

KOCAELİ ÜNİVERSİTESİ  
TEKNOLOJİ FAKÜLTESİ BİLİŞİM SİSTEMLERİ  
MÜHENDİSLİĞİ  
Proje B

StockMan

Hüseyin Oğuz ÇETİNKAYA  
191307003  
*Bilişim Sistemleri Mühendisliği*  
*Kocaeli Üniversitesi*

huseyinoguzc@gmail.com

Danışman

Öğr.Gör Alper METİN

KOCAELİ 2024

İÇİNDEKİLER

[İÇİNDEKİLER i](#_Toc154277420)

[ÖZET ii](#_Toc154277421)

[Giriş 1](#_Toc154277422)

[BÖLÜM 1: KULLANILAN TEKNOLOJİLER 1](#_Toc154277423)

[1.1 .NET 1](#_Toc154277424)

[1.2 ASP.NET Web Api 2](#_Toc154277425)

[1.3 React 2](#_Toc154277426)

[1.4 PostgreSQL 3](#_Toc154277426)

[1.5 MediatR 3](#_Toc154277426)

[1.6 Refit 4](#_Toc154277426)

[1.7 Transaction 4](#_Toc154277426)

[BÖLÜM 2: Veri Tabanı 8](#_Toc154277427)

[KARŞILAŞILAN PROBLEMLER 8](#_Toc154277432)

[KAYNAKÇA 9](#_Toc154277433)

StockMan

Stok Kontrol Projesi

# **ÖZET**

Dijital çağın gerekliliklerine uyum sağlayan işletmeler için etkin stok yönetimi, rekabet avantajı ve operasyonel verimlilik açısından büyük önem taşımaktadır. Bu ihtiyaca yönelik olarak C# ve ASP.NET Core Web API kullanılarak geliştirilen StoKont, farklı sektörlerdeki işletmeler için gelişmiş bir stok kontrol ve yönetim sistemi sunmaktadır. Veritabanı olarak PostgreSQL kullanılan bu sistem, React tabanlı bir kullanıcı arayüzüne sahiptir. Backend servisleri, mikroservis mimarisi kullanılarak tasarlanmış ve bu süreçte MediatR ve Refit gibi modern teknolojiler entegre edilmiştir. Bu raporda, StockMan’in teknolojik yapısı, modüler tasarımı ve işletmelere sağladığı faydalar detaylandırılmıştır.

**Anahtar Kelimeler**: C#, ASP.NET Core Web API, PostgreSQL, React, Mikroservis Mimarisi, MediatR, Refit,

# **ABSTRACT**

In the digital era, effective stock management is crucial for businesses to adapt to the demands of the market, offering a competitive edge and operational efficiency. Addressing this need, StoKont, developed using C# and ASP.NET Core Web API, presents an advanced stock control and management system for businesses across various sectors. This system employs PostgreSQL as its database and features a user interface based on React. The backend services are designed using a microservices architecture, integrating modern technologies such as MediatR and Refit. This report details the technological structure of StockMan, its modular design, and the benefits it provides to businesses.

**Keywords** C#, ASP.NET Core Web API, PostgreSQL, React, Mikroservis Mimarisi, MediatR, Refit,

Giriş

Günümüzde, teknolojinin hızla evrim geçirdiği bir dönemdeyiz ve bu değişim, işletmelerin operasyonel süreçlerinde de kendini göstermektedir. Özellikle stok yönetimi, dijitalleşme sürecinde önemli bir dönüşüm yaşamaktadır. StockMan, C# ve ASP.NET Core Web API kullanılarak geliştirilmiş, PostgreSQL veritabanı ile desteklenen ve React tabanlı bir kullanıcı arayüzüne sahip bir stok kontrol ve yönetim sistemidir. Bu giriş bölümünde, StockMan’ın geliştirilmesine yönelik temel motivasyonlar, hedefler ve bu projenin işletmeler için neden önemli olduğu üzerinde durulacaktır.

Rapor, bir dizi teknolojinin nasıl entegre edildiğini ve bu teknolojilerin StoKont'un genel mimarisine nasıl katkıda bulunduğunu ayrıntılı bir şekilde inceler. Bölüm 1, kullanılan temel teknolojileri (.NET, ASP.NET Web API, React, PostgreSQL, MediatR, Refit, Transaction) ve bu teknolojilerin projenin geliştirme sürecinde nasıl kullanıldığını ele alır. Bölüm 2 ise, StockMan’ın veritabanı yapısını ve bu yapının stok yönetim süreçlerindeki etkinliğini tartışır.

Raporun sonraki kısımlarında, proje geliştirme sürecinde karşılaşılan zorluklar ve bu zorlukların nasıl üstesinden gelindiği üzerinde durulacak ve son olarak, projenin genel bir değerlendirmesi sunulacaktır. Bu rapor, hem teknik uzmanlara hem de sektör profesyonellerine, modern stok yönetim sistemlerinin geliştirilmesindeki teknik ve operasyonel yönleri hakkında derinlemesine bir bakış açısı sunmayı amaçlamaktadır.

BÖLÜM 1: KULLANILAN TEKNOLOJİLER

1.1 .NET

.NET, Microsoft tarafından geliştirilen ve yazılım geliştiricilerin çeşitli uygulama türleri oluşturabilmesi için kapsamlı bir programlama altyapısıdır. Bu çerçeve, dil bağımsızlığı ve geniş kütüphane desteği ile tanınır, bu sayede geliştiricilere hızlı ve etkili bir şekilde çözümler sunar. .NET, esneklik ve güvenlik gibi özellikleri sayesinde, geniş çaplı uygulamaların geliştirilmesi için ideal bir platformdur.

StockMan projesinde .NET'in tercih edilmesinin temel nedenleri, performans, güvenlik ve platformlar arası uyumluluk özellikleridir. .NET, yüksek performanslı uygulamalar geliştirmek için gereken araçları ve kütüphaneleri sağlar. Bu, özellikle stok yönetimi gibi sürekli veri işleme ve güncelleme gerektiren uygulamalar için kritik önem taşır. Güvenlik, her türlü iş uygulamasının temel bir gereksinimidir ve .NET, güçlü şifreleme ve veri koruma mekanizmaları ile bu ihtiyacı karşılar.

Projenin çapraz platform gereksinimleri göz önüne alındığında, .NET Core'un platformlar arası uyumluluk özelliği de önemli bir faktördür. Bu, StockMan sisteminin farklı işletim sistemlerinde sorunsuz bir şekilde çalışabilmesini sağlar, böylece sistem daha geniş bir kullanıcı kitlesine ulaşabilir. Ayrıca, .NET'in sağladığı geniş kütüphane ve topluluk desteği, geliştirme sürecini hızlandırır ve daha etkili çözümler üretmeyi mümkün kılar.

1.2 ASP.NET Web Api

ASP.NET Web API, modern web uygulamalarında RESTful API'lar oluşturmak için kullanılan güçlü bir .NET çerçevesidir. Bu çerçeve, HTTP hizmetlerini kolayca oluşturup, geniş bir yelpazede istemciler tarafından tüketilmesini sağlar, böylece mobil, web ve masaüstü uygulamalar arasında verimli bir şekilde iletişim kurulabilir. ASP.NET Web API, geliştiricilere, veri odaklı ve HTTP tabanlı servisler geliştirme konusunda geniş esneklik ve kontrol imkanı sunar.

StockMan projesinde ASP.NET Web API'nin kullanılmasının temel nedeni, mikroservis mimarisine uygun, hızlı ve etkili bir şekilde RESTful hizmetler sunabilmesidir. Bu çerçeve, veri tabanı ile etkileşimde bulunan ve iş mantığını işleyen backend servislerinin geliştirilmesinde kritik bir rol oynar. ASP.NET Web API'nin sunduğu modüler yapı, proje geliştirme sürecini hızlandırır ve servislerin daha kolay bakımını sağlar.

Ayrıca, ASP.NET Web API'nin güvenlik, hata yönetimi ve veri serileştirme gibi gelişmiş özellikleri, StoKont'un güvenli ve stabil bir şekilde çalışmasını destekler. Bu özellikler, özellikle hassas verilerin işlendiği ve yüksek düzeyde kullanılabilirlik gerektiren stok yönetimi sistemlerinde büyük önem taşır. ASP.NET Web API, RESTful prensiplerine uygun bir şekilde tasarlanmış olup, basit, temiz ve anlaşılır API endpoint'leri oluşturulmasına olanak tanır.

1.3 React

Projemin frontend kısmında kullandığım dil olan React, Facebook tarafından geliştirilen ve modern web uygulamalarının arayüz geliştirme süreçlerinde yaygın olarak kullanılan bir JavaScript kütüphanesidir. Kullanıcı arayüzlerini etkili bir şekilde oluşturmak ve yönetmek için tasarlanmış olan React, özellikle tek sayfa uygulamaları (SPA) için ideal bir çözümdür. Bu kütüphane, yenilikçi ve etkileşimli kullanıcı deneyimleri tasarlamak isteyen geliştiriciler arasında popülerdir.

StockMan projesinde React'in tercih edilmesinin başlıca sebepleri arasında, bileşen tabanlı mimarisi, yüksek performansı ve esnekliği bulunmaktadır. Bileşen tabanlı yaklaşım, uygulamanın daha modüler ve yeniden kullanılabilir yapılarda geliştirilmesine olanak tanır. Bu, özellikle büyük ve karmaşık kullanıcı arayüzlerinde, kodun bakımını ve güncellemesini kolaylaştırır.

React'in sunduğu "Sanal DOM" (Virtual DOM) özelliği, web uygulamalarının performansını önemli ölçüde artırır. Sanal DOM, gerçek DOM ile karşılaştırıldığında daha hızlı güncellenebilir ve bu sayede kullanıcı arayüzünün daha akıcı ve hızlı bir şekilde yanıt vermesini sağlar. Bu özellik, StoKont gibi dinamik veri ile çalışan ve gerçek zamanlı güncellemelere ihtiyaç duyan uygulamalar için büyük önem taşır.

Ayrıca, React'in geniş ekosistemi ve topluluk desteği, geliştirme sürecini kolaylaştırır ve projeye eklenen yeni özellikler ve güncellemeler için güçlü bir temel sunar. React, ayrıca, StockMan’ın kullanıcı arayüzünün modern, etkileşimli ve kullanıcı dostu olmasını sağlayarak, son kullanıcı deneyimini önemli ölçüde iyileştirir.

1.4 PostgreSQL

PostgreSQL, açık kaynaklı, güçlü ve ölçeklenebilir bir ilişkisel veritabanı yönetim sistemidir. Nesne-ilişkisel veritabanı modelini benimseyen bu sistem, gelişmiş sorgulama dili ve sağlam veri bütünlüğü özellikleri ile bilinir. PostgreSQL, karmaşık veri işlemlerini yönetmek ve büyük veri kümeleri üzerinde yüksek performans sağlamak için tasarlanmıştır, bu yüzden genellikle büyük, çok kullanıcılı uygulamalar ve yüksek hacimli veri depolama gereksinimleri için tercih edilir.

StockMan projesinde PostgreSQL'in kullanılmasının sebepleri arasında, güçlü veri bütünlüğü özellikleri, gelişmiş sorgulama kapasitesi ve yüksek düzeyde ölçeklenebilirlik bulunmaktadır. StockMan, sürekli güncellenen stok verileri ve karmaşık veri ilişkileriyle çalışan bir sistemdir. Bu nedenle, veri tutarlılığı ve güvenilirliği, projenin başarısı için hayati öneme sahiptir. PostgreSQL, bu tür veri yoğunluğu ve karmaşıklığına uygun gelişmiş özellikler sunarak, StockMan’ın bu ihtiyaçlarını karşılamaktadır.

PostgreSQL'in ölçeklenebilir yapısı, StoKont'un farklı büyüklükteki işletmelerin ihtiyaçlarına uyum sağlamasına olanak tanır. Bu, sistem büyüdükçe veya veri hacmi arttıkça, veritabanının performansının korunmasını ve veri işleme kapasitesinin artırılmasını sağlar. Ayrıca, PostgreSQL'in genişletilebilir yapısı, özel veri türleri, işlevler ve operatörler oluşturarak StockMan’ın özelleştirilmiş veri işleme gereksinimlerini karşılamaya yardımcı olur.

1.5 MediatR

MediatR, .NET uygulamaları için tasarlanmış bir aracı yazılım (middleware) kütüphanesidir ve CQRS (Command Query Responsibility Segregation) prensiplerine dayalı uygulamaların geliştirilmesini kolaylaştırır. Bu kütüphane, uygulama içindeki bileşenler arasında düşük bağlılık sağlayarak, komutları ve istekleri yönetmekte ve işlemekte kullanılır. MediatR'ın temel amacı, uygulama katmanlarını birbirinden ayırmak ve böylece daha temiz, daha yönetilebilir ve test edilebilir kod yapısı oluşturmaktır.

StockMan projesinde MediatR kullanılmasının ana sebebi, uygulamanın modüler ve esnek bir yapıda geliştirilmesini sağlamaktır. MediatR, uygulamanın farklı bölümlerini birbirinden bağımsız hale getirerek, her bir modülün kendi sorumluluğunu daha etkin bir şekilde yerine getirmesine olanak tanır. Bu yaklaşım, özellikle büyük ve karmaşık sistemlerde, sistemin genel bakımını ve güncellenmesini kolaylaştırır.

MediatR'ın sağladığı mesajlaşma ve aracılık mekanizması, StockMan’ın backend mimarisinde önemli bir role sahiptir. Bu, uygulama içindeki iş akışlarının daha düzenli ve anlaşılır hale gelmesini sağlar ve ayrıca, uygulama içindeki bağımlılıkları azaltır. Komutlar ve sorgular aracılığıyla, veri akışı ve işleme süreçleri daha net bir şekilde ayrıştırılır, bu da sistemdeki hataların tespit edilmesini ve giderilmesini kolaylaştırır.

1.6 Refit

Refit, .NET uygulamalarında HTTP API çağrılarını yönetmek için kullanılan modern bir HTTP istemci kütüphanesidir. Bu kütüphane, API çağrılarını daha sade, daha anlaşılır ve daha yönetilebilir hale getirmek için tasarlanmıştır. Refit, REST API ile etkileşim kurarken kullanılan tipik uzun ve karmaşık HTTP istemci kodlarını, basit ve temiz arayüz tanımları ile değiştirmeyi amaçlar. Böylece, geliştiriciler API ile etkileşimde bulunurken daha az kod yazarak, daha verimli çalışabilirler.

StockMan projesinde Refit kullanılmasının temel sebebi, mikroservis mimarisi içindeki servisler arası iletişimi kolaylaştırmaktır. Refit, servisler arasındaki HTTP tabanlı iletişimi basitleştirerek, mikroservislerin birbirleriyle daha etkili bir şekilde iletişim kurmasını sağlar. Bu, özellikle çok sayıda mikroservis içeren ve bu servisler arasında sürekli veri alışverişi gerektiren StoKont gibi projeler için büyük önem taşır.

Refit'in sağladığı, önceden tanımlanmış arayüzler ve entegre edilmiş hata yönetimi, StockMan projesinin API tüketim süreçlerini daha güvenilir ve yönetilebilir hale getirir. Ayrıca, bu yaklaşım, kodun okunabilirliğini ve bakımını önemli ölçüde iyileştirir. Refit, API çağrılarını standart bir yapıya dönüştürerek, projenin genelinde tutarlı bir API tüketim modeli oluşturur.

1.7 Transaction

Transaction, yani işlem, veritabanında gerçekleştirilen bir dizi işlemin bütünlüğünü ve güvenliğini sağlamak için kullanılan bir kavramdır. İşlem yönetimi, veritabanı sistemlerinde kritik bir rol oynar, çünkü bütünleşik ve tutarlı veri durumunu korumak için gereklidir. Bir işlem, ya tamamen başarılı olmalı ya da hiçbir etki bırakmamalıdır; bu, veritabanı yönetiminde "Atomiklik" olarak bilinen bir prensiptir.

StockMan projesinde, işlem yönetimi, DbContext'in sağladığı DbContext.Database.BeginTransaction metodunu kullanarak gerçekleştirilir. Bu yaklaşım, veritabanı işlemlerinin güvenliğini ve tutarlılığını sağlamak için önemlidir. Özellikle, stoğun yönetimi ve güncellenmesi gibi karmaşık veri işlemleri sırasında, birden fazla adımın başarılı bir şekilde tamamlanmasını gerektiren durumlarda, işlem yönetimi hayati önem taşır.

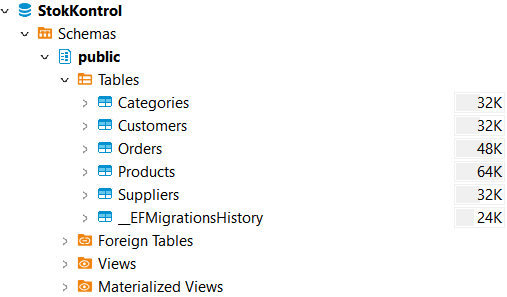
DbContext üzerinden gerçekleştirilen transaction yönetimi, StockMan’ın veri bütünlüğünü korurken esneklik ve ölçeklenebilirlik sağlar. Bu yöntem, birden fazla veritabanı işlemini bir araya getirerek, bu işlemlerin hepsinin başarılı bir şekilde tamamlanmasını veya hata durumunda hepsinin geri alınmasını garanti eder. Bu, veri tutarlılığını korumanın yanı sıra, uygulama hatalarının ve veritabanıyla ilgili sorunların önlenmesine de yardımcı olur.

BÖLÜM 3: Veritabanı

Veritabanı olarak PostgreSQL kullanmak benim açımdan yeni bir tecrübe oldu. Önceden geliştirmiş olduğum projeler arasında ilişkisel veri yapısı destekleyen MsSQL kullanarak proje geliştirdiğimden ötürü PostgreSQL kullanımında yani entity tablolarımın tasarımı gibi durumlarda çok zorlanacağımı düşünmüyordum.

Postgre kullanmamın sebebi açık kaynak bir veritabanı teknolojisi olması ve daha önce bu veritabanı ile bir proje geliştirmemem oldu.

PostgreSQL kullanırken yönetim arayüzü olarak Dbeaver aracını kullandım. Bu araç Sql Management Studio ile biraz farklı gelse de kullandıkça alıştığım bir araç oldu.



Şekil 1

KARŞILAŞILAN PROBLEMLER

Projeyi geliştirirken back-end servislerini yazmada tecrübem vardı. Projeyi geliştirirken hiç tecrübem olmayan konulara da bulaşmak istedim. Bunların en büyüğü olarak front kısmını geliştirdiğim React.js bulunuyor. Kendimi şu anda back-end üzerine geliştiriyorum. Front bilgim neredeyse 0 diyebiliriz. Bu durumda olunca React.js ile en basitinden http isteği atmak bile çok zor/karışık bir duruma geldi. İsteği atabilmeyi başardım e peki ekrana ürün/kategori gibi servis alanlarını eklemek için gerekli olan form ekranı nasıl oluşacak ? Bu da oluştu güncelleme yaparken ilgili satırlar dolu şekilde nasıl gelecek gibi bir sürü bilmediğim durumla karşılaştım. Bu durumları bol bol araştırarak ve bilgisi olan arkadaşlarım ile iletişim kurarak ilerlettim. Bu projede front olarak güzel bir yapı kuramadım fakat bunun sebebi react ile ilgili bilgimin düşük seviyede olması.

Veritabanı aşamasında da ilişkisel veri yapılarını kullanan veritabanlarında tecrübem olduğunu daha önceki projelerde geliştirdiğimi söylememe rağmen beklemediğim kadar uğraştırıcı bir serüven oldu. Bu durumda entity sınıflarını oluşturmak gibi durumlar etkili değildi. PostgreSQL kurulumu benim de bazı durumları gözden kaçırmamdan ötürü uzun sürdü. Bununla birlikte Dbeaver gibi PostgreSQL yönetimi aracını da ilk defa kullanmak SQL Management Studio’dan sonra biraz garip geldi. Veritabanıyla ilgili bir başka uğraştırıcı olan durum da migration konusuydu. En son ASP.Net 6.0 projelerinde migration işlemi uygulamıştım ve o zamandan bu zamana yani .Net 8.0 sürümüne gelindiğinde migration ve entityframework core işlemleriyle ilgili güncellemeler gelmiş ve ben bu güncellemeleri kaçırmıştım. Eski 6.0’daki notlarıma bakarak yapmaya çalıştığımda bir süre hatalar ile uğraştım ki bunlar hata değilmiş bunlar eski versiyon hatasıymış. İnternet üzerinde biraz araştırmadan sonra bu işlemleri de hallettim ve devam ettim.

Back-end servislerini geliştirirken de zorlandığım durumlar aslında çok olmadı. Projeyi geliştirirken farklı teknolojiler kullnamak istedim bunlar MediatR ve Refit gibi teknolojilerdi. Bu teknolojileri projelere entegre etmek ilk denemelerini yapmak biraz kafa karıştırıcı olsa da kullanmaya başladıktan sonra yararlı bir durum sağladığını ve kullanmanın faydalı olduğunu gördüm.

KAYNAKÇA

1. https://www.gencayyildiz.com/blog/cqrs-pattern-nedir-mediatr-kutuphanesi-ile-nasil-uygulanir/ https://aws.amazon.com/tr/what-is/net/
2. https://www.gencayyildiz.com/blog/net-core-refit-kutuphanesi-nedir-nasil-kullanilir/ https://www.hosting.com.tr/blog/postgresql/
3. https://www.gencayyildiz.com/blog/category/microservices/ https://tr.tradingview.com/news/coinkolik:b42df9f32d9e8:0/
4. https://www.gencayyildiz.com/blog/entity-framework-code-first-migrations-islemleri/ https://redux.js.org/introduction/getting-started
5. https://react-hook-form.com/
6. https://legacy.reactjs.org/docs/react-component.html https://legacy.reactjs.org/docs/hooks-reference.html

Projenin Github Linki:

<https://github.com/hoguzcetinkaya/StokKontrol/tree/sunum>

Projenin Driver Linki:

https://drive.google.com/drive/folders/1e7wO5xY3SIRQvo7JYiLG3tZzsynCxc60?usp=sharing