Computer Systems Project (PSI)

Assignment 2 Group 1391-02

Daniel Varela & Guillermo Martín-Coelli Juárez

DJANGO ARCHITECTURE

PROJECT AND APPLICATIONS

First we have a Django project which contains three important things:

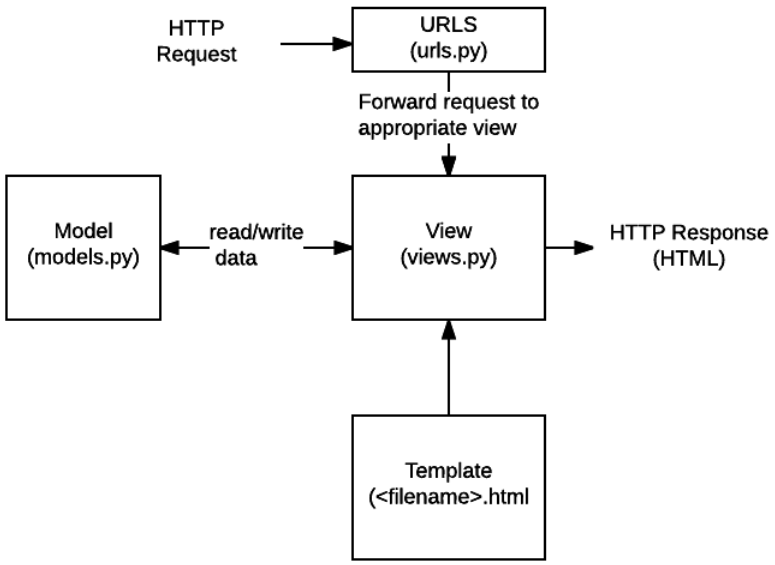
The database (where all the data our application needs to work)
The root URL parser (which controls the applications URLS)
The base HTML templates (where we make the web interface)

In the Django project we then can add different applications, and when we do so, the database will get synchronized with the models created on the application, the project URL parser will get connected with the application URL parser and the project HTML will be the same as the application HTML templates. After that we will have a working Django application inside a Django project.

MODEL-VIEW CONTROLLER

When a request is made to the web server, the first step is to go to the urls.py in the Django project which redirects to urls.py inside the Django application. The urls.py checks the request and decides which view it has to show from views.py. Once this is decided, views.py picks what it needs to fulfill the request from models.py (which accesses a database in the Django project) and the html templates and proceeds to return the HTML with the data to the client.

This is a basic mockup of how the Django architecture works:



MODEL

The Django applications use the database with python objects called models. These specify some basic parameters and the structure of the data in the database. These models are not restricted to one only database, they can use different ones and also re-use others (this is set in the project configuration). Once the database to use is selected, Django will take care of the communication with the database. We just need to write the architecture of the models.

CONTROLLER

First of all, in this part, the root URL parser calls to the application URL parser which then calls to the corresponding view in views.py.

Here, in views.py, we just have to create the view of the data we want to show, we can add filters, different ways of showing the data, join different datas, etc… We do this through an API of basic consulting provided by Django and by accessing the Models in models.py. This allowed us to do for example the challenge of Part 5 from the tutorial where we were asked to make a count of the books that contained a specific word case insensitive. Once we apply these filters and modifications of the data we will request for the view (HTML templates).

VIEW

Here we have the HTML templates where we decide the aspect of the page we will show in HTML language, to make this more interesting we use in this practice Jinja2 which allows us to call Python functions while in HTML to make a dynamic content. These templates will be called from views to be shown in response to a specific URL.

COVERAGE

Here we add a screenshot of out test coverage:

Name	Stmts	Miss	Cover	Missing
catalog/__init__.py	0	0	100%	
catalog/admin.py	24	0	100%	
catalog/apps.py	4	0	100%	
catalog/forms.py	13	0	100%	
catalog/migrations/0001_initial.py	7	0	100%	
catalog/migrations/0002_language.py	4	0	100%	
catalog/migrations/0003_rename_language_language_name.py	4	0	100%	
catalog/migrations/0004_book_language.py	4	0	100%	
catalog/migrations/0005_auto_20210927_1736.py	5	0	100%	
catalog/migrations/0006_auto_20211012_1741.py	6	0	100%	
catalog/migrations/0007_alter_bookinstance_options.py	4	0	100%	
catalog/migrations/0008_alter_bookinstance_options.py	4	0	100%	
catalog/migrations/0009_alter_bookinstance_options.py	4	0	100%	
catalog/migrations/0010_alter_bookinstance_options.py	4	0	100%	
catalog/migrations/0011_alter_bookinstance_options.py	4	0	100%	
catalog/migrations/0012_alter_author_date_of_death.py	4	0	100%	
catalog/migrations/__init__.py	0	0	100%	
catalog/models.py	58	2	97%	60, 109
catalog/urls.py	8	0	100%	
catalog/views.py	85	0	100%	
TOTAL	246	2	99%	