HANDBUCH ZUM NUTZUNG SERVER-KOMPONENTE UND CLIENT-KOMPONENTE

Inhaltsverzeichnis:

EINLEITUNG

Vorwort

Die Anwendung ist browserbasiert. Die Adminstration läuft in folgenden Browsern:

```
- Internet Explorer
- Mozilla Firefox
- Safari
- Chrome
```

Dieses Handbuch ist zur Erläuterung der Verwendung des gemeinschaftliche Gruppenabschlussprojekt PRG-Praktikum WiSe 2021/2022 "Parliament Sentiment Radar" zuständig.
Die Lösung dieses Gruppenprojekts haben sich die Teilnehmerinnen jeweils in Gruppen zusammengefunden. Hierzu verwenden wir, wie auch schon in den vorherrigen Übungen, GitLab.

Um alle Anwendungsfunktionen nutzen zu können, müssen Sie in Ihren Browsereinstellungen Folgendes zulassen:

```
- Ausführung von Java Script
- Verwendung von Sitzungscookies
- Öffnen von Popup-Fenstern
```

Vorbereitung

▼ Hide Information
Unsere Gruppe hat zur Bearbeitung der Abschlussaufgabe ein privates Projekt namens "Gruppe_8_Mittwoch_3_ParliamentSentimentRadar" in Gitlab gestellt und hat unserer Tutorin das Projekt mit der Berechtigung lesend frei gegeben.
Alles JavaScript-Dateien wurden ausführlich dokumentiert.

Das "Parliament Sentiment Radar" besteht aus zwei Komponenten: einer Server-Komponente, implementiert in Java, sowie einer Client-Komponente, implementiert in JavaScript unter Verwendung von SB Admin 2 und beinhaltet folgenden Funktionalitäten:

```
1.  Einlesen von Parlamentsdebatten des Deutschen Bundestages im XML-Format.
2.  Analysieren, Aufbereiten und datenbankgestütztes Abspeichern der abgefragten Parlamentsreden.
3.  Verarbeitung der Parlamentsreden mittels angegebener NLP-Verfahren.
4.  Datenbankgestütztes Abfragen und Auswerten der Ergebnisse.
5.  Implementierung eines RESTful Webservice zur Bereitstellung  der Parlamentsreden für die Client-Komponente.
6.  Implementierung einer Visualisierungskomponente unter Verwendung von SB Admin 2 gemäß Aufgabe 3.
```

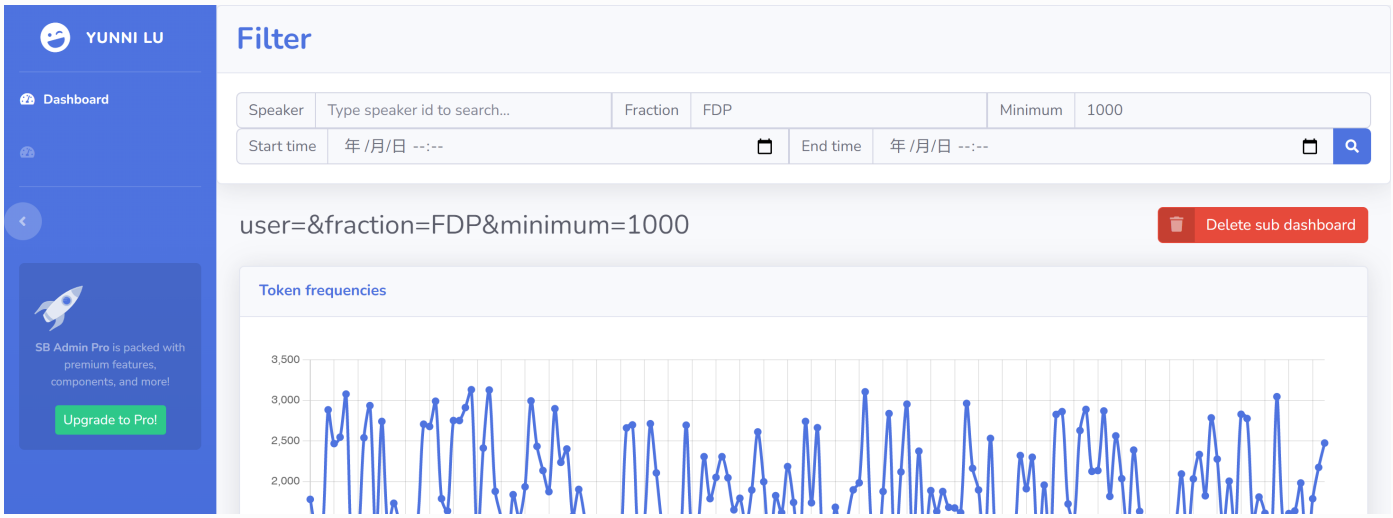
Unsere Server-Komponente ist unter Verwendung von Java Spark als RESTful Webservice implementiert.

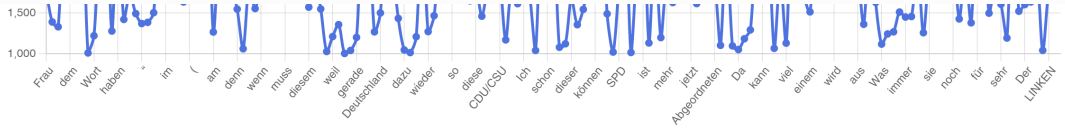
Zum Testen und Dokumentieren der REST-Abfragen verwenden wir **swagger.io**.

FRONTEND

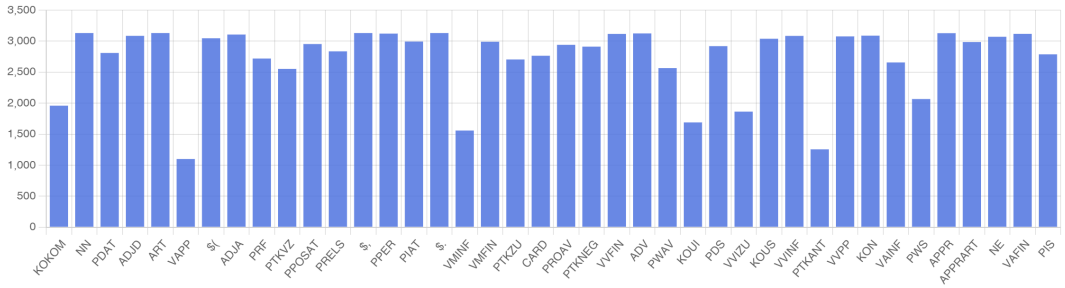
Die Hauptaufgabe von Frontend-Teil dient dazu, die Ergebnisse mittels SB Admin 2 gemäß der definierten Aufgaben zu visualisieren und analysieren. Es gibt zwei Websites in unsere Frontend-Teil.

Main Website - Dashboard

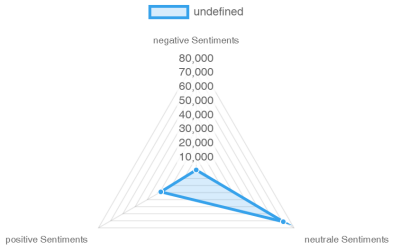




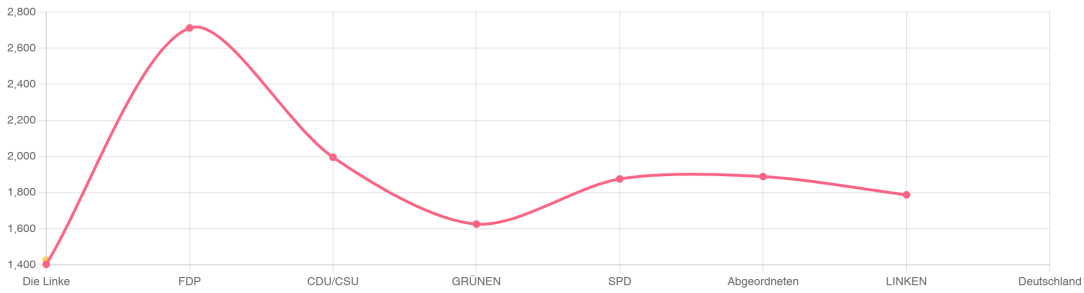
POS frequencies



Sentiment frequencies



NamedEntity frequencies



Speaker frequencies

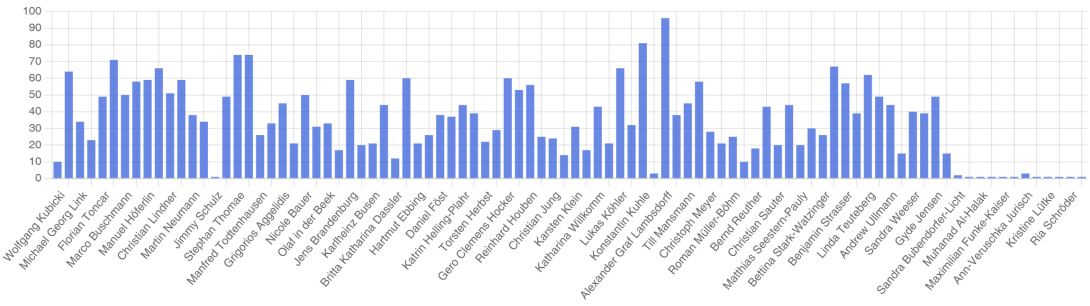


Chart Visualisation

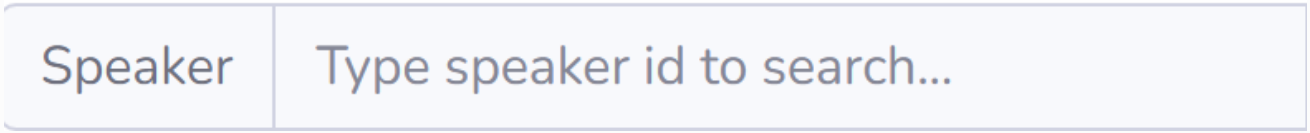
Das Dashboard "**Übersicht**" visualisiert die gesamte Parlament Daten aus unsere Datenbank:

```
- Tokenchart
- Poschart
- Sentimentchart
- Named Entities chart
- Rednerchart mit Redner-Bild als Hover-Effekt
```

Filter Funktion

Man kann im oberen Seitenbereich je nach dem Suchfeld frei entscheiden, ob man durch Dropdown-Menü oder selbst tippen, die entsprechenden Filterergebnisse suchen:

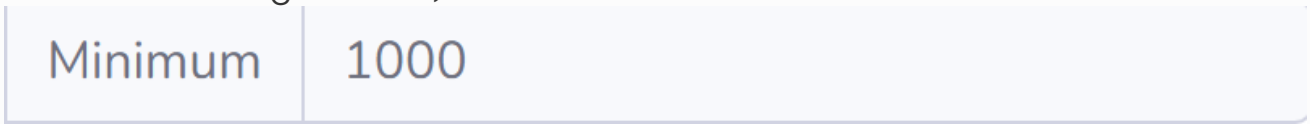
- Speaker-ID (Format: Dropdown-Menü oder 8-stellige ID-Zahlen wie z.B. 11004908)

- Fraktionsnamen,
  hier ist in dem Bild als Beispiel "FDP" geschrieben (Format: Dropdown-Menü oder Fraktionszugehörigkeit)

- minimale Anzahl z.B. für Token, Pos usw.
  hier ist in dem Bild als Beispiel "1000" geschrieben (Format: Dropdown-Menü oder 3 bis 4-stellige Zahlen)

- Beginn- und End-Zeitraum (Format: Dropdown-Menü)
- 

Es wird nach Klicken  ein sogenannte "**Sub-Dashboard**" angezeit (mit Suchanfragetitel), dies enthält die gleichen Visualisierungen und Optionen wie dem "Main" Dashboard, jedoch nur für die Suchanfrage.

In dem Bild sieht man, dass man kein SpeakerID aber Fraktion "FDP" und einen minimale Anzahl "1000" eingegeben hat.

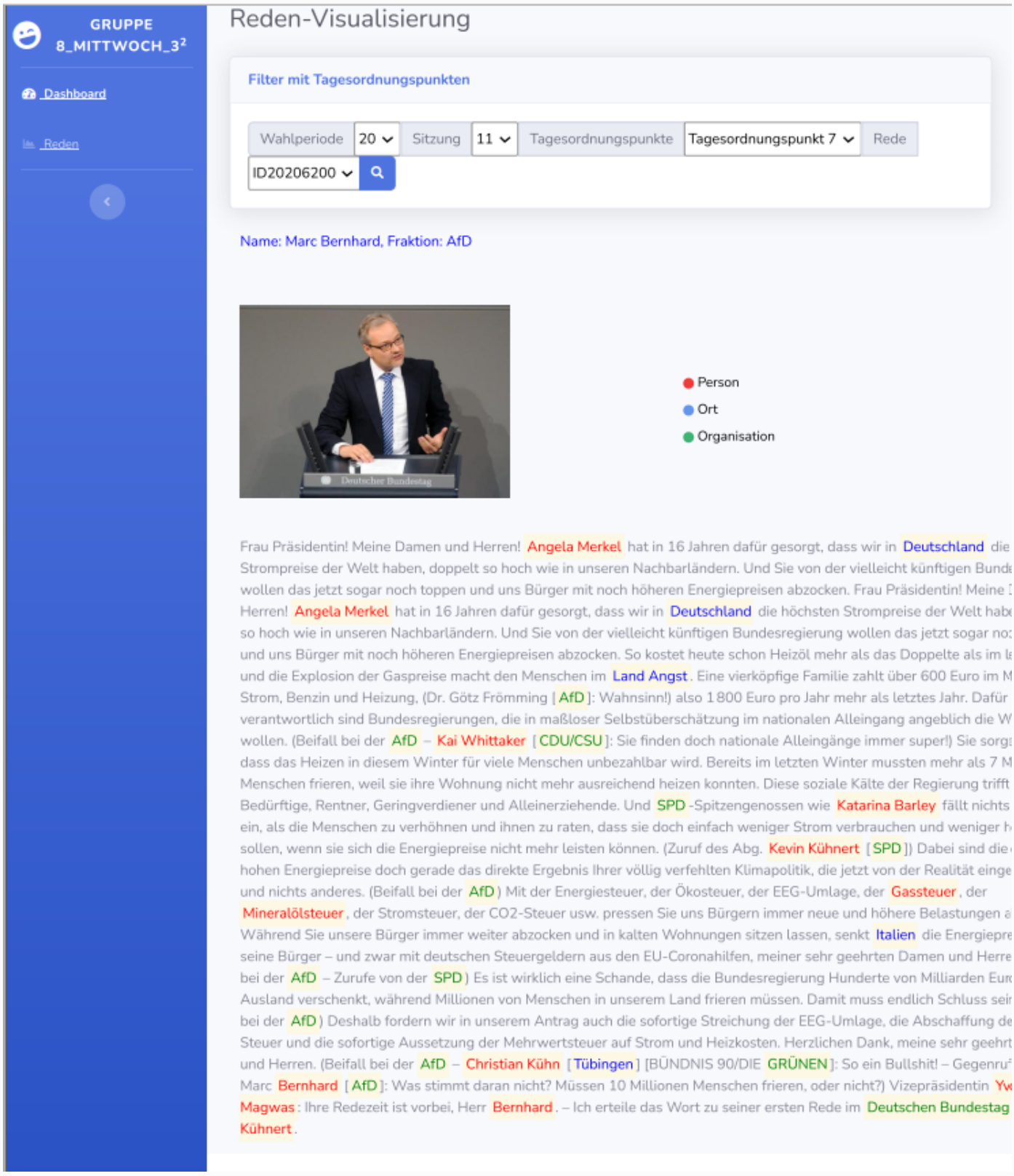
```
user=&fraction=FDP&minimum=1000
```

Das Sub-Dashboard kann durch Button "Delete sub dashboard" wieder entfernt werden.

Sub Website - Reden

Volltext Funktion

Links bei der Sidebar unter "Dashboard" sieht man die Website "reden". Die Volltext-Visualisierung ist für jede einzelne Parlaments-Rede implementiert und beinhaltet folgenden Funktionen:

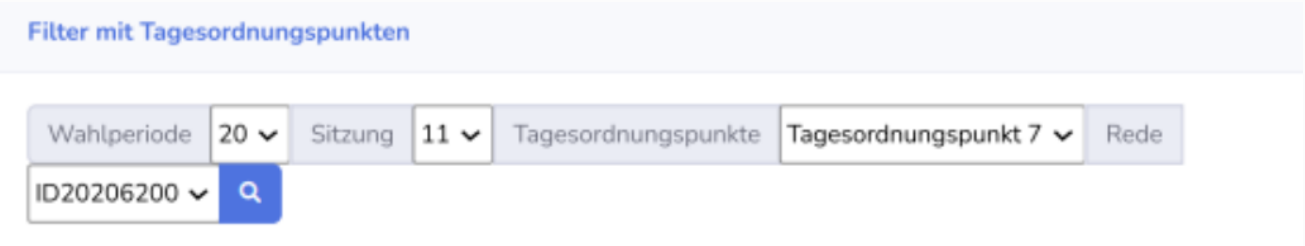


Alle im Text gefundenen Named Entities werden farblich hinterlegt.

Person

Ort

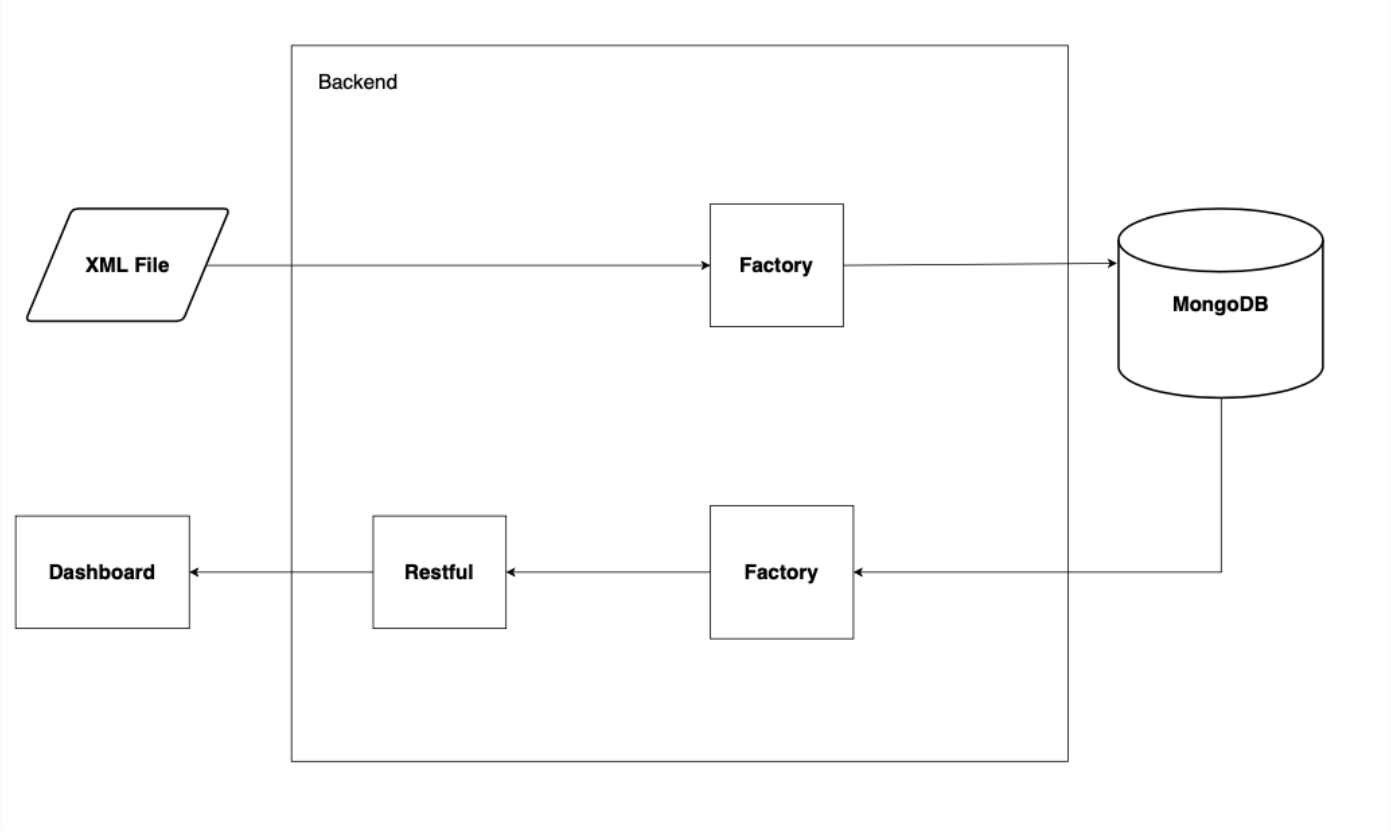
Organisation

- Der Sprech*innenname, Informationen zu seiner/ihrer Partei- und Fraktionszugehörigkeit, sowie ein Foto(sofern vorhanden) werden im Kopf der Rede angezeigt. (Bei dem obigen Beispiel ist der Abgeordnete Name: Marc Bernhard, Fraktionszugehörigkeit: AfD)
- Ermöglichung eines Filters über eine Navigation durch die Tagesordnung der Protokolle.
- 

- Man wählt mit dem DropDown-Menü in der Reihenfolge:
    1. Wahlperiode
    2. Sitzung
    3. Tagesordnungspunkte
    4. RedeID

BACKEND

Flussdiagramm für Backend:



Zuerst lesen die XML File ein. Dann verwendet Factory, um die Daten in unsere eigene Datenstruktur zu speichern. Dann schreiben die Daten in MongoDB ein. Verwendet wieder die Fabrik, um Daten aus MongoDB zu lesen. Und implementiert Restful -Schnittstelle. Auf diese Weise kann der Frontend über diese Restful-Schnittstelle visualisiert werden.

Funktionsweise von Backend ist nicht relevant für Benutzer. Deshalb werden hier keine Details genannt. Die Details werden während des Vorrechnen erläutert.

XML File einzulesen

Führt die *Main-Methode* der ProtocolMongoDBWriter-Klasse aus. Der Object *handler* für die Klasse MongoDBConnectionHandler wird über die Main-Methode erstellt. Wir können dann eine Verbindung zu MongoDB herstellen. Die Parlamentstexte werden direkt vom Bundestag heruntergeladen. Die XML-Datei wird dann über ParliamentFactory eingelesen.

```
public static void main(String[] args) {
  MongoDBConnectionHandler handler = new
MongoDBConnectionHandler("config/config.json");
  String protocolDirectory = "Daten/20. Wahlperiode";
  ParliamentFactory factory = new ParliamentFactory_Impl();
  factory.initFromDirectory(protocolDirectory);
//    writeProtocols(factory.getProtocols(), handler);
  writeSpeakers(factory.getParliamentMembers(), handler);
//    writeSpeechs(factory.getSpeeches(), handler);
//    writeFractions(factory.getFractions(), handler);
}
```

Methode von MongoDB

Über das Objekt handler von MongoDBConnectionHandler kann man folgende Methode verwenden.
Diese Methode werden in Klasse MongoDBConnectionHandler gespeichert.

- Erstellen
- Einschreiben
- Lesen
- Updaten
- Löschen

Erstellen

Wenn wir das Project ausführen, wird MongoDBConnectionHandler schon ergestellt.

Einschreiben

1.Schreib nur ein Dokument in MongoDB ein:
Verwenden diese Methode *writeDocument* von Klass MongoDBConnectionHandler:

```
public void writeDocument(String collectionName, Document
document){
  try{
      MongoCollection<Document> collection =
this.getCollection(collectionName);
      collection.insertOne(document);
    } catch (Exception e){
        e.printStackTrace();
    }
}
```

Mit diesem Befehl kann man einen Dokument in eine spezielle Collection in MongoDB einschreiben.

```
handler.writeDocument(collectionName, document);
```

2.Schreib mehrerer als eins Dokumente in MongoDB ein:
Verwenden diese Methode *writeDocuments* von Klass MongoDBConnectionHandler:

```
public void writeDocuments(String collectionName,
List<Document> documents){
  try{
      MongoCollection<Document> collection =
this.getCollection(collectionName);
      collection.insertMany(documents);
  } catch (Exception e){
      e.printStackTrace();
  }
}
```

Mit diesem Befehl kann man mehrere Dokumente in eine spezielle Collection in MongoDB einschreiben.

```
handler.writeDocuments(collectionName, documents);
```

Lesen

Suche nach einem bestimmten Dokument in einer bestimmten Collection in MongoDB.
Verwenden diese Methode *getDocument* von Klass MongoDBConnectionHandler:

```
1  public Document getDocument(String collectionName, Bson
   query){
2    try{
3        MongoCollection<Document> collection =
   this.getCollection(collectionName);
4        return collection.find(query).first();
5    } catch (NullPointerException e){
6
    System.err.println("MongoDBConnectionHandler.getDocument:
   collection " + collectionName + " not found!");
7        return null;
8    } catch (Exception e) {
9        e.printStackTrace();
1        return null;
0    }
1  }
```

Kann man durch diesem Befehl erreichen:

```
1  handler.getDocument(collectionName, query);
```

Updaten

Mit diesem Method kann man die Dokument, der in eine spezielle Collection in MongoDB ist, updaten.
Verwenden diese Methode *updataDocument* von Klass MongoDBConnectionHandler:

```
1  public boolean updataDocument(String collectionName, Bson
   query, Document newDocument){
2    MongoCollection<Document> collection =
   this.getCollection(collectionName);
3    try{
```

```
4       UpdateResult result = collection.replaceOne(query,
newDocument);
5       if (result != null && result.getModifiedCount() == 0 &&
result.getMatchedCount() == 0){
6           try {
7
 this.getCollection(collectionName).insertOne(newDocument);
8               return true;
9           }
1           catch (Exception e){
0               System.out.println(e.getMessage());
1           }
2       }
3   } catch (Exception e){
4       e.printStackTrace();
5   }
6   return false;
7 }
```

Kann man mit diesem Befehl erreichen:

```
1 handler.updataDocument(collectionName, query, newDocument);
```

Wenn die Dokument erfolgreich updatet wird, geben ture aus.

Löschen

Mit diesem Method kann man die Dokument, der in eine spezielle Collection in MongoDB ist, löschen.
Verwenden diese Methode *deleteDocument* von Klass MongoDBConnectionHandler:

```
public Boolean deleteDocument(String collectionName, Bson query){
  try {
      MongoCollection<Document> collection = this.getCollection(collectionName);
      DeleteResult result = collection.deleteOne(query);
      System.out.println("Deleted document count: " + result.getDeletedCount());
      return true;
  } catch (Exception e) {
//      System.err.println("Unable to delete due to an error: " + e);
      e.printStackTrace();
      return false;
  }
}
```

Kann man mit diesem Befehl erreichen:

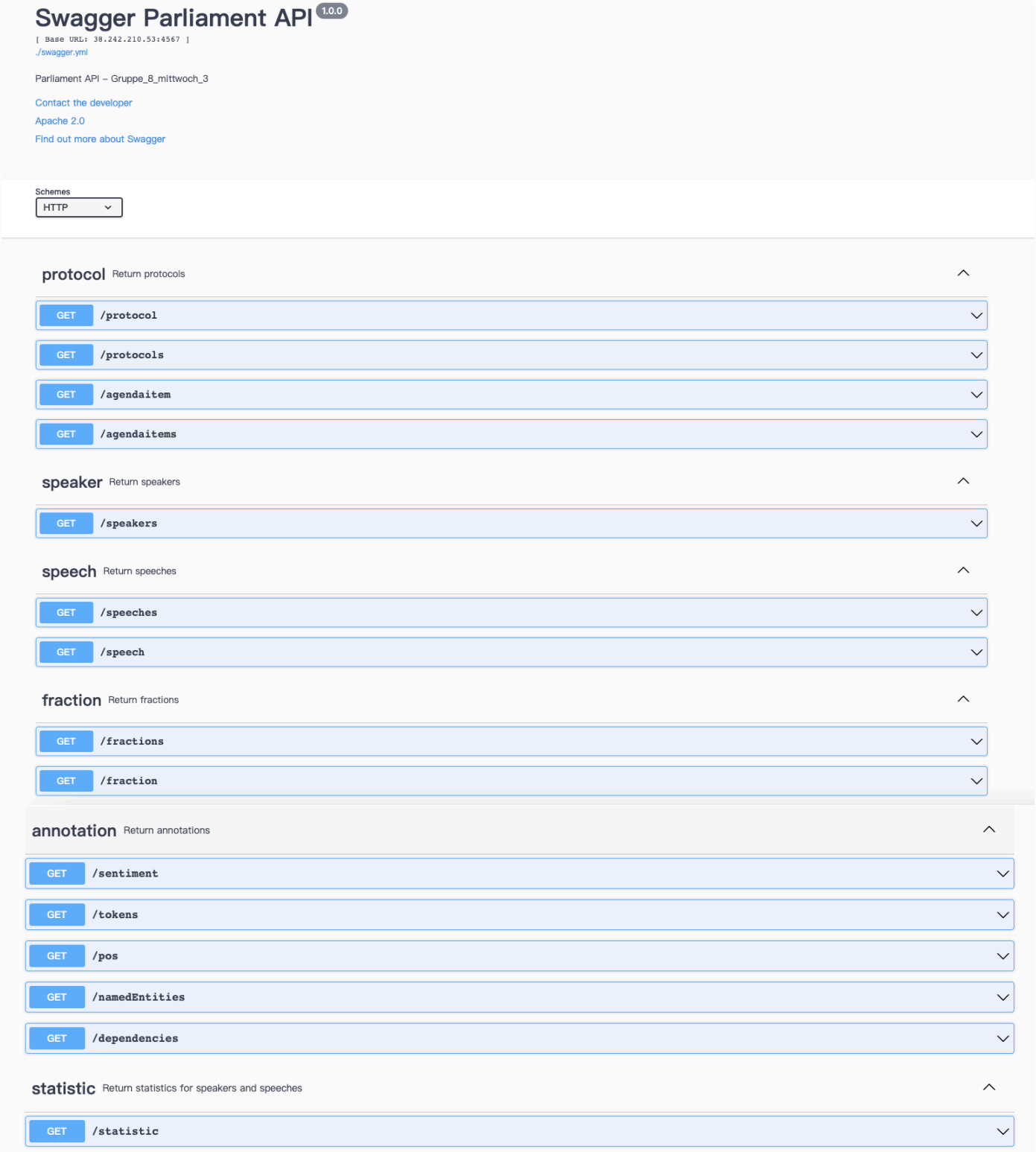
```
handler.deleteDocument(collectionName, query);
```

Wenn die Dokument erfolgreich gelöscht wird, geben ture aus.

RESTful Schnittstelle

Wenn die XML File eingelesen und NLP analysiert werden, werden durch ParliamentFactory in MongoDB eingeschriben. Aber das Frontend kann jedoch keine Daten direkt von MongoDB für die Visualisierung abrufen. Deshalb verwenden wir hier RESTful Schnittstelle.

Hier ist unsere Swagger.

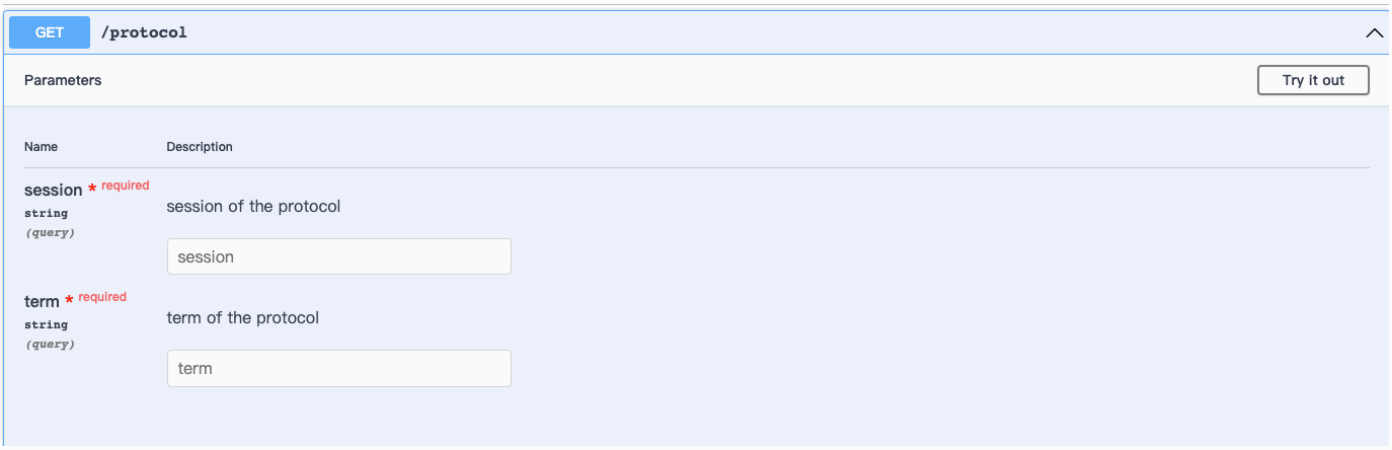


Return protocols

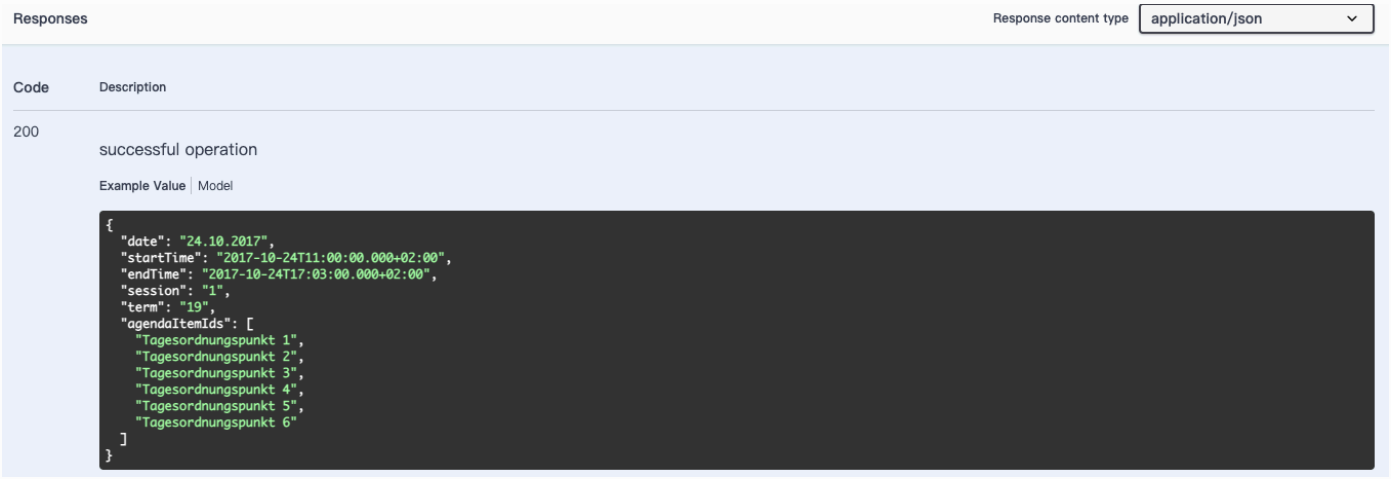
Get protocol:

Sollt man hier zwei Parameters *session* und *term* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:



Struktur der Ergebnisse:

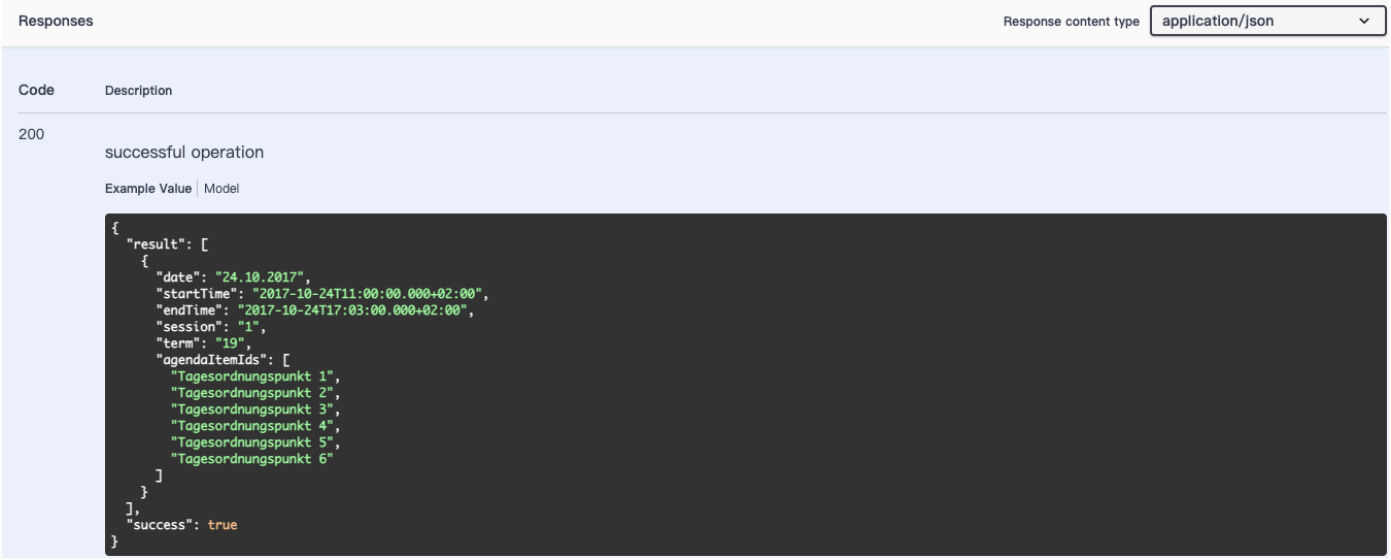


Get protocols:

Hier müssen keine Parameter eingegeben werden.

Klicken Sie auf die Schaltfläche *Try it out*:

Struktur der Ergebnisse:

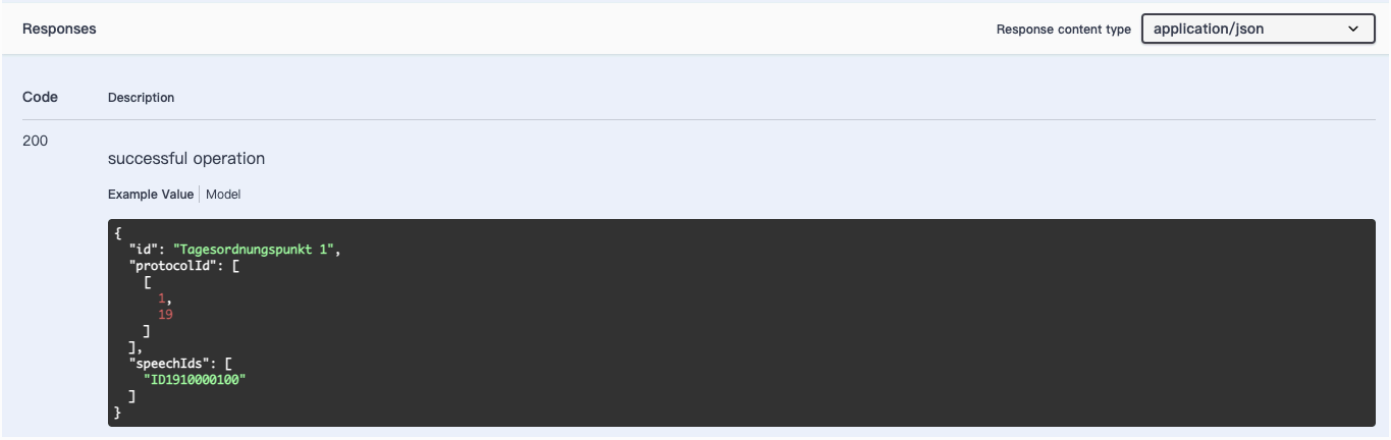


Get agendaitem:

Sollt man hier drei Parameters *session*, *term* und *id* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:

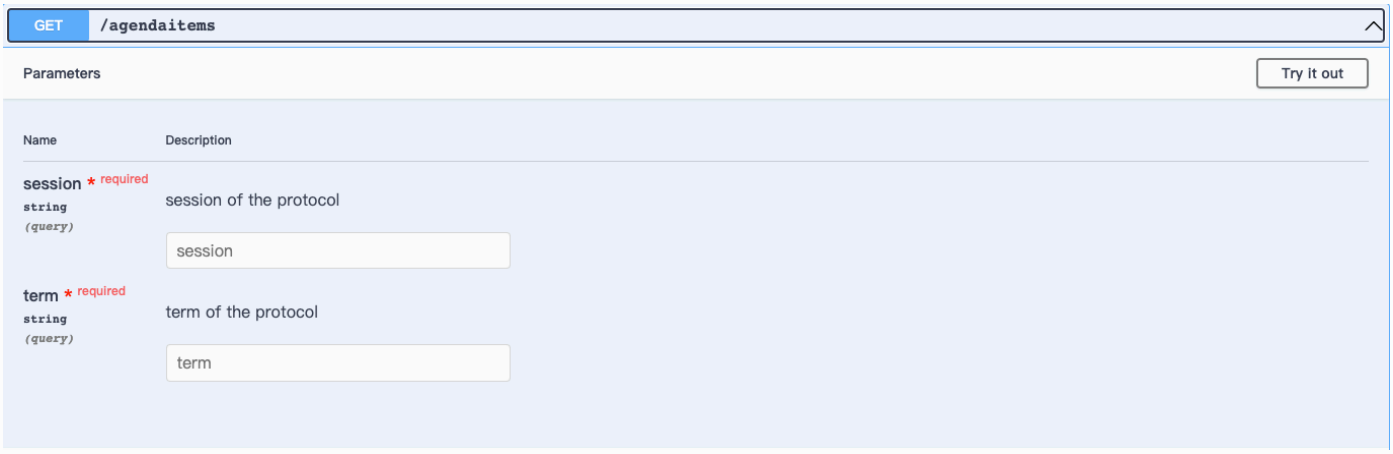
Struktur der Ergebnisse:



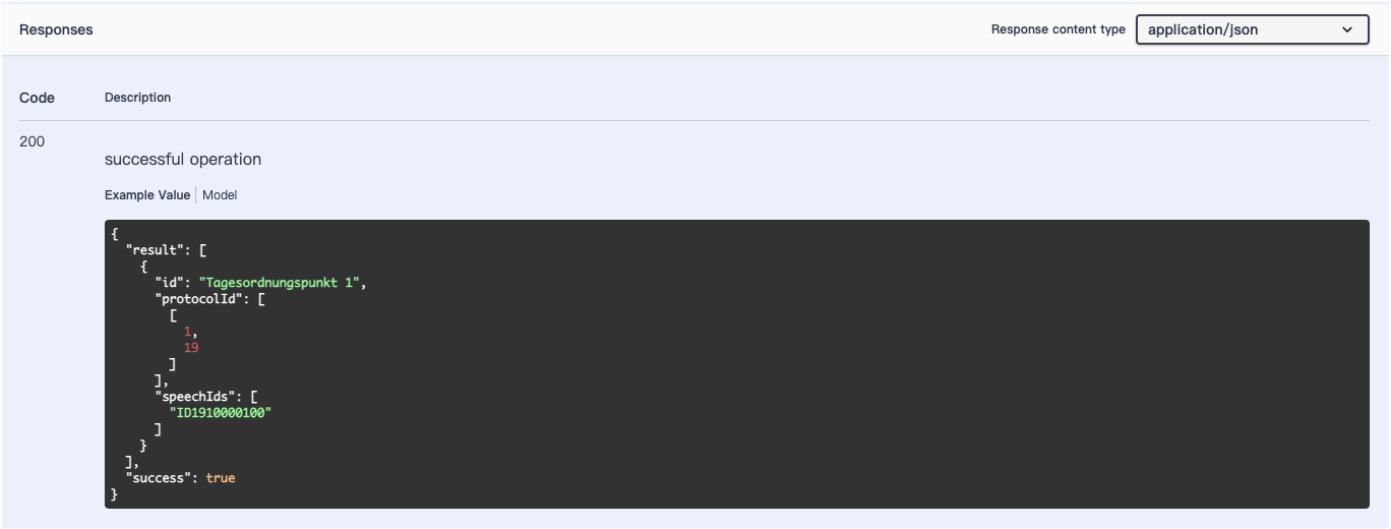
Get agendaitems:

Sollt man hier zwei Parameters *session* und *term* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:



Struktur der Ergebnisse:

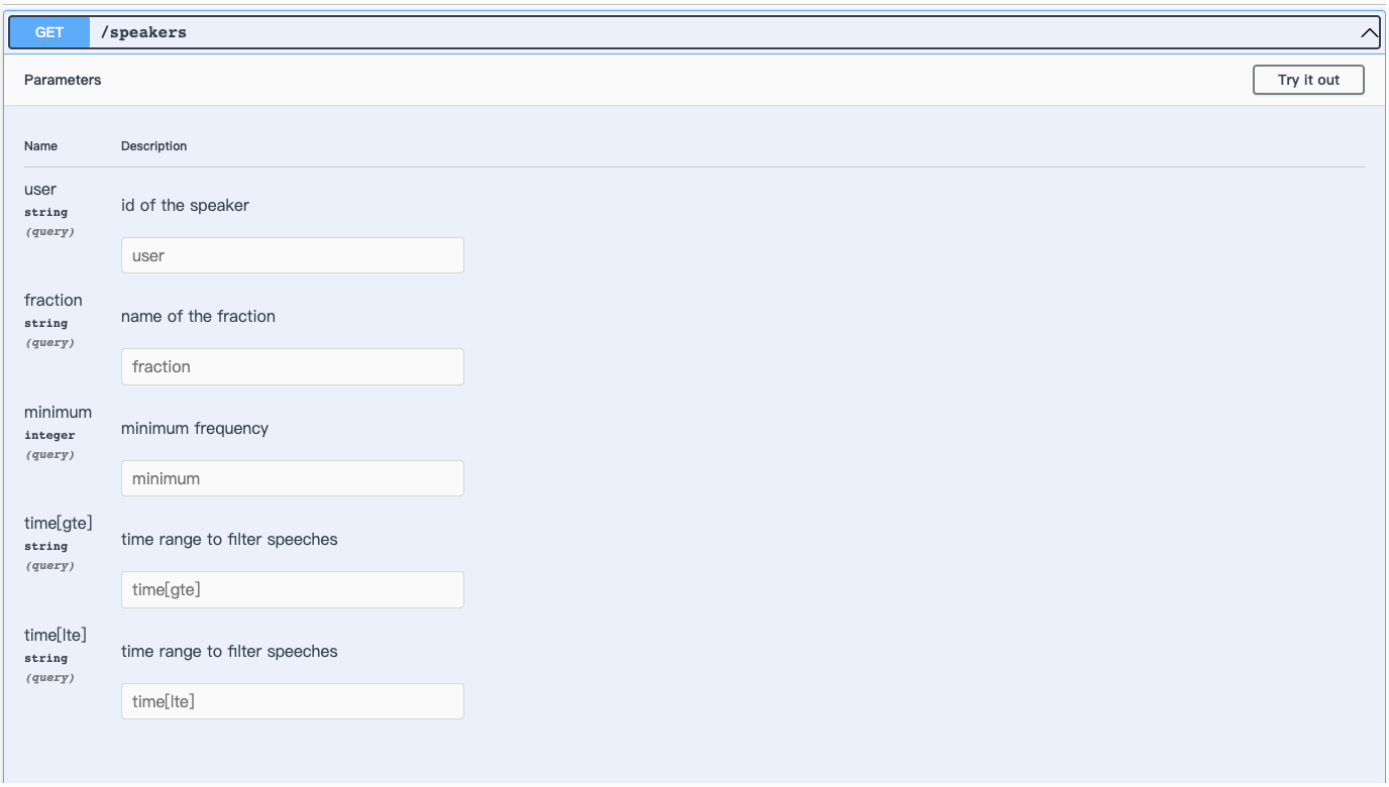


Return speakers

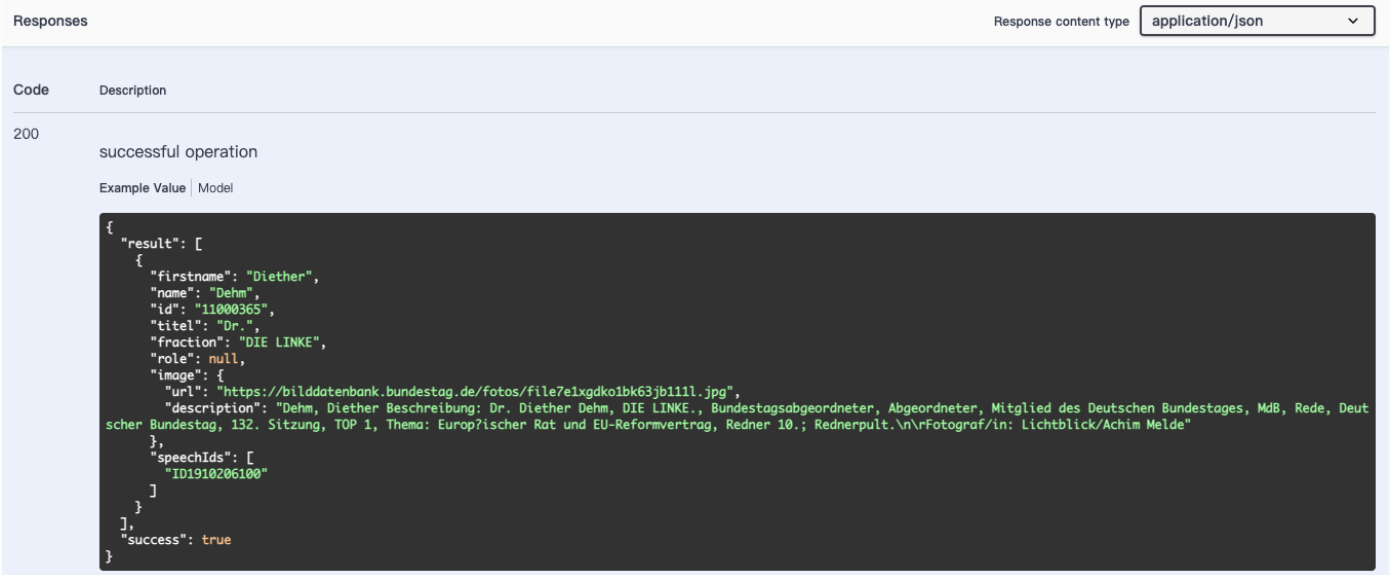
Get speakers:

Sollt man hier fünf Parameters *user*, *fraction*, *minimum*, *time* and *time* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:



Struktur der Ergebnisse:

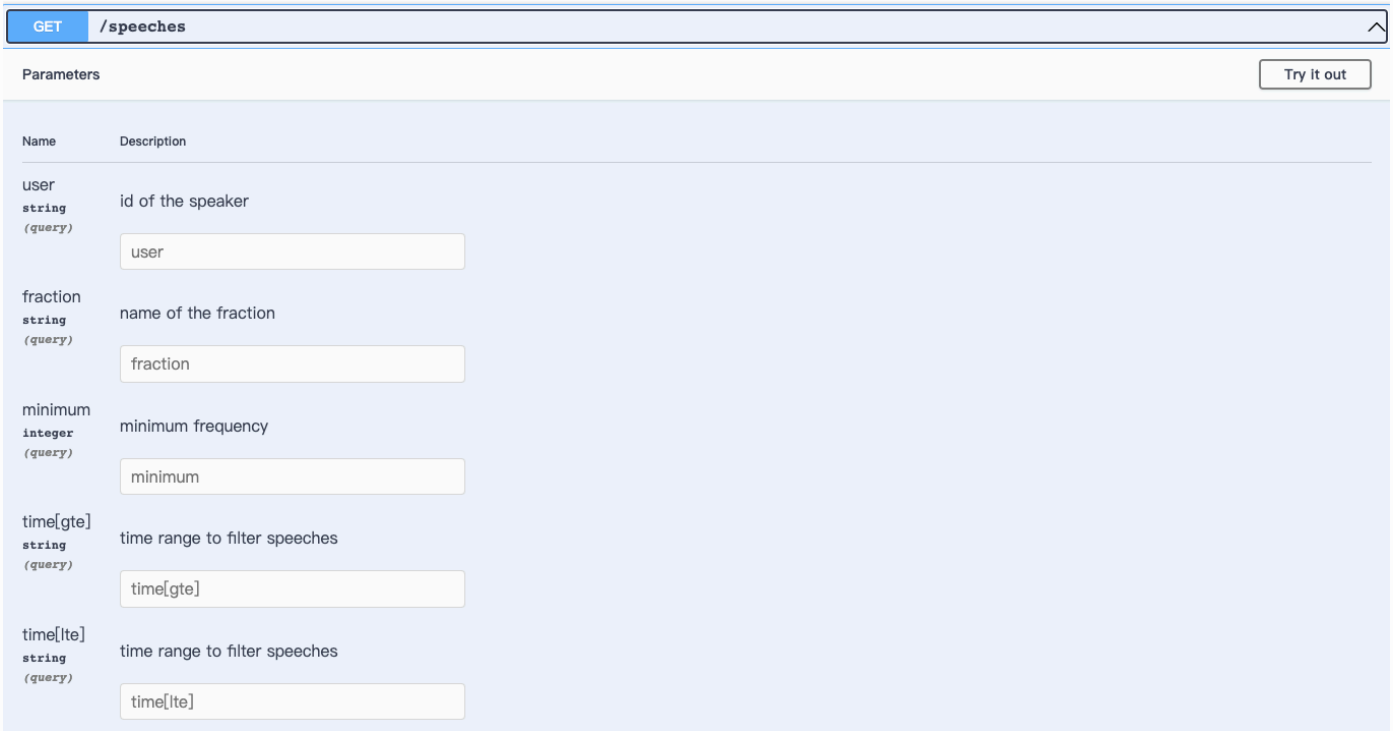


Return speeches

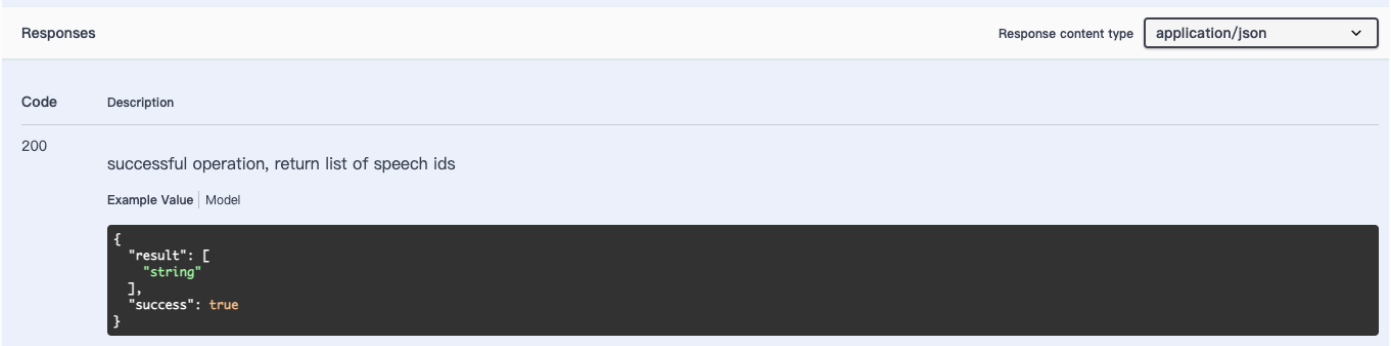
Get speeches:

Sollt man hier fünf Parameters *user*, *fraction*, *minimum*, *time* and *time* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:



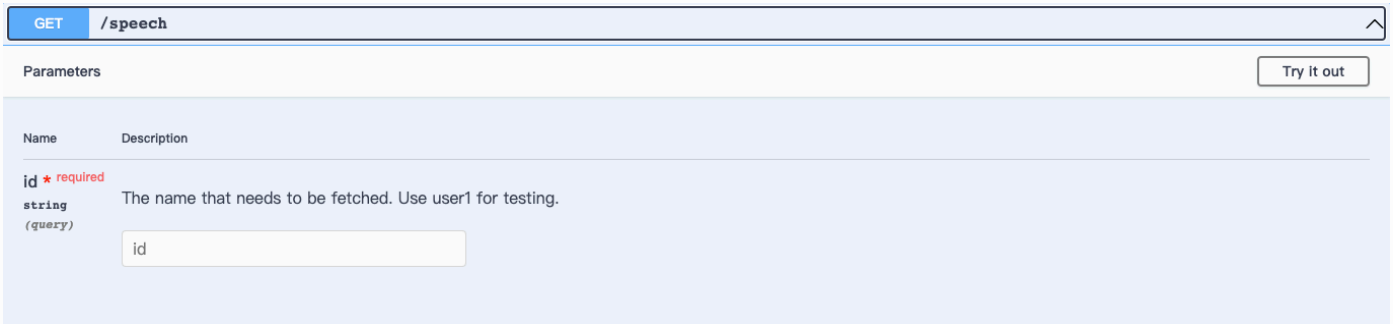
Struktur der Ergebnisse:



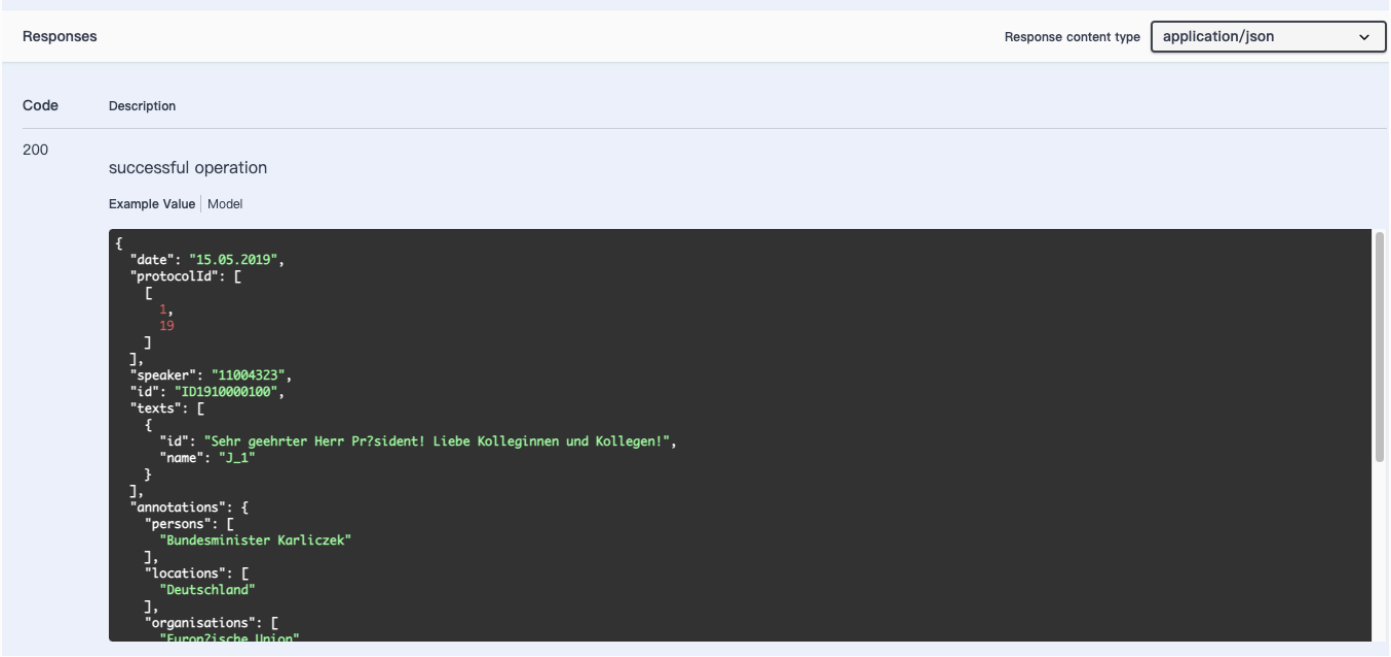
Get speech:

Sollt man hier ein Parameter *id* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:



Struktur der Ergebnisse:

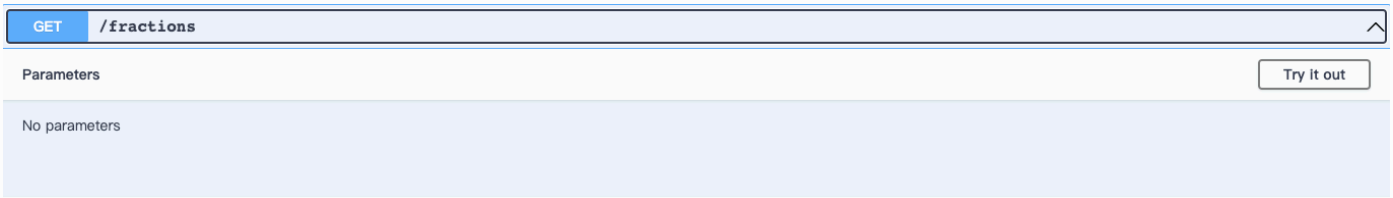


Return fractions

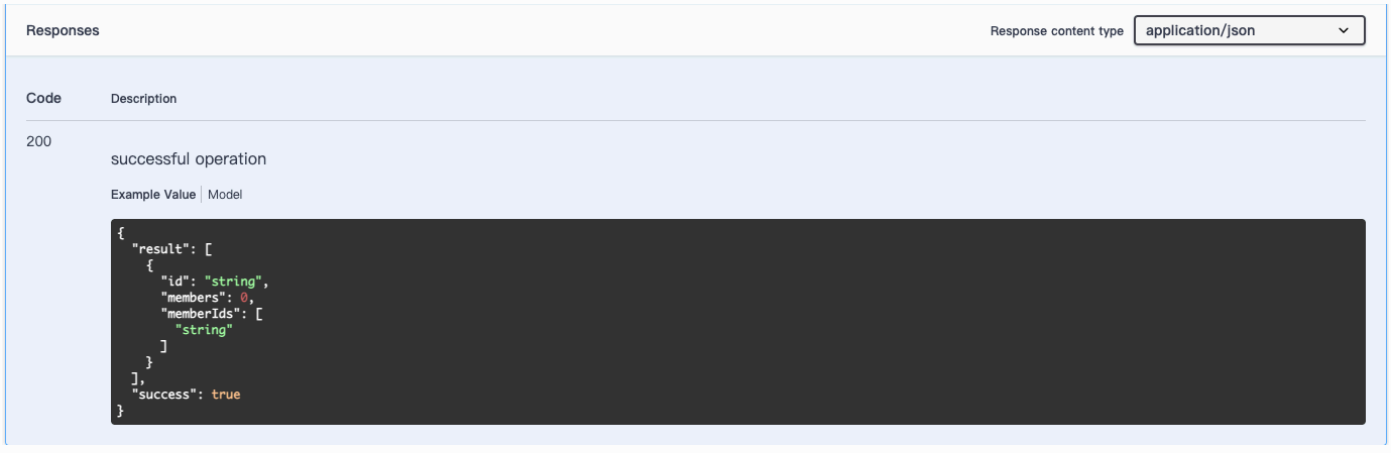
Get fractions:

Hier müssen keine Parameter eingegeben werden.

Klicken Sie auf die Schaltfläche *Try it out*:



Struktur der Ergebnisse:

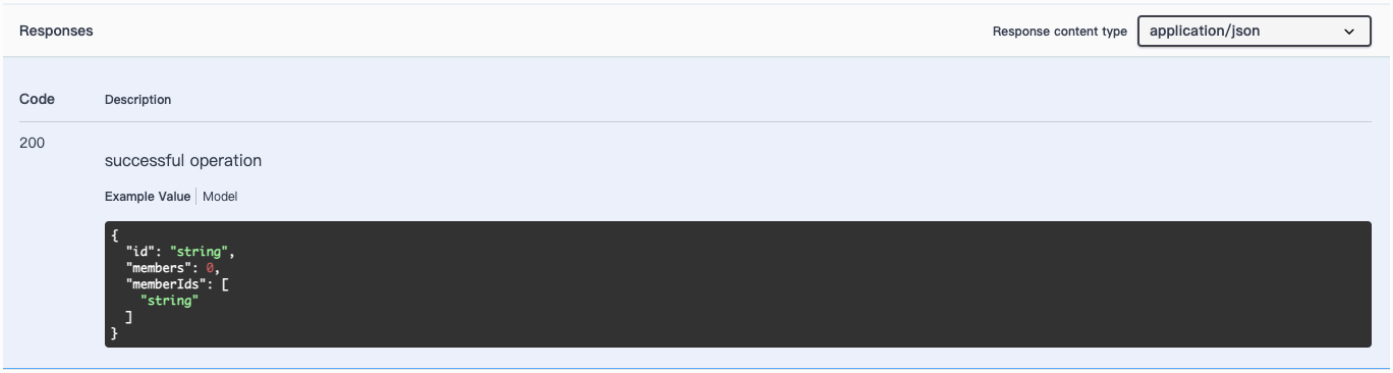


Get fraction:

Sollt man hier ein Parameter *name* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:

Struktur der Ergebnisse:

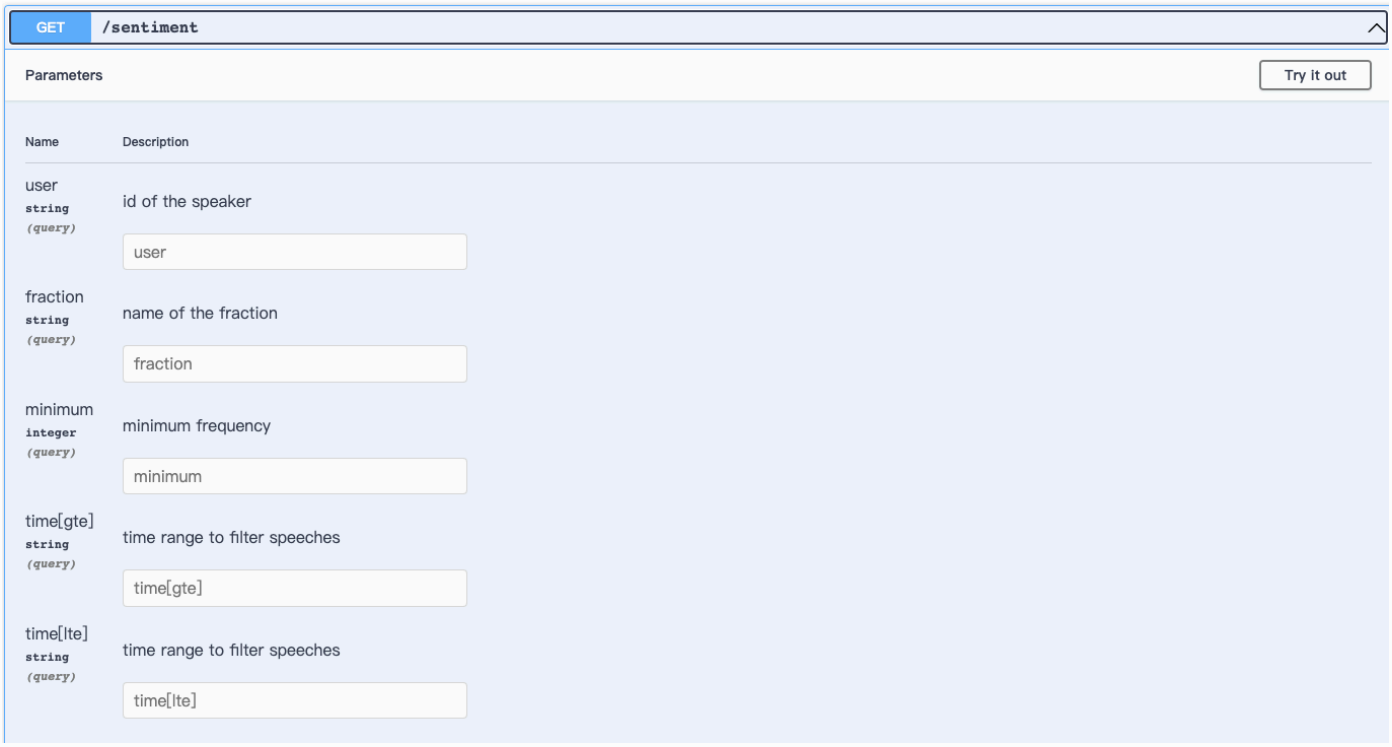


Return annotations

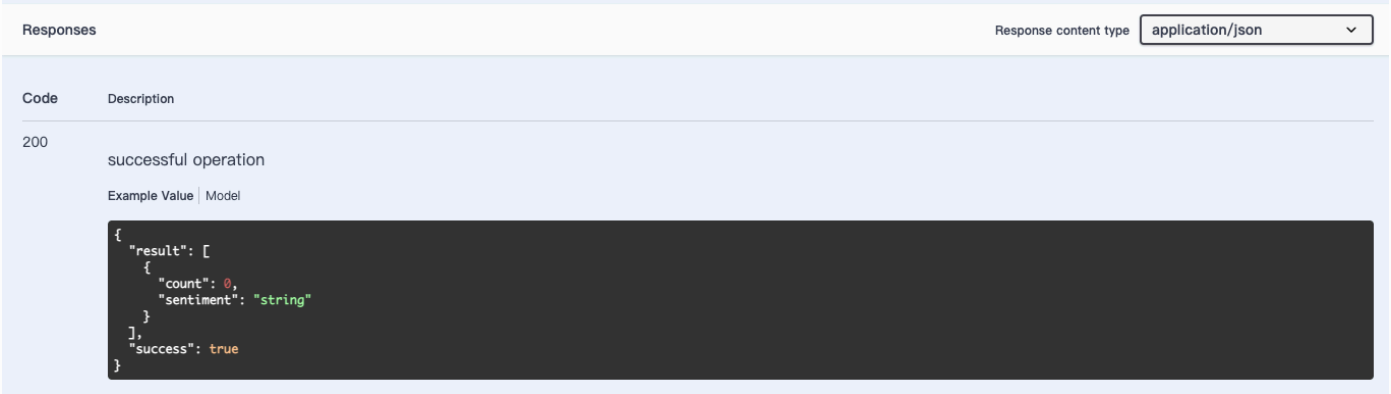
Get sentiment:

Sollt man hier fünf Parameters *user*, *fraction*, *minimum*, *time* and *time* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:



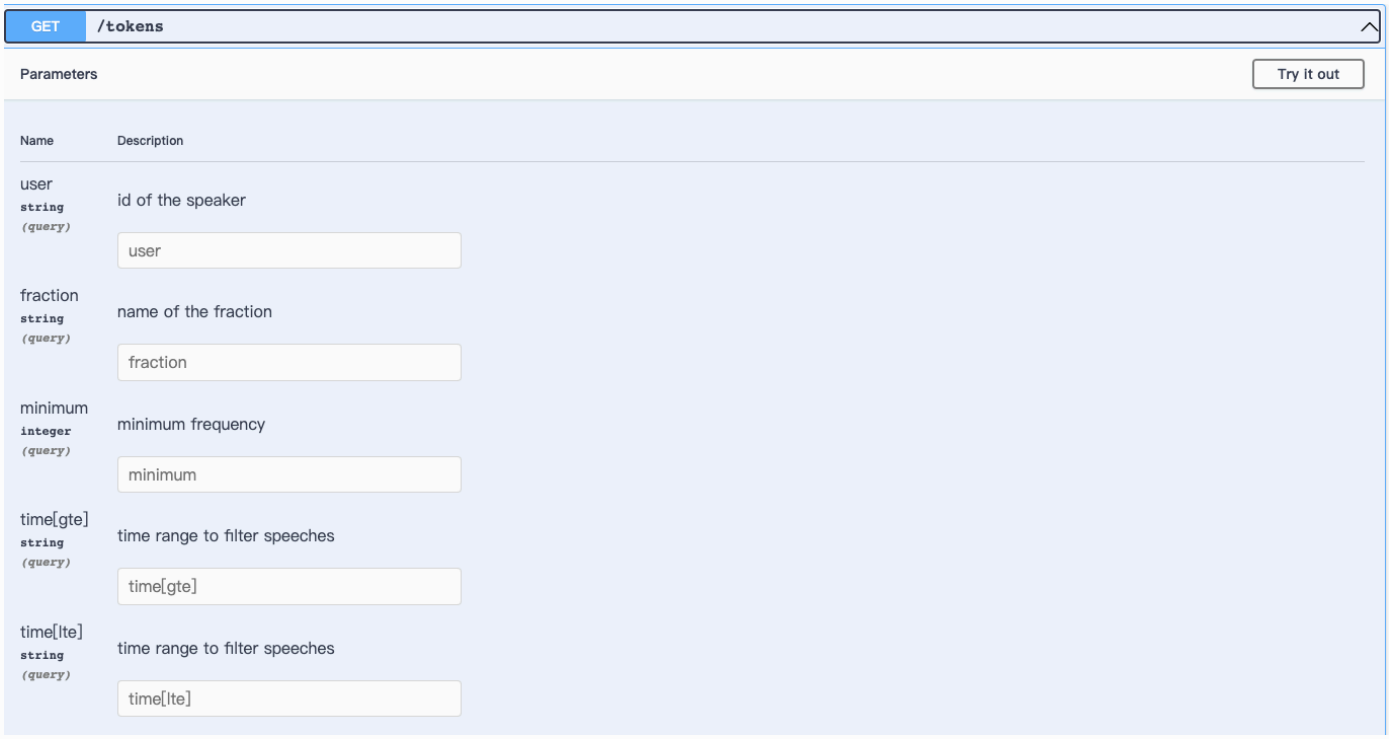
Struktur der Ergebnisse:



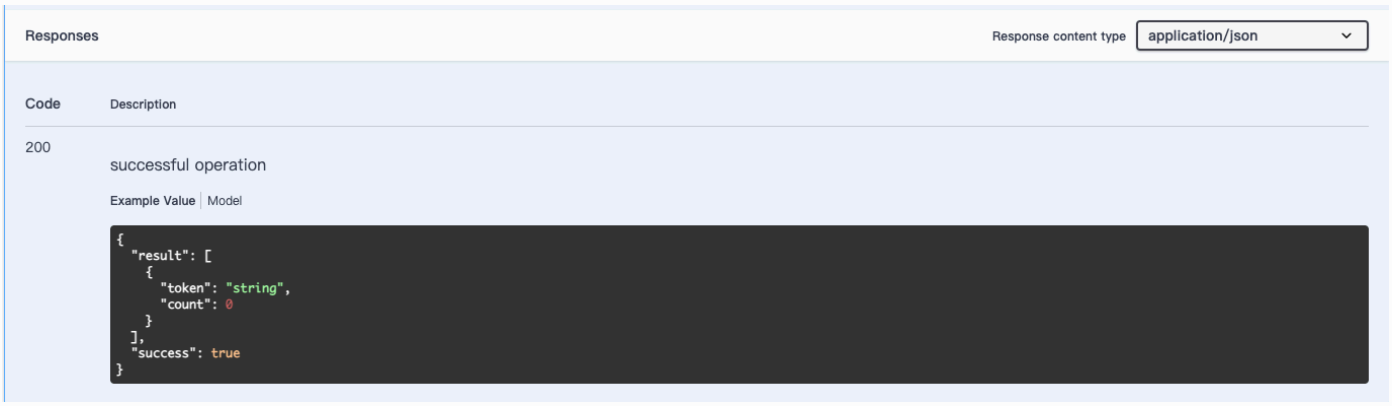
Get tokens:

Sollt man hier fünf Parameters *user*, *fraction*, *minimum*, *time* and *time* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:



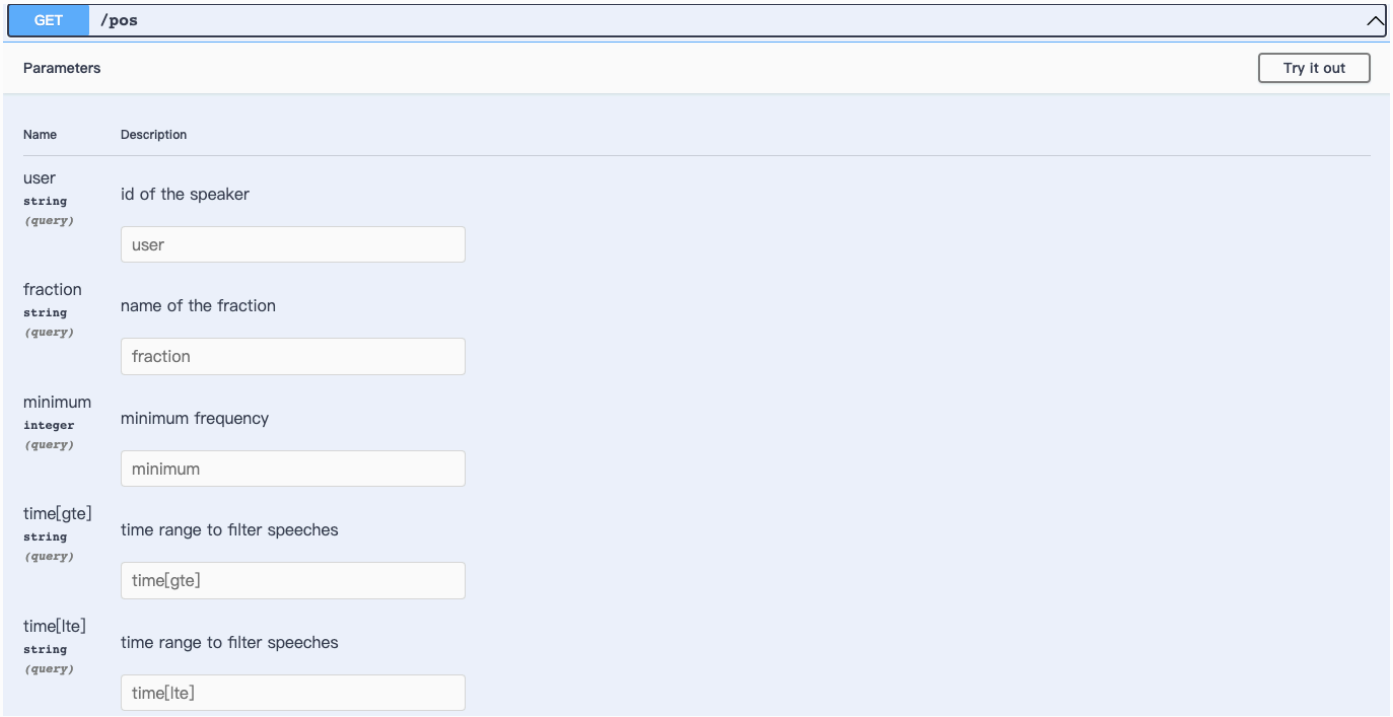
Struktur der Ergebnisse:



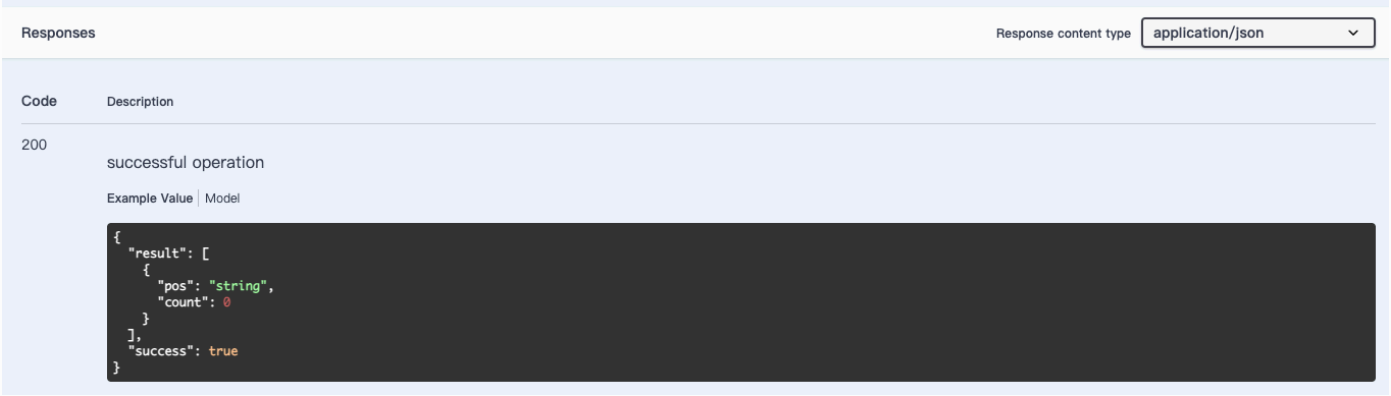
Get pos:

Sollt man hier fünf Parameters *user*, *fraction*, *minimum*, *time* and *time* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:



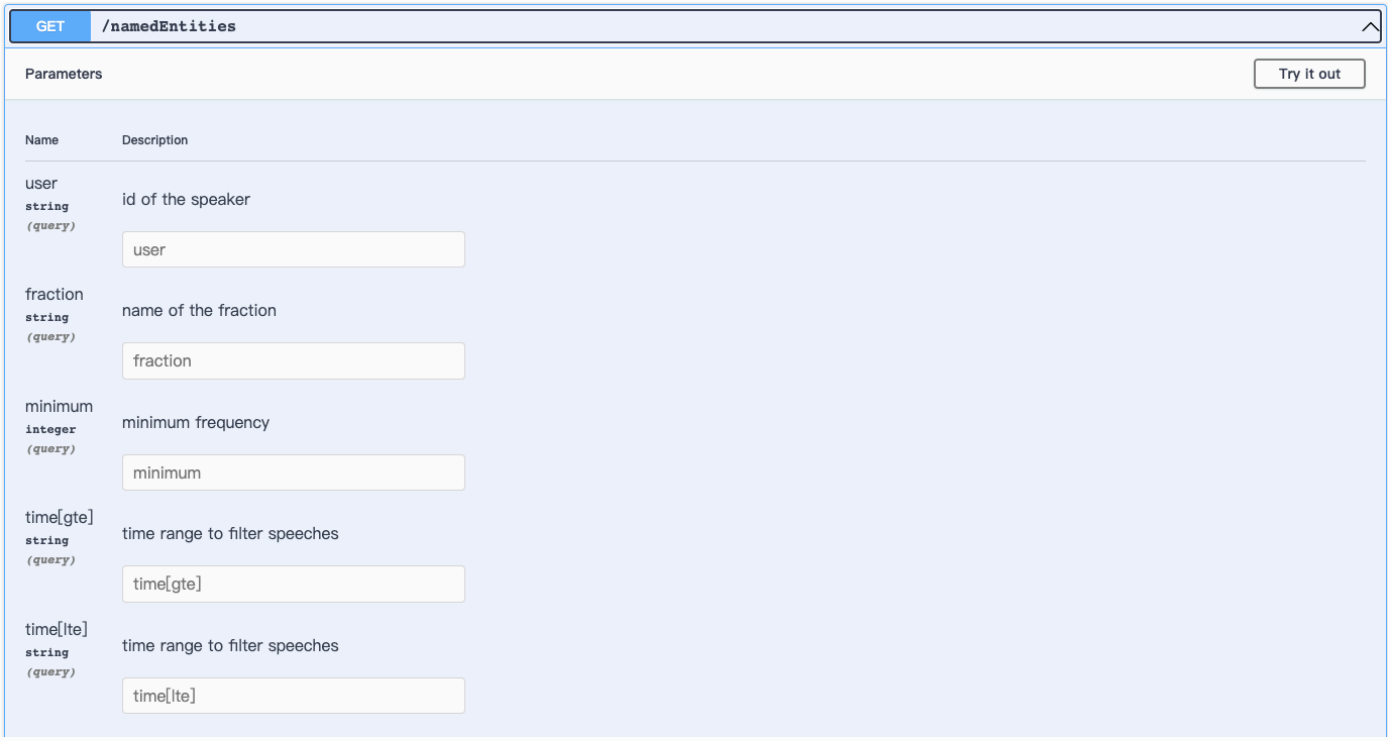
Struktur der Ergebnisse:



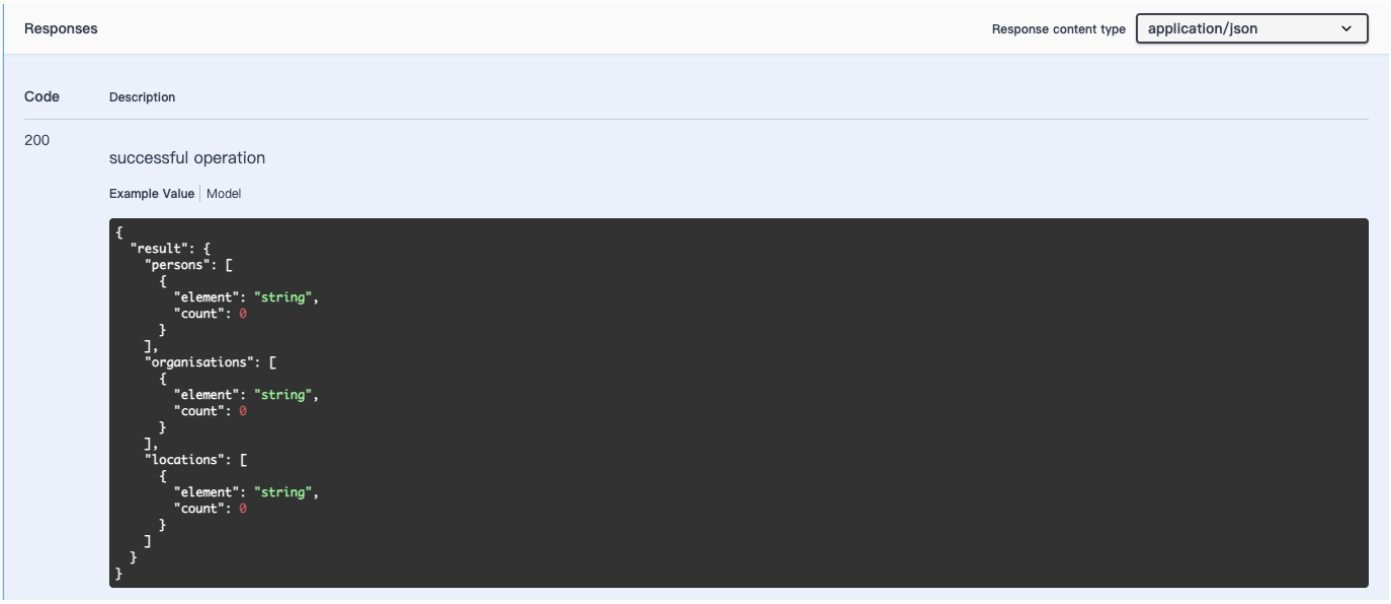
Get namedEntities:

Sollt man hier fünf Parameters *user*, *fraction*, *minimum*, *time* and *time* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:



Struktur der Ergebnisse:

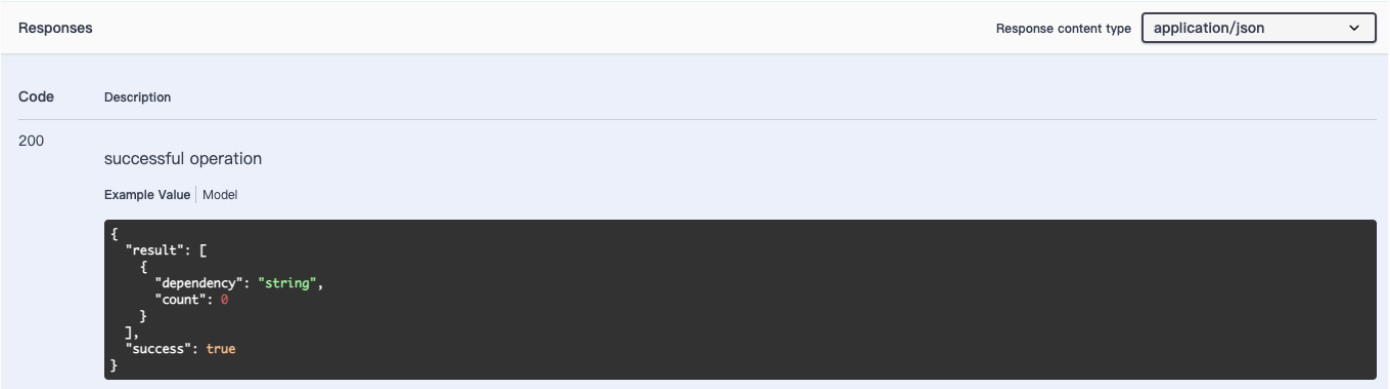


Get dependencies:

Sollt man hier fünf Parameters *user*, *fraction*, *minimum*, *time* and *time* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:

Struktur der Ergebnisse:

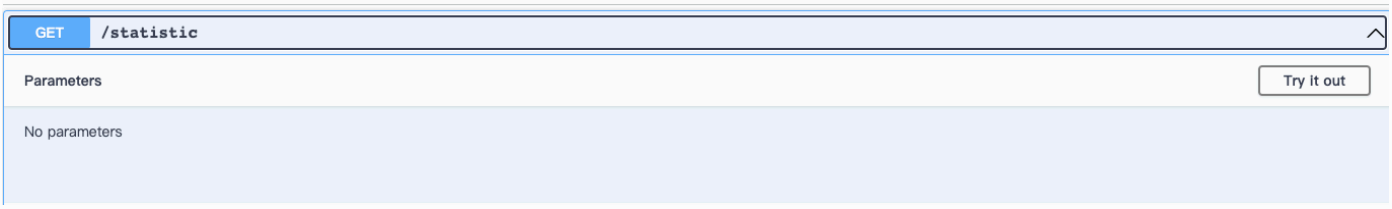


Return statistics for speakers and speeches

Get statistic:

Hier müssen keine Parameter eingegeben werden.

Klicken Sie auf die Schaltfläche *Try it out*:



Struktur der Ergebnisse: