T.C.

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Web Tabanlı Konteyner Orkestrasyon Sistemi

Aleyna ÇELİK

IBRAHIM KHALIL ATTEIB YACOUB

BİTİRME ÇALIŞMASI

DANIŞMANI :

Prof. Dr. Cihan KARAKUZU

Dr. Öğr. Üyesi Burakhan ÇUBUKÇU

BİLECİK

18 Mayıs 2023

T.C.

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Web Tabanlı Konteyner Orkestrasyon Sistemi

Aleyna ÇELİK

IBRAHIM KHALIL ATTEIB YACOUB

BİTİRME ÇALIŞMASI

DANIŞMANI :

Prof. Dr. Cihan KARAKUZU

Dr. Öğr. Üyesi Burakhan ÇUBUKÇU

BİLECİK

18 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: 18 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, 35 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, 35 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 ATTEIB YACOUB

18 Mayıs 2023

İÇİNDEKİLER

ŞEKİLLER TABLOSU

1 Proje Konusunun Seçilme Nedenleri

Günümüzde teknolojik projelerin karmaşıklığı ve iş yükünün artmasıyla birlikte, verimli ve ölçeklenebilir bir altyapı tasarımı büyük önem taşımaktadır. Bu noktada, Docker ile çoklu konteynerlar kullanarak paralel iş yapan sistem tasarımı, proje için ideal bir çözüm olarak karşımıza çıkmıştır. Bu yazıda, neden Docker ve çoklu konteynerlar üzerine bir sistem tasarımı seçtiğimiz açıklandı.

- Hafif ve Taşınabilir Konteynerler: Docker, hafif ve taşınabilir konteyner teknolojisi sunmaktadır. Her bir konteyner, uygulamaların ve bileşenlerin izole bir şekilde çalışmasını sağlar. Bu özellik, projemizde her bir iş parçacığını ayrı bir konteynerde çalıştırarak paralel işlem yapmamıza olanak tanır. Ayrıca, konteynerlerin taşınabilir olması, farklı ortamlarda kolayca dağıtım yapmamızı sağlar.
- Ölçeklenebilirlik: Docker ve çoklu konteynerlar, projemizin ölçeklenebilirliğini artırmak için ideal bir yapı sağlar. İhtiyaç duydukça yeni konteynerler oluşturabilir ve sistemi yatay olarak genişletebiliriz. Bu sayede, artan iş yüküne kolayca uyum sağlayabilir ve sistem performansını istenilen seviyede tutabiliriz.
- Kolay Dağıtım ve Yönetim: Docker, konteynerlerin dağıtımını ve yönetimini kolaylaştıran bir araç sunar. Konteynerlerin Docker Hub gibi bir merkezi depoda paylaşılabilmesi, projemizin dağıtım sürecini hızlandırır ve yazılım sürümlerini güncelleme kolaylığı sağlar. Ayrıca, Docker'ın sağladığı araçlar ve komutlarla konteynerleri kolayca yönetebilir, izleyebilir ve sorunları hızla tespit edebiliriz.
- İzolasyon ve Güvenlik: Her bir konteyner, izole bir çalışma ortamı sağlar. Bu, her bir iş parçacığının birbirinden bağımsız olarak çalışmasını ve bir sorun durumunda diğerlerini etkilememesini sağlar. Böylece, projemizin güvenlik ve kararlılık açısından daha güçlü bir yapıya sahip olmasını sağlarız.

Docker ile çoklu konteynerlar kullanarak paralel iş yapan sistem tasarımını seçmemizin temel nedenleri yukarıda belirtildiği gibi hafiflik, taşınabilirlik, ölçeklenebilirlik, kolay dağıtım ve yönetim, izolasyon ve güvenlik gibi avantajlardır. Bu tasarım yaklaşımı, projenin gereksinimlerini karşılamak ve iş yükünü etkin bir şekilde yönetmek için ideal bir

çözüm sunmaktadır.

Docker'ın konteynerleştirme teknolojisi, iş parçacıklarını izole ederek paralel işlem yapmayı sağlar. Bu da projenin performansını artırırken, ölçeklenebilirliğini ve esnekliğini de sağlar. Ayrıca, Docker'ın sunduğu araçlar ve kolay yönetim özellikleri, projenin dağıtımını ve güncellemelerini hızlandırırken, sistem yönetimini de kolaylaştırır.

Tüm bu avantajlar göz önüne alındığında, Docker ile çoklu konteynerlar kullanarak paralel iş yapan sistem tasarımını seçmek, projenin başarılı bir şekilde büyümesini ve gereksinimlerini karşılamasını sağlayacak stratejik bir tercihtir.

2 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 sa-

yesinde, 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.

3 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.

3.1 Konteyner 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 Doc-

ker 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.

3.2 Konteyner 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 kullanımı 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.

3.3 Konteyner 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 karşılaşabileceği güvenlik zafiyetleri ve tehditler incelenir. Ayrıca, Docker için en iyi uygulamalar, konteyner güvenliğini artırmak için alınması gereken önlemleri ve konteynerlerin güvenliğini sağlamak için kullanılabilecek 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 kullanılabilecek çeşitli çözümler sunulur. Örnek olarak, konteyner güvenliği için izolasyon önlemleri, imaj güvenliği, ağ güvenliği ve erişim kontrolleri gibi konular ele alınır.

4 PROJE TANITIMI

Bu proje, Docker kullanarak paralel işlem yapabilen sistemler için çoklu konteynerlarla bir sistem tasarlamayı amaçlar. Proje kapsamında Docker'ın izolasyon özellikleri ve optimize edilebilirlik imkanı kullanılarak her bir konteyner farklı işlevler için optimize edilir. Laravel yapısıyla esneklik sağlanır ve güvenlik önlemleri alınır. Docker konteynerleme işlemleriyle uygulama performansı artırılır ve işler daha etkin bir şekilde gerçekleştirilir. İlerleme takibi ve bildirim sistemi eklenerek kullanıcıların işlerin durumunu kolaylıkla takip etmeleri sağlanır. Projenin detayları aşağıda detaylı şekilde açıklanmıştır.

4.1 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.

4.2 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 templateleri ihtiyaçlarımıza göre özelleştirildi ve değişiklikler yapıldı. Daha sonra backend ile entegrasyonu gerçekleştirerek ayarlamaları tamamlandı.

4.3 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.

4.4 Docker ile Konteynerleme

Son aşamada, Docker işlemlerini gerçekleştirerek konteynerleme işlemleri tamamlandı. Docker'ın avantajlarından yararlanarak, uygulamayı farklı konteynerlar içinde ça-

lıştırılabildi. 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.

4.5 İlerleme Takibi ve İletişim 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.

5 KULLANILAN TEKNOLOJİLER VE YÖNTEMLER

5.1 Kullanılan Bileşenlerin Tercih Sebepleri

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.

5.2 Laravel'inSağladığı Özelliklerden Kullandıklarımız

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.

5.2.1 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.

5.2.2 Laraval'de MVC Yapısı

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

5.2.3 ORM Katmanı

Laravel, veritabanı işlemlerini kolaylaştıran bir ORM (Object-Relational Mapping) olan Eloquent'i içerir. Eloquent, veritabanı sorgularını nesne odaklı bir şekilde yapmanızı sağlar ve veritabanı işlemlerini hızlı ve verimli bir şekilde gerçekleştirmenizi sağlar.

5.2.4 Veritabanı Şeması:

Laravel'in sağladığı göç (migration) sistemini kullanarak, veritabanı şemasını kolayca oluşturabilir ve değişiklikleri yönetebilirsiniz. Bu, uygulamanızın veritabanı yapısını güncellemek ve yönetmek için esneklik sağlar.

5.2.5 Güvenlik ve Oturum Yönetimi

Laravel, güvenlik önlemleriyle donatılmıştır. Oturum yönetimi, kimlik doğrulama güvenlik işlemlerini kolayca uygulamanızı sağlar.

5.2.6 Geniş Ekosistem ve Topluluk Desteği

Laravel, büyük bir geliştirici topluluğuna sahiptir ve aktif bir ekosistemi vardır. Laravel Forge, Laravel Nova, Laravel Horizon gibi araçlar ve paketlerle işleri kolaylaştırabilir ve projenizi hızlandırabilirsiniz. Ayrıca, Laravel ile ilgili belgeler, kaynaklar ve topluluk destekleri kolayca erişilebilir.

5.3 Konteynerize Etmek İçin Docker

5.4 Konteyner Tanımı

Bir konteyner, uygulama veya hizmetlerin çalıştırıldığı ve izole edildiği bir sanal ortamdır. Konteynerler, bir işletim sistemi çekirdeğini paylaşarak çalışırken, uygulama ve bileşenlerini bir arada bulundurur ve bu şekilde taşınabilirlik, hızlı dağıtım ve ölçeklenebilirlik gibi avantajlar sunar. Konteynerler, birbiriyle çakışmadan çalışabilir ve kaynakları (bellek, işlemci, ağ vb.) etkin bir şekilde paylaşabilirler. Bunun yanı sıra, konteynerlerin oluşturulması, başlatılması ve durdurulması gibi işlemler hızlı ve kolaydır. Konteyner teknolojisi, uygulama geliştirme, test, dağıtım ve çalıştırma süreçlerini kolaylaştırarak verimliliği artırır.

5.4.1 Docker Tanımı

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.

5.4.2 Docker'ın çalışma yapısı

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

5.4.3 Konteyner tekonolojisinin avantajları

Konteyner teknolojisinin birçok avantajı vardır:

- **İzolasyon:** Konteynerler, uygulamaları ve hizmetleri birbirlerinden izole eder. Her konteyner, kendi dosya sistemini, kütüphanelerini ve diğer bağımlılıklarını içerir. Bu izolasyon, bir konteynerdeki bir uygulamanın diğer konteynerler üzerindeki performansa veya güvenliğe olumsuz etki yapmasını önler.

- **Taşınabilirlik:** Konteynerler, uygulamaları ve bileşenleri, kullandıkları ortamla bağımsız olarak taşınabilir hale getirir. Bir kez oluşturulan ve yapılandırılan konteynerler, farklı işletim sistemleri veya bulut platformları üzerinde kolaylıkla çalıştırılabilir. Bu, uygulamaların farklı ortamlarda sorunsuz bir şekilde dağıtılmasını ve çalıştırılmasını sağlar.
- **Hızlı Dağıtım:** Konteynerler, uygulama dağıtım sürecini hızlandırır. Konteynerlerin hızlı bir şekilde başlatılması ve durdurulması, yeni sürümlerin ve güncellemelerin kolayca dağıtılabilmesini sağlar. Bu da geliştirme sürecini hızlandırır ve hızlı geri dönüşlerin elde edilmesini sağlar.
- **Ölçeklenebilirlik:** Konteynerler, ölçeklenebilir bir altyapı sağlar. Birden fazla konteyneri aynı anda çalıştırabilir ve yükü dengeleyebilirsiniz. İhtiyaç duyduğunuzda konteyner sayısını artırabilir veya azaltabilirsiniz. Bu, talebe göre kaynakları esnek bir şekilde yönetmenizi sağlar ve sistem performansını artırır.
- **Kaynak Verimliliği:** Konteynerler, kaynakların daha etkin kullanılmasını sağlar. Bir konteyner, yalnızca gerektiği kadar kaynağı kullanır ve diğer konteynerlerle kaynakları paylaşır. Bu, sunucu kaynaklarının daha verimli bir şekilde kullanılmasını ve daha fazla uygulamanın veya hizmetin aynı sunucu üzerinde çalışmasını sağlar.
- **Kolay Yönetim:** Konteyner teknolojisi, uygulamaların ve bileşenlerin yönetimini kolaylaştırır. Konteyner yönetim araçları, konteynerleri oluşturmayı, yapılandırmayı, izlemeyi ve yönetmeyi sağlar. Konteynerlerin otomatik olarak yeniden başlatılması, hata durumlarında kurtarma mekanizmalarının devreye alınması gibi işlemler kolayca gerçekleştirilebilir.

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

Konteynerlar, yazılım geliştirme ve işletim ekipleri arasındaki işbirliğini güçlendirerek DevOps prensiplerini destekler. Konteynerlar, uygulamaları paketleyip taşınabilir birimler haline getirerek hızlı geliştirme, sorunsuz çalışma ve hızlı dağıtım imkanı sunar.

Bu sayede, yazılım geliştirme ekipleri uygulamaları kolayca test eder ve işletim ekipleri de hızlıca dağıtabilir. Sürekli entegrasyon ve dağıtım yöntemleri ile otomasyon sağlanır, böylece hızlı ve güvenilir bir dağıtım süreci oluşturulur. Konteynerlar sayesinde ekipler arasındaki işbirliği artar ve daha iyi hizmet kalitesi sağlanır.

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-

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 komutlarını ve açıklamalarını içeren Tablo 5.1.'da görebilirsiniz.

Tablo 5.1: Docker Komutları ve Açıklamaları

Komut	**Açıklama**
docker run	Bir konteynerin başlatılması ve çalıştırılması için kullanılır.
docker stop	Çalışan bir konteyneri durdurmak için kullanılır.
docker build	Bir Docker imajının oluşturulması için kullanılır.
docker push	Bir Docker imajının uzak bir imaj deposuna yüklenmesi için kullanılır.
docker pull	Bir Docker imajının uzak bir imaj deposundan indirilmesi için kullanılır.
docker exec	Çalışan bir konteynere komut çalıştırmak için kullanılır.
docker ps	Çalışan konteynerleri listelemek için kullanılır.
docker images	Oluşturulmuş Docker imajlarını listelemek için kullanılır.

5.5 Veri Tabanı İçin MYSQL

MySQL Veritabanı, performansı, güvenliği, ölçeklenebilirliği ve açık kaynak avantajıyla web uygulamalarının veritabanı ihtiyaçlarını etkili bir şekilde karşılayan bir çözümdür.Aşağıda detaylı şekilde anlatılmıştır.

5.5.1 MySQL Veritabanının Özellikleri ve Avantajları

- Veritabanı Yönetimi: : MySQL, kullanıcı dostu bir arayüzle karmaşık veritabanı yapılarının yönetilmesine olanak tanır. Veritabanı oluşturulması, tabloların tanımlanması, veri eklenmesi ve sorgulama gibi işlemler kolaylıkla gerçekleştirilebilmektedir. Bu sayede kullanıcılar, veritabanı yönetimi süreçlerinde etkin bir şekilde rol alabilmektedirler.

- Veri Tutma ve İşleme: MySQL, farklı veri türlerini (sayılar, metinler, tarihler vb.) destekleyerek kullanıcılara geniş bir veri tutma ve işleme imkanı sunulur. Bu veriler üzerinde veri ekleme, güncelleme, silme, sıralama ve filtreleme gibi çeşitli işlemler kolaylıkla gerçekleştirilebilir. Bu sayede kullanıcılar, verileriyle etkileşimde bulunurken esneklik ve işlevsellik sağlanır.
- Veritabanı Güvenliği: MySQL, veritabanı güvenliğini sağlamak için kullanıcı yetkilendirme ve erişim kontrolü mekanizmalarıyla donatılmıştır. Kullanıcılar, belirli işlemleri gerçekleştirme yetkilerini belirleyebilir ve veri güvenliğini korumak için gerekli önlemleri alabilir. Ayrıca, veri şifreleme yöntemleri kullanarak verilerinizi güvende tutmanız da mümkündür
- Açık Kaynaklı: MySQL, açık kaynaklı bir veritabanıdır. Kullanıcılar, kaynak koduna erişebilir ve ihtiyaçlarına göre özelleştirme yapabilir. Geniş bir kullanıcı topluluğu, destek ve kaynak paylaşımı açısından avantaj sağlar.

5.5.2 Kullanım Alanları

MySQL, geniş ölçekteki web uygulamaları ve büyük sistemlerde yaygın olarak kullanılan etkili bir veritabanı çözümüdür. Verilerin etkin bir şekilde saklanmasını ve yönetilmesini sağlayarak projelerin gereksinimlerini karşılar. Güçlü ve esnek yapısıyla veritabanı yönetimi için tercih edilen MySQL, veri tabanlarının güvenli ve verimli bir şekilde işlenmesini sağlar.

6 PROJE'NİN İŞLEYİŞİ

6.1 Arayüz Kısmı

6.1.1 Template Hazırlığı

Proje için uygun bir template seçilmiş ve projenin gereksinimlerine uygun hale getirilmiştir. Template içerisinde yer alan gereksiz kısımlar temizlenerek sağ menü bölümü oluşturulmuştur. Bu sayede kullanıcılar, sağ menü üzerinden kolayca erişebilecekleri Dashboard, görevler (Ping, Theharvester), sistem logları ve Profil gibi önemli bölümlere ulaşabileceklerdir. Bu düzenlemeler, arayüzün daha kullanıcı dostu ve projeye özgü hale getirilmesini sağlamaktadır.

6.1.2 Kullanıcı Girişi

Projeyi ilk açtığınızda, kullanıcı login sayfasıyla karşılaşacaksınız. Bu sayfada kimlik bilgilerinizi girdikten sonra doğrulama işlemi gerçekleştirilir ve başarılı bir şekilde doğrulandığınızda diğer sayfalara erişim sağlanır. Bu sayede projenin güvenliği ve kullanıcı gizliliği sağlanırken, yetkilendirme mekanizması sayesinde sadece yetkili kullanıcılar projenin içeriğine erişebilir. Bu işlem, kullanıcıların projenin sunduğu özelliklere erişebilmeleri ve projeyi kullanmaya başlayabilmeleri için önemli bir adımdır.

6.1.3 Kontrol Paneli

Giriş yaptıktan sonra karşılaşacağınız ilk sayfa Kontrol Paneli olacaktır. Kontrol paneli, kullanıcılara projede gerçekleştirilebilecek işlemleri açıklayan ve sunan bir sayfadır. Bu sayede kullanıcılar, projenin farklı özelliklerini keşfedebilir, verileri yönetebilir, raporları görüntüleyebilir veya diğer kullanıcılarla etkileşimde bulunabilir. Kontrol paneli, projenin merkezi bir noktasıdır ve kullanıcılara projenin sunmuş olduğu işlevselliği keşfetme ve kullanma imkanı sağlar. Bu sayede kullanıcılar projenin potansiyelini tam olarak kullanabilir ve projenin amaçlarına yönelik işlemleri gerçekleştirebilirler.

6.1.4 Görevler Bölümü

Sağ menüde bulunan Görevler bölümünden, kullanıcılar yeni görevler oluşturabilir ve önceden oluşturulmuş görevlere erişebilirler. Görevler bölümü, kullanıcılara farklı işlemleri gerçekleştirmek için bir arayüz sağlar. Örneğin, bir Ping görevi oluşturulduğunda, kullanıcılar konteynırların çıktılarına, kayıp yüzdesine, maksimum, minimum ve ortalama geri dönüş sürelerine erişebilirler. Bu bilgiler, görevin durumu ve performansı hakkında önemli bilgiler sunar. Kullanıcılar, görevlerin sonuçlarını inceleyebilir, performans analizleri yapabilir ve gerektiğinde görevleri düzenleyebilirler. Görevler bölümü, kullanıcıların projede yapılan işlemleri izleme ve yönetme sürecine katkı sağlar, verimliliği artırır ve kullanıcılara kontrol sağlar.

6.1.5 Sistem Logları

Sağ menüde bulunan Sistem Logları bölümü, kullanıcılara log kayıtlarını görüntüleme imkanı sunar. Bu log kayıtları, kullanıcının hangi görevleri ne zaman çalıştırdığını, görevlerin tamamlanma sürelerini, başarı durumunu ve oturum açma/kapatma bilgilerini içerir. Sistem Logları bölümü, kullanıcılara gerçek zamanlı olarak yapılan işlemleri takip etme ve denetleme yeteneği sağlar. Bu sayede kullanıcılar, projenin geçmişinde gerçekleşen işlemleri izleyebilir, performans analizleri yapabilir ve gerektiğinde sorun giderme yapabilir. Sistem Logları bölümü, kullanıcılara projenin izlenebilirliğini artırırken, işlemlerin güvenliğini ve bütünlüğünü de sağlar.

6.1.6 Hesabım Bölümü

Hesabım bölümü, kullanıcılara profil düzenlemelerini gerçekleştirme imkanı sunar. Bu bölümde kullanıcılar, profil bilgilerini güncelleyebilir, şifrelerini değiştirebilir, iletişim tercihlerini yönetebilir ve gizlilik ayarlarını düzenleyebilir. Profil düzenlemeleri sayesinde kullanıcılar, kendi hesaplarını kişiselleştirme ve özelleştirme imkanına sahip olurlar. Bu şekilde, kullanıcılar projedeki profil bilgilerini istedikleri şekilde yönetebilir ve hesaplarını kendi ihtiyaçlarına göre ayarlayabilirler. Hesabım bölümü, kullanıcılara projedeki kişisel bilgilerini güncelleme ve yönetme kolaylığı sağlar, böylece kullanıcılar projede daha iyi bir kullanıcı deneyimi yaşayabilirler.

6.2 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.

6.2.1 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.

6.2.2 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

6.3 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.

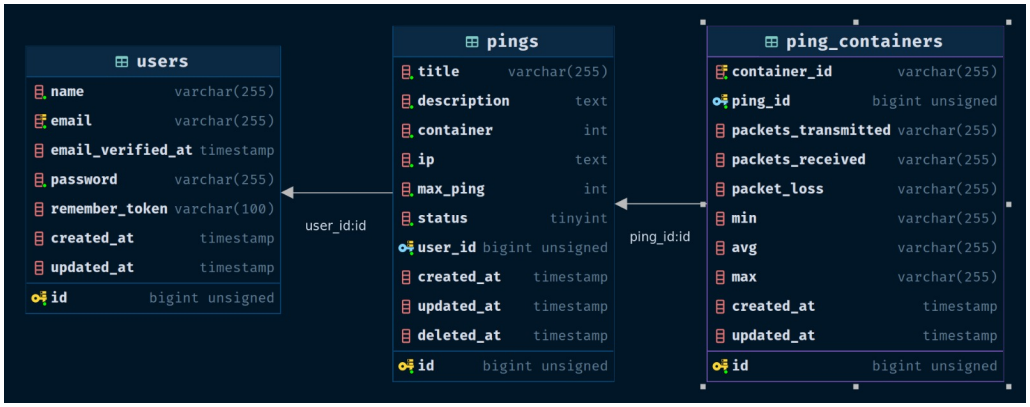
"users" tablosu, kullanıcıların temel bilgilerini içerir ve her kullanıcıya birincil anahtar (ID) ile benzersiz bir kimlik atar. Ayrıca kullanıcı adı, e-posta adresi ve diğer ilgili bilgiler gibi sütunlar da içerir.

"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 6.1: Sql ping tabloları

7 SONUÇLAR VE ÖNERİLER

Docker projesinde büyük çaplı bir projeye ihtiyaç duyulmadığı için Kubernetes veya Swarm gibi Docker'in oluşturduğu yönetim araçlarını kullanmaya gerek yoktur. Docker, projesinde ihtiyaçları karşılamak için yeterli olacaktır. Bu nedenle, projenin ölçeği ve karmaşıklığı dikkate alındığında, Docker'in hafif ve taşınabilir konteynerler sağlaması, izole çalışma ortamı sunması ve uygulama dağıtımını kolaylaştırması avantajlıdır. Bu sayede projedeki gereksinimleri etkili bir şekilde yönetebilir ve geliştirilebilir.

Proje büyüdükçe ve veritabanı gereksinimleri karmaşıklaştıkça, MySQL yerine PostgreSQL tercih edilebilir. PostgreSQL, karmaşık veri yapıları ve büyük veri hacimlerini etkili bir şekilde işleyebilme özelliğiyle öne çıkar. Ayrıca veri bütünlüğü ve güvenlik konularında da güçlü bir seçenektir. PostgreSQL'in ölçeklenebilirlik yetenekleri, büyük ölçekli sistemlerde yaygın olarak kullanılmasını sağlar.

8 EKLER

KAYNAKLAR

[1] Brogi,A.,Pahl,C.,andSoldani,J.2020.Onenhancingtheorchestrationofmulti-containerdockerapplications.InAdvancesinService-OrientedandCloudComputing:WorkshopsofESOCC2018,Como,Italy,September12âĂŞ14,2018,RevisedSelectedPapers,volume7,pages21âĂŞ33.SpringerInternationalPublishing. [Ziyaret Tarihi: 30 Mayıs 2022]

[2] Docker2023.Dockeroverview. [Ziyaret Tarihi: 12 Mart 2023]

[3] Ibrahim,M.H.,Sayagh,M.,andHassan,A.E.2021.Astudyofhowdockercomposeisusedtocomposemulti-componentsystems.EmpiricalSoftwareEngineering,26:1-27 [Ziyaret Tarihi: 6 Nisan 2022]

[4] Sharma,V.,Saxena,H.K.,andSingh,A.K.(2020).Docker3formulti-containerswebapplication.In20202ndInternationalConferenceonInnovativeMechanismsforIndustryApplications(ICIMIA),pages589âĂŞ592.IEEE. [Ziyaret Tarihi: 6 Nisan 2022]

[5] McKendrick,R.(2020).MasteringDocker:EnhanceyourcontainerizationandDevOpsskillstodeliverproduction-readyapplications.PacktPublishingLtd. [Ziyaret Tarihi: 6 Nisan 2022]

[6] Warrier,A.(2020).Containersvsvirtualmachines(vms). [Ziyaret Tarihi: 16 Mart 2023.]

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

İŞ DENEYİMLERİ

Yıl	**:**
Kurum	**:**
Stajlar	**:**

İLGİ ALANLARI:

YABANCI DİLLER: Ingilizce(B1)

BELİRTMEK İSTEDİĐİNİZ DİĞER ÖZELLİKLER:

ÖZGEÇMİŞ

KİŞİSEL BELGELER

Adı Soyadı : IBRAHIM KHALIL ATTEIB 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 :

Kurum :

Stajlar :

İLGİ ALANLARI: Yapay Zeka, Web Uygulama : PHP Laravel, Vue.js

YABANCI DİLLER: Fransizca (C2), Arapça(C1), ˙Ingilizce(A2)

BELİRTMEK İSTEDİĐİNİZ DİĞER ÖZELLİKLER: