#Task Manager API

This is an example of a working API for a basic task manager app. Also, this README will serve you as an example for documenting an API, and to understand the documentation of others. The documentation approach is based on Blueprint (https://github.com/apiaryio/api-blueprint/blob/master/Tutorial.md).

##Documentation

FORMAT: 1A

###Task manager
Task manager is an API for managing tasks of individual users.

####Authentication [/authentication]

Get token [POST]

- Request (application/json)

```
{
    "username":"albertofer",
    "password":"12345678"
}
```

- Response (application/json)

```
{
    "user": {
        "_id": "5544f0e2dbf7ed6c06cdff21",
        "username": "albertofer",
        "tasks": [
            "5544f661cb41f99206900f22"
        ],
        "is_admin": false,
        "__v": 1
    },

"token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJfaWQiOiI1NTQ0ZjBlMmRiZjdlZDZjMDZjZGZmMjEiLCJ1c2VybmFtZSI..."
}
```

####Users collection [/users]
#####List All Users [GET]

- Response 200 (application/json)

```
[
    {
        "username": "jedi",
        "password": "12345678",
        "tasks": ['']
    },
    {
        "username": "bertofer",
        "password": "983dsaas",
        "tasks": ['']
    }
]
```

#####Create new user [POST]

- Request (application/json)

```
{
    "username": "albertofer",
    "password": "12345678"
}
```

- Response 200 (application/json)

```
{
    "_id": "5544f0e2dbf7ed6c06cdff21",
    "username": "albertofer",
    "tasks": [],
    "is_admin": false,
    "__v": 1
}
```

####User [/user]
#####Change password [PATCH]

- Request (application/json)

```
{
    "oldPassword": "12345678",
    "newPassword": "87654321"
}
```

- Response 200
  "Password changed correctly"

####Tasks collection [/user/:user_id/tasks]

List all tasks [GET]

- Response 200 (application/json)

```
 [
     {
         "_id": "554506e59130a45f09cd0a82",
         "title": "Do the shopping",
         "description": "Buy a new chess board",
         "__v": 0,
         "subtasks": [ ],
         "created": "2015-05-02T17:18:29.765Z",
         "completed": false
     },
     {
         "_id": "554507939130a45f09cd0a83",
         "title": "Buy a 20sided dice",
         "description": "Volunteers in jedi courses not a problem
anymore",
         "__v": 0,
         "subtasks": [ ],
         "created": "2015-05-02T17:21:23.884Z",
         "completed": false
     }
 ]
```


Create new task [POST]

- Request (application/json)


```
 {
     "title": "Do the shopping",
     "description": "Buy a new chess board"
 }
```


- Response 200 (application/json)


```
 {
     "__v": 0,
     "title": "Do the shopping",
     "description": "Buy a new chess board",
     "_id": "554506e59130a45f09cd0a82",
     "subtasks": [ ],
     "created": "2015-05-02T17:18:29.765Z",
     "completed": false
 }
```


####Task [/user/:user_id/task/:task_id]

Modify task [PATCH]

The fields that can be modified are: title, description, completed, dueDate, subtasks.

- Request (application/json)

```
{
    "completed":true
}
```

- Response 200 (application/json)

```
{
    "_id": "554507939130a45f09cd0a83",
    "title": "Buy a 20sided dice",
    "description": "Volunteers in jedi courses not a problem anymore",
    "__v": 0,
    "subtasks": [ ],
    "created": "2015-05-02T17:21:23.884Z",
    "completed": true
}
```

Delete task [DELETE]

+Response 200 (application/json)

```
    "Deleted correctly"
```

##Other libraries used
In this example, some new libraries and modules has been user. There you have a list of them:

- lodash (https://lodash.com/) Refactor of other library called underscore, useful for managing objects and collections.
- bcrypt (https://www.npmjs.com/package/bcrypt) Library used to hash passwords and verify hashes.

##Other considerations for complex APIs
Error handling, versioning, processmanager, extra-services(mail, valid, password...)

###Error handling
To the api not being so complex, error handling hasn't been taken in much consideration. But in production-ready apis, this is very important. An example of more complex error handling is to return a JSON with error messages for users and developers, e.g:

```
    {
        dev_message:"The server responded with 500 due to very high load",
        users_message:"We couldn't do that, wait a moment!",
        status: 500,
        ...
    }
```

Here (http://expressjs.com/guide/error-handling.html) you can read an article on error handling in express apps.

###Versioning
When your api is on production, and you need to modify and create new versions of your api, and the different versions of the api grow, it's incredibly difficult to manage users from different versions of the api. One detail that facilitates the process is to put the version somewhere, for example in the url:

```
    /v1/user/:user_id/tasks
```

###Process managers
A process manager is a system process that manages the execution process of our api. Usually these process managers can launch more than one instance of the api (to use more cpu cores), or relaunch the process if it fails at any time. Here you have some process managers for node.js:

- Forever (https://github.com/foreverjs/forever)
- PM2 (https://github.com/Unitech/pm2)

HTTP Server

An http server is the layer that receives the http requests and redirects it to the process listening in localhost. E.G we have the node.js listening on port 8080, but the http port is usually 80. An http server could redirect mydomain.com (on port 80) to localhost:8080. The http server also serves the SSL certificates. Examples of http servers are:

- nginx (recommended for node.js) (http://nginx.org/)
- Apache HTTP server (http://httpd.apache.org/)