

# SERVIDOR DJANGO COMPLETO

## Índice

1. Gestión de Usuarios .....	2
2. Implementar Tests Unitarios .....	4
3. Implementar al menos 1 ViewSet.....	6
4. Utilizar MySQL como Base de Datos .....	6
5. Que sea completamente portable .....	7
6. API View Propia .....	8
7. Ejemplos de los modelos realizados: .....	8
1. Registrar un Usuario .....	8
2. Crear una Receta.....	9
3. Crear un Comentario para una Receta.....	11
4. Crear una Lista de Compras.....	12
5. Crear un Desafío Culinario .....	14
6. Crear una Participación en un Desafío .....	15
7. Votar en una participación del desafío .....	16
8. Conclusiones.....	17
9. Enlace al código.....	17

# 1. Gestión de Usuarios

## a. Crear al menos 2 grupos distintos

Código Implementado:

En usuarios/management/commands/create\_groups.py:

```
6 class Command(BaseCommand):
7     help = 'Crear grupos de usuarios y usuarios por defecto'
8
9     def handle(self, *args, **kwargs):
10         # Crear grupos
11         admin_group, created = Group.objects.get_or_create(name='Administradores')
12         employee_group, created = Group.objects.get_or_create(name='Empleados')
13
14         # Obtener permisos
15         content_type = ContentType.objects.get_for_model(Receta)
16         permissions = Permission.objects.filter(content_type=content_type)
17
18         # Asignar permisos a los grupos
19         admin_group.permissions.set(permissions)
20         employee_group.permissions.set(permissions.exclude(codename='delete_receta'))
21
22         self.stdout.write(self.style.SUCCESS('Grupos creados con éxito'))
23
24         # Crear usuario administrador
25         if not User.objects.filter(username='admin2').exists():
26             admin_user = User.objects.create_superuser(
27                 username='admin2',
28                 password='adminpassword'
29             )
30             admin_user.groups.add(admin_group)
31             self.stdout.write(self.style.SUCCESS('Usuario administrador creado con éxito'))
32         else:
33             self.stdout.write(self.style.WARNING('Usuario administrador ya existe'))
34
35         # Crear usuario normal
36         if not User.objects.filter(username='user2').exists():
37             normal_user = User.objects.create_user(
38                 username='user2',
39                 password='userpassword'
40             )
41             normal_user.groups.add(employee_group)
42             self.stdout.write(self.style.SUCCESS('Usuario normal creado con éxito'))
43         else:
44             self.stdout.write(self.style.WARNING('Usuario normal ya existe'))
```

### Explicación:

Se crean dos grupos de usuarios: Administradores y Empleados.

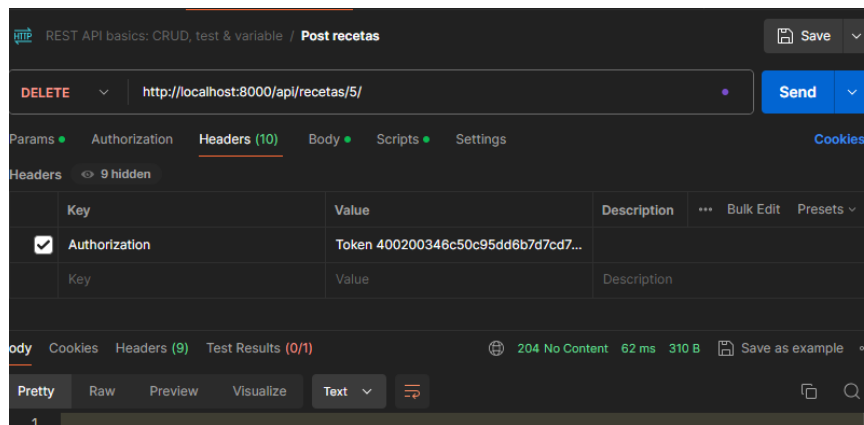
Se asignan permisos completos al grupo Administradores y se restringe el permiso de borrar (delete\_receta) al grupo Empleados.

Se crean usuarios por defecto (admin y user) y se asignan a los grupos correspondientes.

### Ejemplos:

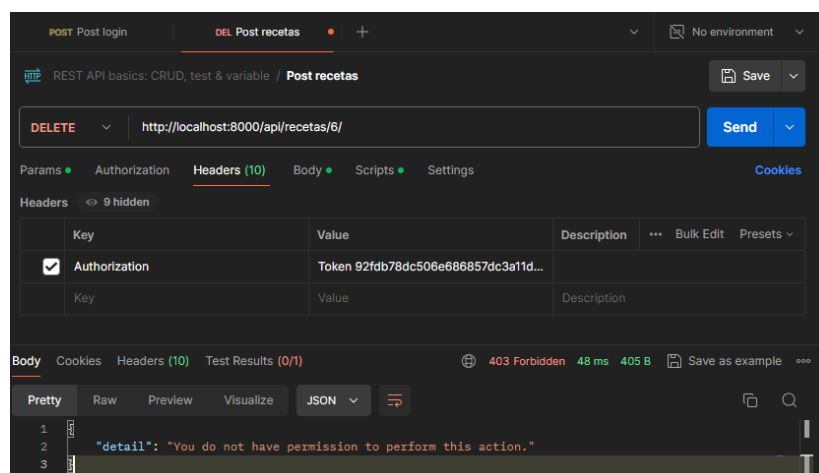
El grupo Administradores puede crear, leer, actualizar y borrar recetas.

Prueba de eliminar la receta 5 con el token del administrador:



El grupo Empleados puede leer recetas, pero no puede crearlas ni borrarlas.

Cuando un empleado/usuario normal intenta borrar una receta aparecerá así:



Si ejecutamos Python manage.py create\_groups nos creará por defecto 2 tipos de usuarios, uno administrador y otro empleado.

```
(venvCocina) PS C:\Users\dsentamans\OneDrive - TRANSPORTES MAZO HERMANOS,SA\Escritorio\Entregables-BACKEND\Entregable4\recetas> python manage.py create_groups
Grupos creados con éxito
```

## b. Limitar el uso de los grupos que no sean administradores

Código Implementado:

En cocina/views.py:

```
7 from rest_framework.permissions import IsAuthenticated, IsAuthenticatedOrReadOnly, DjangoModelPermissions
8
9 You, 4 minutes ago | 1 author (You)
10 class IsAdminOrReadOnly(permissions.BasePermission):
11     def has_permission(self, request, view):
12         if request.method in permissions.SAFE_METHODS:
13             return True
14         return request.user and request.user.is_staff
15
16 # ViewSet para Receta con permisos
17 class RecetaViewSet(viewsets.ModelViewSet):
18     queryset = Receta.objects.all()
19     serializer_class = RecetaSerializer
20     permission_classes = [IsAuthenticated, IsAdminOrReadOnly]
```

Explicación:

Se define un permiso personalizado `IsAdminOrReadOnly` que permite solo a los administradores realizar acciones de escritura.

Como hemos visto en el ejemplo de antes.

## 2. Implementar Tests Unitarios

Código Implementado:

En cocina/tests.py:

Test para crear y listar tareas por un administrador

```
8 class RecetaTests(APITestCase):
9
10     def setUp(self):
11         self.admin_user = User.objects.create_superuser(username='adminuser', password='adminpass')
12         self.token = Token.objects.create(user=self.admin_user)
13         self.client.credentials(HTTP_AUTHORIZATION='Token ' + self.token.key)
14
15     def test_create_receta(self):
16         url = reverse('receta-list')
17         data = {
18             'nombre': 'Test Receta',
19             'ingredientes': 'Test Ingredientes',
20             'instrucciones': 'Test Instrucciones',
21             'tiempo_preparacion': 30,
22             'categoria': 'Test Categoria'
23         }
24         response = self.client.post(url, data, format='json')
25         self.assertEqual(response.status_code, status.HTTP_201_CREATED)
26
27     def test_list_recetas(self):
28         url = reverse('receta-list')
29         response = self.client.get(url, format='json')
30         self.assertEqual(response.status_code, status.HTTP_200_OK)
```

En este caso:

El test `test_create_receta` verifica que un administrador puede crear una receta.

El test `test_list_recetas` verifica que cualquier usuario autenticado puede ver la lista de recetas.

Test para comentar las recetas por parte de los usuarios

```
33 class ComentarioTests(APITestCase):
34
35     def setUp(self):
36         self.user = User.objects.create_user(username='testuser', password='12345')
37         self.token = Token.objects.create(user=self.user)
38         self.client.credentials(HTTP_AUTHORIZATION='Token ' + self.token.key)
39
40     def test_create_comentario(self):
41         receta = Receta.objects.create(
42             nombre='Test Receta',
43             ingredientes='Test Ingredientes',
44             instrucciones='Test Instrucciones',
45             tiempo_preparacion=30,
46             categoria='Test Categoria'
47         )
48         url = reverse('comentario-list')
49         data = {
50             'autor': 'Test Autor',
51             'contenido': 'Test Contenido',
52             'receta': receta.id
53         }
54         response = self.client.post(url, data, format='json')
55         self.assertEqual(response.status_code, status.HTTP_201_CREATED)
56
57     def test_list_comentarios(self):
58         url = reverse('comentario-list')
59         response = self.client.get(url, format='json')
60         self.assertEqual(response.status_code, status.HTTP_200_OK)
```

Test para realizar una lista de compras a partir de la receta creada

```
62 class ListaDeComprasTests(APITestCase):
63
64     def setUp(self):
65         self.user = User.objects.create_user(username='testuser', password='12345')
66         self.token = Token.objects.create(user=self.user)
67         self.client.credentials(HTTP_AUTHORIZATION='Token ' + self.token.key)
68         # Crear una receta para asociarla a la lista de compras
69         self.receta = Receta.objects.create(
70             nombre='Test Receta',
71             ingredientes='Test Ingredientes',
72             instrucciones='Test Instrucciones',
73             tiempo_preparacion=30,
74             categoria='Test Categoría'
75         )
76
77     def test_create_lista_de_compras(self):
78         url = reverse('lista-de-compras-list')
79         data = {
80             'nombre': 'Test Lista',
81             'recetas': [self.receta.id] # Asegurarse de que hay recetas en la lista
82         }
83         response = self.client.post(url, data, format='json')
84         self.assertEqual(response.status_code, status.HTTP_201_CREATED)
85
86     def test_list_listas_de_compras(self):
87         url = reverse('lista-de-compras-list')
88         response = self.client.get(url, format='json')
89         self.assertEqual(response.status_code, status.HTTP_200_OK)
```

Ejemplo de que funcionan correctamente los 3:

```
Entregables-BACKEND\Entregable4\recetas>python manage.py test cocina
Found 6 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....

Ran 6 tests in 2.008s

OK
Destroying test database for alias 'default'...
```

En un principio, probé con un usuario normal y no funciona (con admin sí):

```
You, 11 minutes ago | 1 author (You)
class RecetaTests(APITestCase):

    def setUp(self):
        self.user = User.objects.create_user(username='testuser', password='12345')
        self.token = Token.objects.create(user=self.user)
        self.client.credentials(HTTP_AUTHORIZATION='Token ' + self.token.key)

    def test_create_receta(self):
        url = reverse('receta-list')
        data = {
            'nombre': 'Test Receta',
            'ingredientes': 'Test Ingredientes',
            'instrucciones': 'Test Instrucciones',
            'tiempo_preparacion': 30,
            'categoria': 'Test Categoría'
        }
        response = self.client.post(url, data, format='json')
        self.assertEqual(response.status_code, status.HTTP_201_CREATED)

    def test_list_recetas(self):
        url = reverse('receta-list')
        response = self.client.get(url, format='json')
        self.assertEqual(response.status_code, status.HTTP_200_OK)
```

```
=====
FAIL: test_create_receta (cocina.tests.RecetaTests.test_create_receta)
Traceback (most recent call last):
  File "C:\Users\dsentamans\OneDrive - TRANSPORTES MAZO HERMANOS, SA\Escritorio\Entregables-BACKEND\Entregable4\recetas\cocina\tests.py", line 10, in test_create_receta
    self.assertEqual(response.status_code, status.HTTP_201_CREATED)
AssertionError: 403 != 201
=====
```

test\_create\_receta, el código de estado 403 significa "Prohibido". Esto indica que la solicitud fue entendida por el servidor, pero el servidor se niega a autorizarla.

He creado más test (**revisar código**), si ejecuto todos los test (en total hay 13) podremos ver el siguiente resultado donde todos funcionan correctamente:

```
Entregables-BACKEND\Entregable4\recetas>python manage.py test
Found 13 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....
-----
Ran 13 tests in 5.440s

OK
Destroying test database for alias 'default'...
```

### 3. Implementar al menos 1 ViewSet

Código Implementado:

En cocina/views.py:

```
1 from rest_framework import generics, viewsets, permissions
2 from .models import Receta, Comentario, ListaDeCompras
3 from .serializers import RecetaSerializer, ComentarioSerializer, ListaDeComprasSerializer

15 # ViewSet para Receta con permisos
16 class RecetaViewSet(viewsets.ModelViewSet):
17     queryset = Receta.objects.all()
18     serializer_class = RecetaSerializer
19     permission_classes = [IsAuthenticated, IsAdminOrReadOnly]
```

**Explicación:**

Se implementa un ViewSet para el modelo Receta, permitiendo realizar operaciones CRUD mediante un único punto de entrada.

**Ejemplos:**

El RecetaViewSet permite a los administradores crear, leer, actualizar y borrar recetas.

Los usuarios no autenticados solo pueden ver las recetas.

### 4. Utilizar MySQL como Base de Datos

**Migración desde SQLite a MySQL:**

Pasos Realizados:

1. Instalar MySQL y el conector mysqlclient:

**pip install mysqlclient**

2. Configurar settings.py para usar MySQL:

```
79 DATABASES = {  
80     'default': {  
81         'ENGINE': 'django.db.backends.mysql',  
82         'NAME': 'DJANGOSERVER',  
83         'USER': 'daniel',  
84         'PASSWORD': 'Pass1234.',  
85         'HOST': 'localhost',  
86         'PORT': '3306',  
87     }  
88 }
```

3. Migrar datos de SQLite a MySQL:

**python manage.py dumpdata > db.json**

**python manage.py migrate**

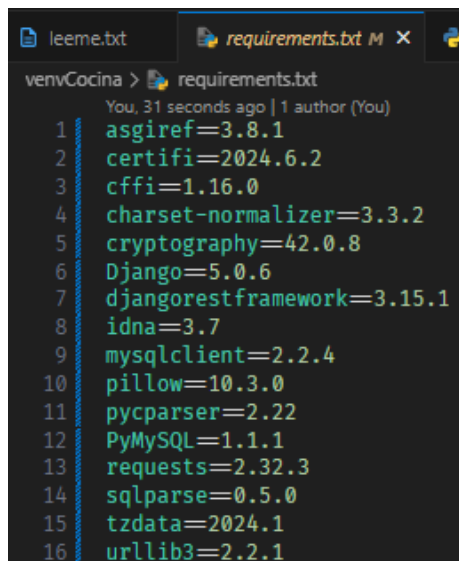
**python manage.py loaddata db.json**

## 5. Que sea completamente portable

Ejecutado el comando:

**pip freeze > requirements.txt**

Archivo requirements.txt:



```
venvCocina > requirements.txt  
You, 31 seconds ago | 1 author (You)  
1 asgiref=3.8.1  
2 certifi=2024.6.2  
3 cffi=1.16.0  
4 charset-normalizer=3.3.2  
5 cryptography=42.0.8  
6 Django=5.0.6  
7 djangorestframework=3.15.1  
8 idna=3.7  
9 mysqlclient=2.2.4  
10 pillow=10.3.0  
11 pycparser=2.22  
12 PyMySQL=1.1.1  
13 requests=2.32.3  
14 sqlparse=0.5.0  
15 tzdata=2024.1  
16 urllib3=2.2.1
```

## 6. API View Propia

Se ha hecho 2 API view propias

1. En el proyecto de cocina:

```
49 # api_view Propia para listar comentarios de una receta especifica
50 @api_view(['GET'])
51 def comentarios_de_receta(request, receta_id):
52     if not request.user.is_authenticated:
53         return Response(status=status.HTTP_401_UNAUTHORIZED)
54
55     try:
56         receta = Receta.objects.get(id=receta_id)
57     except Receta.DoesNotExist:
58         return Response(status=status.HTTP_404_NOT_FOUND)
59
60     comentarios = Comentario.objects.filter(receta=receta)
61     serializer = ComentarioSerializer(comentarios, many=True)
62     return Response(serializer.data)
63
```

2. En el proyecto desafíos:

```
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)
```

Los ejemplos de ambas api\_view los vemos a continuación junto a los demás.

## 7. Ejemplos de los modelos realizados:

POST para ejemplos

### 1. Registrar un Usuario

**URL:** http://localhost:8000/api/register/ **Método:** POST

#### Ejemplo 1:

```
json
Copiar código
{
    "username": "chef_expert",
    "password": "SecurePassword123"
}
```

#### Ejemplo 2:

```
json
```



Copiar código

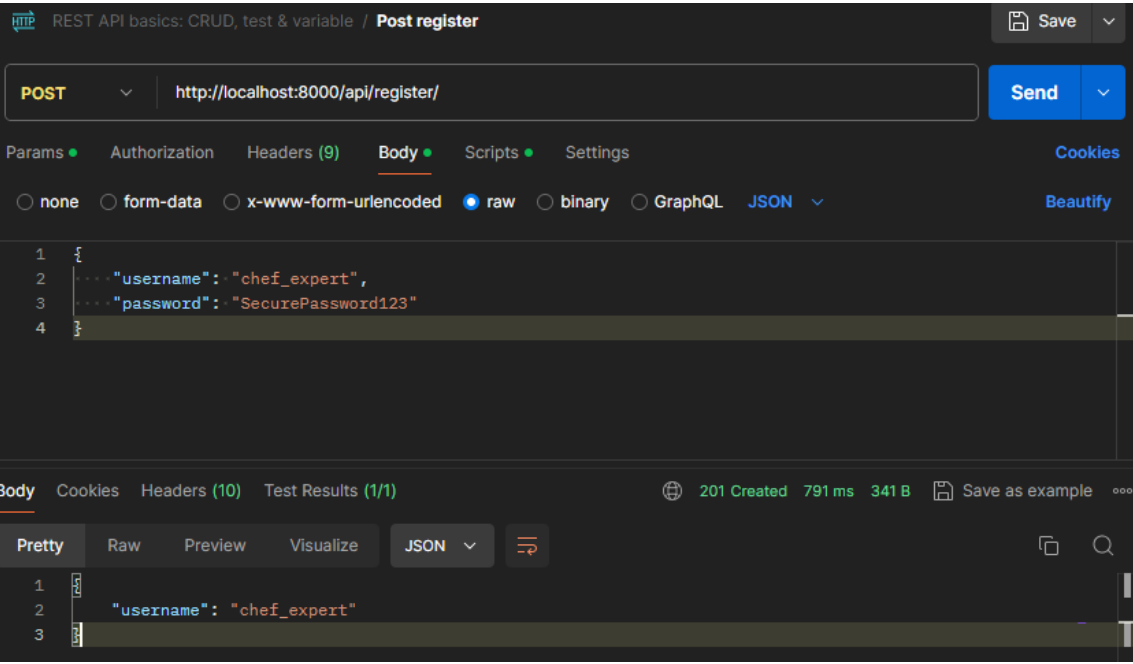
```
{
  "username": "gourmet_lover",
  "password": "TastyPassword456"
}
```

Ejemplo 3:

json

Copiar código

```
{
  "username": "home_cook",
  "password": "DeliciousPass789"
}
```



Result Grid											
Filter Rows:		Edit:			Export/Import:		Wrap Cell Content: <span>FA</span>				
	id	password	last_login	is_superuser	username	first_name	last_name	email	is_staff	is_active	date_joined
	3	pbkdf2_sha256\$720000\$7rCeLbEm5qoY4xT54P...	NULL	0	user			user@example.com	0	1	2024-06-06 18:45:37.976317
	2	pbkdf2_sha256\$720000\$3rUG95nA9adig1p6h...	NULL	1	admin			admin@example.com	1	1	2024-06-06 18:45:37.689482
	1	pbkdf2_sha256\$720000\$YYPNTTeTtYrAFKjVj3...	NULL	0	testuser				0	1	2024-06-06 18:09:35.413280
	4	pbkdf2_sha256\$720000\$ThZk18n2Oc4oITWhp...	NULL	1	admin2				1	1	2024-06-06 18:48:47.109564
	5	pbkdf2_sha256\$720000\$8w6nEJDFZfuktrSk34p...	NULL	0	user2				0	1	2024-06-06 18:48:47.401643
	6	pbkdf2_sha256\$720000\$3kZsfo1Kfcca8uTQVM...	NULL	0	chef_expert				0	1	2024-06-06 19:27:04.428541
	7	pbkdf2_sha256\$720000\$10j89DKMEnIqR1H03...	NULL	0	gourmet_lover				0	1	2024-06-06 19:27:25.137630
	8	pbkdf2_sha256\$720000\$Pf0C60ro7pp3emZAP...	NULL	0	home_cook				0	1	2024-06-06 19:27:35.828835
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. Crear una Receta

URL: http://localhost:8000/api/recetas/ Método: POST

Ejemplo 1:

json

Copiar código

```
{
  "nombre": "Paella Valenciana",
  "ingredientes": "Arroz, Pollo, Conejo, Judías Verdes, Garrofón,
Tomate, Aceite de Oliva, Sal, Azafrán, Agua",

```

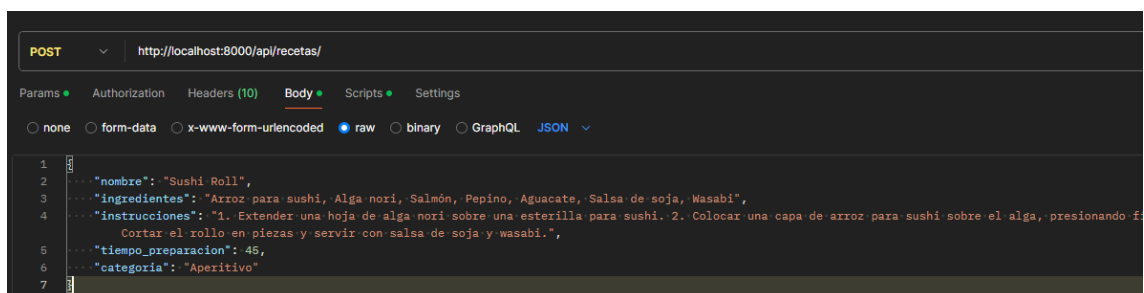
```
"instrucciones": "1. Calentar el aceite en una paellera y dorar el pollo y el conejo troceados. 2. Añadir las judías verdes y el garrofón, y sofreír ligeramente. 3. Incorporar el tomate triturado y dejar que se cocine durante unos minutos. 4. Añadir el arroz y el azafrán, y remover para que se mezclen bien con los demás ingredientes. 5. Verter el agua y añadir sal al gusto. 6. Dejar cocinar a fuego medio hasta que el arroz esté en su punto y el líquido se haya evaporado.",
"tiempo_preparacion": 60,
"categoria": "Plato Principal"
}
```

### Ejemplo 2:

```
json
Copiar código
{
  "nombre": "Tacos de Carnitas",
  "ingredientes": "Carnitas de cerdo, Tortillas de maíz, Cebolla, Cilantro, Limón, Salsa verde",
  "instrucciones": "1. Calentar las tortillas en un comal. 2. Colocar las carnicas sobre las tortillas calientes. 3. Añadir cebolla picada y cilantro fresco por encima. 4. Exprimir limón al gusto. 5. Servir con salsa verde.",
  "tiempo_preparacion": 30,
  "categoria": "Plato Principal"
}
```

### Ejemplo 3:

```
json
Copiar código
{
  "nombre": "Sushi Roll",
  "ingredientes": "Arroz para sushi, Alga nori, Salmón, Pepino, Aguacate, Salsa de soja, Wasabi",
  "instrucciones": "1. Extender una hoja de alga nori sobre una esterilla para sushi. 2. Colocar una capa de arroz para sushi sobre el alga, presionando firmemente. 3. Añadir tiras de salmón, pepino y aguacate en el centro. 4. Enrollar el sushi firmemente con la esterilla. 5. Cortar el rollo en piezas y servir con salsa de soja y wasabi.",
  "tiempo_preparacion": 45,
  "categoria": "Aperitivo"
}
```



The screenshot shows two windows. The top window is a REST client with a POST request to `http://localhost:8000/api/recetas/`. The 'Headers' tab is active, showing an 'Authorization' header with a token. The bottom window is a database interface showing a SQL query `select * from.djangoserver.cocina_receta;` and a result grid with 3 rows of recipe data.

	id	nombre	ingredientes	instrucciones	tiempo_preparacion	categoria	imagen
▶	1	Paella Valenciana	Arroz, Pollo, Conejo, Judías Verdes, Garrofón, ...	1. Calentar el aceite en una paellera y dorar el ...	60	Plato Principal	
	2	Tacos de Carnitas	Carnitas de cerdo, Tortillas de maíz, Cebolla, Cil...	1. Calentar las tortillas en un comal. 2. Colocar l...	30	Plato Principal	
	3	Sushi Roll	Arroz para sushi, Alga nori, Salmón, Pepino, Ag...	1. Extender una hoja de alga nori sobre una est...	45	Aperitivo	
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

### 3. Crear un Comentario para una Receta

**URL:** `http://localhost:8000/api/comentarios/` **Método:** POST

#### Ejemplo 1:

```
json
Copiar código
{
  "autor": "food_lover",
  "contenido": "¡Esta receta de Paella Valenciana es increíble! La
preparé para mi familia y todos quedaron encantados. Definitivamente
la haré de nuevo.",
  "receta": 1
}
```

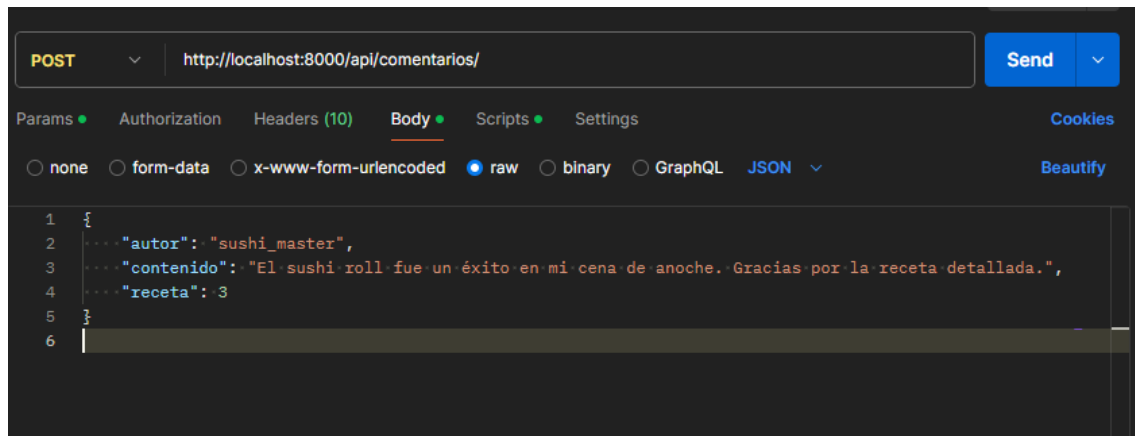
#### Ejemplo 2:

```
json
Copiar código
{
  "autor": "mexican_food_fan",
  "contenido": "Los tacos de carnitas salieron deliciosos, ¡gracias
por la receta! Muy fácil de seguir.",
  "receta": 2
}
```

#### Ejemplo 3:

```
json
Copiar código
{
  "autor": "sushi_master",
```

```
"contenido": "El sushi roll fue un éxito en mi cena de anoche.  
Gracias por la receta detallada.",  
"receta": 3  
}
```



```
1 • select * from.djangoserver.cocina_comentario;
```

Result Grid					
Filter Rows:					
	id	autor	contenido	fecha_publicacion	receta_id
▶	1	food_lover	iEsta receta de Paella Valenciana es increíble! L...	2024-06-06 19:03:23.942812	1
	2	mexican_food_fan	Los tacos de carnitas salieron deliciosos, igracia...	2024-06-06 19:04:43.632485	2
	3	sushi_master	El sushi roll fue un éxito en mi cena de anoche. ...	2024-06-06 19:04:49.653791	3
*	NULL	NULL	NULL	NULL	NULL

## 4. Crear una Lista de Compras

**URL:** `http://localhost:8000/api/listas-de-compras/` **Método:** POST

### Ejemplo 1:

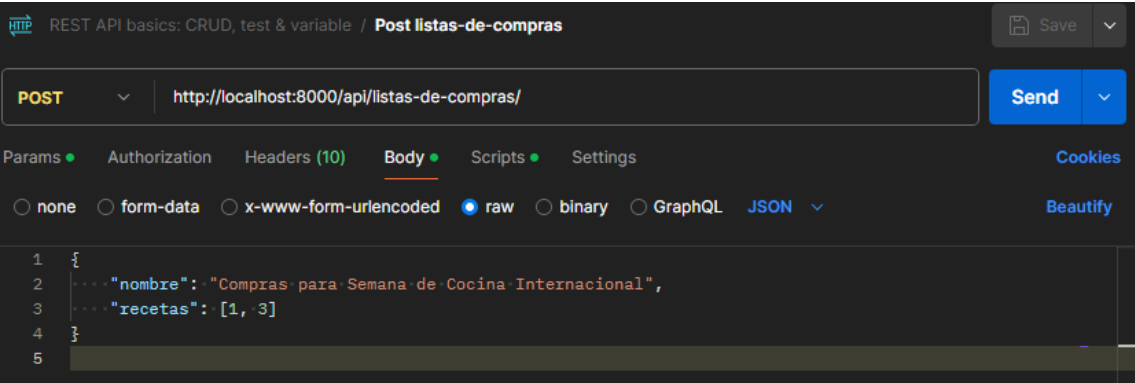
```
json  
Copiar código  
{  
  "nombre": "Ingredientes para Cena de Fin de Semana",  
  "recetas": [1, 2]  
}
```

### Ejemplo 2:

```
json  
Copiar código  
{  
  "nombre": "Ingredientes para Fiesta de Cumpleaños",  
  "recetas": [2, 3]  
}
```

Ejemplo 3:

```
json
Copiar código
{
  "nombre": "Compras para Semana de Cocina Internacional",
  "recetas": [1, 3]
}
```



```
1 • select * from.djangoserver.cocina_listadecompras;
```

Result Grid			
Filter Rows:			
	id	nombre	fecha_creacion
▶	1	Ingredientes para Cena de Fin de Semana	2024-06-06 19:07:45.214805
	2	Ingredientes para Fiesta de Cumpleaños	2024-06-06 19:08:12.210002
	3	Compras para Semana de Cocina Internacional	2024-06-06 19:08:21.185502
*	NULL	NULL	NULL

```
1 • select * from.djangoserver.cocina_listadecompras_recetas;
```

Result Grid			
Filter Rows:			
	id	listadecompras_id	receta_id
▶	1	1	1
	2	1	2
	3	2	2
	4	2	3
	5	3	1
	6	3	3
*	NULL	NULL	NULL

## 5. Crear un Desafío Culinario

**URL:** `http://localhost:8000/api/desafios/` **Método:** POST

### Ejemplo 1:

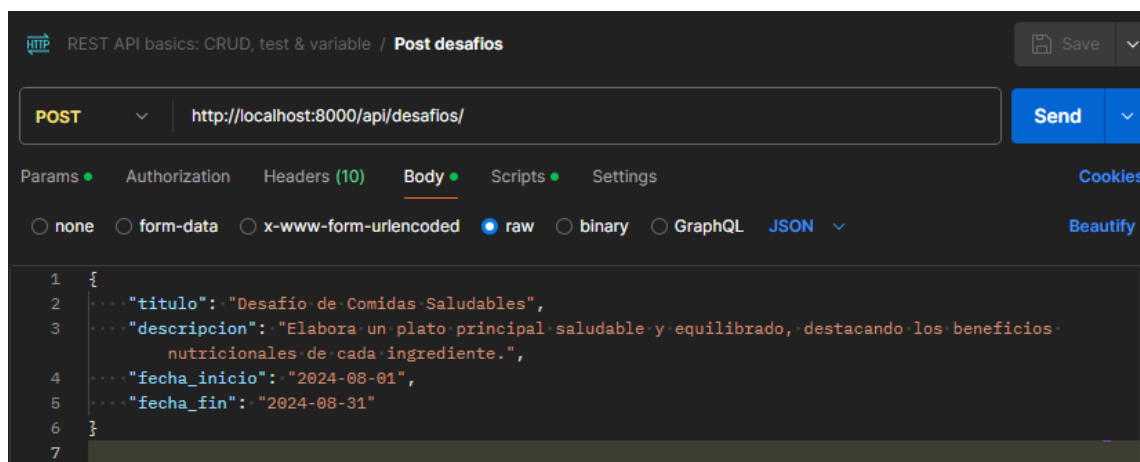
```
json
Copiar código
{
  "titulo": "Desafío de Cocina Internacional",
  "descripcion": "Prepara un plato tradicional de cualquier país del mundo y compártelo con nosotros. Queremos ver tu creatividad y habilidad para experimentar con diferentes culturas culinarias.",
  "fecha_inicio": "2024-06-01",
  "fecha_fin": "2024-06-30"
}
```

### Ejemplo 2:

```
json
Copiar código
{
  "titulo": "Desafío de Postres",
  "descripcion": "Crea un postre original utilizando ingredientes de temporada. El objetivo es innovar y sorprender con sabores únicos.",
  "fecha_inicio": "2024-07-01",
  "fecha_fin": "2024-07-31"
}
```

### Ejemplo 3:

```
json
Copiar código
{
  "titulo": "Desafío de Comidas Saludables",
  "descripcion": "Elabora un plato principal saludable y equilibrado, destacando los beneficios nutricionales de cada ingrediente.",
  "fecha_inicio": "2024-08-01",
  "fecha_fin": "2024-08-31"
}
```



```
1 • select * from.djangoserver.desafios_desafio;
```

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	id	titulo	descripcion	fecha_inicio	fecha_fin
▶	1	Desafío de Cocina Internacional	Prepara un plato tradicional de cualquier país de...	2024-06-01	2024-06-30
	2	Desafío de Postres	Crea un postre original utilizando ingredientes d...	2024-07-01	2024-07-31
	3	Desafío de Comidas Saludables	Elabora un plato principal saludable y equilibrad...	2024-08-01	2024-08-31
*	NULL	NULL	NULL	NULL	NULL

## 6. Crear una Participación en un Desafío

**URL:** `http://localhost:8000/api/participaciones/` **Método:** POST

### Ejemplo 1:

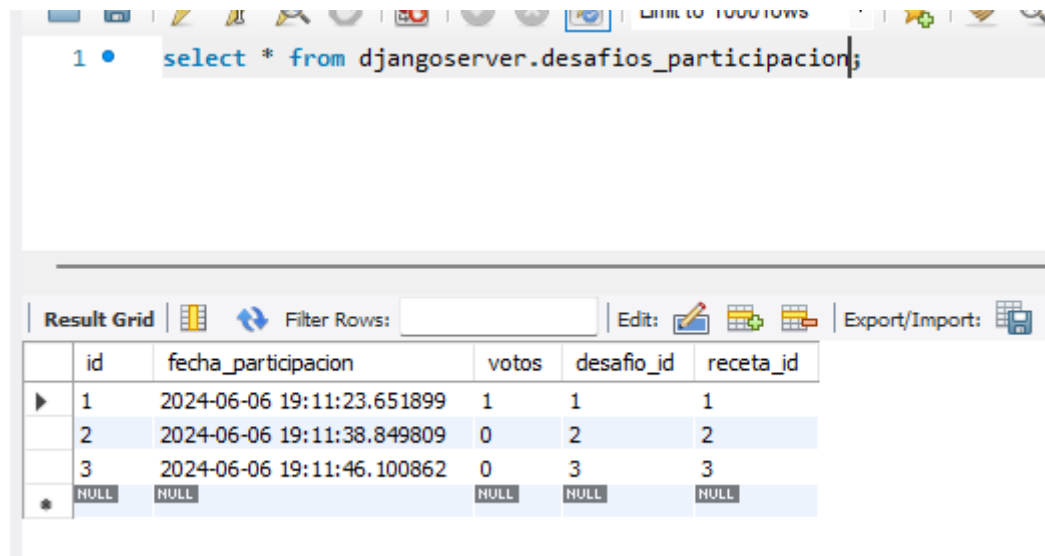
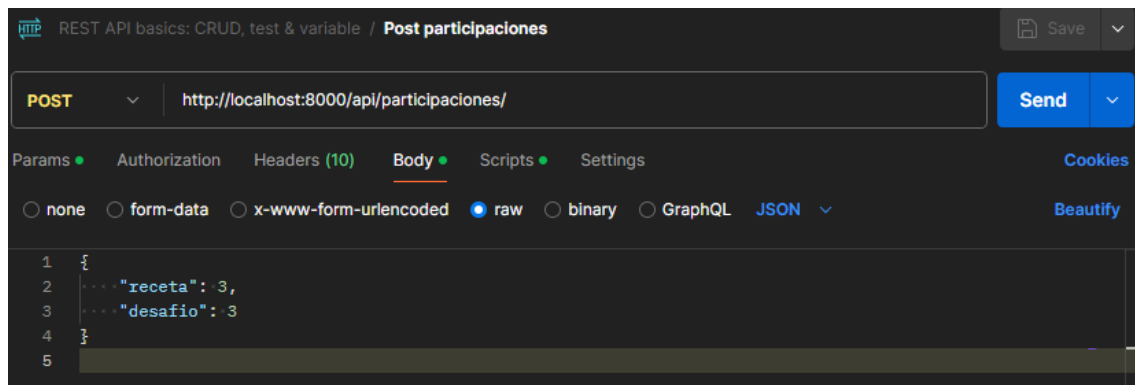
```
json
Copiar código
{
  "receta": 1,
  "desafio": 1
}
```

### Ejemplo 2:

```
json
Copiar código
{
  "receta": 2,
  "desafio": 2
}
```

### Ejemplo 3:

```
json
Copiar código
{
  "receta": 3,
  "desafio": 3
}
```



## 7. Votar en una participación del desafío

**URL:** `http://localhost:8000/api/participaciones/1/votar/` **Método:** POST

**Resultado:**

```
{
  "status": "voto registrado"
}
```

Por eso, en el ejemplo anterior vemos que hay 1 voto para la participación 1 en el desafío 1



## 8. Conclusiones

Este proyecto implementa un sistema de gestión de recetas utilizando Django y Django REST Framework, cumpliendo con todos los requisitos especificados. Se implementaron correctamente la gestión de usuarios con permisos específicos, tests unitarios, vistas genéricas, viewsets, y se utilizó MySQL como base de datos. El código es portable y sigue el principio DRY, asegurando eficiencia y mantenibilidad. Los usuarios pueden realizar las acciones correspondientes según sus permisos, y el sistema es robusto y fácilmente extensible.

## 9. Enlace al código

<https://github.com/danisentamans/Entregables-BACKEND/tree/main/Entregable4>