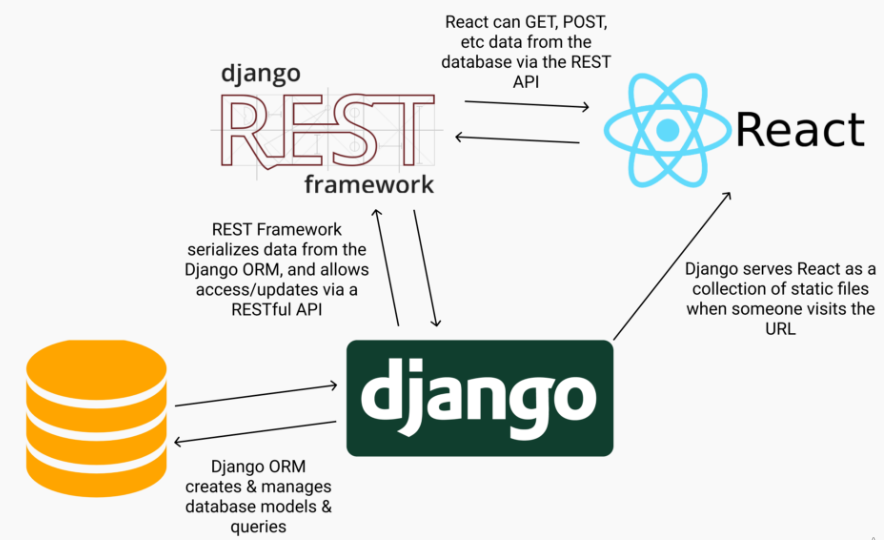
To-do list to create a REST API in Django

* Set up Django
* Create a model in the database that the Django ORM will manage
* Set up the Django REST Framework
* Serialize the model from step 2
* Create the URI endpoints to view the serialized data

There are a few key options for a REST API request:

* GET — The most common option, returns some data from the API based on the endpoint you visit and any parameters you provide
* POST — Creates a new record that gets appended to the database
* PUT — Looks for a record at the given URI you provide. If it exists, update the existing record. If not, create a new record
* DELETE — Deletes the record at the given URI
* PATCH — Update individual fields of a record

Typically, an API is a window into a database. The API backend handles querying the database and formatting the response. What you receive is a static response, usually in JSON format, of whatever resource you requested.



Open Conda Promt

* Virtual Environment Setup

pyenv virtualenv django-rest

pyenv local django-rest

* Install Django

pip install django

* StartProject

django-admin startproject mysite --> project folder name is mysite

cd mysite/ --> go to created folder and check all files are present (using dir command)

python manage.py runserver --> to check if project is good to go on local server (Open browser and go to http://127.0.0.1:8000/)

* Create an app ( This is for api or anything you are creating inside django)

python manage.py startapp myapi --> here myapi is app name .

You can set to anything you want

* Register the myapi app with the mysite project

edit mysite/settings.py

INSTALLED\_APPS = [

'myapi', --> whatever app name you created.

... # Leave all the other INSTALLED\_APPS as it is.

]

* Migrate the database

python manage.py migrate

* Create Super User

python manage.py createsuperuser

Remember to save name and password you are entering after above command. Will be useful to see admin interface.

* Verify that it works. Start up the Django server:

python manage.py runserver

Go to http://127.0.0.1:8000/admin --> add ur username and pwd from above step.

* Create a model in the database that Django ORM will manage

Open myapi/models.py

from django.db import models

class Hero(models.Model):

name = models.CharField(max\_length=60)

alias = models.CharField(max\_length=60)

def \_\_str\_\_(self):

return self.name

* + - * name and alias are character fields where we can store strings.
      * \_\_str\_\_ method just tells Django what to print when it needs to print out an instance of the Hero model.
* Make migrations

python manage.py makemigrations

python manage.py migrate

* Register Hero Model created with the admin site

Open myapi/admin.py

from django.contrib import admin

from .models import Hero

admin.site.register(Hero)

* Check Hero Model created at django admin interface using your login credentials:

python manage.py runserver

* Create some new heroes at dashboard at ur screen .

Click “Add.” Then, make your heroes!

* Set up Django REST Framework

pip install djangorestframework

* Tell Django that we installed the REST Framework in mysite/settings.py:

INSTALLED\_APPS = [

# All your installed apps stay the same

...

'rest\_framework',

]

* Serialize the Hero model

create a new file — myapi/serializers.py and in that :

Import the Hero model

Import the REST Framework serializer

Create a new class that links the Hero with its serializer

# serializers.py

from rest\_framework import serializers

from .models import Hero

class HeroSerializer(serializers.HyperlinkedModelSerializer):

class Meta:

model = Hero

fields = ('name', 'alias')

* Display the data

Views -- We need to render the different heroes in JSON format.

To do so, we need to:

Query the database for all heroes

Pass that database queryset into the serializer we just created, so that it gets converted into JSON and rendered.

* myapi/views.py

from rest\_framework import viewsets

from .serializers import HeroSerializer

from .models import Hero

class HeroViewSet(viewsets.ModelViewSet):

queryset = Hero.objects.all().order\_by('name')

serializer\_class = HeroSerializer

* Site URLs

Point a URL at the viewset we just created

# mysite/urls.py

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('myapi.urls')),

]

* API URLs

#myapi/urls.py:

from django.urls import include, path

from rest\_framework import routers

from . import views

router = routers.DefaultRouter()

router.register(r'heroes', views.HeroViewSet)

# Wire up our API using automatic URL routing.

# Additionally, we include login URLs for the browsable API.

urlpatterns = [

path('', include(router.urls)),

path('api-auth/', include('rest\_framework.urls', namespace='rest\_framework'))

]

* Test it out!

python manage.py runserver