



**T.C.**

**BİLECİK ŞEYH EDEBALI ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**Docker ile çoklu konteynerlar kullanarak paralel iş yapan sistem tasarımı**

**Aleyna ÇELİK**

**İBRAHİM KHALİL ATTEIB YACOUB**

**BİTİRME ÇALIŞMASI**

**DANIŞMANI : Prof. Dr. Cihan KARAKUZU**

**Dr. Öğr. Üyesi Burakhan ÇUBUKÇU**

**BİLECİK**

**13 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**  
**13 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: 13 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, 26 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, 26 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**

13 Mayıs 2023

# İÇİNDEKİLER

<b>ÖNSÖZ</b>	<b>v</b>
<b>ŞEKİLLER TABLOSU</b>	<b>vii</b>
<b>1 GİRİŞ</b>	<b>1</b>
<b>2 PROJE TANITIMI</b>	<b>4</b>
<b>3 KULLANILAN YAZILIMLAR VE YÖNTEMLER</b>	<b>6</b>
3.1 Web Uygulama Geliştirmek İçin . . . . .	6
3.1.1 Laravel MVC . . . . .	6
3.2 Konteynerize Etmek İçin . . . . .	8
3.2.1 Docker . . . . .	8
3.3 Veri Tabanı İçin . . . . .	11
3.3.1 MYSQL . . . . .	11
<b>4 PROJE’NİN İŞLEYİŞİ</b>	<b>14</b>
4.1 Arayüz Kısmı . . . . .	14
4.2 Arka paln işleyişi . . . . .	14
4.3 Veritabanı kısmı Tabloların oluşturulması . . . . .	14
<b>5 SONUÇLAR VE ÖNERİLER</b>	<b>15</b>
<b>6 EKLER</b>	<b>16</b>
<b>KAYNAKLAR</b>	<b>17</b>
<b>ÖZGEÇMİŞ</b>	<b>18</b>

## ŞEKİLLER TABLOSU



# 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 PROJE TANITIMI

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.

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

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

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.

Özetlemek gerekirse, projemizde Docker kullanarak paralel işlem yapabilen bir sistem tasarlandı. Laravel'in MVC yapısını kullanarak esnek bir yapı oluşturuldu. Güvenlik önlemlerini alarak kullanıcıların ve verilerin güvenliğini sağlandı. Docker ile konteynerleme işlemlerini gerçekleştirerek uygulamayı optimize ettik ve performansı artırıldı. Arayüzde canlı ilerleme bilgilerini gösteren bir özellik ekleyerek kullanıcıların işlerin durumunu takip etmelerini kolaylaştırıldı.

Projemizin başarıyla tamamlanması için bir dizi adım takip edildi. İlk olarak, gereksinimlerimizi belirlendi ve Docker'ı seçtik çünkü konteynerleme teknolojisi olarak bize izolasyon ve esneklik sağlayan bir çözüm sunuyordu. Ardından, Laravel'in MVC yapı-

sını kullanarak arayüz ve backend arasında etkili bir iletişim ve veri akışı sağlandı.

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

Backend tarafında, güvenlik önlemleri almak için login sayfasını tasarlandı ve kullanıcıların kimlik doğrulama sürecini sağlamlaştırıldı. Bu sayede, yalnızca yetkilendirilmiş kullanıcıların sistemimizi kullanabilmesi sağlandı ve veri güvenliğini sağlandı.

Docker kullanarak konteynerleme işlemleri gerçekleştirildi. 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 eklendi. Örneğin, kullanıcılar işleri önceliklendirebilir ve farklı konteynerlarda çalışacak işlere öncelik atayabildi. Bu, daha hızlı tamamlanması gereken işlerin öncelikli olarak çalıştırılmasını sağladı ve genel performansı artırdı.

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

Projenin güvenliği için, izinsiz erişim girişimlerini engellemek için oturum açma sayfasına ek güvenlik önlemleri eklendi. Kullanıcıların güvenliği ve verilerin korunması en üst düzeyde önem taşıırken, sistemimize giriş yapmak isteyen herhangi bir yetkisiz kişinin engellenmesi sağlandı.

Performansı ve ölçeklenebilirliği daha da artırmak için, proje üzerinde sürekli iyileştirmeler ve optimizasyonlar gerçekleştirildi. Örneğin, konteynerler arasında yük dengelemesi yaparak işlerin daha dengeli bir şekilde dağıtılmasını sağlandı ve böylece sistem üzerindeki aşırı yüklenmeyi önlenildi.

Arayüzde canlı ilerleme bilgilerini gösteren bir özellik eklendi. Kullanıcılara toplam iş miktarını, tamamlanma yüzdesini ve en hızlı çalışan konteynerları göstererek işlerin durumunu takip etmelerini sağlandı. Bu, kullanıcıların proje ilerlemesini daha iyi yönetmelerini ve gerekirse müdahalede bulunmalarını sağlayan değerli bir geribildirim mekaniz-

ması sağlandı.

Son olarak, proje yönetimi için bir kontrol paneli eklendi. Bu panel, proje yöneticilerinin işlerin ilerlemesini takip etmelerini, konteyner performansını analiz etmelerini ve gerektiğinde müdahale etmelerini sağlandı. Kontrol paneli, genel bir bakış açısı sunarak proje yönetiminin daha etkin bir şekilde yapılmasına yardımcı oldu.

Tüm bu adımları takip ederek, Docker kullanarak paralel işlem yapabilen bir sistem tasarlandı. Projemiz, Laravel MVC yapısıyla esneklik sağladı, güvenliği ön planda tuttu, konteynerleme ile performansı artırdı ve kullanıcılarımıza canlı ilerleme bilgileri sunarak daha iyi bir deneyim sağladı.

## **3 KULLANILAN YAZILIMLAR VE YÖNTEMLER**

### **3.1 Web Uygulama Geliştirmek İçin**

#### **3.1.1 Laravel MVC**

Laravel, popüler bir PHP framework'üdür ve MVC (Model-View-Controller) mimarisini kullanır. MVC, uygulamanın farklı katmanlarını birbirinden ayıran ve kodu düzenli, okunabilir ve bakımı kolay hale getiren bir tasarım desendir. Laravel, bu deseni kullanarak uygulama geliştirmeyi kolaylaştırır.

Laravel MVC yapısının temel bileşenleri:

- **Model:** Model, uygulamanın veri katmanını temsil eder. Veritabanı ile etkileşimde bulunur, verileri alır, günceller ve saklar. Model, veritabanı tablolarını temsil eden sınıflardan oluşur. Laravel'de model sınıfları, genellikle veritabanı tablolarının adlarıyla eşleşir ve veri manipülasyonu için işlevler sağlarlar. Model, veriye erişim, sorgulama yapma, ilişkili veri işleme gibi görevleri üstlenir.
- **View:** View, kullanıcı arayüzünü temsil eder. Kullanıcıya sunulan bilgilerin nasıl görüntüleneceğini belirler. HTML, CSS ve bazen PHP kodları içeren şablon dosyalarından oluşur. View, kullanıcıdan gelen istekleri işler, gerekli verileri alır ve düzenli bir şekilde sunar. Laravel'de Blade isimli bir template motoru kullanıla-

rak view'lar oluşturulur. Blade, basit ve esnek bir yapıya sahiptir ve kod tekrarını azaltmak için özelleştirilmiş dil yapısı sunar.

- **Controller:** Controller, model ve view arasındaki arabirimdir. Kullanıcıdan gelen istekleri alır, modeli kullanarak gerekli verileri işler, view'a iletmek üzere verileri hazırlar. Controller, uygulamanın mantıksal işlevlerini yönetir. Laravel'de controller sınıfları, istemci tarafından yapılan isteklerin yönlendirilmesini ve işlenmesini sağlar. Controller, model ve view arasındaki iletişimi sağlayarak uygulamanın nasıl tepki verdiğini kontrol eder.
- **Route:** Route, gelen istekleri doğru controller'a yönlendiren URL yönlendirme sistemidir. Laravel'de route'lar, web.php veya api.php gibi dosyalarda tanımlanır. Route dosyasında URL'lerle ilgili işlemler, controller eşleştirmeleri ve istek metodları belirtilir. Bu sayede gelen talepler doğru controller'a yönlendirilir ve işlenir.

MVC yapısı, Laravel'de uygulama geliştirirken kodu düzenli ve modüler tutmayı sağlar. Her bileşen kendi sorumluluk alanına odaklanır ve birbirinden bağımsızdır. Bu sayede geliştirme süreci daha etkili ve yönetilebilir hale gelir. Aynı zamanda, MVC yapısı sayesinde iş mantığı, veritabanı etkileşimi ve kullanıcı arayüzü birbirinden ayrı tutulur. Bu da kodun daha yeniden kullanılabilir, test edilebilir ve bakımı kolay olmasını sağlar.

Laravel'de MVC yapısını kullanarak bir uygulama geliştirmek için aşağıdaki adımlar izlenebilir :

1. **Model oluşturma:** Veritabanı tablosuna karşılık gelecek bir model oluşturun. Bu model, veritabanı işlemleri için kullanacağınız sınıfı temsil eder. Laravel'de model sınıfları, app/Models dizini altında yer alır ve Eloquent ORM (Object-Relational Mapping) kullanılarak veritabanı işlemlerini gerçekleştirir.
2. **Controller oluşturma:** İstekleri işleyecek ve yönlendirecek bir controller oluşturun. Controller, kullanıcıdan gelen istekleri alır, gerekli işlemleri yapar ve sonuçları view'a iletmek üzere hazırlar. Controller, app/Http/Controllers dizini altında yer alır ve Controller sınıfından türetilir.
3. **Route tanımlama:** İstekleri doğru controller'a yönlendirmek için route'lar tanımlayın. Bu, URL'lerin belirli bir controller'a ve işlemine eşleştirilmesini sağlar. La-

ravel’de route’lar, routes/web.php veya routes/api.php gibi dosyalarda tanımlanır. Route tanımlarken HTTP metodu, URL ve ilgili controller metodu belirtilir.

4. View oluşturma: Kullanıcı arayüzünü temsil eden view’lar oluşturun. Laravel’de Blade template motoru kullanılarak view’lar oluşturulur. View’lar, resources/views dizini altında yer alır ve HTML, CSS ve Blade dilini içeren şablon dosyalarından oluşur. View, kullanıcıya sunulan bilgilerin nasıl gösterileceğini belirler.
5. Controller ve Model arasındaki iletişim: Controller, gerekli verileri model aracılığıyla alır veya günceller. Model, veritabanı işlemlerini gerçekleştirir ve gerekli verileri controller’a döndürür. Controller, aldığı verileri view’a ileterek kullanıcı arayüzünde gösterilmesini sağlar.

MVC yapısı sayesinde, Laravel’de uygulama geliştirmek daha düzenli, okunabilir ve bakımı kolay bir şekilde gerçekleştirilebilir. Her bileşenin belirli bir sorumluluğu olduğu için, kodun yeniden kullanılabilirliği ve test edilebilirliği artar. Ayrıca, geliştirme sürecinde hız ve verimlilik sağlayarak uygulamanın daha iyi ölçeklenebilmesini ve yönetilebilmesini sağlar.

## 3.2 Konteynerize Etmek İçin

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

Docker'ın çalışma süreci şu adımları içerir:

1. **Bir Docker imajı oluşturma:** Dockerfile kullanarak bir Docker imajı oluşturulur. Dockerfile'da, uygulamanın gereksinimleri, bağımlılıkları, çalışma zamanı ve diğer yapılandırmalar belirtilir.
2. **Docker imajının derlenmesi (build):** Dockerfile'ı kullanarak, Docker imajı oluşturulur. Bu adımda Docker, Dockerfile'da tanımlanan adımları takip eder, gerekli paketleri indirir, bağımlılıkları kurar ve uygulamayı imaj haline getirir.
3. **Docker konteynerinin çalıştırılması:** Oluşturulan Docker imajı temel alınarak Docker konteyneri çalıştırılır. Docker konteyneri, Docker imajının bir örneğidir ve çalışan bir uygulamayı temsil eder. Docker, konteynerin izole bir ortamda çalışmasını sağlar ve uygulamanın sistem üzerindeki kaynaklara erişmesini sınırlar.
4. **Konteynerler arası iletişim:** Docker, birden fazla konteynerin birlikte çalışmasını sağlar. Konteynerler arasındaki iletişim, Docker ağı ve bağlantı noktaları aracılığıyla gerçekleştirilir. Bu sayede, farklı konteynerler arasında veri ve kaynak paylaşımı yapılarak uygulamanın işlevselliği artırılabilir.
5. **Konteyner yönetimi:** Docker, konteynerlerin yönetimini kolaylaştırır. Konteynerleri başlatma, durdurma, yeniden başlatma, silme gibi işlemleri yapabilirsiniz. Ayrıca, konteynerlerin kaynak kullanımını izleyebilir, logları görüntüleyebilir ve güncelle-

meleri gerçekleştirebilirsiniz.

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.

### 3.3 Veri Tabanı İçin

#### 3.3.1 MYSQL

Mysql, PHP programlama dili ile MySQL veritabanıyla etkileşim sağlamak için kullanılan bir PHP eklentisidir. mysql (MySQL Improved) eklentisi, MySQL veritabanı sunucusuna bağlantı kurmayı, sorguları yürütmeyi, veri alma ve veri güncelleme gibi işlemleri gerçekleştirmeyi sağlar.

Mysql eklentisi, önceki MySQL eklentisi olan mysql'nin geliştirilmiş ve geliştirilmiş bir sürümüdür. mysql, MySQL veritabanı sunucusunun daha yeni özelliklerini destekler ve daha iyi performans sağlar. Ayrıca, , daha güvenli ve daha kolay kullanım imkanı sunar.

Mysql kullanarak MySQL veritabanıyla çalışmak için aşağıdaki adımlar izlenir :

1. Bağlantı kurma: **mysql\_connect** fonksiyonunu kullanarak MySQL veritabanı sunucusuna bir bağlantı kurabilirsiniz. Bu fonksiyon, sunucu adı, kullanıcı adı, parola ve veritabanı adı gibi bağlantı bilgilerini alır. Bağlantı başarılı bir şekilde kurulursa, bir bağlantı nesnesi döndürülür.
2. Sorguları yürütme: **mysql\_query** fonksiyonunu kullanarak SQL sorgularını yürütebilirsiniz. Bu fonksiyon, bağlantı nesnesi ve yürütülecek sorguyu alır. Sorgu başarılı bir şekilde yürütülürse, sonuç seti veya etkilenen satır sayısı gibi sonuçları döndürür.
3. Veri alma: **mysql\_fetch\_assoc**, **mysql\_fetch\_array** veya **mysql\_fetch\_row** gibi fonksiyonları kullanarak sonuç setinden veri alabilirsiniz. Bu fonksiyonlar, sonuç setindeki her bir satırı dizi veya ilişkisel dizi olarak döndürür.
4. Veri güncelleme: **mysql\_query** fonksiyonunu kullanarak UPDATE, INSERT veya DELETE gibi güncelleme işlemlerini gerçekleştirebilirsiniz. Bu tür bir sorgu yürütüldüğünde, etkilenen satır sayısı döndürülür.

5. Bağlantıyı kapatma: **mysql\_close** fonksiyonunu kullanarak MySQL bağlantısını kapatmanız önerilir. Bağlantı kapandıktan sonra, veritabanı sunucusu ile bağlantı sonlandırılır ve kaynaklar serbest bırakılır.

Mysql, daha güncel bir PHP eklentisi olduğu için, daha fazla özellik ve işlevsellik sunar. Örneğin, hazır ifadelerle sorgu yapma, işlemi yürütme, parametre bağlama, işlem geçmişini takip etme gibi gelişmiş özelliklere sahiptir. Mysql, MySQL veritabanıyla güvenli ve etkili bir şekilde etkileşim sağlamak için aşağıdaki özelliklere sahiptir:

- Hazır ifadelerle sorgu yapma: Mysql, hazır ifadeleri kullanarak sorguların daha güvenli bir şekilde yürütülmesini sağlar. Bu özellik, SQL enjeksiyon saldırılarına karşı koruma sağlar. Hazır ifadeler, parametrelerin değerlerini güvenli bir şekilde eklemek için kullanılır.
- İşlemi yürütme (Transaction): Mysql, işlemleri (transaction) destekler. İşlemler, birden çok sorgunun birleştirilerek atomik bir şekilde yürütülmesini sağlar. İşlem kullanılarak veritabanı üzerinde bir dizi sorguyu gerçekleştirirken, tüm sorguların başarılı bir şekilde tamamlanması veya hiçbiri tamamlanmaması prensibiyle çalışır.
- Parametre bağlama (Parameter Binding): Mysql, parametreleri sorguya bağlamak için kullanılır. Bu, sorguları daha verimli hale getirir ve veritabanı sunucusunda ön işlemci tarafından daha iyi optimize edilmesini sağlar. Parametre bağlama, sorguların daha hızlı çalışmasını ve tekrar kullanılabilirliğini artırır.
- İşlem geçmişini takip etme: Mysql, gerçekleştirilen sorguların ve işlemlerin geçmişini takip etmek için özel işlevler sunar. Bu, hata ayıklama ve sorun giderme süreçlerinde yardımcı olur. Örneğin, `mysql_error` fonksiyonu ile son gerçekleştirilen sorguyla ilgili hata mesajını alabiliriz.
- Önceden hazırlanmış bildirimler (Prepared Statements): Mysql, önceden hazırlanmış bildirimleri kullanarak sorguları daha verimli hale getirir. Bu, aynı sorgunun tekrar tekrar yürütülmesi durumunda performans avantajı sağlar. Önceden hazırlanmış bildirimler, sorguları oluştururken parametrelerin yerine geçecek yer tutucular kullanmayı sağlar.

Mysql, güvenlik, performans ve veri bütünlüğü açısından daha gelişmiş bir seçenek ol-

duđu için MySQL veritabanıyla çalışırken tercih edilen bir PHP eklentisidir. Veritabanı işlemleri sırasında güvenlik açıklarını azaltır, daha verimli sorgular yapmanızı sağlar ve verilerin doğru bir şekilde yönetilmesini sağlar.

## **4 PROJE’NİN İŞLEYİŞİ**

### **4.1 Arayüz Kısmı**

Bu projede Arayüz kısmı için işleyiş aşağıda belirtilmiştir. İsteklere uygun bir template bulundu ve template üzerinde projeye uygun hale getirildi. Gereksiz kısımlar temizlendi ve sağ menü bölümünde sırasıya Dashboard , görevler(Ping ,Theharvester), sistem logları ve Profil bölümleri oluşturuldu.

Proje ilk açıldığında kullanıcı login sayfası ile karşılaşacaktır.Kimlik bilgileri girilecektir.Doğrulama yaptıktan sonra diğer sayfalara erişebilecektir.Giriş yaptıktan sonra karşılaşılabilecek ilk sayfa Kontrol Panelidir.Kontrol panelinde oluşabilecek işlemler yazılacak Sağ kısımda menü bulunabilmektedir.Burda görevler kısmından istediği zaman görev oluşturup önceden oluşturulan o görevle ilgili işlemleri de görebilir.Bir ping görevi açıldığında oluşan konyetnerlar bu konteynarların çıktılarını , kayıp yüzdesi ,pingin maximum geri dönüş süresi ,minimum geri dönüş süresi ve ortalama milisaniye geri dönüş süresi bulunmaktadır .Yine sağ menü kısmında sistem log kayıtları da görüntülenebilir .Bu log kayıtları altına ise kullanıcı Kim hangi görevi ne zaman çalıştırmış?, Görev kaç saniyede tamamlanmış?,Görev başarılı mı başarısız mı?,Oturum açma ve kapatma sorularının cevabını da kolaylıkla görüntüleyebilecektir.

Hesabım kısmından profil ile ilgili düzenlemeler yapılabilir.

### **4.2 Arka plan işleyişi**

### **4.3 Veritabanı kısmı Tabloların oluşturulması**

## **5 SONUÇLAR VE ÖNERİLER**

## **6 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: