Artículo de Investigación

Aibi revista de investigación, administración e ingeniería. Volumen 13, Número 3, Pág 01-11, Septiembre - Diciembre de 2025, ISSN 2346-030X

# Propuesta y aplicación de una guía técnica para adoptar API-FIRST.
# Proposal and application of a Technical Guide to adopt API-FIRST.

Juan Gabriel Enriquez[1], Héctor Reinaga[2], Sandra Isabel Casas[3]
[1,2,3]*Universidad Nacional de la Patagonia Austral, Río Gallegos - Argentina*

Recibido: 22 de febrero de 2025.
Aceptado: 14 de julio de 2025.
Publicado: 01 de septiembre de 2025.

***Resumen-*** Una innovación de la industria de las APIs web es el enfoque de desarrollo de software API-First. Según diversos autores en el ámbito académico, aún son escasos los estudios que lo han abordado. Sin embargo, los defensores de este modelo aseguran que genera múltiples beneficios para el desarrollo de software y las empresas. Para lograr un acercamiento al mismo, en este trabajo se propone una guía técnica para su aplicación, obtenida de su exploración y descripción exhaustiva. Así también, se evalúa y demuestra dicho instrumento en un problema real y local. La guía está fuertemente orientada al uso de tecnologías y herramientas específicas que impactaron en la duración y complejidad de la API desarrollada.

**Palabras clave**: API-First, spring boot, open API, guía técnica**.**

***Abstract—*** An innovation in the web API industry is the API-First software development approach. According to various academic authors, studies have still been limited. However, proponents of this model assert that it generates multiple benefits for software development and businesses. To gain a better understanding of this approach, this paper proposes a technical guide for its application, obtained from its exploration and exhaustive description. It also evaluates and demonstrates this tool in a real, local problem. The guide is strongly oriented toward the use of specific technologies and tools that impacted the duration and complexity of the developed API.

**Keywords**: API-First, spring boot, open API, technical guide.

*Autor para correspondencia.
Correo electrónico: sicasas@uarg.unpa.edu.ar (Sandra Casas).
La revisión por pares es responsabilidad de la Universidad de Santander.

Como citar este artículo: J. G. Enriquez, H. Reinaga y S. Casas, "Propuesta y aplicación de una guía técnica para adoptar API-FIRST", Aibi revista de investigación, administración e ingeniería, vol. 13, no. 3, pp. 01-11 2025, doi: 10.15649/2346030X.4164

# I. INTRODUCCIÓN

Las APIs (*Application Programming Interface*) web [1] constituyen uno de los servicios de intercambio de información y acceso a datos más comunes. Actualmente son el mecanismo establecido para la interacción entre aplicaciones, asegurando el intercambio de información y funcionalidades en los sistemas internos de una organización y con sistemas de terceros.

En el ámbito de las aplicaciones de software, las APIs web (REST, RESTful, RPC, SOAP, etc.) se utilizan habitualmente con distintos fines. Suele crearse una capa de API común para cumplir los requisitos empresariales de aplicaciones que se ejecutan en diferentes plataformas y front-ends (web, escritorio, móvil, etc.). Además, poner las APIs a disposición del público es la forma más usada de ampliar el impacto y el uso de aplicaciones populares. Los principales ejemplos de este enfoque son Facebook a través de su Facebook Graph API y la red social X (anteriormente Twitter) [2]. Las APIs web se han constituido en el pilar del software moderno y la transformación digital. La comercialización de activos por medio de las APIs o Economía API [3], se ha instalado en la industria como un objetivo central de negocio, lo que ha generado una innovación en el sector del software, hacia nuevas actividades y herramientas (administración, gobierno, monetarización, etc.).

Otra innovación que se ha impulsado refiere a la forma de desarrollar el software. En los últimos años en las comunidades de desarrollo de APIs web, se destaca la adopción de un enfoque de desarrollo de software denominado “API-First”. Los principios API-First sugieren que todas las capacidades de una organización y sus sistemas, se exponen a través de una API web, y que la base del diseño del sistema es la definición de APIs claras y bien definidas [4]. Según [5] lo que caracteriza al enfoque es que se prioriza las APIs al comienzo del proceso de desarrollo, posicionando las APIs como los componentes básicos del software, lo cual implica desarrollar las APIs antes de escribir cualquier otro código, en lugar de tratarlas como una ocurrencia tardía.

Es decir, este enfoque postula el diseño y el desarrollo de una API antes de la implementación. El diseño API-First es una innovación para el diseño de sistemas empresariales; en esta etapa inicial de la metodología, la investigación, documentación y herramientas que ayuden a adoptar un enfoque de diseño API-First son mínimos [6]. Los defensores de este modelo de desarrollo aseguran, que la adopción de API-First tiene varios beneficios, tales como, aumentar la rapidez de los proyectos de software, ya que se acelera el desarrollo y entrega, y reducir los fallos del software [5].

Dado que API-First se encuentra en su infancia, el interrogante que se plantea es, cómo adoptar API-First. En este trabajo, se propone una guía técnica que condensa en un conjunto de pasos, las actividades principales para adoptar API-First. Dicho procedimiento ha sido aplicado y validado con un problema real.

Este trabajo se estructura de la siguiente manera: la Sección 2 muestra el enfoque API-First, la Sección 3 describe el encuadre metodológico, la Sección 4 y 5 presenta los resultados, la Sección 6 expone las amenazas y limitaciones; y la Sección 7 presenta la discusión y los resultados.

# II. MARCO TEÓRICO

## *API-FIRST*

API-First es una metodología emergente, que está atrayendo la atención de los profesionales del software en la industria. La cantidad de desarrolladores, líderes y empresas que utilizan API-First crece rápidamente, de acuerdo a la última encuesta de la industria API de Postman [7]. Estos resultados sugieren que todas las capacidades comerciales de un sistema se expongan a través de una API. La metodología también sugiere que el diseño se centre en la definición de APIs claras y bien definidas como base [4].

El enfoque API-First implica realizar el diseño y la API antes de la implementación. El equipo de trabajo planifica la interfaz de la aplicación. A continuación, el equipo se basará en esa interfaz para construir el resto de la aplicación [8]. Según [9], pensar en API-First significa que una API es la primera interfaz de las aplicaciones. En consecuencia, los desarrolladores de las APIs son además usuarios de las mismas.

Según [10], existen varios beneficios al usar API-First que impactan en diferentes aspectos del desarrollo y el negocio:

- Reutilización: el enfoque tiene como objetivo cubrir todas las funciones necesarias con una llamada a la API, así se pueden utilizar estas mismas llamadas a la API en un desarrollo web o móvil y también reutilizarlos en otras aplicaciones,
- Documentación: API-First comienza escribiendo la especificación de la API en un formato estandarizado, esta definición crea la documentación correspondiente en el proceso,
- Consistencia: la estructura y documentación de las APIs siguen las mismas pautas,
- Acelera el desarrollo y la entrega: una API bien diseñada desde el principio es más fácil de reutilizar en diferentes partes de una aplicación o incluso en diferentes aplicaciones, promoviendo la escalabilidad y la eficiencia en el desarrollo de software,
- Pruebas: los desarrolladores pueden realizar pruebas y ajustes en el diseño antes de la implementación de la interfaz de usuario o en otros componentes del sistema. Esto permite una iteración más temprana y una reducción de costos en caso de cambios necesarios. Se pueden realizar pruebas automatizadas de manera más efectiva y temprana en el proceso de desarrollo, lo que contribuye a la calidad del producto final,
- Colaboración: permite obtener comentarios y opiniones sobre la API en funcionamiento tempranamente. La definición de una API puede y debe ser compartida con las partes interesadas desde el principio para hacer que el diseño sea lo mejor posible.

Según [7] los adoptantes de API-FIRST producen APIs más rápido y encuentran menos fallas que el promedio. Y cuando una API falla, pueden restaurarla más rápido, generalmente en menos de una hora.

### *a. API-FIRST en la Industria*

Existen guías y directrices para adoptar o aplicar de API-First, provenientes principalmente de la literatura gris o empresas importantes que marcan tendencia en la industria. A continuación, se mencionan algunas de ellas.

Postman [5] indica que para que una organización adopte un modelo de desarrollo API-First, debe priorizar las API, reconocer el papel de las API públicas, privadas y de socios en las organizaciones, y comprender el ciclo de vida de las API y las herramientas necesarias para convertirse en API-First. Se enuncia un ciclo de vida API de ocho fases (definir, diseñar, desarrollar, testear, seguridad, deploy, observar y distribuir). Se hace hincapié en que las plataformas API como sistemas de software (con herramientas y procesos integrados que permiten a los productores y consumidores crear, gestionar, publicar y consumir APIs) son un facilitador clave de API-First, e identifica los componentes clave: herramientas para el ciclo de vida de API, capacidades de colaboración para productores y consumidores, capacidades de gobernanza, e integraciones con el ciclo de vida de desarrollo de software. También presenta una guía de cinco pasos para que una organización se inicie en API-First.

Swagger [11] recomienda varios lineamientos para la planificación e implementación de un programa de API-First: identificación (servicios clave, tipos de API, servicios que ofrecerán mediante APIs , casos de uso de cada API, posibles puntos finales); establecer Stakeholders (la mayor cantidad de personas posible debe participar en su iniciativa API); diseñar contratos de APIs (describir y documentar todas las API siguiendo estándares y prácticas previamente establecidas); crear una guía de estilo (guía de estilo integral, coherente y estandarizada, para códigos de estado, control de versiones, manejo de errores, etc.); implementar la gobernanza de la APIs; automatizar procesos (utilizar herramientas como SwaggerHub para automatizar la generación de documentación de API, validación de estilos, simulación de API y control de versiones); seguimiento y administración de la cartera de APIs; implementar un sistema de rastreo y administrar las APIs; y crear un portal para desarrolladores internos.

La comunidad [12] ofrece implementar API-First con las APITools, se trata de un conjunto de cuatro herramientas open source (OpenAPI2postman, doSonarAPI, APIGEN Java o .Net y OpenAPI2soapui) que, a partir de la especificación OpenAPI, permiten generar todas las pruebas y validaciones de formato necesarias para cada entorno; validar la definición de las APIs en base al estándar corporativo, generar arquetipos en Spring Boot y generar pruebas con SoapUI.

Desde [13] [14] se aboga por un ciclo de vida API-First (creación de la API, diseño, prueba documentación, monitoreo, compartir, colaboración, etc.), basado en API-First Design, usando la herramienta Postman para diseñar y construir las APIs, y los formatos de especificación OpenAPI, y JSON Schema. El autor del reconocido blog considera que es un proceso iterativo en el que deben participar todos los interesados.

Las seis fases de API-First aportados por la fundadora de NordicAPI, recuperados por [10] indican los siguientes pasos:

- Planificar: debe proporcionar respuestas a las preguntas de por qué y quién.
- Diseñar y validar: cubre el diseño de la API de acuerdo con la planificación. Se recomienda una herramienta que permita probar las solicitudes de API antes de cualquier implementación).
- Bloquear la especificación API: se crea una especificación de API de acuerdo con el diseño y se debe bloquear para que pueda implementarse y mostrarse como la API final para todos los involucrados en el desarrollo, incluidas las partes interesadas y los desarrolladores.
- Probar: las funcionalidades, coherencia y experiencia del desarrollador deben ser objeto de las pruebas).
- Implementar: Codificar la API para que sea accesible para que las partes interesadas la prueben.
- Operar y participar: se implementa la API y los clientes deberían poder dar su opinión al respecto.

### *b. API-FIRST en la Investigación*

Algunos artículos y tesis se han planteado en torno al API-First, a continuación, destacamos algunos de ellos.

En [4] se aborda la práctica del diseño de API desde una perspectiva tanto académica como industrial. El trabajo se centra en examinar la adopción y las tendencias del diseño API-First en estos dos ámbitos. En el ámbito académico se mencionan algunos artículos que tratan este enfoque, pero generalmente lo consideran como un tema secundario, aún es un tema incipiente en la investigación académica. En resumen, se destaca que el diseño API-First es un tema emergente en este ámbito y se señala la necesidad de más investigación en este campo para cerrar la brecha entre la academia y la industria. A diferencia de la limitada investigación sobre el diseño API-First en entornos académicos, la industria ha generado una gran cantidad de contenido en forma de white papers, blogs, revistas y videos que discuten este enfoque. En el ámbito industrial se destaca la importancia y beneficios del enfoque, desde Amazon hasta empresas más pequeñas, y su potencial impacto en diversos sectores.

En [15] se describe una metodología utilizada para diseñar una plataforma Backend as a Service (BaaS) basada en microservicios, siguiendo un enfoque API-First. En resumen, la metodología comienza con la selección de servicios centrales que se proporcionarán en la plataforma BaaS, diseño detallado de la API, implementación de microservicios y la integración de estos microservicios para permitir su funcionamiento conjunto en la plataforma BaaS. Se destaca que la plataforma BaaS propuesta simplifica y acelera considerablemente el proceso de desarrollo web, especialmente en lo que respecta a la parte del servidor. También señala que los resultados obtenidos dependen en gran medida del contexto del proceso de desarrollo web, incluyendo el dominio de la aplicación, las funcionalidades, los frameworks, lenguajes y otras herramientas de desarrollo.

El objetivo principal de [10] es comprender qué es el diseño API-First, sus beneficios y cómo aplicar el método API-First Design en el entorno de Azure API Management, utilizando herramientas de terceros para hacer que el proceso sea eficiente. La investigación se centra en determinar hasta qué punto se puede acelerar el proceso de construcción de API mediante el uso de herramientas adecuadas. El estudio evaluó cuatro herramientas en total, dos de las cuales no fueron eficaces para un proceso de API-First Design efectivo, cuando se combinaron con Azure API Management. El trabajo concluye que API-First Design es una metodología prometedora y que las herramientas adecuadas pueden mejorar significativamente el proceso de desarrollo de API. También señala que el impacto de API-First Design en el trabajo en equipo, es un tema interesante para futuras investigaciones, así como su implementación con productos ya existentes.

En [16] se describe cómo implementar prácticas de gobierno de APIs en EPM (empresa de servicios públicos con altos estándares de calidad en energía eléctrica, gas, agua y saneamiento). Utilizan la gestión de APIs para la interacción entre aplicaciones (internas y externas) utilizando el enfoque API-First con el fin de reducir los tiempos, la complejidad, y los costos en el desarrollo de APIs en los equipos de trabajo de la

empresa. En particular, como validar contratos de APIs de alta calidad. Concluyen que, al utilizar la herramienta Spectral para automatizar la validación de contratos de APIs en el proyecto P2P energía transactiva, se logró reducir significativamente los tiempos de validación (entre el 95% y el 99.9%, dependiendo de la longitud del contrato). Esto condujo a la reducción de tiempos, complejidad y costos de desarrollo, especialmente en la fase de diseño de la API.

En [17] se investiga el valor del enfoque "API-First" al desarrollar APIs, comparándolo con el enfoque "Code-First". Se desarrollaron dos APIs, una utilizando el enfoque API-First y la otra el enfoque Code-First. Se utilizaron entrevistas semiestructuradas para recopilar datos y se destacaron las diferencias en eficiencia de prueba, experiencia del usuario y corrección de la API. La conclusión señala que el enfoque API-First mejora la calidad de la experiencia del usuario en términos de documentación, eficiencia de prueba y precisión en los requisitos comerciales. Se enfatiza que el enfoque API-First facilita la corrección de errores y la comunicación continua, mientras que el enfoque Code-First enfrenta desafíos en este sentido. Además, API-First contribuye a una documentación más completa, lo que facilita el uso de la API por terceros a corto y largo plazo. Finalmente se sugiere que este enfoque es adecuado para proyectos grandes y complejos debido al diálogo continuo entre desarrolladores y clientes.

## III. METODOLOGÍA O PROCEDIMIENTOS

Este estudio adopta el enfoque Design Science Research (DSR) [18], ampliamente utilizado en investigaciones de ingeniería, informática y sistemas de información para la producción de artefactos que resuelvan problemas específicos. DSR se caracteriza por su naturaleza prescriptiva y su enfoque en la creación de soluciones innovadoras mediante el desarrollo sistemático de construcciones, modelos, métodos o instancias.

### *a. Aplicación del Proceso DSR*

La investigación siguió los pasos metodológicos establecidos por Peffers et al. [18], como se ilustra en la Figura 1. El proceso se estructuró en cuatro fases principales:

- Identificación del problema y motivación: Se realizó un análisis exhaustivo de antecedentes académicos y en la industria, para determinar la problemática relacionada con la adopción del enfoque API-First en contextos organizacionales, la cual se halla resumida en la primera sección de este documento.
- Definición de objetivos: Se establecieron los objetivos específicos del estudio, orientados hacia la creación de una solución práctica y aplicable.
- Diseño y desarrollo del artefacto: Se diseñó y desarrolló una guía técnica para la adopción de API-First, tomando como referencia las mejores prácticas identificadas en Lin [8] y Larsson & Åkermark [17]. La guía constituye el artefacto central de esta investigación DSR.
- Demostración y evaluación: Se implementó un proceso de validación mediante Technical Action Research (TAR) para evaluar la efectividad y utilidad del artefacto en condiciones reales de práctica.

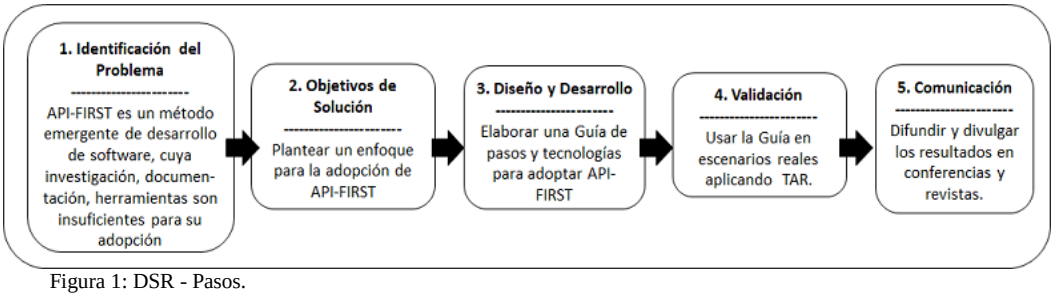



Figura 1: DSR - Pasos.
Fuente: Elaboración propia.

### *b. Validación mediante Technical Action Research*

Para la validación del artefacto se aplicó Technical Action Research (TAR) [19], un enfoque específicamente diseñado para validar nuevos artefactos en condiciones de práctica real. TAR permite responder preguntas fundamentales sobre la efectividad y utilidad del artefacto desarrollado en contextos aplicados, proporcionando evidencia empírica de su valor práctico.

La validación se estructuró mediante un ciclo iterativo que integra acción y reflexión sistemática. Cada iteración del ciclo TAR corresponde a la aplicación de un paso específico de la guía técnica desarrollada, permitiendo una evaluación granular de cada componente del artefacto. La Figura 2 esquematiza este proceso cíclico y define los roles específicos de los investigadores durante la validación.

El diseño metodológico contempla escenarios de aplicación reales, donde cada iteración se configura como la implementación práctica de un paso de la guía técnica. La observación sistemática de estas aplicaciones, se lleva a cabo a partir de las respuestas a preguntas de investigación predefinidas, lo que generó los datos necesarios para evaluar la efectividad del artefacto en condiciones operacionales.

| Iteración | GUÍA |
|---|---|
| 1 | Definición de requisitos de la API |
| 2 | Selección de tecnologías |
| 3 | Especificación |
| 4 | Generación código API |

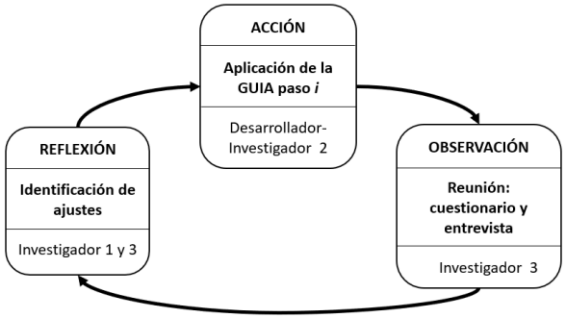



Figura 2: TAR - Ciclo Iterativo.
Fuente: Elaboración propia.

### *c. Integración DSR-TAR*

La combinación de DSR y TAR proporciona un marco metodológico robusto que permite tanto el desarrollo sistemático del artefacto como su validación empírica rigurosa. DSR aporta la estructura para el diseño y desarrollo de la guía técnica, mientras que TAR facilita su evaluación en contextos reales de aplicación, asegurando que el artefacto desarrollado sea tanto teóricamente sólido como prácticamente viable.

Esta integración metodológica responde a la necesidad identificada en la literatura de cerrar la brecha entre el desarrollo teórico de artefactos y su aplicabilidad práctica, proporcionando evidencia empírica del valor y utilidad de la guía técnica desarrollada para la adopción del enfoque API-First.

## IV. RESULTADOS, ANÁLISIS E INTERPRETACIÓN

### *Guía técnica para la adopción de Api-First*

La Figura 3, presenta los cuatro pasos de la guía y las actividades que en cada uno de estos se requiere hacer. El paso "Definición de requisitos de la API", es específico de la API que se requiera desarrollar, al igual que el paso "Especificación" y "Generación código API", por ello existe una secuencialidad entre las mismas. En cuanto al paso "Selección de tecnologías", refiere a un conjunto instrumental de herramientas que se utilizarán, debe realizarse siempre antes de las actividades de especificación. Pudiendo ser, el primero o segundo paso a cumplir. Para futuros desarrollos, este paso no sería necesario, si el equipo decide emplear las mismas tecnologías.

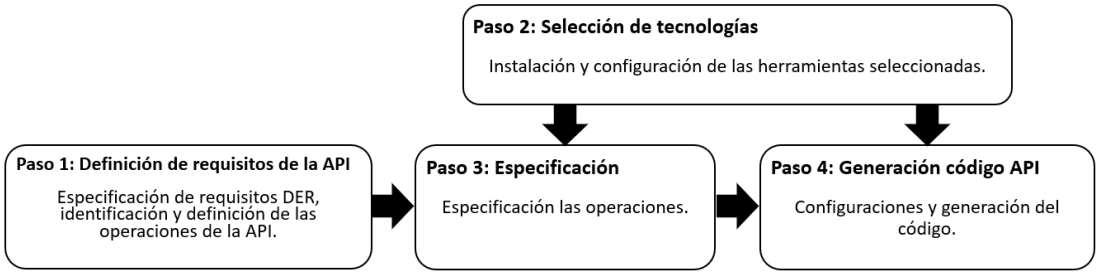



Figura 3: Pasos y Actividades de la Guía Técnica.
Fuente: Elaboración propia.

Paso 1: Definición de requisitos de la API

Se comienza identificando y documentando claramente los requisitos funcionales y no funcionales de la API. Dado que es fundamental comprender qué propósito cumplirá la API, qué recursos ofrecerá y qué acciones permitirá realizar a los clientes. A partir de la especificación de requisitos de la aplicación (casos de uso, historias de usuario u otra técnica) las actividades que deben realizarse son:

1) Especificación de requisitos. Definir un diagrama entidad-relación (y atributos) relacionados a la API.
2) Identificación y definición de las operaciones de la API, cada una de estas operaciones se transformará en un endpoint de la API. Por cada operación se define una descripción clara y sencilla de lo que hace cada operación y como se accederá a la misma. Esta descripción incluye: operación (debe refleja su función o propósito), método HTTP (GET/POST/PUT, etc.), descripción (breve descripción de lo que hace la operación), entradas (los parámetros, se enumera y describe las entradas que se deben proporcionar a la operación y se indica si son obligatorios u opcionales), salidas (será la respuesta, se describe el contenido y el formato JSON u otro), excepciones (se indican las condiciones que producen errores en la operación), consideraciones de seguridad (se menciona las consideraciones de seguridad o autenticación que se deben tener en cuenta al invocar a la operación), notas adicionales (se agrega cualquier información adicional que sea relevante para comprender y utilizar la operación).

Paso 2: Selección de tecnologías

Este paso está fuertemente centrado en las tecnologías y herramientas que se van a utilizar, ya que permitirán la automatización de ciertas actividades del proceso, hasta concluir con la generación del código de la API. Las herramientas deben dar soporte a diversas tareas y funciones, tales como edición de la especificación, documentación de la API, pruebas, implementación, etc.

Paso 3: Especificación

A partir de la documentación generada en el paso “Definición de requisitos de la API”, se especifica la interface de la API en algún formato estandarizado. Existen varios formatos como RAML, API Blueprint, OpenAPI, siendo quizás esta última las más conocida y usada. La especificación debe cubrir los requisitos, las restricciones, las características y las funcionalidades que debe cumplir la API, y ser independiente del lenguaje de programación. La especificación constituye el contrato de la API.

Paso 4: Generación código de la API

A partir de un documento de especificación estructurado definido en los pasos previos y las herramientas seleccionadas se implementa la API. Este paso suele ser semi-automático, ya que podrá obtenerse una implementación prototipo o esqueleto de la API e incluso el modelo de datos, pero manualmente el desarrollador deberá completar la codificación de la lógica específica de las operaciones, y realizar configuraciones y asociaciones más concretas.

### *Aplicación y experiencia en contexto*

La guía fue validada a partir de su aplicación en contexto, para ello, desde una empresa local planteó un problema real que debía ser resuelto con la guía presentada en el apartado anterior. Se formularon las siguientes preguntas, para validar si la guía técnica propuesta facilita la adopción de API-First: P1: ¿Qué actividad presentó obstáculos?; P2: ¿Qué tareas o actividades extras debieron ser realizadas?; P3: ¿Cuál fue la duración del paso?; y P4: ¿Cuál fue la complejidad (alta, media, baja) de la tarea?

Paso 1: Definición de requisitos de la API

Se consideró la construcción de una API a partir de distintos escenarios [20] relacionados con la gestión de Fondos de Operaciones y Mantenimiento (FOM) de la empresa Servicios Públicos Sociedad del Estado (https://spse.ar/). El FOM es una cantidad de dinero que se deposita en la cuenta bancaria de una gerencia o distrito de la empresa que podrá usar para compras directas. Posteriormente, estas gerencias y distritos deben rendir el FOM, es decir, presentar todas las facturas que totalizan el dinero otorgado. El FOM se establece mediante una resolución de directorio de la empresa y define por las distintas gerencias y distritos de la empresa el importe total del FOM y el importe máximo que cada factura del FOM podrá tener. El diagrama entidad-relación inicial se presenta en la Figura 4.

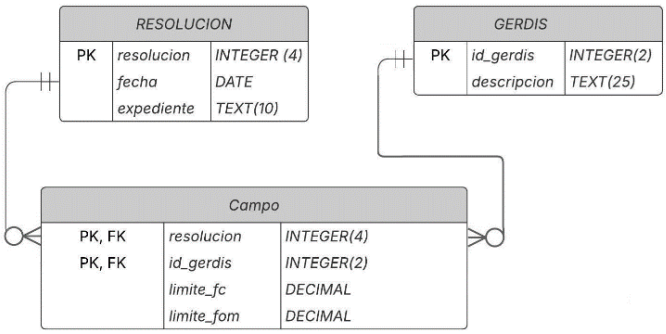



Figura 4: Diagrama entidad-relación.
Fuente: Elaboración propia.

En la Figura 5, se enumeran las operaciones identificadas para la API, y a continuación, la definición de las mismas, el esquema establece los elementos necesarios para cada una de ellas.

| Operaciones |
|---|
| **Crear un FOM –** |
| **Recuperar una Resolución** |
| **Recuperar los FOM de una Gerencia o Distrito** |
| **Recuperar las Resoluciones de un año** |

| | |
|---|---|
| Nombre | Crear FOM |
| Método | POST |
| Descripción | Agregar un nuevo FOM en la BD |
| Entradas | Resolución (obligatorio), fecha (obligatorio), expediente (obligatorio), limite-fom (obligatorio), {id-gerdis (obligatorio), limite_fc(obligatorio), limite-fom (obligatorio} |
| Salidas | OK o Error |
| Excepciones (error) | resolución ya existe,<br>fecha invalida,<br>id_gerdis no existe,<br>limite_fc>limite-fom,<br>(limite_fc \|\| limite_fom) <=0 |
| Seguridad | Solo usuarios del sistema FOM habilitados |
| Notas Adicionales | -- |

| | |
|---|---|
| Nombre | Recuperar una Resolución |
| Método | GET |
| Descripción | Recuperar los datos de una Resolución de la BD |
| Entradas | Resolución (obligatorio) |
| Salidas | fecha, expediente, limite-fom, {id-gerdis (obligatorio), limite_fc, limite-fom}<br>o Error |
| Excepciones (error) | resolución no existe |
| Seguridad | Solo usuarios del sistema FOM habilitados |
| Notas Adicionales | -- |

| | |
|---|---|
| Nombre | Recuperar los FOM de una Gerencia o Distrito |
| Método | GET |
| Descripción | Recuperar los FOM de una Gerencia o Distrito de la BD |
| Entradas | id-gerdis (obligatorio) |
| Salidas | {fecha, expediente, resolución, limite_fc, limite-fom}<br>o Error |
| Excepciones (error) | id-gerdis no existe |
| Seguridad | Solo usuarios del sistema FOM habilitados |
| Notas Adicionales | -- |

| | |
|---|---|
| Nombre | Recuperar las Resoluciones de un año |
| Método | GET |
| Descripción | Recuperar las Resoluciones de un determinado año de la BD |
| Entradas | año (obligatorio) |
| Salidas | {fecha, expediente, resolución, limite_fom}<br>o Error |
| Excepciones (error) | Año invalido |
| Seguridad | Solo usuarios del sistema FOM habilitados |
| Notas Adicionales | -- |

Figura 5: Identificación y Definición de Operaciones.
Fuente: Elaboración propia.


Aibi revista de investigación, administración e ingeniería. Volumen 13, Número 3, Septiembre - Diciembre de 2025, ISSN 2346-030X PP 01-11.

Paso 2: Selección de tecnologías

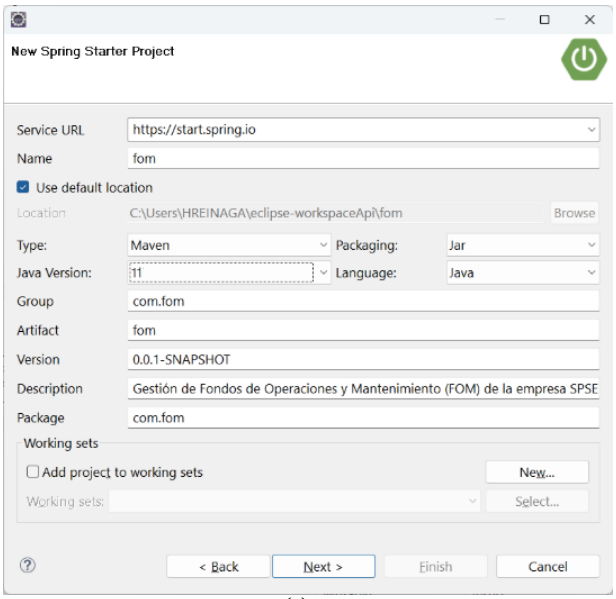
En la Tabla 1 se listan las herramientas elegidas para el caso de estudio y la función o uso que se le dará en el mismo. Diversos motivos llevaron a la selección de estas herramientas, son libres y gratuitas, y existe suficiente documentación accesible en la red. Asimismo, facilitan el desarrollo y despliegue de aplicaciones y además son tecnologías maduras, ampliamente utilizadas en el desarrollo de software empresarial en Java.

Tabla 1: Herramientas utilizadas.

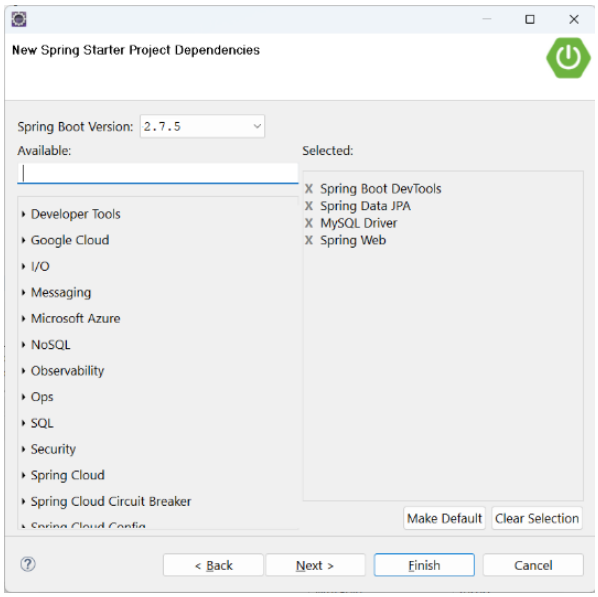
| **Herramienta** | **Función / Usos** |
|---|---|
| OpenAPI | Estándar para la descripción de una API. |
| Swagger Editor | Desarrollo de la especificación OpenAPI. |
| Eclipse IDE para Java EE Developers | Versión de Eclipse para desarrollar aplicaciones web Java EE. |
| Spring Boot DevTools | Permite recargar la aplicación cada vez que se hace una modificación al proyecto. |
| Spring Data JPA SQL | Permite leer los datos de la base de datos usando objetos Java. |
| MySQL driver | Servicio de conexión al gestor de base de datos MySQL |
| Spring Web | Dependencia para crear aplicaciones web, incluye un servidor embebido. |
| Spring Boot 2.7.5 | Facilita la creación de aplicaciones independientes Spring. |
| OpenAPI (Swagger) Editor 1.0.38 | Complemento que proporciona soporte para crear y editar archivos OpenAPI. |
| Spring Tools 4 | Complemento de Eclipse para desarrollar aplicaciones en Spring-Boot. |
| Apache Maven | Herramienta de gestión y comprensión de proyectos de software, basado en el concepto de modelo de objetos de proyecto (POM) |
| Spring Boot Maven Plugin | Proporciona soporte para Spring Boot en Apache Maven. Se añaden las dependencias Springfox, SpringBoot Starter Validation, OpenAPI tools, y OpenAPI Generator, editando el archivo pom.xml del proyecto. |
| Springfox | Librería que genera la documentación de una API. |
| SpringBoot Starter Validation | Implementa la lógica de validación en un ecosistema Java. |
| OpenAPI tools | Herramientas para el desarrollo de interfaz OpenAPI. |
| OpenAPI Generator | Herramienta que genera el código a partir de la especificación OpenAPI. |

Fuente: Elaboración propia.

Dado el conjunto de herramientas seleccionadas, las acciones se dividen en tres sub-pasos; preparar el entorno de trabajo, crear el proyecto y agregar dependencias. Se instaló la versión de Eclipse 2022, y se instalaron los complementos “OpenAPI (Swagger) Editor 1.0.38” y "Spring Tools 4". Luego, desde Eclipse se creó un proyecto “Spring Starter Project” denominado FOM (Figura 6.a), y se seleccionó la versión de “Spring Boot 2.7.5”, con las cuatro dependencias correspondientes (Figura 6.b). Una vez creado el proyecto, se editó el archivo pom.xml, y se agregaron las dependencias “Springfox” y “SpringBoot Starter Validation” (Figura 6.c).




(a)




(b)

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.openapitools</groupId>
            <artifactId>jackson-databind-nullable</artifactId>
            <version>0.2.1</version>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.9.2</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
    </dependencies>
```

(c)

Figura 6: Instalación y configuración de tecnologías.
Fuente: Elaboración propia.

Paso 3: Especificación

La especificación de la API se realizó en un archivo en formato YAML, se definieron las secciones (*info, servers, paths, y components*). En la sección *Paths*, se definieron los *endpoints* (resoluciones, gerencias, y resolucionesFOM) y se implementan las operaciones para crear un FOM, recuperar una Resolución, recuperar los FOM de una Gerencia o Distrito, entre otras; y así también, los distintos tipos de métodos HTTP, con las respuestas de retorno por cada método a través de un código de estado HTTP. Finalmente, la sección components, se describen los modelos de la implementación, más precisamente en la subsección schemas, en la que se definen las entidades Resoluciones, ResoFOM y GerDis.

Para mayor descripción, en la Figura 7 se presenta la definición de las secciones *info* y *servers*; y en la Figura 8, las secciones *paths*, y *components*.

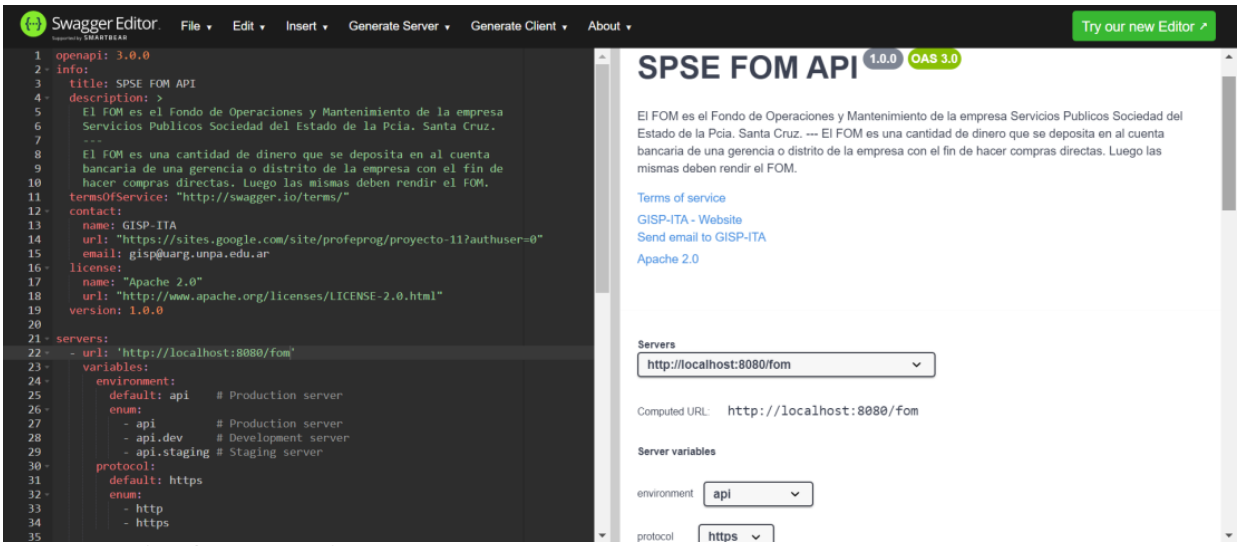



Figura 7: Especificación OpenAPI (*info* y *servers*).
Fuente: Elaboración propia.

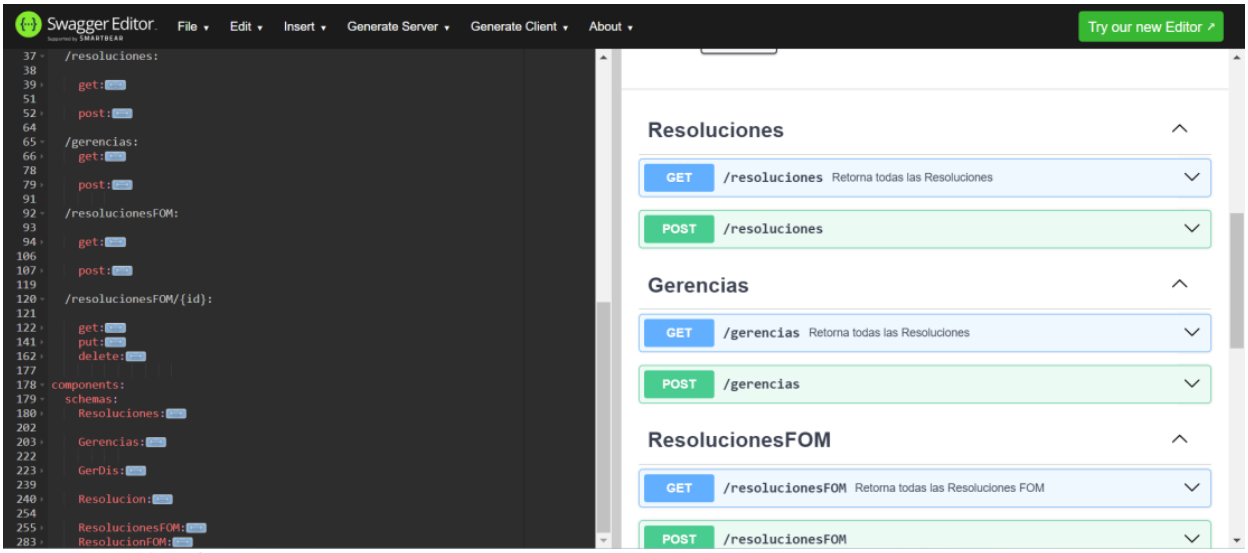



Figura 8: Especificación OpenAPI (*paths* y *components*).
Fuente: Elaboración propia.

Paso 4: Generación código de la API

Dado el conjunto de herramientas seleccionadas, las actividades y el orden de las mismas, se configuraron tres sub-pasos:

a. Configuración plugin Openapi-Generator (facilita el paso de la especificación OpenAPI a la implementación para que los desarrolladores 'solo' se preocupen de implementar la lógica de negocio. El plugin puede generar tanto la parte cliente como la parte servidor y todos los objetos especificados en el contrato).
b. Generación del código (para generar el código relacionado a la API y al modelo de datos, hay que construir el proyecto mediante la *función Build Project*).
c. Configuración de persistencia (con base a la sección <schema> de la especificación OpenAPI se generan las clases del modelo de datos. En la definición de las clases se escribe la configuración con anotaciones. Para persistir las tablas en la base de datos se debe agregar la etiqueta @Entity a las clases generadas del modelo).

En la Figura 9.a se refleja el resultado del sub-paso a), dado que antes de generar el código API, en el archivo pom.xml se agregó el plugin Openapi-generator en el apartado <build plugin>, y en la etiqueta <inputSpec> se configuró el *path* donde se encuentra la especificación

OpenAPI. Así también, en las etiquetas <apiPackage> y <modelPackage>, se ingresaron los paquetes donde se generan el código controlador y el modelo clases/objetos respectivamente.

A continuación, construyó el proyecto a través de la función Build Project de Eclipse, lo cual generó el código relacionado a la API y el modelo de datos. Luego, se procedió a comentar en el archivo pom.xml, el plugin *OpenAPI Generator* para que no vuelva a crear el código generado, lo que corresponde al sub-paso b).

Finalmente, y como se muestra en la Figura 9.b, para configurar la persistencia de clases del modelo en la base de datos, primero se creó la base de datos FOM desde MySQL. Posteriormente, en Eclipse se configuró el archivo application.properties del proyecto, ingresando el nombre de la base de datos, información relacionada a la conexión y el driver; y para generar las tablas automáticamente se definió la propiedad spring.jpa.hibernate.ddl-auto=update.

Para persistir las tablas en la base de datos, se agregó la etiqueta @Entity a las clases Resolucion, GerDis, y ResoFOM (Figura 9.c). Una vez creada, se comentó la propiedad hibernate para evitar que se vuelvan a crear las tablas.

```
<configuration>
    <inputSpec>
        ${project.basedir}/src/main/resources/api.yml</inputSpec>
    <generatorName>spring</generatorName>
    <apiPackage>openapi.api</apiPackage>
    <modelPackage>openapi.model</modelPackage>
</configuration>
```

(a)

```
server.servlet.contextPath=/fom
server.port=8080
spring.datasource.url=jdbc:mysql://localhost:3306/fom
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=update
logging.level.org.apache.tomcat=DEBUG
org.apache.catalina.session.level=ALL
java.util.logging.ConsoleHandler.level=ALL
```

(b)

```
package openapi.model;

import java.util.Objects;
import com.fasterxml.jackson.annotation.*;
import com.fasterxml.jackson.annotation.JsonProperty.Access;
import io.swagger.annotations.ApiModelProperty;
import javax.persistence.*;

@Entity
public class ResoFom   {
  @JsonProperty("id")
  @Id
  @GeneratedValue(strategy=GenerationType.AUTO)
  private Integer id;

  @JsonProperty("limite_fc")
  private Double limiteFc;

  @JsonProperty("limite_fom")
  private Double limiteFom;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "gerdis_id")
    @JsonProperty(access = Access.WRITE_ONLY)
    private GerDis gerdis;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "resolucion_id")
    @JsonProperty(access = Access.WRITE_ONLY)
    private Resolucion resolucion;
```

(c)

Figura 9: Generación de código.
Fuente: Elaboración propia.

Por último, desde Eclipse se ejecutó la opción Spring Boot App para generar las tablas de la base de datos. En la Tabla 2, se resumen las respuestas a las preguntas formuladas en cada iteración del proceso TAR ejecutado que se corresponde con un paso de la Guía.

Tabla 2: Respuestas proceso TAR.

| **PREGUNTA** | **PASOS** | | | |
|---|---|---|---|---|
| | **Definición de requisitos de la API** | **Selección de tecnologías** | **Especificación** | **Generación código API** |
| **¿Qué actividad presentó obstáculos?** | Ninguno | Incompatibilidades de algunos complementos de acuerdo a la versión de Eclipse | Diseñar la especificación de acuerdo al problema planteado. | Crear las relaciones entre las tablas con anotaciones. |
| **¿Qué tareas o actividades extras debieron ser realizadas?** | Entrevistas a usuarios, relevamiento de documentación, especificación de escenarios. | Instalar los complementos de acuerdo a la versión de Eclipse. | Aprendizaje de la estructura, comandos, secciones para el diseño de una especificación. | -Aprendizaje de los distintos tipos de anotaciones para crear relaciones entre entidades.<br>-Comentar el plugin "OpenAPI Generator" en el archivo pom.xml, para que no vuelva a crear el código generado.<br>-Comentar la propiedad spring.jpa.hibernate.ddl-auto=update, del archivo application.properties, para evitar que se vuelvan a crear las tablas. |
| **Duración** | 7 (días) | 20 (días) | 15 (días) | 15 (días) |
| **Complejidad** | Baja | Media | Media | Media |

Fuente: Elaboración propia.

## V. AMENAZAS A LA VALIDEZ Y LIMITACIONES

En el marco de DSR, la validez se fundamenta en demostrar que el artefacto desarrollado efectivamente resuelve el problema identificado y cumple con los objetivos de diseño establecidos. Peffers et al. [18] incorporan la evaluación como un paso crítico del proceso DSR, la cual en este estudio se operacionalizó mediante la aplicación de TAR [19]. Esta elección metodológica se justifica en que la Investigación en Acción, y específicamente TAR, constituye un método confiable y sistemático para evaluar y validar artefactos DSR en contextos reales, como lo demuestran Cleven et al. [21].

Para minimizar los sesgos inherentes a la evaluación del artefacto, se implementaron las recomendaciones metodológicas propuestas por Wieringa y Morali [19]:

- Separación de roles: La aplicación de la guía al escenario definido fue ejecutada por un investigador independiente, distinto al equipo que desarrolló el artefacto, reduciendo así el sesgo de confirmación y aumentando la objetividad en la evaluación.
- Estructura iterativa controlada: La validación se diseñó como un proceso cíclico con pasos claramente definidos, donde cada iteración involucra diferentes actores con roles y funciones específicas, asegurando múltiples perspectivas en la evaluación.
- Mecanismos de detección de fallas: Los datos recolectados en cada ciclo TAR se estructuraron mediante consignas específicas para la identificación sistemática de deficiencias en la guía, permitiendo mejoras iterativas del artefacto.
- A pesar de las medidas implementadas, persisten ciertas limitaciones:
- Validez externa: La instanciación se limitó a un caso específico, lo que puede restringir la generalización de los resultados a otros contextos o dominios de aplicación.
- Validez de constructo: La evaluación se centró en la aplicabilidad práctica de la guía, pero requiere validación adicional para confirmar que los constructos teóricos subyacentes se manifiestan adecuadamente en diferentes contextos.
- Sesgo del investigador: Aunque se implementó separación de roles, la interpretación de los resultados sigue siendo susceptible a sesgos interpretativos inherentes al proceso de investigación cualitativa.

Estas limitaciones, sin embargo, son consistentes con las características exploratorias del enfoque DSR y proporcionan direcciones claras para investigaciones futuras que amplíen la validación del artefacto desarrollado.

## VI. DISCUSIONES Y CONCLUSIONES

API-First es definido de diversas formas por sus promotores, como enfoque, metodología, modelo de desarrollo, a la vez existen diversos esquemas de aplicación (API-First design, API-First Code, API-First Prototype, etc.), siendo API-First Design y API-First Code, las más mencionadas. También en algunos casos, se relaciona a actividades como el ciclo de vida de la API, gobierno y/o administración de API, pero estas interacciones no son claras. Es necesario, una mayor indagación que unifique, integre y clarifique estos conceptos y establezca con mayor precisión las clasificaciones. Según los promotores de API-First, la adopción de este enfoque logra optimizar distintos aspectos del desarrollo de software que son pilares en la ingeniera de software (reutilización, menores costos y disminución de errores, etc.), priorizando el diseño, implementación y pruebas de APIs, pero aún no son claros y evidentes los requisitos o condiciones necesarios para dicha adopción.

Como se ha descripto en la Sección 2, el enfoque API-First está más desarrollado en la industria que en la academia, se proporcionan guías o recomendaciones para la adopción que guardan cierto nivel de similitud y a la vez diferencias, en algunos casos importantes. Por otro lado, estas guías tienden a orientarse a las herramientas ofrecidas por las empresas. La limitación que se observa, es que los pasos o etapas de dichas guías, se formulan en función de las capacidades de las herramientas que se proponen, como sucede en Postman o las comunidades API Evangelist y APIAdicct.

En este estudio se ha presentado una guía técnica para la adopción de API-First. La misma cumple con los principios básicos del enfoque API-First Design. A partir de cuatro pasos, se distribuyen actividades para desarrollar una API. La guía propone un paso que consiste en seleccionar tecnologías y herramientas, de esta manera se logra que las mismas no influyan en la guía (en forma general). Sin embargo, este paso es clave, puesto que el conjunto de herramientas incide en la determinación de las actividades de los pasos especificación y generación de código y según los estudios de [10], estas herramientas podrían no ser adecuadas para la adopción del API First. Las herramientas pueden influir en la complejidad y adopción del enfoque, ya que requieren acciones (instalaciones, configuraciones, etc.) muy específicas en los restantes pasos. La evaluación y demostración de la misma en el caso real usado, mostró que la duración y complejidad de los pasos y actividades, están relacionados con la falta de experiencia y manejo en las tecnologías, cuya aplicación puede requerir que el aprendizaje se convierta en actividades adicionales, aunque se trate de tecnologías conocidas y maduras. Por eso, se sostiene que el paso de selección de tecnologías de esta propuesta, permite definir tecnologías alternativas existentes, lo que la hace más flexible. La guía técnica cumple así con el objetivo planteado y resulta útil para potenciales adoptantes al indicar los pasos claves a realizar y las decisiones y actividades más críticas de este proceso.

Así y a partir de las experiencias registradas, se concluye que API-First es dependiente de las herramientas específicas que se elijan para la automatización de las tareas de desarrollo de las APIs, pero de acuerdo a la evaluación presentada en este estudio, también es dependiente de la experiencia de los desarrolladores en el uso de esas herramientas, como lo indica en [10]. Desde una perspectiva de investigación falta mayor andamiaje conceptual y técnico para describir completamente a API-First.

El trabajo futuro está dirigido a continuar la indagación del enfoque API-First desde fuentes de la industria, y recolectar aquellas características comunes que presenten las distintas guías a efectos de incorporarlas, y hacer que la guía técnica propuesta evolucione a un framework genérico.

## VII. REFERENCIAS

[1] L. Richardson, M. Amundsen y S. Ruby, RESTful Web APIs, O'Reilly Media, 2013.
[2] J. M. Rivero, S. Heil y J. Grigera, «MockAPI: An Agile Approach Supporting API-first Web Application Development,» (eds) Web Engineering. ICWE 2013. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg., vol. 7977, pp. 7-21, 2013.
[3] W. Tan, Y. Fan, A. Ghoneim, M. A. Hossain y S. Dustdar, «From the Service-Oriented Architecture to the Web API Economy,» IEEE Internet Computing, vol. 20, nº 4, pp. 64-68, 2016.
[4] N. Beaulieu, S. M. Dascalu y E. Hand, «API-First Design: A Survey of the State of Academia and Industry,» ITNG 2022 19th International Conference on Information Technology-New Generations. Advances in Intelligent Systems and Computing, vol. 1421, pp. 73-79, 2022.
[5] P. (s.f.), «Guide to API-first,» [En línea]. Available: https://www.postman.com/api-first. [Último acceso: 29 11 24].
[6] N. Beaulieu, S. Dascalu y E. Hand, «API Integrator: A UI Design and Code Automation Application Supporting API-First Design,» ACIT '22: Proceedings of the 9th International Conference on Applied Computing & Information Technology, pp. 36-40, 2023.

[7] Postman, «State of the API Report,» 2023. [En línea]. Available: https://www.postman.com/state-of-api/2023/. [Último acceso: 29 11 2023].
[8] J. Lin, «Medium. API-first software development for modern organizations,» 2018. [En línea]. Available: https://medium.com/better-practices/api-first-software-development-for-modern-organizations-fdbfba9a66d3. [Último acceso: 11 09 2023].
[9] L. Trieloff, «Medium. Three Principles of API First Design,» 2017. [En línea]. Available: https://blog.developer.adobe.com/three-principles-of-api-first-design-fa6666d9f694. [Último acceso: 01 09 2023].
[10] O. Hämäläinen, «API-First Design with Modern Tools. Thesis, School of Business, Degree Programme in Business Information Technology,» 2019. [En línea]. Available: https://urn.fi/URN:NBN:fi:amk-2019060615295. [Último acceso: 23 10 2023].
[11] J. Wagner, «Swagger. Plan Your API-First Program,» [En línea]. Available: https://swagger.io/resources/articles/adopting-an-api-first-approach. [Último acceso: 07 09 2023].
[12] API: Addicts (s.f.), «Implementa API-First con las APITools,» [En línea]. Available: https://www.apiaddicts.org/apitools. [Último acceso: 10 10 2023].
[13] K. Lane, The API-First Transformation, Postman Inc., 2022, p. 262.
[14] K. Lane, «APIscene,» [En línea]. Available: https://www.apiscene.io/author/kin-lane. [Último acceso: 29 07 2024].
[15] M. Dudjak y G. Martinovic, «An API-first methodology for designing a microservice-based Backend as a Service platform.,» Information technology and control, vol. 49, nº 2, pp. 206-223, 2020.
[16] J. J. Londoño Tirado, «Gobierno de APIs, implementación y experimentación con API-First y OpenAPI en el proyecto P2P energía transactiva. Trabajo de grado. Universidad de Antioquia,» [En línea]. Available: https://hdl.handle.net/10495/29121. [Último acceso: 20 10 2023].
[17] J. Larsson y L. Åkermark, «The value of implementing API-First as a methodology when developing APIs (Dissertation) - Jonkoping University,» 2021. [En línea]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:hj:diva-54311. [Último acceso: 26 09 2023].
[18] K. Peffers, T. Tuunanen, M. A. Rothenberger y S. Chatterjee, «A design science research methodology for information systems research.,» Journal of Management Information Systems: JMIS, vol. 24, nº 3, pp. 45-77, 2007.
[19] R. Wieringa y A. Morali, «Technical Action Research as a Validation Method in Information Systems Design Science,» Design Science Research in Information Systems. Advances in Theory and Practice, vol. 7286, pp. 220-238, 2012.
[20] G. Hadad, J. Doorn y G. Kaplan, «Explicitar Requisitos del Software usando Escenarios,» Workshop em Engenharia de Requisitos, 2009.
[21] A. Cleven, P. Gubler y K. M. Hüner, «Design alternatives for the evaluation of design science research artifacts,» International Conference on Design Science Research in Information Systems and Technology, 2009.