### /

Co-Add Development Guide

(Draft)

Contents

[- 0 -](#_Toc143504656)

[INTRODUCTION - 4 -](#_Toc143504657)

[Overview - 4 -](#_Toc143504658)

[adjcoadd - 6 -](#_Toc143504659)

[Static - 6 -](#_Toc143504660)

[Quick Start with new module - 8 -](#_Toc143504661)

[New schema and database Router - 8 -](#_Toc143504662)

[Start with New Application - 8 -](#_Toc143504663)

[Building - 8 -](#_Toc143504664)

[apputil - 13 -](#_Toc143504665)

[apputil folder tree - 13 -](#_Toc143504666)

[Models - AuditModel - 14 -](#_Toc143504667)

[Models - ApplicationUser - 15 -](#_Toc143504668)

[Models - Dictionary - 17 -](#_Toc143504669)

[Document - 20 -](#_Toc143504670)

[Middle Ware - 20 -](#_Toc143504671)

[Signals - 21 -](#_Toc143504672)

[Utils in Apputil - 21 -](#_Toc143504673)

[ddrug - 22 -](#_Toc143504674)

[ddrug tree structure - 22 -](#_Toc143504675)

[Drug - 23 -](#_Toc143504676)

[Ketcher Structure drawing and Update sMol - 25 -](#_Toc143504677)

[dgene - 25 -](#_Toc143504678)

[dorganism - 26 -](#_Toc143504679)

[dorganism application folder tree - 26 -](#_Toc143504680)

[Taxonomy - 27 -](#_Toc143504681)

[Organism - 29 -](#_Toc143504682)

[Model List - 32 -](#_Toc143504683)

[Database Schema - 34 -](#_Toc143504684)

[Python Utils - 35 -](#_Toc143504685)

[apputil - 35 -](#_Toc143504686)

[dcollab - 39 -](#_Toc143504687)

[ddrug - 39 -](#_Toc143504688)

[dgene - 39 -](#_Toc143504689)

[dorganism - 39 -](#_Toc143504690)

[dscreen - 40 -](#_Toc143504691)

[Template Tags - 40 -](#_Toc143504692)

[Custom\_filters: - 40 -](#_Toc143504693)

[Pagination: - 41 -](#_Toc143504694)

[Application Templates - 41 -](#_Toc143504695)

[templates/Base - 41 -](#_Toc143504696)

[apputil/templates - 41 -](#_Toc143504697)

[dcollab/templates - 43 -](#_Toc143504698)

[ddrug/templates - 43 -](#_Toc143504699)

[ddrug/ - 43 -](#_Toc143504700)

[ddrug - 46 -](#_Toc143504701)

[dgene/templates - 47 -](#_Toc143504702)

[dgene - 47 -](#_Toc143504703)

[dorganism/templates - 49 -](#_Toc143504704)

[dorganism - 49 -](#_Toc143504705)

[dscreen - 50 -](#_Toc143504706)

[Append Templates Utils - 50 -](#_Toc143504707)

[1. card.html - 50 -](#_Toc143504708)

[2. data\_download\_form.html - 51 -](#_Toc143504709)

[3. datatable\_filter.html - 51 -](#_Toc143504710)

[4. datatable\_general.html - 52 -](#_Toc143504711)

[5. datatable\_sm.html - 53 -](#_Toc143504712)

[6. django\_form\_lgtable.html - 53 -](#_Toc143504713)

[7. ketcher.html - 53 -](#_Toc143504714)

[8. leftbar.html - 54 -](#_Toc143504715)

[9. lefticon.html - 54 -](#_Toc143504716)

[10. main\_horizalbar.html - 54 -](#_Toc143504717)

[11. message.html - 54 -](#_Toc143504718)

[12. messagebox.html - 55 -](#_Toc143504719)

[13. navbar.html - 55 -](#_Toc143504720)

[14. pagination.html - 55 -](#_Toc143504721)

[15. row\_editabletable.html - 55 -](#_Toc143504722)

[16. scripts.html - 55 -](#_Toc143504723)

[17. search\_organism\_id.html - 55 -](#_Toc143504724)

[18. search\_organism.html - 56 -](#_Toc143504725)

[19. sidebar.html - 56 -](#_Toc143504726)

[20. wizard\_single\_page.html - 56 -](#_Toc143504727)

[21. drag\_n\_drop - 56 -](#_Toc143504728)

[22. miscellaneous - 56 -](#_Toc143504729)

[23. modal - 57 -](#_Toc143504730)

[JavaScript Functions - 57 -](#_Toc143504731)

[js\_utils - 57 -](#_Toc143504732)

[ketcher - 57 -](#_Toc143504733)

[childtable - 58 -](#_Toc143504734)

[destroy\_childtable - 58 -](#_Toc143504735)

[create\_childtable - 58 -](#_Toc143504736)

[editableTable - 58 -](#_Toc143504737)

[send\_organism / send\_organism\_id - 59 -](#_Toc143504738)

[sorting\_table - 59 -](#_Toc143504739)

[Project Structure Tree - 60 -](#_Toc143504740)

## INTRODUCTION

The Development Guide is designed to display each application’s structures, functions and with code implementation example. It has 4 sections: section 1 overview is about main functions, project directories and applications inside the project, including CSS style change, JavaScript usages, data handling generally…; section 2 introduces how to quick start implement a new module; section 3 is about how to implement and add codes in each application mainly for adding CRUD views; section 4 is appended list of functions, models, code overview and explanations.

## Overview

adjCOADD is a database-driven web application, which contains 6 main applications: apputil, dcollab, ddrug, dgene, dorganism, dscreen. The main functions are providing database CRUD views, screening data from OrgDB database and on data visualization.

Based on the database postgresql, each application represents one schema with the same name (refer to [Append Database Schema](#_Database_Schema)), and in general involves following parts:

* **Model**s: for Object Relational Mapping (ORM) to a database ([Append Model List](#_Append_Model_List))
* **Templates** folder: Htmls containing variables/context passing from the Views.
* **Views**: python functions/classes containing business logic to take web requests and return web response, like html documents.
* utils folder: functions / classes called by views and models to provide a base view, forms, logs, base logical control functions, data type convertor, storage / path definition…
* urls: application urls wrapped by project urls and contains links in templates HTML to call corresponded views function/classes.
* template tags folder: a package contains python functions as filter (convert datatype, retrieve data…) or urls pattern and will be applied as customized template tag in HTML.

The project overview tree structure is:

.

├── MANIFEST.in (used to package useful modules make pypi)

├── README.rst (used for project and usage introduction)

├── **adjcoadd**

│   ├── \_\_init\_\_.py

│   ├── asgi.py

│   ├── constants.py

│   ├── routers.py

│   ├── settings.py

│   ├── urls.py

│   └── wsgi.py

├── **applog**

│   ├── django\_server.log

│   └── user.log

├── **apputil**

│   ├── \_\_init\_\_.py

│   └── …

├── **dcollab**

│   ├── \_\_init\_\_.py

│   └── …

├── **ddrug**

│   ├── \_\_init\_\_.py

│   └── …

├── deploy\_gunicorn.txt

├── **dgene**

│   ├── \_\_init\_\_.py

│   └── …

├── **dorganism**

│   ├── \_\_init\_\_.py

│   └── …

├── **dscreen**

│   ├── \_\_init\_\_.py

│   └── …

├── environment.yml (create development environment)

├── environment\_prod.yml (create product environment)

├── impdata (import functions module)

│   ├── Data

│   │

│   └── …

├── manage.py (Django command manager)

├── requirements.txt (required python packages with pip manager)

├── setup.cfg (used to package useful modules make pypi)

├── setup.py (used to package useful modules make pypi)

├── **static (overall used css, js, images)**

├── **templates (base template and util templates used project overall)**

├── **tests (teat cases for all applications)**

├── **z\_accessAPI (example to use python script access api data tables (drug, VITEK\_AST))**

├── **z\_design (project design)**

│   └── adjOrgDB\_Schema.pptx

└── z\_docs (project related documentations)

In the tree structure, the folder **adjcoadd** is a Django project core folder with general settings (asgi.py, settings.py, urls.py and wsgi.py) and extra settings (constants.py and routers.py). The constants contain project widely used constants and links while routers are used to put application into assigned migrating database. Folder **static** and **templates** contains HTMLs and JavaScript functions project widely used during development.

And in the following application sections will introduce each application with:

* displaying the models’ relationship in database way in entity relationship diagram (ERD)- an overview of models with the primary keys and foreign keys inside application
* the application’s sub tree structure – overview of folders and scripts in the application
* Then description each model in the application based on Model – View - Template:
  + MVT diagram to display Model, View, Template and list function names, classes and urls…
  + Code implementation examples.

There are two types of code implementation. 1) code implementation for base classes or functions will on developing code in the application apputil, which used for other applications inside the project. They are generally python script or python packages; 2) The second type of development is based on apputil create database CRUD views and specific function related to one application. These views will generally inherited base classes from apputil / utils with

-> create model based on auditmodel in <application> / models.py

-> create view class based on views\_base (e.g., class GeneCreateView(SimplecreateView) based on class SimplecreateView(LoginRequiredMixin, View)) in <application> / views.py to rend template with context.

-> create corresponded url address in application / urls.py, for calling view class to render and send response to html page at the address.

-> create the corresponded template in <application> / templates / <application> / <model> / <model template>.html. It usually including several html templates in the utils / templates in base templates folder.

In the section Append displaying a list of overviews of models, util functions, templates, javascript functions, css class and folders’ tree structure.

### adjcoadd

It is Django core folder:

url – project level urls are admin, authentication and including the application level urls;

constant – the constants and links can be used in any python script inside project.

routers – python script for application models migration with assigned data schema;

settings – contains project settings for django general settings, database, backend, middlewares… for extra authentication Ldap configurations, and website logs path.

### Static

Static folder contains css files, javascripts and images.

In templates html element has style classes from bootstrap 5 and css files in static folder overall. To be specific:

* The class property for each HTML element contains class- names in the order: “[layout class] [space class] [position class] [text-color] [text-size] [border][background color][special effect] [function class name]”.

Layout class (e.g., col or row or d-flex), space class (e.g., m-auto or p-auto), position class (justify-content-between), border (border-0…), text-color (text-info…), text-size, and background color are based on bootstrap 5.

* In case to change bootstrap font-size and customize a font-size for html elements, go to overideboot\_size.css to override font size, font weight values etc.,
* In custom.css mainly defined special effects, functional classes, and customized colors to override bootstraps colors which listed on top lines of the file:
* @import url('https://fonts.googleapis.com/css2?family=Open+Sans:wght@300&family=Roboto&display=swap');
* /\* functional classes \*/
* /\*
* 1 Login Form
* 2 Navbar
* …
* key up search input
* \*/
* /\*  Special Effects \*/
* /\*
* (sorting table duble arrows effect
* …\*/
* /\* Theme Color \*/
* /\*
* How to use theme colors:
* var(--body-bg): body background, input background,
* …

Also for general used html element styles: a, p, input, select, span… Within this file could implement changes on theme colours and functional classes (except tables), just go to the top line -> copy name -> use find function to find the code block’s location.

* There is separated css file to override table style from boostrap and define functional classes for tables – table\_wrap.css, with functional name listed on the top lines:
* /\* This css only for all kinds of tables \*/
* /\*
* 1. general table style
* 2. partial render datatable in detail view
* 3. fixed table header
* 4. pivottable style
* 5. Draggable and Dropable and sortable
* \*/

Changes on table’s class will be here.

* The file “main.6a646761.css” is only used for ketcher chemical draw – embedded application canvas.

The list of css inherited by templates in the order(with override priority high to low): table\_warp.css -> main.6a646761.css -> custom.css -> overideboot\_size.css -> bootstrap, jquery others…(cdn links).

JavaScript files in Static are referred to append [JavaScript functions](#_Append_JavaScript_Functions). JavaScript will used to

Images are used for css style and object displaying like image model and drug models

## Quick Start with new module

To add a simple new module(application including templates) just following these steps:

* New schema and database Router script configuration
* startapp <new application>
* build data models and views based on utils.py in apputil, config urls and create templates.

### New schema and database Router

Add new application to routers.py in core app(adjcoadd):

route\_app\_labels = {

'default': {'auth', 'contenttypes','apputil'},

...,

# here add new

# 'schema':{'newapplication'}

}

In settings.py database config, adding:

DATABASES = {

#...,

'new schema': {

# config as above.

},

},

### Start with New Application

manage.py startapp <new>

### Building

#### Add new model in models.py:

#Import third package

#Import django build in package

#Import coadd package

from adjcoadd.constants import \*

from apputil.models import AuditModel, Dictionary

class newmodel(AuditModel):

#here create attribute: HEADER\_FIELDS, CARD\_FIELDS, FORM\_GROUPS and others

#here model fields

#model meta

#model methods

#### New modelform, filterform in forms.py:

#Import third package

#Import django build in package

#Import coadd package

from apputil.models import Dictionary, ApplicationUser, **NewModel**

from apputil.utils.filters\_base import Filterbase

class newmodel\_form(forms.ModelForm):

#here formfields

#form init method and other methods

#form meta

class newmodel\_filter(Filterbase):

#here filterfields

#form init method and other methods

#form meta

#### Add view classes in views.py:

#Import third package

#Import django build in package

#Import coadd package

from adjcoadd.constants import \*

from apputil.utils.filters\_base import FilteredListView

from apputil.utils.views\_base import SimplecreateView, SimpleupdateView,HtmxupdateView, SimpledeleteView

from .models import \*

from .forms import \*

# list view

class Modellistview(FilteredListView):

#here define view attribute:

#login\_url, model, template\_name, filterset\_class, model\_fields, app\_name, model\_name and others

#or customize view methods:

#get\_queryset, get\_context\_data, get\_paginate\_by, get\_order\_by

#detailview

def modeldetail(request, pk):

# set and return render context in template

# create view

class Modelcreateview(SimplecreateView):

#here define view attribute: form\_class, template\_name, transaction\_use

#or customize view methods

# update view

class Modelupdateview(SimpleupdateView or HtmxupdateView):

#here define view attribute: form\_class, template\_name,template\_partial(if use HtmxupdateView), model, transaction\_use

#or customize view methods

# delete view

class Modeldeleteview(SimpledeleteView):

#here define view attribute: model, transaction\_use

#or customize view methods

#### Config urls in urls.py:

in core(adjcoadd)/urls.py

path('<new application>/', include('<new application.urls>')),

in new application/urls.py

path('<modelname\_list>/', Viewclass.as\_view(), name="modelname\_list"),

path('<modelname\_detail>/<str:pk>', viewfunction, name="modelname\_detail"),

path('<createmodel>/', Viewclass.as\_view(), name="...create"),

path('<update...>/<str:pk>', Viewclass.as\_view(), name="...update"),

path('<delete...>/<str:pk>', Viewclass.as\_view(), name="...delete"),

#### Create template

in new application/templates/new application/model create:

* model\_c.html includes ‘utils/modal/modal\_form.html’ and {% url ‘model\_create’ as createurl %}
* model\_u.html includes update url, form with including {%include ‘utils/row\_editabletable.html’ with form\_label=field.label data\_type=field.label form\_field=field %}
* model\_d.html includes:
  + {%include ‘utils/miscellaneous/delete\_btn.html’ with target=’#model\_detail\_del’%},
  + {% include ‘utils/modal/delete.html’ with title=’Delete Organism’ deleteurl=deleteurl entry=object %} with delete url
* model\_list.html includes:
  + {% url ‘pivoted-table’ app\_model=’applicationname-Modelname’ as url\_pivotedtable%}
  + {% include ‘utils/leftbar.html’ with list\_url=’model\_list’ card\_url=’model\_card’ create\_object=’createModelname’ create\_objectModal=’createModelnameModal’ url\_pivotedtable=url\_pivotedtable %}
  + {% include “utils/sidebar.html” %}
  + {% include ‘utils/main\_horizalbar.html’ with title=’Modelname’ model\_name=’Modelname’ application=’applicationname’ %}
  + {% include ‘utils/datatable\_general.html’ with modelname=’Modelname’ %}
* model\_detail.html includes:
  + {% include ‘utils/lefticons.html’ with title=”Model” %}
  + {% include ‘utils/message.html’ %}
  + {% include ‘./model\_u.html’ %}
* Add function line to templates/miscellaneous/create\_a.html:
  + loadModal("#createNewModelModal", "#createNewModel", "{% url newmodel\_create'%}");

Create Template Link:

Add http link to new model view in “templates/utils/navbar.html” and “apputil/templates/home.html”.

## apputil

Model entities in application apputil are displaying in the following database ERD:

Diagram

Description automatically generated

Figure apputil ERD

### apputil folder tree

├── apputil

│   ├── \_\_init\_\_.py

│   ├── admin.py

│   ├── api\_filterclass.py

│   ├── apps.py

│   ├── clear\_session\_middleware.py

│   ├── forms.py

│   ├── migrations

│   ├── models.py

│   ├── signals.py

│   ├── templates

│   │   ├── apputil

│   │   │   ├── appUserProfile.html

│   │   │   ├── appUsers.html

│   │   │   ├── appUsersCreate.html

│   │   │   ├── appUsersDel.html

│   │   │   ├── appUsersUpdate.html

│   │   │   ├── appuser\_tr.html

│   │   │   ├── async.html

│   │   │   ├── dictCreate.html

│   │   │   ├── dictList.html

│   │   │   ├── dictionary\_tr.html

│   │   │   └── importhandler\_excel.html

│   │   ├── home.html

│   │   └── registration

│   │   └── login.html

│   ├── templatetags

│   │   ├── \_\_init\_\_.py

│   │   ├── custom\_filters.py

│   │   └── myapp\_extras.py

│   ├── urls.py

│   ├── utils

│   │   ├── api\_filterclass.py

│   │   ├── data.py

│   │   ├── data\_style.py

│   │   ├── files\_upload.py

│   │   ├── filters\_base.py

│   │   ├── flex\_pivottable.py

│   │   ├── form\_wizard\_tools.py

│   │   ├── validation\_log.py

│   │   └── views\_base.py

│   └── views.py

In the tree structure including:

* admin.py: register models to admin page
* api\_filterclass.py: contain API list view class as a base class to create drug and vitek-ast API view.
* apps.py
* clear\_session\_middleware.py: contains functions control project level request and response to clear session storage.
* forms.py: contains forms for create / update entries in database and filterset forms for each model in dorganism.
* migrations
* models.py contains models ([refer to models](#_Append_Model_List)list).
* signals.py contains functions to record applications response, errors, catch exceptions, user login in logs for analyse and debug once application running on the server.
* templates, the templates in application for each model are generally including the shared templates in utils. Each model has own template folder with this model’s CRUD templates HTML files inside. Their functions are passing model variables to these templates.
* Templatetags contains two scripts: 1) custom\_filter.py including a list of filter functions for working on context result via python functions in templates which used in project level’; 2) pagination\_url.py to keep filtered result consistence with changed page numbers.
* urls.py contains urls for each view class / function.
* utils, utils in apputil are shared functions and classes overall inside project (refer to [apputils](#_Part_I_Utilities))
* views.py contains views functions and classes: (refer to [model’s mvt diagram](#_ApplicationUser_Diagram))

### Models - AuditModel

The AuditModel is an abstract model, which been inherited by other data model except ApplicationUsers. The main functions it is used for are:

* Displaying objects status in database and objects with visible status (>=0) for users at the front. This will be controlled by the class attributes:

DELETED = -9; INVALID = -1; UNDEFINED = 0; VALID = 1; CONFIRMED = 2

and model field – astatus.

* Define default username “orgdb” and Tracking objects changes with modelfields and instance methods: acreated\_at / aupdated\_at / adeleted at / acreated / aupdated / adeleted; save and delete.
* The instance methods to group and regroup objects fields displaying at the front with HEADER\_FIELDS (table view and pivote view) and CARDS\_FIELDS (card view), are:
  + get\_fields,
  + iter\_foreignkey,
  + get\_values
  + get\_fieldsandvalues
* The instance method to return unique column values (query in distinct) to make a selecting options list, is:
  + get\_field\_choices

- VALID\_STATUS = False

- ID\_SEQUENCE = ""

- ID\_PREFIX = ""

- ID\_PAD = 0 (??)

### Models - ApplicationUser

#### ApplicationUser Diagram



Figure ApplicationUser-MVT diagram

#### Code Implementation

##### Create Model

Firstly, create model class – class ApplicationUser(AbstractUser) in <application>/models.py



Then in the views.py build view class or function to process **workflow** (CRUD) between database and UI.

##### CRUD View Class

The crud Views are inheirated from [utils/view\_base and filters\_base](#_filters_base):

* Create View: class AppUserCreateView(SimplecreateView)
* Read View: class AppUserListView(LoginRequiredMixin, FilteredListView); With detail view class: class AppUserDetailView(DetailView):
* Update View: class ApplicationUserUpdateView(HtmxupdateView)
* Delete View: class AppUserDeleteView(SuperUserRequiredMixin, UpdateView). Instead of remove an entry physically from the database, delete view is used to update field astatus(for ApplicationUser is\_appuser field to False) to -9, filtering by queryset filter and leaving invisible on UIs.

##### Correspondent URLs are:

* path('user\_list/', AppUserListView.as\_view(), name="userslist"),
* path('user\_create/', AppUserCreateView.as\_view(), name="createAppUser"),
* path('user\_update/<str:pk>', ApplicationUserUpdateView.as\_view(), name="updateAppUser"),
* path('user\_delete/<str:pk>', AppUserDeleteView.as\_view(), name="deleteAppUser"),
* path('user\_profile/<str:pk>', AppUserDetailView.as\_view(), name='userprofile' ),

The views and urls build connection between UI and database operations.

##### Templates Rendering

UI or the templates rendering includes main template with included partial templates(htmls).

* **For the Read View – overview(list)**

The main template is …/apputil/templates/apputil/[appUsers.html](#_./appUsers.html), which includs : ‘[utils/leftbar.html](#_leftbar.html)’, [‘utils/sidebar.html’](#_sidebar.html), [‘utils/main\_horizalbar.html’](#_Append_3_Templates), [‘utils/datatable\_general.html’](#_datatable_general.html) and Javascripts included in ‘utils/script.html’

With the URl in navbar menu/user section button(Only show when the user is admin role) with link {% url “userslist” %} (codes: path('user\_list/', AppUserListView.as\_view(), name="userslist"), ) leads to the view page.

* **For the Create View**

The corresponded template’[appUsersCreate.html](#_./appUsersCreate.html)’ is a modal html as the following:

It includes the partial html ‘[utils/modal/modal\_form.html’](#_modal_form.html). And as modal-dialog called by JS function [loadModal](#_Append_JavaScript_Functions). With the URI in a partial html ‘[create\_a.html’](#_create_a.html), wrapped by another partial html – ‘[utils/leftbar.html](#_leftbar.html)’ leads to the View.

* **For the Update View**

The template update view template is a partial template “[apputil/templates/apputil/appUsersUpdate.html](#_./appUsersUpdate.html)”, which contains htmx attributes: hx-put, hx-include, hx-trigger.

URl links are in hx-put attribute.

* **For the Delete View**

The template for the delete view class is a partial modal template – [appUsersDel.html](#_./appUsersDel.html) included in appuer\_tr.html. The delete view template used bootstrap default modal form template. In the partial template self also includes utils template ‘['utils/miscellaneous/delete\_btn.html'’](#_Append_3_Templates) and ['utils/modal/delete.html'](#_Append_3_Templates). The first one is a standard delete button and the second one is deleting modal form submitting the “delete” request.

* **Difference between ajax called modal form and Bootstrap triggered modal form by default.**

The two types of modal-forms are both wrapped by a parent html code blocks (for example generaltable.html or leftbar.html). The main difference between the two types is that using external ajax the modal-form will be able to render at new urls with the form from the view function while the default modal is triggered within the URL, which used to render the wrapper html and the view function at this URL might or might not contain the required form.

calling delete modal template with bootstrap default modal call. Compared to create entry modal, delete modal is include on the template appuser\_tr.html not rendered on a separated URL address ‘delete URL’, as the following:

Create user (a modal body with create form rendered at create URL):

Text

Description automatically generated with medium confidence

Delete user (a modal body inside the same list url with blank form point to delete URL):

Diagram

Description automatically generated

So, there is no delete form body rendered on the modal but the delete URL will be included in form action attribute which can perform the post request.

### Models - Dictionary

#### Dictionary diagram

Diagram

Description automatically generated

Figure Dictionary MVT diagram

#### Code Implementation

The code implementation approach is generally the same referred to [Application User](#_Code_Implementation) Model with Create Model -> View ->Template

##### Create Model

##### CRUD View Class:

The create and read Views are inheirated from [utils/view\_base and filters\_base](#_filters_base):

* Create View: class DictionaryCreateView(SuperUserRequiredMixin, SimplecreateView):
* Read View: class DictionaryView(LoginRequiredMixin, FilteredListView):
* Update View is a function-based view not an inherited view

The difference is Dictionary primary key contain characters not suitable for URL names and as it simply only 4 fields model, update and delete use jQuery ajax call return a JsonResponse not rendering the whole template.



* Delete View as well is a function based view:

Correspondent URLs are:

* path('dict/', DictionaryView.as\_view(), name='dict\_view' ),
* path('dict\_create/', DictionaryCreateView.as\_view(), name='dict\_create' ),
* path('dict\_update/', updateDictionary, name='dict\_update' ),
* path('dict\_delete/', deleteDictionary, name='dict\_delete' )

##### Templates Rendering

UI or the templates rendering includes main template with included partial templates(htmls).

- For the Read View – overview(list)

The main template is …/apputil/templates/apputil/dictList.html including dictionary\_tr.html

- For the Create View

The corresponded template’dictCreate.html’ is a modal html included by another partial html – ‘utils/leftbar.html’

- For the Update View

Updating in dictionary\_tr.html included by main template, called update function by ajax.

- For the Delete View

Delete as above.

### Models - Document

#### Document diagram

#### Code Implementation

##### Create Model

Models are inherited from Audit Model

##### Upload / Delete View Class:

Image and document have related model organism or organism\_batch which is in application dorganism, where the displaying(read)/upload/ delete views will be implemented. Displaying will as a related foreign key field in organism entry only have upload view and delete View here.

* Upload View: image and document will uploaded to directory uploads/images or /documents via class CreateimageView(CreateFileView) and class CreatedocumentView(CreateFileView) inherited [from CreateFileView](#_views_base) in [utils/view\_base](#_views_base)
* Delete View inherited from SimpledeleteView in [utils/view\_base](#_views_base)

Correspondent URLs are:

* path('img/<str:pk>', CreateimageView.as\_view(), name="addimg"),
* path('doc/<str:pk>', CreatedocumentView.as\_view(), name="adddoc"),
* path('img-delete/<str:pk>', ImageDeleteView.as\_view(), name='org\_img\_delete'),
* path('doc-delete/<str:pk>', DocDeleteView.as\_view(), name='org\_doc\_delete'),

##### Templates Rendering

- refer to [organism detail view](#_Organism)

### Forms

Forms script functions are similar in each application, including:

* Create filter forms for Django\_filter (List View and Card View). Filter forms are inherited from the class Filterbase or Filterbase\_base(only for model ApplicationUser) in apputil/utils/filters\_base.py
* Create django modelforms for update/create view class and the forms will be submitted to change records from UI.

### Middle Ware

There is one customized middle ware used in project: class ClearSessionMiddleware installed in settings.py “MIDDLEWARE” block with code line: 'apputil.clear\_session\_middleware.ClearSessionMiddleware'.

The function of this middle ware is to clear session storage for cached queryset.

For each datatable searching and filtering, results stored in request.session with the key datable\_modelname\_cached\_queryset. They will be used to generate pivot table with selected results. They need to be clear up when switching to another view or leaving current data model view This middel ware used to check view name per each request and perform the two logic:

1. request changed to another data model, delete stored cached queryset.

2. request stay in current view or process data from current view, NOT delete.

### Signals

Used to track app user login, logout or log in errors. The log will be record in project\_root/applog/user.log. And log format configured in settings.py.

### Utils in Apputil

Refer to appended [python util](#_Part_I_Utilities)

## ddrug

Application ERD

A picture containing text, screenshot, font, number

Description automatically generated

Drug application has 7 models. Among them, the drugs’CRUD views functions are inherited from view classes in view\_base in apputil/utils. Other models don’t have create, update and delete view, changes will be made directly via data import.

### ddrug tree structure

├── ddrug

│   ├── \_\_init\_\_.py

│   ├── admin.py

│   ├── apps.py

│   ├── forms.py

│   ├── migrations

│   ├── models.py

│   ├── serializers.py

│   ├── templates

│   │   └── ddrug

│   │   ├── breakpoint

│   │   │   └── breakpoint\_list.html

│   │   ├── drug

│   │   │   ├── drug\_c.html

│   │   │   ├── drug\_card.html

│   │   │   ├── drug\_d.html

│   │   │   ├── drug\_detail.html

│   │   │   ├── drug\_detail\_structure.html

│   │   │   ├── drug\_list.html

│   │   │   └── drug\_u.html

│   │   ├── importhandler\_drug.html

│   │   ├── importhandler\_vitek.html

│   │   ├── mic\_coadd

│   │   │   ├── mic\_coadd\_card.html

│   │   │   └── mic\_coadd\_list.html

│   │   ├── mic\_pub

│   │   │   ├── mic\_pub\_card.html

│   │   │   └── mic\_pub\_list.html

│   │   ├── vitek\_ast

│   │   │   └── vitekast\_list.html

│   │   ├── vitek\_card

│   │   │   ├── vitekcard\_detail.html

│   │   │   └── vitekcard\_list.html

│   │   └── vitek\_id

│   │   └── vitekid\_list.html

│   ├── tests\_api.py

│   ├── upload\_views.py

│   ├── urls.py

│   ├── utils

│   │   ├── bio\_analysis.py

│   │   ├── bio\_data.py

│   │   ├── bio\_updates.py

│   │   ├── import\_drug.py

│   │   ├── molecules.py

│   │   ├── tables.py

│   │   └── vitek.py

│   └── views.py

ddrug tree structure contains:

* admin.py: register models to admin page
* apps.py
* forms.py: contains forms for create / update entries in database and filterset forms for each model in ddrug.
* migrations
* models.py contains models ([refer to models](#_Append_Model_List)list).
* serializer.py: serializer forms for drug and vitekast APIs.
* templates, the templates in application for each model are generally including the shared templates in utils. Their functions are passing model variables to these templates.
* tests\_api.py: test functions for test api site accessibility
* upload\_view.py: separated view class only for uploads View classes for models: Drug, Vitekcard/ID/Ast
* urls.py contains urls for each view class / function.
* utils, utils in apputil are shared functions and classes overall inside project (refer to [ddrug utils](#_ddrug))
* views.py contains views functions and classes: (refer to  [model’s mvt diagram](#_ApplicationUser_Diagram))

### Drug

#### Drug Diagram

#### Code Implementation

Create Model

Firstly, create model class – class Drug (AuditModel) in drug / models.py

Then in the views.py build view class or function to process **workflow** (CRUD) between database and UI.

CRUD View Class

The crud Views are inheirated from [utils/view\_base and filters\_base](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_filters_base):

* Create View: class AppUserCreateView(SimplecreateView)
* Read View: class AppUserListView(LoginRequiredMixin, FilteredListView); With detail view class: class AppUserDetailView(DetailView):
* Update View: class ApplicationUserUpdateView(HtmxupdateView)
* Delete View: class AppUserDeleteView(SuperUserRequiredMixin, UpdateView). Instead of remove an entry physically from the database, delete view is used to update field astatus(for ApplicationUser is\_appuser field to False) to -9, filtering by queryset filter and leaving invisible on UIs.

Correspondent URLs are:

* path('user\_list/', AppUserListView.as\_view(), name="userslist"),
* path('user\_create/', AppUserCreateView.as\_view(), name="createAppUser"),
* path('user\_update/<str:pk>', ApplicationUserUpdateView.as\_view(), name="updateAppUser"),
* path('user\_delete/<str:pk>', AppUserDeleteView.as\_view(), name="deleteAppUser"),
* path('user\_profile/<str:pk>', AppUserDetailView.as\_view(), name='userprofile' ),

The views and urls build connection between UI and database operations.

Templates Rendering

UI or the templates rendering includes main template with included partial templates(htmls).

* **For the Read View – overview(list)**

The main template is …/apputil/templates/apputil/[appUsers.html](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_./appUsers.html), which includs : ‘[utils/leftbar.html](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_leftbar.html)’, [‘utils/sidebar.html’](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_sidebar.html), [‘utils/main\_horizalbar.html’](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_Append_3_Templates), [‘utils/datatable\_general.html’](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_datatable_general.html) and Javascripts included in ‘utils/script.html’

* **For the Create View**

The corresponded template’[appUsersCreate.html](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_./appUsersCreate.html)’ is a modal html as the following:

It includes the partial html ‘[utils/modal/modal\_form.html’](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_modal_form.html). And as modal-dialog called by JS function [loadModal](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_Append_JavaScript_Functions). The call trigger is a partial html ‘[create\_a.html’](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_create_a.html), wrapped by another partial html – ‘[utils/leftbar.html](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_leftbar.html)’

* **For the Update View**

The template update view template is a partial template “[apputil/templates/apputil/appUsersUpdate.html](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_./appUsersUpdate.html)”, which contains htmx attributes: hx-put, hx-include, hx-trigger.

### Ketcher Structure drawing and Update sMol

Implementation ketcher application via static/js/ketcher/index-bundle.min.js.

Index-bundle.min.js is compiled js file from React-Django-Ketcher (on the server "I:\DEEPMICROB-Q3967\Code\Python\Django\React-Django-Ketcher" ) with Ketcher standalone React packages.

In version 1.1 implemented drug structure editor Ketcher on each drug detail view, which used to update drug’s mol field and get query structures.

#### Update smol field

* Select or create new drug to detail view
* Click drug image, guiding to ketcher editor section
* Editing structure inside the canvas, then click getsmile button to get smile or click get mol button to see molblock
* Once smile updated in textarea, click update mol button on the top of the canvas, open update form and click save button
* Smol field should be updated.

## dgene

(To display data)

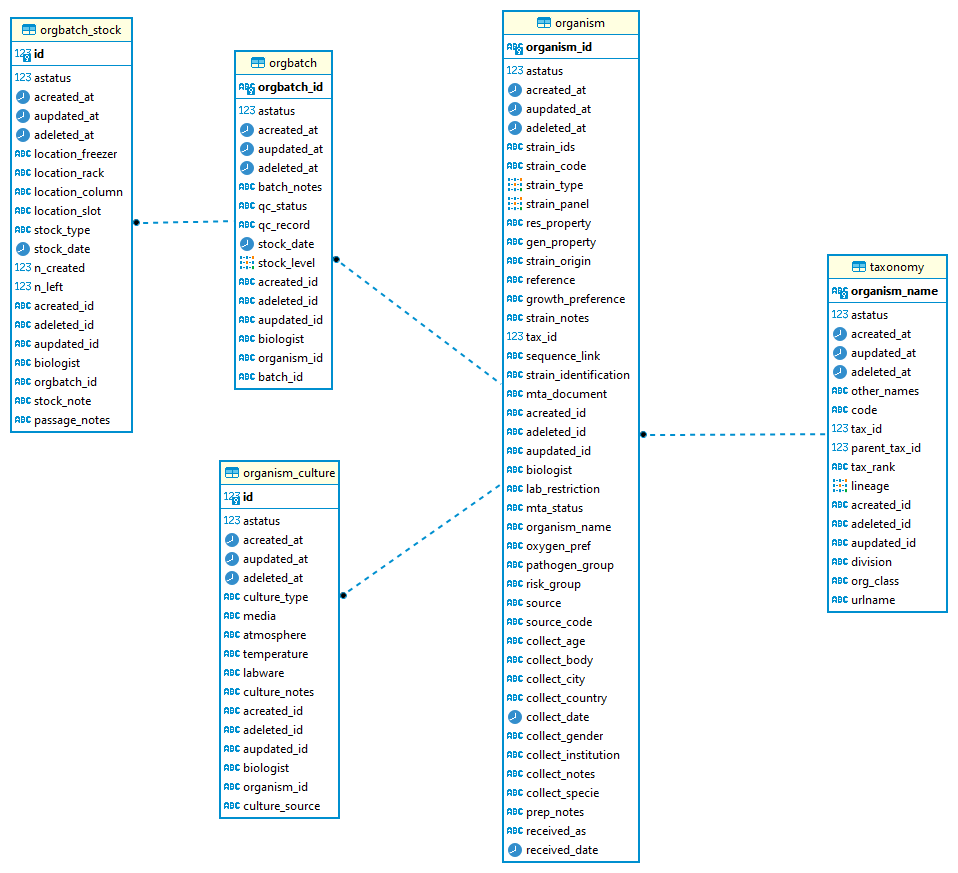
A screenshot of a computer

Description automatically generated with medium confidence

dgene application has 5 models. Among them, the Model Gene’s CRUD views functions are inherited from view classes in view\_base in apputil/utils. Other models’ changes in database will be made directly via data import and they all have table list view class inherited from view\_base as well.

## dorganism

All models in application dorganism are displaying in the following database ERD:



The main models are taxonomy, organism, orgbatch, organism\_culture, orgbatch\_stock, MIC\_COADD, MIC\_Pub.

### dorganism application folder tree

├── dorganism

│   ├── \_\_init\_\_.py

│   ├── admin.py

│   ├── apps.py

│   ├── forms.py

│   ├── migrations

│   ├── models.py

│   ├── templates

│   │   └── dorganism

│   │   ├── organism

│   │   │   ├── batch

│   │   │   │   ├── batch\_c.html

│   │   │   │   ├── batch\_card.html

│   │   │   │   ├── batch\_d.html

│   │   │   │   ├── batch\_tr.html

│   │   │   │   └── batch\_u.html

│   │   │   ├── batch\_stock

│   │   │   │   ├── stock\_c.html

│   │   │   │   ├── stock\_d.html

│   │   │   │   ├── stock\_list.html

│   │   │   │   └── stock\_u.html

│   │   │   ├── culture

│   │   │   │   ├── culture\_c.html

│   │   │   │   ├── culture\_d.html

│   │   │   │   ├── culture\_tr.html

│   │   │   │   └── culture\_u.html

│   │   │   ├── datamap.html

│   │   │   ├── organism\_c.html

│   │   │   ├── organism\_card.html

│   │   │   ├── organism\_d.html

│   │   │   ├── organism\_detail.html

│   │   │   ├── organism\_file.html

│   │   │   ├── organism\_image.html

│   │   │   ├── organism\_list.html

│   │   │   ├── organism\_mic.html

│   │   │   └── organism\_u.html

│   │   └── taxonomy

│   │   ├── taxonomy\_c.html

│   │   ├── taxonomy\_card.html

│   │   ├── taxonomy\_d.html

│   │   ├── taxonomy\_detail.html

│   │   ├── taxonomy\_list.html

│   │   └── taxonomy\_u.html

│   ├── urls.py

│   ├── utils

│   │   ├── data\_visual.py

│   │   └── utils.py

│   └── views.py

Application dorganism tree structure contains:

* admin.py: register models to admin page
* apps.py
* forms.py
* migrations
* models.py contains models ([refer to models](#_Append_Model_List)list).
* templates
* urls.py
* utils, utils in apputil are shared functions and classes overall inside project (refer to [dorganism utils)](#_dorganism)
* views.py contains views functions and classes: (refer to  [model’s mvt diagram](#_Organism_Diagram))

### Taxonomy

#### Taxonomy Diagram

#### Code Implementation

##### Create Model

##### CRUD View Class

The implementation of CRUD of Taxonomy is simply making a new class inherited from Simple…View Class from utils/view\_base. For example, the update view will be:

class TaxonomyUpdateView(SimpleupdateView):

    form\_class=Taxonomy\_form

    template\_name='dorganism/taxonomy/taxonomy\_u.html'

    model=Taxonomy

    transaction\_use = 'dorganism'

Detail view is a function to pass context with individual object and update form: Taxonomy\_form.

@login\_required

def detailTaxonomy(req, slug=None):

    context={}

    object\_=get\_object\_or\_404(Taxonomy, urlname=slug)

    context["object"]=object\_

    context['form']=Taxonomy\_form(instance=object\_)

    return render(req, "dorganism/taxonomy/taxonomy\_detail.html", context)

##### Templates Rendering

* **For the Read View – overview(list)**

The main template is …/dorganism/templates/dorganism/taxonomy/taxonomy\_card.html, which includs : ‘[utils/leftbar.html](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_leftbar.html)’, [‘utils/sidebar.html’](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_sidebar.html), [‘utils/main\_horizalbar.html’](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_Append_3_Templates), [‘utils/datatable\_general.html’](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_datatable_general.html) and Javascripts included in ‘utils/script.html’.

With the URl in navbar menu button or landing page button with link {% url “taxo\_card” %} (codes: path('taxonomy\_card', TaxonomyCardView.as\_view(), name="taxo\_card") ) leads to the view page.

* **For the Read View – detailview**

With the URl - link to {% url “taxo\_detail” %} positioned in each taxonomy name in table/card views leads to the view page.

* **create View**

Modal-form called by Ajax with main template 'dorganism/organism/organism\_c.html' and trigger button included in 'utils/leftbar.html'

* **update View**

./organism/organism\_u.html with UpdateOrganism\_form passed by context

* **delete View**

./organism/organism\_u.html include 'utils/miscellaneous/delete\_btn.html' and

### Organism

#### Organism Diagram

#### Code Implementation

The code implementation approach: Create Model -> View -> Template.

##### Create Model

##### CRUD View Class

- Create View: There are two steps to create an organism entry:

1) Ajax Call(/search\_organism/) find Foreignkey orgname, so the function based view has a parameter: req.POST.get('search\_organism’) passed by ajax view function def search\_organism(req) at url: path('search\_organism/', search\_organism, name="search\_organism");

2) Submit request form with step 1 ‘s result.

- Read View:

1) for an overview, class OrganismListView(LoginRequiredMixin, FilteredListView) and class OrganismCardView(OrganismListView) are used in List View and Card View UIs.

2) for a detailed view, including following functions:

* + Single Organism entry’s detail information display, update and delete.
  + Related data-model batch, stock, culture overview display, update, create and delete:
  + Grouped information in dataframe: Antibiogram table and a pivot table based on it

The detail view is a function with a context not only containing single entry’s information of Organism in UpdateOrganism\_form also data from Organism\_Batch, Orgbatch\_Stock, Organism\_Culture data model as well as data for antibiogram from ddrug application.

- Update View: def updateOrganism(req, pk) contains steps:

1) update Organism Name with logic control for NOT updating name in different class and parameter: req.POST.get('search\_organism')

2) update other fields

- Delete View: class OrganismDeleteView([SimpledeleteView](#_views_base)) .

Correspondent URLs are:

* path('organism\_card', OrganismCardView.as\_view(), name="org\_card"),
* path('organism\_list', OrganismListView.as\_view(), name="org\_list"),
* path('organism/<str:pk>', detailOrganism, name="org\_detail"),
* path('createOrg/', createOrganism, name="org\_create"),
* path('updateOrg/<str:pk>', updateOrganism, name="organism\_update"),
* path('deleteOrg/<str:pk>', OrganismDeleteView.as\_view(), name="organism\_delete"),

##### Templates Rendering

* **For the Read View – overview(list)**

The main template is …/dorganism/templates/dorganism/organism/organism\_card.html, which includs : ‘[utils/leftbar.html](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_leftbar.html)’, [‘utils/sidebar.html’](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_sidebar.html), [‘utils/main\_horizalbar.html’](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_Append_3_Templates), [‘utils/datatable\_general.html’](file:///C:\Users\uqwzhon4\02%20Orgdbapp_Development%20Guide.docx#_datatable_general.html) and Javascripts included in ‘utils/script.html’

* **For the Read View – detailview**

Organism\_Batch, OrgBatch\_Stock, Organism\_Culture, Antibiogram(mic\_coadd), images and documents (which are Manytomany fields of organism) instance and data are passing by context to template detailview. So the main template …/dorganism/templates/dorganism/organism/organism\_detail.html including:

* + ./organism/organism\_u.html – template with UpdateOrganism\_form with initial values
  + 'utils/miscellaneous/entrylog.html' – for the entry create / update date and user
  + 'utils/lefticons.html' – contains update Organism, create batch/stock/culture modal trigger button.
  + 'utils/miscellaneous/showentries\_badge.html' – showing the amount of datamodel entries
  + 'utils/datatable\_sm.html' – Batch / Stock / Culture CRUD table view
  + 'dorganism/organism/organism\_image.html' – Image upload / display / delete view template
  + 'dorganism/organism/organism\_file.html' – Document upload / display / delete view template
  + ‘./organism\_mic.html’ – antibiogram dataframe and pivottable

With implementation of htmx and ajax functions to make update, delete and create these models’ objects on one web page. static/js/ajax\_deleteModal.js, create\_modal.js, childtable.js, create\_childtable.js, destroy\_childtable.js, loadModal.js (refer to [utils](#_views_base) and [javascript functions](#_Append_2_JavaScript))

* **create View**

Modal-form called by Ajax with main template 'dorganism/organism/organism\_c.html' and trigger button included in 'utils/leftbar.html'

* **update View**

./organism/organism\_u.html with UpdateOrganism\_form passed by context

* **delete View**

./organism/organism\_u.html include 'utils/miscellaneous/delete\_btn.html' and

* **Pivot Table View**

Pivot Table Views will be widely used in all applications. In organism Detailview there are a grouped table view for antibiogram from model VITEK\_AST in ddrug application and a pivoted table view based on the grouped table. Since this is special case in Organism Detail View, put utils functions for generate table in Ddrug/utils/tables

## Model List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Application | Model | Pk | Foreign key | Description |
| apputil | ApplicationUser | name(char) |  | authenticated users datatable |
| AuditModel | id(int) | acreated/ aupdated/adelete(ApplicationUser**, for all ChildClasses**) | an abstract model for other chemical datatables |
| Dictionary | dict\_value(char) |  | provide customized options for objects in other chemical datatables |
| ApplicationLog | id(int) | Log\_user(ApplicationUser) | save objects changes records |
| Document | id(int) |  | for file objects upload to server and save server links in datatables |
|  |  |  |  |  |
| docllab | Organisation | org\_id(char) | org\_type (Dictionary) | List of Organisation |
| Collab\_User | user\_id(char) | organisation (Organisation)  group (Collab\_Group) | List of Collaborative Groups |
| Collab\_Group | group\_id(char) | organisation (Organisation)  pi (Collab\_User)  mta\_status (Dictionary) | List of Collaborative Groups |
| Data\_Source | data\_id(char) | data\_type (Dictionary)  collab\_group (Collab\_Group) | List of Data sources |
|  |  |  |  |  |
| ddrug | Drug | drug\_id(char) | drug\_type (Dictionary) | List of Drugs, DrugCombinations, DrugScreens |
| Breakpoint | id(int) | drug\_id(Drug); org\_rank / notorg\_rank / bp\_type(Dictionary) | List of Breakpoints |
| VITEK\_Card | card\_barcode  (char) | orgbatch\_id(organism\_Batch); card\_type(Dictionary) | List of VITEK Cards |
| VITEK\_AST | id(int) | card\_barcode(VITEK\_Card); drug\_id(Drug) | Antimicrobial Suceptibility Testing (AST) data from VITEK Cards |
| VITEK\_ID | id(int) | card\_barcode(VITEK\_Card); | Identification Testing (ID) data from VITEK Cards |
| MIC\_COADD | id(int) | orgbatch\_id(Organism\_Batch); drug\_id(Drug); mic\_type(Dictionary); plate\_size(Dictionary); plate\_material(Dictionary) | Antibiogram from CO-ADD screening |
| MIC\_Pub | id(int) | organism\_id(Organism); drug\_id(Drug); mic\_type(Dictionary) | Antibiogram from Public sources |
|  |  |  |  |  |
| dgene | Gene | gene\_id(char) | gene\_type(Dictionary); | List of Genes |
| ID\_Pub | id(int) | organism\_id(Organism); id\_type(Dictionary); | Identification from Public or Collaborative sources |
| ID\_Sequence | id(int) | orgbatch\_id(Organism\_Batch); id\_type(Dictionary) | Identification from Whole Genome Sequencing |
| WGS\_FastQC | id(int) | orgbatch\_id(Organism\_Batch); | FastQC outcome - Import only |
| WGS\_CheckM | id(int) | orgbatch\_id(Organism\_Batch); | CheckM outcome - Import only |
|  |  |  |  |  |
| dorganism | Taxonomy | organism\_name(char) | org\_class(Dictionary); division(Dictionary); | Based on the NCBI Taxonomy at https://www.ncbi.nlm.nih.gov/taxonomy  with available link https://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?mode=Info&id=<Tax\_ID>  NCBI has an API but with Search/Fetch procedures https://www.ncbi.nlm.nih.gov/books/NBK25501/   NCBI Taxonomy specific fields (could be defined as models.TextChoice classes):  Tax\_Rank subspecies,species,species group,genus,order,class,family,phylum,no rank,varietas  Division Rodents, Bacteria, Mammals, Plants and Fungi, Primates  Division\_Code ROD, BCT, MAM, PLN, PRI |
| Organism | organism\_id(char) | organism\_name(Taxonomy); mta\_status / lab\_restriction / risk\_group / pathogen\_group / oxygen\_pref (Dictionary) ; biologist (ApplicationUser); assoc\_images (Image); assoc\_documents(Document) | Main class of Organisms/Bacterias/Fungi/Cells in Isolate Collection |
| Organism\_Batch | orgbatch\_id(char) | organism\_id(Organism); qc\_status(Dictionary); biologist(ApplicationUser) | Organism/Isolate Batch Collection |
| OrgBatch\_Image | Id(int) | Orgbatch\_id(Organism\_Batch) |  |
| OrgBatch\_Stock | id(int) | orgbatch\_id(Organism\_Batch); stock\_type(Dictionary); biologist(ApplicationUser) | Stock of Organism/Isolate Batches |
| Organism\_Culture | id(int) | organism\_id(Organism); culture\_type / culture\_source (Dictionary); biologist(ApplicationUser) | Recommanded and optimised Growth/Culture conditions |
|  |  |  |  |  |
| dscreen | Screen\_Run | run\_id(char) | run\_status, run\_type(Dictionary | # Screening Application Model |

## Database Schema

DATABASES = {

    'default': {

        "ENGINE": PG\_ENGINE,

        'OPTIONS':{'options': '-c search\_path=apputil,dorganism,public', 'isolation\_level': psycopg2.extensions.ISOLATION\_LEVEL\_SERIALIZABLE,},

        'NAME': database\_name,

        'USER': database\_user,

        'PASSWORD':database\_password,

        'HOST': HOST\_NAME,

        'PORT': '5432',

    },

    'dorganism': {

        "ENGINE": PG\_ENGINE,

        'OPTIONS':{'options': '-c search\_path=dorganism,apputil,public', 'isolation\_level': psycopg2.extensions.ISOLATION\_LEVEL\_SERIALIZABLE,},

        'NAME': database\_name,

        'USER': database\_user,

        'PASSWORD':database\_password,

        'HOST': HOST\_NAME,

        'PORT': '5432',

    },

    'ddrug': {

        "ENGINE": PG\_ENGINE,

        'OPTIONS':{'options': '-c search\_path=ddrug,dscreen,dorganism,apputil,public', 'isolation\_level': psycopg2.extensions.ISOLATION\_LEVEL\_SERIALIZABLE,},

        'NAME': database\_name,

        'USER': database\_user,

        'PASSWORD':database\_password,

        'HOST': HOST\_NAME,

        'PORT': '5432',

    },

    'dscreen': {

        "ENGINE": PG\_ENGINE,

        'OPTIONS':{'options': '-c search\_path=dscreen,apputil,public', 'isolation\_level': psycopg2.extensions.ISOLATION\_LEVEL\_SERIALIZABLE,},

        'NAME': database\_name,

        'USER': database\_user,

        'PASSWORD':database\_password,

        'HOST': HOST\_NAME,

        'PORT': '5432',

    },

    'dgene': {

        "ENGINE": PG\_ENGINE,

        'OPTIONS':{'options': '-c search\_path=dgene,dorganism,apputil,public', 'isolation\_level': psycopg2.extensions.ISOLATION\_LEVEL\_SERIALIZABLE,},

        'NAME': database\_name,

        'USER': database\_user,

        'PASSWORD':database\_password,

        'HOST': HOST\_NAME,

        'PORT': '5432',

    },

    'dcollab': {

        "ENGINE": PG\_ENGINE,

        'OPTIONS':{'options': '-c search\_path=dcollab,apputil,public', 'isolation\_level': psycopg2.extensions.ISOLATION\_LEVEL\_SERIALIZABLE,},

        'NAME': database\_name,

        'USER': database\_user,

        'PASSWORD':database\_password,

        'HOST': HOST\_NAME,

        'PORT': '5432',

    }

}

DATABASE\_ROUTERS = ['adjcoadd.routers.DatabaseRouter',]

## Python Utils

### apputil

#### api\_class

provide base list view for API view, including:

* + CustomPagination: define pagination class for API List View
  + API\_ListView: generic List view.

#### data\_style

Data Visualization functions, including:

* + highlight\_val: set background of data-frame table or pivoted table cell depending on the value of this cell with the following arguments and settings:
    - val -> value of table cell
    - threshold\_number -> number to decide which background color should be chose.
    - Empty value with white color, bigger than threshold\_number with #dfba9f, else with yellow
  + highlight\_val2: same functionalities as above with different color settings and color\_threshold is ‘R’,’S’, ‘L’ as Char value.
  + convert\_heatmap: apply style functions highlight\_val and \_val2 to uploaded data tables with the following arguments:
    - xls\_file: choose uploaded xls.
    - XlsSheet: sheet in xls\_file
    - upload: default False
    - uploaduser: application user
    - lower: default True

#### data

General utilities for processing List, Dictionaries

#### files\_upload

basic functions and classes used in files uploading process for example import data by pdf via form\_wizard\_tools, including:

* + function file\_location: to create user folder in project MEDIA\_ROOT (e.g., in server project/uploads/<username>)
  + OverwriteStorage: Override filename in FileStorage, remove the existed file when new uploading file has the same name.
  + FileValidator with deconstructible(to make class instance serializable, in case will be needed for a model form field and makemigrations): used to check virus and file type. For file size check is commented out.
  + validate\_file: instance validator of FileValidator, used directly in form fields
  + MultiFileUploadForm (Legacy)
  + FileUploadForm (Legacy)
  + Importhandler (Legacy), replaced by ImportHandler\_WizardView in form\_wizard\_tools
  + run\_impProcess: run an upload process only for command line

#### filters\_base

Django-filter utilities functions and classes, including:

* + get\_all\_fields\_q\_object: called by Filter set Class, define a method to search different fields. Currently it includes CharField, Foreinkey(search primary key field), IntegerField, Array Field which should be cover all the field in current models.

**!! If there are Fields not included, then append elif condition in the function (see code comments).**

* + get\_all\_fields\_q\_object\_deep: The difference between get\_all\_fields\_q\_object and \_deep is searching related table, with the first one only searching primarykey field while the second one searching all fields (except excluded fields) in related tables.
  + Filterbase\_base: basic filter set class provided for model filter forms, it contains Search\_all\_fields with the following methods:
    - multichoices\_filter for arrayfield search with overlap condition.
    - filter\_all\_fields excluded fields with password, astatus, in foreignkey field searching related tables’ pk
    - filter\_all\_fields\_deep excluded fields with password, astatus, search through all related tables
    - inside init method receive user request ‘deep’ to switch filter all fields methods between normal and deep; override charfilter’s lookup\_expr to ‘icontains’
  + Filterbase: inherited from Filterbase\_base class, conditionally set qs with astatus\_\_gte=0 as property.
  + find\_item\_index, used to check the field in HEADFIELDS is a dictionary or not. Then choose the correct field’s name to use in the method *def ordered\_by* in class *FilteredListView*
  + FilteredListView: is inherited Django ListView and used for all model general views. With the following attributes and methods:

The attributes are,

* + - filterset\_class, integrated with model’s filterset class
    - paginated\_by (inherited pagination settings from Django Listview, default is 50 and value changed by the method self.get\_paginated\_by)
    - model\_fields, model HEADERFIELDS.
    - order\_by, refer to method get\_order\_by
    - filter\_Count, used to save qs.count()
    - app\_name, model’s app label
    - model\_name

The methods are,

* + - get\_queryset: build in method, based on default model queryset to return the filtered queryset with filter methods(deep or normal search) under the orders from method self.get\_order\_by (). And assign qs.count() to self.filter\_Count
    - get\_context\_data, data need to be passed to html
    - get\_paginate\_by
    - get\_order\_by.

#### Flex\_pivottable

Utility functions to create pivotable based on filtered or chosen data and customized columns / index / values. Contains 2 functions:

* get\_pivottable is a util function used to convert a data set via selected aggfunc, columns, index, values, which are all passed by flex\_pivottable function to pivoted tables
* flex\_pivottable is a view function used to parse submitted request form from html -> pass them to get\_pivottable -> get table result -> add styling -> rend and pass table as context to html. The received request arguments are,
  + - app\_model as url param and parsed to app name and model name
    - values
    - column
    - index
    - data\_function\_type

#### form\_wizard\_tools

Steps process driven by form submission requests. for importing data with PDFs. Including:

* + UploadFileForm: The steps are start with uploading though submitting a UploadFileForm.
  + StepFrom\_x: the between steps’ form customized based on different model requirements.
  + FinalizForm: the last step, return any user input log and back to the first step
  + ImportHandler\_WizardView: inherited from SuperUserRequiredMixin and SessionWizardView provide base view for each application, sharing class method (delete\_file) and class attribute file\_storage. The customized parts inlcude:

1. Class attributes: step1(more steps, define steps name), form\_list(for each step/form pair) , template\_name
2. \_\_init\_\_, precess\_step(main steps controller), done(called by finalizform), get\_context\_data(put each steps result and info here)

#### validation\_log

In-process Logging class to capture outcomes of validation and processing tasks as logTypes = ['Error','Warning','Info']

#### views\_base

View classes for CRUD and permission functions, including:

* + Permission\_not\_granted: return a response with text “Permission Not Granted”
  + SuperUserRequiredMixin: check user permission return true if user has Admin permission, otherwise false.
  + SimplecreateView: creating data with regular model form, saved with creator(request.user)
  + SimpleupdateView: update data with regular model form, saved with editor(request.user)
  + SimpledeleteView: inherited from SimpleupdateView, to change astatus to -9, used overall except model ApplicationUser and Dictionary
  + HtmxupdateView: update data in an editable row in ListView table implemented with HTMX
  + CreateFileView: for create image\_batch and document model instance
  + DataExportBaseView: base view for export selected data

### dcollab

### ddrug

#### bio\_data

#### bio\_updates

#### import\_drug

* + imp\_Drug\_fromDict
  + imp\_VitekCard\_fromDict
  + imp\_VitekID\_fromDict
  + imp\_VitekAST\_fromDict
  + imp\_MICCOADD\_fromDict
  + imp\_MICPub\_fromDict

#### molecules

util functions for configuration of molecules’ datatype and queryset

* + smiles2mol, return xmol
  + def molecule\_to\_svg, convert and save structures as SVGs, including:
    - file\_path: project/static/images
  + clearIMGfolder(legacy): used to clean folder static/images
  + get\_mfp2\_neighbors: used to Similarity Query

#### tables

#### vitek

Functions to process uploading and parsing Vitek pdfs, when import Vitek data.

### dgene

#### import\_gene

#### parse\_wgs

### dorganism

#### reformat\_OrganismID

#### reformat\_OrgBatchID

#### search\_organism

User searching organism name in Taxanomy (Ajax searching on-Key-up), called by url “dorganism/search\_organism” with:

* + User input words

Return:

* + data: organism name, organism class or “noClass by Import or…”

### dscreen

## 

## Template Tags

### Custom\_filters:

from django import template

import re

from dorganism.models import OrgBatch\_Stock

from adjcoadd.constants import LinkList

register = template.Library()

@register.filter

def count\_filtered\_stock(object\_batch):

    return OrgBatch\_Stock.objects.filter(orgbatch\_id=object\_batch, astatus\_\_gte=0, n\_left\_\_gt=1).count()

@register.filter

def is\_dict(value):

    return isinstance(value, dict)

@register.filter

def is\_list(value):

    return isinstance(value, list)

@register.filter

def to\_valid\_selector(value):

    return "a" + re.sub(r"[\s\.\#\[\]\(\)\+\>\~\=\'\\*\^\$]", "\_", str(value))

@register.filter

def to\_int(value):

    if value != "":

        return int(value)

    else:

        return 0

@register.filter

def get\_linkname(value):

    if value in LinkList.keys():

        return LinkList[value]

    else:

        return None

### Pagination:

from django import template

register= template.Library()

@register.simple\_tag

def pagination\_url(value, field\_name, urlencode=None):

    url='?{}={}'.format(field\_name, value)

    if urlencode:

        querystring=urlencode.split('&')

        filtered\_querystring=filter(lambda p: p.split('=')[0]!=field\_name, querystring)

        encoded\_querystring='&'.join(filtered\_querystring)

        url='{}&{}'.format(url, encoded\_querystring)

    return url

## Application Templates

Applications’ Templates are used to render Views’ UIs, which has the hierarchy:

Base (base.html)

* application templates (model\_view htmls)
  + - utils templates

### templates/Base

#### base.html

including all CDN links, meta data and most javascripts (shared by application templates)

### apputil/templates

#### home.html

Application home page with including:

* landbtn\_primary button html:

example:

{% url 'org\_card' as org\_url%}

{% include 'utils/miscellaneous/landbtn\_primary.html' with title='ORGANISM' url=org\_url totalnumber=nOrg %}

* landbtn\_secondary button html:

example:

{% url 'taxo\_card' as taxonomy\_url %}

{% include 'utils/miscellaneous/landbtn\_secondary.html' with title='Taxonomy' url=taxonomy\_url totalnumber=nTax %}

#### registration

#### ./login.html

for webapp login includes: message.html

#### ./error400.html

Render page when 400 happens

#### ./error404.html

Render page when 404 happens

#### ./error500.html

Render page when 500 happens

#### apputil

#### ./appuser\_tr.html

partial render data row includes:

* update button for htmx: utils/miscellaneous/update\_btn\_htmx.html
* delete pages: apputil/appUsersDel.html

#### ./appUserProfile.html (not in use)

Application users profile page, includes.

* utils/sidebar.html

#### [./appUsers.html](#_Templates_Rendering)

Render Application users’ data overview, including:

* utils/leftbar
* ./sidebar
* ./main\_horizalbar
* ./datatable\_general,

#### [./appUsersCreate.html](#_Templates_Rendering)

Render create view, modal view. Including utils/modal/modal\_form.html

#### [./appUsersDel.html](#_Code_Implementation)

Render delete view, modal view. Including 'utils/miscellaneous/delete\_btn.html'

#### [./appUsersUpdate.html](#_Templates_Rendering)

Partially render (replace [./appuser\_tr.html](#_apputil/appuser_tr.html)) update view with hx-put (update url) and includes 'utils/miscellaneous/cancel\_btn\_reloc.html'.

#### ./dictCreate.html

Render create view, modal view. Including utils/modal/modal\_form.html

#### ./dictList.html

Render Dictionary data’s overview, including:

* utils/main\_horizalbar.html
* ./dictionary\_tr.html

#### ./dictionary\_tr.html

Render data row view and update/delete view via ajax-script (static/js/editablerow.js)

#### ./importhandler\_excel.html (Not in use)

Render one step import data via excel sheets, including utils/import\_singlestep.html.

### dcollab/templates

(…)

### ddrug/templates

### ddrug/

#### breakpoint/breakpoint\_list.html

Render Breakpoint’ data overview, including:

* utils/leftbar
* ./sidebar
* ./main\_horizalbar
* ./datatable\_filter.html,

#### drug

#### ./drug\_c.html

Render create view, modal view. Including utils/modal/modal\_form.html

#### ./drug\_card.html

Render Drug’s data overview, including:

* utils/leftbar (no variable *import\_module* inside leftbar, ignore if it exists in template tag include.)
* ./sidebar
* ./main\_horizalbar
* ./card.html,

#### ./drug\_d.html

Render delete view, modal view. Including:

* 'utils/miscellaneous/delete\_btn.html'
* 'utils/modal/delete.html'

#### ./drug\_detail.html

Render single row detail view page, including:

* utils/lefticons.html
* utils/message.html
* [ddrug/drug/drug\_u.html](#_./drug_u.html)
* utils/miscellaneous/entrylog.html

#### ./drug\_detail\_structure.html(not in use)

#### ./drug\_list.html

Render Drug’s data overview, including:

* utils/leftbar (no variable *import\_module* inside leftbar, ignore if it exists in template tag include.)
* ./sidebar
* ./main\_horizalbar
* ./datatable\_general.html,

#### ./drug\_u.html

Render row data in detail view, partial replace data row with form groups via javascript static/js/editableTable.js and integrate Ketcher Editor for drug’s molecular structure / mol / smile field. Including:

* utils/row\_editabletable.html
* update form for drug and
* utils/miscellaneous/save\_btn.html and utils/miscellaneous/cancel\_btn\_reloc.html
* utils/ketcher.html

#### mic\_coadd

#### . / mic\_coadd\_card.html (not in use)

#### . / mic\_coadd\_list.html

Render Drug’s data overview, including:

* utils/leftbar (no variable *import\_module* inside leftbar, ignore if it exists in template tag include.)
* ./sidebar
* ./main\_horizalbar
* ./datatable\_general.html

#### mic\_pub

#### . / mic\_pub\_card.html (not in use)

#### . / mic\_pub\_list.html

Render MIC\_Pub’s data overview, referring to mic\_coadd\_list

#### vitek\_ast

#### . / vitekast\_list.html

Render VITEK\_AST’s data overview, referring to mic\_coadd\_list

#### vitek\_card

#### . / vitekcard\_list.html

Render VITEK\_CARD’s data overview, referring to mic\_coadd\_list

#### vitek\_id

#### . / vitek\_list.html

Render VITEK\_ID’s data overview, including:

* Referring to mic\_coadd\_list
* ./datatable\_filter.html

#### Importhandler\_vitek.html

Render import data view, including:

* utils/wizard\_single\_page.html

#### Importhandler\_drug.html (not in use)

### ddrug

│   ├── templates

│   │   └── ddrug

│   │   ├── breakpoint

│   │   │   └── breakpoint\_list.html

│   │   ├── drug

│   │   │   ├── drug\_c.html

│   │   │   ├── drug\_card.html

│   │   │   ├── drug\_d.html

│   │   │   ├── drug\_detail.html

│   │   │   ├── drug\_detail\_structure.html

│   │   │   ├── drug\_list.html

│   │   │   └── drug\_u.html

│   │   ├── importhandler\_drug.html

│   │   ├── importhandler\_vitek.html

│   │   ├── mic\_coadd

│   │   │   ├── mic\_coadd\_card.html

│   │   │   └── mic\_coadd\_list.html

│   │   ├── mic\_pub

│   │   │   ├── mic\_pub\_card.html

│   │   │   └── mic\_pub\_list.html

│   │   ├── vitek\_ast

│   │   │   └── vitekast\_list.html

│   │   ├── vitek\_card

│   │   │   ├── vitekcard\_detail.html

│   │   │   └── vitekcard\_list.html

│   │   └── vitek\_id

│   │   └── vitekid\_list.html

### dgene/templates

### dgene

#### gene (hidden)

#### ./gene\_c.html

Render create view, modal view. Including utils/modal/modal\_form.html

#### ./gene\_detail.html

Render single row detail view page, including:

* utils/lefticons.html
* utils/message.html
* [dgene/gene/gene\_u.html](#_./gene_u.html)
* utils/miscellaneous/entrylog.html

#### ./gene\_list.html

Render Gene’s data overview, including:

* utils/leftbar (no variable *import\_module* inside leftbar, ignore if it exists in template tag include.)
* ./sidebar
* ./main\_horizalbar
* ./datatable\_filter.html,

#### ./gene\_u.html

Render row data in detail view and partial replace data row with form groups via javascript static/js/editableTable.js. Including:

* utils/row\_editabletable.html
* update form for drug and
* utils/miscellaneous/save\_btn.html and utils/miscellaneous/cancel\_btn\_reloc.html

#### id\_pub (hidden)

#### ./id\_pub\_c.html

Render create view, modal view. Including utils/modal/modal\_form.html

#### ./id\_pub\_detail.html (not in use)

#### ./id\_pub\_list.html

Render ID\_Pub’s data overview, including:

* utils/leftbar (no variable *import\_module* inside leftbar, ignore if it exists in template tag include.)
* ./sidebar
* ./main\_horizalbar
* ./datatable\_filter.html,

#### ./id\_pub\_u.html (Not in Use)

#### id\_sequence

#### . / sequence\_c.html (Not in use)

#### . / sequence\_list.html

Render ID\_Sequence’s data overview, referring to [mic\_coadd\_list](#_._/_mic_coadd_list.html)

#### . / sequence\_c.html (Not in use)

#### wgs\_checkm

#### . / checkm\_list.html

Render WGS\_CheckM’s data overview, referring to [id\_pub\_list](#_./id_pub_list.html)

#### wgs\_fastqc

#### . / fastqc\_list.html

Render WGS\_FastQC’s data overview, referring to [id\_pub\_list](#_./id_pub_list.html)

**dgene**

│   ├── templates

│   │   └── dgene

│   │   ├── gene

│   │   │   ├── gene\_c.html

│   │   │   ├── gene\_detail.html

│   │   │   ├── gene\_list.html

│   │   │   └── gene\_u.html

│   │   ├── id\_pub

│   │   │   ├── id\_pub\_c.html

│   │   │   ├── id\_pub\_detail.html

│   │   │   ├── id\_pub\_list.html

│   │   │   └── id\_pub\_u.html

│   │   ├── id\_sequence

│   │   │   ├── sequence\_c.html

│   │   │   ├── sequence\_list.html

│   │   │   └── sequence\_u.html

│   │   ├── wgs\_checkm

│   │   │   └── checkm\_list.html

│   │   └── wgs\_fastqc

│   │   └── fastqc\_list.html

### dorganism/templates

### dorganism

#### organism

./batch

#### ./batch\_c.html

Render create view, modal view. Including utils/modal/modal\_form.html

#### ./batch\_d.html

Render delete view, modal view. Including:

* 'utils/miscellaneous/delete\_btn.html'
* ‘utils/modal/delete.html’

#### ./batch\_list.html (Not in use)

As batch listed in it’s related Organism detail view.

#### ./batch\_tr.html

Render batch data row view inside utils/datatable\_sm.html via variable ‘*html\_name*’, with implemented htmx function: partially rendering update view (./batch\_u.html).

#### ./batch\_u.html

Partially render update view via htmx.

#### ./batch\_c.html

Render create view, modal view. Including utils/modal/modal\_form.html

#### ./batch\_d.html

Render delete view, modal view. Including:

* 'utils/miscellaneous/delete\_btn.html'
* ‘utils/modal/delete.html’

#### ./batch\_list.html (Not in use)

As batch listed in it’s related Organism detail view.

#### ./batch\_tr.html

Render batch data row view inside utils/datatable\_sm.html via variable ‘*html\_name*’, with implemented htmx function: partially rendering update view (./batch\_u.html).

#### ./batch\_u.html

Partially render update view via htmx.

**dorganism**

│   ├── templates

│   │   └── dorganism

│   │   ├── organism

│   │   │   ├── batch

│   │   │   │   ├── batch\_c.html

│   │   │   │   ├── batch\_card.html

│   │   │   │   ├── batch\_d.html

│   │   │   │   ├── batch\_tr.html

│   │   │   │   └── batch\_u.html

│   │   │   ├── batch\_stock

│   │   │   │   ├── stock\_c.html

│   │   │   │   ├── stock\_d.html

│   │   │   │   ├── stock\_list.html

│   │   │   │   └── stock\_u.html

│   │   │   ├── culture

│   │   │   │   ├── culture\_c.html

│   │   │   │   ├── culture\_d.html

│   │   │   │   ├── culture\_tr.html

│   │   │   │   └── culture\_u.html

│   │   │   ├── datamap.html

│   │   │   ├── organism\_c.html

│   │   │   ├── organism\_card.html

│   │   │   ├── organism\_d.html

│   │   │   ├── organism\_detail.html

│   │   │   ├── organism\_file.html

│   │   │   ├── organism\_image.html

│   │   │   ├── organism\_list.html

│   │   │   ├── organism\_mic.html

│   │   │   └── organism\_u.html

│   │   └── taxonomy

│   │   ├── taxonomy\_c.html

│   │   ├── taxonomy\_card.html

│   │   ├── taxonomy\_d.html

│   │   ├── taxonomy\_detail.html

│   │   ├── taxonomy\_list.html

│   │   └── taxonomy\_u.html

### dscreen

## Append Templates Utils

### card.html

* Usage: used as partial html in <model>\_card.html with include, with tag
* Functions:
  + display all the entry or all query result in one table’s List View Template.
* Content:
  + context variables “object”- data model instance, send from Django views;
  + template variables*:*

*url\_name* – link to single object detailview;

*object\_name or object\_name\_ita –* object name for link url\_name;

*object\_img* – object image name if exists(e.g., Drug)*;*

*object\_id\_slug* – to control expand button needs unique ID with object\_id or slug*, card\_fields* – {instance field: value}, if exists in model;

*object\_var\_link\_label and object\_var\_link* – name of external web link for TAX\_ID ;the rest variables are customized to display instance fields and values: *object\_var2\_label;…object\_var9\_label; object\_var2;…object\_var9*

* + Html...

### data\_download\_form.html

* Usage:
* Functions:
  + Data export from model tables or antibiogram table.
* Content:
  + context variables “object”- data model instance, send from Django views;
  + template variables*:*

*url\_name* – link to single object detailview;

*object\_name or object\_name\_ita –* object name for link url\_name;

*object\_img* – object image name if exists(e.g., Drug)*;*

*object\_id\_slug* – to control expand button needs unique ID with object\_id or slug*, card\_fields* – {instance field: value}, if exists in model;

*object\_var\_link\_label and object\_var\_link* – name of external web link for TAX\_ID ;the rest variables are customized to display instance fields and values: *object\_var2\_label;…object\_var9\_label; object\_var2;…object\_var9*

* + Html...

### datatable\_filter.html

* Usage:
  + used as partial html in <model>\_list.html
  + **Important Note**: when use datatable\_filter, the HEAD\_FIELDS will be used as form fields in model’s django-filter-form:

A screenshot of a computer program

Description automatically generated

* Functions:
  + display all the entry or all filtered query result in one table’s List View Template.
* Content: It is made from:
  + context variables: “field/filter.form”and “object.get\_values/object/page\_obj” send from Django views
  + form – with id=django-filter-table
    - A table with boostrap classes/css styling (refer to [datatable\_general.html](#_datatable_general.html))
    - Input element under each thead/tr/td for typing query key words.
    - JavaScript to submit form.
  + Integrate with custom filter: is\_dict to check instance value data structure, return dictionary key and value with the href link

### datatable\_general.html

* Usage: used as partial html in <model>\_list.html
* Functions:
  + display all the entry or all query result in one table’s List View Template.
  + For ApplicationUser Model including editable function
* Content: It is made from:
  + context variables: “field/fields”(from Model’s HEAD\_FIELDS)and “object.get\_values/object/page\_obj” send from Django views
  + field in fields are “customized/verbose model fields name”, an external web link or internal link to detail view is in fields with data type: dict.
  + Object.get\_value in page\_obj are single object’s values list according to field, same as above for data type: dict.
  + custom\_filters – is\_dict: to check field or value is dict or not
  + a div container: with boostrap classes/css styling defined the whole block displaying size
  + a table with boostrap classes/css styling:
    - table thead / tr / td:
      * First column is “spare for selecting or editing”;
      * The rest column contains data fields from views.
    - table tbody:
      * /tr / td : 1. First column is included partial html “utils/ miscellaneous/selectbox.html” ; 2. Rest columns are data values according to the fields in the thead**, or**
      * Inlcude <model>\_tr.html with tr/td, which used for updating via partial rendered html (htmx)

### datatable\_sm.html

* Usage: used as partial html in <model>\_detail.html for a single object detail view(Organism Detail View) and displaying entries of foreignkey tables of this single objects.
* Functions:
  + Editable and generate child tables.
* Content: It has made from:
  + context variables and template variables: *table\_id, hiddenfield, html\_name, data\_fields*.
  + JavaScript: HTMX CSDN, create\_childtable.js, destroy\_childtable.js, childtable.js, scripts for defining jQuery DataTable and call child table scripts.
  + a table with boostrap classes/css styling:
    - table thead / tr / td:
      * First column is “spare for selecting or editing”;
      * May contain Second column depending on hiddenfield exists or not;

The rest column contains data fields from data\_fields(from Model’s HEAD\_FIELDS)

* + - table tbody:
      * /tr / td : 1. First column is object.organism; 2. Rest columns are data values according to the fields in the thead, or
      * Include a html html\_name,

### django\_form\_lgtable.html

* Usage: partial html wrapped by create and update entry bootstrap modal form
* Functions: render django table form in the same style
* Content:
  + context variables: ‘*form.visible\_fields*’ – default groups of djano form fields with visible attribute, ‘*field*’ – itera item in form.visible\_fields, *‘field.help\_text*’ – showing fields’ information if defined in forms, ‘*field.errors*’ – display when exists during form validation
  + javascripts used to style html text in <option></option>

### ketcher.html

* Usage: to implement Ketcher react application
* Functions: render ketcher canvas, mol / smiles and other functions
* Content: a react virtual DOM as div element with root id, for rendering all react elements

### leftbar.html

* Usage: wrapping function buttons: hidefilter, listview, cardview, pivoteview and create new entry, placing on the main readview most left side, included by all model overview template (e.g., organism\_card.html…)
* Functions: hiding fitler, switch between list and card view, to pivoteview for current model, create new entry for current model.
* Content:
  + Variables passed by main overview template:
    - “list\_url” – listview context variable,
    - “card\_url” – cardview context variable,
    - “url\_pivotedtable” – pivote url, e.g,
  + Includes 'utils/miscellaneous/create\_a.html'

Example in organism\_card.html:

{% url 'pivoted-table' app\_model='dorganism-Organism' as url\_pivotedtable%}

{% include 'utils/leftbar.html' with list\_url='org\_list' card\_url='org\_card' create\_object='createOrganism' create\_objectModal='createOrganismModal'  url\_pivotedtable=url\_pivotedtable %}

### lefticon.html

* Usage: used in detail view and contains different cases.
* Functions: create, update, delete, retrieve data,
* Content:

### main\_horizalbar.html

1. Usage:
2. Functions:
3. Content: It has made from:

### message.html

1. Usage:
2. Functions:
3. Content: It has made from:

### messagebox.html

1. Usage:
2. Functions:
3. Content: It has made from:

### navbar.html

1. Usage:
2. Functions:
3. Content: It has made from:

### pagination.html

1. Usage:
2. Functions:
3. Content: It has made from:

### row\_editabletable.html

1. Usage:
2. Functions:
3. Content: It has made from:

### scripts.html

1. Usage:
2. Functions:
3. Content: It has made from:

### search\_organism\_id.html

1. Usage:
2. Functions:
3. Content: It has made from:

### search\_organism.html

1. Usage:
2. Functions:
3. Content: It has made from:

### sidebar.html

1. Usage:
2. Functions:
3. Content: It has made from:

### wizard\_single\_page.html

1. Usage:

Used in importhandler\_<model>.html

1. Functions:

Steps operations for import data from single or multiple .pdf or .excel files

Steps will be controlled by template language:

{% if wizard.steps.current == 'upload\_file' %} // this is to check process steps

<h2>Upload File</h2> // here do something within each step

Please only use one of path.

{{ wizard.form.as\_p }}

1. Content: It has made from:
   * context variables and template variables*: data\_modal – indicate which model data import; stepsname, step1…stepn – indicate steps number or names; wizard.management\_form*, *wizard.form.as\_p*
   * JavaScript: to displaying a file list uploaded.
   * A form and control button “previous”, “next”, “cancel” with boostrap classes/css

### drag\_n\_drop

#### selectivedrag\_hr.html

#### selectivedrag\_vt.html

### miscellaneous

#### cancel\_btn\_htmx.html

#### cancel\_btn\_reloc.html

#### cancel\_btn.html

#### create\_a.html

#### delete\_btn.html

#### entrylog.html

#### preloader.html

#### save\_btn.html

#### selectAllExp.html

#### selectbox.html

#### showentries\_badge.html

#### update\_btn.html

### modal

#### delete.html

#### modal\_form.html

## JavaScript Functions

All Javascript written in jQuery, depended on jquery-3.6.1.min.js, unless with extra explanation.

### js\_utils

* third package javascirpt(dataTables.min.js)
* django framework template default javascripts (getCookie.js)
* templates js (not used directly)

### ketcher

* index-bundle.min.js (ketcher standalone application)

### childtable

Main function for create a Stock table as each batch’s child table, based on dataTables.min.js.

* Event trigger: the ‘a’ element with class “dt-control” in each Batch table row
* Behavior call function: destroyChild(), createChild()

Dependencies:

* dataTables.min.js
* destroy\_childtable.js
* create\_childtable.js

### destroy\_childtable

* Behavior: called by childtable
* Function: remove stock table.

### create\_childtable

* Behavior: called by childtable
* Function: generate stock table with.

editablerow

utility function for class “editablerow”

**input parameter**

**usage**

import script to html, using class name ‘editablerow’ for tr html element.

editableTable

utility function for class “editableTable”

**input parameter**

**usage**

import script to html, using class name ‘editableTable’ for td html element.

### editableTable

utility function for class “editableTable”

**input parameter**

**usage**

import script to html, using class name ‘editableTable’ for td html element. prevented from keypress event

### send\_organism / send\_organism\_id

utility function to search and select organism name or organism\_id in the taxonomy and organism table respectively and assign the selected value to create and update Django form input

**input parameter**

scheduled\_function – functionally delay the sending data to server for a certain ms, initial value is False and will be assigned with a setTimeout function with 500ms inside the keyup event function.

searchInput – searching input

resultsBox – displaying selectable searching results

**usage**

import script to search\_organism.html / search\_organism\_id.html, using in organism create and update view, batch and culture create view.

### sorting\_table

utility function to sorting data entries through data table for list/card view.

**input parameter**

localStorage.setItem( ‘dorder’, ‘-‘) – desc ordering

localStorage.removeItem(‘dorder’) – asc ordering

order\_name – ordering field, input field in Django filter form

**usage**

import script to the html, add ‘order\_field’ class to th html element

vitek\_datamap(not used)

utility function to create pivot table for vitekcard.

**input parameter**

**usage**

## Project Structure Tree

.

├── MANIFEST.in

├── README.rst

├── adjcoadd

│   ├── \_\_init\_\_.py

│   ├── asgi.py

│   ├── constants.py

│   ├── routers.py

│   ├── settings.py

│   ├── urls.py

│   └── wsgi.py

├── applog

│   ├── django\_server.log

│   └── user.log

├── apputil

│   ├── \_\_init\_\_.py

│   ├── admin.py

│   ├── api\_filterclass.py

│   ├── apps.py

│   ├── clear\_session\_middleware.py

│   ├── forms.py

│   ├── migrations

│   ├── models.py

│   ├── signals.py

│   ├── templates

│   │   ├── apputil

│   │   │   ├── appUserProfile.html

│   │   │   ├── appUsers.html

│   │   │   ├── appUsersCreate.html

│   │   │   ├── appUsersDel.html

│   │   │   ├── appUsersUpdate.html

│   │   │   ├── appuser\_tr.html

│   │   │   ├── async.html

│   │   │   ├── dictCreate.html

│   │   │   ├── dictList.html

│   │   │   ├── dictionary\_tr.html

│   │   │   └── importhandler\_excel.html

│   │   ├── home.html

│   │   └── registration

│   │   └── login.html

│   ├── urls.py

│   ├── utils

│   │   ├── api\_filterclass.py

│   │   ├── data.py

│   │   ├── data\_style.py

│   │   ├── files\_upload.py

│   │   ├── filters\_base.py

│   │   ├── flex\_pivottable.py

│   │   ├── form\_wizard\_tools.py

│   │   ├── validation\_log.py

│   │   └── views\_base.py

│   └── views.py

├── dcollab

│   ├── \_\_init\_\_.py

│   ├── admin.py

│   ├── apps.py

│   ├── models.py

│   └── views.py

├── ddrug

│   ├── \_\_init\_\_.py

│   ├── admin.py

│   ├── apps.py

│   ├── forms.py

│   ├── migrations

│   ├── models.py

│   ├── serializers.py

│   ├── templates

│   │   └── ddrug

│   │   ├── breakpoint

│   │   │   └── breakpoint\_list.html

│   │   ├── drug

│   │   │   ├── drug\_c.html

│   │   │   ├── drug\_card.html

│   │   │   ├── drug\_d.html

│   │   │   ├── drug\_detail.html

│   │   │   ├── drug\_detail\_structure.html

│   │   │   ├── drug\_list.html

│   │   │   └── drug\_u.html

│   │   ├── importhandler\_drug.html

│   │   ├── importhandler\_vitek.html

│   │   ├── mic\_coadd

│   │   │   ├── mic\_coadd\_card.html

│   │   │   └── mic\_coadd\_list.html

│   │   ├── mic\_pub

│   │   │   ├── mic\_pub\_card.html

│   │   │   └── mic\_pub\_list.html

│   │   ├── vitek\_ast

│   │   │   └── vitekast\_list.html

│   │   ├── vitek\_card

│   │   │   ├── vitekcard\_detail.html

│   │   │   └── vitekcard\_list.html

│   │   └── vitek\_id

│   │   └── vitekid\_list.html

│   ├── tests\_api.py

│   ├── upload\_views.py

│   ├── urls.py

│   ├── utils

│   │   ├── bio\_analysis.py

│   │   ├── bio\_data.py

│   │   ├── bio\_updates.py

│   │   ├── import\_drug.py

│   │   ├── molecules.py

│   │   ├── tables.py

│   │   └── vitek.py

│   └── views.py

├── deploy\_gunicorn.txt

├── dgene

│   ├── \_\_init\_\_.py

│   ├── admin.py

│   ├── apps.py

│   ├── forms.py

│   ├── migrations

│   ├── models.py

│   ├── templates

│   │   └── dgene

│   │   ├── gene

│   │   │   ├── gene\_c.html

│   │   │   ├── gene\_detail.html

│   │   │   ├── gene\_list.html

│   │   │   └── gene\_u.html

│   │   ├── id\_pub

│   │   │   ├── id\_pub\_c.html

│   │   │   ├── id\_pub\_detail.html

│   │   │   ├── id\_pub\_list.html

│   │   │   └── id\_pub\_u.html

│   │   ├── id\_sequence

│   │   │   ├── sequence\_c.html

│   │   │   ├── sequence\_list.html

│   │   │   └── sequence\_u.html

│   │   ├── wgs\_checkm

│   │   │   └── checkm\_list.html

│   │   └── wgs\_fastqc

│   │   └── fastqc\_list.html

│   ├── urls.py

│   ├── utils

│   │   ├── import\_gene.py

│   │   └── parse\_wgs.py

│   └── views.py

├── dorganism

│   ├── \_\_init\_\_.py

│   ├── admin.py

│   ├── apps.py

│   ├── forms.py

│   ├── migrations

│   ├── models.py

│   ├── templates

│   │   └── dorganism

│   │   ├── organism

│   │   │   ├── batch

│   │   │   │   ├── batch\_c.html

│   │   │   │   ├── batch\_card.html

│   │   │   │   ├── batch\_d.html

│   │   │   │   ├── batch\_tr.html

│   │   │   │   └── batch\_u.html

│   │   │   ├── batch\_stock

│   │   │   │   ├── stock\_c.html

│   │   │   │   ├── stock\_d.html

│   │   │   │   ├── stock\_list.html

│   │   │   │   └── stock\_u.html

│   │   │   ├── culture

│   │   │   │   ├── culture\_c.html

│   │   │   │   ├── culture\_d.html

│   │   │   │   ├── culture\_tr.html

│   │   │   │   └── culture\_u.html

│   │   │   ├── datamap.html

│   │   │   ├── organism\_c.html

│   │   │   ├── organism\_card.html

│   │   │   ├── organism\_d.html

│   │   │   ├── organism\_detail.html

│   │   │   ├── organism\_file.html

│   │   │   ├── organism\_image.html

│   │   │   ├── organism\_list.html

│   │   │   ├── organism\_mic.html

│   │   │   └── organism\_u.html

│   │   └── taxonomy

│   │   ├── taxonomy\_c.html

│   │   ├── taxonomy\_card.html

│   │   ├── taxonomy\_d.html

│   │   ├── taxonomy\_detail.html

│   │   ├── taxonomy\_list.html

│   │   └── taxonomy\_u.html

│   ├── templatetags

│   │   ├── \_\_init\_\_.py

│   │   ├── custom\_filters.py

│   │   └── myapp\_extras.py

│   ├── urls.py

│   ├── utils

│   │   ├── data\_visual.py

│   │   └── utils.py

│   └── views.py

├── dscreen

│   ├── \_\_init\_\_.py

│   ├── admin.py

│   ├── apps.py

│   ├── models.py

│   └── views.py

├── environment.yml

├── environment\_prod.yml

├── impdata

│   ├── Data

│   │   ├── ApplicationData\_v03.xlsx

│   │   ├── ApplicationData\_v04.xlsx

│   │   ├── DrugData\_v03.xlsx

│   │   ├── LMIC\_Data\_v06.xlsx

│   │   ├── OrganismData.xlsx

│   │   └── OrganismData\_v01.xlsx

│   ├── DictV03\_K7KrWZM.csv

│   ├── \_\_init\_\_.py

│   ├── a\_upload\_AppUtil.py

│   ├── b\_upload\_dOrganism.py

│   ├── c\_upload\_dDrug.py

│   ├── d\_upload\_Vitek.py

│   ├── e\_upload\_MIC.py

│   ├── f\_upload\_gene.py

│   ├── get\_missingDrug.ps1

│   ├── get\_missingOrgBatch.ps1

│   ├── test\_FK.py

│   └── upload\_OrgDB\_Data.py

├── manage.py

├── requirements.txt

├── setup.cfg

├── setup.py

├── static

│   ├── css

│   │   ├── custom.css

│   │   ├── main.6a646761.css

│   │   ├── overideboot\_size.css

│   │   └── table\_wrap.css

│   ├── images

│   │   ├── app

│   │   │   ├── CO-ADD\_Logo\_White\_NoBackground.png

│   │   │   ├── ...

│   │   └── brand

│   │   ├── ...

│   │   └── logo-fullsize.png

│   └── js

│   ├── ajax\_deleteModal.js

│   ├── ajax\_simpleupload.js

│   ├── call\_jsme\_editor.js

│   ├── childtable.js

│   ├── create\_childtable.js

│   ├── create\_modal.js

│   ├── destroy\_childtable.js

│   ├── editableTable.js

│   ├── editablerow.js

│   ├── filter\_toggle.js

│   ├── formWizard

│   │   └── jquery.steps.min.js

│   ├── initialPivottable.js

│   ├── jquery-3.6.1.min.js

│   ├── js\_utils

│   │   ├── ajax\_submit.js

│   │   ├── custom\_option.js

│   │   ├── dataTables.min.js

│   │   ├── getCookie.js

│   │   ├── saveData.js

│   │   └── sendtoserver.js

│   ├── ketcher

│   │   └── index-bundle.min.js

│   ├── loadModal.js

│   ├── resizable.js

│   ├── search\_organism.js

│   ├── search\_organism\_id.js

│   ├── sorting\_table.js

│   └── vitek\_datamap.js

├── templates

│   ├── base.html

│   ├── base2\_nonav.html

│   └── utils

│   ├── card.html

│   ├── data\_download\_form.html

│   ├── datatable\_filter.html

│   ├── datatable\_general.html

│   ├── datatable\_sm.html

│   ├── django\_form\_lgtable.html

│   ├── drag\_n\_drop

│   │   ├── selectivedrag\_hr.html

│   │   └── selectivedrag\_vt.html

│   ├── editable\_tr.html

│   ├── import\_singlestep.html

│   ├── ketcher.html

│   ├── ketcher\_test.html

│   ├── leftbar.html

│   ├── lefticons.html

│   ├── main\_horizalbar.html

│   ├── message.html

│   ├── messagebox.html

│   ├── miscellaneous

│   │   ├── cancel\_btn.html

│   │   ├── cancel\_btn\_htmx.html

│   │   ├── cancel\_btn\_reloc.html

│   │   ├── create\_a.html

│   │   ├── delete\_btn.html

│   │   ├── entrylog.html

│   │   ├── landbtn\_primary.html

│   │   ├── landbtn\_secondary.html

│   │   ├── preloader.html

│   │   ├── save\_btn.html

│   │   ├── selectAllExp.html

│   │   ├── selectbox.html

│   │   ├── showentries\_badge.html

│   │   ├── update\_btn.html

│   │   └── update\_btn\_htmx.html

│   ├── modal

│   │   ├── delete.html

│   │   ├── modal\_form.html

│   │   └── simpleupload.html

│   ├── navbar.html

│   ├── pagination.html

│   ├── pivotedtable.html

│   ├── pivottable.html

│   ├── row\_editabletable.html

│   ├── scripts.html

│   ├── search\_organism.html

│   ├── search\_organism\_id.html

│   ├── sidebar.html

│   ├── simpleupload.html

│   ├── uploadtable.html

│   └── wizard\_single\_page.html

├── tests

│   ├── \_\_init\_\_.py

│   ├── dev\_tests

│   │   ├── GN\_0544\_Antibio.xlsx

│   │   ├── agg\_Antibio.ipynb

│   │   ├── test\_antibiogram.py

│   │   └── test\_bp.py

│   └── unit\_tests

│   ├── test\_apputil

│   │   ├── \_\_init\_\_.py

│   │   ├── test\_urls.py

│   │   └── test\_views.py

│   ├── test\_ddrug

│   │   └── tests\_api.py

│   └── test\_dorganism

│   └── tests.py

├── tree.txt

├── z\_accessAPI

│   ├── api\_request.py

│   └── api\_request\_with\_credentials.py

├── z\_design

│   └── adjOrgDB\_Schema.pptx

└── z\_docs

├── GetStartwithCoAdd.md

├── InitialiseNewDatabase.md

├── config

│   ├── nginx.conf

│   └── nginx\_django.conf

├── dj42py311\_v01.yaml

├── setup\_deployment.md

└── win10

├── dj42py311\_win10.yaml

└── python\_ldap-3.4.3-cp311-cp311-win\_amd64.whl

85 directories, 419 files