Office Sign Up

Description

Sign up system for office. Based on Python micro web framework Flask 0.10.1 and connected with MySQL database by python official MySQL connector.

Installation

- Execute `db/tables.sql` on MySQL command line or phpMyAdmin to create all necesaries tables.
- Edit `model/__init__.py` to set your database access info.
- If you wont specific port to run on your server, edit `__main__.py` and set your custom port on line 123:

  ```
  app.run(port=<int:port>)
  ```

- Execute `python __main__.py`

Index

Content

1. AUTH

Send to admin user generated token to execute actions with employees and sessions.

1.1 Check token

Check if token is valid or expired.

Request GET `/auth`

Headers

```
    token: <string:user_token>
```

Response OK 200

Response ERROR 403

Body [type: plain text]

```
    Forbidden
```

1.2 Login

Request access token from HTTP Basic Auth protocol.

Request POST `/auth`

Headers

```
    Authorization: <string:base64_encode(username:password)>
```

Response OK 200

Body

```
    {
        username: <string:username>,
        token: <string:token_generated>,
        email: <string:email address>,
        expire: <datetime:token_expire_date>
     }
```

Response ERROR 4XX

Body [type: plain text]

```
    <string:error message>
```

Response ERROR 5XX

Body [type: plain text]

```
    <string:error message>
```

1.3 Logout

Logout and set token to empty string and expire to zero datetime.

Request DELETE `/auth/<string:username>`

Headers

```
    Authorization: <string:base64_encode(username:password)>
```

Response OK 200

Response ERROR 4XX

Body [type: plain text]

```
    <string:error message>
```

Response ERROR 5XX

Body [type: plain text]

```
<string:error message>
```

SESSION

Module to display and control employees session info.

2.1 Get sessions

Show all sessions on database.

Request GET `/session`

Headers

```
token: <string:user_token>
```

Body [Optional]

```
date: <date:sessions day>
```

or

```
user: <int:user id>
```

Response OK 200

Body [type: json]

```
[
    {
        id: <int:session id>,
        start: <datetime: session start>,
        end: <datetime: session end>,
        user: <int:user id>
    },
    ...
 ]
```

Response ERROR 4XX

Body [type: plain text]

```
<string:error message>
```

Response ERROR 5XX

Body [type: plain text]

```
<string:error message>
```

2.2 Get session by ID

Show sessions filtered by session id.

Request GET `/session/<int:session id>`

Headers

```
token: <string:user_token>
```

Response OK 200

Body [type: json]

```
{
    id: <int:session id>,
    start: <datetime: session start>,
    end: <datetime: session end>,
    user: <int:user id>
}
```

Response ERROR 4XX

Body [type: plain text]

```
<string:error message>
```

Response ERROR 5XX

Body [type: plain text]

```
<string:error message>
```

2.3 Create session

Creates new session.

Request POST `/session`

Headers

```
token: <string:user_token>
```

Body

```
start: <datetime:session start>,
user: <int:user id>
```

Response OK 200

Response ERROR 4XX

Body [type: plain text]

```
<string:error message>
```

Response ERROR 5XX

Body [type: plain text]

```
<string:error message>
```

2.4 Update session

Update session based on its id.

Request PUT `/session/<int:session id>`

Headers

```
token: <string:user_token>
```

Body

```
    end: <datetime:session start>
```

Response OK 200

Response ERROR 4XX

Body [type: plain text]

```
    <string:error message>
```

Response ERROR 5XX

Body [type: plain text]

```
    <string:error message>
```

2.5 Delete session

Delete session from database.

Request DELETE `/session/<int:session id>`

Headers

```
    token: <string:user_token>
```

Response OK 200

Response ERROR 4XX

Body [type: plain text]

```
    <string:error message>
```

Response ERROR 5XX

Body [type: plain text]

```
    <string:error message>
```

EMPLOYEE

Module to display and control employees info.

3.1 Get employees

Return list with all employees.

Request GET `/employee`

Headers

```
token: <string:user_token>
```

Response OK 200

Body [type: json]

```
[
    {
        id: <int:user id>,
        name: <string:full name>,
        email: <string:email adress>,
        hours: <int:anual working hours>
    },
    ...
 ]
```

Response ERROR 4XX

Body [type: plain text]

```
<string:error message>
```

Response ERROR 5XX

Body [type: plain text]

```
<string:error message>
```

3.2 Get employee by ID

Return employee info based on id.

Request GET `/employee/<int:employee id>`

Headers

```
token: <string:user_token>
```

Response OK 200

Body [type: json]

```
{
    id: <int:user id>,
    name: <string:full name>,
    email: <string:email adress>,
    hours: <int:anual working hours>
}
```

Response ERROR 4XX

Body [type: plain text]

```
<string:error message>
```

Response ERROR 5XX

Body [type: plain text]

```
<string:error message>
```

3.3 Create employee

Create new employee.

Request POST `/employee`

Headers

```
token: <string:user_token>
```

Body

```
    name: <string:full name>,
    email: <string:email adress>,
    hours: <int:anual working hours>
```

Response OK 200

Response ERROR 4XX

Body [type: plain text]

```
    <string:error message>
```

Response ERROR 5XX

Body [type: plain text]

```
    <string:error message>
```

3.4 Update employee

Update employee based on id.

Request PUT `/employee/<int:employee id>`

Headers

```
    token: <string:user_token>
```

Body

```
    name: <string:full name>,
    email: <string:email adress>,
    hours: <int:anual working hours>,
    fingerprint: <string:finguerprint_signature>
```

Response OK 200

Response ERROR 4XX

Body [type: plain text]

```
    <string:error message>
```

Response ERROR 5XX

Body [type: plain text]

```
<string:error message>
```

3.5 Delete employee

Delete employee.

Request DELETE `/employee/<int:employee id>`

Headers

```
token: <string:user_token>
```

Response OK 200

Response ERROR 4XX

Body [type: plain text]

```
<string:error message>
```

Response ERROR 5XX

Body [type: plain text]

```
<string:error message>
```