Identificación de Anomalías de APIs web en Mashup

Cinthia Lima, Graciela Vidal y Sandra Casas

GISP - Instituto de Tecnología Aplicada
Universidad Nacional de la Patagonia Austral
Campus universitario Piloto Lero Rivera s/nro. Río Gallegos Santa Cruz
cinty.calderon.15@gmail.com , gvidal@uarg.unpa.edu.ar y sicasas@uarg.unpa.edu.ar

Resumen Las aplicaciones web mashup son el resultado de extraer y combinar información o datos de diversas fuentes externas. Las APIs web son fundamentales en el proceso de desarrollo de una web mashup ya que permiten el acceso a información de distinto tipo (texto, imágenes, videos). Luego, cuando la aplicación mashup está disponible, es posible que se produzcan anomalías en las respuestas de las APIs web, esto ocasiona fallas o pérdidas de información. Por este motivo, es necesario que los desarrolladores identifiquen estas anomalías para manejar estas situaciones. Este trabajo presenta un mashup que integra información de las redes sociales Facebook, Twitter y YouTube, e incorpora un mecanismo para identificar y registrar anomalías conocidas como respuestas vacías y respuestas mal formadas. A partir de un archivo JSON personalizado se registran las anomalías para su análisis. Los resultados se presentan mediante ejemplos sobre la aplicación web y el reporte de anomalías encontradas.

1. Introducción

Un mashup es una aplicación web que combina el contenido de más de una fuente externa y lo integra para que se pueda acceder desde un único lugar [1]. Estas aplicaciones son consideradas híbridas, ya que extraen información o datos de diversas fuentes, con lo cual se obtiene como resultado un nuevo sitio web [2]. Según [3], una web mashup es una aplicación web que integra datos, una aplicación lógica y/o interfaces de usuario de origen web, típicamente un mashup integra y orquesta dos o más elementos. Un componente típico suelen ser las APIs web.

Las APIs son interfaces de programación de aplicaciones que brindan métodos y facilidades para acceder al contenido de alguna aplicación [4]. Las APIs de servicios web, ofrecen un enfoque sistemático y extensible, a diferencia de las APIs vinculadas estáticamente o locales [5][6] para acceder a recursos, servicios y datos a través de la red. En particular, las APIs web facilitan el intercambio de datos inter e intra organizacionales disponibles a través de puntos finales específicos [7].

Las redes sociales se han convertido en los mayores proveedores en este contexto, ofrecen sus APIs y las comparten con el fin de que los desarrolladores integren la información y funcionalidades características de cada una de ellas en nuevas aplicaciones.

Durante la ejecución de una web mashup pueden suceder eventos no esperados, principalmente en el acceso a la respuesta de las APIs, algunos de ellos pueden ser respuestas mal formadas o respuestas vacías [8]. La aplicación mashup se ejecutará erróneamente o presentará información incompleta. Por lo tanto, es necesaria una estrategia para la identificación de estas anomalías con el fin de analizar su alcance sobre la aplicación. De este modo, el desarrollador puede manejar estas situaciones,

ya sea cambiando de API web, modificando la implementación del mashup, o realizar otra acción.

Este trabajo presenta el diseño e implementación de un esquema de resolución para el problema planteado. Se trata de un mashup web que integra y compone las novedades de cuentas en redes sociales (Facebook, Twitter y YouTube) de diferentes áreas de la Universidad Nacional de la Patagonia Austral y además identifica y registra anomalías producidas en las respuestas de las APIs web consumidas (respuestas vacías y respuestas mal formadas).

Este estudio se organiza de la siguiente forma, en la Sección 2 se presenta una definición general de API web y anomalías relacionadas. En la Sección 3 se presenta el caso de estudio, en la Sección 4 se muestran ejemplos del registro de anomalías, en la Sección 5 se exponen los trabajos relacionados, finalmente se presentan las conclusiones en la Sección 6.

2. APIs web y anomalías

El desarrollo de software moderno es inseparable del uso de APIs [9]. Las APIs web pertenecen a una nueva generación de APIs, denominadas API de servicios web, estas ofrecen un enfoque sistemático y extensible, a diferencia de las API vinculadas estáticamente o locales [5][6]. Las APIs web facilitan el intercambio de datos inter e intra organizacionales disponibles a través de puntos finales específicos [7]. Las APIs web se utilizan como un mecanismo de interconectividad clave para acceder a servicios de software a través de Internet. Las aplicaciones interconectadas pueden proporcionar un mejor servicio a un menor costo para sus usuarios [10]. Los catálogos online como API Harmony, PublicAPI o ProgrammableWeb enumeran miles de APIs web y dan cuenta de la proliferación de este tipo recursos, además las grandes empresas IT como Google, Amazon, Facebook, Twitter y YouTube, ofrecen compartir información, recursos y servicios a partir de las APIs web.

Existen diversos problemas relacionados con las respuestas obtenidas de las consultas y/o llamadas a las APIs web, algunos de ellos son las respuestas mal formadas y las respuestas vacías [8]. Estos problemas afectan el funcionamiento de las aplicaciones mashups, por lo tanto, su identificación y análisis es necesario para llevar a cabo su gestión, debido a que impactan de manera negativa en las mismas.

Las respuestas mal formadas se refieren a la generación de respuestas en un formato inadecuado, que no corresponde con el esperado. Por ejemplo, cuando se codifica una cadena JSON que contiene comillas dobles y estas no se ubican correctamente (es decir, "foo": "b" ar " en contraposición al formato válido " foo "=" b\"ar"). Otro ejemplo, ocurre en documentos XML, si se rompe una etiqueta de la respuesta (en este caso, '<data>' se convierte en '<data').

Por otro lado, una API puede devolver respuestas vacías, esto sucede por diferentes razones, por ejemplo si el servidor web está al límite de su capacidad máxima o si la conexión se interrumpe debido a problemas de comunicación, entre otras. En ambas situaciones, la aplicación web mashup no se ejecutará normalmente, presentando información incompleta o no podrá cumplir con otras operaciones [8].

3. Caso de estudio: Portal de publicaciones de redes sociales UNPA

Se desarrolló un portal web de publicaciones realizadas por diferentes áreas de la UNPA en sus redes sociales. Además, sobre este sitio web es posible identificar y reportar anomalías en las respuestas de las APIs web consumidas. En la Fig. 1 se presenta la propuesta desarrollada.

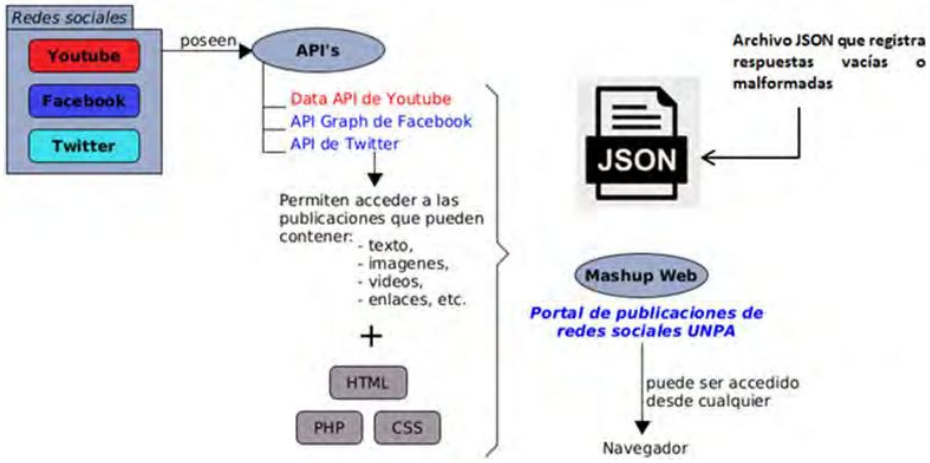

Figura 1: Representación de la propuesta

En este portal se integraron las APIs de las redes sociales, YouTube, Facebook y Twitter. A través de las mismas es posible acceder a diferentes contenidos que luego son presentados en el portal. Además, mientras el portal está en ejecución se registran las anomalías detectadas en las respuestas a las APIs web consumidas.

Para esto se llevaron a cabo tareas considerando los datos incluidos en la respuesta de cada API y los requeridos en el portal. Primero, se analizó la estructura de las respuestas, esto facilitó la identificación de los campos relevantes para el portal.

Aunque el formato de las respuestas de las APIs es el mismo, las estructuras de las respuestas no son idénticas. Por lo tanto, también fue necesario filtrar y ordenar los campos de cada respuesta con el fin de presentar sólo aquellos que el mashup necesitaba mostrar.

Luego, se estudió las anomalías: respuestas vacías y respuestas mal formadas. Se realizaron pruebas sobre cada una de las APIs web con el fin de observar el comportamiento del mashup ante las mismas. Para ello, se incluyó un registro de actividad, el mismo consiste en un archivo JSON, el cual se actualiza cada vez que se detecta pérdida de información en el caso de que se detecten las respuestas vacías o pérdida de la respuesta total, en relación a las respuestas mal formadas.

En la Fig. 2 se muestra la página de inicio del portal de publicaciones de redes sociales UNPA desarrollado. La información se presenta en tres columnas, una para cada red social. Los contenidos incluyen texto, imágenes y videos.

Figura 2: Página de inicio del portal

En la Fig. 3 se describen las consultas y las respuestas obtenidas de la API Graph de Facebook, Twitter y Youtube. En (a) se presenta la Url base (texto negro), seguida del identificador de la página de Facebook, en este caso se trata del área de extensión.unpa.uarg. Luego se especifican los campos que se requieren en la respuesta (texto rojo), en este caso los posts, el texto y la imagen de cada uno. También se indica la cantidad de posts que debería incluir la respuesta (texto naranja), por último se debe incluir el token de acceso del usuario (texto verde). Con el campo access token y los permisos adecuados es posible realizar diversas solicitudes a esta API web (lectura, eliminación, modificación y agregación). En (b) se muestra la respuesta con los 25 últimos posts, con la imagen y texto como se especificó en la consulta.

Luego se replica la consulta y la respuesta de la API de Twitter, en (c) se observa la url base seguida del identificador de la cuenta (texto azul), se indican los campos que se esperan recuperar de los tweets (texto rojo), a continuación se define la cantidad de tweets requeridos (texto violeta). El campo de texto de tweet es retornado por default, sin embargo es posible especificar algunos campos más como la fecha de creación y la imagen asociada si existen (texto color naranja). Por último se debe indicar el tipo de autorización. En (d) se presenta la respuesta al igual que el caso anterior, con los cinco últimos tweets de la cuenta sobre la que se realizó la consulta. Cada tweet posee un id de identificación, la fecha de creación, el texto del tweet y una imagen. La API de Twitter no solicita permisos para realizar lectura de contenido público dentro de la red social. Para las acciones de modificación, eliminación y agregación es necesaria la autenticación OAuth 1.0.

Por último, se especifica la consulta y la respuesta de la API de YouTube, en (e) se identifica la Url base utilizada (texto negro). La búsqueda se realiza con el ID del canal (texto azul), la cantidad máxima de resultados deseados (texto fucsia), a continuación se ordenan por fecha (texto verde), luego se indica el tipo de contenido requerido (color violeta), en este caso video. Finalmente se debe proporcionar la clave de api (texto naranja). En (f) se muestra la respuesta resultante en formato JSON, es

extensa por lo tanto se deben filtrar solo los campos que serán presentados en la aplicación final, a diferencia de las otras APIs que devuelven respuestas resumidas.

Figura 3. Consultas y respuestas a las APIs consumidas.

3.1 Registro de anomalías

El registro de anomalías (RA) genera como resultado un archivo JSON, en esta instancia los algoritmos verifican solo respuestas vacías y respuestas mal formadas. El RA se crea desde el inicio de la aplicación, cada vez que se ejecuta y/o invoca una consulta a una API web, un proceso analiza el archivo de respuesta y crea un objeto por cada anomalía detectada. Cada objeto se describe mediante los siguientes campos: tipo, API, contenido, número de Post o Tweet, fecha y hora. En la Fig. 4 se presenta un objeto creado en el archivo que registra la actividad.

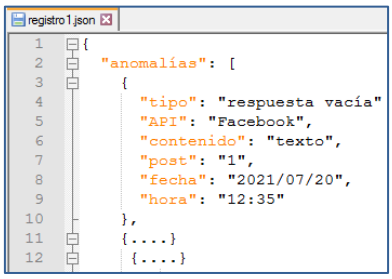

Figura 4. Registro de anomalías

Para verificar la existencia de anomalías en las respuestas de las APIs web consumidas por el mashup desarrollado se han diseñado dos algoritmos. En ambos casos, los posteos se copian en arreglos bidimensionales, y se recorren analizando las condiciones que representan las anomalías. El algoritmo 1 verifica si la respuesta contiene respuestas vacías, y el algoritmo 2 examina la respuesta en busca de respuestas mal formadas. A continuación, se expresan estos algoritmos.

```
Algoritmo 1
Entrada: Json con los posts (JP)
Salida : entradas en el registro de actividades (RA)
MP ← Generar un arreglo bidimensional con JP
FOR EACH fila de MP
  FOR EACH columna de fila
      IF MP[fila][columna] = vacia
         registrar [fila][columna] en RA
      endIf
  endForeach
endForeach
```

```
Algoritmo 2
Entrada: Json con los posts (JP)
Salida : entradas en el registro de actividades (RA)
MP ← Generar un arreglo bidimensional con JP
FOR EACH fila de MP
  FOR EACH columna de fila
      IF jsondecode(MP[fila][columna])
         registrar MP[fila], [columna]en RA
      endIf
  endForeach
endForeach
```

La estructura flexible del RA posibilita su utilización en otros entornos. La información que contiene es de utilidad para tomar de decisiones respecto de las APIs. Con este fin se plantean distintas consultas sobre las anomalías que afectan la aplicación mashup. En la Fig.5 se destacan los campos a ser chequeados para responder a algunas de estas consultas.

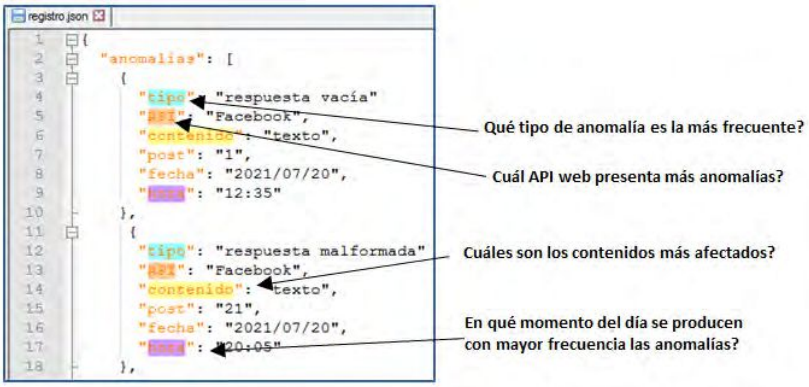

Figura.5. Consultas sobre el registro de anomalías

4. Ejemplos del RA

A continuación, se presentan ejemplos del registro de anomalías detectadas sobre el portal de publicaciones de la UNPA.

En la Fig.6 se expone un ejemplo de respuesta obtenida de la API de YouTube, que retorna cuatro campos con respuestas vacías (a). En el primer post no fue posible recuperar el video, en el segundo la publicación está completa, con texto y video. En el tercer post los campos de texto y video están vacíos por lo tanto se pierde el post completo. En el cuarto posteo el campo de video está vacío al igual que en el primer post. Los detalles de los problemas detectados se presentan en (b).

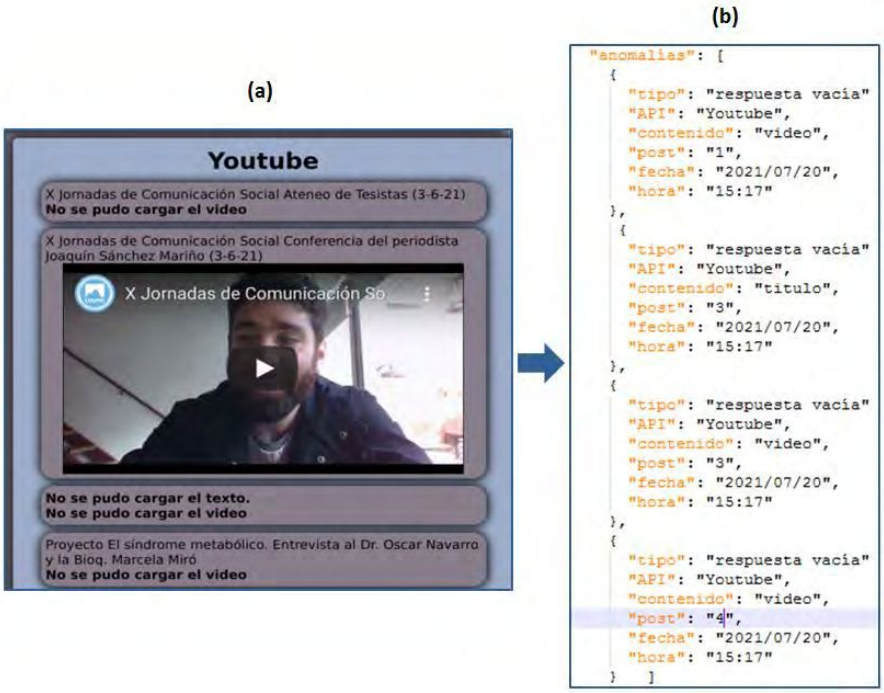

Figura 6. Respuestas vacías en YouTube

En la Fig.7 se muestra el portal de publicaciones y el contenido del RA. En este caso, (a) se obtuvo una respuesta mal formada de la API de Twitter por esta razón la columna correspondiente a esta red social se encuentra vacía y sólo aparece un mensaje al usuario informando que no se pudieron recuperar los tweets. En las otras APIs no se detectaron problemas en el contenido de las respuestas. La anomalía se refleja en (b) con la información sobre la respuesta mal formada identificada. En este caso, además del tipo, del nombre de la API web, la fecha y hora se incorpora un campo con un mensaje.

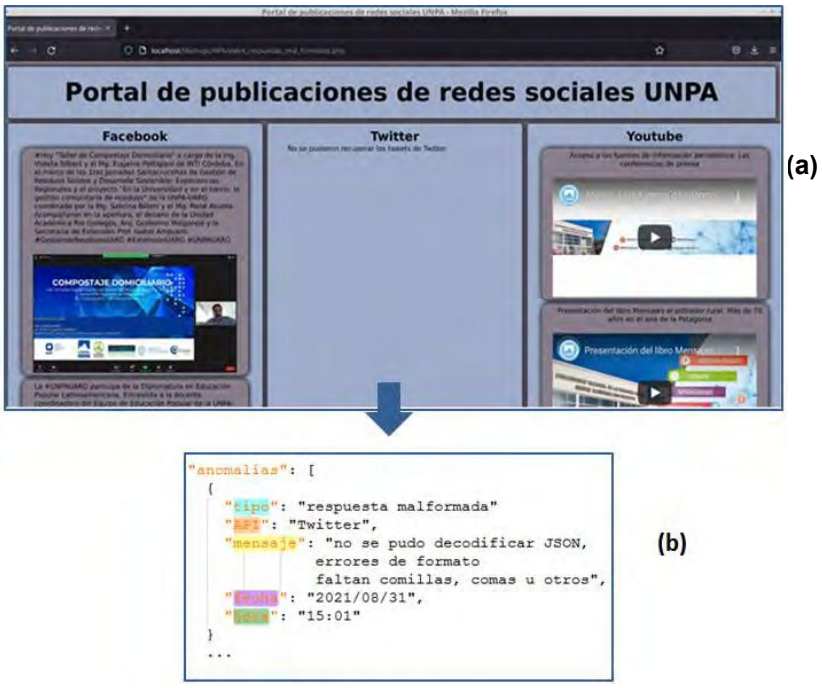

Figura 9. Respuesta mal formada de Twitter.

5. Trabajos relacionados

Existen diversos trabajos que presentan estudios enfocados a problemas originados en el uso de las APIs web. En [11][12] proponen herramientas para identificar errores y usos indebidos de APIs en aplicaciones JavaScripst. Ambas propuestas a partir de las especificaciones de las APIs realizan comprobaciones estáticas, que pueden identificar errores en la codificación de las consultas. Estas herramientas pueden resolver algunos de los errores de ejecución por la evolución de las APIs o los enumerados por [13], sin embargo, no atiende las causas de las anomalías de respuestas vacías o mal formadas

Las APIs web pueden generar diversos problemas de ejecución en la web mashup y/o cualquier aplicación que las consuman, por diversos factores, entre estos, los cambios en las APIs, producto de la evolución de las mismas. Cambios en los puntos finales, en las operaciones, en los parámetros y respuestas de la API web son reportados en diversos estudios, tales como [8][10][14][15]. Algunos de estos cambios podrían generar problemas en la ejecución, pero no parecen estar relacionadas a las anomalías que este trabajo aborda.

En [13] analizan 2,43 millones de respuestas (logs) de error de la API del servicio de pago Adyen. El objetivo del estudio es comprender las fallas que ocurren en la integración de la API web, las cuales potencialmente pueden resultar en problemas de producción para los consumidores de una API. Los resultados muestran que, (a) las fallas en la integración de la API se pueden agrupar en 11 causas generales: entrada de usuario no válida, entrada de usuario faltante, datos de solicitud caducados, datos de

solicitud no válidos, datos de solicitud faltantes, permisos insuficientes, procesamiento doble, configuración, datos de servidor faltantes, internos y de terceros, (b) la mayoría de las fallas pueden atribuirse a solicitudes no válidas o datos faltantes. Observamos que las anomalías de respuestas mal formadas y respuestas vacías, no están explícitamente abordadas, podría ser debido a que no son directamente registradas en el log del servidor de la API o bien porque su frecuencia de ocurrencia es baja o suceden en forma inadvertida para la API web. También podrían ser atribuidas a la tipificación como causa de fallos "internos". Por otro lado, el estudio se basa en el análisis de errores (log) del lado del productor de la API, mientras que nuestra propuesta plantea el análisis de errores del lado del consumidor (mashup).

Otros trabajos [16][17][18] han usado los registros de logs de las APIs web para realizar diversos análisis de estructura, calidad, usabilidad, etc. Nuestro trabajo propone usar la técnica de logs personalizados, pero para uso de los consumidores de APIs web.

6. Conclusiones

Este trabajo abordó la problemática para desarrolladores de web mashup relacionada a anomalías en la ejecución de los APIs web, conocidas como respuestas mal formadas y vacías. En particular, a partir del desarrollo de un web mashup de publicaciones en redes sociales, se implementaron algoritmos para detectar estas situaciones en las respuestas de las APIs de Facebook, Twitter y YouTube y registrar dichos eventos de manera específica.

Este registro de actividades (RA) o log personalizado es una estrategia que permite a los desarrolladores de mashup identificar y manejar estas situaciones. Esta estructura, permite al desarrollador, no solamente identificar los problemas en las respuestas de las APIs que utiliza la aplicación mashup, además es posible utilizar la información detallada en el mismo en otras etapas o por otras aplicaciones.

Una mejora a nuestra propuesta consistirá en proporcionar una solución más reusable y modular, que, a partir de una librería, diversos métodos o funciones, analicen las respuestas de las APIs y en consecuencia actualicen el registro de actividades. Sin embargo, es un gran desafío manejar la heterogeneidad de las estructuras de las respuestas de las APIs web.

El trabajo futuro consiste en continuar la personalización del RA, incorporar más anomalías al análisis, y proponer un enfoque más reusable.

7. Referencias

1. Yee, R. Pro Web 2.0 Mashups: Remixing Data and Web Services. Editorial: Apress. ISBN: 978-1-59059-858-0. (2008)
2. Dávila Macías & A. M.. Tesis. Recuperado a partir de http://repositorio.ug.edu.ec/handle/redug/6767. (2012)
3. Daniel F., Muhammand I., Soi S., De Angeli A., Wikinson C., Casati F., & Marchese M. “Developing Mashup Tools for End.Users: On the Importance of the Application Domain”, International Journal on Next-generation Computing, Vol 2. No 2, (2012).
4. Robbes R., Lungu M. & Janes A.“API fluency” ICSE-NIER '19: Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results. pp 97–100. https://doi.org/10.1109/ICSE-NIER.2019.00033 (2019)

5. Curbera F., Duftler M., Khalaf R., Nagy W., Mukhi N., & Weerawarana S. “Unraveling the web services web: an introduction to SOAP, WSDL, and UDDI,” Internet Computing, vol. 6, no. 2, pp. 86–93. (2002)
6. Vinoski S. “Restful web services development checklist,” IEEE Internet Computing, vol. 12, no. 6, pp. 96–95. (2008)
7. Dekel U. & Herbsleb J.D. “Reading the documentation of invoked API functions in program comprehension” ICPC’09, pp. 168–177.(2009)
8. Espinha T., Zaidaman A. & Gross HG. Web API Fragility: How Robust Is Your Web API Client?. Software Engineering Research Group - Department of Software Technology. ISSN 1872-5392. (2014)
9. Raemaekers S., van Deursen A., & Visser J. “Measuring software library stability through historical version analysis,” in Proc. Int’l Conf. on Software Maintenance (ICSM). IEEE CS. pp. 378–387. (2012)
10. Sohan S.M, Anslow C. & Maurer F. A Case Study of Web API Evolution. IEEE World Congress on Services. Doi: 10.1109/SERVICES.2015.43.(2015)
11. Writtern E., Ying A., Zheng Y., Dolby J., & Laredo J. Statically checking web API requests in JavaScript. In Proceedings of the 39th International Conference on Software Engineering. IEEE Press.(2017).
12. SungGyeong B., Hyunghun Ch., Inho L. & Sukyoung R. SAFEWAPI: web API misuse detector for web applications. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. DOI:https://doi.org/10.1145/2635868.2635916. (2014)
13. Aué, J., Aniche, M., Lobbezoo, M., & van Deursen, A. An Exploratory Study on Faults in Web API Integration in a Large-Scale Payment Company. In ICSE-SEIP '18: 40th International Conference on Software Engineering: Software Engineering in Practice Track (pp. 13-22). doi.org/10.1145/3183519.3183537.(2018)
14. Li J., Xiong Y., Liu X., & Zhang L. “How does web service API evolution affect clients?” 20th Conf. on Web Services (ICWS). IEEE, pp. 300–307.(2013)
15. Fokaefs M., Mikhaiel R., Tsantalis N., Stroulia E., Lau A.An Empirical Study on Web Service Evolution. IEEE International Conference on Web Services.(2011)
16. Koçi, R., Franch, X., Jovanovic, P., & Abelló, A. Improving Web API Usage Logging. arXiv preprint arXiv:2103.10811.(2021)
17. Suter P. & Wittern E. Inferring web API descriptions from usage data. In HotWeb,IEEE, DOI: 10.1109/HotWeb36442.2015 (2015).
18. Macvean A., Daughtry J., Church J.& Citro C. “API Usability at Scale." Proceedings of the 26th annual workshop of the Psychology of Programming Interest Group (2016)