**EVALUATE THE IMPACT OF MICROFRONTENDS ON CLOUD-BASED APPLICATION ARCHITECTURE**

**MAI CHUNG TUAN**

**INTERIM REPORT**

**MAY 2024**

# **DEDICATION**

This thesis work is dedicated to my wife, Quynh Nguyen, who has been a constant source of support and encouragement during the challenges of graduate school and life. I am truly thankful for having you in my life. This work is also dedicated to my parents who have always loved me unconditionally and whose good examples have taught me to work hard for the things that I aspire to achieve.

# **ACKNOWLEDGEMENTS**

I would like to thank Upgrad and LJMU for giving me the chance to do this thesis work for them, and especially, big thanks to my supervisor at LJMU, Dr. Ehsan Rana, for her constructive feedback and unwavering guidance throughout the thesis process.

I must also extend my sincere appreciation to all the interviewees and participants who generously shared their time and invaluable insights during this research. Their contributions form the foundation of this thesis, and I am truly grateful for their perspectives.

Lastly, thank you to all who have played a role, large or small, in making this endeavor possible.

# **ABSTRACT**

This study investigates the impact of applying micro-frontend development to cloud-based applications. It will explore the impact of micro-frontend to the scalability, cost-effectiveness, and securing of cloud-based applications. The primary objectives are structured around comprehensive case studies, performance evaluations, and the formulation of best practices. Through these objectives, the research seeks to provide actionable guidelines and insights for effectively integrating micro-frontends within cloud-based applications. Real-world implementations of micro-frontends will be examined to gain practical insights, performance assessments will be conducted in simulated cloud environments, and best practices will be developed for seamless integration. The scope of the study is broad, encompassing a thorough examination of how micro frontend development influences the scalability, cost-effectiveness, and security of cloud-based application architectures. Employing a multifaceted approach, including case studies and performance evaluations, the study aims to address key aspects of this integration. By considering a variety of real-world scenarios and diverse cloud platforms, the research aims to provide a holistic understanding of the challenges and opportunities associated with the integration of micro-frontends into cloud-based applications. The methodologies, case studies, and analyses detailed in subsequent sections contribute to the overarching goal of offering valuable insights to the dynamic landscape of web application development, enabling organizations to make informed architectural decisions and achieve optimal outcomes.

***Keywords:*** *S*calability, security, cost-effectiveness, micro-frontends, cloud-based application

# **LIST OF ABBREVIATIONS**

MFE Micro frontends

CEA Cost effective on ...

# **LIST OF FIGURES**

Figure 1.1: ... 12

# **LIST OF TABLES**

Table 1.1: ...

# **LIST OF DIAGRAMS**

Diagram 1.1: ... 21

[**DEDICATION** i](#_Toc167010550)

[**ACKNOWLEDGEMENTS** ii](#_Toc167010551)

[**ABSTRACT** iii](#_Toc167010552)

[**LIST OF ABBREVIATIONS** iv](#_Toc167010553)

[**LIST OF FIGURES** v](#_Toc167010554)

[**LIST OF TABLES** vi](#_Toc167010555)

[**LIST OF DIAGRAMS** vii](#_Toc167010556)

[**TABLE OF CONTENTS** viii](#_Toc167010557)

[**Chapter 1 : INTRODUCTION** 1](#_Toc167010558)

[**1.1** **Background** 1](#_Toc167010559)

[**1.2** **Problem statement** 1](#_Toc167010560)

[**1.3** **Significance of the Study** 2](#_Toc167010561)

[**1.4** **Scope of the Study** 2](#_Toc167010562)

[**1.5** **Aim and objective of the study** 2](#_Toc167010563)

[**Chapter 2 : RELATED WORK** 4](#_Toc167010564)

[**2.1** **Basic knowledgement** 4](#_Toc167010565)

[**2.1.1** **Definition of micro-frontends** 4](#_Toc167010566)

[**2.1.1.1** **What is micro-frontend?** 4](#_Toc167010567)

[**2.1.1.2** **How do micro-frontends work?** 5](#_Toc167010568)

[**2.1.1.3** **Types of micro-frontends?** 5](#_Toc167010569)

[**2.1.1.4** **Micro-frontends architecture** 6](#_Toc167010570)

[**2.1.2** **Definition of cloud-based architecture** 6](#_Toc167010571)

[**2.1.3** **Scalability in cloud-based applications** 7](#_Toc167010572)

[**2.1.3.1** **Vertical scaling in cloud computing** 7](#_Toc167010573)

[**2.1.3.2** **Horizontal scaling in cloud computing** 8](#_Toc167010574)

[**2.1.4** **Security in cloud-based applications** 8](#_Toc167010575)

[**2.1.5** **Cost-effectiveness in cloud-based applications** 8](#_Toc167010576)

[**Chapter 3 : RESEARCH METHOD** 9](#_Toc167010577)

[**3.1** **Data Collection and Analysis of Security Tactics for Microservices** 9](#_Toc167010578)

[**3.2** **Case Study Preparation and Development of Model Data Set** 9](#_Toc167010579)

[**3.3** **Definition of ADDs and Metrics** 9](#_Toc167010580)

[**3.4** **Case Study-Based Analysis and Evaluation** 10](#_Toc167010581)

[**3.5** **Model Generation** 10](#_Toc167010582)

[**3.6** **Extraction of Architecture Models from Microservice Code** 10](#_Toc167010583)

[**1.1.** **Mô hình nghiên cứu các yếu tố ảnh hưởng đến hành vi vì môi trường (PEB) tại công ty Kuehne + Nagel** 12](#_Toc167010584)

[**1.1.1.** **Mô hình nghiên cứu đề xuất** 12](#_Toc167010585)

[**1.1.2.** **Ý nghĩa của các biến nghiên cứu** 16](#_Toc167010586)

[**TÓM TẮT MỤC 3** 19](#_Toc167010587)

[**2.** **PHÂN TÍCH VẤN ĐỀ** 19](#_Toc167010588)

[**TÓM TẮT MỤC 4** 20](#_Toc167010589)

[**3.** **GIẢI PHÁP VÀ KẾ HOẠCH TRIỂN KHAI** 20](#_Toc167010590)

[**3.1.** **Giải pháp chi tiết của đề án** 20](#_Toc167010591)

[**3.2.** **Kế hoạch triển khai** 20](#_Toc167010592)

[**TÓM TẮT MỤC 5** 20](#_Toc167010593)

[**3.1.1.1.** **Quan điểm Tài chính** 20](#_Toc167010594)

[**3.1.1.2.** **Quan điểm Quy trình nội bộ** 23](#_Toc167010595)

[**3.1.1.3.** **Quan điểm Các bên liên quan** 24](#_Toc167010596)

[**3.1.1.4.** **Quan điểm Học tập và Phát triển** 24](#_Toc167010597)

[**3.1.1.5.** **Quan điểm Môi trường** 25](#_Toc167010598)

[**3.1.1.6.** **Quan điểm Xã hội** 27](#_Toc167010599)

[**Chapter 2 KẾT LUẬN** 28](#_Toc167010600)

[**Chapter 3 DANH MỤC TÀI LIỆU THAM KHẢO** 28](#_Toc167010601)

[Chapter 4 **PHỤ LỤC 🡺** **PENDING** 1](#_Toc167010602)

# **TABLE OF CONTENTS**

# **: INTRODUCTION**

## **Background**

Over the last few years, the world of web application development and deployment has undergone significant changes. These changes have been largely driven by the increasing popularity of micro-frontends and cloud computing (Universitet et al., n.d.). The rise of micro-frontends, which is an architectural approach that applies the principles of microservices to the frontend layer, has completely transformed the process of designing and building web applications. At the same time, cloud-based application architectures have emerged as the standard choice for organizations that want infrastructure that is scalable, cost-effective, and security. Our method involves detecting architectural decisions related to these tactics automatically and using formal metrics to assess system conformance. We apply this approach to a dataset of 10 open-source microservice systems and 20 variants, evaluating the validity of our metrics through statistical analysis.

## **Problem statement**

The significant changes in the world of web application development and deployment, driven by the increasing popularity of micro-frontends and cloud computing, have presented new challenges and opportunities. These changes have transformed the traditional process of designing and building web applications, with a shift towards micro-frontends and cloud-based application architectures. This prompts the need to explore the potential drawbacks, caveats, and best practices associated with these emerging trends in web application development and deployment.

The widespread adoption of micro-frontends and cloud-based application architectures requires a thorough examination of their impact on the security implications, scalability, and cost-efficiency aspects. Understanding the practical considerations and potential challenges associated with these innovative approaches is crucial for organizations and developers to make informed decisions when implementing modern web application solutions.

As organizations increasingly transition towards micro-frontends and cloud computing for their web applications, it becomes essential to address any potential limitations or drawbacks associated with these approaches. Additionally, strategies and best practices need to be defined to optimize the development, deployment, and management of web applications in the context of micro-frontends and cloud-based architectures.

The ongoing transformation in web application development necessitates a comprehensive exploration of the implications of adopting micro-frontends and cloud computing. This exploration will guide the future development of web applications and provide insights into ensuring the efficient, secure, and scalable deployment of next-generation web applications.

## **Significance of the Study**

The importance of this study is underscored by the intersection of two revolutionary architectural concepts: micro-frontends and cloud-based application architectures. As businesses embrace these paradigms at an escalating rate, it becomes crucial to grasp how they interact synergistically or present challenges. The critical need to explore the implications of integrating micro-frontends into cloud-based application architectures, particularly with regard to scalability, cost-effectiveness, and security measures, arises from a variety of essential factors.:

1. Rapid Technological Advancements: With the constant evolution of technology, particularly in cloud computing and web development, understanding the implications of incorporating micro-frontends is crucial to stay current and competitive in the industry.
2. Increasing Demand for Scalability: As the demand for scalable and responsive web applications continues to grow, exploring the scalability benefits of micro-frontends in cloud architectures becomes imperative for meeting user expectations and business requirements.
3. Cost-Efficiency Considerations: Cost optimization is a key concern for organizations operating in the cloud. Investigating the cost-effectiveness of micro-frontends could lead to more efficient resource utilization and budget management strategies.
4. Heightened Focus on Security: In an era of heightened cybersecurity threats, enhancing the security measures of cloud-based applications is paramount. Understanding how micro-frontends impact security indexes is vital for ensuring robust protection against potential vulnerabilities.
5. Industry Competitiveness: Delving into this topic is essential for organizations looking to differentiate themselves in the competitive landscape by leveraging the advantages of micro-frontends within cloud infrastructures.
6. User Experience Enhancement: Improved user experience is a top priority for web applications. Exploring how micro-frontends influence user experience within cloud environments can lead to enhanced customer satisfaction and retention.
7. Alignment with Industry Trends: Researching this topic aligns with current industry trends and best practices, allowing organizations to adapt proactively to new methodologies and technologies shaping the future of cloud-based web development.

By addressing these urgent considerations, this research can provide timely insights and recommendations to industry professionals, researchers, and decision-makers navigating the complexities of cloud-based application development.

## **Scope of the Study**

This study will encompass a comprehensive examination of the impact of micro frontend development within cloud-based application architectures. It will concentrate on evaluating the influence of micro-frontends on the scalability, cost-effectiveness, and security of such architectures. The research will employ a multifaceted approach, including case studies, performance evaluations, and the development of best practices, to address these key aspects. The study's scope will encompass a range of real-world scenarios and cloud platforms to provide a holistic understanding of the challenges and opportunities arising from the integration of micro-frontends into cloud-based applications. In the subsequent sections, we delve into the methodologies, case studies, and analyses that will be employed to investigate and evaluate the impact of micro-frontends on cloud-based application architecture. This research aims to contribute valuable insights to the ever-evolving landscape of web application development, helping organizations in making informed architectural choices and achieving optimal outcomes.

## **Aim of the study**

This research aims to investigate the impact of applying micro-frontends to cloud-based application architectures with a focus on scalability, cost-effectiveness and security indexes. By delving into a wide array of real-world scenarios and conducting thorough interviews with 250 developers and technical architects, the objective of this research is to provide profound insights into the evolving landscape of web application development within the cloud environment.

The research questions are formulated based on the aim of this study which are as follows:

1. **[Need a question related to scalability]**
2. How can we automatically assess conformance to Architectural Design Decisions on security tactics in the context of micro-frontend systems?
3. How to address the optimal resource provisioning to achieve the cost optimality within a performance guarantee?

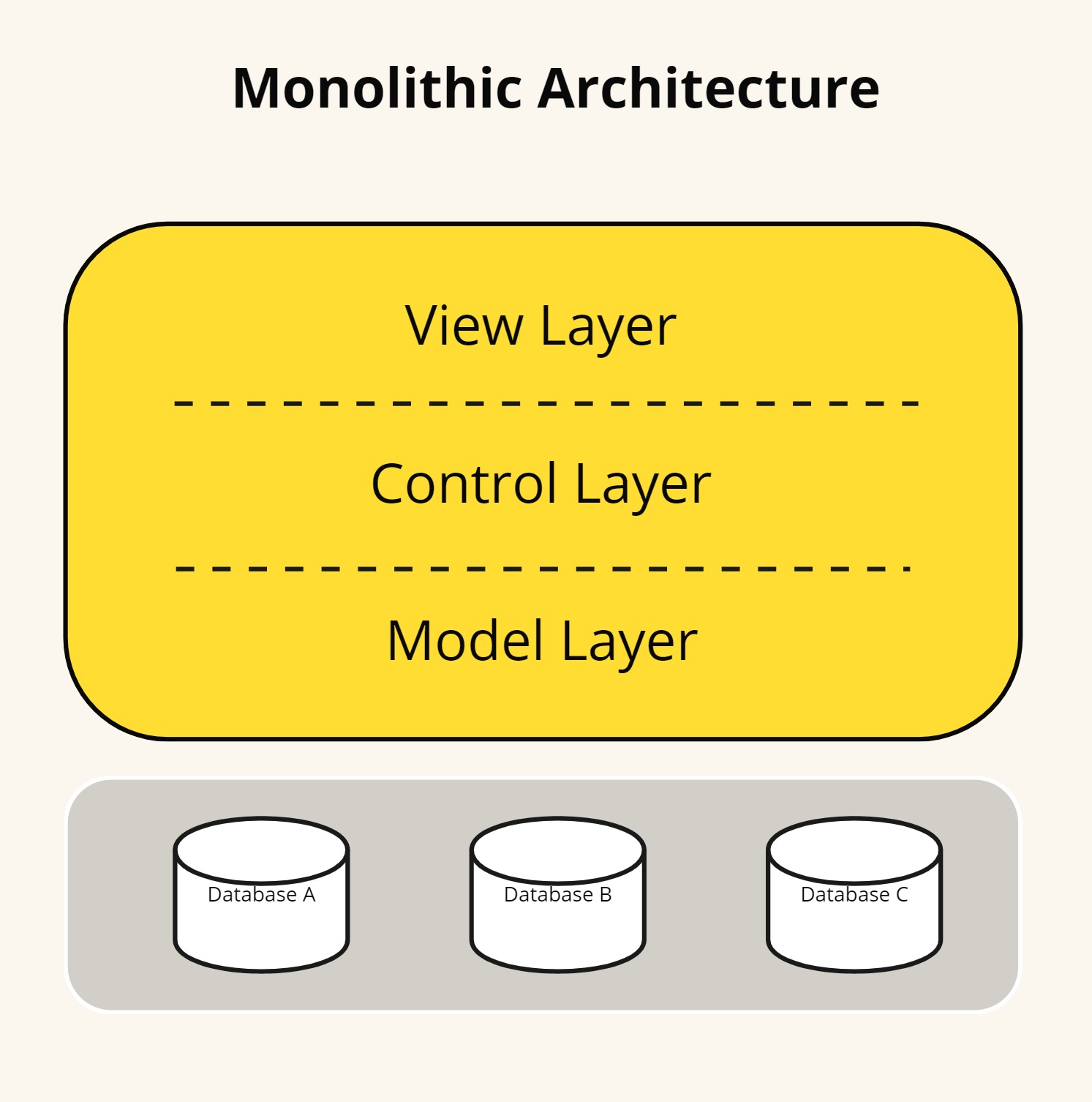
# **: RELATED WORK**

In this chapter, we provide the background information of micro-frontend and establish the context for our research work. By reviewing related work, researchers can understand the existing knowledge in their field and situate their own work within the broader scientific community.

## **Definitions**

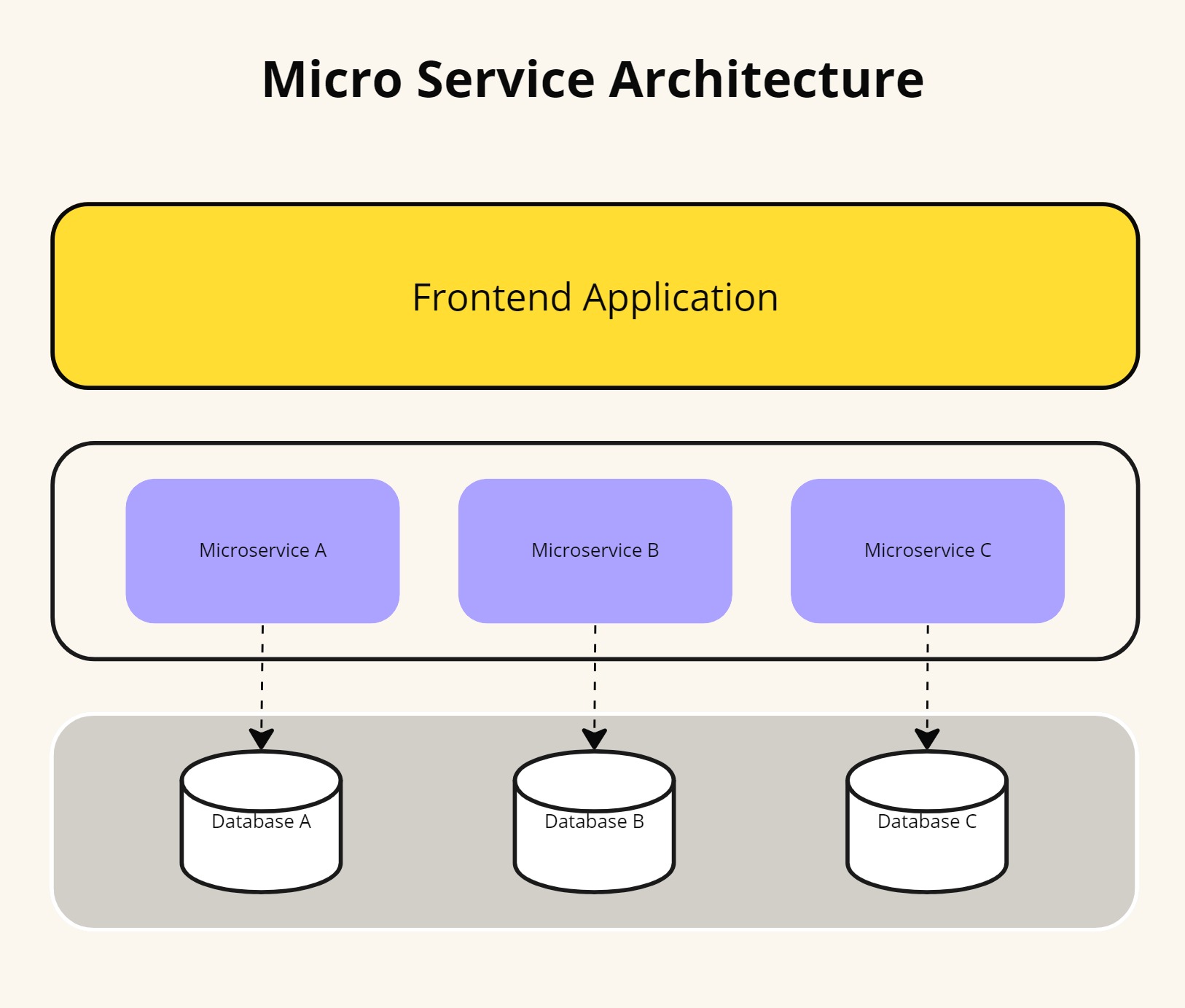
### **Background of Microservice**

Previously, software applications were commonly created with a Monolithic Architecture approach, which involved merging all application processes into a single self-contained unit. This monolith integrated the user interface, business logic, and data layer within one application (Alpers et al., 2015). While this architecture offered simplicity in deployment and operation, it also presented limitations.



*Fig.*

Monolithic architectures have significant issues, such as a lack of flexibility, instability, and inefficiency (Thatikonda, 2023). The more giant the monolith, the more complicated it gets to comprehend and modify. A significant challenge with scalability is that it requires altering the entire application rather than just specific components. Transitioning arose from challenges to Microservices Architecture experienced with monolithic systems. The goal is to dissect complex applications into more minor, more manageable services that are loosely connected. These services can be developed, deployed, and scaled independently (Megargel, Shankararaman, & Walker, 2020). This transition aimed to enhance software systems’ maintainability, scalability, and productivity. It also facilitates rapid and secure execution of changes, ensuring that the system can adapt to increasing workloads.

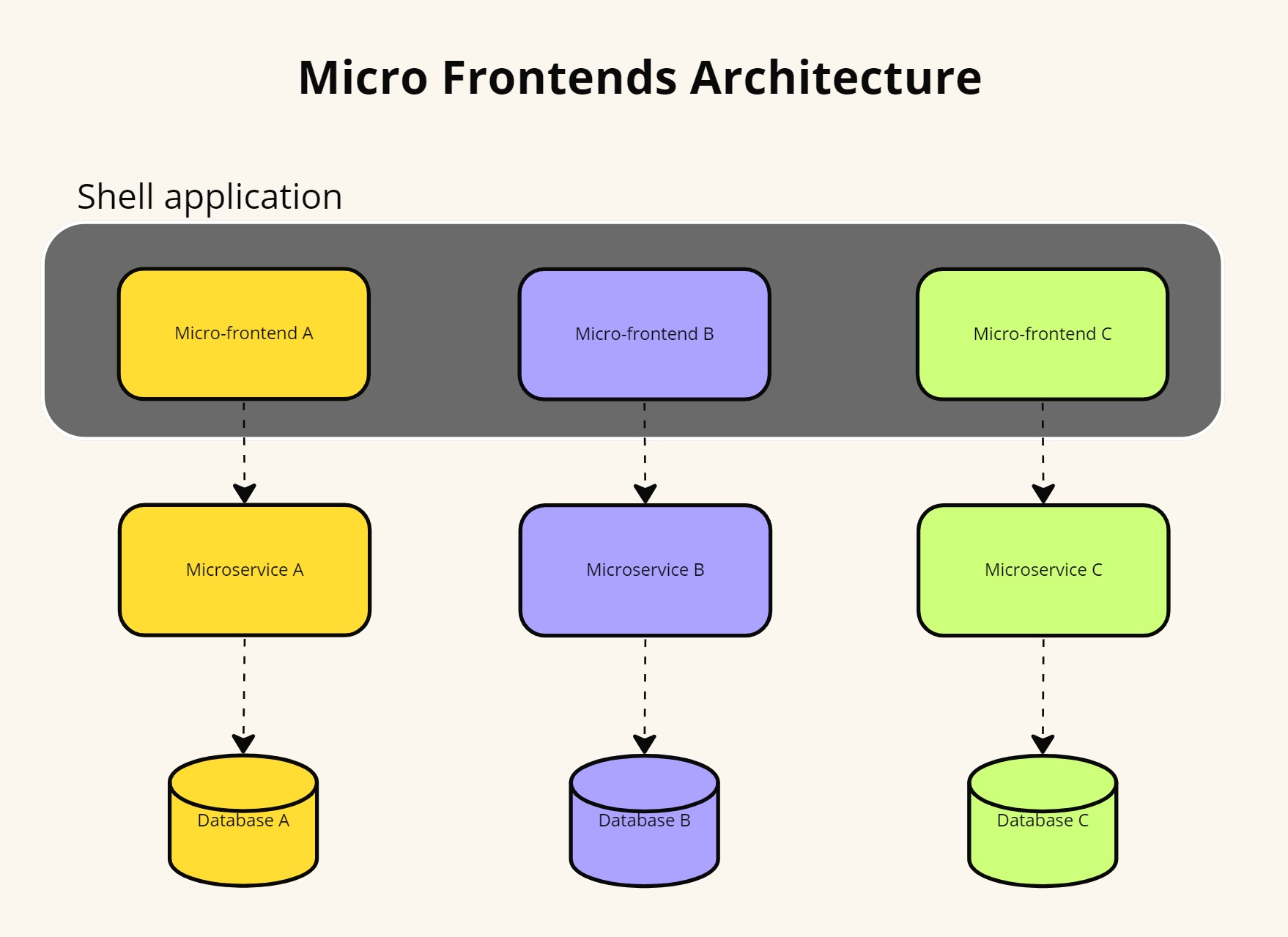
*Fig.*

### **Micro-frontends architecture**

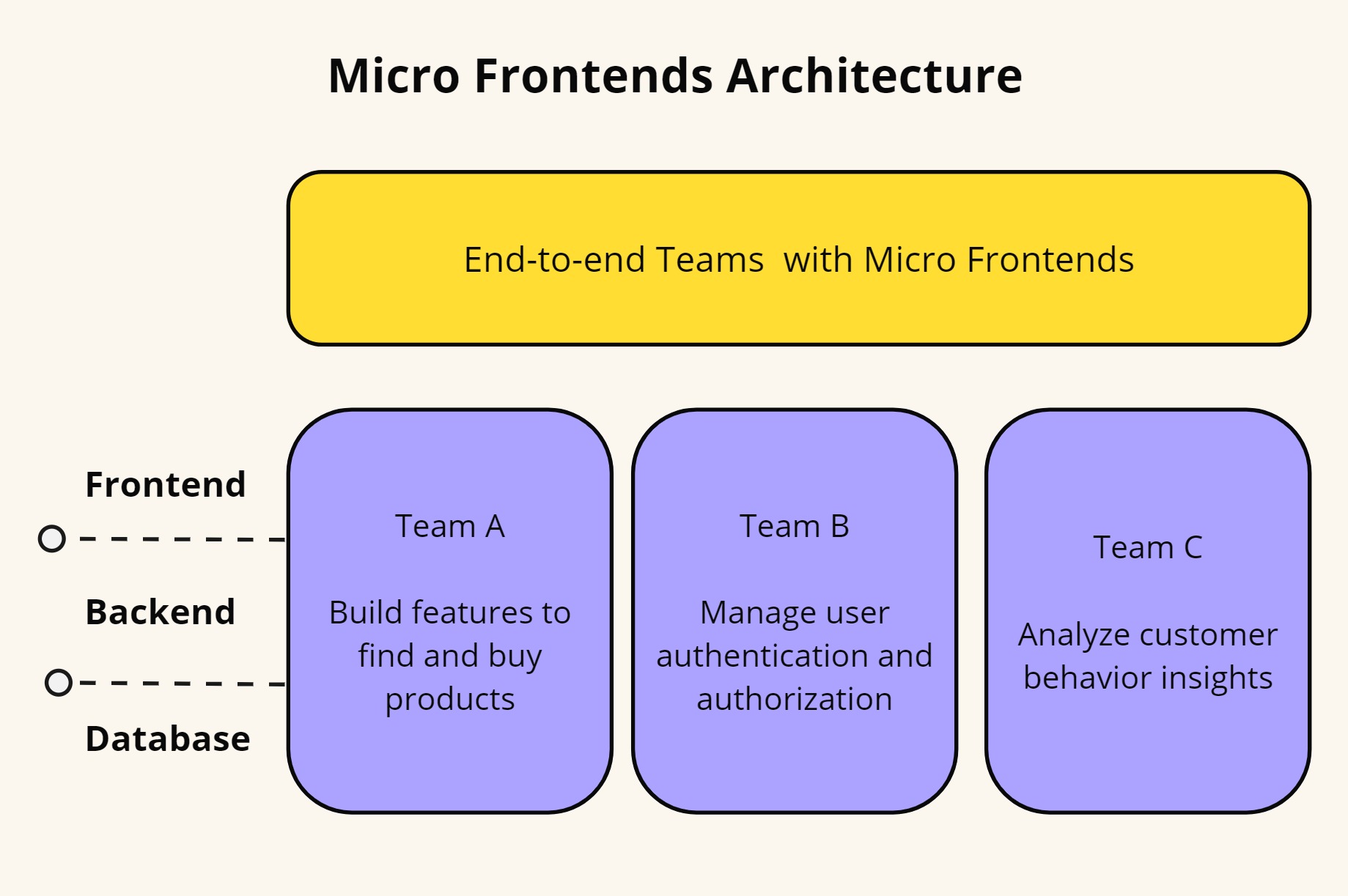
Microservices are commonly used to separate different components of an application, primarily in the backend. They structure a distributed application as a collection of services, dividing the backend into distinct functional units. Each application function becomes an independent service, which helps avoid bottlenecks in the database.

Micro frontends, however, extend the microservice concept to the frontend domain. In today’s web applications, the frontend is growing larger, while the backend is becoming less significant. Inspired by the microservices model, micro frontends offer a solution for modern complex web applications by allowing them to be divided into smaller, independent modules. This architectural approach involves splitting a web application into individual modules or functions, each implemented autonomously, providing frontend teams with flexibility and speed similar to what microservices offer to backend teams.

The term micro frontends or micro-frontends (MFE) first appeared in 2016, in the [Thought Works Technology Radar](https://www.thoughtworks.com/es-es/about-us/news/2016/technology-radar-nov-2016) guide (Cam Jackson, 2019). A micro frontend can be a complete page or specific fragments of the page, which other teams can use to add somewhere in the page they are developing. Unlike reusable components, micro frontends can be implemented independently as individual projects. The technique for implementing micro frontends is to develop everything separately, extracting the other components at runtime to use them. The micro frontend consists of several independent and modular components that will be displayed, as required. That is, for a particular page only the necessary components are loaded. These components interact directly with the data and do not require a centralized server to route requests or process the data. In addition to the components that display content, a micro frontend can also have several utility components to interact with the application environment, it could be a user or other components.

*Fig.*

Thus comes the vertical organization, with a less complex architecture thanks to micro frontends, which divide the application into small independent functions, each one implemented simultaneously, from the backend to the frontend, by a work team. This approach has become popular because of the problems with the monolithic approach. The frontend has grown rapidly and with a monolithic architecture, it becomes more difficult to maintain. With micro frontends, the same scalability, security, and cost-effectiveness is guaranteed as with the backend microservices architecture. The application created is less cumbersome and more user-friendly. In addition, each micro frontend can be developed with different frameworks.

*Fig.*

This effort enhances development speed and streamlines the digital product's complexity by reducing inter-team dependency.

### **Cloud-based architecture with Micro-frontends**

The architectural landscape of cloud-based applications has witnessed a transformative shift, marked by the integration of micro-frontends as a key architectural paradigm. Traditionally, cloud-based applications have relied on monolithic structures, where the frontend, backend, and database components are tightly coupled into a single unit. However, the emergence of micro-frontends has challenged conventional approaches to frontend development, promoting a more modular approach (Perlin et al., 2023). Micro-frontends enable the decomposition of the user interface into smaller, independent components, each serving a specific feature or function (Cam Jackson, 2019). The adoption of cloud-based applications has brought about a significant shift in architectural design. This shift has allowed for increased scalability, improved performance, and enhanced flexibility, making cloud-based applications a valuable asset for businesses looking to stay ahead of the curve. The modular nature of micro-frontends facilitates independent development, testing, and deployment of frontend components, enhancing agility and scalability.

### **Scalability in cloud-based applications**

Cloud scalability is the ability of a cloud computing system to adapt to changing computing requirements by either increasing or decreasing its resources, such as computing power, storage, or network capacity on demand (Kerimovs, 2023). It allows the system to adjust its resources to the workload to meet the required performance levels. This scalability often involves increasing or decreasing the number of servers, storage, or other computing resources.

This type of scalability is essential because it allows organizations to quickly adjust to the changes in their computing needs while also providing efficient use of computing resources. The goal of cloud scalability is to make sure that the cloud service can scale cost-effectively and ensure that the service can handle greater loads by adding physical or virtual resources (Rashid Dar, 2016). And this scalability is a crucial advantage of cloud computing. It allows businesses to quickly and easily scale their operations as needed without making significant upfront investments in hardware and other infrastructure.

### **Vertical scaling in cloud computing**

Vertical scaling in cloud computing involves augmenting the resources of an existing instance or server to enhance its capacity and functionalities. This process facilitates the system to dynamically adjust its resource allocation based on evolving demands. Typically, vertical scaling entails enhancing the computing power of the server by increasing RAM, CPU cores, or storage capacities like hard disks or solid-state drives (Debski et al., 2018).

Vertical scaling enables your applications to run faster and handle more load without purchasing a new server or instance. And vertical scaling is a popular choice for cloud computing because it is relatively easy to do and does not require any changes to the existing infrastructure.

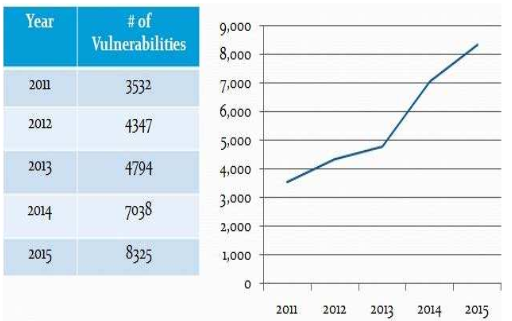
### **Horizontal scaling in cloud computing**

Horizontal scaling in cloud computing involves expanding a system by incorporating additional nodes or servers into the infrastructure. This approach is commonly employed to enhance the processing capacity of the cluster, enabling applications and services to manage a higher volume of concurrent requests or to process larger quantities of data efficiently (Rashid Dar, 2016).

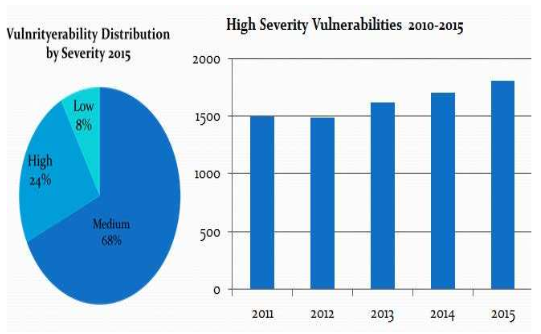
In cloud computing, horizontal scaling is usually achieved by adding additional virtual machines (VMs), containers, or other resources to an existing cluster (Li et al., 2020). This type of scaling is often used to improve performance or to handle increased traffic. When done correctly, horizontal scaling can be an effective way to enhance a system's performance.

### **Security in cloud-based applications**

The global distribution of these applications makes them prone to attacks that uncover and maliciously exploit a variety of security vulnerabilities [1]. ISO 27005 defines vulnerability as “a weakness of an asset or group of assets that can be exploited by one or more threats where an asset is anything that can has value to the organization, its business operations and their continuity, including information resources that support the organization's mission” [2]. According to National Vulnerability Database (NVD) [3] the number of vulnerabilities has approximately three times increased since 2011.



Although OWASP’s 2012 report indicated that investment in security was increasing but NTA Monitor’s 2014 Web Application Security Report demonstrated that Web security had actually decreased compared to the previous year. In fact, Web application vulnerabilities represent huge problems for companies and organizations. According to WhiteHat Security’s most recent Website Security Statistics Report, 63 percent of assessed websites are vulnerable, each having an average of six unsolved flaws [4]. These vulnerabilities create and feed an underground economy based on attacking and stealing data and resources. Figure \_ shows the vulnerability distribution by severity of the year 2015.



On the other hand, the report of \_ states that the security of the users’ data is at risk and securing the web applications in all areas including institutional activities is the foremost priority of the security practitioners. This is due to the size and complexity of microservice systems, the need for continuous evolution and frequent release of these systems. In this context, manually validating whether security features are used as intended throughout the system is a time-consuming and error-prone task. For architecturally relevant security features, architectural abstraction can help focus only on the relevant aspects, but still substantial effort is required e.g., to check a large-scale system’s architecture for conformance to security tactics.

### **Cost-effectiveness in cloud-based applications**

In additional, some existing researches focus on the issue of cost-benefit analysis in cloud computing. In [21], Selvarani and Sadhasivam discussed a job grouping algorithm which was used to allocate the task-groups to different available resources. This scheduling algorithm measured both resource cost and computation performance, it also improved the computation/communication ratio by grouping the user tasks according to a particular cloud resource’s processing capability and sent the grouped jobs to the resource. A cost-based resource scheduling paradigm was presented in [22] by leveraging market theory to schedule compute resources and meet user’s requirement. The set of computing resources with the lowest price were assigned to the user according to current suppliers’ resource availability and price. An algorithm and protocol were designed for cost-based cloud resource scheduling. The scheduling algorithm and protocol were described in the pure Java based platform, which had three-tiered hierarchical and extensible architecture. In [23], a minimum cost maximum flow algorithm was proposed for resources (e.g. virtual machines) placement in clouds. Hadji and Zeghlache focused on the optimal dynamic placement of virtual resources in data centers and cloud infrastructures to serve multiple users and tenants with time varying demands and workloads. Providers could use the minimum cost maximum flow algorithm to opportunistically select the most appropriate physical resources. m for service operators to minimize their service provision cost. In the first phase, a mathematical formula was proposed to compute the optimal amount of long-term reserved resources. In the second phase, the Kalman filter was used to predict resource demand and adaptively change the subscribed on-demand resources such that provision cost could be minimized. They exploited a predictive based resource management to adaptively configure VMs. To the best of our knowledge, the problems of achieving a cost-effective cloud system with finite buffer analyses and impatient job concerns have not been studied.

# **: SCALABILITY FOR MICRO-FRONTENDS**

[Need to add content here]

# **: SECURITY TACTICS MODEL FOR MICRO-FRONTENDS**

## **Data Collection and Analysis of Security Tactics for Micro-frontends**

To initiate our data collection and analysis of microservice security tactics, we began by consulting existing microservice-specific recommendations from industry organizations such as NIST, OWASP, and the Cloud Security Alliance. These recommendations aggregate industry best practices at a broad level.

Currently, we are conducting a comprehensive study of gray literature related to microservice security practices and tactics. Our analysis has delved into 30 practitioner sources in-depth. From the initial data collected, our focus is on addressing security tactics within the core software architecture. Specifically, we aim to model these tactics within the software architecture decomposition view.

## **Case Study Preparation and Development of Model Data Set**

We conducted an in-depth study of 10 open-source microservices systems. For each system, we meticulously analyzed the source code, manually annotating each security feature. These systems were published by practitioners with a background in microservices.

Our approach involved using an existing static code analysis method for reconstructing the architecture of polyglot microservice systems. We leveraged this foundation to extract models from the source code of the 10 studied open-source system cases and identify security tactics within these models.

To develop our modeling approach, we embarked on an iterative study, drawing insights from various microservice-related knowledge sources. Gradually, we refined a meta-model that encompasses all the essential elements necessary for reconstructing existing microservice-based systems.

To expand our dataset and explore the design space, we created 20 additional models representing system variants. These variants were adapted from a published example, guided by discussions in relevant literature. Aside from the specific variations described in Tables 1 and 2, all other system aspects remained consistent with the base models. In total, we analyzed 30 models, summarized in Tables 1 and 2. We consider these evaluation systems to be practical examples that either reflect or closely resemble real-world microservice architectures.

## **Definition of ADDs and Metrics**

In our study, we carefully curated a subset of security tactics for microservices. These tactics were selected based on input from industrial security experts who co-authored our article and analyzed collected data. The chosen subset represents widely used security practices in the context of microservices.

Next, we formulated a set of metrics with the specific goal of automating decision-making at each individual decision point within our Architectural Decision Diagrams (ADDs). These metrics are formally defined in Section 6 of our work.

To validate and refine these metrics, we conducted extensive testing using numerous small examples related to our tactics. The formalization process involved a combination of set theory and first-order logic.

By combining expert judgment, empirical data, and rigorous formalization, our approach enhances decision-making in microservice architectures.

## **Case Study-Based Analysis and Evaluation**

In our study, we conducted a systematic assessment to evaluate the support or violation of the collected security tactics. Here are the steps we followed:

1. Recommendation as Ordinal Ratings: The other authors applied our recommendations as an ordinal rating scheme to each model variant summarized in Tables 1 and 2. This process allowed us to create a ground truth for our study.
2. Expert Review: Three industrial security experts from our author team and two experts from another company reviewed the rating scheme and the ratings in the ground truth. Their insights and expertise contributed to the robustness of our findings.
3. Statistical Analysis:

* Spearman Rank Correlation: We initially examined how well the independent variables correlate with the dependent variable using Spearman rank correlation. This method is widely used for analyzing the relationship between continuous and discrete ordinal variables.
* Ordinal Regression: To assess how well the hypothesized metrics predict the ground truth data, we performed an ordinal regression analysis. Ordinal regression models the dependence of an ordinal response on a set of independent predictors. We utilized the lrm function from the rms package for this analysis.

By combining expert judgment and rigorous statistical methods, our study provides valuable insights into the effectiveness of the proposed metrics.

## **Model Generation**

To model microservice architectures, we adopted the approach described in our prior work. Leveraging our existing tool called CodeableModels, a Python implementation for precisely defining meta-models, models, and model instances in code, we achieved the following:

1. Automated Code Generators: These tools automatically generate graphical visualizations of all meta-models and models using PlantUML. This visualization aids in understanding the architecture and relationships between components.
2. Detectors: We developed detectors capable of identifying relevant aspects of metrics within the models. These detectors analyze the architecture and provide valuable insights for evaluation and optimization.
3. Metric Calculators: Additionally, we created generators that automatically calculate metrics based on the architecture. These metrics help assess the performance, scalability, and other critical characteristics of the microservices.

By combining these components, our approach facilitates effective modeling and analysis of microservice architectures.

## **Extraction of Architecture Models from Microservice Code**

This article builds upon our previous work, focusing on the automatic extraction of architecture models from code. Our approach extends the method introduced by Haitzer et al. We propose a detector-based technique that supports polyglot and continuously evolving systems. Additionally, our approach allows for the use of reusable detectors. In this context, detectors are software components that parse relevant sections of source code and generate model abstractions. Reusable detectors can be employed across various model abstraction tasks and projects.

Our approach assumes that a system expert has already identified the high-level architecture and the architectural security features of the system. This expert familiarity is essential. The next step involves modeling this knowledge in an execution script that iterates the detectors over each system element

# **: COST MODEL FOR MICRO-FRONTENDS**

## **Data Collection of cost-effective for Micro-frontends**

In a cloud system, operational costs include resource provisioning (related to server quantity, power consumption, and buffer capacity), system losses (due to impatient users, system blocking, and VM activation), and performance considerations (including system rejection penalties and congestion).

In general, virtual machines (VMs) are activated once a job request has been accepted and forwarded to the buffer. However, some impatient users enter the queue but then abandon the VMs immediately (a phenomenon known as “renege”). Consequently, evaluating the cost overhead associated with activating VMs without subsequent release becomes crucial for systems with impatient users. The following summarizes the cost notations:

C1 = Expected server provisioning cost per server per unit time;

C2 = Expected power consumption cost per service rate per unit time;

C3 = Expected cost incurred by preparing per buffer space per unit time;

C4 = Impatient users and system blocking losses incurred by per request;

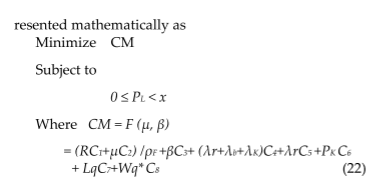
C5 = Starting-up cost incurred by activating per VM;

C6 = System rejection penalty;

C7 = Cost incurred by holding jobs in the system per unit time;

C8 = Cost incurred by jobs waiting in the system per unit time;

Given that system performance, loss probability, and operational costs significantly rely on buffer space and service rate, an expected cost function per unit time is developed. In this function, both the service rate and buffer size serve as the primary decision variables. Naturally, users prefer not to encounter blockages or abandon services due to insufficient buffer space or intolerable system delays. Therefore, a loss probability guarantee is essential in a service system, representing one of the most critical performance metrics for measuring service levels. Specifically, the Service Level Agreement (SLA) constraint ensures that the loss probability remains below a certain threshold, denoted as SLA (x%), where x represents the maximum acceptable value. Mathematically, cost minimization (CM) with a loss probability guarantee can be expressed as:



## **4.2 The Proposed CEA policy**

In this report, we are introducing the Cost-Effective Abandoned (CEA) policy, which is designed to address the optimal solution (μ\*, β\*) in order to minimize operational costs without violating SLA constraints. The main focus of this policy is to resolve the conflicting objectives of reducing system losses while conserving operational costs. The cost function's complexity makes it difficult to get an analytical solution for the optimal parameters. Therefore, we are proposing the CEA heuristic algorithm, which aims to find the minimum total cost by solving nonlinear constrained optimization problems. This algorithm considers various incurred costs and user behavior variations in order to provide a comprehensive and effective solution

