http://localhost:3000/img/logo.png bits, please.

**“bits, please”** is a web application that provides automation which pulls and analyzes the latest images from Flickr and allows you to sort them by color. Learn more at www.bitzplz.com .

**“bits, please” Overview**

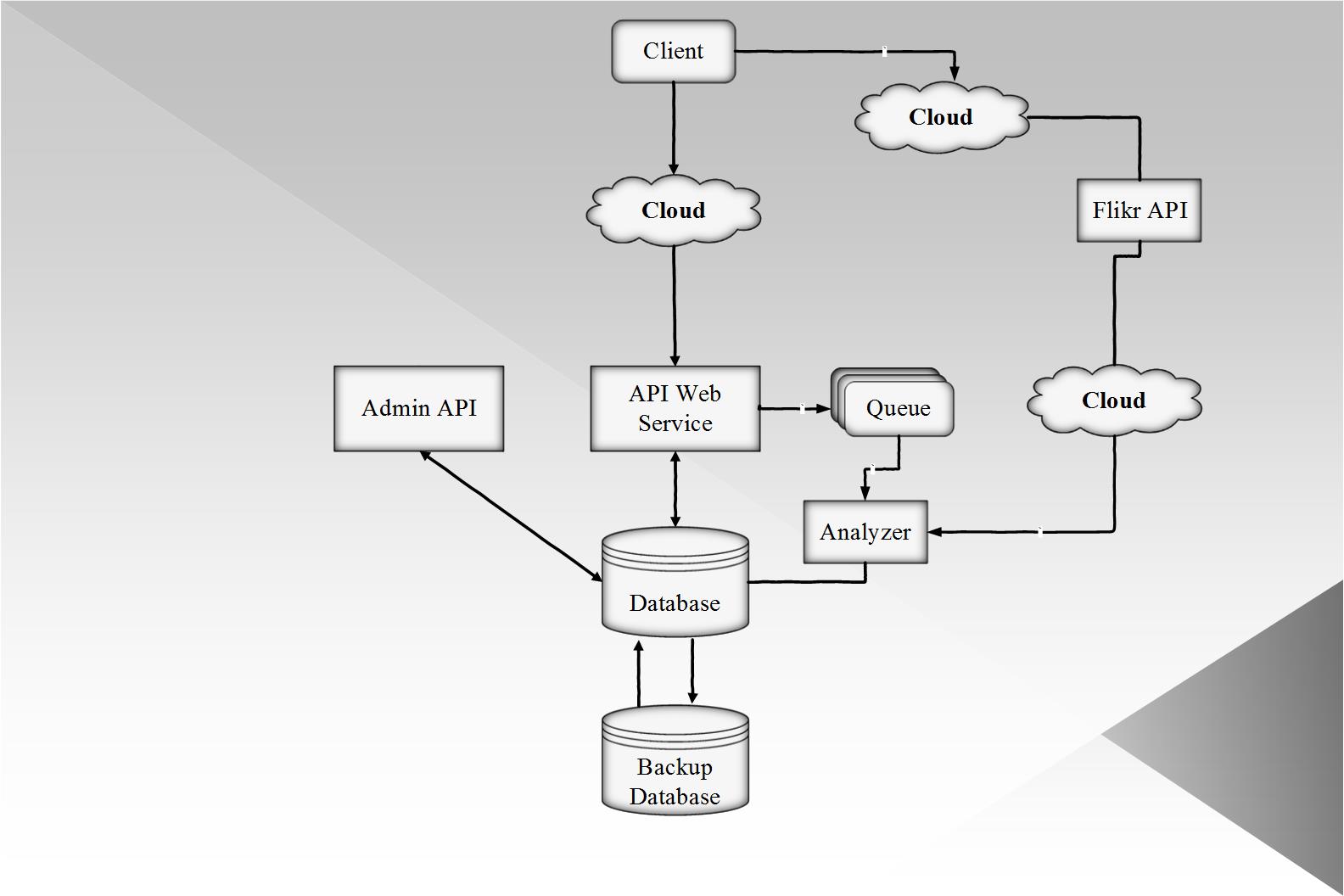
**About**

A web application that provides Flickr image library integration and automation across the tool to access Flickr’s massive image collections. It utilizes the Flickr API among other middleware to analyze and present images with meaningful data in real-time.

“bits, please” helps solve business problems by responding in real-time to requests:

* **Facilitated Image Searching** – Being able to use the API to retrieve the latest amount of pictures uploaded onto Flickr and sort that data into relevant groups having to reference the dominant color.
* **Automated Color Analysis** - Identifying and verifying the different colors in predominantly in each image and turning it into a graph that displays the colors for user’s to evaluate in percentage.
* **Continuous Deployment** - Build and update the meta-data in the database, continuously adding and building the amount of items that are populating in the demo page. Actively adding items to the page as the user scrolls down instead of waiting for a complete request of 500 images to be completed first.

**How it Works?**

**Components:**

* **Client** – The interface in which users interact with the application
* **API Webservice** – The interface in which the logic of the APIs are handled.
* **Analyzer** – The logic that performs queries with the Flickr API.
* **Database** – The database in which objects used in the application are stored.
* **Backup Database** – The permanent archival storage in which the current database can be restored from.

****

**Web Application Architecture using MEAN Stack**:

**MongoDB** - Serves as the Database component of the application

**AngularJS** - Serves as the Client component of the application

**NodeJS** - Serves as the API Webservice component of the application; includes modules for ExpressJS (webserver) and Mongoose (data mapper for MongoDB)

**Pomax's node-flickr-api** - Serves as the Analyzer component of the application; the "official" Flickr API wrapper for NodeJS

**\*SQL?** - A possible solution to restore the MongoDB database in case of failure (MongoDB exists as a cache of documents and does not truly function as a proper database).

**Interface Definitions**

These are some of proposed interface definitions for core functionality.

**API Webservice\***

**createRun -** Generates a new runID; launches the AnalyzerRun

**expireRun -** Marks run object as expired (set expiration date = now)

**restartRun -** Keeps image list from the createRun request, updates the expiration date of the run instance, and restarts AnalyzerRun

\*= the "run" verb is used in place of the word session to denote that no sensitive data is being held. For all intents and purposes, instances of the word "run" is considered a "session".

**Analyzer**

**startAndWaitForAnalyzerRun** - Starts an instance of AnalyzerRun and waits until analysis has been completed.

**analyzerComplete** - Denotes that the analysis on objects has been completed as to not overload the Analyzer with too many requests (implementation of singleton session)

**Analyzer** - Checks session periodically during run and halts process if session has expired.

$baseURL/$reportId/$imgId

**Future considerations**

We might want to consider a job queue in order to control the process of requests to the Analyzer from the API Webservice.

**Installation Instructions for application testing:**

**\*\*\* WARNING: OUR APPLICATION REQUIRES A LOT OF DEPENDENCIES AND RESOURCES THAT MAY TAKE UP A LOT OF TIME AND CONFIGURATIONS TO SET UP PROPERLY TO ACCESS IT VIA THIS METHOD. SUGGESTED TO USE THE WEB SITE FOR APPLICATION TESTING PURPOSES AND EVALUATIONS. \*\*\* :** [**http://www.bitzplz.com:3000/**](http://www.bitzplz.com:3000/)

**Install node.js**

To install node.js on Windows, just go to <https://nodejs.org/> official site and download Windows installer, then execute the installer.

**Install MongoDB**

Go to mongoDB official site <https://www.mongodb.org/downloads> to download zip file for Windows, and unzip the contents to a specific location. MongoDB will read data at \data\db by default, but mongoDB won’t create this folder for us, so we must create it individually, you can create this folder in Windows Explorer, or type the following command in terminal:

C:\> mkdir \data

C:\> mkdir \data\db

After creating \data\db, double click mongod.exe in your\_mongodb\_path\bin or type the following command in terminal to turn on mongoDB:

C:\> cd your\_mongodb\_path\bin

C:\> mongod

Then you can double click mongo.exe or type the following command in terminal to get into administrative shell：

C:\> cd your\_mongodb\_path\bin

C:\> mongo

Complete. You’ve successfully installed node.js and mongoDB on Windows!!!

**Installing Github via GUI & Bash:**

Download and install the latest version of GitHub for Windows. This will automatically install Git and keep it up-to-date for you.

On your computer, open the Git Shell application.

Tell Git your name so your commits will be properly labeled. Type everything after the $ here:

$ git config --global user.name "YOUR NAME"

Tell Git the email address that will be associated with your Git commits. The email you specify should be the same one found in your email settings. To keep your email address hidden, see "Keeping your email address private".

$ git config --global user.email "YOUR EMAIL ADDRESS"

To run Git and be able to clone the repository. Navigate to the directory you wish to clone the repository.

$ git init

$ git clone <https://github.com/jbirds/ITC2015.git>

Complete!

**How to setup a node.js & MongoDB development environment on Mac OSX Lion**

Please follow this link for Mac OSX installation procedures.

<http://dreamerslab.com/blog/how-to-setup-a-node-js-development-environment-on-mac-osx-lion/>

**How to setup a node.js development environment on Ubuntu 11.04**

Please follow this link for Ubuntu installation procedures

<http://dreamerslab.com/blog/how-to-setup-a-node-js-development-environment-on-ubuntu-11-04/>

**Running the Application Methods:**

**Method 1 – Bin/StartMongo.sh**

MongoDB runs as a standard program. You can start MongoDB from a command line by issuing the “mongod” command and specifying options. The following assume the directory containing the mongod process is in your system paths. The mongod process is the primary database process that runs on an individual server.

**Method 2 – You may run mongod/sudo & node app.js after.**

Issue the following command to start mongod:

Sudo service mongod start

Now verify that MongoDB has started successfully. Verify that the mongod process has started successfully by checking the contents of the log file at

/var/log/mongod/mongod.log for a line reading

[initandlisten| waiting for connections on port <port>

Where <port> is the port configured in /etc/mongod.conf, for our project port 3000 is used.

**Method 3 – Accessing the Web Application on the hosted website provided. (Preferred Method)**

Please use the link described or click on this link to access the site:

<http://www.bitzplz.com:3000/>