URL KISALTICI WEB UYGULAMASI

IEE RAPORU



Kubilay Birer

211307086

Bilişim Sistemleri Mühendisliği

kubilaybirer@hotmail.com

*Özet*— Bu projede, Django web çerçevesi kullanılarak geliştirilmiş bir URL kısaltma uygulaması ele alınmıştır. Uygulama, kullanıcıların uzun URL'lerini kısaltmalarına ve kısaltılmış URL'leri paylaşmalarına olanak tanır. Ayrıca, uygulama Docker ve Alpine Linux üzerinde çalışacak şekilde sanallaştırılmıştır. Proje, web uygulaması geliştirmenin yanı sıra, bu uygulamayı sanal bir makine içinde çalıştırma ve Docker konteynerına dağıtma süreçlerini içermektedir. URL kısaltma uygulaması, masaüstü ve sunucu sanallaştırma teknolojilerini kullanarak, esnek ve taşınabilir bir ortamda çalışabilir. Bu proje, Django, Docker, ve Alpine Linux gibi teknolojileri anlama, kullanma ve bu teknolojilerle entegrasyonu sağlama yeteneğini değerlendirir. Ayrıca, bulut bilişim ve sanallaştırma konularına dair temel bilgileri içermektedir.

Keywords: Django, URL Kısaltma , Docker , Sanallaştırma , Konteyner , Python , Alphine Linux , Web Geliştirme

# GİRİŞ

Bu proje raporu, günümüzdeki uygulama geliştirme süreçlerinde önemli bir rol oynayan modern web teknolojileri, sanallaştırma yöntemleri ve bulut bilişim ilkelerini bir araya getiren bir uygulamayı ele almaktadır. Projemizin merkezinde, kullanıcıların URL'lerini etkili bir şekilde kısaltmalarını ve paylaşmalarını sağlayan bir URL kısaltma uygulaması bulunmaktadır. Bu uygulama, Django web çerçevesi üzerinde geliştirilmiş ve Docker konteynerları ile taşınabilir hale getirilmiştir.

Django'nun sunduğu güçlü özellikler ve hızlı geliştirme imkanları, uygulamanın temelini oluştururken, Docker konteynerları sayesinde uygulamanın bağımsız bir şekilde çalışabilir, taşınabilir ve ölçeklenebilir olmasına odaklandık. Docker, sanallaştırma teknolojisinin sağladığı izolasyon ve kolay dağıtım avantajlarıyla, uygulamamızın farklı platformlarda sorunsuz bir şekilde çalışmasını sağlar.

Bu rapor, projenin nasıl başladığını, kullanılan teknolojileri, uygulamanın temel özelliklerini ve Docker üzerindeki dağıtım sürecini detaylı bir şekilde açıklamaktadır.

**Proje Senaryosu: URL Kısaltma Servisi Oluşturma**

Giriş:

Günümüzde internet kullanımı arttıkça, URL'lerin paylaşımı ve yönetimi önemli bir konu haline gelmiştir. Bu proje, kullanıcıların uzun URL'leri kısaltmalarına ve bu kısaltılmış URL'leri paylaşmalarına olanak tanıyan bir URL kısaltma servisi oluşturmayı amaçlamaktadır. Bu servis, özellikle sosyal medya paylaşımları ve iletişimde URL'leri daha kullanıcı dostu ve yönetilebilir hale getirmek için tasarlanmıştır.

Senaryo:

1. Ana Sayfa ve URL Kısaltma:

* Kullanıcılar, ana sayfada basit bir arayüzle karşılaşırlar.
* Kullanıcılar, kısaltmak istedikleri URL'yi sisteme girerler.
* Sistem, kısaltılmış bir URL oluşturur ve kullanıcıya gösterir.

1. Kısaltılmış URL Kullanımı:

* Kullanıcılar, kısaltılmış URL'yi paylaşabilirler.
* Paylaşılan kısaltılmış URL'yi tıklayan kullanıcılar, doğrudan orijinal uzun URL'ye yönlendirilirler.

1. Teknolojiler:

* Django (Web Çerçevesi)
* Docker (Konteynerizasyon)
* SQLite (Sanal Veritabanı)
* HTML, CSS, JavaScript (Arayüz)

Bu senaryo, kullanıcı dostu bir arayüze sahip, güvenli ve ölçeklenebilir bir URL kısaltma servisinin oluşturulmasını amaçlamaktadır. Projenin aşamaları arayüz

Kullanılan Teknolojilerin Genel Tanımı

Bu projede, güçlü ve esnek bir web uygulama geliştirmek için çeşitli modern teknolojiler kullanılmıştır. İşte projede temelini oluşturan ana teknoloji bileşenleri:

* Django

Django, Python tabanlı açık kaynaklı bir web çerçevesidir. Güçlü bir Model-View-Controller (MVC) mimarisi sunan Django, hızlı ve modüler web uygulamaları oluşturmak için kullanılır. Django, veritabanı yönetimi, URL yönlendirmesi, şablona dayalı görünümler ve bir dizi kullanışlı araç içerir.

* Docker

Docker, uygulamaları hafif, taşınabilir ve ölçeklenebilir konteynerlara paketleme ve dağıtma konusunda popüler bir platformdur. Docker, uygulamaların bağımlılıklarını izole etmek ve herhangi bir ortamda tutarlı bir şekilde çalışmasını sağlamak için kullanılır. Proje, Docker'ın sunduğu konteyner teknolojisinin avantajlarından yararlanarak geliştirilmiştir.

Bu temel teknoloji bileşenleri, projenin geliştirilmesi ve dağıtılması süreçlerinde kritik rol oynar. Django ile güçlü bir web çerçevesi, Docker ile taşınabilir konteynerler ve bir araya gelerek projenin başarılı bir şekilde hayata geçirilmesini sağlar.

Bu teknolojilerin birleşimi, hızlı geliştirme, kolay dağıtım ve ölçeklenebilirlik sağlayarak projenin başarıyla tamamlanmasına katkıda bulunur.er yerden erişilebilir olmasına vurgu yapmaktadır.

# BULUT BİLİŞİM VE SANALLAŞTIRMA TEKNOLOJİLERİ İLE İLGİLİ ELDE EDİLEN BİLGİLER

## BULUT BİLİŞİM

Bulut bilişim, internet üzerinden erişilebilen ve paylaşılan kaynaklara dayalı olarak hizmet sağlayan bir bilişim modelidir. Geleneksel yerel sunucu altyapılarına kıyasla daha esnek, ölçeklenebilir ve maliyet etkin bir çözüm sunar. Bulut bilişimde temel olarak üç hizmet modeli bulunur:

1. Altyapı Servisleri (IaaS - Infrastructure as a Service)

* IaaS, fiziksel donanım kaynaklarının sanal makineler, depolama ve ağ gibi hizmetlerle kullanıcıya sunulduğu bir modeldir.
* Kullanıcılar, bu hizmetler aracılığıyla kendi uygulama ortamlarını oluşturabilir ve yönetebilirler.
* Amazon Web Services (AWS), Microsoft Azure ve Google Cloud Platform (GCP) gibi büyük bulut sağlayıcıları IaaS hizmetleri sunar.

1. Platform Servisleri (PaaS - Platform as a Service):

* PaaS, uygulama geliştirme sürecini kolaylaştıran bir hizmet modelidir.
* Kullanıcılar, altyapıyı düşünmeden sadece uygulama geliştirmeye odaklanabilirler.
* Heroku, Google App Engine gibi platformlar PaaS hizmetleri sunan örneklerdir.

1. Yazılım Servisleri (SaaS - Software as a Service):

* SaaS, kullanıcılara genellikle bir tarayıcı üzerinden erişilen yazılımları sunar.
* Kullanıcılar, yazılımı kullanmak için herhangi bir yükleme veya yapılandırma yapmadan sadece bir abonelikle erişebilirler.
* Google Workspace, Microsoft 365 gibi bulut tabanlı ofis uygulamaları SaaS örneklerindendir.

Bulut bilişim, esnek kaynak kullanımı, hızlı dağıtım ve sürekli güncellemeler gibi avantajlarıyla günümüzde birçok organizasyon tarafından tercih edilmektedir.

## SANALLAŞTIRMA TEKNOLOJİLERİ

Sanallaştırma, fiziksel donanım kaynaklarının sanal ortamlara bölünmesini ve bu sanal ortamların bağımsız bir şekilde çalışmasını sağlayan bir teknolojidir. Sanallaştırma teknolojileri, özellikle kaynakların daha etkin kullanılması ve uygulamaların izolasyonu için önemlidir.

1. **Masaüstü Sanallaştırma (Desktop Virtualization):**

Kullanıcıların masaüstü ortamlarını sanal bir sunucu üzerinde çalıştırmalarını sağlayan bir teknolojidir. VMware Horizon, Microsoft Remote Desktop gibi çözümler masaüstü sanallaştırma sunan platformlardır.

1. **Uygulama Sanallaştırma (Application Virtualization):**

Uygulamaların bağımsız bir şekilde çalışmasını sağlayan bir teknolojidir. Uygulama sanallaştırma, uygulamaların birlikte çalışabilmesini ve çatışmaların önlenmesini sağlar. Docker, uygulama sanallaştırma için sıkça kullanılan bir platformdur.

1. **Sunucu Sanallaştırma (Server Virtualization):**

Fiziksel sunucuların birden çok sanal sunucuya bölünmesini sağlayan bir teknolojidir. Her bir sanal sunucu, bağımsız bir işletim sistemine sahip gibi davranabilir. VMware vSphere, Microsoft Hyper-V gibi platformlar sunucu sanallaştırma sağlar.

1. **Ağ Sanallaştırma (Network Virtualization):**

Ağ kaynaklarının sanal olarak oluşturulması ve yönetilmesini sağlayan bir teknolojidir. Ağ sanallaştırma, ağ topolojilerini daha esnek hale getirir ve uygulamalar arasındaki iletişimi iyileştirir. VMware NSX, Cisco ACI gibi çözümler ağ sanallaştırma sunar.

1. **Depolama Sanallaştırma (Storage Virtualization):**

Depolama kaynaklarının sanal ortamlar üzerinden yönetilmesini sağlayan bir teknolojidir. Depolama sanallaştırma, depolama sistemlerini daha esnek ve ölçeklenebilir hale getirir. VMware vSAN, Microsoft Storage Spaces gibi platformlar depolama sanallaştırma sunar.

Sanallaştırma teknolojileri, kaynakların daha etkin kullanılmasını, donanım bağımsızlığını ve uygulama izolasyonunu sağlayarak bilişim altyapılarını daha yönetilebilir hale getirir.

# TASARIM , Temel Özellikler ve Uygulama

**1. Tasarım**

**1.1. Mimari Tasarım**

Proje, Django web çerçevesi üzerine inşa edilmiş ve Model-View-Controller (MVC) mimarisini temel alarak geliştirilmiştir. Django'nun sunduğu güçlü ORM sistemi, veritabanı yönetimi için kullanılmıştır. Ayrıca, uygulamanın modüler yapısı, gelecekteki genişlemelere olanak tanıyan bir temel oluşturur.

**1.2. Konteynerizasyon ve Dağıtım**

Uygulama, konteyner teknolojisi kullanılarak paketlenmiş ve dağıtılmıştır. Konteynerizasyon, uygulamanın bağımlılıklarını ve çalışma ortamını bir konteyner içinde izole etme sürecidir. Bu projede Docker kullanılmıştır. Docker konteynerleri, uygulamanın herhangi bir ortamda tutarlı bir şekilde çalışmasını sağlar ve geliştirme, test ve üretim ortamları arasında sorunsuz bir taşınabilirlik sağlar.

Docker kullanılarak oluşturulan konteyner, uygulamanın tüm bağımlılıklarını (Python sürümü, Django çerçevesi, gerekli kütüphaneler) içerir. Ayrıca, uygulama kodu ve statik dosyalar da konteyner içinde bulunur.

**2. Temel Özellikler**

**2.1. Kısa URL Oluşturma**

Kullanıcılar, uzun URL'leri kısaltmak için uygulama üzerinden kolayca kısa URL'ler oluşturabilirler. Bu özellik, URL'lerin paylaşımını kolaylaştırır ve izlenme istatistikleri ile birlikte gelir.

**3. Uygulama**

**3.1. Çalışma Ortamı**

Uygulama, Docker konteyneri içinde geliştirme aşamasında yerel bilgisayarlar üzerinde çalıştırılmış ve test edilmiştir.

**3.2. Dağıtım**

Uygulama, belirlenen sunucu üzerine Docker konteyneri manuel olarak dağıtılmıştır. Bu dağıtım süreci, Docker konteynerinin sunucuya yüklenmesini, çalıştırılmasını ve gerektiğinde güncellenmesini içermektedir.

## Uygulama Aşaması

**1. Tek Kişilik Geliştirme ve Dinamikler**

HammyBurger projesini tek başıma geliştirme süreci, temel teknolojilerin (HTML, CSS, JavaScript) uyumlu bir şekilde entegre edilmesini gerektirdi. Bu süreç, özellikle responsive tasarımın sağlanması noktasında belirli zorluklarla karşılaştım.

**2. Temel Teknolojilerin Uyumu**

HTML, CSS ve JavaScript gibi temel teknolojilerin entegrasyonu, projenin geliştirilmesinde sorunsuz ilerledi. Ancak, Docker üzerinde çalışacak bir uygulama geliştirmek, bu temel teknolojileri bir araya getirerek uyumlu bir ortam sağlamayı gerektirdi.

**3. Docker ve Konteyner Sanallaştırma**

Proje geliştirme sürecinde, Docker konteynerları kullanılarak uygulama bir izole ortamda çalıştırılmış ve test edilmiştir. Docker, uygulamanın bağımlılıklarını, çevresel yapılandırmalarını ve çalışma zamanı ortamını içeren hafif, taşınabilir konteynerlara paketleme konusunda popüler bir teknolojidir.

Docker, her biri belirli bir görevi yerine getiren bağımsız konteynerler olarak adlandırılan sanal ortamları yönetir. Bu konteynerler, uygulamanın çalışması için gereken tüm bileşenleri içerir ve bu sayede uygulamanın farklı ortamlarda tutarlı bir şekilde çalışmasını sağlar.

Konteynerler, uygulamaların bağımlılıklarını izole etmek, dağıtımı kolaylaştırmak ve ölçeklenebilirliği artırmak için kullanılır. Proje geliştirilirken, Docker konteynerları sayesinde her bir bileşenin (veritabanı, uygulama sunucusu vb.) ayrı bir konteynerde çalışması sağlanmış ve bu sayede geliştirme ve test süreçleri daha yönetilebilir hale getirilmiştir.

Docker'ın sunduğu bu izolasyon, geliştiricilere lokal ortamlarında uygulamayı hızlıca başlatma, test etme ve dağıtma imkanı sağlamıştır. Ayrıca, uygulamanın çeşitli ortamlarda sorunsuz bir şekilde çalışabilmesi, Docker konteynerlarının taşınabilirliği sayesinde elde edilmiştir.

Docker, projenin lokal geliştirme ortamından üretim ortamına geçişini kolaylaştırdı. Konteynerler sayesinde uygulama, farklı sistemlerde ve altyapılarda sorunsuz bir şekilde çalışabilir hale geldi.

## Karşılaşılan Zorluklar ve Kazanımlar

Proje sürecinde Docker konteynerları kullanılarak uygulamanın geliştirilmesi, dağıtılması ve yönetilmesi bir dizi zorluk ve kazanımı beraberinde getirmiştir. İlk aşamada, uygulamanın konteynerlara izole edilmiş bir şekilde taşınması ve çalıştırılmasıyla ilgili olarak ortaya çıkan konfigürasyon zorlukları yaşandı. Ancak, Docker'ın sunduğu kapsamlı belgeler ve topluluk desteği sayesinde bu konudaki bilgi eksiklikleri aşılabildi.

Docker konteynerlarının kullanılmasıyla birlikte uygulamanın hızlı dağıtılabilirliği ve taşınabilirliği arttı. Ancak, bu avantajların yanında konteynerlar arası iletişim, ağ konfigürasyonları ve kaynak paylaşımı gibi konularda başlangıçta bazı zorluklar yaşandı. Bu noktada, Docker Compose ve Docker Networking gibi araçlar kullanılarak çözümler geliştirildi.

Özellikle uygulamanın farklı ortamlarda (development, staging, production) sorunsuz bir şekilde çalışabilmesi için Docker'ın sunduğu çok katmanlı yapıların doğru anlaşılması ve kullanılması gerekti. Bu sayede her bir ortam için uygun konfigürasyonların sağlanması ve uygulamanın tutarlı bir şekilde çalışması sağlandı.

Projemde Django ile geliştirdiğim URL kısaltıcı web uygulamasını Docker konteynerlerinde başarıyla çalıştırdıktan sonra, AWS Elastic Container hizmetini kullanarak bu konteynerleri dağıtmak istedim. Ancak bu süreçte birkaç zorlukla karşılaştım.

Repository Ekleme Sürecinde Sorunlar: AWS Elastic Container Registry'e (ECR) projemin Docker imajını yüklemek istediğimde, yerel Docker ortamımda başarılı olan imajın AWS üzerindeki ECR'ye yüklenmesinde sıkıntılar yaşadım. İmajın doğru şekilde etiketlenmesi ve ECR iletişiminde bazı yetki sorunlarıyla karşılaştım.

Task Definition Oluşturma Zorlukları: Oluşturduğum Docker imajını temel alarak AWS üzerinde Task Definition oluşturmada bazı uyumsuzluklar yaşadım. Özellikle, konteynerin konfigürasyonu, CPU ve bellek ayarları gibi noktalarda eksik ya da hatalı bilgiler verdiğimde Task Definition'ı oluşturamadım.

Cluster'da Task'ı Çalıştırma Engelleri: AWS üzerinde oluşturduğum Task Definition'ı belirlediğim Cluster'a ekledikten sonra, bu task'ı başlatırken ağ bağlantısı veya izinler gibi sorunlarla karşılaştım. Konteynerin çalıştırılmasını engelleyen ve detaylı hata mesajlarıyla sunulmayan problemlerle karşılaştım.

Bu zorluklar, projenin AWS Elastic Container'da dağıtımını tamamlamamı engelledi. Sorunlarla başa çıkmak için çeşitli AWS belgelerini, topluluk forumlarını ve teknik dokümantasyonları inceledim. Ancak belirli bir çözüme ulaşamadım ve projenin bu aşamasında ilerleme kaydedemedim.

Bu deneyimler, AWS Elastic Container gibi bulut tabanlı dağıtım hizmetlerinin karmaşıklığını anlamama ve bu tür platformlarda karşılaşılabilecek teknik zorlukları aşma konusundaki ihtiyacımı ortaya koydu.

Sonuç olarak, Docker konteynerları kullanılarak yapılan bu projede yaşanan zorluklar, sürekli iyileştirme ve öğrenme sürecine katkıda bulundu. Konteyner teknolojilerinin esnekliği, hızlı dağıtım imkanları ve izolasyon yetenekleri sayesinde uygulamanın geliştirilmesi ve yönetilmesi daha etkili hale geldi.

1. *Docker Konteyner ve Sürüm Bilgileri*

Bu bölümde, projenin Docker konteyner yapısı ve kullanılan sürümler hakkında detaylı bilgiler yer almaktadır.

* **Kullanılan Docker İmajı**

Django uygulaması için Docker konteynerı oluşturulurken, hafif ve verimli bir seçenek olan Alpine Linux tabanlı Python 3.11.4 imajı tercih edilmiştir. İşte bu tercihin nedenleri:

Yer Tasarrufu: Alpine Linux, diğer Linux dağıtımlarına göre daha düşük depolama alanı kullanır. Bu, Docker konteynerının boyutunu minimize eder ve daha hızlı dağıtım sağlar.

Düşük Maliyet: Alpine Linux, minimal bir yapıya sahiptir ve gereksinim duyduğu paketleri sadece içerir. Bu dağıtım, düşük maliyetli bulut hizmetlerinde ve kaynak kısıtlı ortamlarda ideal bir seçenektir.

* **Kullanılan Python Sürümü**

Python 3.11.4 sürümü, Django uygulamasının gereksinim duyduğu Python paketlerini ve özelliklerini desteklemek için kullanılmıştır. Bu sürüm, güncel ve sağlam bir Python versiyonunu temsil eder.

Bu sürüm seçimi, projenin ihtiyaçlarına uygun bir Python sürümünü kullanarak hem güvenilirliği hem de performansı artırmayı amaçlamaktadır.

Bu bilgiler, Docker konteynerının oluşturulma sürecindeki temel kararları ve tercihleri anlamak adına önemlidir.

# Sonuç

Bu projede, Django web çerçevesi, Docker konteynerları ve Alpine Linux gibi modern teknolojilerin entegrasyonuyla bir URL kısaltma uygulaması geliştirildi. Uygulama, uzun URL'leri etkili bir şekilde kısaltma ve paylaşma imkanı sunarken, Docker konteynerları sayesinde bağımsız, taşınabilir ve ölçeklenebilir bir yapıya kavuştu. Projenin başarılı olması için çeşitli zorluklarla karşılaşıldı. Özellikle, Docker konteynerlarının konfigürasyonu ve farklı ortamlarda uyumlu çalıştırılması başlangıçta teknik meydan okumalar sunsa da, bu zorluklar, sürekli öğrenme ve gelişme fırsatlarına dönüştü. Docker'ın esnekliği ve avantajları, uygulamanın hızlı bir şekilde geliştirilip dağıtılmasına olanak tanıdı. Projenin kazanımları arasında, Django ve Docker gibi teknolojilerle entegrasyon yeteneğinin artması, bulut bilişim ve sanallaştırma prensiplerinin uygulama içinde başarılı bir şekilde kullanılması bulunmaktadır. Bu proje, günümüzün bilişim trendlerine uygun bir uygulama geliştirme deneyimi sunarken, yaşanan zorluklar da projenin sürekli iyileştirme ve öğrenme fırsatlarına dönüşmesine katkı sağladı.

##### Kaynakça

1. Django. (s.a.). Django web çerçevesi. https://www.djangoproject.com/
2. Docker. (s.a.). Docker belgeleri. https://docs.docker.com/
3. W3Schools. (s.a.). HTML öğrenme kaynağı. https://www.w3schools.com/html/
4. Bootstrap. (s.a.). Bootstrap belgeleri. https://getbootstrap.com/docs/5.0/getting-started/introduction/
5. JavaScript MDN. (s.a.). JavaScript belgeleri. <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>
6. *What is Virtualization? | IBM*. (2023). Ibm.com. https://www.ibm.com/topics/virtualization
7. ‌ what-is-cloud-computing. (2023). Amazon Web Services, Inc. https://aws.amazon.com/tr/what-is-cloud-computing/