**API**

**API**

Application program Interface

Literarilly, A software that allows applications to communicate to each other

It is a messenger that delivers your request to the provider and it also sends response

**How API works**

Restaurent have a waiter

A customer comes to order food

The waiter takes the order and the goes to the kitchen

To fetch the order

Customer – Service

Waiter - API

Kitchen - AutoService

**Types of API**

1. REST – Representational State Transfer
2. SOAP –Simple Object Access Protocol
3. GRAPHQL

We have more. But the common one is **REST**

**How to develop API with Django**

**Django itself is used for creating those app that you use templating to get html. But u can have the frontend independently (using REACT framework) then integrating it with the backend.**

**There should be a standard format of communication. This could be XML, JSON. But the mostly used is JSON(Javascript Object Notation)**

**REST framework is built on django. It is used for building web API.**

**It provides us with serialization. It provides lots of functionalities such that you wont write lots of codes. It is a tool that makes life easier.**

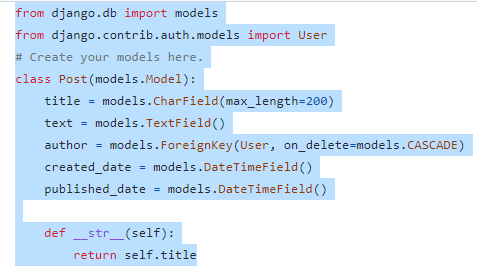
**NB:**

**We create a simple django project first:**

1. **Create a repo called “DJANGO\_REST”**
2. **Clone this repo**
3. **Create virtual environment “venv”**
4. **We install dependencies:**

* **Django**
* **Django\_extensions**
* **Python-decouple**

1. **We create django project: “mysite”**
2. **We create django\_app: ”blog”**
3. **Now we create our models: “Post” as follows:**

****

1. **Next, we run the followingmigrations:**

* **Python manage.py migrate**
* **Python manage.py makemigrations**
* **Python manage.py migrate**

1. **We create our “templates” folder inside the “blog” app**
2. **Next we introduce rest framework**

**We use the REST documentation throughouht**

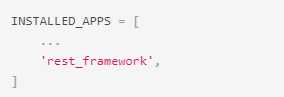
1. **We install as follows**

****

1. **Next we add the REST framework to installed apps**

* **We go to settings.py**
* **Under INSTALLED\_APPS , we include:**

**‘rest\_framework’**

****

1. If you're intending to use the browsable API you'll probably also want to add REST **framework's login and logout views**. Add the following to your root urls.py file.



1. Next we create **Rest API** documentation under **settings.py** file as follows:

[[1]](#footnote-2)

1. Now we need to work on our model.

Notice that when we query **Post** field or we create an instance for Post field, the response is a queryset; an object that is being returned. So we need to deserialize the object. That is: all the information in the object, I want to be able to send them as json file. This is the idea of serialization comes in.

Serialization helps to interprete in a queryset and helps in deserializing in a Rest framework

1. We create a new file inside the “Blog” app and call it “serializers.py”
2. We include the following in **serializers.py**

From **rest\_framework** import **serializers**

From **.models** import **Post**

**Class PostSerializer(serializers.ModelSerializer):**

**Class Meta:**

Model = Post

Fields = ‘\_\_all\_\_’

1. Next we create the **viewset** to describe **the behavior of the API** as follows:

NB: There are different ways of doing this.

From **rest\_framework** import viewsets

From .models import Post

From **.serializers** import **PostSerializer**

Class **PostViewSet**(viewsets, ModelViewSet):

Queryset = Post.objects.all() **# It query the post Table and return all objects**

Serializer\_class = PostSerializer **#It serializes the response of queryset in a JSON format**

**# with that we can create, get post , update, delete and get details of a particular instance**

NB: To do the above, we need to create a **url** for it, to do this we use **router: it is like creating a path to access a view function. Rest router is easy cuz we only use one route to do many things**

1. We go to the blog apps **urls.py, and we add the following:**

**From rest\_framework.routers import DefaultRouter**

**From .views import PostViewSet**

**# Now we register a route**

**Router = DefaultRouter()**

**#We register the route**

**router.register(r’posts’, PostViewSet, basename=’posts’)**

**urlpatterns = [] + router.urls**

**#127.0.0.1:8007/posts**

**NB: There is a tool generally used to make django Rest API create, update, delete: “POSTMAN”**

1. **We go to https://**[**www.postman.com**](http://www.postman.com) **, the we download postman**

Or type **download postman** on google search bar

1. Now, we got to our web browser and search for: **127.0.0.1:8007/posts**
2. **We log in then create instance of Post as follows:**

**Title: Free Plan**

**Body: This is a free plan**

**NB: when we click post, we see our JSON format. It returns “I”d automatically.**

**When we click the post, it came to urls.py “registered route”.**

**Then it comes to view with the info and deserializes it,then create the following underground:**

**Payload = {**

**‘title’: ‘My first blog post’,**

**‘body’: ‘This is my first blog post’**

**}**

**Post.objects.create(title=payload[‘title’], body = ayload[‘body’])**

**Then it serializes the above and return the JSON format as we have seen**

1. **Now after installing out postman, we can open it and explore:**

* **We insert the word “Posts” under new collection**
* **The we select http verb:**
  + **GET: To make a request, and u get a response**
  + **POST: you put in data**
  + **PUT: to update an existing data**
  + **PATCH: same as PUT**
  + **DELETE : to delete an instance from database**
* **We select GET**
* **Then we include our endpoint; that is: 127.0.0.1:8007/posts**
* **Then we click send. After which we get response in JSON format.**
* **NB: To get a single info: we could specify as follows: 127.0.0.1:8007/posts/1**

**NB: Django is a python framework, and Django Rest is a tool/framework built on django**

**NB:** **Thunder Client** is the name of the **vscode extension** for running REST framework – we search for it n **vs-code extension** then we install

1. Next, we select **POST, and we put in our data as follows:**

**NB: We select json format.**

**‘title’: ‘Second Post’**

**‘body’: ‘This is the second post’**

1. **Next we select PUT: to update e.g id3 as follows:**

* **127.0.0.1:8007/posts/3**
* We edit, then we send
* When we go back selecting get, we observe that from the list, the **id3** has been updated

1. Next we select delete: e.g **id 2** as follows:

* **127.0.0.1:8007/posts/3**
* We click send, it return 204 response: i.e empty
* When we go back selecting get, we observe that from the list, the id3 has been deleted

1. Change the **.DjangoModelPermisssionsORAnnonReeadOnly** to **.AllowAny**

   **Notethat: this allow any developer have access to the API** [↑](#footnote-ref-2)