UNIVERSIDAD PERUANA UNIÓN

FACULTAD DE INGENIERIA Y ARQUITECTURA

Escuela Profesional de Ingeniería de Sistemas

Una Institución Adventista

La arquitectura de software basada en microservicios: Una revisión sistemática de la literatura

Por:

Cesar Benjamin Pareja Valerio

Leandro Jair Burgos Robles

Asesor:

Mg. Fredy Abel Huanca Torres

Lima – Perú

Diciembre 2019

DECLARACIÓN JURADA
DE AUTORÍA DEL TRABAJO
DE INVESTIGACIÓN

Fredy Abel Huanca Torres, de la Facultad de Ingeniería y Arquitectura, Escuela Profesional de Ingeniería de Sistemas, de la Universidad Peruana Unión.

DECLARO:

Que el presente trabajo de investigación titulado: "LA ARQUITECTURA DE SOFTWARE BASADA EN MICROSERVICIOS: UNA REVISIÓN SISTEMÁTICA DE LA LITERATURA" constituye la memoria que presentan los estudiantes Leandro Jair Burgos Robles y Cesar Benjamin Pareja Valerio para aspirar al grado de bachiller en Ingeniería de Sistemas, cuyo trabajo de investigación ha sido realizado en la Universidad Peruana Unión bajo mi dirección.

Las opiniones y declaraciones en este trabajo de investigación son de entera responsabilidad del autor, sin comprometer a la institución.

Y estando de acuerdo, firmo la presente declaración en Lima, al 3 de diciembre del año 2019.

Mg. Fredy Abel Huanca Torres

La arquitectura de software basada en microservicios: Una revisión sistemática de la literatura

Trabajo de investigación

Presentado para optar al grado de bachiller en Ingeniería de Sistemas

JURADO CALIFICADOR

Dra. Erika Inés Acuña Salinas
Presidente

Mg. Nemias Saboya Rios
Secretario

Ing. Jenson Daniel Chambi Aguilar
Vocal

Mg. Daniel Levano Rodriguez
Vocal

Ing. Fredy Abel Huanca Torres
Asesor

Lima, 02 de diciembre del 2019

La arquitectura de software basada en microservicios: Una revisión sistemática de la literatura

Burgos Robles Leandro Jair[1] and Pareja Valerio Cesar Benjamin[2]

[1,2]Universidad Peruana Unión, Km 19 Carretera Central, Ñaña, Lurigancho, Lima, Perú

Resumen. El estilo de arquitectura de software basado en microservicios se ha ido convirtiendo en un tema de interés debido a los múltiples beneficios que otorga a las organizaciones que lo adoptan para el desarrollo de sus sistemas de software. Este estilo propone desarrollar la aplicación como un conjunto de pequeños servicios, cada uno ejecutándose independientemente y comunicándose entre sí. El presente artículo busca identificar patrones, buenas prácticas y técnicas relacionadas con la arquitectura de software basada en microservicios. Para la identificación de los elementos mencionados se realizó una revisión sistemática de la literatura en las bases de datos reconocidas. De un total de 394 artículos revisados, se identificaron 24 artículos que hacen referencia a la arquitectura basada en microservicios y sus elementos. Luego de realizar una revisión sistemática de la literatura se pudo identificar 8 patrones en los cuales se destaca la contenerización de servicios y la descomposición de monolitos, 1 técnica relacionada a la migración de aplicaciones monolíticas a microservicios y 4 buenas prácticas destacadas las cuales pertenecen a DevOps.

Palabras claves: Arquitectura de Software, Microservicios, Patrones, Buenas Prácticas, Técnicas.

1 Introducción

Las innovaciones tecnológicas han producido cambios beneficiosos en la sociedad, especialmente en las comunicaciones y el manejo de la información [1] mediante aplicaciones que están al alcance de las personas. Además, se ha generado oportunidades de crecimiento y competitividad en las organizaciones mediante los sistemas de información que permiten gestionar sus finanzas, logística, procesos y demás actividades [2]. Para obtener todos los beneficios mencionados, el desarrollo de software ha evolucionado a lo largo de los años y se ha ido adaptando a las necesidades del entorno.

La creación de software sigue un proceso al cual se le conoce como el Ciclo de vida del Desarrollo de Software especificado en la ISO 12207. Dentro de este proceso se encuentra la fase de diseño del software, en la que se define la arquitectura que se utilizará en el proyecto con el fin de orientar la construcción del software. Nuseibeh [2] describe el paradigma en el desarrollo que suele separar la definición de los requerimientos y la arquitectura de software al momento de gestionar el proyecto, el cual crea una brecha entre estas fases del ciclo de vida, causando la independencia en las especificaciones tanto de requerimientos como de estructura.

Para estructurar un proyecto es necesario conocer los distintos estilos de arquitectura que existen y escoger cuál se adecua a las características y necesidades del proyecto. Dentro de los distintos estilos se encuentra el estilo de la arquitectura basado en microservicios, la cual ha captado la atención de los diversos mercados. Muchas organizaciones han empezado a migrar sus tradicionales sistemas monolíticos a la arquitectura de microservicios debido a los beneficios (escalabilidad, disponibilidad, etc.) que trae esta arquitectura en comparación con la arquitectura monolítica [3].
El objetivo principal de esta revisión sistemática es identificar patrones, buenas prácticas y técnicas relacionados a la arquitectura de software basada en microservicios.
El artículo se distribuye de la siguiente manera: la sección II expone el marco conceptual; la sección III presenta la metodología utilizada; la sección IV contiene los resultados obtenidos y por último la sección V abarca las conclusiones.

2 Marco Conceptual

En esta sección se presentan definiciones que abarcan el entorno en el que se realiza el estudio y el objeto de análisis.

2.1 Arquitectura de Software

La arquitectura de software fue declarada una disciplina por Dewayne Perry y Alexander Wolf's en su artículo "Foundations for the Study of Software Architecture" [4] e inicialmente su propósito era "resolver problemas básicos de estructuras de software estáticos", pero en la actualidad se busca gestionar la comunicación entre los componentes de software que conforman un sistema.
En las últimas 4 décadas la arquitectura de software ha evolucionado de forma significativa debido al desarrollo de nuevas tecnologías. Sin embargo, aún son usadas para distintos tipos de proyectos [5].
En la década de 1980 el estilo de desarrollo predominante era de tipo monolítica, con la tendencia a ejecutar los sistemas en computadoras individuales tanto en servidores centrales (mainframe) como en las computadoras de usuario único. En esta etapa se destacan las aplicaciones de escritorio las cuales cumplían estas características.
Ingresando en la década de 1990 se incorpora el concepto de distribución de esfuerzos. Es decir, el procesamiento sería compartido entre el Servidor y el Cliente. Sin embargo, los programas heredan la estructura de código de las aplicaciones monolíticas que tiene como característica la rigidez de ser un solo paquete ejecutable.
Para lograr ordenar y estilizar el código, Jim Althoff crea el patrón de diseño MVC como una librería en Smalltalk-80.
Para inicios de los años 2000 ya era una realidad la evolución de Internet. La incorporación de los sistemas a una red mundial significaba cambios, especialmente en las cualidades no funcionales como el rendimiento y la escalabilidad. El objetivo principal de los sistemas se convirtió en "Always available", es decir, mantenerse siempre en línea.
A partir de 2010, Internet ya era un servicio básico con el que se contaba. La arquitectura de software adoptó estilos que permitían la interconexión entre aplicaciones o

incluso componentes de software por medio de APIs. Es decir, la flexibilidad se volvió una propiedad necesaria para este estilo de arquitectura. A inicios de esta década se establecieron el protocolo de servicios web SOAP y RESTFUL, quienes permiten la comunicación entre aplicaciones independientemente del lenguaje en el que fueron desarrolladas [5].
REST es un estilo arquitectónico de servicios web el cual usa la URI para exponer la lógica del negocio. REST a menudo tiene muchos recursos y es una arquitectura orientada a recursos mientras que SOAP es un protocolo el cual usa interfaces de servicios para exponer la lógica del negocio. Las aplicaciones desarrolladas bajo REST son llamadas aplicaciones RESTful [6].
A partir de esto, se introduce un nuevo concepto, la arquitectura de software basada en microservicios adopta estas características y asegura la disponibilidad de una aplicación además de su independencia total en la codificación, siendo la arquitectura más moderna y base para los avances futuros.
Se identifica las tendencias que serán una realidad en la siguiente década debido al gran avance que se tiene en la actualidad. El internet de las cosas basa sus conceptos y estructura en las características de los microservicios. En primer lugar, se tiene la disponibilidad en la nube como una de las características que se utiliza para esta tecnología.

2.2 Arquitectura basada en microservicios

Fowler y Lewis (2014) definen el estilo arquitectónico de microservicios como un enfoque para el desarrollo de una aplicación integrada por un conjunto de pequeños servicios, cada uno ejecutándose independientemente y comunicándose con mecanismos ligeros, siendo en la mayoría de las veces un API de recursos HTTP [7].
En cierto grado, el estilo arquitectónico de microservicios puede ser visto como una extensión natural de SOA (Service-Oriented Architecture), la cual pone énfasis la independencia y autogestión de servicios, y en su ligera naturaleza [8].
Los microservicios son elementos arquitectónicos de bajo acoplamiento que pueden ser desplegados independientemente. A menudo, estos despliegues independientes son logrados mediante el uso de plataformas ligeras basadas en contenedores [9], cuyos principales beneficios de una arquitectura basada en microservicios incluyen escalabilidad, alta disponibilidad, tolerancia a fallos, aislamiento de problemas, elasticidad horizontal de los recursos, entre otros beneficios organizacionales [10].
Las aplicaciones que utilizan este estilo arquitectónico tienden a ser diseñadas para tolerar fallas y errores de manera resiliente [10].

2.3 P atrones, Buenas Prácticas y Técnicas

Un patrón es la solución a un problema definido ya sea en la creación , estructura o comportamiento de los componentes de software [11]. En el libro Design Patterns Elements Reusable Object Oriented se agrega que los patrones permiten reutilizar soluciones para diferentes problemas que mantengan similitud.
En [12] se define a las buenas prácticas en el campo del desarrollo de software al conjunto de técnicas recopiladas de la experiencia de los desarrolladores en distintos proyectos.

En [13] se define a las técnicas como un “conjunto de reglas, normas o protocolos, que tienen como objetivo obtener un resultado determinado”. En este estudio se utiliza el término para identificar procedimientos.

3 Definición de la revisión sistemática de la literatura

3.1 Necesidad de la revisión sistemática

La metodología conocida como Revisión Sistemática de la Literatura [14] se utilizó para la recolección y análisis de información relacionada con el objeto de estudio.
Debido a que el objeto de estudio es un tema de interés actual, se vio necesario utilizar una metodología que permita delimitar la búsqueda bajo criterios específicos para obtener una correcta literatura.
Esta revisión sistemática surge con la necesidad de identificar patrones, buenas prácticas y técnicas que están relacionadas a la arquitectura de software basada en microservicios. Esta necesidad se manifiesta debido a la evolución que han tenido los estilos de arquitecturas de software en las últimas 4 décadas lo que causa una interrogante al momento de definir los elementos necesarios para un proyecto de software con una arquitectura basada en microservicios [8]. Se utilizó la plantilla Goal, Question, Metric (GQM por sus siglas en inglés) como se observa en la Tabla I.
Debido a la naturaleza de esta investigación no se está aplicando ni evaluando de manera experimental.

Tabla I ELABORACIÓN DEL OBJETIVO DE LA INVESTIGACIÓN

CAMPO	VALOR
Objeto de estudio	Arquitectura de software basada en microservicios
Propósito	Identificar
Foco	Patrones, buenas prácticas, y técnicas
Involucrados	Arquitectura de software, desarrollo de software, industria de software, proceso de desarrollo de software, estándares de desarrollo, ingeniería de sistemas
Factores de contexto	No aplica para este caso

3.2 Preguntas para la revisión sistemática

Las preguntas realizadas están basadas en el propósito de esta investigación, el cual fue mencionado en la sección anterior. En la Tabla II se encuentran las preguntas propuestas para la revisión sistémica y las motivaciones en la que se basan. En la Tabla III están las preguntas de bibliometría con la finalidad de distinguir la evolución de los estudios sobre el objeto de estudio en un lapso definido.

3.3 Protocolo de la investigación

El siguiente protocolo está basado en la guía de la Revisión Sistemática de la Literatura [14] en el cual se describe los criterios a utilizar para la ejecución de la búsqueda de información relevante para desarrollar este estudio.

Tabla II PREGUNTAS DE INVESTIGACIÓN Y MOTIVACIÓN

ID	Pregunta	Motivación
PI-01	¿Cuáles son los patrones arquitectónicos relacionados al estilo de arquitectura basada en microservicios?	Identificar los patrones que se existen y están relacionados con la arquitectura de software basada en microservicios
PI-02	¿Cuáles son las técnicas relacionadas a la arquitectura basada en microservicios?	Identificar las técnicas que existen y están relacionados con la arquitectura de software basada en microservicios
PI-03	¿Cuáles son las buenas prácticas relacionadas a la arquitectura basada en microservicios?	Identificar las buenas prácticas que existen y están relacionados con la arquitectura de software basada en microservicios

Tabla III PREGUNTAS DE BIBLIOMETRÍA

ID	Pregunta	Motivación
PB-01	¿Cuál es la cantidad de publicaciones por tipo de artículo?	Identificar la cantidad de estudios publicados por tipo de artículo
PB-02	¿Cómo ha evolucionado en el tiempo la frecuencia de las publicaciones sobre este tema?	Identificar la frecuencia de las publicaciones dentro de un intervalo de tiempo de 5 años
PB-03	¿Cuál es la cantidad de publicaciones del tema por base de datos?	Identificar en qué base de datos cuenta con mayor cantidad de artículos relacionados con la arquitectura basada en microservicios

3.3.1 Definición de las cadenas de búsqueda

Población

- Término Principal: software architecture based on microservices

- Término Alterno: Architecture Styles; Architecture Models; Software Architecture
- Justificante: Se selecciona el término por ser el objeto de estudio de la revisión a ejecutar y se obtienen los términos alternos que representan las variantes o cercanos al término principal.

Intervención

- Término Principal: Microservicios
- Término Alterno: independent services; microservices
- Justificante: Por ser el término parte del objeto de estudio al cual se enfoca el presente trabajo de revisión

Resultados

- Término Principal: Best Practices
- Término Alterno: Patterns; Tactics
- Justificante: Por ser el foco de esta investigación se incluyen estas palabras en la cadena de búsqueda.

Tabla IV Plantilla GQM

CONCEPTO	**TÉRMINOS**
Población	("Software architecture" or "Architecture styles" or "Architecture models")
Intervención	(“Independent services" or "Microservices")
Comparación	-
Resultado	"techniques" or “patterns” or “best practices”
Contexto	-

3.3.2 Criterios de inclusión y exclusión

Criterios de Inclusión

CI.1. Se consideran todos aquellos artículos provenientes de librerías digitales indexadas y del indexador Google Scholar.

CI.2. Los artículos deben provenir del área de Ingeniería de Software.

CI.3. Se aceptarán artículos que contengan estudios o análisis comparativos de herramientas o metodologías de software.

CI.4. Se considerarán todos los artículos que se encuentren dentro del rango de temporalidad definido.

CI.5. Se aceptarán artículos provenientes de revistas científicas y conferencias, workshop papers, special issue papers.

CI.6. Solo se aceptarán artículos redactados en inglés

C.I.7. Se aceptarán artículos de mayor antigüedad si es que son de contenido conceptual

Criterios de Exclusión

CE.1. Serán excluidos los artículos duplicados.

CE.2. Serán rechazados los artículos de contenido similar, quedándose solo los que tengan el contenido más completo.

CE.3. Serán excluidos los estudios secundarios, estudios terciarios y resúmenes.

CE.4. Serán excluidos los artículos cuyo título no tenga relación con el objeto de estudio

Temporalidad. Se consideró como estado del arte a las publicaciones de los 5 últimos años, a excepción de los conceptos sobre arquitectura de software. Teniendo en cuenta que pese a la variabilidad de las tecnologías y herramientas existen principios que aún tienen vigencia a la actualidad.

Fuentes de datos. Las librerías digitales indexadas consideradas para este estudio fueron:

- IEEEXplore (http://www.ieee.org/web/publications/xplore/)
- ACM Digital Library
- Google Scholar

Procedimiento. Se aplicó la metodología RSL siguiendo los siguientes pasos:

- Paso 1: Se ejecutó las cadenas con caracteres booleanos (AND, OR) en las librerías mencionadas enlazando los términos previamente mostrados.
- Paso 2: Se aplicó el filtro del paso 2 (Tabla V) para obtener resultados que apoyen a la investigación y sean relacionados con el tema de estudio.
- Paso 3: Los artículos se limitaron con los criterios del paso 3 (Tabla V) para descartar aquellos que no están en el rango de tiempo definido
- Paso 4: Se excluye el uso de artículos que sean solo recopilaciones de otros autores siguiendo los criterios del paso 4 (Tabla V)

Tabla V Criterios de selección

PROCEDIMIENTO	CRITERIO DE SELECCIÓN
Paso 1	-
Paso 2	CI1 , CI6 y CE1
Paso 3	CI4 , CI5 , CE5 y CI7
Paso 4	CE3 y CE4

3.3.3 Criterios de calidad

- **¿La metodología seleccionada para llevar a cabo el estudio ha sido documentada apropiadamente?**
  - S: La metodología seleccionada documentada apropiadamente.
  - P: La metodología seleccionada documentada parcialmente.
  - N: No se ha documentado la metodología seleccionada.
- **¿El estudio menciona las amenazas sobre la validez de la investigación?**
  - S: El estudio cubre las amenazas totalmente

  - P: El estudio cubre las amenazas parcialmente.
  - N: No se detallan amenazas.
- **¿Se han documentado las limitaciones del estudio de manera clara?**
  - S: Las limitaciones se han documentado claramente.
  - P: Las limitaciones se han documentado parcialmente.
  - N: No se han documentado las limitaciones.
- **¿Los aportes del estudio para las comunidades científica, académica o para la industria han sido descritos?**
  - S: Los aportes del estudio han sido mencionados claramente.
  - P: Los aportes del estudio han sido mencionados parcialmente.
  - N: No se han mencionado aportes.
- **¿Los resultados han contribuido a responder las preguntas de investigaciones planteadas?**
  - S: Los resultados han contribuido a responder todas las preguntas de investigación.
  - P: Los resultados han contribuido a responder algunas preguntas de investigación.
  - N: Los resultados no han contribuido a responder las preguntas de investigación.

Tabla VI FORMULARIO PARA LA EXTRACCIÓN DE DATOS

Criterio	Detalle	Relevancia
Identificador		
Fuente		
Título		
Autores		
Publicación		
Año de publicación		
Tipo de publicación		
Tipo de análisis comparativo		
Objetivo del análisis		
Elementos comparados		
Criterios de comparación utilizados		

Dominio de aplicación		

4 Resultados

4.1 Resultados de la búsqueda

Para obtener mejores resultados se realizó modificaciones de sintaxis en la cadena de búsqueda para cada librería.
La búsqueda en IEEE Xplore requería agrupar 2 elementos y un operador booleano dentro de paréntesis
La búsqueda en Google Scholar requería modificar la búsqueda predeterminada, seleccionar la configuración avanzada y cambiar el intervalo de tiempo según lo definido en los criterios de inclusión.

Tabla VII Resultados de Búsqueda

Base de Datos	Fecha	Total
Cadena de Búsqueda		
IEEE Explore	mayo 2019	15
(((("Software architecture") OR "Architecture styles") OR "Architecture models") AND (("microservices") OR ("Independent services")) AND ((("best practices") OR ("experiences")) OR ("proposal"))		
ACM Digital Library	mayo 2019	30
(((("Software architecture") OR "Architecture styles") OR "Architecture models") AND (("microservices") OR ("Independent services")) AND ((("best practices") OR ("experiences")) OR ("proposal"))		
Google Scholar (Indexador)	mayo 2019	349
("Software architecture" OR "Architecture styles" OR "Architecture models") AND ("microservices") AND ("best practices" OR "experiences" OR "proposal") -IOT -book -"Internet of things"		

4.2 Seleccionar los estudios primarios

- Paso 1: La lista de artículos obtenidos de la cadena de búsqueda ejecutada fue filtrada para no tener artículos duplicados, que provengan de una librería digital no indexada, no pertenezcan al área de Ingeniería de Software y no estén redactados en inglés.
- Paso 2: Se descarta de la lista aquellos artículos que no hayan sido publicados en los 5 últimos años, a excepción de referencias conceptuales vigentes. También se descartan los artículos que no guardan relación con el objeto de estudio.
- Paso 3: En el último paso se excluyen los artículos que sean versiones pasadas de sí mismos o solo contenga un resumen.

Tabla VIII Filtración de Artículos

Base de datos	**Artículos descubiertos**	**Paso 1**	**Paso 2**	**Paso 3**
Google Scholar (Indexador)	349	296	31	15
IEEE Xplore	15	15	15	4
ACM	30	28	26	5
Total	394	339	72	24

4.3 Evaluar la calidad de los estudios

A los 24 artículos seleccionados luego del paso 3 se aplicaron los criterios de calidad mencionados en la sección III. Podemos observar que el 50% de los artículos sobrepasa la media y el 91.7% supera la puntuación 3 indicando que cumplen como mínimo con el 60% de la evaluación de calidad mencionada en la sección III.

4.4 Extraer los datos relevantes

Al evaluar la calidad del contenido de los artículos con los criterios establecidos en la sección III se obtuvo la puntuación presentada en la tabla VIII.

Tabla IX Evaluación de la calidad del contenido

ID	Nombre del Artículo	C1	C2	C3	C4	C5	Total
1	The pains and gains of microservices: A Systematic grey literature review	1	1	0,5	0	1	3,5
2	Microservices tenets	1	1	0,5	0	1	3,5
3	Microservices in the modern software world	0	1	0,5	1	1	3,5
4	Architecting microservices	1	1	0,5	0,5	1	4
5	Software Architecture in a Changing World	1	1	0,5	0,5	1	4
6	Continuous experimentation in start-up: Approaches and Improvements	1	1	1	1	0,5	4,5
7	From monolith to microservices: Lessons learned on an industrial migration to a web oriented architecture	1	1	1	1	0,5	4,5
8	Model-Integrating Microservices: A Vision Paper.	1	1	0,5	0,5	0,5	3,5
9	Correlations study and clustering from SPI experiences in small settings	1	1	0,5	0,5	0,5	3,5
10	Extraction of microservices from monolithic software architectures	1	1	1	1	0,5	4,5
11	U-RPC: a Protocol for Microservices in DHT	1	1	1	1	0,5	4,5
12	Designed and delivered today, eroded tomorrow?: towards an open and lean architecting framework balancing agility and sustainability	1	1	0,5	0,5	0,5	3,5
13	Performance engineering for microservices: research challenges and directions	0	1	1	0,5	0,5	3
14	The evolution of Java based software architectures	0,5	1	1	0,5	0,5	3,5

15	Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud	1	1	1	1	1	5
16	Benchmark Requirements for Microservices Architecture Research	1	1	1	0,5	0,5	4
17	Towards a practical maintainability quality model for service-and microservice-based systems	1	1	1	0	0	3
18	Model-driven Generation of Microservice Architectures for Benchmarking Performance and Resilience Engineering Approaches	1	1	0,5	0,5	0,5	3,5
19	Exploration of academic and industrial evidence about architectural tactics and patterns in microservices	1	1	1	0	1	4
20	Disambiguation and Comparison of SOA, Microservices and Self-Contained Systems	1	1	0,5	1	1	4,5
21	Microservices Architecture Enables DevOps: An Experience Report on Migration to a Cloud-Native Architecture	1	1	1	1	0,5	4,5
22	Containerized development and microservices for self-driving vehicles: Experiences & best practices	1	1	1	1	0	4
23	Actor based business process modeling and execution: A reference implementation based on ontology models and microservices	1	1	0,5	1	0	3,5
24	Microservices in Practice, Part 1: Reality Check and Service Design	1	1	0,5	0,5	0,5	3,5

En la tabla IX se muestra un ejemplo de la extracción de datos importantes a un artículo.

Tabla X Ejemplo de Extracción de Datos

Criterio	Detalle	Relevancia
Identificador	22	-
Fuente	IEEE Xplore	PB-01,PB-05
Título	Containerized development and microservices for self-driving vehicles: Experiences & best practices	PB-01
Autores	Christian Berger, Björnborg Nguyen, Ola Benderius	PB-01
Publicación	2017 IEEE International Conference on Software Architecture Workshops (ICSAW)	PB-03

Año de publicación	2017	PB-02
Tipo de publicación	Conference Paper	PB-01
Objetivo del análisis	Experiences and best practices from using containerized software microservices for self-driving vehicles are shared	PI-06
Dominio de aplicación	Technology	-

4.5 Análisis bibliométrico

4.5.1 PB-01 ¿Cuál es la cantidad de publicaciones por tipo de artículo?

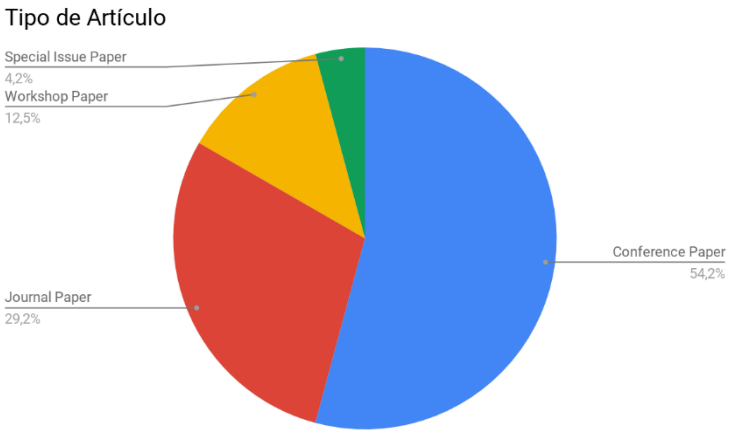

Gráfico 1.Tipos de Artículo

En el gráfico se observa el porcentaje que corresponde a cada tipo de artículo de la siguiente manera:
13 son de tipo “Conference Paper” y representa un 54.17% de la cantidad total, en segundo lugar, tenemos 7 de tipo “Journal Paper” con un 29.17%, luego se tiene 3 artículos de tipo “Workshop Paper” con un 12.5% y por último 1 artículo de tipo “Special Issue Paper” representado por un 4.17% del total.
Se observa una cantidad importante de artículos publicados en conferencias tecnológicas debido al trabajo de investigación constante en el área de desarrollo de software.

4.5.2 PB-02 ¿Cómo ha evolucionado en el tiempo la frecuencia de las publicaciones sobre este tema?

Luego de un análisis de los resultados conseguidos en la tabla V se puede observar en el Gráfico 2 los artículos seleccionados que fueron publicados en los últimos 5 años. En total se tiene 24 artículos de los cuales, 13 (54.17%) fueron publicados en el año 2017, 5 (20.83%) en el año 2016, 4 (16.67%) en el año 2018, 1(4.17%) en el año 2015 y, por último, 1 (4.17%) en el año 2019.

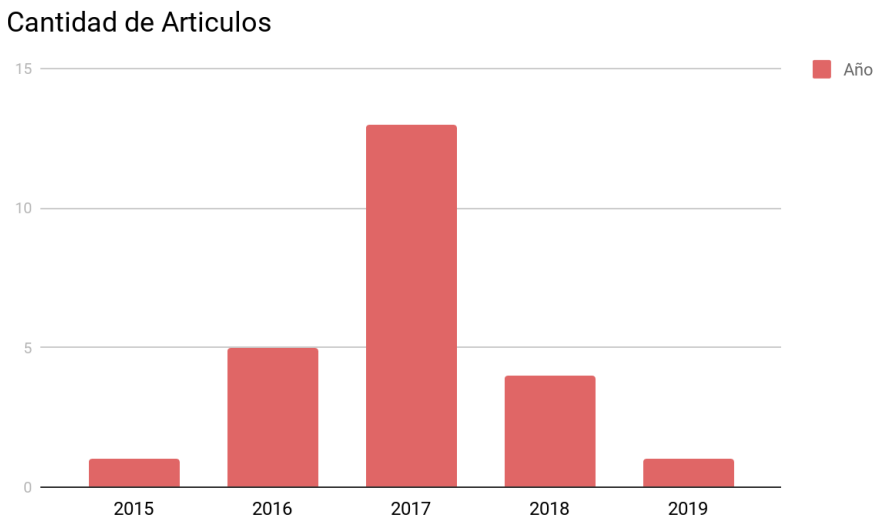

Gráfico 2. Año de publicación

4.5.3 PB-03 ¿Cuál es la cantidad de publicaciones del tema por base de datos?

Se ejecutó la cadena de búsqueda en 3 bases de datos científicas y se tuvo como resultado lo siguiente: 15 (62.5%) artículos encontrados en Google Scholar, 4 (16.7%) artículos se obtuvieron de IEEE Xplore y 5 (20.8%) artículos fueron seleccionados de ACM Digital Library como se muestra en el Gráfico 3.

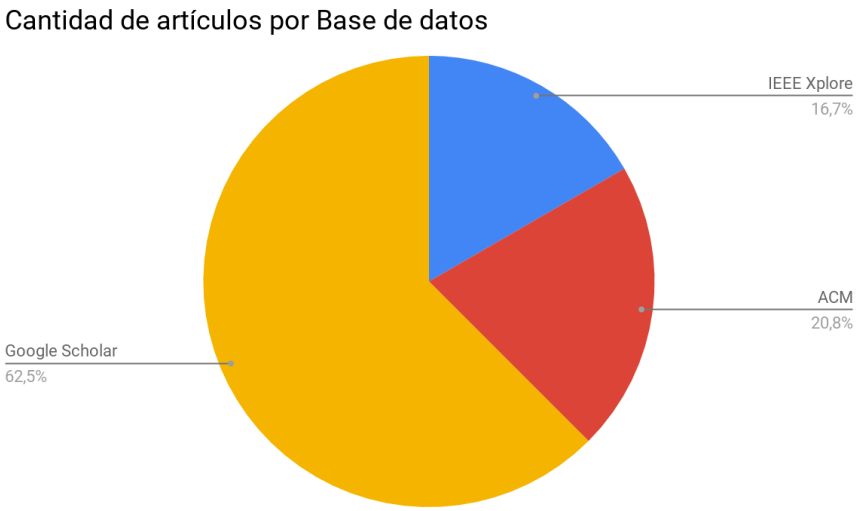

Gráfico 3. Base de datos de investigación

4.6 Sintetizar los datos extraídos

4.6.1 PI-01 ¿Cuáles son los patrones arquitectónicos relacionados al estilo de arquitectura basada en microservicios?

Los patrones de arquitectura orientada a microservicios encontrados fueron 38. Entre estos patrones hay similitudes, debido a que surgen adaptaciones para distintos proyectos, ya que cada proyecto de desarrollo de software busca cumplir con distintos requerimientos. El patrón arquitectónico debe ser escogido considerando este propósito. En la tabla XI se agrupó a los patrones identificados y la cantidad de adaptaciones que surgieron a partir de su estructura.

Tabla XI Patrones de microservicios

Nombre	Propósito	Relacionados
Modern Web Architecture	Combinar componentes de front-end con componentes de back-end	5
Single Page Application	Desarrollar nueva aplicación web o refactorizar una sección a web.	3

Native Mobile Application	Crear de aplicaciones que necesitan soporte en varias plataformas	5
Near Cache	Operar de forma eficiente cuando la conectividad a internet es mala.	4
Modular	Operar con módulos independientes	3
Decompose the Monolith	Descomponer el sistema en pequeños componentes	8
Containerize the Services	Aislar un servicio en una máquina virtual	7
DBaaS	Estructurar la base de datos como un servicio	3

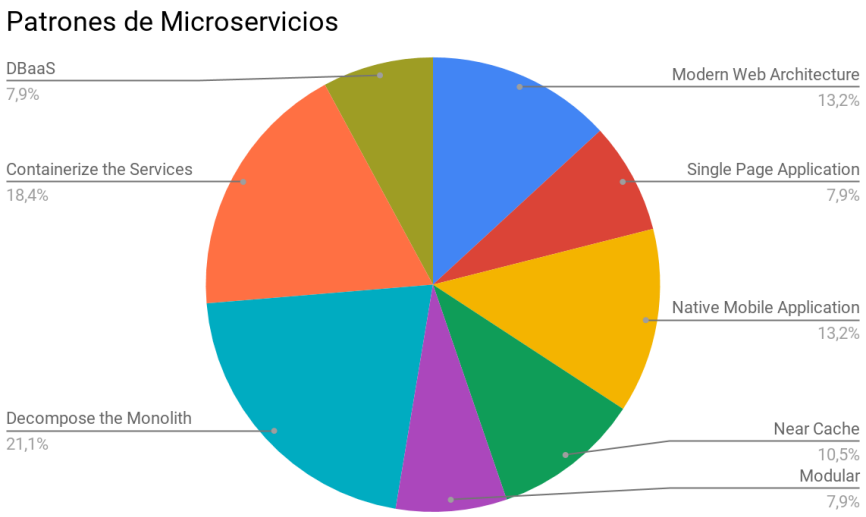

Gráfico 4. Patrones de Microservicios

Como se aprecia en el Gráfico 4 los patrones que destacan son: Descompose the Monolith con 21%, Containerize the Service con 18,4% y Modern Web Architecture con 13,2%. Los patrones más relevantes son los siguientes:

Descompose the Monolith : Descomponer el monolito. Para la incorporación de este patrón en un sistema es necesario acompañarlo con técnicas de migración. La evolución de las tecnologías ha abierto oportunidades de innovación en el desarrollo de software. La implementación de microservicios permite mejorar la calidad de software, de

manera especial en los atributos de calidad no observables. Esto lleva a la migración de arquitectura utilizando el patrón de descomposición de código en fragmentos pequeños no relacionados y con una función única.

Containerize the Service : Contenerizar el servicio. Este patrón busca que el despliegue de los microservicios se realice mediante contenedores. Por lo general está asociado con otros patrones de arquitectura y se toma como un complemento. La importancia de contenerizar los microservicios se centra en la facilidad para gestionar su contenido y su independencia.

Native Mobile Application : Aplicación móvil nativa. Este patrón propone una estructura que permita el funcionamiento de una aplicación siendo soportado por microservicios con características distintas como su lenguaje.

Modular. Este patrón permite a una aplicación tener uno o más módulos soportado por microservicios, es decir, parte del sistema puede poseer un patrón arquitectónico distinto. Se suele utilizar este patrón al hacer una migración por módulos de un sistema.

4.6.2 PI – 02 ¿Cuáles son las técnicas relacionadas a la arquitectura basada en microservicios?

Se encontró una técnica relacionada a la migración de aplicaciones monolíticas a microservicios. Esta es una técnica de extracción de microservicios bajo un enfoque de agrupamiento de microservicios basado en gráficos (Graph based microservice clustering approach) el cual usa acoplamiento estático y evolutivo de las clases de software. En este enfoque se representa el sistema de software como un gráfico donde se muestra las diferentes relaciones y acoplamiento entre las clases. El clusterizado agrupa el gráfico e identifica los posibles microservicios.
La técnica de extracción cuenta con dos módulos, el módulo de análisis evolutivo y el módulo de análisis estático, el módulo de análisis evolutivo extrae y guarda la información de los cambios del software usando repositorios, se extrae los acoplamientos evolutivos al detectar cambios en los commits consecutivos. El módulo de análisis estático extrae los acoplamientos estáticos en la versión final del software analizando árboles de sintaxis abstracta. Luego se crea el gráfico de relación de software con los acoplamientos estáticos y evolutivos para proceder a la clusterización.[15]

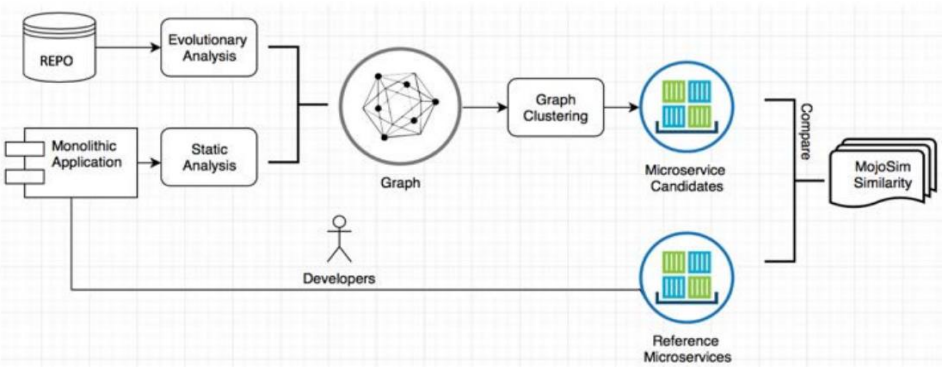

Gráfico 5. Técnica de extracción de Arquitectura de Software

4.6.3 PI - 03 ¿Cuáles son las buenas prácticas relacionadas a la arquitectura basada en microservicios?

Luego de analizar los resultados se observó que las prácticas encontradas pertenecían a DevOps, el cual es un conjunto de prácticas que no solo pretende reducir el tiempo entre la aplicación de un cambio en un sistema y la transferencia al entorno de producción, sino que también busca mantener la calidad del software en términos de código y de entrega.

Dentro de las prácticas más relevantes se encuentran:

- Continuous Delivery: Entrega Continua. Esta práctica permite el despliegue de componentes de software cuando se requieran en cualquier entorno. Junto con la Integración contínua y el Despliegue Continuo, es una de las prácticas fundamentales de DevOps. El paso de desarrollo a producción es su función principal, sin embargo, requiere de un componente humano que tome la decisión de ejecutar esta transición.
- Domain Driven Design: Diseño guiado por el dominio. Esta práctica busca la colaboración entre el equipo técnico y los expertos del negocio, al cual está dirigido el sistema. DDD crea una integración usando lenguaje semántico buscando que los dominios de la organización puedan reducir la complejidad al comunicarse
- Ports and Adapters Pattern: Patrón de puertos y adaptadores. Esta práctica permite gestionar la publicación de los contenedores de microservicios. Dependiendo de la demanda de un microservicio se tendrá un número de puertos con mucha variabilidad. Es por eso que esta práctica propone administrar la carga de los puertos en adaptadores de redistribución como nginx.
- Virtualization Platforms: Plataformas de Virtualización. Esta práctica propone crear imágenes a demanda, es decir, tener microservicios temporales publicados en máquinas virtuales. Dependiendo de su necesidad de requerimiento estos deben ser escalables de manera horizontal replicando su contenido para alivianar la carga.

5 Conclusiones

En este estudio se muestran los resultados de una revisión sistémica aplicada a 24 artículos seleccionados de librerías digitales indexadas. Estos artículos fueron escogidos luego de pasar por un análisis bibliométrico en el cual se busca encontrar aquellos estudios que apoyan al objetivo de esta investigación y hayan sido publicados dentro de los últimos 5 años.

Este estudio busca identificar patrones, buenas prácticas y técnicas relacionadas con la Arquitectura de Software basada en Microservicios.

Como se puede apreciar en los resultados de la presente investigación, los patrones que más se encontraron son: "decompose the monolith" y "containerize the services", la técnica encontrada es "Graph based microservice clustering approach" aplicada en la migración de las aplicaciones monolíticas hacia microservicios y las buenas prácticas encontradas se encuentran relacionadas también con DevOps. Esto muestra la creciente tendencia de las investigaciones no a construir nuevas aplicaciones basadas en microservicios sino a migrar sus aplicaciones monolíticas existentes hacia la arquitectura orientada a microservicios.

6 Referencias

[1] A. Van de Ven *et al.*, "Increasing benefits & reducing social costs of technological innovations," *Behav. Sci. Policy*, vol. 3, no. 1, pp. 92–103, 2017.

[2] B. Nuseibeh, "Weaving Together Requirements and Architectures," *Computer (Long. Beach. Calif).*, vol. 34, no. 3, pp. 115–119, 2016.

[3] P. Di Francesco, "Architecting microservices," *Proc. - 2017 IEEE Int. Conf. Softw. Archit. Work. ICSAW 2017 Side Track Proc.*, pp. 224–229, 2017.

[4] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture," *ACM SIGSOFT Softw. Eng. Notes*, vol. 17, no. 4, pp. 40–52, 1992.

[5] E. Woods, "Software Architecture in a Changing World," 2016.

[6] S. Malik and D. H. Kim, "A comparison of RESTful vs. SOAP web services in actuator networks," *Int. Conf. Ubiquitous Futur. Networks, ICUFN*, pp. 753–755, 2017.

[7] M. Fowler and J. Lewis, "Microservices : a definition of this new architectural term," *Microservices : a definition of this new architectural term*, 2014. [Online]. Available: http://martinfowler.com/articles/microservices.html.

[8] O. Zimmermann, "Microservices tenets: Agile approach to service development and deployment," *Comput. Sci. - Res. Dev.*, vol. 32, no. 3–4, pp. 301–310, 2017.

[9] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, "Cloud Container Technologies: a State-of-the-Art Review," *IEEE Trans. Cloud Comput.*, vol. 7161, no. c, pp. 1–14, 2017.

[10] A. Kwan, H.-A. Jacobsen, A. Chan, and S. Samoojh, “Microservices in the modern software world,” *CASCON ’16 Proc. 26th Annu. Int. Conf. Comput. Sci. Softw. Eng.*, pp. 297–299, 2017.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*, vol. 5. 2015.

[12] E. G. Maida and J. Pacienzia, “Metodologías de desarrollo de software,” Universidad Católica de Argentina, 2015.

[13] G. Arias, *El Proyecto de Investigación*, 6ta ed., vol. 6. Caracas: EDITORIAL EPISTEME, C.A., 2012.

[14] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, “Systematic literature reviews in software engineering - A systematic literature review,” *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, 2009.

[15] S. Eski and F. Buzluca, “An Automatic Extraction Approach - Transition to Microservices Architecture from Monolithic Application,” in *Proceedings of the 19th International Conference on Agile Software Development Companion - XP ’18*, 2018, pp. 1–6.