



Máster Universitario En Investigación En Ingeniería De Software Y Sistemas Informáticos

**Desarrollo De Líneas De Producto Software Mediante Un
Enfoque Generativo** TRABAJO DE INVESTIGACIÓN

Autor

César Hugo Bárzano Cruz

—
2019/2020

APIGENA: Generación de APIs como linea de productos software

César Hugo Bárzano Cruz¹

UNED,
hugobarzano@gmail.com

Resumen En el ámbito de la web 2.0 y las soluciones cloud, el uso y consumo de servicios o productos software basados en el protocolo HTTP se ha incrementado notablemente. A raíz de la gran demanda de este tipo de productos software, esta publicación propone la implementación de una linea de productos software mediante un enfoque generativo, concretamente de aplicaciones API-REST con el objetivo de generalizar este tipo de aplicaciones para alcanzar una producción en escala. En esta publicación se muestra una implementación capaz de dado un modelo de específico de dominio definido en formato JSON producir el código fuente y configuración necesarias para realizar operaciones CRUD sobre este tipo de modelos mediante una interfaz API-REST. La implementación que aquí se muestra es capaz de producir código fuente en los siguientes lenguajes: Go, Python, Javascript. Las APIs generadas incluyen todo lo necesario para ser auto-contenidas, es decir, contienen todas las librerías o dependencias necesarias para funcionar correctamente. De manera adicional, esta publicación muestra como el uso de Docker como entorno virtual, facilita el desarrollo y la portabilidad del software aquí propuesto.

Keywords: Lineas de Productos Software, Programación Generativa, API-Rest, Cloud, Golang, Docker

1. Introducción

1.1. Problema

El problema que aquí se propone consiste en ¿ Como generalizar y automatizar el proceso de especificación, documentación y desarrollo de servicios web de naturaleza API-REST? ¿ Es posible abordar el problema para producir APIs en distintos lenguajes de programación?

1.2. Estado del Arte

Actualmente existen herramientas capaces de facilitar y automatizar el proceso de desarrollo de APIs mediante un enfoque generativo. Las herramientas que se proponen a continuación tiene en común el uso de una especificación

OpenAPI[1] que permite describir APIs para ser usadas posteriormente a la hora de producir, consumir y visualizar servicios web. Por otro lado tambien se proponen algunas herramientas las cuales son capaces de dada una especificación de API son capaces de generar documentación e interfaz de usuario asociada. Todas las herramientas propuestas son de naturaleza Open-Source. Existen otros tipos de especificación que son igual de validos, como por ejemplo:

- API Blueprint
- WSDL
- RAML
- Swagger 1.0
- Swagger 2.0

OpenAPI Generator Permite la generación de bibliotecas de cliente API (SDK), stubs de servidor, documentación y configuración dada una especificación OpenAPI. Para obtener mas información acerca de este software, puede consultarse el repositorio del proyecto openapi-generator.[2]

Swagger Codegen De igual forma que OpenApi Generator, Swagger Codegen permite la generación de clientes, servidores y documentación de API dada una especificación OpenApi, para obtener mas información sobre esta herramienta puede consultarse el repositorio del proyecto.[3]

LucyBot DocGen Permite la generación estática de documentación así como una consola de pruebas para las operaciones especificadas para la API. También permite configurar el dominio y el estilo con el que se produce la salida. Pueden consultarse ejemplos de uso en la documentación de la herramienta Lucybot[4]

DapperDox Permite generar documentación relacionada con la API especificada así como explorador para navegador web permitiendo una mayor comprensión de las operaciones generadas. Esta herramienta tiene la particularidad de poder consumir diversas especificaciones de diversas APIs lo que facilita trabajar con distintos micro-servicios.

Widdershins Genera documentación estática de la API especificada así como vistas HTML para probar métodos, parámetros de los métodos y respuestas de los métodos. La peculiaridad de Widdershins[5] reside en que es capaz de producir código fuente de ejemplo en distintos lenguajes como por ejemplo:

- Ruby
- Python
- JavaScript
- Java

1.3. Solución

Tras el análisis realizado sobre las actuales herramientas capaces de satisfacer en mayor o menor medida el problema propuesto, se aprecia una dependencia directa en todas ellas a la especificación de la API, sea en un formato o en otro, por ello, en este generador se pretende ir un paso mas allá y dado un modelo especificó de dominio en formato JSON:

```
1 {  
2   "type": "Apple",  
3   "name": "Golden",  
4   "price": 1.5,  
5   "inStock": true  
6 }
```

producir dicha especificación en formato OpenApi o Swagger 2.0 para posteriormente ser aprovechada por el código fuente generado para satisfacer operaciones CRUD sobre el modelo de entrada.

El generador aquí propuesto es capaz de producir código fuente en los siguientes lenguajes de programación:

- Go
- Python
- JavaScript

Debido a que el generador esta implementado en GO, esto facilitar la generación de modelos de datos complejos de manera nativa. En el caso de Python y JavaScript, el código fuente generado generaliza el tratamiento de datos y no define un modelo concreto si no que utiliza la entidad definida en la especificación. De igual forma, el código fuente generado para Python y JavaScript no define rutas del tipo /api o /api/object si no que las rutas de la API son inferidas de la especificación generada. Este enfoque generativo proporciona de manera adicional una interfaz de usuario donde consultar la documentación del API generada, así como una consola para ejecutar operaciones de creación, consulta, borrado y actualización, incluyendo ejemplos de las peticiones a realizar y de las posibles respuestas obtenidas. Las APIs generadas son de carácter auto-contenido, es decir junto a la especificación y al código fuente se produce la configuración necesaria para poder instalar las librerías o dependencias que cada tecnología necesita para ejecutar su APIs.

Como parte adicional, el generador incluye una imagen Docker para facilitar la portabilidad, ejecución y pruebas de las APIs generadas. En las siguientes secciones se muestran detalles de implementación, un caso de uso a modo de ejemplo y los posibles trabajos futuros que pueden resultar del aquí propuesto.

2. Implementación

El generador que aquí se plantea ha sido desarrollado utilizando GO como lenguaje de programación. De este lenguaje se aprovecha el uso de interfaces y su motor de templates que facilita la generación del código fuente.

2.1. Interface

```

1 // Generator interface definition
2 type Generator interface {
3     Init() Generator
4     WithName(name string) Generator
5     WithPort(port int) Generator
6     WithInputSpec(spec interface{}) Generator
7     WithOutputPath(path string) Generator
8     Generate()
9 }

```

2.2. Template Engine

3. Caso de Uso

4. Trabajos Futuros

En el campo de la robótica, la percepción del entorno donde estos agentes han de desempeñar ciertas tareas es parcial, el entorno es cambiante, los sensores que utilizan pueden sufrir variaciones y las tareas a realizar pueden no estar especificadas de ante mano.

Referencias

1. S. Software, What is openapi?
URL <https://swagger.io/docs/specification/about/>
2. S. Software, Openapitools/openapi-generator.
URL <https://github.com/openapitools/openapi-generator>
3. S. Software, swagger-api/swagger-codegen.
URL <https://github.com/swagger-api/swagger-codegen>
4. L. Inc, Lucybot docs.
URL <http://docs.lucybot.com/Introduction>
5. Mermade, Mermade/widdershins.
URL <https://github.com/mermade/widdershins>

Notas y Comentarios. Esta entrega es una tentativa de lo que será el trabajo completo. Con ella, se espera una pequeña aproximación de lo que será