Final Project:

Indexing and retrieving Metadata from an Instagram Dataset

Jordi Morera Trujillo
Eric Presas Valga
Adrià Carbó Duch

Introduction

With the raise of the new kind of web applications that most of the companies are starting to use, we are going to implement a simple web application to retrieve some images and it's metadata from a certain Instagram image set.

As it's said in the proposal document, the purpose of this project, is to build a web application that retrieves some data from an Instagram images dataset given by the teacher. Specifically consists on, first indexing the metadata from our images dataset to a Non relational database (MongoDB) to later retrieve this data using some kind of filters.

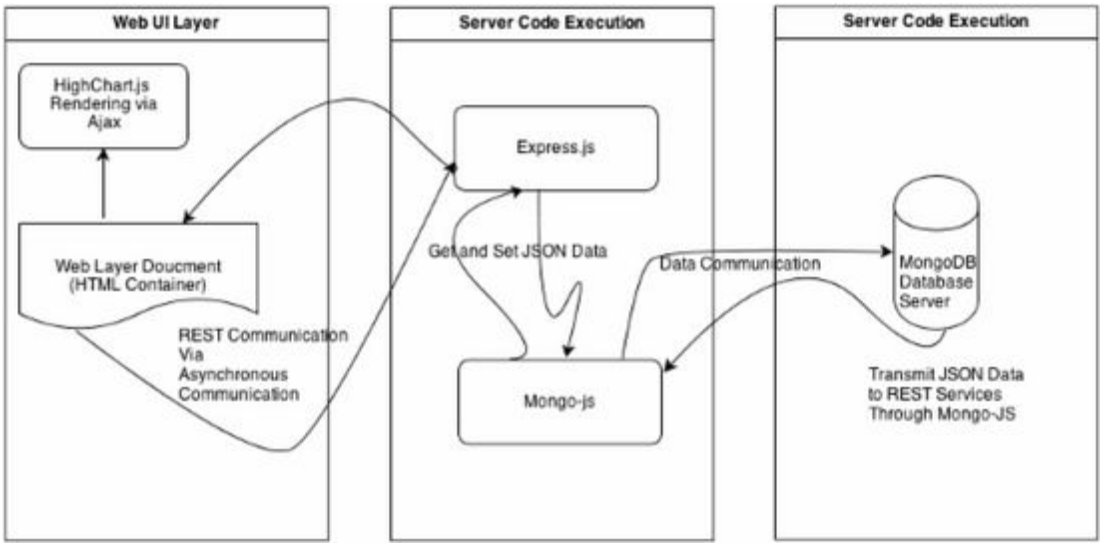
Since our metadata set is provided in JSON format, we are taking use of databases that store information in collections (Nosql).

We also propose to make a web application as a support for giving some "real" application to our work.

System architecture

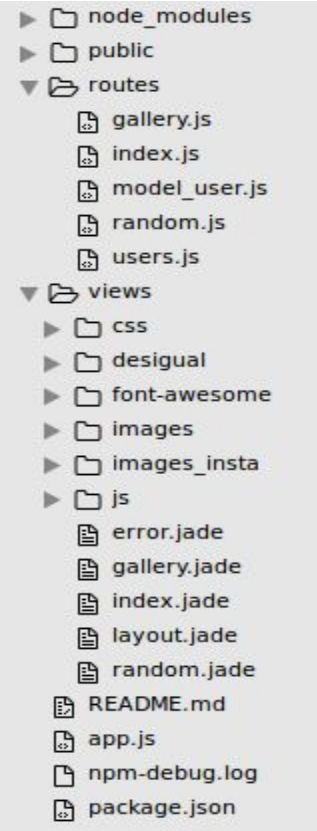
In this section the architecture of our web application will be explained::
Server-side is structured in two main parts,

- NodeJS back-end: Handles client requests, retrieve information from the database, and sends the response to the client with the desired information
- Database: Stores all the information of the app.
- Client: Interacts with the user and sends requests when the client performs an action.



Project structure

On this section, project structure will be explained in order to understand how a Node project is organized and structured.



As we can see, on the left image, the folders routes and views are those that contain the scripts that describe and define our project.

Main folder: Contains the script app.js that links to every script on routes folder.

Routes folder: Contains the back-end functions and functionalities of every part of the web application.

Views folder: Contains the front-end visualization of the web application (images, styles and layout).

Node_modules: Contains the libraries used on the project.

Data integration: PENTAHO ETL

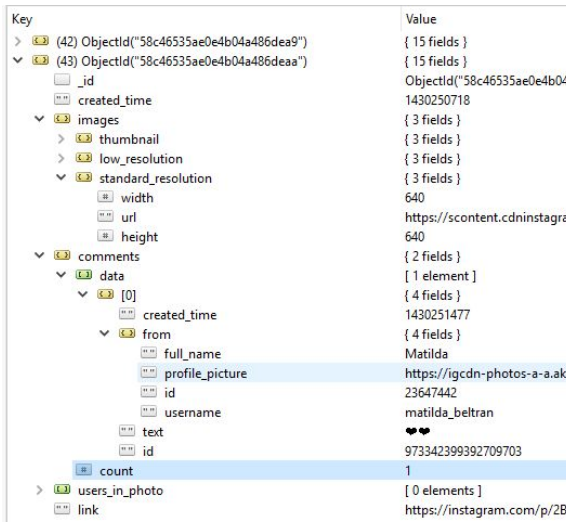
Pentaho data integration (PDI) is the component of Pentaho responsible for the Extract, Transform and Load (ETL) processes. Though ETL tools are most frequently used in data warehouses environments, PDI can also be used for other purposes:

- Migrating data between applications or databases
- Exporting data from databases to flat files
- Loading data massively into databases
- Data cleansing
- Integrating applications

We used it in order to fill our database with Instagram's metadata.

Metadata structure: Images

MongoDB is a NoSQL document-oriented database program that uses JSON-like documents. In our case we have the metadata of images taken from instagram which structure looks as follows:



Front-end

For the view in the browser, that what's called the front end of our project or the client side, HTML is applied in a different way that we are used to.

By the necessity of having a dynamic HTML content to show our results, i.e: in tables or container boxes, on the client-side, a HTML template engine called JADE will be used in order to solve this problem.

Jade (or pug)

Jade is an elegant and a high performance templating language and engine focused on enabling quick HTML coding. Influenced by Haml and implemented with JavaScript for nodeJS and browsers.

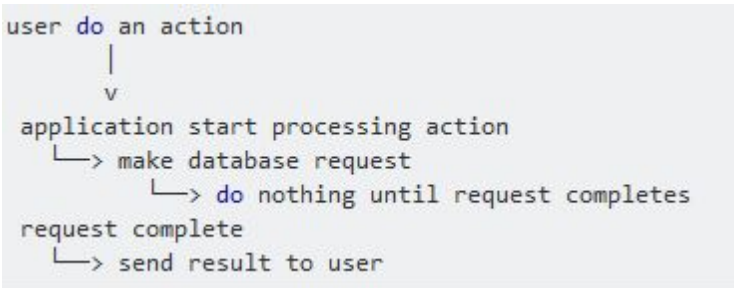
No more XML situps and percent prefixes for tags. It is HTML templating how it should be.

Back-end

NodeJS

Node JS is an event-oriented single-thread requests processing. It handles concurrency by callbacks.

The schema that follows shows how it works:



We implemented the following functionalities:

- Retrieve random photo with its comments and likes.
- Search engine by tags,username and fullname.

Routes

App.js handles the routes to the back-end functionalities depending on client's actions. Depending of the functionality we have basically 4 options:

Index

It's the principal page of the web application where the specified route is the root path.

This main page have some functionalities:

- It shows randoms photos on a slider
- It has links to the rest of functionalities

Random

This route has to be accessed by clicking a button placed in the index page, after that a random photo will be provided with its comments likes and username.

Gallery:

This route can be accessed by clicking a button in the index page, after that it shows 9 random photos.
When the user adds information (#tags,user…) on the form and performs the submit a post method is called to retrieve the images which metadata fits on the search parameters.

Conclusions

After implementing the project we realized that:

- Having the same information structure and type of object (JSON) both on the database and the communication server-client avoids a lot of work on parsing and processing the information.
- Complex data that is not heavily-related like image's metadata is easier to store on non-relational databases than relational ones.
- It was such a nice idea to choose NodeJS because it's code-friendly and makes it easy to implement the server-side functionalities.

** Link to the repository:
https://github.com/ericpresas/NodeJS_Photo_Metadata.git