GOBIERNO DE APIs, IMPLEMENTACIÓN Y EXPERIMENTACIÓN CON API-FIRST Y OPENAPI EN EL PROYECTO P2P ENERGÍA TRANSACTIVA

Juan José Londoño Tirado

Informe de prácticas para optar al título de Ingeniero de sistemas

Asesor
Diego Iván Oliveros Acosta
Magister en Arquitecturas de Tecnologías de Información

Universidad de Antioquia
Facultad de ingeniería
Ingeniería de sistemas
El Carmen de Viboral, Antioquia
2022

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

API	Application Programming Interface
CI	Continuous Integration
DEA	Dirección de Estrategia y Arquitectura de TI
EPM	Empresas públicas de Medellín
JSON	JavaScript Object Notation
MD	Markdown
OAS	OpenAPI Specification
P2P	Peer to Peer
PDF	Portable Document Format
YAML	YAML Ain't Markup Language

RESUMEN

EPM es una empresa de servicios públicos domiciliarios con una responsabilidad social y ambiental. EPM imprime los más altos estándares internacionales de calidad a los servicios que presta: energía eléctrica, gas por red, agua y saneamiento. La gestión de APIs es la estrategia elegida para la interacción entre aplicaciones en EPM, asegurando el intercambio de información y funcionalidades en los sistemas internos de la empresa y con sistemas de terceros.

A partir de lo anterior, la implementación de buenas prácticas para el gobierno de APIs, junto con el enfoque de desarrollo de APIs, API-First, son claves para poder lograr la estandarización de estas y, por lo tanto, reducir tiempo, complejidad y costo de desarrollo de las APIs, además de aumentar su calidad y promover su economía.

La implementación del enfoque API-First se llevó a cabo mediante la experimentación en el proyecto P2P energía transactiva en donde, haciendo uso de Spectral, herramienta conocida como API-linter, en el proceso de validación de contratos de APIs. Gracias a esta herramienta se logró obtener porcentajes entre el 95% y el 99.9% de reducción de tiempos de validación. Herramienta que a su vez se trajo como lineamiento en la estrategia de gobierno de APIs en EPM.

Palabras clave — **API-First, Gobierno de APIs, OpenAPI, Spectral, linting, Diseño de APIs, estandarización, gestión de APIs.**

ABSTRACT

EPM is a home public services company with a social and environmental responsibility. EPM has the highest international quality standards for the services it provides: electricity, gas, water and sanitation. API management is the chosen strategy for application interaction, assuring information and functionalities exchange between internal and external systems.

From the above, implementing good practices in API government, along with the API development approach, API-First, are key ways to standardize EPM's API, therefore, reduce time, complexity, and cost of developing API as well as increasing quality and promote their economy.

The API-First approach implementation was carried out by experimenting in the P2P transactive energy project and using Spectral, a tool known as API-linter, in the API contract validation process. By using Spectral, the validation time taken in the validation process was reduced between 95% and 99.9%. This tool also was used as a guideline in the EPM's API governance strategy.

Keywords — **API-First, API government, OpenAPI, Spectral, linting, API design, standardization, API management.**

I. INTRODUCCIÓN

EPM es una empresa de servicios públicos domiciliarios, fue creada el 6 de agosto de 1955, con una responsabilidad social y ambiental, que le da sentido a su origen, a su desarrollo y a su estrategia de negocios. EPM imprime los más altos estándares internacionales de calidad a los servicios que presta: energía eléctrica, gas por red, agua y saneamiento.

Las prácticas tuvieron lugar en la vicepresidencia de nuevos negocios, innovación y tecnología, la cual, en parte, está compuesta por la gerencia de tecnología de información, de la misma manera, esta está compuesta por la dirección de estrategia y arquitectura de TI (DEA). La DEA es la encargada de liderar la definición, diseño y evolución del desarrollo de estrategias, gobiernos, modelos de operación, arquitecturas de TI y los procesos de la función TI; con sus respectivos mapas de ruta que permiten responder de forma ágil y flexible a los requerimientos del negocio.

Durante los últimos años se ha venido desestimulando el uso de aplicaciones monolíticas, donde la interoperabilidad es solo entre aplicaciones internas. Al mismo tiempo, se han incorporado enfoques actuales, entre ellos la interoperabilidad y estrategias de desarrollo modernas, como lo son los buses empresariales, las plataformas de integración en la nube y el enfoque de desarrollo API-First.

Debido a esto, la mediación y gestión de APIs es el mecanismo establecido para la interacción entre aplicaciones, asegurando el intercambio de información y funcionalidades en los sistemas internos de la empresa y con sistemas de terceros, con el fin de tener eficiencia en las operaciones de los negocios y atención a los clientes, promoviendo así la economía de las APIs.

II. OBJETIVOS

A. Objetivo general

Implementar buenas prácticas de gobierno de APIs en EPM utilizando y experimentando con el enfoque de desarrollo API-First, con el que se pretende reducir los tiempos, complejidad y costos de desarrollo de las APIs en los diferentes equipos de trabajo de la empresa.

B. Objetivos específicos

- Apoyar en la definición de la estrategia, proceso y lineamientos de gobierno de APIs de la empresa, a través de la participación semanal en la mesa de gobierno de APIs.

- Apoyar en la validación de contratos de APIs de alta calidad implementando experimentos con el enfoque API-First en el proyecto P2P energía transactiva para reducir tiempo, complejidad y costo de desarrollo en los equipos.

- Apoyar en la actualización del catálogo de APIs, cambiando la información de prueba por información real y verdadera de todas las APIs utilizadas actualmente en EPM para que puedan ser consultadas por los diferentes equipos de trabajo.

III. MARCO TEÓRICO

A. API

Application Programming Interface (API) es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. Las APIs permiten que los productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados, facilitando que los desarrolladores puedan crear programas concretos para bases de datos, sistemas operativos, plataformas online y/o redes sociales [1].

B. Economía de APIs

Este concepto se refiere todas las características, técnicas o métodos que puede hacer a una API más costo-eficiente, esto significa que haga que se minimicen los costos y el tiempo de desarrollo, y se maximicen la ganancias directas o indirectas a través de ventas o eficiencia, respectivamente; y la popularidad de la API. La principal característica de las APIs es que promueve su propia economía, exponiendo sus funcionalidades con otras aplicaciones sin necesidad de exponer cómo están implementadas, esto simplifica el desarrollo de otras aplicaciones y permite ahorrar tiempo y dinero [2]. Otros conceptos que promueven la economía de APIs que se hablarán en este documento son API-First, ciclo de vida y gobierno de APIs.

C. API-First

Es un enfoque de desarrollo de APIs el cual toma a la API como "cliente de primera clase" y propone que se diseñe y desarrolle en primer momento. Luego de que la API haya sido desarrollada y se haya creado un mock de la misma, el equipo podrá implementar el resto de la aplicación en paralelo, lo cual hace que se pueda exponer a producción en un menor tiempo, la experiencia de desarrollo sea mejor y se puedan crear casos de prueba desde el inicio. El enfoque API-First se divide en dos conceptos importantes:

1. *Diseño API-First*

Este concepto se refiere al diseño de la API y la planeación de qué funcionalidades tendrá, qué información va a exponer, cómo va a escalar y qué nuevas funcionalidades podrá tener en el futuro.

2. *Desarrollo API-First*

Este concepto se refiere al desarrollo de la API creando un contrato donde se expongan las funcionalidades previamente diseñadas. Luego se genera un mock de la API con el que los desarrolladores podrán implementar el resto de la aplicación en paralelo. Los desarrolladores son considerados como el primer consumidor de la API y podrán dar feedbacks desde el inicio [3].

D. Ciclo de vida de las APIs

Las API también son software, por lo tanto, también tienen un ciclo de vida que va desde su análisis hasta el retiro de esta. Aquí se describe lo que se debe realizar en cada una de sus cuatro fases: Análisis, desarrollo, operación y retiro.

1. *Análisis*

En esta primera fase se tiene en cuenta cosas como la estrategia que se utilizará para el desarrollo de la API, el modelo de monetización y quiénes serán los consumidores.

2. *Desarrollo*

En esta fase se implementará y probará la API siguiendo la estrategia de desarrollo elegida en la fase de análisis.

3. Operación

En esta fase la API ya está implementada, se tendrán en cuenta cosas como despliegue a producción y monitorización (rendimiento, salud de la API, métricas, etc.).

4. Retiro

En esta última fase donde el API se saca de producción, se tendrán en cuenta cosas como cambios de versión, problemas de seguridad, utilidad y uso por parte de los consumidores.[4]

E. Gobierno de APIs

Es el proceso más importante dentro de la gestión de APIs. Su objetivo principal es estandarizar los procedimientos dentro del ciclo de vida para garantizar estabilidad y coherencia en las APIs. El gobierno de APIs se encarga de definir los estándares a utilizar dentro de la empresa para temas como documentación, herramientas tecnológicas, políticas de versionamiento y escalado; monetización, pruebas, publicación en el catálogo de APIs, seguridad, monitorización, nombramiento de endpoints, códigos de respuesta y manejo de errores [5].

F. OpenAPI

Es una especificación que define los estándares para la descripción de APIs Rest que permite que a partir de un archivo OpenAPI máquinas y humanos puedan descubrir y entender las capacidades de un servicio sin tener que acceder al código fuente de este. En el archivo OpenAPI se describen cosas como la información de contacto, licencias, términos de uso, métodos de autenticación, los endpoints disponibles con sus respectivos métodos http y sus parámetros de entrada y salida [6].

G. API-linter

Los API linters son herramientas que se utilizan para hacer **linting** a los contratos de las APIs. Linting es el proceso de validar que un código esté construido como se esperaba; este proceso se hace generalmente utilizando un conjunto de reglas que analizan la estructura, errores de estilo, bugs, etc. [7].

H. Spectral

Es un API linter open-source creado por la empresa Stoplight, entre sus muchas características tiene que su implementación es muy sencilla, tiene documentación completa, y comunidad creciente. Como herramienta de validación de contratos puede validar OpenAPI versión 2.0 y 3.0; además de, AsyncAPI, los contratos pueden estar en archivos JSON/YAML. Su más importante característica es que se puede integrar en un proceso de CI, ya que es un paquete de npm; y también se puede utilizar localmente [8].

IV. METODOLOGÍA

Para alcanzar los objetivos planteados, el desarrollo de las prácticas se realizó en dos fases principales: inducción y apoyo y participación.

A. Fase de inducción

En esta primera fase se envió al estudiante la documentación necesaria para realizar la inducción en la empresa, para poder familiarizarse con el entorno de trabajo y las posibilidades que ofrece la empresa.

Como actividad principal de esta fase se hizo una investigación por parte del estudiante sobre los temas principales que se trabajaron en el período de las prácticas académicas: Gobierno de APIs y API-First.

B. Fase de apoyo y participación

En esta fase, en primera instancia, se contextualizó al estudiante sobre el gobierno de APIs que se utilizaba en la empresa y la estrategia que se tenía en ese entonces. También se introdujo al estudiante en el proyecto P2P energía transactiva, el cual se pretendía desarrollar utilizando el enfoque de desarrollo API-First.

Posteriormente, participó en la mesa de gobierno de APIs donde el estudiante apoyó en la definición de la estrategia de gobierno de APIs, específicamente, apoyando en la definición de lineamientos para el diseño de APIs.

También se asignó al estudiante la validación de los contratos de las APIs en el proyecto P2P energía transactiva; para esto se basó en el enfoque API-First utilizando la herramienta Spectral; validando que se cumplieran los lineamientos definidos desde la mesa de gobierno de APIs.

En el proceso de las prácticas el estudiante apoyó en la actualización del catálogo de APIs en EPM, donde apoyó en el ingreso de la información de algunas de las APIs de EPM en el catálogo de APIs.

V. RESULTADOS

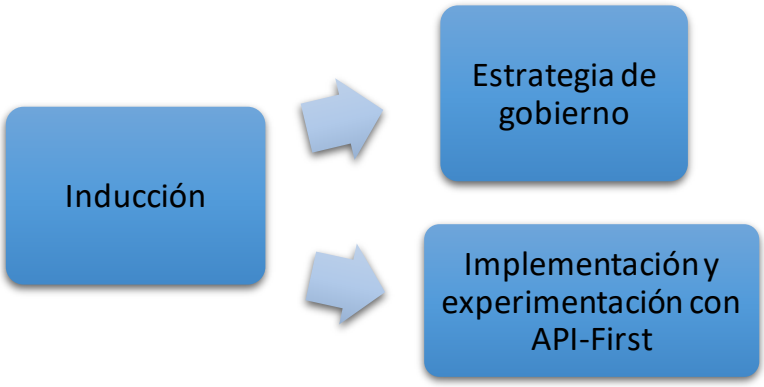

Fig. 1. Proceso de desarrollo de las prácticas. Fuente: El autor

Las prácticas académicas realizadas en EPM entre las fechas **agosto 30 de 2021** hasta **marzo 1 de 2022**, tenían como propósito principal implementar buenas prácticas en la estrategia de gobierno de APIs y, por lo tanto, mejorar su calidad. Según esto y basándose en la metodología anteriormente planteada se describen las actividades realizadas durante el proceso y los resultados obtenidos en cada una:

A. Estrategia de gobierno de APIs en EPM

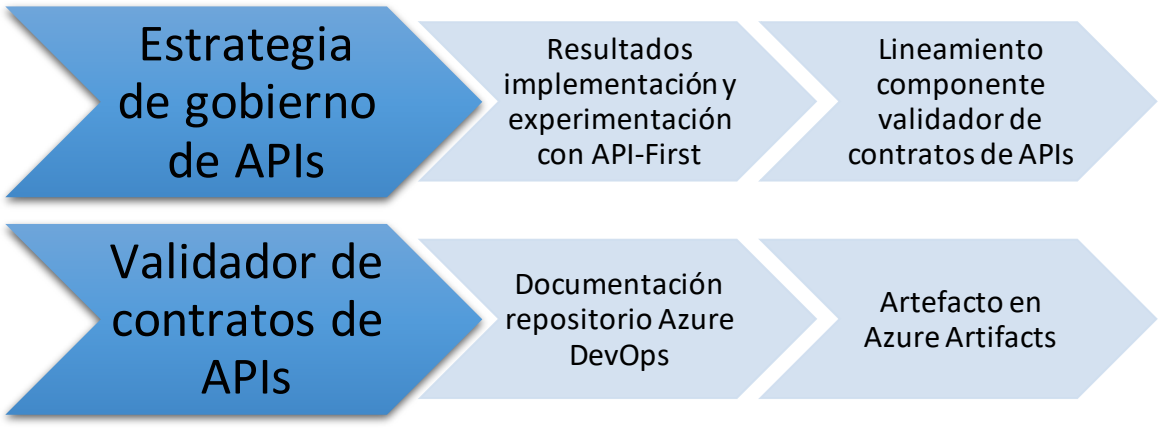

Fig. 2. Procesos para apoyo a la estrategia de gobierno de APIs. Fuente: El autor

Esta actividad se realizó mediante la participación en la mesa de gobierno de APIs donde el estudiante apoyó en la definición de la estrategia de gobierno de APIs, específicamente, en la

definición del lineamiento para el diseño de APIs llamado **Componente validador de contratos de APIs.**

Este "componente validador de contratos de APIs" se propuso como lineamiento de la mesa de gobierno de APIs según los resultados de **implementación y experimentación con API-First** en el proyecto P2P energía transactiva, donde se pudo reducir considerablemente el tiempo, complejidad y costo de desarrollo.

Para la implementación de este nuevo lineamiento en los proyectos que utilicen APIs alrededor de EPM, se creó un nuevo repositorio en la plataforma organizacional de EPM, Azure DevOps [9]. En este repositorio se hizo entrega de los archivos resultados de la implementación y experimentación con API-First (**Actividad B**), además de, archivos de documentación con extensión .md, en los cuales se explica:

1. *Propósito del repositorio*

El objetivo del repositorio es almacenar los documentos necesarios del validador de contratos para posteriormente publicarlos en un artefacto de Azure Artifacts [10].

2. *Propósito del validador*

El principal objetivo del validador es asegurar que los contratos de APIs cumplan con los lineamientos establecidos desde la mesa de gobierno de APIs.

3. *Propósito del artefacto en Azure Artifacts*

El objetivo del artefacto es empaquetar los documentos necesarios para el uso del validador de contratos. Azure Artifacts es la herramienta para distribuir el validador como componente alrededor de EPM.

4. Estructura del repositorio

Carpeta Docs

Almacena la documentación necesaria para la implementación del validador en un proyecto, por ejemplo, paso a paso para utilizar el validador. Esta carpeta y todo su contenido se empaqueta en el artefacto.

Carpeta DocumentacionSpectral

Almacena la documentación de la herramienta Spectral, la cual se utilizó para la implementación del validador.

Carpeta OpenAPI

Almacena ejemplos de contratos de APIs útiles para crear archivos guías en la carpeta docs.

Carpeta Pipelines

Almacena los pipelines que se utilizan para el proceso de validación de contratos. Esta carpeta y todo su contenido se empaqueta en el artefacto.

Carpeta ValidatorRules

Almacena los conjuntos de reglas que se utilizan para el proceso de validación de contratos. Esta carpeta y todo su contenido se empaqueta en el artefacto.

Posteriormente, para facilitar el uso y la distribución de este "componente validador de contratos de APIs" a cualquier proyecto que utilice APIs, se hizo uso de la herramienta Azure Artifacts, ubicado también en la plataforma Azure DevOps. Para publicar el nuevo artefacto que contiene los archivos necesarios para el uso de este componente, se hizo mediante un pipeline

nombrado como "A2V_Componente-Validador-CI", el cual contiene las siguientes tareas y configuraciones clave:

1. *Tarea copiar archivos*

Haciendo uso de la tarea Copy Files proporcionada por Azure Pipelines [11], el agente del pipeline copia los contenidos de las carpetas Docs, Pipelines y ValidatorRules ubicadas en el repositorio mencionado anteriormente en la ruta definida por la variable $(Build.ArtifactStagingDirectory) [12], la cual es una variable de entorno en Azure Pipelines.

2. *Tarea publicar en Azure Artifacts*

Haciendo uso de la tarea Universal Packages [13] proporcionada por Azure Pipelines, se publica el artefacto en el feed de nivel organizacional de EPM, org-epm-artefactos, con el nombre "gobiernoapis-validador" en su versión 1.0.0, empaquetando los archivos copiados de la tarea anterior.

B. Implementación y experimentación con API-First

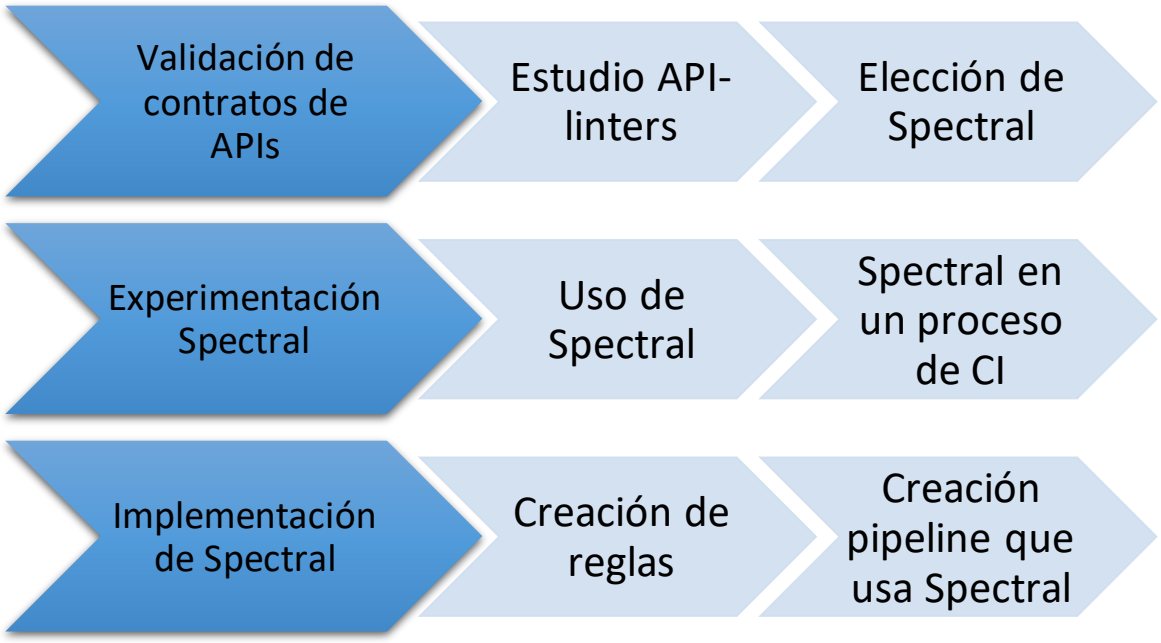

Fig. 3. Procesos para implementación de API-First. Fuente: El autor

Esta actividad se realizó mediante la participación en el proyecto P2P energía transactiva, donde el objetivo era, siguiendo el enfoque de desarrollo API-First, validar que los contratos de las APIs que se utilizaran en el proyecto cumplieran con los lineamientos de estandarización de APIs establecidos desde la mesa de gobierno de APIs.

Siguiendo que uno de los objetivos del enfoque API-First es reducir tiempo de desarrollo, se llega a la automatización de este proceso de validación, ya que puede ser repetitivo, lleva mucho tiempo hacerlo manualmente y tiene el riesgo de errores cometidos por humanos. Para lograr la automatización de este proceso de validación, se hizo utilizando las herramientas para este propósito conocidas como API-linter.

Debido a que en la web hay una gran cantidad de estas herramientas open-source, se hizo un estudio (**TABLA I**) de cuatro de estas herramientas para elegir la que más se acoplaba a la estrategia de EPM, este estudio se basó en las siguientes preguntas:

- ¿Qué tan fácil es aprender a usarlo?
- ¿Qué tan conocido es y su apoyo por la comunidad?
- ¿Qué tan constante su desarrollador lo actualiza?
- ¿Qué versiones de la especificación OpenAPI puede validar?
- ¿Qué tipos de archivos son compatibles para que valide?

TABLA I:
RESULTADO ESTUDIO API-LINTERS

	IBM OpenAPI Validator	openapi-lint	Spectral	oas-kit
Facilidad de uso	Fácil	Medio	Fácil	Medio
Comunidad	Parcialmente conocido	Parcialmente conocido	Conocido y creciente	Parcialmente conocido
Actualizaciones	Constantes	Última en 2020	Constantes	Última en julio de 2021
Versiones OpenAPI	OpenAPI v2 y v3	OpenAPI v3	OpenAPI v2 y v3; AsyncAPI v2	OpenAPI v2
Tipo de archivos	JSON/YAML	JSON/YAML	JSON/YAML	JSON/YAML

Fuente: El autor

Como resultado de este estudio, se concluye que Spectral es la herramienta que más se acopla con la estrategia de EPM, principalmente por su comunidad y desarrollador activos; y su capacidad para validar OpenAPI v2 y v3, y AsyncAPI v2.

Posteriormente a la elección de la herramienta a utilizar, se procede a investigar cómo utilizar la herramienta Spectral e implementarla en un proceso de CI utilizando pipelines en la plataforma organizacional de EPM, Azure DevOps, en el ambiente del proyecto P2P energía transactiva. Como resultado de la investigación, se hizo entrega de:

- Documentación sobre el uso de Spectral.
- Documentación sobre Spectral en un proceso de CI.
- Archivo con reglas a validar.
- Pipeline de validación en YAML.

1. *Documentación sobre el uso de Spectral*

Esta documentación sobre el uso de Spectral se entregó en formato .md, y este archivo contiene los conceptos básicos para utilizar la herramienta Spectral [14] en el proceso de validación de contratos de APIs, los cuales son:

Archivos OpenAPI y AsyncAPI

Los archivos OpenAPI y AsyncAPI en formato JSON o YAML, también conocidos como contratos de las APIs, son los archivos donde se muestra cómo es el comportamiento de una API Rest o una API asíncrona, según las especificaciones OpenAPI y AsyncAPI, respectivamente.

Ruleset: ¿qué es y de qué se compone?

Es un archivo YAML que actúa como contenedor de las reglas que se utilizarán para garantizar la calidad de las APIs. Con Spectral se pueden crear varios *rulesets* que contengan las

reglas para validar las diferentes partes de la API, y luego un *ruleset* principal que contenga las referencias a los demás.

Un *ruleset* está compuesto por propiedades para que Spectral valide las reglas que se creen en él. Estas propiedades clave son:

- **Extends:** Permite importar las reglas de otros *rulesets* para utilizarse en el proceso de validación de los contratos de APIs. Se pueden utilizar las reglas de diferentes *rulesets* al mismo tiempo.
- **Rules:** Es donde se crean todas las reglas personalizadas para utilizar en el proceso de validación de los contratos de APIs.

¿Cómo crear una regla?

```
rules:
    tags-description:
        description: All tags must have a description.
        message: "{{description}}"
        recommended: false
        severity: error
        given: $.tags[*]
        then:
            field: description
            function: truthy
```

Fig. 4. Ejemplo de una regla en Spectral. Fuente: El autor

Una regla personalizada en Spectral (**Fig. 1**) está compuesta por los siguientes campos:

- **Description:** Es donde se describe lo que se quiere validar, lleva relación directa con el lineamiento.

- **Message:** Es opcional y se utiliza para dar resultados de la validación más claros y precisos.

- **Given:** Contiene un JsonPath que apunta a una parte específica del objeto JSON/YAML que se quiere validar.

- **Severity:** Especifica la gravedad de la regla en el proceso de validación. Sus valores pueden ser error, warn, info o hint.

- **Recommended:** Es opcional y especifica si la regla se toma en cuenta o no en el proceso de validación. Sus valores pueden ser true o false.

- **Then:** Es el contenedor de las validaciones que se le aplicarán al objeto JSON/YAML durante el proceso de validación.

- **Field:** Es opcional y se refiere a una propiedad específica del objeto JSON/YAML a validar.

- **Function:** Se refiere a qué tipo de validación se aplicará a lo obtenido por Given y Field. Estas funciones vienen con Spectral y algunas de ellas tienen opciones para personalizar aún más su funcionamiento [15].

2. Documentación sobre Spectral en un proceso de CI

Esta documentación sobre el uso de Spectral en un proceso de CI [16] se entregó en formato .md, y este archivo contiene las configuraciones y tareas a ejecutar en un pipeline de Azure Pipelines, el cual hace uso de Spectral. Estos pasos son:

Configuración general del pipeline

Las siguientes configuraciones del pipeline indican los aspectos generales para el uso de un nuevo pipeline alrededor de EPM, tales como:

- **Agent pool:** Se ejecuta en Azure Pipelines, el cual es el agente establecido por EPM para la ejecución de pipelines.

- **Agent Specification:** Se ejecuta en windows-2019, el cual indica el sistema y la versión del sistema operativo a utilizar durante la ejecución del pipeline.

- **Recursos:** El pipeline debe apuntar al repositorio y a la rama donde están alojados los archivos OpenAPI, AsyncAPI y el *ruleset*.

Tareas para utilizar Spectral

Las siguientes tareas para la instalación de Spectral son generales para cualquier pipeline en el cual se requiera utilizar la herramienta Spectral:

- **Versión de NodeJs:** Para poder ejecutar la herramienta Spectral en un pipeline de Azure Pipelines se requiere que el agente que lo ejecuta tenga la versión 14.18.1 de NodeJs. Para hacerlo de una manera más sencilla se puede hacer uso de la utilidad que provee Azure Pipelines llamada "Node.js tool installer".

- **Instalar npm:** Otra de las tareas requeridas para la ejecución de Spectral es instalar npm en el agente que ejecuta el pipeline. Para esto se puede hacer uso de la utilidad que provee Azure Pipelines llamada "npm", seleccionando como comando la opción install.

- **Instalar Spectral:** Como tarea que todo pipeline debe de incluir para el uso de Spectral en un proceso de CI, se debe instalar la consola de Spectral en el agente que ejecuta el pipeline. Para esto se puede hacer uso de la utilidad que provee Azure Pipelines llamada "npm", seleccionando como comando la opción custom y en el campo que se habilita llamado "Command and arguments" se escribe el comando "install -g @stoplight/spectral-cli".

- **Ejecutar Spectral:** Como última tarea para utilizar Spectral en un proceso de CI, es la ejecución de la propia herramienta Spectral en el agente que ejecuta el pipeline. Para esto se puede hacer uso de la utilidad que provee Azure Pipelines llamada "Command Line" y en su campo llamado Script se escribe el comando "spectral lint \$(Build.SourcesDirectory)/[ruta archivos] -r $(Build.SourcesDirectory)/[ruta *ruleset*]" donde [ruta archivos] y [ruta *ruleset*] son las rutas dentro del repositorio donde están ubicados los archivos OpenAPI y las reglas a validar, respectivamente.

3. Archivo con reglas a validar

Para garantizar que las APIs que se creaban en el proyecto P2P energía transactiva cumplían con los lineamientos establecidos mediante el uso de la herramienta Spectral, se hizo un archivo en formato YAML que contiene las reglas a utilizar durante el proceso de validación, y siguiendo la Documentación sobre el uso de Spectral mencionada anteriormente.

En total, se crearon 9 reglas basadas en los lineamientos establecidos desde la mesa de gobierno de APIs, las cuales verificaban que las APIs dentro del proyecto P2P energía transactiva cumplieran con ciertos diseños, tales como: nombramiento, versionamiento, manejo de errores y requerir campos que la especificación OpenAPI no lo hace. Estas reglas son:

TABLA II:
LINEAMIENTO PARÁMETROS EN CAMEL-CASE

Nombre	camel-case-parameter-name
Description	Parameter names must be camel cased.
Message	Parameter '{{value}}' must be camel cased.
Given	$.paths[*][get,post,put,delete,patch].parameters[*]
Severity	error
Field	name
Function	casing
FunctionOptions	type: camel

Fuente: El autor

TABLA III:
LINEAMIENTO NO "/" AL FINAL

Nombre	path-not-end-with-slash
Description	Paths should not end with /.
Message	{{description}}
Given	$.paths[*]~
Severity	Warn
Field	-
Function	pattern
FunctionOptions	notMatch ".+\/$"

Fuente: El autor

TABLA IV:
LINEAMIENTO DESCRIPCIÓN DE OPERACIONES

Nombre	path-description
Description	Each operation must have a description which cannot be empty
Message	Path must have a description
Given	$.paths[*][get,post,put,delete,patch]
Severity	Warn
Field	Description
Function	Truthy
FunctionOptions	-

Fuente: El autor

TABLA V:
LINEAMIENTO OPERACIONES EN KEBAB-CASE

Nombre	kebab-case-path
Description	Paths must be kebab cased
Message	{{description}}
Given	$.paths[*]~
Severity	Error
Field	-
Function	Pattern
FunctionOptions	match: "^\/([a-z0-9]+(-[a-z0-9]+)*)?(\/[a-z0-9]+(-[a-z0-9]+)*|\/{.+})*$"

Fuente: El autor

TABLA VI:
LINEAMIENTO REQUERIR EJEMPLOS

Nombre	property-examples-in-responses
Description	All responses must have property examples
Message	{{description}}
Given	$.paths[*][*].responses[*].content[*]
Severity	Warn
Field	Examples
Function	Defined
FunctionOptions	-

Fuente: El autor

TABLA VII:
LINEAMIENTO REQUERIR ESQUEMA DE ERROR

Nombre	valid-error-response-schema
Description	Error response schema must be present
Message	{{description}}
Given	$.components.schemas
Severity	Warn
Field	ResponseApi
Function	Defined
FunctionOptions	-

Fuente: El autor

TABLA VIII:
LINEAMIENTO VERSION EN LA URI

Nombre	version-in-uri
Description	API version should be in the URI
Message	{{description}}
Given	$.paths[*]~
Severity	Warn
Field	-
Function	Pattern
FunctionOptions	match: "^\/.+\/v([1-9]+(\\.[1-9]+)*?)(\/.+)$"

Fuente: El autor

TABLA IX:
LINEAMIENTO PROPIEDADES EN ESQUEMA DE ERROR

Nombre	required-properties-error-response-shema
Description	Error response schema must contain code and message properties
Message	Missing '{{property}}' property
Given	$.components.schemas.ResponseApi.properties
Severity	Warn
Field	Code/Message
Function	Defined
FunctionOptions	-

Fuente: El autor

TABLA X:
LINEAMIENTO ESQUEMA DE ERROR

Nombre	error-responses-schema
Description	All error responses must follow 'Error response' schema
Message	{{description}}
Given	$.paths[*][*].responses[?(@property >= '400' && @property <= '500' && @property !== 'default')].content[*].examples[*]
Severity	Warn
Field	Value
Function	Schema
FunctionOptions	type: object required: - code - message properties: code: type: integer format: int32 message: type: string innerMessage: type: object properties: title: type: string titleTag:

	type: string detail: type: string

Fuente: El autor

4. Pipeline de validación en YAML

Como última actividad realizada en la Implementación y experimentación con API-First en el proyecto P2P energía transactiva y para facilitar el uso de Spectral en los proyectos que utilizan APIs alrededor de EPM, se creó un archivo en formato YAML que contiene la configuración de un pipeline de Azure Pipelines para ejecutar la herramienta Spectral en la plataforma Azure DevOps.

Este archivo se creó siguiendo los pasos mencionados anteriormente en la actividad "Documentación sobre Spectral en un proceso de CI" y el resultado se puede ver a continuación:

```
jobs:
  - job: Job_1
    displayName: Validation process
    pool:
      vmImage: windows-2019
    steps:
      - checkout: self
      clean: true
      - task: NodeTool@0
      displayName: NodeJs version
      inputs:
        versionSpec: 14.18.1
      - task: Npm@1
      displayName: Install npm
      inputs:
```

```
      verbose: false
    - task: Npm@1
    displayName: Install Spectral
    inputs:
      command: custom
      verbose: false
      customCommand: install -g @stoplight/spectral-cli
    - task: CmdLine@2
    displayName: Lint OpenAPI contracts with Spectral
    inputs:
        script: spectral lint $(Build.SourcesDirectory)/OpenAPI/*.{json,yml,yaml}
                -r $(Build.SourcesDirectory)/ValidatorRules/oas-rules.yml
```

Fuente: El autor

5. Resultado de la experimentación

Como resultados finales de la "Implementación y experimentación con API-First" en el proyecto P2P energía transactiva, utilizando el API-linter llamado Spectral y todos los archivos y la documentación mencionada anteriormente sobre este, se obtuvieron los siguientes resultados:

TABLA XI:
RESULTADO EXPERIMENTACIÓN API-FIRST

longitud contrato de api	Tiempo manual	Tiempo Spectral	% de reducción
corto	20-30 min	1 min	95-97%
medio	30-60 min	1 min	97-98%
largo	+1 horas	1 min	98-99.9%

Fuente: El autor

En la TABLA XI se observan los porcentajes de reducción de las métricas evaluadas. Se evalúan los tiempos que se toman en el proceso de validación utilizando la herramienta Spectral y haciéndolo manualmente.

Se puede observar que al hacer uso de la herramienta Spectral en el proceso de validación de contratos de APIs, hay una reducción entre el 95% y un 99.9% en el tiempo de validación, aumentando su porcentaje de reducción siendo directamente proporcional a la longitud del contrato de API que se quiere validar.

Otros beneficios implícitos del uso de la herramienta Spectral en el proceso de validación de contratos de APIs son:

- Reducción de errores humanos en el proceso de validación de contratos de APIs, ya que la validación se hace con las reglas mencionadas anteriormente en el "Archivo con reglas a validar" y no manualmente con un checklist que se pude llegar a extender mucho.

- Reducción de costos de desarrollo, ya que al automatizar este proceso de validación y obtener las reducciones de tiempos de validación en más de un 95%, los pagos a los desarrolladores se hacen por desarrollar y no por validar los contratos de APIs.

- Reducción en tiempo de entregas, ya que al acortar tanto el proceso de validación, las entregas finales de los contratos también se reducen, por lo tanto, se puede empezar a desarrollar más rápido.

Basándose en los porcentajes de reducción de tiempos de desarrollo de APIs utilizando la herramienta Spectral, se propuso su utilización como nuevo lineamiento en la mesa de gobierno de APIs, implementándolo con el nombre de "Componente validador de APIs" como se mencionó anteriormente en los resultados obtenidos en "Estrategia de gobierno de APIs en EPM".

C. Actualización del catálogo de APIs

Durante el desarrollo de esta actividad, se hicieron reuniones con una persona del equipo para apoyar en el ingreso de la información de las APIs de EPM en el catálogo de APIs, Alejandría APM. Se logró el ingreso de la información de algunas APIs como los nombres de la aplicación,

endpoints y flujos de información. También durante las reuniones con la persona del equipo se pudo notar que había información previa faltante y requerida para el ingreso de las APIs, por lo que no se ingresó toda la información completa.

VI. CONCLUSIONES

Se logró aportar a la estrategia de gobierno de APIs, la cual estaba en desarrollo en la mesa de gobierno de APIs. Esto se logró con la creación de lineamiento para el uso del "Componente validador de APIs" propuesto. Con este lineamiento se pretende que su uso sea tanto en proyectos nuevos como en proyectos ya iniciados, alrededor de EPM que utilicen APIs en sus ciclos de vida para así lograr la estandarización de estas.

Debido al uso de la herramienta Spectral para automatizar el proceso de validación de las APIs en el proyecto P2P energía transactiva, se logró obtener porcentajes entre el 95% y el 99.9% de reducción de tiempos de validación de contratos de APIs, dependiendo directamente de la longitud del contrato. La reducción de estos tiempos de validación también conlleva a la reducción de complejidad y costos de desarrollo dentro del ciclo de vida de las APIs, específicamente, en la fase de diseño de la API, ya que se puede desarrollar más rápidamente.

Gracias a la implementación y experimentación con el enfoque de desarrollo de APIs, API-First, se logró implementar buenas prácticas de gobierno de APIs alrededor de todos los proyectos en EPM que las utilicen, reduciendo los tiempos, complejidad y costos de desarrollo de las APIs en los diferentes equipos de trabajo de la empresa, promoviendo así la calidad y economía de las APIs.

REFERENCIAS

[1] M. Boyd, "Developing the API Mindset", Nordic APIs AB, 2015

[2] B. Doerrfeld, C. Wood, A. Anthony, and K. Sandoval, "The API Economy", Nordic APIs AB, 2016

[3] B. Doerrfeld, C. Wood, K. Sandoval, and O. Santolalla, "How to Successfully Market an API", Nordic APIs AB, 2016

[4] B. Doerrfeld, B. Pedro, K. Sandoval, and A. Krohn, "The API Lifecycle", Nordic APIs AB, 2015

[5] A. Blanco, "Gobierno de APIs", 2021. https://bit.ly/3vf3x3U

[6] SWAGGER, "OpenAPI Specification", 2021. https://bit.ly/3Hgr2vC

[7] B. Doerrfeld, "8+ OpenAPI Linters", 2019. https://bit.ly/3Hgrav6

[8] Stoplight, "Spectral, an Open Source JSON/YAML Linter", 2022. https://bit.ly/361oQLR

[9] Microsoft, "¿Qué es Azure DevOps?", 2022. https://bit.ly/3Cw8zdP

[10] Microsoft, "Información general sobre Azure Artifacts", 2022. https://bit.ly/365ZHPW

[11] Microsoft, "¿Qué es Azure Pipelines?", 2022. https://bit.ly/3KyFNfj

[12] Microsoft, "Use predefined variables", 2022. https://bit.ly/3J1sgws

[13] Microsoft, "Publicar y descargar Universal Packages en Azure Pipelines", 2022. https://bit.ly/3J2k2V3

[14] Stoplight, “Custom Rulesets”, [Citado: 8 mar. 2022]. https://bit.ly/3686SXQ

[15] Stoplight, “Core Functions”, [Citado: 8 mar. 2022]. https://bit.ly/3MFlTB8

[16] Stoplight, “Spectral CLI”, [Citado: 8 mar. 2022]. https://bit.ly/3vXp4OJ