Atlas of Innovation: Overview

MakerNet.work

ATLAS OF INNOVATION

NATION OF MAKERS

Contents

List of Figures

List of Tables

1 Technologies

Atlas of innovation is writting in Python language using the Django framework in the 2.0.1 version, other modules are installed automatically using PIP as show in itemize 1, this packages are defined in a requirements.txt file, aditionally the instalation of tlsh package and postgresql DB is perfom in comandline as defined in the vagrantfile, the path of the files are described in the Folder organization section .

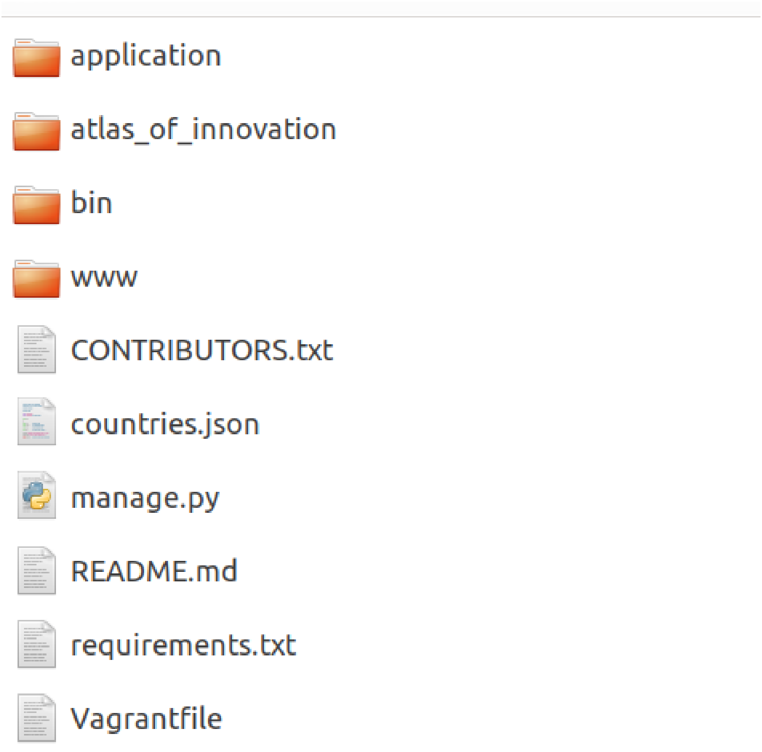
- certifi==2018.1.18
- chardet==3.0.4
- Django==2.0.1
- django-countries==5.1.1
- django-recaptcha==1.4.0
- djangorestframework==3.7.7
- idna==2.6
- psycopg2==2.7.3.2
- pytz==2017.3
- urllib3==1.22
- django-post_office==3.0.4
- tlsh

As is well knowed django uses MVT(model-view-template) instead the clasic MVC(model-view-controler) pattern,"The Model-View-Template (MVT) is slightly different from MVC. In fact the main difference between the two patterns is that Django itself takes care of the Controller part (Software Code that controls the interactions between the Model and View), leaving us with the template. The template is a HTML file mixed with Django Template

Language (DTL)" [1]. The views are the place where we code the logic of the app,It will ask for information of the model and will pass it to the template

2 Folder organization

This section aims to explain the organization of project folders, so that the developer can identify the files that can be modified to add or modify some system functionality, when you download from github and install the project using vagrant,the root folder project are named atlas_of_inovation, inside we could find the folders and files as show in the image 1.



application
atlas_of_innovation
bin
www
CONTRIBUTORS.txt
countries.json
manage.py
README.md
requirements.txt
Vagrantfile

Figure 1: Main folder

2.1 Installation and configuration files

The installation of all the environment for development is performed using the files

- **Vagrantfile**.
- **requirements.txt**
- **box_setup.sh** inside bin folder

- **base.py, development.py and production.py** inside **atlas_of_innovation/settings** folder.

The basic installations requirements and configurations are specified inside this files.

2.1.1 VM configuration: Vagrantfile and setup.sh

Atlas of innovation runs in vagrant Virtual Machine(VM),the installation and configuration of this VM is in charge of vagrantfile and setup.sh.

The vagrant file describes the vm installation,the primary function of the Vagrantfile is to describe the type of machine required for a project, and how to configure and provision these machines.The syntax of Vagrantfiles is Ruby, but knowledge of the Ruby programming language is not necessary to make modifications to the Vagrantfile, since it is mostly simple variable assignment. [2]

The setup.sh file is part of the vagrantfile, is the provision shell script that execute all the comands for provide the box all necesary software for atlas aplication,Provisioners in Vagrant allow you to automatically install software, alter configurations, and more on the machine as part of the vagrant up process.

The setup.sh includes the django framework, python and dependences installation.

2.2 Atlas configuration files

The packages necessaries and configurations are especified in the files: requirements.txt, base.py, development.py and production.py.

2.2.1 Packages required : requirements.txt

Atlas of innovation has a set of dependencies that are required for that application to work. The requirements file is a way to specify and install specific set of package dependencies at once. The package name and version is specified inside this file, including django, and then these are installed during provision task of vm automatically.

2.2.2 Django configurations: base.py, development.py and production.py.

This files are used to configure the django project including

INSTALLED_APPS A list of strings designating all applications that are enabled in this Django installation.

MIDDLEWARE Middleware is a framework of hooks into Django's request/response processing. It's a light, low-level «plugin» system for globally altering Django's input or output.

Database configuration etc.

this files can include many others configuration items, see full list in Django website

2.3 Atlasproject: code and data

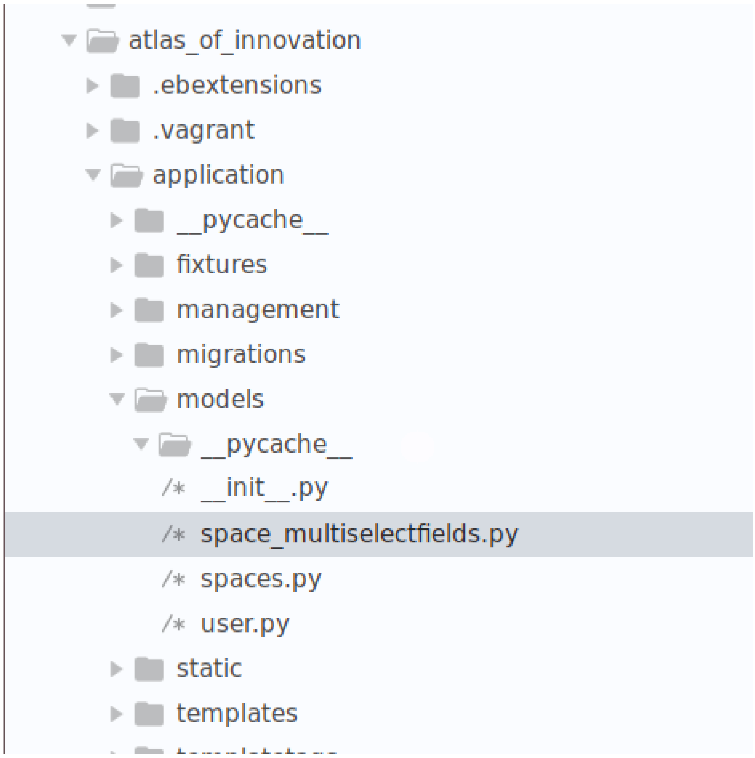
the code and data for atlas are stored in application folder, atlas_of_innovation folder and contributors.txt and countries.json files.

2.3.1 models views and templates

The models path is **application/models**, in the spaces.py, user.py and space_multiselectedfields.py files, as show in figure 2. aditionally the admin model is located in **application/admin.py**
As the name of each file describes the spaces.py contain all models related to spaces, user.py all user models and space_multiselectedfields.py the models of multiselected field

The views are located in **application/views**,as show in figure 4, there you can find the function and classes used to manage the logic of the page, including user signup, send mails, and all the logic for space/provisional_spaces edit, create, upload, analisis, and csv importer.

finally the Templates path are **application/templates** and **atlas_of_innovation/templates**, in the last could find the templates related to admin page



```
▼ atlas_of_innovation
  ▶ .ebextensions
  ▶ .vagrant
  ▼ application
    ▶ __pycache__
    ▶ fixtures
    ▶ management
    ▶ migrations
    ▼ models
      ▼ __pycache__
        /* __init__.py
        /* space_multiselectfields.py
        /* spaces.py
        /* user.py
    ▶ static
    ▶ templates
```

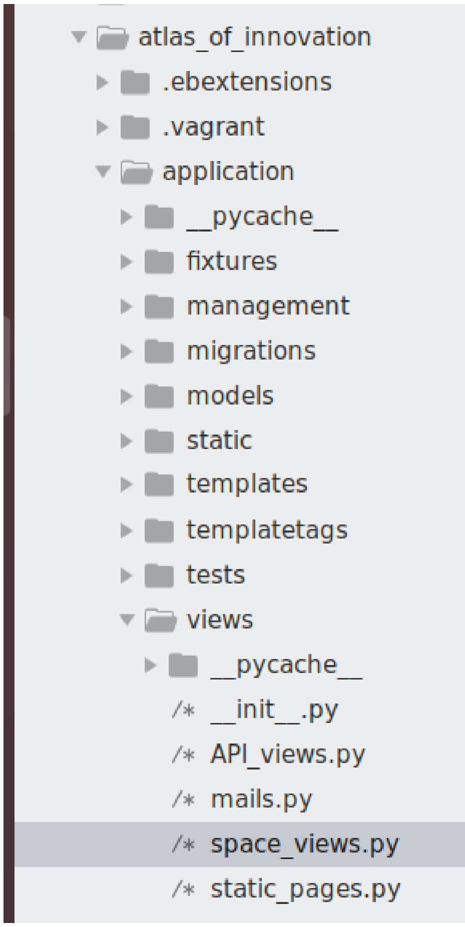
Figure 2: Models folder

2.4 Static Files

inside the Static folders can find the images, css and javascript files used in atlas project as show in figure 5 5, the files used by interactive map are:leafted.css,makercluster.css,makercluster-default.css,leafted.js,makercluster,css

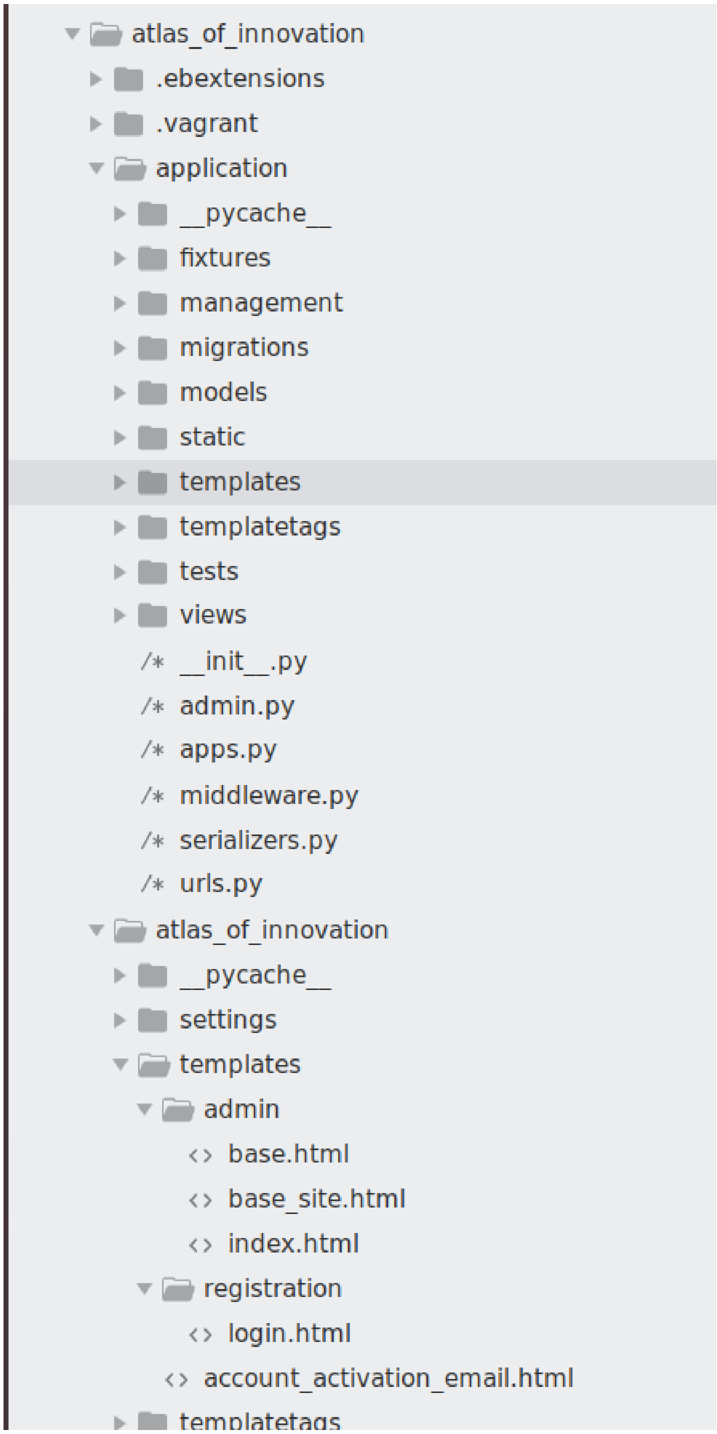
the files used by list section are:wikipage.css,wiki.js and wikilist.js

finally the general css and js are stylesheet.css and page.js



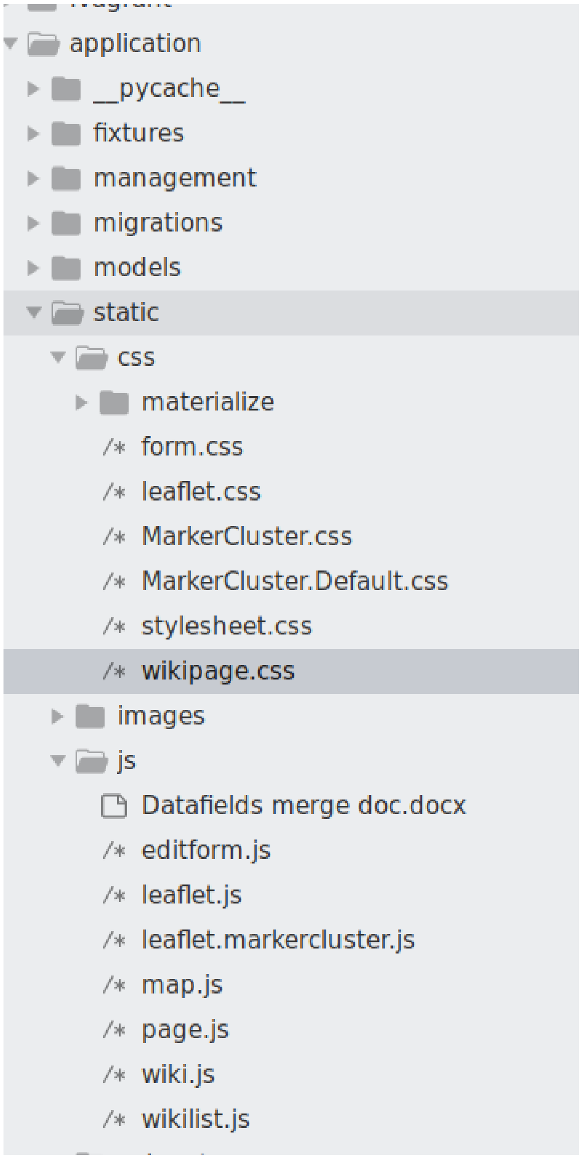
```
atlas_of_innovation
  .ebextensions
  .vagrant
  application
    __pycache__
    fixtures
    management
    migrations
    models
    static
    templates
    templatetags
    tests
    views
      __pycache__
      __init__.py
      API_views.py
      mails.py
      space_views.py
      static_pages.py
```

Figure 3: views folder



```
▾ atlas_of_innovation
  ▸ .ebextensions
  ▸ .vagrant
  ▾ application
    ▸ __pycache__
    ▸ fixtures
    ▸ management
    ▸ migrations
    ▸ models
    ▸ static
    ▸ templates
    ▸ templatetags
    ▸ tests
    ▸ views
      __init__.py
      admin.py
      apps.py
      middleware.py
      serializers.py
      urls.py
  ▾ atlas_of_innovation
    ▸ __pycache__
    ▸ settings
    ▾ templates
      ▾ admin
          base.html
          base_site.html
          index.html
      ▾ registration
          login.html
        account_activation_email.html
    ▸ templatetags
```

Figure 4: views folder

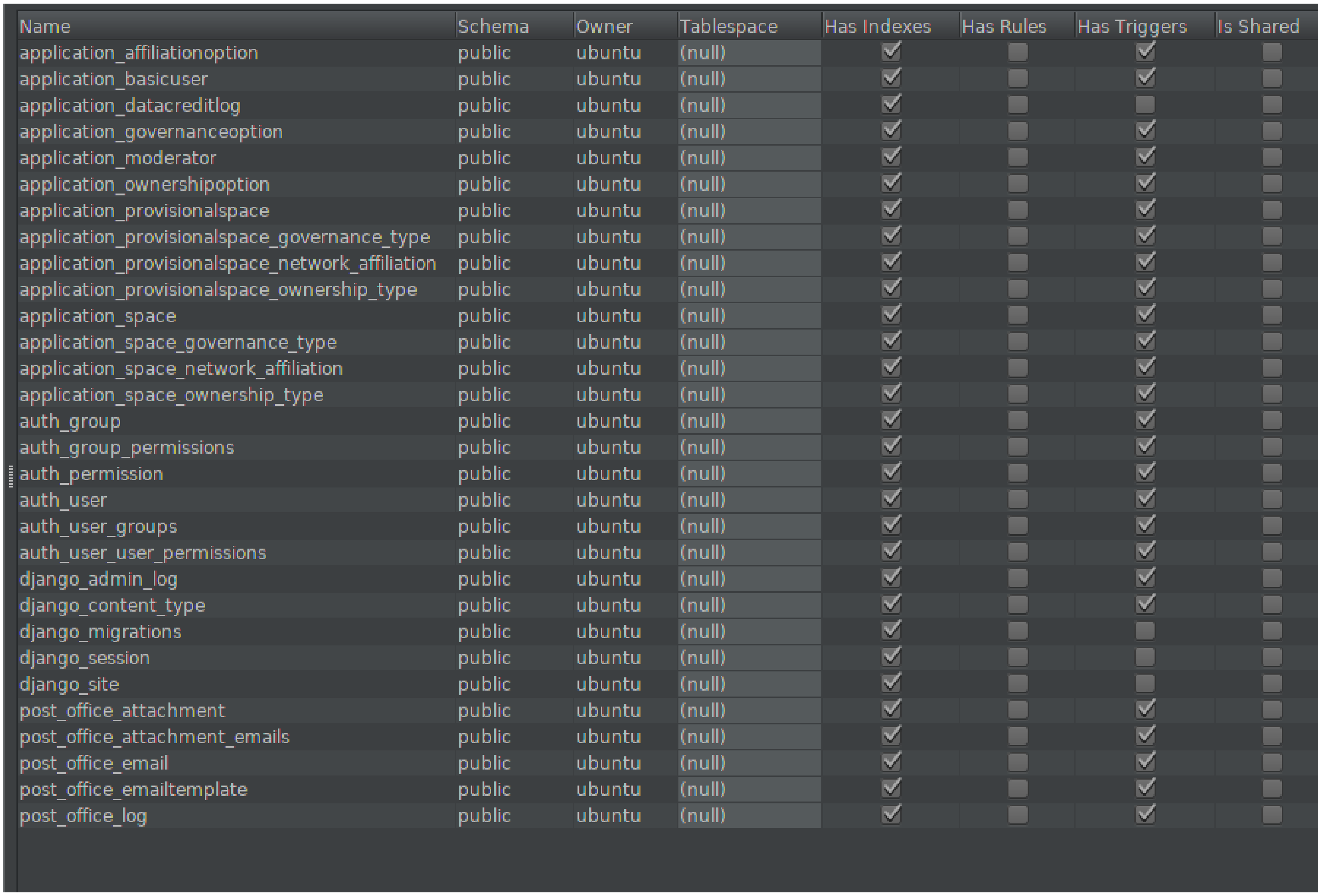


```
.vagrant
application
    __pycache__
    fixtures
    management
    migrations
    models
    static
        css
            materialize
            /* form.css
            /* leaflet.css
            /* MarkerCluster.css
            /* MarkerCluster.Default.css
            /* stylesheet.css
            /* wikipage.css
        images
        js
            Datafields merge doc.docx
            /* editform.js
            /* leaflet.js
            /* leaflet.markercluster.js
            /* map.js
            /* page.js
            /* wiki.js
            /* wikilist.js
```

Figure 5: static folder

3 Database

The db used in atlas is prosgresql, the list of tlables and the schema are showed in figures 6, 8, 7 and 9



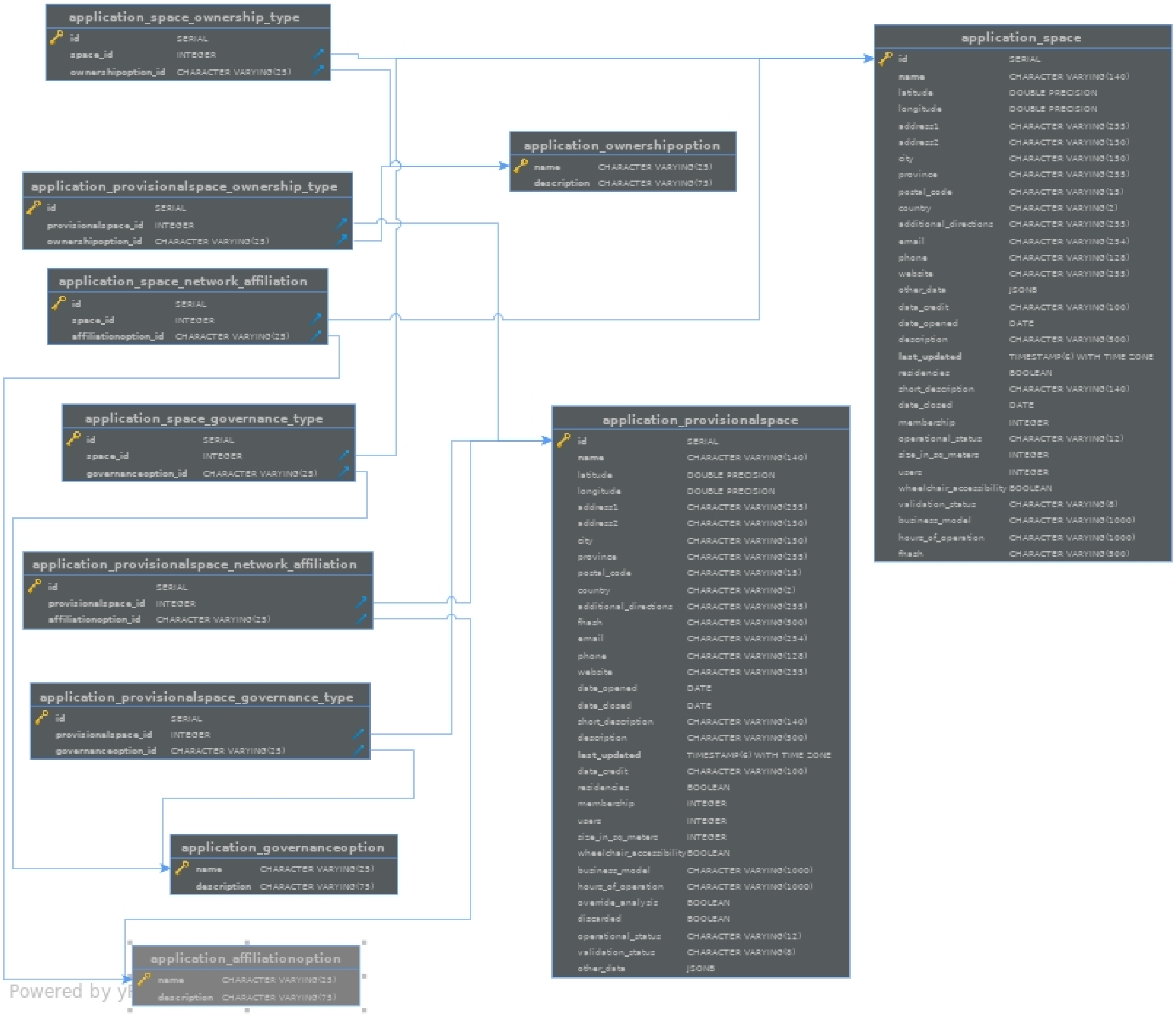
Name	Schema	Owner	Tablespace	Has Indexes	Has Rules	Has Triggers	Is Shared
application_affiliationoption	public	ubuntu	(null)	☑	□	☑	□
application_basicuser	public	ubuntu	(null)	☑	□	☑	□
application_datacreditlog	public	ubuntu	(null)	☑	□	□	□
application_governanceoption	public	ubuntu	(null)	☑	□	☑	□
application_moderator	public	ubuntu	(null)	☑	□	☑	□
application_ownershipoption	public	ubuntu	(null)	☑	□	☑	□
application_provisionalspace	public	ubuntu	(null)	☑	□	☑	□
application_provisionalspace_governance_type	public	ubuntu	(null)	☑	□	☑	□
application_provisionalspace_network_affiliation	public	ubuntu	(null)	☑	□	☑	□
application_provisionalspace_ownership_type	public	ubuntu	(null)	☑	□	☑	□
application_space	public	ubuntu	(null)	☑	□	☑	□
application_space_governance_type	public	ubuntu	(null)	☑	□	☑	□
application_space_network_affiliation	public	ubuntu	(null)	☑	□	☑	□
application_space_ownership_type	public	ubuntu	(null)	☑	□	☑	□
auth_group	public	ubuntu	(null)	☑	□	☑	□
auth_group_permissions	public	ubuntu	(null)	☑	□	☑	□
auth_permission	public	ubuntu	(null)	☑	□	☑	□
auth_user	public	ubuntu	(null)	☑	□	☑	□
auth_user_groups	public	ubuntu	(null)	☑	□	☑	□
auth_user_user_permissions	public	ubuntu	(null)	☑	□	☑	□
django_admin_log	public	ubuntu	(null)	☑	□	☑	□
django_content_type	public	ubuntu	(null)	☑	□	☑	□
django_migrations	public	ubuntu	(null)	☑	□	□	□
django_session	public	ubuntu	(null)	☑	□	□	□
django_site	public	ubuntu	(null)	☑	□	□	□
post_office_attachment	public	ubuntu	(null)	☑	□	☑	□
post_office_attachment_emails	public	ubuntu	(null)	☑	□	☑	□
post_office_email	public	ubuntu	(null)	☑	□	☑	□
post_office_emailtemplate	public	ubuntu	(null)	☑	□	☑	□
post_office_log	public	ubuntu	(null)	☑	□	☑	□

Figure 6: Database tables

Exist 2 identical tables, spaces and provisional spaces, because when import new spaces from csv the new spaces first are stored as provisional spaces until the analisis and moderator determines that space is valid or invalid.

Atlas have 3 usertypes, administer, moderator and basic user, basic user only can add a space, moderator can analize and impor spaces from csv and administer have all privileges on page.

Actually use postoffice to manage the mails templates languages etc



application_space_ownership_type

id	SERIAL
space_id	INTEGER
ownershipoption_id	CHARACTER VARYING(25)

application_provisionalspace_ownership_type

id	SERIAL
provisionalspace_id	INTEGER
ownershipoption_id	CHARACTER VARYING(25)

application_space_network_affiliation

id	SERIAL
space_id	INTEGER
affiliationoption_id	CHARACTER VARYING(25)

application_space_governance_type

id	SERIAL
space_id	INTEGER
governanceoption_id	CHARACTER VARYING(25)

application_provisionalspace_network_affiliation

id	SERIAL
provisionalspace_id	INTEGER
affiliationoption_id	CHARACTER VARYING(25)

application_provisionalspace_governance_type

id	SERIAL
provisionalspace_id	INTEGER
governanceoption_id	CHARACTER VARYING(25)

application_governanceoption

name	CHARACTER VARYING(25)
description	CHARACTER VARYING(75)

application_affiliationoption

name	CHARACTER VARYING(25)
description	CHARACTER VARYING(75)

application_ownershipoption

name	CHARACTER VARYING(25)
description	CHARACTER VARYING(75)

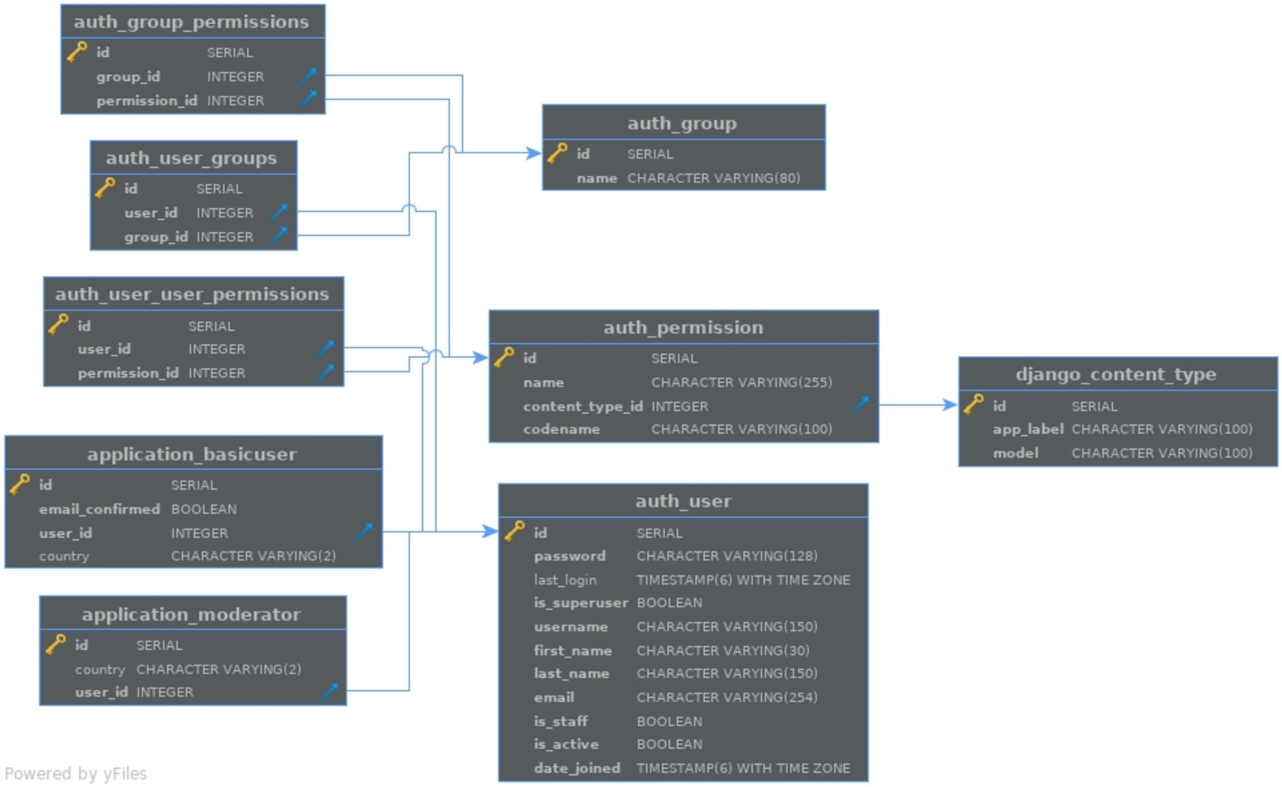
application_provisionalspace

id	SERIAL
name	CHARACTER VARYING(140)
latitude	DOUBLE PRECISION
longitude	DOUBLE PRECISION
address1	CHARACTER VARYING(255)
address2	CHARACTER VARYING(150)
city	CHARACTER VARYING(150)
province	CHARACTER VARYING(255)
postal_code	CHARACTER VARYING(15)
country	CHARACTER VARYING(2)
additional_directions	CHARACTER VARYING(255)
fhash	CHARACTER VARYING(300)
email	CHARACTER VARYING(254)
phone	CHARACTER VARYING(128)
website	CHARACTER VARYING(255)
date_opened	DATE
date_closed	DATE
short_description	CHARACTER VARYING(140)
description	CHARACTER VARYING(300)
last_updated	TIMESTAMP(6) WITH TIME ZONE
data_credit	CHARACTER VARYING(100)
residencies	BOOLEAN
membership	INTEGER
users	INTEGER
size_in_sq_meters	INTEGER
wheelchair_accessibility	BOOLEAN
business_model	CHARACTER VARYING(1000)
hours_of_operation	CHARACTER VARYING(1000)
override_analysis	BOOLEAN
discarded	BOOLEAN
operational_status	CHARACTER VARYING(12)
validation_status	CHARACTER VARYING(8)
other_data	JSONB

application_space

id	SERIAL
name	CHARACTER VARYING(140)
latitude	DOUBLE PRECISION
longitude	DOUBLE PRECISION
address1	CHARACTER VARYING(255)
address2	CHARACTER VARYING(150)
city	CHARACTER VARYING(150)
province	CHARACTER VARYING(255)
postal_code	CHARACTER VARYING(15)
country	CHARACTER VARYING(2)
additional_directions	CHARACTER VARYING(255)
email	CHARACTER VARYING(254)
phone	CHARACTER VARYING(128)
website	CHARACTER VARYING(255)
other_data	JSONB
data_credit	CHARACTER VARYING(100)
date_opened	DATE
description	CHARACTER VARYING(300)
last_updated	TIMESTAMP(6) WITH TIME ZONE
residencies	BOOLEAN
short_description	CHARACTER VARYING(140)
date_closed	DATE
membership	INTEGER
operational_status	CHARACTER VARYING(12)
size_in_sq_meters	INTEGER
users	INTEGER
wheelchair_accessibility	BOOLEAN
validation_status	CHARACTER VARYING(8)
business_model	CHARACTER VARYING(1000)
hours_of_operation	CHARACTER VARYING(1000)
fhash	CHARACTER VARYING(300)

Figure 7: spaces schema



auth_group_permissions

id	SERIAL	
group_id	INTEGER	↗
permission_id	INTEGER	↗

auth_user_groups

id	SERIAL	
user_id	INTEGER	↗
group_id	INTEGER	↗

auth_user_user_permissions

id	SERIAL	
user_id	INTEGER	↗
permission_id	INTEGER	↗

application_basicuser

id	SERIAL	
email_confirmed	BOOLEAN	
user_id	INTEGER	↗
country	CHARACTER VARYING(2)	

application_moderator

id	SERIAL	
country	CHARACTER VARYING(2)	
user_id	INTEGER	↗

auth_group

id	SERIAL
name	CHARACTER VARYING(80)

auth_permission

id	SERIAL	
name	CHARACTER VARYING(255)	
content_type_id	INTEGER	↗
codename	CHARACTER VARYING(100)	

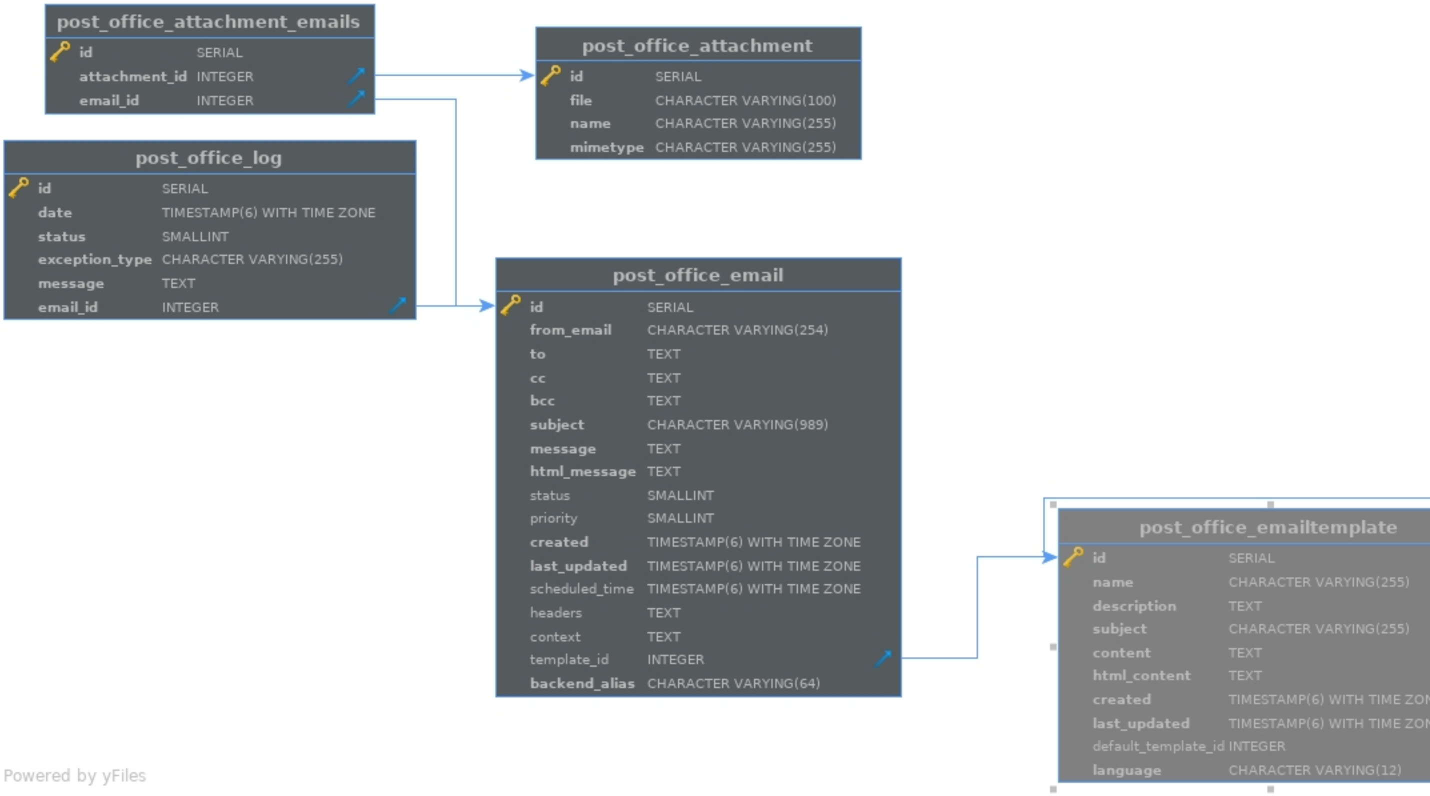
django_content_type

id	SERIAL
app_label	CHARACTER VARYING(100)
model	CHARACTER VARYING(100)

auth_user

id	SERIAL
password	CHARACTER VARYING(128)
last_login	TIMESTAMP(6) WITH TIME ZONE
is_superuser	BOOLEAN
username	CHARACTER VARYING(150)
first_name	CHARACTER VARYING(30)
last_name	CHARACTER VARYING(150)
email	CHARACTER VARYING(254)
is_staff	BOOLEAN
is_active	BOOLEAN
date_joined	TIMESTAMP(6) WITH TIME ZONE

Figure 8: users schema



post_office_attachment_emails	
id	SERIAL
attachment_id	INTEGER
email_id	INTEGER

post_office_attachment	
id	SERIAL
file	CHARACTER VARYING(100)
name	CHARACTER VARYING(255)
mimetype	CHARACTER VARYING(255)

post_office_log	
id	SERIAL
date	TIMESTAMP(6) WITH TIME ZONE
status	SMALLINT
exception_type	CHARACTER VARYING(255)
message	TEXT
email_id	INTEGER

post_office_email	
id	SERIAL
from_email	CHARACTER VARYING(254)
to	TEXT
cc	TEXT
bcc	TEXT
subject	CHARACTER VARYING(989)
message	TEXT
html_message	TEXT
status	SMALLINT
priority	SMALLINT
created	TIMESTAMP(6) WITH TIME ZONE
last_updated	TIMESTAMP(6) WITH TIME ZONE
scheduled_time	TIMESTAMP(6) WITH TIME ZONE
headers	TEXT
context	TEXT
template_id	INTEGER
backend_alias	CHARACTER VARYING(64)

post_office_emailtemplate	
id	SERIAL
name	CHARACTER VARYING(255)
description	TEXT
subject	CHARACTER VARYING(255)
content	TEXT
html_content	TEXT
created	TIMESTAMP(6) WITH TIME ZOI
last_updated	TIMESTAMP(6) WITH TIME ZOI
default_template_id	INTEGER
language	CHARACTER VARYING(12)

Figure 9: post office schema

4 References

References

[1] DJANGO SITE *https://www.tutorialspoint.com/django/django_ overview.htm*

[2] Vagrant site *https://www.vagrantup.com/docs/vagrantfile/*