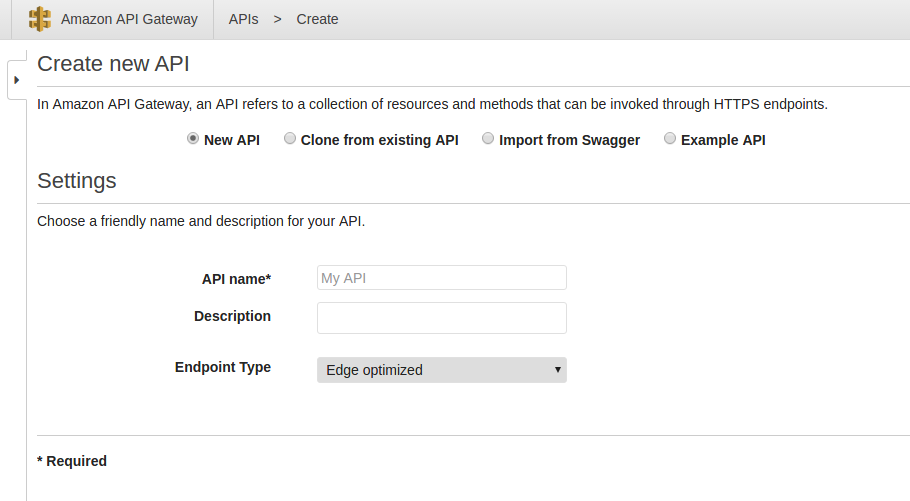
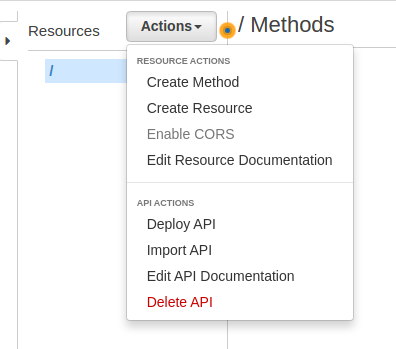
**Implementación de seguridad de una APP usando API GATEWAY**

Lo primero será crear una nueva API. Esto lo conseguimos directamente sobre la consola de Amazon API Gateway clicando sobre el botón **Create API**, asignamos un nombre y lo tenemos.



Ya que se basa en RESTful, para hacer la publicación de algún servicio, tendremos que definir el recurso y los métodos por los que atenderá a las peticiones.

En el menú **Actions** encontramos todas las operaciones que podemos realizar sobre una API.

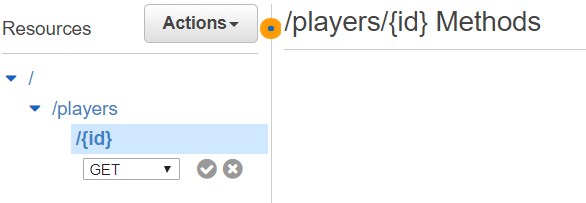


Primero, creamos el recurso que queremos exponer.

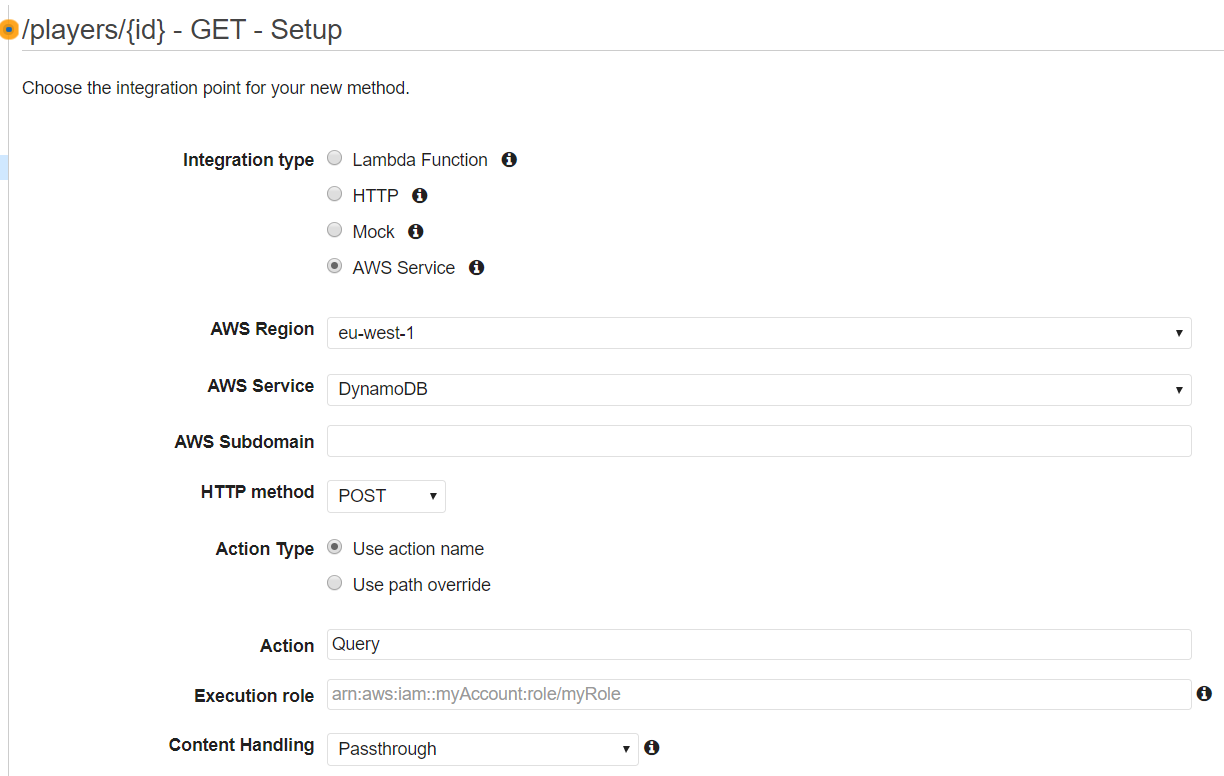


En este caso, crearemos un recurso llamado **players/{id}**.

Segundo, creamos el método HTTP al que vamos a asociar la operación.



En el siguiente paso debemos definir el tipo de integración que deseamos realizar. En este caso concreto, sería enlazar el recurso a la tabla en base de datos (recordemos que la misma debe ser previamente creada en DynamoDB).



Tal como mencionamos previamente, en esta pantalla pueden observarse todas las opciones que ofrece API Gateway para ser integrado.

Para continuar con el ejemplo, en este caso debemos rellenar toda la plantilla con los datos de localización y nombre del servicio con el que deseamos integrar, así como también la acción que será invocada.

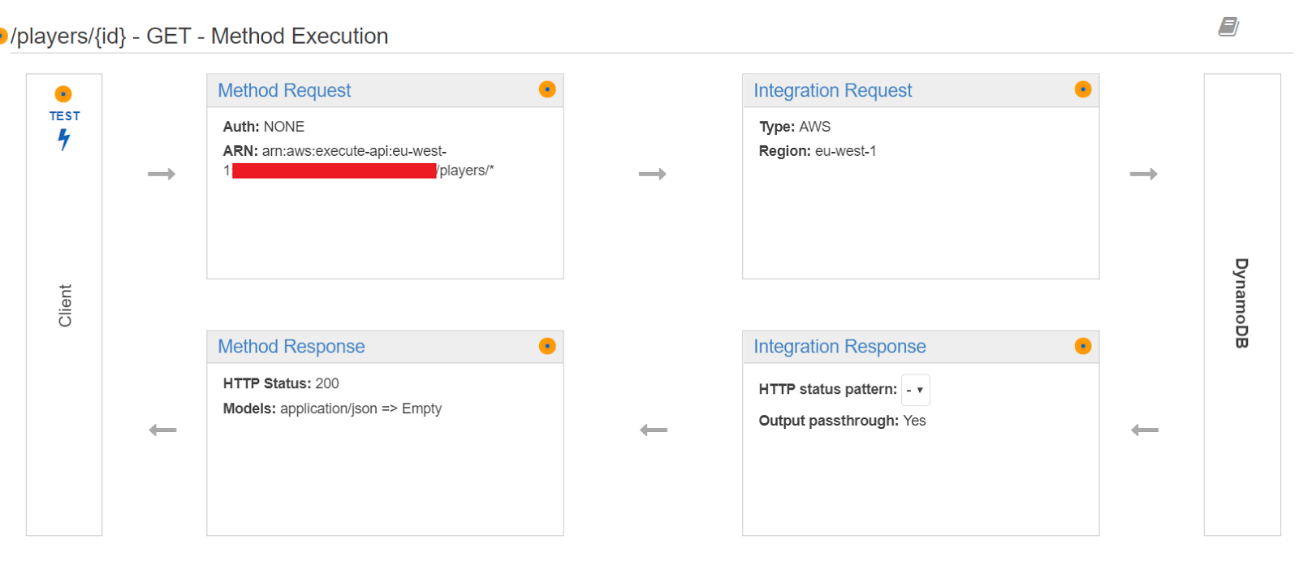
En este punto hay que hacer énfasis en lo siguiente:

1. Es importante resaltar que en la selección de HTTP method no se trata del mismo método que estamos exponiendo en la API que hemos creado previamente. Este se refiere al método que debemos utilizar para conectar con un servicio el particular al que vamos a conectar. Para el servicio DynamoDB, todas las peticiones deben ser realizadas a través del método POST, aún cuando lo que vayamos a realizar se una consulta de lectura.
2. Action, se refiere a la acción o función que queremos invocar dentro del servicio. En este caso vamos a realizar una consulta sobre la tabla player filtrando por el id. La API DynamoDB ofrece varias funcionalidades, tanto de consulta como de modificación (GetItem, UpdateItem, DeleteItem, PutItem, etc). El detalle del servicio DynamoDB está fuera del ámbito de este post.
3. Por razones obvias de seguridad, en AWS, para establecer la comunicación entre servicios, es estrictamente necesario definir las políticas y unos roles donde indiquemos las reglas de visibilidad entre servicios, e incluso qué funciones pueden ser invocadas entre sí. Esto se conoce dentro del ámbito de AWS como el Execution Role. Todo esto se define en la consola IAM, específicamente en los apartados de Policies y Roles\*\*\*.\*\*\* En el primero definimos acceso y visibilidad (reglas del juego), y el segundo es el perfil a quién le asignamos tales políticas y reglas de acceso.

Dado que estamos hablando de peticiones HTTP y que los parámetros pueden viajar bien sea a través de la URL (path/querystring), en el cuerpo o en la cabecera de esta (body y header respectivamente), la configuración de los mismos vendrá condicionada al servicio que se desee invocar.

Ahora bien, para el ejemplo que hemos propuesto, para invocar la acción Query dentro del servicio DynamoDB, es necesario proporcionar nombre de la tabla y los campos por los que que realizará el filtro (requisitos mínimos de la acción).

En todo caso, el mapeo y transformación tanto de los argumentos de entrada como los de salida, los podemos configurar dentro de la sección Method Execution.

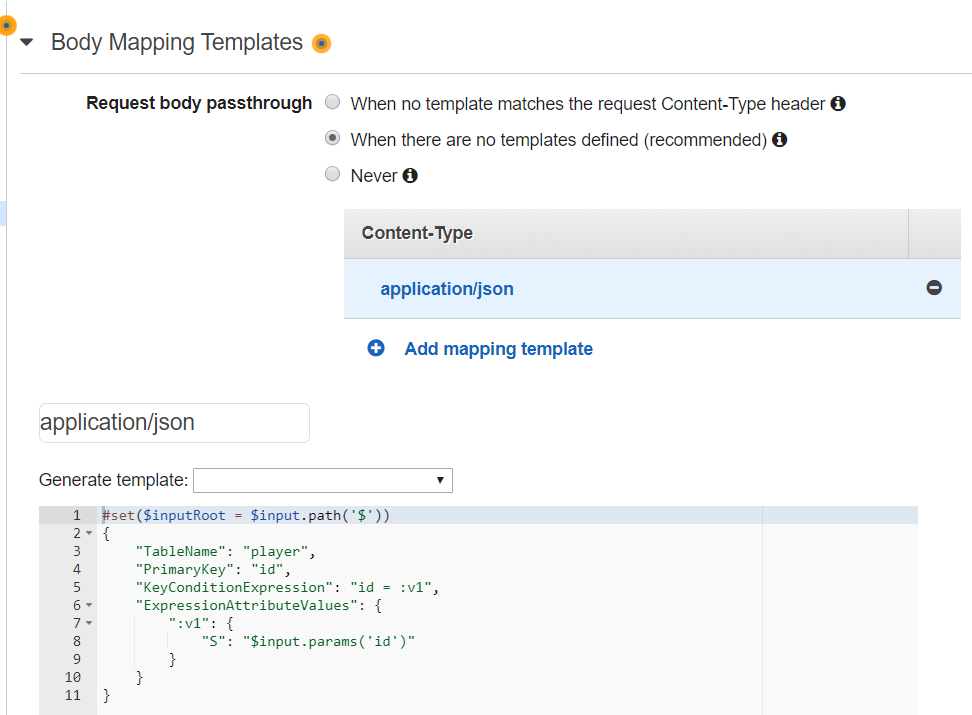


Donde:

* En el Method Request se indican los parámetros de la petición, cadena de parámetros, el cuerpo y cabeceras.
* El Integration Request se refiere a la captura y transformación de los datos que enviaremos al servicio a invocar (bien sea una función Lambda, un endpoint interno o externo, u otro servicio dentro de AWS). Este paso puede ser también visto como el pase de parámetros de la API al servicio.
* Por lo general, las respuestas de los servicios no vienen expresadas en el formato más adecuado, o mejor dicho, no vienen en el formato que requerimos. De igual manera que sucede en los parámetros de entrada, es posible hacer un mapeo o transformación de la respuesta recibida desde el servicio que invocamos y así podamos presentarla en nuestra API de la forma deseada. Esto lo podemos conseguir editando el apartado Integration Response.
* Por último, tenemos el Method Response en el que podemos hacer alguna transformación o proyección de datos con respecto a la presentación de resultados desde la API.

Con respecto al ejemplo:

* El Method Request no lo vamos a modificar, ya que nuestro método recibirá el parámetro de entrada desde el mismo recurso.
* El Integration Request, este apartado sí que lo configuraremos, ya que para la integración con el servicio DynamoDB es necesario indicar una serie de valores tales como nombre de la tabla a consultar, campos que deseamos mostrar, filtro a aplicar, ordenación de resultados entre otros, que son requeridos para la ejecución de la acción.



El cuerpo de la petición lo establecemos desde la opción Body Mapping Templates del Integration Request. En este punto podemos capturar los datos que viajan desde la entrada hasta el método, parsearlos (si es necesario) y luego inyectarlos a la llamada del servicio con el que estamos enlazando.

El cuerpo de la petición se expresa en formato JSON, por lo que el Content-Type será application/json. Siempre podemos partir de una plantilla que exista o una en blanco.

En la instrucción #set($inputRoot = $input.path(‘$’)) obtendremos todos los datos con la que se ha realizado la petición. De allí podremos extraer: parámetros, cabeceras, cuerpo y contexto; bien para añadir algún tipo de lógica o transformación, o bien para redirigirlos como entrada del siguiente punto de enlace que no es más que la invocación al servicio.

El JSON definido establece lo siguiente:

1. TableName. nombre de la tabla a consultar.
2. PrimaryKey. campo primario dentro de la tabla.
3. KeyConditionExpression. condición de filtrado (campo *id*).
4. ExpressionAttributeValues. es aquí donde hacemos la magia, es decir, aquí indicamos de dónde provienen los datos de entrada ($input.params(‘id’)) y dónde los inyectamos.

Con esto último nuestra API está preparada para consultar el registro de la tabla player a través del método GET del recurso /players/{id}. Lo podemos comprobar ejecutando la prueba que nos provee la consola.

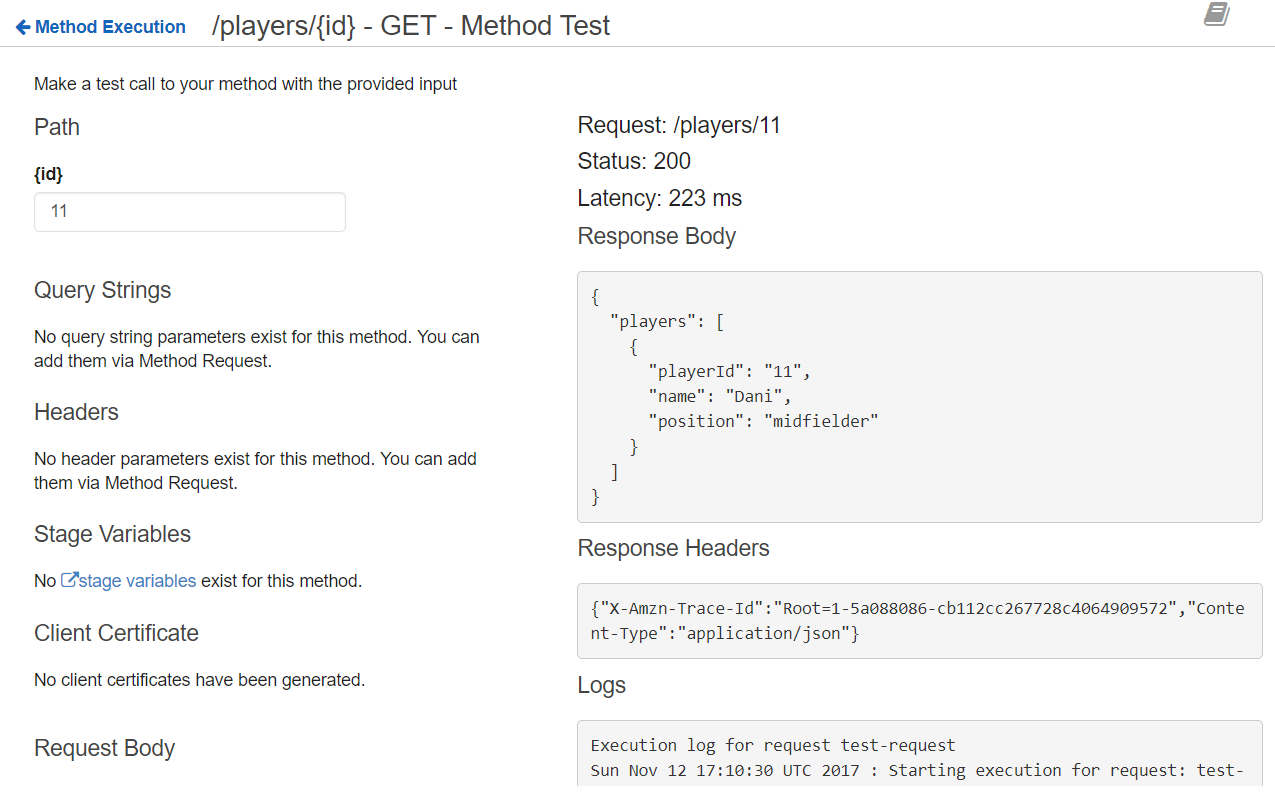


De forma similar que ocurre en el Integration Request, en el Integration Response podemos configurar el apartado Body Mapping Templates. Pero en este caso, para hacer una proyección de los datos obtenidos, es decir, podemos aplicar alguna transformación a la respuesta, y de esta manera presentar la salida como sea más conveniente.



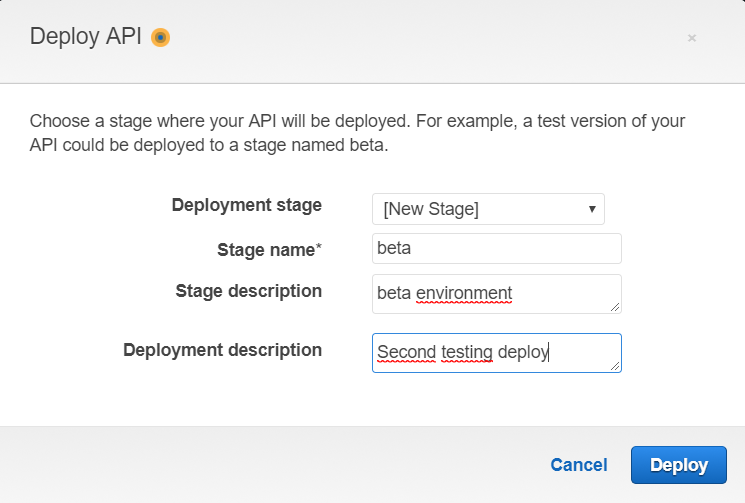
Como podemos observar, aquí preparamos la salida para que tenga una estructura más manejable. Esto es siempre que lo necesitemos, ya que este punto no es imprescindible para llevar a cabo la integración de la API con un servicio.

Si ejecutamos nuevamente el test veremos claramente la diferencia:



El despliegue de una API es más sencillo de lo que podríamos imaginar, solo es necesario ejecutar la acción Deploy API.

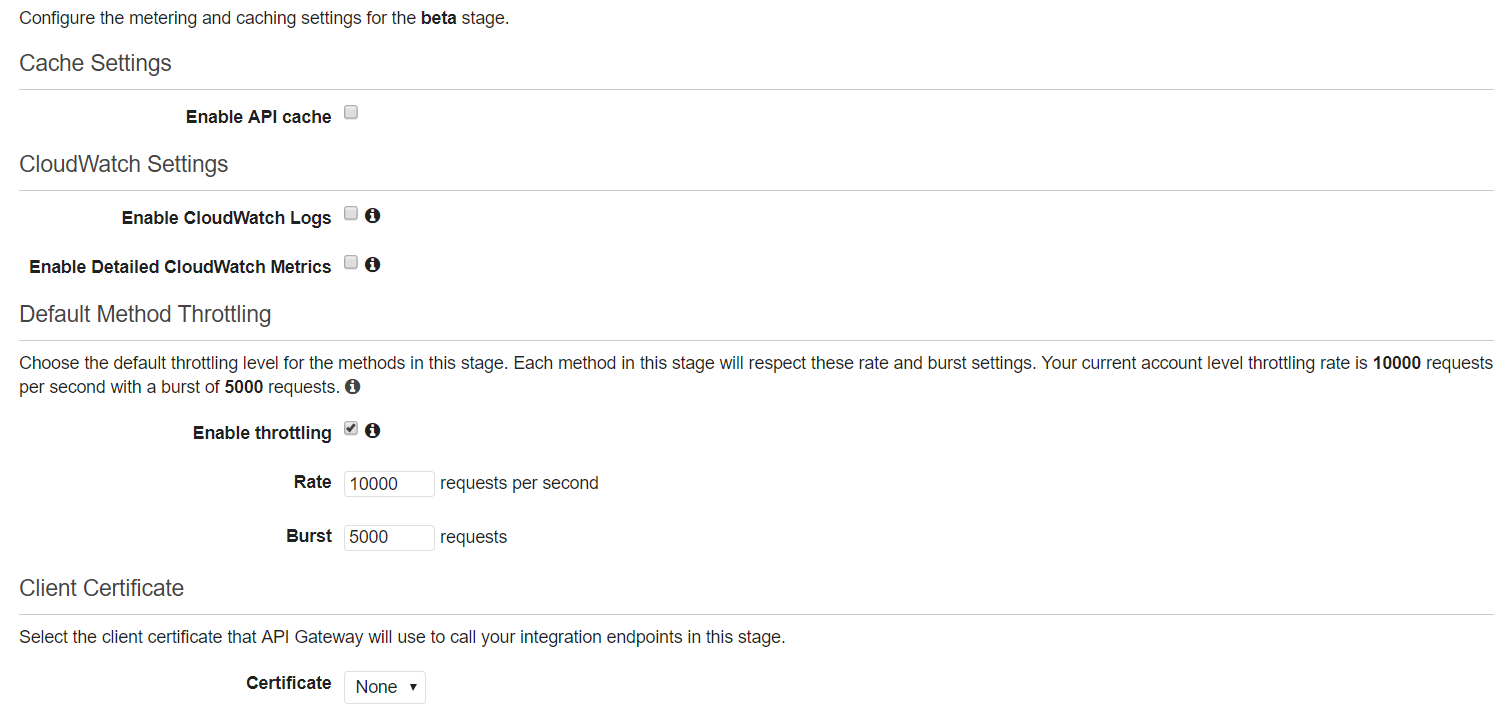
Seguidamente mostrará un popup donde indicaremos la fase o entorno, acompañado por un nombre y una breve descripción.



Con este paso ya hemos conseguido publicar los servicios a través de nuestra API.

Cada Stage que creamos nos suministrará una URL a través de la cual podremos atracar un servicio utilizando el método adecuado.

Una vez realizado el despliegue podemos realizar una serie de configuraciones con respecto al rendimiento, seguridad y monitorización de la API. Esto lo podemos conseguir editando el apartado **Stages**.



Aquí podemos activar el uso de la caché para mejorar los tiempos de respuesta. También podemos ajustar el umbral de peticiones por segundo y ráfagas de peticiones concurrentes.

Y en caso de que la integración así lo requiera, también podemos habilitar la opción de envío de certificado digital. Los certificados podemos añadirlos directamente en el apartado Client Certificates.

Finalmente, cada Stage proporciona una URL con el que podemos consumir la API.

