Junior Full Stack Web Developer Interview Task

This is a brief documentation to describe the provided solution and explain how to test it.

> **Important**: the mongodb server instance should be running.

Required Software

- Node.js
- MongoDB

Import data into database (1st part of the task)

Download data from provided url :

```
$ curl -o ~/data.json.gz  http://data.githubarchive.org/2015-01-01-15.json.gz
```

Unzip downloaded .gz file:

```
$ gunzip ~/data.json.gz
```

Import data into mongodb database:

```
$ mongoimport --db DATABASE_NAME --drop --collection events --file ~/data.json
```

> **Important**:
>
> - Replace **DATABASE_NAME** by the name of your database.
> - **--drop** option is used to drop the database in case it exists.

Instructions to test the API Server (2nd part of the task)

API Server Structure

```
- app/
---- config/
-------- db.js
---- models/
-------- event.js
---- routes.js
- node_modules/
- app.js
- package.json
- documentation.pdf
```

> **db.js** : contains database url. **event.js** : contains Event model (which corresponds to one record in the dataset). **routes.js** : manages routes to the api.**node_modules/**: directory containing the installed nodejs modules.**app.js**: contains node.js application. **documentation.pdf**: contains documentation of the application.

Launch the node.js instance

> **Important**: the mongodb server instance should be running.

> Unzip api_zerver.zip:

```
$ unzip interview_task.zip
```

Open db.js located in app/config/ and replace the url with the url of the database:

```
module.exports = {
    'url': 'mongodb://127.0.0.1/mydb'
}
```

> **Important**: The application uses port 3000 by default. If this port is used, make sure to change it in app.js file:

```
...
var PORT = 3000;
...
```

Go to interview_task directory:

```
$ cd interview_task
```

Run the node.js instance:

```
$ node app.js
```

Application Routes

Route	HTTP Verb	Description
/api/records	POST	Gets records filtered by repository id and event type. `Note: repo_id and actor_login are both required`
/api/actor	POST	Returns actor details and list of contributed repositories by actor login `Note: actor_login is an optional parameter that allows to retrieve records only for the specified login`
/api/toprepo	POST	Finds the repository with the highest number of events from an actor (by login). If multiple repos have the same number of events, it returns the one with the latest event. `Note: actor_login is required`
/api/repos	GET	Returns list of all repositories with their top contributor (actor with most events).
/api/delete_history	POST	Deletes the history of actor's events by login and returns number of records deleted. `Note: actor_login is required.`

Testing the API with Postman

The API can be tested easily with tools such as Postman

> **Important**: Make sure to use `x-www-form-urlencoded`

Data sample to test /api/records:

```
{ "repo_id": "28688495",   "event_type": "PushEvent" }
```

Data sample to test /api/actor, /api/toprepo, /api/delete_history:

{ "actor_login": "hex7c0"}