# Gutendexer: Thought Process

## Choice of technologies

Beginning the project, I had to choose the technologies. The web framework(fastapi) was chosen, because I want to get more familiar with it and because you already use it.

Mongodb was chosen because I have already worked with it in the past. The problem with this choice was getting familiar with Motor driver, which I had never used before.

Poetry package manager was used to get familiar with it, since I have never used it and I have seen it a lot in python repos, lately.

## Process

Initially, I went through the gutendex api and got familiar with it.

I started with setting up the fastapi service and connecting to the db. Then I tried to have the minimum viable product ready.

For the search part, I noticed that gutendex(found in its repo), is searching the terms inside title, and author's name, so I try to filter the results I get to only be found in the title, since the requirement said to search a book title.

I also finished the other endpoints and then I started setting up the tests.

I had problems setting them up, because it was my first time working with asyncio and fastapi, so I merged the endpoints to develop and then started on doing some of the extra points mentioned, to save up some time.

Once I finished the extra points that I wanted to have, I got back to the test branch and I managed to set it up properly.

One point I would like to mention is that, for the pagination point, I thought of using the gutendex pagination that it offers(search results are already paginated). And I also assumed that the results from gutendex are correct. That means that I did not filter the results for checking the title, cause then in some cases the results for each page would differ from the gutendex page results.

The other extra tasks I did are pretty straightforward.

I did not do the caching part and the high-level system diagram part.

For the first one, I did not have much time to develop it and test it. My plan was to use redis for the caching. And checking if the data exists in it. If it exists then return it and if not get the result and run a background task to store the result to redis.

## Things that should have been included

Due to lack of time, I:

- Did not add more thorough tests. Like data validation from fastapi
- Did not add any logger