```
@Query("""
{"bool": {
"filter": [
{ "match": { "geoTags": "?0" } },
{ "match": { "brands" : "?1" } },
{ "exists": { "field" : "name" } }]}}""")
fun findByGeotagForBrands(geotag: String, brands: Set<String>):
    Flux<Hotel>
```

Die Ergebnisse müssen sowohl in der angefragten Region liegen, was über die geoTags sichergestellt wird, als auch der angefragten Brand angehören. Hierfür enthält das „brands“ Feld der Hotels eine Liste der zugehörigen Brands. Eine Brand ist dabei ein Veranstalter der DER Touristik Gruppe, wie dies eingangs im Abschnitt „Tourismus“ erklärt wurde. Ein Hotel kann dabei für unterschiedliche Brands verfügbar sein.

6.3 Testen des Geo Serivces

Abschnitt komplett umbauen, kein Erzähler wechsel, in Read.ME des Gits auslagern.

In diesem Abschnitt wird der Geo Service in seiner vollen Funktion getestet. Dies können Sie begleiten, in dem Sie sich das Projekt von GitHub clonen. Zu finden ist dieses unter dem folgendem Link:
https://github.com/Brebeck-Jan/Spring-Kotlin-Project-with-RabbitMQ-Elasticsearch

6.3.1 Producer zum lokalen Testen

Da zur Entwicklung des Geo Service nicht der bestehende Hotelimport genutzt werden soll, da dieser große Datenmengen bei externen Dienstleistern abfragt, wird ein Producer entwickelt, welcher Hotelobjekte zu Testzwecken erstellen soll. Der Producer soll beim Aufruf Hotelobjekte erzeugen und diese auf dieselbe RabbitMQ-Queue legen, wie dies der Hotelimport tut. Die Daten für die Testobjekte werden in einer JSON-Datei spezifiziert. Der Producer wird als Service implementiert, bei dem durch einen Anfrage an localhost und den entsprechenden Port das Erstellen von Hotelobjekte gestartet werden kann. Die Daten der Hotelobjekte, die der Producer erzeugt, werden in der Hotels.json Datei spezifiziert. Diese

Datei enthält Informationen zu Namen und Koordinaten der zu erstellenden Testobjekte und sieht wie folgt aus:

```
{"Hotels":[{"name":"Hotel Eins",
"coord":[13.031665402239936,52.58441548551115]},...]}
```

In der JSON-Datei können zu Testzwecken beliebig viele Hotels spezifiziert werden. Der Producer wird ebenfalls als Spring-Boot-Service entwickelt und ist unter localhost:8080 zu erreichen. Mit einer Post-Anfrage an localhost:8080/triggerImport/ stößt man die Erstellung der Testobjekte an.

6.3.2 Benötigte Infrastruktur starten

Der Geo Service benötigt zum Arbeiten Instanzen von ElasticSearch und RabbitMQ. Darüber hinaus empfiehlt sich noch eine Kibana Insanz zur Überwachung der ElasticSearch. Alle diese benötigenden Bestandteile sind in einem docker-compose.yml gebündelt. Dies muss über die Kommandozeile mithilfe von Docker ausgeführt werden. Stellen Sie hierfür sicher, das Docker Desktop auf ihrem Computer arbeitet. Das docker-compose wird mit dem folgenden Befehl ausgeführt:

```
docker-compose up -d
```

6.3.3 Import der Daten

Nach dem alle Infrastrukturbestandteile hochgefahren sind, muss der Import der Geodaten durchgeführt werden. Dazu sind im Git beispielhaft die Daten für einige Länder hinterlegt. Für den Import muss das geo-import.py Skript ausgeführt werden, dieses ist im Ordner elastic_setup zu finden. Ausgeführt wird dieses Skript ebenfalls über die Kommandozeile. Sie benötigen für die Ausführung des Skriptes eine Python 3.X Version. Je nach Ihrer Installation wird das Skript mit einem der hier folgenden Befehle gestartet:

namen ändern

```
python geo-import.py
python3 geo-import.py
```

Sollte die Infrastruktur noch nicht vollständig hochgefahren sein, werden sie die folgenden oder eine ähnliche Fehlermeldung erhalten:

```
File "/Library/Frameworks/Python.framework/Versions/3.8/lib/
    python3.8/site-packages/urllib3/util/connection.py", line
    74, in create_connection
sock.connect(sa)
ConnectionRefusedError: [Errno 61] Connection refused
```

Das Skript wird versuchen sich alle 10 Sekunden erneut mit der Elasticsearchinstanz zu verbinden, bis die Verbindung etappliert ist oder Sie dies über den Tastenbefehhl Strg+C stoppen.
Während des Imports werden unterschiedliche Informationen von dem Skript in die Kommandozeile ausgegeben. Hier sind Beispielhaft die Ausgaben eines Importdurchlaufs zu sehen:

```
start import
connected to ES
./data/countries.geojson
246
{'count': 245, '_shards': {'total': 1, 'successful': 1, '
    skipped': 0, 'failed': 0}}
```

In Zeile 3 wird stets zu Beginn der Bearbeitung einer Datei dessen Dateipfad ausgegeben. In der vierten Zeile wird die Anzahl der Bearbeiteten JSON Einträge ausgegeben. In der darauf folgenden Zeile 5 ist dem Attribut „count“ der Wert der tatsächlich in der Datenbank gespeicherten Werte zu entnehmen. Sollte zwischen den beiden Werten eine Differenz bestehen ist bei der Bearbeitung oder der Speicherung eines beziehungsweise einiger Einträge ein Fehler aufgetreten. Die Namen der betroffenen Einträge wird in das error.txt geschrieben, so kann im Nachgang überprüft werden, welche Probleme mit diesem Eintrag bestehen.
Nach dem der Import der Geodaten abgeschlossen ist, kann das Ergebniss in Kibana überprüft werden. Kibana ist dabei ein Toll zur Visualisierung und Explorierung der Daten einer Elasticsearch. Die mit dem docker-compose gestartete Instanz ist unter localhost:5601 zu finden. Dort muss man auf der linken Seite zu dem Reiter **Dev Tools** navigieren. Damit öffnet sich eine Console, mit der man Abfragen an die Elasticsearchinstanz auf localhost stellen kann. Beispielhaft zum Überprüfen des Imports kann folgenden Abfrage verwendet werden:

```
GET geo-index/_search
{"query": {"match_all": {}}}
```

Die Ausgabe der Abfrage enthält unter anderem Informationen über die benötigte Zeit für die Abfrage und die Anzahl der Ergebnisse, darüber hinaus sind nach diesen Informationen die Ergebnisse der Suche aufgelistet. Ein Ergebnis sieht dabei beispielhaft verkürzt wie folgt aus:

```
"name" : "Ireland",
    "location" : {
        "type" : "multipolygon",
        "coordinates" : [[[[-7.3551,55.380675],...]]]}
```

6.3.4 Producer starten und anstoßen

Nach dem nun alle benötigten Infrastrukturbestandteile verfügbar sind und die Daten importiert wurden, können nun die Services gestartet werden. Dazu muss mit der Kommandozeile im producer Ordner der folgende Befehl ausgeführt werden:

```
./gradlew bootRun --args='--spring.profiles.active=dev'
```

Anschließen kann mit der bereits beschriebenen Anfrage an den Service, der Import getriggert werden. Diese kann unter anderem mit der Kommandozeile und dem folgenden Befehl angestoßen werden:

```
curl -X POST localhost:8080/triggerImport
```

Wenn der Import läuft, werden die Objekte in die Konsole ausgegeben. Hierfür ein Beispiel: "name":"Hotel Eins","coord":[11.031665402239936,48.58441548551115]. Des Weiteren kann das Ergebnis des Producers in dem Dashboard der RabbitMQ überprüft werden. Dies ist unter http://localhost:15672 zu erreichen und die benötigten Einloggdaten sind: Username: producer; Password: password. Unter dem Reiter *Queues* werden alle Queues gelistet. Wenn man dort auf die Hotel-Queue klickt, gelangt man zur Übersicht dieser Queue. Auf dieser Seite sieht man unter anderem wie viele Messages aktuell auf der Queue liegen. Unter dem Reiter *Get messages* kann man sich eine Message von der Queue ausgeben lassen. Dies sieht wie folgt aus:

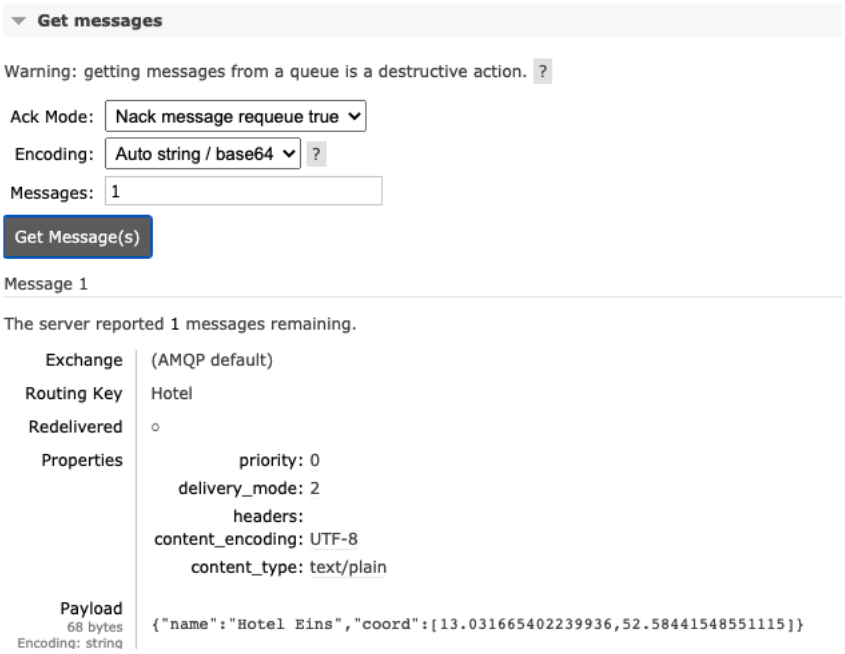

Abbildung 6.2: Screenshot einer Beispiel Message von dem RabbitMQ Dashboard

6.3.5 Geo Service starten und überprüfen

Anschließend muss der Geo Service mit dem gleichen Befehl wie der producer gestartet werden: **./gradlew bootRun –args='–spring.profiles.active=dev'**. Der Service konsumiert nach dem hochfahren die Messages von der Queue, die wir vorher mit dem producer erzeugt haben. Nach der Verarbeitung einer Message wird das Objekt, das auf die Hotel-Queue gelegt wird, in die Konsole ausgegeben. Ein solches Objekt sieht dabei wie folgt aus:

```
1 Hotel(name=Hotel Eins, coord=[13.031665402239936, 52.58441548551115],
2 tags=[Germany])
```

Anschließend kann erneut auf dem RabbitMQ Dashboard unter der GeoTags-Queue überprüft werden, wie die Hotelobjekte auf der Queue abgelegt werden. Dies sieht wie folgt aus: "name":"Hotel Eins","coord":[13.031665402239936,52.58441548551115],"tags":["Germany"]. Mit diesem Schritt ist die Verarbeitung von dem Geo Service abgeschlossen und auch der Test dessen abgeschlossen. Der Geo Service funktioniert wie erwartet und arbeitet zuverlässig.