


Open Opened 1 day ago by  **Rubén Montero**

Opcional: Enviar petición POST en Godot

Resumen

- Ejecutaremos el *backend* disponible en `django-backend/` en nuestro ordenador local
- Modificaremos *Dodge the Creeps!* para que mande un POST cuando se completa cada partida

Descripción

En esta tarea *opcional* se plantea un objetivo que te ayudará a familiarizarte con Godot y a tener más confianza para trabajar en *tu propio juego*.

¡Ojo! Esta tarea no es evaluable. Sólo *tu juego* contará para la nota del *sprint*.

Completando la tarea anterior

Al principio de la tarea anterior indicamos dos objetivos:

1. Leer un archivo JSON de disco con el *username*
2. Mandar petición POST para publicar la puntuación de la partida

¡Vamos a por el segundo!

La tarea

En primer lugar, desde un *terminal* (cmd.exe) **cambia** el directorio activo (`cd`) hasta la carpeta `django-backend/scoreboard/` de nuestro repositorio local.

Luego, **lanza** el API REST con:

```
python manage.py runserver
```

¿Qué es este API REST?

Un *sencilísimo* API REST con un único *endpoint* `/score` .

- Si mandas un `GET` , verás la lista de puntuaciones publicadas. Puedes visitarlo desde el navegador: <http://localhost:8000/score>
- Si mandas un `POST` debidamente formado, publicarás una nueva puntuación

¿Cómo es el POST?

Como este ejemplo:

```
POST /score
Content-Type: application/json
Client-Secret: abc

{
  "username": "TestUser04",
  "score": 39
}
```

¿Y cómo lo lanzamos?

¡Muy fácil! **Abre** Godot con tu proyecto *Dodge the Creeps!*. En el *script* principal `Main.gd` tenemos una función llamada `game_over()` . **Invoca** un nuevo método ahí:

```
func game_over():
    $ScoreTimer.stop()
    $MobTimer.stop()
    $HUD.show_game_over()
    $Music.stop()
    $DeathSound.play()
```

```
+ send_post_new_score()
```

...y luego, **añade** dicho método. El primer paso será ejecutar una *condición* de guarda, por si hubo algún problema cargando el `username` . Así:

```
func send_post_new_score():
    if username == null:
        printerr("Will NOT send POST data with score due to invalid username")
        printerr("There might have been an error loading user_data file")
        return
```

HTTPRequest

Es el tipo de *nodo* que se usa para mandar peticiones HTTP. Puede añadirse a la escena desde el editor. Nosotros lo haremos *programáticamente*, dentro del *script*. Para ello, **añade** este código:

```
func send_post_new_score():
    if username == null:
        printerr("Will NOT send POST data with score due to invalid username")
        printerr("There might have been an error loading user_data file")
        return

    # Username OK. Let's send the request
    var http_request = HTTPRequest.new()
    add_child(http_request)
```

Y, a continuación, **envía** una petición con cuerpo JSON y cabeceras adecuadas, así:

```
func send_post_new_score():
    if username == null:
        printerr("Will NOT send POST data with score due to invalid username")
        printerr("There might have been an error loading user_data file")
        return

    # Username OK. Let's send the request
    var http_request = HTTPRequest.new()
    add_child(http_request)
    var body = to_json({"username": username, "score": score})
    var headers = ["Content-Type: application/json", "Client-Secret: abc"] # CLIENT_SECRET should never be public! If lea
    http_request.request("http://127.0.0.1:8000/score", headers, false, HTTPClient.METHOD_POST, body)
```

Parámetros de `.request`

1. El *host* y su *endpoint*. En nuestro caso, `http://127.0.0.1:8000/score`
2. Las cabeceras, en formato *lista* de *Strings*
3. `true` si se usa HTTPS. `false` para HTTP sencillo. En producción, una aplicación publicada deberá emplear HTTPS siempre. De lo contrario, el `Client-Secret` carece de sentido pues viajará en texto claro por la red y se puede espiar fácilmente
4. Método HTTP
5. Cuerpo de la petición

¿Y ya está?

¡La petición ya debe enviarse! Ahora, para *escuchar* la respuesta del servidor, **añade** el siguiente código que conecta la señal `"request_completed"` :

```
func send_post_new_score():
    if username == null:
        printerr("Will NOT send POST data with score due to invalid username")
        printerr("There might have been an error loading user_data file")
        return

    # Username OK. Let's send the request
    var http_request = HTTPRequest.new()
    add_child(http_request)
+    http_request.connect("request_completed", self, "_on_server_has_responded")
    var body = to_json({"username": username, "score": score})
    var headers = ["Content-Type: application/json", "Client-Secret: abc"] # CLIENT_SECRET should never be public! If lea
    http_request.request("http://127.0.0.1:8000/score", headers, false, HTTPClient.METHOD_POST, body)
```

```
+  
+func _on_server_has_responded(result, response_code, headers, body):  
+    var response = parse_json(body.get_string_from_utf8())  
+    print("Server response:")  
+    print(response)
```

Esto es especialmente útil si queremos lanzar un **GET** para obtener información de un *endpoint*.

Nosotros, en este breve ejemplo, sólo lanzaremos un **POST** que publica la puntuación.

¡Adelante! Con el servidor REST en ejecución, **juega** una partida. Al terminar, deberías ver lo siguiente en la consola de Godot:

```
Godot Engine v3.5.stable.official.991bb6ac7 - https://godotengine.org  
OpenGL ES 3.0 Renderer: Intel(R) UHD Graphics  
Async. shader compilation: OFF  
  
username loaded correctly  
Server response:  
{created:True}
```

Si es así, **¡enhorabuena!** Has creado tu primer juego en Godot que se comunica con un API REST.



[Rubén Montero @ruben.montero](#) changed milestone to [%Sprint 4](#) 1 day ago