Tie-Die List - Yes, Yet Another To Do List App

Business Case

What?

I often use to do list apps and for several reasons I decided that I would try to make my own.

Why?

Firstly, I always loved Wunderlist, but this app is no longer being maintained and has been superseded by Microsoft ToDo. I like ToDo, but it is missing some key features which I really liked about Wunderlist, for example natural language recognition, which is not present in any other app that I have tried since. I would therefore like to attempt to make my own ToDo list which has natural language recognition. I will also attempt to include a few other features, which are laid out in the roadmap below.

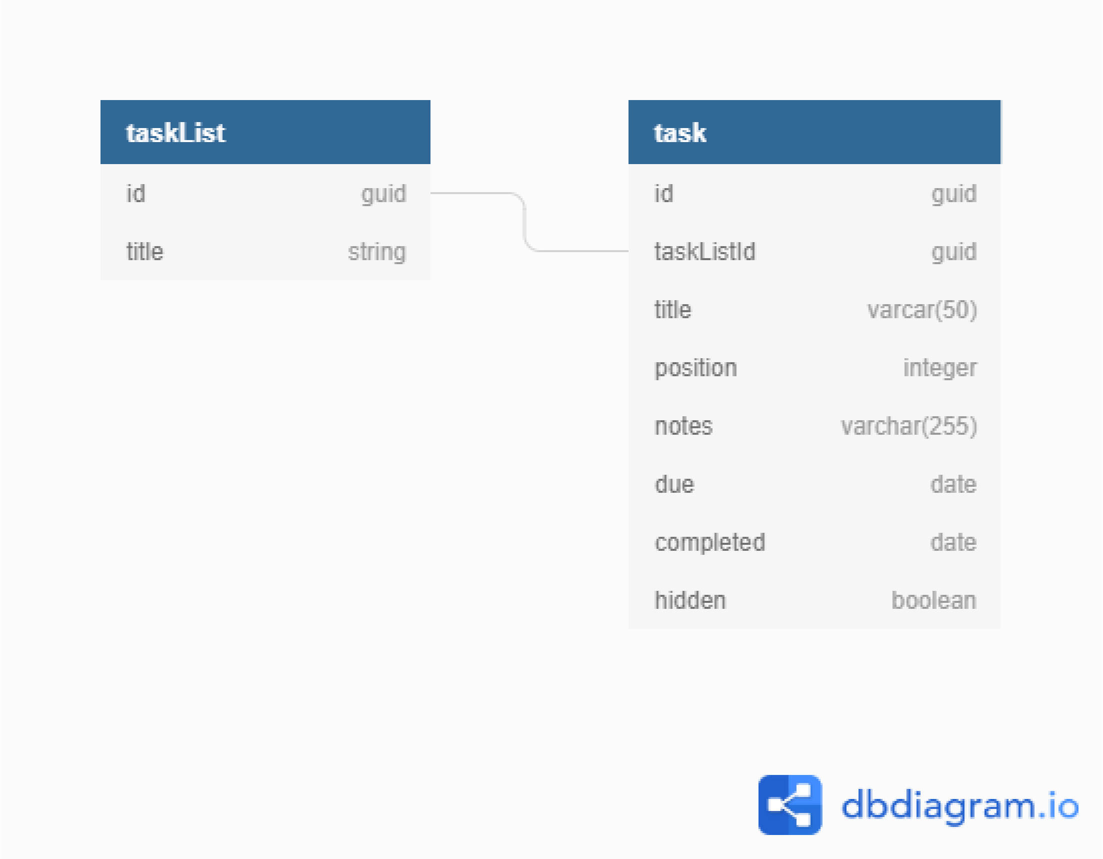
I believe that a ToDo List app is a good project for PD, because it contains all the rudiments of a CRUD application.

Technology Stack

What technologies have I used and why?

- Angular TypeScript Front End: It is easy to quickly build a responsive GUI using Angular and connect it to an API
- Java Spring Backend: Easy to quickly set up a REST API in Java Spring. This will give me more power over how to collect data from the DB and send it to the front end.
- MySQL v8 Database - free relational DB, which will allow me to create separate lists of tasks and query them quickly
- Keycloak authentication provider: easy out of the box application to manage OAuth2 authentication and token granting
- Source Control: GitHub. The code will be globally available on GitHub, although the credentials will not be stored there.
  - Front End: https://github.com/joshp93/tie-dye-list
  - Back End: https://github.com/joshp93/tie-dye-list-api
  - DB (MySQL SPs, Functions etc.): https://github.com/joshp93/tie-dye-list-mysql-sps

DB Diagram



taskList	
id	guid
title	string

task	
id	guid
taskListId	guid
title	varcar(50)
position	integer
notes	varchar(255)
due	date
completed	date
hidden	boolean

Roadmap

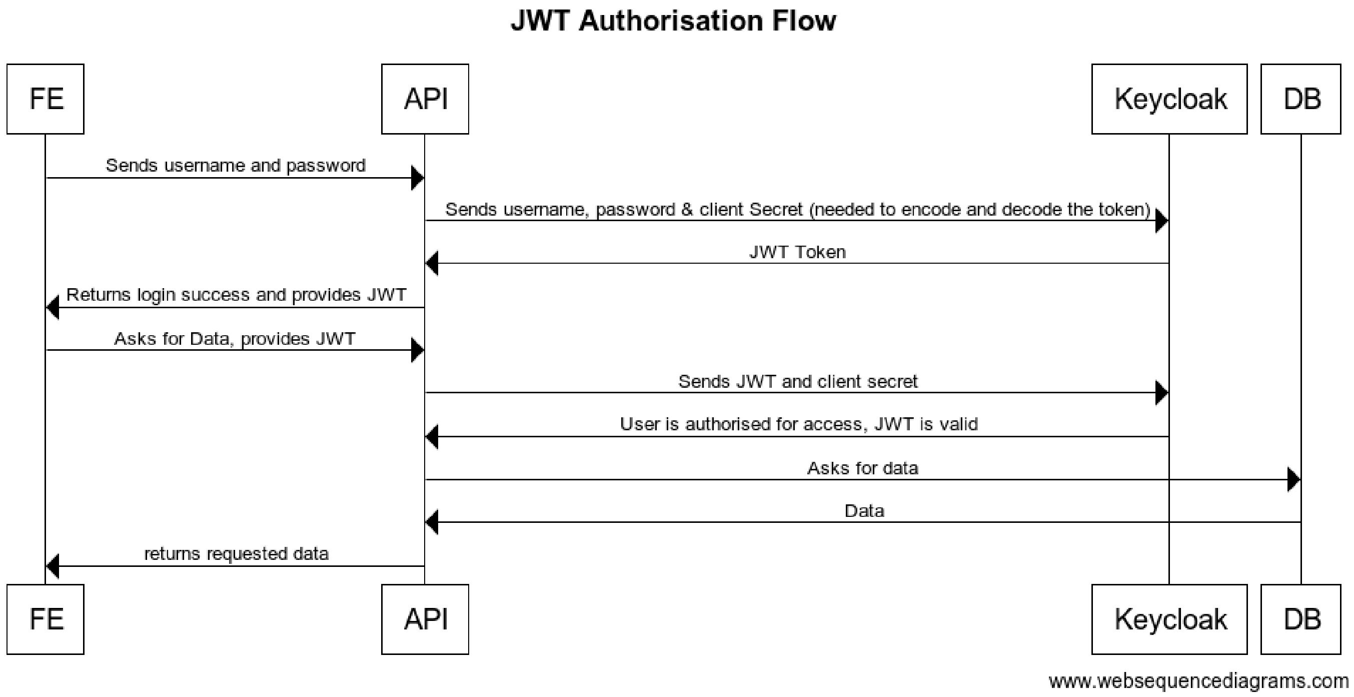
Note: I have already created the front end and the back end and the basic styling, RESTful framework have already been implemented. This roadmap therefore begins with Authentication and DB Integration.

Authentication

I decided to use Keycloak rather that write the code to create my own authentication provider.

Keycloak is an application which can be run on the server to automatically handle authentication to multiple endpoints. It can be easily integrated into Spring Boot, which is why I have chosen to use it.

Below is the authentication flow used:

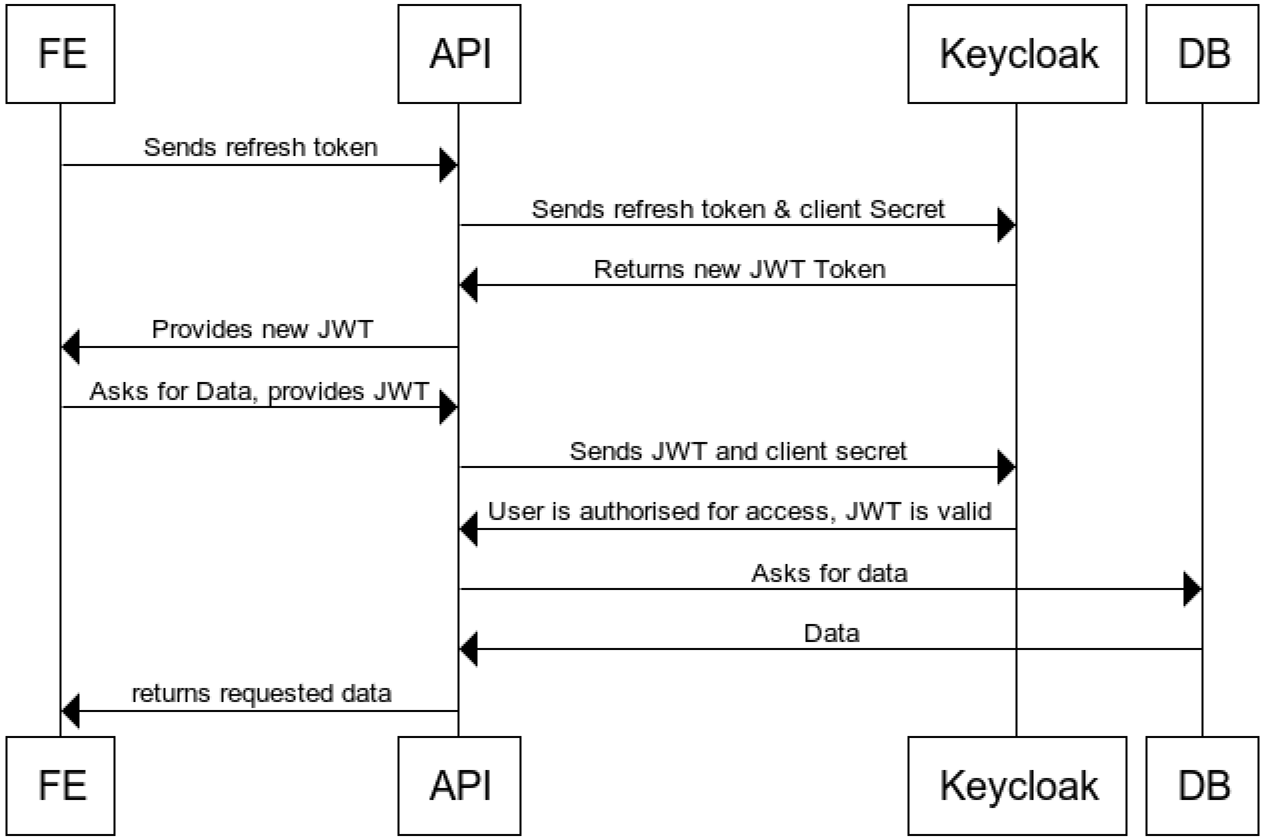

JWT Authorisation Flow
FE
API
Keycloak
DB
Sends username and password
Sends username, password & client Secret (needed to encode and decode the token)
JWT Token
Returns login success and provides JWT
Asks for Data, provides JWT
Sends JWT and client secret
User is authorised for access, JWT is valid
Asks for data
Data
returns requested data
FE
API
Keycloak
DB
www.websequencediagrams.com

The access token will expire after a short period of time. However the refresh token can now be used to retrieve a new token from Keycloak without having to re-enter username and password:

Before sending a request to the tasks API, the front end will check if the token has expired, based on the expires at information returned from Keycloak.

If the token has expired then the front end will send a request to Keycloak using the refresh token to ask for a new access token

JWT Authorisation Flow


FE
API
Keycloak
DB
Sends refresh token
Sends refresh token & client Secret
Returns new JWT Token
Provides new JWT
Asks for Data, provides JWT
Sends JWT and client secret
User is authorised for access, JWT is valid
Asks for data
Data
returns requested data
FE
API
Keycloak
DB

www.websequencediagrams.com