**Django**, a Python framework to create web applications, is based on Model-View-Template (MVT) architecture. **MVT** is a software design pattern for developing a web application. It consists of the following three entities:

1. Model
2. View
3. Template

**Model**

A **Model** is an object that defines the structure of the data in the Django application.

It is responsible for maintaining the entire application’s data for which it provides various mechanisms to add, update, read and delete the data in the database.

**View**

A **View** is a handler function that accepts HTTP requests, processes them, and returns the HTTP response.

It retrieves the necessary data to fulfil the request using Models and renders them on the user interface using Templates.

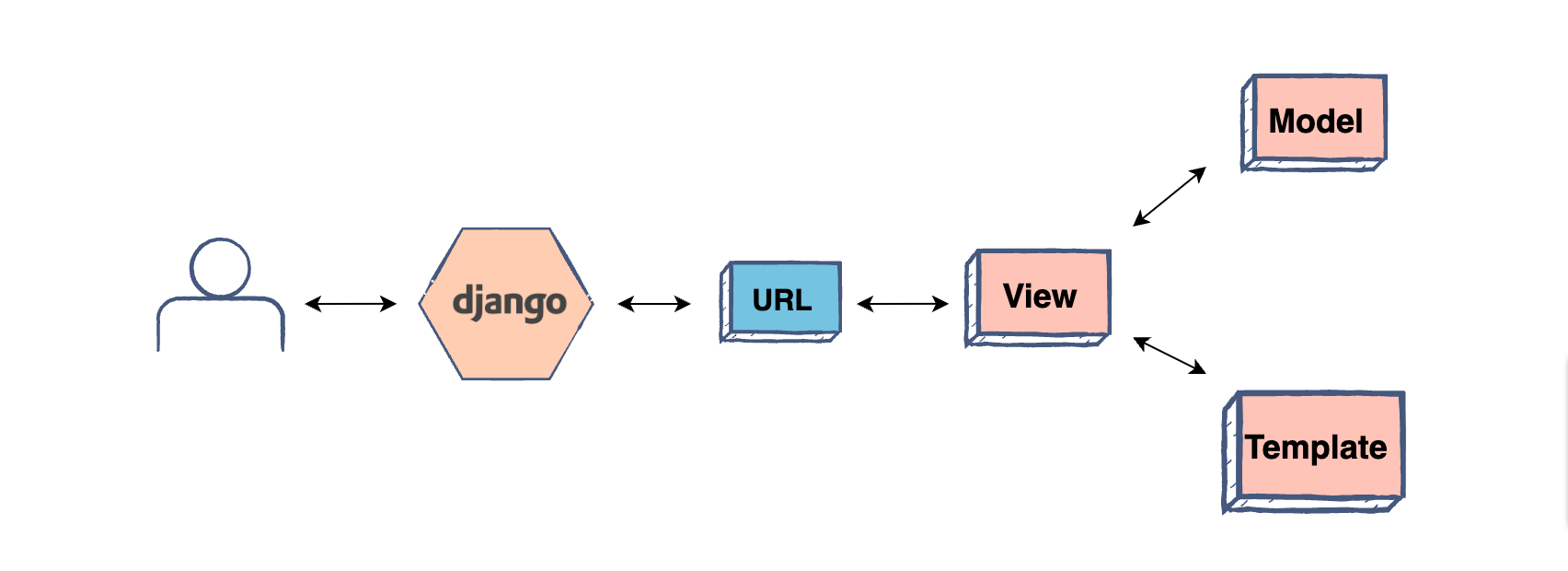
It can also create an HTML page using an HTML template dynamically, and populate it with data fetched from the model.

**Template**

A **Template** is a text file that defines the structure or layout of the user interface. The text file can be any type of file; for example HTML, XML, etc.

It can accept data from the view and render it using jinja syntax.

**Control flow in MVT architecture**

****

* The user interacts with a Django application using a URL that is passed to the MVT architecture. A URL mapper is used to redirect the requests to the appropriate view based on the request URL.
* If an appropriate view is found, it will be invoked.
* The View will interact with the Model and retrieve the necessary data from the database via Model.
* The View will render an appropriate template along with the retrieved data to the user.

**Django Project Structure**

* **django\_project/manage.py**

This file is used basically as a command-line utility and for deploying, debugging, or running our web application.

**runserver**: This command is used to run the server for our web application.

**migration**: This is used for applying the changes done to our models into the database. That is if we make any changes to our database then we use migrate command. This is used the first time we create a database.

**makemigration**: this is done to apply new migrations that have been carried out due to the changes in the database.

* **django\_project/django\_project/\_\_init\_\_.py**

This file remains empty and is present their only to tell that this particular directory(in this case django\_project) is a package.

* **django\_project/django\_project/settings.py**

This file is present for adding all the applications and the middleware application present. Also, it has information about templates and databases. Overall, this is the main file of our Django web application.

* **django\_project/django\_project/urls.py**

This file handles all the URLs of our web application. This file has the lists of all the endpoints that we will have for our website.

**URL**: **Universal Resource Locator** is used to provide the addresses of the resources (like image, website, etc) that are present there on the internet.

* **django\_project/django\_project/wsgi.py**

This file mainly concerns with the WSGI server and is used for deploying our applications on to servers like Apache etc.

WSGI, short for **Web Server Gateway Interface** can be thought of as a specification that describes how the servers interact with web applications.

* **django\_project/django\_project/asgi.py**

In the newer versions of Django, you will also find a file named as asgi.py apart from wsgi.py. ASGI can be considered as a succeeder interface to the WSGI.

**ASGI**, short for **Asynchronous Server Gateway interface** also has the work similar to WSGI but this is better than the previous one as it gives better freedom in Django development. That’s why WSGI is now being increasingly replaced by ASGI.

**Django App Structure**

* **\_init\_.py**

This file has the same functionality just as in the \_init\_.py file in the Django project structure. It remains empty and is present just to indicate that the specific app directory is a package.

* **admin.py**

As the name suggests, this file is used for registering the models into the Django administration.

The models that are present have a superuser/admin who can control the information that is being stored.

* **apps.py**

This file deals with the application configuration of the apps. The default configuration is sufficient enough in most of the cases and hence we won’t be doing anything here in the beginning.

* **models.py**

This file contains the models of our web applications (usually as classes).

Models are basically the blueprints of the database we are using and hence contain the information regarding attributes and the fields etc of the database.

* **views.py**

This file is a crucial one, it contains all the Views(usually as classes). Views.py can be considered as a file that interacts with the client. Views are a user interface for what we see when we render a Django Web application.

* **urls.py**

Just like the project urls.py file, this file handles all the URLs of our web application. This file is just to link the Views in the app with the host web URL. The settings urls.py has the endpoints corresponding to the Views.

* **tests.py**

This file contains the code that contains different test cases for the application. It is used to test the working of the application.

We won’t be working on this file in the beginning and hence it is going to be empty as of now.

**Django Template Tags and Filters**

The Django template language has three ways of controlling what gets rendered: **values**, **tags**, and **filters**. Everything you put into a template that is not one of these three gets rendered as you have written it. In this tutorial, you’ll go over the three main parts of the template language:

1. **Interpreted data**, which you note by double braces, **{{ value }}**
2. **Tags**, which you note by braces and percent signs, **{% tag\_name %}**
3. **Filters**, which modify interpreted data and you apply with the pipe operator (|), like in **{{ value | filter }}**

**Useful Commands**

pwd # Tell you your present working directory in terminal

python3 -m venv venv\_name

source venv\_name/bin/activate

(venv\_name)

pip install django==3.2

django-admin --version

django-admin startproject project\_name or location/project\_name

cd project\_name

python manage.py startapp app\_name

python manage.py runserver

python manage.py makemigrations

python manage.py migrate

python manage.py createsuperuser

python manage.py runserver 0.0.0.0:8000

We might need run below commands

sudo apt-get install python-dev python3-dev

sudo apt-get install libmysqlclient-dev

pip install MySQL-python

pip install pymysql

pip install mysqlclient