Testen des Geo Serivces

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

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.

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
```

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. namen ändern
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:

```
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:

```
1 File "/Library/Frameworks/Python.framework/Versions/3.8/lib/
     python3.8/site-packages/urllib3/util/connection.py", line
     74, in create_connection
2 sock.connect(sa)
3 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:

```
1 start import
2 connected to ES
3 ./data/countries.geojson
4 246
5 {'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:

```
1 GET geo-index/_search
2 {"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:

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

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:

```
1 ./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:

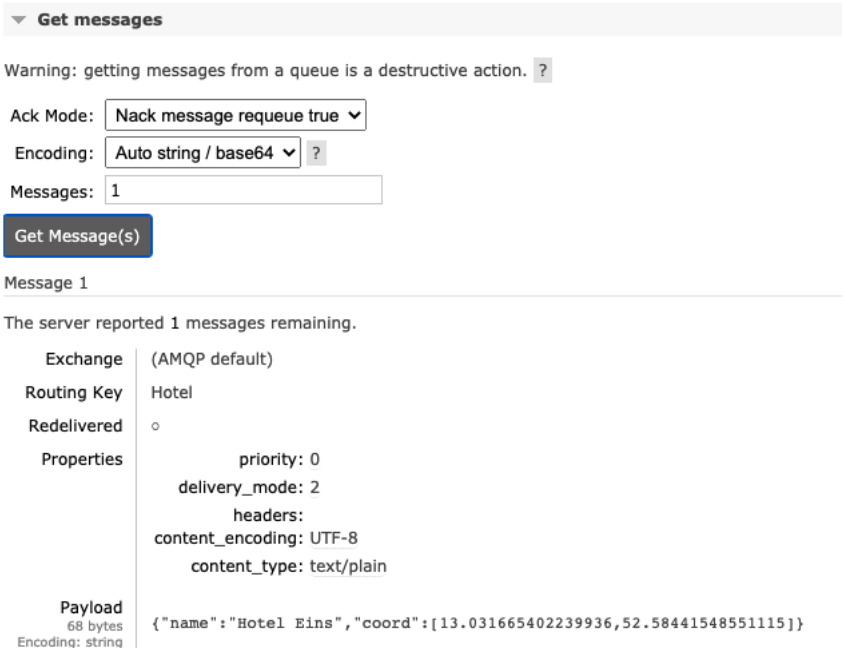


Abbildung 6.2: Screenshot einer Beispiel Message von dem RabbitMQ Dashboard

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.

6.3 Beurteilung der Umsetzung

In diesem Abschnitt wird die Umsetzung mit den in Kapitel 3 „Wissenschaftliche Methodik“ aufgestellten Kriterien beurteilt. Dazu werden die definierten Akzeptanzkriterien verwendet und auf die Erfüllung überprüft. Dazu wird hier zunächst eine Checkliste abgebildet, die die erfüllten und nicht erfüllten Akzeptanzkriterien aufzeigt. Im Anschluss werden die einzelnen Punkte detailliert beschrieben.

Akzeptanzkriterium	Erfüllt	Nicht Erfüllt
Hotelsuche mithilfe von Städten	☒	☐
Hotelsuche mithilfe von Bundesländern	☒	☐
Hotelsuche mithilfe von Ländern	☒	☐
Hotelsuche mithilfe von politischer Regionen	☒	☐
Hotelsuche mithilfe von touristischer Regionen	☒	☐
Ergebnis der Destinationssuche sind fachlich korrekt	☒	☐
Hotels sind mehreren Regionen zugeordnet	☒	☐
Regionsdaten sind abgespeichert und können bearbeitet werden	☒	☐
Regionstyp einer Destination ist ersichtlich	☒	☐
Alle Hotels in einer touristischer Regionen werden gefunden	☐	☒

Der Geo Service ermöglicht die Suche nach Hotels mit jeglicher Art von Regionen. Der vorab definierte Test mit dem „LUITPOLDPARK-HOTEL“ ist erfolgreich. Das Hotel wird unter anderem mit den Suchbegriffen Füssen, Bayern, Deutschland und Allgäu gefunden. Damit sind die ersten 5 Akzeptanzkriterien für den Geo Service erfüllt.

Die fachliche Korrektheit der Ergebnisse ist erfüllt. Dazu wurde 1/3 der Suchergebnisse für das Suchwort „Allgäu“ überprüft. Alle überprüften Ergebnisse liegen tatsächlich in der Region des Allgäus.

Die Mehrfachzuordnung von Regionen ist erfolgreich und wurde anhand des Beispiel Hotels „LUITPOLDPARK-HOTEL“ getestet. Dieses erhält erfolgreich die Regionslabel Füssen, Bayern, Deutschland und Allgäu. Darüber hinaus ist die Mehrfachzuordnung außerdem

notwendig um die ersten 5 Akzeptanzkriterien zu erfüllen und wird dadurch ebenfalls überprüft.

Die Regionsdaten des Datensatzes sind in einer Datenbank gespeichert, verfügbar und damit ist die Möglichkeit gegeben, diese Mittels Abfragen zu verändern. Damit ist dieses Akzeptanzkriterium erfüllt.

Der Regionstyp ist bei der Abfrage gegen die Elasticsearch, zur Bestimmung der Regionslabels für ein Hotel, ersichtlich und erfüllt damit das Akzeptanzkriterium.

Der aktuell dem Geo Service zu Grunde liegende Datensatz birgt einige Schwächen in der Datenqualität. Deswegen ist das letzte Akzeptanzkriterium nicht erfüllt. Bei der Suche nach Hotels im Allgäu werden lediglich Hotels in Bayern gefunden. Die Region im Datensatz erstreckt sich nur über Bayern und nicht über Bayern und Baden-Württemberg, wie diese in Realität ist. Dadurch wird lediglich das Hotel Mohren in Oberstdorf, Bayern gefunden und nicht das Beispielhotel in Baden-Württemberg.

Unter Berücksichtigung der aufgeführten Punkten kommt der Geo Service zum produktiven Einsatz. Die Hotels werden nun um Regionslabels angereichert im Hotel Service abgespeicher und die Suche nach Hotels mittels Regionen steht über die Schnittstelle des Productrepositories zur Verfügung. Diese Funktionalität wird vom Frontend aufgegriffen und steht damit dem Endkunden zur Verfügung. Ein Auszug aus einer solchen Suche ist im Anhang unter F „Beispielhafte Geosuche auf der neuen Plattform“ zu finden.

6.4 Erkenntnisse der Umsetzung

In work

Aus der Umsetzung für das spezifische Problem der Suche nach Hotels können...

Verallgemeinerungen finden, die auch für die Umsetzung bei anderen Problemstellungen zum Einsatz kommen können.
Z.B. Wahlbüro mit PLZ suchen oder POI in der Nähe des eigenen Standorts, einer gesetzen Position, etc.

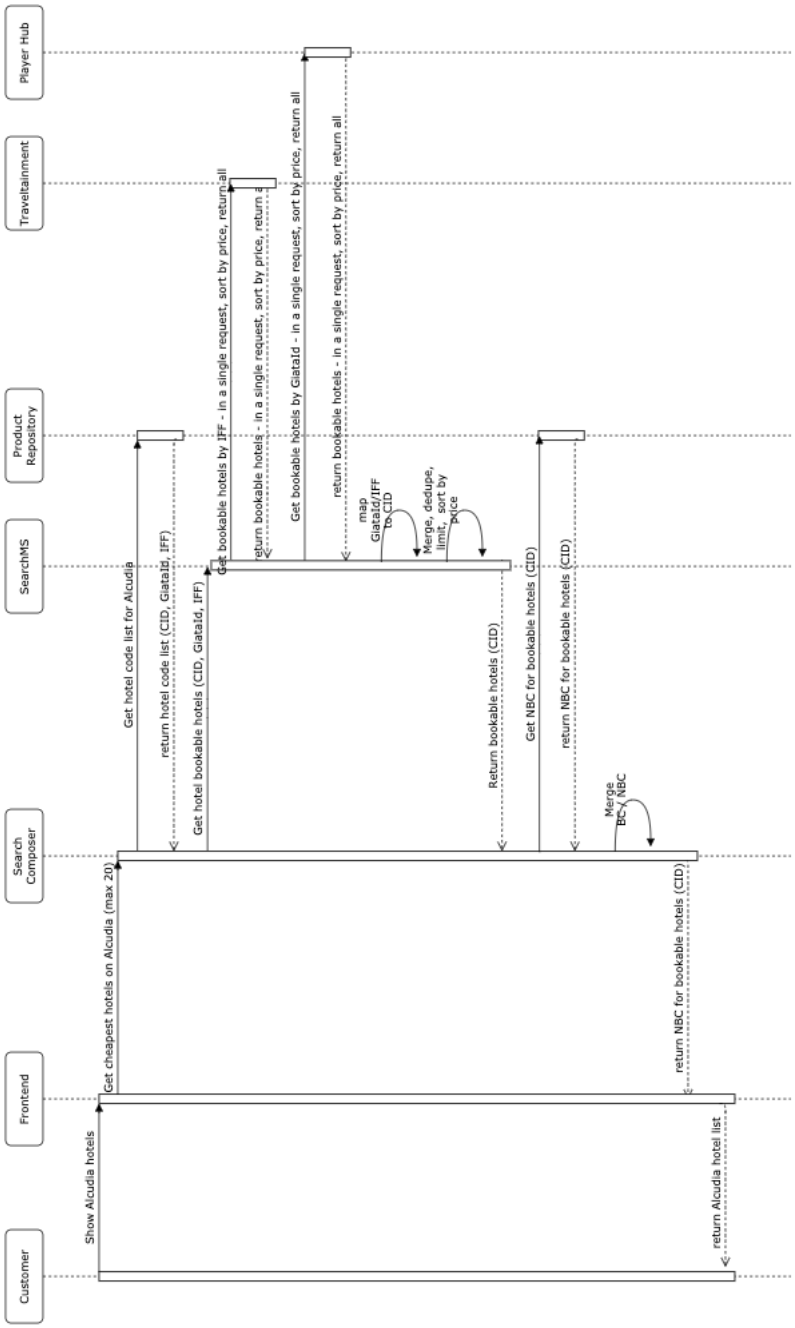
- Batch verarbeitung mit Vertaggung - Aufteilen der Suchanfragen - Nicht Notwendigkeit von Geosuchabfragen bei Kunden-/Nutzeranfragen

7 Fazit

In dieser Bachelorarbeit sollte die Entwicklung eines Geo Service beschrieben werden, der das Problem der Suche nach Hotels mit Hilfe von Regionen löst. Zur Lösung dieses Problems sollte unter anderem eine Geodatenbank zum Einsatz kommen. Zunächst wurde die aktuell vorherrschende Situation analysiert. In diesem Zuge wurde beschrieben, wie die aktuell verwendete Suche funktioniert und welche Anforderungen an den Geo Service zum Aufbau des zentralen Elements der Geosuche gestellt werden. Die Anforderungen wurden in fachliche und technische Anforderungen unterteilt und mit Akzeptanzkriterien spezifiziert. Diese enthalten messbare Kriterien, anhand deren bestimmt wird, ob ein Kriterium erfüllt wurde. Mit diesen aufgestellten Anforderungen wurde das Konzept zur Umsetzung erarbeitet und dazu zunächst eine geeignete Datenbank evaluiert. Dafür wurde ein Datensatz verwendet, um damit die Datenbanken PostGIS, MongoDB und ElasticSearch anhand vorher definierter Use Cases miteinander zu vergleichen. Bei diesem Vergleich stellte sich die ElasticSearch als beste Option für den Geo Service heraus. Anschließend daran wurde eine geeignete Architektur für den Import von Hotels in das Productrepository mit der Erweiterung des Geo Service evaluiert. Dazu wurden zwei Ansätze gegenübergestellt und abgewägt, welche die bessere Option darstellt. Dabei fiel die Entscheidung auf eine Architektur mit einer RabbitMQ, Elasticsearch und mehreren Services. Anschließend musste ein Konzept für den Geo Service erstellt werden. Hierbei wurde das Datenbankschema definiert und die Funktionsweise des Geo Service beschrieben. Das spezifizierte Konzept wurde folgend in dem Kapitel 6 „Umsetzung des Geo-Service“ umgesetzt. Dabei wurde eine SpringBoot Application in Kotlin erstellt und die notwendigen Daten für den Service importiert. Abschließend an die Implementierung wurde die Umsetzung anhand der Akzeptanzkriterien beurteilt. Im Zuge dessen wurde festgestellt, das alle Akzeptanzkriterien mit der Ausnahme, das alle Hotels in einer touristischen Region gefunden werden, erfüllt wurden. Trotzdessen kommt der Geo Service produktiv zum Einsatz und ermöglicht es, über das Frontend, dem Endkunden, Hotels mit Hilfe von Regionen zu suchen. Abschließend wurden im Abschnitt 6.4: „Erkenntnisse der Umsetzung“ die Bestandteile der Umsetzung zur Lösung des spezifischen Problems der Hotelsuche aufgegriffen, die auch bei anderen Problemen im Bereich der Geosuche angewendet werden können.
Die Suche nach Hotels über Geosuchen beziehungsweise über Regionen konnte mit dem Geo Service dem Endkunden ermöglicht werden. Damit wurden die Suchmöglichkeiten für

den Endkunden um den Kernbestandteil der Suche erweitert. Es wurde geschafft, die Last die durch die neue Suchmöglichkeit entsteht, von der Suchanfrage des Kunden zu entkoppel. Dafür wurde auf eine Batchimportarchitektur gesetzt die nächtlich asynchron den Import durchführt und dabei in dieser Zeit nicht genutzte Ressourcen verwendet. In diesem Batch wird die Vorarbeit für die Geosuche geleistet, damit die Kundenanfrage zum späteren Zeitpunkt schnellst möglich beantwortet werden kann. Mit diesem Verfahren konnte der Geo Service ohne Probleme in die bestehende Architektur eingebunden werden und arbeitet nun mit den anderen bereits bestehenden Komponenten zusammen. Nicht optimal bei der aktuellen Umsetzung ist jedoch, das der Kunde nur nach Städten, Regionen, Bundesländern und Ländern suchen kann, die auch im Datensatz des Geo Service enthalten sind.
Abschließend ist zu sagen, dass das Projekt mit dem produktiven Einsatz des Geo Service sein Ziel erreicht hat. In der weiteren Entwicklung gilt es den Geo Service zu verbessern und um zusätzliche Funktionen zu erweitern. So soll unter anderem der Datensatz optimiert werden. Hierfür wird angedacht diesen entweder durch eine bessere Alternative zu ersetzen oder diesen manuell aufzubessern. Darüber hinaus wird ein Tool notwendig, das es ermöglicht die Regionen einfach zu editieren, neue Regionen hinzuzufügen und bestehende Regionen zu löschen. Dafür muss ein Tool gefunden werden, welches dies auch nicht technikaffine Kollegen ermöglicht, die genannten Änderungen vorzunehmen. Außerdem soll die Möglichkeit geprüft werden, ob weitere externe Api's angebunden werden können, um auch Suchen nach Städten, Regionen und Ländern zu ermöglichen, die nicht im Datensatz des Geo Service enthalten sind.

A Sequenzdiagramm der „neuen“ Suche



B Architekturdiagram des Productrepositories

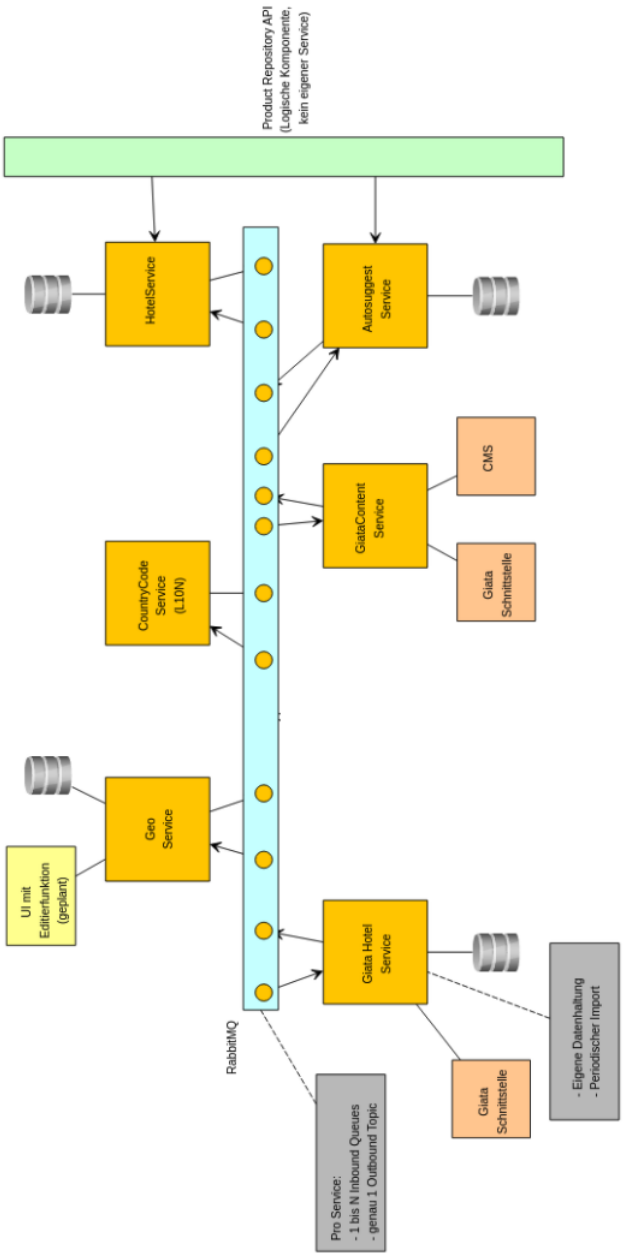


Abbildung B.1: Architekturdiagramm des Productrepositories

C Übersprungene Polygone beim Import in die MongoDB

35 Stk MarketingRegionen:
Emirat Abu Dhabi
Marketing Region
Queensland
Belize-Süd
Papua-Neuguinea

Citys:
Battlement Mesa
Summit Park
&Re
Chiu-Wu-Kung
Da Nang
Estacion Mata Ortiz
Ho-Chih-Hsien
Onokhoy
Guamuchil
Hoi An
Phan Thiet
Lao Giao
Pleiku
Bloomfield
Farmington
Delta
Ascension
Zhong Xian
Nuevo Casas Grandes
None
Hermagor-Pressegger See

Sand
Sosua
Phan Thiet
Worland
Sand
None
Horumersiel-Schillig
Kyle
Greybull

D Auszug von Geoabfragen aus (PostGIS Project, 2021, PostGIS Reference)

ST_Contains — Returns true if and only if no points of B lie in the exterior of A, and at least one point of the interior of B lies in the interior of A.
ST_ContainsProperly — Returns true if B intersects the interior of A but not the boundary (or exterior). A does not contain properly itself, but does contain itself.

ST_Covers — Returns 1 (TRUE) if no point in Geometry B is outside Geometry A
ST_CoveredBy — Returns 1 (TRUE) if no point in Geometry/Geography A is outside Geometry/Geography B

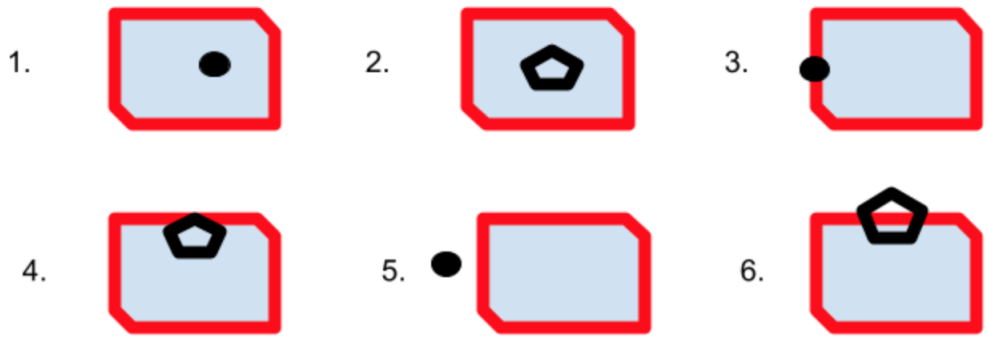
ST_Intersects — Returns TRUE if the Geometries/Geography ßpatially intersect in 2D- (share any portion of space) and FALSE if they don't (they are Disjoint). For geography tolerance is 0.00001 meters (so any points that close are considered to intersect)
ST_Intersection — (T) Returns a geometry that represents the shared portion of geomA and geomB.

ST_Within — Returns true if the geometry A is completely inside geometry B

ST_Distance — Returns the distance between two geometry or geography values.
ST_3DDistance — Returns the 3D cartesian minimum distance (based on spatial ref) between two geometries in projected units.
ST_DistanceSphere — Returns minimum distance in meters between two lon/lat geometries using a spherical earth model.
ST_DistanceSpheroid — Returns the minimum distance between two lon/lat geometries using a spheroidal earth model.

ST_GeomFromText — Return a specified ST_Geometry value from Well-Known Text representation (WKT).

Zum Weiteren Verständniss der Abfragen sind hier einige der Abfragen visualisiert:



1. ST_Intersects = true, ST_Covers = true, ST_Contains = true.
2. ST_Intersects = true, ST_Covers = true, ST_Contains = true.
3. ST_Intersects = true, ST_Covers = true, ST_Contains = **false**.
4. ST_Intersects = true, ST_Covers = true, ST_Contains = true.
5. ST_Intersects = **false**, ST_Covers = **false**, ST_Contains = **false**.
6. ST_Intersects = true, ST_Covers = **false**, ST_Contains = **false**.

Abbildung D.1: Quelle: (Entin, 2019)

E GeoJSON Beispiel

```
{
    "type": "FeatureCollection",
    "name": "Countrys",
    "crs": { "type": "name", "properties": { "name": "urn:ogc:
       def:crs:OGC:1.3:CRS84" } },
    "features": [
    {
    "type": "Feature", "properties": { "OBJECTID": 37.0, "NAME
       ": "Bahrain", "LEGACYID": "L_24", "CUSTOM": 0.0, "
       COMPANYTYP": "DER", "SHAPE_AREA": 0.0, "SHAPE_LEN": 2.0
       }, "geometry": { "type": "MultiPolygon", "coordinates":
       [ [ [ [ 50.602439, 26.22354 ], [ 50.602736, 26.218932 ],
        ... ]]] } },
    {
    "type": "Feature", "properties": { "OBJECTID": 37.0, "NAME
       ": "Germany", "LEGACYID": "L_24", "CUSTOM": 0.0, "
       COMPANYTYP": "DER", "SHAPE_AREA": 0.0, "SHAPE_LEN": 2.0
       }, "geometry": { "type": "MultiPolygon", "coordinates":
       [ [ [ [ 13.602439, 52.22354 ], [ 12.22354, 51.218932 ],
       ... ]]] }
    },
     ...
    ]
}
```

Komplettes Hotelobjekt aus dem PR anhägen

F Beispielhafte Geosuche auf der neuen Plattform

Screenshot einfügen

Literaturverzeichnis

Bundesverband der Deutschen Tourismuswirtschaft (BTW) e.V. (n. d.). *Wirtschaftsfaktor Tourismus*. Verfügbar 17. Februar 2021 unter http://www.btw.de/tourismus-in-zahlen/wirtschaftsfaktor-tourismus.html

Der deutsche Reisemarkt: Zahlen und Fakten 2019. (n. d.). *Deutscher Reiseverband*.

DER Touristik Central Europe GmbH. (2020a). *REWE Group*. Verfügbar 22. Februar 2021 unter https://www.dertouristik.com/de/marken/reiseveranstalter

DER Touristik Central Europe GmbH. (2020b). *REWE Group*. Verfügbar 22. Februar 2021 unter https://www.dertouristik.com/de/unternehmen/rewe-group/

DER Touristik Central Europe GmbH. (2020c). *Spezialisten*. Verfügbar 22. Februar 2021 unter https://www.dertouristik.com/de/marken/spezialisten

DER Touristik Central Europe GmbH. (2020d). *Unternehmen*. Verfügbar 22. Februar 2021 unter https://www.dertouristik.com/de/unternehmen/

DER Touristik Online GmbH. (2020). *WIR ÜBER UNS*. Verfügbar 22. Februar 2021 unter https://www.dertouristik.online/ueber-uns/

Dörnberg, A. v., Freyer, W. & Sülberg, W. (2017). *Reiseveranstalter- und Reisevertriebs-Management: Funktionen – Strukturen – Prozesse*. Walter de Gruyter GmbH & Co KG.

Entin, M. (2019). *Contains/Covers/Intersects/Within?* Verfügbar 4. März 2021 unter https://mentin.medium.com/which-predicate-cb608b470471

Esri. (2018). *Datenbankverbindungen in ArcGIS for Desktop*. Verfügbar 8. März 2021 unter https://desktop.arcgis.com/de/arcmap/10.3/manage-data/databases/database-connections-desktop.htm#ESRI_SECTION1_B0ED7CB88C2C4928B74088FEC5583C84

Freyer, W. (2015). *Tourismus - Einführung in die Fremdenverkehrsökonomie*. Walter de Gruyter GmbH & Co KG.

Fuchs, W. (2021). *Tourismus, Hotellerie und Gastronomie von A bis Z*. Walter de Gruyter GmbH & Co KG.

Kennedy, M. D. (2013). *Introducing Geographic Information Systems with ArcGIS - A Workbook Approach to Learning GIS*. John Wiley Sons.

Meier, A. & Kaufmann, M. (2016). *SQL- NoSQL-Datenbanken* -. Springer-Verlag.

PostGIS Project. (2021). *PostGIS Reference*. Verfügbar 4. März 2021 unter https://postgis.net/docs/reference.html

QGIS project. (2021, 8. März). *13.1.2. The DB Manager*. Verfügbar 8. März 2021 unter https://docs.qgis.org/3.16/en/docs/user_manual/managing_data_source/opening_data.html#the-db-manager

Reiseanalyse 2020: Erste ausgewählte Ergebnisse der 50. Reiseanalyse zur ITB 2020. (n. d.). *FUR Forschungsgemeinschaft Urlaub und Reisen e.V.* https://reiseanalyse.de/wp-content/uploads/2020/03/RA2020_Erste-Ergebnisse_DE.pdf

Robert Koch-Institut. (2021). *Robert Koch-Institut: COVID-19-Dashboard*. Verfügbar 8. März 2021 unter https://experience.arcgis.com/experience/478220a4c454480e823b17327b2bf1d4

Schicker, E. (2017). *Datenbanken und SQL - Eine praxisorientierte Einführung mit Anwendungen in Oracle, SQL Server und MySQL*. Springer-Verlag.

Statistisches Bundesamt. (2017). *Automobilindustrie trägt 4,5 % zur Bruttowertschöpfung in Deutschland bei*. Verfügbar 17. Februar 2021 unter https://www.destatis.de/DE/Presse/Pressemitteilungen/2017/09/PD17_326_811.html