

Introducción

Este proyecto de cocina desarrollado con Django incluye diversas funcionalidades que permiten gestionar recetas, comentarios, listas de compras y desafíos culinarios. La estructura del proyecto se divide en varias aplicaciones: cocina y desafíos, cada una con modelos, vistas y rutas propias. A través de una API REST, los usuarios pueden interactuar con los diferentes recursos disponibles, desde la creación de recetas hasta la participación en desafíos culinarios.

Estructura del Proyecto y código

- **Aplicación cocina:**

Modelos: Receta, Comentario, ListaDeCompras

```
3 class Receta(models.Model):
4     nombre = models.CharField(max_length=255)
5     ingredientes = models.TextField()
6     instrucciones = models.TextField()
7     tiempo_preparacion = models.IntegerField(help_text="Tiempo en minutos")
8     categoria = models.CharField(max_length=100)
9     imagen = models.ImageField(upload_to='imagenes_recetas/', null=True, blank=True)
10
11     def __str__(self):
12         return self.nombre
13
14 class Comentario(models.Model):
15     autor = models.CharField(max_length=100)
16     contenido = models.TextField()
17     fecha_publicacion = models.DateTimeField(auto_now_add=True)
18     receta = models.ForeignKey(Receta, related_name='comentarios', on_delete=models.CASCADE)
19
20     def __str__(self):
21         return f'Comentario de {self.autor} en {self.receta.nombre}'
22
23 class ListaDeCompras(models.Model):
24     nombre = models.CharField(max_length=255)
25     recetas = models.ManyToManyField(Receta)
26     fecha_creacion = models.DateTimeField(auto_now_add=True)
27
28     def __str__(self):
29         return self.nombre
```

Serializadores: Serializadores para cada modelo.

```
4 class RecetaSerializer(serializers.ModelSerializer):
5     class Meta:
6         model = Receta
7         fields = '__all__'
8
9 class ComentarioSerializer(serializers.ModelSerializer):
10     class Meta:
11         model = Comentario
12         fields = '__all__'
13
14 class ListaDeComprasSerializer(serializers.ModelSerializer):
15     class Meta:
16         model = ListaDeCompras
17         fields = '__all__'
```

Vistas Genéricas: Vistas para listar, crear, actualizar y eliminar recetas, comentarios y listas de compras.

```

8 # Vistas Genéricas para Receta
You, 17 hours ago | 1 author (You)
9 class RecetaList(generics.ListCreateAPIView):
10     queryset = Receta.objects.all()
11     serializer_class = RecetaSerializer
12
13 You, 17 hours ago | 1 author (You)
14 class RecetaDetail(generics.RetrieveUpdateDestroyAPIView):
15     queryset = Receta.objects.all()
16     serializer_class = RecetaSerializer
17
18 # Vistas Genéricas para Comentario
19 You, 17 hours ago | 1 author (You)
20 class ComentarioList(generics.ListCreateAPIView):
21     queryset = Comentario.objects.all()
22     serializer_class = ComentarioSerializer
23
24 You, 17 hours ago * Primer paso práctica django hecha
25
26 You, 17 hours ago | 1 author (You)
27 class ComentarioDetail(generics.RetrieveUpdateDestroyAPIView):
28     queryset = Comentario.objects.all()
29     serializer_class = ComentarioSerializer
30
31 # Vistas Genéricas para ListaDeCompras
32 You, 17 hours ago | 1 author (You)
33 class ListaDeComprasList(generics.ListCreateAPIView):
34     queryset = ListaDeCompras.objects.all()
35     serializer_class = ListaDeComprasSerializer
36
37 You, 17 hours ago | 1 author (You)
38 class ListaDeComprasDetail(generics.RetrieveUpdateDestroyAPIView):
39     queryset = ListaDeCompras.objects.all()
40     serializer_class = ListaDeComprasSerializer

```

Vistas Personalizadas: Vista para listar comentarios de una receta específica.

```

35 # api_view Propia para listar comentarios de una receta específica
36 @api_view(['GET'])
37 def comentarios_de_receta(request, receta_id):
38     try:
39         receta = Receta.objects.get(id=receta_id)
40     except Receta.DoesNotExist:
41         return Response(status=status.HTTP_404_NOT_FOUND)
42
43     comentarios = Comentario.objects.filter(receta=receta)
44     serializer = ComentarioSerializer(comentarios, many=True)
45     return Response(serializer.data)

```

URLs:

Listar y crear recetas: <http://localhost:8000/api/recetas/>

Crear con el método POST: (2 ejemplos)

Nombre: Arroz al horno

Ingredientes: Arroz bomba 4 cacitos
Caldo de cocido 8 cacitos

Instrucciones: Empezamos poniendo el caldo a calentar. Mientras tanto, lavamos, pelamos y cortamos en rodajas la patata, luego la doramos brevemente en una sartén y reservamos.

Si tenemos fuego a gas, colocamos la cazuela de barro sobre el fuego y ponemos a dorar la carne y la panceta troceada con un poco de aceite. Si no, lo hacemos en una sartén y luego lo añadiremos a la cazuela antes de hornear. Doramos también brevemente la morcilla y rehogamos el arroz y los garbanzos.

Antes de añadir el caldo, ajustamos de sal (hay quien añade azafrán o colorante para que coja color), distribuimos las rodajas de patata, colocamos una cabeza de ajo en el centro y ponemos un tomate partido por la mitad. También hay quien rehoga el tomate con un poco de pimentón o de tomate frito.

Por último vertemos el caldo y hornearmos durante 19 minutos con el horno a 250 °C. Recordad que la proporción de caldo es exactamente dos a uno. Es decir, dos partes de caldo por una de arroz. Antes de servir, dejamos que el arroz repose unos minutos tapado con un paño, para que acabe la cocción.

Tiempo preparacion: 60
Tiempo en minutos

Categoria: Arroces

Imagen: Seleccionar archivo | Arroz al horno.jpg

POST

Nombre

Paella Valenciana

Ingredientes

Arroz bomba 1500 g
Pollo de corral 1

Instrucciones

Toda paella que se precie comienza por un buen sofrito. En una paella cuanto más grande mejor, se sofríe en abundante aceite el pollo, el conejo, las judías, las alcachofas y los caracoles (la que veis en la foto no tiene garrofó porque no es temporada y el congelado no es igual), sazonando con un poco de sal y pimentón hacia el final. Cuando esté bien dorado se añade el tomate triturado y se rehoga.

Con el sofrito listo se debe de añadir el agua. Las proporciones dependen mucho del fuego, del calor que haga, del grado de humedad y de lo grande que sea la paella, pero para comenzar, una buena proporción es la de añadir tres veces el volumen de agua que de arroz, aunque es la experiencia la que os hará ajustar y perfeccionar estas cantidades, que acabareís haciendo a ojo, como hicieron la tía y la madre de mi novia, que eran las encargadas de esta paella (a pesar de que la tradición marca que sea el hombre de la casa el que la prepare).

Echamos ahora algunos troncos más al fuego para que suba de potencia y se haga bien el caldo durante 25 o 30 minutos. Es un buen momento de echar el azafrán o, en su defecto, el sazónador de paella (el más popular es "el paellador", que lleva sal, ajo, colorante y un poco de azafrán).

Luego añadimos el arroz "en caballete" (en diagonal) y lo distribuimos por la paella. Cocemos entre 17 y 20 minutos, aunque aquí el tiempo lo marca de nuevo el grano de arroz y la potencia del fuego, que debemos ir dejando consumirse. Tiene que quedar completamente seco y

Tiempo preparación

90

Categoría

Arroces

Imagen

Seleccionar archivo

paella.png

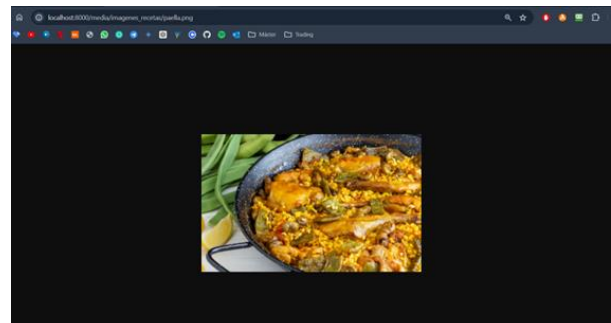
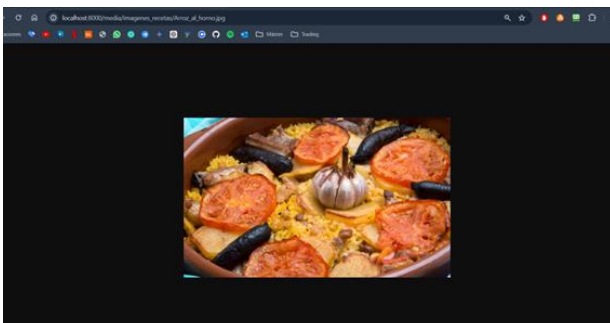
Listar las recetas con el método GET:

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

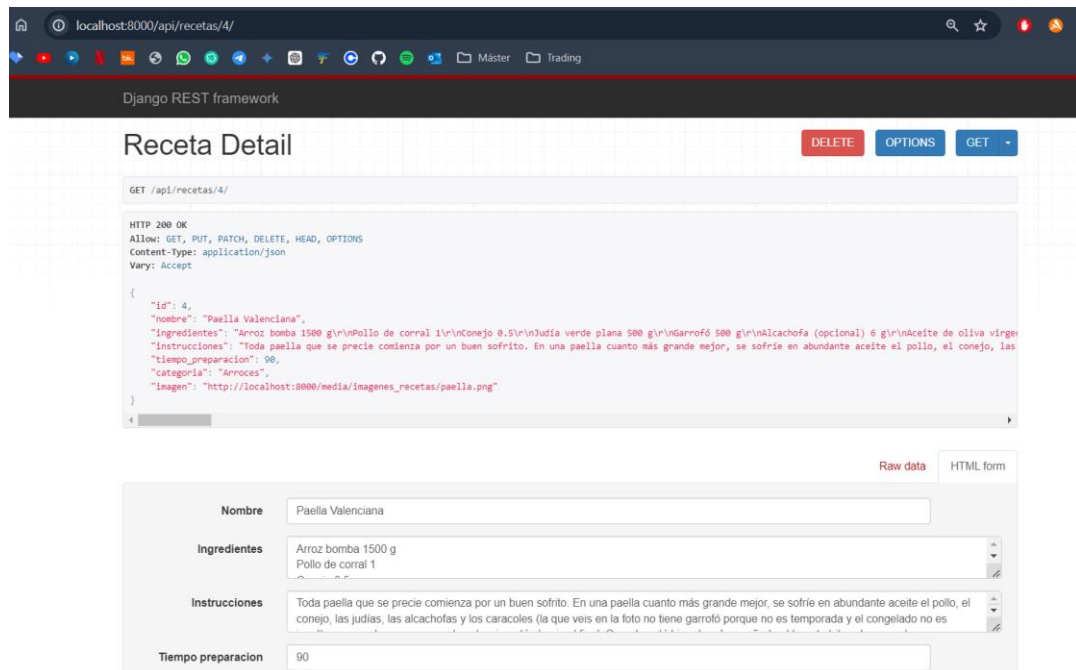
```

Así como mostrar las imágenes adjuntas:

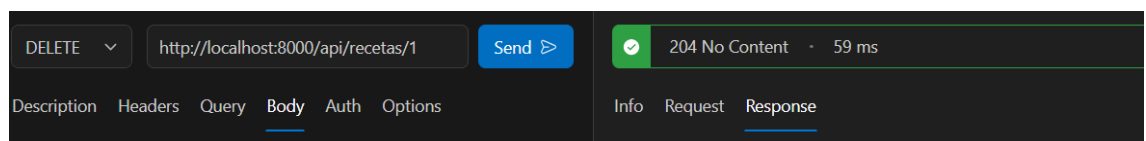


Detallar, actualizar y eliminar una receta: <http://localhost:8000/api/recetas/<id>/>

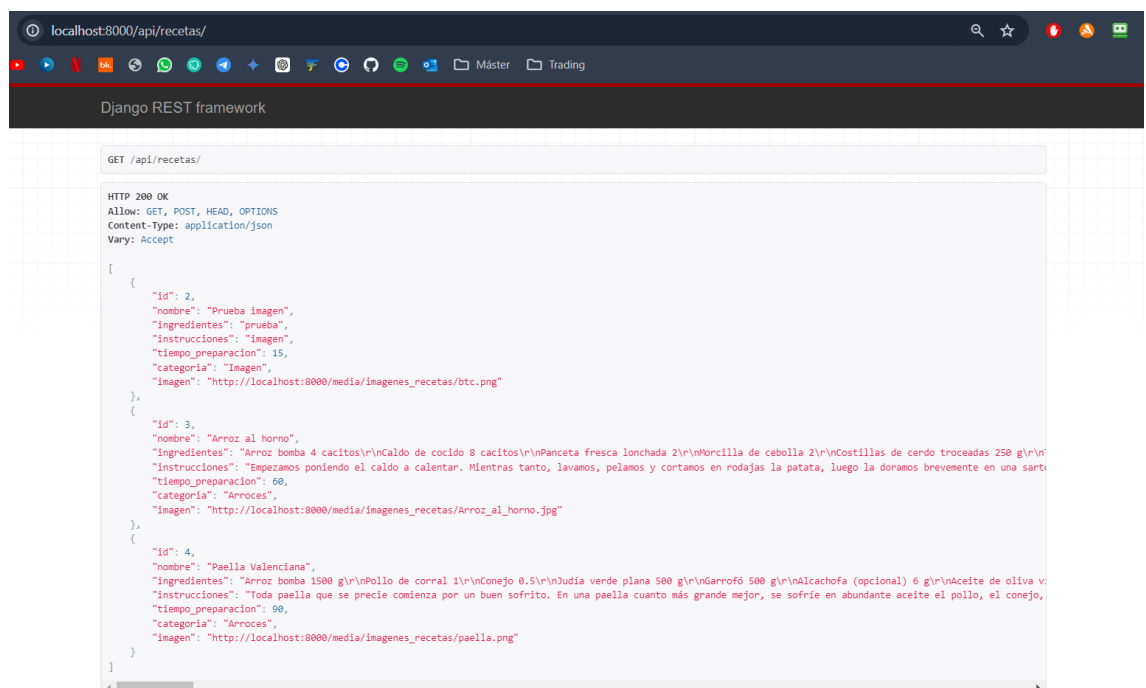
Detalle de una única receta:



Como había 2 recetas de arroz al horno, hemos borrado la primera que era de prueba

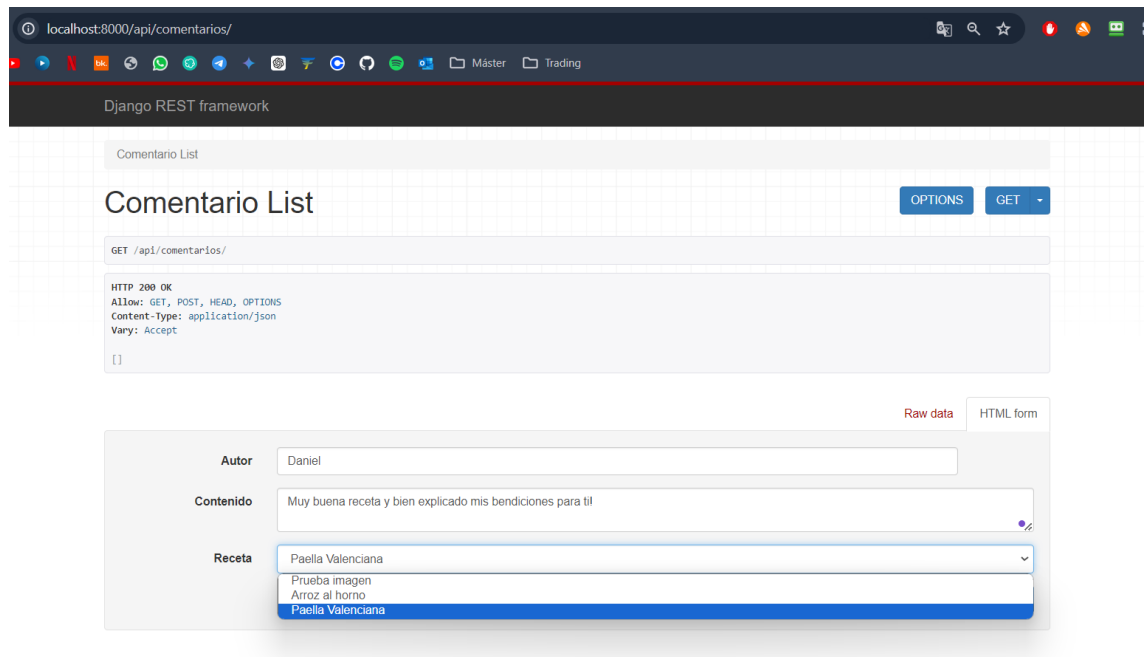


Hecho el delete, podemos observar que ya no sale cuando hacemos el método GET:

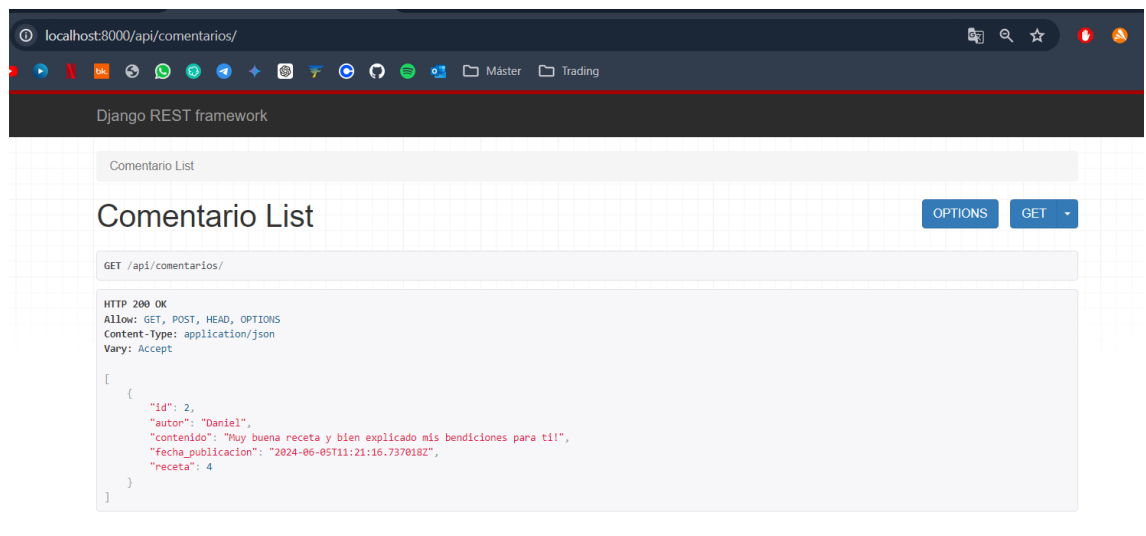


Listar y crear comentarios: <http://localhost:8000/api/comentarios/>

Creamos el comentario con el método POST, relacionado con la receta de la paella valenciana:



Y lo mostramos con el método GET:



Detallar, actualizar y eliminar un comentario:

<http://localhost:8000/api/comentarios/<id>/>

Por otro lado, detallamos los comentarios:

localhost:8000/api/comentarios/2/

Django REST framework

Comentario List / Comentario Detail

Comentario Detail

DELETE OPTIONS GET

GET /api/comentarios/2/

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

```
{
  "id": 2,
  "autor": "Daniel",
  "contenido": "Muy buena receta y bien explicado mis bendiciones para ti!",
  "fecha_publicacion": "2024-06-05T11:21:16.737018Z",
  "receta": 4
}
```

Raw data HTML form

Autor Daniel

Contenido Muy buena receta y bien explicado mis bendiciones para ti

Receta Paella Valenciana

PUT

Y hacemos una modificación del contenido del comentario con el método PUT

Django REST framework

Comentario List / Comentario Detail

Comentario Detail

DELETE OPTIONS GET

GET /api/comentarios/2/

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

```
{
  "id": 2,
  "autor": "Daniel",
  "contenido": "Muy buena receta y bien explicado mis bendiciones para ti, gracias por todo",
  "fecha_publicacion": "2024-06-05T11:21:16.737018Z",
  "receta": 4
}
```

Raw data HTML form

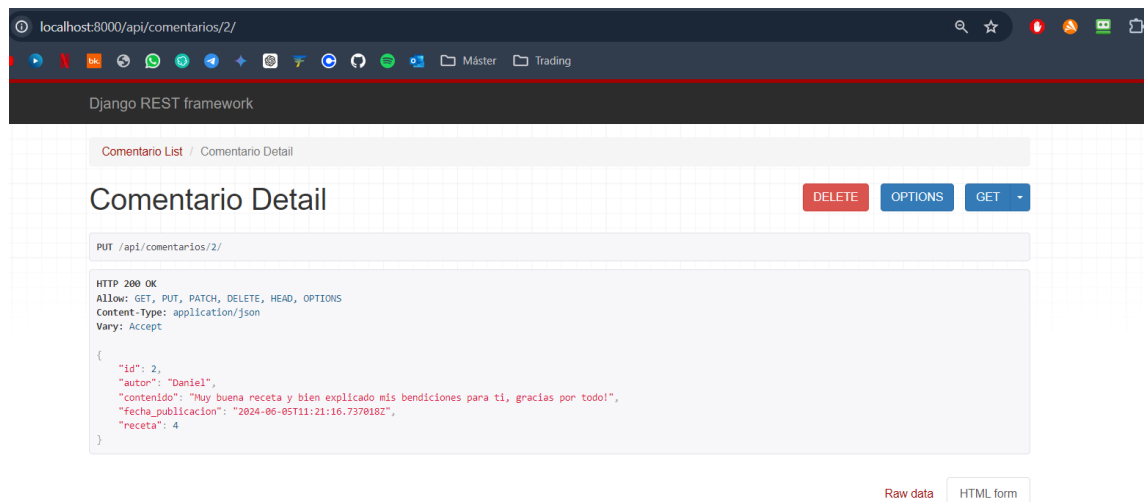
Autor Daniel

Contenido Muy buena receta y bien explicado mis bendiciones para ti, gracias por todo

Receta Paella Valenciana

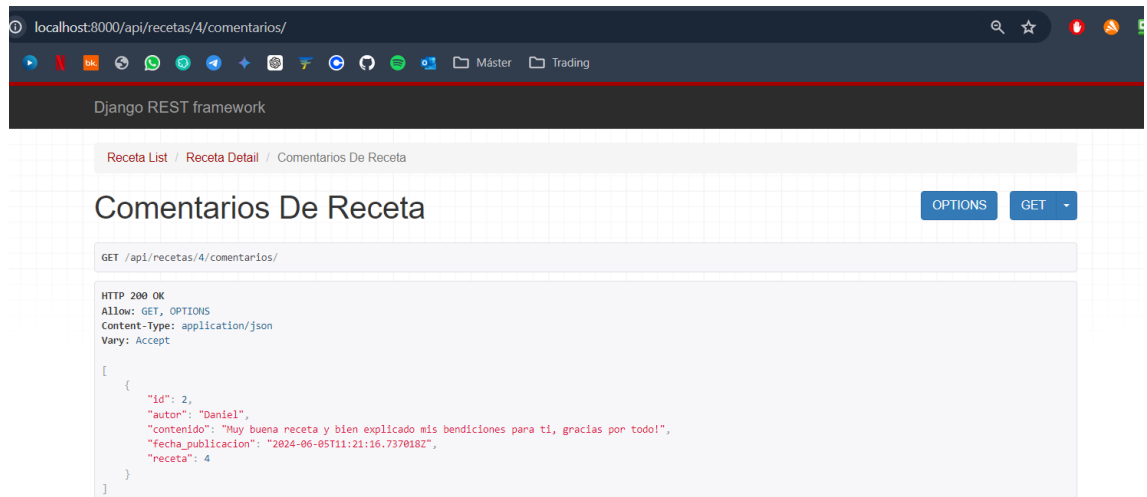
PUT

Modificado correctamente:



Listar comentarios de una receta específica:

http://localhost:8000/api/recetas/<receta_id>/comentarios/



Listar y crear listas de compras: <http://localhost:8000/api/listas-de-compras/>

Creamos una lista de la compra para la receta del arroz al horno

Django REST framework

Lista De Compras List

OPTIONS GET

POST /api/listas-de-compras/

HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "id": 3,
  "nombre": "Comprar ingredientes arroz al horno",
  "fecha_creacion": "2024-06-05T11:33:59.620894Z",
  "recetas": [
    3
  ]
}
```

Raw data HTML form

Nombre:

Recetas:

Prueba imagen
Arroz al horno
Paella Valenciana

Detallar, actualizar y eliminar una lista de compras:

<http://localhost:8000/api/listas-de-compras/<id>/>

Por ejemplo, el primer caso de lista de compra lo cree vacío a modo de test:

GET /api/listas-de-compras/

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

```
[
  {
    "id": 1,
    "nombre": "Compra 1",
    "fecha_creacion": "2024-06-04T21:58:30.931362Z",
    "recetas": []
  },
  {
    "id": 2,
    "nombre": "Compra 2",
    "fecha_creacion": "2024-06-05T11:33:06.889548Z",
    "recetas": [
      4
    ]
  },
  {
    "id": 3,
    "nombre": "Comprar ingredientes arroz al horno",
    "fecha_creacion": "2024-06-05T11:33:59.620894Z",
    "recetas": [
      3
    ]
  }
]
```

Lo eliminamos de la siguiente manera:

DELETE Send

204 No Content · 33 ms

Info Request Response

Description Headers Query Body Auth Options

Vemos que ya se ha eliminado correctamente:

Lista De Compras List

OPTIONS

GET

GET /api/listas-de-compras/

```

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

[
  {
    "id": 2,
    "nombre": "Compra 2",
    "fecha_creacion": "2024-06-05T11:33:06.889548Z",
    "recetas": [
      4
    ]
  },
  {
    "id": 3,
    "nombre": "Comprar ingredientes arroz al horno",
    "fecha_creacion": "2024-06-05T11:33:59.620894Z",
    "recetas": [
      3
    ]
  }
]

```

- **Aplicación desafíos:**

Modelos: Desafio, Participación

```

4 class Desafio(models.Model):
5     titulo = models.CharField(max_length=255)
6     descripcion = models.TextField()
7     fecha_inicio = models.DateField()
8     fecha_fin = models.DateField()
9
10    def __str__(self):
11        return self.titulo
12
13    class Participacion(models.Model):
14        receta = models.ForeignKey(Receta, on_delete=models.CASCADE)
15        desafio = models.ForeignKey(Desafio, related_name='participaciones', on_delete=models.CASCADE)
16        fecha_participacion = models.DateTimeField(auto_now_add=True)
17        votos = models.PositiveIntegerField(default=0)
18
19    def __str__(self):
20        return f'Participación de {self.receta.nombre} en {self.desafio.titulo}'

```

Serializadores: Serializadores para cada modelo.

```

4 class DesafioSerializer(serializers.ModelSerializer):
5     class Meta:
6         model = Desafio
7         fields = '__all__'
8
9 class ParticipacionSerializer(serializers.ModelSerializer):
10    class Meta:
11        model = Participacion
12        fields = '__all__'

```

Vistas Genéricas: Vistas para listar, crear, actualizar y eliminar desafíos y participaciones.

```

8  # Vistas genéricas Desafio y Participacion
9  class DesafioList(generics.ListCreateAPIView):
10     queryset = Desafio.objects.all()
11     serializer_class = DesafioSerializer
12
13  class DesafioDetail(generics.RetrieveUpdateDestroyAPIView):
14     queryset = Desafio.objects.all()
15     serializer_class = DesafioSerializer
16
17  class ParticipacionList(generics.ListCreateAPIView):
18     queryset = Participacion.objects.all()
19     serializer_class = ParticipacionSerializer
20
21  class ParticipacionDetail(generics.RetrieveUpdateDestroyAPIView):
22     queryset = Participacion.objects.all()
23     serializer_class = ParticipacionSerializer

```

Vistas Personalizadas: Vista para votar por una participación.

```

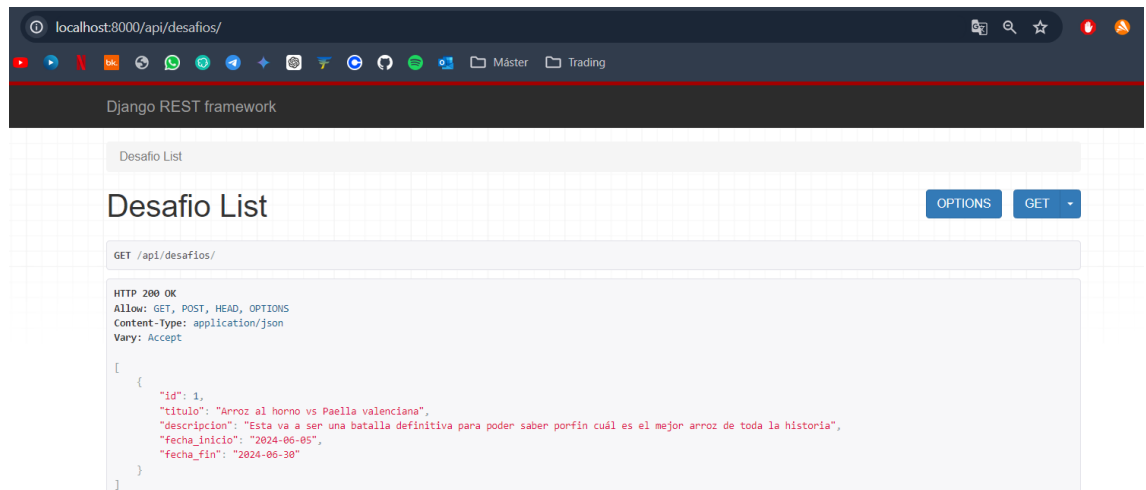
25  # Vista de api_view para votar en participaciones
26  @api_view(['POST'])
27  def votar_participacion(request, participacion_id):
28      try:
29          participacion = Participacion.objects.get(id=participacion_id)
30          participacion.votos += 1
31          participacion.save()
32          return Response({'status': 'voto registrado'}, status=status.HTTP_200_OK)
33      except Participacion.DoesNotExist:
34          return Response(status=status.HTTP_404_NOT_FOUND)

```

URLs:

Listar y crear desafíos: <http://localhost:8000/api/desafios/>

Previamente hemos creado un desafío con el método POST y lo mostramos con el método GET:



Detallar, actualizar y eliminar un desafío: <http://localhost:8000/api/desafios/<id>/>

Mostramos detalladamente el desafío por la búsqueda por id con el método GET:

localhost:8000/api/desafios/1/

Django REST framework

Desafio Detail

DELETE OPTIONS GET

GET /api/desafios/1/

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

```
{
  "id": 1,
  "titulo": "Arroz al horno vs Paella valenciana",
  "descripcion": "Esta va a ser una batalla definitiva para poder saber porfin cuál es el mejor arroz de toda la historia",
  "fecha_inicio": "2024-06-05",
  "fecha_fin": "2024-06-30"
}
```

Raw data HTML form

Titulo Arroz al horno vs Paella valenciana

Descripcion Esta va a ser una batalla definitiva para poder saber porfin cuál es el mejor arroz de toda la historia

Fecha inicio 05/06/2024

Fecha fin 30/06/2024

PUT

Listar y crear participaciones: <http://localhost:8000/api/participaciones/>

Creamos participaciones con el método POST:

localhost:8000/api/participaciones/

Django REST framework

Participacion List

OPTIONS GET

GET /api/participaciones/

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

```
[]
```

Raw data HTML form

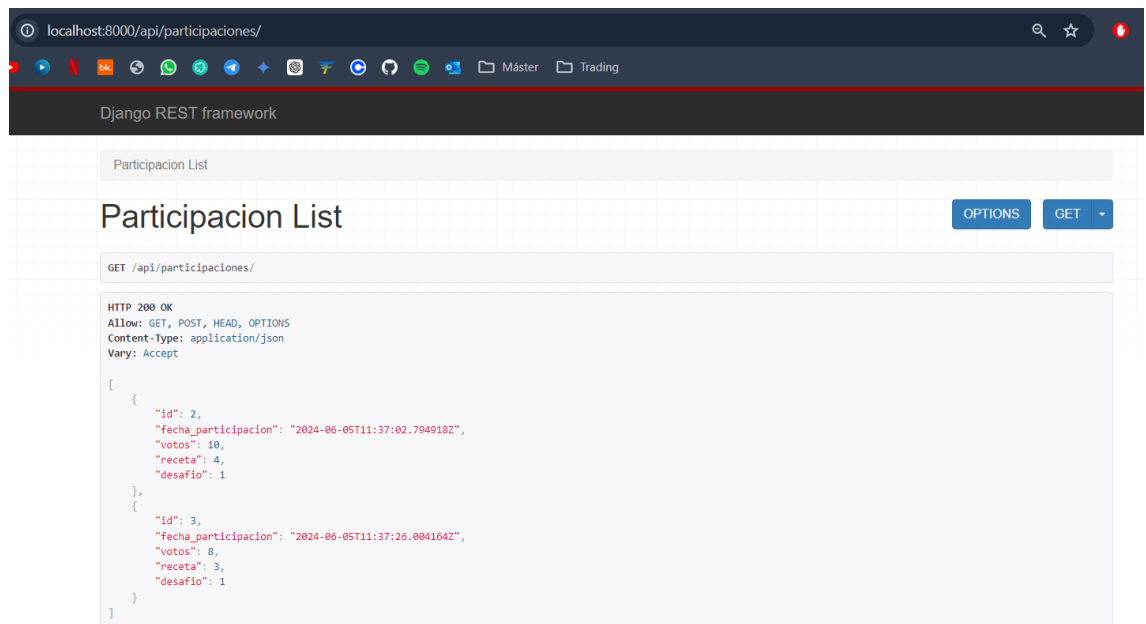
Votos 10

Receta Paella Valenciana

Desafio Arroz al horno vs Paella valenciana

POST

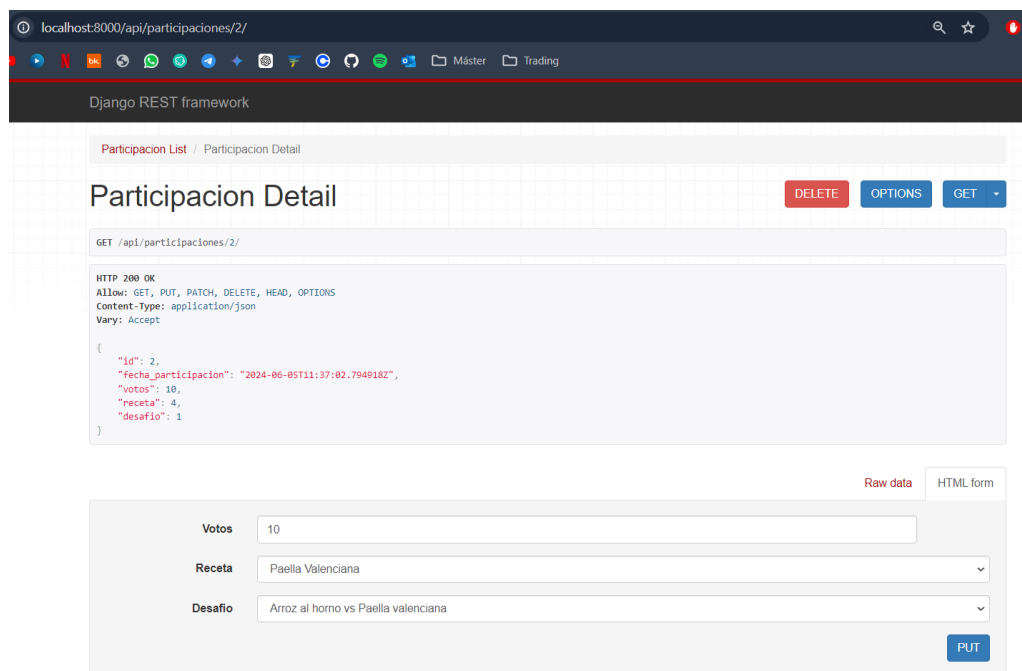
Mostramos el listado de las participaciones al desafío.



Detallar, actualizar y eliminar una participación:

<http://localhost:8000/api/participaciones/<id>/>

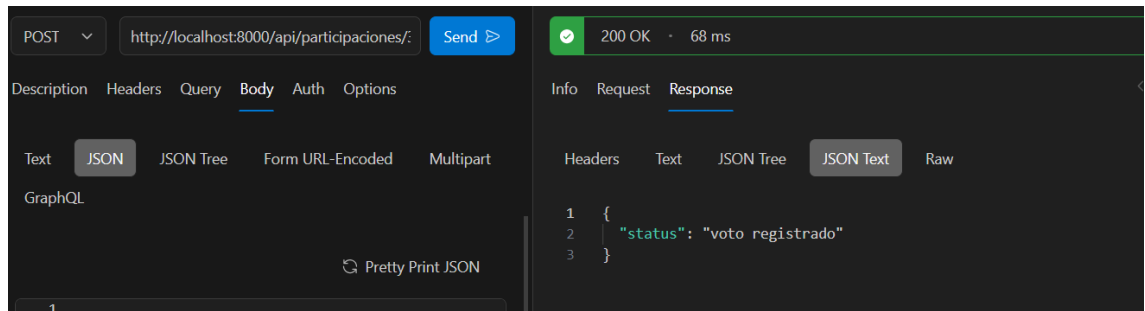
Mostramos detalladamente la participación en el desafío:



Votar por una participación:

http://localhost:8000/api/participaciones/<participacion_id>/votar/

Gracias al método POST y el api_view personalizado que hemos creado, podemos votar a una participación indicada, como es este caso:



Como vemos, la participación 3 aumentaría en 1 su voto



Conclusiones

Este proyecto de cocina ofrece una plataforma versátil para gestionar recetas, comentarios, listas de compras y desafíos culinarios. La implementación de vistas genéricas y personalizadas, junto con una API REST robusta, permite una interacción fluida y eficiente con los datos. Las funcionalidades añadidas, como la votación en desafíos y la gestión de listas de compras, enriquecen la experiencia del usuario, haciendo de esta aplicación una herramienta completa para los entusiastas de la cocina.

La estructura modular del proyecto facilita su escalabilidad y mantenimiento, permitiendo la incorporación de nuevas características de manera ordenada y eficiente. El uso de herramientas como Postman y cURL para probar y consumir la API asegura que las interacciones sean claras y manejables, tanto para desarrolladores como para usuarios finales.

Enlace del código

<https://github.com/danisentamans/Entregables-BACKEND/tree/main/Entregable3>