

50ª Conferencia Latinoamericana de Informática (L CLEI 2024)

Bahía Blanca, 12 al 16 de agosto de 2024
Universidad Nacional del Sur, SADIO

Trabajo Final – Curso de introducción al desarrollo de software cuántico

Generación y despliegue de servicios cuánticos haciendo uso de una especificación OpenAPI

Antes de dar comienzo a este trabajo final hay que tener un concepto claro, y es el de la tecnología OpenAPI. OpenAPI es una especificación que se usa para describir las interfaces de API de REST. Describe la API sin necesidad de acceso al código fuente ni a documentación adicional. La especificación es legible tanto por humanos como por máquinas. Para obtener más información, consulta la [documentación de especificación de OpenAPI](#).



Dado que la descripción de OpenAPI es legible por máquina, puede usarla para hacer cosas como las siguientes:

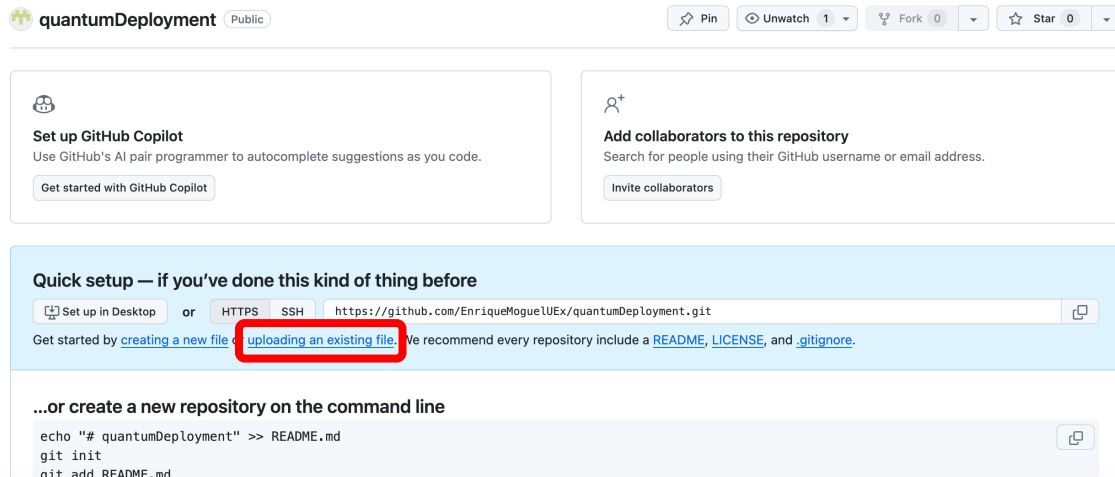
- Generar bibliotecas para facilitar el uso de la API de REST;
- Validar y probar una integración que usa la API de REST;
- Explorar e interactuar con la API de REST mediante herramientas de terceros como Insomnia o Postman.

Por ejemplo, GitHub usa la descripción de OpenAPI para generar los SDK de Octokit. GitHub también usa la descripción de OpenAPI para generar la documentación de referencia de la API de REST para cada punto de conexión.

Aprovechando el potencial de OpenAPI y las técnicas de DevOps de integración continua y despliegue continuo, desplegaremos un servicio cuántico siguiendo este manual paso a paso:

PASO 1: Descargar el repositorio quantumDeployment del siguiente repositorio:
<https://github.com/EnriqueMoguelUEx/quantumDeployment>

PASO 2: Subir el proyecto (descomprimido y el contenido de dentro de la carpeta) a tu repositorio de GitHub a través de los siguientes pasos:
Create Repository > Repository Name > Uploading an existing file > Commit changes



- - - -

Bahía Blanca			
Nombre	Fecha de modificación	Tamaño	Clase
openapi_old.yaml	ayer, 16:04	6 KB	YAML
openapi_tutorial.yaml	ayer, 16:04	19 KB	YAML
openapi-generator-cli.jar	ayer, 16:04	25,6 MB	Archiv...de Java
openapi.yaml	ayer, 16:04	44 KB	YAML
openapiBase.yaml	ayer, 16:04	4 KB	YAML
README.md	ayer, 16:04	124 bytes	Archivo md
requirements.txt	ayer, 16:04	773 bytes	Texto
sonar.py	ayer, 16:04	1 KB	py

PASO 3: Crear Action a través de la pestaña Actions:
Actions > set up a workflow yourself

Copiar en dicha caja de texto el código fuente del siguiente archivo YAML:
<https://github.com/EnriqueMoguelUEx/quantumDeployment/blob/main/.github/workflows/main.yml>

> Commit changes

PASO 4: Ver Action > main.yml > Validate

El Action ya se está ejecutando

```
validate (3.8)
Started 42s ago

Build with Maven 6s
120 circuit.cx(qreg_q[1], qreg_q[3])
121 circuit.cx(qreg_q[2], qreg_q[3])
122 circuit.barrier(qreg_q[0], qreg_q[1], qreg_q[2], qreg_q[3])
123 circuit.x(qreg_q[0])
124 circuit.h(qreg_q[1])
125 circuit.x(qreg_q[2])
126 circuit.h(qreg_q[0])
127 circuit.h(qreg_q[2])
128 circuit.barrier(qreg_q[0], qreg_q[1], qreg_q[2], qreg_q[3])
129 circuit.measure(qreg_q[0], creg_c[0])
130 circuit.measure(qreg_q[1], creg_c[1])
131 circuit.measure(qreg_q[2], creg_c[1])
132
133 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
134 from numpy import pi
135 Es circuito en quirk
136 Es circuito en código
139
```

PASO 5: Una vez finalizada la ejecución, acceder a Swagger:

En el apartado **Deploy Stager** pulsa el enlace que viene en el path:

```
validate (3.8)
succeeded now in 1m 1s

Deploy Stage 12s
20 100 341 0 0 100 341 0 33 0:00:10 0:00:10 --:--:-- 0
27 100 341 0 0 100 341 0 30 0:00:11 0:00:11 --:--:-- 0
28 100 440 100 99 100 341 8 28 0:00:12 0:00:12 --:--:-- 20
29 100 440 100 99 100 341 8 28 0:00:12 0:00:12 --:--:-- 26
30 {"container_name":"quantumdeployment", "path":"http://quantumservicesdeployment.spilab.es:8084/ui"}

Post Set up Python 3.8 0s
1 Post job cleanup.

Post Build python API with YAML 0s
1 Post job cleanup.

Post Run actions/checkout@v3 0s
1 Post job cleanup.
2 /usr/bin/git version
3 git version 2.46.0
4 Temporarily overriding HOME='/home/runner/work/_temp/9dafab9b-4995-4024-8950-dedad6d6b3a4' before making global git config changes
5 Adding repository directory to the temporary git global config as a safe directory
6 /usr/bin/git config --global --add safe.directory /home/runner/work/CursoBahiaBlanca/CursoBahiaBlanca
7 /usr/bin/git config --local --name-only --get-regexp core\.sshCommand
8 /usr/bin/git submodule foreach --recursive sh -c "git config --local --name-only --get-regexp 'core\.sshCommand' && git config --local --unset-all 'core.sshCommand' || :"
9 /usr/bin/git config --local --name-only --get-regexp http\.https\:\/\/github\.com\/\.extraheader
10 http.https://github.com/.extraheader
```

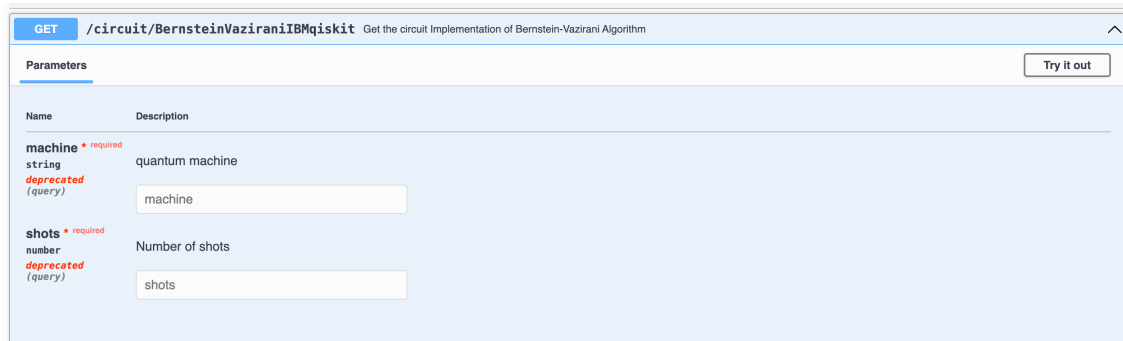
!!!Enhorabuena!!! Ya tienes tus servicios cuánticos desplegados.

PASO 6: Vamos a probarlo.

Pulsa sobre el circuito que quieras ejecutar > *Try it out*

Para la variable Machine -> “local”.

Y para la variable Shots debe ser un número entero.



Ahora tienes que rellenar el siguiente cuestionario:

<https://forms.gle/U8jppqM6twQUqwoHGA>

One more thing!!!

Si ahora quisieras ejecutar tu propio circuito, deberías modificar el archivo `openapi_tutorial.yaml`, como por ejemplo:

```
CursoBahiaBlanca / openapi_tutorial.yaml
Code Blame 610 Lines (591 loc) · 18.6 KB Code 55% faster with GitHub Copilot

311
312 /circuit/GroverIBMquirk:
313   get:
314     tags:
315       - quantum_code
316     summary: Get the circuit implementation of Grover Algorithm
317     description: ''
318     operationId: GroverIBMquirk
319     parameters:
320       - name: machine
321         in: query
322         description: quantum machine
323         required: true
324         style: form
325         explode: false
326         deprecated: true
327         schema:
328           type: string
329       - name: shots
330         in: query
331         description: Number of shots
332         required: true
333         style: form
334         explode: false
335         deprecated: true
336         schema:
337           type: number
338     responses:
339       '200':
340         description: successful operation
341       '405':
342         description: Invalid execution
343     x-quantumCode: 'https://algassert.com/quirk#circuit={["H","H",[1,"X"],[1,"H"],["*","X"],[1,"H"],[1,"X"],[1,"H"],[1,"Measure"]]'
344     x-quantumProvider: 'ibm'
```

Y en la línea `x-quantumCode` debe ir vuestro código quirk o qiskit.