# NS - Architecture Documentation

**Date: 24 abr 2023**

**Team:**

Stephan Guingor Falcón

Miguel Arriaga Velasco

Diego Araque Fernandez

Salvador Salgado Normandía

Pablo Rocha Ojeda

Marco Antonio Torres Sepúlveda

## Objective

The objective of this natural search team was to make a simple search experience that doesn't rely on filters.

## User Stories

US40 - NL Search: As a user I want to search for cars without the need of filters or advanced details to be able to have an easy experience in searching my car.

US42 - Filter Search: As a user I want to utilize traditional filters To be able to select a car with the specific requirements that I want (model, price range, color, number of seats, etc).

## Concepts

Text-Embedding: Text-embedding refers to the process of representing text data as a dense vector of numerical values. Text-embedding algorithms use techniques such as neural networks to learn the relationships between words in a text corpus, and represent each word as a high-dimensional vector. These vectors capture the meaning and context of words, and can be used in various natural language processing tasks such as text classification, text clustering, and information retrieval.

Custom NER: Custom NER (Named Entity Recognition) refers to the process of training a machine learning model to identify and extract specific named entities from text data. Named entities can include things like people's names, locations, organizations, dates, and more. By creating a custom NER model, it is possible to extract specific named entities that are relevant to a particular domain or task, improving the accuracy and efficiency of downstream natural language processing tasks.

Elastic Search: Elastic Search is a search engine based on the Lucene library, used for full-text search and analytics. It is designed to be scalable and distributed, and can be used for a wide range of use cases such as enterprise search, logging, and security

analytics. Elastic Search is based on a RESTful API, making it easy to integrate with other applications and services.

IP Geolocation: IP Geolocation refers to the process of determining the geographical location of an IP address. This is typically done using a database that maps IP addresses to specific locations such as cities, regions, or countries. IP Geolocation can be used for a wide range of applications such as targeting advertising, fraud detection, and security monitoring.

Semantic Search: Semantic Search is a search technique that uses natural language processing and machine learning algorithms to improve the accuracy and relevance of search results. Unlike traditional keyword-based search, which matches exact words or phrases, semantic search takes into account the meaning and context of the query, and returns results that are semantically related to the user's intent. Semantic search can be used for a wide range of applications such as e-commerce, customer support, and content recommendation.

## Overview

Docs for API

To create the best search experience we implemented a search endpoint that has two modes.

- Neural Search
    - The results for the users will be based on context, this is also known as semantic search, and we implemented it using text-embeddings in elastic search. The embeddings where generated by a model based on the pretrained model bert
- Keyword Search
    - This makes use of classic full text search, using elastic search. We use the user's query and full text search in different fields and rank the results based on the importance of the field and number of hits.

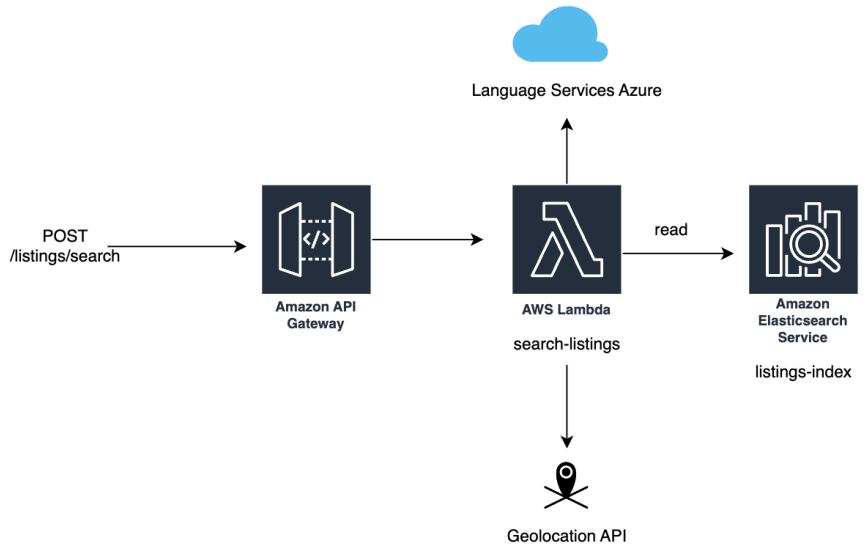
For both modes we can add filters on top, like ranges, or exact matches on fields, this is very important as users often want to limit the search results, we can also enable ‘near me‘ searches, which looks for cars that are being sold in less than a specific distance. ( 1000KM )

We also created an autocomplete endpoint, so that the user is given suggested queries as they type. For this endpoint, we store all the previous queries in the elastic search database, as well as a score for each one of them. The more a query is used, the higher its score will be and it will have more priority in the autocomplete recommendations.

In addition if custom ner is enabled while searching, we will extract different entities from the plain text query. If the service from azure has a confidence score higher than 50% (should be higher but the model is not trained yet to give high scores) then we will add them to our filters and increase the efficiency of our results.

## Architecture diagram

Search Diagram




Search-Listings Lambda: This service is in charge of determining the queries mode, if it needs to call the geolocation api and build the query that will be sent to elastic-search.

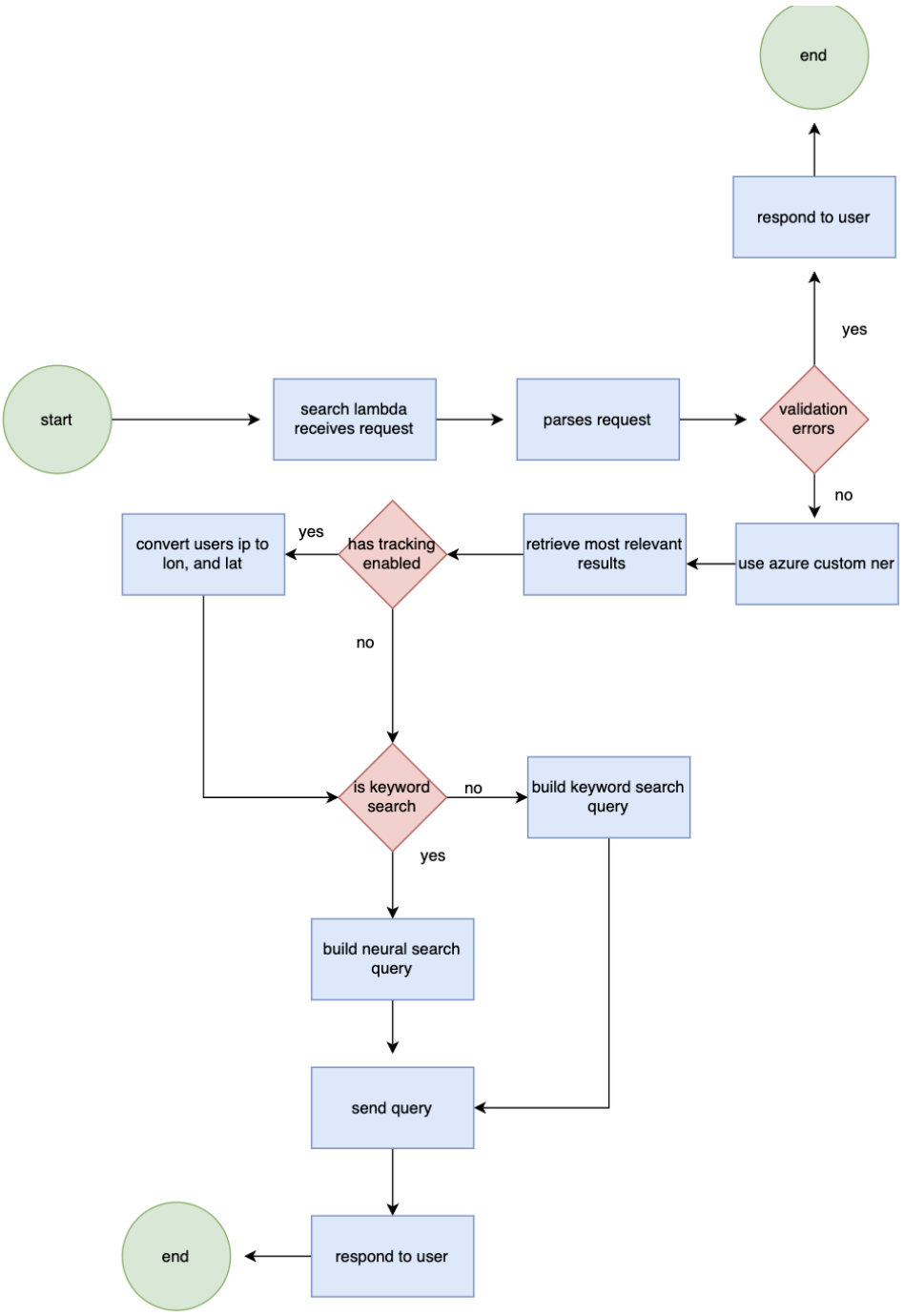
Language Services Azure: Custom NER model was trained and deployed using Azure Language Services, the service is in charge of receiving the user's query and returning a list of entities related to the car, for example the make, model, year. This in order to reduce the search space, without the user specifying the filters.

Geolocation API: If enable_tracking parameter is passed in to the search endpoint then this service will receive the users ip address and convert it to a coordinate, then we can use the coordinate to filter the nearest hits from elasticsearch.

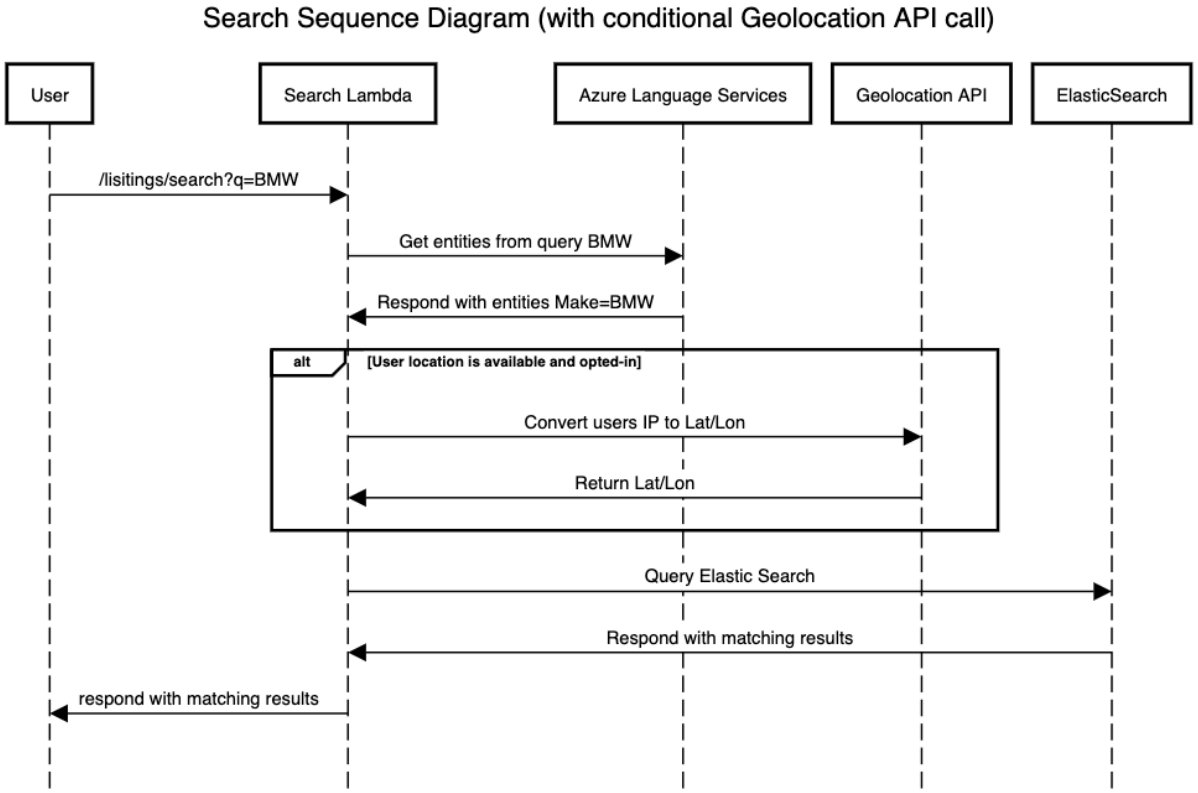
Elasticsearch: This is a NoSQL DB, and search engine we use to retrieve our information, we have setup an ingestion pipeline that when used creates a text-embedding out of the description field, and when we query data we also create an embedding for the query in order for us to look in the database for contextually similar results. It is used for all of our queries, full text search, finding results that are geographically close and contextually.

## Control Flow Diagram

Search Listings Flow Diagram


end
respond to user
yes
start
search lambda receives request
parses request
validation errors
no
convert users ip to lon, and lat
yes
has tracking enabled
retrieve most relevant results
use azure custom ner
no
is keyword search
no
build keyword search query
yes
build neural search query
send query
end
respond to user

## Sequence Diagrams




## Risks and mitigations:

| **Risk** | **Mitigation** |
|---|---|
| ElasticSearch is not in a VPC | Use APIKey authentication |
| Getting entities from azure is too slow ( 5s ) | Make it an optional feature of search |
| Synchronization issues between postgres and elastic. | Use an existing service like PGSync / research ways to trigger lambdas on db change. |
| The listings descriptions are not written in natural language and that might impact search. | When an agency uploads a listing, they should provide a description for it / use LUIS to identify properties in listings in natural languages / use a model to build these descriptions. |

## Future Improvements

- Creating a recommendation system that sends search results to the end user based on queries made in the month.
- Show trending searches based on number of first hit appearances or views.
- As the azure NER search is very slow, a future improvement would be to use a web sockets connection to bring in those results after the initial semantic search results are displayed.
- The system is created in a way in which the model used for semantic search can be easily changed. In the future we could use a more powerful model, like GPT-3.