ABDIKATA: Jurnal Abdi Loka Wisata
Program Diploma Kepariwisataan Universitas Merdeka Malang
Vol. 1 (01) November 2025; 26-34
p-ISSN: XXXX-XXXX | e-ISSN: XXXX-XXXX
https://jurnal.unmer.ac.id/index.php/abdikata

Cloud-native microservices architecture to support the scalability of the traditional village tourism platform in Bali

Harianto
ITB Dewantara
Acropolis, RUKO Blk. LC, Jl. Bojong Depok Baru III Blok LC 19, Karadenan, Kec. Cibinong, Kabupaten Bogor, Jawa Barat 16913, Indonesia

ARTICLE INFO:

Received: 30-09-2025
Review: 15-10-2025
Accepted: 30-11-2025

Corresponding author:

Harianto
masharry.net@gmail.com

Keywords:

cloud-native; microservices architecture; scalability; tourism platform; traditional village

DOI:

ABSTRACT

Tourism based on local communities in Bali's traditional villages requires technological support to optimally develop. Digital tourism platforms for these villages in the era of digital transformation must handle surges in users and information requests, especially during holidays or special events. A microservices architecture with a cloud-native approach is proposed as a solution to improve platform scalability and reliability. This study is a literature review that analyzes the microservices and cloud-native architecture concepts in the context of village tourism platforms. Microservices is an architectural style that breaks down an application into small services that can be developed, deployed, and scaled independently. This approach, supported by cloud computing, allows tourism platforms to adjust service capacity in real-time according to load, avoiding system failures during traffic spikes. A conceptual case study in Bali's traditional villages shows that a cloud-native microservices architecture can enhance feature development flexibility, ease of integrating external services, and maintain system availability even if one component encounters issues. The results of this study are expected to inform the design of modern, scalable, and resilient tourism platforms for traditional villages, thereby supporting empowerment and self-reliance in community-based tourism management.

How to cite: Harianto, H. (2025). A Cloud-Native Microservices Architecture to Support Scalability of Traditional Village Tourism Platform in Bali: Designing a Resilient and Scalable Digital Ecosystem for Community-Based Tourism in Bali through Cloud-Native Microservices. *Abdikata: Jurnal Abdi Loka Wisata*, *1*(01). Retrieved from https://jurnal.unmer.ac.id/index.php/abdikata/article/view/16237

Introduction

The tourism industry has undergone rapid digital transformation in the last decade. The concept of smart tourism has developed, where information technology is integrated to improve tourist experiences and destination management. Bali, as an icon of Indonesian tourism, has also followed this trend, including through the development of tourist villages or traditional villages that have become cultural tourist destinations. Traditional villages in Bali play an important role in cultural preservation while attracting local and foreign tourists. However, the challenge that arises is how to effectively manage and promote the tourism potential of these traditional villages

in the digital era. Previous studies have shown that the use of digital platforms can increase promotion and tourist visits to tourist villages in Bali. Nirmala and Lavianto (2019) emphasize the importance of digital enablers in the marketing of community-based tourist villages in Bali to support an increase in tourist visits. Similarly, Warmayana (2018) states that the use of digital marketing in tourism promotion is a necessity in the Industry 4.0 era to reach a wider range of tourists.

Although some tourism villages already have websites or social media accounts, many are still managed conventionally and separately, so they are not yet optimal in responding to the needs of modern tourists. Existing tourism platforms are often simple monolithic applications that display basic information and contacts. A monolithic platform means that all functions (e.g., destination information, reservations, payments) are integrated into one application. This monolithic architecture has limitations in terms of scalability and maintenance. When the number of users increases rapidly, monolithic applications can generally only be scaled by running instances of the entire application in parallel, which is inefficient if the surge in load only occurs in certain functions. As a result, during holidays or special events, monolithic platforms risk slowing down or even failing to handle the surge in requests. In addition, changes or additions to features in a monolithic architecture can be difficult and risk disrupting the entire system *(tight-coupling)*. The monolithic approach is also prone to total downtime if an error occurs in one module, because all components run in a single, interdependent unit.

To overcome these limitations, a more modular, flexible, and scalable software architecture is needed. *Microservices* architecture emerged as the answer to these challenges (Lewis & Fowler, 2014; Dragoni et al., 2017). *Microservices* is an architectural style that breaks down applications into small, independent services that communicate through lightweight mechanisms (e.g., REST API). Each service focuses on a specific business capability and can be developed, tested, deployed, and scaled independently (Oyeniran et al., 2024; Newman, 2015). This approach increases system modularity, making maintenance and the addition of new features easier (Atmojo et al., 2022; Newman, 2015). In the context of a tourism platform, microservices architecture allows features such as tourist attraction information, homestay booking, payment, and tourist reviews to be placed on separate services. Thus, changes to the booking module, for example, will not affect the destination information module, as long as the services communicate through a clear interface (API). This significantly reduces the risk *of* overall *downtime* and increases the agility of the system to adapt to new needs (Dragoni et al., 2017; Atmojo et al., 2022).

The adoption of *microservices* architecture is increasingly enabled and reinforced by advances in cloud computing. The concept of cloud-native refers to an approach to building and operating applications that fully leverage the *cloud* environment, including the use of containers, managed cloud services, and automated orchestration. Cloud platforms provide the ease of horizontal scaling, which is the ability to quickly add service instances when the load increases and reduce them when the load decreases, automatically. For large-scale travel companies, migrating from *legacy* systems to cloud-based *microservices* architecture has been proven to improve their performance and operational flexibility. Barua et al. (2025) found that cloud *microservices* frameworks on online travel platforms were able to handle traffic spikes smoothly and reduce downtime compared to traditional monolithic architectures. In the travel and hospitality industry, the implementation of cloud computing provides benefits such as cost savings, reliable scalability, and flexible access from anywhere. With *cloud* infrastructure, tourism platforms can serve global users 24/7 without relying on local servers with limited capacity.

Based on the above description, this paper aims to examine cloud-native *microservices* architecture as a solution to support the scalability of traditional village tourism platforms in Bali. The study was conducted conceptually through a literature review of the latest primary references related to *microservices*, cloud computing, and their application in the tourism industry. It is hoped that the results of this study can contribute knowledge and architectural recommendations for developers or stakeholders who wish to build a modern, scalable, and reliable community tourism *platform*.

Method

This research is a descriptive-qualitative *literature review*. The methods used include the collection, evaluation, and synthesis of information from various primary sources related to *microservices* architecture, cloud-native concepts, and their application in tourism systems. The research stages began with identifying the problem, namely the need for scalability of the traditional village tourism platform, followed by a search of scientific literature through journal databases, conference proceedings, and other reliable sources. The prioritized literature criteria were s published in the last 10 years (2015–2025) that were relevant to the topic, in accordance with the minimum requirement of 15 primary references (80% of the latest publications). Some of the keywords used in the search included: *microservices architecture, cloud-native, tourism platform, scalability,* and tourism village. The collected literature was then selected based on content suitability and quality (preference was given to indexed journal articles and IEEE/ACM proceedings). Next, content analysis was conducted to understand key concepts: the definition *of microservices*, the benefits and challenges *of microservices*, cloud-native principles, and case studies of the application of this architecture. Information from various sources was compared and integrated to formulate a conceptual architecture framework appropriate to the context of tourist villages in Bali.

In the discussion stage, this study combines theoretical findings with practical case study scenarios. The case study is conceptual, using the scenario of a traditional village in Bali that wants to develop an online tourism platform. The analysis was conducted deductively by applying the concept of *cloud-native microservices* architecture to the scenario, accompanied by architectural diagrams and technical flowcharts.

Results and Discussion

Microservices Architecture for Scalability

Microservices are defined as an application architecture composed of a set of small, cohesive services that run in separate processes and communicate via messages or lightweight protocols (e.g., HTTP/REST). Each service generally implements a single business function in its entirety (single responsibility) so that changes to one service do not affect the entire system (isolated failure). This differs from monolithic architecture, where all components are interdependent within a single application. Dragoni et al. (2017) explain that microservices architecture emerged to address the weaknesses of monoliths, such as difficulty in maintenance, dependency hell, and the scalability limitations of certain modules. With microservices, developers can focus on implementing specific services and choose the most suitable technology for each service (heterogeneity), for example, one service uses a SQL database while another uses NoSQL, without interfering with each other (polyglot persistence). The main advantages of microservices include scalability and resilience. Scalability is achieved because each service can be scaled out independently according to workload requirements. For example, if the reservation

module experiences a surge in traffic, only the reservation service needs to have its instances increased, without having to duplicate the entire application as in a monolith (resource efficiency). In terms of resilience, a failure in one service (e.g., the payment service) does not necessarily cause the entire system to crash, because other services can continue to operate and the failed service can be restarted in isolation (fault isolation). Atmojo et al. (2022) proved this in the implementation of a microservices-based village information system: the performance of village apparatus services improved, the addition of business processes became easier, and system security was better maintained than in the previous monolithic version.

However, microservices architecture also brings its own challenges. Compared to monoliths, microservices add complexity to management because there are many services that must be orchestrated and integrated (development sprawl). Developing a system with dozens of microservices requires the implementation of mature DevOps practices for deployment continuity and monitoring. Benavente et al. (2022), through a comparative study, concluded that for small-scale systems with simple logic, monolithic architecture can have slightly better performance and be developed more quickly. However, for large and complex systems, microservices provide significant advantages in terms of code simplification, long-term scalability, and ease of maintenance, even though they require a greater initial development time investment. Tapia et al. (2020) in a performance test study also found that microservices architecture has slightly more overhead on throughput than monoliths when the load is low, but is able to handle high loads more efficiently and stably without the drastic performance degradation that occurs in monolithic architecture. In other words, microservices excel in elasticity—the ability of a system to dynamically adjust capacity—which is a key factor for tourism platforms with fluctuating user demand (e.g., surging during holidays and declining on regular days).

Cloud-Native Approach

In order to fully utilize the potential of microservices, implementation is usually carried out in conjunction with cloud-native principles. Cloud-native architecture means that applications are designed to run in a cloud computing environment by utilizing cloud features, such as containerization, orchestration, auto-scaling, and managed services. Container technology (e.g., Docker) allows each service to be packaged with its dependencies in a single isolated unit that can be easily moved between server environments. Pahl (2015) explains that containerization on Platform as a Service (PaaS) platforms facilitates the deployment of portable and efficient applications in the cloud. Each microservice can be run in its own container, thereby avoiding conflicts between environments and optimizing resource usage. Above the containers, there are orchestrators such as Kubernetes that manage the scheduling of these containers on server clusters, perform health checks, and automate the scaling process. With this orchestration, when the load on a service increases, the system can automatically add replicas of the service container (horizontal scaling) and reduce them again when the load returns to normal (auto-scale on demand). Oyeniran et al. (2024) emphasize that the integration of microservices with cloud platforms results in high elasticity, where the system can automatically adjust capacity according to workload, while increasing resilience and fault tolerance through process isolation.

Cloud-native microservices architecture also supports the implementation of CI/CD (Continuous Integration/Continuous Deployment) practices, which accelerate the release of new features without disrupting other services. This is important so that the village tourism platform can continue to grow and innovate in line with market needs. In addition, the cloud environment

provides many supporting services (such as managed databases, authentication services, monitoring) that can be directly integrated into applications, so managers do not need to build everything from scratch. Integration with third-party services such as payment gateways, digital maps (GIS), or external reservation systems is also easier through APIs. According to Pencarelli (2020), digital transformation in the tourism industry is characterized by the interconnection of various services through APIs and centralized data analysis to enhance the customer experience. With a distributed architecture, tourism platforms can collect data from each service (e.g., destination search data, booking data, tourist feedback) into a centralized data lake in the cloud, then apply analytics for insights such as tourist visit patterns or promotion effectiveness (which can improve tourism village marketing strategies as mentioned by Warmayana (2018)).

Security is also a vital aspect of microservices architecture. Distributed systems present a broader attack surface due to the many open services and integration points. Sun, Nanda, and Jaeger (2015) introduced the concept of Security-as-a-Service for microservices-based cloud applications, emphasizing the importance of authentication, authorization, and encryption mechanisms for communication between services from the system design stage. In implementation, each service should be equipped with a security token (e.g., JWT) and managed by an API Gateway that functions as a unified gateway. The API Gateway is a microservices design pattern component that provides a single access point for clients to various services behind it, handling tasks such as authentication, routing requests to related services, rate limiting, and response aggregation. With a gateway, the internal architecture of microservices is protected from direct access, adding a layer of security and simplifying API consumption for users.

Design of a Microservices-Based Tourism Platform for Traditional Villages

Based on the above concept, the following is a proposed cloud-native microservices architecture design to support a tourism platform for traditional villages in Bali. Figure 1. Cloud-Native Microservices Architecture illustrates the main components and interaction flows in the proposed platform.

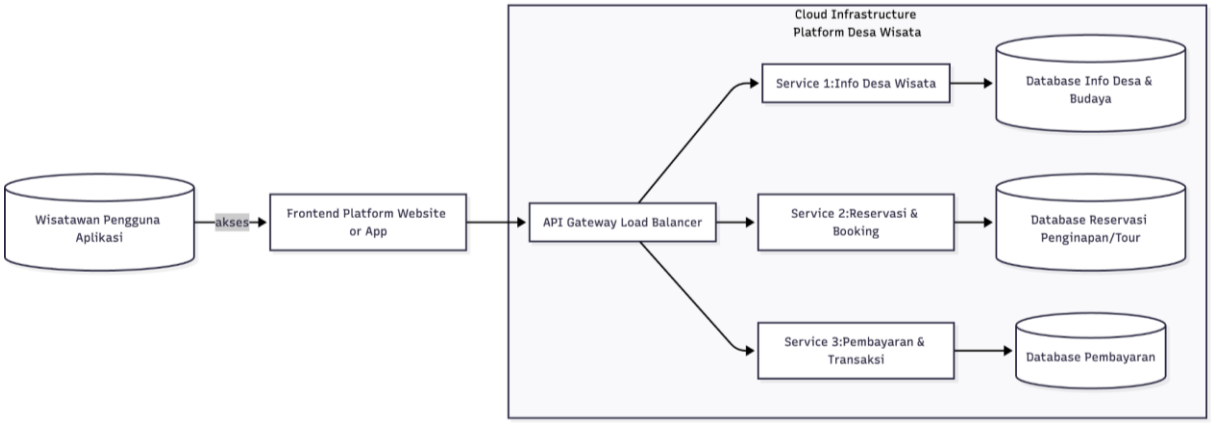

Figure 1. Illustration of *cloud-native microservices* architecture for the tourism platform for traditional villages.

In this design, the front-end of the application (website or mobile app) is not directly connected to the services in the back-end, but rather through an API Gateway. This gateway will forward requests to the relevant microservice services according to the accessed endpoint.

Several core services identified for the traditional village tourism platform include: Tourism Information Service, Reservation/Booking Service, Payment Service, Review & Rating Service, and Local MSME Management Service.

The Tourism Information Service provides details about village tourist attractions (e.g., temples, cultural attractions, traditional event calendars), traditional village profiles, tour packages, and guide information for tourists. This service is connected to a tourism content database that can be updated by village administrators.

The Reservation Service handles tourism service bookings, such as homestay or guest house reservations in the village, local guide bookings, or traditional art performance tickets. This service communicates with the payment service for transaction processing and can be connected to a calendar system to manage availability. A study by Martinez-Garcia et al. (2018) shows that the integration of *marketplace* and trip planning features in a single platform can be achieved with a *microservices* architecture, where each feature runs as a separate service but is coordinated through an API. This is relevant for tourist villages that may want to provide custom tour packages for visitors (trip planning).

Payment Services manage electronic payment processes (e.g., credit cards, e-wallets). This service can utilize third-party APIs (midtrans, etc.) so that the payment module on the platform functions as a secure intermediary. With microservices, the payment module can be isolated for security and compliance (e.g., PCI DSS) without affecting other services.

The Reviews & Ratings Service allows tourists to provide feedback, reviews, and ratings on their tourism experiences in traditional villages. This review data is important for improving service quality and transparency. This service can be integrated with text analytics modules (e.g., sentiment analysis) in the future to analyze tourist perceptions, as in the concept of the "StreetLines" platform, which processes tourist feedback from various sources.

The Local MSME Management Service supports online sales of local products (crafts, souvenirs). This is in line with several community service programs that help tourist villages sell MSME products through e-commerce. The MSME service allows tourists to order products before or after their visit, supporting the local economy.

Each of the above services has its own database or data storage as needed (database per service principle). For example, the reservation service has a reservation transaction database, separate from the review database. This separation improves performance and scalability, as each database can be optimized for different access patterns and can be scaled separately.

Communication between services can use synchronous patterns (REST API) for direct requests, or asynchronous patterns (message queue) for processes that do not require real-time responses. For example, when a payment is successful, the payment service sends a message to the reservation service to confirm the booking without waiting for an immediate response (asynchronous event). The use of message brokers (such as RabbitMQ or Kafka) is common in microservices architecture to loosely integrate services and increase system throughput.

In Figure 1 *(placeholder)*, it can be seen that all services are run in separate containers managed by an orchestrator in the cloud. Cloud environments that can be used include Kubernetes services from cloud providers (GCP, AWS EKS, etc.) or private clouds. Each service container replicates instances as needed: for example, the Information Service may have 2 container instances under normal load, but when there is a large event and many tourists access information, the auto-scaler increases it to, say, 5 instances. Meanwhile, the Payment Service may normally have only 1 instance and increase to 2 instances during peak transactions at night. This auto-scaling occurs automatically based on metrics (CPU usage, number of requests) that are

continuously monitored (metrics are obtained from monitoring tools such as Prometheus and Grafana). This approach ensures efficient resource utilization, so that cloud operating costs can be optimized—given that tourism villages may have limited budgets, the elasticity feature of the cloud helps them pay only for the resources they use *(cost efficiency).*

The advantages of the above architectural design for the traditional village tourism platform can be summarized as follows:

1. Guaranteed scalability: Able to serve significant traffic increases without a decline in performance, because critical services can be scaled horizontally quickly (Barua et al., 2025).
2. High reliability: A disruption in one service (e.g., payment gateway error) does not bring down the entire system, and a circuit breaker mechanism can be implemented to temporarily divert or disconnect the problematic service until it recovers.
3. Flexibility and ease of development: Development teams can work in parallel on different services (e.g., team A develops the MSME module, team B develops the reservation module) with different technology stacks as needed. New features can be implemented more quickly by simply adding new services or updating related services without touching the entire application. For traditional villages, this means that the platform can be continuously improved in stages, for example, by adding an AR *(Augmented Reality)* guide feature at a later date as a new service, without the need to redeploy the entire system.
4. External service integration: Through an API approach, the platform can be easily connected to third-party services such as government tourism portals (e.g., Jadesta), travel marketplaces, or international review platforms. This supports the networking of tourist villages to a broader tourism ecosystem.
5. Better user experience: With a robust and responsive system, tourists get fast and reliable information and booking services. Minimal downtime and fast performance increase user satisfaction, thereby maintaining a positive image of the tourist village.

From an implementation perspective, resources and technical expertise are required to realize this architecture. Traditional village managers may need assistance from IT experts or collaboration with technology providers. Various open-source platforms can be utilized, such as using headless WordPress for content exposed via API, combined with other custom services, or utilizing BaaS (Backend as a Service) to accelerate development. The involvement of stakeholders (tourism agencies, academics, local IT communities) in the planning and development of this platform will determine its success as a sustainable solution. Although microservices offer many benefits, this approach must be balanced with the capacity of the development team and the available infrastructure. If the number of services is still small and the management team is limited, a hybrid approach can be chosen: for example, starting with separate modules for critical components, while gradually breaking down other monoliths as the scale increases (evolutionary principle).

Overall, this conceptual study indicates that cloud-native microservices architecture has great potential for adoption in indigenous village tourism platforms. Future trends indicate that more and more community tourism destinations are utilizing digital technology and data to improve competitiveness and sustainability. Traditional villages in Bali, with their rich culture, can seize this opportunity to offer a more interactive, personalized, and reliable tourism experience through the support of a strong digital platform.

Conclusion

Digital transformation in the management of indigenous village tourism is a necessity in the modern era. Based on this literature study, cloud-native microservices architecture has proven to be a superior solution to support the scalability and availability of community tourism platforms. Microservices architecture breaks down the system into independent services that can be scaled and updated independently, enabling it to handle surges in visitors during busy periods without sacrificing system performance or stability. Supported by the implementation of cloud computing, the platform can perform auto-scaling on demand, optimize resource usage, and minimize downtime.

In the context of traditional villages in Bali, the application of this architecture will enable the integration of various features (cultural information, accommodation reservations, payments, MSME promotions, etc.) into a flexible and reliable digital ecosystem. Tourists can obtain faster, more integrated, and personalized services, while village managers can monitor and develop the platform continuously as needed. This conceptual study is in line with global tourism industry trends that increasingly rely on distributed digital technology to enhance the tourist experience and operational efficiency (Pencarelli, 2020; Gretzel et al., 2015).

Of course, the implementation of cloud-native microservices requires commitment in terms of human resources and infrastructure. Therefore, it is recommended that tourism village managers establish partnerships with technology providers or universities for technical assistance. Digital skills training for local human resources is also important so that platform management can be carried out independently in the future. With careful planning and stakeholder support, a tourism platform based on microservices architecture can become an enabler for the empowerment of traditional villages, expand marketing reach, increase tourist satisfaction, and ultimately support the economic and cultural sustainability of local communities. The concepts and recommendations from this research are expected to be tested and further refined through real implementation projects in tourism villages, thereby transferring knowledge from theory to practice that benefits the community.

References

Atmojo, S., Utami, R., Dewi, S., & Widhiyanta, N. (2022). Implementation of a Village Information System Based on Microservices Architecture. SMATIKA Journal, 12(1), 55–66.

Barua, B., Mozumder, M. J. U., Kaiser, M. S., & Barua, I. (2025, February). Building Scalable Airline Reservation Systems: A Microservices Approach Using AI and Deep Learning for Enhanced User Experience. In Proceedings of the 2025 4th International Conference on Sentiment Analysis and Deep Learning (ICSADL) (pp. 941–950). IEEE.

Benavente, V., Yantas, L., Moscol, I., Rodriguez, C., Inquilla, R., & Pomachagua, Y. (2022). Comparative Analysis of Microservices and Monolithic Architecture. In Proceedings of the 2022 14th International Conference on Computational Intelligence and Communication Networks (CICN) (pp. 177–184). IEEE.

Dragoni, N., Giallorenzo, S., Lluch Lafuente, A., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: Yesterday, Today, and Tomorrow. In M. Mazzara et al. (Eds.), Present and Ulterior Software Engineering (pp. 195–216). Springer, Cham.

Gretzel, U., Sigala, M., Xiang, Z., & Koo, C. (2015). Smart tourism: foundations and developments. Electronic Markets, 25(3), 179–188.

Lewis, J., & Fowler, M. (2014). Microservices: a definition of this new architectural term. Retrieved from https://martinfowler.com/articles/microservices.html

Martinez-García, L., Aciar, S., Mendoza, R., & Puello, J. J. (2018). Smart Tourism Platform Based on Microservice Architecture and Recommender Services. In Proceedings of the 15th International Conference on Mobile Web and Intelligent Information Systems (MobiWIS 2018) (pp. 39– Fifty). Springer, Cham.

Newman, S. (2015). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media.

Nirmala, B. P. W., & Lavianto, S. (2019). Utilization of Digital Enablers in the Transformation of Community-Based Tourism Village Marketing in Bali. Journal of Information Technology and Computers, 5(1), 148–157.

Oyeniran, O. C., Adewusi, A. O., Adeleke, A. G., Akwawa, L. A., & Azubuko, C. F. (2024). Microservices Architecture in Cloud-Native Applications: Design Patterns and Scalability. Computer Science & IT Research Journal, 5(9), 2107–2124

Pahl, C. (2015). Containerization and the PaaS Cloud. IEEE Cloud Computing, 2(3), 24–31.

Pencarelli, T. (2020). The digital revolution in the travel and tourism industry. Information Technology & Tourism, 22(3), 455–476.

Sun, Y., Nanda, S., & Jaeger, T. (2015, November). Security-as-a-Service for Microservices-Based Cloud Applications. In 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom) (pp. 50–57). IEEE.

Tapia, F., Mora, M. Á., Fuertes, W., Aules, H., Flores, E., & Toulkeridis, T. (2020). From Monolithic Systems to Microservices: A Comparative Study of Performance. Applied Sciences, 10(17), 5797.

Warmayana, I. G. A. K. (2018). Utilization of Digital Marketing in Tourism Promotion in the Industry 4.0 Era. Cultural Tourism: Journal of Religion and Culture, 3(2), 81–92.