


Open Opened 2 days ago by  **Rubén Montero**

Permitidme criticar eso...

Resumen

- Modificaremos el `endpoint /entries/<int:path_param_id>/comments` para que permita **añadir** datos a la base de datos si recibe un **POST**

Descripción

En nuestro primer `endpoint` (`GET /entries`) permitimos **recuperar** todas las *Entradas*.

En nuestro segundo `endpoint` (`POST /entries`) permitimos **añadir** una nueva *Entrada*.

En nuestro tercer `endpoint` (`GET /entries/<int:path_param_id>/comments`) permitimos **recuperar** todos los *Comentarios* de una *Entrada*.

Ahora toca...

POST /entries/<int:path_param_id>/comments

Vamos a modificar la función...

```
def entry_comments(request, path_param_id):
    if request.method != "GET":
        return JsonResponse({"error": "HTTP method not supported"}, status=405)
    comments = # Tu opción preferida para recuperar los comentarios
    json_response = []
    for row in comments:
        json_response.append(row.to_json())
    return JsonResponse(json_response, safe=False)
```

...y **prepararla** para procesar un POST:

```
@csrf_exempt
def entry_comments(request, path_param_id):
    if request.method == "GET":
        comments = # Tu opción preferida para recuperar los comentarios
        json_response = []
        for row in comments:
            json_response.append(row.to_json())
        return JsonResponse(json_response, safe=False)
    elif request.method == "POST":
        # Añadiremos un nuevo comentario
        # ...
    else:
        return JsonResponse({"error": "HTTP method not supported"}, status=405)
```

¿Qué datos añadiremos?

Vamos a esperar que el cliente HTTP envíe un **cuerpo de petición** como este:

```
{
  "new_content": "Excelente, me gusta mucho"
}
```

Y con respecto a *qué Entrada* asignar el nuevo *Comentario*... ¡eso ya viene en la URL `path_param_id` !

cURL

Al terminar nuestro `endpoint`, el siguiente POST enviado desde **cURL** debería generar un nuevo **Comment** :

```
curl http://localhost:8000/entries/1/comments -d '{"new_content": "Excelente, me gusta mucho"}'
```

Vamos allá

1) Sacar los datos de la petición HTTP

```
elif request.method == "POST":
    # Añadiremos un nuevo comentario
    client_json = json.loads(request.body)
    client_content = client_json.get("new_content", None)
    # ...
```

Podemos aprovechar a devolver un **400 Bad Request** si el cuerpo es incorrecto:

```
elif request.method == "POST":
    # Añadiremos un nuevo comentario
    client_json = json.loads(request.body)
    client_content = client_json.get("new_content", None)
    if client_content is None:
        return JsonResponse({"error": "Missing new_content"}, status=400)
    # ...
```

2) Crear una nueva instancia del modelo y guardarla con **.save()**

```
elif request.method == "POST":
    # Añadiremos un nuevo comentario
    client_json = json.loads(request.body)
    client_content = client_json.get("new_content", None)
    if client_content is None:
        return JsonResponse({"error": "Missing new_content"}, status=400)
    # Creamos una nueva instancia de Comment
    # Nótese pasamos entry_id=... en vez de entry=...
    # para indicar que nos referimos a la clave primaria (id)
    # del atributo. Esto se hace así para los models.ForeignKey
    new_comment = Comment(content=client_content, entry_id=path_param_id)
    new_comment.save()
    return JsonResponse({"new_comment_saved": True}, status=201)
```

También devolvemos un **201 Created** si todo va bien.

¡Terminado!

Los usuarios ya podrán añadir *Comentarios* a las publicaciones.

Por último

Comprueba que tu código pasa el *test* asociado a la tarea.

Haz **commit** y **push** para subir los cambios al repositorio.



[Rubén Montero @ruben.montero](#) changed milestone to [%Sprint 4](#) 2 days ago