



**T.C.**  
**BİLECİK ŞEYH EDEBALI ÜNİVERSİTESİ**  
**MÜHENDİSLİK FAKÜLTESİ**  
**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**Web Tabanlı Konteyner Orkestrasyon Sistemi**

**ALEYNA ÇELİK**  
**İBRAHİM KHALİL ATTEIB YACOUB**  
**BİTİRME ÇALIŞMASI**

**DANIŞMAN**  
**Prof. Dr. Cihan KARAKUZU**  
**Dr. Öğr. Üyesi Burakhan ÇUBUKÇU**

**BİLECİK**  
**20 Haziran 2023**



**T.C.**  
**BİLECİK ŞEYH EDEBALI ÜNİVERSİTESİ**  
**MÜHENDİSLİK FAKÜLTESİ**  
**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**Web Tabanlı Konteyner Orkestrasyon Sistemi**

**ALEYNA ÇELİK**  
**İBRAHİM KHALİL ATTEİB YACOUB**  
**BİTİRME ÇALIŞMASI**

**DANIŞMAN**  
**Prof. Dr. Cihan KARAKUZU**  
**Dr. Öğr. Üyesi Burakhan ÇUBUKÇU**

**BİLECİK**  
**20 Haziran 2023**

## **BİLDİRİM**

Bu çalışmada bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

## **DECLARATION**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

**İmza**

**ALEYNA ÇELİK & İBRAHİM KHALİL ATTEIB YACOUB**

**Tarih: 20 Haziran 2023**

# ÖZET

## BİTİRME ÇALIŞMASI

### Web Tabanlı Konteyner Orkestrasyon Sistemi

ALEYNA ÇELİK

IBRAHİM KHALİL ATTEİB YACOUB

Bilecik Şeyh Edebali Üniversitesi  
Mühendislik Fakültesi  
Bilgisayar Mühendisliği Bölümü

Danışman

Prof. Dr. Cihan KARAKUZU

Dr. Öğr. Üyesi Burakhan ÇUBUKÇU

2023, 42 Sayfa

Jüri Üyeleri

İmza

.....  
.....  
.....

.....  
.....  
.....

Bu çalışma, konteyner teknolojilerini kolaylaştırarak kullanımını geliştirmeyi hedeflemektedir. Docker Engine API'sini kullanarak konteyner oluşturma, yürütme ve yönetme işlemlerini kolaylaştırmaktadır. çalışmada API'ya başarılı bir şekilde bağlantı sağlanmakta ve API çağrılarını verimli bir şekilde gerçekleştirmek için Guzzle kütüphanesi kullanılmaktadır. Ayrıca, "Theharvester" servisinin çalışmaya dahil edilmesi gibi servis entegrasyonunu da desteklemektedir. Gelecekteki geliştirmeler arasında web tabanlı bir arayüz, geliştirilmiş hata yönetimi, kapsamlı dokümantasyon ve genişletilebilirlik bulunabilir. Genel olarak, bu çalışma konteyner kullanımını basitleştirmeyi ve üretkenliği artırmayı amaçlamaktadır.

# **ABSTRACT**

## **THESIS**

### **Web Based Container Orchestration System**

**ALEYNA ÇELİK**

**IBRAHİM KHALİL ATTEİB YACOUB**

**Bilecik Şeyh Edebali University  
Engineering Faculty  
Department of Computer Engineering**

#### **Advisors**

**Prof. Dr. Cihan KARAKUZU**

**Dr. Burakhan ÇUBUKÇU**

**2023, 42 Pages**

**Jury**

**Sign**

.....  
.....  
.....

.....  
.....  
.....

This study simplifies container technologies and enhances their usability. Leveraging the Docker Engine API, it enables users to effortlessly create, execute, and manage containers. The study successfully establishes a connection with the API and utilizes the Guzzle library for efficient API calls. It also supports service integration, exemplified by the inclusion of the "Theharvester" service. Future improvements may involve a web-based interface, improved error handling, comprehensive documentation, and enhanced extensibility. Overall, this study aims to streamline container usage and boost productivity.

# ÖNSÖZ

Bitirme çalışmasında başından sonuna kadar emeği geçen ve bizi bu konuya yönlendiren saygı değer hocalarımız ve danışmanlarımız Sayın Prof. Dr. Cihan KARAKUZU , Dr. Öğr. Üyesi Burakhan ÇUBUKÇU ve Öğr. Gör. Murat ÖZALP'e tüm katkılarından ve hiç eksiltmediği desteklerinden dolayı teşekkür ederiz.

**ALEYNA ÇELİK**

**IBRAHİM KHALİL ATTEİB YACOUB**

20 Haziran 2023

# İÇİNDEKİLER

<b>ÖNSÖZ</b>	<b>v</b>
<b>ŞEKİLLER TABLOSU</b>	<b>vii</b>
<b>KODLAR LİSTESİ</b>	<b>viii</b>
<b>1 GİRİŞ</b>	<b>1</b>
<b>2 LİTERATÜR TARAMASI</b>	<b>3</b>
<b>3 KULLANILAN TEKNOLOJİLER</b>	<b>5</b>
3.1 Docker . . . . .	5
3.2 Laravel . . . . .	6
3.3 MySQL . . . . .	7
3.4 Konteyner ile İlgili Temel Bilgiler . . . . .	7
<b>4 Web Tabanlı Konteyner Orkestrasyon Sistemi</b>	<b>10</b>
4.1 Kullanıcı Arayüz . . . . .	10
4.2 Arkaplanda Çalışan Servisler (Backend) . . . . .	17
4.3 Uygulamada Servis Ekleme İşlemi . . . . .	25
<b>5 SONUÇLAR VE ÖNERİLER</b>	<b>27</b>
<b>KAYNAKLAR</b>	<b>29</b>
<b>ÖZGEÇMİŞ</b>	<b>31</b>

## ŞEKİLLER TABLOSU

Şekil 1.1	Proje Çalışma Yapısı . . . . .	2
Şekil 3.1	Veritabanı şeması . . . . .	8
Şekil 4.1	Giriş sayfası . . . . .	10
Şekil 4.2	Kontrol paneli sayfası . . . . .	11
Şekil 4.3	Ping oluşturma sayfası . . . . .	12
Şekil 4.4	Ping görev listesi . . . . .	13
Şekil 4.5	konteyner grafikleri . . . . .	14
Şekil 4.6	Ping görev detayı . . . . .	14
Şekil 4.7	Konteyner detayı . . . . .	14
Şekil 4.8	Konteyner çıktısı . . . . .	15
Şekil 4.9	Konteyner hataları . . . . .	15
Şekil 4.10	Kullanıcılar listesi . . . . .	16
Şekil 4.11	Kullanıcı kayıtları . . . . .	16
Şekil 4.12	Oturum açan kullanıcı hesabı . . . . .	17
Şekil 4.13	Ping görevi çalışma şeması . . . . .	24
Şekil 4.14	Bir görevi çalışma şeması . . . . .	25



## KODLAR LİSTESİ

1	Ping sınıfının yapıcısı . . . . .	19
2	Ping görevi servis metodu . . . . .	20
3	Ping görevi servis metodu . . . . .	22
4	Konteyner çıktısı parse işlemi . . . . .	23

# 1 GİRİŞ

Günümüzde teknolojik çalışmaların karmaşıklığı ve iş yükünün artmasıyla birlikte, verimli ve ölçeklenebilir bir altyapı tasarımı büyük önem taşımaktadır. Bu nedenle, Web Tabanlı Konteyner Orkestrasyon Sistemi, çalışma olarak ideal bir çözüm değerlendirilmiştir.

Konteyner kavramı, ilk olarak 1979 yılında ortaya atılmış ve o tarihten itibaren gelişimini sürdürmüştür. Bu süreçte Unix V7 ile ilk kez kullanılmıştır. FreeBSD Jails, Linux VServer, Oracle Solaris Containers, Open VZ, Process Containers (Google), LXC, Warden ve Lmctfy gibi gelişim süreçlerinden geçerek günümüze kadar gelmiştir.

Konteyner teknolojisinin daha yaygın hale gelmesi, 2000'li yılların başında FreeBSD Jail ile gerçekleşmiştir. Ardından Jacques Gélinas'ın VServer çalışması ile Linux ortamına dahil olmuştur. Bu temel altyapının oluşturulmasının ardından, günümüz Linux konteynerlerinin yapısı şekillenmeye başlamıştır.

Ancak, konteyner teknolojisi bu gelişmelere rağmen hala genel kullanım için yaygın değildi. 2008 yılında Docker, bu alanda devrim niteliğinde bir adım atmıştır. Google gibi büyük bilgi teknolojileri şirketlerinin kullandığı bu sistem, son kullanıcıların da hizmetine sunulmuş ve konteyner teknolojisinin hızla gelişmesine yol açmıştır. Docker, kendi adını taşıyan konteyner teknolojisiyle (dotCloud aracılığıyla) kullanıcıları tanıştırmıştır.[10]

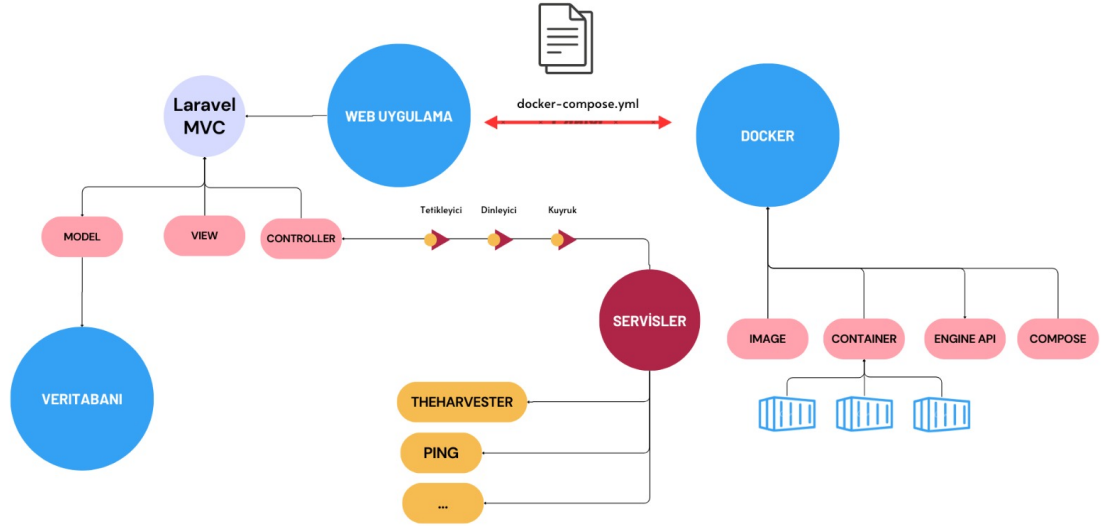
Bu çalışma için Docker'ın kullanılması uygun görülmüştür. Bunun temel nedenleri arasında hafiflik, taşınabilirlik, ölçeklenebilirlik, kolay dağıtım ve yönetim, izolasyon ve güvenlik avantajları bulunmaktadır. Bu tasarım yaklaşımı, çalışmanın gereksinimlerini etkin bir şekilde karşılamak ve iş yükünü yönetmek için ideal bir çözüm sunmaktadır.

Bu çalışmanın amacı, kullanıcıların konteyner teknolojileriyle etkileşimde bulunabilmesini sağlayan bir web arayüzü sunmak ve aynı zamanda konteynerlerin paralel bir şekilde çalışmasını sağlamaktır. Bu arayüz sayesinde kullanıcılar konteyner oluşturabilir, bu konteynerleri paralel olarak çalıştırabilir ve işlemlerin çıktılarını görüntüleyebilirler.

Çalışma sonucunda, konteyner kullanımını kolaylaştırmak amacıyla konteyner oluşturma, çalıştırma, paralel işlemler yapma ve sonuçları görme işlemleri başarıyla gerçekleştirilmiştir. Bu sayede kullanıcılar, konteyner teknolojilerini daha etkili bir şekilde kullanabilir ve işlemlerini daha verimli bir şekilde yönetebilirler.

Web arayüzü sayesinde kullanıcılar, konteynerlerin oluşturulması için gerekli parametreleri belirleyebilir, bu konteynerleri istedikleri işlemleri gerçekleştirmek için kullanabilir ve işlemlerin sonuçlarını takip edebilirler. Ayrıca, paralel işlemler sayesinde birden fazla konteyner aynı anda çalıştırılarak işlem hızı artırılabilir ve daha fazla veri işlenebilir.

Bu çalışma, konteyner teknolojilerinin kullanımını daha erişilebilir hale getirerek, kullanıcıların konteynerlerle etkileşimde bulunmasını ve paralel işlemler yapmasını sağlamaktadır. Bu da hem geliştiricilerin hem de sistem yöneticilerinin işlerini kolaylaştırabilir ve verimliliklerini artırabilir. Şekil 'de bu projenin çalışma yapısı gösterilmiştir.



Şekil 1.1: Proje Çalışma Yapısı

Çalışmayla ilgili yapılan araştırmanın bir bölümü olan Literatür Taraması, sonraki bölümde detaylı olarak açıklanmıştır.

## 2 LİTERATÜR TARAMASI

Konteynerizasyon konusunda akademik çalışmalar 2008 yıllarında başlamış, 2013 yıllarından sonra yoğunlaşmıştır. Bu çalışmada konuya yakın olan daha önce yapılan çalışmalar aşağıda özetlenmiştir.

"An Overview of Containerization Technology: Advantages, Challenges, and Future Directions" adlı çalışmada, yazarlar containerization teknolojisinin genel bir bakışını, avantajlarını, zorluklarını ve gelecekteki yönelimlerini ele almaktadır. Makale, containerization teknolojisinin nasıl çalıştığına, sanallaştırma teknolojileriyle karşılaştırılmasına ve çeşitli kullanım senaryolarına odaklanmaktadır [8].

"Docker for Multi-containers Web Application" başlıklı makalede, Sharma, Saxena ve Singh, çoklu konteyner web uygulamaları için Docker teknolojisini ele almaktadır. Makale, 2020 2. Uluslararası Sanayi Uygulamaları İçin Yenilikçi Mekanizmalar konferansında sunulmuştur. Yazarlar, Docker teknolojisinin kullanımıyla çoklu konteyner web uygulamalarının nasıl hazırlanacağını ve dağıtılacağını ele almaktadırlar. Makalede, Docker'in kullanımının avantajları ve web uygulamaları için uygunluğu da tartışılmaktadır [13].

"Containerization: A Systematic Literature Review" başlıklı çalışma, containerization teknolojisi hakkında sistematik bir literatür taraması sunmaktadır. Yazarlar, containerization teknolojisinin farklı yönlerini, kullanım alanlarını, avantajlarını ve zorluklarını analiz etmektedirler. Ayrıca, çalışmada gelecekteki araştırma yönelimleri ve containerization teknolojisinin geliştirilmesi için öneriler de sunulmaktadır [2].

"Mastering Docker" adlı kitap, Docker teknolojisi hakkında kapsamlı bir kılavuz sunmaktadır. Okuyucular, Docker'ın nasıl kullanılacağı, konteynerlerin nasıl tasarlanacağı, dağıtılacağı ve yönetileceği hakkında bilgi edinebilirler. Kitap, uygulama geliştirme ve dağıtım süreçlerinde Docker'ın nasıl kullanılabileceği konusunda pratik bilgiler sunmaktadır. Sonuç olarak, "Mastering Docker" kitabı, Docker teknolojisi hakkında kapsamlı bir kılavuz sunarak, okuyucuların konteynerleştirme ve DevOps becerilerini geliştirmelerine yardımcı olmayı amaçlamaktadır [11].

"On Enhancing the Orchestration of Multi-container Docker Applications" başlıklı makale, Docker teknolojisi ile oluşturulmuş çoklu konteynerli uygulamaların orkestrasyonunu geliştirmek için Docker Compose ile bir alternatif yaklaşım sunmaktadır. Bu çalışma, yazılım geliştiricilere ve sistem yöneticilerine fayda sağlamayı amaçlamaktadır [1].

"Docker Compose'un Çok Bileşenli Sistemleri Oluşturmak İçin Kullanımının İncelenmesi" başlıklı makale, Docker Compose'un kullanımının çoğu zaman çok bileşenli sistemlerin oluşturulması için yararlı olduğunu inceliyor. Bununla birlikte, Docker Compose dosyalarının kullanımı konusunda belirli zorluklar da bulunmaktadır. Örneğin, Docker Compose dosyalarının oluşturulması ve yönetilmesi karmaşık olabilir ve Docker Compose dosyalarının sürdürülebilirliği sorunları ortaya çıkabilir [7].

Bu literatür taraması, Docker teknolojisi ve konteynerleştirme üzerine yapılmış bazı çalışmaları özetlemektedir. Bu çalışmalar, Docker'ın çoklu konteyner uygulamaları, orkestrasyon yöntemleri ve kullanım zorlukları gibi farklı yönlerini ele almaktadır. Bu bilgiler, bu çalışmada kullanılan teknolojik seçimlerin nedenleri ve özellikleri hakkında bilgi vermektedir.

### 3 KULLANILAN TEKNOLOJİLER

Bu çalışmada kullanılan teknolojilerin tercih sebepleri ve özellikleri detaylıca aşağıda bahsedilmiştir.

#### 3.1 Docker

Docker, yazılım uygulamalarını konteynerlere paketleme ve dağıtma konusunda kullanılan bir platformdur. Bu teknoloji, uygulamaların bağımsız bir şekilde çalışabilmesini sağlar ve kurulum ve dağıtım süreçlerini kolaylaştırır. Docker, bir uygulamanın çalışması için gerekli olan tüm bağımlılıkları içeren bir konteyner oluşturmaya sağlar. Bu konteynerler, farklı ortamlarda (geliştirme, test, üretim) aynı şekilde çalışabilir, uyumluluk sorunlarını en aza indirir ve uygulamaların taşınabilirliğini artırır. Docker, yüksek verimlilik ve izolasyon sağlar, kaynakların daha etkili kullanılmasını sağlar ve sistem yönetimini kolaylaştırır [3].

Bu çalışmada kullanılan Docker özellikleri:

- **Docker compose:** Docker ortamında birden fazla konteyneri yönetmek için kullanılan bir araçtır. Docker Compose, bir YAML dosyası aracılığıyla konteynerleri tanımlamanıza ve yapılandırmanıza olanak sağlar. Bu YAML dosyasında, farklı konteynerlerin konfigürasyonları, ağ bağlantıları, bağımlılıkları ve diğer özellikleri belirtilebilir. Docker Compose, bu dosyayı okuyarak ve yorumlayarak belirtilen konteynerleri oluşturur, başlatır ve durdurur. Böylece birden fazla konteynerin aynı ortamda birlikte çalışmasını kolaylaştırır.
- **Docker Engine API:** Docker ortamını yönetmek için kullanılan bir programlama arabirimidir. Docker Engine, Docker'ın temel bileşenidir ve konteynerlerin oluşturulması, çalıştırılması ve yönetilmesi gibi işlemleri gerçekleştirir. Docker Engine API, Docker ortamının komut satırı arayüzünün (CLI) ötesine geçerek programatik olarak Docker ortamını kontrol etmenizi sağlar. Bu API, Docker ile etkileşim kurmanızı ve Docker işlemlerini otomatikleştirmenizi sağlayan bir dizi yöntem ve işlev sağlar. Docker Engine API, HTTP üzerinden erişilebilen bir RESTful API olarak sunulur ve Docker komutlarını programlamaya entegre etmenize olanak tanır [12].

## 3.2 Laravel

Laravel, PHP tabanlı bir web uygulama geliştirme framework'üdür. MVC (Model-View-Controller) tasarım desenini benimser ve geliştiricilere web uygulamaları oluşturmak için bir dizi kullanışlı özellik sunar. Laravel, güçlü bir yönlendirme sistemi, otomatik olarak oluşturulan SQL sorguları, oturum yönetimi, veritabanı migrasyonları, ön-bellekleme, form doğrulama gibi birçok bileşeni içerir. Bu bileşenler, geliştirme sürecini hızlandırır ve kod tekrarını azaltır. Laravel'in geniş bir topluluğu vardır ve bu da destek almak ve kaynaklara erişmek açısından avantaj sağlar [5].

Bu çalışmada kullanılan Laravel özelliklerine kısaca bir göz atalım:

- **Migration (migrasyon)**: Laravel, veritabanı tablolarını oluşturmak ve yönetmek için migrasyonları kullanır. Migrasyonlar, veritabanı şemalarını kod olarak temsil eder ve veritabanı yapısının kolayca değiştirilmesini ve sürdürülmesini sağlar.
- **Model**: Laravel'de model, veritabanı tablolarıyla ilişkilendirilen veri erişim katmanını temsil eder. Model sınıfları, veritabanı işlemlerini gerçekleştirmek ve verileri işlemek için kullanılır.
- **Request**: Laravel, HTTP isteklerini işlemek için request (istek) sınıflarını kullanır. Bu sınıflar, gelen istek verilerini doğrulamak, işlemek ve manipüle etmek için kullanılır.
- **Controller**: Laravel'de controller (denetleyici), HTTP isteklerini yöneten ve ilgili iş mantığını uygulayan sınıflardır. Bir controller, bir veya daha fazla işlem (action) içerir ve bu işlemler, isteklere yanıt olarak çalıştırılır.
- **Route**: Laravel, yönlendirme (routing) mekanizmasıyla istekleri doğru controller ve işlemle eşleştirir. Route dosyalarında, URL'leri belirleyebilir, istek yönlendirmelerini tanımlayabilir ve parametreleri yakalayabilirsiniz.
- **Queue**: Laravel, işleri (jobs) arkaplanda sıralı olarak çalıştırmak için kuyruk (queue) sistemini destekler. Kuyruklar, yoğun işlem yükü altında olan uygulamalarda işleri geciktirir ve daha sonra işlerin işlenmesini sağlar.
- **Auth**: Laravel'in sağladığı kimlik doğrulama (authentication) sistemi. Auth bileşeni, kullanıcı kaydı, oturum yönetimi, şifre sıfırlama gibi kullanıcı kimlik doğrulama işlemlerini kolaylaştırır.

- **Event/Listener** : Laravel’de "Event" ve "Listener" kavramları, olay tabanlı programlama yaklaşımını desteklemek için kullanılır. Bir "Event" (olay), uygulamınızda gerçekleşen belirli bir eylemi veya durumu temsil eder.
- **View** : Laravel’de "View" (görünüm), kullanıcılara sunulan HTML veya JSON gibi çıktıları oluşturmak için kullanılan şablon dosyalarını temsil eder. Görünümler, uygulama mantığını içermeyen, yalnızca kullanıcı arayüzünü temsil eden yapıları içerir.

### 3.3 MySQL

MySQL, popüler bir açık kaynaklı ilişkisel veritabanı yönetim sistemidir. MySQL, verilerin depolanması, yönetilmesi ve erişilmesi için kullanılır. Ölçeklenebilir, güvenilir ve performanslı bir veritabanı sunucusu olarak bilinir. MySQL, geniş bir kullanıcı tabanına ve gelişmiş özelliklere sahiptir. SQL (Structured Query Language) tabanlı bir veritabanı yönetim sistemi olması, veritabanı işlemlerini kolaylaştırır ve standart bir dil kullanmasını sağlar. MySQL, birçok programlama diliyle entegre edilebilir ve çeşitli platformlarda kullanılabilir.

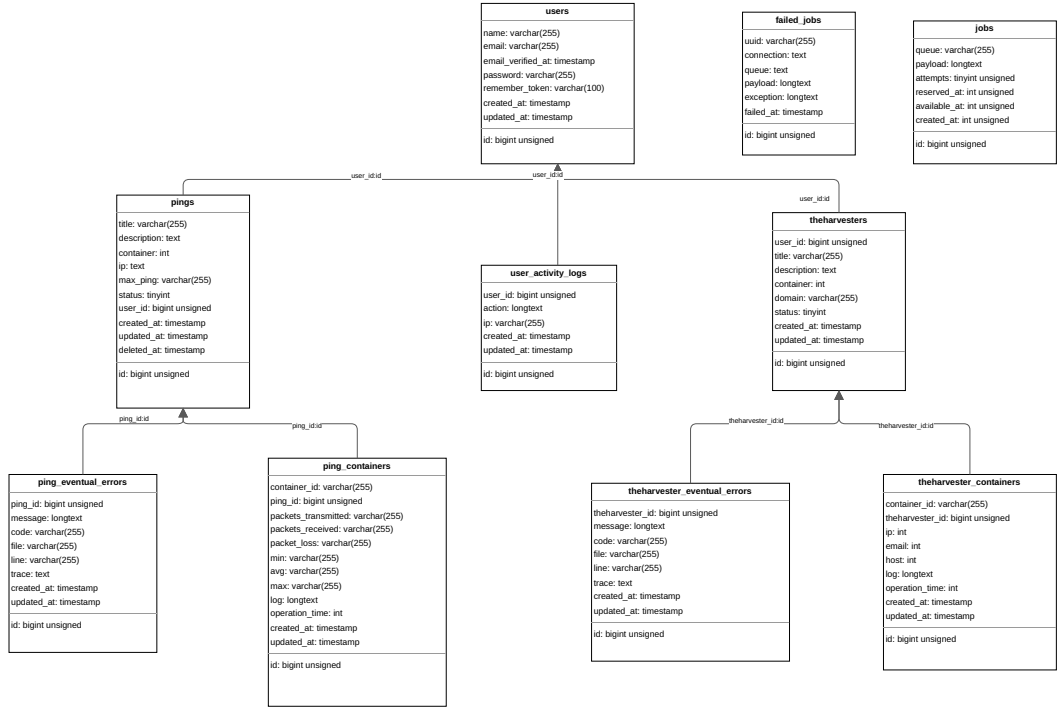
Şekil 3.1, Laravel uygulamanızın veritabanı şemasını temsil ediyor. Bu şema, veritabanı tablolarını ve bu tablolar arasındaki ilişkileri gösterir. Her tablo, Laravel migrasyonlarıyla oluşturulan bir veritabanı tablosunu temsil eder ve tablolardaki sütunları ve ilişkileri gösterir. Şekil 3.1 incelenmesi zor olabilecek büyük boyutta bir şema olduğundan, daha yüksek çözünürlüklü bir resim rapor reposunda bulunmaktadır.

Bu teknolojilerin kullanılması, konteyner tabanlı uygulama dağıtımının sağlanmasını, web uygulamasının geliştirilmesini ve veritabanı yönetimini etkin bir şekilde gerçekleştirmeyi hedeflemektedir.

### 3.4 Konteyner ile İlgili Temel Bilgiler

Konteyner, bir uygulamanın çalışması için gereken tüm bağımlılıkları ve bileşenleri bir araya getiren ve bu bileşenlerin izole edilmiş bir ortamda çalışmasını sağlayan bir yazılım paketleme ve dağıtım teknolojisidir. Konteynerler, bir uygulamanın tüm çalışma





Şekil 3.1: Veritabanı şeması

zamanı bağımlılıklarını içeren taşınabilir bir ortam sunar ve böylece uygulamaların farklı platformlarda tutarlı bir şekilde çalışmasını sağlar [6].

Konteynerlerin avantajları şunlardır:

- **Taşınabilirlik:** Konteynerler, uygulamaların bir ortamdan diğerine sorunsuz bir şekilde taşınmasını sağlar. Konteynerlerin bağımsız bir şekilde çalışabilmesi, farklı işletim sistemleri, bulut platformları veya dağıtım ortamları arasında sorun yaşamadan hareket etmelerini sağlar.
- **İzolasyon:** Konteynerler, uygulamaların birbirlerinden ve ana işletim sisteminden izole bir şekilde çalışmasını sağlar. Her konteyner, kendi dosya sistemine, ağ bağlantılarına ve kaynaklara sahiptir. Bu, uygulamaların birbirlerinin kaynaklarını etkilemeden güvenli bir şekilde çalışmasını sağlar.
- **Hızlı Dağıtım:** Konteynerler, hızlı ve tutarlı bir şekilde dağıtılabilir. Konteyner imajları, uygulamaların ve bağımlılıklarının bir araya getirildiği taşınabilir bir formattır. Bu imajlar hızlı bir şekilde oluşturulabilir, paylaşılabilir ve dağıtılabilir, böylece

uygulamaların hızlı bir şekilde çalıştırılması ve güncellenmesi mümkün olur.

- Ölçeklenebilirlik: Konteynerler, uygulamaların kolayca ölçeklendirilmesini sağlar. Konteyner tabanlı bir uygulamanın birden fazla kopyası aynı anda çalıştırılabilir ve bir yük dengeleyici kullanılarak trafiğin bu kopyalar arasında dengeli bir şekilde dağıtılması sağlanabilir. Bu, yüksek talepler altında uygulamaların performansını artırır.

Konteynerlerin dezavantajları şunlardır:

- Karmaşıklık: Konteyner teknolojileri, bazı kullanıcılar için karmaşık olabilir. Konteynerlerin oluşturulması, yönetimi ve yapılandırılması konusunda ek bilgi ve beceri gerektirebilir. Bu nedenle, konteyner teknolojilerini kullanmak isteyen kullanıcıların bu teknolojilere aşina olmaları ve gerektiğinde destek almaları önemlidir.
- Bellek ve İşlemci Kullanımı: Konteynerlerin izolasyonu sağlamak için ek sistem kaynaklarına ihtiyaçları olabilir. Konteynerler, her biri kendi işletim sistemleri gibi davranırken, her bir konteynerin bellek ve işlemci kullanımı ek yük getirebilir. Bu, sistem kaynaklarının daha dikkatli bir şekilde yönetilmesini gerektirebilir.
- Veri Yönetimi: Konteynerler, genellikle veri yönetimi konusunda bazı zorluklar sunabilir. Konteynerlerin geçici doğası ve izole edilmiş dosya sistemleri, verilerin nasıl depolanacağı, paylaşılacağı ve korunacağı konusunda bazı ek adımlar gerektirebilir. Bu, uygulama geliştiricilerinin veri yönetimi stratejilerini dikkate almalarını gerektirir.

Konteyner teknolojileri, uygulamaları hızlı bir şekilde dağıtmak, taşımak ve ölçeklendirmek için güçlü bir araçtır. Ancak, her çalışmanın ihtiyaçlarına ve altyapısına bağlı olarak avantajları ve dezavantajları dikkate almak önemlidir.

Çalışmanın ne olduğunu nasıl çalıştığı Bölüm 4'te detaylıca anlatılmıştır.

## 4 Web Tabanlı Konteyner Orkestrasyon Sistemi

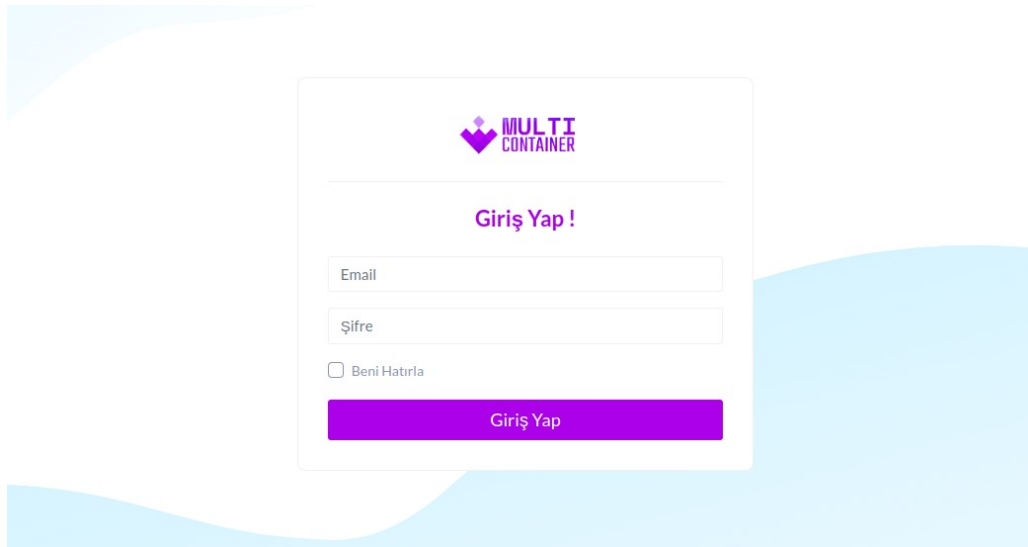
Bu çalışma, kullanıcıların konteyner teknolojilerini kullanarak konteynerlerle etkileşimde bulunmasını sağlayan bir web arayüzü sağlar. Çalışma, kullanıcıların hazırlanan bir formu doldurarak konteyner oluşturma, çalıştırma ve konteynerden gelen çıktıları grafik ve tablo şeklinde elde etme imkanı sunar.

Ayrıca, çalışmanın kullanıcı arayüzü ve arkaplan (backend) işleyişi hakkında da bilgi verilmektedir.

### 4.1 Kullanıcı Arayüz

Kullanıcı arayüzü, özel bir template seçilerek çalışmanın gereksinimlerine uygun hale getirilmiştir. Template içerisinde gereksiz kısımlar temizlenerek sağ menü bölümü oluşturulmuştur. Bu sayede kullanıcılar, sağ menü üzerinden kolayca erişebilecekleri Dashboard, görevler (Ping, Theharvester), sistem kayıtları ve Profil gibi önemli bölümlere ulaşabilmektedir. Bu düzenlemeler, arayüzün kullanıcı dostu ve çalışmaya özgü bir hale gelmesini sağlamaktadır.

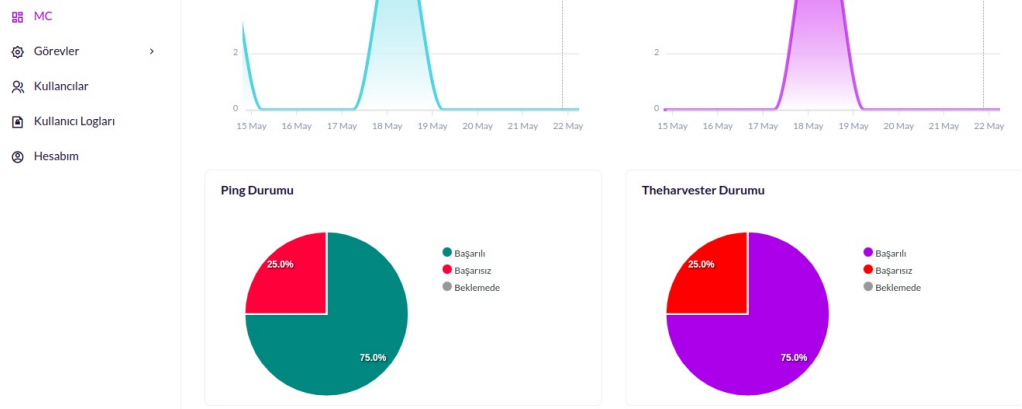
**Oturum Açma :** Kullanıcıların uygulamayı kullanabilmek için oturum açmalarını sağlar. Kullanıcı, kayıtlı e-posta adresi ve şifresini girerek oturum açma işlemini gerçekleştirir.



Şekil 4.1: Giriş sayfası

Şekil 4.1’de görüldüğü gibi oturum açma sayfası, kullanıcıya e-posta ve şifre alanlarını doldurarak oturum açma imkanı sunar. Kullanıcının girdiği e-posta adresi ve şifre, kayıtlı bilgilerle doğrulandıktan sonra oturum açma işlemi tamamlanır.

Kullanıcı oturumu başarıyla açtığı anda, **Kontrol Paneli** sayfasıyla karşılaşacaktır. Şekil 4.2’de görüldüğü gibi, bu sayfa kullanıcıya genel bir bakış sağlar ve Ping ve Theharvester konteynerlerine ilişkin istatistikleri grafik ve tablo şeklinde sunar.



Şekil 4.2: Kontrol paneli sayfası

Kontrol Paneli sayfasında kullanıcı, Ping ve Theharvester konteynerlerine ait verileri kolayca görüntüleyebilir. İlk olarak, kullanıcının sahip olduğu konteynerlerin sayısı bir grafikte gösterilir. Bu grafik, belirli bir zaman diliminde kullanıcının konteynerlerinin değişimini görsel olarak sunar.

Ardından, Ping ve Theharvester görevlerinin istatistikleri gösterilir. Kullanıcı, başarılı, başarısız ve beklemede olan görevlerin sayısını görüntüleyebilir. Bu bilgi, kullanıcının görevlerinin durumunu hızlıca anlamasına yardımcı olur.

Sayfanın alt kısmında, en son 5 görev tablo şeklinde listelenir. Bu liste, kullanıcının en son gerçekleştirdiği görevleri ve ilgili bilgileri içerir. Kullanıcılar, bu tablo üzerinden son görevlerini takip edebilir ve detaylı bilgilere erişebilir.

Kontrol Paneli sayfası, kullanıcıların konteynerlerine ilişkin genel bir bakış elde etmelerini ve önemli istatistikleri görsel ve tablo şeklinde görüntülemelerini sağlar. Bu sayede kullanıcılar, uygulamanın sağladığı konteyner orkestrasyon sisteminin performansını hız-

İlca değ erlendirebilirler.

Panelden yeni g revler oluřturmak i in **Ping Oluřturma Sayfası** kullanılır. řekil 4.3'te g sterilen form, kullanıcının Ping g revlerini oluřturmasını saęlar.

The screenshot shows a web form titled 'Yeni Ping' with a breadcrumb path 'MC / G revler / Ping / Yeni Ping'. The form contains the following fields:

- Bařlık**: A text input field.
- Tanım**: A larger text area for description.
- IP**: A text input field for the target IP address.
- Konteyner Sayısı**: A text input field with the value '1'.
- Maksimum Ping Sayısı**: A text input field with the value '1'.

A green button labeled 'Ekle' is located at the bottom right of the form.

řekil 4.3: Ping oluřturma sayfası

Formda, kullanıcından ařaęıdaki bilgileri girmesi istenir:

- G rev Bařlıęı: Oluřturulacak g revin bařlıęı.
- A ıklama: G revle ilgili detaylı a ıklama.
- Ping IP Adresi: Ping iřleminin ger ekleřtirileceęi hedef IP adresi.
- Ping Sayısı: Ka  adet ping atılacaęı.
- Konteyner Sayısı: Bu g revi ger ekleřtirecek konteynerlerin sayısı.

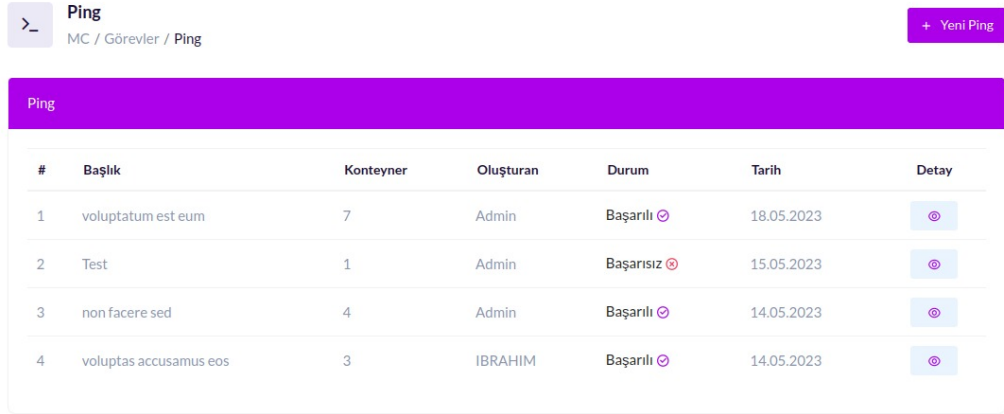
Bu bilgilerin tamamı zorunlu alanlardır ve kullanıcı tarafından doldurulması gerekmektedir. Kullanıcılar, bu formu doldurarak yeni Ping g revleri oluřturabilir ve bu g revlerin konteynerler tarafından ger ekleřtirilmesini saęlayabilir.

Ping Oluřturma Sayfası, kullanıcılara kolay ve kullanıcı dostu bir řekilde yeni g revler oluřturma imkanı sunar. Kullanıcılar, bu form  zerinden gerekli bilgileri girerek istedikleri Ping g revini tanımlayabilir ve uygulamanın konteynerler aracılıęıyla bu g revi

gerçekleştirmesini sağlayabilirler.

Oluşturulan Ping görevleri, **Ping Görev Listesi** adı verilen bir tablo şeklinde görüntülenir. Şekil4.4’de gösterilen tabloda, her görevin aşağıdaki bilgileri listelenir:

- Görev Başlığı: Oluşturulan görevin başlığı.
- konteyner: Oluşturulan konteyner sayısı.
- Oluşturan: Görevin kimin tarafından oluşturuldu
- Durumu: Görevin mevcut durumu, başlangıçta "Beklemede" olarak gösterilir.
- Görev Oluşturma Tarihi
- Detay Botunu: Görev detayı görüntülebilmektir

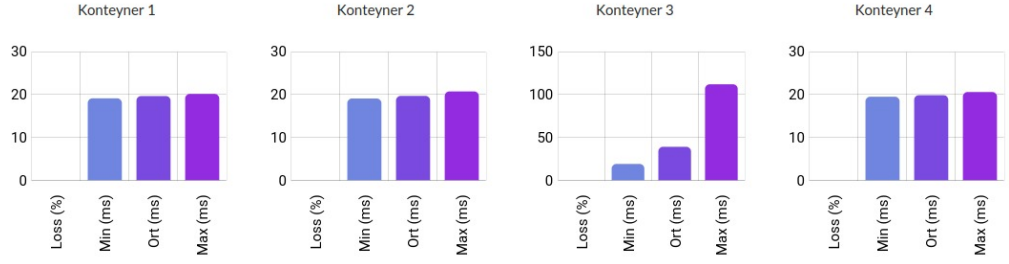


#	Başlık	Konteyner	Oluşturan	Durum	Tarih	Detay
1	voluptatum est eum	7	Admin	Başarılı	18.05.2023	
2	Test	1	Admin	Başarısız	15.05.2023	
3	non facere sed	4	Admin	Başarılı	14.05.2023	
4	voluptas accusamus eos	3	IBRAHİM	Başarılı	14.05.2023	

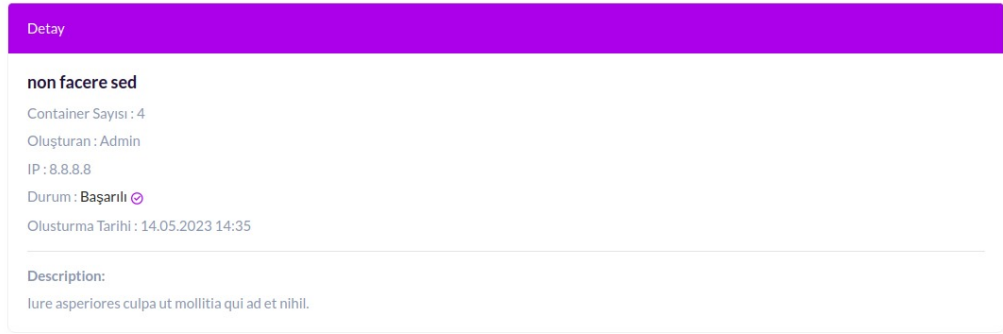
Şekil 4.4: Ping görev listesi

Görevlerin işlem durumu tamamlandığında, kullanıcılar detaylarına erişebilir. Görev detayı aşağıdaki bölümlerden oluşur:

- Konteyner Grafikleri: Görevin çalıştırıldığı konteyner sayısına göre grafikler oluşturulur. Her grafikte, minimum, ortalama, maksimum ping değerleri ve ping kaybı gösterilir. Şekil4.5’de bu grafiklerin bir örneği görülebilir.
- Görev Detayı: Görevin genel detayları, başlığı, açıklaması, atılan ping IP adresi, ping sayısı ve kullanılan konteyner sayısı gibi bilgiler içerir. Şekil4.6’de bu detayların bir örneği görülebilir.
- Konteyner Detayları ve Çıktıları: Görevin çalıştırıldığı konteynerlerin ayrıntıları ve çıktıları bu bölümde görüntülenir. Konteynerlerin detaylarını içeren bir tablo ve her



Şekil 4.5: konteyner grafikleri



Şekil 4.6: Ping görev detayı

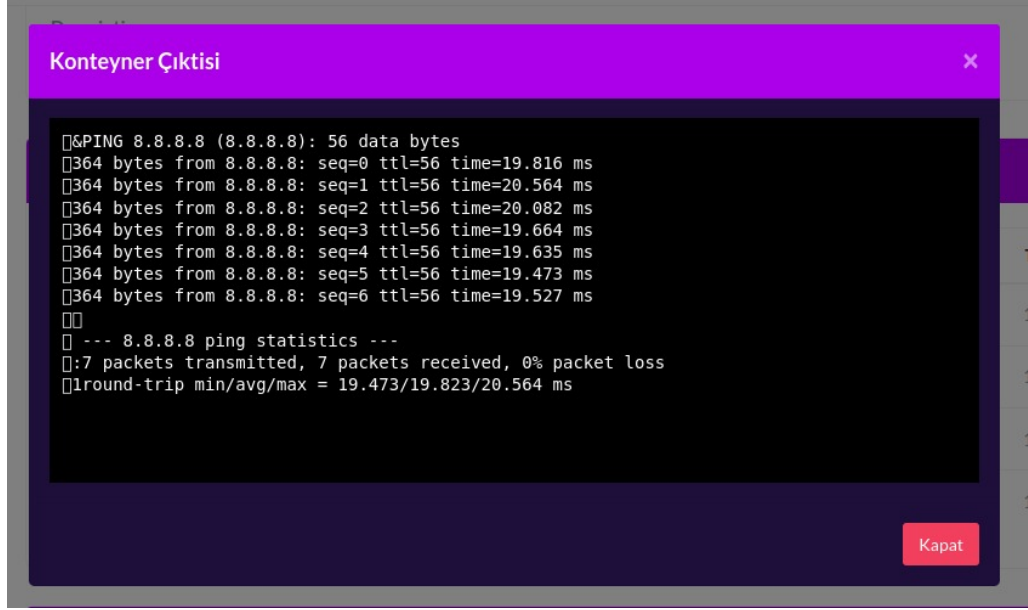
konteynerin çıktılarını gösteren ayrı bir tablo bulunur. Şekil 4.7 ve şekil 4.8’de bu detayların örneklerini gösterir.

Konteynerler										
#	Konteyner ID	Ping Sayısı	Başarılı	Kayıp	Min (Ms)	Avg (Ms)	Max (Ms)	Süre (S)	Tarih	Detay
1	1b0c453d7a7a8bcd7d2e...	7	7	0%	19.090	19.625	20.129	6	14/05/2023 14:36	<a href="#">Detay</a>
2	96f0313612f5d8e8a54e...	7	7	0%	19.054	19.665	20.703	6	14/05/2023 14:37	<a href="#">Detay</a>
3	11c18b3c284d32a73ac0...	7	7	0%	18.988	38.967	111.851	6	14/05/2023 14:37	<a href="#">Detay</a>
4	380b949a42ed930f55b9...	7	7	0%	19.473	19.823	20.564	6	14/05/2023 14:37	<a href="#">Detay</a>

Şekil 4.7: Konteyner detayı

- Görev sırasında oluşan hatalar bu tabloda listelenir. Her hata, hata mesajı ve hatanın olduğu konteyner bilgileriyle birlikte gösterilir. Şekil 4.9’de bu hata listesinin bir örneği görülebilir.

Bu görev detayları, kullanıcılara görevlerin ayrıntılı bilgilerini, konteyner performansını, çıktıları ve oluşan hataları inceleme imkanı sunar. Kullanıcılar, bu detaylar sayesinde



Şekil 4.8: Konteyner çıktısı

Hatalar					
#	Hata Kodu	Mesajı	Satır	Tarihi	Detay
1	0	cURL error 7: Failed to connect to 10.86.5.160 port 2375 after 0 ms: Connection...	210	15/05/2023 12:32	

Şekil 4.9: Konteyner hataları

görevlerin çalışma durumunu, konteynerlerin performansını ve olası sorunları kolayca analiz edebilirler.

**Kullanıcılar** bölümünde, kullanıcıların oluşturulması, görüntülenmesi, güncellenmesi ve silinmesi gibi işlemler gerçekleştirilebilir. Şekil 4.10'de, **Kullanıcılar Listesi** adı verilen bir tablo şeklinde kullanıcıların listelendiği görülmektedir.

Her kullanıcı için aşağıdaki bilgiler listelenir:

- Kullanıcı Adı ve Soyadı.
- E-posta: Kullanıcının kayıtlı e-posta adresi.
- Oluşturma Tarihi: Kullanıcının oluşturulma tarihi.
- İşlem sütununda mevcuttur.



Bu tablo, sisteme kayıtlı olan tüm kullanıcıların genel bilgilerini görüntüler. Kullanıcılar, bu tablo üzerinden kullanıcıları inceleyebilir, güncelleyebilir veya silme işlemleri gerçekleştirebilir. Bu sayede sistem yöneticileri, kullanıcılarla ilgili işlemleri kolaylıkla yapabilir ve kullanıcı verilerini yönetebilir.

Kullanıcılar			
ID	Ad Ve Soyad	Email	İşlem
1	Ibrahim	ibrahim@app.com	<a href="#">👁</a> <a href="#">✎</a> <a href="#">✖</a>
2	Aleyna	celikaleyna@gmail.com	<a href="#">👁</a> <a href="#">✎</a> <a href="#">✖</a>

Şekil 4.10: Kullanıcılar listesi

**Kullanıcı kayıtları** bölümünde, her kullanıcının yaptığı aktivitelerin bir log listesi olarak görüntülediği bir tablo bulunmaktadır. Şekil 4.11’de, Kullanıcı kayıtları tablosu örneği gösterilmektedir. Bu tabloda her log girdisi için aşağıdaki bilgiler listelenir:

- Kullanıcı: İlgili kullanıcının adı ve Soyadı.
- IP: Kullanıcının oturum açtığı IP adresi.
- İşlem: Kullanıcının gerçekleştirdiği işlem veya aktivite açıklaması.
- Tarih: İşlemin gerçekleştiği tarih.
- Saat: İşlemin gerçekleştiği saat.

 **Kullanıcı Logları**  
MC / Kullanıcı Logları

Kullanıcı Logları					
#	Kullanıcı	IP	İşlem	Tarih	Saat
1	Ibrahim	172.22.0.1	Kullanıcı logları inceledi	22/05/2023	05:56:34
2	Ibrahim	172.22.0.1	Kullanıcı Kullanıcılar listesi görüntüledi.	22/05/2023	05:56:10
3	Ibrahim	172.22.0.1	Kullanıcı Kullanıcılar listesi görüntüledi.	22/05/2023	05:56:06
4	Ibrahim	172.22.0.1	Kullanıcı Kullanıcılar listesi görüntüledi.	22/05/2023	05:55:43
5	Ibrahim	172.22.0.1	Kullanıcı Kullanıcılar listesi görüntüledi.	22/05/2023	05:55:05

Şekil 4.11: Kullanıcı kayıtları

Bu kayıtlar, kullanıcı aktivitelerinin izlenmesi, takibi ve güvenlik amaçlı olarak kullanılabilir. Sistem yöneticileri veya yetkilileri, bu kayıtları kullanarak kullanıcıların yaptığı

işlemleri inceleyebilir ve gerekirse uygun önlemleri alabilir. Ayrıca, kullanıcı kayıtları, sistemdeki kullanıcı etkinliği hakkında bilgi sahibi olmak için kullanılabilir. **Hesabım** bölümünde, oturum açmış olan kullanıcının kendi hesap bilgilerini güncelleyebileceği bir form bulunmaktadır. Şekil 4.12’de, "Oturum açan kullanıcı hesabı" formu örneği gösterilmektedir.

The screenshot displays a user interface for 'My Account' (MC / Hesabım). It features two main sections: 'Hesab Bilgilerim' (My Account Details) and 'Şifreyi Güncelle' (Update Password). The 'Hesab Bilgilerim' section includes fields for 'Ad ve Soyad' (Name and Surname) with the value 'Aleyna', 'Email' with the value 'aleyna@app.com', and a 'Güncelle' (Update) button. The 'Şifreyi Güncelle' section includes fields for 'Şimdiki Şifre' (Current Password) with masked characters, 'Yeni Şifre' (New Password), and a 'Güncelle' (Update) button. Below these sections is a 'Loglarım' (My Logs) section with a table header showing columns for '#', 'IP', 'İşlem' (Action), 'Tarih' (Date), and 'Saat' (Time).

Şekil 4.12: Oturum açan kullanıcı hesabı

Ayrıca, kullanıcının kendi kayıtları da tablo şeklinde görüntülenmektedir. Bu kayıtlar, kullanıcının kendi aktivitelerini takip etmek ve geçmiş işlemlerini gözden geçirmek için kullanılabilir.

## 4.2 Arkaplanda Çalışan Servisler (Backend)

Daha önce 3.1 bölümünde bahsettiğimiz gibi, Docker Engine PHP özelliğini kullanarak Laravel uygulamasıyla iletişim sağlayan "PingService" adında bir servis geliştirilmiştir. Bu servis sayesinde konteyner oluşturma, çalıştırma, detaylarını görüntüleme gibi işlemler gerçekleştirilebilir.

Konteyner oluşturmak için Docker Engine API’sini kullanarak POST metoduyla <https://example.com/v14/containers/create> adresine bir istek atılması gerekmektedir. İstek gövdesi (body) iki parametre içermektedir:

- Image: Hangi imajın kullanılacağı

- Cmd: Hangi komutun çalıştırılacağı

Bu parametreler, JSON formatında gönderilmektedir.

Aşağıda, bir ping görevi oluşturma örneği verilmiştir:

```
1  $response = $this->client->post('/containers/create', [  
2  'json' => [  
3  'Image' => 'alpine',  
4  'Cmd' => ['ping', '-c', $this->ping->max_ping, $this->ping->ip]  
5  ]  
6  ] );
```

Bu örnek, **alpine** imajını kullanarak **ping** komutunu belirtilen parametrelerle çalıştırmak için bir konteyner oluşturur.

“Alpine, hafif ve güvenli bir Linux dağıtımdır. Docker tarafından resmi olarak desteklenen ve sıkça kullanılan bir imajdır. Alpine, minimal bir yapıya sahiptir ve gereksiz bileşenleri içermez, bu nedenle küçük boyutlu ve hızlı çalışan konteynerler oluşturmak için tercih edilir [9].”

İşte belirli bir işlem için oluşturulan konteynerin geri döndürdüğü JSON yanıtını kullanarak konteyner kimliğini ve uyarıları elde etmek için PHP kodu:

```
1  $responseJson = '{  
2      "Id": "  
3          a468317ac3533ff0dde4857bb7e50c51953293073e9ae5975e7cb5a114b8463a  
4          ",  
5      "Warnings": []  
6  }';  
7  
8  $responseArray = json_decode($responseJson, true);  
9  $containerId = $responseArray['Id'];  
10 $warnings = $responseArray['Warnings'];
```

Bu kod, JSON yanıtını doğru bir şekilde bir PHP dizisine dönüştürür. Ardından, konteyner kimliğini (\$containerId) ve uyarıları (\$warnings) ilgili değişkenlere atar.

Daha sonra, elde edilen konteyner kimliğiyle yapılacak işlemleri gerçekleştirebilirsiniz. Örneğin:

```
1
2     $this->startContainer($containerId);
3
4     $this->stopContainer($containerId);
5
6     $this->getContainerLogs($containerId);
7
8     $this->storeContainerLogs($parsedLogs, $containerId,
        $operation_time);
```

Bu kod parçacığı, belirli bir işlemi gerçekleştirmek için konteyner kimliğini kullanılması olarak sağlar. İlgili işlemler **startContainer**, **stopContainer**, **getContainerLogs** ve **storeContainerLogs** olarak adlandırılan fonksiyonlar özel olarak yazılmıştır.

Aşağıda, Docker Engine API ile Laravel'in nasıl bağlandığını gösteren PHP kodu yer almaktadır. Bu kod, guzzle[4] kütüphanesini kullanarak Docker Engine API'ye bağlantı sağlamaktadır. 'Ping' sınıfı bağımlılığı da burada enjekte edilmektedir. ndfigure

#### Listing 1: Ping sınıfının yapısı

```
1     public function __construct(public Ping $ping)
2     {
3         $this->logger = Log::channel('single');
4
5         try {
6             $this->client = new Client([
7                 'base_uri' => config('services.docker.endpoint'),
8                 'timeout' => config('services.docker.timeout')
9             ]);
10        } catch (\Exception $e) {
11            $this->saveEventualErrors($e);
12            $this->logger->error($e->getMessage());
13        }
14    }
```

Şekil 1’deki kodda, Docker Engine API’ye bağlantı sağlamak için ‘Client’ sınıfı oluşturulmaktadır. ‘base uri’ parametresi, Docker API endpointini belirtmektedir ve ‘timeout’ parametresi, API çağrılarının zaman aşımı süresini belirlemektedir. Oluşturulan ‘Client’ nesnesi, sınıfın diğer metotlarında kullanılmak üzere ‘this->client’ değişkenine atanmaktadır.

Hata durumunda, ‘try-catch’ bloğu kullanılarak hata yakalanmakta ve ilgili işlemler gerçekleştirilmektedir. Hata durumunda, hata kaydedilip gerekli kayıt alma işlemleri yapılmaktadır.

Şekil 2’deki kodda, createPingContainer fonksiyonunu içerir. İlk olarak, IP adresinin doğruluğu kontrol edilir ve geçerli değilse bir hata fırlatılır. Daha sonra, mevcut konteynerleri siler.

Döngü kullanarak, belirli sayıda ping konteyneri oluşturulur. Her konteyner oluşturulduğunda, Docker API’ye bir POST isteği gönderilir ve konteyner oluşturulur. Oluşturulan konteyner başlatılır ve belirli bir süre (burada 10 saniye) beklenir. Ardından konteyner durdurulur.

Konteynerin çalışma süresi ve kayıtları elde edilir ve ilgili işlemler gerçekleştirilir. Son olarak, ping nesnesinin durum güncellemesi yapılır ve ilgili kayıt alma işlemleri gerçekleştirilir.

Hata durumunda, hata yakalanır, gerekli işlemler gerçekleştirilir ve hata mesajı kayıt edilir.

Bu şekilde, createPingContainer fonksiyonu, ping görevi için gerekli konteynerleri oluşturur ve ilgili işlemleri gerçekleştirir.

#### Listing 2: Ping görevi servis metodu

```
1 public function createPingContainer(): void
2 {
3     // Validate the IP address
4     if (!filter_var($this->ping->ip, FILTER_VALIDATE_IP)) {
```

```

5         throw new \InvalidArgumentException('Invalid IP address:
           ' . $this->ping->ip);
6     }
7     $this->ping->containers()->delete();
8
9     try {
10         $this->logger->info('start creating container');
11         for ($i = 0; $i < $this->ping->container; $i++) {           //
            Create the container
12             $response = $this->client->post('/containers/create'
                , [
13                 'json' => [
14                     'Image' => 'alpine',
15                     'Cmd' => ['ping', '-c', $this->ping->
                        max_ping, $this->ping->ip]
16                 ]
17             ]);
18
19             $container = json_decode((string)$response->getBody
                (), true);
20
21             // Start the container
22             $this->logger->info('Started container with ID: ' .
                $container['Id']);
23             $this->startContainer($container['Id']);
24
25
26             Sleep::for(10)->seconds();
27
28             $this->stopContainer($container['Id']);
29
30             $operation_time = $this->containerRunTime($container
                ['Id']);
31
32             $logs = $this->getContainerLogs($container['Id']);
33
34             $parsedLogs = $this->parseContainerLogs($logs);

```

```

35
36         $this->storeContainerLogs($parsedLogs, $container['
           Id'], $operation_time);
37     }
38     $this->ping->update(['status' => 1]);
39
40     $this->logger->info('container is created');
41 } catch (\Exception $e) {
42     $this->saveEventualErrors($e);
43     $this->logger->error($e->getMessage());
44 }
45 }

```

Konteynerden komut çıktısı alma işlemi şekil 3 gibi gerçekleştirilmektedir:

#### Listing 3: Ping görevi servis metodu

```

1  public function getContainerLogs(string $containerId): string
2  {
3      try {
4          // Stream the container logs (include both stdout and
           stderr)
5          $response = $this->client->get('/containers/' .
           $containerId . '/logs?stdout=1&stderr=1&follow=1');
6          $stream = $response->getBody();
7
8          // Read the logs until the stream is closed
9          $logs = '';
10         while (!$stream->eof()) {
11             $logs .= $stream->read(1024);
12         }
13
14         return $logs;
15     } catch (\Exception $e) {
16         $this->logger->error($e->getMessage());
17         return '';
18     }
19 }

```

Konteyner çıktısının parse edilme işlemi şekil 4 gibi yapılmaktadır:

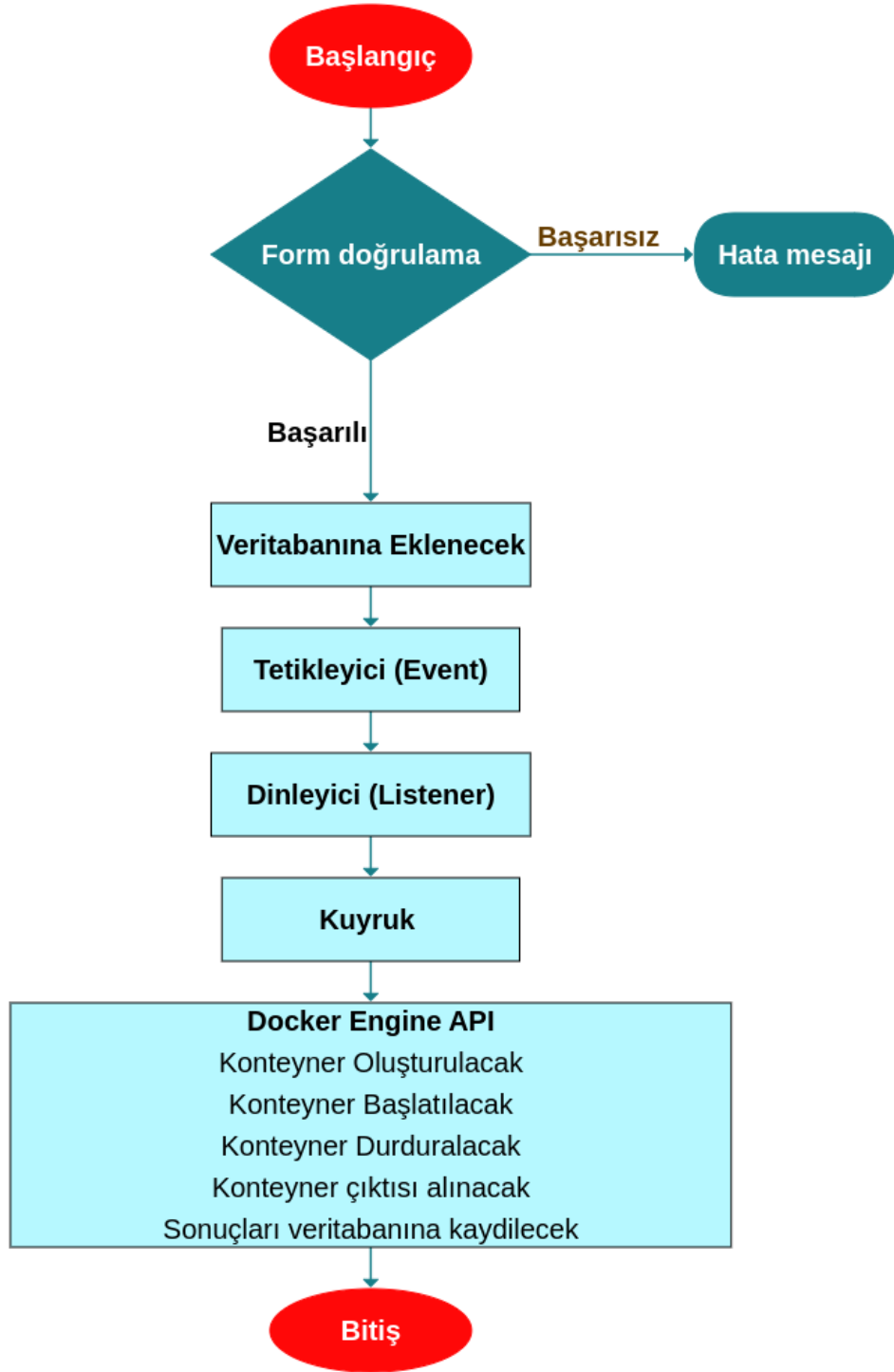
#### Listing 4: Konteyner çıktısı parse işlemi

```
1  protected function parseContainerLogs(string $log): array
2  {
3      $data = [
4          'packets_transmitted' => '',
5          'packets_received' => '',
6          'packet_loss' => '',
7          'min' => '',
8          'avg' => '',
9          'max' => '',
10         'log' => $log
11     ];
12
13     preg_match('/(\d+) packets transmitted, (\d+) packets
14         received, (\d+)% packet loss/', $log, $results);
15     if (isset($results)) {
16         $data['packet_loss'] = $results[3] ?? '';
17         $data['packets_transmitted'] = $results[1] ?? '';
18         $data['packets_received'] = $results[2] ?? '0';
19     }
20
21     preg_match('/(\d+\.\d+)\s\/(\d+\.\d+)\s\/(\d+\.\d+)\s\/', $log,
22         $matches);
23     if (isset($matches)) {
24         $data['min'] = $matches[1] ?? '0';
25         $data['avg'] = $matches[2] ?? '0';
26         $data['max'] = $matches[3] ?? '0';
27     }
28
29     return $data;
30 }
```

Bu şekilde, Docker Engine API ile Laravel arasında bağlantı sağlanmakta ve ping görevi için ilgili fonksiyonlar kullanılmaktadır.



Bir Ping görevi işlem akış diyagramı Şekil çizilmiştir.



Şekil 4.13: Ping görevi çalışma şeması

Bir servis çalışma süreci şekilde gösterilmiştir



Şekil 4.14: Bir görevi çalışma şeması

### 4.3 Uygulamada Servis Ekleme İşlemi

Bu çalışmada, varsayılan olarak Ping görevi servisi bulunmaktadır. Ekstra servisler eklemek için Laravel paketleri geliştirebilir ve bu paketleri Composer aracılığıyla çalışmaya dahil edilebilir. Örneğin, Theharvester servisi geliştirilmiş ve çalışmaya eklenmiştir.

Theharvester servisini çalışmaya dahil etmek için aşağıdaki adımlar takip edildi:

1. Terminalde çalışmanın bulunduğu dizine aşağıdaki komut kullanıldı:

```
1 $ composer require ikay/theharvester-service
```

Bu komut, Theharvester servisini çalışmaya Composer aracılığıyla ekleyecektir. Composer, bağımlılıkları otomatik olarak yönetir ve paketi çalışmaya kurar.

2. Son olarak, paketin sağladığı komutları kullanabilmek için çalışmanın yeniden yüklenmesi gerekmektedir. Terminalde aşağıdaki komut çalıştırıldı:

```
1 $ php artisan theharvester:install
```

Bu komut, Theharvester paketinin bu çalışmada gereken yapılandırmalarını yapacaktır.

Böylece Theharvester servisi çalışmayı başarıyla dahil edilmiş olacaktır. Servisi kullanmaya başlamak için paketin sağladığı komutları veya yöntemleri kullanılabilir. çalışmanın ihtiyaçlarına göre bu yöntemler özelleştirilebilir.

Not : Yukarıdaki adımlar varsayılan bir senaryo üzerinde örneklenmiştir. Paketinizin yapısına ve gereksinimlerine göre adımları uyarlamayı unutmayın.

Theharvester açık kaynak olarak geliştirilmiştir bu github linkinden ulaşabilmektedir : <https://github.com/ikhalilatteib/theharvester-service>

## 5 SONUÇLAR VE ÖNERİLER

Bu çalışma, konteyner teknolojilerinin yaygınlaşmasıyla birlikte ortaya çıkan karmaşıklığı azaltmak amacıyla geliştirilmiştir. Konteyner kullanımını kolaylaştırmayı hedefleyen bu çalışmada, konteyner oluşturma, çalıştırma ve sonuçları başarıyla gerçekleştirilmektedir.

Çalışmanın başarılı sonuçları aşağıdaki şekillerde değerlendirilebilir:

- **Konteyner İşlemleri:** Çalışmada, Docker Engine API ile entegre olarak konteyner oluşturma, çalıştırma, durdurma ve silme gibi işlemleri başarılı bir şekilde gerçekleştirmektedir. Bu sayede kullanıcılar, Docker konteynerlerini kolaylıkla yönetebilmektedir.
- **Çalışma Performansı:** Çalışmanın uygulama performansı, Guzzle kütüphanesinin etkili kullanımı ve Docker Engine API'nin sağladığı hızlı yanıtlar sayesinde yüksek seviyede tutulmaktadır. Bu da kullanıcı deneyimini olumlu yönde etkilemektedir.
- **Theharvester Servisi:** Çalışmada, servislerin eklenmesini desteklemektedir. Örnek olarak Theharvester servisi geliştirilmiş ve başarıyla çalışmaya entegre edilmiştir. Bu sayede kullanıcılar, çalışmalarına farklı servisleri ekleyerek genişletme imkânına sahip olmaktadır.

### Öneriler

- **Kullanıcı Arayüzü Geliştirme:** Gelecekte, web tabanlı bir kullanıcı arayüzü oluşturulabilir. Bu arayüz sayesinde kullanıcılar, konteyner oluşturma, silme, güncelleme ve sonuçları inceleme gibi işlemleri görsel bir şekilde gerçekleştirebilir. Bu, kullanım kolaylığı sağlayacak ve çalışmanın kullanıcı tabanını genişletecektir.
- **Hata Yönetimi ve Güncelleme:** Çalışmada, hataların ve istisnaların yönetimini etkin bir şekilde gerçekleştirmektedir. Gelecekte, kullanıcı geri bildirimleri ve testlerle çalışmanın daha da güçlendirilmesi ve hata sayısının en aza indirgenmesi sağlanabilir. Ayrıca, Docker Engine API'nin güncellemelerini takip etmek ve çalışmayı güncel tutmak da önemlidir.

- Dokümantasyon ve Destek: Çalışmaya ilişkin kapsamlı bir dokümantasyon oluşturmak, kullanıcıların çalışmayı daha iyi anlamalarını ve kullanmalarını sağlayacaktır. Ayrıca, kullanıcıların karşılaştıkları sorunlar için destek sağlamak da önemlidir. Sorunların çözümüne yönelik bir destek kanalı veya topluluk oluşturmak, kullanıcı memnuniyetini artıracaktır.
- Genişletilebilirlik: Çalışmaya ekstra servisler eklemek için Laravel paketlerinin geliştirilmesi teşvik edilmelidir. Geliştiricilere açık API ve belgeleme sağlanarak, yeni servislerin kolayca entegre edilebilmesi ve çalışmanın kullanım alanının genişlemesi sağlanabilir.

Sonuç olarak, bu çalışma konteyner teknolojilerinin karmaşıklığını azaltarak kullanım kolaylığı sağlamak amacıyla geliştirilmiştir. Başarılı sonuçları ve önerilerle birlikte, çalışmanın gelecekteki gelişim potansiyeli ve kullanıcı memnuniyetini artırma imkanı vardır.

Bu rapor reposunda, readme.md dosyasında aşağıdaki maddelerin linklerine ulaşabilirsiniz:

- Projenin Github deposu
- Theharvester servisinin github deposu
- Veritabanı şeması
- Windows ortamında Docker'ın çalıştırılması

Bu raporun readme linki:

<https://github.com/ikhalilatteib/mutli-container-project/blob/master/readme.md>

## KAYNAKLAR

- [1] Antonio Brogi, Claus Pahl ve Jacopo Soldani. “On enhancing the orchestration of multi-container Docker applications”. İçinde: *Advances in Service-Oriented and Cloud Computing: Workshops of ES OCC 2018, Como, Italy, September 12–14, 2018, Revised Selected Papers*. C. 7. Springer International Publishing, 2020, ss. 21–33.
- [2] Jennifer Davis David Brown. “Containerization”. İçinde: *Journal of Software Engineering Research ve Development*. 2020.
- [3] Docker. *Docker overview*. Erişim tarihi: 12 Mart 2023. 2023. URL: <https://docs.docker.com/get-started/overview/>.
- [4] Guzzle Documentation. *Guzzle, HTTP istemcisi olarak kullanılan bir PHP kütüphanesidir ve Docker Engine API ile iletişim kurmak için kullanılmıştır*. Erişim tarihi: 12 Nisan 2023. 2023. URL: <http://docs.guzzlephp.org/en/stable/>.
- [5] Laravel Documentation. *Laravel hakkında ayrıntılı bilgi ve Laravel projeleri geliştirmek için rehberlik sunar*. Erişim tarihi: 14 Mart 2023. 2023. URL: <https://laravel.com/docs>.
- [6] J. Domenici Bravo. “Docker orchestration: Use containerization to deploy and manage applications at scale”. İçinde: Packt Publishing Ltd. 2020.
- [7] Mohamed H. Ibrahim, Mohamed Sayagh ve Ahmed E. Hassan. “A study of how Docker Compose is used to compose multi-component systems”. İçinde: *Empirical Software Engineering* 26 (2021), ss. 1–27.
- [8] Emily Johnson John Smith. “An Overview of Containerization Technology”. İçinde: *International Journal of Computer Science ve Applications*. 2019.
- [9] Alpine Linux. *About Alpine*. Erişim tarihi: 17 Mayıs 2023. 2022. URL: <https://www.alpinelinux.org/about/>.
- [10] Global Tech Magazine. *Konteyner (container) Teknolojisi Nedir*. Erişim tarihi: 03 Mart 2023. 2021. URL: <https://www.globaltechmagazine.com/2021/04/13/konteyner-container-teknolojisi-nedir/>.
- [11] Russ McKendrick. *Mastering Docker: Enhance your containerization and DevOps skills to deliver production-ready applications*. Packt Publishing Ltd, 2020.

- [12] Docker API Reference. *Docker Engine API hakkında resmi belgelendirme ve API referansı*. Erişim tarihi: 23 Mart 2023. 2023. URL: <https://docs.docker.com/engine/api/>.
- [13] V. Sharma, H. K. Saxena ve A. K. Singh. “Docker for multi-containers web application”. İçinde: *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*. IEEE. Mar. 2020, ss. 589–592.

## ÖZGEÇMİŞ

### KİŞİSEL BELGELER

**Adı Soyadı** : ALEYNA ÇELİK  
**Uyruğu** : T.C.  
**Doğum Yeri ve Tarihi**: 29.09.2000  
**Adres** : Etimesgut/Eryaman/Ankara  
**Telefon** : 5511045713  
**E-mail** : celikaleyna71@gmail.com

### EĞİTİM DURUMU

**Lisans Öğrenimi**: BŞEÜ Bilgisayar Mühendisliği Bölümü  
**Bitirme Yılı** : 2023  
**Lise** : Batıkent Anadolu Lisesi-Ankara

### İŞ DENEYİMLERİ

**Yıl** : Şubat 2023 - Devam  
**Kurum**: Bilecik Şeyh Edebali Üniversitesi Bilgi İşlem Daire Başkanlığı  
**Stajlar** : Staj I ve Staj II tamamlandı



### İLGİ ALANLARI

Siber Güvenlik, Ağ ve Sistem, Network Yapılandırması

### YABANCI DİLLER

İngilizce: B1

### BELİRTMEK İSTEDİĞİNİZ DİĞER ÖZELLİKLER

 <https://github.com/Aleyna06>  
 <https://www.linkedin.com/in/aleyna-çelik/>



## **ÖZGEÇMİŞ**

### **KİŞİSEL BELGELER**

**Adı Soyadı** : IBRAHİM KHALİL ATTEİB YACOUB  
**Uyruğu** : ÇAD  
**Doğum Yeri ve Tarihi:** 03.03.1998  
**Adres** : Cumhuriyet Mah. Atatürk Bul. No:70 D:13 Bilecik/Merkez Türkiye  
**Telefon** : 5433044170  
**E-mail** : ibrahimalkhalilatteib@gmail.com

### **EĞİTİM DURUMU**

**Lisans Öğrenimi:** BŞEÜ Bilgisayar Mühendisliği Bölümü  
**Bitirme Yılı** : 2023  
**Lise** : Kuveyt Merkez Lisesi - Encemine/ÇAD

### **İŞ DENEYİMLERİ**

**Yıl** : Ağustos 2022 - Devam  
**Kurum:** Şef Bilişim Hizmetleri A.Ş. (Plus Clouds) / Yazılım Departmanı  
**Stajlar** : Staj I tamamlandı

### **İLGİ ALANLARI**


Devops, Web Uygulama, Girişimcilik

### **YABANCI DİLLER**

Fransızca: C2  
Arapça : C1  
İngilizce : A2

## **BELİRTMEK İSTEDİĞİNİZ DİĞER ÖZELLİKLER**

 <https://github.com/ikhalilatteib>

 <https://www.linkedin.com/in/ikhalilatteib>