



MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES



**DESIGN OF A QUEUE-BASED
MICROSERVICES ARCHITECTURE AND
PERFORMANCE COMPARISON WITH
MONOLITH ARCHITECTURE**

KENAN CEBECİ

MASTER THESIS

Department of Computer Engineering

ADVISOR

Assist. Prof. Ömer KORÇAK

ISTANBUL, 2019



**MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES**



**DESIGN OF A QUEUE-BASED
MICROSERVICES ARCHITECTURE AND
PERFORMANCE COMPARISON WITH
MONOLITH ARCHITECTURE**

KENAN CEBECİ

(524111011)

MASTER THESIS

Department of Computer Engineering

ADVISOR

Assist. Prof. Ömer KORÇAK

ISTANBUL, 2019

MARMARA UNIVERSITY

INSTITUTE FOR GRADUATE STUDIES IN PURE AND APPLIED SCIENCES

Kenan CEBECİ, a Master of Science student of Marmara University Institute for Graduate Studies in Pure and Applied Sciences, defended her thesis entitled “**Design of A Queue-Based Microservices Architecture and Performance Comparison with Monolithic Architecture**”, on June 21, 2019 and has been found to be satisfactory by the jury members.

Jury Members

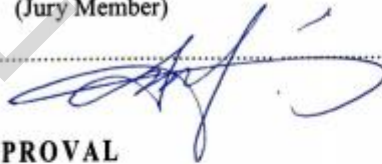
Assist. Prof. Dr. Ömer KORÇAK (Advisor)
Marmara University



Assoc. Prof. Dr. Murat Can GANİZ (Jury Member)
Marmara University



Assist. Prof. Dr. Ali NİZAM (Jury Member)
Fatih Sultan Mehmet Vakıf University



APPROVAL

Marmara University Institute for Graduate Studies in Pure and Applied Sciences Executive Committee approves that Kenan CEBECİ be granted the degree of Master of Science in Department of Computer Engineering, Computer Engineering Program on 07.08.19 (Resolution no: 2019/16-02).

Director of the Institute

Prof. Dr.
Bülent EKİCİ



ACKNOWLEDGMENT

I would like to express my gratitude to my thesis supervisor, Assist. Prof. Ömer Korçak, for his guidance, support and encouragement throughout my graduate study and completion of this thesis. I would also thank to Assoc. Prof. Murat Can Ganiz and Assist. Prof. Ali NİZAM for participating my thesis committee and their useful comments.

I would like to thank Assist. Prof. Gökay Burak Akkuş, Assist. Prof. Yaşar Safkan and İdil Gülnihal Sağlam for their support and friendship. I would like to Ramazan Çamcı , Onur Doğan and Ozan Tek for their helps during the measurements.

Finally, I would like to thank my family for their patience, encouragement and support during my whole life.

July, 2019

Kenan CEBECİ

TABLE OF CONTENTS

TABLE OF CONTENTS.....	iii
ÖZET.....	v
ABSTRACT.....	vi
SYMBOLS.....	vii
ABBREVIATIONS.....	viii
LIST OF FIGURES.....	x
LIST OF TABLES.....	xii
1. INTRODUCTION.....	1
1.1. Developer Operations (DevOps).....	2
1.2. Cloud Computing.....	3
1.3. Software Architectures.....	5
1.3.1. Monolith.....	6
1.3.2. Microservices.....	7
1.4. Related works and motivation.....	8
2. METHODOLOGICAL CONSIDERATION.....	11
2.1. Architectural Evaluation.....	11
2.1.1. Monolithic architecture.....	11
2.1.2. Service-based architectures.....	13
2.1.3. Service-oriented architecture versus microservices.....	15
2.1.4. Monolith versus microservices.....	16
3. PROPOSED DESIGN OF MICROSERVICES ARCHITECTURE.....	21
3.1. Communication.....	23
3.1.1. API gateway.....	24
3.1.2. Messaging Data Format Selection.....	28
3.1.3. Inter-microservices communication.....	28
3.2. Service Registry and Discovery.....	30
3.3. Modularity.....	35
3.4. Security.....	36
3.4.1. Authentication and Authorization.....	37
3.5. Database Selection.....	39
4. EXPERIMENTAL RESULTS.....	41
4.1. Test Environment.....	41

4.2. Performance of The Implemented Prototype Application	42
4.3. Database Performance of Monolith and Proposed MSA.....	45
5. CONCLUSION AND FUTURE WORK.....	48
REFERENCES	49
RESUME.....	1

PREVIEW

ÖZET

KUYRUK TABANLI BİR MİKROSERVİS MİMARİSİ TASARIMI VE MONOLİTİK MİMARİ İLE KARŞILAŞTIRILMASI

Kurumsal bir yazılım sisteminin oluşturması veya dönüşümü, iş ihtiyaçlarının tam olarak tanımlanmasını gerektiren meşakkatli bir işlemdir. İş gereksinimlerinin karşılanabilmesi için iyi düşünülmüş, uygun yazılım mimarisi kararlaştırılmalı ve tasarlanmalıdır. Genel olarak sorunlara çözüm bulmak için takip edilebilecek iki yöntem vardır. Birincisi geleneksel monolitik mimaride olduğu gibi problemi, doğru çözümü bulmak için bir bütün olarak ele almak. İkincisi ise problemi daha kolay anlaşılabilen ve çözülebilen küçük parçalara ayırmaktır. Eğer yazılım dünyasında ikinci yöntem takip edilecek olursa, mikroservis mimarisi gündeme gelmektedir. Kurumsal ölçekli yazılım sistemi tasarlanmak istendiğinde, bildiğimiz kadarıyla yazılım mimarilerini değerlendiren, iletişim protokolü, veri modeli ve veritabanının seçimini üzerine yol gösterici deneysel bir araştırma bulunmamaktadır. Bu tezde, kolay ölçeklenebilir, bakım yapılabilir, erişilebilirliği yüksek, güvenilir ve gözlemlenebilir mikroservis tabanlı bir yazılım sistemi tasarlanmıştır. Ayrıca amacına uygun yazılım mimarisi ve modellerini seçmeye yardımcı olabilecek şekilde farklı mimarilerin, iletişim protokollerinin ve veri modellerinin karşılaştırıldığı deneysel çalışmalar sunulmuştur. Tüm makale sadece sunucu servis tasarımı ile ilgili olup istemci tipi ve teknolojileri bu çalışmanın kapsamı dışındadır.

July, 2019

Kenan CEBECİ

ABSTRACT

DESIGN OF A QUEUE-BASED MICROSERVICES ARCHITECTURE AND PERFORMANCE COMPARISON WITH MONOLITH ARCHITECTURE

Building or transformation of an enterprise software system is an onerous process which requires precise definition of business demands. Then to enable the satisfaction of business requirements, the well-thought-of and convenient software architecture must be determined and designed. According to common sense, there are two methods to be followed in order to find the right solution for a problem. One is to handle the problem as a whole; like the traditional monolith architecture. The second method is to divide the problem into easily understandable and soluble fine-grains. If the second path is chosen in software world, the microservices architecture can be shown. When the entire enterprise level system design is considered, to the best of our knowledge, there is no any leading empirical research on the evaluation of software architectures, selection of communication protocol, data formats, and database. In this thesis, an easily scalable, maintainable, highly-available, reliable and observable software system is designed by comparing variant architectures, communication methods, and data models that would help to choose the most appropriate architecture or model for the right purpose. All the thesis is about designing a backend API system. The client types or technologies are out of scope.

July, 2019

Kenan CEBECİ

SYMBOLS

PREVIEW

ABBREVIATIONS

CSE	: Continuous Software Engineering
CPU	: Central Processing Unit
DevOps	: Developer Operations
SDLC	: Software Development Lifecycle
QoS	: Quality of Services
OS	: Operating System
IaaS	: Infrastructure-as-a-service
HaaS	: Hardware-as-a-service
PaaS	: Platform-as-a-service
SaaS	: Software-as-a-service
API	: Application Programming Interfaces
SOA	: Service Oriented Architecture
DDD	: Domain Driven Design
SRP	: Single Responsibility Principle
MSA	: Microservices Architecture
REST	:Representative State Services
RDBMS	: Relational Database Management System
EA	: Enterprise Architecture
SoC	: Separation of Concerns
XML	: Extensible Markup Language
JSON	: Java Object Notation
ACID	: Atomicity, consistency, isolation and durability
PoC	: Proof-of-Concept
IoT	: Internet of Things
AI	: Artificial Intelligence
IPC	: Inter-Process Communication
SOAP	: Simple Object Access Protocol

WSDL : Web Services Description Language
AMQP : Advanced Message Queuing Protocol
ESB : Enterprise Service Bus
JWT : Json Web Token

PREVIEW

LIST OF FIGURES

Figure 1.1 A standard monolith architecture design.....	6
Figure 1.2 Microservices architecture design.....	8
Figure 2.1 Service choreography.....	16
Figure 2.2 Service orchestration.....	16
Figure 2.3 SOA Scaling.....	18
Figure 3.1 Proposed Enterprise Software Architecture Design.....	22
Figure 3.2 Request lifecycle in API Gateway	26
Figure 3.3 A sample request message JSON.....	27
Figure 3.4 A sample response message JSON.....	27
Figure 3.5 Private queue usage for inter-microservices communication	30
Figure 3.6 [44] Client-side service discovery.....	32
Figure 3.7 [44] Server-side service discovery	33
Figure 3.8 Flow of the message director	34
Figure 3.9 Providing JWT token	38
Figure 3.10 Authentication and authorization flow.....	39
Figure 4.1 RabbitMQ and HTTP RestAPI performance comparison.....	43
Figure 4.2 Bubble Sort Response time for an integer array of 10000 items while instance count increase	44
Figure 4.3 Message processing velocity for Figure 4.2 test case	45
Figure 4.4 Performance comparison of monolith and proposed microservices for database bounded operations.....	46
Figure 4.5 Comparison of error rates percentages of monolith and proposed microservices for database bounded operations	47

PREVIEW

LIST OF TABLES

Table 4-1 Server Dedication Demonstration 41

Table 4-2 CPU Usage Percentage According to Concurrent Thread Count..... 42

PREVIEW