ABSTRACCIÓN DE TECNOLOGÍA SUBYACENTE Y GESTIÓN DE DOMINIOS EN *FRAMEWORKS* DE DESARROLLO: REVISIÓN SISTEMÁTICA DE LITERATURA

NICOLÁS SIMONCINI
https://orcid.org/0009-0003-6267-115X
nsimoncini@frba.utn.edu.ar
Facultad Regional Buenos Aires, Universidad Tecnológica Nacional, Argentina

PABLO PYTEL
https://orcid.org/0000-0002-3920-1710
ppytel@frba.utn.edu.ar
Facultad Regional Buenos Aires, Universidad Tecnológica Nacional, Argentina

Recibido: 26 de enero del 2025 / Aprobado: 14 de abril del 2025
doi: https://doi.org/10.26439/interfases2025.n021.7729

RESUMEN. Tanto las compañías de desarrollo de *software* como las empresas que implementan sus aplicativos *in house* suelen enfocar el diseño de sus productos de *software* utilizando diversos *frameworks* de desarrollo que facilitan, *out of the box*, la rápida implementación de los requisitos funcionales orientados a nichos específicos, como es el caso de aplicaciones web. No obstante, estos suelen perder el foco en la correcta gestión del *backend* en lo que respecta a la gestión de dominios de aplicación y el encapsulamiento y abstracción de las tecnologías subyacentes, es decir, no brindar herramientas o mecanismos de diseño para facilitar dicha abstracción. En este artículo, se pretende, a través de una revisión sistemática de literatura, analizar la existencia de investigaciones académicas que arrojen como resultado metodologías o protocolos válidos para el modelado de *frameworks* orientados al diseño de *backends* que permitan el encapsulamiento y la abstracción de la tecnología subyacente, a través del uso de dominios de aplicación válidos desde la perspectiva ontológica.

PALABRAS CLAVE: revisión sistemática de literatura / desarrollo de *software* / *framework* de desarrollo de *software* / ingeniería de dominio / diseño de ontologías

ABSTRACTION OF UNDERLYING TECHNOLOGY AND DOMAIN MANAGEMENT IN DEVELOPMENT FRAMEWORKS: A SYSTEMATIC LITERATURE REVIEW

ABSTRACT. Both software development companies and companies that implement in-house their applications usually focus the design of their software products using various development frameworks that facilitate "out of the box" the rapid implementation of functional requirements aimed at specific niches as is the case with Web applications, but they often lose focus on the correct management of the backend regarding to the management of application domains and the encapsulation and abstraction of the underlying technologies (that is, not providing tools or mechanisms of design to facilitate such abstraction). The present work aims to analyze, through a systematic literature review, the existence of academic research that results in valid methodologies or protocols for the modeling of frameworks oriented to the design of backends that allow the encapsulation and abstraction of the underlying technology and make use of ontologically valid application domains.

KEYWORDS: systematic literature review / software development / software development framework / domain engineering / ontology design

INTRODUCCIÓN

Es ampliamente conocido en la ingeniería del *software* que la utilización de *frameworks* de desarrollo aporta significativos beneficios en la creación de productos de *software* indistintamente de la plataforma donde se desea realizar la implementación. Entre los principales aportes se pueden destacar los siguientes (Ramirez et al. 2019; Riehle, 2000; Swacha & Kulpa, 2023):

- la posibilidad de reutilizar los artefactos desarrollados en un dominio específico y, por ende, la reducción en los tiempos de desarrollo y el consecuente incremento de la productividad
- la estandarización de patrones de diseño que permite a los desarrolladores enfocarse, más bien, en desarrollar funcionalidades en vez de resolver problemas de arquitectura
- la facilidad en la portabilidad al introducir abstracción y separación de las capas lógicas de las capas técnicas orientadas a la plataforma
- la escalabilidad que permite expandir el *software* desarrollado a medida que este incrementa su grado de complejidad, con el menor costo posible
- la sostenibilidad que se logra gracias a que, por norma general, los *frameworks* evolucionan en línea con las nuevas modas, requisitos de seguridad o nuevas tendencias en tecnología
- el soporte de la comunidad de usuarios para los casos de *frameworks* populares

Paralelamente, el proceso de encapsulamiento de los modelos y requisitos funcionales en dominios de aplicación se ha convertido en una tendencia que ha crecido de forma sostenida en el tiempo (Bjørner, 2006; Cambarieri et al. 2020; Evans, 2004; Johnson & Foote, 1988). Asimismo, la calidad de las estructuras y algoritmos definidos en los modelos puede ser medible y normalizada desde la perspectiva de las ontologías (Alrumaih et al., 2020; Lopez et al., 2011), lo que genera un valor agregado extra al momento de validar calidad.

Por otra parte, dependiendo del contexto del problema a solucionar, existen situaciones en las cuales el producto de *software* debe contar con el requisito de acceder o integrarse con tecnologías desarrolladas por terceras partes (clasificadas como subyacentes), que sirven como herramienta para cumplir su objetivo. Por citar algunos ejemplos, se puede encontrar con motores de bases de datos, librerías de uso común para la gestión de requisitos no funcionales (como la gestión de la resiliencia en los algoritmos) o *hardware* especializado, entre otros. Al mantenerse alineado a las buenas prácticas del diseño de *software*, resulta esperable que lo anterior sea implementado bajo las reglas establecidas por los principios de diseño SOLID (Martin, 2000; Martin

et al., 2018, capítulo III) y, más específicamente, su componente principio de responsabilidad única (*simple responsability principle*, SRP), con el fin de garantizar tanto la abstracción como el encapsulamiento de las funcionalidades internas.

En este contexto, se considera de importancia verificar, al menos conceptualmente, la existencia de alguna definición de un *framework* que tenga a bien agrupar los conceptos mencionados. En el presente artículo, se tiene como objetivo hallar dichas definiciones en el campo académico a través de una revisión sistemática de literatura. Para ello, primero se describen los principales fundamentos del trabajo (sección 2) y se detalla la metodología aplicada (sección 3). Luego, se indican los resultados obtenidos (sección 4) y su correspondiente análisis (sección 5). Por último, se presenta las conclusiones y principales líneas de trabajo futuro (sección 6).

FUNDAMENTOS

A medida que el campo de la ingeniería del *software* avanzó a lo largo del tiempo hacia la reutilización de componentes o piezas de *software* para lograr la optimización de recursos y uso del tiempo, diversos conceptos como patrones de diseño/arquitectura (Gamma et al., 2011; Johnson & Foote, 1988; Richards & Ford, 2020), *frameworks* de desarrollo de *software* (Martin, 2000; Stanojevic et al., 2011), gestión de dominios de aplicación (Bjørner, 2006; Cambarieri et al., 2020; Evans, 2004), entre otros, tomaron relevancia de forma progresiva en la materia con el objetivo de proveer las herramientas para tal fin. Si bien el campo académico provee diversos estudios en estos conceptos, se cuenta con pocas evidencias de investigaciones que tengan a bien agrupar y consolidar al menos de forma conceptual una metodología de desarrollo de *frameworks* suficientemente maduro (Ramirez et al., 2019).

En contraparte, se observa en el campo profesional un notorio avance en la oferta y evolución tecnológica de dichas herramientas, pero sin contar con fundamentos académicos de base. Tal es el caso de los *frameworks* de desarrollo orientados a plataformas web (Swacha & Kulpa, 2023). Esto quiere decir que existen diversas opciones, pero orientadas a dominios específicos donde la literatura académica relacionada se encuentra dispersa y carente de un enfoque centralizado y común para cada especificidad (Ramirez et al., 2019). Además, puede observarse casos en los que se plantean *frameworks* de desarrollo cuyo objetivo es la rápida puesta en marcha de proyectos de *software*, pero tienden a obviar las herramientas o mecanismos para la implementación de modelos del dominio recomendables para su aplicación (Ramirez et al., 2019). Entonces, si bien los modelos son tratados como artefactos manipulables dentro de la solución, dejan este espacio a la creatividad del equipo de implementación, sin contar con lineamientos o estructuras de base.

Por otra parte, aunque existen estándares para el diseño y arquitectura de *software* que brindan las herramientas que los proyectos carecen (Evans, 2004), existe poca evidencia científica que corrobore su implementación y uso a través de *frameworks* de desarrollo de software. Asimismo, se cuenta con evidencia académica orientada al estudio de soluciones de *software* para la abstracción de tecnología subyacente, pero que adolece de un enfoque, al menos básico, del uso de dominios de aplicación (Lei et al., 2011).

Finalmente, mediante el proceso de revisión sistemática de literatura propuesto en el presente trabajo de investigación, se intenta hallar puntos en común o equilibrio entre las evidencias académicas relacionadas con *frameworks* de desarrollo de *software*, dominios de aplicación y utilización de la tecnología subyacente a través de diversas preguntas de investigación y bajo un riguroso método de análisis de calidad sobre la información colectada.

METODOLOGÍA

Con el fin de garantizar la calidad del presente trabajo de investigación, se hace uso exhaustivo de la metodología y los lineamientos propuestos por Kitchenham y Charters (2007) para las revisiones sistemáticas de literatura. Entonces, las revisiones sistemáticas tienen como objetivo presentar una evaluación justa de un tema de investigación mediante el uso de una metodología confiable, rigurosa y auditable (Kitchenham & Charters, 2007).

Por lo tanto, se propone lo siguiente:

- analizar los diferentes enfoques, mecanismos, ejemplos o buenas prácticas para la definición o utilización de *frameworks* de desarrollo de *software* orientados al dominio de forma validable y con la capacidad de abstraer la utilización de tecnologías subyacentes
- evidenciar las posibles áreas de vacancia obtenidas en la investigación con el fin de dejar abierta la expansión de la temática abordada en el marco académico y profesional

Para conseguir dichas metas, en las siguientes subsecciones se detallan los pasos aplicados para llevar a cabo la identificación y procesamiento de los artículos relevantes.

Formulación de preguntas

Teniendo en cuenta las metas propuestas, en la Tabla 1 se presentan las preguntas de investigación definidas junto con su correspondiente justificación. La incorporación de estas justificaciones permite sustentar adecuadamente la pertinencia de las preguntas planteadas.

Tabla 1

Preguntas de investigación

Clave	Pregunta de investigación	Justificación
RQ1	¿Qué enfoques son utilizados para la generación de *frameworks* orientados al diseño de *backends* y agnósticos a la tecnología subyacente dentro del campo de la ingeniería de *software*?	Identificar cuál es el estado del arte vinculado al diseño de *frameworks* para la implementación de modelos de *backends* que permitan abstraer la tecnología subyacente indistintamente de la taxonomía.
RQ2	¿Qué enfoques son utilizados para la definición de dominios de aplicación con fundamentos ontológicos en el diseño de *backends* dentro del marco de la ingeniería de *software*?	Identificar cuál es el estado del arte vinculado al modelado de dominios de aplicación ontológicamente válidos.
RQ3	¿Existen estudios que combinen la generación de *frameworks* capaces de abstraer tecnología subyacente y contar con definición de dominios de aplicación ontológicamente válidos en el diseño de *backends* dentro del campo de la ingeniería de *software*?	Validar si existen casos de estudio que puedan responder RQ1 y RQ2 de forma combinada.
RQ4	¿Qué propuestas de mejora o lecciones aprendidas existen para el proceso de diseño de *frameworks* agnósticos a la tecnología subyacente con el uso de dominios de aplicación con fundamentos ontológicos para la implementación de *backends*?	Analizar los casos que cumplan RQ3 con el fin de identificar recomendaciones para futuras líneas de investigación en la materia.
RQ5	¿Qué conclusiones pueden deducirse de los estudios encontrados?	Validar si existe la posibilidad de plantear futuras líneas de investigación relacionadas al diseño de *frameworks* para *backends* que abstraigan la tecnología subyacente y hagan uso de dominios de aplicación ontológicamente válidos.

Búsqueda

Con el fin de recuperar de forma efectiva artículos que guarden relación con el objetivo del trabajo de investigación y las preguntas generadas, se crea una cadena de búsqueda booleana capaz de considerar de la mejor manera posible todos los puntos de interés. Resulta primordial en esta etapa contar con estrategias de búsqueda imparciales (Kitchenham & Charters, 2007). Para ello, como primera instancia se identifican las palabras clave y su correspondiente categoría a partir de las preguntas de investigación, tal como se detallan en la Tabla 2.

Tabla 2

Selección de palabras clave para la búsqueda con base en las categorías relacionadas

Categoría	Palabras clave
Framework de desarrollo de *software*	*Software*, *design*, *development*, *framework*
Dominio de aplicación	*Software*, *domain*, *"application domain"*
Ontología	*Ontology*
Tecnología subyacente	*Underlying*, *beneath*, *below*, *technology*

Definición de criterios de selección

Con el fin de garantizar la obtención de los estudios más relevantes, se define un proceso de selección. Para ello, se definen los criterios de inclusión y exclusión que deben basarse en las preguntas de investigación (Kitchenham & Charters, 2007) y cuya finalidad es asegurar la inclusión de estudios relevantes al trabajo de investigación. En la Tabla 3, se muestran los criterios de inclusión y exclusión específicos a ser aplicados con los datos obtenidos inicialmente tras la ejecución de la cadena de búsqueda booleana definida en la sección anterior.

Tabla 3

Criterio de inclusión y exclusión definidos para el presente trabajo de investigación

Criterio	Inclusión	Exclusión
Contenido	Cualquier estudio que tenga como foco principal o secundario el diseño o desarrollo de un *framework* de desarrollo	Cualquier estudio que no haga mención del desarrollo o diseño de un *framework* de desarrollo
Idioma	Inglés y español	Cualquier otro idioma
Tipo de estudio	Cualquier tipo de estudio académico	Fuentes no académicas, artículos de opinión no avaladas o editoriales
Tipo de acceso	Cualquier tipo de acceso académico	Cualquier acceso de pago con imposibilidad de obtener artículos por representación académica
Antigüedad	Publicación en los últimos veinte años	Publicaciones desactualizadas mayores a veinte años
Disponibilidad	Texto completo	Solo resumen o texto completo no disponible
Disciplina	Relacionada al campo de la ingeniería en sistemas de información	Cualquier disciplina que no tenga relación con el campo de la ingeniería en sistemas de información

Asimismo, se han seleccionado las fuentes de búsqueda indicadas en la Tabla 4. Debe notarse que también se toman como válidos los hallazgos provenientes de búsquedas manuales, recomendaciones o por referencias obtenidas desde estudios

previamente analizados, los cuales suelen denominados como seguimiento de referencias o *snowballing* (Greenhalgh & Peacock, 2005).

Tabla 4

Fuentes de consulta bibliográfica utilizadas

Nombre	Link	Descripción
IEEE Xplore	https://ieeexplore.ieee.org	Base de datos de investigación para el descubrimiento y acceso a artículos de revistas, actas de congresos, normas técnicas y materiales relacionados sobre informática, ingeniería eléctrica y electrónica y campos afines perteneciente al Instituto de Ingenieros Eléctricos y Electrónicos (IEEE)
ACM Digital Library	https://dl.acm.org/	Plataforma de investigación, descubrimiento y creación de redes perteneciente a la Asociación de Maquinaria Computacional (ACM) que contiene una completa base de datos bibliográfica centrada exclusivamente en el campo de la informática
Springer Link	https://link.springer.com/	Plataforma de información electrónica del campo de las ciencias, la técnica y las ciencias sociales que forma parte del ecosistema Springer Nature
Science Direct	https://www.sciencedirect.com/	Sitio web que brinda acceso a una gran base de datos bibliográfica de publicaciones científicas y médicas de la editorial holandesa Elsevier
SEDICI	http://sedici.unlp.edu.ar/	Repositorio institucional de la Universidad de La Plata, Argentina

Entonces, con el fin de poder refinar el resultado de la búsqueda de los artículos inicialmente obtenidos, tras la ejecución de la cadena de búsqueda en las fuentes de consulta bibliográfica, se define un mecanismo de filtrado sucesivo y progresivo a través de etapas o fases. Para el presente trabajo de investigación, se definieron cuatro filtros (Tabla 5).

Tabla 5

Filtros aplicados al trabajo de investigación

Filtro	Descripción
Rf0	Ejecución de cadena de búsqueda en las fuentes de consulta bibliográfica
Rf1	Refinamiento mediante el análisis de distancia mínima de dos palabras entre palabras clave
Rf2	Eliminación de artículos repetidos y lectura de título y resumen
Rf3	Lectura completa de artículos con el fin de garantizar la relevancia para el trabajo de investigación

Para poder aplicar el último filtro (Rf3), un hito muy importante en el proceso de revisión sistemática de la literatura dentro de la etapa de selección de artículos primarios fue la evaluación de la calidad (Kitchenham & Charters, 2007). Esta última fase de filtrado proporcionó criterios de inclusión y exclusión aún más detallados y sirvió como medio para ponderar la importancia de los estudios individuales cuando se sintetizaron los resultados. Si bien no existía una definición acordada en lo que se refiere al estudio de calidad, se sugirió que esta debía relacionarse con la medida en que un estudio minimice el sesgo y maximice la validez interna y externa, tal como se muestra en la Tabla 6.

Tabla 6

Definiciones del concepto de calidad

Término	Sinónimos	Definición
Sesgo	*Error sistemático*	La tendencia a producir resultados que se alejan sistemáticamente de los "verdaderos"
Validez interna	*Validez*	La medida en la que la realización del estudio probablemente evite errores sistemáticos
Validez externa	*Universalidad, aplicabilidad*	La medida en que los efectos observados en el estudio son aplicables fuera de este

Nota. Adaptado de *Guidelines for performing systematic literature reviews in software engineering* (*EBSE 2007-001*), de B. A. Kitchenham y S. Charters, 2007, p. 21 (https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf).

La verificación de la calidad se realiza por medio de la aplicación de cada estudio sobre una lista de verificación en donde cada elemento contiene una pregunta que debe responderse luego de la lectura y comprensión del estudio. Cada pregunta contiene asociado un puntaje de acuerdo con la respuesta específica, por lo que el resultado de la suma de todos los puntajes se denomina puntaje de calidad. Para definir si un artículo posee la calidad suficiente, su puntaje deberá hallarse dentro de un rango numérico específico. Entonces, en la Tabla 7, se detallan los criterios de evaluación de calidad y los puntajes relacionados con este trabajo de investigación.

Tabla 7

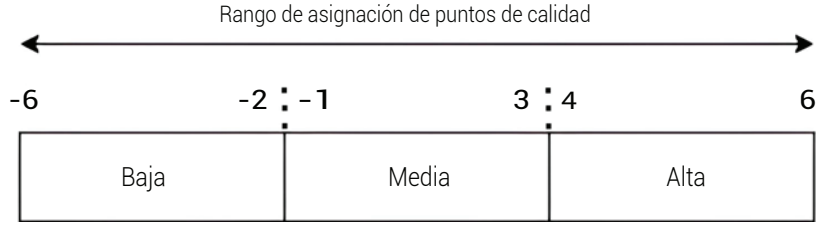
Criterios de evaluación de calidad definidos para el presente trabajo de investigación

Clave	Criterio de evaluación	Puntaje	Descripción
QA1	¿El artículo es referenciado por otros estudios de otros autores que guardan relación en la materia con este?	1	2 artículos o más
		0	1 artículo
		-1	Ningún artículo
QA2	¿El artículo responde de forma apropiada las preguntas de investigación?	1	Sí, todas
		0	Al menos la mitad
		-1	No, ninguna
QA3	¿El artículo hace uso de fundamentos técnicos comprobables y basándose en estándares de ingeniería de *software*?	1	Sí
		0	Parcialmente
		-1	No
QA4	¿El artículo se encuentra sesgado por intereses particulares o comerciales?	1	No
		0	Parcialmente
		-1	Sí
QA5	¿El artículo menciona problemas, limitaciones o lecciones aprendidas como parte de sus conclusiones?	1	Sí
		0	Parcialmente
		-1	No
QA6	¿El artículo propone la realización de futuras líneas de investigación alineados al objetivo de este?	1	Sí
		0	Parcialmente
		-1	No

Asimismo, debe considerarse que todos los criterios de evaluación cuentan con el mismo peso. Esto quiere decir que cada criterio cuenta con una ponderación media de 1/6 puntos. En la Figura 1, se detalla el rango de asignación de puntos de calidad seleccionado.

Figura 1

Rango de asignación de puntos de calidad



De la Figura 1 puede interpretarse que un artículo, cuya evaluación de calidad se encuentre en el rango comprendido entre -6 y -2 puntos, será tomado como de baja calidad; mientras que, si se halla en el rango de -1 a 3, será tomado como de calidad media; y, finalmente, dentro del rango entre 4 y 6, será de calidad alta. Para el presente trabajo de investigación, se ha optado incluir todos los estudios cuya calidad supere el valor 1. Es decir, cualquier estudio de calidad media o superior cuenta con un nivel de calidad lo suficientemente óptimo como para ser incluido en la lista de artículos primarios de la revisión.

Aplicación de proceso de selección

Al comenzar a trabajar con los filtros de selección Rf0 y Rf1, definidos en la Tabla 5, y luego de realizar varias pruebas de ejecución de las cadenas de búsqueda en las fuentes de consulta académica consideradas, se observaron varios problemas que dificultaban el tratamiento homogéneo de la información recabada de lo siguiente:

- La mayoría de las herramientas provistas por las fuentes de consulta carecían de mecanismos de evaluación booleana que utilizan operadores avanzados como NOT, NAND, NOR o XOR, y se limitaron al uso de operadores simples tales como AND y OR, así como el uso de paréntesis para separar palabras y sentencias para generar agrupaciones. Esto dificultó la generación de expresiones booleanas complejas o el uso de reglas de álgebra booleana.
- El proceso de búsqueda se limitó a consultas directas hacia las bases de datos y no solían considerar el uso mecanismos o filtros contextuales en las expresiones. Es decir, no brindaron herramientas para indicar la relación contextual entre palabras clave, como la distancia mínima o máxima requerida entre estas o bien la búsqueda por aproximación, semejanza, pluralización o concordancia entre palabras o sentencias.
- El criterio de agrupamiento por disciplina o subdisciplina no solía ser estándar entre las diversas opciones, por lo que existía la posibilidad de que algunos artículos no fueran incluidos en los resultados de las búsquedas al utilizar dichos filtros.
- En muchos casos existía una limitación de la cantidad máxima de operadores permitidos en una cadena de búsqueda booleana. Esto limitó sensiblemente la correspondencia de la cadena de búsqueda deseada por sobre la calidad y cantidad de resultados esperados.
- No todas las fuentes permitieron la búsqueda sobre los mismos campos. Por lo general, se intentó hacer foco inicialmente sobre los campos más representativos de los estudios e intentar que estos fueran los mismos en todas las fuentes, como el título, el resumen y las palabras clave. Esto solía

denominarse TAK (*title*, *abstract* y *keywords*), tal como se ejemplifica en Hinderks et al. (2022, p. 5).

- En determinadas situaciones, y sin un patrón aparente, algunas fuentes de consulta muestran comportamientos erráticos, fallas en sus interfaces de usuario y, en menor medida, los resultados obtenidos pueden no ser determinísticos.
- La cantidad de resultados máximos retornados puede verse fija a un valor arbitrario lo que provoca la necesidad de redefinir la cadena de consulta o los filtros para adaptarse al caso en particular. Esto deriva en un sesgo sobre los resultados y una incompatibilidad entre las diversas fuentes de búsqueda.

Entonces, con el fin de mitigar estos problemas, se concluyó que resultaba necesario implementar un mecanismo genérico que permitiera obtener los resultados por medio de algún método de exportación, desde las fuentes de búsqueda en línea, para luego ser manipulados programáticamente mediante alguna herramienta de *software ad hoc*, y así contar con resultados normalizados y facilitar su manipulación de forma centralizada.

Después de una verificación de las fuentes de consulta seleccionadas, se observó que, en todos los casos, era posible exportar los resultados a diversos formatos estándares como archivos BibTeX, los cuales contienen listas con registros en formato de estructura de datos (BibTex, s. f.), o bien archivos CSV (Shafranovich, 2005), que contienen listas con registros compuestos por valores separados por comas. Por lo tanto, los autores decidieron implementar la herramienta TFEHelper (Simoncini, 2024), cuyo objetivo era la colección y curación de referencias de artículos académicos. Con esta herramienta fue posible lo siguiente:

- *Importar y exportar*: contenido en formato BibTeX y CSV.
- *Aplicar filtros*: ejecutar consultas avanzadas sobre los datos importados y realizar búsquedas por distancia entre palabras por medio de la expresión lógica NEAR / ONEAR, que permite buscar texto que cumpla con la condición de que dos palabras se hallen separadas por una cantidad específica de palabras intermedias.
- *Interoperar*: acceder de forma homogénea a fuentes de consulta bibliográfica por medio de API públicas para colectar referencias (arXiv, CrossRef, Doaj, Eric, Pubmed, Scopus y Springer Link). De esta manera, se evitó acceder a los sitios web de las fuentes para recolectar manualmente los datos.

Entonces, el nuevo proceso de selección inició a partir de la ejecución de la cadena de búsqueda simple en las fuentes académicas de literatura (filtro Rf0). Tomando las palabras clave indicadas en la Tabla 2, sobre la base de recomendaciones académicas

como en Aliyu (2017) y aplicado un proceso de mejora y refinamiento, se obtuvo la cadena de búsqueda de la Tabla 8, con la cual se consiguió un total de 35 005 resultados.

Tabla 8

Cadena de búsqueda

(software AND development AND framework) OR (software AND development AND framework AND ontology AND domain)

Estos resultados fueron exportados al formato CVS y BibTeX para luego ser importados (manualmente o a través de la API correspondiente) en TFEHelper. Haciendo uso de esta herramienta, se filtró Rf1 y se aplicó el filtro NEAR/ONEAR con una distancia máxima de dos palabras para los campos título y resumen. Una vez aplicados estos filtros, se obtuvieron 1142 artículos, que fueron luego procesados para llevar a cabo las fases de filtrado Rf2, Rf3 y QA. Para ello, se decidió utilizar la herramienta Parsifal (Freitas, s. f.), por lo que los datos generados por TFEHelper fueron exportados al formato BibTeX para que, a su vez, pudieran ser importados en Parsifal. Una vez importados los artículos primarios con la herramienta Parsifal, se volcaron de forma manual los contenidos distintivos de cada estudio en la lista de campos definida en la Tabla 9 y se calculó el puntaje de calidad correspondiente. De esta manera, se obtuvieron los once artículos primarios que se detallan a continuación:

- "A Framework for Syntactic and Semantic Quality Evaluation of Ontologies" (Iyer et al., 2021)
- "A Reference Architecture for Ontology Engineering Web Environments" (Braun et al., 2019)
- "A Review of Approaches to Detecting Software Design Patterns" (Assad & Avksentieva, 2024)
- "Application Framework Development and Design Patterns: Current State and Prospects" (Jurišić & Kermek, 2014)
- "Design, Implementation and Evolution of Object Oriented Frameworks: Concepts and Guidelines" (Van Gurp & Bosch, 2001)
- "EDON: A Method for Building an Ontology as Software Artefact" (Reynares et al., 2012)
- "Implementación de una arquitectura de *software* guiada por el dominio" (Cambarieri et al., 2020)
- "Introduction to a Framework for Multi-Modal and Tangible Interaction" (Lo et al., 2010)

- "Models and Frameworks: A Synergistic Association for Developing Component-Based Applications" (Alonso et al., 2014)
- "Ontologías en arquitecturas dirigidas por modelos" (Lopez et al., 2011)
- "Software Framework Construction Based on Plug-In Technology" (Lei et al., 2011)

Finalmente, luego de volcar las referencias en los campos del formulario (Tabla 9) y obtener los once artículos primarios, estas fueron importadas a la herramienta Zotero (s. f.) para ser utilizadas en la documentación del análisis de las secciones 4 y 5 de este artículo.

Tabla 9

Lista de campos del formulario de extracción de datos

Nombre	Descripción
Fuente	Lista de fuentes académicas desde donde se obtiene el estudio. Para este caso son los anteriormente mencionados: IEEE Xplore, ACM Digital Library, Springer Link, Science Direct y SEDICI. Asimismo, se incluye la fuente manual y *snowballing*.
Autores	Los autores principales del estudio
Fecha de publicación	La fecha de publicación del estudio
País de publicación	El país de origen de la publicación
Institución de la publicación	Lista de instituciones a las cuales el estudio hace referencia. Puede darse el caso de que el artículo haya sido desarrollado en colaboración entre varias instituciones.
Referencias importantes	Lista de referencias que, en el contexto en el que fueron citadas, resultan relevantes para el estudio por lo que es recomendable ser analizadas. De aquí se desprenden potenciales artículos candidatos a convertirse en primarios para el presente trabajo de investigación. A esta técnica se la denomina *backward snowballing* (Wohlin, 2014).
Enfoque	Los dos posibles tipos de enfoque del artículo en el cual el estudio está interesado: "dominio y ontología" o bien "tecnología subyacente"
Orientación	Lista de posibles orientaciones temáticas del artículo, como, por ejemplo: "arquitectura de *software*", "desarrollo de *software*", "diseño de *software*" o "gestión de las configuraciones"
Ejemplificación	Indica la forma en que el artículo ejemplifica de forma práctica su contenido, por ejemplo: "código fuente", "modelado conceptual" o "ninguno".
Patrones de arquitectura referenciados	Breve descripción del patrón, o los patrones, de arquitectura mencionados en el artículo, si los hubiere
Frameworks referenciados	Breve descripción de los *frameworks* de desarrollo de *software* mencionados en el artículo, si los hubiere

(*continúa*)

(*continuación*)

Nombre	Descripción
Lenguajes de programación referenciados	Breve descripción de los lenguajes de programación mencionados en el artículo, si los hubiere
Respuestas a preguntas de investigación	Se compone de cinco campos, uno para cada pregunta de investigación (es decir, de RQ1 a RQ5) en donde, brevemente, se resumen las ideas y posibles respuestas a ser revisadas más adelante.
Respuestas a análisis de calidad	Se compone de seis campos, uno para cada pregunta de calidad (es decir, de QA1 a QA6) en donde se indica el puntaje específico para cada una.
Puntaje de calidad	Sumatoria del valor de los puntos de calidad

A modo de resumen, en las figuras 2 y 3, se puede observar la hoja de ruta completa que resume de forma gráfica todo el proceso aplicado para la selección de los datos para el presente trabajo de investigación.

Figura 2

Hoja de ruta de la selección de datos

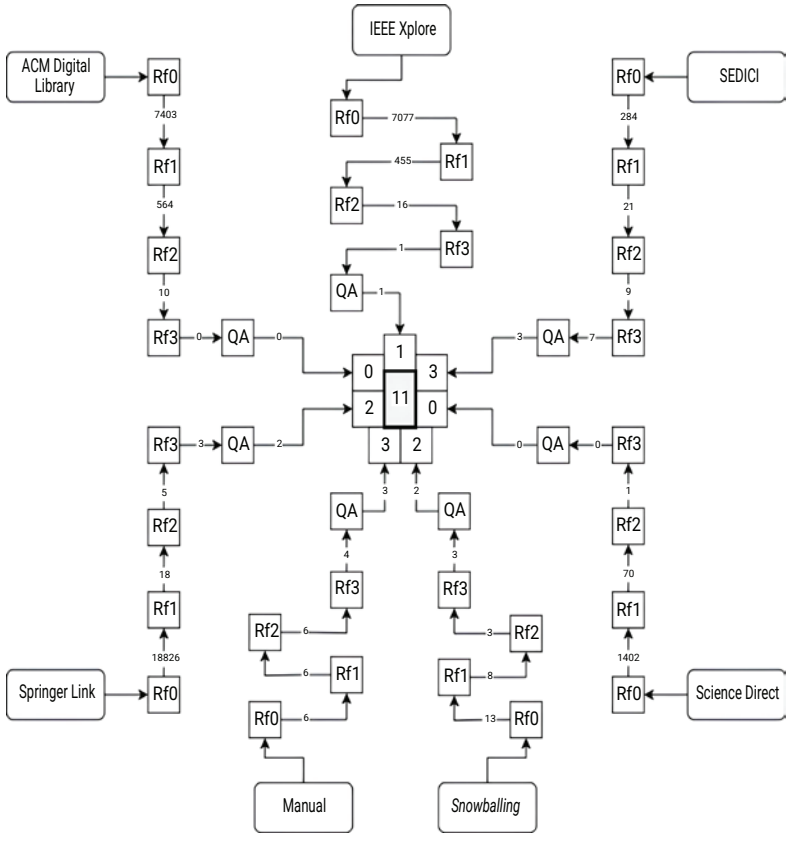
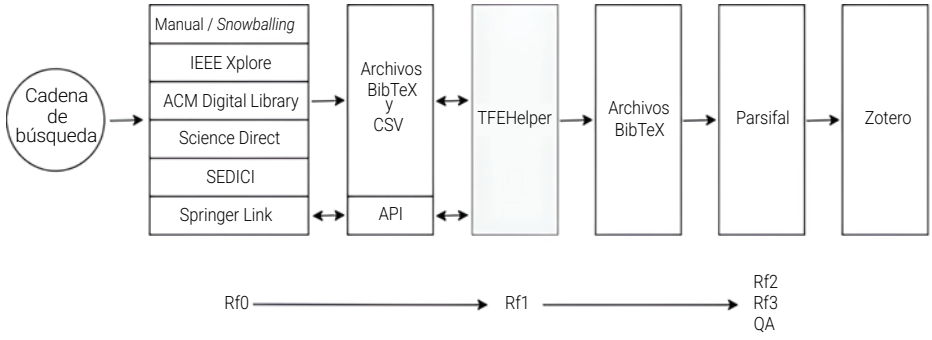


Figura 3

Proceso de selección completo



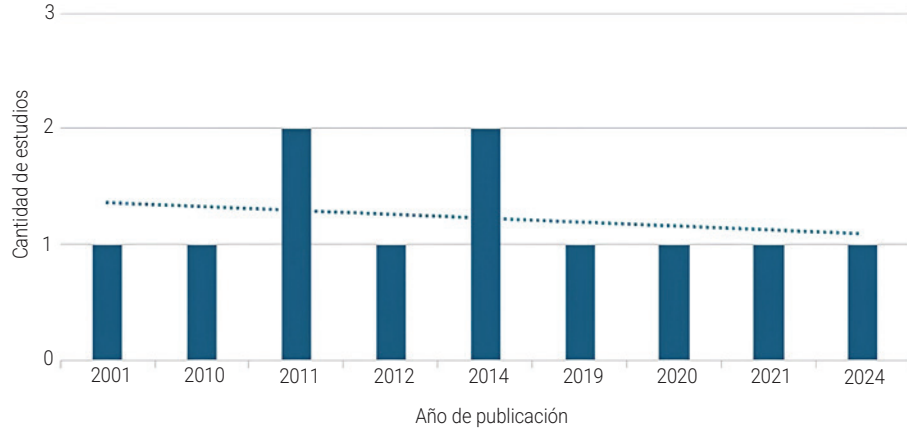
RESULTADOS

En esta sección se presenta un resumen de los principales hallazgos de la aplicación de los pasos de la revisión sistemática explicada en la sección anterior.

En la Figura 4, se expone la distribución de los estudios primarios obtenidos en la revisión sistemática según su año de publicación. Esto permitió realizar un análisis de la evolución cronológica de las publicaciones en el campo de estudio. Como puede observarse, la cantidad de artículos primarios obtenidos marcó una tendencia decreciente a través de los años al observar la línea de puntos. Esto explica la notable carencia de interés en el tema de investigación en el campo académico.

Figura 4

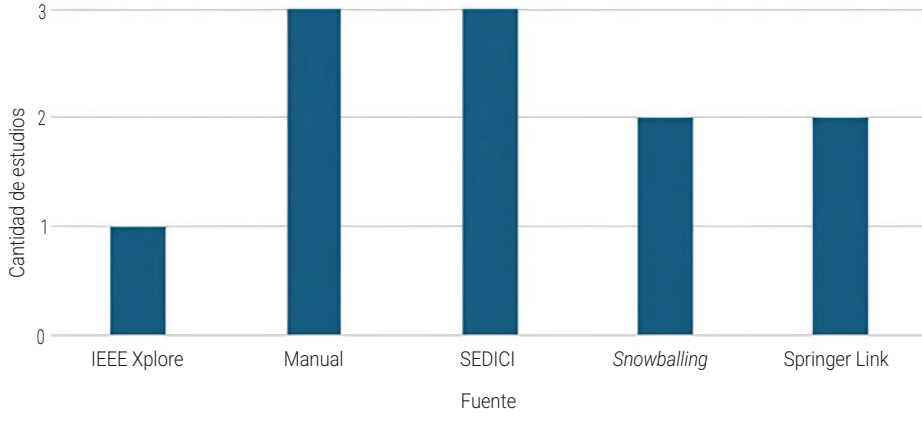
Distribución de estudios por año de publicación



Por otro lado, en la Figura 5, puede observarse la distribución de artículos por fuente de consulta tomada de los resultados obtenidos. En este caso, llamó la atención la cantidad de artículos obtenidos por fuentes no convencionales como manual y *snowballing*. Se observa que la suma de la cantidad de artículos de ambas fuentes compone aproximadamente el 45 % del total de los artículos primarios. Esto evidencia el alto grado de complejidad del tema de investigación del presente trabajo, así como una limitada información disponible.

Figura 5

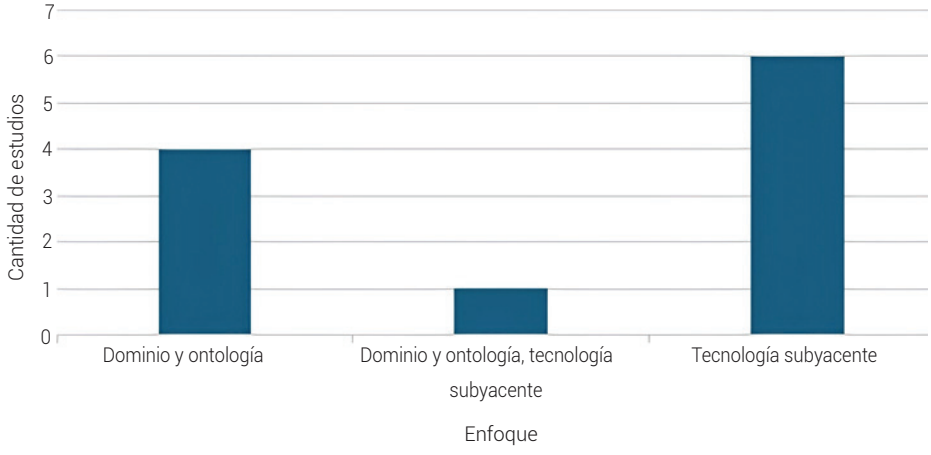
Distribución de estudios por fuente de consulta



Desde la perspectiva de las preguntas de investigación, en la Figura 6, puede observarse la clasificación de los artículos obtenidos por el enfoque RQ1 (tecnología subyacente), RQ2 (dominio y ontología) y RQ3 (combinación de ambas). Puede notarse la poca cantidad de artículos que abordan la combinación de ambos enfoques en contraposición con la cantidad de estudios que lo hacen separadamente. Esto explicaría la existencia del área de vacancia en lo referente a *frameworks* de desarrollo de *software* orientados al diseño de *backends* agnósticos a la tecnología subyacente en conjunto con la utilización de dominios de aplicación con fundamentos ontológicos.

Figura 6

Distribución de estudios por tipo de enfoque



Por último, en la Tabla 10, se analiza la calidad de los artículos recabados indicando de forma discriminada los valores obtenidos del cálculo de los puntajes de los criterios indicados en la Tabla 7. A partir de los valores obtenidos, se pudo detectar que la mayoría de los artículos fue referenciada por otros artículos relacionados con la materia de estudio, por lo que se utilizaron fundamentos técnicos comprobables basándose en los estándares de la ingeniería de *software*; además, no se encontró sesgado por intereses particulares o comerciales y se propusieron futuras líneas de investigación alineadas a la materia. Sin embargo, también se pudo ver que muy pocos artículos respondieron de forma apropiada las preguntas de investigación ni se mencionaron problemas, limitaciones o lecciones aprendidas como parte de sus conclusiones. Este hecho parece indicar la existencia de un área de vacancia para el tema de investigación propuesto en este documento.

Tabla 10

Lista de campos del formulario de extracción de datos

Criterio de evaluación	Suma de puntos	Porcentaje de artículos
QA1: ¿el artículo es referenciado por otros estudios de otros autores que guardan relación en la materia con este?	6	77,27
QA2: ¿el artículo responde de forma apropiada las preguntas de investigación?	-1	45,45
QA3: ¿el artículo hace uso de fundamentos técnicos comprobables y basándose en estándares de ingeniería de *software*?	10	95,45

(*continúa*)

(*continuación*)

Criterio de evaluación	Suma de puntos	Porcentaje de artículos
QA4: ¿el artículo se encuentra sesgado por intereses particulares o comerciales?	10	95,45
QA5: ¿el artículo menciona problemas, limitaciones o lecciones aprendidas como parte de sus conclusiones?	-5	27,27
QA6: ¿el artículo propone la realización de futuras líneas de investigación alineadas al objetivo de este?	4	68,18

DISCUSIÓN

Sobre la base de los datos extraídos, en esta sección se realiza un análisis que permite dar cuenta de las publicaciones identificadas para responder las preguntas de investigación definidas.

RQ1: abstracción de tecnología subyacente

La pregunta RQ1 plantea la necesidad de poder identificar cuál es el estado del arte vinculado al diseño de *frameworks* para la implementación de modelos de *backends* que permitan abstraer la tecnología subyacente indistintamente de la taxonomía. Ante ello, la revisión de la literatura reveló que existen principalmente tres enfoques aplicados a la abstracción de tecnología subyacente en el diseño de *frameworks* de desarrollo orientados al *backend*, los cuales son descritos a continuación:

- *Arquitectura basada en componentes*: también conocida como ingeniería de *software* basada en componentes (*component-based software engineering*, CBSE) o bien como desarrollo orientado a componentes (*component-based development*, CBD) (Heineman & Councill, 2001; Szyperski et al., 2009). Esta metodología cumple con el objetivo de proveer mecanismos para lograr la separación de conceptos por medio del encapsulamiento de las piezas de *software* y la comunicación entre sí por medio de interfaces o contratos específicos. Con esto se logra un bajo nivel de acoplamiento y se incrementa la reutilización. La recomendación por el uso de componentes se refuerza aún más gracias a las evidencias halladas en los artículos primarios donde se referencia el uso de determinados patrones de diseño, tales como *factory* (Gamma et al., 2011, p. 87) y *flyweight* (Gamma et al. 2011, p. 195) (también conocidos como patrones GOF).
- *Arquitectura basada en complementos*: se trata de un patrón arquitectónico de *software*, también conocido como *plugin* (Evans, 2004, p. 475), que tiene como fin brindar la capacidad de interacción entre diversas tecnologías encapsuladas en componentes y basadas en las mismas abstracciones, pero

diseñadas de forma independiente. Por lo general, dichos componentes se conectan a un concentrador central o núcleo que admite los protocolos que necesitan y sabe cómo comunicarse con las interfaces que proporcionan. Esta arquitectura difiere de la orientada a componentes en el hecho de no tener que compartir el núcleo en un mismo espacio de direcciones, sino más bien utilizarlo por medio de contratos o interfaces a través de una librería externa. La incorporación o eliminación de los complementos a la estructura del sistema, por norma general, no debe implicar el reacondicionamiento del núcleo cada vez (es decir, no se requiere rediseño ni reprogramación).

- *Arquitectura basada en capas:* también conocida como estilo de arquitectura n-capas (Richards & Ford, 2020, capítulo 10). Toma las bases establecidas por la arquitectura basada en componentes y hace que estos se agrupen y organicen en conjuntos (capas) horizontales lógicos, cada uno desempeñando de forma aislada una función específica dentro de la aplicación (como lógica de presentación, lógica de negocios o lógica de persistencia de datos). Los artículos primarios obtenidos en el presente trabajo de investigación hacen mención a evoluciones de la arquitectura basada en capas, tales como las arquitecturas hexagonales (también conocidas como "puertos y adaptadores") (Cockburn, 2005; Cockburns & Garrido de Paz, 2024)y las arquitecturas *clean* (Martin et al., 2018).

RQ2: utilización de dominios de aplicación

La RQ2 plantea la necesidad de poder identificar cuál es el estado del arte vinculado al modelado de dominios de aplicación ontológicamente válidos.

En relación con el enfoque de los *frameworks* de desarrollo orientados a *backends* sobre la aplicación de dominios, puede observarse como principal a las arquitecturas dirigidas por modelos (*model driven architecture*, MDA), que se trata de una metodología que proporciona un conjunto de guías y normas para estructurar especificaciones expresadas como modelos (Standards Development Organization, s. f.). Tiene como premisa la definición de la funcionalidad de un sistema por medio de la creación de modelos independientes de la plataforma (*platform independent model*, PIM) a través de un lenguaje específico para el dominio en particular. Los modelos PIM pueden luego traducirse a modelos específicos de la plataforma (*property management system*, PMS) por medio de herramientas automatizadas. MDA persigue como objetivo mantener separado el diseño de la arquitectura de las tecnologías de construcción y facilita que ambos puedan ser alterados independientemente. En este sentido (Gašević et al., 2006), indica que es posible hacer uso de ontologías en la combinación de MDA y dominios de aplicación; además, refuerza esta teoría proponiendo un marco de trabajo cuyo punto de partida es una ontología y que, por medio de transformaciones, es convertida en un modelo (Lopez et al., 2011).

En el contexto del diseño de *frameworks* de desarrollo de *software* para *backends*, la inclusión de la gestión de dominios de aplicación facilita la adopción de los modelos necesarios en la creación de aplicaciones. Tal como indica Evans (2004, capítulo 4), esto puede realizarse por medio de la inclusión de una capa específica en la que el dominio pueda ser completamente abstraído, tal como propone en su patrón arquitectónico DDD. Idealmente, un *framework* de desarrollo de *software* podría proveer un modelo base (abstracto) de dominio para ser utilizado por la implementación específica, como así también proveer de los mecanismos necesarios para verificar su validez ontológica.

En los artículos primarios recabados, no resulta posible hallar vínculos de forma directa entre la utilización de ontologías en los dominios de aplicación y los *frameworks* de desarrollo. No obstante, diversos estudios presentan variados enfoques en relación con el tratamiento ontológico de dominios de aplicación que pueden ser utilizados para tal fin.

En Iyier et al. (2021), se reconoce como óptimo el análisis de calidad de las ontologías por medio del análisis sintáctico (capacidad de adherencia a las pautas o reglas de desarrollo de una ontología) y semántico (validez de conceptos y relaciones). Toma como premisa que un dominio deba definirse en lenguaje de ontologías (*ontology language overview*) (W3C, s. f.) y propone realizar su evaluación de forma semiautomática por medio del desarrollo de un *framework* materializado en dos aplicaciones de *software*.

Desde la perspectiva sintáctica, define la herramienta SynEvaluator que utiliza un marco de creación de reglas de validación sintáctica que permite a los usuarios definirlas de forma no programática durante el tiempo de ejecución y establecer prioridades específicas para cada tarea. Respecto a la semántica, define la herramienta SemValidator que utiliza el *crowdsourcing* e inteligencia artificial para validar la calidad semántica de las ontologías de acuerdo con los resultados obtenidos.

Por su parte, Reynares et al. (2012) plantea la definición de ontologías a través de una metodología iterativa denominada *evolutionary development of ontologies* (EDON). El resultado de esta metodología se observa en componentes reutilizables y ontológicamente validados. Esta metodología propone desarrollar de forma recurrente una ontología que cumpla con los requisitos del ciclo de desarrollo al que pertenece. Partiendo de la conceptualización de una ontología haciendo uso de técnicas para la elicitación de los requisitos, como preguntas de competencia (*competency questions*, CQ) y lenguaje léxico extendido (*language extended lexicon*, LEL), se obtiene la información necesaria sobre el dominio que luego se captura como objetos, relaciones y propiedades usando herramientas destinadas a la creación de ontologías, como Protégé (Stanford Center for Biomedical Informatics Research, s. f.). Finalmente, las salidas de dichas herramientas son traducidas a lenguaje de programación para luego ser importadas en los proyectos de *software* específicos.

RQ3: abstracción de tecnología subyacente y uso de dominios de aplicación

La RQ3 plantea la necesidad de validar la existencia de casos de estudio que respondan las preguntas RQ1 y RQ2 de forma combinada. Es decir, estudios que abarquen el estado del arte vinculado al diseño de *frameworks* para la implementación de modelos de *backends* que permitan abstraer la tecnología subyacente indistintamente de la taxonomía y que, a su vez, contemplen el modelado de dominios de aplicación ontológicamente válidos.

No obstante, a partir del proceso de análisis aplicado sobre los once artículos resultantes de la búsqueda, no ha sido posible encontrar casos en los que se desarrolle la combinación planteada en la pregunta de investigación RQ3. Esto indica que, al momento de la confección del presente documento de investigación, no hubo artículos académicos disponibles que cumplieran con los criterios de búsqueda establecidos para validar la existencia de estudios que abarcaran tal combinación.

Si bien este trabajo de investigación se enfoca en mantener los lineamientos establecidos por un protocolo asociado a una revisión sistemática de literatura formal, la escasez de artículos académicos presenta como oportunidad considerar la existencia de artículos producidos por profesionales en la ingeniería de sistemas de información o ingeniería de *software* que se hallen fuera de los repositorios académicos y, por ende, sin referato. Para ello, se toma como guía lo propuesto por Vahid Garousi et al. (2019), con el fin de incluir evidencias obtenidas de la literatura gris en el campo profesional sin afectar los resultados y la calidad del trabajo de investigación.

Por lo anterior, se tomaron las siguientes consideraciones para la búsqueda de literatura producida en el campo profesional:

- artículos que respondan a la cadena de búsqueda establecida en la sección 3.2
- artículos que cuenten con una antigüedad no superior a los diez años
- artículos cuyo contenido pueda ser reproducido y validado empíricamente, así como refiera a tecnologías o mecanismos conocidos y demostrables
- autores que cuenten con más de cincuenta seguidores en la plataforma donde se publica el artículo o bien cuente con al menos cinco artículos publicados
- fuentes o repositorios conocidos en el campo profesional

A continuación, se evidencian algunos casos obtenidos fuera del marco académico.

En Negrin (2024), el autor demostró cómo es posible combinar el uso de Spring (s. f.), un *framework* de desarrollo para el lenguaje de programación Java utilizando los lineamientos establecidos por DDD (Evans, 2004) y la arquitectura hexagonal

(Cockburn, 2005; Cockburn & Garrido de Paz, 2024). El *framework* Spring ofrece abstracción de tecnología subyacente por medio de sus componentes, mientras que el uso de DDD, entre otras cosas, provee un mecanismo de encapsulamiento de dominios de aplicación. Si bien el artículo puede ser tomado como evidencia para responder la pregunta de investigación RQ3, resulta importante destacar que no menciona mecanismos para la validación de dominios desde la perspectiva de la ontología.

Por otro lado, Martins (2024) explica cómo la empresa Mercado Libre ha evolucionado su mecanismo para el proceso de desarrollo de *software* utilizando el modelo PaaS. De esta manera, ha logrado proveer a sus ingenieros no solo las herramientas necesarias por medio de *frameworks* de desarrollo de *software* para garantizar el encapsulamiento de la tecnología subyacente al momento de generar las aplicaciones, sino también una plataforma centralizada para el gobierno de la infraestructura de forma agnóstica al proveedor de servicios de la nube. Si bien el artículo no menciona la gestión de dominios, la empresa cuenta con una alianza estratégica con la empresa OpenAI, lo que probablemente posibilite considerar dicho aspecto en el futuro.

Finalmente, tenemos a ASP.NET Boilerplate (s. f.), un *framework* de desarrollo de *software* para la plataforma .NET que hace uso de la arquitectura multicapa y se encuentra basado en los principios establecidos por DDD, lo que facilita la gestión de dominios de aplicación. Si bien la documentación del *framework* no especifica soportar la abstracción de tecnología subyacente, esto puede ser implementado gracias a que cuenta con la capacidad de inyección de dependencias de forma nativa. Asimismo, provee la capacidad de conectarse al ORM nativo de .NET (*entity framework*), por lo que, de forma indirecta, provee un mecanismo de abstracción al acceso a la base de datos. Al igual que en los casos anteriores, el *framework* no provee mecanismos para validar ontológicamente los dominios utilizados, pero, gracias a su extensibilidad, esta característica podría ser implementada en un futuro.

RQ4: propuestas de mejora o lecciones aprendidas

La RQ4 plantea la necesidad de analizar los casos que cumplan RQ3 con el fin de identificar recomendaciones para futuras líneas de investigación en la materia, es decir, en los artículos analizados que cumplan con la combinación de estudios relacionados a *frameworks* de desarrollo de *software* capaces de posibilitar la abstracción de tecnología subyacente y contar con el uso de dominios de aplicación ontológicamente válidos en el diseño de *backends*.

Tal como se indicó anteriormente, no ha sido posible encontrar casos en los que se desarrolle la combinación planteada en la pregunta de investigación RQ3, lo que impide el desarrollo formal de esta sección. No obstante, se plantea como alternativa analizar las propuestas de lecciones aprendidas de los artículos analizados agrupando los que corresponden a las preguntas RQ1 y RQ2 de forma separada.

Respecto a la abstracción de tecnología subyacente (RQ1), Cambarieri et al. (2020) plantean la necesidad de un cambio revolucionario en la forma de escribir *software*, en donde prime el enfoque en dominios, DDD y arquitecturas hexagonales, con el fin de agilizar los tiempos en el desarrollo de *software*. No obstante, tanto Cambarieri et al. (2020) como Van Gurp y Bosch (2001) reconocen que la implementación de *frameworks* de desarrollo de *software* puede ser desafiante como también puede presentar dificultades al momento de ser utilizado en *software* preexistente, por lo que se plantea que los *frameworks* sean definidos como pequeños módulos o componentes combinables, de acuerdo con las necesidades particulares de los diseñadores de *software*.

Respecto a la utilización de dominios de aplicación (RQ2), todos los estudios coinciden en la recomendación de definir los dominios como piezas de *software* independientes capaces de ser utilizados por las aplicaciones por medio de una interfaz común y estandarizada. Sin embargo, Reynares et al. (2012) reconocen que, en su caso de estudio, no cuentan con las pautas suficientes como para agrupar y clasificar correctamente los modelos y validar las ontologías. Asimismo, Cambarieri et al. (2020) plantean mejorar el suyo utilizando lenguajes específicos de dominio (*domain specific languages*, DSL) y la programación generativa (*generative programming*, GP) para contar con un enfoque de reutilización proactivo, de familias de productos relacionados por algún dominio en particular, en lugar de productos independientes.

RQ5: conclusiones obtenidas de los estudios primarios

Finalmente, la RQ5 plantea la necesidad de validar si existe la posibilidad de proponer futuras líneas de investigación relacionadas al diseño de *frameworks* para *backends* que abstraigan la tecnología subyacente y hagan uso de dominios de aplicación ontológicamente válidos.

De los artículos primarios obtenidos se puede concluir lo siguiente:

- El interés en el marco académico en los temas planteados en el presente trabajo de investigación ha disminuido a través de los años, mientras que, en el marco profesional, se ha incrementado notoriamente, lo que deja un valle temporal carente de información formal y actualizada sobre la materia.
- Los sistemas modernos no utilizan ontologías debido a diversas causas, por ejemplo, complejidad, evolución continua del negocio, la no existencia de mecanismos estandarizados, no contar con *frameworks* de desarrollo que promuevan su utilización o bien por desconocimiento en la materia.
- El uso de inteligencia artificial puede ayudar significativamente en la ejecución de tareas para las cuales se requiere experiencia en campos particulares, como los dominios de aplicación y la validación de las ontologías.

Por todo lo anterior, se ha detectado como área de vacancia la definición de un *framework* de desarrollo orientado a la creación de aplicaciones empresariales que combinen la abstracción de tecnología subyacente con el uso de dominios de aplicación, y que tengan como característica la capacidad de validar los dominios de aplicación por medio de ontologías.

CONCLUSIONES

En cuanto a la implementación de la abstracción de tecnología subyacente, los estudios primarios obtenidos indican que los *frameworks* de desarrollo de *software* podrían hacer uso de arquitecturas basadas en componentes (Alonso et al., 2014, Asaad & Avksentieva, 2024; Lo et al., 2010; Van Gurp & Bosch, 2001), complementos (Lei et al. 2011) o bien en capas (Cambarieri et al., 2020; Jurišić & Kermek, 2014; Lo et al., 2010). De acuerdo con la cantidad de referencias y calidad asociada a los artículos que la refieren, la primera opción resulta ser la más válida.

Respecto a la gestión de dominios de aplicación, se observa como recomendación el uso de arquitecturas dirigidas por modelos (Lopez et al., 2011) o componentes (Reynantes et al., 2012) y la implementación de algún mecanismo que permita la ejecución de análisis sintáctico y semántico en los dominios (Braun et al., 2019; Iyer et al., 2021), aunque la cantidad y calidad de los artículos pudiera no ser suficiente para tal conclusión.

No se han hallado artículos primarios que aborden la combinación de ambos enfoques, por lo que no puede inferirse recomendaciones al respecto. Asimismo, queda evidenciado que la investigación en este campo se ha visto mermada a través de los años en contraposición con la gran adopción de las tecnologías relacionadas, por lo que resulta importante aprender las causas.

Se observan las siguientes propuestas de investigación:

- estudios de casos y experiencias en la utilización de *frameworks* de desarrollo de *software* propietario en el ambiente empresarial
- estudios sobre la usabilidad de ontologías en la ingeniería del *software* en la actualidad
- exploración del uso de la inteligencia artificial para la validación de dominios y modelos por medio de ontologías

Tal como se adelantó en el acápite sobre RQ5, se detectó como área de vacancia la definición de los lineamientos necesarios para la creación de *frameworks* de desarrollo de *software* orientados a *backends* que permitan abstraer el uso de tecnología subyacente y que cuenten con la capacidad de validar dominios de aplicación por medio de ontologías.

Por otro lado, se observa como oportunidad el desarrollo de un trabajo de investigación que tenga como objetivo definir formalmente un modelo de *framework* de desarrollo de *software* basado en el patrón arquitectónico hexagonal y limpio, orientado a componentes y DDD que pueda ser tomado como base para la generación de *frameworks* de desarrollo para la construcción de *software backend* en el ámbito empresarial. La definición de dicho *framework* de desarrollo podría contar con la capacidad de brindar herramientas para la abstracción de tecnología subyacente, así como el tratamiento de dominios de aplicación y las correspondientes validaciones ontológicas.

Finalmente, resulta evidente que quedan amplias oportunidades para que futuros estudios continúen aportando conocimientos relevantes, tanto para la comunidad científica como para la práctica profesional.

REFERENCIAS

Aliyu, M. (2017). Efficiency of Boolean search strings for information retrieval. *American Journal of Engineering Research*, *6*(11), 216-222. https://www.ajer.org/papers/v6(11)/ZA0611216222.pdf

Alonso, D., Sánchez-Ledesma, F., Sánchez, P., Pastor, J. A., & Álvarez, B. (2014). Models and frameworks: A synergistic association for developing component-based applications. *The Scientific World Journal*, *2014*(1), 687346. https://doi.org/10.1155/2014/687346

Alrumaih, H., Mirza, A., & Alsalamah, H. (2020). Domain ontology for requirements classification in requirements engineering context. *IEEE Access*, *8*, 89899-89908. https://doi.org/10.1109/ACCESS.2020.2993838

Asaad, J., & Avksentieva, E. (2024). A review of approaches to detecting software design patterns. *2024 35th Conference of Open Innovations Association (FRUCT)*, 142-148. https://doi.org/10.23919/FRUCT61870.2024.10516345

ASP.NET Boilerplate. (s. f.). *Web application framework*. https://aspnetboilerplate.com/

BibTeX. (s. f.). *Your BibTeX resources*. https://www.bibtex.org/

Bjørner, D. (2006). *Software engineering. 3: Domains, requirements, and software design*. Springer.

Braun, G. A., Estevez, E., & Fillottrani, P. (2019). A reference architecture for ontology engineering web environments. *Journal of Computer Science and Technology*, *19*(01), e03. https://doi.org/10.24215/16666038.19.e03

Cambarieri, M., Difabio, F., & García Martínez, N. (2020). Implementación de una arquitectura de *software* guiada por el dominio. En *XXI Simposio Argentino de Ingeniería de Software (ASSE 2020) - JAIIO 49 (Modalidad virtual)* [Simposio]. http://sedici.unlp.edu.ar/handle/10915/115198

Cockburn, A. (2005, 4 de enero). The hexagonal (ports & adapters) architecture. *Alistair Cockburn*. https://alistair.cockburn.us/hexagonal-architecture/

Cockburn, A., & Garrido de Paz, J. M. (2024). *Hexagonal architecture explained. How the ports & adapters architecture simplifies your life, and how to implement it*. Humans and Technology Inc.

Evans, E. (2004). *Domain-driven design: Tackling complexity in the heart of software*. Addison-Wesley.

Freitas, V. (s. f.). *Parsifal* [Registro de datos]. GitHub. https://github.com/vitorfs/parsifal

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (2011). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.

Garousi, V., Felderer, M., & Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology*, *106*, 101-121. https://doi.org/10.1016/j.infsof.2018.09.006

Gašević, D., Djurić, D., & Devedžić, V. (2006). *Model driven architecture and ontology development*. Springer.

Greenhalgh, T., & Peacock, R. (2005). Effectiveness and efficiency of search methods in systematic reviews of complex evidence: Audit of primary sources. *BMJ*, *331*(7524), 1064-1065. https://doi.org/10.1136/bmj.38636.593461.68

Heineman, G. T., & Councill, W. T. (Eds.). (2001). *Component-based software engineering: Putting the pieces together*. Addison-Wesley.

Hinderks, A., Domínguez Mayo, F. J., Thomaschewski, J., & Escalona, M. J. (2022). Approaches to manage the user experience process in agile software development: A systematic literature review. *Information and Software Technology*, *150*, 106957. https://doi.org/10.1016/j.infsof.2022.106957

Iyer, V., Sanagavarapu, L. M., & Reddy, R. (2021). *A framework for syntactic and semantic quality evaluation of ontologies*. En R. Krishnan, H. R. Rao, S. K. Sahaym, S. Samtani & Z. Zhao (Eds), *Secure knowledge management in the artificial intelligence era* (pp. 73-93). https://doi.org/10.1007/978-3-030-97532-6_5

Johnson, R. E., & Foote, B. (1988). Designing reusable classes. *Journal of Object-Oriented Programming*, *1*(2), 22-35.

Jurišić, M., & Kermek, D. (2014). Application framework development and design patterns: Current state and prospects. *Central European Conference on Information and Intelligent Systems*, 306-344.

Kitchenham, B. A., & Charters, S. (2007). *Guidelines for performing systematic literature reviews in software engineering version 2.3*. Keele University. https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf

Lei, G., Yu, F., & Shulin, P. (2011). Software framework construction based on plug-in technology. En *2011 International Conference on Computational and Information Sciences* (pp. 762-764). IEEE. https://doi.org/10.1109/ICCIS.2011.255

Lo, K. W. K., Tang, W. W. W., Ngai, G., Chan, S. C. F., & Tse, J. T. P. (2010). Introduction to a framework for multi-modal and tangible interaction. *2010 IEEE International Conference on Systems, Man and Cybernetics* (pp. 3001-3007). IEEE. https://doi.org/10.1109/ICSMC.2010.5641977

Lopez, G., Servetto, A. C., Echeverría, A., Jeder, I., Grossi, M. D., & Jiménez Rey, E. M. (2011). Ontologías en arquitecturas dirigidas por modelos. En *XIII Workshop de Investigadores en Ciencias de la Computación*. Repositorio Institucional de la UNLP. http://sedici.unlp.edu.ar/handle/10915/20065

Martin, R. C. (2000). Design principles and design patterns. *Object Mentor*, *1*(34), 597. http://labs.cs.upt.ro/labs/ip2/html/lectures/2/res/Martin-PrinciplesAndPatterns.PDF.

Martin, R. C., Grenning, J., Brown, S., & Henney, K. (2018). *Clean architecture: A craftsman's guide to software structure and design*. Prentice Hall.

Martins, J. M. (2024, 26 de febrero). The technological evolution at Mercado Libre. *Mercado Libre Tech*. https://medium.com/mercadolibre-tech/the-technological-evolution-at-mercado-libre-fb269776a4e8

Negrin, J. (2024, mayo 7). Construyendo una RESTful API con Spring Boot: integración de DDD y arquitectura hexagonal. *Medium*. https://medium.com/@juannegrin/construyendo-una-restful-api-con-spring-boot-integraci%C3%B3n-de-ddd-y-arquitectura-hexagonal-af824a3a4d05

Ramirez, M. O. G., De-la-Torre, M., & Monsalve, C. (2019). Methodologies for the design of application frameworks: Systematic review. En *2019 8th International Conference On Software Process Improvement (CIMPS)* (pp. 1-10). IEEE. https://doi.org/10.1109/CIMPS49236.2019.9082427

Reynares, E., Caliusco, L., & Galli, R. (2012). EDON: A method for building an ontology as software artefact. En *XIII Argentine Symposium on Software Engineering (ASSE 2012) (XLI JAIIO, La Plata, 27 al 31 de agosto de 2012)* (pp. 324-338). http://sedici.unlp.edu.ar/handle/10915/124015

Richards, M., & Ford, N. (2020). *Fundamentals of software architecture: An engineering approach*. O'Reilly.

Riehle, D. (2000). *Framework design: A role modeling approach* [Tesis de doctorado, Swiss Federal Institute of Technology Zurich]. Research Collection ETH Zurich. https://doi.org/10.3929/ethz-a-003867001

Shafranovich, Y. (2005). *RFC 4180: Common format and MIME type for comma-separated values (CSV) files*. IETF Datatracker. https://datatracker.ietf.org/doc/html/rfc4180

Simoncini, N. (2024). *TFEHelper: A tool for the collection and curation of academic articles references* [Registro de datos]. GitHub. https://github.com/nicolassimoncini/TFEHelper

Spring. (s. f.). *Spring framework* [*Software*]. https://spring.io/

Standards Development Organization. (s. f.). *MDA Specifications*. http://www.omg.org/mda/specs.htm

Stanford Center for Biomedical Informatics Research. (s. f.). *Protégé* [*Software*]. Universida de Standford. https://protege.stanford.edu/

Stanojevic, V., Vlajic, S., Milic, M., & Ognjanovic, M. (2011). Guidelines for framework developmentprocess. *2011 7th Central and Eastern European Software Engineering Conference (CEE-SECR)*, 1-9. https://doi.org/10.1109/CEE-SECR.2011.6188465

Swacha, J., & Kulpa, A. (2023). Evolution of popularity and multiaspectual comparison of widely used web development frameworks. *Electronics*, *12*(17), 3563. https://doi.org/10.3390/electronics12173563

Szyperski, C., Gruntz, D. W., & Murer, S. (2009). *Component software: Beyond object-oriented programming* (2.ª ed.). Addison-Wesley.

Van Gurp, J., & Bosch, J. (2001). Design, implementation and evolution of object oriented frameworks: Concepts and guidelines. *Software: Practice and Experience*, *31*(3), 277-300. https://doi.org/10.1002/spe.366

W3C. (s. f.). *OWL. Web ontology language. Overview*. https://www.w3.org/TR/owl-features/

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. En M. Shepperd, T. Hall & I. Myrtveit (Eds.), *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering* (pp. 1-10). Association for Computing Machinery. https://doi.org/10.1145/2601248.2601268

Zotero. (s. f.). *Your personal research assistant*. Digital Scholarship. https://www.zotero.org/