**T.C.**

**SOSYAL GÜVENLİK KURUMU BAŞKANLIĞI**

**MİKROSERVİS MİMARİSİNİN İNCELENMESİ: BİR SOSYAL GÜVENLİK KURUMU YAZILIM UYGULAMASI ÖRNEĞİ**

**Sosyal Güvenlik Uzmanlığı Tezi**

**Hazırlayan**

**Hallaç Mansur AKBAŞ**

**Tez Danışmanı**

**Murat BİNAY**

**Sosyal Güvenlik Uzmanı**

**Ağustos 2024**

**Ankara**

Hallaç Mansur AKBAŞ tarafından hazırlanan bu çalışmanın Sosyal Güvenlik Uzmanlığı Tezi olarak uygun olduğunu onaylarım. …./…./2024

Tez Danışmanının Adı SOYADI: Murat BİNAY

İmzası:

Bu çalışma, Kurulumuz tarafından oy birliği / oy çokluğu ile Sosyal Güvenlik Uzmanlığı Tezi olarak kabul edilmiştir.

|  |  |  |
| --- | --- | --- |
|  | **Adı Soyadı ve Unvanı** | **İmzası** |
| Başkan | : ………………………………………………… | …....………... |
| Üye | : …………………………………………………. | …....………... |
| Üye | : ………………………………………………… | …....………... |
| Üye | : …………………………………………………. | …....………... |
| Üye | : ………………………………………………… | …....………... |

Tarih: …./…./………

# 

# TEŞEKKÜR

Bu çalışma sırasında tecrübe ve bilgisi ile beni yönlendiren tez danışmanım Emre MUTLU’ya, ihtiyaç duyduğum konularda desteklerini esirgemeyen çalışma arkadaşlarım Hasan İbrahim KOÇAK’a ve Ramazan SARIALTIN’a, ayrıca her koşulda yanımda olan anneme, babama ve kardeşlerime teşekkürü borç bilirim.

**Ankara 2024 Hallaç Mansur AKBAŞ**

# BEYAN

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde akademik kurallara ve bilimsel etik davranış ilkelerine uyduğumu, ayrıca, bu kural ve ilkelerin gereği olarak, çalışmada bana ait olmayan tüm veri, düşünce ve sonuçları andığımı ve kaynağını gösterdiğimi beyan ederim.

…./…./……..

Adı ve SOYADI: Hallaç Mansur AKBAŞ

İmzası:

# ÖZET

**Sosyal Güvenlik Uzmanlığı Tezi**

**MİKROSERVİS MİMARİSİNİN İNCELENMESİ:**

**BİR SOSYAL GÜVENLİK KURUMU YAZILIM UYGULAMASI ÖRNEĞİ**

**Hallaç Mansur AKBAŞ**

İnce istemci, uzak birimlerdeki kullanıcı ihtiyaçlarını zamanla izler ve uzak birime merkezdeki kaynaklardan gerekli ve yeterli miktardakini ayırarak etkin çözümlerle ve daha az kaynak kullanımıyla önemli bir iyileştirme sağlar.

İnce istemci sanallaştırma teknolojisi kullanır. Sanallaştırma; fiziksel olarak merkezde olan büyük bir havuzu (yazılım sayesinde) ihtiyaç olduğu kadar mantıksal birimlere bölerek kaynakları daha verimli hale getirmektir. Böylece eskiden her birimde ayrı ayrı bulunan fiziksel kaynaktaki kullanılmayan kapasite daha etkin bir biçimde kullanılacak ve merkezi yönetimle sistem performansı kat kat artacak ve yönetim maliyetinin de düşük olmasını sağlayacaktır.

SGK, sunucu sanallaştırma altyapısı ile bütün hizmetlerini kamuya sunmaktadır. Sunucu sanallaştırma, merkezdeki etkinliği sağlamakla birlikte mevcut taşra altyapılarında bir sanallaştırma altyapısı olmadığından buralardaki performanslara doğrudan bir etki yapamaz. İşte tam da burada ince istemci mimarisi ve alt yapısının gerekliliği ortaya çıkmaktadır. Bu çalışmada ilk olarak mevcut sanallaştırma altyapısı incelenmiş, topolojisi çıkartılmış ve bileşenleri anlatılmıştır. Ardından ince istemci mimarisinin SGK taşra birimleri kullanıcı masa üstü ortamına uygulanabilirliği test edilmiştir ve olumlu sonuçlar alınmıştır. Ayrıca Bilişim Ağı, Enerji Verimliliği, Yönetim Politikası ve Felaket Kurtarma konularında kurum bilişim teknolojileri altyapısında uygulanabilecek çözümler belirlenmiştir. Tezin amacı İnce İstemci tekonolojileri kullanımı halinde Kurum’da haberleşme yönetim ve yedekleme açısından ortaya çıkan faydaların değerlendirilmesi ve ince istemci ile ilgili bilinç oluşmasıdır.

**Anahtar Kelimeler:** Mikroservis, Yazılım Mimarisi, Monolitik Mimari, Yazılım Tasarımı, Ölçeklenebilirlik, Konteynerizasyon, DevOps

# ABSTRACT

**Social Security Expertise Thesis**

**EXAMINATION OF MICROSERVICE ARCHITECTURE: A SAMPLE OF A SOCIAL SECURITY INSTITUTION SOFTWARE APPLICATION**

**Hallaç Mansur AKBAŞ**

Thin client, the remote unit monitor user needs over time and with the necessary and adequate amount of resources in the next one effective solution by separating the central resources and remote unit it provides a significant improvement with less resource use.

Thin client uses virtualization technology. Virtualization, with a large pool in the center physically (thanks to software) to logically divide up our resources that we need to become more efficient units. Thus enabling each unit formerly unused capacity on the physical resources available separately to be used more effectively and will increase central management with system performance and management many times that of the low.

SGK, offers all the services to the public with the server virtualization infrastructure. Server virtualization; while providing a direct impact on activity in the center performance can’t be a direct impact on province performance as there is not a virtualization infrastructure around the existing provincial infrastructure. It is arises here is exactly the thin client architecture and requirements of the infrastructure. In this study first of all, examined the existing virtualization infrastructure, topology designated and components described. Then SGK provincial units of the thin client architecture have been tested the applicability of the user desktop environment and positive results were. In addition, IT network, energy efficiency, management policies and disaster recovery solutions identified are applicable to corporate information technology infrastructure. The purpose of this thesis is if the use of thin client technologies at the institution, evaluation of the benefits in terms of communication management and backup and aimed to create awareness about thin client.

**Key Words:** Microservice, Software Architecture, Monolithic Architecture, Software Design, Scalability, Containerization, DevOps

# YÖNTEM

Bu çalışmada ulusal ve yabancı kaynaklardan faydalanılarak hazırlanmıştır. Sanallaştırmanın ve sanallaştırma türlerinin detaylı araştırılmasında yurtiçi ve yurtdışındaki makalelerden, analizlerden, dergi ve kitaplardan, kurumların yapmış oldukları çalışmalardan yararlanılmıştır. Ayrıca konu ile ilgili ulusal ve yabancı internet sitelerinin de ağırlıklı olarak kullanımı düşünülmektedir. Tezin yazımının üç aşamadan meydana gelmesi düşünülmektedir.

Birinci bölümde Sosyal Güvenlik Kurumu bilgi işlem ağı ve altyapısı, ince istemcinin temeli olan sanallaştırma kavramanın tarihçesinden, kurum veri merkezlerinden ve hali hazırda yürütüyor olunan projelerden bahsedilecektir. Ayrıca diğer sanallaştırma yöntemleri olan; sunucu sanallaştırma, masaüstü sanallaştırma, uygulama sanallaştırma, veri depolama sanallaştırma, ağ sanallaştırma, veri ve veritabanı sanallaştırma, bellek sanallaştırma gibi sanallaştırma türleri detaylı bir şekilde incelenecektir.

İkinci bölümde ise ince istemcinin ne olduğu, mimarisi, uygulama alanları ve ince istemci ile sanallaştırma modeli uygulanırken, hangi hususlara dikkat edilmesi gerektiği açıklanmıştır. Ayrıca ince istemcinin geleneksel masaüstü bilgisayarlardan mantıksal farkı ve ince istemciye geçiş süreci de yine bu bölümde incelenecektir.

Son bölümde ise ince istemci teknolojisi kullanılarak taşralarda sanallaştırmaya geçiş sonucu elde edilebilecek faydalar ve geçiş süreci anlatılmıştır. Yapılması düşünülen yeni sanallaştırma teknolojisinin Sosyal Güvenlik Kurumu’na olası getirileri riskleri ve götürüleri üzerinde durulmuştur. Ayrıca ince istemci teknolojisi kullanılarak oluşturulan sanallaştırmanın ne gibi güvenlik getirileri olduğu ve enerji verimliliği açısından (otuz bin kullanıcılı üç yüz ayrı ve uzak lokasyonda bulunan bir kurumda) ne gibi getirileri olduğu da bu bölümde anlatılmıştır. Ayrıca pilot bölge seçilen bir taşra biriminde birkaç kullanıcının ince istemci kullanarak günlük işlerini yapması olanağını oluşturup, bu durumun sonuçlarını kurum geneline göre detaylı analiz edilmesi düşünülmekte ve ince istemci teknolojisinin kullanıldığı sanal bir platform kurup bu platform üzerinden günlük işler yapıp buna göre bir inceleme yapılması düşünülmektedir.

# İÇİNDEKİLER

[TEŞEKKÜR i](#_Toc165367899)

[BEYAN ii](#_Toc165367900)

[ÖZET iii](#_Toc165367901)

[ABSTRACT iv](#_Toc165367902)

[YÖNTEM v](#_Toc165367903)

[İÇİNDEKİLER vi](#_Toc165367904)

[ŞEKİLLER LİSTESİ vii](#_Toc165367905)

[TABLOLAR LİSTESİ viii](#_Toc165367906)

[KISALTMALAR ix](#_Toc165367907)

[GİRİŞ 1](#_Toc165367908)

[BİRİNCİ BÖLÜM 2](#_Toc165367909)

[YAZILIM MİMARİLERİ 2](#_Toc165367910)

[**1. YAZILIM MİMARİSİ 2**](#_Toc165367911)

[**1.1. Yazılım Mimarisi Tanımı 2**](#_Toc165367912)

[**1.2. Yazılım Mimarisinin Yararları 2**](#_Toc165367913)

[**1.3. Yazılım Mimarisi Çeşitleri 3**](#_Toc165367914)

[**1.3.1. Monolitik Mimari 3**](#_Toc165367915)

[**1.3.2. Servis Yönelimli Mimari 3**](#_Toc165367916)

[**1.3.3. Mikroservis Mimarisi 3**](#_Toc165367917)

[İKİNCİ BÖLÜM 4](#_Toc165367918)

[MİKROSERVİSLERLE KULLANILAN TEKNOLOJİLER 4](#_Toc165367919)

[**1. KURUM BİLGİ İŞLEM MİMARİSİ 4**](#_Toc165367920)

[**1.1.** **Aaaaaa aaaaaaaaa aaaaaaaaa** 4](#_Toc165367921)

[**1.2.** **Bbbbbbbbb bbbbbb** 4](#_Toc165367922)

[**1.2.1. Ccccccc cccccc cccccccc ccccccccccccccccccc 4**](#_Toc165367923)

[**1.2.2. Ddddddddddd 4**](#_Toc165367924)

[**1.2.2.1. Ddddddddddd 4**](#_Toc165367925)

[ÜÇÜNCÜ BÖLÜM 5](#_Toc165367926)

[UYGULAMA ÖRNEĞİ 5](#_Toc165367927)

[**1. KURUM BİLGİ İŞLEM MİMARİSİ 5**](#_Toc165367928)

[**1.1. Aaaaaa aaaaaaaaa aaaaaaaaa 5**](#_Toc165367929)

[**1.2. Bbbbbbbbb bbbbbb 5**](#_Toc165367930)

[**1.2.1. Ccccccc cccccc cccccccc ccccccccccccccccccc 5**](#_Toc165367931)

[**1.2.2. Ddddddddddd 5**](#_Toc165367932)

[**1.2.2.1. Ddddddddddd 5**](#_Toc165367933)

[DÖRDÜNCÜ BÖLÜM 6](#_Toc165367934)

[UYGULMA ÖRNEĞİ 6](#_Toc165367935)

[**1. KURUM BİLGİ İŞLEM MİMARİSİ 6**](#_Toc165367936)

[**1.1. Aaaaaa aaaaaaaaa aaaaaaaaa 6**](#_Toc165367937)

[**1.2. Bbbbbbbbb bbbbbb 6**](#_Toc165367938)

[**1.2.1. Ccccccc cccccc cccccccc ccccccccccccccccccc 6**](#_Toc165367939)

[**1.2.2. Ddddddddddd 6**](#_Toc165367940)

[**1.2.2.1. Ddddddddddd 6**](#_Toc165367941)

[SONUÇ ve ÖNERİLER 7](#_Toc165367942)

[KAYNAKÇA 8](#_Toc165367943)

# ŞEKİLLER LİSTESİ

# TABLOLAR LİSTESİ

# KISALTMALAR

# GİRİŞ

Günümüzün hızla gelişen ve değişen teknoloji ortamında yazılım geliştirme süreç ve yöntemleri ihtiyaçlara cevap verebilmek için sürekli bir dönüşüm içindedir. Hız, performans ve depolama kapasiteleri artan bilgisayar sistemleri daha karmaşık ve güçlü yazılımların geliştirilmesine imkan sağlamıştır. Bunun sonucunda üretilen büyük ölçekli yazılımları geliştirmek ve yönetmek için yeni mimari yaklaşımlar ortaya çıkmıştır.

Klasik yazılım geliştirme mimarisinde tek bir kod tabanı üzerinde geliştirme yapılır ve geliştirilen yazılım bir bütün halinde kullanıma alınır. Özellikle büyük ölçekli yazılımların geliştirilmesi ve yönetilmesi konusunda bu yaklaşım bazı sınırlamaları ve zorlukları beraberinde getirir. Bu mimarinin getirdiği zorlukları aşmak ve daha esnek, ölçeklenebilir ve yönetilebilir yazılım çözümleri sunabilmek için farklı bir mimari model olan mikroservis mimarisi ortaya çıkmıştır.

Mikroservis mimarisinin kullanımı Netflix, Amazon, Spotify gibi dünyadaki önemli kurumsal şirketlerin bu mimariyi kullanmada öncülük etmesiyle yazılım dünyasında yaygınlaşmıştır (Cebeci & Korçak, 2020). Bu mimari büyük monolitik uygulamaların yerine küçük, bağımsız ve birbiri ile entegre çalışabilen servislerden oluşan bir uygulama yapısı önermektedir. Her bir mikroservis, belirli bir işlevi gerçekleştirir ve bu servisler bir araya gelerek büyük uygulamaları oluşturur.

SGK bünyesinde hizmet veren yazılımlar, milyonlarca insanın hayatını doğrudan etkileyen kritik öneme sahip yazılımlardır. Bu sistemlerin doğru bir şekilde tasarlanması ve geliştirilmesi hem kurumun işleyişi hem de hizmet verilen vatandaşlar için büyük önem taşımaktadır. Gerek büyük uygulama yazılımlarının karmaşıklığının üstesinden gelmek gerekse yazılım ekiplerinin rahat ve uyumlu bir şekilde çalışabilmesini sağlamak için sunulan bir çözüm olan mikroservis mimarisinin kurum yapısına uygunluğu ve kullanılabilirliği değerlendirilecektir. Avantajları ve dezavantajları göz önüne alınarak ne kadar optimize bir çözüm olduğu hem teknik açıdan hem kültürel açıdan ele alınacaktır.

# BİRİNCİ BÖLÜM

# YAZILIM MİMARİLERİ

## YAZILIM MİMARİSİ

### Yazılım Mimarisi Tanımı

Yazılım mimarisi, bir yazılım sisteminin oluşturulması için gerekli olan yapıların ve bu yapıları oluşturmak için izlenen disiplinin bütünüdür. Yazılım mimarisi, yazılım bileşenlerini, bu bileşenler arasındaki ilişkileri ve hem bileşenlerin hem de ilişkilerinin özelliklerini içerir (Bass, et al., 2012, p. 3).

Yazılımlar belirli bir işi yerine getirmek için yapılır. Yazılım planının soyut bir hedef halinden bir işleme çözüm getiren somut bir hale gelmesi kompleks bir süreçtir. Yazılım mimarisi, süreçleri yönetebilmek ve iş hedeflerini gerçekleştirebilmek için teorik bir çerçeve sağlar (Bass, et al., 2012, p. 2). Bu çerçeve, iş gereksinimlerini, kullanıcı ihtiyaçlarını ve teknik kısıtlamaları dikkate alarak somut ürünün meydana gelmesinde kritik bir rol oynar.

### Yazılım Mimarisinin Yararları

Yazılım mimarisinin sağladıkları yararlar aşağıdaki gibidir:

* Yazılım sisteminin temel yapısını ve bileşenlerinin birbiriyle nasıl etkileşime girdiğini tanımlar. Tüm paydaşlar için daha anlaşılabilir hale getirir (Bass, et al., 2012, p. 2).
* Süreç yönetimi ve maliyet tahmini yapılmasını sağlar (Perry & Wolf, 1992).
* Paydaşlar arasındaki iletişimi güçlendirir (Bass, et al., 2012, p. 21).
* Yapılan yazılım mimarisinin parçaları farklı sistemlerde yeniden kullanılabilir. Bu geliştirme süresini ve maliyetini kısaltmaya yardımcı olur (Bass, et al., 2012, p. 21).
* Yazılım gelecekteki muhtemel değişikliklere uyum sağlayabilecek bir yapıda geliştirilmesini sağlar.
* Yazılımın esnekliğini ve ölçeklenebilirliğini artırır. Böylece yazılımın büyütülebilmesine ve değiştirilebilmesine olanak sağlar.

### Yazılım Mimarisi Çeşitleri

Yazılım mimarisinin evrimi, değişen teknolojik gereksinimler ve iş zorluklarına cevap olarak gelişmiştir. Yazılım mimarisi literatüründe adı geçen üç mimari stili incelenecektir.

### Monolitik Mimari

Monolitik mimari, yazılım geliştirme dünyasında uzun süredir kullanılan geleneksel bir mimari yaklaşımdır. Bu yaklaşım, yazılımın tüm fonksiyonlarının tek bir birim olarak geliştirilmesini ve dağıtılmasını ifade eder (Chen, et al., 2017, pp. 466-475). Bu yaklaşımda kullanıcı arayüzü, iş mantığı, veri erişim katmanı ve uygulama entegrasyonu işlevlerinin tümü tek parça halindedir. Tek bir kod tabanı üzerine kurulu bu sistemler, başlangıçta geliştirilmesi ve anlaşılması kolay sistemlerdir. Çünkü tüm bileşenler tek bir programlama dilinde ve çalışma zamanı ortamında geliştirilir.

Zamanla yazılımın ihtiyaçları karşılaması için yapılan değişiklikler ve eklemelerle yazılımın boyutu ve kod karmaşıklığı artar. Bu durum yazılımın ölçeklendirme ve sürekli entegrasyon gibi ihtiyaçları karşılayamamasına neden olur. Kodda yapılan herhangi bir değişiklik bütün yapının yeniden konuşlandırılmasını(deploy) gerektirir. Ayrıca, yazılımın herhangi bir yerindeki hata tüm yazılımın çalışmamasına neden olabilir. Bu durum hata izolasyonunun ve sistem esnekliğinin olmaması demektir.

Bu mimari yaklaşımın özellikle daha küçük yazılımlar için veya bir yazılımın boyutu ve kapsamı nispeten sabit ve yönetilebilir olduğunda çeşitli avantajları da vardır. Tüm bileşenler aynı uygulamada olduğundan uygulamanın farklı bölümleri arasındaki iletişim hızlıdır. Hata ayıklama ve test etme işlemleri basittir. Harici hizmetlere herhangi bir bağımlılık olmaksızın yönetilecek tek bir uygulama olduğundan konuşlandırma(deploy) kolaydır.

### Servis Odaklı Mimari

Servis-odaklı mimari (SOA), sistem bileşenlerinin bağımsız hizmetler olduğu dağıtılmış sistemler geliştirmenin bir yoludur (Sommerville, 2010). Bileşenler birbirleriyle ağ üzerinden bir iletişim protokolü ile haberleşir. SOA’da temel amaç işlevleri farklı servislere ayırmaktır.

SOA kullanımının yazılım sistemlerine esnek verimli ve güvenli bir şekilde çalışabilmesi için getirdiği bazı avantajlar vardır. Bunlar:

* SOA’da her servis spesifik bir işlevi yerine getirmek üzere oluşturulur. Bu servisler gerektiğinde yeniden kullanılabilir. Bu da iş ve zamandan tasarruf edilmesini sağlar.
* Servisler işlevlerini yerine getirirken iç uygulamanın mantığını yani nasıl yaptıklarını kullanıcıdan gizlerler. Kullanıcıya basit bir arayüz sunarak karmaşıklığı ortadan kaldırır.
* Servisler birbirinden bağımsız olarak geliştirilebilir ve konuşlandırılabilir.
* Değişen iş gereksinimlerini karşılamak için gelişmiş esneklik sağlar. Yeni servisler eklenebilir, var olan servisler güncellenebilir ve iş süreçleri yeniden yapılandırılabilir.
* Farklı sistemlerin birlikte çalışabilmesi ve veri paylaşması kolaydır.
* Modüler bir yapıya sahiptir ve sistem bileşenleri izole bir şekilde çalıştığından güncellenmesi durumunda birbirinden etkilenmez.
* Ölçeklenebilirliği sayesinde artan kullanıcı sayısı ve/veya iş yükü durumunda sistem performansını korumak için sistem büyütülebilir.

Bunun yanı sıra SOA bazı dezavantajları da beraberinde getirir. Bunlar:

* Ağ performansı ve servisler arası iletişim gecikmeleri sistemin çalışma performansını yavaşlatabilir.
* Sistem birçok servis içermesi durumunda bu servisler arasındaki iletişimi yönetmek ve izlemek zor olabilir. Bu karmaşıklık hataları tespit etmeyi ve çözmeyi zorlaştırır.
* Birçok farklı servis bağlantı noktası içerdiğinden güvenlik riskleri doğar. Servislerin arasındaki iletişimin güvenliğini sağlamak ek güvenlik önlemleri gerektirir.

### Mikroservis Mimarisi

# İKİNCİ BÖLÜM

# MİKROSERVİSLERLE KULLANILAN TEKNOLOJİLER

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

## KURUM BİLGİ İŞLEM MİMARİSİ

### Aaaaaa aaaaaaaaa aaaaaaaaa

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

### Bbbbbbbbb bbbbbb

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

### 1.2.1. Ccccccc cccccc cccccccc ccccccccccccccccccc

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

### 1.2.2. Ddddddddddd

### 1.2.2.1. Ddddddddddd

# ÜÇÜNCÜ BÖLÜM

# UYGULAMA ÖRNEĞİ

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

## KURUM BİLGİ İŞLEM MİMARİSİ

### Aaaaaa aaaaaaaaa aaaaaaaaa

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

### Bbbbbbbbb bbbbbb

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

### 1.2.1. Ccccccc cccccc cccccccc ccccccccccccccccccc

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

### 1.2.2. Ddddddddddd

### 1.2.2.1. Ddddddddddd

# DÖRDÜNCÜ BÖLÜM

# UYGULMA ÖRNEĞİ

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

## KURUM BİLGİ İŞLEM MİMARİSİ

### Aaaaaa aaaaaaaaa aaaaaaaaa

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

### Bbbbbbbbb bbbbbb

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

### 1.2.1. Ccccccc cccccc cccccccc ccccccccccccccccccc

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

### 1.2.2. Ddddddddddd

### 1.2.2.1. Ddddddddddd

# SONUÇ ve ÖNERİLER

# KAYNAKÇA