



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

Docker ile çoklu konteynerlar kullanarak paralel iş yapan sistem tasarımı

Aleyna ÇELİK
IBRAHİM KHALİL ATTEİB YACOUB
BİTİRME ÇALIŞMASI

DANIŞMANI : Prof. Dr. Cihan KARAKUZU
Dr. Öğr. Üyesi Burakhan ÇUBUKÇU

BİLECİK
16 Mayıs 2023



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

Docker ile çoklu konteynerlar kullanarak paralel iş yapan sistem tasarımı

Aleyna ÇELİK
Ibrahim Khalil Atteib YACOUB

BİTİRME ÇALIŞMASI

DANIŞMANI : Prof. Dr. Cihan KARAKUZU
Dr. Öğr. Üyesi Burakhan ÇUBUKÇU

BİLECİK
16 Mayıs 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

Ibrahim Khalil Atteib

YACOUB

Tarih: 16 Mayıs 2023

ÖZET

BİTİRME ÇALIŞMASI

Docker ile çoklu konteynerlar kullanarak paralel iş yapan sistem tasarımı

Aleyna ÇELİK

Ibrahim Khalil Atteib YACOUB

Bilecik Şeyh Edebali Üniversitesi

Mühendislik Fakültesi

Bilgisayar Mühendisliği Bölümü

Danışman: Prof. Dr. Cihan KARAKUZU

Dr. Öğr. Üyesi Burakhan ÇUBUKÇU

2023, 34 Sayfa

Jüri Üyeleri

İmza

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

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

Docker ile çoklu konteynerlar kullanarak paralel işlem yapan sistemler, uygulama ve servislerin farklı konteynerlar içinde izole edilerek çalıştırılmasını sağlayan bir sistem tasarımıdır. Bu sayede her bir konteyner, farklı özellikler ve işlevler için optimize edilebilir ve uygulama ölçeklenebilir hale gelir. Ayrıca, verilen işi belirtilen sayıda konteyner ile bölerek daha esnek hale getirmek amaçlanmıştır. Arayüz, canlı olarak ekranda ilerleme bilgileri göstererek, kullanıcılara toplam iş miktarı ile tamamlanma yüzdesi ve hangi konteynerların en hızlı olduğu gibi önemli bilgileri verir. Bu arayüz, güvenlik testleri veya izinsiz giriş girişimi gibi meşru olmayan amaçlarla kullanılmamalıdır.

ABSTRACT

THESIS

System design using Docker with multiple containers for parallel processing

Aleyna ÇELİK

Ibrahim Khalil Atteib YACOUB

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

Advisor : Prof. Dr. Cihan KARAKUZU

Assoc. Prof. Dr. Burakhan ÇUBUKÇU

2023, 34 Pages

Jury

Sign

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

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

Using multiple containers with Docker to perform parallel processing is a system design that allows applications and services to be run in different containers, isolated from each other. This allows each container to be optimized for different features and functions, making the application scalable. Additionally, the system is designed to make the given task more flexible by dividing it into a specified number of containers. The interface displays progress information live on the screen, providing users with important information such as the total amount of work completed, the completion percentage, and which containers are the fastest. However, this interface should not be used for illegitimate purposes such as security testing or unauthorized access attempts.

ÖNSÖZ

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

Aleyna ÇELİK

Ibrahim Khalil Attuib YACOUB

16 Mayıs 2023

İÇİNDEKİLER

ÖNSÖZ	v
ŞEKİLLER TABLOSU	viii
1 GİRİŞ	1
2 Literatür Taraması	4
2.1 Docker ve Mikro Hizmetler:	4
2.2 Docker ve Bulut Bilişim:	5
2.3 Docker ve Güvenlik:	6
3 PROJE TANITIMI	8
3.1 Giriş	8
3.2 Yapı ve Teknolojiler:	8
3.3 Güvenlik Önlemleri:	8
3.4 Docker ile Konteynerleme:	8
3.5 İlerleme Takibi ve Bildirim Sistemi:	8
3.6 Sonuç:	9
4 KULLANILAN YAZILIMLAR VE YÖNTEMLER	10
4.1 Neden Tercih Ettik ?	10
4.2 Web Uygulama Geliştirmek İçin	11
4.2.1 Laravel	11
4.2.2 Kullanım Kolaylığı:	11
4.2.3 MVC Mimari ve Yapılandırma:	12
4.3 Konteynerize Etmek İçin	12
4.3.1 Docker	12
4.4 Veri Tabanı İçin	16
4.4.1 MYSQL	16
5 PROJE’NİN İŞLEYİŞİ	18
5.1 Arayüz Kısmı	18

5.2	Arka plan işleyişi	18
5.2.1	Docker Engine API	19
5.2.2	Laravel Özellikleri	19
5.3	Veritabanı kısmı İçin	20
6	SONUÇLAR VE ÖNERİLER	22
7	EKLER	23
	KAYNAKLAR	24
	ÖZGEÇMİŞ	25

ŞEKİLLER TABLOSU

Şekil 4.1	Docker vs Sanal Makina	15
Şekil 5.1	Sql ping tabloları	21

1 GİRİŞ

Paralel işlem, modern bilgisayar sistemlerinde performansı artırmak için yaygın olarak kullanılan bir tekniktir. Ancak, bu teknik, karmaşık paralel işlem yapısı tasarımı ve uygulanması zor ve zaman alıcı olabilir. Bu sorunları çözmek için Docker, çoklu konteynerlar kullanarak paralel işlem yapmak için ideal bir çözüm sunar. Docker, 2013 yılında dotCloud adlı bir Platform as a Service (PaaS) sağlayıcısı tarafından geliştirilmiştir ve kurucusu ve CTO'su Solomon Hykes olarak bilinmektedir.

Docker, uygulamaların ve hizmetlerin paketlenmesi, dağıtımı ve çalıştırılmasını kolaylaştıran bir yazılım platformudur. Bu teknoloji, farklı konteynerlar arasında görevleri bölüştürerek, daha hızlı ve verimli bir şekilde çalışan bir sistem oluşturmak için kullanılabilir. Çoklu konteynerlar, birbirleriyle iletişim halinde olan ve her biri farklı bir görevi yerine getiren bir dizi konteynerden oluşan bir yapıdır. Bu yapı, görevler arasında paralelleştirme yapılarak daha hızlı ve verimli bir sistem oluşturulabilir. Konteynerler arasındaki iletişim, Docker Swarm veya Kubernetes gibi konteyner orkestrasyon araçları kullanılarak kolayca yönetilebilir. Bu sayede, tasarım ölçeklenebilir hale getirilebilir.

Bu çalışmada, Laravel PHP kullanılarak bir örnek üzerinde Docker ile çoklu konteynerlar kullanarak paralel işlem yapmanın nasıl oluşturulacağına ve yönetileceğine dair detaylı bilgiler verilecektir. Örneğin veritabanı olarak MySQL kullanılacak. Docker sayesinde, örneğin birden fazla sunucuda birden fazla örneği çalıştırmak mümkün olacak. Böylece, örneğin daha yüksek trafik hacimlerine veya daha karmaşık iş yüklerine ihtiyaç duyulduğunda sistem ölçeklendirilebilir hale getirilebilir.

Docker ayrıca, geliştirme sürecini de hızlandırabilir. Örneğin, farklı uygulama ve hizmetleri ayrı ayrı test etmek yerine, Docker sayesinde bunları farklı konteynerlar içinde birleştirmek ve birlikte test etmek mümkün olacak. Bu sayede, uygulama ve hizmetlerin bütünlüğü ve performansı daha iyi bir şekilde test edilebilir. Ayrıca, Docker sayesinde, uygulama ve hizmetlerin herhangi bir sistemde kolayca çalıştırılması ve dağıtılması da mümkün hale gelir.

Sonuç olarak, Docker çoklu konteynerlar kullanarak paralel işlem yapmak için ideal

bir züm sunan bir teknolojidir. Bu sayede, karmaşık paralel işlem yapısı tasarımı ve uygulanması daha kolay ve zaman açısından daha verimli hale gelir.

Docker'ın çoklu konteynerlar kullanarak paralel işlem yapmaya sağladığı avantajlardan biri, iş yükünü parçalara bölerek her bir parçayı ayrı bir konteyner içinde çalıştırabilme esnekliğidir. Bu, işlemleri paralel olarak gerçekleştirerek performansı artırır. Örneğin, bir web uygulaması üzerinde çalışırken, kullanıcı taleplerini işleyen bir konteyner, veritabanı işlemlerini yapan başka bir konteyner ve resim işleme gibi farklı bir görevi üstlenen bir başka konteyner olabilir. Bu konteynerler, bağımsız olarak çalışabilir ve birbirleriyle iletişim kurabilirler. Bu şekilde, her bir konteyner kendi kaynaklarını etkin bir şekilde kullanarak iş yükünü hızlı ve verimli bir şekilde işleyebilir.

Docker ayrıca, ölçeklendirme kolaylığı sağlar. Yüksek talep veya büyüyen iş yükleri durumunda, Docker Swarm veya Kubernetes gibi konteyner orkestrasyon araçları kullanarak yeni konteyner örnekleri oluşturabilir ve bu konteynerleri mevcut konteyner gruplarına ekleyebilirsiniz. Bu, daha fazla kaynak sağlayarak sistemi ölçeklendirmenize ve iş yükünü dengelemenize yardımcı olur. Aynı şekilde, talep düştüğünde veya iş yükü azaldığında, gereksiz konteyner örneklerini kolayca kaldırabilirsiniz. Bu dinamik ölçeklendirme yeteneği, kaynak kullanımını optimize ederek performansı artırır ve maliyetleri düşürür.

Docker ayrıca, geliştirme sürecini kolaylaştırır ve hızlandırır. Konteyner tabanlı bir geliştirme ortamında, her bir bileşen veya hizmeti ayrı bir konteynerde çalıştırarak, geliştiricilerin uygulama ve hizmetlerin birlikte çalışmasını ve entegrasyonunu daha iyi test etmelerini sağlar. Ayrıca, her bir konteyneri bağımsız olarak güncelleyebilir veya değiştirebilirsiniz, bu da hızlı ve sorunsuz bir şekilde yeni özellikler eklemenize veya hataları düzeltmenize olanak tanır.

Sonuç olarak, Docker'ın çoklu konteynerlar kullanarak paralel işlem yapmaya yönelik sunmuş olduğu çözüm, performansı artırır, ölçeklenebilirlik sağlar, geliştirme sürecini hızlandırır ve sistem yönetimini kolaylaştırır. Docker, paralel işlem ihtiyaç duyan projeler için ideal bir çözümdür. Özellikle büyük ölçekli uygulamalar, mikro hizmet mimarileri veya veri yoğun iş yükleriyle çalışan sistemler Docker'ın sağladığı çoklu kontey-

nerlar ile paralel işlem yapma yeteneklerinden büyük ölçüde faydalanabilir.

Docker'ın çoklu konteynerlar kullanarak paralel işlem yapmanın yanı sıra birçok avantajı vardır. Bunlardan biri, uygulama ve hizmetlerin bağımsız ve izole bir şekilde çalıştırılabilmesidir. Her bir konteyner, kendi ortamını ve bağımsız olarak çalışan bileşenlerini içerir. Bu, uygulama ve hizmetlerin birbirinden etkilenmeden güvenli bir şekilde çalışmasını sağlar.

Ayrıca, Docker'ın konteyner tabanlı mimarisi, uygulama ve hizmetlerin farklı ortamlarda sorunsuz bir şekilde çalışmasını sağlar. Bir konteyneri bir sistemden diğerine taşımak veya farklı bir ortamda çalıştırmak oldukça kolaydır. Bu da geliştirme sürecini hızlandırır ve dağıtım süreçlerini kolaylaştırır.

Docker ayrıca, kaynakların etkin bir şekilde kullanılmasını sağlar. Her bir konteyner, sadece ihtiyaç duyduğu kaynakları kullanır ve izole edilmiş olduğu için diğer konteynerlere etki etmez. Bu sayede, sistemdeki kaynakların verimli bir şekilde dağıtılması ve kullanılması sağlanır.

Son olarak, Docker ekosistemi geniş bir topluluk tarafından desteklenmektedir. Docker Hub gibi bir merkezi depo, hazır konteyner görüntülerini paylaşmanıza ve kullanmanıza olanak tanır. Ayrıca, Docker, zengin bir API ve komut satırı arayüzü ile birlikte gelir, bu da otomasyon ve yönetim süreçlerini kolaylaştırır.

Docker'ın çoklu konteynerlar kullanarak paralel işlem yapma yetenekleri, modern bilgisayar sistemlerinde performansı artırmak ve verimliliği sağlamak için güçlü bir araçtır. Karmaşık iş yüklerini parçalara ayırarak, her bir parçayı ayrı bir konteynerde çalıştırarak ve konteynerler arasında iletişimi kolayca yöneterek, daha hızlı, ölçeklenebilir ve etkili bir sistem oluşturmak mümkündür. Bu nedenle, Docker, paralel işlem yapmak isteyen geliştiriciler ve sistem yöneticileri için önemli bir teknoloji ve çözüm sunmaktadır.

2 Literatür Taraması

Docker, konteynerleştirme teknolojisi olarak bilinen bir platformdur ve yazılım uygulamalarının hızlı bir şekilde paketlenmesi, taşınması ve dağıtılmasını sağlar. Docker, uygulamaları bağımsız ve taşınabilir bir şekilde çalıştıran hafif konteynerler oluşturmak için bir dizi araç ve API'ler sunar. Bu literatür taraması, Docker'ın kullanımıyla ilgili güncel çalışmaları incelemektedir.

2.1 Docker ve Mikro Hizmetler:

- **"Microservices Deployment with Docker: Challenges and Solutions" (Docker ile Mikro Hizmetler Dağıtımı: Zorluklar ve Çözümler):** Bu çalışma, Docker kullanarak mikro hizmetlerin dağıtımıyla ilgili karşılaşılan zorlukları ve bu zorluklara yönelik çözümleri inceler. Mikro hizmet mimarisinde, bir uygulama birden fazla küçük hizmete (mikro hizmetlere) bölünür ve her bir hizmet kendi konteynerinde çalışır. Docker, mikro hizmetlerin bağımsız olarak paketlenmesini ve dağıtılmasını sağlayan bir konteynerleştirme teknolojisi olduğu için bu alanda yaygın olarak kullanılır. Bu çalışma, mikro hizmetlerin Docker konteynerleri içinde nasıl dağıtıldığına odaklanarak, ortaya çıkan zorlukları ve bu zorluklara yönelik çeşitli çözüm önerilerini sunmaktadır.
- **"Docker-based Microservice Architecture for Scalable and Fault-Tolerant Systems" (Ölçeklenebilir ve Hata Toleranslı Sistemler İçin Docker Tabanlı Mikro Hizmet Mimarisi):** Bu çalışma, ölçeklenebilir ve hata toleranslı sistemler için Docker tabanlı bir mikro hizmet mimarisini araştırır. Mikro hizmetlerin Docker konteynerleri içinde çalıştırılması, sistemlerin daha iyi ölçeklenebilirlik ve hata toleransı elde etmesini sağlar. Bu çalışmada, Docker'ın sağladığı avantajlar ve bu mimarinin nasıl tasarlandığı, uygulandığı ve yönetildiği üzerinde durulur. Ayrıca, ölçeklendirme, yük dengeleme ve hata toleransı gibi konulara odaklanılarak, Docker tabanlı mikro hizmet mimarisinin performans ve dayanıklılık açısından nasıl optimize edilebileceği incelenir.

- **"Microservices Deployment with Docker: Challenges and Solutions" (Docker ile Mikro Hizmetler Dağıtımı: Zorluklar ve Çözümler):** Bu çalışma, Docker kullanarak mikro hizmetlerin dağıtımıyla ilgili karşılaşılan zorlukları ve bu zorluklara yönelik çözümleri inceler. Mikro hizmet mimarisinde, bir uygulama birden fazla küçük hizmete (mikro hizmetlere) bölünür ve her bir hizmet kendi konteynerinde çalışır. Docker, mikro hizmetlerin bağımsız olarak paketlenmesini ve dağıtılmasını sağlayan bir konteynerleştirme teknolojisi olduğu için bu alanda yaygın olarak kullanılır. Bu çalışma, mikro hizmetlerin Docker konteynerleri içinde nasıl dağıtıldığına odaklanarak, ortaya çıkan zorlukları ve bu zorluklara yönelik çeşitli çözüm önerilerini sunmaktadır.
- **"Docker-based Microservice Architecture for Scalable and Fault-Tolerant Systems" (Ölçeklenebilir ve Hata Toleranslı Sistemler İçin Docker Tabanlı Mikro Hizmet Mimarisi):** Bu çalışma, ölçeklenebilir ve hata toleranslı sistemler için Docker tabanlı bir mikro hizmet mimarisini araştırır. Mikro hizmetlerin Docker konteynerleri içinde çalıştırılması, sistemlerin daha iyi ölçeklenebilirlik ve hata toleransı elde etmesini sağlar. Bu çalışmada, Docker'ın sağladığı avantajlar ve bu mimarinin nasıl tasarlandığı, uygulandığı ve yönetildiği üzerinde durulur. Ayrıca, ölçeklendirme, yük dengeleme ve hata toleransı gibi konulara odaklanılarak, Docker tabanlı mikro hizmet mimarisinin performans ve dayanıklılık açısından nasıl optimize edilebileceği incelenir.

2.2 Docker ve Bulut Bilişim:

- **"Docker Containers in Cloud Computing Environments: A Survey" (Bulut Bilişim Ortamlarında Docker Konteynerleri: Bir Araştırma):** Bu çalışma, bulut bilişim ortamlarında Docker konteynerlerinin kullanımını araştırmaktadır. Docker konteynerleri, bulut bilişim altyapısında önemli bir rol oynamaktadır çünkü uygulamaların hızlı bir şekilde dağıtılmasını, ölçeklendirilmesini ve yönetilmesini sağlar. Bu çalışmada, Docker konteynerlerinin bulut bilişim ortamlarındaki yaygın kulla-

nımı ve avantajları incelenir. Ayrıca, Docker konteynerlerinin bulut bilişim ortamlarında karşılaşılan zorluklar, performans etkisi, ağ iletişimi ve veri yönetimi gibi konulara da değinilir.

- **"Docker Orchestration Tools for Container Cluster Management in Cloud Environments" (Bulut Ortamlarında Konteyner Kümesi Yönetimi İçin Docker Orkestrasyon Araçları):** Bu çalışma, bulut ortamlarında konteyner kümesi yönetimi için Docker orkestrasyon araçlarını araştırır. Docker, tek bir konteynerin yönetimini sağlamak için kullanılabilirken, birden fazla konteynerden oluşan karmaşık uygulama yapılarının yönetimi için orkestrasyon araçlarına ihtiyaç duyulur. Bu çalışmada, bulut ortamlarında Docker konteyner kümesinin nasıl yönetildiği, dağıtım stratejileri, otomatik ölçeklendirme, yük dengelemesi ve hata toleransı gibi konular ele alınır. Ayrıca, popüler Docker orkestrasyon araçları olan Kubernetes, Docker Swarm ve Apache Mesos gibi araçlar incelenir.
- **"Security Considerations for Docker Deployments in Cloud Infrastructure" (Bulut Altyapısında Docker Dağıtımları İçin Güvenlik Düşünceleri):** Bu çalışma, bulut altyapısında Docker dağıtımları için güvenlik düşüncelerini ele almaktadır. Docker konteynerlerinin bulut altyapısında kullanılması, güvenlik açısından dikkate alınması gereken bazı zorlukları beraberinde getirebilir. Bu çalışmada, Docker konteynerlerinin güvenlik riskleri, izolasyon önlemleri, veri güvenliği ve ağ güvenliği gibi konular incelenir. Ayrıca, Docker'da güvenlik en iyi uygulamaları ve bu uygulamaların bulut altyapısında nasıl uygulanabileceği tartışılır.

2.3 Docker ve Güvenlik:

- **"Container Security: Vulnerabilities, Threats, and Docker Best Practices" (Konteyner Güvenliği: Zafiyetler, Tehditler ve Docker En İyi Uygulamaları):** Bu çalışma, konteyner güvenliği konusunda zafiyetleri, tehditleri ve Docker için en iyi uygulamaları araştırmaktadır. Docker konteynerleri, paylaşılan bir işletim sistemi çekirdeği üzerinde çalıştığından, kötü niyetli kullanıcıların bir konteynerden

diğerine erişmesi veya güvenlik açıklarından yararlanması potansiyeli vardır. Bu çalışmada, Docker konteynerlerinin karşılaşılabileceği güvenlik zafiyetleri ve tehditler incelenir. Ayrıca, Docker için en iyi uygulamalar, konteyner güvenliğini artırmak için alınması gereken önlemleri ve konteynerlerin güvenliğini sağlamak için kullanılacak teknolojileri içerir.

- **"Securing Docker Containers: Challenges and Solutions" (Docker Konteynerlerinin Güvenliği: Zorluklar ve Çözümler):** Bu çalışma, Docker konteynerlerinin güvenliğini araştırırken karşılaşılan zorlukları ve bu zorluklara yönelik çözümleri ele almaktadır. Docker konteynerlerinin güvenliği, konteynerlerin izolasyonu, güvenlik açıkları, saldırı yüzeyi ve veri güvenliği gibi çeşitli faktörleri içerir. Bu çalışmada, Docker konteynerlerinin güvenlik zorluklarına odaklanarak, bu zorlukların nasıl üstesinden gelinebileceği ve konteynerlerin güvenliğini sağlamak için kullanılacak çeşitli çözümler sunulur. Örnek olarak, konteyner güvenliği için izolasyon önlemleri, imaj güvenliği, ağ güvenliği ve erişim kontrolleri gibi konular ele alınır.

3 PROJE TANITIMI

3.1 Giriş

Projemizin amacı, Docker kullanarak paralel işlem yapabilen sistemler, uygulamalar ve servisler için çoklu konteynerlar kullanarak bir sistem tasarlamaktır. Docker, izolasyon sağlayarak her bir konteyneri farklı özellikler ve işlevler için optimize etme imkanı sunar ve bu da uygulamanın ölçeklenebilir olmasını sağlar.

3.2 Yapı ve Teknolojiler:

Projede, Laravel yapısı kullanıldı. Bu sayede, arayüz ve backend arasında esneklik sağlayarak birlikte çalışma imkanı elde edildi. Arayüz için başlangıçta bir hazır template kullanıldı, ancak bu template'leri ihtiyaçlarımıza göre özelleştirildi ve değişiklikler yapıldı. Daha sonra backend ile entegrasyonu gerçekleştirerek ayarlamaları tamamlandı.

3.3 Güvenlik Önlemleri:

Projenin güvenliği için, login sayfası tasarlandı ve erişim kısıtlamaları eklendi. Bu sayede, giriş yapmayan kullanıcıların diğer sayfalara erişmesi engellendi. Kullanıcıların güvenliğini ve verilerin gizliliğini korumak önemli bir hedefimiz oldu.

3.4 Docker ile Konteynerleme:

Son aşamada, Docker işlemlerini gerçekleştirerek konteynerleme işlemleri tamamlandı. Docker'ın avantajlarından yararlanarak, uygulamayı farklı konteynerlar içinde çalıştırılabilirdi. Böylelikle, her bir konteynerin optimize edilmiş bir şekilde çalışmasını sağlayarak işleri belirtilen sayıda konteynera bölerek daha esnek bir yapı oluşturuldu. Bu da performansı artırırken sistem üzerindeki yükü dengelememize yardımcı oldu.

3.5 İlerleme Takibi ve Bildirim Sistemi:

Projenin arayüzünde, canlı olarak ilerleme bilgilerini gösteren bir özellik bulunmaktadır. Kullanıcılara, toplam iş miktarı, tamamlanma yüzdesi ve en hızlı çalışan konteynerlar

gibi önemli bilgiler sunarak işlerin durumunu takip etmeleri sağlandı. Bu sayede kullanıcılar, işlerin ne kadar ilerlediğini ve hangi konteynerların en etkili olduğunu görebilmektedir.

3.6 Sonuç:

Özetlemek gerekirse, projemizde Docker kullanarak paralel işlem yapabilen bir sistem tasarlandı. Laravel yapısını kullanarak esnek bir yapı oluşturuldu. Güvenlik önlemleri olarak kullanıcıların ve verilerin güvenliğini sağlandı. Docker ile konteynerleme işlemlerini gerçekleştirerek uygulamayı optimize ettik ve performansı artırdık. Arayüzde canlı ilerleme bilgilerini gösteren bir özellik ekleyerek kullanıcıların işlerin durumunu takip etmelerini kolaylaştırdık.

Projenin başarıyla tamamlanması için bir dizi adım takip edildi. İlk olarak, gereksinimlerimizi belirledik ve Docker'ı seçtik çünkü konteynerleme teknolojisi olarak bize izolasyon ve esneklik sağlayan bir çözüm sunuyordu. Ardından, Laravel yapısını kullanarak arayüz ve backend arasında etkili bir iletişim ve veri akışı sağladık.

Arayüz tarafında, hazır bir template kullanarak başladık, ancak bunu kendi tasarım ve işlevsel gereksinimlerimize uygun şekilde özelleştirdik. Kullanıcılarımızın deneyimini en üst düzeye çıkarmak için kullanıcı dostu ve sezgisel bir arayüz tasarladık.

Backend tarafında, güvenlik önlemleri almak için login sayfasını tasarladık ve kullanıcıların kimlik doğrulama sürecini sağladık. Bu sayede, yalnızca yetkilendirilmiş kullanıcıların sistemimizi kullanabilmesini sağladık ve veri güvenliğini sağladık.

Docker kullanarak konteynerleme işlemlerini gerçekleştirdik. Her bir konteyner, farklı özelliklere ve işlevlere sahip olduğu için optimize edilebilir hale geldi. İşleri parçalara bölerek belirli sayıda konteynera dağıttık, bu da sistemimizin esnekliğini artırdı ve işleri daha hızlı ve etkin bir şekilde gerçekleştirebildi.

Projenin devamında, kullanıcıların işleri daha etkili bir şekilde yönetmelerini sağlamak için bazı ek özellikler ekledik. Örneğin, kullanıcılar işleri önceliklendirebilir ve farklı

konteynerlarda çalışacak işlere öncelik atayabilir. Bu, daha hızlı tamamlanması gereken işlerin öncelikli olarak çalıştırılmasını sağladı ve genel performansı artırdı.

Ayrıca, proje takımının iletişimini kolaylaştırmak için bir bildirim sistemi entegre ettik. Kullanıcılar, tamamlanan işler hakkında anlık bildirimler aldı ve gerektiğinde müdahale edebildi. Bu, ekip üyelerinin işlerin durumunu takip etmesini ve projenin ilerlemesini koordine etmesini kolaylaştırdı.

Sonuç olarak, Docker kullanarak paralel işlem yapabilen bir sistem tasarladık ve bu sistemde Laravel yapısını kullanarak esneklik sağladık. Güvenlik önlemleri olarak kullanıcıların ve verilerin güvenliğini koruduk. Docker ile konteynerleme işlemlerini gerçekleştirerek optimize edilmiş bir sistem oluşturduk. Arayüzde ilerleme takibi ve bildirim sistemi gibi özellikler ekleyerek kullanıcıların işleri daha etkili bir şekilde yönetmelerini sağladık.

Bu proje sayesinde, paralel işlem yapabilen sistemlerin performansını artırma, esneklik sağlama ve güvenliği sağlama konularında önemli bir deneyim kazandık.

4 KULLANILAN YAZILIMLAR VE YÖNTEMLER

4.1 Neden Tercih Ettik ?

Laravel, web uygulamaları için tercih ettiğimiz bir framework'tür. MVC yapısı ve geliştirme kolaylığı sağlar. Hazır bileşenleriyle hızlı ve verimli bir şekilde uygulama geliştirebiliriz. Geniş topluluğuyla destek alabileceğimiz bir kaynağa sahiptir. Bu nedenlerle, Laravel'i tercih ettik.

Docker'ı tercih etmemizin nedeni, uygulamaların izole edilmiş, taşınabilir ve hızlı bir şekilde dağıtılmasını sağlayan konteyner teknolojisiyle geliştirme ve dağıtım süreçlerini kolaylaştırmasıdır.

MySQLi, MySQL veritabanı ile PHP arasında bir bağlantı sağlayan bir PHP eklentisidir. MySQLi'nin tercih edilmesinin nedeni, gelişmiş özellikleri, performans iyileştirmeleri ve güvenlik önlemleriyle MySQL ile daha etkili ve güvenli bir şekilde etkileşim sağlamasını

sağlamasıdır.

4.2 Web Uygulama Geliştirmek İçin

4.2.1 Laravel

Laravel, popüler bir PHP framework'üdür ve MVC (Model-View-Controller) mimarisini kullanır. Bu mimari, uygulamanın farklı katmanlarını ayırarak kodun düzenli, okunabilir ve bakımı kolay hale gelmesini sağlar. Laravel, model, view, controller ve route gibi temel bileşenlerden oluşur. Model, veri katmanını temsil ederken, view kullanıcı arayüzünü ve controller iş mantığını yönetir. Route ise istekleri doğru controller'a yönlendirir.

Laravel'in MVC yapısı, kodun düzenli tutulmasını, bileşenlerin bağımsız olmasını ve geliştirme sürecinin daha etkili olmasını sağlar. Bu sayede uygulama daha yeniden kullanılabilir, test edilebilir ve bakımı kolay hale gelir.

4.2.2 Kullanım Kolaylığı:

Laravel, kullanıcı dostu ve kolay bir kullanım deneyimi sunar. Gelişmiş komut satırı araçları, hazır bileşenler ve entegre özellikler sayesinde hızlı bir şekilde uygulama geliştirebilirsiniz. Laravel'in sunduğu Artisan komut satırı aracı, geliştirme sürecini hızlandırır ve tekrarlayan görevleri otomatikleştirir. Artisan komutları, veritabanı yönetimi, migrasyonlar, testlerin çalıştırılması, güvenlik kontrolleri ve daha birçok işlemi kolaylıkla gerçekleştirmenizi sağlar.

Ayrıca, Laravel'in içerdiği hazır bileşenler ve entegre özellikler, geliştirme sürecini daha da kolaylaştırır. Örneğin, kimlik doğrulama ve yetkilendirme sistemleri gibi önemli işlevlerin birçoğu Laravel'de hazır olarak sunulur ve uygulamanızı güvenli hale getirmek için kolayca kullanılabilir. Ayrıca, oturum yönetimi, önbellekleme, e-posta gönderme, dosya depolama ve diğer yaygın kullanılan işlevler için Laravel'in entegre özelliklerinden yararlanabilirsiniz.

4.2.3 MVC Mimari ve Yapılandırma:

Laravel'in Model-View-Controller (MVC) mimarisi, uygulamanın farklı katmanlarını ayırarak kodun düzenli tutulmasını ve bakımını kolaylaştırır. Model, view ve controller bileşenleri arasındaki net ayırım, iş mantığının yönetimini ve kullanıcı arayüzünün ayrı tutulmasını sağlar.

4.3 Konteynerize Etmek İçin

4.3.1 Docker

Docker, uygulamaların ve hizmetlerin paketlenmesi, dağıtımı ve çalıştırılmasını kolaylaştıran bir yazılım platformudur. Docker, konteyner teknolojisi kullanarak uygulamaların izole edilmiş ve taşınabilir bir şekilde çalışmasını sağlar.

Docker'ın çalışma yapısı şu şekildedir:

- **Konteyner:** Docker'ın temel yapı birimidir. Konteyner, uygulamanın tüm bağımlılıklarını (kod, çalışma zamanı, kütüphaneler, ortam değişkenleri vb.) bir araya getirir ve izole bir ortamda çalışmasını sağlar. Konteynerlar, bir uygulamayı çalıştırmak için gerekli olan tüm bileşenleri içerir ve bu sayede uygulamaları farklı ortamlarda tutarlı bir şekilde çalıştırabilirsiniz.
- **Docker Image (Docker İmajı):** Bir Docker konteynerini oluşturmak için kullanılan bir şablondur. Bir Docker imajı, bir veya daha fazla katman (layer) olarak adlandırılan yapı taşlarından oluşur. Her katman, imajın farklı bir bileşenini veya yapılandırmasını temsil eder. Bir Docker imajı, bir veya birden fazla konteyneri oluşturmak için kullanılabilir.
- **Dockerfile (Docker Dosyası):** Docker imajlarının nasıl oluşturulacağını tanımlayan bir metin dosyasıdır. Dockerfile, bir uygulamanın çalıştırılması için gerekli olan adımları ve komutları belirtir. Dockerfile, uygulama kodunu, çalışma zamanını, bağımlılıkları ve diğer yapılandırmaları imaja dahil etmek için kullanılır.
- **Docker Registry:** Docker imajlarının depolandığı ve paylaşıldığı bir merkezi kaynak. Docker Hub, en popüler ve yaygın kullanılan Docker Registry'dir. Docker Hub'da, birçok hazır Docker imajı bulabilir ve kendi imajlarınızı da paylaşabilir-

siniz. Ayrıca, Docker Registry'lerini yerel olarak da kurabilir ve kullanabilirsiniz.

Docker'ın avantajları şunlardır:

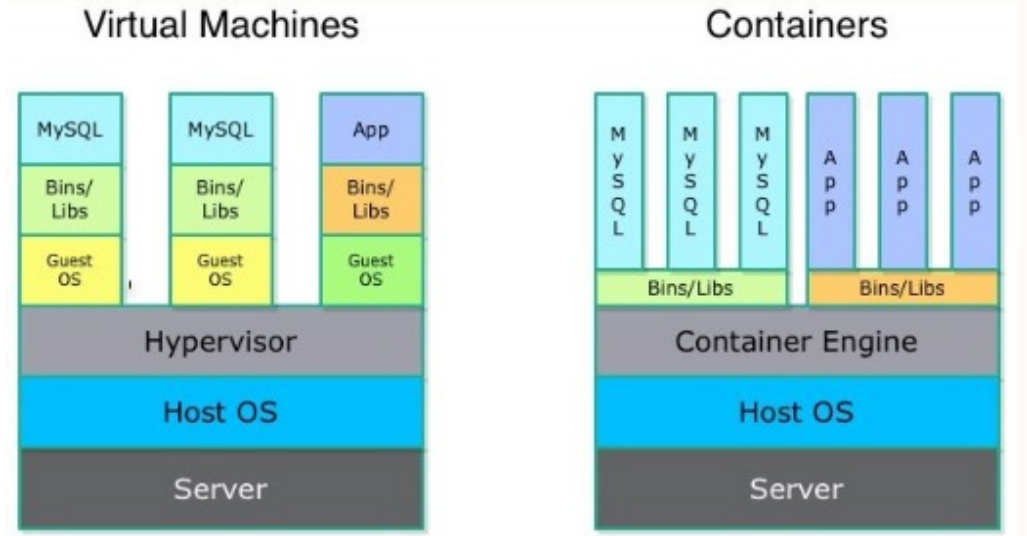
- **Taşınabilirlik:** Docker, uygulamaların farklı platformlarda ve ortamlarda sorunsuz bir şekilde çalışmasını sağlar. Bir kez oluşturulan Docker imajı, farklı sistemlerde kolayca dağıtılabilir ve çalıştırılabilir.
- **İzolasyon:** Docker konteynerleri, uygulamaları ve bağımlılıklarını izole ederek, bir konteynerin diğerlerine etkileşimde bulunmasını engeller. Bu, güvenlik ve istikrar açısından önemli bir avantaj sağlar.
- **Hızlı dağıtım ve ölçeklendirme:** Docker, uygulamaların hızlı bir şekilde dağıtılmasını ve ölçeklendirilmesini sağlar. İmajların hızlı bir şekilde oluşturulması ve konteynerlerin hızlı bir şekilde başlatılması sayesinde uygulamaların hızlıca yayınlanması mümkün olur.
- **Verimlilik:** Docker, kaynakların etkin bir şekilde kullanılmasını sağlar. Konteynerler, daha az bellek ve işlemci gücü tüketerek daha fazla uygulamanın çalışmasını sağlar.
- **Veri bütünlüğü:** Docker, verilerin konteynerlerle birlikte taşınmasını ve yönetilmesini kolaylaştırır. Konteynerler, verileri izole bir şekilde tutar ve bu da veri bütünlüğünü sağlar.
- **Hızlı geri alma ve güncelleme:** Docker, imaj ve konteyner tabanlı bir yaklaşım kullanılarak hızlı geri alma ve güncelleme süreçleri sağlar. Eski bir imaja geri dönerek veya güncellenmiş bir imajla yeni bir konteyner oluşturarak sistemde yapılan değişiklikleri geri almak veya güncellemek kolaydır.
- **Kaynak verimliliği:** Docker, paylaşılan çekirdek özelliklerini kullanarak kaynakların daha verimli bir şekilde kullanılmasını sağlar. Birçok konteyner, tek bir işletim sistemi çekirdeği üzerinde çalışır ve bu da sistem kaynaklarının daha etkin bir şekilde kullanılmasını sağlar.
- **Topluluk desteği:** Docker, geniş bir kullanıcı ve geliştirici topluluğuna sahiptir. Bu topluluk, Docker hakkında bilgi paylaşımı, sorun giderme ve yenilikçi çözümler sunma konusunda yardımcı olur. Docker Hub gibi kaynaklar, hazır kullanıma uygun imajlar ve araçlar sağlar.

Docker'ın bu avantajları, uygulama geliştirme ve dağıtım süreçlerinde hızlilik, esneklik, güvenlik ve ölçeklenebilirlik sağlar. Ayrıca, Docker'ın genişletilebilirlik ve entegrasyon yetenekleri, farklı platformlar ve araçlarla uyumlu çalışmayı kolaylaştırır.

Neden Docker ?

Docker, bir yazılımın uygulama ve bağımlılıklarını bir "konteyner" olarak paketlemek ve çalıştırmak için kullanılan bir platformdur. Docker'ın sanal makinelerden farklı olarak konteynerizasyon teknolojisi üzerine kurulu olması, birkaç önemli avantaj sunmaktadır:

- **Daha hafif ve hızlı:** Sanal makineler (VM'ler), her biri kendi işletim sistemini çalıştıran ayrı bir sanal makine olduğundan, daha fazla kaynak tüketir ve yavaş çalışabilir. Docker konteynerleri ise ana işletim sistemini paylaşır ve sadece uygulama ve bağımlılıkları içerir. Bu nedenle, Docker konteynerleri daha hafif ve daha hızlıdır.
- **Daha verimli kaynak kullanımı:** Docker, aynı fiziksel makine üzerinde birden çok konteyner çalıştırabilme yeteneği sayesinde kaynakları daha verimli kullanmanızı sağlar. Sanal makinelerde her bir VM, kendi işletim sistemini çalıştırdığından kaynak israfı yaşanabilirken, Docker konteynerleri aynı işletim sistemini paylaşarak daha etkili bir kaynak yönetimi sağlar.
- **Taşınabilirlik ve uyumluluk:** Docker konteynerleri, uygulama ve bağımlılıklarını bir araya getirdiği için taşınabilirlik ve uyumluluk sağlar. Konteynerler, herhangi bir ortamda, herhangi bir makinede aynı şekilde çalışabilir. Bir kez oluşturulduklarında, Docker konteynerleri kolayca dağıtılabilir ve başka bir makineye taşınabilir.
- **Hızlı dağıtım ve ölçeklendirme:** Docker konteynerleri, uygulamaların hızlı dağıtımını ve ölçeklendirilmesini sağlar. Konteynerlerin hızlı bir şekilde başlatılması ve durdurulması mümkündür. Ayrıca, bir uygulamayı birden çok konteyner olarak çalıştırarak yüksek kullanılabilirlik ve ölçeklenebilirlik elde etmek kolaydır.
- **İzolasyon ve güvenlik:** Docker konteynerleri, birbirinden izole edilmiş çalışma or-



Şekil 4.1: Docker vs Sanal Makina

tamları sağlar. Her konteyner, kendi dosya sistemini, ağ bağlantılarını ve süreçlerini izole eder. Bu, bir konteynerin diğerlerinden etkilenmeyeceği anlamına gelir, böylece daha güvenli bir ortam sağlar.

Docker vs Sanal Makianlar

Sanal makine (VM) teknolojisi, ayrı işletim sistemleri ve bağımsız çalışma ortamları sağlar. Her bir sanal makine, kendi işletim sistemi, yazılım yığını ve kaynaklara sahip bir sanal bilgisayardır. VM'ler, fiziksel makine üzerinde çalışırken, her biri kendi kaynaklara sahip olduğundan izolasyon sağlarlar. Bu, farklı işletim sistemlerini aynı fiziksel makinede çalıştırmanın bir yolunu sunar. Ancak, VM'lerin daha fazla bellek, işlemci gücü ve depolama alanı gibi kaynaklara ihtiyaç duyması ve daha yavaş çalışması dezavantajlarından bazılarıdır.

Docker konteynerleri, hafif ve taşınabilir uygulama çalıştırma birimleridir. Docker, aynı çekirdek üzerinde çalışan ve paylaşılan kaynakları kullanan konteynerleştirilmiş uygulamaları yönetmek için bir platform sağlar. Her bir Docker konteyneri, izole bir çalışma ortamı sunar, ancak farklı bir işletim sistemi çalıştırmak yerine, ana makinedeki işletim sistemi ve çekirdek üzerinde çalışır. Bu, daha düşük bellek kullanımı ve daha hızlı başlatma süreleri gibi avantajlar sağlar. Ayrıca, Docker konteynerleri uygulama dağıtımını kolaylaştırır ve yazılımın farklı ortamlarda tutarlı bir şekilde çalışmasını sağlar.

Biz sanal makina olarak VirtualBox'ı baz alacağız.

Docker vs VirtualBox

Docker	VirtualBox
Konteyner tabanlı sanallaştırma	Tam sanallaştırma
Aynı çekirdek üzerinde çalışır	Farklı işletim sistemlerini çalıştırır
Hafif ve taşınabilir konteynerler	Tam işletim sistemine sahip sanal makineler
İzole çalışma ortamı sağlar	İzole çalışma ortamı sağlar
Daha düşük bellek kullanımı ve hızlı başlatma	Daha fazla kaynak gerektirir
Uygulama dağıtımını kolaylaştırır	Farklı işletim sistemlerinin testi için kullanılır
Docker Hub'da konteyner görüntüleri paylaşılabilir	Sanal makine görüntüleri kullanılabilir
Microservisler ve ölçeklenebilir uygulama geliştirmeleri	Özel izolasyon gerektiren durumlar

Tablo 4.1: Docker vs. VirtualBox

4.4 Veri Tabanı İçin

4.4.1 MYSQL

MySQL, açık kaynaklı bir ilişkisel veritabanı yönetim sistemidir. İlişkisel veritabanı yönetim sistemleri, verileri tablolarda saklayarak veriler arasındaki ilişkileri yönetmeyi sağlar. MySQL, istemci-sunucu modeline dayalı olarak çalışır ve bir sunucu üzerinde çalışan çok kullanıcılı bir veritabanı sistemidir.

MySQL, hızlı, güvenilir ve geniş bir kullanıcı tabanına sahip olmasıyla bilinir. Veritabanı yönetimi, veri manipülasyonu, sorgulama, güvenlik, yedekleme ve geri yükleme gibi birçok temel veritabanı işlemini destekler. Ayrıca, MySQL'in geniş bir uyumluluk listesi vardır ve farklı programlama dilleri ve platformlarla kolayca entegre olabilir.

MySQL, birçok web uygulamasında ve büyük ölçekli sistemlerde kullanılan yaygın bir veritabanı çözümüdür. Veri bütünlüğünü koruma, veritabanı güvenliği, yüksek performans, ölçeklenebilirlik ve kullanıcı dostu bir arayüz gibi önemli özelliklere sahiptir.

MySQL, veri tabanı yönetimi için güçlü ve esnek bir seçenek olup, çeşitli projelerde verilerin etkin bir şekilde saklanması ve yönetilmesini sağlar.

5 PROJE’NİN İŞLEYİŞİ

5.1 Arayüz Kısmı

Bu projede arayüz kısmı için belirtilen işleyiş aşağıda açıklanmaktadır. Uygun bir template bulunmuş ve projeye uygun hale getirilmiştir. Gereksiz kısımlar temizlenerek sağ menü bölümünde Dashboard, görevler (Ping, Theharvester), sistem logları ve Profil bölümleri oluşturulmuştur.

Projeyi ilk açtığınızda, kullanıcı login sayfasıyla karşılaşacaksınız. Kimlik bilgilerinizi girdikten sonra doğrulama yapıp diğer sayfalara erişebileceksiniz. Giriş yaptıktan sonra karşılaşacağınız ilk sayfa Kontrol Paneli olacaktır. Kontrol panelinde gerçekleştirilebilecek işlemler açıklanacaktır.

Sağ kısımda menü bulunmaktadır. Buradan istediğiniz zaman görevler bölümünden görev oluşturabilir ve önceden oluşturulan görevle ilgili işlemleri görebilirsiniz. Bir Ping görevi oluşturulduğunda, bu konteynırların çıktıları, kayıp yüzdesi, pingin maksimum geri dönüş süresi, minimum geri dönüş süresi ve ortalama milisaniye geri dönüş süresi gibi bilgilere erişebilirsiniz. Ayrıca sağ menüde sistem log kayıtlarını görüntüleyebilirsiniz. Log kayıtlarının altında kullanıcının hangi görevi ne zaman çalıştırdığı, görevin tamamlanma süresi, başarılı veya başarısız olma durumu ile oturum açma ve kapatma bilgileri kolaylıkla görüntülenebilir.

Hesabım bölümü, profili ile ilgili düzenlemeler yapmayı sağlar. Bu bölümde profil bilgilerini güncelleyebilir, şifreniği değiştirebilir, iletişim tercihlerini yönetebilir ve gizlilik ayarlarını düzenlenebilir. Bu şekilde, profili yönetebilir ve hesap istediğimiz şekilde kişiselleştirebilir hale getirildi.

5.2 Arka plan işleyişi

Projenin alt yapısı, PHP framework’ü olan Laravel kullanılarak geliştirilmiştir. Projenin çalışması için bir docker-compose.yml dosyası kullanılarak Docker’da bir servis oluşturulmuştur. Bu servis, iki konteyner oluşturarak projenin çalışmasını sağlamaktadır. Docker yönetimi için Docker Engine API kullanılmıştır.

5.2.1 Docker Engine API

Docker, Docker Engine API olarak adlandırılan arka plan programı sayesinde etkileşim için bir API sağlamaktadır. Docker Engine API, birçok modern programlama dilinde yer alan HTTP kütüphanesi tarafından erişilen bir RESTful API'dir. Projedeki PHP SDK güncel olmadığı için, Docker API'yi kullanarak Laravel uygulaması ile haberleşme sağlayan DockerService adında bir servis yazılmıştır. Bu sayede konteyner oluşturma, çalıştırma, detaylarını görüntüleme gibi işlemler yapılabilmektedir.

5.2.2 Laravel Özellikleri

Bu projede, Laravel'in sağladığı özellikler kullanılarak geliştirme yapılmıştır. Bir IP'ye ping atma görevi oluşturma süreci şu şekildedir:

Kullanıcı, ping oluşturma görevi formunu doldurmalıdır. Formda, görev başlığı, görev açıklaması, atılacak IP adresi, oluşturulacak konteyner sayısı ve her bir konteynerda kaç adet ping atılacağı gibi fieldler bulunmaktadır.

Form doldurulduktan sonra, PingRequest sınıfı devreye girecektir. Bu, Laravel'in sağladığı bir özellik olan Request'tir ve formdan gelen bilgilerin doğruluğunu sağlamaktadır. Eğer bilgiler doğruysa, PingController'a geçilir. Eğer bilgiler doğrulanamazsa, form sayfasına geri yönlendirilir ve eksik olan fieldlerin altında bir hata mesajı yazılır.

PingController, yeni bir görev oluşturarak kullanıcıyı görev listesi sayfasına yönlendirir ve "oluşturuldu" mesajını yazdırır.

Yeni bir görev oluşturulduğunda, Laravel'in bir diğer özelliği olan "Event (Tetikleyici) & Listener (Dinleyici)" devreye girer.

Event & Listener: Bu proje, yeni bir görev oluşturulduğunda event tetiklenir ve Listener çalışmaya başlar. Listener aracılığıyla DockerService çalıştırılır. Kullanıcıyı Form sayfasında bekletmemek için Listener kuyruğa alınır ve DockerService işlemi tamamlandıktan sonra görev durumu güncellenir. Başarılıysa 1, başarısızsa 2 olarak bildirilir

5.3 Veritabanı kısmı İçin

MySQL veritabanı kullanılarak bir proje için optimize edilmiş bir veritabanı şeması oluşturuldu. Bu şema, "users" (kullanıcılar), "ping" (ping kayıtları) ve "pingcontainers" (ping konteynerleri) adında üç tabloyu içermektedir. Bu düzenleme, veritabanının performansını artırmak ve veri bütünlüğünü sağlamak amacıyla yapılmıştır.

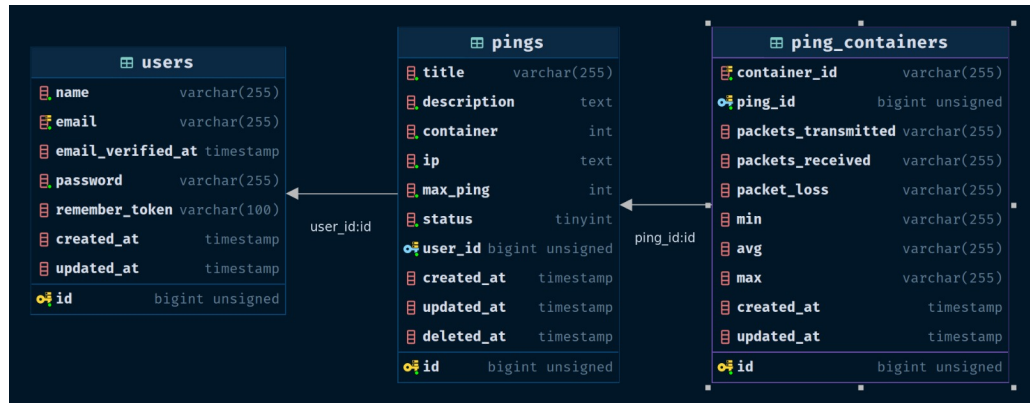
"users" tablosu, kullanıcıların temel bilgilerini içerir ve her kullanıcıya birincil anahtar (ID) ile benzersiz bir kimlik atar. Ayrıca kullanıcı adı, e-posta adresi ve diğer ilgili bilgiler gibi sütunlar da içerir.

"pingcontainers" tablosu, kullanıcıların oluşturabileceği ping konteynerlerini temsil eder. Her konteyner birincil anahtar ile tanımlanır ve kullanıcıya ait olacak şekilde kullanıcı kimliği (user ID) ile ilişkilendirilir. Bu tablo, kullanıcının birden fazla konteyner oluşturabilmesine olanak sağlar.

"ping" tablosu, gerçekleşen ping olaylarını kaydetmek için kullanılır. Her ping kaydı, birincil anahtar ile tanımlanır ve bir konteynere (container ID) ve kullanıcıya (user ID) bağlanır. Bu ilişki, her ping kaydının hangi kullanıcı ve konteyner tarafından oluşturulduğunu gösterir.

Bu veritabanı şeması, kullanıcıların birden fazla konteyner ve görev oluşturabilmesini sağlar. Ayrıca, her görevin bir kullanıcı tarafından oluşturulduğu ilişkisi sayesinde veri bütünlüğünü korur.

Bu düzenleme, veritabanının daha etkili bir şekilde veri saklamasını ve verilere hızlı erişim sağlamasını amaçlamaktadır. Ayrıca, veri bütünlüğünü korumak için gereken ilişkileri de sağlamaktadır.



Şekil 5.1: Sql ping tabloları

6 SONUÇLAR VE ÖNERİLER

7 EKLER

KAYNAKLAR

- [1] Brogi, A., Pahl, C., and Soldani, J. 2020. On enhancing the orchestration of multi-container docker applications. In *Advances in Service-Oriented and Cloud Computing: Workshop of ES OCC 2018, Como, Italy, September 12–14, 2018, Revised Selected Papers*, volume 7, pages 21–33. Springer International Publishing. [Ziyaret Tarihi: 30 Mayıs 2022]
- [2] Docker2023. Docker overview. [Ziyaret Tarihi: 12 Mart 2023]
- [3] Ibrahim, M. H., Sayagh, M., and Hassan, A. E. 2021. A study of how docker compose is used to compose multi-components systems. *Empirical Software Engineering*, 26: 1-27 [Ziyaret Tarihi: 6 Nisan 2022]
- [4] Sharma, V., Saxena, H. K., and Singh, A. K. (2020). Docker3 for multi-containers web application. In *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pages 589–592. IEEE. [Ziyaret Tarihi: 6 Nisan 2022]
- [5] McKendrick, R. (2020). *Mastering Docker: Enhance your containerization and DevOps skills to deliver production-ready applications*. Packt Publishing Ltd. [Ziyaret Tarihi: 6 Nisan 2022]
- [6] Warriar, A. (2020). Containers vs virtual machines (vms). [Ziyaret Tarihi: 16 Mart 2023.]

ÖZGEÇMİŞ

KİŞİSEL BELGELER

Adı Soyadı :
Uyruğu : T.C.
Doğum Yeri ve Tarihi:
Adres :

Telefon :
E-mail :

EĞİTİM DURUMU

Lisans Öğrenimi : BŞEÜ Bilgisayar Mühendisliği Bölümü
Bitirme Yılı :
Lise :

İŞ DENEYİMLERİ

Yıl :
Kurum :
Stajlar :

İLGİ ALANLARI:

YABANCI DİLLER:

BELİRTMEK İSTEDİĞİNİZ DİĞER ÖZELLİKLER: