# 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 desafios, 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

Texto

Descripción generada automáticamente

Serializadores: Serializadores para cada modelo.

Texto

Descripción generada automáticamente

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

Texto

Descripción generada automáticamente

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

Texto

Descripción generada automáticamente

**URLs:**

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

Crear con el método POST: (2 ejemplos)

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Listar las recetas con el método GET:

Texto

Descripción generada automáticamente

Así como mostrar las imágenes adjuntas:

Captura de pantalla de computadora

Descripción generada automáticamentePantalla de computadora con una pizza

Descripción generada automáticamente con confianza media

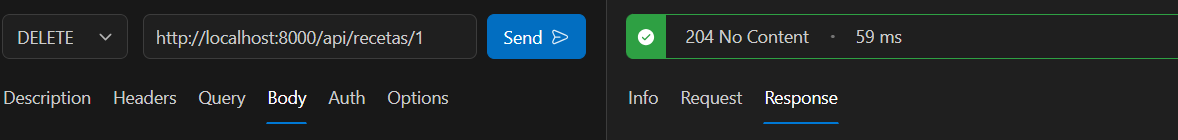
Detallar, actualizar y eliminar una receta: [http://localhost:8000/api/recetas/<id>/](http://localhost:8000/api/recetas/%3cid%3e/)

Detalle de una única receta:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

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:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

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

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

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Y lo mostramos con el método GET:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Detallar, actualizar y eliminar un comentario: [http://localhost:8000/api/comentarios/<id>/](http://localhost:8000/api/comentarios/%3cid%3e/)

Por otro lado, detallamos los comentarios:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

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

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Modificado correctamente:

Interfaz de usuario gráfica, Texto, Aplicación, Sitio web

Descripción generada automáticamente

Listar comentarios de una receta específica: [http://localhost:8000/api/recetas/<receta\_id>/comentarios/](http://localhost:8000/api/recetas/%3creceta_id%3e/comentarios/)

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Sitio web

Descripción generada automáticamente

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

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Detallar, actualizar y eliminar una lista de compras: [http://localhost:8000/api/listas-de-compras/<id>/](http://localhost:8000/api/listas-de-compras/%3cid%3e/)

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

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Lo eliminamos de la siguiente manera:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Vemos que ya se ha eliminado correctamente:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* **Aplicación desafios:**

Modelos: Desafio, Participación

Texto

Descripción generada automáticamente

Serializadores: Serializadores para cada modelo.

Texto

Descripción generada automáticamente

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

Texto

Descripción generada automáticamente

Vistas Personalizadas: Vista para votar por una participación.

Texto

Descripción generada automáticamente

**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:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Detallar, actualizar y eliminar un desafío: [http://localhost:8000/api/desafios/<id>/](http://localhost:8000/api/desafios/%3cid%3e/)

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

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

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

Creamos participaciones con el método POST:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Mostramos el listado de las participaciones al desafío.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Detallar, actualizar y eliminar una participación: [http://localhost:8000/api/participaciones/<id>/](http://localhost:8000/api/participaciones/%3cid%3e/)

Mostramos detalladamente la participación en el desafío:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Votar por una participación: [http://localhost:8000/api/participaciones/<participacion\_id>/votar/](http://localhost:8000/api/participaciones/%3cparticipacion_id%3e/votar/)

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

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Como vemos, la participación 3 aumentaría en 1 su voto

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

# 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>