. Proses *scraping* dan *crawling list url* hotel dilakukan pada *method HotelList*. *List url* hotel yang berhasil di ekstraksi akan dimasukkan kedalam *list* yaitu *listURL*. *List url* hotel yang sudah diekstraksi dapat dilihat dalam *listURL* pada bagian *variables*.

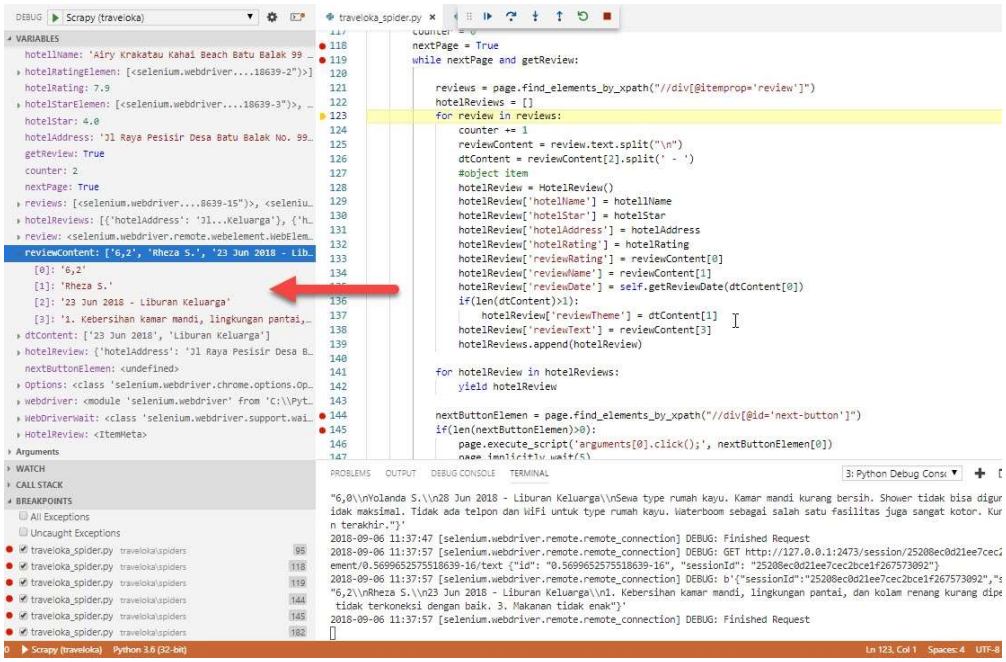
Selanjutnya pengujian untuk ekstraksi data hotel juga dilakukan dengan cara *debugging*. Proses ekstraksi data hotel dilakukan pada *method HotelPage*. *List url* yang sudah di ekstraksi akan dibuka oleh *browser* setelah itu ekstraksi pada data hotel dilakukan oleh sistem. Pada Gambar 4-22 di dalam *local* terlihat hasil data hotel yang telah berhasil diekstraksi.



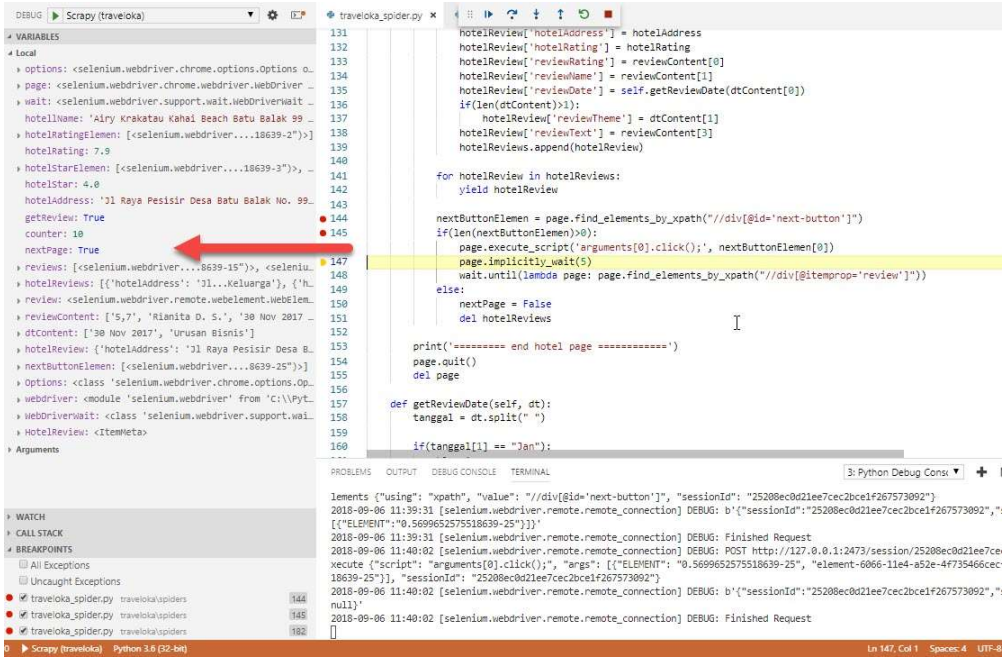
Gambar 4-22. Pengujian ekstraksi data hotel

Setelah melakukan ekstraksi data pada halaman hotel, proses selanjutnya adalah ekstraksi pada data *review* hotel. Proses ekstraksi *review* pada halaman hotel berhasil dilakukan. Proses ekstraksi review pada halaman hotel terjadi pada halaman yang sama pada setiap data hotel yang diekstraksi.

Pada Gambar 4-23 dapat dilihat hasil pengujian untuk proses ekstraksi *review* hotel. Pada bagian *variables*, didalam *review content* adalah data *review* yang telah berhasil diekstraksi berupa nama, tanggal, tema, rating dan teks review.



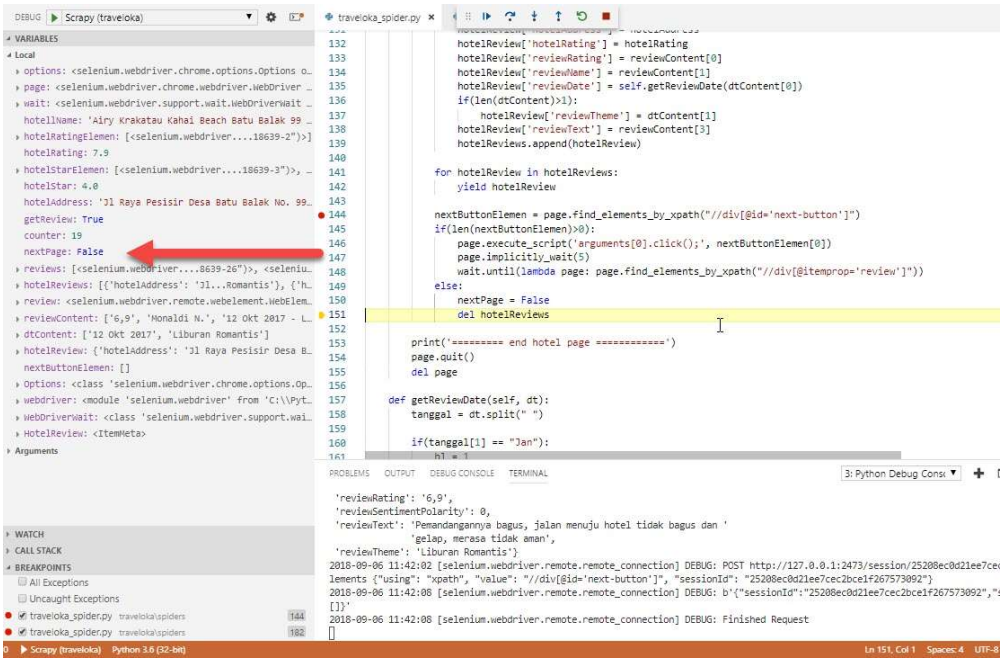
Gambar 4-23. Pengujian ekstraksi *review* hotel



Gambar 4-24. Pengujian *crawling* 1

Pada Gambar 4-24 menujukukan proses *crawling* pada sistem berhasil dilakukan. Proses *crawling* dilakukan pada saat ekstraksi pada *url* hotel dan *review*

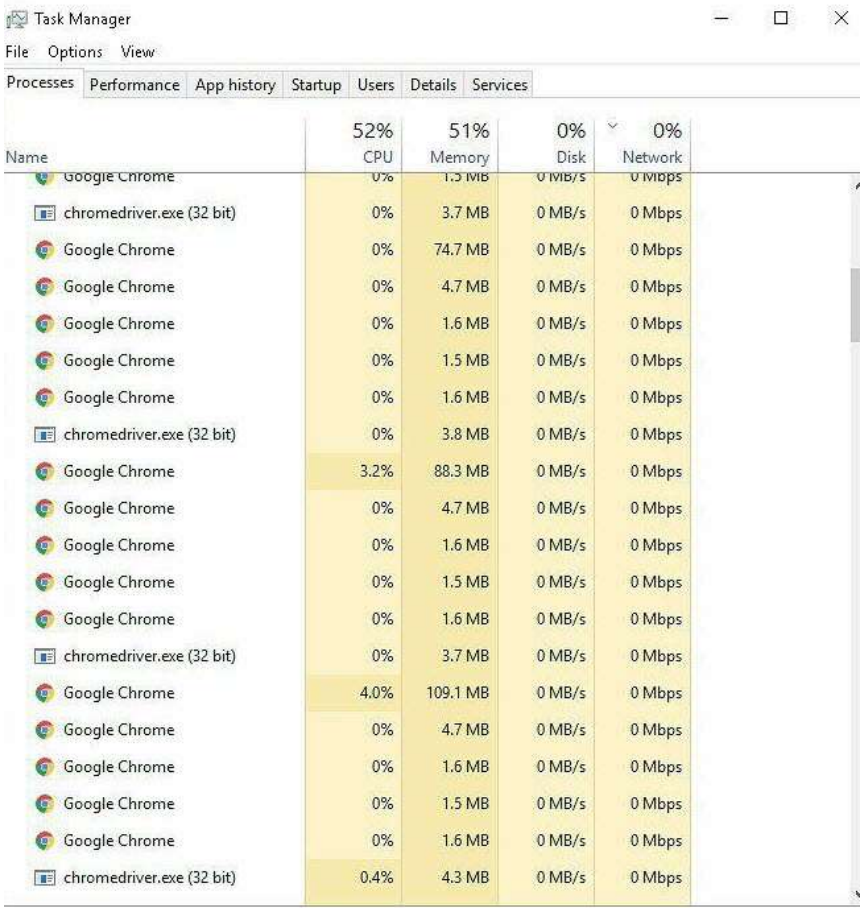
hotel. Jika terdapar *button-next* atau halaman selanjutnya yang terdapat dalam *ajax*, maka sistem akan otomatis melakukan *crawling* untuk memuat daftar selanjutnya. Namun, jika tidak terdapat *button-next*, maka sistem tidak akan melakukan *crawling* dan langsung menuju proses selanjutnya seperti pada Gambar 4-25.



Gambar 4-25. Pengujian *crawling* 2

4.5.2. Pengujian Proses Scraping dan Crawling

Pengujian pada proses *scraping* dan *crawling* dilakukan saat sistem dijalankan. Pengujian ini dilakukan dengan cara melihat pada *task manager* terkait *concurrent* dan melihat hasil akhir dari proses *scraping* dan *crawling* pada *terminal*.



Gambar 4-26. Pengujian proses *concurrent*

Pengujian pada *setting concurrent* dapat dilihat pada *task manager* pada Gambar 4-26 ketika sistem dijalankan. *Concurrent* request yang diatuar pada *setting.py* adalah 8, yaitu menunjukan 8 *browser* dapat dijalankan secara bersamaan untuk membuka *url* hotel dan melakukan proses ekstraksi pada halaman hotel.

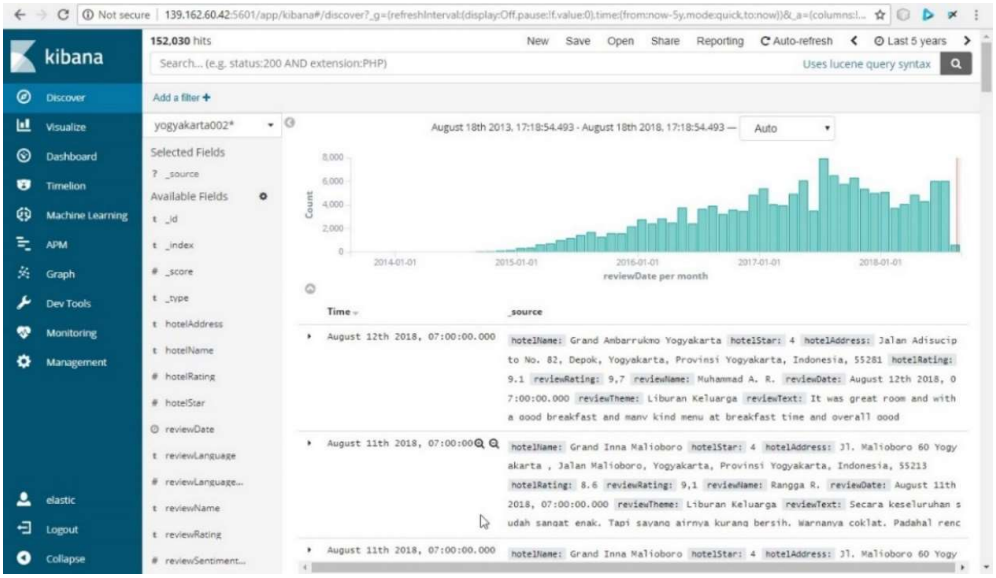
```
      2018-08-14 07:57:47 [scrapy.statscollectors] INFO: Dumping
Scrapy stats:
      {'downloader/exception_count': 70,
      'downloader/exception_type_count/twisted.internet.error.Time
outError': 1,
      'downloader/exception_type_count/twisted.web._newclient.Resp
onseNeverReceived': 69,
       'downloader/request_bytes': 155522,
       'downloader/request_count': 491,
       'downloader/request_method_count/GET': 491,
       'downloader/response_bytes': 36882418,
       'downloader/response_count': 421,
       'downloader/response_status_count/200': 421,
       'finish_reason': 'finished',
       'finish_time': datetime.datetime(2018, 8, 14, 0, 57, 47,
748515),
       'item_scraped_count': 174770,
       'log_count/DEBUG': 1165453,
       'log_count/ERROR': 14,
       'log_count/INFO': 743,
       'log_count/WARNING': 3226,
       'memusage/max': 136798208,
       'memusage/startup': 64970752,
       'request_depth_max': 1,
       'response_received_count': 421,
       'retry/count': 62,
       'retry/max_reached': 8,
      'retry/reason_count/twisted.web._newclient.ResponseNeverRece
ived': 62,
       'scheduler/dequeued': 491,
       'scheduler/dequeued/memory': 491,
       'scheduler/enqueued': 491,
       'scheduler/enqueued/memory': 491,
       'spider_exceptions/NoSuchElementException': 1,
       'spider_exceptions/ValueError': 3,
       'start_time': datetime.datetime(2018, 8, 13, 17, 36, 4,
295672)}
      2018-08-14 07:57:47 [scrapy.core.engine] INFO: Spider closed
(finished)
```

Laporan hasil *scraping* dan *crawling* akan muncul pada tampilan terminal setelah proses selesai dilakukan. Pada laporan tersebut terdapat detail hasil proses *scraping* dan *crawling* berupa proses berhasil dilakukan, jumlah *items* yang telah di ekstraksi sebanyak 174770, jumlah antrian pada scheduler sebanyak 491, crawling yang berhasil dilakukan sebanyak 421, crawling yang gagal dilakukan sebanyak 8, dan crawling yang melakukan retry sebanyak 62.

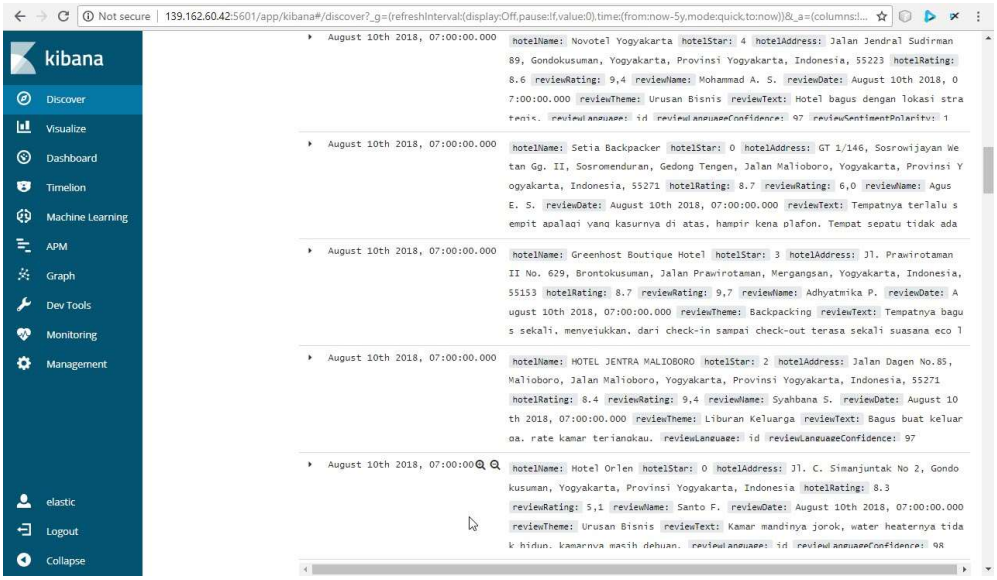
4.6. Hasil

Data hasil ekstraksi akan disimpan pada *elasticsearch* melalui *pipelines*. Gambar 4-27 dan Gambar 4-28 adalah hasil data yang sudah berada pada

elasticsearch. Semua data yang didapat dari hasil ekstraksi akan masuk dalam *index elasticsearch* melalui *items pipelines*. Data yang diekstraksi adalah data hotel (nama, alaman, *rating* dan bintang hotel) dan *review* hotel (nama, tanggal, tema, *rating*, teks *review*) yang ada di Yogyakarta. Dibagian kiri atas halaman terdapat jumlah semua data yang telah diekstraksi.

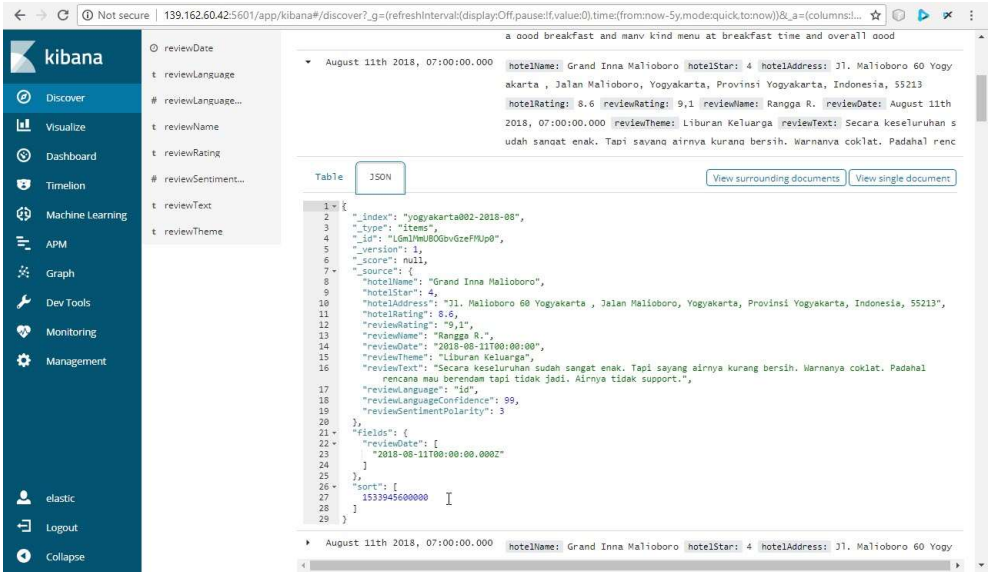


Gambar 4-27. Hasil ekstraksi 1



Gambar 4-28. Hasil ekstraksi 2

Gambar 4-28 merupakan tampilan data yang ada di *elasticsearch*. Data ditampilkan berdasarkan *review* hotel. Masing-masing data *review* hotel dapat dibuka dan akan menampilkan data json sesuai isi dari data *review* tersebut. Data yang disimpan di dalam *elasticsearch* telah berhasil disimpan berdasarkan model data yang telah dibuat.



Gambar 4-29. Hasil ekstraksi (*json*)

Masing-masing data *review* hotel yang ada pada *elasticseacrh* dapat dilihat dalam *format json* seperti pada Gambar 4-29. Data hasil ekstraksi yang ditampilkan di *elasticsearch* merupakan data yang terstruktur dengan model data sesuai dengan yang telah didefinisikan pada *class item*.