vager# **Instal·lació del Virtualenv

Aquesta és una guia pas a pas per a la instalació de **Mailman Suite**, també anomenat **Mailman 3 Suite** o simplement **Mailman 3**.

Instal·lació de dependències

El primer de tot serà instal·lar **Mailman Core**, per tant, haurem d'afegir els següents paquets al nostre sistema:

- sudo dnf install python3-devel
- sudo dnf install epel-release
- sudo dnf install sassc
- sudo dnf install lynx

Configuració de la base de dades

Aquesta guia es basa en el motor de base de dades **Postgresql**. Els passos per a configurar Postgresql SQL son els següents:

- sudo dnf install postgresql-server postgresql-contrib
- sudo postgresql-setup --initdb
- sudo systemctl enable --now postgresql

Iniciem la comanda "**sudo -u postgres psql**" per fer login a l'usuari de postgres i poder entrar les següents dades a la BBDD:

- CREATE DATABASE mailman;
- CREATE DATABASE mailmanweb;
- CREATE USER mailman WITH ENCRYPTED PASSWORD 'MYPASSWORD';
- GRANT ALL PRIVILEGES ON DATABASE mailman TO mailman;
- GRANT ALL PRIVILEGES ON DATABASE mailmanweb TO mailman;
- \q

> **NOTA**
>
> S'han creat dues bases de dades anomenades **mailman** i **mailmanweb** per a **Mailman Core** i **Mailman Web (Django)** respectivament. També hem creat un nou usuari **mailman** i li hem concedit privilegis a les dues bases de dades.

Configurar l'usuari del Mailman

Crearem un nou usuari des d'on executarem tots els serveis de **Mailman**. Afegirem les següents comandes:

- sudo useradd -m -d /opt/mailman -s /usr/bin/bash mailman
- sudo chown mailman:mailman /opt/mailman
- sudo chmod 755 /opt/mailman
- sudo su mailman

Configuració del Virtualenv

Virtualenv és el mecanisme de Python per crear entorns d'execució aïllats.

> **NOTA**
>
> A partir d'ara, ens hem d'assegurar que estem executant les ordres com a usuari **mailman**.

Creem el virtualenv per a Mailman:

- cd /opt/mailman
- python3 -m venv venv

Activació del virtualenv

Activem el virtualenv que acabem de crear:

- source /opt/mailman/venv/bin/activate

Per a que l'entorn virtual **s'activi automàticament** al fer login amb l'usuari mailman, podem executar la següent comanda:

- echo 'source /opt/mailman/venv/bin/activate' >> /opt/mailman/.bashrc

A partir d'ara, cada cop que iniciem l'usuari **mailman**, també s'activarà el **virtualenv** de forma automàtica.

Instal·lació de Mailman Core

Mailman Core és el responsable **d'enviar i rebre** els correus electrònics. Exposa un REST API que diferents clients poden utilitzar per interactuar a través d'un protocol HTTP.

Així doncs, per instal·lar el core executarem:

- (venv)$ pip install wheel mailman psycopg2-binary

Això instal·larà l'última versió de **Mailman Core** i **enllaços Python** per a la base de dades Postgresql. Les versions posteriors de Django funcionaran amb psycopg2-binary 2.9.x.

Després d'això, creem un arxiu de configuració a **/etc/mailman3/mailman.cfg** per a Mailman Core:

> **NOTA**
>
> Com l'usuari **mailman** no té permisos d'administrador, haurem de sortir del usuari amb "exit" i crear el directori i l'arxiu des d'un usuari amb privilegis utilitzant "sudo"

- sudo mkdir /etc/mailman3
- sudo touch /etc/mailman3/mailman.cfg

Contingut de l'arxiu **mailman.cfg**:

```
# /etc/mailman3/mailman.cfg
[paths.here]
var_dir: /opt/mailman/mm/var

[mailman]
layout: here
# This address is the "site owner" address.  Certain messages which must be
# delivered to a human, but which can't be delivered to a list owner (e.g. a
# bounce from a list owner), will be sent to this address.  It should point to
# a human.
site_owner: user@example.com

[database]
class: mailman.database.postgresql.PostgreSQLDatabase
url: postgresql://mailman:MYPASSWORD@localhost/mailman

[archiver.prototype]
enable: yes

# For the HyperKitty archiver.
[archiver.hyperkitty]
class: mailman_hyperkitty.Archiver
enable: yes
configuration: /etc/mailman3/mailman-hyperkitty.cfg

[shell]
history_file: $var_dir/history.py
```

```
[mta]
verp_confirmations: yes
verp_personalized_deliveries: yes
verp_delivery_interval: 1

incoming: mailman.mta.postfix.LMTP
outgoing: mailman.mta.deliver.deliver
lmtp_host: 127.0.0.1
lmtp_port: 8024
smtp_host: 127.0.0.1
smtp_port: 25
```

Seguidament, crearem el fitxer **/etc/mailman3/mailman-hyperkitty.cfg** amb els següents paràmetres:

```
[general]
base_url: http://127.0.0.1:8000/archives/
api_key: Secret_Hyperkitty_API_Key
```

Configuració de l'MTA

Un agent de transferència de correu (MTA) és responsable d'enviar i rebre correus electrònics al servidor. Aquesta guia es basa en Postfix MTA, tot i que Mailman accepta d'altres com Exim4, etc..

- sudo dnf install postfix

Per configurar Postfix per retransmetre correus electrònics cap a i des de Mailman, modificarem i afegirem la següent configuració a l'arxiu **/etc/postfix/main.cf** :

```
inet_interfaces = all

recipient_delimiter = +
unknown_local_recipient_reject_code = 550
owner_request_special = no

transport_maps = hash:/opt/mailman/mm/var/data/postfix_lmtp
local_recipient_maps = hash:/opt/mailman/mm/var/data/postfix_lmtp
relay_domains = hash:/opt/mailman/mm/var/data/postfix_domains
```

Inici automàtic del Mailman

Per iniciar Mailman Core automàticament a l'arrencada, podeu configurar un servei a systemd. Crearem un fitxer nou al directori: **/etc/systemd/system/mailman3.service**

```
[Unit]
Description=GNU Mailing List Manager
After=syslog.target network.target postgresql.service

[Service]
Type=forking
PIDFile=/opt/mailman/mm/var/master.pid
User=mailman
Group=mailman
Environment="MAILMAN_CONFIG_FILE=/etc/mailman3/mailman.cfg"
ExecStart=/opt/mailman/venv/bin/mailman start
ExecReload=/opt/mailman/venv/bin/mailman restart
ExecStop=/opt/mailman/venv/bin/mailman stop
Restart=on-failure
RestartSec=15

[Install]
WantedBy=multi-user.target
```

Per carregar aquesta informació al sistema, executarem:

- sudo systemctl daemon-reload

Abans d'iniciar el servei de mailman correctament, hem d'obrir l'arxiu **/var/lib/pgsql/data/pg_hba.conf** i realitzar la següent configuració:

```
# Autenticación local

local   all             mailman                                 md5

# Autenticación de red

host    all             mailman        127.0.0.1/32             md5

host    all             mailman        ::1/128                  md5
```

Seguidamente reiniciem el servei de PostgreSQL amb:

- sudo systemctl restart postgresql

> **NOTA**
>
> S'ha d'executar la comanda **systemctl enable mailman3.service** abans de **daemon-reload** per aplicar la configuració del servei d'inici de forma correcta.

Verifiquem que el servei de Mailman Core està funcionant correctament:

- sudo systemctl start mailman3
- sudo systemctl status mailman3

Després d'això, fem login amb l'usuari mailman (virtualenv) i executem la comanda "**mailman info**", si està funcionant correctamente hauriem de veure la següent informació:

```
(venv) [mailman@testvm2 vagrant]$ mailman info
GNU Mailman 3.3.10 (Tom Sawyer)
Python 3.9.21 (main, Dec  5 2024, 00:00:00)
[GCC 11.5.0 20240719 (Red Hat 11.5.0-2)]
config file: /etc/mailman3/mailman.cfg
db url: postgresql://mailman:MYPASSWORD@localhost/mailman
devmode: DISABLED
REST root url: http://localhost:8001/3.1/
REST credentials: restadmin:restpass
```

Configurar feines CRON

Mailman Core requereix alguns treballs cron per a **accions periòdiques**. Per configurar feines cron per a Mailman executarem:

- sudo nano -u mailman crontab -e

Afegirem el següent a l'editor:

- **@daily** /opt/mailman/venv/bin/mailman digests --periodic
- **@daily** /opt/mailman/venv/bin/mailman notify

Això executarà les ordres a mitjanit cada dia (poder canviar-ho si volem).

Instal·lació d'Apache

Si no tenim instal·lat el servei d'Apache al nostre servidor, l'instal·larem amb la comanda:

- sudo dnf install httpd

Instal·lació i configuració de Django

Amb l'usuari **mailman** dins de l'entorn virtual instal·lem django amb:

- pip install django-mailman3

Ara crearem el directori "**mkdir ~/projectes**".

Una vegada realitzat l'anterior, crearem un **projecte de django** dins la carpeta "**/projectes**" amb la comanda:

- django-admin startproject mailman3web

Dintre de la carpeta del projecte "**mailman3web**" executem la següent comanda per **crear una aplicació** per al nostre projecte:

- python manage.py startapp mailman_web

Obrim l'arxiu del directori "**opt/mailman/projectes/mailman3web/mailman3web/settings.py**" y afegim el nom de l'aplicació "**mailman_web**" a l'apartat "**INSTALLED_APPS**" y les direccions IP a l'apartat "**ALLOWD_HOSTS** ("Ip_del_host","127.0.0.1", "localhost"):

```
ALLOWED_HOSTS = ['192.168.56.32','127.0.0.1','localhost']

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'mailman_web',
]
```

Una vegada realitzada la configuració, tornem al directori "**opt/mailman/projectes/mailman3web/**" on es troba l'arxiu "**manage.py**" y executarem les següents ordres:

- python manage.py migrate
- python manage.py runserver 0.0.0.0:8000

> **NOTA**

Per a que funcionin les conexions d'Apache (:80) i Django (:8000) hem de permitir el trànsit al firewalld amb "**sudo firewall-cmd --permanent --add-service=http**" y "**sudo firewall-cmd --permanent --add-port=8000/tcp**"

Instal·lació de la interfície d'usuari web

P**ostorius i Hyperkitty** són la interfície d'usuari web oficial de Mailman i Archiver. Mailman-web proporciona un paquet únic convenient per instal·lar tots dos paquests.

Per instal·lar els components web, executarem les ordres següents amb el virtualenv activat:

- (venv) $ pip install mailman-web mailman-hyperkitty

Configuració inicial

A continuació, crearem un fitxer de configuració a "**/etc/mailman3/settings.py**" amb els següents paràmetres:

```
# Mailman Web configuration file.
# /etc/mailman3/settings.py

# Get the default settings.
from mailman_web.settings.base import *
from mailman_web.settings.mailman import *

# Settings below supplement or override the defaults.

#: Default list of admins who receive the emails from error logging.
ADMINS = (
    ('Mailman Suite Admin', 'root@localhost'),
)

# Postgresql database setup.
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'mailmanweb',
        'USER': 'mailman',
        # TODO: Replace this with the password.
        'PASSWORD': 'MYPASSWORD',
        'HOST': 'localhost',
        'PORT': '5432',
```

```
        # For MySQL/MariaDB, in addition to changing the 'ENGINE' setting,
        # uncomment the following to enable utf8 4-byte encodings.
        # 'OPTIONS': {'charset': 'utf8mb4'},
    }
}

# 'collectstatic' command will copy all the static files here.
# Alias this location from your webserver to `/static`
STATIC_ROOT = '/opt/mailman/web/static'

# enable the 'compress' command.
COMPRESS_ENABLED = True

# Make sure that this directory is created or Django will fail on start.
LOGGING['handlers']['file']['filename'] = '/opt/mailman/web/logs/mailmanweb.log'

#: See https://docs.djangoproject.com/en/dev/ref/settings/#allowed-hosts
ALLOWED_HOSTS = [
    "localhost",  # Archiving API from Mailman, keep it.
    "127.0.0.1",
    "192.168.56.31",
    # "lists.your-domain.org",
    # Add here all production domains you have.
]

#: See https://docs.djangoproject.com/en/dev/ref/settings/#csrf-trusted-origins
#: For Django <4.0 these are of the form 'lists.example.com' or
#: '.example.com' to include subdomains and for Django >=4.0 they include
#: the scheme as in 'https://lists.example.com' or 'https://*.example.com'.
CSRF_TRUSTED_ORIGINS = [
    # "lists.your-domain.org",
    # Add here all production domains you have.
]

#: Current Django Site being served. This is used to customize the web host
#: being used to serve the current website. For more details about Django
#: site, see: https://docs.djangoproject.com/en/dev/ref/contrib/sites/
SITE_ID = 1

# Set this to a new secret value.
SECRET_KEY = 'MyVerrySecretKey'

# Set this to match the api_key setting in
```

```
# /opt/mailman/mm/mailman-hyperkitty.cfg (quoted here, not there).
MAILMAN_ARCHIVER_KEY = 'Secret_Hyperkitty_API_Key'

# The sender of emails from Django such as address confirmation requests.
# Set this to a valid email address.
DEFAULT_FROM_EMAIL = 'admin@example.com'

# The sender of error messages from Django. Set this to a valid email
# address.
SERVER_EMAIL = 'admin@example.com'

# add to /etc/mailman3/settings.py
HAYSTACK_CONNECTIONS = {
    'default': {
        'PATH': "/opt/mailman/web/xapian_index",
        'ENGINE': 'xapian_backend.XapianEngine'
    },
}
```

NOTA

Afegirem els dominis/adreces IP des dels quals vulguem servir Mailman web a l'apartat "**ALLOWED_HOSTS**"

NOTA2

Ens hem d'assegurar de crear el camí de registre dels logs, per això podem fer servir la comanda "**sudo mkdir -p /opt/mailman/web/logs**"

També crearem el directori del "**STATIC_ROOT**":

- *sudo mkdir -p /opt/mailman/web/static*
- *sudo chown -R mailman:mailman /opt/mailman/web/static*
- *sudo chmod -R 755 /opt/mailman/web/static*

Y per últim, serà necessari donar permisos a l'usuari **mailman** per modificar la carpeta "**/opt/mailman/web/logs:**"

- sudo chown -R mailman:mailman /opt/mailman/web/logs
- sudo chmod -R 755 /opt/mailman/web/logs

Executar migracions de bases de dades

Dins de l'entorn virtual de mailman, ens posicionarem a la carpeta del nostre projecte de Django "**~/projectes/mailman3web**" que és on està ubicat l'arxiu **manage.py** y executarem el següent:

- (venv)$ mailman-web migrate

Recollir fitxers estàtics

Per copiar tots els fitxers estàtics (css, js, imatges) a l'executació de la configuració inicial:

- (venv)$ mailman-web collectstatic

Comprimeix fitxers CSS

Per comprimir els diferents fitxers CSS fora de línia, executeu:

- (venv)$ mailman-web compress

Configuració de Gunicorn

Per utilitzar Gunicorn en lloc d'uWSGI no cal instal·lar Gunicorn ja que ja està instal·lat com a dependència del nucli de Mailman. Per tant, haurem de crear un fitxer de configuració per a Gunicorn a "**/etc/mailman3/gunicorn.conf.py**":

```
# /etc/mailman3/gunicorn.conf
#
bind = ['0.0.0.0:8000']
proc_name = "mailman-web"
chdir = "/opt/mailman/projectes/mailman3web"
pidfile = "/opt/mailman/projectes/mailman3web/var/gunicorn.pid"
accesslog = "/opt/mailman/projectes/mailman3web/var/logs/access.log"
errorlog = "/opt/mailman/projectes/mailman3web/var/logs/error.log"
```

Haurem de crear el directori "**/opt/mailman/projectes/mailman3web/var/logs/**":

- sudo mkdir -p /opt/mailman/projectes/mailman3web/var/logs/

Y li donarem drets a l'usuari mailman:

- sudo chown -R mailman:mailman /opt/mailman/projectes/mailman3web/var/logs/
- sudo chmod -R 755 /opt/mailman/projectes/mailman3web/var/logs/

Ara crearem l'arxiu "**error.log**" i farem propietari a l'usuari mailman:

- sudo touch /opt/mailman/projectes/mailman3web/var/logs/error.log
- sudo chown mailman:mailman /opt/mailman/projectes/mailman3web/var/logs/error.log

Corretgim permisos per al directori /var:

- sudo chown -R mailman:mailman /opt/mailman/projectes/mailman3web/var/
- sudo chmod -R 755 /opt/mailman/projectes/mailman3web/var/

Y finalment donem propietat i drets per al directori /home/vagrant a l'usuari mailman:

- sudo chown -R mailman:mailman /home/vagrant
- sudo chmod -R 755 /home/vagrant

> **NOTA**
>
> Si volem executar manualment Gunicorn per comprovar que funciona correctament, executarem la comanda "**gunicorn --config /etc/mailman3/gunicorn.conf.py mailman3web.wsgi:application**" desde l'entorn virtual.

Configuració de la cerca de text complet (xapian)

Instal·larem python3-xapian al sistema:

- dnf install -y python3-xapian

Després executarem:

- sudo ln -s /usr/lib/python3.9/site-packages/xapian /opt/mailman/venv/lib/python3.9/site-packages/

Y per últim:

- /opt/mailman/venv/bin/python -c "import xapian"

Ya tindriem Xapian instal·lat al nsotre entorn virtual.

Inici automàtic de Mailman-Web

Per tal d'iniciar-se automàticament a l'inici, podem crear un segon servei a systemd al directori "**mailmanweb.service/etc/systemd/system/mailmanweb.service**" amb la següent configuració:

```
[Unit]
Description=GNU Mailman Web UI
After=syslog.target network.target postgresql.service mailman3.service
```

```
[Service]
Environment="PYTHONPATH=/etc/mailman3/"
User=mailman
Group=mailman
ExecStart=/opt/mailman/venv/bin/gunicorn -c /etc/mailman3/gunicorn.conf.py
mailman3web.wsgi:application
KillSignal=SIGINT

[Install]
WantedBy=multi-user.target
```

Després de crear aquest fitxer, executarem les ordres següents:

- sudo systemctl daemon-reload
- sudo systemctl start mailmanweb

A continuació, comprovearem l'estat del servei xecutant:

- systemctl status mailmanweb

Configuració d'Apache

A continuació es mostra la configuració per a fer funcionar mailmanweb a través del servidor d'Apache:

```
<VirtualHost *:80>
    ServerAdmin jalsina.q@ccma.cat
    DocumentRoot /var/www/html

    Alias /static "/opt/mailman/web/static"
    <Directory "/opt/mailman/web/static">
        Require all granted
    </Directory>

    <IfModule mod_headers.c>
        RequestHeader unset X-Forwarded-Proto
        <If "%{HTTPS} =~ /on/">
            RequestHeader set X-Forwarded-Proto "https"
        </If>
    </IfModule>
```

```
    <IfModule mod_proxy.c>
        ProxyPreserveHost On
        ProxyPass /mailman3 http://127.0.0.1:8000/mailman3
        ProxyPassReverse /mailman3 http://127.0.0.1:8000/mailman3

        ProxyPass /archives http://127.0.0.1:8000/archives
        ProxyPassReverse /archives http://127.0.0.1:8000/archives

        ProxyPass /accounts http://127.0.0.1:8000/accounts
        ProxyPassReverse /accounts http://127.0.0.1:8000/accounts

        ProxyPass /admin http://127.0.0.1:8000/admin
        ProxyPassReverse /admin http://127.0.0.1:8000/admin

        ProxyPass /user-profile http://127.0.0.1:8000/user-profile
        ProxyPassReverse /user-profile http://127.0.0.1:8000/user-profile
    </IfModule>

    ErrorLog /var/log/httpd/mailman3-error.log
    CustomLog /var/log/httpd/mailman3-access.log combined
</VirtualHost>
```

Configuració arxiu settings.py y urls.py de Django (postorius/hyperkitty)

Per tal que funcionin les adreces de mailmanweb al port :8000, hem de configurar l'arxiu settings.py i urls.py del directori "**/opt/mailman/projectes/mailman3web/mailman3web** " amb els següents paràmetres:

settings.py

```
"""
Django settings for mailman3web project.

Generated by 'django-admin startproject' using Django 4.2.19.

For more information on this file, see
https://docs.djangoproject.com/en/4.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.2/ref/settings/
"""
```

```
import os
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent



# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-768!0z826!ma_&g=qdyk0oi5lse#)57cbbh*w3(-(j#6os_q_i'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['192.168.56.31','127.0.0.1','localhost']



# Application definition
SITE_ID = 1

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.humanize',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.sites',
    'django_mailman3',
    'allauth',                  # Add this if it's missing
    'allauth.account',          # Add this if it's missing
    'allauth.socialaccount',    # Add this if you're using social login
    # Other apps...
# apps propies

   'mailman_web.apps.MailmanWebConfig',
   'postorius',
   'hyperkitty',
   'django_extensions',
   'django_mailman3.lib.auth.fedora',
```

```
    'compressor',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.locale.LocaleMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'allauth.account.middleware.AccountMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'django_mailman3.middleware.TimezoneMiddleware',
    'postorius.middleware.PostoriusMiddleware',

]

ROOT_URLCONF = 'mailman3web.urls'

COMPRESS_ENABLED = True
COMPRESS_OFFLINE = True
COMPRESS_URL = '/static/'
COMPRESS_ROOT = os.path.join(BASE_DIR, 'static')

STATICFILES_FINDERS = [
    'django.contrib.staticfiles.finders.FileSystemFinder',
    'django.contrib.staticfiles.finders.AppDirectoriesFinder',
    'compressor.finders.CompressorFinder',  # Agrega esta línea
]


TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.i18n',
                'django.template.context_processors.media',
```

```
                'django.template.context_processors.static',
                'django.template.context_processors.tz',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'django_mailman3.context_processors.common',
                'postorius.context_processors.postorius',

            ],
        },
    },
]

WSGI_APPLICATION = 'mailman3web.wsgi.application'

# Agregar la configuración de búsqueda de Haystack
HAYSTACK_CONNECTIONS = {
    'default': {
        'ENGINE': 'haystack.backends.whoosh_backend.WhooshEngine',
        'PATH': os.path.join(BASE_DIR, 'whoosh_index'),  # Asegúrate de tener
este directorio
    },
}



# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    'default': {
    'ENGINE': 'django.db.backends.postgresql_psycopg2',
    'NAME': 'mailmanweb',
    'USER': 'mailman',
    'PASSWORD': 'MYPASSWORD',
    'HOST': 'localhost',
}
}



# Password validation
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
```

```
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]




LOGIN_URL = 'account_login'
LOGIN_REDIRECT_URL = 'list_index'
LOGOUT_URL = 'account_logout'

# Internationalization
# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC_URL = 'static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static'),  # Otras ubicaciones de archivos estáticos
```

```
que quieras agregar
]

# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

MAILMAN_REST_API_URL = 'http://localhost:8001'  # Asegúrate de que esta URL esté configurada correctamente.
MAILMAN_REST_API_USER = 'restadmin'  # Reemplaza 'tu_usuario' por el nombre de usuario correcto
MAILMAN_REST_API_PASS = 'restpass'  # Reemplaza 'tu_contraseña' por la contraseña correcta


AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
    'allauth.account.auth_backends.AuthenticationBackend',
)
```

urls.py

```
"""
URL configuration for mailman3web project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    3. Add an import:  from other_app.views import Home
    4. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    5. Import the include() function: from django.urls import include, path
    6. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
```

```
from django.contrib import admin
from django.urls import path
from django.conf.urls import include
from django.urls import re_path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('postorius/', include('postorius.urls')),
    path('mailman3/', include('postorius.urls')),
    path('accounts/', include('allauth.urls')),
    path('hyperkitty/', include('hyperkitty.urls')),

    re_path(r'', include('django_mailman3.urls')),
    re_path(r'^accounts/', include('allauth.urls')),
    re_path(r'', include('django.contrib.auth.urls')),

]
```

SuperUsuari de Django

Per poder fer login com a admin a la interfície web, haurem de crear un usuari administrador o "superuser" a django per poder accedir y modificar la base de dades:

NOTA

(venv)$ mailman-web createsuperuser

NOTA2

S'ha de permetre el trànsit del port 25 a través del firewalld de la máquina virtual per poder rebre emails desde la nostre màquina física amb "sudo firewall-cmd --permanent --add-port=25/tcp"