Dicle University
Journal of Engineering
https://dergipark.org.tr/tr/pub/dumf
duje.dicle.edu.tr

Araştırma Makalesi / Research Article

Çoklu bulut teknolojisi yönetimi için web tabanlı bir araç tasarımı ve geliştirilmesi

Design and development of a web-based tool for multi-cloud technology

Hacer ÖZYURT[1,*], Yusuf Can AKIER[2], Özcan ÖZYURT[3]

[1]Karadeniz Teknik Üniversitesi, Of Teknoloji Fakültesi, Yazılım Mühendisliği, hacerozyurt@ktu.edu.tr
ORCID: https://orcid.org/0000-0001-8621-2335

[2]Karadeniz Teknik Üniversitesi, Of Teknoloji Fakültesi, Yazılım Mühendisliği, yusufcanakier@outlook.com
ORCID: https://orcid.org/0009-0008-9926-268X

[3]Karadeniz Teknik Üniversitesi, Of Teknoloji Fakültesi, Yazılım Mühendisliği, oozyurt@ktu.edu.tr
ORCID: https://orcid.org/0000-0002-0047-6813

MAKALE BİLGİLERİ

Makale Geçmişi:

Geliş 3 Kasım 2023
Revizyon 6 Ocak 2024
Kabul 3 Şubat 2024
Online 29 Mart 2024

Anahtar Kelimeler:

Container yönetimi, Çoklu bulut tabanlı mimari, Hizmet olarak yazılım, Kubernetes

ÖZ

Yazılım hizmeti sağlayıcıları, değişen ve gelişen teknoloji araçlarını takip etmekten ve geliştirici envanterlerini güncel tutmaktan sorumludur. Kuruluşlar genellikle kendi bünyelerinde değişen teknolojilere hızla adapte olabilmek için ekiplerini genişletmeyi ya da proje sürelerini uzatmayı tercih etmektedir. Fiziksel makinelerle başlayan yazılım sektörü, günümüzde sanallaştırma ve konteyner mimarisini takip etmektedir. Mikro servis mimarisinin şirketlerde yaygınlaşmasıyla birlikte konteynerler üzerinde çalışan uygulamaların sayısı ciddi oranda artmıştır. Bu durum konteynerlerin yönetilmesinde zaman, güvenlik, performans gibi sorunları da beraberinde getirmiştir. Konteyner yönetim araçlarından biri olan Kubernetes, bir komut satırı ara yüzü üzerinden haberleşerek konteynerleri yönetme imkânı sağlamaktadır. Bu çalışmada, birden fazla sektöre ve/veya işletmeye hizmet verirken hizmet olarak yazılım sağlayıcı kuruluşların değişen ihtiyaçlarına göre zaman ve ekip maliyeti konularını en aza indirmeyi hedefleyen bir yönetim aracının tasarlanması ve geliştirilmesi amaçlanmıştır. Bu doğrultuda Kubernetes kümeleme, bulut ortamında sanallaştırma ve çoklu bulut tabanlı bir mimaride geliştiricilerin altyapılarını yönetebilecekleri bir web uygulaması geliştirilmiştir. Kullanıcıların ilgili bulut sağlayıcılarından aldıkları erişim anahtarları ile uygulamaya giriş yapabilecekleri ve altyapılarını yönetebilecekleri kullanıcı dostu bir ara yüz tasarlanmıştır. Bu ara yüz ile son kullanıcıların ve geliştiricilerin istedikleri teknolojinin altyapı araçlarını hızlı bir şekilde yönetebildikleri ve basit bir dokümantasyon yöntemi ile birden fazla teknoloji arasında geçiş yapmakta zorlanmadıkları gözlemlenmiştir. Geliştirilen aracın bu alandaki araştırmalara örnek teşkil etmesi beklenmektedir.

ARTICLE INFO

Article history:

Received 3 November 2023
Received in revised form 6 January 2024
Accepted 3 February 2024
Available online 29 March 2024

Keywords:

Container management, Multi-cloud-based architecture, Software as a service, Kubernetes

Doi: 10.24012/dumf.1385760

* Sorumlu Yazar

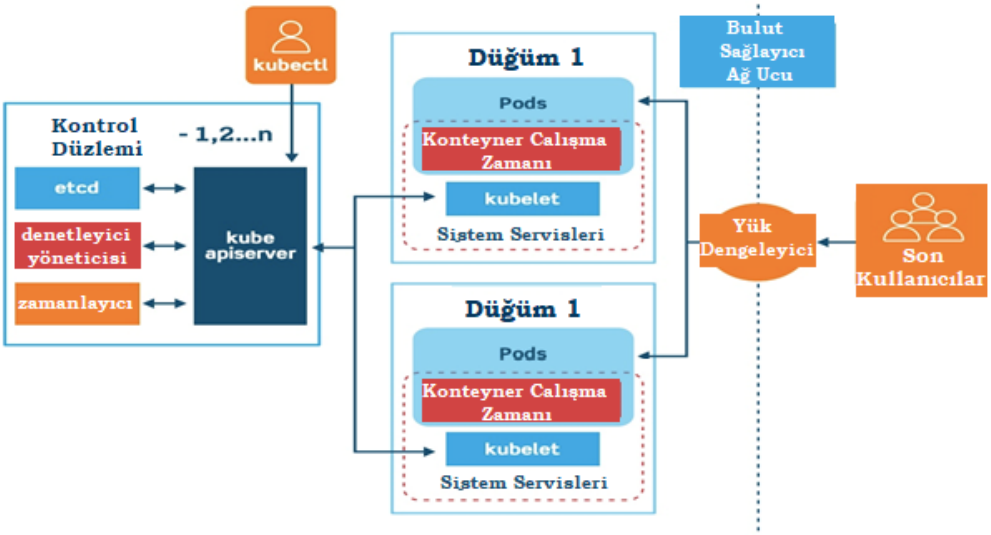
ABSTRACT

Software service providers are responsible for keeping up to date with changing and evolving technology tools and keeping their developer inventories up to date. Organizations generally prefer to expand their teams or extend project durations in order to quickly adapt to changing technologies within their own organization. The software industry, which started with physical machines, today follows virtualization and container architecture. With the widespread use of micro service architecture in companies, the number of applications running on containers has increased significantly. This situation has brought problems such as time, security, and performance in managing containers. Kubernetes, one of the container management tools, provides the ability to manage containers by communicating through a command-line interface. In this study, it is aimed at designing and developing a management tool that aims to minimize time and team cost issues according to the changing needs of software-as-a-service provider organizations while serving multiple sectors and/or businesses. In this direction, Kubernetes Cluster, virtualization in the cloud environment, and a web application where developers can manage their infrastructure in a multi-cloud-based architecture have been developed. A user-friendly interface has been designed where users can log in to the application with the access keys obtained from the relevant cloud providers and manage their infrastructure. With this interface, it has been observed that end users and developers can quickly manage the infrastructure tools of the desired technology and do not have difficulty switching between multiple technologies with a simple documentation method. The developed tool is expected to set an example for research in this field.

Giriş

Dijital dönüşüm altyapılarındaki hızlı teknolojik değişiklikler işletmelerin teknolojiye uyum süreçlerini zorlaştırmış, bu da bulut tabanlı altyapılara geçişe ön ayak olmuştur. Bulut bilişimde birçok katman hazır birer servis olarak sunulduğundan bu altyapı hem zaman hem de güncelleme maliyetinden kaynaklanan işgücünü azaltması açısından avantaj sunmaktadır [1]. Günümüzde bulut bilişim teknolojisi olarak pek çok servis sağlayıcısı ve yönetim aracı bulunmaktadır. Bu araçlar işletmelerin teknik sorunlarına çözüm getirmekle birlikte güvenlik ve yönetim gibi süreçler karmaşık hale gelebilmektedir. Bu durum da kurumlara ekstra maliyet olarak yansımaktadır. Yönetimi ve güvenliği karmaşık hale gelen teknolojilerden biri de konteyner teknolojileridir. Konteynerlerin servis ayarları, dosya yapıları, ağ ayarları, güvenlik ve uygulama yapılandırmaları gibi pek çok noktada yönetimsel karmaşıklık meydana gelmektedir [2]. Teknolojik yaklaşımlar zamanla ara yazılımlar, sanal makineler ve sanal cihazlara kadar değişmiştir. Docker, konteyner (kapsayıcı) tabanlı bir yaklaşımı temsil etmekte ve bu amaçla yaygın olarak kullanılmaktadır [3]. Docker teknolojilerin başında Kubernetes gelmektedir. Kubernetes, bulut ortamında konteyner uygulama dağıtımı için yeni bir standarttır. Konteyner düzenleme sistemi, özellikle büyük ölçekli işletmeler için önemli fırsatlar ve çözümler sunmaktadır. Kubernetes konteynerlerin orkestrasyonunu sağlarken getirdiği otomatik ölçeklendirme, depolamayı yönetme, sürümleri kontrol etme gibi pozitif özellikleri sayesinde sektörde önemli bir yere sahip olmuştur. Konteynerlerin ölçeklendirilmesi, yönetimi ve izlemesi kubernetes teknolojisi ile oldukça kolay hale gelmektedir. Kubernetes altyapısında temel alınan çalışma mantığı istemci-sunucu mimarisidir. Sunucu üzerinde Uygulama Programlama Arabirimi (API), sunucu, etcd (veri tabanının anlık durumunu saklayan yapı), zamanlayıcı, kontrol yöneticisi gibi temel Kubernetes yapıları çalışmaktadır. İstemci düğümleri ise konteyner çalışma zamanı, Kubelet ve Kube-proxy bileşenlerinden oluşmaktadır. Düğümler üzerinde podlar ve servisler çalışmaktadır. Sunucu düğümlerin erişiminin sağlanması için 443 numaralı porttan bağlantı gerekmektedir. Kubernetes üzerindeki sunucu düğümlerde iki farklı erişim yolu mevcuttur. Bunlardan ilki düğümler içerisinde bulunan kubelet bileşeni üzerinden erişim sağlanmasıdır. Şekil 1'de Kubernetes çalışma prensibi gösterilmiştir. Kubernetes çalışma prensibine göre Kubelet aracılığıyla podlara ait kayıtlara erişilmektedir. Kubeletin sağladığı port yönlendirme ile çalışmakta olan podların bağlantısı kurulmaktadır. Bir diğer yöntem ise http bağlantılıları kullanılarak podlara ve servislere doğrudan erişimdir. Bu yöntem güvenlik açığı oluşturabileceğinden halka açık ağlarda kullanılmaması gereken bir yöntemdir. Kubernetes için zorunlu olan cluster kurulumunda iş yükünü hafifletmek için Kubeadm ve Kubespray araçları bütünleşmiş şekilde gelmektedir. Gerçekleştirim için iyi bir ağ bağlantısı, sistem tasarımı, terminal, Linux komutları, Go programlama dili ve konteyner teknolojisine hakimiyet gerekmektedir [4]. Kubernetes ve bulut ortamındaki katmanlı altyapıya yönelik bir çalışma gerçekleştirilmesi, gelişen yazılım sektöründe karmaşık hale gelen mikro servis yönetiminin kolaylaştırılması ve monolitik mimariden uzaklaşılması gerekli görülmektir.

Bu çalışmada Kubernetes altyapılarını bulut ortamında gerçeklemek ve bu altyapıların bir kullanıcı ara yüzü ile son kullanıcı tarafından yönetimini sağlamak için web tabanlı bir yazılım tasarlanmış ve geliştirilmiştir. Geliştirilen bu yazılım (Çalışma kapsamında geliştirilen Web tabanlı Çoklu Bulut Teknolojisi Yönetimi Yazılımı - WtÇBTYY olarak kısaltılmış ve metin boyunca bu isimle anılmıştır) ile DevOps ve site güvenilirlik mühendisliğinde çalışan bireylerin iş yükünün azalması amaçlanmıştır.



Şekil 1. Kubernetes sisteminin çalışma prensibi

Metot

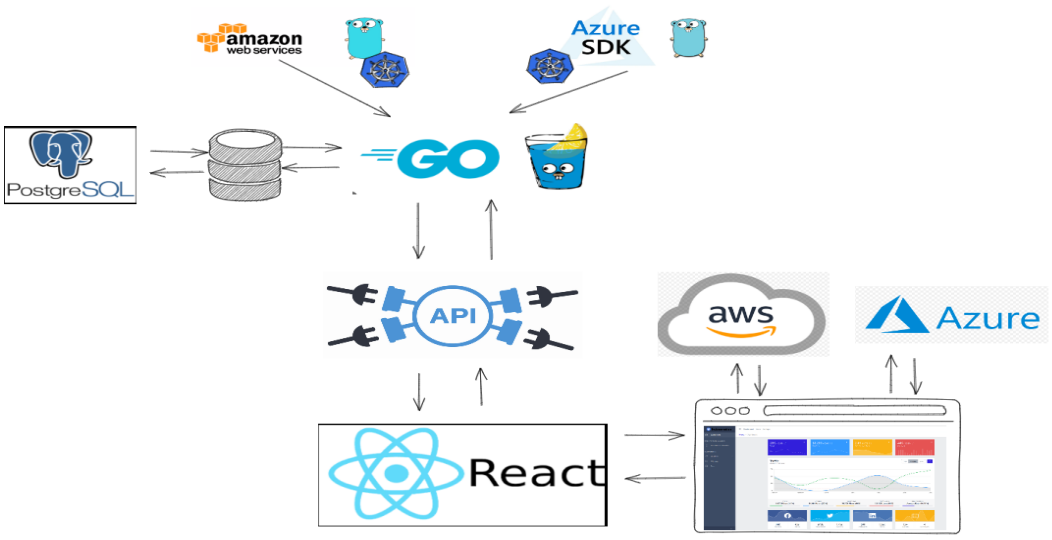
Bu çalışmada çeşitli sektörlere ve işletmelere hizmet sunan yazılım sağlayıcı kuruluşlar, değişen ihtiyaçlara göre zaman ve ekip maliyetini minimize etmeyi hedefleyen bir yönetim aracının (WtÇBTYY) tasarlanması ve geliştirilmesi amaçlanmıştır. Bu amaç doğrultusunda, Kubernetes kümeleme, bulut ortamında sanallaştırma ve çoklu bulut tabanlı bir mimari üzerine odaklanan bir web uygulaması geliştirilmiştir. Bu bölümde WtÇBTYY'nin tasarlanması ve geliştirilmesine yönelik yazılım yaşam döngüsü süreçlerine yer verilmiştir.

Planlama ve analiz

Bu aşamada bulut hizmeti teknolojisi öncüleri olan Google Cloud, Amazon Web Services ve Microsoft Azure platformlarında deneme hesapları oluşturulmuştur. Bu hesaplar ile bulut mimarileri, güvenlik kuralları, kullanıcı ara yüzleri ve performansları incelenmiştir. Bulut bilişimin hizmet türlerinden birisi olan "hizmet olarak yazılım" bulut sağlayıcıların bulut ortamında yazılım hizmeti vermesi ve "kullandığın kadar öde" olanağına mantığına dayanan bir modeldir. Doksanlı yılların sonundan günümüze kadar pek çok hizmet olarak yazılım uygulaması geliştirilmiştir. Bu uygulamaların çoğu tek bir işi yapmak amacıyla tasarlanmış ve yıllık zorunlu abonelikler ile ya da zorunlu güncelleme/bakım hizmetleriyle geliştirici/ kullanıcı için çok büyük maliyetler oluşturmuştur. Her uygulama için yeni bir satın alma, hesap takibi, ara yüz değişimi ve kullanıcı alışkanlığı gibi zaman maliyeti de oluşturabilecek problemler gündeme gelmiştir. WtÇBTYY'nin en önemli özelliği zenginleştirilmiş kullanıcı ara yüzü sunumudur. Geliştirici/kullanıcı hizmet olarak yazılım sağlayıcılarının sunduğu API uygulamaları sayesinde ilgili bütün bulut tabanlı yazılımları tek bir ekran üzerinden yönetebilmektedir. Hizmet olarak yazılım modelinde öne çıkan kavramlardan birisi de DevOps kavramıdır. Ürünün bulut sunucusu ve DevOps tarafındaki arka uç operasyonlarının doğru yürütülebilmesi, işletim sisteminin ihtiyaca yönelik kurulması ve Kubernetes kümelerinin yığın depolayıcıda işlenebilmesi için en dengeli ve kararlı kurulumların yapılması gerekmektedir. Versiyon ve kaynak yönetimi doğru kurgulandığında sürdürülebilirlik açısından önemli bir sorun ortadan kalkmaktadır.

Proje yazılım mimarisi

Bu aşamada genelden özele bütün süreç Birleşik Modelleme Dili (Unified Modelling Language- UML) diyagramları ile şema haline getirilmesi gerekmektedir. Tüm süreçlerin çalışma ve veri tabanı kurgusu belirlenmeli ve sunucudan istemciye her bir adım şema haline getirilmelidir. Şekil 2'de WtÇBTYY'nin tasarım ve mimarisi görülmektedir. Şekil 2'ye göre ilk olarak Go Programlama dili ile arka uç kodların geliştirilmesi gerekmektedir. Bu kodlar ile hizmet olarak yazılım veren yapıların API end-pointlere sunucudan cevap verecek program oluşturulmalıdır. Sağ altta React çerçevesi kullanılarak geliştirilmiş ara yüzün ekran görüntüsü görülmektedir. Bu ara yüz ile kullanıcı etkileşime girdiğinde ilgili sağlayıcı sunucuya POST veya GET isteği göndermekte ve etkileşim sonucunu izleyerek veri tabanına kaydedebilmektedir.



Şekil 2. Altyapı yönetim uygulaması yazılım ve tasarım mimarisi

Hizmet olarak yazılım

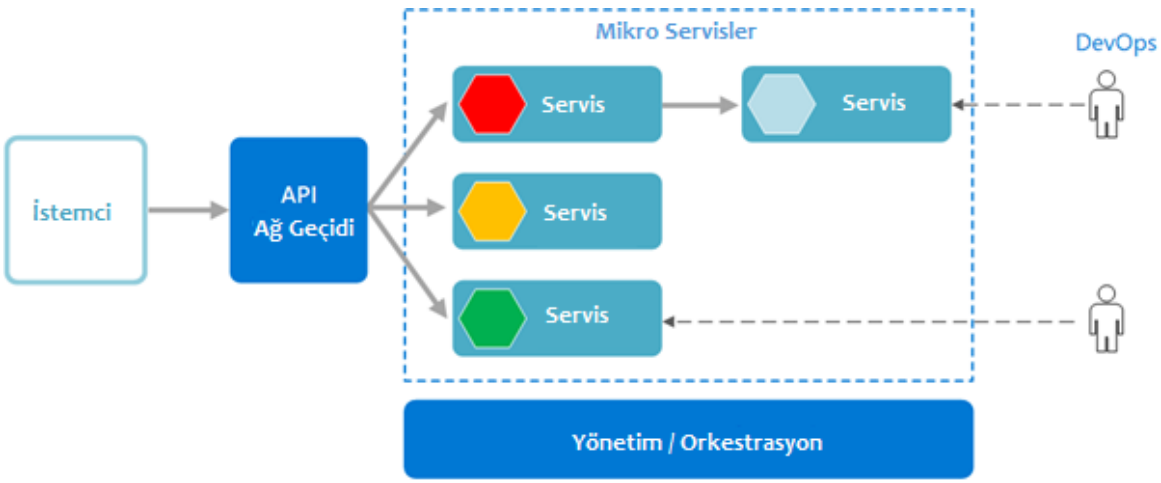
Herhangi bir sistem kurulumu gerekmeksizin, internet tarayıcısı üzerindeki bulut tabanlı platformlardan erişilebilen uygulamalara "Hizmet olarak yazılım" modeli adı verilmektedir. Bu çalışmada geliştirilen ve "Çoklu bulut tabanlı altyapı yönetim aracı" olarak adlandırılan yazılımda olduğu gibi müşteriler alt yapıdaki ağ, sunucu, işletim sistemi ve depolama aygıtları gibi bileşenleri yönetmez veya denetlemez. Bununla birlikte bu modelde kullanıcıya has uygulama ayarları yapılabilir [5]. Tablo 1'de müşteri ve sağlayıcı tarafların "hizmet olarak yazılım" modelindeki sorumluluklar listelenmiştir [6].

Tablo 1. Hizmet olarak yazılım modelinde müşteri ve sağlayıcı sorumlulukları

Müşteri	Sağlayıcı
• Müşteri verilerinin toplanması ve işlenmesi konusundaki veri koruma yasalarına uyum • Kimlik yönetim sisteminin bakımı • Kimlik doğrulama platformunun yönetimi (şifre politikalarının uygulanması dahil)	• Fiziksel destek altyapısı • Fiziksel altyapı güvenliği ve kullanılabilirlik (Sunucular, depolama, ağ bant genişliği) • İşletim sistemi yama yönetimi ve sıkılaştırma prosedürleri • Güvenlik platform yapılandırmaları • Sistem takibi ve izleme • Güvenlik Platformu bakımları (Güvenlik Duvarı, Host IDS, Anti virüs)

Mikro servis mimarisi

Mikro servisler bulut ve konteynerizasyon teknolojileriyle birlikte son dönemlerde vazgeçilmez haline gelmiştir. Bu yapıda, servis yönelimli mimarinin aksine sunulan hizmetlerin her birinin otonom yani kendi kaynaklarının tüketimiyle çalışması beklenmektedir. Servisler arasındaki bağımsızlıkların birbirinden ayrılmasını amaçlamaktadır. Şekil 3'te mikro servis mimarisinde istemciden geliştiriciye mikro servislerin yönetim süreci görülmektedir. Şekil 3'te geleneksel mimariden (monolitik ve katmanlı) farklı olarak mikro servis mimarisinin, bir uygulamanın belirli bir grup fonksiyonu sağlamaya adanmış hizmetlere bölündüğü görülmektedir [7]. Bu bireysel hizmetler, diğer hizmetlerden bağımsız olarak geliştirilip dağıtılabilmekte ve diğer mikro servislerin sağladığı hizmetlere hafif iletişim mekanizmaları aracılığıyla erişebilmektedir [8]. Mikro servis mimarisinin yüksek ölçeklenebilirlik, sürdürülebilirlik, pazara kısa sürede ulaşım, çok dilli geliştirme şeklinde sıralanan avantajları bulunmaktadır. Mikro servis mimarisinin sağladığı bu avantajlar gelişmesinin ve ilerlemesinin en önemli etkenleridir [9].



Şekil 3. Mikro servis mimarisinde istemciden geliştiriciye mikro servislerin yönetim süreci

Şirket içi sistemler

Şirket içi (On-Premise) sistem kavramı bulut tabanlı uygulamaların ortaya çıkması ile birlikte yeniden gündeme gelmiştir. Bu sistemlerde yazılımlar işletmenin özel fiziksel sunucularında faaliyet gösterir ve genellikle şirketin mülkiyetinde veya kiralık olarak işletilen bir veri merkezinde bulunur. Şekil 4'te Şirket içi ve bulut mimari arasındaki fark gösterilmektedir.

Şekil 4. Şirket içi ve bulut mimarisi karşılaştırması

Gin web çerçevesi

Gin web çerçevesi, Go programlama dilinde yazılmış bir çalışma alanı kütüphanesidir. Sunucu ve api gateaway oluşturma kolaylığı sağlamaktadır. go get -u github.com/gin-gonic/gin komut çalıştırıldığında Go bağımsızlıklarına Gin kütüphanesini eklenmektedir. Şekil 5'te Gin web çerçevesi kullanılarak web sunucusu oluşturan kod örneği görülmektedir.

Şekil 5 incelendiğinde ilk aşamada Github üzerinden dağıtılan Gin kütüphanesinin içeriye alındığı görülmektedir. Ana fonksiyon içerisinde varsayılan bir Gin yönlendiricisi tanımlanmaktadır. Index parametresi ana sayfayı içerisinde barındırmakta ve GET fonksiyonu ile yönlendirilmektedir. Son olarak "run" fonksiyonu çalıştırılarak sunucu başlatılmaktadır.

```go
package main

import (
        // kütüphanemizi içeri aktaralım
        "github.com/gin-gonic/gin"
)

func main() {
        //gin'in varsayılan ayarlarında bir yönlendirici oluşturalım.
        router := gin.Default()

        //anasayfayı inde fonksiyonumuz yakalayacak
        router.GET("/", index)

        //daha sonra sunucuyu başlatıyoruz
        router.Run()
}

//anasayfayı yakalayacak olan fonksiyonumuz
func index(c *gin.Context) {
        //c ile gin nesnemize bağlam oluşturduk.
        //c'yi kullanarak artık gin özelliklerine erişebiliriz.

        //sayfaya düz yazı gönderdik
        c.String(200, "Merhaba Dünya")
        //Buradaki 200 sunucudan bir cevap geldiğini anlamına gelir
}
```

Şekil 5. Gin web çerçevesi kullanılarak web sunucusu oluşturan kod örneği

Gerçekleştirim ve test

Tarihsel olarak sanallaştırma teknolojileri, süreçlerin yönetilebilir konteyner birimleri olarak programlanması ihtiyacından doğmuştur. Söz konusu işlemler ve kaynaklar dosya sistemi, bellek, ağ ve sistem bilgileridir [10]. Bulutun temel sanallaştırma yapısı olan sanal makineler, planlama, paketleme ve kaynak erişimi (güvenlik) sorunlarına dikkat ederek WtÇBTYY geliştirilmiştir.

WtÇBTYY geliştirilirken aşağıdaki teknik metot haritasının uygulanmasına karar verilmiştir:

- Kubernetes kümeleri oluşturulmuş ve konteyner-durum geliştirmeleri yapılmıştır.
- Go programlama dili kullanılarak hizmet olarak yazılım veren uygulama sağlayıcılarının API endpointleri ile iletişim kuracak program geliştirilmiştir.
- Ürünün veri tabanı kayıtları ve faturalandırma modülleri geliştirilip ara yüze dahil edilecek tasarlanmıştır.
- Ürünün ara yüzü için gerekli ön uç kütüphaneleri kurulmuş, ilgili kullanıcı etkileşimleri gerçekleştirmiştir.
- Son kullanıcı testleri gerçekleştirilmiştir ve uygulama ürün ortamına alınmıştır.

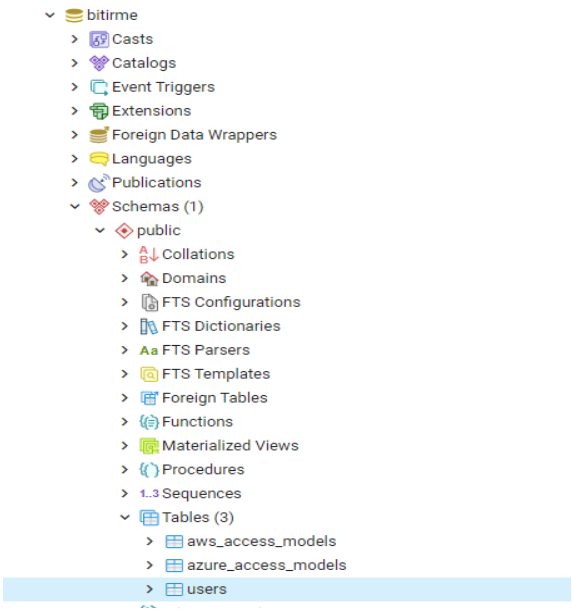
Yetkilendirme altyapısının oluşturulması

Çoklu bulut sağlayıcı ortamlar için kimlik ve erişim yönetimi (IAM) gibi yetkilendirme sistemlerinde kullanılan "Erişim belirteci" yapısı, WtÇBTYY'yedeki kullanıcılar ile ilişkilendirilmek için yazılımda kullanılmıştır. Bu belirteçler kullanıcı dostu bir ara yüz ile kullanıcının ihtiyacına göre eklenebilmekte ya da çıkarılabilmektedir. Kullanıcılar belirteçlerini kendileri isimlendirebilmektedir ancak belirteç ismi, ''Subscribtion Id'' bilgileri kullanılan ilgili bulut sağlayıcısında rastgele oluşmaktadır. Şekil 6'da Amazon web servisi IAM yapısı görülmektedir.

Şekil 6. Amazon web servisi IAM yapısı

Restful API veri tabanı ilişkisinin oluşturulması

REST ve internetin popüler hale gelmesinden önce, web uygulama hizmetleri arasında entegrasyon sağlamak oldukça zor bir süreçtir. API uygulamalar arasında iletişimi sağlayan bir araçtır. Bir istemci tarafından RESTful API üzerinden yapılan bir istekte, kaynak bir uca gönderilmektedir. Alınan veri JSON, XML gibi formatlarla HTTP protokolü üzerinden elde edilmektedir. Diğer bilgiler, meta veriler, yetkilendirmeler, URI'lar, çerezler vb. HTTP başlıkları üzerinden gönderilmektedir [11]. Şekil 7'de Postgre SQL ara yüzünde yazılımın veri tabanı görüntüsü görülmektedir.



Şekil 7. Postgre SQL ara yüzü ekran görüntüsü

İyimser kilitleme

İyimser (optimistik) kilitleme, bir veri tabanı işlemi sırasında aynı kaynağa birden fazla aynı işlemin yapılmasını engellemek için kullanılan bir kısıtlama yöntemidir. RESTful web hizmetleri, her kaynak URI'ye, karşılık gelen kaynağın mevcut durumuna dayalı olarak üretilen temsilci bir durumsal hizmet sağlayıcıdır. Bu nedenle, bağlantı kaynağı güncellendiğinde müşterilerin bir oturum başlatmasını engelleyen durumsuzluk kısıtlamasına uygun bir şekilde eşzamanlı durum değişiklikleriyle başa çıkmak önemlidir. Bu desen tarafından ele alınan sorun, stateless kısıtlamasına uygun bir şekilde eşzamanlı kaynak güncellemeleri ile başa çıkmaktır [12].

Sistem üzerinde oluşturulan müşteri bilgileri ve müşterilere ait erişim belirteci, gizli belirteç, kiracı tanımlayıcı gibi bilgilerin kullanıcı ile ilişkilendirilmiş bir şekilde oluşturulması için veri tabanına ihtiyaç duyulmaktadır. Bu bağlamda WtÇBTYY'de şekil 7 de görüldüğü üzere ilişkisel veri tabanı Postgre SQL tercih edilmiştir. Go programlama dili ile bir nesne-ilişkisel eşleme aracı olan Gorm kullanılarak ilgili modellerin veri tabanı bağlantısı sağlanmıştır. Şekil 8'de kullanıcı ara yüzünden girilecek girişlere sunucudan cevap döndürecek POST metodunun yapısı görülmektedir.

```go
package models

import "gorm.io/gorm"

type User struct {
    gorm.Model
    Username    string    `gorm:"size:255;not null;unique" json:"username,required"`
    Email       string    `gorm:"size:100;not null;unique" json:"email,required"`
    Password string `gorm:"not null" json:"password,required"`
    AwsAccessModel []AwsAccessModel `json:"aws_accesses" gorm:"foreignkey:UserID"`
    AzureAccessModel []AzureAccessModel `json:"azure_accesses" gorm:"foreignkey:UserID"`
}
```

Şekil 8. Kullanıcı ara yüzünden girilecek girişlere sunucudan cevap gönderecek POST metodunun yapısı

Şekil 8 de görüldüğü gibi Gorm kullanarak bir kullanıcı yapısı oluşturulmuştur. Gorm, Ruby'nin ActiveRecord veya Java'nın Hibernate yolunu izleyen Go için bir nesne-ilişkisel eşleme olarak tanımlanmaktadır [13].

Post ve get metotlarının oluşturulması

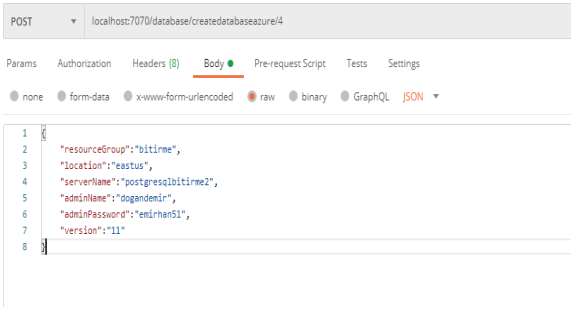
WtÇBYY için kullanılacak olan fonksiyonların POST ve GET metotlarının URL parametreleri ve kontrolcüleri oluşturulmuştur. Şekil 9'da kullanıcı ara yüzünden istek gönderilecek POST metodunun URL ve fonksiyon tanımlamaları görülmektedir.

Şekil 9. Kullanıcı ara yüzünden istek gönderilecek POST metodunun URL ve fonksiyon tanımlamaları

- Create Azure: Bu fonksiyona istek atıldığında giriş yapmış kullanıcı Azure platformunda bir sanal makine oluşturabilmektedir.
- Create Aws: Bu fonksiyona istek atıldığında giriş yapmış kullanıcı Amazon Web Service platformunda bir sanal makine oluşturabilmektedir.
- Get List Aws: Bu fonksiyona istek atıldığında giriş yapmış kullanıcı Aws platformunda oluşturduğu tüm sanal makineleri listeleyebilmektedir.
- Get List Azure: Bu fonksiyona istek atıldığında giriş yapmış kullanıcı Azure platformunda oluşturduğu tüm sanal makineleri listeleyebilmektedir

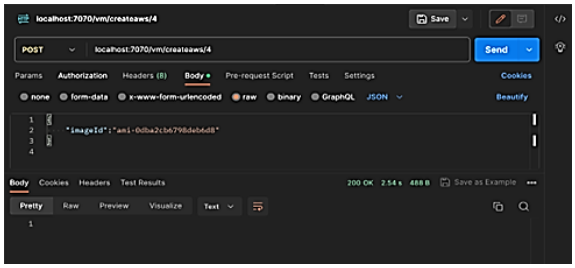
Postman kullanılarak endpointlerin testi

Postman, gerçekçi bir senaryo ile geliştirilen uygulamaları test etmeyi sağlayan bir uç-nokta test uygulamasıdır. Go programlama dilindeki yönlendirme yapısını kullanarak oluşturulan uç-nokta gruplarının çalışıp çalışmadığını test etmek üzere Şekil 10'da Postman üzerinde Json-raw formatındaki Body alanına ilgili parametreler verilmiştir.



Şekil 10. Postman uygulaması kullanarak endpointlerin testi

Şekil 11'de endpointlerin yanıt metinleri analiz edilmiştir. 200 başarılı yanıt uygulamanın doğru çalıştığını göstermektedir.



Şekil 11. Sunucudan "200 OK" dönen Create Aws fonksiyonun testi

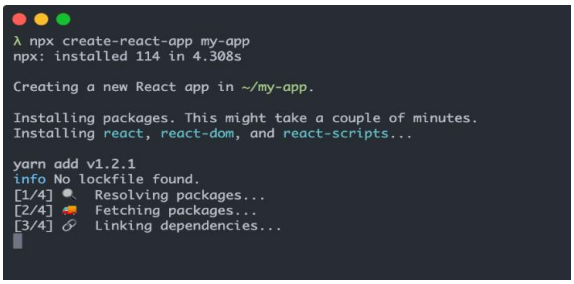
Ön-yüz için gerekli araçların kurulumu ve kullanıcı ara yüzü oluşturulması

WtÇBYY'nin ön-yüz geliştirmelerine başlamadan önce ilk olarak geliştirme yapılacak programlama dilinin, çevresel araç ve kütüphanelerinin belirlenmesi gerekmektedir. Bu araştırmada en popüler ön-yüz geliştirme araçlarından biri olan React.js kütüphanesi tercih edilmiştir. React.js kütüphanesi sunduğu bileşen fonksiyon ve sınıfsal fonksiyon yapıları ile oldukça sık kullanılan bir kütüphanedir. React.js'nin verimliliğinin temelinde sanal Belge Nesnesi Modeli (Document Object Model - DOM)'nin yenilikçi kullanımı yatmaktadır. Performansı optimize etmek için sanal DOM'u anlamak ve kullanmak çok önemlidir. Sanal DOM, gerçek DOM'un hafif bir temsili gibi davranarak React'ın veri değiştiğinde gereken güncelleme ve yeniden oluşturma sayısını en aza indirmesine olanak tanımaktadır. Bu teknik, tarayıcının iş yükünü önemli ölçüde azaltarak oluşturma sürelerinin daha hızlı olmasını sağlamaktadır [14].

React kodlarının sanal bir sunucuda gerçeklenebilmesi için ihtiyaç duyulan en temel çerçeve ise Node kütüphanesidir. İlk etapta projenin çevresel bileşenleriyle uyumlu bir şekilde çalışabilmesi için sağlam çalışan ve pek çok kütüphaneyi sorunsuz çalıştıran bir Node versiyonu belirlenmesi gerekmektedir. Node versiyonu belirlendikten sonra ilgili işletim isteminin terminalinde tercih edilen paket yükleyicisi ile Node Versiyon Yönetimi aracının kurulması gerekmektedir. Node Versiyon Yöneticisi kurulduktan sonra tercih edilen Node versiyonunun sisteme kurulup geliştirmeye hazır hale getirilmesi gerekmektedir. Terminal ekranı açıldıktan sonra ilk olarak *"curl https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh bash"* komutu çalıştırılmalıdır. Bu komut çalıştırıldıktan sonra Node Versiyon Yöneticisi kurulumu tamamlanmaktadır. Daha sonra *"nvm install #version"* komutu çalıştırılarak tercih edilen Node versiyonu kurulumu gerçekleştirilmektedir. Bu çalışmada en stabil versiyon olarak Node 16 versiyonu tercih edilmiştir.

React uygulamasının oluşturulması

Node kütüphanesi kurulduktan sonra, sistem node paket yöneticisinde bulunan tüm kütüphanelere erişebilir hale gelmektedir. Şekil 12'de bir React uygulamasının terminal ekranında oluşturulması gösterilmektedir.



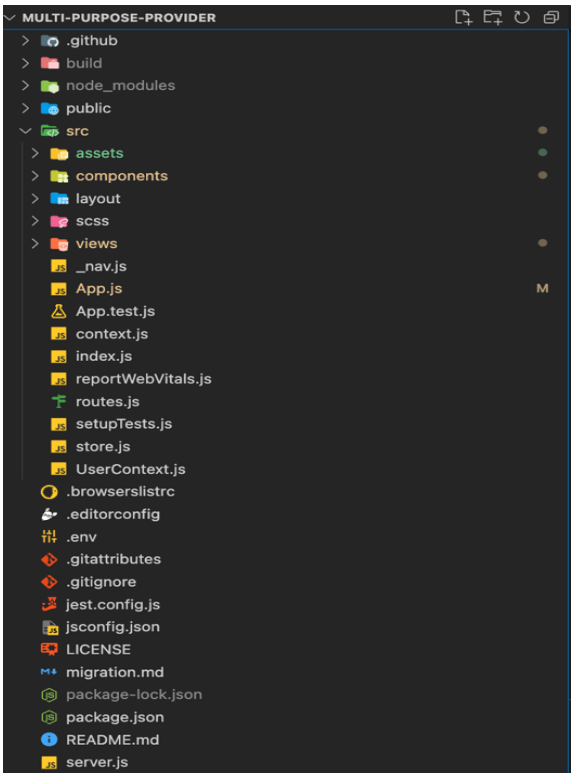
Şekil 12. Terminal ekranında React uygulaması oluşturma

"npx create-react-app #proje-adı" komutu çalıştırılarak istenilen bir dizinde React projesi oluşturulmaktadır. Ancak bunun yerine projenin ihtiyaçlarına göre boş bir node modül

şablonu oluşturarak da proje başlangıcı sağlanabilmektedir. Tüm bu aşamaları gerçekleştirildiğinde sistem ön yüz geliştirmeleri React kütüphanesi kullanarak geliştirmeye hazır hale gelmektedir. Projenin oluşturulduğu dizine giderek *"npm start"* komutuyla React uygulaması başlatılmaktadır.

React projelerinde klasör mimarisi ve standartlar

Ön-yüz geliştirme için tüm kurulumlar gerçekleştirildikten sonra React uygulaması içerisinde uygulama amaçlarına uygun bir şekilde dosya mimarisi kurulumu yapılmaktadır. Dosya mimarisi birlikte çalışan ekip için büyük önem taşımaktadır. Standartların belirlenmesi yönetimi kolaylaştırılırken ürünün uzun dönemde bakımının sağlanabilmesi için oldukça önemlidir. Şekil-13'te bu çalışmanın ön-yüz geliştirmek için tercih edilen dosya mimarisi editör ekranında gösterilmiştir.

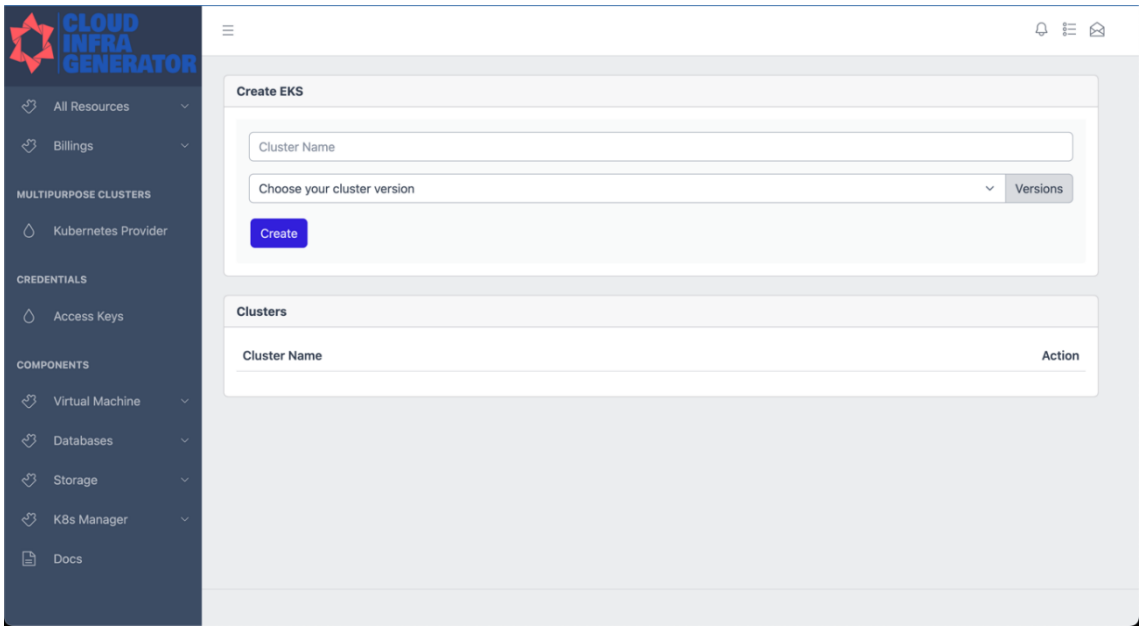


Şekil 13. React uygulamasının dosya mimarisi

Public klasöründe uygulamanın çıktısını veren tarayıcıyla etkileşime geçen index.html vb. dosyaları bulunmaktadır. Src klasöründe uygulamada da kullanılacak componentleri, contextleri, routes yapıları, fetch yapıları gibi tekrar kullanılabilir araçların kaynak kodlarının bulunduğu klasördür. Components klasöründe tekrar kullanılabilir bileşenler oluşturulmaktadır. Örneğin kullanıcıdan birden fazla yerde girdi alınacak bir girdi kutusuna ihtiyaç varsa girdi kutusu bileşeni burada oluşturulmalıdır.

Kullanıcı ara yüzü

Bu aşamaya kadar sunucuya istek gönderen fonksiyonların testi gerçekleştirilmiştir. Son adımda sunucudan dönen istekler ile kullanıcı ara yüzü tasarımı gerçekleştirilmiş ve arka plan etkileşimleri tamamlanmıştır. Şekil 14'te gösterilen yan menü, kullanıcıların sayfalar arasında gezinimini yönlendirmek amacıyla tasarlanmıştır. Bu menü, kullanıcılara "Access Keys" ekranına giderek erişim anahtarlarını içeri aktarma ve ilgili bulut barındırıcılarında sanal makineler oluşturma imkanı sunar. Ayrıca, yan menüde yer alan "K8s Manager" seçeneği üzerinden kullanıcılar, oluşturdukları sanal makineler üzerinde Kubernetes kümelerini oluşturabilir ve ilgili sanal makinelerde bu kümeleri yönetme yeteneğine sahiptir. Menü üzerinde bulunan Databases seçeneğinden ilgili bulut hizmetiyle veri tabanını oluşturmakta, Storages ekranından kaynaklarını takip edebilmektedir. Hizmet olarak yazılım veren uygulamalar kullandığın kadar öde metodunu temel almaktadır. Bu sebeple Billings ekranında kullanımla paralel olarak faturalandırma ekranları takip edilebilmektedir. All Resources ekranında sunucuya gönderilen GET isteğiyle tüm kaynaklar Gösterge Paneline takip edilebilmektedir. Şekil 14'te açılan Create EKS ekranında kullanıcı Küme adını ve sisteme entegre edilmiş küme versiyonunu seçerek Elastic Kubernetes kümesini oluşturabilmektedir. Ayrıca altındaki Clusters şablonundan açık olan tüm kümelerini görüntüleyebilmektedir.



Şekil 14. WtÇBYY kullanıcı ara yüzü ekran görüntüsü

Sonuç ve Öneriler

Bu çalışmada farklı bulut ortamlar üzerinde müşterilerin maliyet analizleri, ortam bilgilerine ilişkin veriler toplanmış olup ilgili şirket ve bireylerin birden fazla altyapı yönetim ortamında uygulamalarının koşturulmasının sağlandığı ve WtÇBYY adı verilen bir uygulama tasarlanmış ve geliştirilmiştir. Bu uygulama sayesinde sanal makine yönetimleri, veri tabanı yönetimleri ve uygulamaların çalışacağı küme ve depolama gibi bilgiler elde edilerek kullanıcıların bu ortamlarda aktif olarak işlem yapması mümkün kılınmıştır.

Sanallaştırma, fiziksel bir sunucu bilgisayarı birden fazla sanal sunucuya bölünerek kaynakların daha etkin kullanılmasına olanak sağlamaktadır. Sanal makineler, her biri kendi işletim sistemine ve uygulamalarına sahip olan izole bir ortamda çalışmaktadır [15]. Konteynerleştirme ise, uygulamaların daha az kaynak tüketerek sistem yükünü hafifleten yapılardır. İşletim sistemi düzeyinde daha performanslı ve verimli çalışmalarına olanak sağlamaktadır. Ara yüz karmaşıklığından kurtulmak basit ve kullanımı kolay bir ara yüz üzerinden işlemleri yapmak iş birimlerinde ve takımlarında öğrenme maliyetini düşürmektedir [15].

Bu çalışmada geliştirici ekipler ve son kullanıcılar için hazırlanmış bir ara yüz oluşturulmuştur. Bir doğrulama metodu ile sisteme giriş yapabilecek ekranlar gerçekleştirilmiştir. WtÇBYY'nin özellikle DevOps kültürüne önem veren kurumlar için oldukça önemli bir uygulama olduğu düşünülmektedir. Hizmet veren kurumlar müşterilerin değişen ihtiyaçlarına hızlıca yanıt verebilme kapasitesiyle doğru orantılı bir şekilde kazanım elde etmektedir. "Hizmet olarak yazılım" hizmeti verebilmek için değişken yapılara hızla uyum sağlayabilmek gerekmektedir. Bu noktada geliştirici ekiplere söz-dizimi, fonksiyon tipleri ve kullanıcı yönetim ekranları tasarlama sürecinde ekiplerde oluşan zaman maliyeti ve yönetimsel problemlere çözüm getirilmiştir.

Çalışmanın bir takım sınırlılıkları bulunmaktadır. Bulut sağlayıcıların end-point testlerindeki en büyük kısıtlardan birisi, Kubernetes kümesi kurup bulut ortamda sanal makine kurma gibi döviz olarak faturalandırılan araçların defalarca kez test edilerek ürün ortamına taşınmasıdır. Geliştirilen altyapı yönetim aracı bu çalışma özelinde yalnızca bulut ortamda Kubernetes kümeleri oluşturup sanal makineler üzerinde iş yapacak yapıları, depolayıcıları oluşturup kaynakları takip etme üzerine sınırlandırılmıştır. Öte yandan geliştirilen WtÇBYY'nin iyileştirilmesine yönelik bu alanda çalışma yapacaklara bir takım öneriler sunulabilir. Bunlar şu şekilde sıralanabilir:

•Altyapı yönetim aracı uygulamasındaki en önemli faktörlerden birisi güvenliktir. Kuvvetli bir kullanıcı doğrulama metodu kullanılabilir. Bunun için en iyi seçeneklerden biri iki faktörlü doğrulama ile kullanıcı girişi sağlanıp, kullanıcı parola, erişim anahtarı gibi veriler uuid şifreleme metodu ile veri tabanında depolanabilir.

•Kullanıcı giriş işlemleri, küme oluşturma, sanal makine oluşturma işlemleri Polygon gibi blok zincirlerinde SHA256 ve daha da üst düzey şifreleme metotlarıyla ilgili blok zinciri ağındaki dağıtık defterlerde sıfır komisyon ile kayıt tutulabilir.

•Kullanıcılar Azure, Google, AWS gibi bulut sağlayıcılarında kısıtlı işlemler yapabilmektedir. Çok daha fazla Bulut sağlayıcısı entegrasyonu yapılabilir. Böylelikle web uygulamasının ideal amacı daha yüksek oranda yakalanmış olacaktır. Kurumlar DevOps ve bulut konseptinde sunucusuz yapı modeline uyumluluk sağlarken, geliştirmeyi yapan ekipler bu çalışmayı çoklu yönetim ve no-code metodunu esas alarak geliştirmeler yapabilir. Daha fazla bulut sağlayıcı, daha fazla altyapı aracı eklenerek geliştirmeler yapılabilir.

•Bu uygulamayı kullanacak ekipler için role yönetimi ekranı ve iş birlikçi yapısı entegrasyonu sağlanmalıdır. Böylelikle

büyük bir proje yönetiminde ekibin görev dağılımı kolaylıkla yapılabilir.

•Web uygulama ara yüzünden tarayıcı eklentisiyle çalışan bildirim özelliği eklenerek kullanıcı deneyimi güçlendirilebilir

Etik kurul onayı ve çıkar çatışması beyanı

Bu çalışmada herhangi bir kişi/kurum ile çıkar çatışması bulunmamaktadır. Ayrıca çalışmanın doğası gereği etik kurul belgesi gerekmemektedir.

Yazar Katkıları

Hacer ÖZYURT, Kavramsal çerçeve, Yazılım ve mimarisel tasarım, Makale yazımı

Yusuf Can AKIER, Yazılım ve mimarisel tasarım, Uygulama geliştirme, Test, Entegrasyon

Özcan ÖZYURT, Kavramsal çerçeve, Sistem tasarımı, Makale yazımı

Kaynaklar

[1] M. Köse, and E. Küçüksille, "Bilişim altyapısı üzerine sunucusuz mimari platformu inşa etme". Mühendislik bilimleri ve tasarım dergisi, Vol. 9, no. 2, pp. 683-700, 2021. doi.org/10.21923/jesd.929649.

[2] C. Koçak, and F. Yardımcı, "Çoklu bulut servis sağlayıcıları üzerinde kubernetes platformu kullanılarak sunucusuz mimari geliştirilmesi", 4th International Scientific Research Congress (IBAD - 2019) Proceedings, April 25-26, pp.42-51, 2009, doi: 10.21733/ibad.554904

[3] K. Noyes, "Docker: A 'Shipping Container' for Linux Code",2013.[Online].Available:https://www.linux.com/news/docker-shipping-container-linux-code/

[4] P. Martin, and P. Martin, "Creating a cluster with kubeadm. kubernetes: preparing for the cka and ckad certifications", 1-9,2021.

[5] Y. Çetiner, "Bulut bilişim ve örnek bir saas uygulaması", Yüksek lisans tezi, Kırıkkale Üniversitesi, Kırıkkale, Türkiye, 2014.

[6] D. Catteddu, "Cloud computing: benefits, risks and recommendations for information security",Web Application Security: Iberic Web Application Security Conference, IBWAS 2009, Madrid, Spain, December 10-11, 2009.

[7] J. P. Gouigoux, and D. Tamzalit, "From monolith to microservices: lessons learned on an industrial migration to a web-oriented architecture", IEEE international conference on software architecture workshops (ICSAW),pp. 62-65, 2017.

[8] M. del Pilar Salas-Zárate, , G. Alor-Hernández, R. Valencia-García, , L. Rodríguez-Mazahua, A. Rodríguez-González, and J. L. L. Cuadrado, "Analyzing best practices on web development frameworks: the lift approach", Science of computer programming, Vol. 102, pp. 1-19, 2015, doi.org/10.1016/j.scico.2014.12.004.

[9] H. Knoche, and W. Hasselbring, "Drivers and barriers for microservice adoption–a survey among professionals in germany", Enterprise Modelling and Information Systems Architectures (EMISAJ)–International Journal of Conceptual Modeling: Vol. 14, Nr. 1,2019.

[10] C. Pahl, "Containerization and the paas cloud". IEEE Cloud Computing, Vol. 2, no. 3, pp. 24-31,2015.

[11] M.F.S. Lazuardy, and D. Anggraini, "Modern front end web architectures with react. js and next. Js", Research Journal of Advanced Engineering and Science, Vol. 7, no.1, pp.132-141, 2022.

[12] C. Pautasso, and E. Wilde, "RESTful web services: principles, patterns, emerging Technologies". In Proceedings of the 19th international conference on World wide web, April,2010, pp. 1359-1360.

[13] S.S. Chang, "Go web programming", Simon and Schuster,2016

[14] A. O'rinboev, "Analyzing the efficiency and performance optimization techniques of react. js ın modern web development", Инновационные исследования в современном мире: теория и практика, Vol. 2, no. 24, pp. 54-57, 2023.

[15] M. J. Scheepers, "Virtualization and containerization of application infrastructure: a comparison." In 21st twente student conference on IT, Vol. 21, pp. 1-7, 2014.