İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010
FEN BİLİMLERİ ENSTİTÜSÜ

Mikroservis Yönetim Sistemi

Yazılım Mühendisliği Ana Bilim Dalı

Dönem Projesi

Tunahan Demirkol

ORCID 0009-0006-0994-1175

Proje Danışmanı: Prof. Dr. Doğan Aydın

Ağustos 2024

Mikroservis Yönetim Sistemi

Öz

Bu çalışma, Flask mikro web çerçevesi ve Python kullanarak geliştirilmiş bir merkezi yönetim sistemi olan "Mikroservis Yönetim Sistemi "ni tanıtmaktadır. Proje, yerel ağ üzerinden erişilebilen kullanıcı ve konfigürasyon yönetimi, kayıt takibi ve istek sınırlandırma özelliklerini sunmaktadır. Kullanıcılar için CRUD (Oluşturma, Yazma, Güncelleme, Silme) işlemleri, günlük istek limitleri ve IP (Internet protokol) tabanlı erişim kontrolü sağlanmaktadır. Flask-Admin modülü ile kullanıcı dostu bir arayüz oluşturulmuştur.

Bu çalışma, küçük ve orta ölçekli işletmelerin merkezi yönetim sistemi ihtiyacını karşılamakta, kullanıcı ve sistem yönetimini kolaylaştırarak operasyonel verimliliği artırmayı hedeflemektedir. Uygulama, Render platformu üzerinde uygulama dağıtılmıştır. Bu çalışma, merkezi yönetim sistemlerinin etkinliğini ve güvenliğini artırmak için önemli bulgular sunmaktadır.

Anahtar Sözcükler: Python, flask, merkezi yönetim sistemi, kullanıcı yönetimi, yerel ağ, render

Microservice Management System

Abstract

This study introduces the "Microservice Management System," a centralized management system developed using the Flask micro web framework and Python. The project provides user and configuration management, log tracking, and rate limiting features accessible over a local network. It offers CRUD (Create, Read, Update, Delete) operations, daily request limits, and IP (Internet Protocol)-based access control for users. A user-friendly interface is created with the Flask-Admin module. This project aims to meet the centralized management system needs of small and medium-sized enterprises, enhancing operational efficiency by simplifying user and system management. The application is deployed on the Render platform. This study presents significant findings to enhance the effectiveness and security of centralized management systems.

Keywords: Python, flask, centralized management system, user management, local network, render

İçindekiler

Şekiller Listesi

Bölüm 1

Giriş

Bu çalışma, Flask mikro web çerçevesi ve Python programlama dili kullanılarak geliştirilmiş bir merkezi yönetim sistemi olan "Mikroservis Yönetim Sistemi"ni tanıtmaktadır. Çalışmanın amacı, küçük ve orta ölçekli işletmelerin kullanıcı ve konfigürasyon yönetimi, kayıt takibi, istek limiti ve yetkilendirme gibi temel yönetim ihtiyaçlarını karşılayan, kullanıcı dostu ve güvenli bir sistem geliştirmektir. Bu sistem, kullanıcıların kolaylıkla erişebileceği, yerel ağ üzerinden erişilebilen bir yapı sunarak, işletmelerin operasyonel verimliliğini artırmayı hedeflemektedir.

1.1 Konunun Önemi ve Seçilme Nedeni

Günümüz işletmelerinde merkezi bir yönetim sistemi, verimlilik ve güvenlik açısından kritik bir rol oynamaktadır. Özellikle küçük ve orta ölçekli işletmeler, bu tür sistemlere olan ihtiyaçlarını uygun maliyetli ve esnek çözümlerle karşılamak istemektedir. Bu bağlamda, Flask ve Python kullanılarak geliştirilen "Mikroservis Yönetim Sistemi", kullanıcı yönetimi, konfigürasyon yönetimi, kayıt takibi ve istek sınırlama gibi özellikleriyle bu ihtiyaca yanıt vermektedir. Çalışmanın önemi, işletmelerin verimliliğini artırırken, aynı zamanda güvenli bir yönetim altyapısı sunmasından kaynaklanmaktadır.

1.2 Ön Bilgiler ve Literatür Değerlendirmesi

Mikroservis mimarisi, büyük ve karmaşık yazılım sistemlerinin yönetimi ve geliştirilmesi sürecinde monolitik yapılardan daha esnek ve yönetilebilir bir yapı sunarak önemli bir paradigma değişikliği yaratmıştır. Mikroservislerin modüler yapısı, her bir servis için bağımsız geliştirme ve dağıtım süreçlerine olanak tanıyarak, hata izolasyonu ve esneklik gibi avantajlar sağlamaktadır (Fowler, 2014; Newman, 2015). Bu yapı, günümüzün değişken iş ihtiyaçlarına daha hızlı uyum sağlamak için ideal bir çözüm sunar.

Mikroservis mimarisi, modern yazılım geliştirme süreçlerinde esnekliği, ölçeklenebilirliği ve bağımsız dağıtımı ile ön plana çıkmaktadır. Bu mimari, uygulamaların çeşitli bağımsız servisler olarak geliştirilip yönetilmesine olanak tanır, bu da yazılım geliştirme süreçlerini hızlandırırken, hataların izole edilmesini ve bağımsız olarak düzeltilmesini kolaylaştırır (Nadareishvili vd., 2016). Mikroservisler, bir API Gateway aracılığıyla dış dünyaya erişim sağlar; bu, farklı mikroservislerin birleşik bir API üzerinden erişilmesini mümkün kılarak, uygulama içinde farklı görevlerin ve işlemlerin kolayca yönetilmesini sağlar (Richardson, 2018).

Flask, Python tabanlı bir mikro web çerçevesi olarak, bu tür sistemlerin geliştirilmesi için yaygın olarak tercih edilmektedir. Minimalist tasarımı ve genişletilebilir yapısıyla, özellikle hızlı prototip geliştirme ve mikroservis mimarileri için uygundur (Grinberg, 2018). Literatürde, Flask kullanılarak geliştirilen sistemlerin hem küçük ölçekli uygulamalarda hem de büyük ölçekli mikroservis tabanlı yapılarda başarıyla uygulandığı çeşitli örnekler bulunmaktadır (Sellers, 2020). Bu çalışmalar, Flask'in sağladığı esneklik ve modülerlik avantajlarının, yönetim sistemlerinin geliştirilmesinde önemli bir rol oynadığını göstermektedir.

API Gateway, mikroservis mimarisi içinde kritik bir bileşen olarak hizmet verir. Bu bileşen, istemcilerin mikroservislerle etkileşim kurmasını sağlayan bir giriş noktası olarak işlev görür ve genellikle yük dengeleme, kimlik doğrulama, yetkilendirme, izleme ve kayıt altına alma gibi görevleri de üstlenir. Bu proje de benzer şekilde, bir merkezi yönetim sistemi üzerinden kullanıcı ve konfigürasyon yönetimi, kayıt takibi ve istek limitleme gibi fonksiyonları yöneterek API Gateway ile paralellikler göstermektedir (Balalaie vd., 2016).

Mevcut literatürde, mikroservis mimarisi ile geliştirilen uygulamaların, özellikle dağıtık sistemlerdeki esneklik, ölçeklenebilirlik ve bağımsız geliştirme süreçleri üzerindeki olumlu etkileri vurgulanmaktadır (Dragoni vd., 2017). Özellikle, mikroservislerin yönetimi ve izlenmesi süreçlerinde karşılaşılan zorluklar, bu mimarinin en kritik unsurlarından biri olarak kabul edilir. Bu bağlamda, bu çalışmada geliştirilen "Mikroservis Yönetim Sistemi", Flask ile bütünleşmiş kayıt takibi, istek sınırlama ve kullanıcı yönetimi özellikleri sunarak, bu alandaki boşlukları doldurmayı hedeflemektedir.

Ayrıca, veri tabanı yönetiminde PostgreSQL'in kullanımı, mikroservis mimarisinin güvenli ve ölçeklenebilir veri yönetimi gereksinimlerini karşılamak için stratejik bir seçim olmuştur. PostgreSQL, özellikle büyük veri yönetimi ve karmaşık sorgu işlemleri için sağladığı performans ve güvenlik özellikleri ile ön plana çıkmaktadır (Gulutzan & Pelzer, 2002). Bu da projede verilerin güvenli bir şekilde saklanmasını ve hızlı erişim sağlanmasını mümkün kılmıştır.

Projenin dağıtılması için Render platformunun tercih edilmesi, bulut tabanlı çözümlerin sağladığı esneklik ve maliyet etkinliğinden yararlanmayı mümkün kılmıştır. Render, Flask uygulamalarının hızlı ve güvenilir bir şekilde dağıtılmasına olanak tanırken, aynı zamanda sürekli entegrasyon ve teslimat süreçlerini de kolaylaştırmaktadır.

1.3 Kullanılan Yöntemler ve Seçilme Nedenleri

Bu çalışmada, Flask mikro web çerçevesi tercih edilmiştir çünkü bu çerçeve, hızlı ve esnek bir geliştirme süreci sunmaktadır. Kullanıcı yönetimi, kayıt takibi ve istek limitleme gibi özelliklerin etkin bir şekilde entegre edilmesi, bu çerçevenin sağladığı modüler yapılar sayesinde mümkün olmuştur. Ayrıca, veri tabanı yönetimi için PostgreSQL tercih edilmiş ve bu veri tabanı ile güvenli ve ölçeklenebilir bir yapı oluşturulmuştur.

Uygulamanın Render platformu üzerinde dağıtılması, projenin ölçeklenebilirliği ve erişilebilirliği açısından stratejik bir karar olmuştur. Bu platform, Flask uygulamalarının hızlı bir şekilde dağıtılmasına olanak tanır ve aynı zamanda maliyeti için etkin bir çözüm sunar.

Bölüm 2

Materyal ve Metot

Bu bölümde, "Mikroservis Yönetim Sistemi" projesinin geliştirilmesi sırasında kullanılan teknikler, araçlar ve yöntemler ayrıntılı bir şekilde ele alınmaktadır. Projenin amacı doğrultusunda belirlenen hedeflere ulaşmak için hangi verilerin, hangi kaynaklardan, hangi tekniklerle ve araçlarla toplandığı, ayrıca bu verilerin nasıl analiz edilip yorumlandığı açıklanmaktadır.

2.1 Materyaller

2.1.1 Yazılım Araçları:

Proje, Python programlama dili ve Flask mikro web çerçevesi kullanılarak geliştirilmiştir. Flask, hafif ve esnek yapısı nedeniyle tercih edilmiştir. Projede kullanılan diğer yazılım araçları şu şekildedir:

- Flask-Admin: Yönetim paneli arayüzünü oluşturmak için kullanılmıştır.
- PostgreSQL: Veri tabanı yönetim sistemi olarak seçilmiştir.
- Render: Uygulamanın bulut ortamında dağıtımı ve barındırılması için tercih edilmiştir.
- Matplotlib: Kayıt verilerinin grafiksel olarak görselleştirilmesi için kullanılmıştır.
- CSS (Cascading Style Sheets): Web sayfalarının stilini belirlemek için kullanılmıştır.
- HTML (HyperText Markup Language): Web sayfalarının yapısını oluşturmak için kullanılmıştır.

2.1.2 Veri Tabanı Yapısı

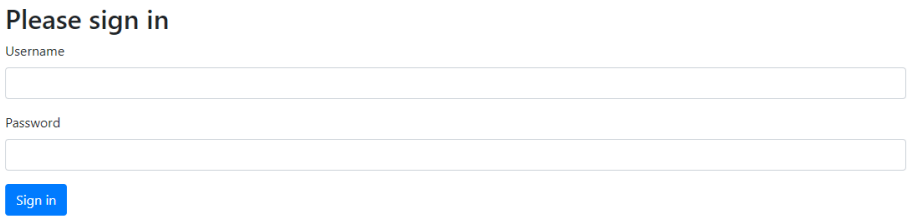
Projede, kullanıcı bilgileri, konfigürasyon ayarları ve istek kayıtlarını yönetmek için PostgreSQL veri tabanı kullanılmıştır. Veri tabanı yapısı şu şekildedir:

- Kullanıcı Tablosu: Kullanıcı adı, şifre, günlük istek limiti ve istek sayacı gibi bilgileri içerir.
- Konfigürasyon Tablosu: IP (Internet Protocol) adresleri ve açıklamalar içerir.
- Kayıt Tablosu: Servis adı, istek yöntemi, cevap durumu, kullanıcı adı ve zaman damgası bilgilerini saklar.

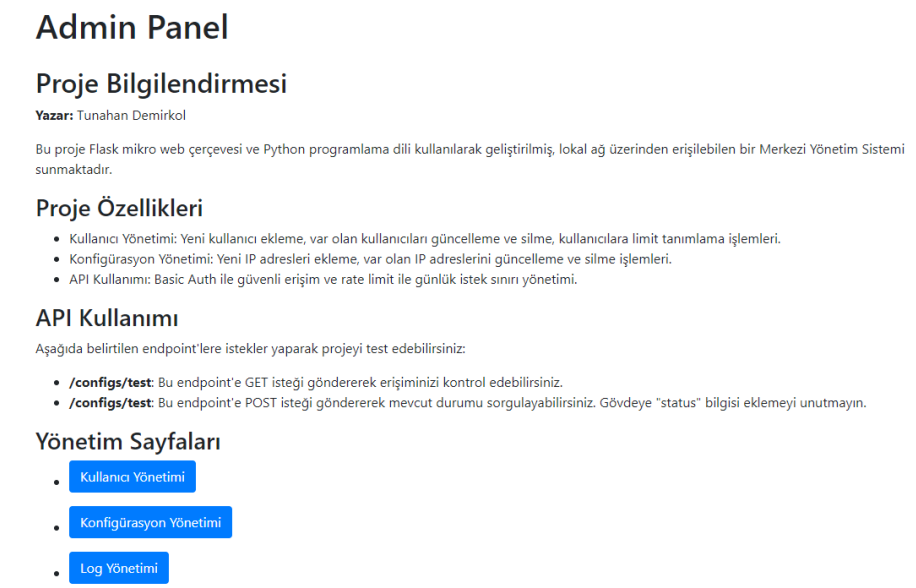
2.2 Metotlar

2.2.1 Kullanıcı Yönetimi

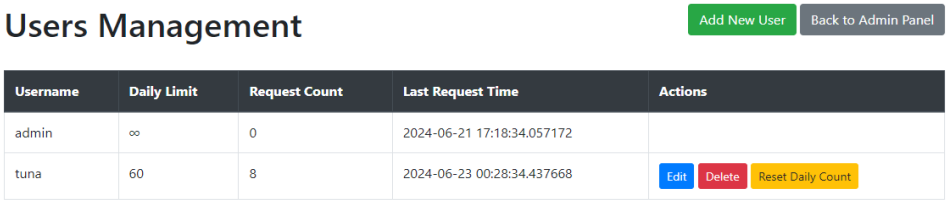
Kullanıcı yönetimi için Flask-Admin eklentisi kullanılmıştır. Bu eklenti, kullanıcıların CRUD (Oluşturma, Okuma, Güncelleme, Silme) işlemlerini kolaylıkla gerçekleştirmesine olanak tanıyan kullanıcı dostu bir arayüz sağlar. Kullanıcı verileri PostgreSQL veri tabanında saklanmakta ve Flask ile işlenmektedir.



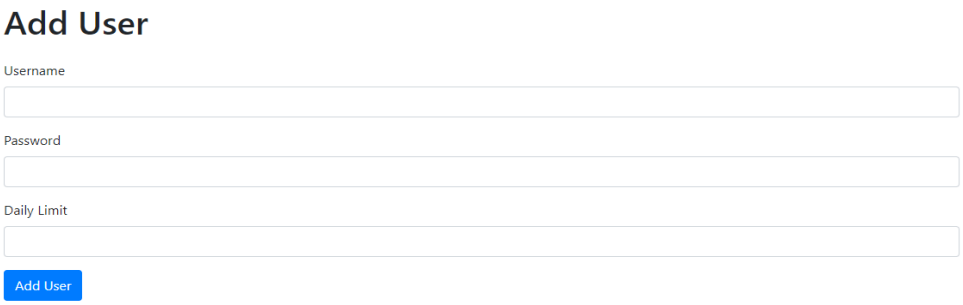
Şekil 2.1: Yönetici giriş ekranı



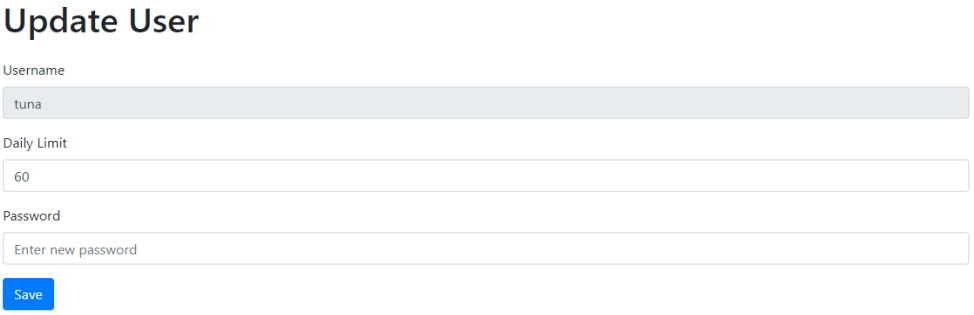
Şekil 2.2: Yönetici paneli ana sayfası



Şekil 2.3: Kullanıcı yönetim paneli



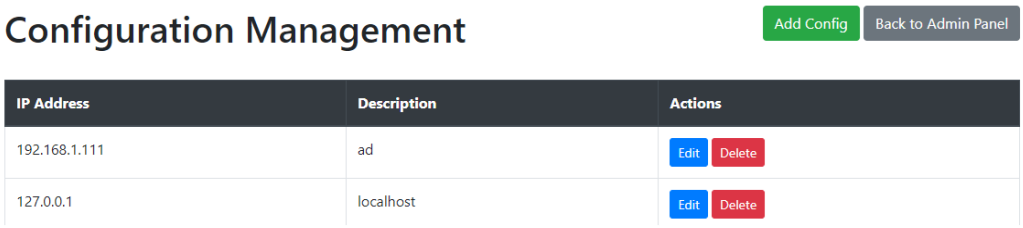
Şekil 2.4: Kullanıcı ekleme paneli



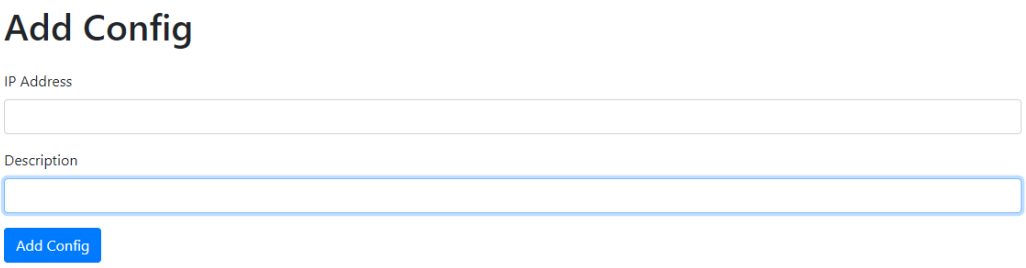
Şekil 2.5: Kullanıcı güncelleme paneli

2.2.2 Konfigürasyon Yönetimi:

Konfigürasyon yönetimi, IP (Internet Protokol) adresleri temelinde erişim kontrolü sağlamak için tasarlanmıştır. Sisteme eklenen IP (Internet Protokol) adresleri belirli servislere erişim izni verir. Yeni IP (Internet Protokol) adreslerinin eklenmesi, mevcut IP (Internet Protokol) adreslerinin güncellenmesi ve silinmesi işlemleri Flask-Admin arayüzü üzerinden gerçekleştirilir.



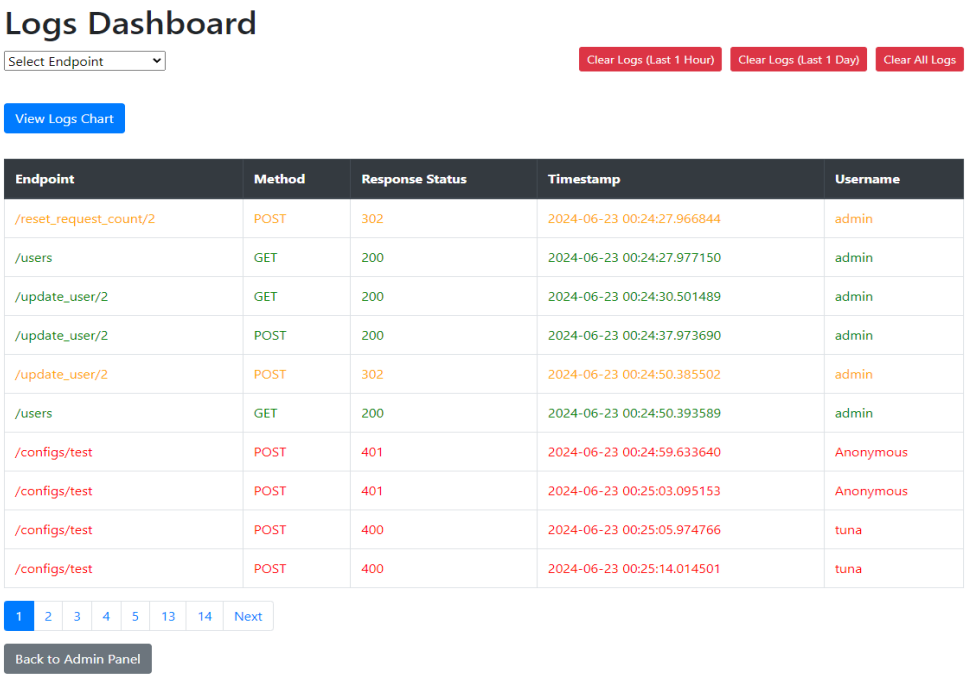
Şekil 2.6: Konfigürasyon yönetim paneli



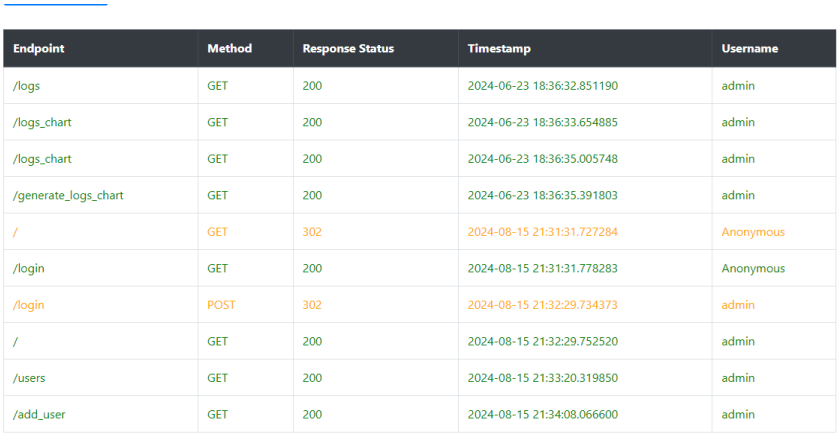
Şekil 2.7: Konfigürasyon ekleme paneli

2.2.3 Kayıt Takibi ve Analizi:

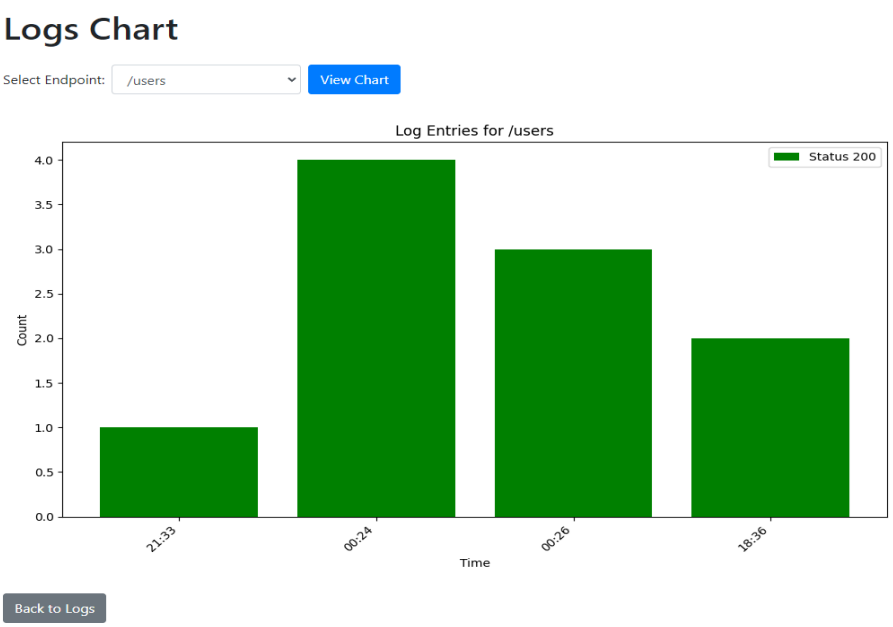
Sistemin güvenliği ve performansı izlemek amacıyla tüm kullanıcı işlemleri ve erişim denemeleri kaydedilmektedir. Bu kayıtlar PostgreSQL veri tabanında saklanmakta ve gerektiğinde Flask uygulaması üzerinden sorgulanabilmektedir. Kayıt verileri, Matplotlib kullanılarak grafiksel olarak görselleştirilmiştir; bu da kullanıcıların sistem üzerindeki etkinliklerini zaman bazında izlemeyi mümkün kılar.



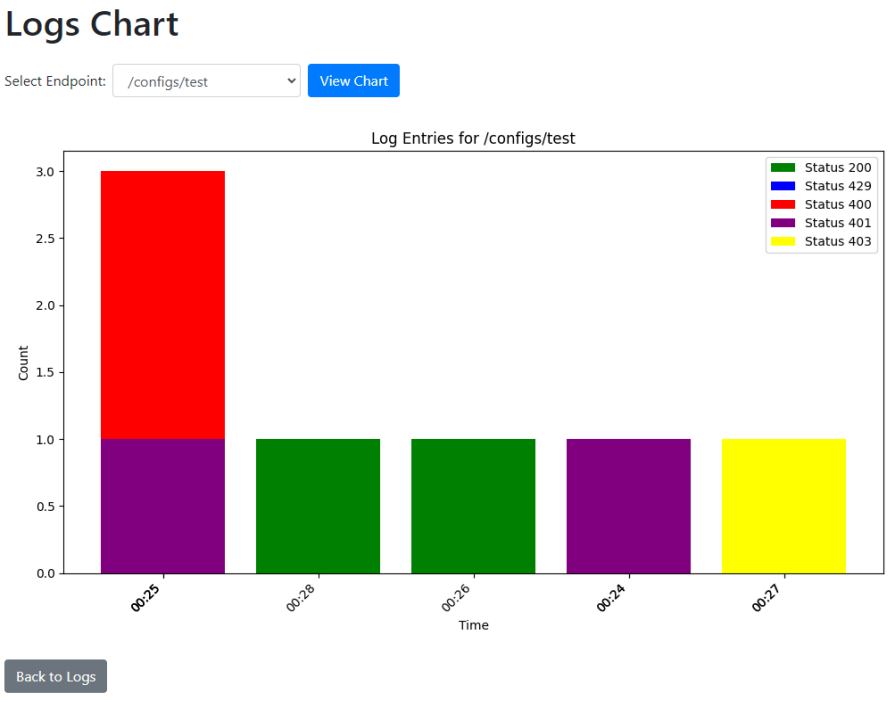
Şekil 2.8: Kayıt listesi (geçmiş)



Şekil 2.9: Kayıt listesi (yeni)



Şekil 2.10: Kayıt listesinin kullanıcılar servisine göre görselleştirilmesi



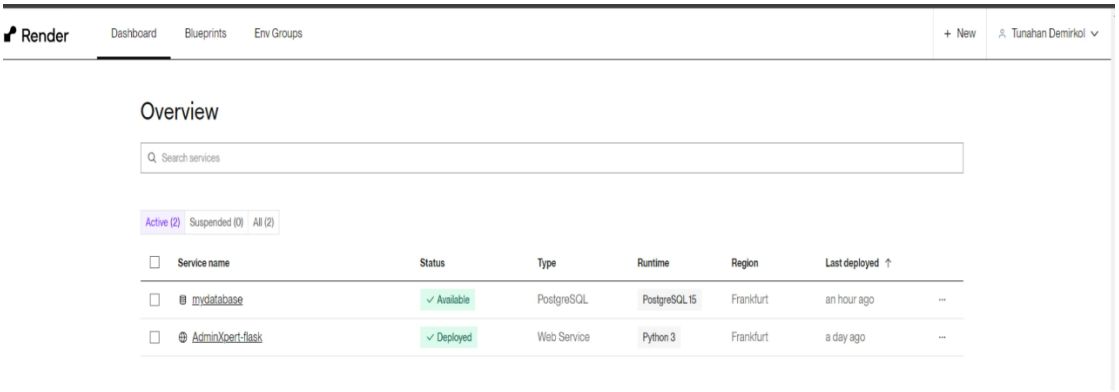
Şekil 2.11: Kayıt listesinin test servisine göre görselleştirilmesi

2.2.4 İstek Sınırlandırma (Rate Limit):

Sistemin kullanıcı başına günlük istek limitlerini kontrol etmek amacıyla istek limitlemesi uygulanmıştır. Her kullanıcı için günlük bir istek limiti belirlenmiş ve bu limit aşıldığında kullanıcıya 429 ("Çok Fazla İstek") hata kodu döndürülmüştür. Bu yöntem, sistemin aşırı yüklenmesini önlemek ve hizmet kalitesini korumak amacıyla uygulanmıştır.

2.2.5 Uygulamanın Dağıtımı:

Geliştirilen uygulama, Render platformu üzerinden dağıtılmıştır. Render, uygulamaların bulut ortamında çalışmasını sağlayan bir platformdur ve PostgreSQL veri tabanı entegrasyonu için gerekli yapılandırmalar yapılmıştır.



Şekil 2.12: Render platformu gösterge paneli

Bölüm 3

Tartışma ve Sonuç

Bu çalışma, merkezi yönetim sistemleri alanında mikroservis tabanlı bir yaklaşım sunarak, özellikle küçük ve orta ölçekli işletmelerin yönetimsel ihtiyaçlarına yanıt vermeyi hedeflemiştir. Flask mikro web çerçevesi ve Python programlama dili kullanılarak geliştirilen bu proje, kullanıcı yönetimi, konfigürasyon yönetimi, kayıt takibi ve istek sınırlandırma gibi önemli özellikleri entegre etmiştir. Çalışma, güvenli ve kullanıcı dostu bir arayüz sağlamak için Flask-Admin modülünü kullanmıştır.

Mikroservis tabanlı bir mimari benimsemenin, büyük sistemlerin yönetimsel zorluklarını azalttığı ve esnekliği artırdığı bilinmektedir. (Fowler, 2014), mikroservislerin, özellikle büyüyen işletmeler için ölçeklenebilir ve modüler çözümler sunduğunu belirtmektedir. Bu çalışma da küçük ölçekli işletmelerin artan yönetim ihtiyaçlarını karşılamak amacıyla benzer bir yaklaşımı benimsemiştir.

Mikroservis mimarisinin getirdiği faydalar, sadece teknik açıdan değil, aynı zamanda işletmelerin stratejik esnekliği açısından da değerlidir. (Nadareishvili vd., 2016), mikroservislerin farklı iş ihtiyaçlarına hızla yanıt verebilme kabiliyeti sayesinde, işletmelerin rekabet avantajı elde etmesine katkıda bulunduğunu savunmaktadır. Bu bağlamda, bu proje, esnek ve modüler bir yapı sunarak, işletmelerin operasyonel süreçlerini optimize etme potansiyeline sahiptir.

(Harrison, 2020) tarafından yapılan çalışmada, merkezi yönetim sistemlerinin özellikle yerel ağlar üzerinde çalışan küçük işletmeler için kritik olduğu vurgulanmıştır. Bu çalışma, bu önermeyi destekler nitelikte olup, yerel ağlar üzerinden güvenli erişim sağlama konusunda önemli bulgular sunmuştur.

Flask gibi mikro web çerçevelerinin kullanımı, özellikle mikroservis mimarilerinde modülerliği artırırken, geliştirme süreçlerini de hızlandırmaktadır. (Grinberg, 2018), Flask'in hızlı prototipleme ve esnek geliştirme süreçleri için ideal bir platform sunduğunu vurgulamaktadır.

Bu projenin en önemli katkılarından biri, Render platformu üzerinde yapılan başarılı bir uygulama dağıtım işlemi ile sistemin sorunsuz bir şekilde çalışmasıdır. Render gibi modern platformlar, mikroservis tabanlı uygulamaların hızlı ve güvenilir bir şekilde dağıtılmasını sağlayarak, işletmelerin operasyonel verimliliğini artırmaktadır (Smith, 2019).

Sonuç olarak, bu çalışma, merkezi yönetim sistemlerinin etkinliğini ve güvenliğini artırmaya yönelik önemli bir katkı sağlamaktadır. Gelecek çalışmalarda, bu sistemin bulut tabanlı ortamlarda daha geniş ölçekli testlerinin yapılması ve farklı güvenlik protokollerinin entegrasyonu ile daha da geliştirilebileceği düşünülmektedir. Özellikle, (Dragoni vd., 2017) tarafından belirtilen, mikroservislerin izlenmesi ve kayıt yönetimi gibi zorlukların daha detaylı incelenmesi, bu projeye gelecekteki geliştirmeler için önemli bir yön verebilir.

Bölüm 4

Kaynaklar

Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2016). *Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture*. IEEE Software, 33(3), 42-52.

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R. (2017). *Microservices: Yesterday, Today, and Tomorrow*. Springer.

Fowler, (2014). **Microservices: a definition of this new architectural term**. https://martinfowler.com/

Gulutzan, P., & Pelzer, T. (2002). *SQL Performance Tuning*. Addison-Wesley.

Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media.

Harrison, T., Phillips, M., & Smith, J. (2020). **The Importance of Centralized Management Systems for Small Enterprises**. *Journal of Small Business Management*, 58(3), 234-245

Nadareishvili, I., Mitra, R., McLarty, M., & Amundsen, M. (2016). *Microservice Architecture: Aligning Principles, Practices, and Culture*. O'Reilly Media.

Newman, S. (2015). *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media.

Richardson, C. (2018). *Microservices Patterns: With examples in Java*.

Sellers, C. (2020). *Mastering Flask Web Development*. Packt Publishing.

Smith, A. (2019). **Deployment Strategies for Microservices on Cloud Platforms**. *International Journal of Cloud Computing*, 7(2), 98-112.