RUG Poliklinieken Planning Tool

Joshua Rubingh

Created on : May, 2020

Last updated : May, 2020

Table of contents

List of Figures

List of Tables

Here you can read more information about the Poliklinieken Planning Tool.

Chapter 1

Installation

In order to install this Polyclinic Schedule Tool, we use the following packages / software.

- NGINX
- MySQL
- Django

First we need to checkout the code.

```
git clone https://git.web.rug.nl/P300021/poli_planning.git /opt/poli_planning
```

1.1 NGINX

Install NGINX. For Ubuntu this would be

```
sudo apt install nginx
```

Also configure SSL (https://letsencrypt.org/) to make the connections secure. This is outside this installation scope.

1.1.1 Setup

After installation of the packages, create a symbolic link in the */etc/nginx/sites-enabled* so that a new VHost is created.

```
ln -s /opt/poli_planning/nginx/vhost.conf /etc/nginx/sites-enabled/poli_planning
```

Important parts of the VHost configuration:

```
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
```

(continues on next page)

(continued from previous page)

```
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##

# Default server configuration
#

server {
    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;

    server_name poli-planning.hpc.rug.nl localhost;

    access_log /var/log/nginx/poli-planning.hpc.rug.nl.access.log;
    error_log /var/log/nginx/poli-planning.hpc.rug.nl.error.log;

    location /static {
        alias /opt/poli_planning/static;
    }

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        # try_files $uri $uri/ =404;

        proxy_pass         http://localhost:8000;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection "upgrade";

        proxy_redirect     off;
        proxy_set_header   Host $host;
        proxy_set_header   X-Real-IP $remote_addr;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Host $server_name;
        proxy_set_header   X-Forwarded-Proto $scheme;
    }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot

    ssl_certificate /etc/letsencrypt/live/poli-planning.hpc.rug.nl/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/poli-planning.hpc.rug.nl/privkey.pem; # managed by Certbot
```

(continues on next page)

(continued from previous page)

```
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}


server {
    if ($host = poli-planning.hpc.rug.nl) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    listen [::]:80;

    server_name poli-planning.hpc.rug.nl localhost;
    return 404; # managed by Certbot
}
```

In order to test if NGINX is configured correctly run *nginx -t* and it should give an OK message:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

1.2 Django

We install Django with standard settings. We could run it in Aync way, but then you need some more steps: https://docs.djangoproject.com/en/3.0/howto/deployment/asgi/ So for now, we keep it simple.

Create a python virtual environment

```
cd /opt/poli_planning
python3 -m venv python3_env
source python3_env/bin/activate
```

Finally we install the required Python modules

```
pip install -r requirements
```

This will install all the needed Python modules we need to run this Django project.

1.2.1 Settings

The settings for Django are set in an *.env* file so that you can easily change the environment from production to testing. There is an *.env.example* file that could be used as a template.

```
# A uniquely secret key
SECRET_KEY=@wb=#(f4uc0l%e!5*eo+aoflnxb(@!l9!=c5w=4b+x$=!8&vy%'

# Disable debug in production
DEBUG=False

# Allowed hosts that Django does server. Take care when NGINX is proxying infront of Django
ALLOWED_HOSTS=127.0.0.1,localhost

# All internal IPS for Django. Use comma separated list
INTERNAL_IPS=127.0.0.1
```

(continues on next page)

(continued from previous page)

```
# Enter the database url connection: https://github.com/jacobian/dj-database-url
DATABASE_URL=sqlite:////opt/poli_planning/polyclinic_scheduling/db.sqlite3

# The location on disk where the static files will be placed during deployment. Setting is required
STATIC_ROOT=

# Enter the default timezone for the visitors when it is not known.
TIME_ZONE=Europe/Amsterdam

# Email settings

# Mail host
EMAIL_HOST=

# Email user name
EMAIL_HOST_USER=

# Email password
EMAIL_HOST_PASSWORD=

# Email server port number to use
EMAIL_PORT=25

# Does the email server supports TLS?
EMAIL_USE_TLS=

# The sender address. This needs to be one of the allowed domains due to SPF checks
# The code will use a reply-to header to make sure that replies goes to the researcher and not this␣
↪address
EMAIL_FROM_ADDRESS=Do not reply<no-reply@rug.nl>
```

Next we have to make the database structure. If you are using SQLite3 as a backend, make sure the database file **DOES** exist on disk.

```
touch /opt/poli_planning/polyclinic_scheduling/db.sqlite3
```

Then in the Python virtual environment we run the following commands:

```
./manage.py migrate
./manage.py compilemessages
./manage.py collectstatic
```

And finally you should be able to start the Django application

```
./manage.py runserver
```

Chapter 2

Models

class lib.models.base.MetaDataModel(*args, **kwargs)

This is an abstract model with some general meta fields.

created_at

The date and time when the model has been created. This will be automatically set once during creating.

Type datetime

updated_at

The date and time when the model has been updated. This will be automatically updated when the model is updated.

Type datetime

class apps.employee.models.Employee(*args, **kwargs)

A model that holds the employee information that is not available in the normal user model. It has a One To One relation with the Djano User model It will inherit the attributes *created_at* and *updated_at* from the Abstract model *MetaDataModel*

user

The Django User model that is the employee.

Type User

hospital

The hospital where this employee is working. You can only choose **one** hospital per employee.

Type *Hospital*

polyclinic

The polyclinic where this employee is working within the hospital. It is possible to have / work for multiple polyclinics.

Type *Polyclinic*

phone

Holds the direct phone number of this employee. Max length is 20 characters.

Type str

exception DoesNotExist

exception MultipleObjectsReturned

class apps.hospital.models.Hospital(*args, **kwargs)

A model that holds the hospital information. This is just basic information just for getting in contact. It will inherit the attributes *created_at* and *updated_at* from the Abstract model *MetaDataModel*

name

The name of the hospital. Max length is 200 characters.

Type str

`address`
The address of the hospital. Street and housenumber. Max length is 200 characters.

Type str

`postal_code`
The postcalcode of the hospital. Max length is 10 characters.

Type str

`city`
The city where this hospital is located. Max length is 60 characters.

Type str

`phone`
The general phone number of this hospital. Max length is 20 characters.

Type str

`exception DoesNotExist`

`exception MultipleObjectsReturned`

`class apps.polyclinic.models.Polyclinic(*args, **kwargs)`
A model that holds the polyclinic information. This is just basic information just for getting in contact. It will inherit the attributes *created_at* and *updated_at* from the Abstract model *MetaDataModel*

`hospital`
The hospital where this polyclinic belongs to.

Type *Hospital*

`name`
The name of the polyclinic. Max length is 200 characters.

Type str

`phone`
The general/direct phone number of this polyclinic. Max length is 20 characters.

Type str

`exception DoesNotExist`

`exception MultipleObjectsReturned`

`class apps.schedule.models.Schedule(*args, **kwargs)`
A model that holds the schedule information. Here we store the form data and let the Peregrine cluster make the calculations. It will inherit the attributes *created_at* and *updated_at* from the Abstract model *MetaDataModel*

`employee`
The employee that is the owner of this schedule.

Type *Employee*

`name`
The name of the schedule. Max length is 100 characters.

Type str

`email`
The email address where the results should be sent to. Max length is 100 characters.

Type str

`status`
The status of the schedule.

Type *ScheduleStatus*

`planning_source`
The complete schedule stored in JSON data object.

Type JSON

`exception DoesNotExist`

`exception MultipleObjectsReturned`

`class ScheduleStatus`
This is a sub class of Schedule which holds all the possible schedule statuses

`NEW`
The schedule is just created and waiting to be picked up by the Peregrine scripts.

`ACCEPTED`
The Peregrine scripts have accepted the new schedule job. And the input is valid.

`PROCESSING`
The Peregrine job is submitted to the job queue and should be starting soon.

`PROCESSED`
The Peregrine job is finished, and the results will be futher processed in order to create a new report with the outcome.

`DONE`
The schedule rapport is created and uploaded to the database. The Peregrine process is done.

`INVALID`
The entered data is invalid. Either directly by the posting of the form. Or when the Peregrine script could not read the input.

`FAILURE`
Something when wrong on Peregrine. Look at the logging output of the Peregrine job

`property done`
Checks if the processing of this schedule is done on the Peregrine cluster. This can be either with the status:

1. *DONE*
2. *INVALID*
3. *FAILURE*

Returns boolean – True when status is one of the above value.

Chapter 3

Vieuws

class apps.schedule.views.ScheduleListView(**kwargs)

This view will give a list of all entered schedules. The list is filtered on the logged in employee.

Only the schedules owned by the logged in user are shown.

The results are shown with 10 items per page. A pager will be shown when there are more then 10 schedules

get_queryset()

Return the list of items for this view.

The return value must be an iterable and may be an instance of *QuerySet* in which case *QuerySet* specific behavior will be enabled.

model

alias of *apps.schedule.models.Schedule*

apps.schedule.views.new_or_update_schedule(*request*, *schedule_id=None*)

This view will create or update an existing schedule. At the moment there is not a real update, but a clone functionality. So every schedule that is updated will be stored as a new schedule.

Only schedules owned by the logged in user can be loaded here and can be cloned.

The data of the form is stored as a JSON object in a single database field for more flexibility.

Parameters HttpRequest -- This is the HTTP request from the Django framework. This will hold the current logged in user info. (*request*) –

Keyword Arguments Schedule -- This is the schedule to be edited. When none, a new schedule will be created (default (*schedule_id*) – {None})

Returns A view with an empty form to create a new schedule or a prefilled form for cloning.

Chapter 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index

U