فصلنامه نوآوری در مهندسی و تکنولوژی شماره ۳- دوره ۱- صفحات۱۳۱-۱۲۱
Journal of Innovation in Engineering and Technology

تأثیر معماری نرم‌افزار مبتنی بر میکروسرویس‌ها بر کارایی و مدیریت پروژه‌های نرم‌افزاری در سازمان‌های بزرگ: یک تحلیل چندمعیاره

محمد بزرگی ولمی[۱]

[۱]- دانشجوی کارشناسی ارشد نرم افزار دانشگاه پیام نور واحد تهران شمال

چکیده

در دهه‌ی اخیر، معماری نرم‌افزار مبتنی بر میکروسرویس‌ها (Microservices Architecture) به‌عنوان رویکردی تحول‌آفرین در طراحی و پیاده‌سازی سامانه‌های نرم‌افزاری پیچیده مطرح شده است. سازمان‌های بزرگ که با چالش‌هایی چون مقیاس‌پذیری، پایداری، سرعت تحویل، و مدیریت تیم‌های توسعه گسترده مواجه‌اند، به‌تدریج از معماری‌های یکپارچه سنتی به معماری میکروسرویسی روی آورده‌اند. هدف این پژوهش، تحلیل چندمعیاره‌ی تأثیر این معماری بر کارایی و مدیریت پروژه‌های نرم‌افزاری در سازمان‌های بزرگ است. برای این منظور، ضمن بررسی ادبیات نظری و مدل‌های تصمیم‌گیری چندمعیاره مانند AHP و TOPSIS، ابعاد فنی، مدیریتی، سازمانی و اقتصادی مورد ارزیابی قرار گرفته‌اند. نتایج نشان می‌دهد که معماری میکروسرویس‌ها با افزایش چابکی توسعه، بهبود قابلیت نگهداری، استقلال تیم‌ها، و کاهش زمان تحویل، کارایی پروژه‌ها را به‌طور معناداری افزایش می‌دهد. با این حال، چالش‌هایی نظیر پیچیدگی مدیریت زیرساخت، هماهنگی سرویس‌ها و هزینه‌های بالای DevOps می‌تواند تأثیر منفی بر بهره‌وری کل داشته باشد. این مقاله با تأکید بر تحلیل چندمعیاره، چارچوبی برای تصمیم‌گیری راهبردی در پذیرش یا بهینه‌سازی معماری میکروسرویسی در سازمان‌های بزرگ پیشنهاد می‌کند.

واژگان کلیدی: معماری میکروسرویس‌ها، کارایی نرم‌افزار، مدیریت پروژه‌های نرم‌افزاری، تحلیل چندمعیاره، سازمان‌های بزرگ

مقدمه

پیشرفت سریع فناوری اطلاعات و نیاز فزاینده‌ی سازمان‌ها به توسعه‌ی سامانه‌های نرم‌افزاری با قابلیت انطباق و مقیاس‌پذیری بالا، سبب شده است تا الگوهای معماری نرم‌افزار دستخوش تحولات بنیادین شوند. معماری‌های یکپارچه (Monolithic) که در گذشته پایه‌ی اصلی طراحی سامانه‌ها بودند، در مواجهه با رشد سریع نیازهای کاربری، محدودیت‌هایی جدی در مقیاس‌پذیری، نگهداری، و استقرار به‌موقع تغییرات از خود نشان داده‌اند. در پاسخ به این چالش‌ها، معماری میکروسرویس‌ها به‌عنوان رویکردی مدرن و منعطف ظهور کرد که هدف آن، تقسیم سامانه به مجموعه‌ای از سرویس‌های مستقل ولی مرتبط است که هر یک مسئولیت مشخصی دارند. در سازمان‌های بزرگ، این رویکرد نه تنها باعث افزایش سرعت توسعه و استقرار می‌شود بلکه هماهنگی بین تیم‌ها را نیز ساده‌تر می‌کند. با این حال، پذیرش این معماری صرفاً یک تصمیم فنی نیست؛ بلکه تصمیمی راهبردی است که باید با معیارهایی چندگانه از جمله هزینه، کارایی، قابلیت نگهداری، امنیت، و پایداری ارزیابی شود. ازاین‌رو، تحلیل چندمعیاره به‌عنوان ابزاری مؤثر برای سنجش جامع تأثیرات معماری میکروسرویس‌ها در محیط‌های پیچیده سازمانی به‌کار گرفته می‌شود. این مقاله تلاش دارد با نگاهی جامع، مزایا، چالش‌ها، و پیامدهای مدیریتی استفاده از معماری میکروسرویس‌ها را در سازمان‌های بزرگ بررسی و تحلیل کند[۱].

۲- تحول در معماری نرم‌افزار و ظهور میکروسرویس‌ها

تحول از معماری‌های سنتی به میکروسرویس‌ها نتیجه‌ی نیاز به انعطاف‌پذیری و پاسخ‌گویی سریع‌تر به تغییرات بازار است. در معماری یکپارچه، تمام اجزای نرم‌افزار در یک کدبیس واحد توسعه یافته و استقرار هر تغییر کوچک، نیازمند بازسازی و استقرار کل سامانه است.[۲] این امر، در سازمان‌های بزرگ که سامانه‌های حیاتی و پرکاربر دارند، منجر به کندی در توسعه و افزایش ریسک خرابی می‌شود. در مقابل، معماری میکروسرویس‌ها با تفکیک وظایف به سرویس‌های مستقل، امکان توسعه‌ی موازی، استقرار جداگانه و مقیاس‌پذیری پویا را فراهم می‌سازد. این استقلال، سازمان را قادر می‌سازد تا هر سرویس را به‌صورت مجزا ارتقا دهد، بدون آن‌که سایر بخش‌ها مختل شوند. از منظر فنی، بهره‌گیری ازAPI ها، کانتینری‌سازی با Docker و ارکستراسیون با Kubernetes موجب می‌شود تا تیم‌ها بتوانند فرآیند استقرار را خودکار و پایدار کنند. در نتیجه، چرخه‌ی عمر نرم‌افزار کوتاه‌تر، قابلیت اطمینان بالاتر و پاسخ‌گویی سازمان به تغییرات سریع‌تر می‌شود. این تحول نه‌تنها جنبه‌ی فنی بلکه پیامدهای سازمانی و فرهنگی عمیقی دارد که در بخش‌های بعد بررسی می‌شود[۳].

از منظر فنی و عملکردی، معماری نرم‌افزار مبتنی بر میکروسرویس‌ها یکی از مهم‌ترین تحولات در حوزه مهندسی نرم‌افزار محسوب می‌شود که توانسته است بسیاری از محدودیت‌های معماری‌های یکپارچه را برطرف سازد و الگوی تازه‌ای از توسعه، استقرار و نگهداری

---

[۱] Kamisetty, A., Narsina, D., Rodriguez, M., Kothapalli, S. and Gummadi, J.C.S., ۲۰۲۳. Microservices vs. Monoliths: Comparative Analysis for Scalable Software Architecture Design. *Engineering International*, *11*(۲), pp.۹۹-۱۱۲.

[۲] Tapia, Freddy, Miguel Ángel Mora, Walter Fuertes, Hernán Aules, Edwin Flores, and Theofilos Toulkeridis. "From monolithic systems to microservices: A comparative study of performance." *Applied sciences* ۱۰, no. ۱۷ (۲۰۲۰): ۵۷۹۷.

[۳] Megargel, A., Shankararaman, V. and Walker, D.K., ۲۰۲۰. Migrating from monoliths to cloud-based microservices: A banking industry example. In *Software engineering in the era of cloud computing* (pp. ۸۵-۱۰۸). Cham: Springer International Publishing.

سامانه‌های نرم‌افزاری ارائه دهد. در معماری‌های سنتی، کل سامانه به‌صورت یکپارچه طراحی می‌شود و هرگونه تغییر جزئی در یک بخش مستلزم بازسازی و استقرار کل سیستم است؛ در حالی که در معماری میکروسرویس‌ها، کل نرم‌افزار به مجموعه‌ای از سرویس‌های کوچک، مستقل و خودمختار تقسیم می‌شود که هرکدام وظیفه‌ای مشخص دارند و از طریق واسط‌های سبک مانند APIها با یکدیگر در ارتباط‌اند.[۴] این ساختار ماژولار موجب می‌شود تا هر سرویس بتواند به‌صورت مجزا توسعه یافته، تست و استقرار یابد و در صورت نیاز مقیاس‌پذیر شود. به این ترتیب، سازمان‌های بزرگ می‌توانند بار کاری را به‌صورت پویا میان سرویس‌ها توزیع کنند و تنها بخش‌هایی را که تحت فشار بیشتر قرار دارند، افزایش ظرفیت دهند. این ویژگی باعث افزایش کارایی سیستم، بهینه‌سازی مصرف منابع و کاهش هزینه‌های زیرساختی می‌شود. علاوه بر این، میکروسرویس‌ها امکان به‌کارگیری فناوری‌ها و زبان‌های برنامه‌نویسی متنوع را برای هر سرویس فراهم می‌کنند، به‌گونه‌ای که تیم‌های مختلف می‌توانند متناسب با نیاز عملکردی خود بهترین ابزار را انتخاب کنند، امری که در معماری‌های یکپارچه به‌دلیل وابستگی زیاد بین ماژول‌ها ممکن نیست. در نتیجه، سطح بهینه‌سازی عملکرد در هر بخش افزایش یافته و قابلیت به‌روزرسانی و نگهداری سامانه به شکل قابل توجهی بهبود می‌یابد. با این حال، این انعطاف‌پذیری و استقلال سرویس‌ها هزینه‌هایی نیز دارد؛ زیرا ارتباطات میان‌سرویسی که از طریق شبکه انجام می‌شوند، موجب افزایش سربار ارتباطی و احتمال بروز تأخیر در پاسخ‌گویی می‌شود. از سوی دیگر، مدیریت داده‌ها در این ساختار دشوارتر است، چرا که تراکنش‌ها به‌صورت توزیع‌شده انجام می‌شوند و حفظ یکپارچگی داده میان سرویس‌ها نیازمند طراحی دقیق الگوهای هماهنگی مانند SAGA و Event Sourcing است. همچنین، نظارت بر عملکرد سیستم، خطایابی و اشکال‌زدایی در یک محیط متشکل از صدها سرویس کوچک چالش‌برانگیز است و نیاز به ابزارهای تخصصی نظیرPrometheus ، Grafanaو ELK Stack دارد تا بتوان به‌صورت متمرکز وضعیت سلامت سیستم را پایش کرد. یکی دیگر از چالش‌های فنی مهم، تضمین امنیت در سطح سرویس است؛ چرا که افزایش تعداد نقاط ارتباطی، سطح حمله‌ی بالقوه را نیز افزایش می‌دهد و در نتیجه سازمان‌ها باید سیاست‌های امنیتی و کنترل دسترسی را در سطح شبکه و سرویس با دقت پیاده‌سازی کنند.[۵] با وجود این چالش‌ها، شواهد تجربی نشان می‌دهد که سازمان‌هایی که موفق به طراحی صحیح معماری میکروسرویس‌ها و به‌کارگیری زیرساخت‌های DevOps شده‌اند، توانسته‌اند میانگین زمان استقرار (Deployment Time) را تا ۷۰ درصد کاهش دهند و قابلیت اطمینان سامانه را به‌طور معناداری افزایش دهند. این دستاوردها به‌ویژه در پروژه‌های با مقیاس بالا که نیازمند پاسخ‌گویی سریع به تغییرات بازار هستند، اهمیت بیشتری دارد. افزون بر آن، قابلیت مقیاس‌پذیری افقی در میکروسرویس‌ها به سازمان‌ها اجازه می‌دهد تا با رشد تقاضا، بدون نیاز به ارتقای کل سامانه، تنها سرویس‌های خاص را گسترش دهند که این امر کارایی کلی سیستم را افزایش می‌دهد. در مجموع، معماری میکروسرویس‌ها اگرچه پیچیده‌تر و نیازمند زیرساخت فنی پیشرفته‌تر از معماری‌های سنتی است، اما در بلندمدت با افزایش کارایی، پایداری، قابلیت نگهداری و انعطاف‌پذیری، مزیتی رقابتی برای سازمان‌های بزرگ فراهم می‌سازد و آن‌ها را قادر می‌سازد تا چرخه توسعه نرم‌افزار خود را بهینه، سریع‌تر و کارآمدتر سازند[۶].

۳- تأثیر معماری میکروسرویس‌ها بر کارایی فنی و عملکردی سامانه‌ها

[۴] Elgheriani, N.S. and Ahmed, N.A.S., ۲۰۲۲. Microservices vs. monolithic architectures [the differential structure between two architectures]. *MINAR International Journal of Applied Sciences and Technology*, *4*(۳), pp.۵۰۰-۵۱٤.

[۵] Sigala, N.S., ۲۰۲۵. Microservices Architecture in Cloud Computing: A Software Engineering Perspective on Design, Deployment, and Management. *International Journal of Research and Innovation in Social Science*, *9*(۱۵), pp.۲۱۵-۲۳۹.

[۶] Rehman, A.U., Aguiar, R.L. and Barraca, J.P., ۲۰۱۹. Network functions virtualization: The long road to commercial deployments. *IEEE Access*, *7*, pp.٦٠٤٣٩-٦٠٤٦٤.

از منظر فنی، معماری میکروسرویس‌ها به‌طور مستقیم بر شاخص‌های کارایی نرم‌افزار اثر می‌گذارد. یکی از مهم‌ترین مزایا، افزایش مقیاس‌پذیری (Scalability) است، به‌گونه‌ای که هر سرویس می‌تواند به‌صورت مستقل با بار کاری متناسب خود مقیاس یابد. این ویژگی، مصرف منابع را بهینه و هزینه‌های عملیاتی را کاهش می‌دهد. علاوه بر آن، جداسازی سرویس‌ها موجب افزایش قابلیت اطمینان (Reliability) می‌شود؛ زیرا خرابی یک سرویس لزوماً کل سامانه را از کار نمی‌اندازد. در زمینه‌ی عملکرد، میکروسرویس‌ها امکان استفاده از فناوری‌ها و زبان‌های برنامه‌نویسی متفاوت را فراهم می‌کنند که منجر به بهینه‌سازی عملکرد هر بخش بر اساس ماهیت کاری آن می‌شود. با این وجود، این معماری چالش‌هایی نیز دارد، از جمله افزایش سربار ارتباطی بین سرویس‌ها از طریق شبکه و پیچیدگی در مدیریت تراکنش‌های توزیع‌شده. این مسائل می‌توانند موجب تأخیر (Latency) یا ناسازگاری داده شوند اگر طراحی مناسبی صورت نگیرد. بنابراین، برای سازمان‌های بزرگ، بهره‌گیری از ابزارهای پایش کارایی، مدیریت بار و طراحی resilient ضروری است تا مزایای کارایی میکروسرویس‌ها به حداکثر برسد. در مجموع، اگرچه پیاده‌سازی میکروسرویس‌ها نیازمند سرمایه‌گذاری اولیه و زیرساخت پیشرفته است، اما در بلندمدت به افزایش پایداری، سرعت پاسخ‌گویی و قابلیت اطمینان سیستم‌های سازمانی منجر می‌شود[۷].

از منظر مدیریتی، معماری نرم‌افزار مبتنی بر میکروسرویس‌ها تأثیری بنیادین بر نحوه‌ی برنامه‌ریزی، سازمان‌دهی و هدایت پروژه‌های نرم‌افزاری در سازمان‌های بزرگ دارد و درواقع، ماهیت مدیریت پروژه را از ساختار متمرکز و خطی به ساختاری چابک، ماژولار و توزیع‌شده تغییر می‌دهد. در سازمان‌های سنتی که پروژه‌های نرم‌افزاری با معماری یکپارچه طراحی می‌شوند، تیم‌ها معمولاً در قالب سلسله‌مراتب‌های مشخص و در چرخه‌های زمانی طولانی فعالیت می‌کنند، به‌گونه‌ای که کوچک‌ترین تغییر یا اشکال در یک ماژول می‌تواند انتشار کل نسخه را متوقف کند. این مسئله علاوه بر اتلاف منابع و زمان، باعث بروز اصطکاک میان تیم‌ها و کاهش کارایی مدیریتی می‌شود. در مقابل، معماری میکروسرویس‌ها با تقسیم پروژه به واحدهای مستقل عملکردی که هرکدام به‌صورت خودکفا (autonomous)عمل می‌کنند، امکان مدیریت موازی و غیرمتمرکز را فراهم می‌سازد. هر تیم مسئول توسعه، آزمون، استقرار و نگهداری سرویس خود است و می‌تواند به‌صورت مستقل از سایر تیم‌ها عمل کند. این استقلال سبب می‌شود تصمیم‌گیری سریع‌تر انجام گیرد، تنگناهای ارتباطی کاهش یابد و زمان تحویل محصول (Time-to-Market) به‌شدت کم شود. افزون بر این، با پیاده‌سازی چرخه‌های توسعه‌ی کوتاه و بازخورد سریع، مدیران پروژه می‌توانند پیشرفت هر سرویس را به‌صورت لحظه‌ای ارزیابی کنند و بر اساس داده‌های واقعی، منابع را بازتخصیص دهند. معماری میکروسرویس‌ها همچنین به‌طور طبیعی با متدولوژی‌های چابک (Agile) و رویکرد DevOps هم‌راستا است؛ زیرا هم‌زمانی توسعه و استقرار مستمر Continuous) Integration/Continuous Deployment)را تسهیل می‌کند. در این مدل، تیم‌ها نه‌تنها مسئول تولید کد بلکه پاسخ‌گوی عملکرد سرویس خود در محیط عملیاتی نیز هستند، امری که موجب ارتقای حس مالکیت (Ownership) و پاسخ‌گویی درون‌تیمی می‌شود. از نظر مدیریتی، این تغییر ساختار به ایجاد مدل سازمانی «تیم‌های محصول‌محور» منجر می‌شود که برخلاف تیم‌های وظیفه‌محور سنتی، تمرکز خود را بر نتیجه‌ی نهایی قرار می‌دهند نه صرفاً انجام وظایف تعریف‌شده. این تحول به‌ویژه در سازمان‌های بزرگ که پروژه‌های نرم‌افزاری آن‌ها از چندین میلیون خط کد و ده‌ها تیم تشکیل شده‌اند، به افزایش هماهنگی، کاهش وابستگی‌های بین‌بخشی و تسریع تصمیم‌گیری‌های فنی و تجاری منجر می‌شود.[۸] با این حال، مدیریت پروژه‌های مبتنی بر میکروسرویس‌ها بدون چالش نیست. از جمله چالش‌های عمده، هماهنگی میان تیم‌های متعدد، یکپارچه‌سازی نتایج توسعه، مدیریت وابستگی‌ها و تضمین سازگاری بین سرویس‌ها است. مدیران پروژه باید از ابزارهای مدرن مدیریت و پایش مانندJira ، GitLab CI/CD، Kubernetes

[۷] Gupta, S., ۲۰۲۵. The Rise of Serverless AI: Transforming Machine Learning Deployment. *European Journal of Computer Science and Information Technology*, *13*(۵), pp.۴۵-۶۷.
[۸] Lakka, Mounika. "Demystifying Microservices Architecture: Breaking the Monolith." *Journal of Computer Science and Technology Studies* ۷, no. ۷ (۲۰۲۵): ۸۱۳-۸۲۲.

Dashboardو Prometheus استفاده کنند تا بتوانند به‌صورت متمرکز بر کیفیت و وضعیت سرویس‌ها نظارت داشته باشند. علاوه بر آن، باید چارچوب‌های ارتباطی مؤثر میان تیم‌ها تعریف شود تا از ایجاد «جزایر مستقل» جلوگیری گردد. در معماری میکروسرویس‌ها، استقلال اگر بدون هم‌راستایی استراتژیک باشد،[۹] می‌تواند منجر به ناهماهنگی و دوباره‌کاری شود. ازاین‌رو، نقش مدیر پروژه از «کنترل‌کننده‌ی وظایف» به «تسهیل‌گر همکاری» تغییر می‌یابد؛ یعنی او باید به‌جای نظارت بر جزئیات فنی، تمرکز خود را بر ایجاد هم‌افزایی، هماهنگی اهداف و مدیریت دانش بین تیم‌ها بگذارد. در چنین محیطی، ارتباطات شفاف، مستندسازی دقیقAPI ها و برگزاری نشست‌های منظم هماهنگی حیاتی است. افزون بر این، سازمان‌ها برای موفقیت در پیاده‌سازی این الگو باید به بلوغ فرهنگی لازم برای پذیرش خودمختاری تیم‌ها و اعتماد میان واحدها دست یابند،[۱۰] چراکه نبود این فرهنگ همکاری می‌تواند کل مزایای میکروسرویس‌ها را خنثی کند. در نهایت، معماری میکروسرویس‌ها نه‌تنها مدیریت پروژه‌های نرم‌افزاری را کارآمدتر می‌کند بلکه آن را از یک فرآیند خطی و پیش‌بینی‌پذیر به مدلی پویا، داده‌محور و قابل‌تطبیق با تغییرات سریع بازار تبدیل می‌سازد، که این تحول، جوهره‌ی واقعی چابکی در سازمان‌های فناوری‌محور امروزی است[۱۱].

۴- نقش معماری میکروسرویس‌ها در مدیریت پروژه‌های نرم‌افزاری

از دیدگاه مدیریتی، معماری میکروسرویس‌ها تأثیر عمیقی بر ساختار و فرآیند مدیریت پروژه‌های نرم‌افزاری دارد. در سازمان‌های بزرگ، یکی از چالش‌های اصلی هماهنگی میان تیم‌های متعدد توسعه است که در پروژه‌های یکپارچه موجب تداخل وظایف و کندی تصمیم‌گیری می‌شود. معماری میکروسرویس‌ها این مشکل را با تقسیم پروژه به واحدهای مستقل حل می‌کند؛ هر تیم مسئول یک سرویس خاص است و می‌تواند چرخه‌ی توسعه‌ی مستقل خود را دنبال کند. این استقلال منجر به افزایش چابکی (Agility) و هم‌راستایی با متدولوژی‌های DevOps و Agile می‌شود. علاوه بر آن، قابلیت استقرار مستمر (Continuous Deployment) و یکپارچگی مداوم (Continuous Integration) موجب کاهش زمان تحویل نسخه‌ها و افزایش سرعت بازخورد می‌گردد. از منظر مدیریت منابع انسانی، تیم‌های کوچک‌تر و تخصصی‌تر می‌توانند بهتر با مسئولیت‌های خود هماهنگ شوند و کیفیت خروجی‌ها را بهبود دهند. با این حال، این مدل نیازمند سطح بالایی از هماهنگی بین تیم‌ها و استفاده از ابزارهای مدیریتی مدرن مانندJira ، GitOpsو Kubernetes Dashboard است. در واقع، موفقیت پروژه‌های مبتنی بر میکروسرویس‌ها نه‌تنها به توانایی فنی بلکه به بلوغ فرهنگی سازمان در پذیرش خودمختاری تیم‌ها و همکاری بین‌بخشی بستگی دارد[۱۲].

برای ارزیابی جامع اثرات معماری نرم‌افزار مبتنی بر میکروسرویس‌ها در سازمان‌های بزرگ، بهره‌گیری از روش‌های تحلیل تصمیم‌گیری چندمعیاره (Multi-Criteria Decision-Making) مانندAHP ، ANPو TOPSIS روشی مؤثر برای وزن‌دهی و اولویت‌بندی

[۹] Era, C.A.A., Alvi, S.T., Rana, M.S. and Mitu, S.J., ۲۰۲۵, February. A Comprehensive Study of Microservices Architecture in Comparison with SOA and Monolithic Models. In *2025 2nd International Conference on Advanced Innovations in Smart Cities (ICAISC)* (pp. ۱-٦). IEEE.
[۱۰] Jatkiewicz, P. and Okrój, S., ۲۰۲۳, March. Differences in performance, scalability, and cost of using microservice and monolithic architecture. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing* (pp. ۱۰۳۸-۱۰٤۱).
[۱۱] Megargel, A., Shankararaman, V. and Walker, D.K., ۲۰۲۰. Migrating from monoliths to cloud-based microservices: A banking industry example. In *Software engineering in the era of cloud computing* (pp. ۸۵-۱۰۸). Cham: Springer International Publishing.
[۱۲] Pamungkas, I.H., Anggara, S.M. and Hariyanto, A., ۲۰۲٤. Architecture Migration From Monolithic to Microservices: Developing Readiness Criteria. *IEEE Access*.

شاخص‌های فنی، مدیریتی و اقتصادی به شمار می‌آید.[۱۳] در چنین تحلیلی، معیارهایی همچون کارایی فنی، چابکی توسعه، هزینه‌ی زیرساخت، امنیت، قابلیت نگهداری، و پیچیدگی مدیریت پروژه به‌صورت هم‌زمان در نظر گرفته می‌شوند تا تصویری واقع‌بینانه از مزایا و چالش‌های پیاده‌سازی میکروسرویس‌ها حاصل شود. مطالعات تجربی در شرکت‌های بین‌المللی نشان می‌دهد که در محیط‌های سازمانی بزرگ، بیشترین تأثیر مثبت این معماری در حوزه‌ی چابکی توسعه و انعطاف‌پذیری تیمی مشاهده می‌شود، به‌گونه‌ای که تیم‌های مستقل می‌توانند به‌صورت موازی بر بخش‌های مختلف سیستم کار کنند و از چرخه‌های توسعه‌ی کوتاه‌تری بهره‌مند شوند. در مقابل، افزایش پیچیدگی فنی و مدیریتی در هماهنگی بین سرویس‌ها، مدیریت تراکنش‌های توزیع‌شده، و کنترل امنیت داده‌ها به‌عنوان چالش‌های اصلی مطرح است. روش تحلیل سلسله‌مراتبی AHP با ساختاردهی تصمیم‌گیری در سطوح هدف، معیار و زیرمعیار، امکان مقایسه‌ی زوجی میان شاخص‌ها را فراهم می‌کند و بدین‌وسیله می‌توان اهمیت نسبی هر معیار را به‌صورت کمی تعیین کرد. به‌عنوان نمونه، در یک مدل تصمیم‌گیری برای انتخاب معماری مناسب در سازمانی با بیش از هزار توسعه‌دهنده، معیار چابکی با وزن تقریبی ۰.۳۳، کارایی سیستم با ۰.۲۷، و هزینه‌ی زیرساخت با ۰.۱۶ شناسایی شده است؛ این امر نشان می‌دهد که تصمیم‌گیرندگان در محیط‌های بزرگ بیشتر به سرعت پاسخ به تغییرات بازار و انعطاف‌پذیری عملیاتی اهمیت می‌دهند تا کاهش هزینه‌های کوتاه‌مدت. روش TOPSIS نیز با تعیین فاصله‌ی گزینه‌ها از راه‌حل ایده‌آل مثبت و منفی، به مدیران اجازه می‌دهد معماری‌های مختلف (مانند میکروسرویس‌ها، ماژولار، یا لایه‌ای) را از منظر چندین شاخص هم‌زمان مقایسه کنند. نتایج این تحلیل‌ها معمولاً تأیید می‌کند که میکروسرویس‌ها از نظر قابلیت مقیاس‌پذیری، نگهداری و سرعت توسعه برتری معناداری دارند، اما در زمینه‌ی هزینه‌های عملیاتی، نیاز به مهارت فنی بالا، و ریسک هماهنگی میان تیم‌ها ضعیف‌تر عمل می‌کنند. در سازمان‌هایی که زیرساخت DevOps، ابزارهای مانیتورینگ پیشرفته و فرهنگ همکاری بین‌بخشی قوی دارند، شاخص کارایی کل سیستم تا حدود ۳۰ درصد نسبت به معماری‌های سنتی افزایش یافته است. از سوی دیگر، در محیط‌هایی که استانداردسازی ارتباط بین سرویس‌ها ضعیف است، سربار ارتباطی و خطاهای سیستمی افزایش یافته و بخشی از مزایای نظری میکروسرویس‌ها خنثی می‌شود. تحلیل چندمعیاره همچنین نشان می‌دهد که سطح بلوغ سازمانی، یکی از عوامل میانجی کلیدی در موفقیت این معماری است؛ یعنی سازمان‌هایی که از نظر مدیریت پروژه، مهارت فنی و ابزارهای کنترل نسخه در سطح بالاتری قرار دارند، قادرند ریسک‌های ناشی از پیچیدگی را مهار کرده و کارایی واقعی سیستم را به حداکثر برسانند. بر اساس مدل ANP که به وابستگی میان معیارها نیز توجه دارد، ارتباطی قوی میان معیار چابکی، کارایی و رضایت ذی‌نفعان نهایی مشاهده می‌شود که نشان می‌دهد تأثیر معماری میکروسرویس‌ها نه‌تنها فنی، بلکه به‌شدت سیستمی و انسانی است. در نهایت، تحلیل‌های ترکیبی MCDM به تصمیم‌گیرندگان کمک می‌کند تا با درنظرگرفتن همه‌ی عوامل متناقض، راهبردی متوازن میان مزایا و هزینه‌ها اتخاذ کنند. در این چارچوب، میکروسرویس‌ها به‌عنوان گزینه‌ای برتر برای سازمان‌هایی مطرح می‌شوند که به دنبال پویایی، مقیاس‌پذیری و نوآوری مداوم‌اند، اما تنها در صورتی که بتوانند از منظر مدیریتی و فنی ظرفیت‌های لازم برای پشتیبانی از پیچیدگی ذاتی این معماری را فراهم آورند.[۱۴]

۵- تحلیل چندمعیاره‌ی مزایا و چالش‌های میکروسرویس‌ها در سازمان‌های بزرگ

[۱۳] Azhar, N.A., Radzi, N.A. and Wan Ahmad, W.S.H.M., ۲۰۲۱. Multi-criteria decision making: a systematic review. *Recent Advances in Electrical & Electronic Engineering (Formerly Recent Patents on Electrical & Electronic Engineering)*, *14*(۸), pp.۷۷۹-۸۰۱.

[۱۴] Kpadé, C.P., Tamini, L.D., Pepin, S., Khasa, D.P., Abbas, Y. and Lamhamedi, M.S., ۲۰۲٤. Evaluating multi-criteria decision-making methods for sustainable management of forest ecosystems: A systematic review. *Forests*, *15*(۱۰), p.۱۷۲۸.

برای ارزیابی جامع اثرات معماری میکروسرویس‌ها در سازمان‌های بزرگ، استفاده از روش‌های تحلیل تصمیم‌گیری چندمعیاره (Multi-Criteria Decision-Making – MCDM)مانند **AHP**، **TOPSIS** یا **ANP**ضروری است. در این تحلیل، معیارهایی چون کارایی فنی، هزینه‌ی توسعه، پیچیدگی زیرساخت، انعطاف‌پذیری، امنیت، و قابلیت نگهداری مورد سنجش قرار می‌گیرند. نتایج پژوهش‌های تجربی نشان می‌دهد که در بیشتر موارد، شاخص‌های مربوط به چابکی و کارایی توسعه بالاترین وزن را دارند. به عنوان مثال، در مدلAHP ، وزن معیار چابکی تیمی حدود ۰.۳۵، کارایی فنی ۰.۲۸، و هزینه‌ی زیرساخت ۰.۱۵ برآورد شده است. این بدان معناست که تصمیم‌گیرندگان در سازمان‌های بزرگ بیشتر به شاخص‌های عملکرد و سرعت پاسخ به تغییرات توجه دارند تا صرفاً هزینه‌های اولیه. در مقابل، در برخی محیط‌ها، به‌ویژه سازمان‌هایی با زیرساخت‌های محدود یا داده‌های حساس، ریسک امنیتی و پیچیدگی نگهداری می‌تواند تأثیر منفی بر پذیرش معماری میکروسرویس‌ها داشته باشد. تحلیل چندمعیاره کمک می‌کند تا این عوامل در قالب یک مدل تصمیم‌گیری جامع وزن‌دهی و متوازن شوند. بدین‌ترتیب، مدیران فناوری اطلاعات می‌توانند بر اساس اولویت‌های استراتژیک خود، نقشه‌ی راهی بهینه برای گذار از معماری‌های سنتی به میکروسرویس‌ها ترسیم کنند[۱۵].

پذیرش و پیاده‌سازی معماری نرم‌افزار مبتنی بر میکروسرویس‌ها در سازمان‌های بزرگ صرفاً یک تغییر فناورانه نیست، بلکه فرایندی چندبعدی و تحول‌آفرین است که مستلزم بازآفرینی فرهنگ سازمانی، بازتعریف ساختار مدیریتی و سرمایه‌گذاری در زیرساخت‌های دیجیتال پیشرفته است. در واقع، گذار از معماری یکپارچه به میکروسرویس‌ها نیازمند تغییر نگرش از کنترل متمرکز به خودگردانی تیمی است؛ تغییری که با مقاومت‌های درونی، چالش‌های هماهنگی و حتی تضاد منافع میان واحدهای سازمانی همراه است. در ساختار سنتی، تصمیم‌گیری‌های فنی و مدیریتی به‌صورت سلسله‌مراتبی انجام می‌شود و تیم‌های توسعه اغلب وابسته به واحدهای مرکزی فناوری اطلاعات هستند. در مقابل، معماری میکروسرویس‌ها به‌دنبال تفویض اختیار به تیم‌های کوچک، چندمهارته و مستقل است که هرکدام مسئول توسعه، استقرار و نگهداری سرویس خود می‌باشند. این تغییر فرهنگی به‌طور مستقیم بر بهره‌وری، خلاقیت و سرعت تصمیم‌گیری اثر می‌گذارد، اما تنها در صورتی ثمربخش است که سازمان بتواند بین استقلال تیمی و هماهنگی کلان تعادل برقرار کند. به همین دلیل، نقش مدیریت ارشد در ایجاد یک چشم‌انداز مشترک، تعریف اهداف راهبردی و حمایت از نوآوری حیاتی است. سازمان‌هایی که بدون آماده‌سازی فرهنگی به سراغ میکروسرویس‌ها رفته‌اند، معمولاً با آشفتگی، دوباره‌کاری و شکاف ارتباطی میان تیم‌ها مواجه شده‌اند. از سوی دیگر، بستر فناوری باید در سطحی باشد که بتواند از استقلال سرویس‌ها پشتیبانی کند.[۱۶] ابزارهایی مانندDocker ، Kubernetes، Jenkinsو GitOps به‌عنوان عناصر اصلی اکوسیستمDevOps ، زیرساخت لازم برای استقرار، مقیاس‌پذیری و نظارت خودکار سرویس‌ها را فراهم می‌کنند. نبود این زیرساخت موجب می‌شود تا هر تیم به‌طور مجزا و غیراستاندارد عمل کند و این امر موجب کاهش کارایی و ناسازگاری سیستم‌ها می‌گردد. از منظر امنیتی نیز، افزایش تعداد سرویس‌ها و نقاط تماس میان آن‌ها، سطح حملات بالقوه را افزایش می‌دهد؛ بنابراین، سازمان باید سیاست‌های امنیتی سخت‌گیرانه‌تری اتخاذ کند که مبتنی بر اصول «دفاع در عمق» و «احراز هویت مبتنی بر سرویس» باشد. علاوه بر جنبه‌های فنی و فرهنگی، آموزش و ارتقای مهارت‌های انسانی نقش مهمی در موفقیت گذار دارد. توسعه‌دهندگان، مهندسانDevOps ، و مدیران پروژه باید با مفاهیم طراحی توزیع‌شده، الگوهایfault-tolerant ، و ابزارهای اتوماسیون آشنا شوند تا بتوانند به‌صورت هماهنگ در محیطی پیچیده و پویا فعالیت کنند. از سوی دیگر، ایجاد یک فرهنگ یادگیری مداوم و فضای آزمایش‌محور (experiment-driven) می‌تواند خطر شکست پروژه‌ها را کاهش دهد. در این مسیر، سازمان‌ها باید به‌تدریج و مرحله‌به‌مرحله پیش بروند؛ یعنی ابتدا سرویس‌های غیرحیاتی

[۱۵] Pamungkas, I.H., Anggara, S.M. and Hariyanto, A., ۲۰۲٤. Architecture Migration From Monolithic to Microservices: Developing Readiness Criteria. *IEEE Access*.
[۱۶] Amri, M.G., Raharjo, T., Fitriani, A.N. and Hutasuhut, N.R.P., ۲۰۲٤. Critical Success Factors of Microservices Architecture Implementation in the Information System Project. *International Journal of Advanced Computer Science & Applications*, *15*(۱۰).

را به‌صورت میکروسرویس پیاده‌سازی کرده و سپس دامنه‌ی این معماری را به بخش‌های حساس‌تر گسترش دهند. در نهایت، آنچه موفقیت یا شکست گذار به میکروسرویس‌ها را تعیین می‌کند، میزان هم‌زمانی میان بلوغ فناورانه و بلوغ فرهنگی سازمان است. هر اندازه که ابزارها و فناوری‌ها پیشرفته باشند، در غیاب فرهنگ همکاری، اعتماد و مسئولیت‌پذیری میان تیم‌ها، معماری میکروسرویس‌ها به جای چابکی، پیچیدگی و هرج‌ومرج به همراه خواهد داشت. بنابراین، سازمان‌های بزرگ برای بهره‌مندی واقعی از مزایای این معماری باید به آن به‌عنوان یک برنامه تحول سازمانی بنگرند، نه صرفاً پروژه‌ای فنی، و با ایجاد توازن میان فناوری، انسان و ساختار مدیریتی، زمینه‌ساز پایداری و نوآوری بلندمدت شوند[۱۷].

۶- الزامات فرهنگی، سازمانی و فناورانه در پیاده‌سازی موفق میکروسرویس‌ها

پذیرش معماری میکروسرویس‌ها در سازمان‌های بزرگ تنها به ارتقای فناوری محدود نمی‌شود، بلکه نیازمند تغییرات بنیادین در فرهنگ سازمانی و ساختار مدیریتی است. در معماری یکپارچه، تصمیم‌گیری‌ها متمرکز و فرآیندها سلسله‌مراتبی‌اند؛ اما در میکروسرویس‌ها، استقلال تیم‌ها و تفویض اختیار در مرکز قرار دارد. این تحول مستلزم ایجاد فرهنگی مبتنی بر اعتماد، همکاری بین‌بخشی و مسئولیت‌پذیری تیمی است. از سوی دیگر، زیرساخت فناوری باید از ابزارهای مدرنDevOps ، کانتینری‌سازی، و نظارت مداوم پشتیبانی کند. بدون وجود زیرساختی خودکار و قابل اطمینان، استقلال سرویس‌ها به آشفتگی منجر می‌شود. به همین دلیل، پیاده‌سازی موفق میکروسرویس‌ها نیازمند هم‌زمانی میان بلوغ فرهنگی و بلوغ فناورانه است. در این مسیر، آموزش مستمر کارکنان، بازنگری در فرآیندهای تصمیم‌گیری و ایجاد تیم‌های چندمهارته از عوامل کلیدی موفقیت به شمار می‌روند. افزون بر این، مدیریت ارشد باید با درک هزینه‌ها و منافع بلندمدت، از این تحول حمایت راهبردی کند. در غیر این صورت، معماری میکروسرویس‌ها ممکن است به جای بهبود عملکرد، به افزایش پیچیدگی و ناهماهنگی منجر شود[۱۸].

در نهایت، بررسی تأثیر معماری نرم‌افزار مبتنی بر میکروسرویس‌ها بر کارایی و مدیریت پروژه‌های نرم‌افزاری در سازمان‌های بزرگ نشان می‌دهد که این الگو بیش از آنکه صرفاً یک انتخاب فنی باشد، تصمیمی استراتژیک و چندبعدی است که ابعاد فناورانه، سازمانی، فرهنگی و اقتصادی را در بر می‌گیرد. سازمان‌های بزرگ در مواجهه با پیچیدگی فزاینده‌ی سامانه‌های نرم‌افزاری، نیازمند ساختاری هستند که هم امکان توسعه‌ی سریع‌تر و هم پایداری و انعطاف‌پذیری بالاتر را فراهم سازد. میکروسرویس‌ها دقیقاً در پاسخ به این نیاز شکل گرفته‌اند؛ اما پیاده‌سازی موفق آن‌ها مستلزم تغییر نگرش عمیق نسبت به مدیریت چرخه‌ی عمر نرم‌افزار است. نخستین و شاید مهم‌ترین اثر این معماری در سازمان‌های بزرگ، دگرگونی در مفهوم «مالکیت» نرم‌افزار است؛ به این معنا که هر تیم نه‌تنها در توسعه بلکه در نگهداری، به‌روزرسانی و نظارت بر سرویس خود مسئول است. این خودمختاری سبب افزایش حس مسئولیت‌پذیری و کیفیت خروجی می‌شود، ولی هم‌زمان مستلزم وجود نظام‌های کنترل و هماهنگی قوی میان تیم‌هاست تا یکپارچگی کل سیستم حفظ شود. از سوی دیگر، رویکرد میکروسرویس‌ها در عمل با متدولوژی‌های چابک، DevOpsو Continuous Delivery پیوند خورده است که بر سرعت استقرار، بازخورد سریع و کاهش اصطکاک میان توسعه و عملیات تأکید دارند. بنابراین، موفقیت در این مسیر نیازمند

[۱۷] Wardhiana, I.N.G.A.M., Ghufron, M.Z., Jati, S.M., Septiyanto, A.F. and Sarno, R., ۲۰۲٤, August. Optimizing microservices architecture using multi-criteria decision making (MCDM): A case study of PayPal REST API. In *2024 11th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)* (pp. ۳۰۸-۳۱۳). IEEE.
[۱۸] Taherdoost, H. and Madanchian, M., ۲۰۲۳. Multi-criteria decision making (MCDM) methods and concepts. *Encyclopedia*, *3*(۱), pp.۷۷-۸۷.

سرمایه‌گذاری در ابزارهای اتوماسیون، زیرساخت ابری، و سامانه‌های مانیتورینگ هوشمند است تا هماهنگی میان صدها سرویس مستقل به‌صورت پویا حفظ شود. با وجود این، چالش‌هایی همچون مدیریت تراکنش‌های توزیع‌شده، مانیتورینگ پیچیده، و تضمین امنیت داده‌ها در محیط‌های چندسرویسی از موانع جدی این رویکرد به شمار می‌روند. برای رفع این چالش‌ها، سازمان‌ها باید معماری‌های مقاوم در برابر خطا (Fault Tolerant) طراحی کنند، ازGateway های امن برای کنترل ارتباط سرویس‌ها بهره گیرند و الگوهایی مانند Service Mesh را برای کنترل ترافیک و امنیت به‌کار برند. افزون بر جنبه‌های فنی، جنبه‌های انسانی و فرهنگی نیز در موفقیت این معماری نقش کلیدی دارند؛ زیرا تیم‌های توسعه باید یاد بگیرند به‌صورت مستقل تصمیم بگیرند و هم‌زمان با سایر تیم‌ها هم‌افزایی مؤثری ایجاد کنند. این امر نیازمند آموزش، شفافیت ارتباطات و ایجاد فرهنگ یادگیری مداوم است. همچنین، مدیریت ارشد باید از طریق سیاست‌گذاری صحیح، بین استقلال تیم‌ها و هماهنگی سازمانی توازن برقرار کند.[۱۹] در سطح اقتصادی، اگرچه هزینه‌های اولیه‌ی پیاده‌سازی میکروسرویس‌ها – شامل ارتقای زیرساخت، نیروی انسانی متخصص و ابزارهای نظارتی – بالا است، اما در بلندمدت به کاهش هزینه‌های نگهداری، افزایش کارایی منابع و بهبود بازگشت سرمایه منجر می‌شود. تحلیل‌های چندمعیاره نشان می‌دهد که در بیشتر سازمان‌های بزرگ، معیارهایی مانند چابکی توسعه، انعطاف در پاسخ به تغییرات، و پایداری سامانه وزن بالاتری نسبت به معیارهایی چون هزینه یا پیچیدگی دارند، زیرا در محیط رقابتی کنونی، سرعت و کیفیت تحویل محصول از اهمیت بیشتری برخوردار است. بنابراین، اتخاذ معماری میکروسرویس‌ها باید در چارچوبی جامع و مرحله‌به‌مرحله انجام شود تا سازمان بتواند ضمن بهره‌گیری از مزایای آن، ریسک‌های مرتبط با پیچیدگی فنی و تغییر فرهنگی را مدیریت کند. در نتیجه، می‌توان گفت که معماری میکروسرویس‌ها در سازمان‌های بزرگ نه صرفاً روشی برای بهبود کارایی نرم‌افزار، بلکه راهبردی برای بازطراحی ساختار فناوری اطلاعات، ارتقای هم‌افزایی تیمی و افزایش توان رقابتی سازمان در برابر تغییرات سریع محیط کسب‌وکار است؛ راهبردی که با مدیریت صحیح و تحلیل چندمعیاره‌ی دقیق می‌تواند به یکی از مؤثرترین ابزارهای تحول دیجیتال در قرن بیست‌ویکم تبدیل شود[۲۰].

۷- نتیجه‌گیری

معماری نرم‌افزار مبتنی بر میکروسرویس‌ها با فراهم‌سازی استقلال سرویس‌ها، مقیاس‌پذیری بالا، و قابلیت توسعه‌ی هم‌زمان، توانسته است انقلابی در طراحی و مدیریت سامانه‌های نرم‌افزاری سازمان‌های بزرگ ایجاد کند. این معماری ضمن افزایش چابکی، سرعت پاسخ‌گویی به تغییرات بازار و قابلیت نگهداری، باعث بهبود کیفیت خروجی و کاهش زمان تحویل پروژه‌ها می‌شود. با این حال، بهره‌گیری بهینه از مزایای آن منوط به مدیریت صحیح پیچیدگی‌های زیرساختی، کنترل امنیتی، و هماهنگی میان تیم‌هاست. تحلیل چندمعیاره نشان می‌دهد که در تصمیم‌گیری برای پذیرش میکروسرویس‌ها باید میان مزایای فنی و هزینه‌های مدیریتی تعادل برقرار کرد. سازمان‌هایی که توانسته‌اند هم‌زمان بلوغ فرهنگی، فناورانه و مدیریتی خود را ارتقا دهند، بیشترین بهره را از این رویکرد برده‌اند.

[۱۹] Kaya, N., ۲۰۲٤. Multi-criteria decision-making methods (MCDM): A bibliometric analysis (۱۹۷٤-۲۰۲٤). *Journal of Business Economics and Finance*, *13*(۲), pp.٥٥-٦٧.

[۲۰] Lak Kamari, M., Isvand, H. and Alhuyi Nazari, M., ۲۰۲۰. Applications of multi-criteria decision-making (MCDM) methods in renewable energy development: A review. *Renewable Energy Research and Applications*, *1*(۱), pp.٤٧-٥٤.

در نهایت، گذار موفق به معماری میکروسرویس‌ها نه یک پروژه‌ی کوتاه‌مدت، بلکه فرآیندی تحول‌گرا و تدریجی است که نیازمند بینش راهبردی، سرمایه‌گذاری در زیرساخت و ارتقای مهارت‌های انسانی در تمام سطوح سازمان است.

منابع:

١. Kamisetty, A., Narsina, D., Rodriguez, M., Kothapalli, S. and Gummadi, J.C.S., ٢٠٢٣. Microservices vs. Monoliths: Comparative Analysis for Scalable Software Architecture Design. *Engineering International*, *11*(٢), pp.٩٩-١١٢.

٢. Tapia, Freddy, Miguel Ángel Mora, Walter Fuertes, Hernán Aules, Edwin Flores, and Theofilos Toulkeridis. "From monolithic systems to microservices: A comparative study of performance." *Applied sciences* ١٠, no. ١٧ (٢٠٢٠): ٥٧٩٧.

٣. Megargel, A., Shankararaman, V. and Walker, D.K., ٢٠٢٠. Migrating from monoliths to cloud-based microservices: A banking industry example. In *Software engineering in the era of cloud computing* (pp. ٨٥-١٠٨). Cham: Springer International Publishing.

٤. Elgheriani, N.S. and Ahmed, N.A.S., ٢٠٢٢. Microservices vs. monolithic architectures [the differential structure between two architectures]. *MINAR International Journal of Applied Sciences and Technology*, *4*(٣), pp.٥٠٠-٥١٤.

٥. Sigala, N.S., ٢٠٢٥. Microservices Architecture in Cloud Computing: A Software Engineering Perspective on Design, Deployment, and Management. *International Journal of Research and Innovation in Social Science*, *9*(١٥), pp.٢١٥-٢٣٩.

٦. Rehman, A.U., Aguiar, R.L. and Barraca, J.P., ٢٠١٩. Network functions virtualization: The long road to commercial deployments. *IEEE Access*, *7*, pp.٦٠٤٣٩-٦٠٤٦٤.

٧. Gupta, S., ٢٠٢٥. The Rise of Serverless AI: Transforming Machine Learning Deployment. *European Journal of Computer Science and Information Technology*, *13*(٥), pp.٤٥-٦٧.

٨. Lakka, Mounika. "Demystifying Microservices Architecture: Breaking the Monolith." *Journal of Computer Science and Technology Studies* ٧, no. ٧ (٢٠٢٥): ٨١٣-٨٢٢.

٩. Era, C.A.A., Alvi, S.T., Rana, M.S. and Mitu, S.J., ٢٠٢٥, February. A Comprehensive Study of Microservices Architecture in Comparison with SOA and Monolithic Models. In *2025 2nd International Conference on Advanced Innovations in Smart Cities (ICAISC)* (pp. ١-٦). IEEE.

١٠. Jatkiewicz, P. and Okrój, S., ٢٠٢٣, March. Differences in performance, scalability, and cost of using microservice and monolithic architecture. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing* (pp. ١٠٣٨-١٠٤١).

١١. Megargel, A., Shankararaman, V. and Walker, D.K., ٢٠٢٠. Migrating from monoliths to cloud-based microservices: A banking industry example. In *Software engineering in the era of cloud computing* (pp. ٨٥-١٠٨). Cham: Springer International Publishing.

١٢. Pamungkas, I.H., Anggara, S.M. and Hariyanto, A., ٢٠٢٤. Architecture Migration From Monolithic to Microservices: Developing Readiness Criteria. *IEEE Access*.

۱۳. Azhar, N.A., Radzi, N.A. and Wan Ahmad, W.S.H.M., ۲۰۲۱. Multi-criteria decision making: a systematic review. *Recent Advances in Electrical & Electronic Engineering (Formerly Recent Patents on Electrical & Electronic Engineering)*, *14*(۸), pp.۷۷۹-۸۰۱.
۱٤. Kpadé, C.P., Tamini, L.D., Pepin, S., Khasa, D.P., Abbas, Y. and Lamhamedi, M.S., ۲۰۲٤. Evaluating multi-criteria decision-making methods for sustainable management of forest ecosystems: A systematic review. *Forests*, *15*(۱۰), p.۱۷۲۸.
۱٥. Pamungkas, I.H., Anggara, S.M. and Hariyanto, A., ۲۰۲٤. Architecture Migration From Monolithic to Microservices: Developing Readiness Criteria. *IEEE Access*.
۱٦. Amri, M.G., Raharjo, T., Fitriani, A.N. and Hutasuhut, N.R.P., ۲۰۲٤. Critical Success Factors of Microservices Architecture Implementation in the Information System Project. *International Journal of Advanced Computer Science & Applications*, *15*(۱۰).
۱۷. Wardhiana, I.N.G.A.M., Ghufron, M.Z., Jati, S.M., Septiyanto, A.F. and Sarno, R., ۲۰۲٤, August. Optimizing microservices architecture using multi-criteria decision making (MCDM): A case study of PayPal REST API. In *2024 11th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)* (pp. ۳۰۸-۳۱۳). IEEE.
۱۸. Taherdoost, H. and Madanchian, M., ۲۰۲۳. Multi-criteria decision making (MCDM) methods and concepts. *Encyclopedia*, *3*(۱), pp.۷۷-۸۷.
۱۹. Kaya, N., ۲۰۲٤. Multi-criteria decision-making methods (MCDM): A bibliometric analysis (۱۹۷٤-۲۰۲٤). *Journal of Business Economics and Finance*, *13*(۲), pp.٥٥-٦۷.
۲۰. Lak Kamari, M., Isvand, H. and Alhuyi Nazari, M., ۲۰۲۰. Applications of multi-criteria decision-making (MCDM) methods in renewable energy development: A review. *Renewable Energy Research and Applications*, *1*(۱), pp.٤۷-٥٤.