software

MDPI

Review

Designing Microservices Using AI: A Systematic Literature Review

Daniel Narváez [1], Nicolas Battaglia [1], Alejandro Fernández [2] and Gustavo Rossi [1,2,*]

[1] CAETI, Facultad de Tecnología Informática, Universidad Abierta Interamericana (UAI), Ciudad Autónoma de Buenos Aires C1270AAH, Argentina; josedaniel.narvaezf@alumnos.uai.edu.ar (D.N.); nicolas.battaglia@uai.edu.ar (N.B.)

[2] Laboratorio de Investigación y Formación en Informática Avanzada (LIFIA), Facultad de Informática, Universidad Nacional de La Plata (UNLP), La Plata 1900, Argentina; alejandro.fernandez@lifia.info.unlp.edu.ar

* Correspondence: gustavo@lifia.info.unlp.edu.ar

Abstract: Microservices architecture has emerged as a dominant approach for developing scalable and modular software systems, driven by the need for agility and independent deployability. However, designing these architectures poses significant challenges, particularly in service decomposition, inter-service communication, and maintaining data consistency. To address these issues, artificial intelligence (AI) techniques, such as machine learning (ML) and natural language processing (NLP), have been applied with increasing frequency to automate and enhance the design process. This systematic literature review examines the application of AI in microservices design, focusing on AI-driven tools and methods for improving service decomposition, decision-making, and architectural validation. This review analyzes research studies published between 2018 and 2024 that specifically focus on the application of AI techniques in microservices design, identifying key AI methods used, challenges encountered in integrating AI into microservices, and the emerging trends in this research area. The findings reveal that AI has effectively been used to optimize performance, automate design tasks, and mitigate some of the complexities inherent in microservices architectures. However, gaps remain in areas such as distributed transactions and security. The study concludes that while AI offers promising solutions, further empirical research is needed to refine AI's role in microservices design and address the remaining challenges.

Keywords: microservices design; artificial intelligence; service decomposition; machine learning; natural language processing; AI in software architecture; microservices performance optimization; AI-driven decision-making; distributed systems; generative AI

Academic Editors: Ian Gorton and Mirko Viroli

Received: 27 January 2025
Revised: 11 March 2025
Accepted: 14 March 2025
Published: 19 March 2025

Citation: Narváez, D.; Battaglia, N.; Fernández, A.; Rossi, G. Designing Microservices Using AI: A Systematic Literature Review. *Software* **2025**, *4*, 6. https://doi.org/10.3390/software4010006

1. Introduction

In recent years, microservices have emerged as the architectural style of choice for building scalable, modular, and independently deployable software systems. This paradigm shift from monolithic to microservices architectures is driven by the need for agility, scalability, and continuous deployment in cloud-based environments [1]. Microservices enable the decomposition of large, complex applications into smaller, autonomous services that can be developed, deployed, and scaled independently, facilitating faster iterations and reduced time-to-market [2].

However, designing microservices architectures presents significant challenges that traditional software engineering methodologies do not fully address. Key activities in

microservices design include identifying individual services, defining their responsibilities and boundaries, and determining their interactions. These tasks involve critical decisions related to service decomposition, communication protocols, and overall architectural structure [3]. Building systems from scratch requires balancing appropriate service granularity with minimizing inter-service communication. Poorly managed designs risk architectural smells, such as excessive coupling or God classes, which hinder maintainability and scalability [4,5].

While approaches such as Domain-Driven Design (DDD) [6] have been widely adopted for defining microservices, practical tools and methodologies remain limited, particularly for architects working on complex, large-scale systems [7,8].

Additionally, the inherent distribution of systems in microservices architectures introduces challenges such as fault tolerance, scalability, and the orchestration of distributed transactions, further complicating the design process [9].

Artificial intelligence (AI) has emerged as a promising tool to address these challenges by automating and enhancing the design of microservices architectures, particularly for new software developments. AI techniques, such as machine learning (ML) and natural language processing (NLP), assist in critical design tasks such as requirement analysis, service identification, and architectural decision-making. Tools like GreenMicro [10], SEMGROMI [11], and GTMicro [12] employ clustering and semantic analysis to enhance service cohesion and reduce coupling during early-stage design in greenfield projects. Similarly, PF4MD [3] uses problem frames and complexity metrics to support informed service decomposition.

Natural language processing (NLP) is fundamental for analyzing unstructured textual data, such as requirements, user stories, and technical specifications, by identifying key entities, relationships, and semantic patterns [13,14]. This process transforms raw text into actionable design insights, as exemplified by SEMGROMI, which leverages NLP to semantically group user stories and facilitate the definition of coherent service boundaries [11]. In parallel, clustering algorithms, such as k-means, are employed to partition data into homogeneous groups, revealing latent structures within monolithic systems that can guide microservice decomposition [10,12]. However, the effectiveness of these techniques heavily depends on the clarity and quality of the input data, and the precise calibration of the optimal number of clusters is critical to achieving meaningful segmentation. This technical foundation underscores the pivotal role of AI in automating and optimizing microservices design.

AI-driven methods leverage diverse design artifacts—such as textual requirements, UML diagrams, and source code—to generate insights and recommendations for architects. For instance, PF4MD [3] integrates problem frames and complexity metrics to support semi-automated service decomposition, while tools like GreenMicro [10], SEMGROMI [11], and GTMicro [12] analyze user stories and use cases to enhance cohesion, reduce coupling, and refine service boundaries. By leveraging these artifacts, AI not only streamlines the design process but also addresses key challenges such as optimizing service granularity and validating architectural decisions. However, issues such as distributed transaction handling, resilience testing, and aligning AI-generated solutions with real-world demands remain areas in need of further empirical investigation [1,15].

Given these limitations, this systematic literature review aims to bridge the gap by analyzing how AI techniques are applied to improve the design and management of microservices in new software developments. Specifically, it explores tools and methods for decomposition, decision-making, and architecture validation. By addressing these gaps, this review contributes to advancing scalable, robust, and adaptable microservices architectures, outlining practical pathways for their implementation in real-world scenarios.

Research Questions

To guide this systematic literature review, we formulated the following research questions (**Q**) along with their respective motivations (**M**):

Q1. What specific AI techniques and methods have been used in the design of microservices for new software developments? **M1**. This question aims to understand the various AI techniques applied in the design of microservices, identifying the methodologies being used. By becoming aware of these techniques, developers and software architects can make more informed decisions when selecting tools, which may contribute to enhancing efficiency and scalability from the conception of the software.

Q2. What are the main challenges encountered when applying AI techniques to the design of microservices for new software developments? **M2.** Identifying the challenges associated with applying AI techniques to microservices design is crucial for developing strategies to mitigate these issues. This includes technical aspects, such as integrating AI models into microservices architectures, and methodological aspects, such as adapting agile processes to effectively incorporate AI.

Q3. What benefits have been reported when using AI in microservices design from scratch compared to traditional design methodologies that do not employ AI techniques? **M3.** Evaluating the benefits of using AI in microservices design provides a clear view of the potential advantages, such as improvements in design automation, performance optimization, and cost reduction. This is valuable for justifying investment in these technologies.

Q4. What are the emerging trends in the research on using AI for microservices design in new software developments? **M4.** Identifying emerging trends allows academics and professionals to stay up to date with the latest and most promising developments in the field. It also helps highlight areas of opportunity for future research and applications, ensuring the continued evolution of microservices design.

Q5. What specific artifacts, such as textual requirements, source code, or design diagrams, are used as inputs when applying AI techniques in microservices design? **M5.** This question seeks to identify the types of artifacts commonly used as inputs when applying AI techniques in microservices design. Understanding which artifacts are most effectively utilized will help software architects adopt a more structured and efficient approach when incorporating AI into microservices design processes.

The rest of the paper is organized as follows: In Section 2, we briefly survey the main problems of microservices design. Section 3 discusses some related work (e.g., other similar reviews). Section 4 presents our review, including the methodology and selection of relevant papers. In Section 5, we discuss the results and how they align with the research questions. Section 6 addresses the threats to the validity of this review. Finally, Section 7 concludes our paper and presents some further work we are pursuing.

2. The Problems of Microservices Design

The transition from building monolithic systems to using the microservices architectural style offers significant advantages, such as improved scalability, agility, and independent deployability. However, this architectural shift also creates several challenges that impact both the technical and operational aspects of software development. One of the most significant challenges in this shift is the decomposition of existing monolithic applications (or for new applications partitioning the application domain) into microservices, which requires that appropriate service boundaries be identified and that inter-service communication be managed. The process is often complex and context-specific, with development teams struggling to avoid high coupling between services. According to a systematic review by Li et al. [3], many organizations find it challenging to define service boundaries, particularly in tightly integrated monolithic systems. Similarly, a review by

Vural et al. [1] highlights the lack of standardized metrics for assessing the quality-of-service decomposition, complicating the evaluation process.

A survey by Liu et al. [16] further elaborates on the difficulties faced during the migration from monolith to microservices, including the need for effective patterns like the Strangler Pattern, which allows for incremental migration. However, managing the complexity of a hybrid system during the transition phase adds another layer of difficulty, often resulting in performance degradation and maintenance challenges.

Another major issue in microservices design is inter-service communication. In distributed systems, services communicate with each other through APIs or message brokers, introducing new complexities like distributed transactions. The work of Nitin et al. [17] states that managing transactions across multiple services often leads to performance issues and increased latency, especially when the system lacks proper coordination mechanisms. The paper by Baškarada et al. [18] highlights that effective API management and security are essential for maintaining the integrity of a microservices architecture, yet these aspects are frequently overlooked during the design phase, leaving systems vulnerable to security breaches.

Data consistency is another significant challenge in microservices design. In a monolithic system, data are typically managed centrally, ensuring consistency across the application. However, in microservices architectures, each service may manage its own data, leading to issues with data synchronization and eventual consistency. According to Alshuqayran et al. [9], the decentralization of data management in microservices can result in conflicting states, especially in systems with high transaction volumes, and these inconsistencies can be exacerbated by failures in communication between services.

Scalability and performance optimization are often cited as major benefits of microservices, but they also present their own challenges. Goli et al. [19] demonstrate that traditional scaling mechanisms, such as the Horizontal Pod Autoscaler (HPA), often fail to consider interdependencies between services, leading to inefficient resource use. The integration of machine learning techniques, as discussed by the authors of "Microservices and API Deployment Optimization using AI" [20], has been proposed as a solution for optimizing both the deployment and performance of microservices by dynamically adjusting resource allocation and predicting workload patterns.

Security and API management are additional concerns. As highlighted by Ierache et al. [21], microservices architectures expose multiple APIs, which can become a point of vulnerability if not properly secured. The decentralized nature of microservices complicates the implementation of centralized security policies, increasing the risk of unauthorized access and data breaches. Ensuring robust security measures across all microservices is critical, but often challenging, as each service operates independently.

Lastly, the lack of specialized skills in microservices architecture can hinder successful implementation. According to a survey conducted by Baškarada et al. [18], many organizations face difficulties in acquiring the necessary expertise to manage distributed systems, DevOps practices, and container orchestration, which are crucial for running microservices efficiently. This skills gap often leads to misconfigurations, over-complicated architectures, and operational inefficiencies.

Building new applications from scratch adds additional complexity, especially in unfamiliar domains where decomposition into meaningful microservices is challenging. Even proven approaches like Domain-Driven Design (DDD) [6] may fall short, requiring architects to balance service granularity and minimizing inter-service communication. SEMGROMI [11] automates service identification by grouping user stories based on semantic similarity using natural language processing (NLP). While this approach shows promise in enhancing microservices design, it encounters challenges with regard to maintaining low

coupling between services when user stories are ambiguously defined. Approaches such as GreenMicro [10] and GTMicro [12] aim to enhance service identification by clustering use cases and related database entities. However, they face challenges when use case definitions overlap or are ambiguous, which can lead to imprecise service boundaries.

All these problems escalate when building new applications, particularly for new application domains or when developers are not familiar with the domain. Decomposing the domain in meaningful and proper microservices might not be straightforward even when using proven approaches like DDD [7]. Finding the "correct" microservices (i.e., defining their proper boundaries) might be even more difficult in Agile approaches [22], since their incremental and iterative nature makes it difficult to obtain all necessary information at the beginning of the process; as a consequence, the resulting microservices might become "smelly" [4].

In this context, we think that personal assistants, using AI concepts and tools, such as [15] might play an important role helping designers during the process.

3. Related Work

The transition from monolithic systems to microservices has significantly transformed software architecture, offering flexibility, scalability, and independent deployability. However, this shift presents challenges such as service decomposition, optimization, and system adaptability, leading researchers to explore artificial intelligence (AI) techniques to address these complexities. Several studies have examined the application of AI in microservices, with a particular focus on service decomposition, performance optimization, and fault detection.

Li et al. [3]. Presented PF4MD, which enhances service decomposition by combining problem frames with complexity metrics, allowing architects to visualize dependencies and make informed decisions. Similarly, CARGO, an AI-guided tool for migrating monolithic applications to microservices, leverages context-sensitive label propagation to reduce distributed transactions and improve system performance by minimizing cross-service communication, thus enhancing throughput and reducing latency [17]. Although CARGO addresses migration challenges, it concentrates on a particular solution without exploring the general application of AI in designing new microservices architectures.

Vural et al. [1], in their systematic literature review, highlighted the use of clustering techniques, such as Word Embeddings and k-Means, to automate the identification of service boundaries in monolithic codebases. These methods aim to enhance modularity and maintain service cohesion during the decomposition process. Clustering remains one of the most widely used machine learning techniques for identifying service candidates and refining service granularity. However, their review primarily focuses on existing methods and does not specifically address the use of AI techniques in the design phase for new software developments.

Building upon NLP techniques, SEMGROMI [11] clusters user stories based on semantic similarity to facilitate microservice identification with improved cohesion and reduced inter-service coupling. However, its effectiveness depends on the clarity and definition of the user stories, facing difficulties when boundaries are unclear or ambiguous.

GreenMicro offers an alternative method by clustering related use cases and database entities to guide microservice identification in greenfield projects, focusing on early design stages [10].

GTMicro [12] presents a methodology for identifying microservices in greenfield development by using hierarchical clustering based on functional decomposition of use cases and their database entities.

AI's role in optimizing microservices' performance, particularly in autoscaling, has also been extensively studied. Traditional autoscalers, such as the Horizontal Pod Autoscaler (HPA), often fail to account for inter-service dependencies, leading to inefficient resource utilization. Goli et al. [19] proposed Waterfall, an ML-based autoscaling solution that predicts the necessary replicas for each microservice while considering interdependencies. This approach outperforms traditional autoscalers, reducing resource waste and improving system throughput. While this contributes to performance optimization, it focuses on runtime resource management rather than design methodologies.

In terms of adaptive resource allocation, Alshuqayran et al. [9] identified that machine learning techniques such as Support Vector Machines (SVMs), reinforcement learning, and neural networks are frequently employed to dynamically adjust resource allocation, optimizing both cost and performance. These adaptive techniques are crucial in DevOps practices, where AI-driven approaches reduce human intervention and enable real-time decision-making to enhance system resilience and scalability. Their study, however, emphasizes operational aspects over the initial design considerations of microservices using AI.

Security and fault detection in microservices have also benefited from AI-driven methods. Ierache et al. [21] discussed the use of machine learning for intrusion detection in software-defined networks (SDNs), which can be adapted to detect anomalous service behaviors in microservices. This work focuses on enhancing security through AI at the operational level, not directly addressing the design phase of microservices architectures.

Despite the significant advancements in AI applications for microservices, there are still notable gaps in the research. Nitin et al. [17] pointed out the lack of empirical studies assessing the practical impact of AI-driven service decomposition and optimization techniques. Additionally, more research is needed in areas like security and compliance, particularly in industries with strict regulatory requirements. The integration of AI in early design stages, such as architectural decision-making and requirement analysis, also remains underexplored. Diaz-Pace et al. [15] proposed an LLM-based assistant to help novice architects in making design decisions, but further validation is required to assess its effectiveness in real-world scenarios.

Overall, while these studies contribute valuable insights into the application of AI in microservices, they often focus on specific tools, operational optimizations, or lack emphasis on the design phase for new software developments. Our systematic literature review addresses this gap by providing a comprehensive analysis of how AI techniques are applied specifically in the design of microservices from scratch, highlighting challenges, benefits, and emerging trends in this area.

4. Research Methodology

This systematic literature review (SLR) follows the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) guidelines [23] to ensure transparency and rigor in reporting the review process. Additionally, we adopt the stages outlined by [24] for conducting systematic reviews in software engineering, ensuring comprehensive coverage of the most relevant research in the field. The review was conducted to identify and analyze the application of AI techniques in the design—such as the identification and definition of microservices and their responsibilities—and optimization—such as performance enhancement and scalability improvements—of microservices architectures. Our review includes the stages of identification, screening, eligibility, and inclusion of relevant studies, as outlined in the PRISMA flow diagram (Figure 1).

Figure 1. Steps in the formulated review protocol.

4.1. Source Selection

We conducted our search across three major academic databases: IEEE Xplore, ACM Digital Library, and Google Scholar. Google Scholar was included to ensure comprehensive coverage and to minimize the risk of omitting relevant studies, given its broad indexing of academic materials. To maintain quality and rigor, strict exclusion criteria were applied, limiting the selection to peer-reviewed studies and excluding non-peer-reviewed works, such as white papers or unpublished studies. These databases were selected for their relevance to software engineering and computer science, as well as their global impact through highly cited publications. The search was restricted to papers published between 2018 and 2024 to capture recent advances. The starting year of 2018 reflects the emergence of significant developments in applying AI to microservices design, particularly with the rise of deep learning techniques and the widespread adoption of microservices in industry.

4.2. Search Strategy

The search strategy employed a combination of Boolean operators and key terms in English to ensure comprehensive coverage of the topic. The following search string was used: ("microservices design" OR "microservices architecture" OR "microservices development" OR "microservice architecture" OR "microservice design" OR "microservices decomposition" OR "service-oriented architecture") AND ("artificial intelligence" OR "machine learning" OR "deep learning" OR "AI" OR "ML" OR "DL" OR "large language models" OR "LLM").

To ensure precision, the search string was refined using wildcard characters supported by the selected databases, allowing for the inclusion of term variations. Additionally, studies focusing exclusively on service-oriented architecture (SOA) without specific references to microservices were excluded during the selection process.

This search string was applied across the selected databases to retrieve studies that focus on the intersection of microservices architecture and AI techniques. The search aimed to capture both theoretical and practical advancements in this area.

4.3. Search Items and Selection Criteria

- Inclusion Criteria:
  - Scientific papers that directly address the application of AI in the design, development, or optimization of microservices.
  - Studies published between 2018 and 2024.
  - Peer-reviewed journal articles and conference papers.
  - Studies available in English.
- Exclusion Criteria:
  - Papers focusing on service-oriented architecture (SOA) without specific reference to microservices.

- Articles published before 2018, except for foundational work.
- Non-peer-reviewed articles, white papers, or unpublished studies.
- Studies without empirical data or validation.

4.4. Study Selection Process

After removing 200 duplicate records, 10,621 studies were excluded because their titles did not align with the scope of this research. Titles were considered irrelevant if they focused on software engineering topics unrelated to the application of AI techniques for microservices design, such as testing, maintenance, or runtime performance. For example, many studies addressed AI in the operational phase of microservices but not during their design. The screening was conducted manually by the authors, Daniel Narváez and Gustavo Rossi, ensuring a rigorous and consistent selection process.

Subsequently, 875 studies were excluded because they did not meet the inclusion criteria, leaving 117 studies for retrieval. Of these, 20 could not be accessed, resulting in 97 studies being selected for full-text review. During this stage, 54 studies were excluded because they did not provide sufficient information to answer the research questions. For instance, some papers discussed AI techniques but lacked a clear focus on their application in defining microservice boundaries or decision-making during design. Ultimately, 43 studies met all criteria and were included in the final analysis.

To further enhance the methodological rigor, two widely recognized papers on systematic review methodologies were deliberately included. Although unrelated to AI or microservices design, these studies strengthen the transparency and reproducibility of this review. They do not contribute to the primary analysis or influence the results, serving only as methodological references.

The selected studies provide insights into AI-driven microservices design, focusing on service decomposition, architectural decision-making, and optimization techniques. The PRISMA flow diagram (Figure 2) illustrates the full selection process. A detailed list of selected studies, including authors, publication dates, and key findings, is publicly available at https://ing-mat-comp.web.app/ms-and-ai, accessed on 18 October 2024.

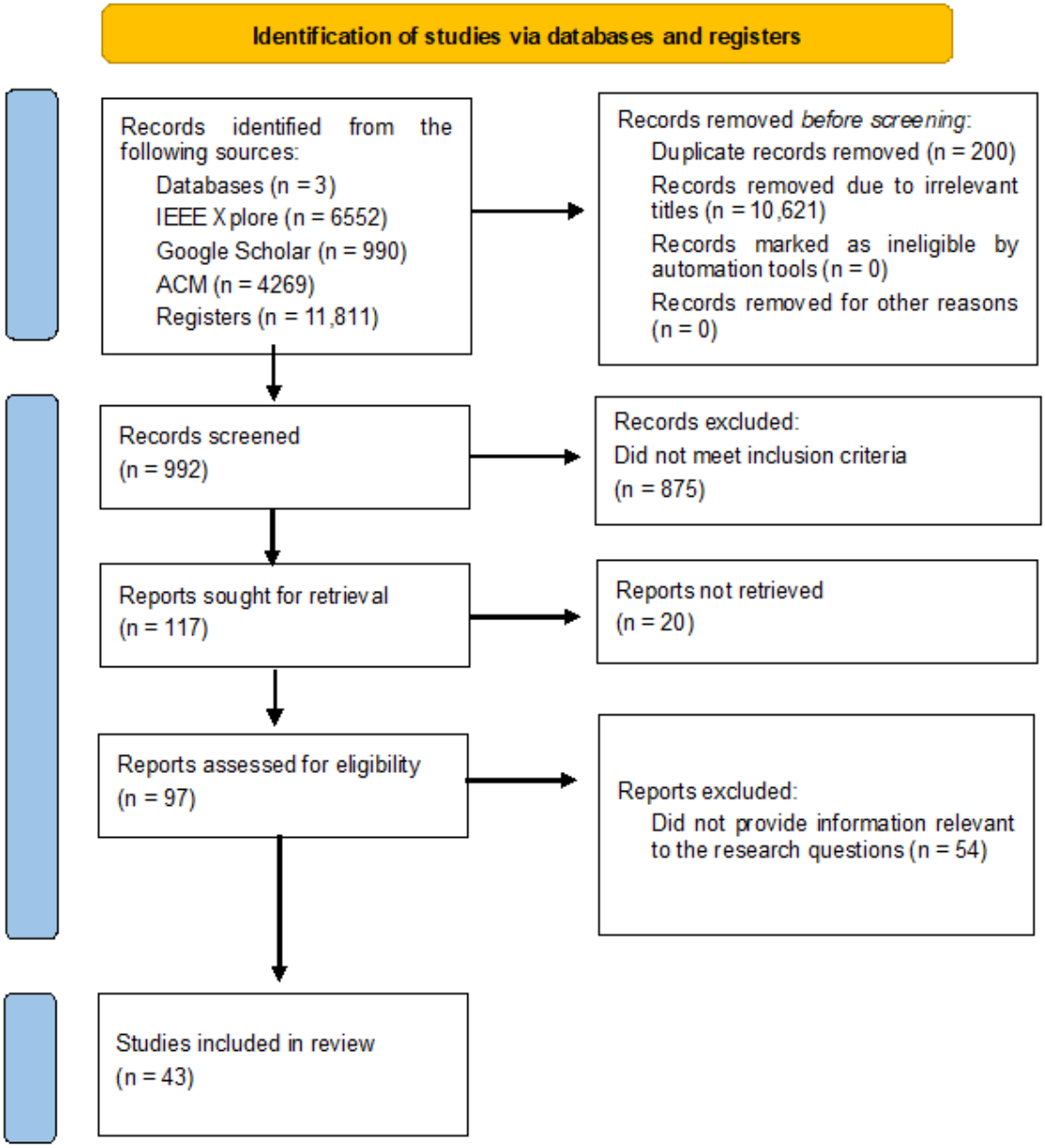

Figure 2. PRISMA 2020 workflow of the identification, screening, eligibility, and inclusion of the studies in the systematic review.

5. Results

Our systematic review analyzed 43 studies focused on the application of artificial intelligence (AI) in microservices design. AI was a key component in all of the studies (100%), while 53.5% specifically employed machine learning (ML) techniques, such as clustering algorithms like k-Means and Word Embeddings, to define service boundaries. Natural language processing (NLP) was used in 32.6% of the studies, primarily to analyze textual requirements and assist in the identification of microservices. Tools such as Mono2Micro, PF4MD, GreenMicro, SEMGROMI, and GTMicro were highlighted across the studies, with an average usage of 1.1 tools per study, reflecting their importance in automating key design tasks. Furthermore, challenges such as data consistency and distributed transactions were mentioned in 95.3% of the studies, emphasizing their critical role in the successful implementation of AI-driven microservices architectures. These findings illustrate the widespread adoption of AI techniques while highlighting the areas where further advancements are needed.

5.1. Overview of Selected Studies

The selected studies apply various AI techniques to different aspects of microservices design, primarily focusing on service decomposition, system optimization, and architectural decision-making. Many studies concentrate on the initial stages of microservices adoption, with particular focus on how AI can assist in decomposing monolithic architectures into microservices. Mono2Micro [25] and PF4MD [3] improve service boundary definition by employing clustering algorithms and evaluating complexity metrics, facilitating a smoother transition from monolithic to microservices architectures.

Other studies explore the optimization of microservices systems through machine learning models that enhance resource allocation and autoscaling. By predicting workload patterns and dynamically adjusting resources, these models improve system performance and scalability. Additionally, recent research emphasizes the potential of large language models (LLMs) in guiding architectural decision-making processes. These works investigate how AI can support architects by analyzing textual requirements and suggesting design patterns or architectural solutions, thereby assisting in complex decision-making tasks.

Overall, these studies demonstrate that AI techniques are being applied across various stages of the microservices design process, from initial decomposition to system optimization and architectural planning. While they hold great promise for automating design tasks and addressing architectural challenges, they also highlight ongoing limitations, particularly in handling distributed transactions and ensuring scalability in large-scale systems.

5.2. Answers to the Research Questions

This systematic review focuses on exploring the application of AI techniques during the design phase (design time) of microservices architectures, addressing key challenges such as service decomposition, architectural decision-making, and optimization. While the primary emphasis is on design-time applications, the review also identifies studies that address runtime aspects, including autoscaling, fault tolerance, and system monitoring. These works, although not directly related to design time, are included to provide a comprehensive view of the broader role of AI in microservices development. This approach highlights the interconnected nature of design-time and runtime considerations, offering valuable insights into how AI can support the full lifecycle of microservices architectures.

Q1. What specific AI techniques and methods have been used in the design of microservices for new software developments?

Throughout the microservices lifecycle, various artificial intelligence (AI) techniques have been applied to improve the design and optimization of microservices. One of the most common is machine learning (ML), especially in the context of monolithic system decomposition into microservices. Techniques like clustering (e.g., k-Means and Word Embeddings) have been used to automatically identify service boundaries in monolithic applications [1,3]. Natural language processing (NLP) has also been effective in transforming textual requirements into formal models and design decisions [13,14]. Mono2Micro [25] applies AI to analyze application behavior, providing recommendations for partitioning monolithic systems into microservices. Other approaches include deep learning for self-adaptive distributed microservices, where reinforcement learning models improve the system's efficiency and performance [26]. Additionally, SEMGROMI [11] employs semantic grouping of user stories through NLP to aid in microservice identification, aiming to enhance service cohesion and minimize inter-service coupling. By leveraging semantic similarity across user stories, SEMGROMI enables a more automated approach to service decomposition from textual artifacts, complementing tools like Mono2Micro in refining service boundaries.

Advanced techniques such as large language models (LLMs) are being used to assist architects in making design decisions based on textual descriptions [27]. Furthermore, the AI-Driven Partitioning Framework provides a dynamic approach for partitioning monolithic applications by analyzing interdependencies between system components, making the partitioning process more efficient and adaptable [28]. In addition to these AI-based methods, GreenMicro [10] offers a clustering approach to identify microservices from use cases in greenfield development. This method concentrates on grouping related use cases and database entities during the early design stages. Although GreenMicro does not directly employ traditional AI techniques, it provides an alternative for initial service decomposition by using use case clustering to provide guidance. In parallel, GTMicro [12] utilizes BERT-based deep learning models to analyze use cases in greenfield projects, focusing on semantic similarity to group related use cases into microservices.

In the context of orchestration and scalability, machine learning techniques such as Long Short-Term Memory (LSTM) and Bidirectional LSTM (Bi-LSTM) networks have been applied for workload prediction in distributed microservices, optimizing resource allocation in cloud environments [29–31].

Q2. What are the main challenges encountered when applying AI techniques to the design of microservices for new software developments?

The primary challenge identified in integrating AI into microservices design is the management of data consistency in distributed systems. Microservices architectures, where each service manages its own data storage, pose synchronization challenges and issues with eventual consistency. Decomposition tools such as PF4MD often fail to fully address these challenges, leaving architects responsible for ensuring consistency across services [3].

Another significant challenge is inter-service communication. In distributed systems, coordinating distributed transactions is complex, increasing the risk of transaction failures or latency issues [17,18]. Additionally, integrating AI into microservice monitoring presents challenges in anomaly detection and identifying issues before they impact system performance [30,32]. Approaches like the one presented in [33] focus on proactive and reactive fault tolerance mechanisms, in which real-time monitoring and predictive analytics based on machine learning can mitigate system failures before they cascade.

Furthermore, the complexity of translating high-level requirements into concrete service boundaries can introduce architectural smells, such as over-coupling between services. While SEMGROMI [11] is advantageous for automating service decomposition using semantic grouping of user stories, it struggles to ensure low coupling and high cohesion when user stories are not well defined or lack clear boundaries. Although GreenMicro is not AI-driven, it encounters challenges in establishing service boundaries when use case definitions are overlapping or unclear, potentially leading to imprecise microservice delineation [10].

GTMicro [12], utilizing a BERT-based approach to cluster use cases, faces difficulties in identifying semantically distinct services when textual descriptions are vague or when there are insufficient training data for the AI models to fully grasp the domain. While promising for greenfield developments, this method encounters challenges when complex domain knowledge needs to be integrated into the clustering process.

AI-based autoscaling also faces difficulties when service interdependencies are not adequately considered. Models like Waterfall optimize scaling, but rely on accurate workload predictions, which may not always reflect real-time system conditions [29]. Sage, a system proposed in [34], leverages advanced machine learning models like Bayesian Networks and Graphical Variational Autoencoders to identify performance bottlenecks in complex microservices architectures. It allows for scalable and practical performance debugging, which is a key benefit in handling interdependent microservices systems efficiently.

Q3. What benefits have been reported when using AI in microservices design from scratch compared to traditional design methodologies that do not employ AI techniques?

The benefits of using AI in microservices design from scratch are numerous and impactful, particularly when it comes to automating complex tasks and optimizing early architectural decisions.

AI in the design phase: A key advantage is the automation of service decomposition, which significantly reduces the time and effort required by architects to identify service boundaries. Tools such as Mono2Micro and PF4MD enhance the accuracy of decomposition, minimize manual intervention, and streamline the creation of modular and cohesive architectures [3,25]. Frameworks like the AI-Driven Partitioning Framework further contribute by dynamically analyzing and recommending efficient partitions. Although initially developed to support the migration of monolithic systems to microservices, these frameworks offer principles—such as dependency analysis and modular decomposition—that are equally beneficial for designing microservices in greenfield projects, enabling architects to achieve optimized service granularity and improved modularity [28].

AI in the runtime phase: Another advantage lies in addressing runtime challenges through design. For example, AI-driven designs can incorporate features that enhance runtime scalability and resilience. AI-based autoscaling techniques, which dynamically adjust resources to maximize efficiency, demonstrate how foundational design decisions enable microservices to handle real-time demands more effectively [29,35]. Similarly, fault tolerance mechanisms benefit from AI-enhanced designs, as machine learning models proactively detect and manage failures, reducing downtime and improving system resilience [33]. Furthermore, predictive analytics and real-time monitoring data inform these designs, allowing architects to anticipate and mitigate potential operational issues during the early stages of planning [36,37].

These benefits demonstrate the dual role of AI in microservices: as a tool for automating and enhancing the design process, and as a foundation for long-term runtime scalability, fault tolerance, and cost optimization. By integrating AI-driven approaches, architects can achieve modular, scalable, and resilient architectures that reflect the value of informed design decisions.

Q4. What are the emerging trends in the research on using AI for microservices design in new software developments?

Emerging trends in AI research for microservices design highlight innovative techniques and tools that enhance the architectural process for new software developments. One notable trend is the use of generative AI models, such as large language models (LLMs), which assist architects in automating complex design tasks. These models can generate architectural patterns, recommend service decompositions, and suggest alternative system configurations based on existing design knowledge [38]. For instance, they enable architects to explore various communication protocols or identify optimal microservice boundaries tailored to specific project requirements, significantly reducing manual effort and improving design consistency. In line with this, recent work [39] demonstrates a prompt pattern sequence approach that harnesses generative AI to assist in software architecture decision-making. Moreover, studies such as [40,41] underscore the importance of effective prompt engineering, illustrating that well-crafted prompt patterns can significantly enhance the reliability and applicability of AI-generated outputs in architectural contexts.

Another critical trend is the integration of deep learning and reinforcement learning techniques for optimizing the adaptability and scalability of microservices. These methods are increasingly used to refine resource allocation and improve system performance in distributed environments. By learning from iterative feedback, these models adapt to real-time conditions, ensuring robust and flexible system designs [26,42]. This trend is

particularly valuable for addressing scalability challenges in greenfield projects, where workload patterns may be unpredictable [1].

Frameworks like the AI-Driven Partitioning Framework demonstrate how automation of partitioning processes has become essential for both migration and greenfield strategies. While initially designed for monolithic to microservices transitions, their principles are increasingly relevant for designing new systems, enabling precise service granularity and optimized architectures [28].

The use of machine learning models to predict workloads and enable dynamic autoscaling is also gaining attention. Techniques such as these allow microservices systems to adjust their resource allocation in real time, optimizing performance and reducing operational costs. For instance, studies such as [43] highlight the role of online machine learning in resource provisioning, showcasing its potential to enhance scalability and efficiency in distributed environments.

The role of AI in documenting and analyzing architectural decisions has also gained traction. Research emphasizes structured decision documentation, where AI tools automate the analysis of trade-offs and suggest configurations aligned with project goals. For example, the use of decision space analysis supports scalable API management and ensures maintainability in complex microservices architectures [44]. These approaches not only enhance decision-making but also establish a repository of reusable architectural patterns, facilitating the design of future systems.

Additionally, AI-driven tools continue to innovate in areas such as service decomposition and fault tolerance. Techniques that combine problem frames with complexity analysis enable more granular decomposition strategies, reducing inter-service communication and improving maintainability [3]. Similarly, the integration of real-time monitoring data with AI models provides architects with actionable insights, bridging the gap between design and runtime. For instance, data from execution traces and system logs help identify potential bottlenecks or high-coupling areas that can inform refined service boundaries [30,31,37].

These trends underscore a paradigm shift in how AI is applied to microservices design. By combining advanced learning models with decision-making frameworks, researchers and practitioners are redefining the possibilities of scalable, adaptable, and efficient architectures for new software developments.

Q5. What specific artifacts, such as textual requirements, source code, or design diagrams, are used as inputs when applying AI techniques in microservices design?

AI techniques applied in microservices design primarily rely on artifacts that provide structured and unstructured information about system requirements and architecture. Textual requirements, such as user stories, software specifications, or functional descriptions, are critical inputs. Tools like Mono2Micro analyze these artifacts using natural language processing (NLP) to extract meaningful information and propose decomposition strategies [25]. Similarly, SEMGROMI employs semantic grouping of user stories to identify microservices, enhancing cohesion and reducing coupling [11].

For greenfield projects, use case descriptions and domain-specific entities are frequently leveraged. Tools like GreenMicro and GTMicro cluster these artifacts into meaningful microservices, enabling structured designs in scenarios where domain knowledge is limited [10,12]. Despite their utility, challenges arise when textual artifacts are ambiguous or incomplete, which can impact the precision of service boundaries.

Formal models, such as UML diagrams and architectural design documents, are also crucial in tools like PF4MD, which integrate these models with complexity metrics to optimize service partitioning [3,25]. These models provide structured representations that AI tools can analyze to guide early-stage design decisions. Architectural patterns like CQRS

and SAGA further serve as templates for orchestrating microservices and maintaining consistency [35].

While studies have primarily focused on design, some studies suggest that runtime data—such as execution traces and performance metrics—can inform design refinement. These artifacts provide insights into system behavior, such as identifying bottlenecks or high coupling, which can guide architects in optimizing service boundaries and improving scalability [30,32,45]. However, their role remains secondary to primary design artifacts.

Generative AI tools further extend the utility of design artifacts by incorporating historical architectural knowledge and patterns. These tools support architects in automating design processes, ensuring consistency, and bridging gaps in manual expertise [38].

5.3. Discussion

The reviewed studies demonstrate that AI is playing an increasingly critical role in the design and optimization of microservices. These tools showcase how AI can automate complex tasks like service decomposition and system performance optimization, reducing the cognitive load on architects and developers. SEMGROMI, in particular, offers an innovative approach by leveraging semantic grouping of user stories, addressing challenges related to service cohesion and inter-service coupling in ways that traditional methods struggle to achieve.

At the same time, the incorporation of AI in microservices exposes persistent challenges that are yet to be fully resolved. Managing distributed transactions, ensuring data consistency across services, and defining precise service boundaries remain intricate tasks, especially when input artifacts such as user stories or use cases are poorly defined. Approaches like GreenMicro, which focus on clustering use cases in greenfield development, add value to early-stage design processes but require clearer guidance when addressing ambiguous or overlapping use cases.

Moreover, recent trends, such as the use of generative AI models like large language models (LLMs), highlight new opportunities in microservices design. These models can assist architects by providing actionable recommendations for service decomposition and suggesting alternative architectural patterns based on textual inputs. However, their practical application in real-world scenarios remains limited and requires further empirical validation.

AI's role in performance optimization through autoscaling mechanisms, as seen in models like Waterfall, reveals its potential in dynamic resource management. Similarly, integrating real-time data, such as execution traces or system logs, with AI models allows for continuous refinement of service boundaries and improved scalability. However, the scalability and robustness of these solutions in real-time environments still require additional validation to ensure effectiveness under unpredictable workloads and inter-service dependencies.

Although AI-driven tools have made significant strides, there remain notable gaps in handling non-functional requirements such as security, fault tolerance, and compliance. Current research has largely focused on functional decomposition, while critical aspects like ensuring system resilience under failure conditions are underexplored. Future efforts must integrate non-functional requirements into AI-based design tools to achieve more holistic microservices architectures.

To provide a clearer comparative overview of these emerging trends, Table 1 summarizes the key techniques currently explored, their applications in microservices design, their advantages, and their limitations.

Table 1. Comparison of emerging AI techniques in microservices design.

Technique/Model	Application	Advantages	Limitations
Generative AI/LLMs	Analyzes historical architectural data and textual requirements to generate design recommendations and propose decompositions.	Automates complex design tasks; provides multiple design options.	Requires high-quality data; may lack interpretability; needs domain-specific fine-tuning.
Deep Learning for Autoscaling	Predicts workload patterns and dynamically adjusts resource allocation in distributed microservices.	Enhances performance; minimizes manual intervention; adapts to dynamic loads.	Requires extensive training data; may struggle with unforeseen scenarios.
Reinforcement Learning for Resource Allocation	Optimizes real-time resource allocation and improves fault tolerance by continuously learning from operational feedback.	Continuously improves performance; adapts to changing conditions.	Complexity in modeling; potential training instability; requires careful reward design.
Hybrid Models (Traditional + AI)	Combines rule-based architectural decision frameworks with AI-driven insights to support both design and runtime optimization.	Integrates strengths of traditional methods and data-driven approaches.	Increases overall system complexity; poses integration challenges.

These trends signal a paradigm shift in microservices design, from static, rule-based approaches to dynamic, data-driven, and generative methodologies. However, further empirical validation and interdisciplinary research are necessary to fully harness the potential of these emerging techniques in practical, large-scale deployments.

The findings suggest that while AI has the potential to transform microservices design, a more comprehensive approach is needed—one that combines AI's strengths in automation with careful consideration of architectural and operational complexities. The development of frameworks capable of addressing both functional and non-functional requirements, as well as adapting to evolving scenarios, will be critical for the next generation of microservices design tools.

6. Threads to Validity

Several threats to the validity of this review must be acknowledged. First, the search strategy may have missed relevant studies that were not indexed in the selected databases or were published outside the chosen timeframe, potentially limiting the comprehensiveness of the review. Additionally, the exclusion of non-English studies may have restricted the scope by omitting relevant research published in other languages. The selection process also introduces a potential bias, as the criteria used to assess the relevance of studies may have inadvertently excluded valuable insights. Moreover, the quality of the studies included in the review was not formally assessed, which may affect the overall reliability of the findings. Finally, the rapidly evolving nature of both microservices architecture and AI techniques means that the findings of this review may quickly become outdated as new research continues to emerge.

7. Conclusions

This systematic literature review has underscored the significant potential of artificial intelligence (AI) in addressing the inherent complexities in microservices design. AI-driven

tools such as Mono2Micro and PF4MD have shown substantial promise by automating critical processes like service decomposition, resource scaling, and fault detection. These advancements reduce manual intervention, improve system performance, and enhance the scalability and reliability of microservices architectures. Additionally, newer tools like SEMGROMI, GreenMicro, and GTMicro have further expanded the landscape by incorporating semantic grouping and clustering techniques to identify microservices from textual artifacts or use cases, providing valuable assistance in greenfield development scenarios.

However, significant challenges remain. Managing distributed transactions, ensuring data consistency, and addressing inter-service communication remain complex tasks, especially in large-scale, distributed environments. The precision of AI-driven service decomposition is still influenced by the quality and clarity of input artifacts, such as user stories or use case descriptions, which can lead to poorly defined service boundaries if ambiguities are present. While AI has made strides in optimizing these areas, it has not fully resolved intricate issues related to fault tolerance, security, and real-time decision-making in microservices architectures.

Emerging trends, such as the use of generative AI models for architectural decision-making and the integration of deep learning techniques for scalability optimization, hold great promise. Tools like GTMicro, which employ deep learning models to analyze use cases, offer novel approaches for early-stage service decomposition in greenfield projects, enhancing the overall design process. These developments signal a future where AI will not only automate routine design tasks but also drive innovation in system architecture, pushing the boundaries of what is possible in microservices design.

To fully leverage AI's potential in microservices design, future research should focus on refining AI models for more seamless integration into real-world systems, particularly addressing challenges like distributed data management, security, and compliance. Additionally, it is crucial to further investigate how AI-driven tools can handle non-functional requirements, such as system resilience and fault tolerance, which are critical for maintaining robust and secure microservices systems. Empirical studies are also needed to validate the effectiveness of AI tools across diverse industry applications and ensure their scalability in large, complex systems.

As the field continues to evolve, AI is poised to become an indispensable component in the toolkit of software architects and developers working with microservices, driving both the automation and innovation of microservices design.

Author Contributions: Conceptualization, D.N. and G.R.; methodology, D.N., N.B. and G.R.; validation, D.N. and A.F.; formal analysis, D.N.; investigation, D.N. and G.R.; resources, D.N. and G.R.; data curation, D.N.; writing—original draft preparation, D.N.; writing—review and editing, D.N., N.B., A.F. and G.R.; supervision, G.R. and A.F.; project administration, D.N.; funding acquisition, G.R. and N.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by CAETI, Facultad de Informática, Universidad Abierta Interamericana.

Data Availability Statement: Our complete data can be found at https://ing-mat-comp.web.app/ms-and-ai, accessed on 18 October 2024.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Vural, H.; Koyuncu, M.; Guney, S. A systematic literature review on microservices. In *Computational Science and Its Applications–ICCSA 2017: 17th International Conference, Trieste, Italy, 3–6 July 2017, Proceedings, Part VI 17*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 203–217.

2. Lewis, J.; Fowler, M. Microservices: A Definition of this New Architectural Term (2014). Available online: http://martinfowler.com/articles/microservices.html (accessed on 20 September 2024).
3. Li, Y.; Li, Z.; Bu, Y.; Xiao, H.; Deng, Y. PF4MD: A Microservice Decomposition Tool Combining Problem Frames. In Proceedings of the 2023 IEEE 31st International Requirements Engineering Conference (RE), Hannover, Germany, 4–8 September 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 359–360.
4. Ponce, F.; Soldani, J.; Astudillo, H.; Brogi, A. Smells and refactorings for microservices security: A multivocal literature review. *J. Syst. Softw.* **2022**, *192*, 111393. [CrossRef]
5. Taibi, D.; Lenarduzzi, V. On the definition of microservice bad smells. *IEEE Softw.* **2018**, *35*, 56–62. [CrossRef]
6. Evans, E. *Domain-Driven Design: Tackling Complexity in the Heart of Software*; Addison-Wesley Professional: Boston, MA, USA, 2004.
7. Vural, H.; Koyuncu, M. Does domain-driven design lead to finding the optimal modularity of a microservice? *IEEE Access* **2021**, *9*, 32721–32733. [CrossRef]
8. Zhong, C.; Li, S.; Huang, H.; Liu, X.; Chen, Z.; Zhang, Y.; Zhang, H. Domain-driven design for microservices: An evidence-based investigation. *IEEE Trans. Softw. Eng.* **2024**, *50*, 1425–1449. [CrossRef]
9. Alshuqayran, N.; Ali, N.; Evans, R. A systematic mapping study in microservice architecture. In Proceedings of the 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), Macau, China, 4–6 November 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 44–51.
10. Bajaj, D.; Goel, A.; Gupta, S.C. GreenMicro: Identifying microservices from use cases in greenfield development. *IEEE Access* **2022**, *10*, 67008–67018. [CrossRef]
11. Vera-Rivera, F.H.; Cuadros, E.G.P.; Perez, B.; Astudillo, H.; Gaona, C. SEMGROMI—A semantic grouping algorithm to identifying microservices using semantic similarity of user stories. *PeerJ Comput. Sci.* **2023**, *9*, e1380. [CrossRef]
12. Bajaj, D.; Bharti, U.; Gupta, I.; Gupta, P.; Yadav, A. GTMicro—Microservice identification approach based on deep NLP transformer model for greenfield developments. *Int. J. Inf. Technol.* **2024**, *16*, 2751–2761. [CrossRef]
13. Liu, K.; Reddivari, S.; Reddivari, K. Artificial intelligence in software requirements engineering: State-of-the-art. In Proceedings of the 2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI), San Diego, CA, USA, 9–11 August 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 106–111.
14. Kochbati, T.; Li, S.; Gérard, S.; Mraidha, C. From user stories to models: A machine learning empowered automation. In Proceedings of the MODELSWARD 2022-9th International Conference on Model-Driven Engineering and Software Development, Online Streaming, France, 8–10 February 2021; SCITEPRESS-Science and Technology Publications: Setúbal, Portugal, 2021; pp. 28–40.
15. Díaz-Pace, J.A.; Tommasel, A.; Capilla, R. Helping Novice Architects to Make Quality Design Decisions Using an LLM-Based Assistant. In *European Conference on Software Architecture*; Springer: Berlin/Heidelberg, Germany, 2024; pp. 324–332.
16. Velepucha, V.; Flores, P. A survey on microservices architecture: Principles, patterns and migration challenges. *IEEE Access* **2023**, *11*, 88339–88358. [CrossRef]
17. Nitin, V.; Asthana, S.; Ray, B.; Krishna, R. Cargo: Ai-guided dependency analysis for migrating monolithic applications to microservices architecture. In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering, Rochester, MI, USA, 10–14 October 2022; pp. 1–12.
18. Baškarada, S.; Nguyen, V.; Koronios, A. Architecting microservices: Practical opportunities and challenges. *J. Comput. Inf. Syst.* **2020**, *60*, 428–436. [CrossRef]
19. Goli, A.; Mahmoudi, N.; Khazaei, H.; Ardakanian, O. A Holistic Machine Learning-based Autoscaling Approach for Microservice Applications. *CLOSER* **2021**, *1*, 190–198.
20. Charankar, N.; Pandiya, D.K. Microservices and API Deployment Optimization Using AI. *Int. J. Recent Innov. Trends Comput. Commun.* **2024**, *11*, 1090–1095. [CrossRef]
21. Ierache, J.; García-Martínez, R.; De Giusti, A. Learning life cycle in autonomous intelligent systems. In Proceedings of the IFIP International Conference on Artificial Intelligence in Theory and Practice, Milano, Italy, 7–10 September 2008; Springer: Boston, MA, USA, 2008; pp. 451–455.
22. Ünlü, H.; Kennouche, D.E.; Soylu, G.K.; Demirörs, O. Microservice-based projects in agile world: A structured interview. *Inf. Softw. Technol.* **2024**, *165*, 107334. [CrossRef]
23. Page, M.J.; McKenzie, J.E.; Bossuyt, P.M.; Boutron, I.; Hoffmann, T.C.; Mulrow, C.D.; Shamseer, L.; Tetzlaff, J.M.; Akl, E.A.; Brennan, S.E.; et al. The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *BMJ* **2021**, *372*, n71. [CrossRef] [PubMed]
24. Kitchenham, B.; Pretorius, R.; Budgen, D.; Brereton, O.P.; Turner, M.; Niazi, M.; Linkman, S. Systematic literature reviews in software engineering—A tertiary study. *Inf. Softw. Technol.* **2010**, *52*, 792–805. [CrossRef]
25. Kalia, A.K.; Xiao, J.; Lin, C.; Sinha, S.; Rofrano, J.; Vukovic, M.; Banerjee, D. Mono2micro: An ai-based toolchain for evolving monolithic enterprise applications to a microservice architecture. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual USA, 8–13 November 2020; pp. 1606–1610.

26. Magableh, B.; Almiani, M. Deep Q Learning for Self Adaptive Distributed Microservices Architecture. *IEEE Acess*, 2019; *in press*.
27. Dhar, R.; Vaidhyanathan, K.; Varma, V. Can LLMs Generate Architectural Design Decisions?—An Exploratory Empirical study. *arXiv* **2024**, arXiv:2403.01709.
28. Ramamoorthi, V. AI-Driven Partitioning Framework for Migrating Monolithic Applications to Microservices. *J. Comput. Soc. Dyn.* **2023**, *8*, 63–72.
29. Imdoukh, M.; Ahmad, I.; Alfailakawi, M.G. Machine learning-based auto-scaling for containerized applications. *Neural Comput. Appl.* **2020**, *32*, 9745–9760. [CrossRef]
30. Singh, A.; Aggarwal, A. Artificial Intelligence Enabled Microservice Container Orchestration to increase efficiency and scalability for High Volume Transaction System in Cloud Environment. *J. Artif. Intell. Res. Appl.* **2023**, *3*, 24–52.
31. Zhong, Z.; Xu, M.; Rodriguez, M.A.; Xu, C.; Buyya, R. Machine learning-based orchestration of containers: A taxonomy and future directions. *ACM Comput. Surv. (CSUR)* **2022**, *54*, 1–35. [CrossRef]
32. Jiang, Y.; Zhang, N.; Ren, Z. Research on intelligent monitoring scheme for microservice application systems. In Proceedings of the 2020 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), Vientiane, Laos, 11–12 January 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 791–794.
33. Power, A.; Kotonya, G. A microservices architecture for reactive and proactive fault tolerance in iot systems. In Proceedings of the 2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Chania, Greece, 12–15 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 588–599.
34. Gan, Y.; Liang, M.; Dev, S.; Lo, D.; Delimitrou, C. Sage: Practical and scalable ML-driven performance debugging in microservices. In Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Virtual USA, 19–23 April 2021; pp. 135–151.
35. Zubov, D.; Kupin, A.; Kosei, M.; Holiver, V. Models and Technologies for Autoscaling Based on Machine Learning for Microservices Architecture. In Proceedings of the COLINS-2024: 8th International Conference on Computational Linguistics and Intelligent Systems, Lviv, Ukraine, 12–13 April 2024.
36. Zhang, Y.; Hua, W.; Zhou, Z.; Suh, G.E.; Delimitrou, C. Sinan: ML-based and QoS-aware resource management for cloud microservices. In Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Virtual USA, 19–23 April 2021; pp. 167–181.
37. Kang, P.; Lama, P. Robust resource scaling of containerized microservices with probabilistic machine learning. In Proceedings of the 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC), Leicester, UK, 7–10 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 122–131.
38. Ozkaya, I. Can architecture knowledge guide software development with generative AI? *IEEE Softw.* **2023**, *40*, 4–8. [CrossRef]
39. JMaranhão, J.O.; Guerra, E.M. A Prompt Pattern Sequence Approach to Apply Generative AI in Assisting Software Architecture Decision-making. In Proceedings of the 29th European Conference on Pattern Languages of Programs, People, and Practices, Irsee, Germany, 3–7 July 2024; pp. 1–12.
40. Phoenix, J.; Taylor, M. *Prompt Engineering for Generative AI: Future-Proof Inputs for Reliable AI Outputs at Scale*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2024.
41. JWhite; Hays, S.; Fu, Q.; Spencer-Smith, J.; Schmidt, D.C. Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. In *Generative AI for Effective Software Development*; Springer: Berlin/Heidelberg, Germany, 2024; pp. 71–108.
42. Moreschini, S.; Pour, S.; Lanese, I.; Balouek-Thomert, D.; Bogner, J.; Li, X.; Pecorelli, F.; Soldani, J.; Truyen, E.; Taibi, D. AI Techniques in the Microservices Life-Cycle: A Survey. *arXiv* **2023**, arXiv:2305.16092.
43. Alipour, H.; Liu, Y. Online machine learning for cloud resource provisioning of microservice backend systems. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 2433–2441.
44. Haselböck, S.; Weinreich, R.; Buchgeher, G.; Kriechbaum, T. Microservice design space analysis and decision documentation: A case study on API management. In Proceedings of the 2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA), Paris, France, 20–22 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–8.
45. Parekh, N.; Kurunji, S.; Beck, A. Monitoring resources of machine learning engine in microservices architecture. In Proceedings of the 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 1–3 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 486–492.