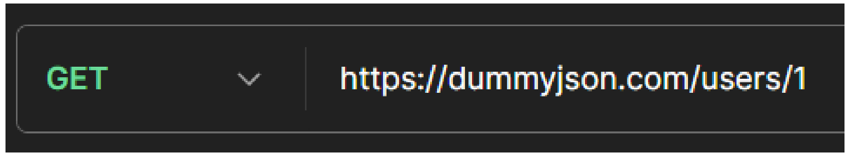
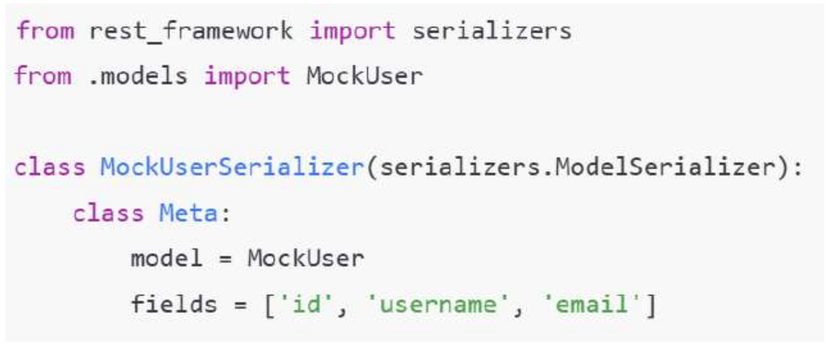
@fpftech.educacional

FPFtech
Escola Tecnológica

FPFtech
Escola Tecnológica

Ciclo de Requests


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


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

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

request

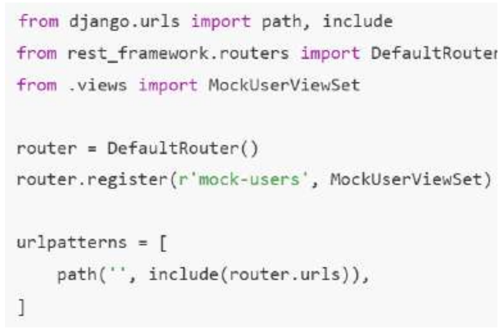
endpoint
urls

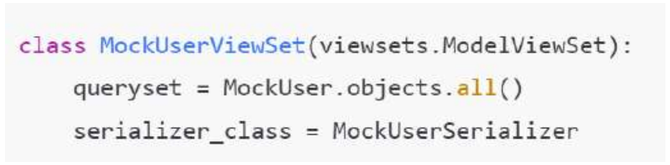
viewsets

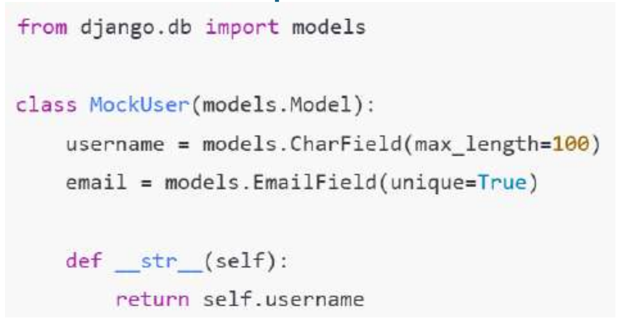
Serializers
JSON=Django

models

database


```python
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)),
]
```

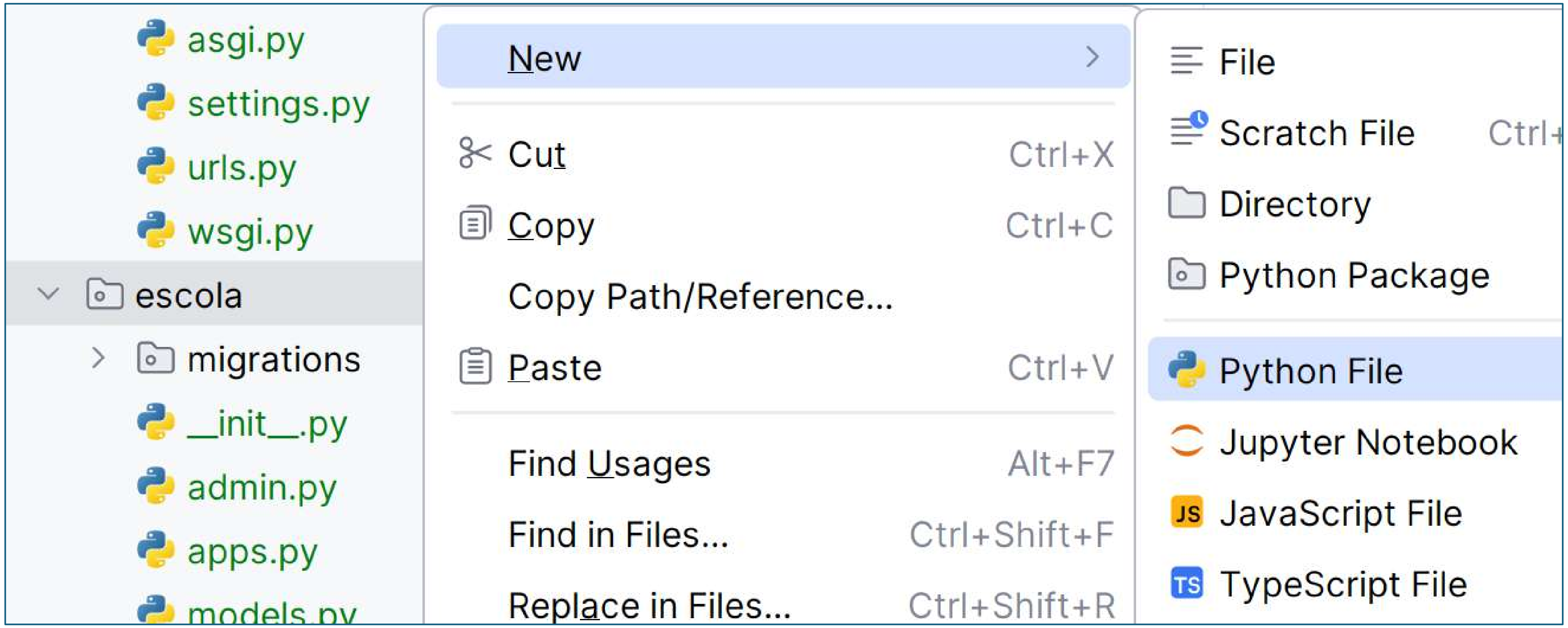

```python
class MockUserViewSet(viewsets.ModelViewSet):
    queryset = MockUser.objects.all()
    serializer_class = MockUserSerializer
```


```python
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
```

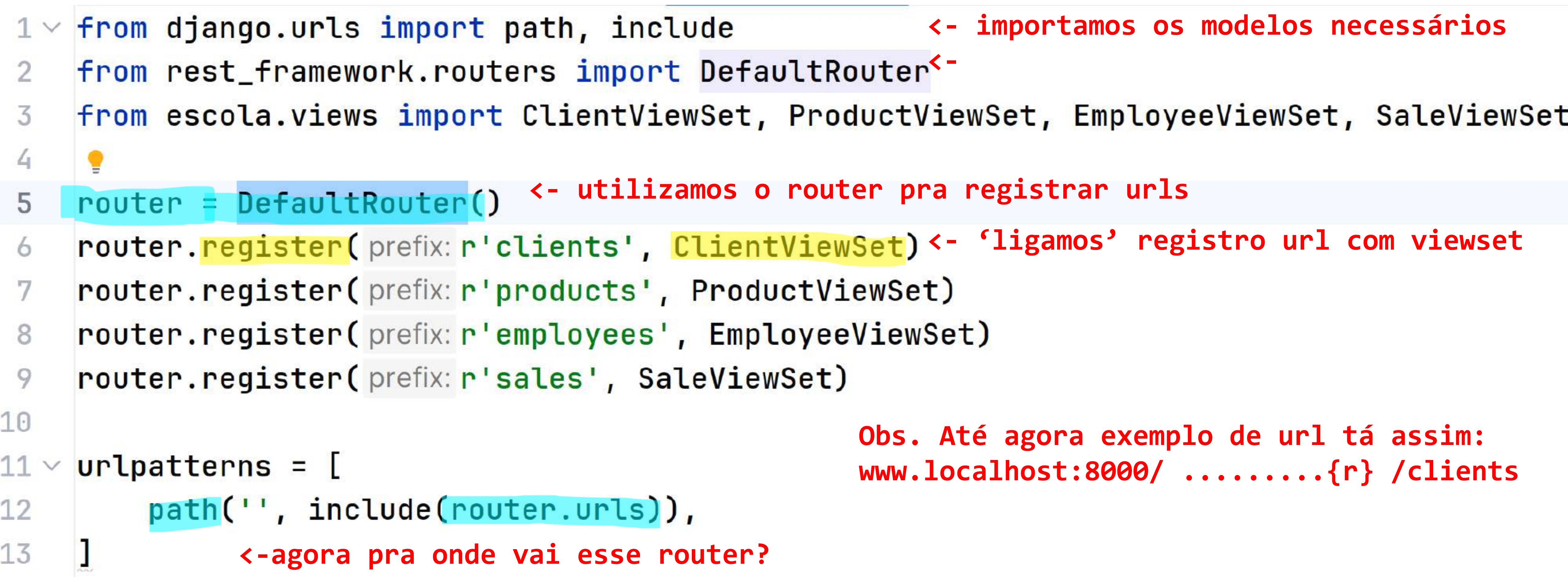
Urls

Urls = arquivo que registra suas urls pros seus viewsets.

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



Urls



```python
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from escola.views import ClientViewSet, ProductViewSet, EmployeeViewSet, SaleViewSet

router = DefaultRouter()
router.register( prefix: r'clients', ClientViewSet)
router.register( prefix: r'products', ProductViewSet)
router.register( prefix: r'employees', EmployeeViewSet)
router.register( prefix: r'sales', SaleViewSet)

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

<- importamos os modelos necessários

<-

<- utilizamos o router pra registrar urls

<- ‘ligamos’ registro url com viewset

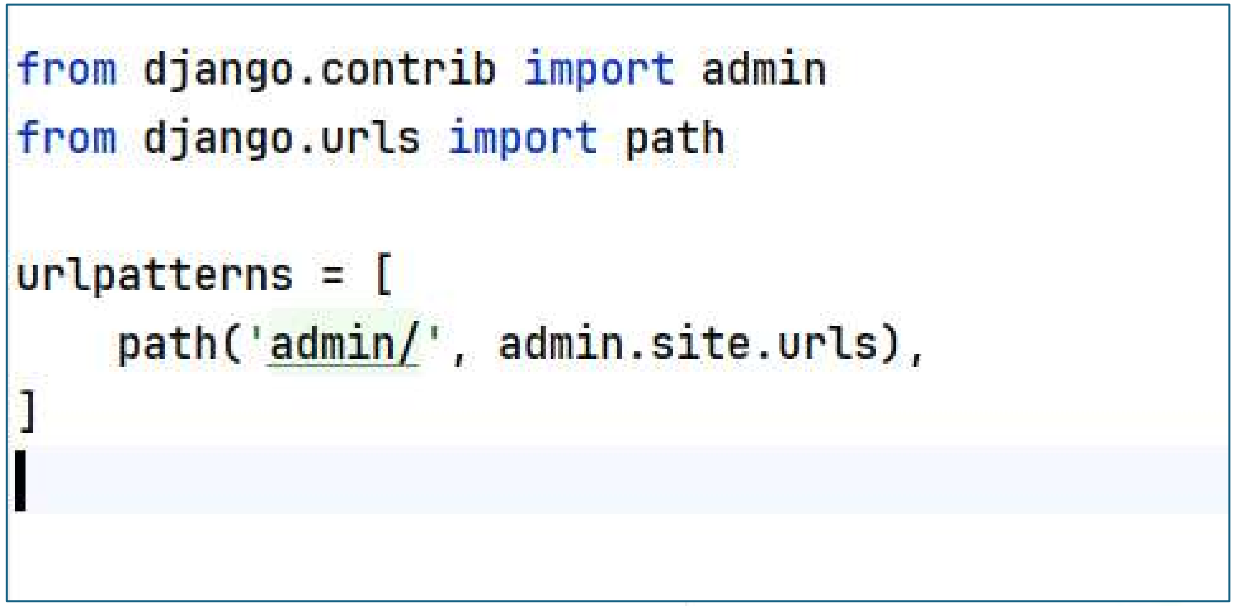
Obs. Até agora exemplo de url tá assim:
www.localhost:8000/ ....{r} /clients

<-agora pra onde vai esse router?

@fpftech.educacional

Urls

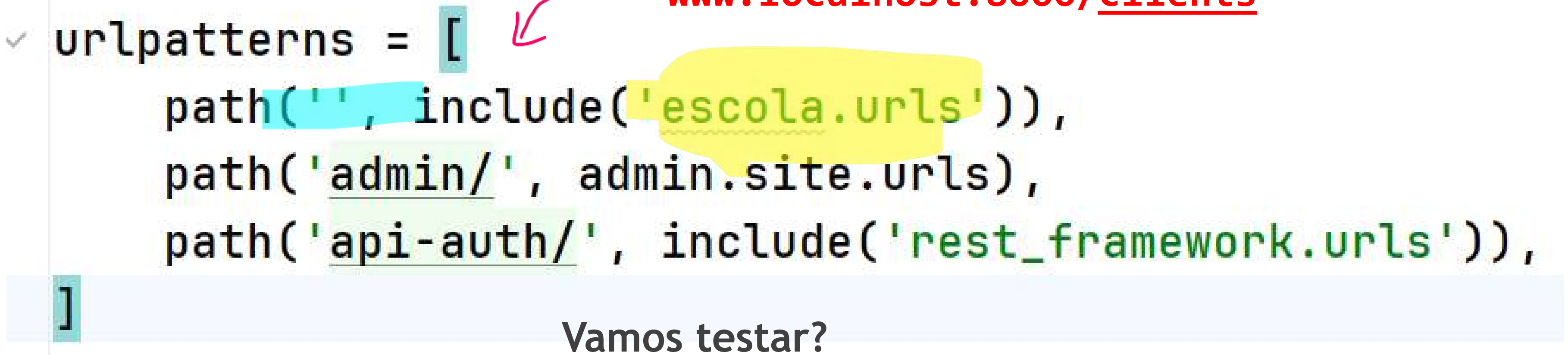
No projetão, acesse o arquivo urls.py e faça a alteração pra incluir as urls que criamos



```
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

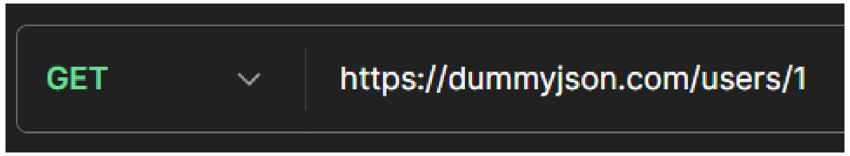
Então, o exemplo de uma url seria: www.localhost:8000/clients

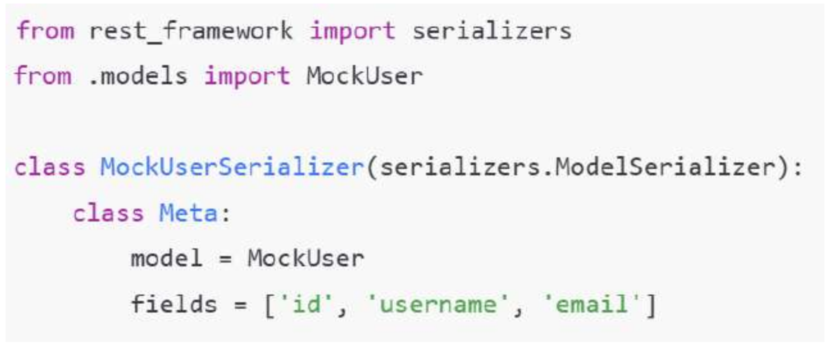


```
urlpatterns = [
    path('', include('escola.urls')),
    path('admin/', admin.site.urls),
    path('api-auth/', include('rest_framework.urls')),
]
```

Vamos testar?

Ciclo de Requests


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


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

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

request

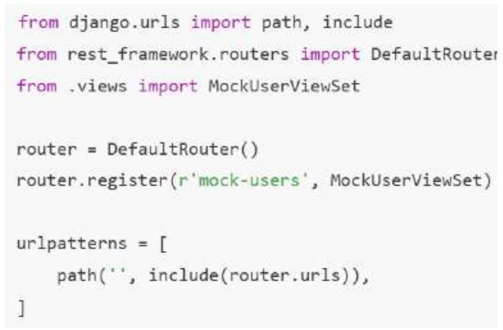
endpoint
urls

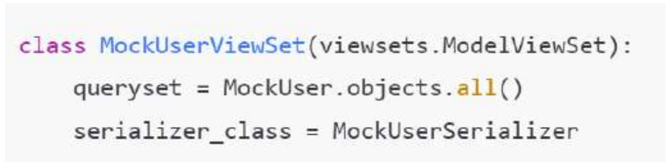
viewsets

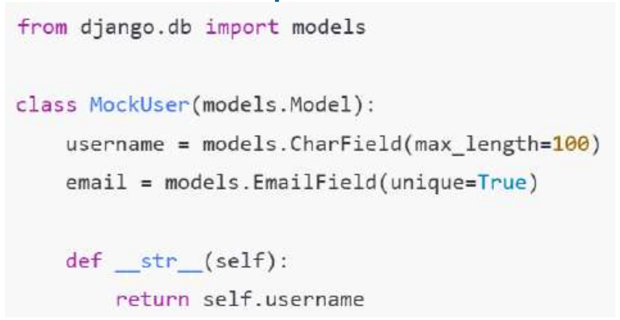
Serializers
JSON=Django

models

database

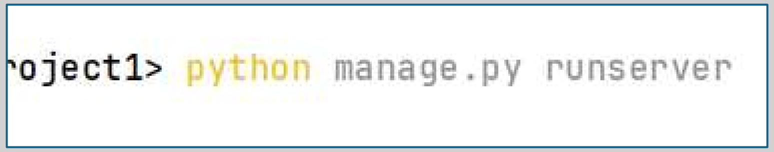

```python
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import MockUserViewSet

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

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


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

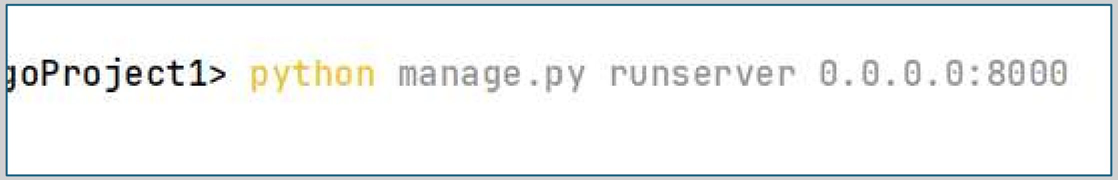

```python
from django.db import models

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

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

Pycharm

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

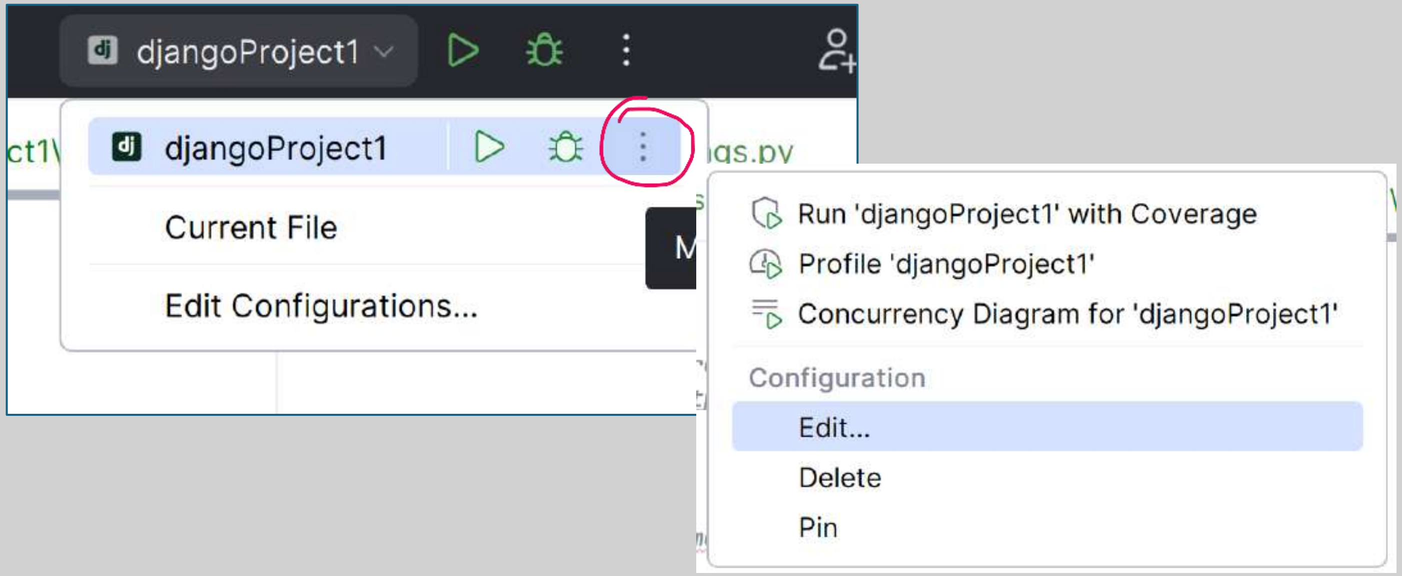


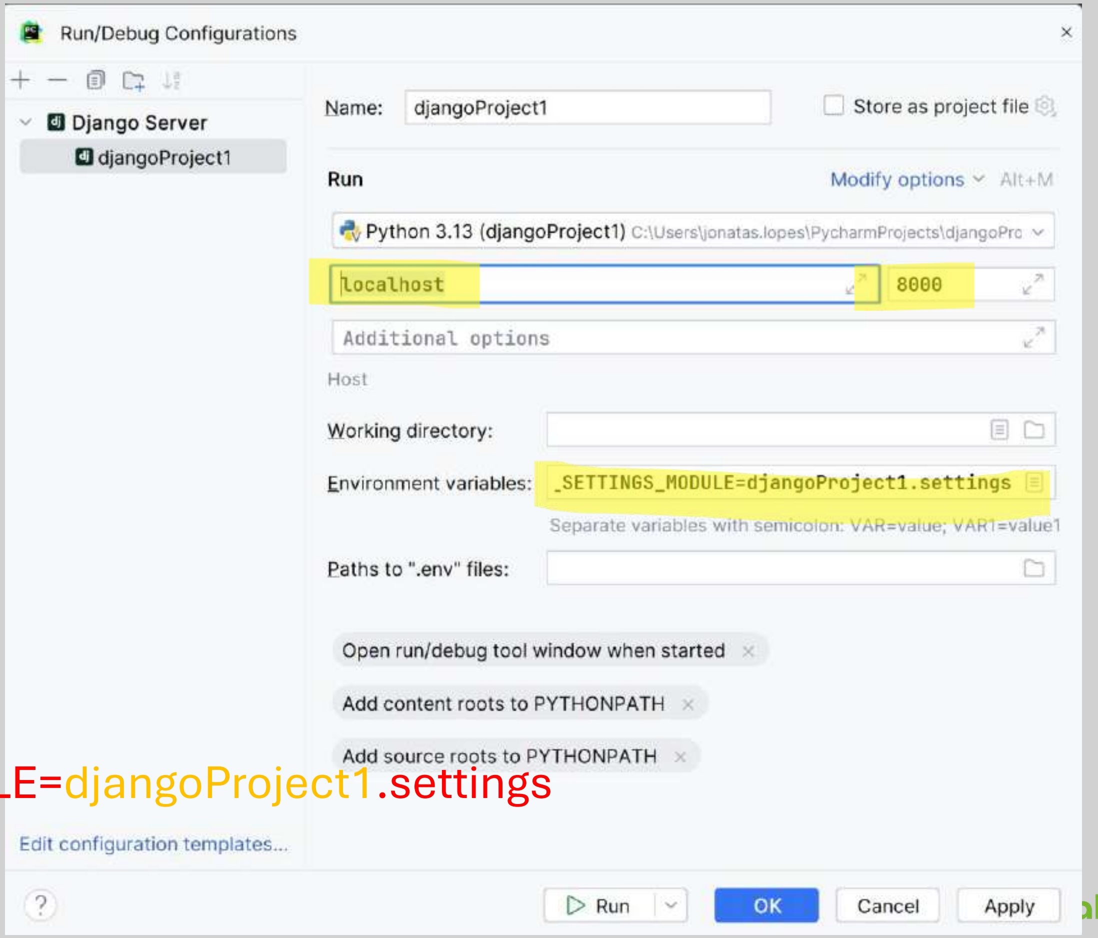
```
roject1> python manage.py runserver
```



```
joProject1> python manage.py runserver 0.0.0.0:8000
```

Ou... Pelo pycharm IDE





Name: djangoProject1

Python 3.13 (djangoProject1) C:\Users\jonatas.lopes\PycharmProjects\djangoPro

localhost 8000

Additional options

Host

Working directory:

Environment variables: _SETTINGS_MODULE=djangoProject1.settings

Separate variables with semicolon: VAR=value; VAR1=value1

Paths to ".env" files:

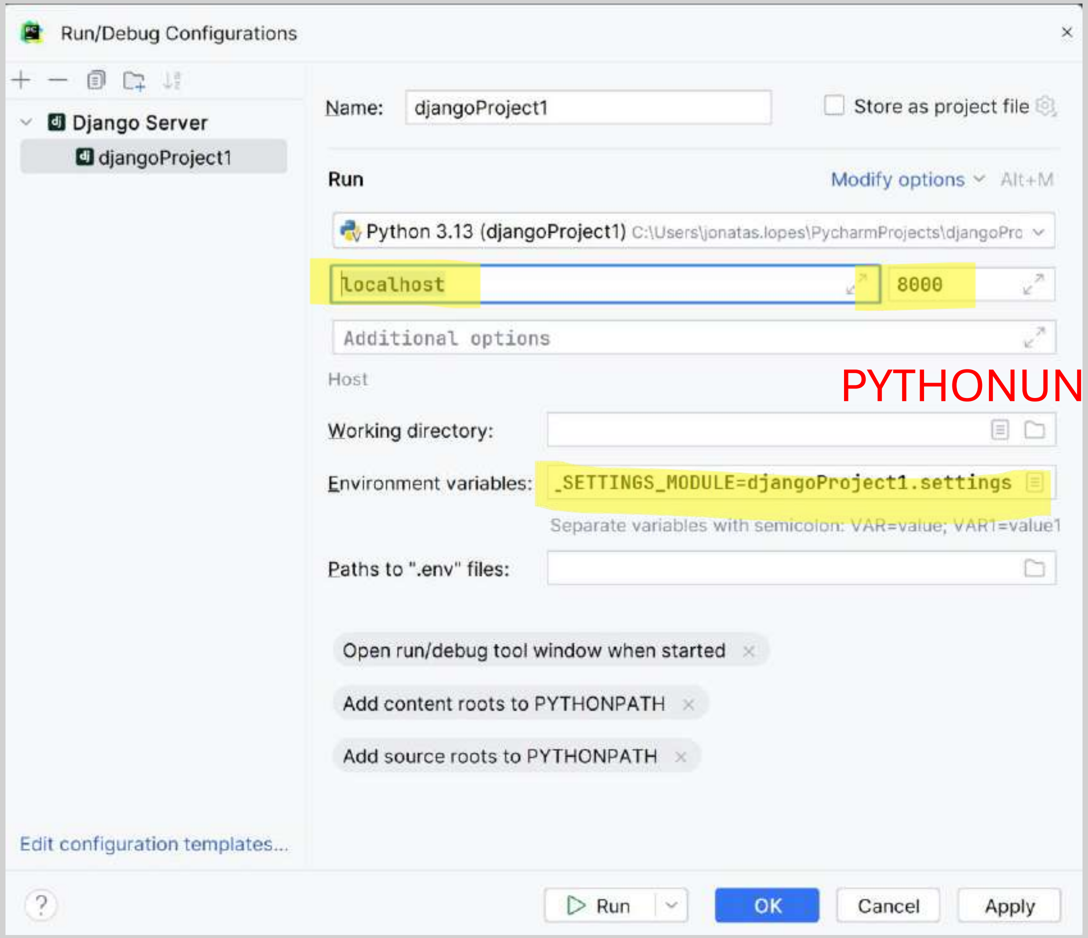
Open run/debug tool window when started

Add content roots to PYTHONPATH

Add source roots to PYTHONPATH

PYTHONUNBUFFERED=1;DJANGO_SETTINGS_MODULE=djangoProject1.settings

Pycharm



Run/Debug Configurations

Django Server
- djangoProject1

Name: djangoProject1
Store as project file

Run — Modify options Alt+M

Python 3.13 (djangoProject1) C:\Users\jonatas.lopes\PycharmProjects\djangoPro

Host: localhost — Port: 8000

Additional options

Working directory:

Environment variables: _SETTINGS_MODULE=djangoProject1.settings

Separate variables with semicolon: VAR=value; VAR1=value1

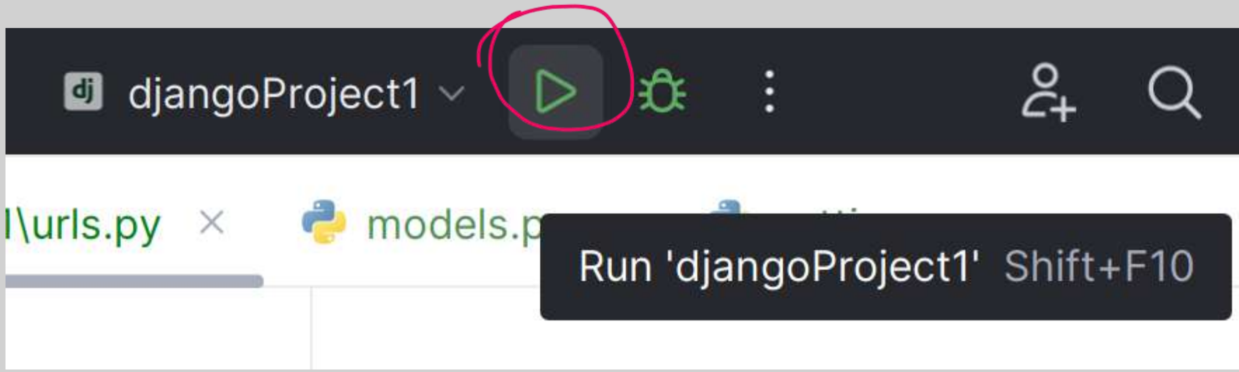
Paths to ".env" files:

- Open run/debug tool window when started
- Add content roots to PYTHONPATH
- Add source roots to PYTHONPATH

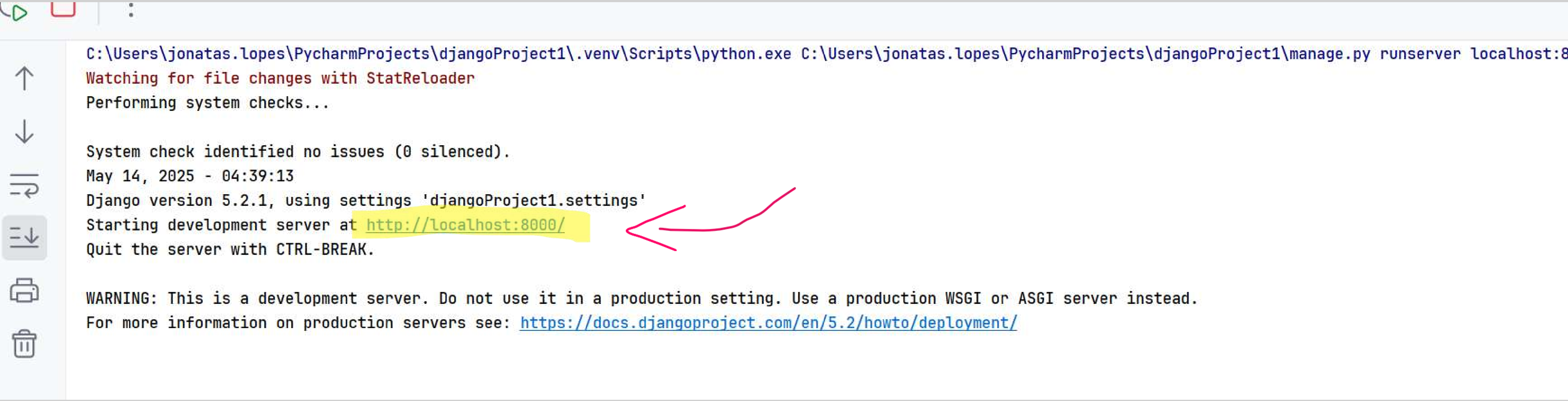
Edit configuration templates...

Run | OK | Cancel | Apply

PYTHONUNBUFFERED=1;DJANGO_SETTINGS_MODULE=djangoProject1.settings



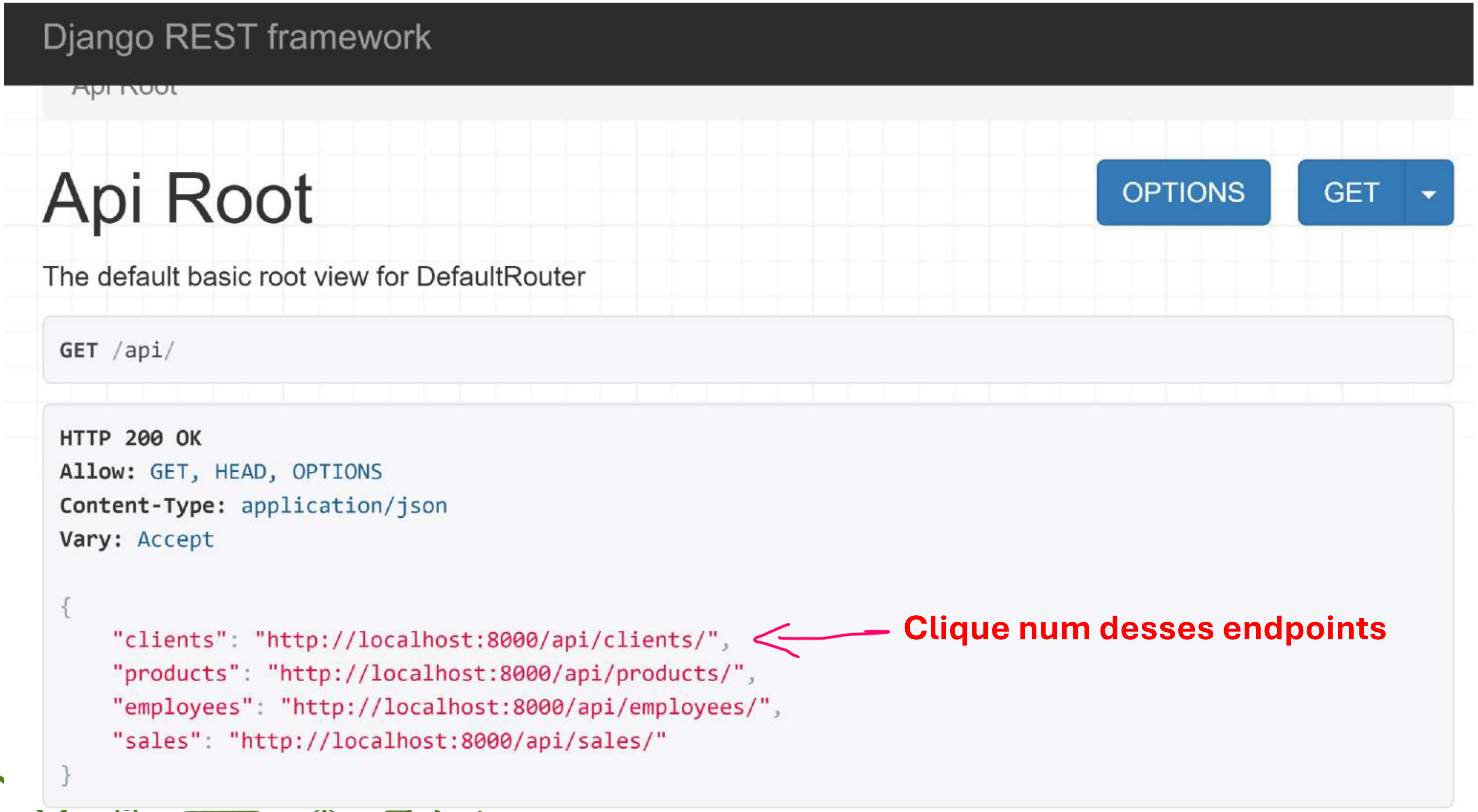
Pycharm



```
C:\Users\jonatas.lopes\PycharmProjects\djangoProject1\.venv\Scripts\python.exe C:\Users\jonatas.lopes\PycharmProjects\djangoProject1\manage.py runserver localhost:8
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/
```

Pycharm

www.localhost:8000/



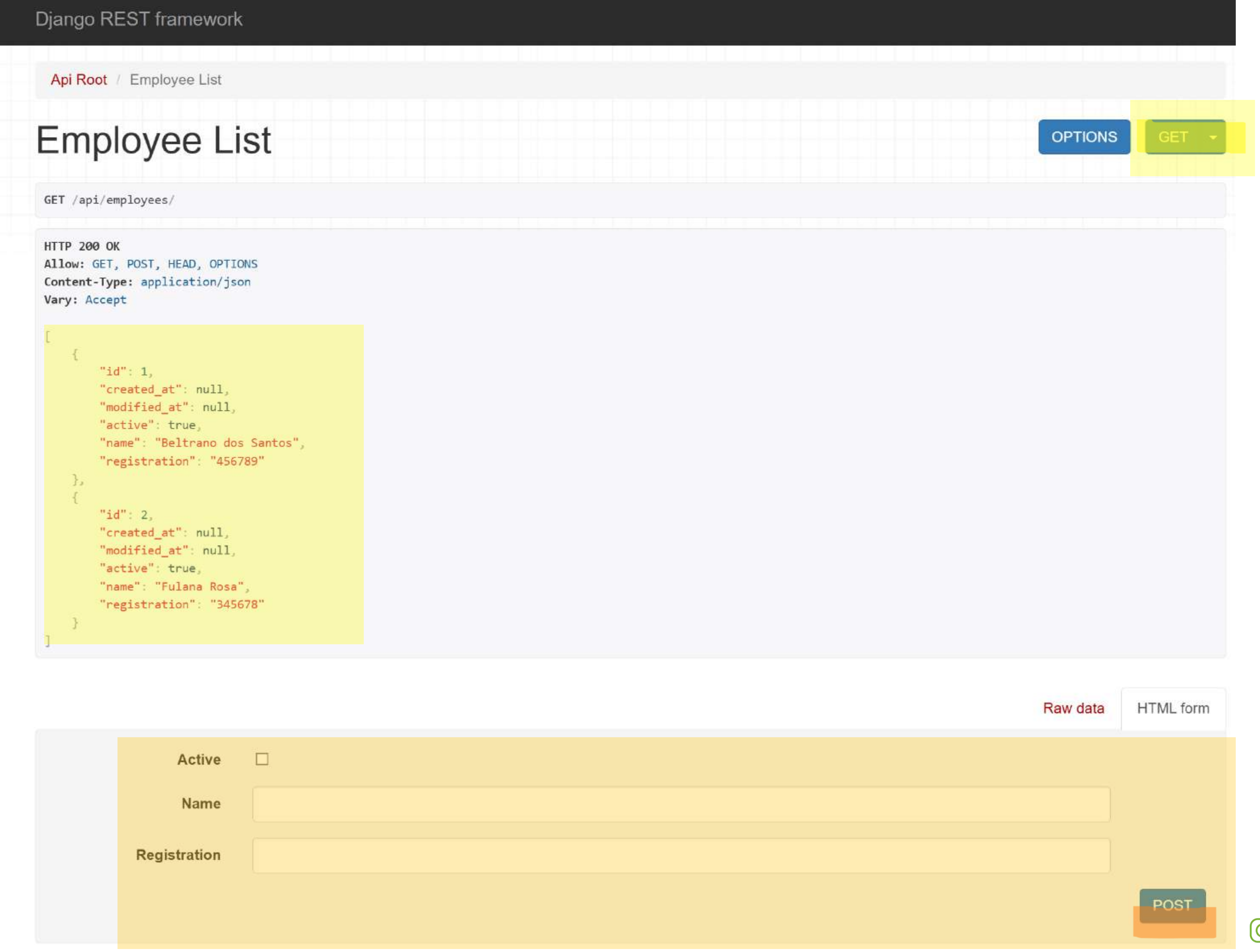
Api Root

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

```
GET /api/employees/
```

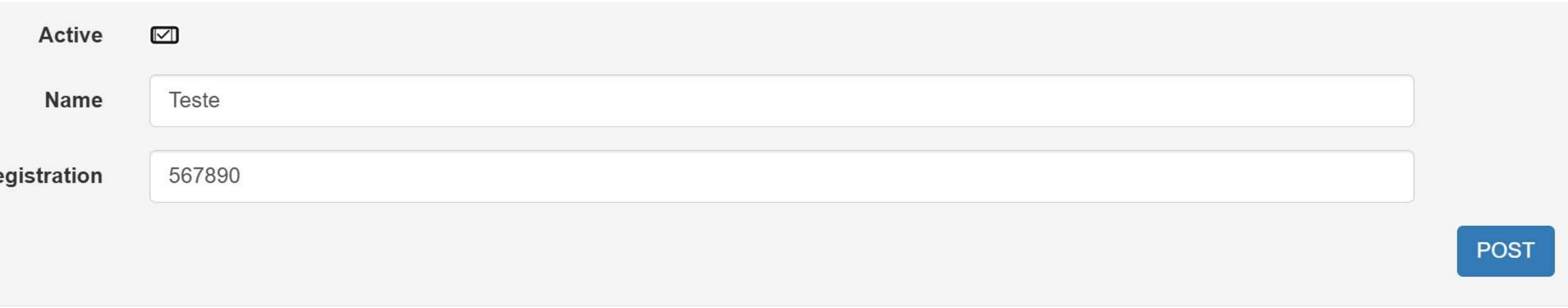
```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "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"
    }
]
```

Active	☐
Name	
Registration	

FPFtech Escola Tecnológica

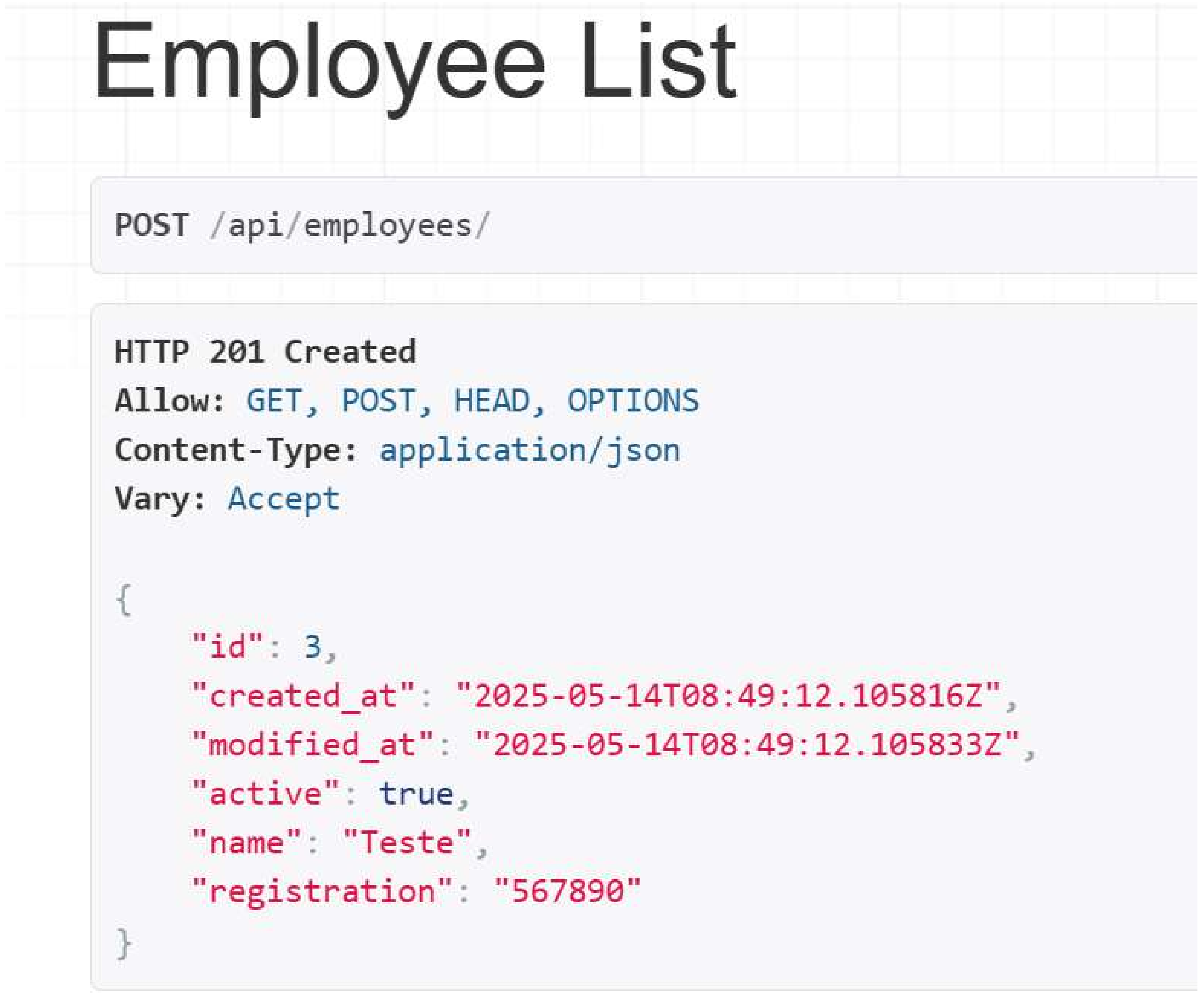
@fpftech.educacional

Exemplo de POST



Active	✓
Name	Teste
egistration	567890

Resultado do POST

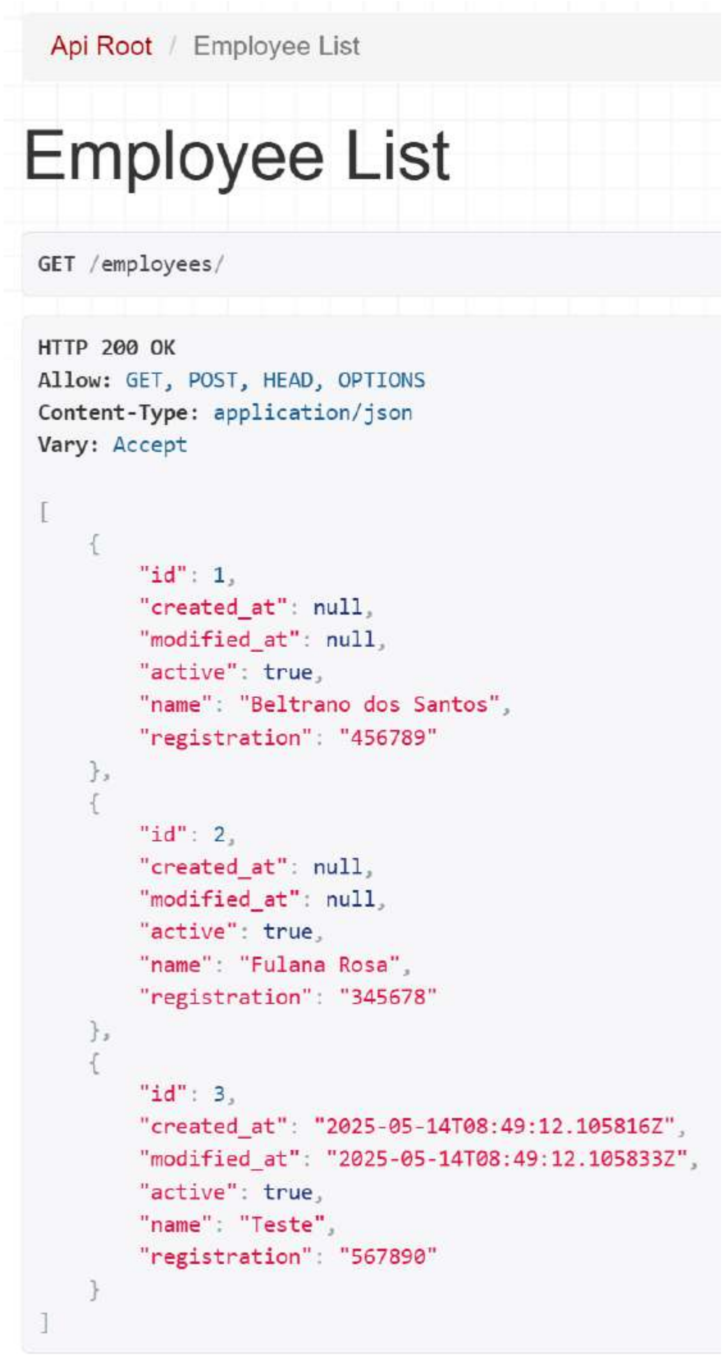


Employee List

```
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

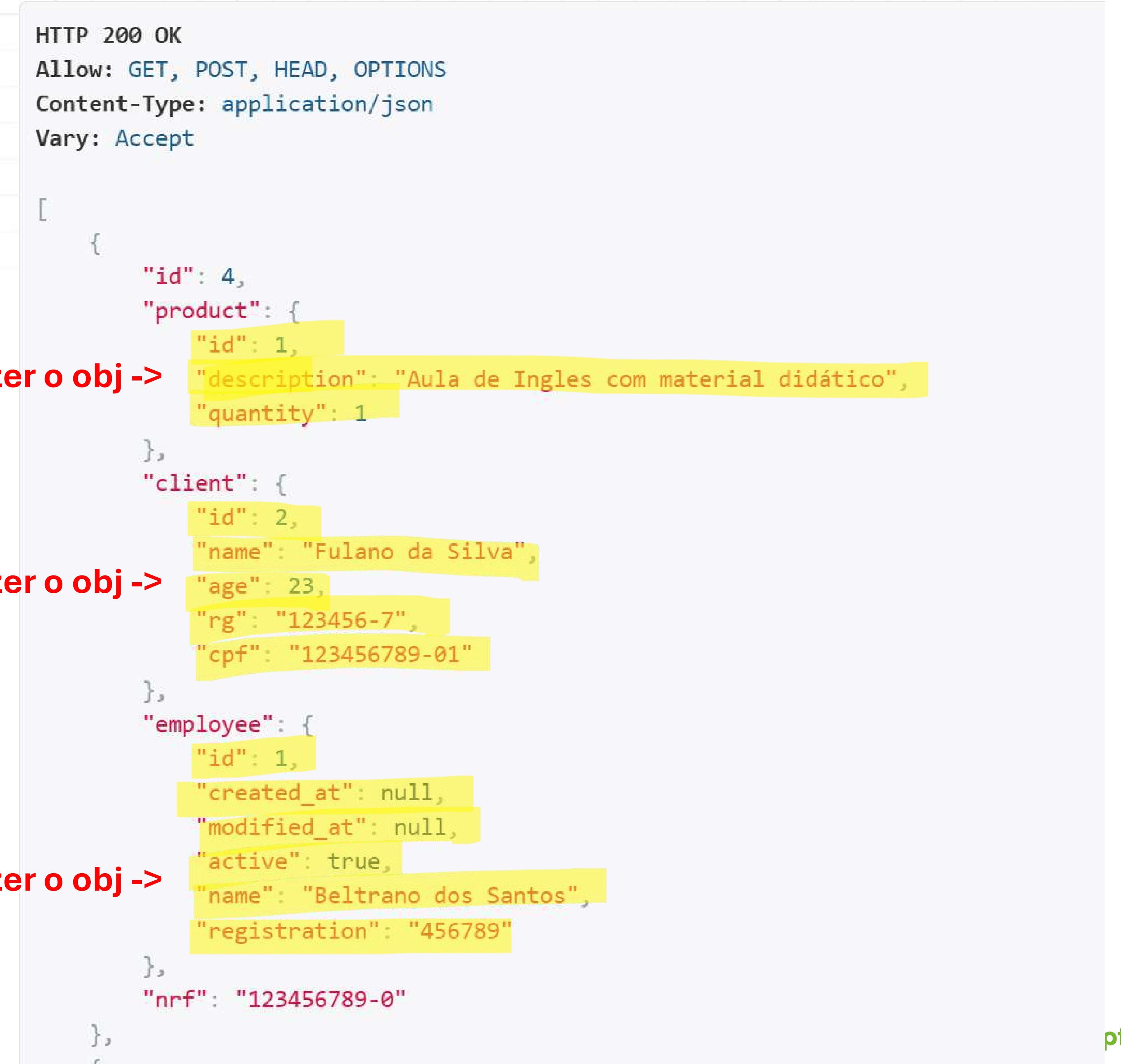
Employee List

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"
    }
]
```

FPFtech Escola Tecnológica

GET do sale



```
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 trazer o obj ->

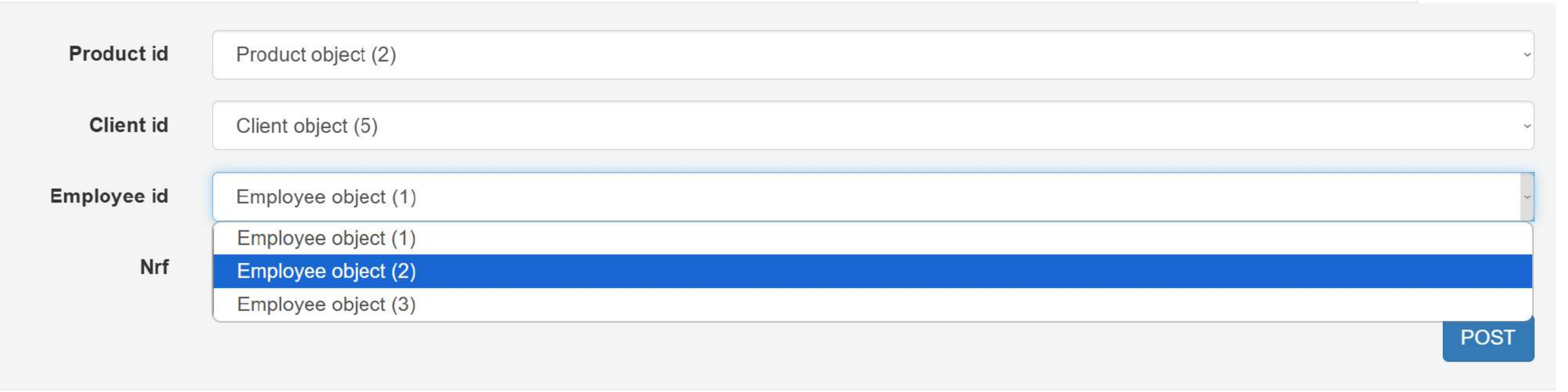
No serializer dizemos pra ele trazer o obj ->

No serializer dizemos pra ele trazer o obj ->

POST do sale

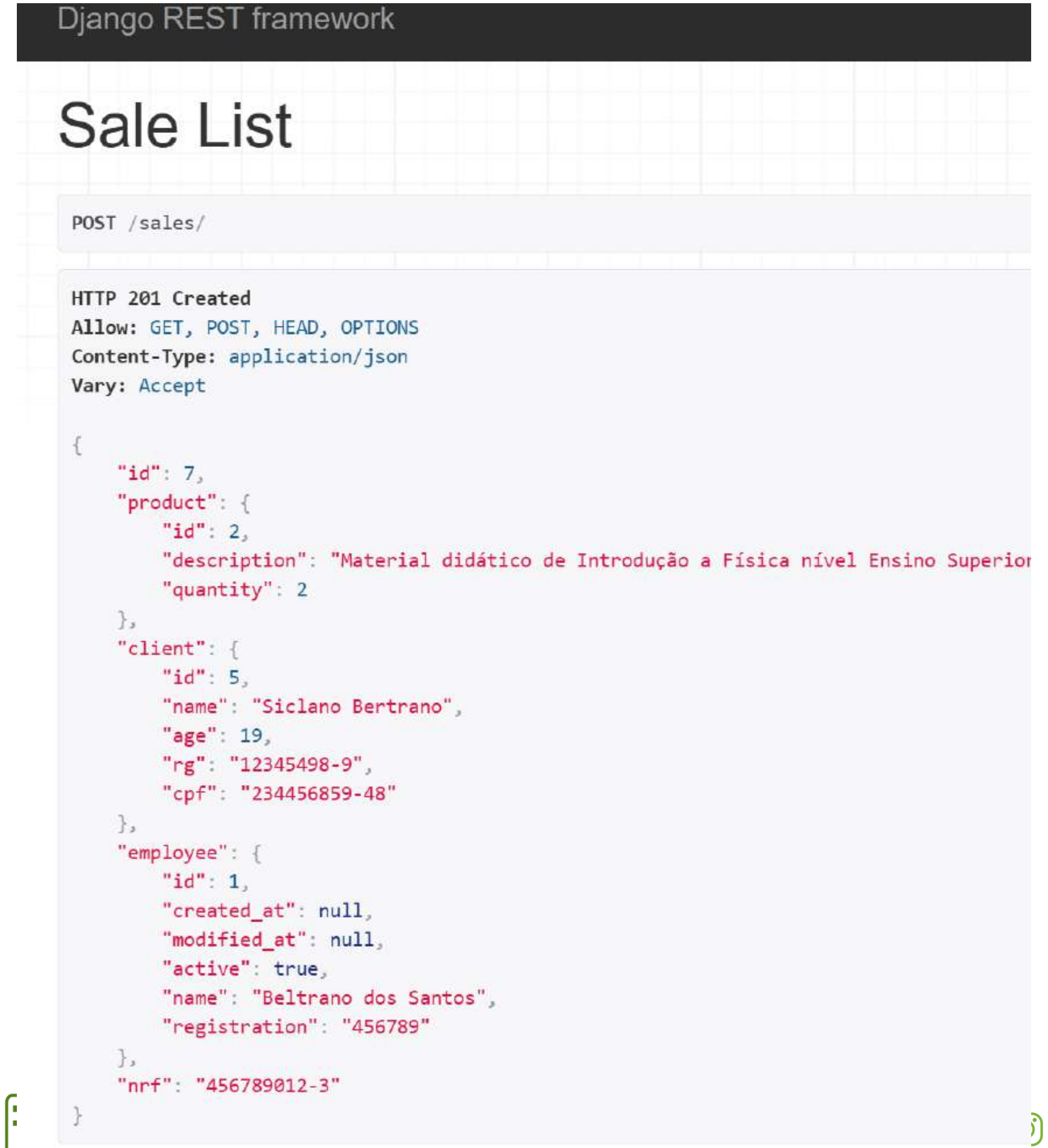
No serializer dizemos pra ele receber só o numero e ir buscar o obj

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	

Resultado do POST do sale



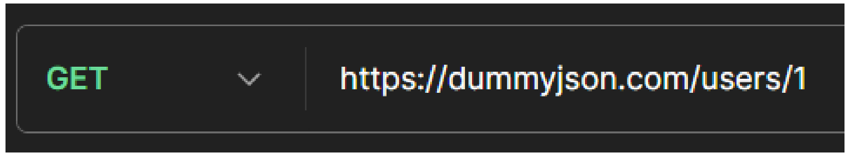
Django REST framework

Sale List

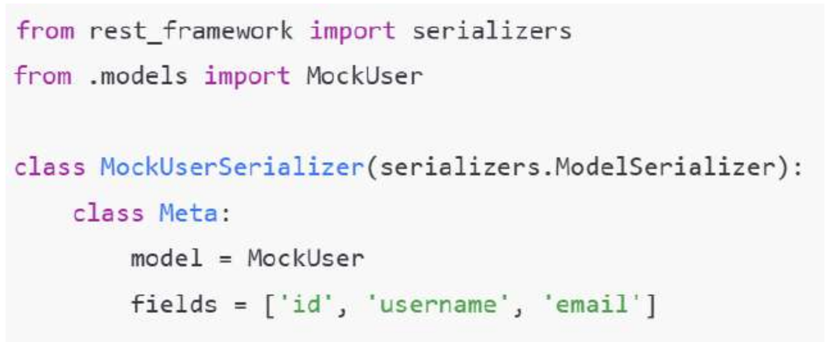
```
POST /sales/
```

```
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"
}
```

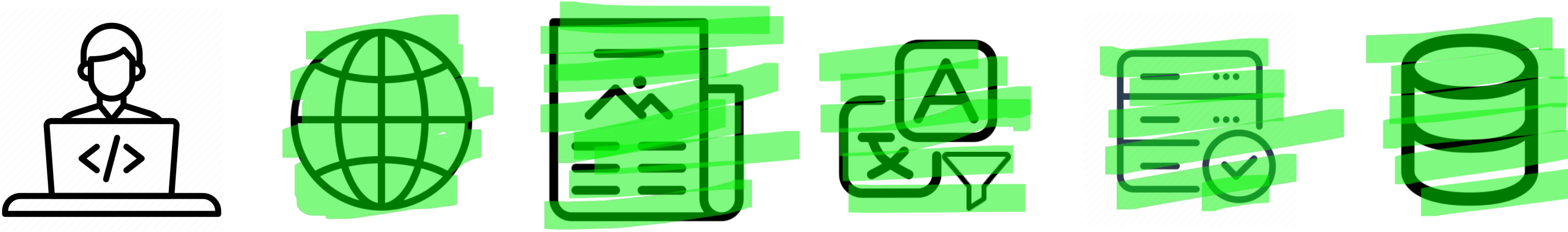
Ciclo de Requests



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



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

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



request

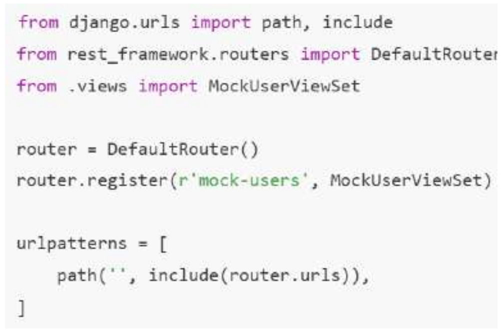
endpoint
urls

viewsets

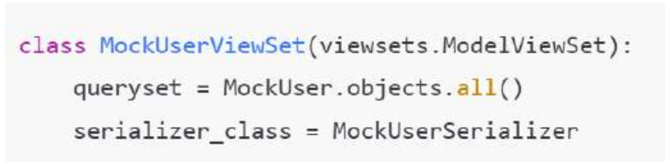
Serializers
JSON=Django

models

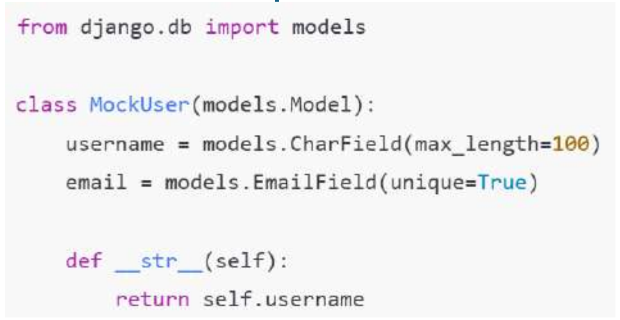
database



```python
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)),
]
```



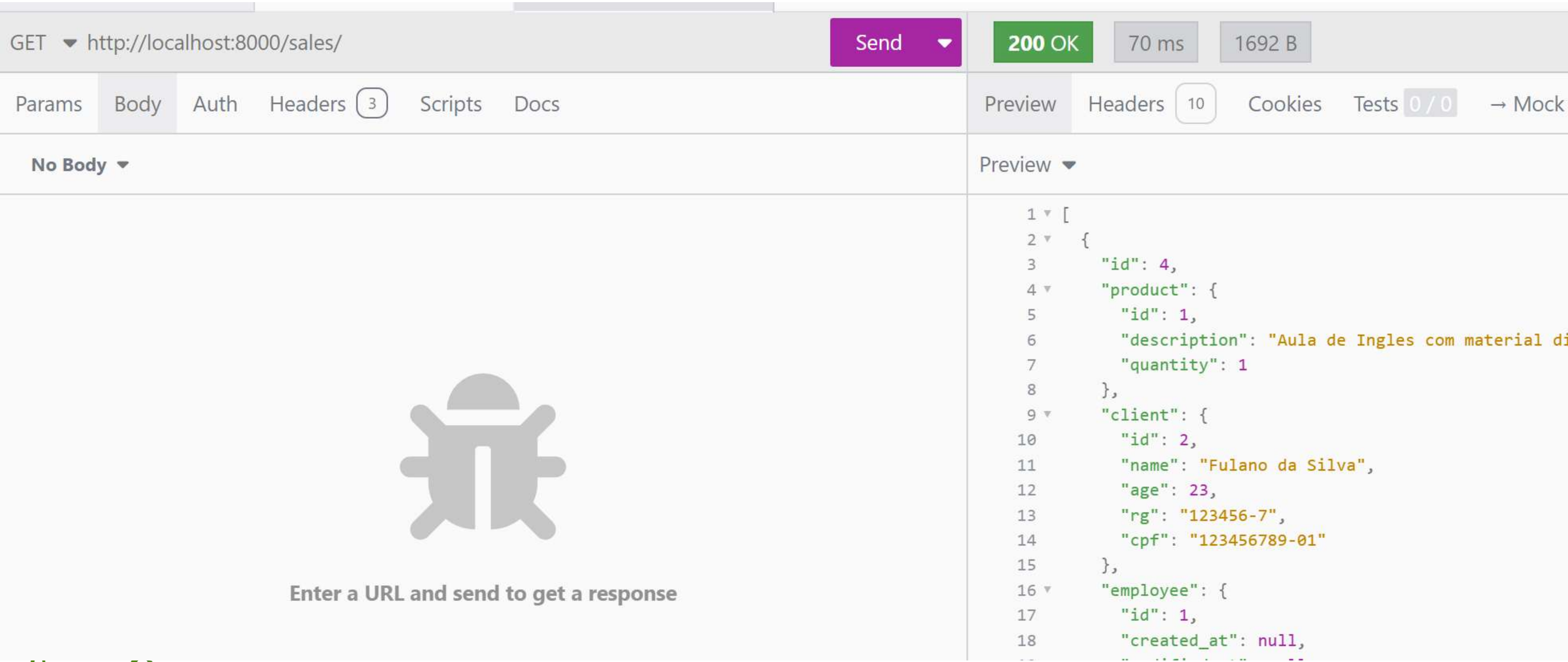
```python
class MockUserViewSet(viewsets.ModelViewSet):
    queryset = MockUser.objects.all()
    serializer_class = MockUserSerializer
```



```python
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

Exemplo GET 'sale' no insomnia

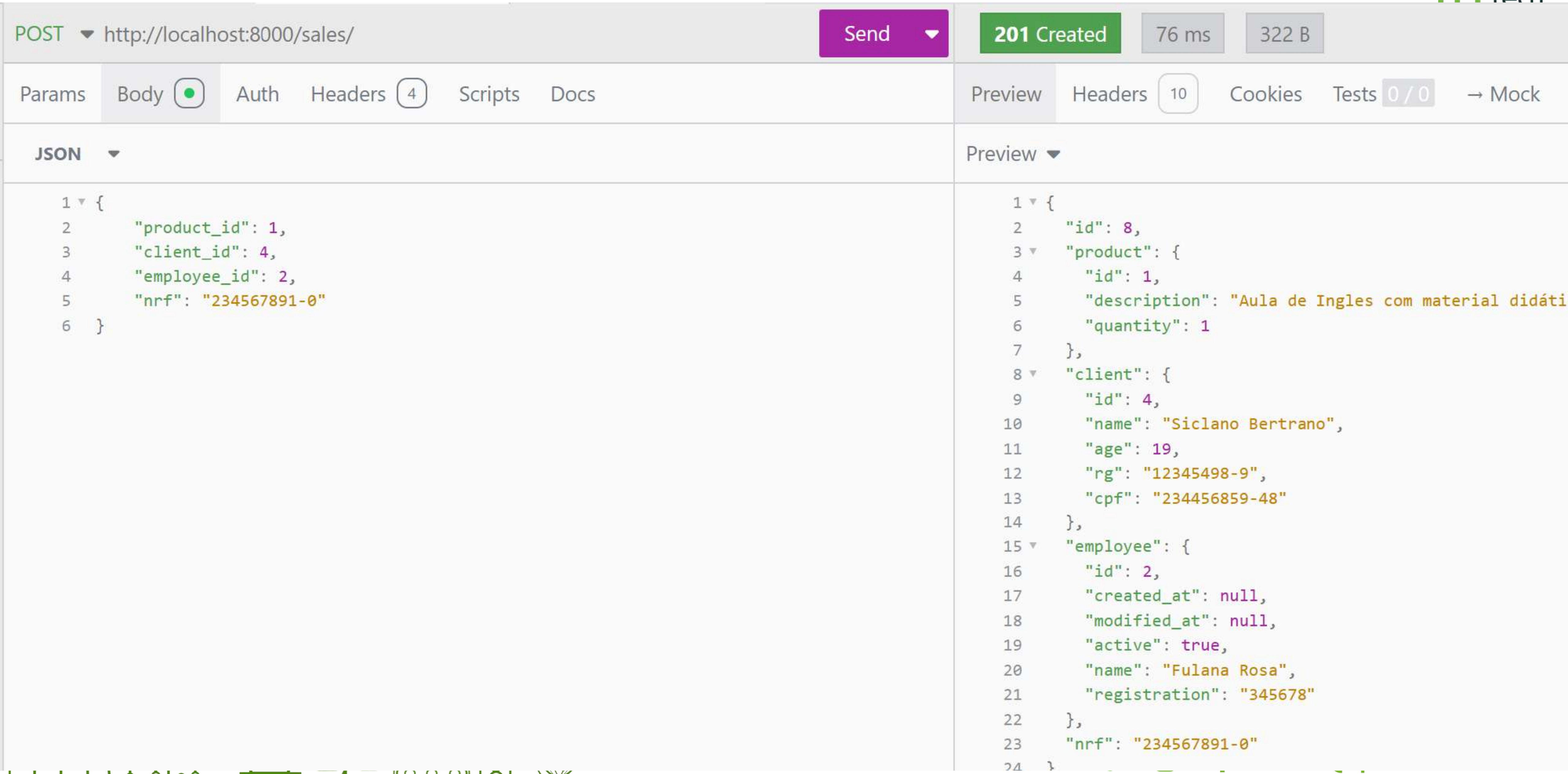


http://localhost:8000/sales/

```
[
  {
    "id": 4,
    "product": {
      "id": 1,
      "description": "Aula de Ingles com material di
      "quantity": 1
    },
    "client": {
      "id": 2,
      "name": "Fulano da Silva",
      "age": 23,
      "rg": "123456-7",
      "cpf": "123456789-01"
    },
    "employee": {
      "id": 1,
      "created_at": null,
```

@fpftech.educacional

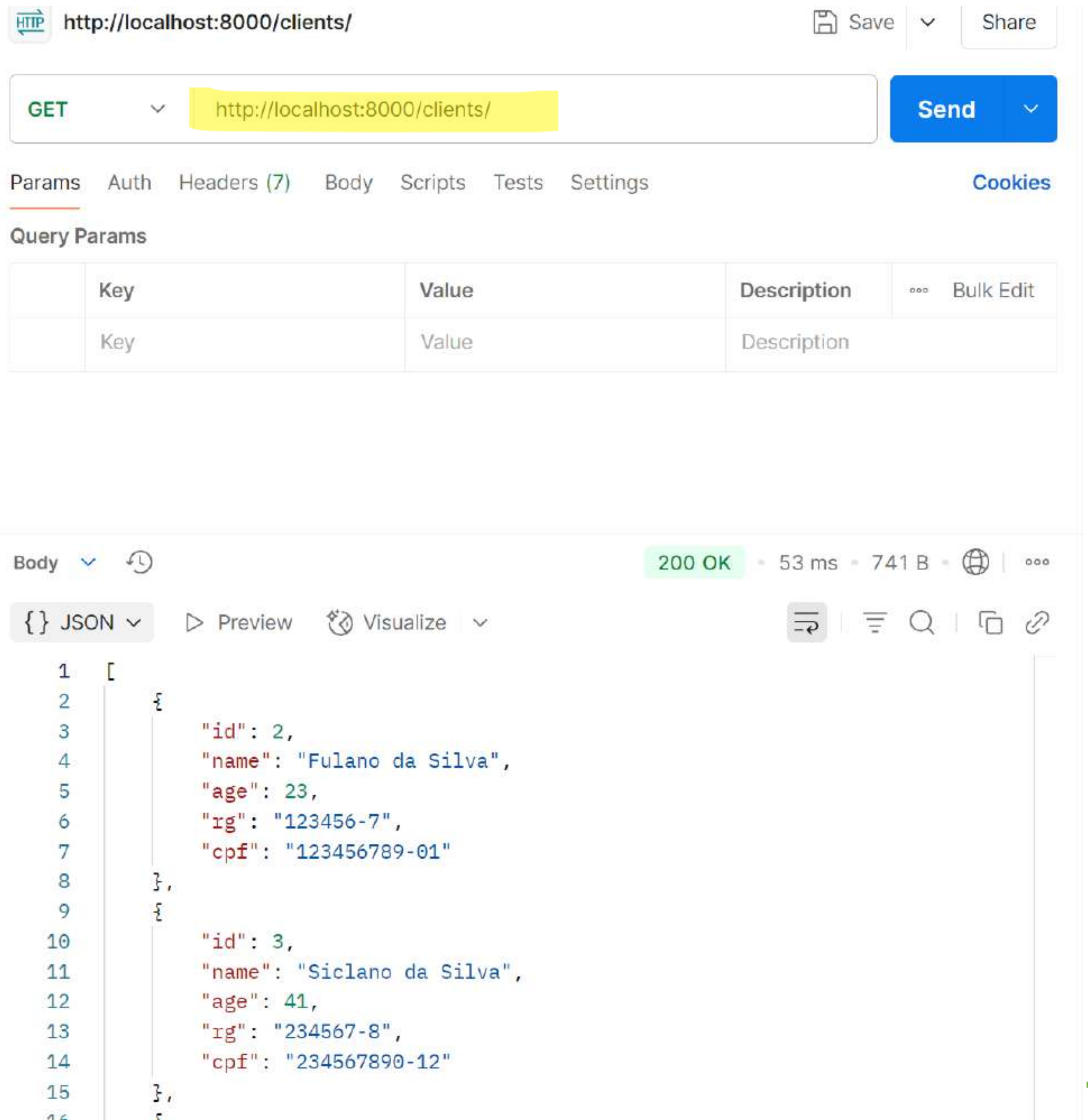
Exemplo POST ‘sale’ no insomnia



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

```
{
  "id": 8,
  "product": {
    "id": 1,
    "description": "Aula de Ingles com material didáti
    "quantity": 1
  },
  "client": {
    "id": 4,
    "name": "Siclano Bertrano",
    "age": 19,
    "rg": "12345498-9",
    "cpf": "234456859-48"
  },
  "employee": {
    "id": 2,
    "created_at": null,
    "modified_at": null,
    "active": true,
    "name": "Fulana Rosa",
    "registration": "345678"
  },
  "nrf": "234567891-0"
```

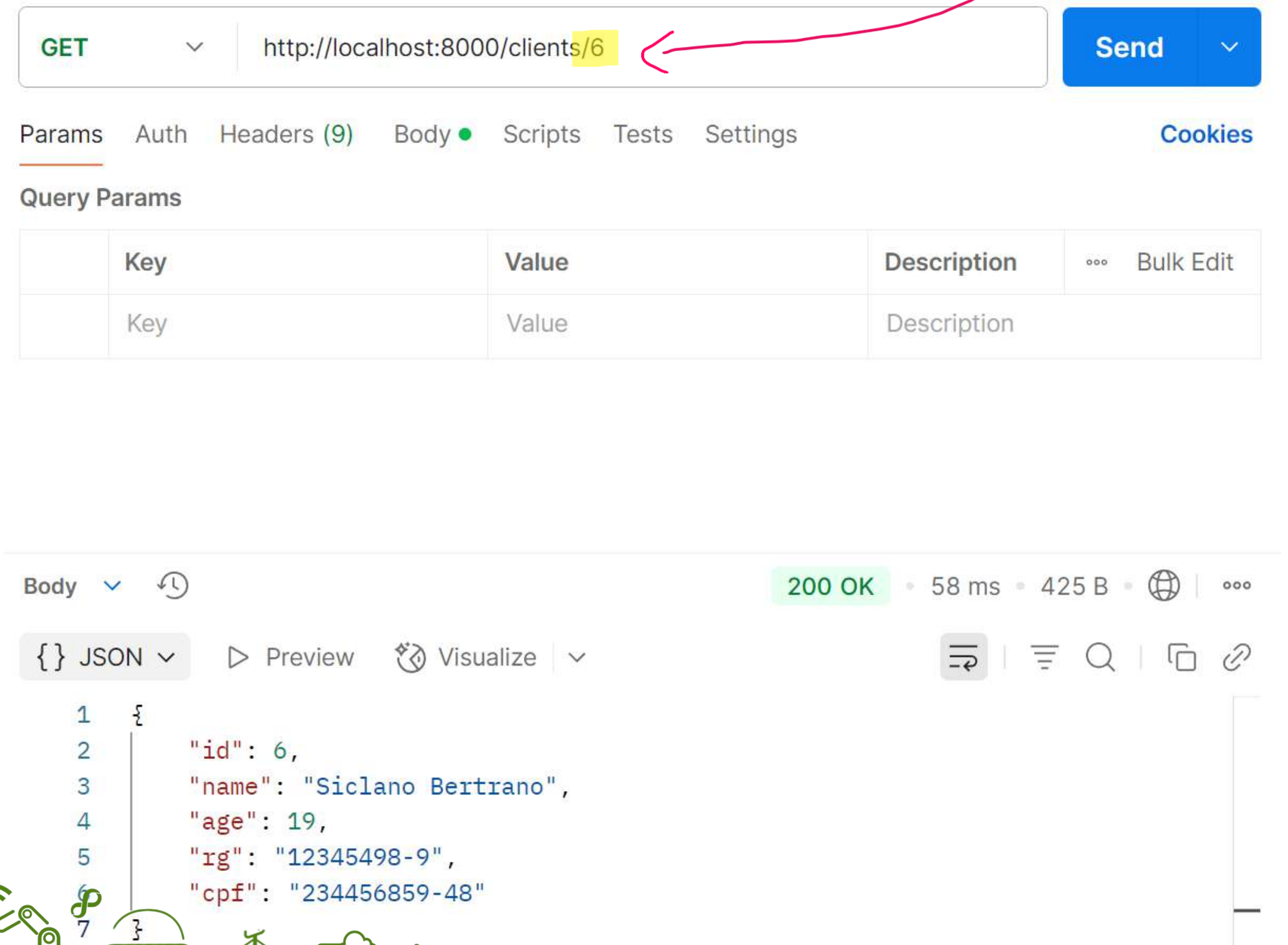
Exemplo GET ‘clients’ no postman



```
[
    {
        "id": 2,
        "name": "Fulano da Silva",
        "age": 23,
        "rg": "123456-7",
        "cpf": "123456789-01"
    },
    {
        "id": 3,
        "name": "Siclano da Silva",
        "age": 41,
        "rg": "234567-8",
        "cpf": "234567890-12"
    },
```

Exemplo GET ‘clients’ específico

Usando o id do cliente



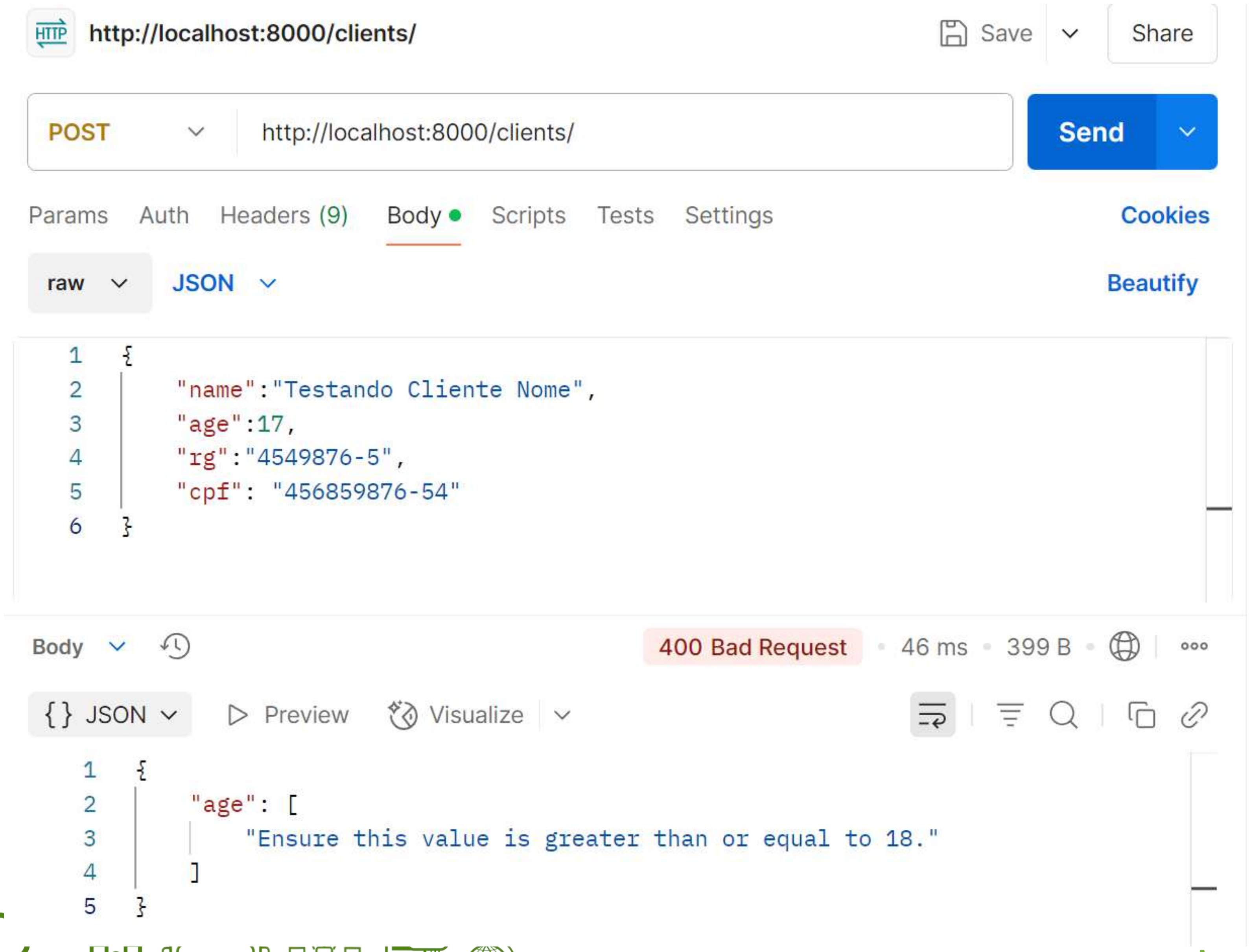
```
http://localhost:8000/clients/6
```

```
{
    "id": 6,
    "name": "Siclano Bertrano",
    "age": 19,
    "rg": "12345498-9",
    "cpf": "234456859-48"
}
```

FPFtech Escola Tecnológica

@fpftech.educacional

Exemplo POST ‘clients’ no postman



POST http://localhost:8000/clients/

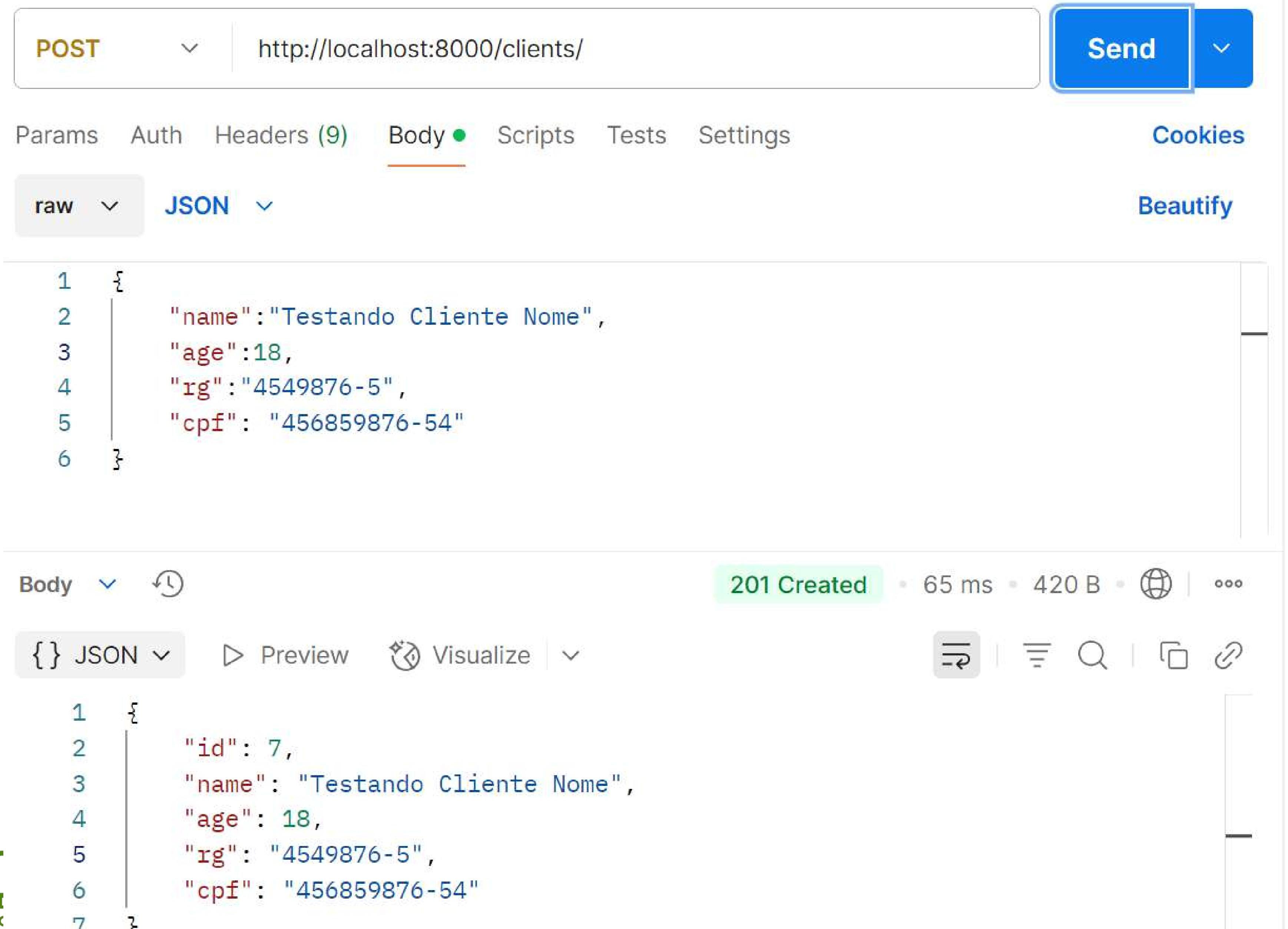
```
{
    "name":"Testando Cliente Nome",
    "age":17,
    "rg":"4549876-5",
    "cpf": "456859876-54"
}
```

400 Bad Request · 46 ms · 399 B

```
{
    "age": [
        "Ensure this value is greater than or equal to 18."
    ]
}
```

Por que deu errado?

Exemplo POST 'clients' no postman



POST http://localhost:8000/clients/

```
{
    "name":"Testando Cliente Nome",
    "age":18,
    "rg":"4549876-5",
    "cpf": "456859876-54"
}
```

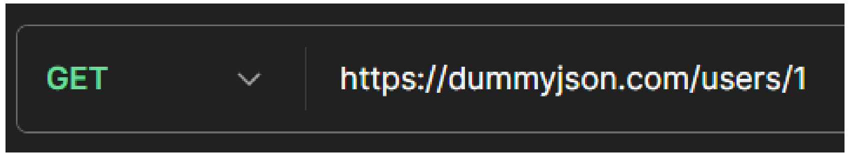
201 Created · 65 ms · 420 B

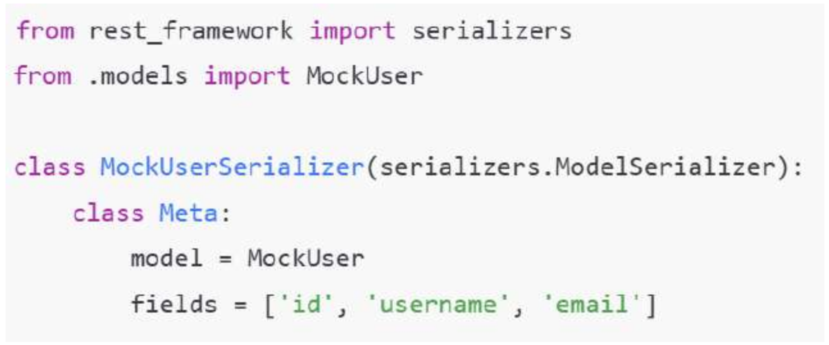
```
{
    "id": 7,
    "name": "Testando Cliente Nome",
    "age": 18,
    "rg": "4549876-5",
    "cpf": "456859876-54"
}
```

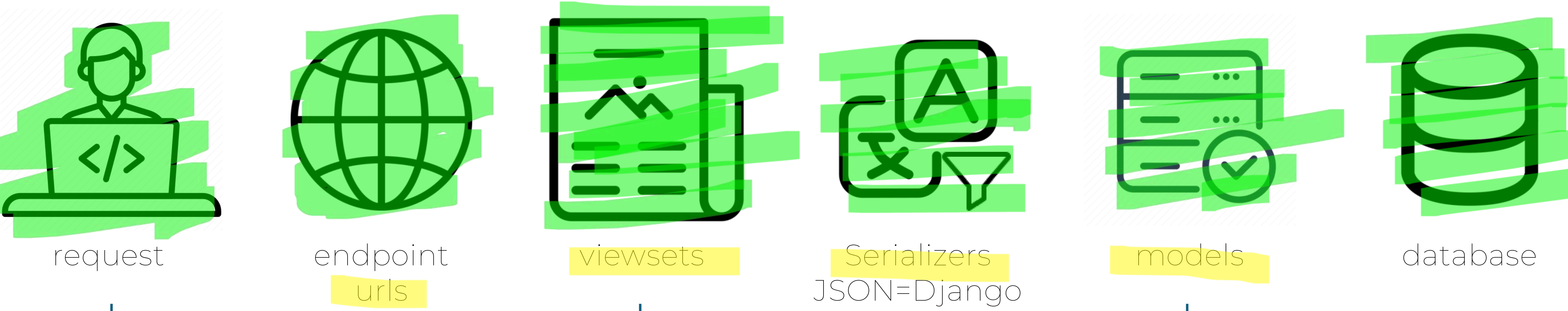
Agora sim!

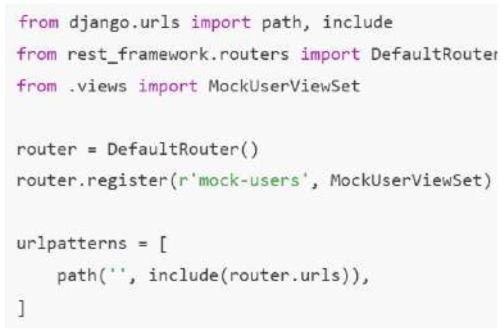
in @fpftech.educacional

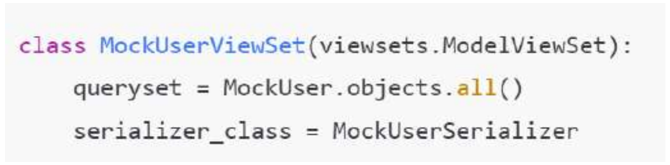
Ciclo de Requests

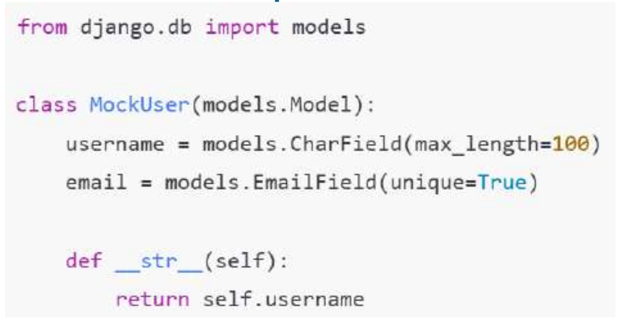

GET https://dummyjson.com/users/1


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

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




```python
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)),
]
```


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


```python
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
Escola Tecnológica

Obrigado!

FPFtech
Escola Tecnológica

@fpftech.educacional