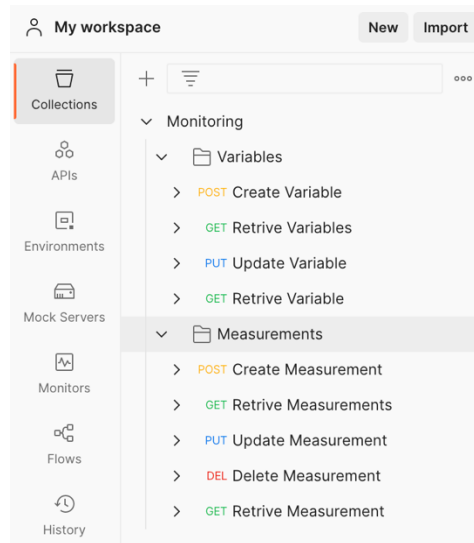


Universidad de Los Andes- Arquitectura y diseño de software
Talle Django – Vistas
Nicole Murillo Fonseca 202025521 n.murillof

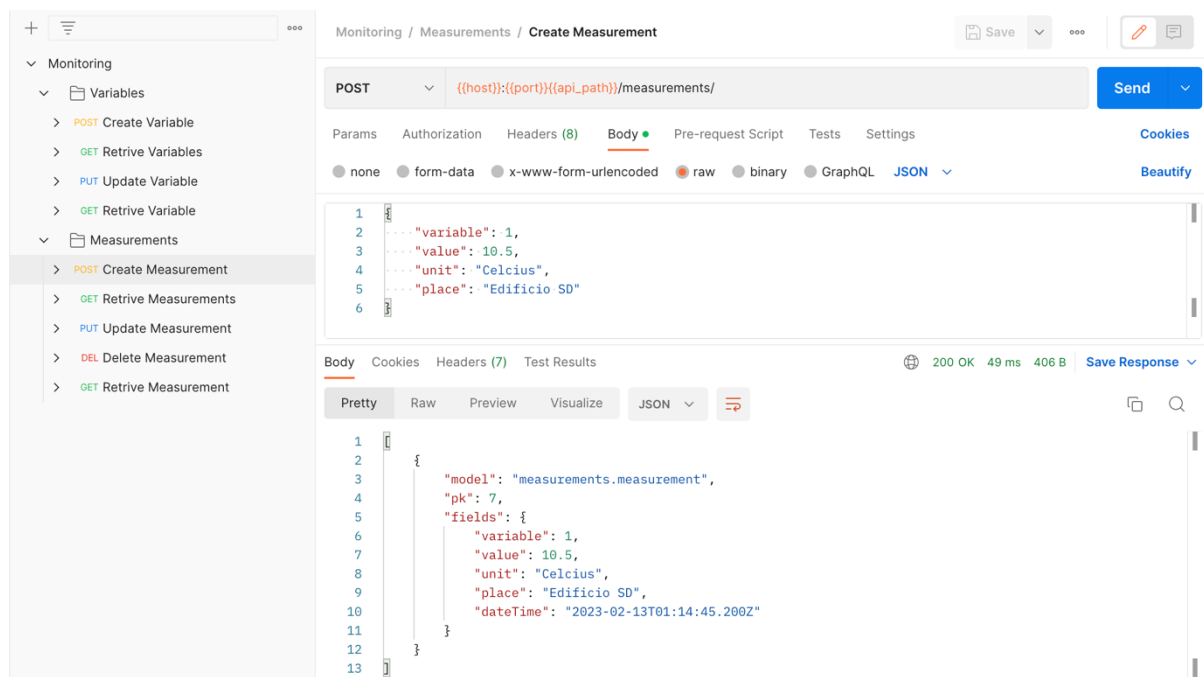
Ejecución de pruebas Postman

Evidencia de la creación de las pruebas para las 5 funciones nuevas

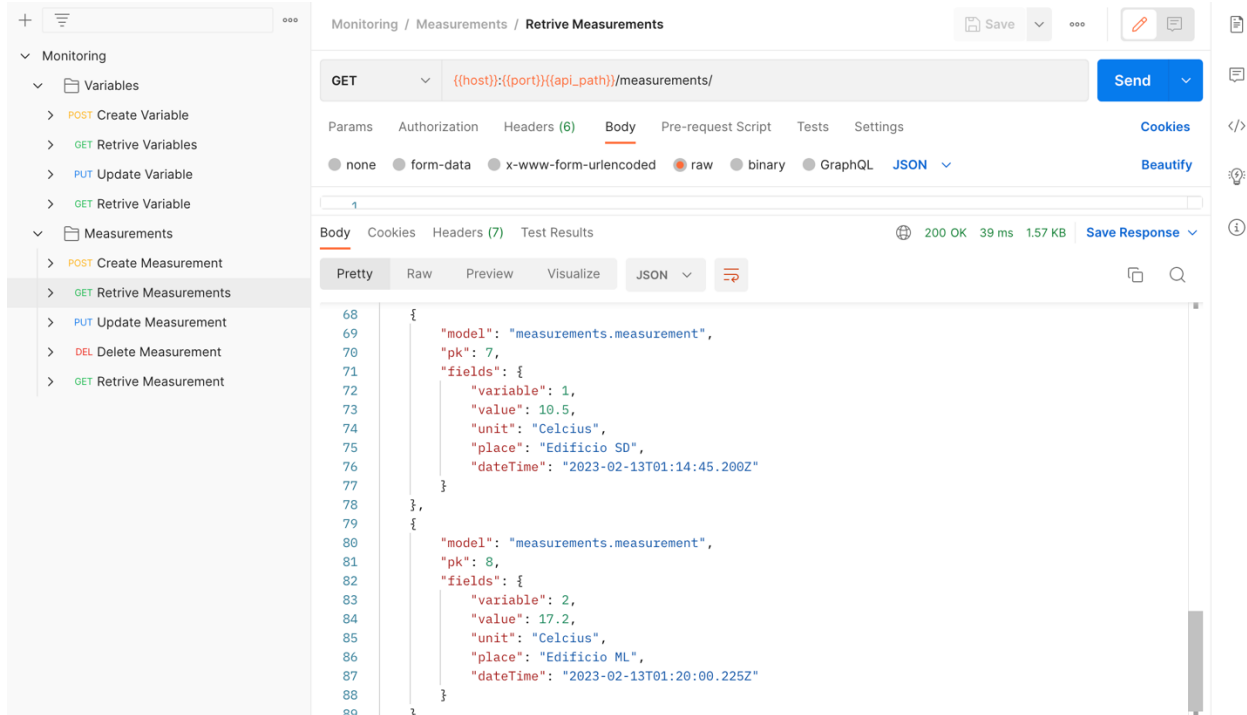


Evidencia de la ejecución exitosa de las pruebas para las 5 funciones

Create Measurement



Retrieve Measurements



Monitoring / Measurements / **Retrive Measurements**

GET `{{(host)}}:{{(port)}}{{(api_path)}}/measurements/` **Send**

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings **Cookies** **Beautify**

none form-data x-www-form-urlencoded raw binary GraphQL JSON

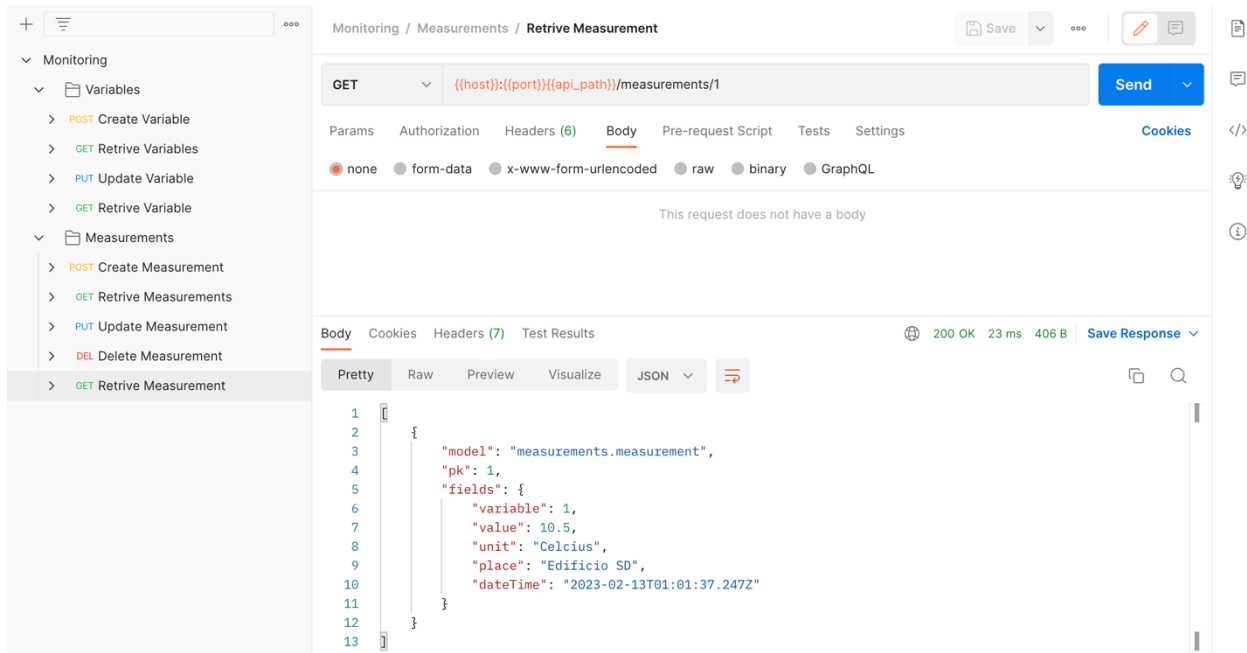
1

Body Cookies Headers (7) Test Results 200 OK 39 ms 1.57 KB **Save Response**

Pretty Raw Preview Visualize JSON

```
68 {
69   "model": "measurements.measurement",
70   "pk": 7,
71   "fields": {
72     "variable": 1,
73     "value": 10.5,
74     "unit": "Celcius",
75     "place": "Edificio SD",
76     "dateTime": "2023-02-13T01:14:45.206Z"
77   }
78 },
79 {
80   "model": "measurements.measurement",
81   "pk": 8,
82   "fields": {
83     "variable": 2,
84     "value": 17.2,
85     "unit": "Celcius",
86     "place": "Edificio ML",
87     "dateTime": "2023-02-13T01:20:00.225Z"
88   }
89 }
90 }
```

Retrieve Measurement



Monitoring / Measurements / **Retrive Measurement**

GET `{{(host)}}:{{(port)}}{{(api_path)}}/measurements/1` **Send**

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings **Cookies** **Beautify**

none form-data x-www-form-urlencoded raw binary GraphQL

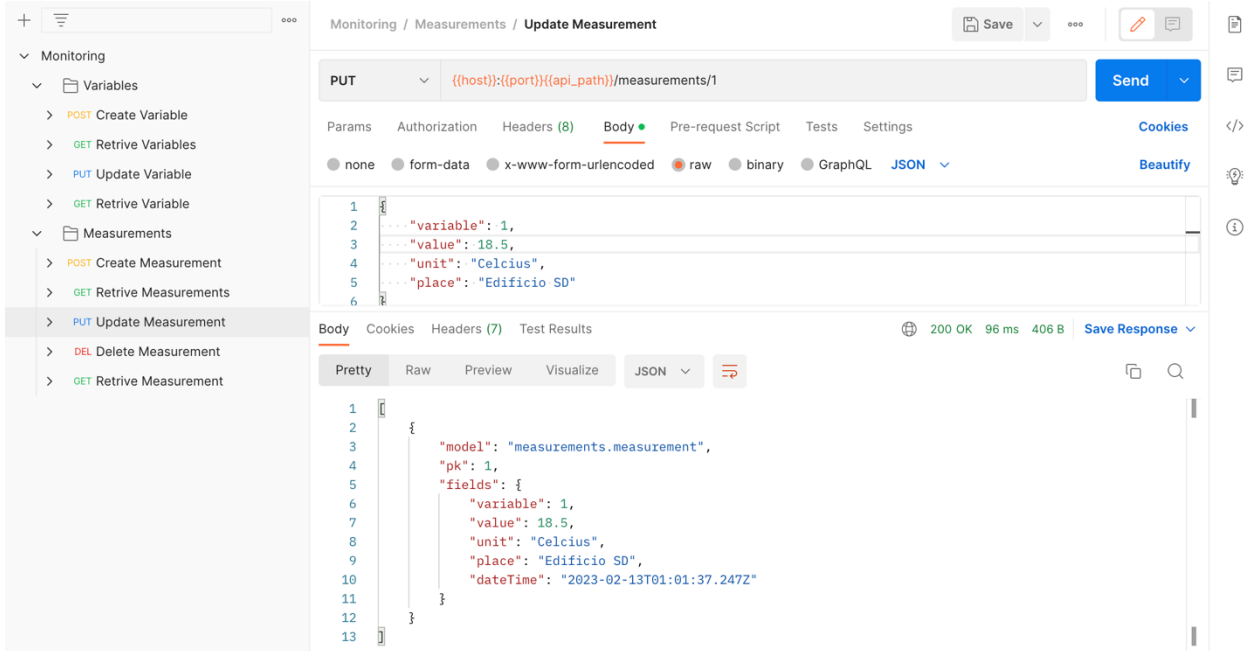
This request does not have a body

Body Cookies Headers (7) Test Results 200 OK 23 ms 406 B **Save Response**

Pretty Raw Preview Visualize JSON

```
1 {
2   "model": "measurements.measurement",
3   "pk": 1,
4   "fields": {
5     "variable": 1,
6     "value": 10.5,
7     "unit": "Celcius",
8     "place": "Edificio SD",
9     "dateTime": "2023-02-13T01:01:37.247Z"
10   }
11 }
12
13 }
```

Update Measurement



Monitoring / Measurements / Update Measurement

PUT {{(host)}}:{{(port)}}/{{(api_path)}}/measurements/1

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "variable": 1,
3   "value": 18.5,
4   "unit": "Celcius",
5   "place": "Edificio SD"
6 }

```

Body Cookies Headers (7) Test Results 200 OK 96 ms 406 B Save Response

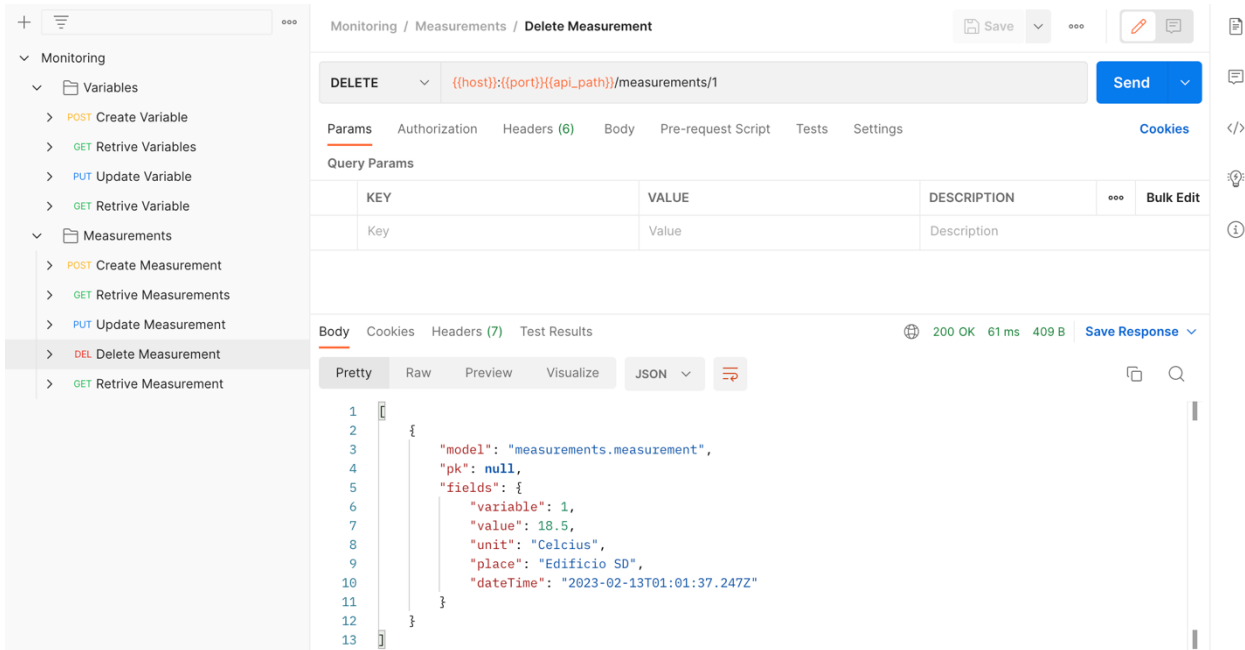
Pretty Raw Preview Visualize JSON

```

1 {
2   "model": "measurements.measurement",
3   "pk": 1,
4   "fields": {
5     "variable": 1,
6     "value": 18.5,
7     "unit": "Celcius",
8     "place": "Edificio SD",
9     "dateTime": "2023-02-13T01:01:37.247Z"
10  }
11 }
12
13

```

Delete Measurement



Monitoring / Measurements / Delete Measurement

DELETE {{(host)}}:{{(port)}}/{{(api_path)}}/measurements/1

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

Body Cookies Headers (7) Test Results 200 OK 61 ms 409 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "model": "measurements.measurement",
3   "pk": null,
4   "fields": {
5     "variable": 1,
6     "value": 18.5,
7     "unit": "Celcius",
8     "place": "Edificio SD",
9     "dateTime": "2023-02-13T01:01:37.247Z"
10  }
11 }
12
13

```

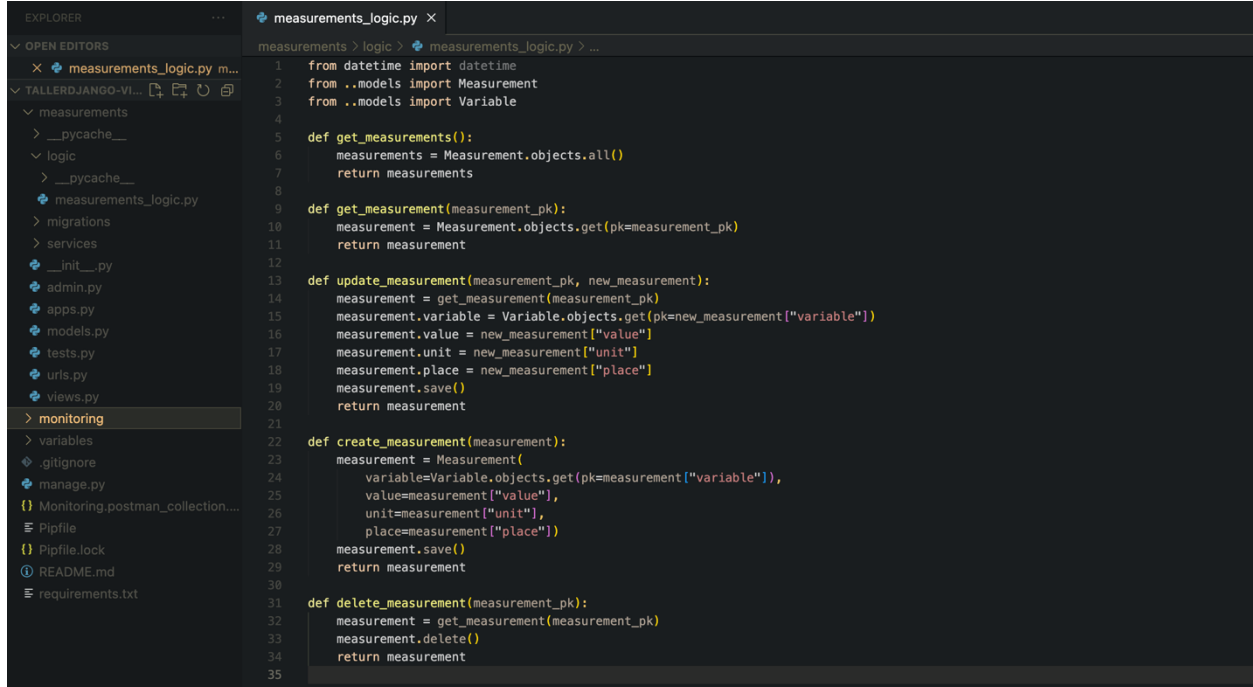
Explicación capturas de pantalla

Se crearon 5 pruebas de Postman para crear un measurement (POST), obtener un measurement (GET), obtener todos los measurements (GET), actualizar un measurement (PUT) y borrar un measurement (DELETE). En el cuerpo de crear y actualizar se pusieron los atributos

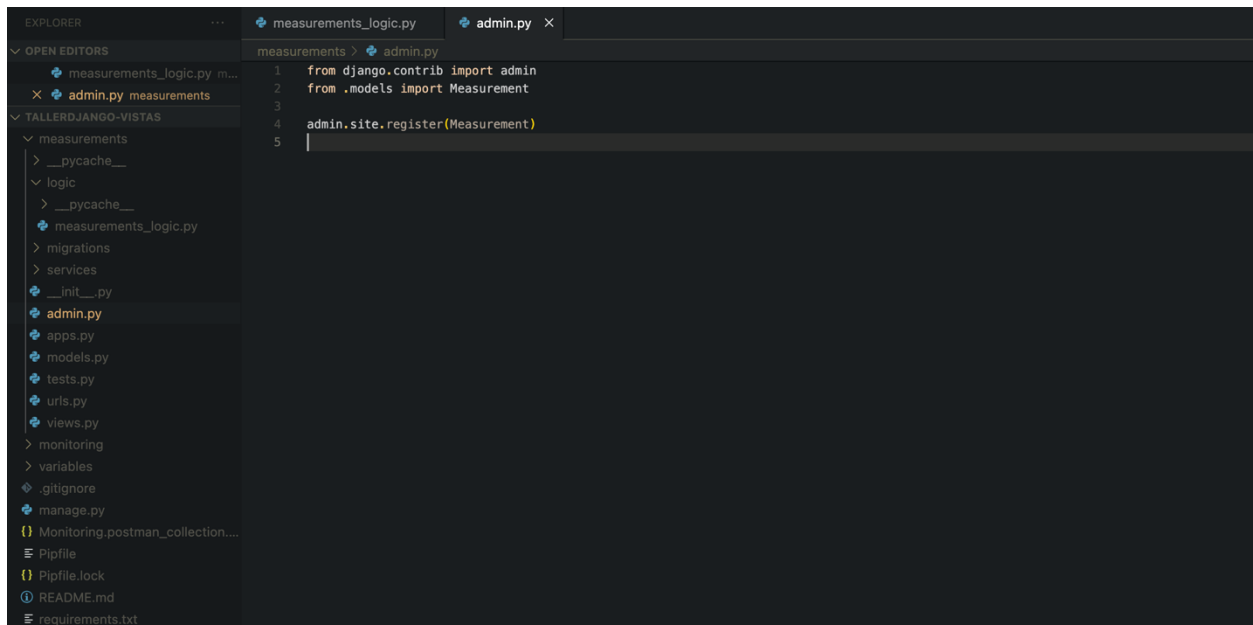
definidos en el models.py de la clase measurements del proyecto. Como se puede observar en las capturas de pantalla todas las pruebas ejecutaron exitosamente.

Definición de nuevas funciones

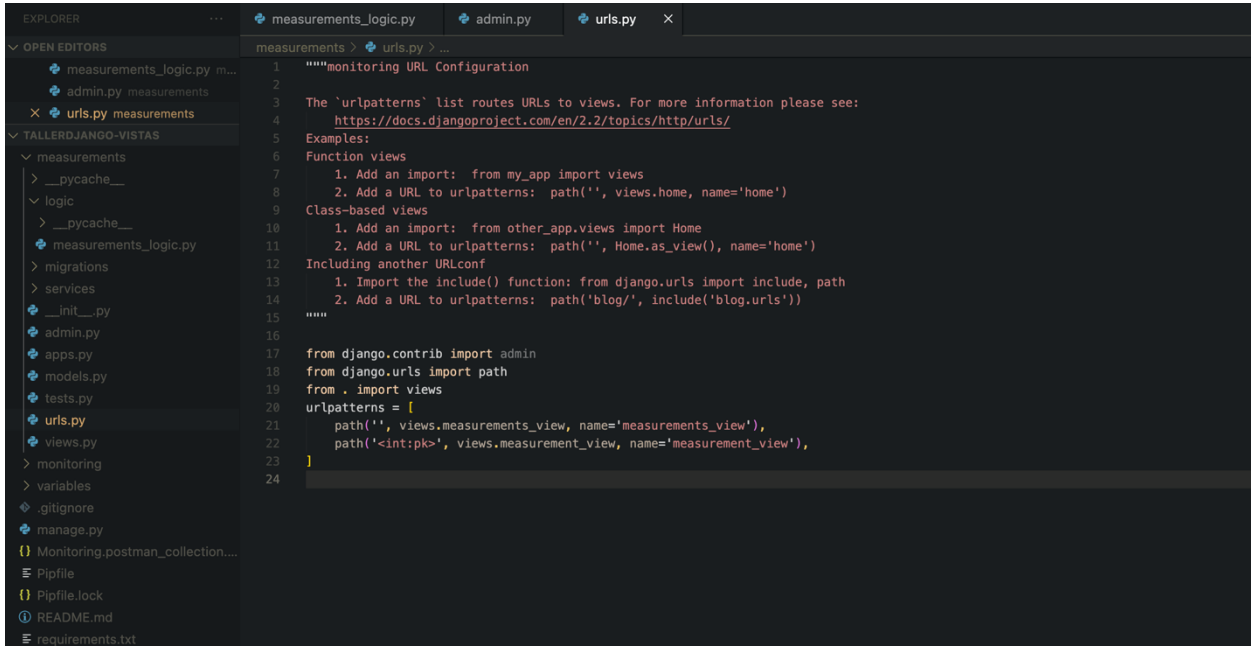
Evidencia de la creación de todas las nuevas funciones



```
1 from datetime import datetime
2 from ..models import Measurement
3 from ..models import Variable
4
5 def get_measurements():
6     measurements = Measurement.objects.all()
7     return measurements
8
9 def get_measurement(measurement_pk):
10    measurement = Measurement.objects.get(pk=measurement_pk)
11    return measurement
12
13 def update_measurement(measurement_pk, new_measurement):
14    measurement = get_measurement(measurement_pk)
15    measurement.variable = Variable.objects.get(pk=new_measurement["variable"])
16    measurement.value = new_measurement["value"]
17    measurement.unit = new_measurement["unit"]
18    measurement.place = new_measurement["place"]
19    measurement.save()
20    return measurement
21
22 def create_measurement(measurement):
23    measurement = Measurement(
24        variable=Variable.objects.get(pk=measurement["variable"]),
25        value=measurement["value"],
26        unit=measurement["unit"],
27        place=measurement["place"])
28    measurement.save()
29    return measurement
30
31 def delete_measurement(measurement_pk):
32    measurement = get_measurement(measurement_pk)
33    measurement.delete()
34    return measurement
35
```

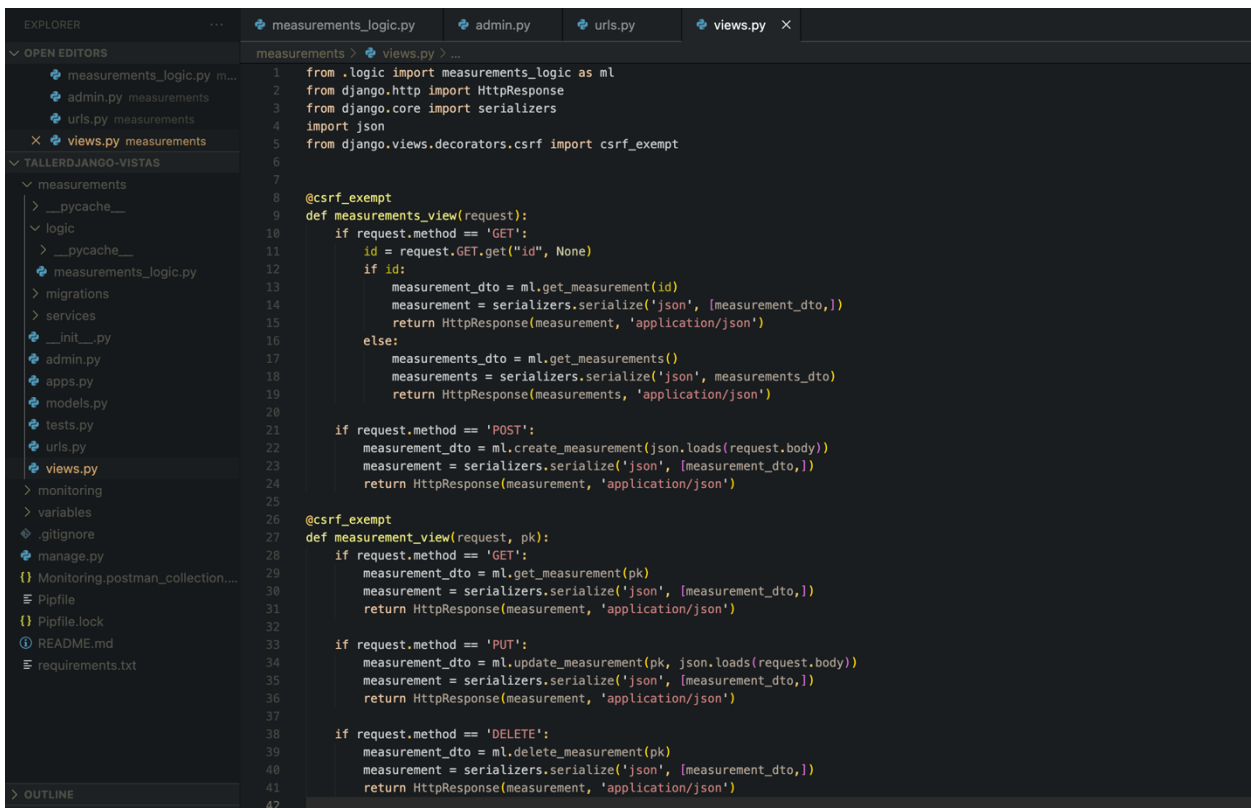


```
1 from django.contrib import admin
2 from ..models import Measurement
3
4 admin.site.register(Measurement)
5
```



```

1  """monitoring URL Configuration
2
3  The 'urlpatterns' list routes URLs to views. For more information please see:
4  https://docs.djangoproject.com/en/2.2/topics/http/urls/
5  Examples:
6  Function views
7      1. Add an import:  from my_app import views
8      2. Add a URL to urlpatterns:  path('', views.home, name='home')
9  Class-based views
10     1. Add an import:  from other_app.views import Home
11     2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
12 Including another URLconf
13     1. Import the include() function: from django.urls import include, path
14     2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
15 """
16
17 from django.contrib import admin
18 from django.urls import path
19 from . import views
20 urlpatterns = [
21     path('', views.measurements_view, name='measurements_view'),
22     path('<int:pk>', views.measurement_view, name='measurement_view'),
23 ]
  
```



```

1  from .logic import measurements_logic as ml
2  from django.http import HttpResponse
3  from django.core import serializers
4  import json
5  from django.views.decorators.csrf import csrf_exempt
6
7
8  @csrf_exempt
9  def measurements_view(request):
10     if request.method == 'GET':
11         id = request.GET.get("id", None)
12         if id:
13             measurement_dto = ml.get_measurement(id)
14             measurement = serializers.serialize('json', [measurement_dto,])
15             return HttpResponse(measurement, 'application/json')
16         else:
17             measurements_dto = ml.get_measurements()
18             measurements = serializers.serialize('json', measurements_dto)
19             return HttpResponse(measurements, 'application/json')
20
21     if request.method == 'POST':
22         measurement_dto = ml.create_measurement(json.loads(request.body))
23         measurement = serializers.serialize('json', [measurement_dto,])
24         return HttpResponse(measurement, 'application/json')
25
26 @csrf_exempt
27 def measurement_view(request, pk):
28     if request.method == 'GET':
29         measurement_dto = ml.get_measurement(pk)
30         measurement = serializers.serialize('json', [measurement_dto,])
31         return HttpResponse(measurement, 'application/json')
32
33     if request.method == 'PUT':
34         measurement_dto = ml.update_measurement(pk, json.loads(request.body))
35         measurement = serializers.serialize('json', [measurement_dto,])
36         return HttpResponse(measurement, 'application/json')
37
38     if request.method == 'DELETE':
39         measurement_dto = ml.delete_measurement(pk)
40         measurement = serializers.serialize('json', [measurement_dto,])
41         return HttpResponse(measurement, 'application/json')
42
  
```

Explicación capturas de pantalla

Se hicieron cambios en measurements_logic.py, admin.py, urls.py y views.py de la carpeta measurements para ejecutar las pruebas de Postman. En los archivos measurements_logic.py y views.py se agregaron 5 funciones nuevas para crear un measurement (POST), obtener un

measurement (GET), obtener todos los measurements (GET), actualizar un measurement (PUT) y borrar un measurement (DELETE); de manera que, estas pudieran responder a las solicitudes (requests) de Postman. Todo ejecutó exitosamente y las funciones respondieron como se esperaba.