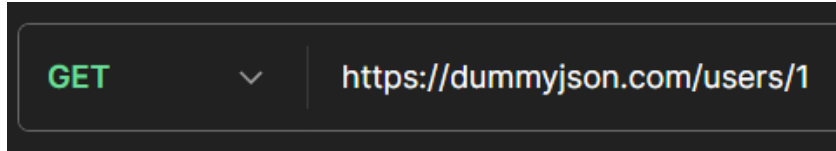
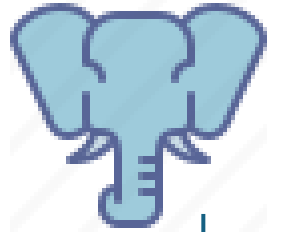


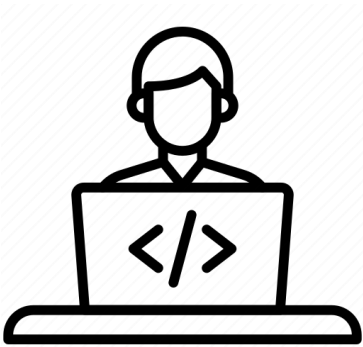


Ciclo de Requests



```
from rest_framework import serializers
from .models import MockUser

class MockUserSerializer(serializers.ModelSerializer):
    class Meta:
        model = MockUser
        fields = ['id', 'username', 'email']
```



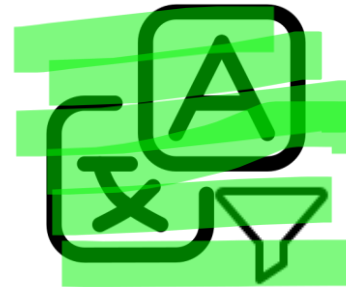
request



endpoint
urls



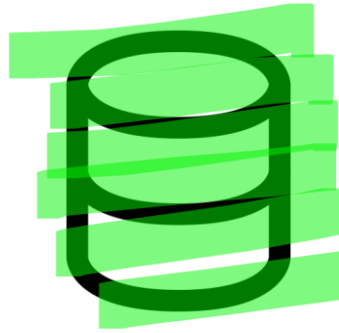
viewsets



Serializers
JSON=Django



models



database



```
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import MockUserViewSet

router = DefaultRouter()
router.register(r'mock-users', MockUserViewSet)

urlpatterns = [
    path('', include(router.urls)),
]
```

```
class MockUserViewSet(viewsets.ModelViewSet):
    queryset = MockUser.objects.all()
    serializer_class = MockUserSerializer
```

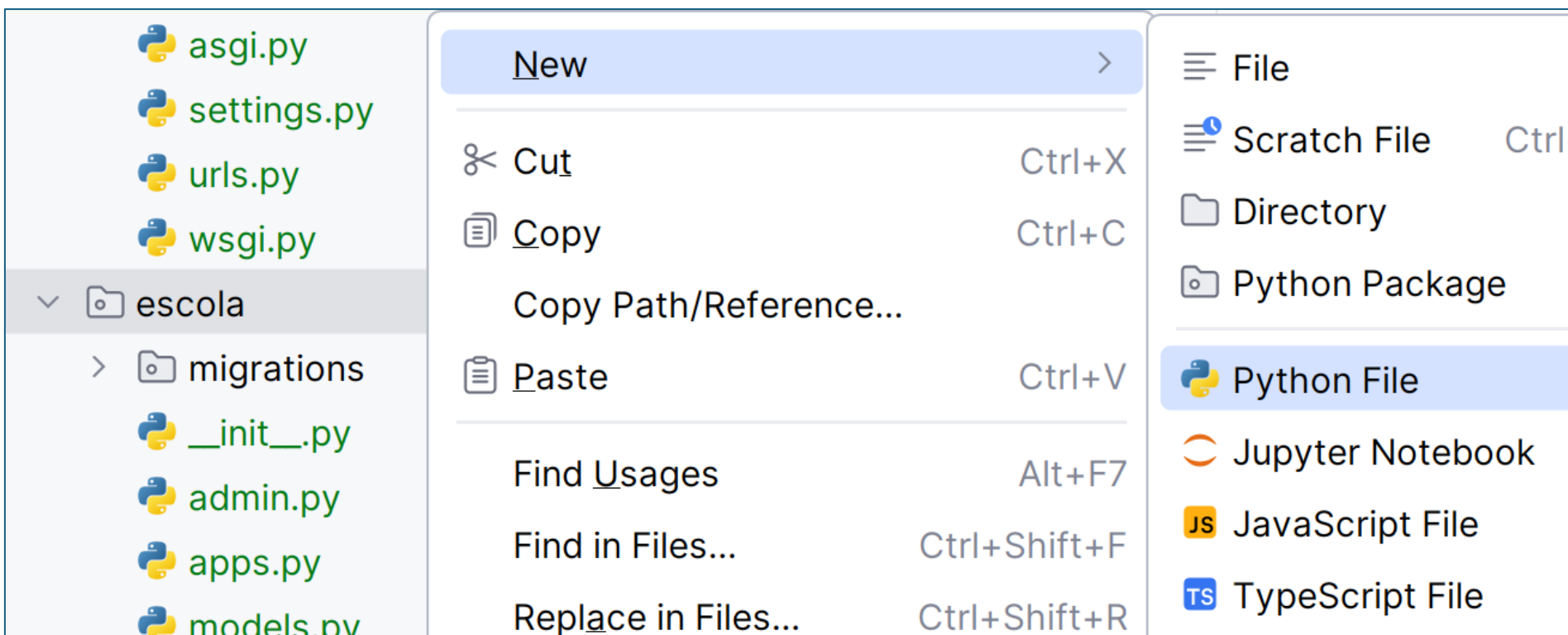
```
from django.db import models

class MockUser(models.Model):
    username = models.CharField(max_length=100)
    email = models.EmailField(unique=True)

    def __str__(self):
        return self.username
```

Urls = arquivo que registra suas urls pros seus viewsets.

1) Vamos criar um novo arquivo python chamado 'urls' dentro da sua aplicação.



```

1  from django.urls import path, include          <- importamos os modelos necessários
2  from rest_framework.routers import DefaultRouter<-
3  from escola.views import ClientViewSet, ProductViewSet, EmployeeViewSet, SaleViewSet
4
5  router = DefaultRouter() <- utilizamos o router pra registrar urls
6  router.register(prefix: r'clients', ClientViewSet) <- 'ligamos' registro url com viewset
7  router.register(prefix: r'products', ProductViewSet)
8  router.register(prefix: r'employees', EmployeeViewSet)
9  router.register(prefix: r'sales', SaleViewSet)
10
11 urlpatterns = [
12     path('', include(router.urls)),
13 ]
    <-agora pra onde vai esse router?

```

Obs. Até agora exemplo de url tá assim:
www.localhost:8000/{r} /clients



No projetão, acesse o arquivo urls.py e faça a alteração pra incluir as urls que criamos

```
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

Estamos dizendo que o path 'caminho' pro nosso router que criamos será limpo

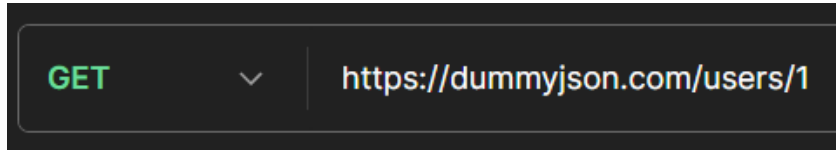
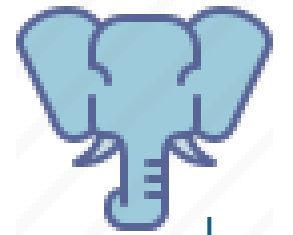
Então, o exemplo de uma url seria:
www.localhost:8000/clients

```
✓ urlpatterns = [
    path('', include('escola.urls')),
    path('admin/', admin.site.urls),
    path('api-auth/', include('rest_framework.urls')),
]
```

Vamos testar?

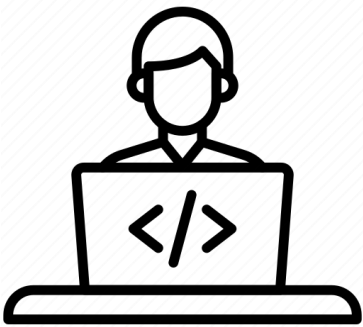


Ciclo de Requests



```
from rest_framework import serializers
from .models import MockUser

class MockUserSerializer(serializers.ModelSerializer):
    class Meta:
        model = MockUser
        fields = ['id', 'username', 'email']
```



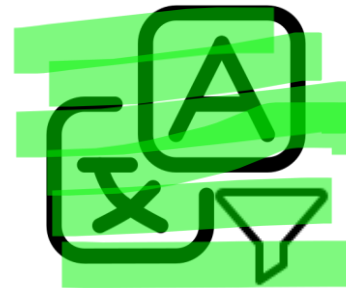
request



endpoint
urls



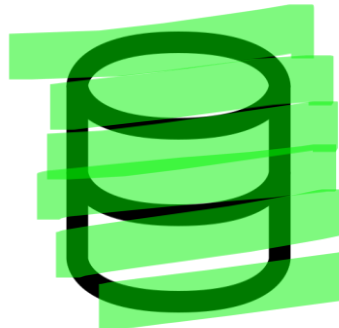
viewsets



Serializers
JSON=Django



models



database

```
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import MockUserViewSet

router = DefaultRouter()
router.register(r'mock-users', MockUserViewSet)

urlpatterns = [
    path('', include(router.urls)),
]
```

```
class MockUserViewSet(viewsets.ModelViewSet):
    queryset = MockUser.objects.all()
    serializer_class = MockUserSerializer
```

```
from django.db import models

class MockUser(models.Model):
    username = models.CharField(max_length=100)
    email = models.EmailField(unique=True)

    def __str__(self):
        return self.username
```



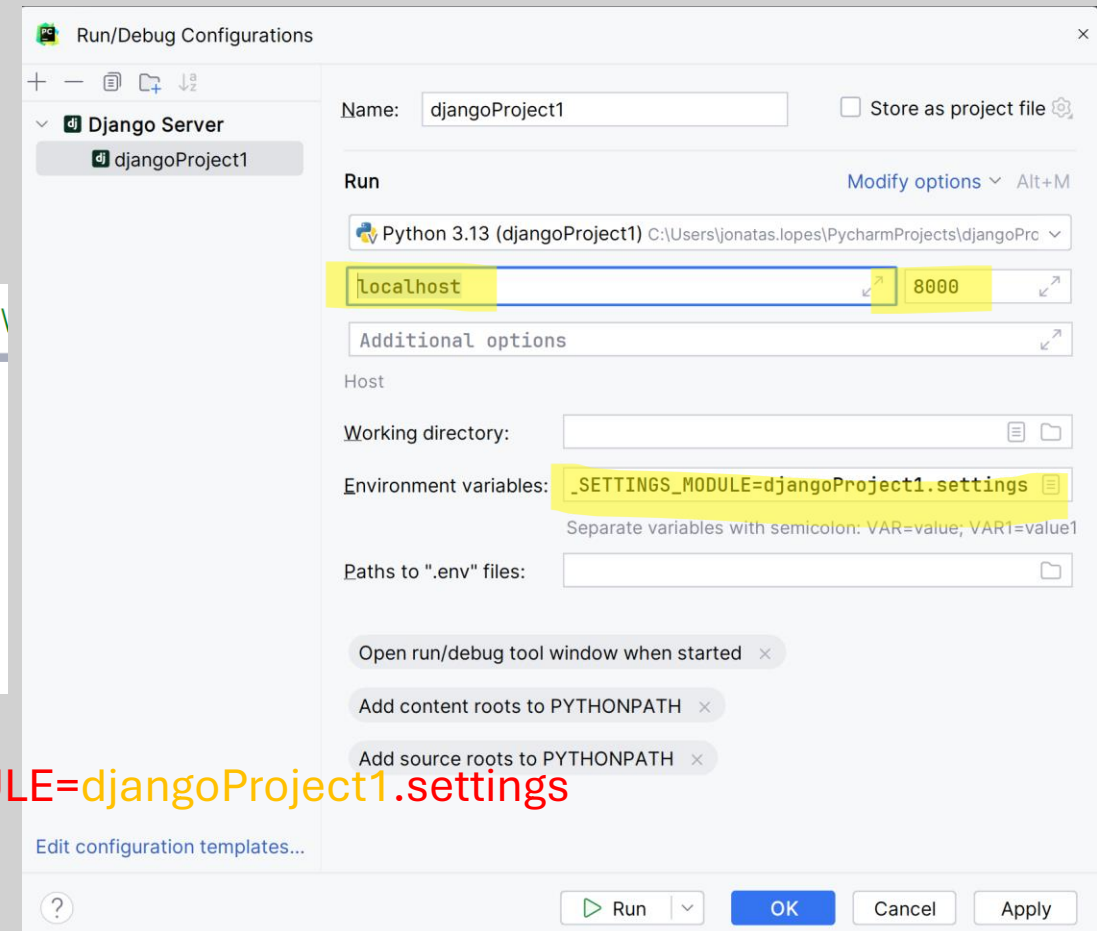
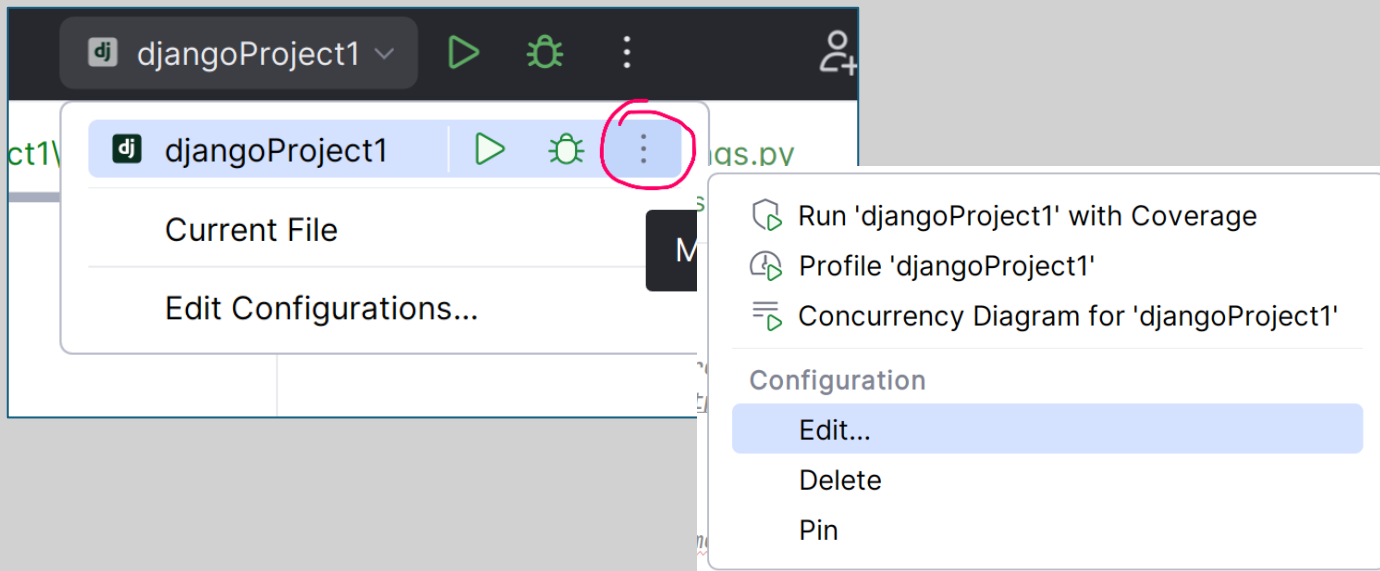
@fpftech.educacional

Primeiro devemos configurar o Django pra rodar localmente.
2 formas de fazer isso:

```
project1> python manage.py runserver
```

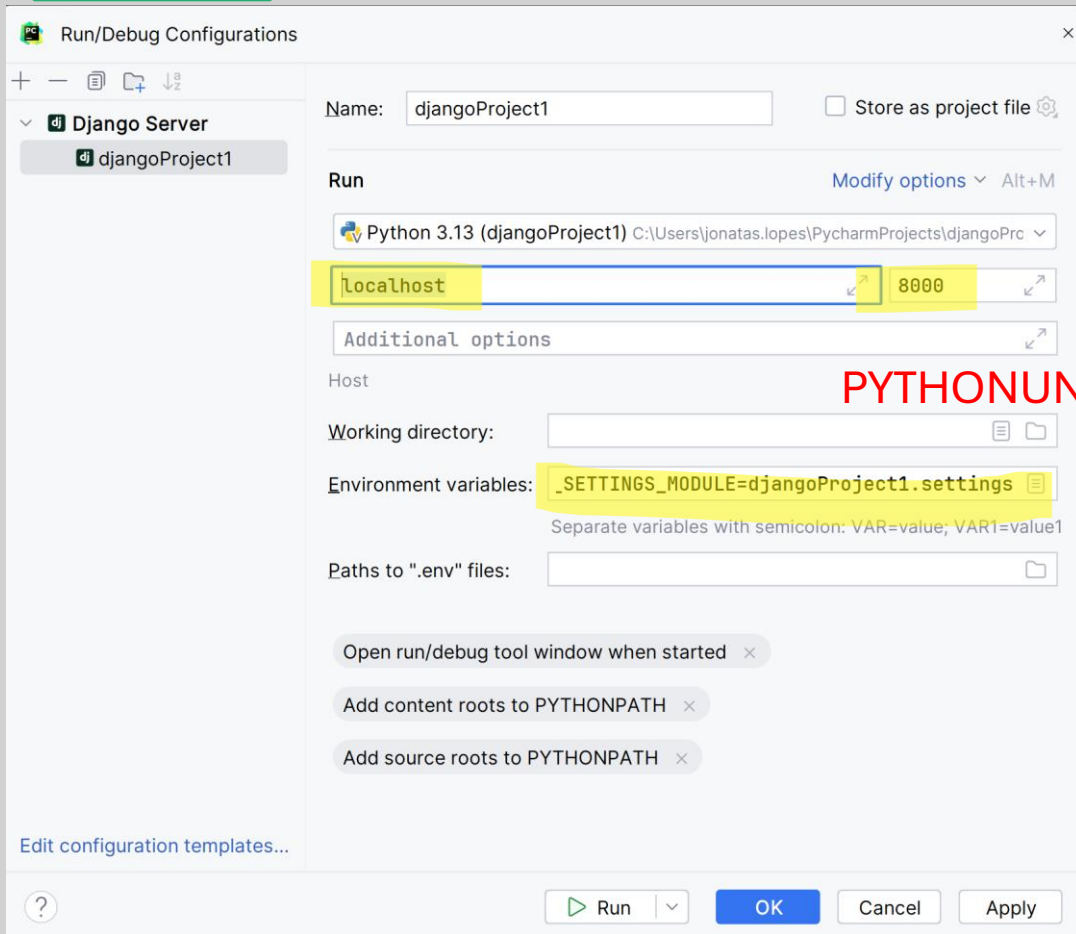
```
goProject1> python manage.py runserver 0.0.0.0:8000
```

Ou... Pelo pycharm IDE

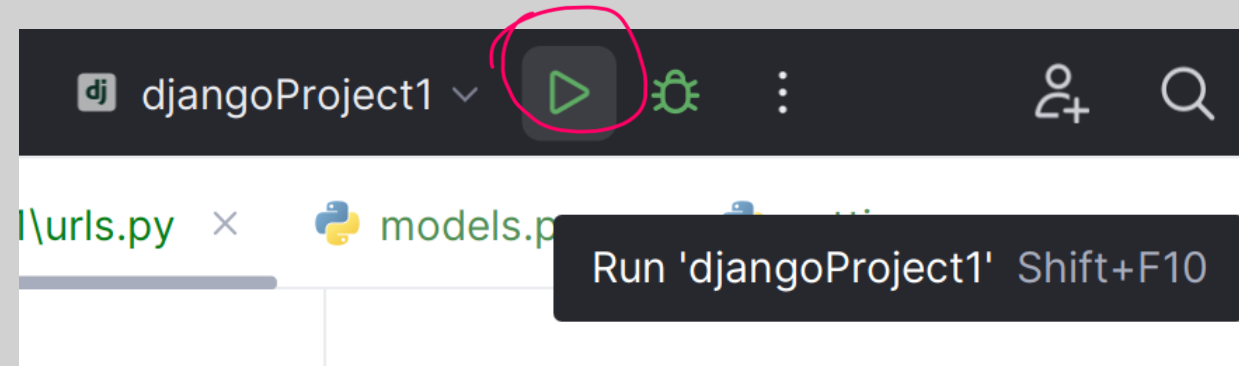


PYTHONUNBUFFERED=1;DJANGO_SETTINGS_MODULE=djangoProject1.settings





PYTHONUNBUFFERED=1;DJANGO_SETTINGS_MODULE=djangoProject1.settings




```
C:\Users\jonatas.lopes\PycharmProjects\djangoProject1\.venv\Scripts\python.exe C:\Users\jonatas.lopes\PycharmProjects\djangoProject1\manage.py runserver localhost:8000
```

```
Watching for file changes with StatReloader
```

```
Performing system checks...
```

```
System check identified no issues (0 silenced).
```

```
May 14, 2025 - 04:39:13
```

```
Django version 5.2.1, using settings 'djangoProject1.settings'
```

```
Starting development server at http://localhost:8000/
```

```
Quit the server with CTRL-BREAK.
```

```
WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server instead.
```

```
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/
```



Django REST framework

Api Root

Api Root

OPTIONS

GET



The default basic root view for DefaultRouter

GET /api/

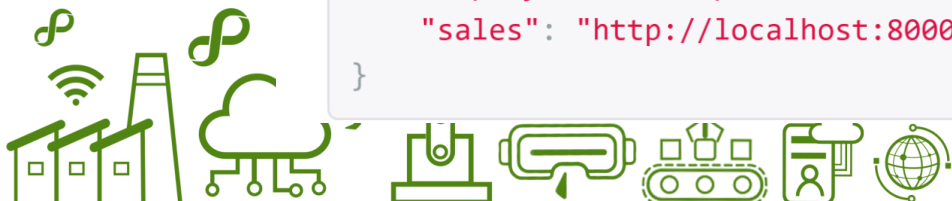
HTTP 200 OK

Allow: GET, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "clients": "http://localhost:8000/api/clients/",
  "products": "http://localhost:8000/api/products/",
  "employees": "http://localhost:8000/api/employees/",
  "sales": "http://localhost:8000/api/sales/"
}
```

 **Clique num desses endpoints**

Employee List

OPTIONS GET

GET /api/employees/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
[
  {
    "id": 1,
    "created_at": null,
    "modified_at": null,
    "active": true,
    "name": "Beltrano dos Santos",
    "registration": "456789"
  },
  {
    "id": 2,
    "created_at": null,
    "modified_at": null,
    "active": true,
    "name": "Fulana Rosa",
    "registration": "345678"
  }
]
```

Raw data HTML form

Active

☐

Name

Registration

POST

Exemplo de POST

Active ☒

Name

Registration

POST



@fpftech.educacional

Employee List

POST /api/employees/

HTTP 201 Created

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "id": 3,
  "created_at": "2025-05-14T08:49:12.105816Z",
  "modified_at": "2025-05-14T08:49:12.105833Z",
  "active": true,
  "name": "Teste",
  "registration": "567890"
}
```



GET com a lista atualizada

Api Root / Employee List

Employee List

GET /employees/

HTTP 200 OK

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
[
  {
    "id": 1,
    "created_at": null,
    "modified_at": null,
    "active": true,
    "name": "Beltrano dos Santos",
    "registration": "456789"
  },
  {
    "id": 2,
    "created_at": null,
    "modified_at": null,
    "active": true,
    "name": "Fulana Rosa",
    "registration": "345678"
  },
  {
    "id": 3,
    "created_at": "2025-05-14T08:49:12.105816Z",
    "modified_at": "2025-05-14T08:49:12.105833Z",
    "active": true,
    "name": "Teste",
    "registration": "567890"
  }
]
```



GET do sale

No serializer dizemos pra ele trazer o obj ->

No serializer dizemos pra ele trazer o obj ->

No serializer dizemos pra ele trazer o obj ->

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

```
[
  {
    "id": 4,
    "product": {
      "id": 1,
      "description": "Aula de Ingles com material didático",
      "quantity": 1
    },
    "client": {
      "id": 2,
      "name": "Fulano da Silva",
      "age": 23,
      "rg": "123456-7",
      "cpf": "123456789-01"
    },
    "employee": {
      "id": 1,
      "created_at": null,
      "modified_at": null,
      "active": true,
      "name": "Beltrano dos Santos",
      "registration": "456789"
    },
    "nrf": "123456789-0"
  }
]
```



No serializer dizemos pra ele receber só o numero e ir buscar o obj

Aqui (já que é o próprio Django Admin) ele já buscou todos (all) obj e nos mostrou por id

Product id	Product object (2)
Client id	Client object (5)
Employee id	Employee object (1)
Nrf	<div>Employee object (1)</div> <div>Employee object (2)</div> <div>Employee object (3)</div>
	POST



Resultado do POST do sale

Django REST framework

Sale List

POST /sales/

HTTP 201 Created

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

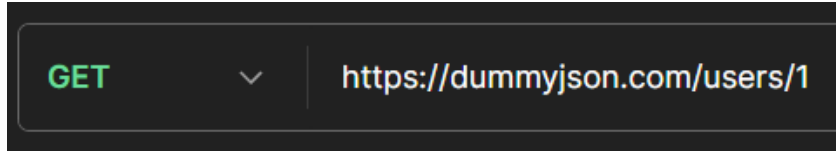
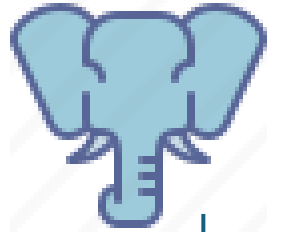
Vary: Accept

```
{
  "id": 7,
  "product": {
    "id": 2,
    "description": "Material didático de Introdução a Física nível Ensino Superior",
    "quantity": 2
  },
  "client": {
    "id": 5,
    "name": "Siclano Bertrano",
    "age": 19,
    "rg": "12345498-9",
    "cpf": "234456859-48"
  },
  "employee": {
    "id": 1,
    "created_at": null,
    "modified_at": null,
    "active": true,
    "name": "Beltrano dos Santos",
    "registration": "456789"
  },
  "nrf": "456789012-3"
}
```



@fpftech.educacional

Ciclo de Requests



```
from rest_framework import serializers
from .models import MockUser

class MockUserSerializer(serializers.ModelSerializer):
    class Meta:
        model = MockUser
        fields = ['id', 'username', 'email']
```



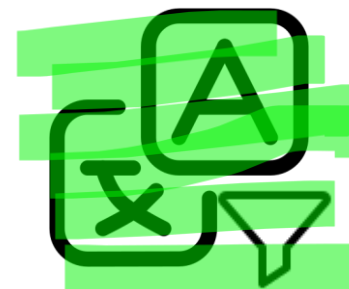
request



endpoint
urls



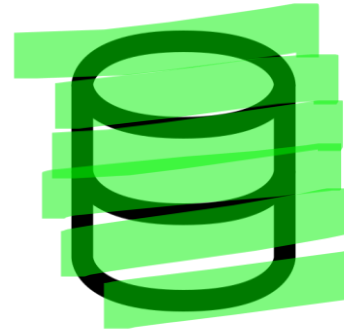
viewsets



Serializers
JSON=Django



models



database

```
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import MockUserViewSet

router = DefaultRouter()
router.register(r'mock-users', MockUserViewSet)

urlpatterns = [
    path('', include(router.urls)),
]
```

```
class MockUserViewSet(viewsets.ModelViewSet):
    queryset = MockUser.objects.all()
    serializer_class = MockUserSerializer
```

```
from django.db import models

class MockUser(models.Model):
    username = models.CharField(max_length=100)
    email = models.EmailField(unique=True)

    def __str__(self):
        return self.username
```



© 2010 Pearson Education, Inc. or its affiliate(s). All rights reserved.

```
1 ▾ [
2 ▾ {
3     "id": 4,
4 ▾   "product": {
5       "id": 1,
6       "description": "Aula de Ingles com material di
7       "quantity": 1
8     },
9 ▾   "client": {
10     "id": 2,
11     "name": "Fulano da Silva",
12     "age": 23,
13     "rg": "123456-7",
14     "cpf": "123456789-01"
15   },
16 ▾   "employee": {
17     "id": 1,
18     "created_at": null,
```



@fpfttech.educacional

Exemplo POST 'sale' no insomnia

POST http://localhost:8000/sales/

Send

201 Created

76 ms

322 B

ParamsBodyAuthHeaders4ScriptsDocs

PreviewHeaders10CookiesTests0/0Mock

JSON

Preview

```
1 {
2   "product_id": 1,
3   "client_id": 4,
4   "employee_id": 2,
5   "nrf": "234567891-0"
6 }
```

```
1 {
2   "id": 8,
3   "product": {
4     "id": 1,
5     "description": "Aula de Ingles com material didáti
6     "quantity": 1
7   },
8   "client": {
9     "id": 4,
10    "name": "Siclano Bertrano",
11    "age": 19,
12    "rg": "12345498-9",
13    "cpf": "234456859-48"
14  },
15  "employee": {
16    "id": 2,
17    "created_at": null,
18    "modified_at": null,
19    "active": true,
20    "name": "Fulana Rosa",
21    "registration": "345678"
22  },
23  "nrf": "234567891-0"
24 }
```


Exemplo GET 'clients' no postman

HTTP <http://localhost:8000/clients/> Save Share

GET

<http://localhost:8000/clients/>

Send

Params Auth Headers (7) Body Scripts Tests Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

200 OK • 53 ms • 741 B •  ...

{ } JSON

Preview

Visualize

```
1  [
2    {
3      "id": 2,
4      "name": "Fulano da Silva",
5      "age": 23,
6      "rg": "123456-7",
7      "cpf": "123456789-01"
8    },
9    {
10     "id": 3,
11     "name": "Siclano da Silva",
12     "age": 41,
13     "rg": "234567-8",
14     "cpf": "234567890-12"
15   },
16 ]
```



Exemplo GET 'clients' específico

Usando o id do cliente

GET Send



Params Auth Headers (9) Body ● Scripts Tests Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body 

200 OK • 58 ms • 425 B •  • 

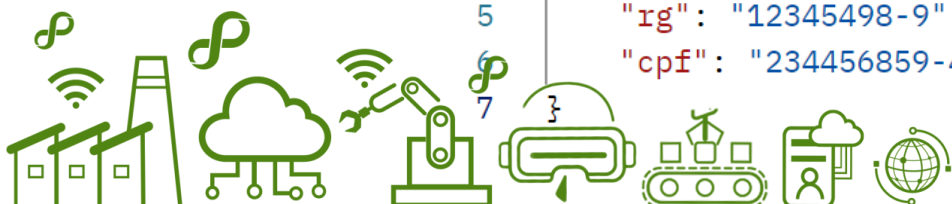
{ } JSON

▶ Preview

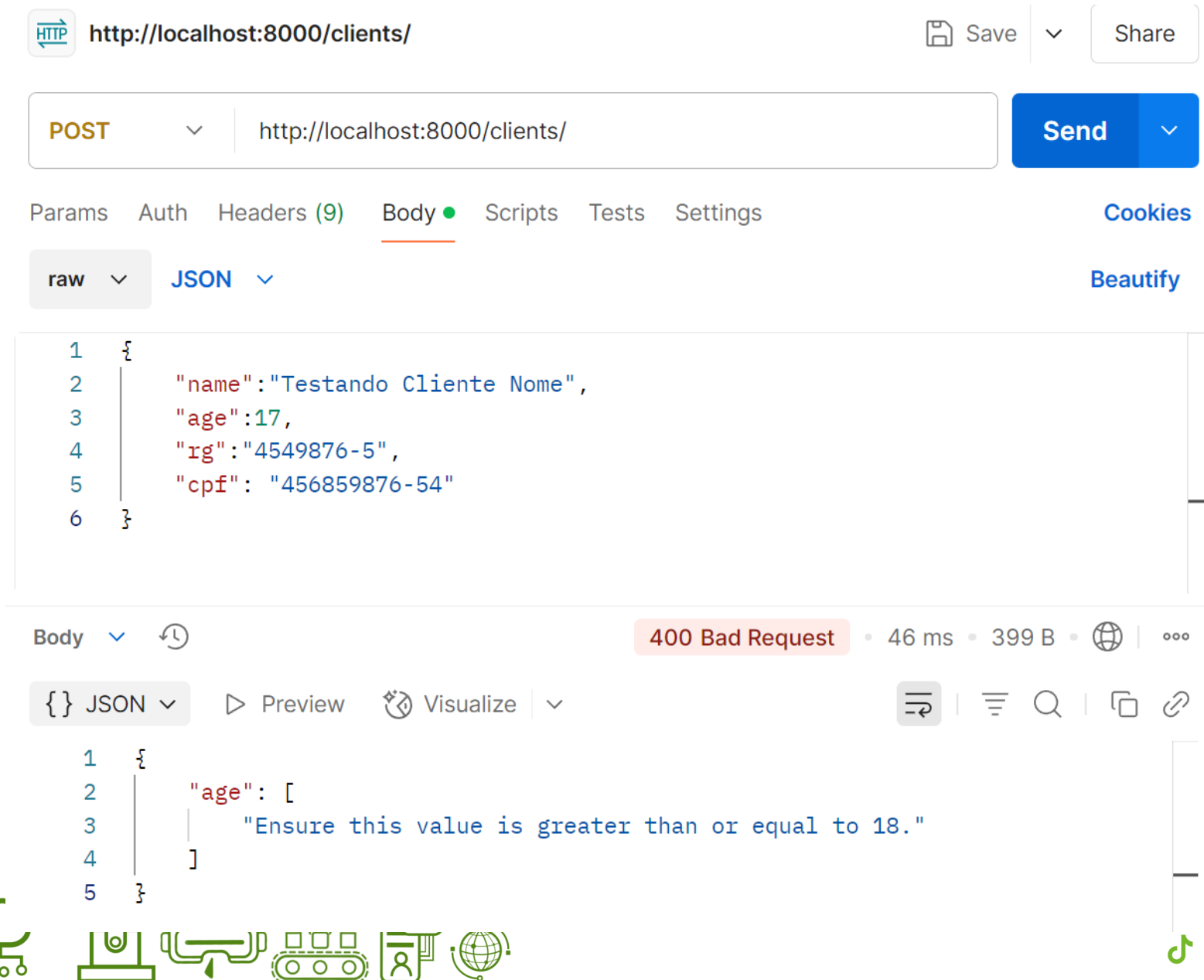
 Visualize



```
1 {
2   "id": 6,
3   "name": "Siclano Bertrano",
4   "age": 19,
5   "rg": "12345498-9",
6   "cpf": "234456859-48"
7 }
```



Exemplo POST 'clients' no postman



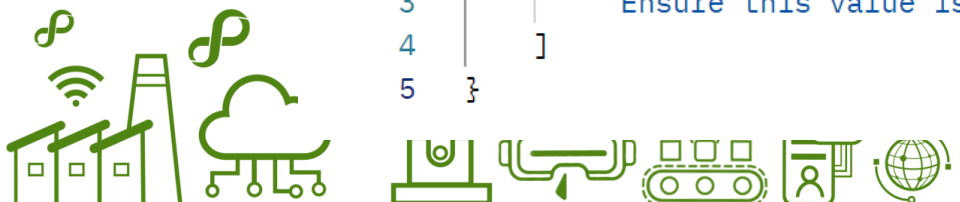
The image shows the Postman interface for a POST request to `http://localhost:8000/clients/`. The request body is a JSON object:

```
1 {  
2   "name": "Testando Cliente Nome",  
3   "age": 17,  
4   "rg": "4549876-5",  
5   "cpf": "456859876-54"  
6 }
```

The response status is **400 Bad Request** (46 ms, 399 B). The response body is a JSON object:

```
1 {  
2   "age": [  
3     "Ensure this value is greater than or equal to 18."  
4   ]  
5 }
```

Por que deu errado?





Exemplo POST 'clients' no postman

POST ▼ http://localhost:8000/clients/ Send ▼

Params Auth Headers (9) Body ● Scripts Tests Settings Cookies

raw ▼ JSON ▼ Beautify

```
1 {  
2   "name": "Testando Cliente Nome",  
3   "age": 18,  
4   "rg": "4549876-5",  
5   "cpf": "456859876-54"  
6 }
```

Body ▼  201 Created • 65 ms • 420 B •  | ⋮

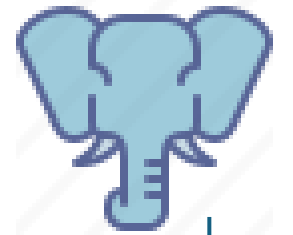
{} JSON ▼ ▶ Preview 🔗 Visualize ▼ ≡ | ≡ | 🔍 | 📄 | 🔗

```
1 {  
2   "id": 7,  
3   "name": "Testando Cliente Nome",  
4   "age": 18,  
5   "rg": "4549876-5",  
6   "cpf": "456859876-54"  
7 }
```

Agora sim!



Ciclo de Requests



```
GET https://dummyjson.com/users/1
```

```
from rest_framework import serializers
from .models import MockUser

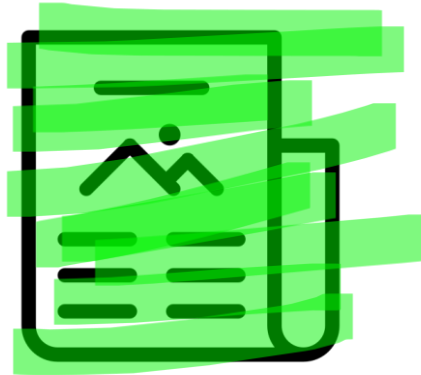
class MockUserSerializer(serializers.ModelSerializer):
    class Meta:
        model = MockUser
        fields = ['id', 'username', 'email']
```



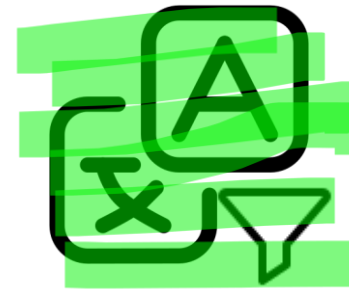
request



endpoint
urls



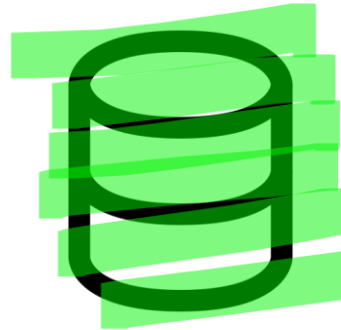
viewsets



Serializers
JSON=Django



models



database

```
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import MockUserViewSet

router = DefaultRouter()
router.register(r'mock-users', MockUserViewSet)

urlpatterns = [
    path('', include(router.urls)),
]
```

```
class MockUserViewSet(viewsets.ModelViewSet):
    queryset = MockUser.objects.all()
    serializer_class = MockUserSerializer
```

```
from django.db import models

class MockUser(models.Model):
    username = models.CharField(max_length=100)
    email = models.EmailField(unique=True)

    def __str__(self):
        return self.username
```



@fpftech.educacional

Obrigado!



    @fpftech.educacional