Unit 9: Developing an API for a Distributed Environment

Required Reading

Pillai, A.B. (2017) Software Architecture with Python. Birmingham, UK. Packt Publishing Ltd.

- Chapter 5.

Summary
This chapter explores the principles and techniques of designing scalable software applications using Python. It delves into key topics like vertical and horizontal scalability, concurrency, and parallelism, emphasizing the importance of efficient resource management. Techniques such as multithreading, multiprocessing, and asynchronous execution are covered in detail, alongside Python's built-in modules like threading, asyncio, and multiprocessing. The text provides practical examples, such as a thumbnail generator and primality checker, to demonstrate the use of these concepts. Additionally, it highlights trade-offs between performance, scalability, and resource constraints, explaining how to choose between threading and multiprocessing based on the application's requirements.
Reflection
The content effectively illustrates the complexity of building scalable systems while balancing performance and resource optimization. By grounding theoretical concepts in real-world examples, it becomes evident how thoughtful architectural decisions can lead to significant efficiency gains. The discussion on Python's Global Interpreter Lock (GIL) and its impact on threading versus multiprocessing was particularly insightful, underscoring Python's inherent trade-offs in scalability. This reflection reiterates the importance of understanding both the tools and the underlying system behavior to build robust, high-performing applications.

Salah, T., Zemerly, M. J., et al. (2016) The Evolution of Distributed Systems towards Microservices Architectures, in Proc. of the 11th International Conference for Internet Technology and Secured Transactions.

Summary
The paper examines the evolution of distributed systems, tracing the journey from the traditional client-server paradigm to the contemporary microservices architecture. Initially, the client-server model laid the groundwork, followed by innovations such as mobile agents and Service-Oriented Architecture (SOA). While SOA introduced loose coupling and reusability, it struggled to meet modern demands for scalability and rapid development. Microservices emerged as a solution, offering modularity, elasticity, and lightweight communication through APIs. However, microservices introduce challenges like increased network communication, coordination complexity, and heightened security vulnerabilities. The paper compares these architectures and concludes that while microservices offer

significant advantages, their adoption requires careful consideration of trade-offs and operational challenges.

Reflection

The study provides a comprehensive exploration of how distributed systems have adapted to evolving technological and business needs. It is fascinating to see how each architectural shift—from monolithic to modular systems—has attempted to address the limitations of its predecessor. The emphasis on microservices as a response to modern demands for agility and scalability resonates with the current trends in software development. However, the challenges highlighted, such as the complexity of managing numerous independent services, underscore the need for robust tools and skilled teams. This reflection emphasizes the importance of balancing innovation with practicality, ensuring that new solutions are both effective and sustainable.

Winkfield, L., Hu, Y-H., Hoppa, & M. A. (2018) A Study of the Evolution of Secure Software Development Architectures, Journal of the Colloquium for Information System Security Education.

Summary

The article explores the evolution of secure software development architectures in response to emerging technologies like containers, microservices, DevOps, and Agile SDLC. These innovations, while enabling flexibility, scalability, and rapid development, also introduce new security challenges. The shift from monolithic to distributed systems, virtualization to containerization, and infrastructure-based to platform-based services has necessitated a rethinking of security practices. Developers are increasingly responsible for integrating security into the software development lifecycle (SDLC), requiring significant cultural, technical, and organizational changes. The article emphasizes the need for ongoing developer training and the adaptation of educational curriculums to address modern cybersecurity demands.

Reflection

The study highlights the interconnected evolution of software architectures and security practices, emphasizing the complex trade-offs between innovation and security. It reflects on the industry's struggle to balance rapid development with robust security measures, suggesting that the future lies in cultivating a security-aware developer workforce. This resonates as a pressing issue, especially in an era where the pace of technological advancements often outstrips the ability to secure them adequately. The emphasis on cultural transformation and education is insightful, as it underscores the necessity of not just technological shifts but also human-centric solutions for sustainable progress.

Additional Reading

Hadiyoso S., Alfaruq A., Hariyani Y.S., Rizal A., Riza T.A. (2021) Design and Implementation of a Registration System with Mobile Application at Public Health Center Based on IoT Using a RESTful API. In: Triwiyanto, Nugroho H.A., Rizal A., Caesarendra W. (eds) Proceedings of the 1st International Conference on Electronics, Biomedical Engineering, and Health Informatics. Lecture Notes in Electrical Engineering, 746. Springer, Singapore. DOI: https://doi.org/10.1007/978-981-33-6926-9_2

Nugroho, L. E., Azis, A., Mustika, I. W. & Selo (2017) Development of RESTful API to support the oil palm plantation monitoring system. 7th International Annual Engineering Seminar (InAES). 1-5. DOI: https://doi.org/10.1109/INAES.2017.8068545.

Sohan, S. M., Anslow, C. & Maurer, F. (2017) Automated example oriented REST API documentation at Cisco. IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP). 213-222. DOI: https://doi.org/10.1109/ICSE-SEIP.2017.11.