MÓDULO: BASE DE DATOS

INTEGRANTES:

- Medrano Cambara Natalia
- Rodriguez Mendoza Denis

DESCRIPCION

Contamos con dos APIs principales una para USUARIOS y otra para KUDOS

API USUARIOS

Implementado sobre NodeJS, utiliza una Base de datos MONGO y tiene los siguientes endpoints:

- **CREAR USUARIOS**

  utilizando el endpoint expuesto, crea un usuario en la base de datos MONGODB y tambien notifica a la cola de eventos "queueCreateUser" que es un RPC para notificar a un escucha en el lado del API de kudos que tambein debe crear un usuario (nickname/nombre) en la Base de datos Neo4J, esto para preparar la data para cuando se necesita insertar un Kudos.

- **LISTADO DE USUARIOS - /*user/*list/**

  Lista los usuarios de la BBDD Mongo, con la particularidad de que este tienen un escucha que recoje los mensajes disponible en la cola de Rabbit "queueStats", la cual indica si sumamos o restamos la QTY de Kudos que tenga cada usuario a ser listado, esto se hace asi porque la BBDD de Kudos esta en Neo4J, con esto queremos reflejar la consistencia eventual. Los resultados pueden ser páginados

- **DETALLE USUARIO – *user/:*_id**

  Detalle de usuario, y obtencion del detalle de kudos del usuario en cuestion. Se usa una cola Rabbit para poder pedir los datos de los ckudos del usuario

- **EIMINAR USUARIO - /user/delete/:_id**

  Elimina un usuario en la Base de datos de MONGO y adicionalmente envia a una cola "queueDeleteUser" el evento de que se esta eliminando un usuario, luego esta cola notifica a la API de Kudos para eliminar a este usuario y sus relaciones en la BBDD de Neo4J (KUDOS)

- **BUSCAR USUARIO - /search**
  Busca en la BBDD de Mongo al usuario mediante su username o nombre; NO implementa SOLR o ELASTICSEARCH

API KUDOS

- **CREAR KUDOS - kudos/add**

  Crea un kudos men la BBDD Neo4J, ademas de relacionar al usuario que ENVIA y al usuario que RECIBE el Kudos, tambien este proceso notifica a a cola "queueStats" que se ha agregado un nuevo kudos al usuario que RECIBE, con lo cual el API de usuarios debera cambiar el campo QTY de la los usuarios

para sumar o restar según corresponda. Se relaciona el Kudos con el que ENVIA por medio del username y el idKudos y lo mismo para la relacion que RECIBE

- **DETALLE DE KUDOS – kudos/:_kudosId**

  Muestra un detalle de los kudos, ademas de mostrar el nombre y username de los usuario que participan de cada Kudos

- **ELIMINAR KUDOS – kudos/delete/:_id**

  Elimina un kudos y las relaciones de este en la BBDD neo4J, ademas notifica a una cola Rabbit llamada "queueStats" para disminuir el QTY de usuarios

- **LISTA DE KUDOS – kudos/list**

  Lista los kudos y el detalle de los usuarios que participan en cada uno

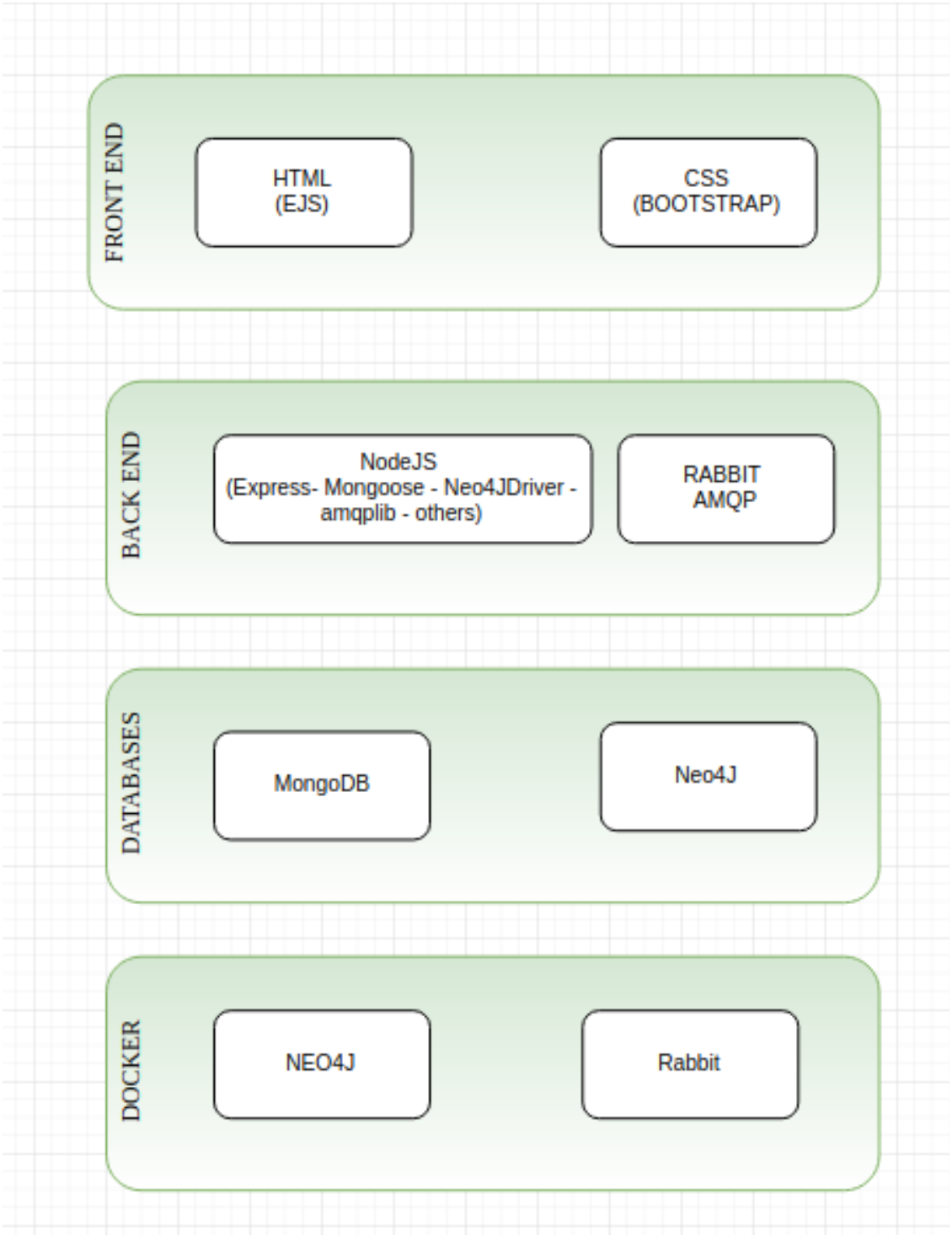
SOFTWARE STACK

Se utilizaron los siguientes elementos:

1. **FRONTEND:**
   - Motor de plantillas HTML EJS
   - Framework CSS Bootstrap

2. **BACKEND**
   - **Nodejs** utilizando librerías NPM, entre las principales destacan:
     - **EXPRESS** : framework para trabajo con aplicaciones web sobre nodeJs
     - **MONGOOSE**: driver para trabajo con base de datos MongoDB desde NodeJS
     - **NEO4J-Driver**: Driver para conexión con una base de datos Neo4J desde NodeJS
     - **AMQPLIB**: Librería para crear conexiones AMQP desde NodeJS
   - **RABBIT** Como negociador de mensajes, esto para realizar las notificaciones de eventos utilizando colas, para poder sincronizar la data entre BBDD
3. **BASES DE DATOS**
   - **MONGODB** para implementar la Base de datos de los usuarios
   - **NEO4J** para implementar la persistencia de los KUDOS

MÓDULO: BASE DE DATOS



FUNCIONAMIENTO

GITHUB

https://github.com/patoCode/bbdd_final.git

DOCKER RABBIT

Ejecutar por consola :

$ docker run -d -p 15672:15672 -p 5672:5672 --hostname my-rabbit --name some-rabbit rabbitmq:3-management

DOCKER NEO4J

Ejecutar por consola :

$ docker run --publish=7474:7474 –publish=7687:7687 --volume=$HOME/neo4j/data:/data neo4j