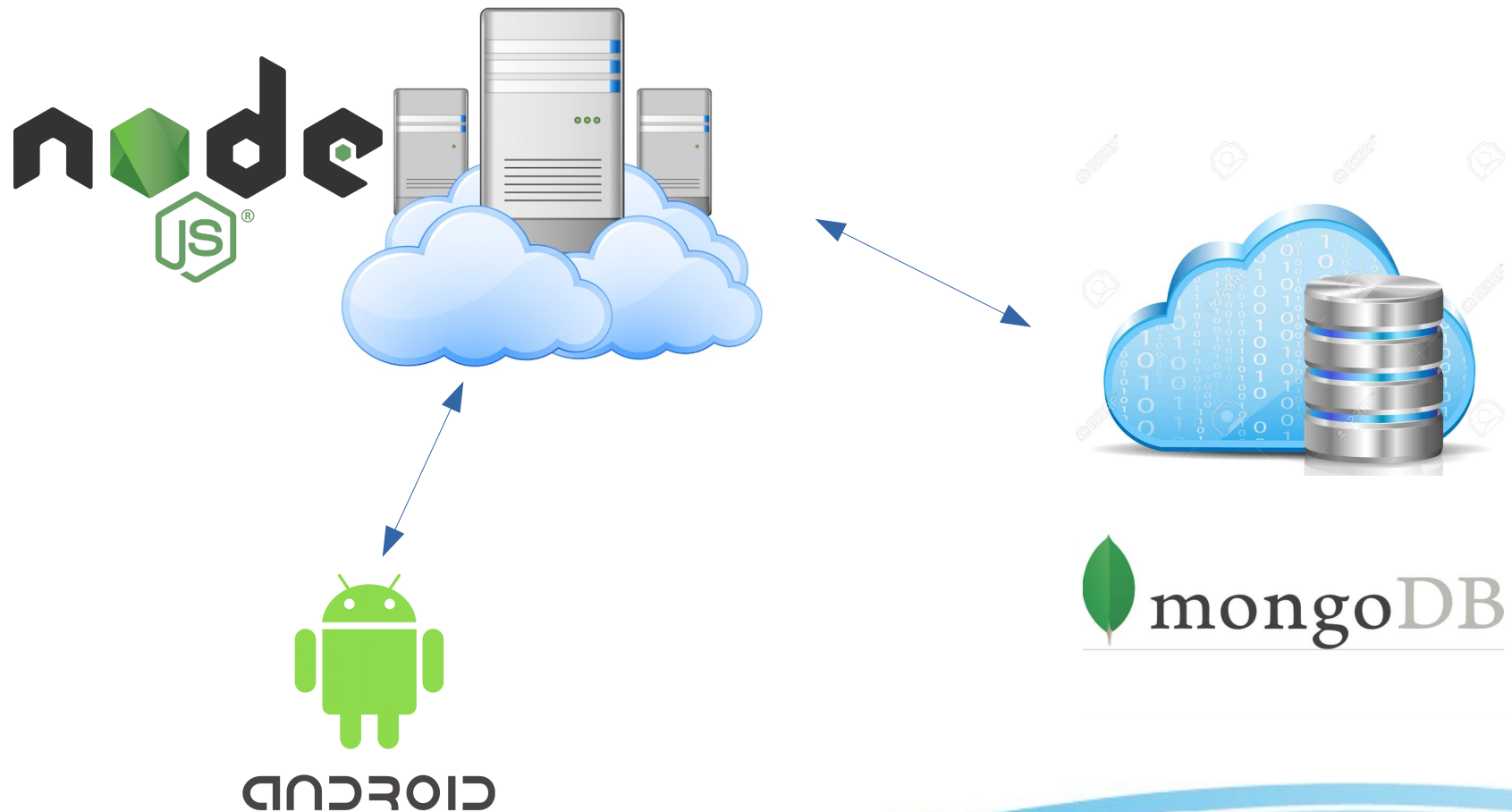


Social Photo Backend



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/>

The screenshot shows the Mlab MongoDB interface for a database named 'mydb'. At the top right, there is a 'Delete database' button. Below this, a box provides connection instructions for the mongo shell and the standard MongoDB URI. A warning message states: 'Sandbox databases do not have redundancy and therefore are not suitable for production. Read our documentation on how to upgrade.' Below the warning, there are tabs for 'Collections', 'Users', 'Stats', 'Backups', and 'Tools'. The 'Collections' tab is active, showing a table with one collection named 'users' containing 2 documents. At the bottom right of the collections list, there are buttons for 'Delete all collections' and 'Add collection'.

Database: mydb Delete database

To connect using the mongo shell:

```
% mongo ds257372.mlab.com:57372/mydb -u <dbuser> -p <dbpassword>
```

To connect using a driver via the standard MongoDB URI ([what's this?](#)):

```
mongodb://<dbuser>:<dbpassword>@ds257372.mlab.com:57372/mydb
```

mongod version: 3.6.7 (MMAPv1)

⚠ Sandbox databases do not have redundancy and therefore are not suitable for production. Read our documentation on [how to upgrade](#).

Collections Users Stats Backups Tools

Collections Delete all collections + Add collection

NAME	DOCUMENTS	CAPPED?	SIZE ⓘ
users	2	false	8.95 KB

⌵

Technologies

- Controller:
 - Written in NodeJS and deployed on Heroku

<https://www.heroku.com/>

The screenshot shows the Heroku dashboard for the 'socialphotobackend' app. The top navigation bar includes the Heroku logo, a search bar, and a user profile icon. Below the navigation bar, the app name 'socialphotobackend' is displayed with a star icon, 'Open app' button, and a 'More' dropdown. The main content area is divided into three sections: 'Installed add-ons', 'Dyno formation', and 'Collaborator activity'. The 'Installed add-ons' section shows a message that there are no add-ons for this app. The 'Dyno formation' section shows that the app is using free dynos and lists the buildpack 'web node server.js' as 'ON'. The 'Collaborator activity' section shows a list of collaborators, with 'carminemansueto95@gmail.com' having 12 deploys. On the right side, the 'Latest activity' section shows a list of recent deployments and builds, all of which were successful.

HEROKU

Jump to Favorites, Apps, Pipelines, Spaces...

Personal > socialphotobackend

Overview Resources Deploy Metrics Activity Access Settings

Installed add-ons \$0.00/month Configure Add-ons

There are no add-ons for this app
You can add add-ons to this app and they will show here. [Learn more](#)

Dyno formation \$0.00/month Configure Dynos

This app is using free dynos

Process	State
web node server.js	ON

Collaborator activity Manage Access

Collaborator	Deploys
carminemansueto95@gmail.com	12 deploys

Latest activity All Activity

- carminemansueto95@gmail.com: Deployed 17ded747
Sep 14 at 6:56 PM · v15
- carminemansueto95@gmail.com: Build succeeded
Sep 14 at 6:55 PM · [View build log](#)
- carminemansueto95@gmail.com: Deployed c26dc239
Sep 14 at 6:49 PM · v14
- carminemansueto95@gmail.com: Build succeeded
Sep 14 at 6:49 PM · [View build log](#)
- carminemansueto95@gmail.com: Deployed 475e1529
Sep 14 at 6:47 PM · v13



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



- 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) {
```


- 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

- 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

Routes

```
1 'use strict';
2 module.exports = function(app) {
3   var todoList = require('../controllers/todoListController');
4
5   app.route('/users')
6     .post(todoList.create_a_user);
7
8   app.route('/users/myUserTrigger')
9     .get(todoList.read_a_user);
10
11  app.route('/users/logOut')
12    .post(todoList.setLoggedOut);
13
14  app.route('/users/myUser')
15    .get(todoList.respond);
16 };
```

Controller methods

```
1 'use strict';
2 const MongoClient = require('mongodb').MongoClient;
3 const utils = require('../utils/');
4
5 //var MongoUrl = "mongodb://localhost:27017/";
6 var MongoUrl = "mongodb://<Username>:<password>@ds245022.mlab.com:45022/mydb";
7
8 > exports.create_a_user = function(req, res){=
28
29 > exports.read_a_user = function(req, res){=
39
40 > exports.setLoggedOut = function(req, res){=
56 > exports.respond = function(req, res){=
```

- 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

- 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