Prod Setup

This project was implemented using Django. Therefore a setup using Gunicorn + nginx is recommended and will be briefly explained in the following sections. But of course you are free to use any other setup, if you'd like to do so.

Preconditions

- MySQL is up and running
- At least one project has already been created by a FAT-Client!!!

Install Requirements

Install the following packages, if you haven't already:

> `sudo apt-get install python-pip python-dev mysql-server python-mysqldb libmysqlclient-dev nginx git`

Download the WBS-Project-timetracker-backend

Move to the directory you'd like to keep the sources in and either download them manually or clone the git repository using:

> `git clone https://github.com/jdepoix/WBS-Project-timetracker-backend.git or download`

A common location would be something like `/var/www/`.

Setup Django Environment

Make a copy of `conf.template.json` and rename it to `conf.json`. Now fill in the settings, needed for your environment.

To setup the project run the setup script. You can find the setup script in the `setup` folder. This will create a virtualenv, install all python requirements, run the django migrations and collect the static assets.

> `sudo ./setup/prod`

You should maybe consider using a dedicated user instead of running as root.

Note that the setup scripts only supports Unix based OS'es.

Gunicorn

After you have setup your Django environment, you can try starting Gunicorn. Do this by running:

> `virtualenv/bin/gunicorn --bind 127.0.0.1:8080 -w 2 wbs_timetracker.wsgi.prod`

Gunicorn can now serve requests on localhost. I would highly recommend, adding Gunicorn to upstart, which provides a more robust way of handling Gunicorns lifecycle.

Adding gunicorn to upstart

Create and edit `/etc/init/gunicorn.conf` with the editor of your choice. Your upstart config could look something like this:

```
description "Gunicorn application server handling myproject"

start on runlevel [2345]
stop on runlevel [!2345]
```

```
respawn
chdir /var/www/WBS-Project-timetracker-backend

exec virtualenv/bin/gunicorn --bind 127.0.0.1:8080 -w 2 wbs_
```

Of course this is only an example configuration and you can adjust this to your server environment.

Now Gunicorn can be started using:

> `sudo service gunicorn start`

Nginx

Now we want to reverse proxy incoming requests to `127.0.0.1:8080`, for Gunicorn to serve them. This is done using Nginx.

To create a new Nginx config, create and open `/etc/nginx/sites-available/wbs_timetracker` in the editor of your choice.

A really simple configuration could look like this:

```
server {
    listen *:80;

    location / {
        proxy_set_header Host      $host;
        proxy_pass http://127.0.0.1:8080;
    }

    location /static {
        alias /var/www/WBS-Project-timetracker-backend/stati
    }
}
```

Now create a link to your config in the `sites-enabled` folder and remove the default config, if there is one:

> `ln -s /etc/nginx/sites-available/wbs_timetracker /etc/nginx/sites-enabled/wbs_timetracker`
>
> `rm /etc/nginx/sites-enabled/default`

and then restart Nginx:

> `sudo service nginx restart`

Of course this also only is an example configuration, which should be adjusted to your server environment. Also using HTTPS should be preferred. In this example HTTP is only used, for simplicities sake. The main points to take away from this is:

- forward incoming requests to the adress gunicorn is listening on
- forward requests to static assets to the static folder

Do some tweaking and adjustments to meet your needs and then you're ready to go!

Dev Setup

If you're settings the project up for development, also go through the steps described in the sections Preconditions, Install Requirements, Download the WBS-Project-timetracker-backend, Setup Django Environment. But instead of running the `prod` setup script, run `dev`.

Run Server

You can start the server by running Djangos `./manage.py runserver`.

Run Tests

You can run the test suite by running `./manage.py test --settings=settings.test`.

API endpoints

Login

`[POST] /api/login/`

- returns a authentication token, if correct credentials are supplied.

  ######POST DATA:

  ```
  {
      username: <String>,
      password: <String>
  }
  ```

- following request can be authenticated by adding "Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b" to the header

Users

`[GET] /api/users/(?(username))`

- lists all users

######PARAMS:

  - `username`: filter by the users username. Note that usernames are unique.

`[POST] /api/users/`

- creates a new user

  ######POST DATA:

  ```
  {
      /** new users username */
      username: <String>,
      /** new users password */
      password: <String>
  }
  ```

`[GET] /api/users/<user_id>/`

- the user with the user id `<user_id>`.

`[PATCH] /api/users/<user_id>/`

- change password of the user with the user id `<user_id>`.

  ######POST DATA:

  ```
  {
      /** the users new password */
      password: <String>
  }
  ```

`[GET] /api/users/<user_id>/projects/`

- get all projects ot the user with the user id `<user_id>`.

`[POST] /api/users/<user_id>/projects/`

- add the user to an already existing project

  ######POST DATA:

  ```
  {
      /** URL of the project you want the user to be adde
      project: <URL>
  }
  ```

Booking session

`[GET] /api/booking-sessions/`

- list the current booking session for the currently logged in user, in case there is an open session.

`[POST] /api/booking-sessions/`

- open a new booking session for the currently logged in user.

  ######POST DATA:

  ```
  {
      /** the URL to the workpackage ressource, this is b
      workpackage: <URL>
  }
  ```

`[GET] /api/booking-sessions/<booking_session_id>/`

- lists the booking session with the id `bookings_session_id`.

`[DELETE] /api/booking-sessions/<booking_session_id>/`

- closes the current booking session with the id `bookings_session_id. But be aware that this doesn't create a new Booking, which has to be done manually using the booking endpoints. ``

  ## Projects

`[GET] /api/projects/`

- list all projects for the currently logged in user.

`[POST] /api/projects/`

- this endpoint is called, when a new project is created by the FAT-Client. It doesn't actually create a new project, it is just needed to update the backend, due to crappy legacy code.

  ######POST DATA: no post data is needed, since the information is already in the database.

`[GET] /api/projects/<project_id>/`

- show specific information about the project with the id `<project_id>`.

Bookings

`[GET] /api/projects/<project_id>/bookings/(?(date|workpackage_id))`

- lists all bookings on this project.

  ######PARAMS:

    - `date` : if date is set, only bookings from this date will be listed. Format: YYYY-MM-DD.
    - `workpackage_id` : if workpackage_id is set, only bookings on this workpackage will be listed.

`[POST] /api/projects/<project_id>/bookings/`

- creates a new booking.

  ######POST DATA:

  ```
  {
      /** link to the workpackage this booking belongs to
      workpackage: <URL>,
      /** the date of this booking. Format: YYYY-MM-DD */
      date: <date>,
      /** the workeffort in workdays (8h) */
      effort: <double>,
      /** the description of what was done */
      description: <String>,
      /** OPTIONAL: the new ETC for the corresponding wo
      newETC: <double>
  }
  ```

`[GET] /api/projects/<project_id>/bookings/<booking_id>/`

- lists all information regarding the booking with the booking id `<booking_id>` .

`[PATCH] /api/projects/<project_id>/bookings/<booking_id>/`

- updates the booking with the id `<booking_id>` . Data format is the same as for POSTs.

`[DELETE] /api/projects/<project_id>/bookings/<booking_id>/`

- deletes the booking with the id `<booking_id>`.

Workpackages

`[GET] /api/projects/<project_id>/workpackages/(?(toplevel_wp|inactive))`

- lists all workpackages for the project with the id `<project_id>`.

  ######PARAMS:

  - `toplevel_wp`: if true only toplevel workpackages are shown, if false only non toplevel workpackages.
  - `inactive`: if true only inactive workpackages are shown, if false only active workpackages.

`[GET] /api/projects/<project_id>/workpackages/<workpackage_id>/`

- lists specific information regarding the workpackage with the id `<workpackage_id>`.

`[PATCH] /api/projects/<project_id>/workpackages/<workpackage_id>/`

- update the ETC of the workpackage with the id `<workpackage_id>`.

  ######PATCH DATA:

  ```
  {
      /** new etc for this workpackage */
      etc: <double>
  }
  ```