Nadine Documentation

Release 2.0.14

Jacob Sayles

Jan 04, 2019

Getting Started

Nadine is a Django web project which runs behind the scenes of coworking spaces.

Nadine has four applications: Members, Staff, Admin, and Tablet. These applications help your management team facilitate your coworking community.

CHAPTER 1

License & Copyright

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

2.1 Requirements

- Python 3.6
- Virtualenv (Virtual environment)
- Homebrew if you are on Mac OS X (http://brew.sh)
- Postgresql

Important: Do not use SQLite.

2.1.1 Base System Installation

On Mac OS X

```
$ git # If you have not installed it, this will prompt you to download it.
$ brew update
$ brew install postgres python3 cairo pango
$ pip3 install virtualenv
```

On Ubuntu/Debian

```
$ sudo apt-get update
$ sudo apt-get install git postgresql postgresql-server-dev-all python3-pip python3-
↪dev virtualenv
$ sudo apt-get install libffi-dev libghc-cairo-dev libghc-pango-dev
```

Once that is ready, you can start the *quickstart*

2.2 Quickstart

Install the *required systems*

Setup the database

```
$ sudo su postgres -c "createuser -s $(whoami)"
$ createdb nadinedb
```

Create a virtual environment for the python project

```
$ virtualenv nadine
$ cd nadine
$ source bin/activate
```

Download the nadine source code from Github

```
$ git clone https://github.com/nadineproject/nadine.git
$ cd nadine
```

Install all the requirements

```
$ pip3 install -r requirements.txt
```

Run these scripts to setup nadine, install the database, and create your admin user

```
$ ./manage.py setup
$ ./manage.py migrate
$ ./manage.py createsuperuser
```

At this point you can run the server

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

Visit your installation of Nadine at http://127.0.0.1:8000/

2.3 Changing Application Settings

All the settings files are in nadine/settings/base.py. You can override these values in your local setttings file. If you run the setup command this will be created for you from the example file.

2.4 Production Setup

In a production environment you want a webserver in front of the Django engine and the preferred one is Nginx. This will handle all inbound requests, server your ssl certificate, redirect http requests to https, and serve up static content in /media and /static.

2.4.1 Create Nadine User

```
$ sudo adduser nadine
$ sudo su - nadine
```

Follow all the instructions in *quickstart* as the nadine user.

Create a few important directories for later.

```
$ mkdir -p /home/nadine/logs/
$ mkdir -p /home/nadine/backups/
$ mkdir -p /home/nadine/webapp/run/
$ mkdir -p /home/nadine/webapp/media/
$ mkdir -p /home/nadine/webapp/static/
```

2.4.2 Install Nginx and Certbot

```
$ sudo apt-get install nginx certbot openssl
```

2.4.3 Get your LetsEncrypt certificate

Follow instructions here: *https://certbot.eff.org/all-instructions/ <https://certbot.eff.org/all-instructions/#debian-9-stretch-nginx>*

If you test your server using the SSL Labs Server Test now, it will only get a B grade due to weak Diffie-Hellman parameters. We can fix this by creating a new dhparam.pem file and adding it to our server block.

```
$ sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
```

2.4.4 Copy configuration files in to place

```
$ cd /home/nadine/webapp/nadine/conf
$ sudo cp etc/nginx/sites-available/nadine /etc/nginx/sites-available/nadine
$ sudo ln -sf /etc/nginx/sites-available/nadine /etc/nginx/sites-enabled/default
$ sudo cp etc/nginx/snippets/ssl-nadine.conf /etc/nginx/snippets/
$ sudo cp etc/nginx/snippets/ssl-params.conf /etc/nginx/snippets/
$ sudo cp etc/uwsgi/apps-available/nadine.yaml /etc/uwsgi/apps-available/
$ sudo ln -s /etc/uwsgi/apps-available/nadine.yaml /etc/uwsgi/apps-enabled/

Edit all configuration files to make sure your domain is correct.
```

2.4.5 Restart Nginx

```
$ sudo nginx -t
$ sudo systemctl restart nginx
```

2.5 Nadine Structure

2.5.1 Overview

Nadine is comprised of four applications: Members, Staff, Admin, and Tablet.

Members is the member facing application. We consider members to be users with active resource allocations. **Staff** is the application used by staff members to help with the overall management of the community. **Admin** is the application in which only application administrators can access as it has absolute access to data. **Tablet** is the application used for user sign-ins and for the interaction expected at the entrance of a space and greeting a new or returning user of the space.

2.5.2 Members Application

This application is what members will use to connect with each other, the coworking space staff, and greater community.

Note: Much of the information given by members and organizations is designated as either public (viewable by current members) or private (only viewable by staff and that specific member). This is clearly indicated to members and they opt in to sharing whatever of that information with which they are comfortable.

In the Members Application there are profiles for both members and organizations/companies. Each have a photo or logo, URLS they would like others to see, a short bio, and tags. While the listed items are public, privately the members and organizations can see emergency contacts, billing history, signed documents (such as a membership agreement), and their user activity.

A member and an organization can edit their profiles as often as they would like. One setting, though, that is set by application administrator is whether or not members are allowed to upload their own user photos. That setting is entitled ALLOW_PHOTO_UPLOAD. More information in *Changing Application Settings*.

Other features of the Members Application are the calendar of events, ability to subscribe and unsubscribe from mailing lists, Slack invitation (if allowed in Settings), and the general ability to see who the other members and organizations are and to make a request to connect.

One feature, tags, are interests they can share which are then sortable and searchable by other members and organizations. This allows a space to see how organizations self identify their industries and also what their members are doing.

The Members Application is laid similarly in the views and the templates.

- connect
- core
- events
- organization
- profile
- tags

2.5.3 Staff Application

The Staff Application allows the staff of a space to best manage memberships and the tasks required to run the space. It is the application designed for staff to easily track space usage, review billing and deposits, and edit member

information if/when needed.

The navigation of the Staff Application includes:

- Tasks/ToDo
- Member List
- Activity
- Billing
- Stats
- Logs
- Lists
- Settings

Tasks/Todo

This section of the Staff Application deals with tasks that staff must complete. Tasks can be assigned to specific staff members or left available for anyone to complete and mark as such. This is the default home page for a staff member.

Member List

Member List shows all members in a sortable manner and then allows a staff member to edit any person's information as needs.

Activity

Activity is for recording any members particular use of the space and to generate reports on past usage and membership levels.

Billing

It is important to know that Nadine does not store any person's credit card information. Nadine in its current iteration uses a USAEpay integration for billing. In this section of the application, staff can track payments, generate reports, and run the daily billing.

Logs

Logs show device and user logins to the local internet. To know the user, the system remembers devices after a member has logged into Nadine from that computer. This allows us to better track use of space and to make sure that a member is using the space per their membership.

Lists

These pages are for the management of whatever mailing lists a space might have and the Slack channel, if a space has it.

Settings

Under Settings, staff has the ability to edit different elements of the application as well as see the settings as set by the Nadine Admin. Options from the Settings dropdown include:

- **Help Texts** - These are the documents for frequently asked questions. Each will have it's own page which members will be able to review.
- **MOTD** - MOTD stands for Message of the Day. This is the greeting which is presented on the home screen of the Tablet Application.
- **Membership Packages** - Allows staff to create and edit the default subscriptions and pricing for membership packages
- **Edit Rooms** - Staff can add and edit the details of rooms which are available for users to reserved

2.5.4 Admin

Like most admin applications, this has absolute access to user and space data. As stated before, though, this does not include any credit card information. Only application administrators have access to this part of Nadine.

2.5.5 Tablet Application

The Tablet is designed for use on an iPad at the entry of a space as a sort of portal. The user has access to sign in and see who else is in the space from this. Additionally, a user can sign documents such as a membership agreement.

2.6 Themes

Nadine's Member application is designed to be changed to better serve each space. Want to include your logo, company color scheme, etc? We tried to make that easy for you.

Currently the Member application is styled to be rather generic and includes the Nadine Logo throughout. What logo is that? That's the cow on the index page of this documentation. You are welcome to stay with this design but we also welcome you to get in the sandbox and make the Member application unique to your coworking space.

2.6.1 Creating a Theme

It is easy to create your own theme and implement it with the Member application.

Create a new project with the following doc tree:

```
THEME_NAME/
├── theme_settings.py
├── static/
│   ├── css/
│   │   └──members.css
│   └── img/
│   └── js/
│   └── fonts/
├── templates/
│   ├── members/
└── .gitignore
```

Static Folder

The static/ folder will contain all of your new styling(css), any particular javascript files you might need, new font files, and images. Here you can include the style sheets for any new CSS framework you might use and/or your own stylesheet.

The **members.css** file will be the most important for your new styling. This is where you can override the stylings from the default theme.

To completely override the layout of a page, you will need to write that page with DTL and HTML and include that in the templates/members folder.

Logo

In in the img/ folder, you can include your logos which you will use. If you do not intend to change the HTML then you will need to include two versions of your logo and save them as **logo.png** and **logo-line.png**. The first one to be used on the homepage jumbotron and the other to be part of the top navigation throughout the app.

Theme Settings

In theme_settings/ you can set the local settings for the application. The settings available include things such as the social media URLS, permissions for registration and photo uploads, and others.

For example, a theme_settings file for Office Nomads might look like:

```
ALLOW_ONLINE_REGISTRATION = False
ALLOW_PHOTO_UPLOAD = False

FACEBOOK_URL = "https://www.facebook.com/OfficeNomads"
TWITTER_URL = 'https://twitter.com/OfficeNomads'
YELP_URL = 'https://www.yelp.com/biz/office-nomads-seattle-2'
INSTAGRAM_URL = 'https://www.instagram.com/officenomads/'
```

Implementing the Theme

First, copy your new theme folder into the themes/ folder. Then, in the terminal:

```
$ cd themes
$ ln -s THEME_NAME active
```

This command tells Nadine to prioritize your theme over the members.css that came with it. Reload the Member App and see how it all looks!

2.7 Django Templating

Nadine uses Django as our backend framework. With Django comes the awesome power of the Django templates. The templates are HTML with the Django template language (DTL) and any necessary JavaScript.

For more information on DTL and Django templates, check out the documentation.

2.7.1 Nadine's Usage of Django Templates

To make the DRY-est html, we have used the templates to manage repeated code. You will notice a base.html in all of the applications. This is the file which will layout the head, navigation, and footer. From there, each of the pages will include a version of:

```
{% extends app_name/base.html %}
```

In the Staff App, the templates are divided up even more and each of those folders include their own base.html which extends the staff/base.html and then sets some basic navigation styling and brings in a stylesheet for that section.

Below the 'extends' code, there might also be such code as:

```
{% load static %}
```

This loading of static or settings or whatever is called is bringing in a variable from the backend to be used.

To maintain our DRY code, there are points in the HTML in which the template 'includes' another HTML page. An example of this would be the date_range_form.html which is repeatedly used in the Staff App with:

```
{% include "staff/date_range_form.html" %}
```

Again, for more info on Django templating, please see the documentation.

2.8 Static Files

Per Django documentation, Nadine serves static files by setting a STATIC_URL in the settings file and then loading the static shortcut then including it in a path like this:

```
{% load static %}
<img src="{% static "img/logo.png" %}" alt="Company Logo"/>
```

The static folder includes all images Nadine will use, stylesheets, JavaScript files, and fonts. Each application has its own static folder in which you must include any necessary item to be used in that application.

2.9 Staff Application

Nadine's Staff Application is really where this whole project got started. The original iteration was just referred to as the 'daily log' and was used by staff to track who had come into the space during the day and then was used to help with billing. Nadine's current Staff Application makes tracking subscriptions, usage, and billing easily.

2.9.1 Tracking Usage

Staff have the ability to see, real-time, who is signed into the space. Based on a user's subscriptions, Nadine will set a visit as billable or waived but staff can override that easily. Nadine will send emails to staff to remind them of any tasks to be completed for checked-in members. With that information, staff can go into Nadine and mark tasks as completed or assign a task to another person. This action will lead to that task to appear on the top of the assigned person's to-do lists until completed.

2.9.2 Tracking Billing

Proper billing allows the space to function well and Nadine is designed to automatically handle bill generation but allow for staff access to editing said bills. Members regularly have questions about billing and Nadine is designed to allow staff the ability to

When a person becomes a member of the space, staff can use Nadine to transfer billing information for use with an online payment platform. If at any time a member needs to update that information, staff can do so on Nadine easily. If there is no billing profile or an issue with billing, staff can set a profile as invalid which will notify the member to find staff to edit their information.

In the 'Daily Charges' page, Nadine displays current transactions and their statuses. Staff from here can void a payment if set in error or a cancelled membership and then can record payments.

Staff can also open current or past bills, edit line items, add custom charges (bought a t-shirt?), and email receipts.

2.9.3 Editing Memberships

To add, edit, or end subscriptions, Nadine is set to allow staff easy access to these tools. Nadine is not designed to allow members to change their own memberships so staff must manage this.

Staff may also edit the billing day for a user. Initially, when you start running Nadine, you should set the DEFAULT_BILLING_DAY in your local settings. The integer is the day of the month which all bills will be run. If set to 0 then billing days will be based on when a members signs up.

2.9.4 Member Actions

There are a number of actions staff may take with Nadine to update member information or reach out to a member.

On a member's Staff Application profile, staff may send emails to a member, reset their password, add a note about the member only viewable by other staff, upload or delete files regarding the member (i.e. Membership Agreement), assign or delete a keycard, and edit their profile information and photo.

2.9.5 Settings Pages

As described in the Settings section, the are there to allow staff to create and edit elements of the space without accessing the admin interface. The editable elements are the **Help Texts, Messages of the Day** (displayed on tablet at sign in), **Membership Packages** and **reservable rooms**.

2.10 Testing

Nadine is written with front-end and back-end tests. You are welcome to run the tests locally. If you do run into any issues, please enter it as an issue in Github.

2.10.1 Front-End Testing

For Front-End testing, Nadine uses CasperJS which is a 'navigation scripting & testing utility for the PhantomJS (WebKit) and SlimerJS (Gecko) headless browsers, written in Javascript.'

To run these tests, you first must have CasperJS installed and make sure that PhantomJS is installed.

```
$ brew install casperjs
$ phantomjs --version
```

If you do not have phantomjs, then use brew to install it.

To run all tests:

```
$ ./manage.py runserver  #make sure you have the server or running it will error out
$ casperjs test frontend-testing/tests --username='YOUR_USERNAME' --password='YOUR_
↪PASSWORD' --path='/PAGE_TO_TEST/'
```

To run a singular test, include the filename after tests/ in the path. In particular, to run tests to verify all links return a status code of 200, we have a test for that. Include a new variable called 'path' and either assign it '/member/' or '/staff/'.

```
$ ./manage.py runserver
$ casperjs test frontend-testing/tests/linktesting.js --username=YOUR_USERNAME --
↪password=PASSWORD --path='/PAGE_TO_TEST/'
```

Suggested paths:

- '/member/'
- '/member/view/'
- '/staff/'
- '/staff/members/members/'
- '/staff/members/detail/USERNAME'

Some of the tests are checking for a lot of information so it might take a minute or so to run.

2.10.2 Back-End Testing

Django is wonderful and includes its own ability to run unit tests. According to the docs, 'Django's unit tests use a Python standard library module: unittest. This module defines tests using a class-based approach.' For more detailed information on Django testing then please see the documentation

To run backend tests, you can be specific or more broad. To run all tests:

```
$ ./manage.py test nadine.tests
```

To run one specific test, like from the room booking test suite:

```
$ ./manage.py test nadine.tests.test_room.RoomTestCase.test_available_straddling
```

2.11 Mailing Lists

In the interest of shipping more quickly, we have made certain assumptions about the interlink mailing lists which may or may not suit everyone's needs.

- the reply-to address for mail from a list is the original sender, not the entire list
- attachments are neither saved nor sent to the list, but a removal note is appended to the message
- incoming messages are parsed for a single text message and a single html message (not multiple MIME messages)

- loops and bounces are silently dropped
- any email sent to a list which is not in a subscriber's user or membership record is moderated
- the sender of a message receives a copy of the message like any other subscriber

2.12 LDAP Syncing

Installation on a Debian server requires the python3-dev libsasl2-dev packages to be installed:

```
$ sudo apt-get install python3-dev libsasl2-dev
```

2.13 Events

For many coworking spaces, the ability to host and track events is very important. In Nadine, we have made it so that you can display a styled Google calendar or a built-in calendar using the PostgreSQL saved events.

2.13.1 Google Calendar Setup

If you would like to use a Google Calendar then you must first create a Google calendar attached to a Google email account. It is with that account that you can add and edit events. Once that is done, you can proceed with connecting Nadine to this calendar then you just need to make a couple of additions to the local_settings.

To use a Google Calendar in the place of a local calendar, you will need to acquire a Google API key and the Google Calendar ID for previously created calendar. In **nadine/local_settings**, set each appropriately as GOOGLE_API_KEY and GOOGLE_CALENDAR_ID.

This calendar is only set to be viewable by members but not editable from the Members application. Members can click on events on the calendar and they will be redirected to the Google page including event details. Additionally, this calendar will not display member room bookings made within the application.

2.13.2 Built-In Calendar

To use Nadine's built-in calendar, it is important to make sure that you do not have a GOOGLE_CALENDAR_ID in local_settings or comment it out if you do. There is only one optional setting in local_settings and that is to set colors for different rooms/spaces in the calendar. If interested in doing this, set it like the example below, with the room name and then color.

```
CALENDAR_DICT = {'Pike':  'RGBA(249, 195, 50, 1)', 'Pine':  'RGBA(249, 195,
50, 1)'}
```

The built-in calendar allows members and staff to add public events to the calendar. These are events that do not require an exclusive room booking and are along the lines of a Member Lunch, Lunch & Learn, Open House, etc. Members can click on the individual events to see time and host information. Additionally, members can add events to the calendar from clicking right on the date. These are only public events and not for a specific meeting room.

2.13.3 Room/Event Booking

If you opt to have room/event booking then you have a few items to set up. In local_settings set the hours when the space opens and closes. See example:

```
OPEN_TIME = '8:00'
CLOSE_TIME = '18:00'
```

In the Staff Application you can add rooms which members can book under settings/edit_rooms. The option of 'Members Only' means that a room is free for members to use and not bookable by general public. Once rooms are created, you can edit the CALENDAR_DICT with the room name.

If members can receive discounts on rooms then set MEMBER_DISCOUNT with a decimal amount. This will be applied for any room not set as 'Members Only'.

Members can have subscriptions for a certain number of room booking hours. Once they exceed an allowance then they will be charged the default hourly rate for the rooms.