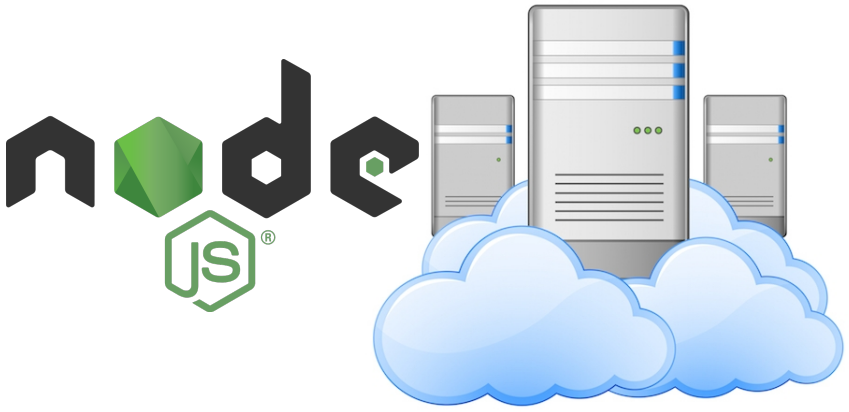
Social Photo Backend



mongoDB

android

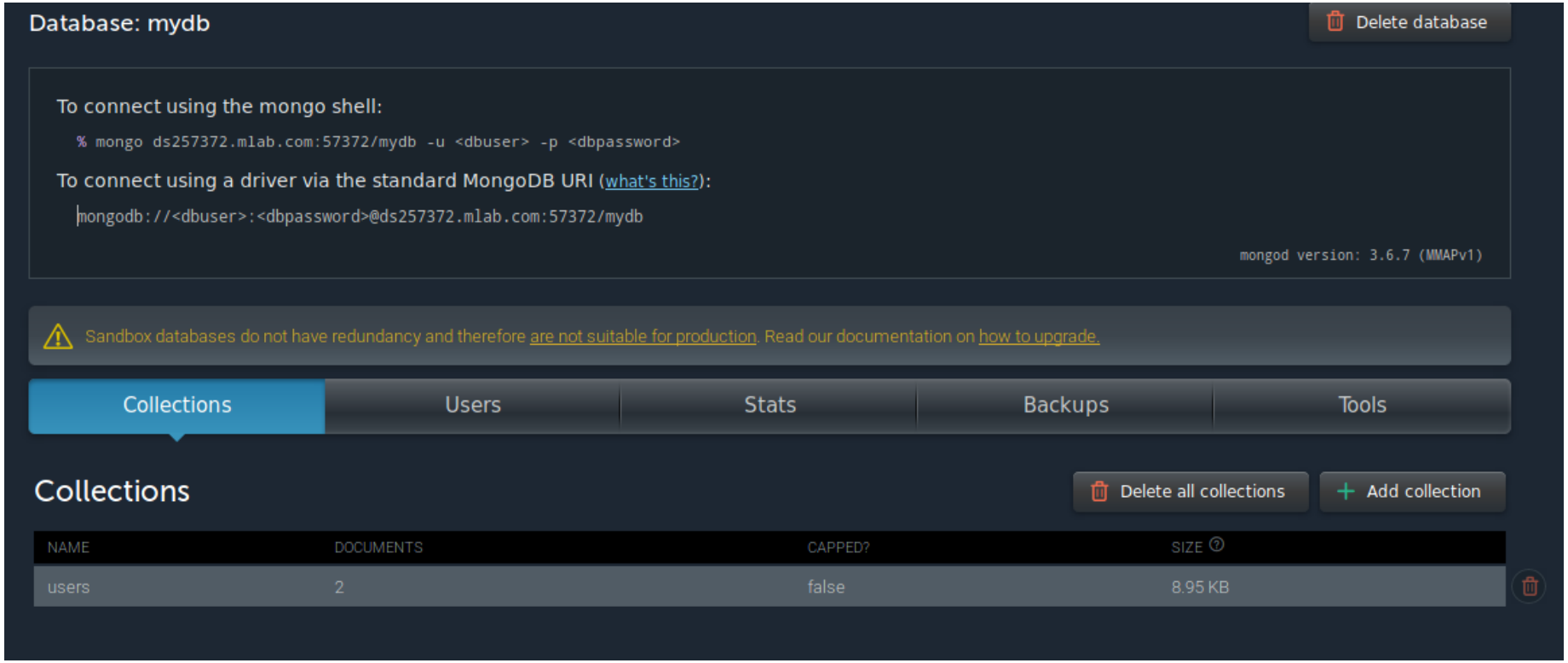
Candidati:
Carmine Maria Mansueto 1646454
Ivano Mazzola 1814282
Federico Bianca 1650901

Overview

- Backend is composed by two layers:
  - Model
  - Controller
- These two interact each other by means of API’s while the Client communicates only with the Controller
- Client is unaware of the architecture which allows the two layers to communicate

Technologies

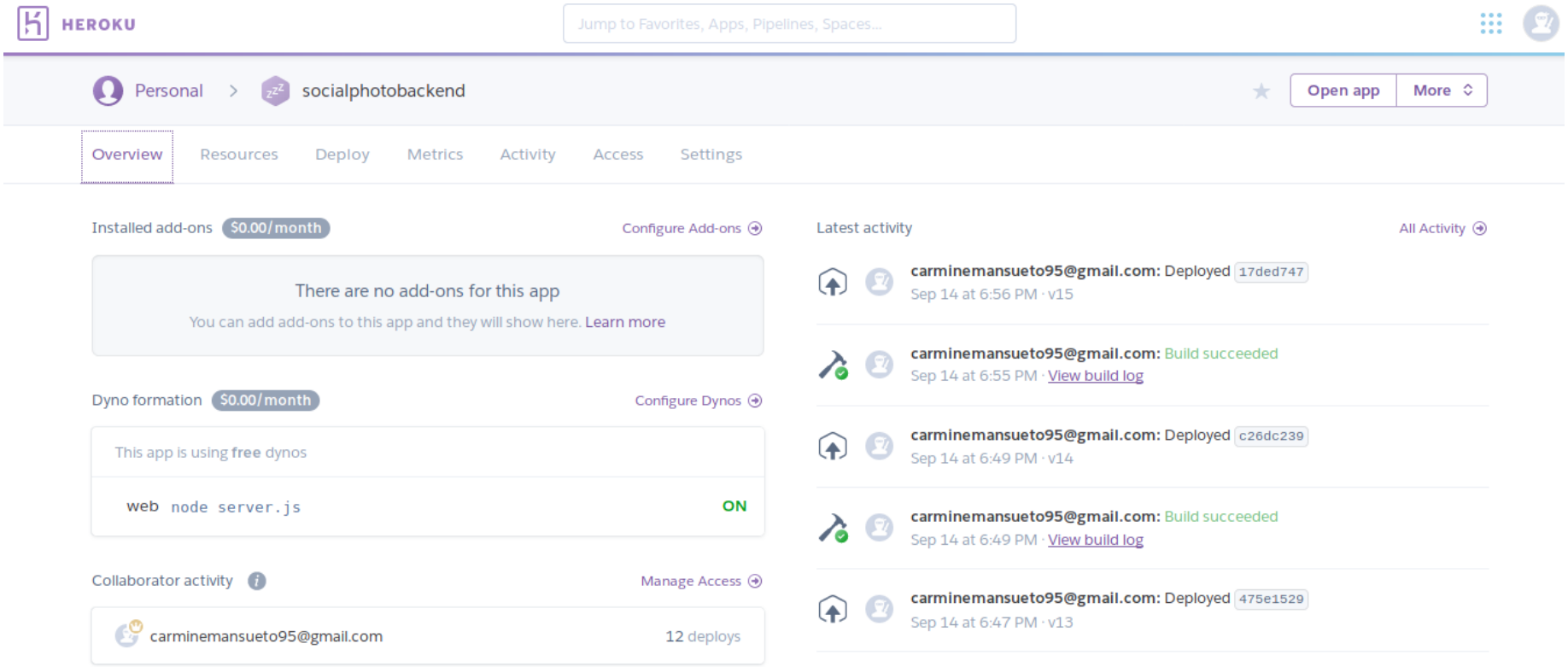
- Model:
  - Composed by a MongoDB database hosted on Mlab https://mlab.com/



Technologies

- Controller:
  - Written in NodeJS and deployed on Heroku

    https://www.heroku.com/



MongoDB

- MongoDB is a free and open-source cross-platform document-oriented database
- Classified as a NoSQL database
- Tables in DBMS match with Collections in Mongodb
- Tuples in DBMS match with Documents in MongoDB

MongoDB

- Collections in MongoDB (as in every NoSQL db) do not follow a precise schema:
  - Documents can have different number of fields
  - Fields type can be different among different documents in the same collection
  - Fields can be complex structures (this avoids join operations which are demanding in terms of computation)

MongoDB

- What we store:
  - DB is composed by a single collection ‘users’
  - In ‘users’ each document represents an Android’s app end user identified by its facebook id
  - For each end user are stored informations about his most liked photo and most commented photo involving a certain period as requested by him

MongoDB

- How we store it:
  - Thanks to the MongoDB API's it's pretty simple to interact with the DB through CRUD based methods like:
  - InsertOne

```
dbo.collection("users").findOne({FACEBOOK_ID: fb_id}, function(err, user){
```

  - FindOne

```
dbo.collection("users").findOne({FACEBOOK_ID: fb_id}, function(err, user){
```

  - UpdateOne

```
dbo.collection("users").updateOne({FACEBOOK_ID: fb_id}, newValues, function (err,res) {
```

NodeJS

- Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser

- Controller is composed by:
  - Server
  - Routes
  - Controller methods associated to routes

NodeJS

- Server:
  - Nodejs needs a .js file in order to set up preliminaries operations like port configuration, db initialization and routes setup

```
1   const MongoClient = require('mongodb').MongoClient;
2 › const express = require('express'),…
5   const utils = require('./utils');
6   var routes = require('./api/routes/todoListRoutes');
7
8   routes(app);
9   app.listen(port);
10  var MongoUrl = "mongodb://<Username>:<password>@ds245022.mlab.com:45022/mydb";
11  //Create a collection
12 › MongoClient.connect(MongoUrl, { useNewUrlParser: true }, function(err, db) {…
```

NodeJS

```
'use strict';
module.exports = function(app) {
  var todoList = require('../controllers/todoListController');

  app.route('/users')
    .post(todoList.create_a_user);

  app.route('/users/myUserTrigger')
    .get(todoList.read_a_user);

  app.route('/users/logOut')
    .post(todoList.setLoggedOut);

  app.route('/users/myUser')
    .get(todoList.respond);
};
```

Routes

Controller methods

```
'use strict';
const MongoClient = require('mongodb').MongoClient;
const utils = require('../../utils');

//var MongoUrl = "mongodb://localhost:27017/";
var MongoUrl = "mongodb://<Username>:<password>@ds245022.mlab.com:45022/mydb";

exports.create_a_user = function(req, res){…
exports.read_a_user = function(req, res){…
exports.setLoggedOut = function(req, res){…
exports.respond = function(req, res){…
```

NodeJS

- Dependencies and libraries:
  - **Express**: let us to be able to set up a server listening on a specific port and to build a route controller pattern
  - **Mongodb:** allows us to specify the db's URL and so to connect to it and to use its services (make queries, create collections etc.)
  - **Fb:** A simple library which let us to be able to interact with Facebook's Graph API

NodeJS

- FB APIs usage example

```
var urlPhotos = "me/photos?type=uploaded&limit=500000";
const FB = require('fb');
FB.setAccessToken(token);
FB.api(urlPhotos, function(res1) {
  //Salvo gli ID di tutte le foto in arrayID
  var i=0;
  while(res1.data[i] != undefined){
    if(res1.data[i].created_time.substring(0,7) == year+"-"+month){
      arrayID.push(res1.data[i].id);
      mapLikes.set(res1.data[i].id, "");
      mapComments.set(res1.data[i].id, "");
    }
    i++;
  }
```

- Through FB.api() it is possible to retrieve any kind of info, simply specifying the FB Graph URL
- The result is a JSON object which can be simply parsed using the dot notation(as a java developer does when accesses class members