Brain Fog Blog

Author: Daniele Campagnoli

Index

[Introduction 2](#_Toc92386897)

[Project Repository Structure 3](#_Toc92386898)

[System Architecture 4](#_Toc92386899)

# Introduction

The Brain Fog Blog (BFB) is a blog abut technology and generative art.

The main scope of this project is to demonstrate my skills on software engineering.

# Project Repository Structure

The project is hosted at the following git repository:

<https://github.com/DanieleCampagnoli/brainfog>

The repository is divided in 2 folders:

1. docs: contains the project documentation
2. sources: contains the project source code

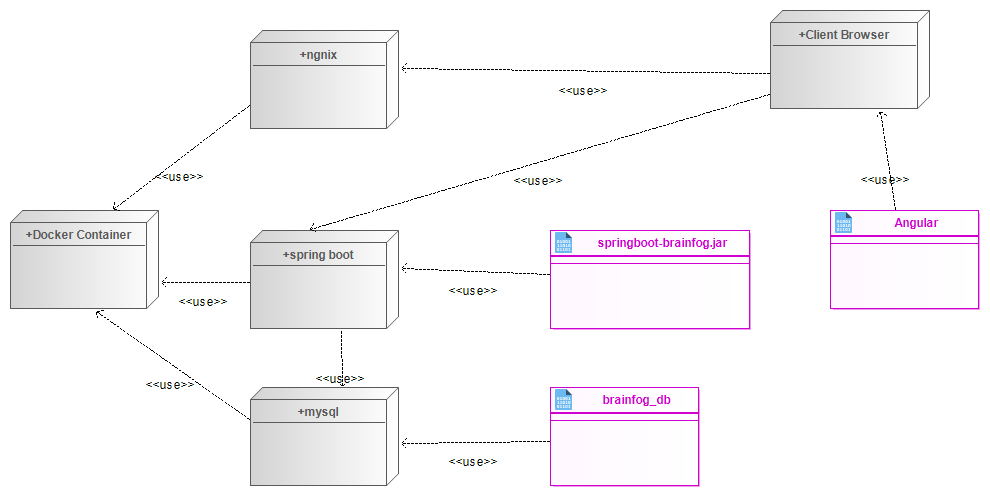
The docs folder is organized with the following files

|  |  |
| --- | --- |
| file | description |
| documentation.docx | The project documentation (this document). |
| modelio.zip | A modelio project with UMLs.  The tool can be downloaded from this website  <https://www.modelio.org/> |
| brain\_fog.xd | This file contains a mockup of the website made with Adobe XD. |

The source folder is organized in the following way:

|  |  |
| --- | --- |
| file | description |
| angular-brainfog | Angular application source code (UI) |
| springboot-brainfog | Spring boot application that contains the business logic of the application. This application exposes a rest API that is consumed by the Angular application. |
| AngularBrainfog.Dockerfile | Docker configuration for the angular-brainfog application. |
| docker-compose.yml | This docker configuration file is used to configure and startup all the docker containers in order to start the overall system. |
| MysqlDB.Dockerfile | Docker configuration of MySql ( Persistence). |
| SpringBootBrainfog.Dockerfile | Docker container for springboot-brainfog. |

# System Architecture



# How CCS is structured

The css follows the BEM notation <http://getbem.com/naming/>.

The file src\app\app.component.css contains the global css rules of all the components.

On top of the file we have css custom properties

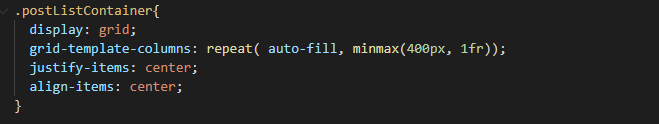
<https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_custom_properties>

They are variables that can be called from other css classes. This is useful to define colors and other utility values like fonts and so on.

## Grid system

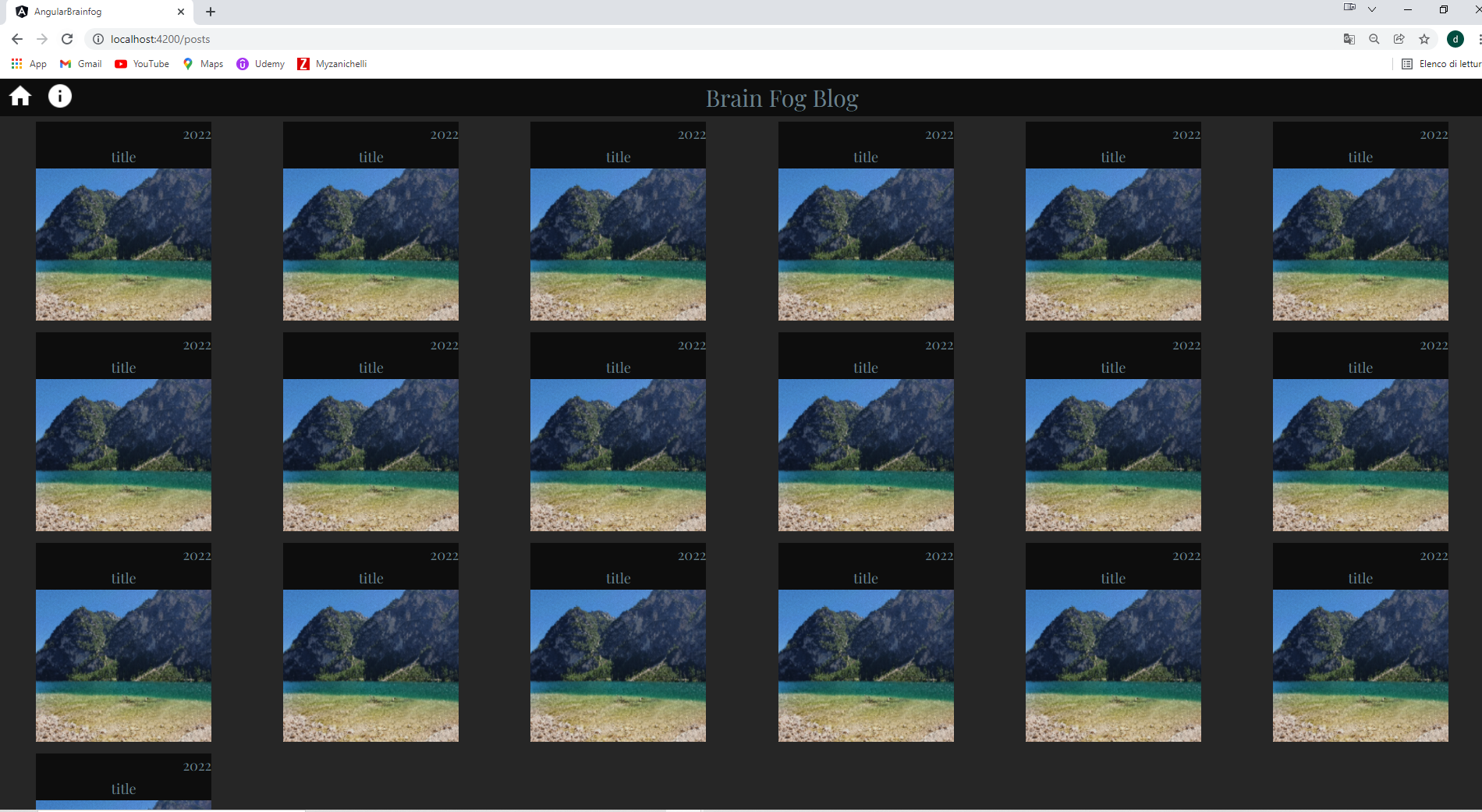
The application is using the standard CCS grid system.

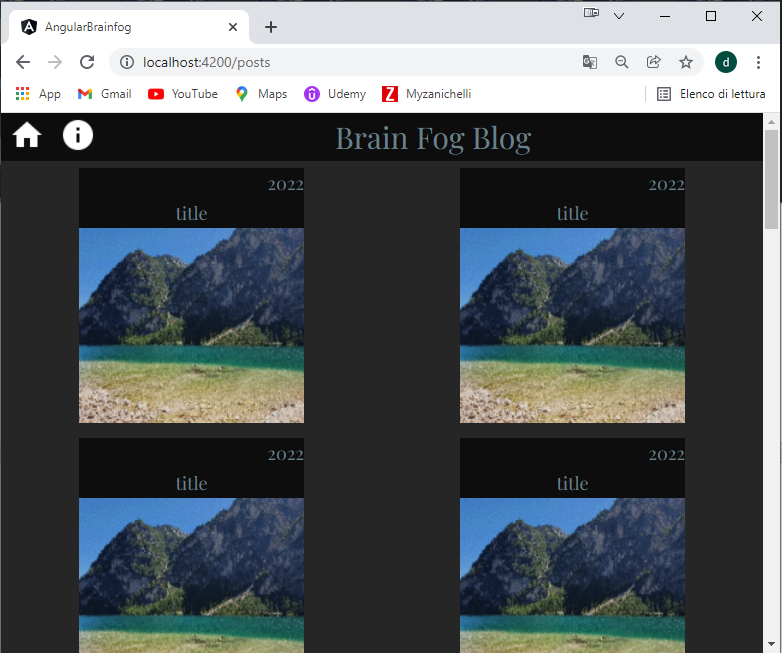
The grid system is also able to arrange the items on the page in a responsive way.



This snippet of code defines a grid with columns that are minimum 400 px wide or they have all the same dimension on the page (1fr).

This allows the list to shrink if needed.





As you can notice from the two screenshots the grid has adjusted the number of columns from 6 to 2 based on the window dimension.

# CAOS TO BE ORGANIZED – how to structure the application

**Main idea**

Angular will be deployed on a dedicated web server on a dedicated docker container.

Spring will be deployed on a dedicated web server with the standard spring boot setup and it will use one docker container.

MySQL will be deployed also on a dedicated docker container.

**Angular And Docker**

Angular And Docker Deployment taken from the following tutorial

https://www.indellient.com/blog/how-to-dockerize-an-angular-application-with-nginx/

Command to build the angular docker image

docker build -t angular-brainfog -f AngularBrainfog.Dockerfile .

Command to start the angular docker image

docker run -d -p 8080:80 angular-brainfog

**Spring Boot And Docker**

<https://spring.io/guides/topicals/spring-boot-docker>

docker build -t springboot-brainfog -f SpringBootBrainfog.Dockerfile .

docker run -d -p 8080:8080 springboot-brainfog

**MySQL and Docker**

The configuration is specified by the docker-compose.yml. For now I’m not using any special configuration.

**Docker Compose**

Start all the applications required by the system: db, application server and web server with the angular application.

Run the following commands from the project folder.

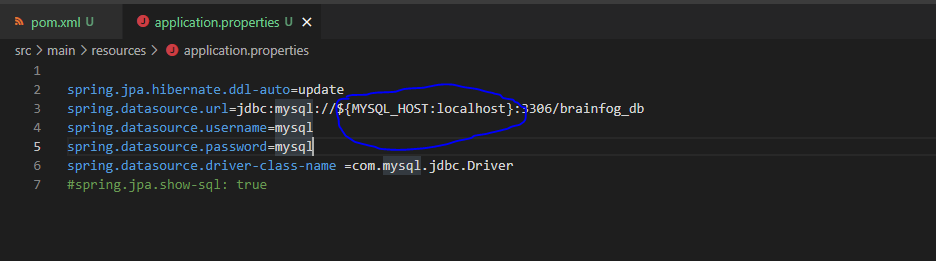
docker-compose build

docker-compose up

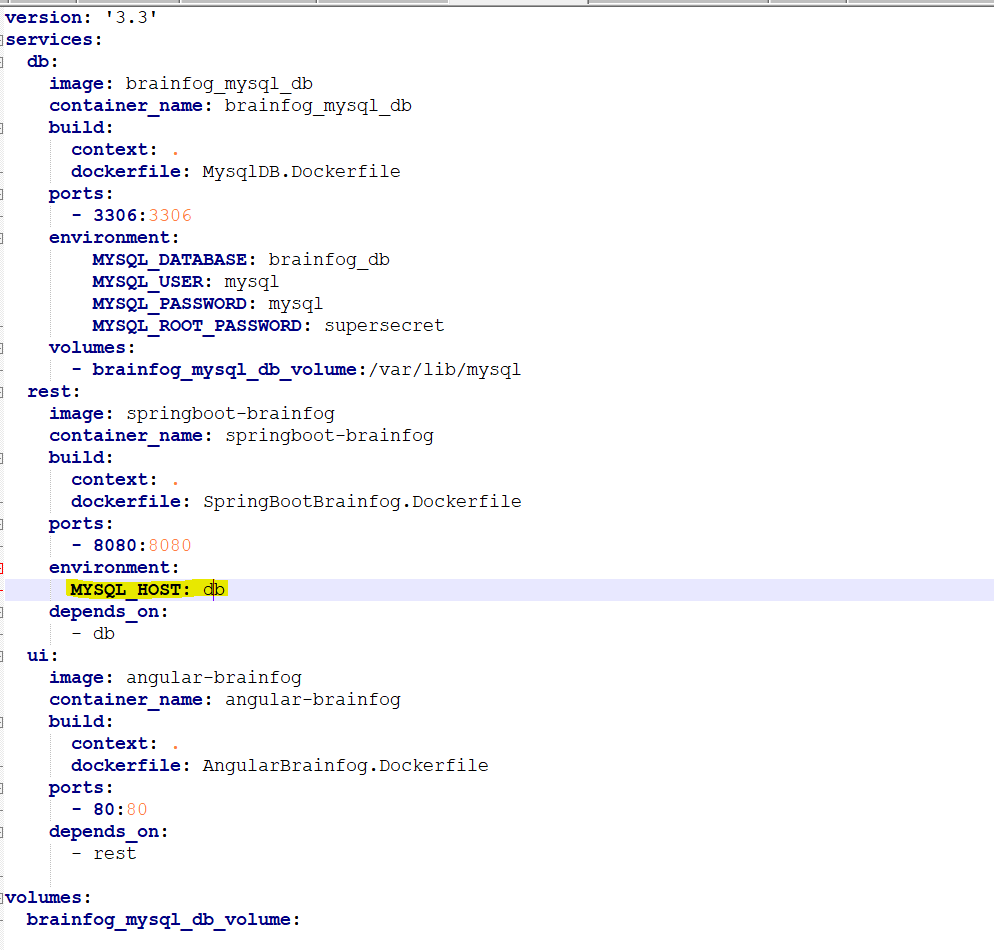
The docker compose creates a virtual network.

<https://docs.docker.com/compose/networking/>

In order to make it work there is a special configuration to implement on Spring.



Docker compose configuration

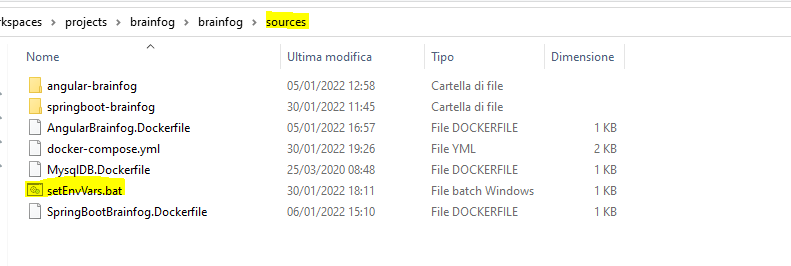


COME STRUTTURARE IL DOCUMENTO??? Forse in awd all’università avevo già fatto qualcosa.

Questo documento per la struttura del progetto, poi un documento per ogni post.

**Password Management**

The application is published as a public project on github. For this reason the passwords as specified inside system variables and are populated with a script.



1. run the script to populate the system configurations
2. run docker and the various applications

**Version 1.0.0**

Every blog post is mainly implemented in angular. The blog text is hardcoded in the html page.

The spring backend and the persistence is called only to feed the UI with the algorithm output.

**First post: Voronoi space**

Download an image of the space and apply the voroni diagram on it.

[**https://en.wikipedia.org/wiki/Voronoi\_diagram**](https://en.wikipedia.org/wiki/Voronoi_diagram)

[**https://api.nasa.gov/**](https://api.nasa.gov/)

**How to switch Angular from development to production mode**

[**https://angular.io/guide/build**](https://angular.io/guide/build)