First We Need To Install Django And Django Rest Framework:

- Run The Command in Venv: **pip install django**
- Run The Command in Venv: **pip install djangorestframework**

****

Then To Start The First Project:

- Run The Command: **django-admin startproject first_api .**

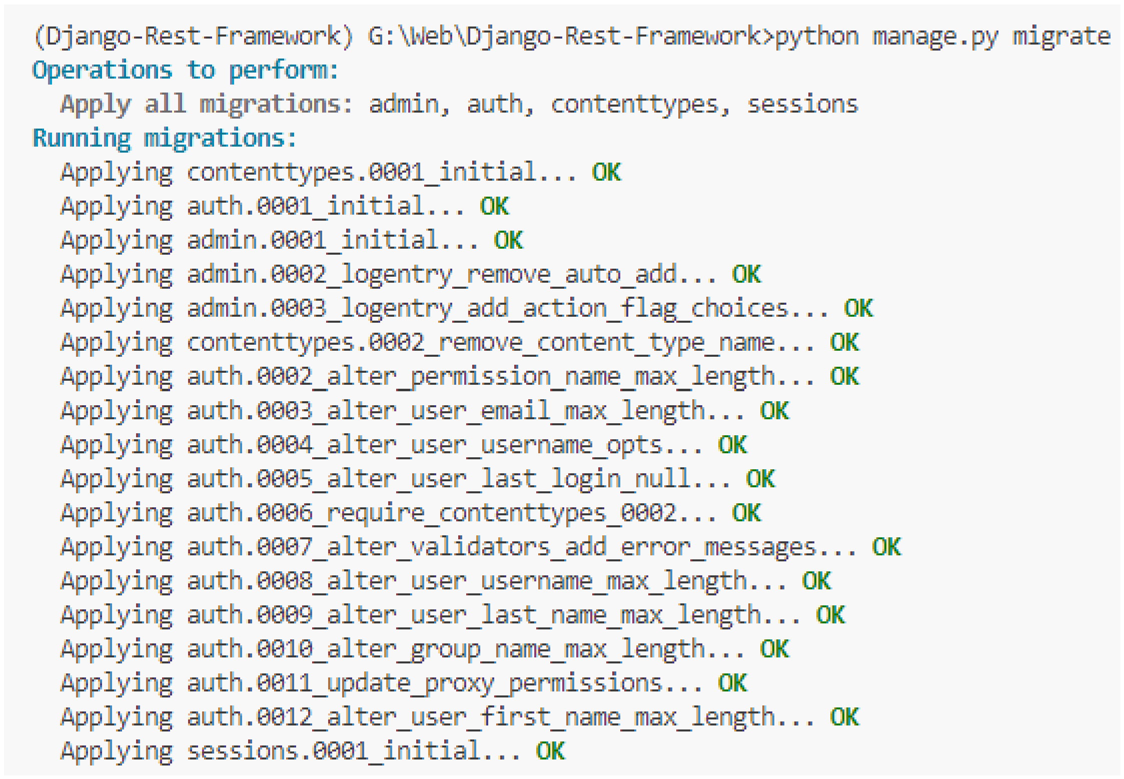
****

To Run The Server, We Run Command: **python manage.py runserver**.

- Note: To Run On Different Port: **python manage.py runserver 127.0.0.1:5001**

****

To Avoid Any Errors, We Run Command: **python manage.py migrate**

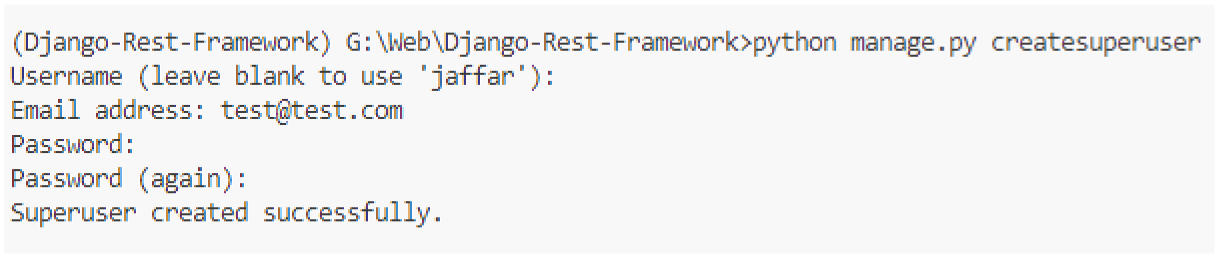


```
(Django-Rest-Framework) G:\Web\Django-Rest-Framework>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

****

To Create Super User For Our Admin Page:

- Run The Command: **python manage.py createsuperuser**
- Then After Creating Super User, We Can Go to: **http://127.0.0.1:5001/admin**



```
(Django-Rest-Framework) G:\Web\Django-Rest-Framework>python manage.py createsuperuser
Username (leave blank to use 'jaffar'):
Email address: test@test.com
Password:
Password (again):
Superuser created successfully.
```

****

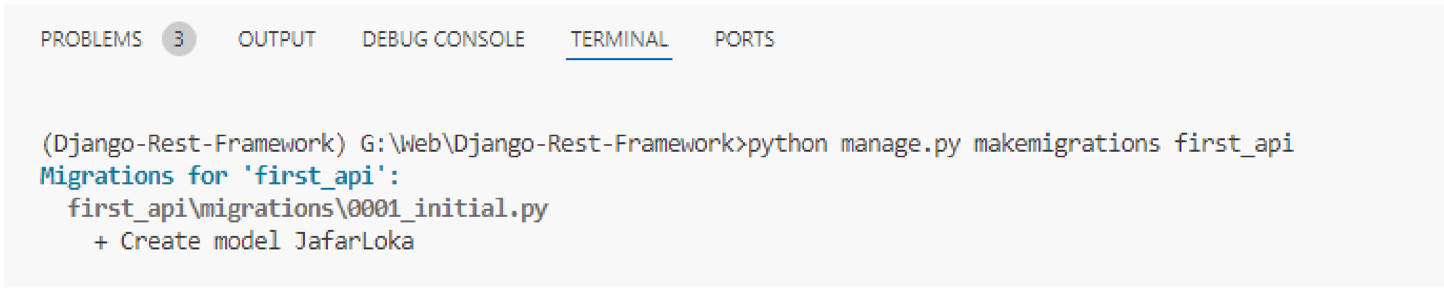
To Create Models:

- First Create models.py-File Inside The Project That We Created Directory.
- Then, Import: **from django.db import models**

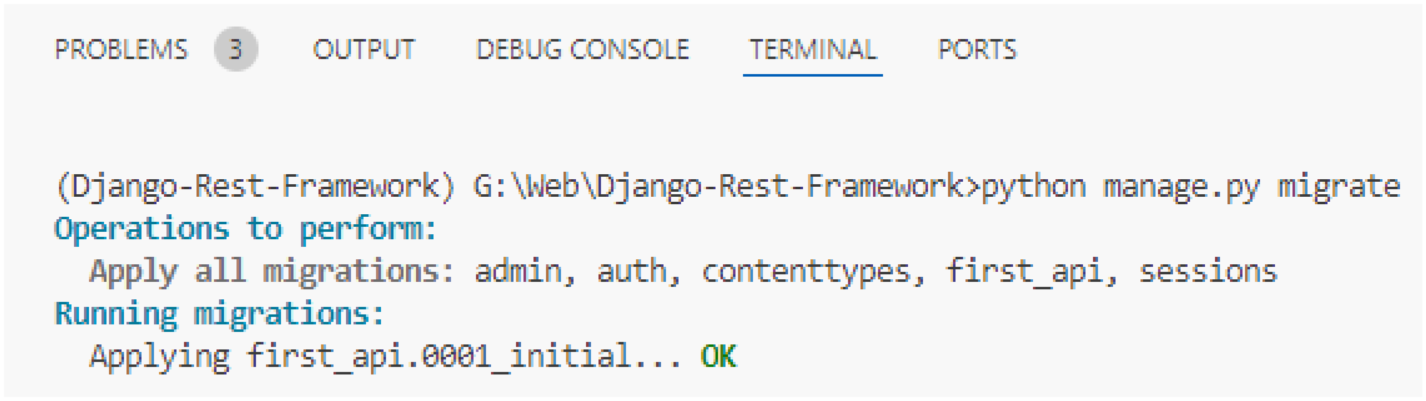
****

To Make Our Migrations From Our Models:

- First We Set Our App Inside The **settings.py**:
- Then We Run The Command: **python manage.py makemigrations app_name**
    - Command: **python manage.py makemigrations first_api**
- Then We Run: **python manage.py migrate**



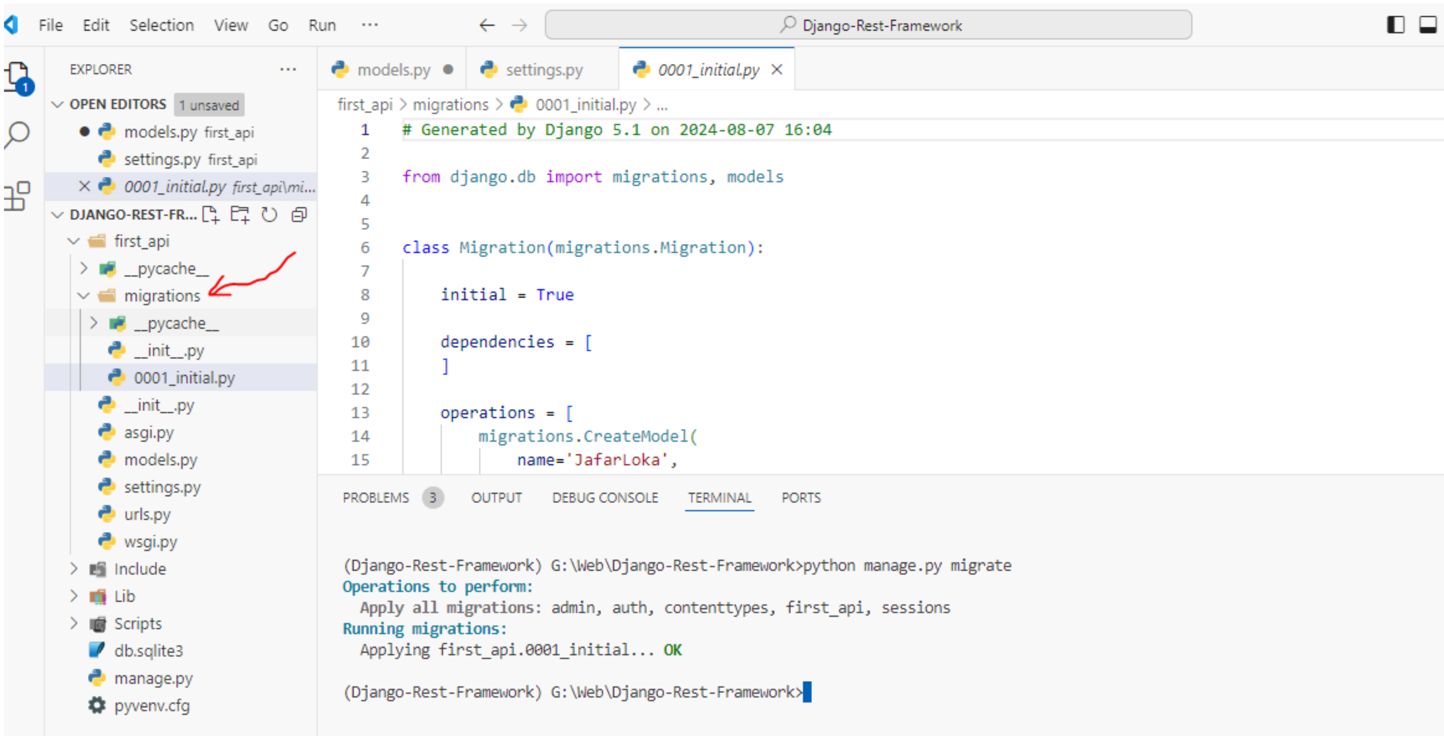
```
(Django-Rest-Framework) G:\Web\Django-Rest-Framework>python manage.py makemigrations first_api
Migrations for 'first_api':
  first_api\migrations\0001_initial.py
    + Create model JafarLoka
```



```
(Django-Rest-Framework) G:\Web\Django-Rest-Framework>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, first_api, sessions
Running migrations:
  Applying first_api.0001_initial... OK
```

****

Note: After We Run makemigrations New Directory Will Be Created, **migrations-Folder**:



```python
# Generated by Django 5.1 on 2024-08-07 16:04

from django.db import migrations, models


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='JafarLoka',
```

```
(Django-Rest-Framework) G:\Web\Django-Rest-Framework>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, first_api, sessions
Running migrations:
  Applying first_api.0001_initial... OK

(Django-Rest-Framework) G:\Web\Django-Rest-Framework>
```

****

Then To Display Our Models Inside The **admin-panel**, we:

- Create New File: **admin.py** Inside Our App

Then We Set Inside admin.py:

```python
from django.contrib import admin
from .models import JafarLoka

admin.site.register(JafarLoka)
```

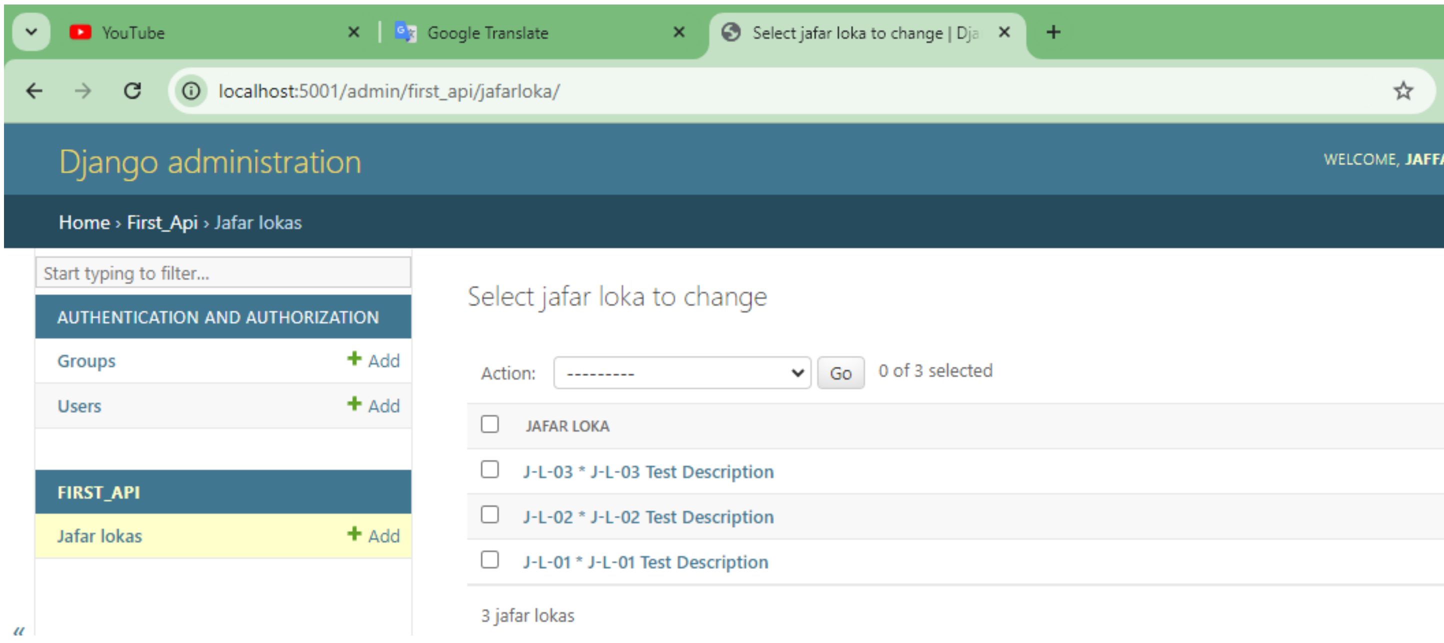
Then Re-Run The Server Again (Only If The Model Not Appear In Admin Panel).

****

If We Want To Change The Format Of Model Representation Inside The Admin-Panel:

- Define The __str__-method
- Return The Representation That We Want:

```
def __str__(self):
        return self.name + ' * ' + self.description;
```



YouTube
Google Translate
Select jafar loka to change | Dja
localhost:5001/admin/first_api/jafarloka/
Django administration
WELCOME, JAFF
Home › First_Api › Jafar lokas
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups
Add
Users
Add
FIRST_API
Jafar lokas
Add
Action:
Go

Select jafar loka to change

0 of 3 selected

JAFAR LOKA
J-L-03 * J-L-03 Test Description
J-L-02 * J-L-02 Test Description
J-L-01 * J-L-01 Test Description

3 jafar lokas

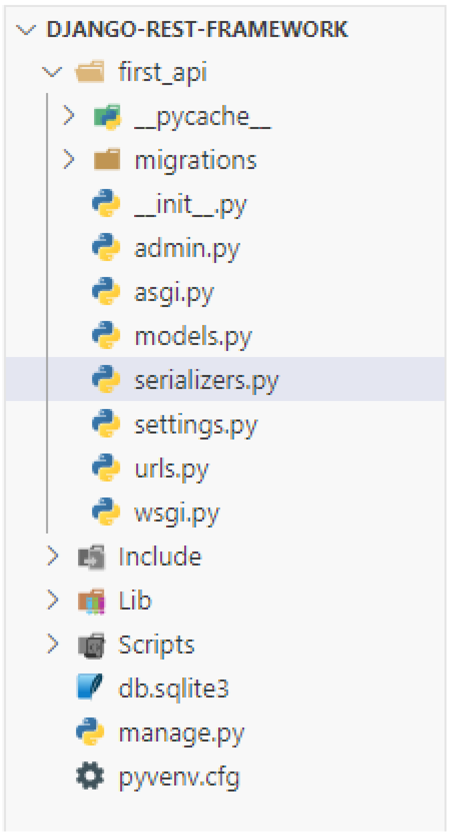
****

To Register The Django Rest Framework, We Add This Line:

```
INSTALLED_APPS = [
    'rest_framework', // This Line Here
    'first_api',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```
****

Then Inside Our App We Create serializers.py-File.

This File Describe How Objects Converted To Json Objects.



****

To Register Our Model Serializer We Must Do:

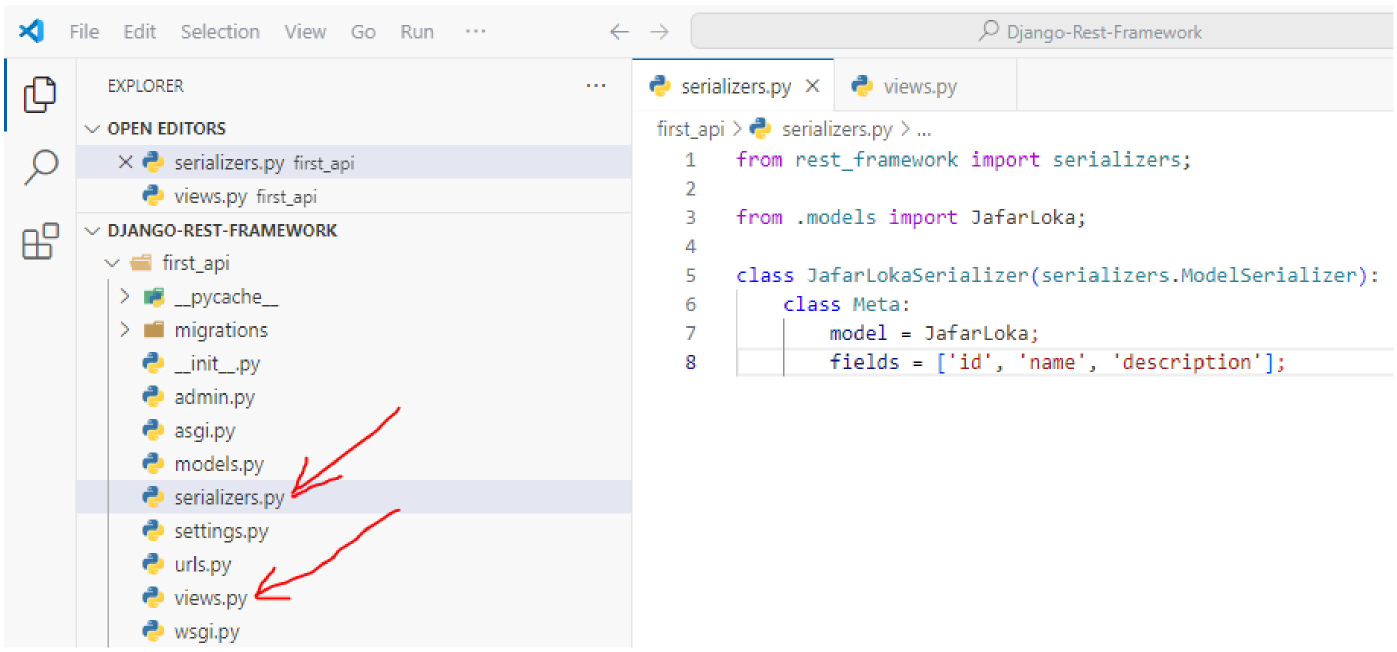
```
from rest_framework import serializers;
from .models import JafarLoka;

class JafarLokaSerializer(serializers.ModelSerializer):
    class Meta:
        model = JafarLoka;
        fields = ['id', 'name', 'description'];
```

Where The:

- The **model** is The Model That We Want To Serialize It.
- The **fields** Is The Fields Data That We Want To Return When Displaying Data.

Then For Our App, We Must create File Name: **views.py**



```
from rest_framework import serializers;

from .models import JafarLoka;

class JafarLokaSerializer(serializers.ModelSerializer):
    class Meta:
        model = JafarLoka;
        fields = ['id', 'name', 'description'];
```

****

The File: **views.py** is where We Add All Our End-Points.

****

Inside The views.py We Import This Modules, And Define Our End-Point Only:

Note 1: We Must Set **safe=False**, If We Want Our Object To Display The Data On The Browser.

```
from django.http import JsonResponse;
from .models import JafarLoka;
from .serializers import JafarLokaSerializer;


def jafar_loka_list(request):
    # Get All Jafar Loka List.
    # Serialize Them.
    # Return The Serialized Data As JSON.
    data = JafarLoka.objects.all();
    serialized_data = JafarLokaSerializer(data, many=True);
    return JsonResponse(serialized_data.data, safe=False);
```
****

In This Way, We Return The Serializer Data As Dict, Without Set **safe=False**:

```
return JsonResponse({'jafar_loka_data': serialized_data.data});
```
****

To Use HTTP-VERB Methods Using Django Rest Framework, We Can Use Decorators:

```
from rest_framework.decorators import api_view;

@api_view(['GET'])
def jafar_loka_list(request):
    # Get All Jafar Loka List.
    # Serialize Them.
    # Return The Serialized Data As JSON.
    data = JafarLoka.objects.all();
    serialized_data = JafarLokaSerializer(data, many=True);
    return JsonResponse({'jafar_loka_data': serialized_data.data});
```
****

Note 1: For Post Data We Need To Add / To The URL In The Client Side.

****

```
from rest_framework.decorators import api_view;
from rest_framework.response import Response;
from rest_framework import status;

@api_view(['GET', 'POST'])
def jafar_loka_list(request):
    # Get All Jafar Loka List.
    # Serialize Them.
    # Return The Serialized Data As JSON.
    if request.method == 'GET':
        data = JafarLoka.objects.all();
        serialized_data = JafarLokaSerializer(data, many=True);
        return JsonResponse({'jafar_loka_data': serialized_data.data});
    elif request.method == 'POST':
        serialized_data = JafarLokaSerializer(data=request.data);
        if serialized_data.is_valid():
            serialized_data.save();
            return Response(serialized_data.data,
status=status.HTTP_201_CREATED);
```

****

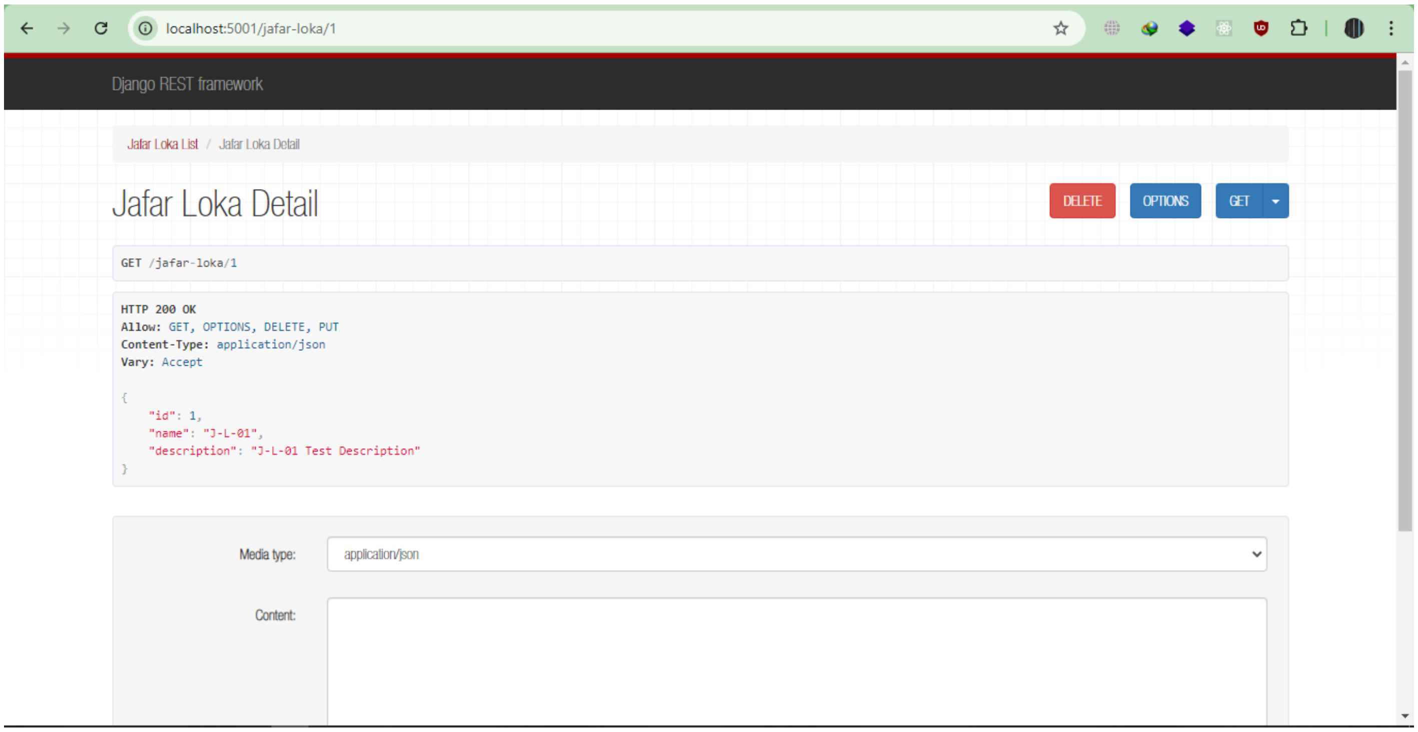
To Get The Data By Id:

Note 1: In This Way If No Object Found It Will Throw Error.

```
try:
        JafarLoka.objects.get(pk=id);
except JafarLoka.DoesNotExist:
        return Response(status=status.HTTP_404_NOT_FOUND);
```

****

If We Use Response Of Django Rest Framework, Then We Can Access To This Page To Get The Details:



Jafar Loka Detail

```
GET /jafar-loka/1
```

```
HTTP 200 OK
Allow: GET, OPTIONS, DELETE, PUT
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "name": "J-L-01",
    "description": "J-L-01 Test Description"
}
```

****

The Right Way To Return The Data In Formatted Way:

```
data = JafarLoka.objects.all();

serialized_data = JafarLokaSerializer(data, many=True);

return Response(serialized_data.data);
```
****

To Solve The URL Issues Of Formatting The Data Like:

- Ex 1: http://localhost:5001/jafar-loka.json
- Ex2: http://localhost:5001/jafar-loka.xml

We Can Use URL Patterns Formatter:

```
from rest_framework.urlpatterns import format_suffix_patterns;


urlpatterns = [
    path('admin/', admin.site.urls),
    path('jafar-loka/', views.jafar_loka_list),
    path('jafar-loka/<int:id>', views.jafar_loka_detail),
]


urlpatterns = format_suffix_patterns(urlpatterns);
```

And In Each Method, We Add **format=None-Parameter** In The views.py-File:

```
def jafar_loka_list(request, format=None):
```

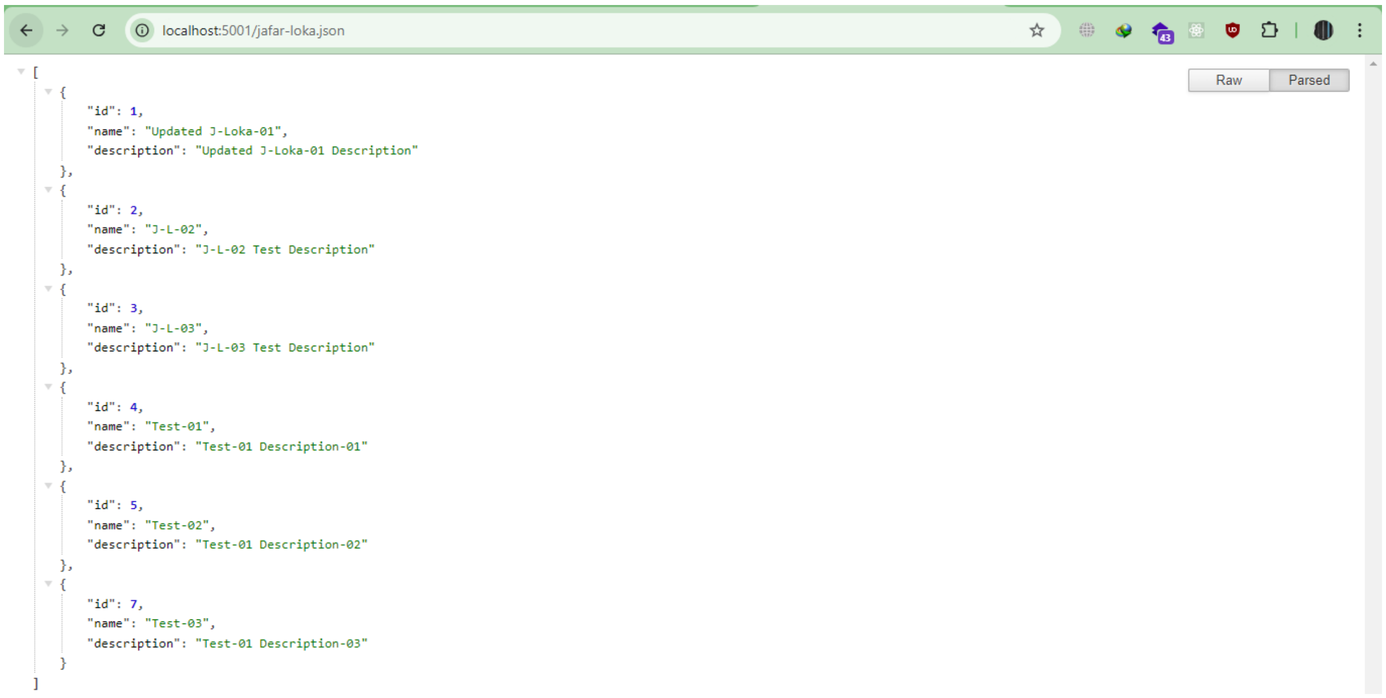
... Here We Define The Body ...

```
def jafar_loka_detail(request, id: int, format=None):
```

... Here We Define The Body ...

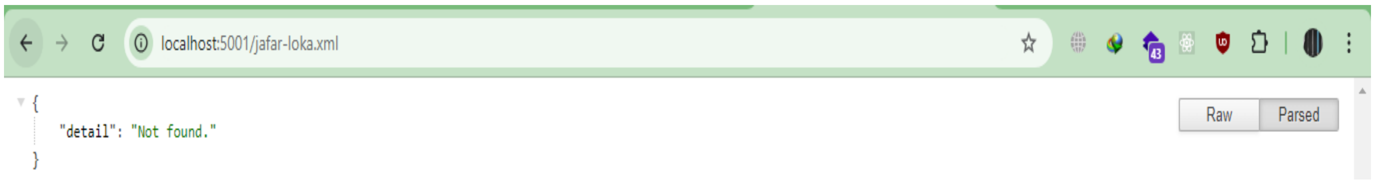
After That These URLS Are Valid:

- Ex1: http://localhost:5001/jafar-loka.json
- Ex2: http://localhost:5001/jafar-loka.xml


localhost:5001/jafar-loka.json

Raw Parsed

```
[
    {
        "id": 1,
        "name": "Updated J-Loka-01",
        "description": "Updated J-Loka-01 Description"
    },
    {
        "id": 2,
        "name": "J-L-02",
        "description": "J-L-02 Test Description"
    },
    {
        "id": 3,
        "name": "J-L-03",
        "description": "J-L-03 Test Description"
    },
    {
        "id": 4,
        "name": "Test-01",
        "description": "Test-01 Description-01"
    },
    {
        "id": 5,
        "name": "Test-02",
        "description": "Test-01 Description-02"
    },
    {
        "id": 7,
        "name": "Test-03",
        "description": "Test-01 Description-03"
    }
]
```


localhost:5001/jafar-loka.xml

Raw Parsed

```
{
    "detail": "Not found."
}
```

****