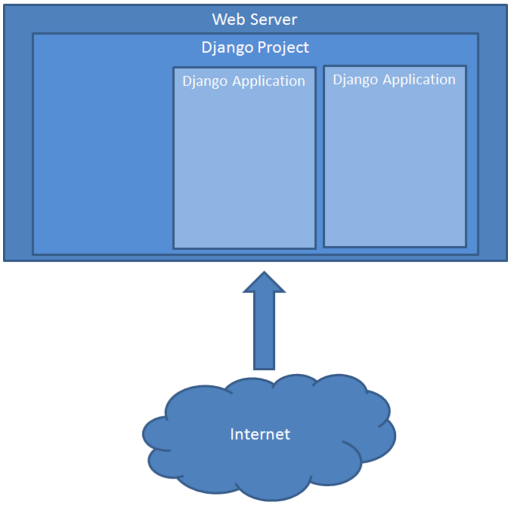
Luna Mancebo Rodríguez
Laura de Paz Carbajo

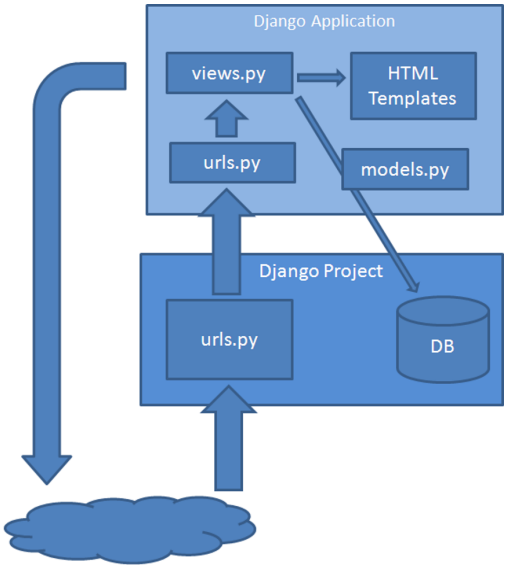
Report Assignment 2

How does Django work?

Following the guideline provided in the Moodle presentation, we are going to explain how Django works, based on the acquired knowledge from Practice 2.



In this first picture we can see how Django is included inside the web server. Django is a Framework that is used as an interpreter between the web server and the Python application. Inside the Django project, we can define as many Django applications as we like. In our case we created the application “catalog”.

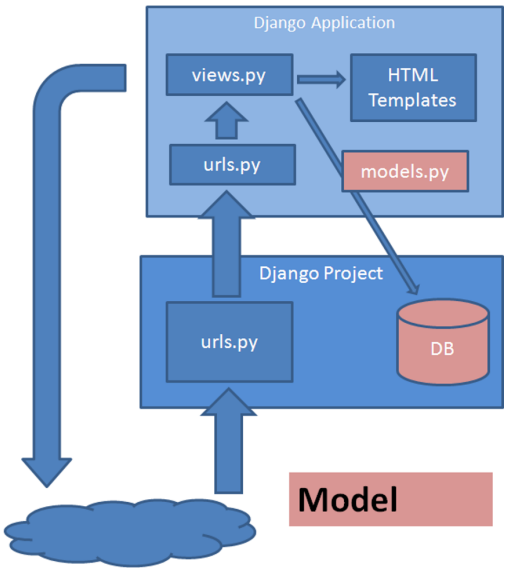


Django follows the MVC pattern, with a slight variation, therefore, it is actually considered an MTV Framework. The design pattern for an MTV is:

M (Model), is the layer to access the database . This layer contains all the information about the data, how to access and validate it, its behaviour and the relations between data.

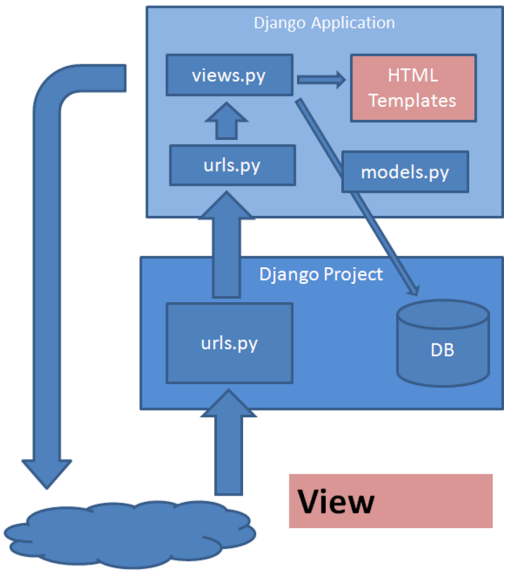
T (Template), is the presentation layer and it contains the decisions related to the presentation (how everything is displayed in a web page or document).

V (View), is the business logic. This layer contains the logic that accesses the model and it relates it to the appropriate template (it’s like the bridge between the model and the templates).

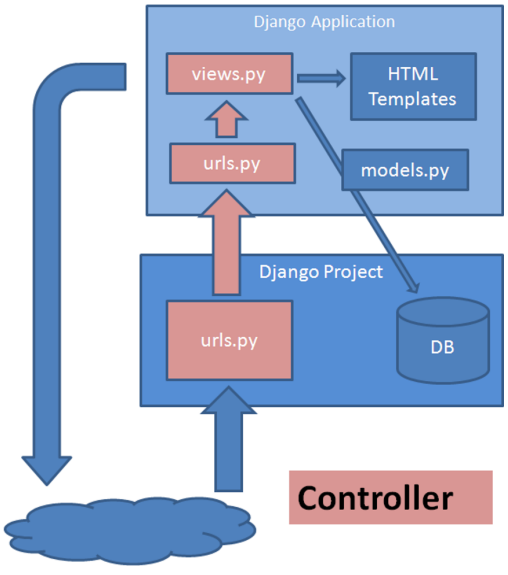


As we just explained, the model is the layer that accesses the database, and contains all the information related to the data.

Django facilitates the process of creating the database, as we only have to define the data as classes in Python. Django itself carries the job of translating these classes into SQL and building the corresponding tables.



The templates are HTML files whose parameters are replaced by data from the model. It is the presentation layer, meaning it's main functionality is to manage how the data is displayed to the user.



When the server receives an HTML request, Django looks in the *urls.py* file for a pattern that matches the request. Following, the request is sent to the *urls.py* file of a certain application, based on the matched pattern. If no match is found an error occurs.

Each URL corresponds with a view defined in *views.py*. In these views we define the variables that will be used in the HTML templates (located in the template directory). In this file (*views.py*) is where we make the queries to the database, and in the templates is where we define how they are shown.

Coverage of the tests

```
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_auto_20210920_2351.py            5      0   100%
catalog/migrations/0003_alter_author_date_of_birth.py    4      0   100%
catalog/migrations/0004_bookinstance_borrower.py         6      0   100%
catalog/migrations/0005_alter_bookinstance_options.py    4      0   100%
catalog/migrations/0006_alter_author_date_of_death.py    4      0   100%
catalog/migrations/__init__.py                           0      0   100%
catalog/models.py                                       56      2    96%   55, 100
catalog/urls.py                                          3      0   100%
catalog/views.py                                        86      0   100%
----------------------------------------------------------------------------------
TOTAL                                                  216      2    99%
```

This is the report of the coverage of the tests generated thanks to the coverage utility. As we can see, the tests cover every line of code in all the files except for models.py. The two lines that are not being tested are 55 and 100, which correspond with the display_genre function in the Book model and the function __str__ of BookInstance, respectively. The aim of these 2 functions is to parse strings so even if the test doesn't cover them, we can be certain that they won't trigger any error.

Our **Heroku** application can be accessed with the following link:

https://safe-citadel-40470.herokuapp.com/