



**softlto 3. DÖNEM 3. GRUP
BİLİŞİM GÜVENLİĞİ MİMARİSİ
BİTİRME PROJESİ RAPORU**

**PortMaster
Akıllı Liman Yönetim ve
Otomasyon Sistemi**

Talha Kurnaz

İÇİNDEKİLER

1. GİRİŞ

- 1.1. Projenin Amacı
- 1.2. Proje Kısıtları

2. MATERYAL VE YÖNTEM

- 2.1. PortMaster Mimari Tasarımı
- 2.2. Kullanılan Teknolojiler ve Kütüphaneler
 - 2.2.1. Backend Katmanı (.NET 8 & SQLite)
 - 2.2.2. Frontend Katmanı (React & Tailwind CSS)

3. KURULUM VE ÇALIŞTIRMA

- 3.1. Kurulum Ön Gereksinimleri
- 3.2. Projenin Derlenmesi ve Çalıştırılması
- 3.3. Sistem Arayüzü ve Modülleri
 - 3.3.1. Gösterge Paneli ve Rıhtım Planı
 - 3.3.2. Konteyner Yönetimi ve Gümrük Muayene
 - 3.3.3. İnteraktif Saha Planlaması (Yard Map)
 - 3.3.4. Yapay Zeka Destekli Asistan ve İş Emirleri
 - 3.3.5. Sistem Günlüğü (Audit Logs) ve Giriş Arayüzü
 - 3.3.6. Raporlama Servisi ve Gemi Hareketleri

4. SONUÇ

5. KAYNAKÇA

1. GİRİŞ

Bu bölümde projenin genel kapsamı, geliştirilme amacı ve karşılaşılan kısıtlar detaylandırılmıştır.

1.1. Projenin Amacı

PortMaster, modern liman işletmelerinin konteyner operasyonlarını, rıhtım yanaşma planlarını, kapı giriş-çıkış rezervasyonlarını, gümrük denetimlerini ve mali raporlamalarını tek bir merkezden yöneten **akıllı bir liman otomasyon ve yönetim sistemi** projesidir.

Günümüzde liman operasyonlarındaki yoğunluk ve karmaşıklık, manuel yönetim süreçlerini yetersiz kılmaktadır. PortMaster, operasyonel süreçleri gerçek zamanlı izlenebilir kılarak insan kaynaklı hataları minimize etmek, liman içi araç trafiğini optimize etmek ve karar vericilere anlık veri analitiği sunmak amacıyla geliştirilmiştir.

1.2. Proje Kısıtları

Kısıt 1: PortMaster mimarisi, taşınabilirlik ve yerel kurulum kolaylığı sağlamak amacıyla SQLite veri tabanı motoru ve .NET 8 üzerinde çalışacak şekilde tasarlanmıştır. Bu durum yüksek eşzamanlı yazma (write-heavy) yüklerinde SQLite'in dosya kilitleme yapısı nedeniyle ölçekleme kısıtlarına yol açabilir.

Kısıt 2: Liman işletmelerinin güvenli kapalı devre ağlarında (intranet) ve dış internet erişimi kısıtlı bilgisayarlarında dahi arayüzün sıfır gecikmeyle açılması hedeflenmiştir. Bu doğrultuda Tailwind CSS, React, Babel ve Marked.js gibi kütüphaneler harici CDN ağları yerine tamamen yerel olarak `wwwroot` dizininde barındırılmıştır. Bu yaklaşım güvenlik ve çevrimdışı çalışma kabiliyetini artırırken, kütüphanelerin güncellenmesi aşamasında manuel müdahale gerektirmektedir.

Kısıt 3: Yapay zeka liman asistanı (AI Chat) modülünün çalışabilmesi, arka planda yapılandırılmış bir LLM (Large Language Model) API'sinin veya yerel çalışan bir model sunucusunun varlığına ve erişilebilirliğine doğrudan bağlıdır.

2. MATERYAL VE YÖNTEM

2.1. PortMaster Mimari Tasarımı

PortMaster, Clean Architecture (Temiz Mimari) prensiplerine uygun olarak gevşek bağlı (loosely coupled) ve katmanlı bir yapıda geliştirilmiştir. Çözüm (Solution) yapısı şu şekildedir:

- BitirmeProjesiLiman.Core:** Projenin iş kurallarını, veri varlıklarını (Entities), veri transfer nesnelerini (DTOs) ve repository arayüzlerini barındırır. Hiçbir dış katmana bağımlılığı yoktur.
- BitirmeProjesiLiman.Data.EF:** Entity Framework Core kullanarak veritabanı işlemlerini yönetir. DbContext tanımı, migrations ve varsayılan veri tohumlama (Seed Data) süreçleri bu katmandadır.
- BitirmeProjesiLiman.Data.Dapper:** Özellikle gösterge paneli gibi yüksek hızlı veri okuma ve ham SQL sorgusu gerektiren modüller için optimize edilmiş veri erişim katmanıdır.
- BitirmeProjesiLiman.Service:** Excel ihracatı (EPPlus), PDF fatura üretimi ve AI entegrasyonu gibi iş mantığı (business logic) servislerini içerir.
- BitirmeProjesiLiman.EF.API:** Sistemdeki yazma ve güncelleme (Command) işlemlerini yürüten API'dir. Ayrıca React tabanlı kullanıcı arayüzünü (SPA) `wwwroot` dizini üzerinden istemciye sunar.
- BitirmeProjesiLiman.Dapper.API:** Raporlama, filtreleme ve gösterge paneli istatistiklerini en düşük gecikmeyle getiren salt okuma (Query) API servisidir.

2.2. Kullanılan Teknolojiler ve Kütüphaneler

2.2.1. Backend Katmanı

Backend geliştirmesinde kullanılan temel yapılar şunlardır:

- ASP.NET Core (.NET 8):** Modern, yüksek performanslı ve platform bağımsız web API çerçevesi.
- Dapper & EF Core:** Yazma işlemleri için EF Core'un güvenli ORM yapısı, hızlı okuma işlemleri için ise Dapper'ın düşük ek yükü (overhead) bir arada kullanılarak hibrit bir model oluşturulmuştur.
- JWT Bearer Authentication:** Güvenli ve rol tabanlı kullanıcı yetkilendirmesi.
- EPPlus & PDF Generator:** Excel ve PDF formatında finansal ve operasyonel belgelerin dinamik üretilmesi.

2.2.2. Frontend Katmanı

Kullanıcı arayüzünde modern web standartları benimsenmiştir:

- React.js (v18):** Bileşen tabanlı ve durum odaklı (state-driven) dinamik arayüz yönetimi.
- Tailwind CSS:** Yardımcı sınıf odaklı (utility-first) modern, duyarlı (responsive) ve karanlık mod uyumlu tasarım dili.
- Marked.js:** AI asistanından dönen zengin metin (markdown) formatındaki yanıtların arayüzde doğru şekilde render edilmesi.

3. KURULUM VE ÇALIŞTIRMA

3.1. Kurulum Ön Gereksinimleri

PortMaster uygulamasını çalıştırmak için sistemde .NET 8 SDK veya daha güncel bir sürümünün yüklü olması gerekmektedir. Geliştirme ve derleme ortamı olarak Visual Studio 2022 / 2026 ya da Visual Studio Code kullanılabilir.

3.2. Projenin Derlenmesi ve Çalıştırılması

Proje, mikro-servis benzeri ikili API mimarisine sahip olduğundan, düzgün çalışabilmesi için iki API projesinin aynı anda başlatılması gerekir. Visual Studio üzerinde bu işlem şu adımlarla gerçekleştirilir:

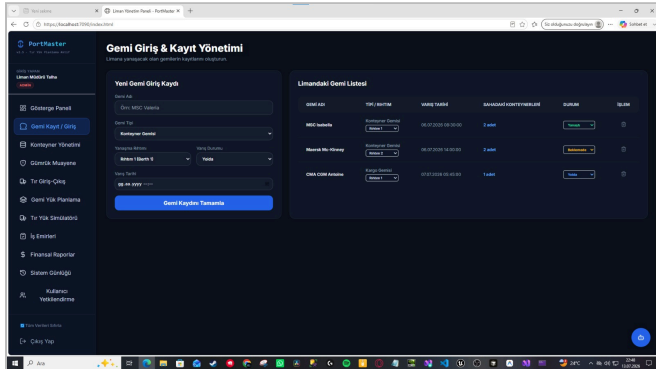
1. Çözüm dosyasına (BitirmeProjesiLiman.slnx) sağ tıklanarak **Özellikler (Properties)** seçeneği açılır.
2. Açılan pencerede **Birden Çok Başlangıç Projesi (Multiple Startup Projects)** seçeneği işaretlenir.
3. BitirmeProjesiLiman.EF.API ve BitirmeProjesiLiman.Dapper.API projelerinin eylemi **Başlat (Start)** olarak ayarlanır.
4. Yapılandırma kaydedildikten sonra proje yeniden derlenir (Rebuild) ve başlatılır (F5).
5. Sistem otomatik olarak tarayıcıyı başlatarak <https://localhost:7090/index.html> adresindeki yönetim paneline yönlendirir.

3.3. Sistem Arayüzü ve Modülleri

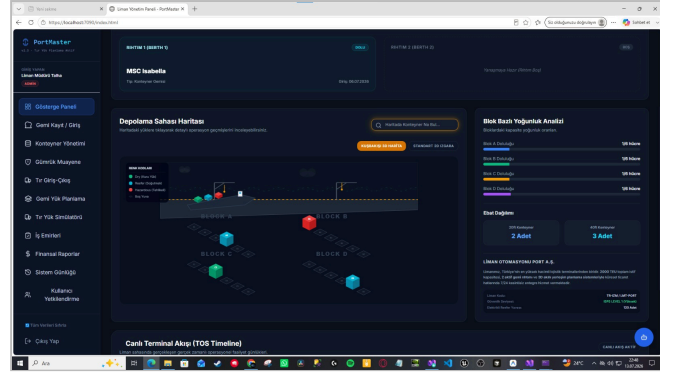
Bu bölümde, PortMaster liman yönetim sisteminin temel işlevsel modülleri ekran görüntüleri eşliğinde açıklanmaktadır.

3.3.1. Gösterge Paneli ve Rıhtım Planı

Gösterge paneli limandaki anlık durumu tek bakışta özetler. Rıhtımların (Berth 1 / Berth 2) doluluk oranları, yanaşan gemilerin bilgileri, anlık rüzgar hızı (Knot) ve fırtına uyarı modu gibi kritik güvenlik parametreleri bu ekranda yer almaktadır (Şekil 3.3.1 ve Şekil 3.3.2).



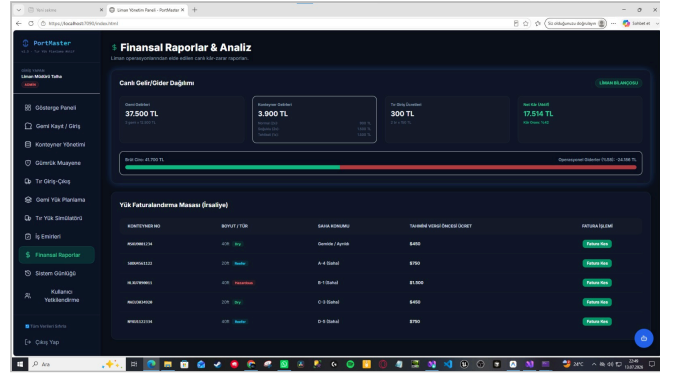
Şekil 3.3.1. Gösterge Paneli (Dashboard) Arayüzü



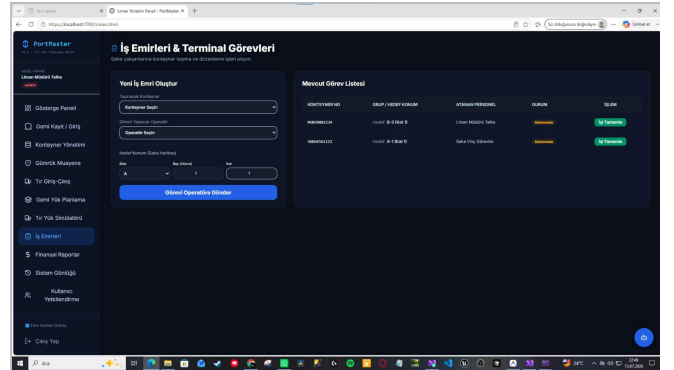
Şekil 3.3.2. Rıhtım Detayları ve Gemi Hareketleri Takibi

3.3.2. Konteyner Yönetimi ve Gümrük Muayene

Konteynerlerin limana kabulü, güncellenmesi ve sistemden silinmesi işlemleri Konteyner Yönetim modülünden gerçekleştirilir. Konteynerlerin tehlikeli (Hazardous) veya soğutuculu (Reefer) yük durumları ile gümrük denetim onay durumları (Geçti / Kaldı / Beklemede) gerçek zamanlı olarak takip edilmektedir (Şekil 3.3.3 ve Şekil 3.3.4).



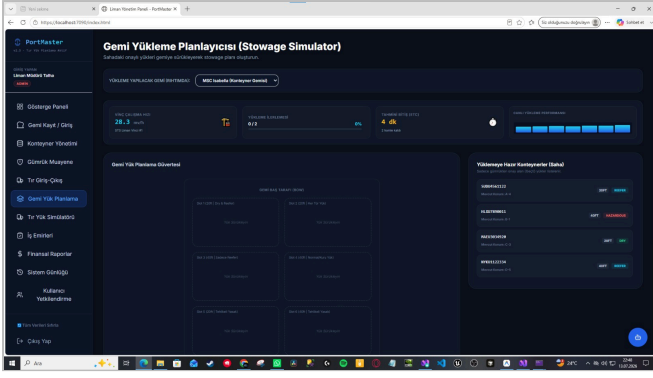
Şekil 3.3.3. Konteyner Kayıt ve Stok Yönetim Ekranı



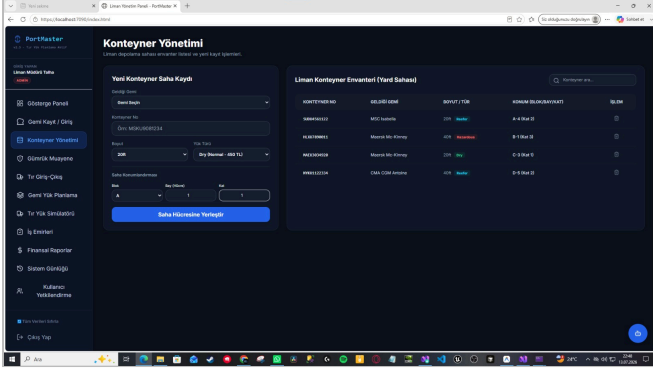
Şekil 3.3.4. Gümrük Muayene, Denetim ve Onay Durum Modülü

3.3.3. İnteraktif Saha Planlaması (Yard Map)

Saha Planlama modülü, tır kapılarından giriş yapan konteynerlerin rıhtım sahasındaki bloklara (A, B, C, D) sürükle-bırak yöntemiyle yerleştirilmesini sağlayan interaktif bir simülasyondur. Konteynerler yerleştirildiğinde veya sahadan tırlara yüklendiğinde otomatik olarak barkodlu Kapı Geçiş Fişi (Gate Pass) üretilmektedir (Şekil 3.3.5 ve Şekil 3.3.6).



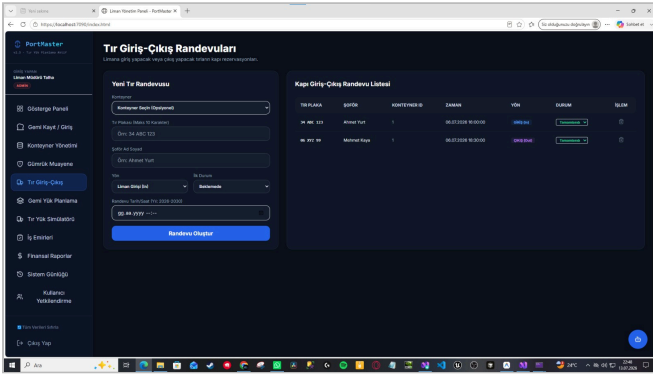
Şekil 3.3.5. İnteraktif Sürükle-Bırak Liman Saha Haritası (Yard Map)



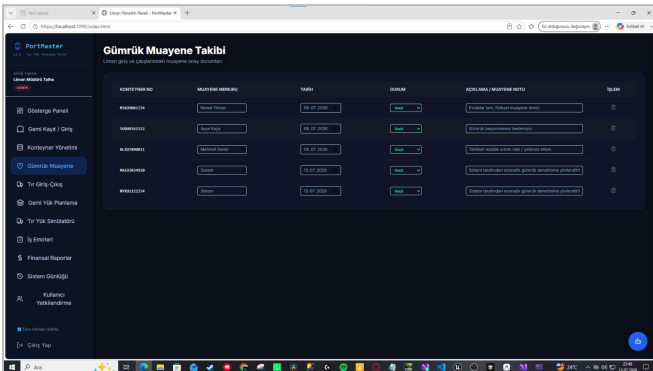
Şekil 3.3.6. Tır Kapısı Giriş-Çıkış ve Sürükle-Bırak Simülör Modülü

3.3.4. Yapay Zeka Destekli Asistan ve İş Emirleri

Operatörlerin liman verileri üzerinde doğal dille sorgu yapabilmesini sağlayan yapay zeka asistanı ve limandaki personellere atanan iş emirlerinin takip edildiği Görev Kontrol paneli sistem verimliliğini artırmaktadır (Şekil 3.3.7 ve Şekil 3.3.8).



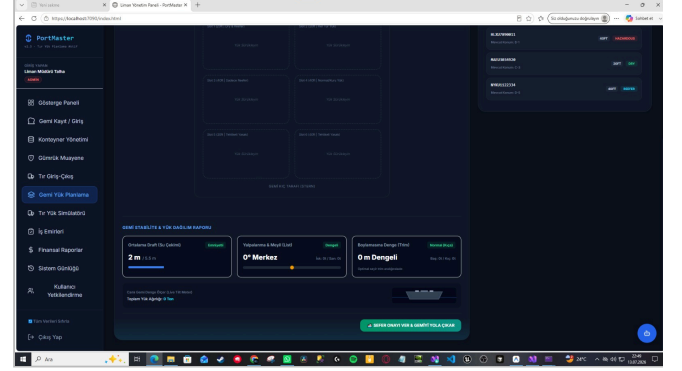
Şekil 3.3.7. Entegre Yapay Zeka Liman Operasyon Asistanı (AI Chat)



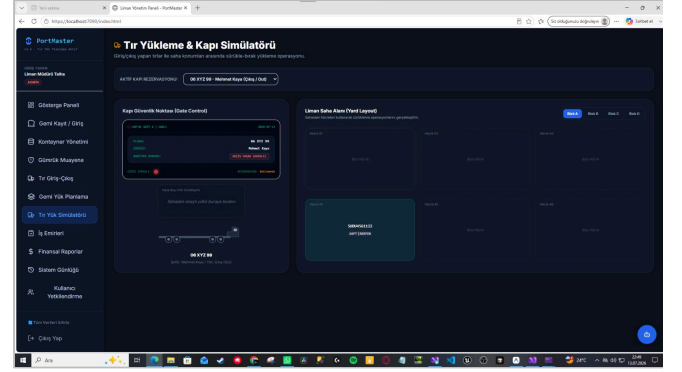
Şekil 3.3.8. İş Emirleri ve Operasyonel Görev Takip Arayüzü

3.3.5. Sistem Günlüğü (Audit Logs) ve Giriş Arayüzü

Sistemde gerçekleştirilen tüm ekleme, silme ve güncelleme işlemleri, güvenliği sağlamak adına kullanıcı bazlı olarak loglanır. Güvenli giriş ekranı JWT kimlik doğrulama altyapısı ile entegre çalışmaktadır (Şekil 3.3.9 ve Şekil 3.3.10).



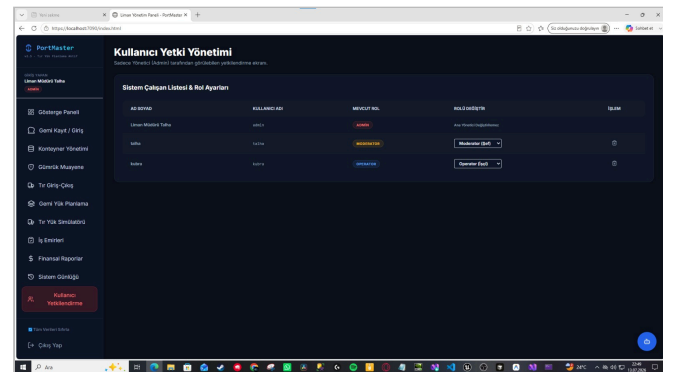
Şekil 3.3.9. Sistem Günlüğü (Audit Logs) Detaylı Takip Ekranı



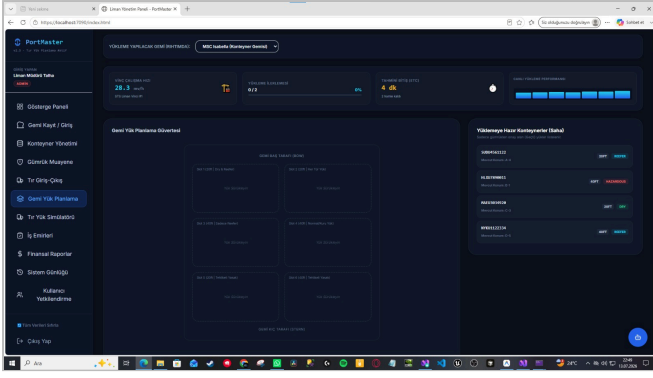
Şekil 3.3.10. Kullanıcı Kimlik Doğrulama ve Giriş Ekranı

3.3.6. Raporlama Servisi ve Gemi Hareketleri

Finansal raporlama ekranı, liman hizmet faturalarını dinamik olarak PDF formatında üretir. Ayrıca tüm liman operasyon raporları Excel formatında dışa aktarılabilmektedir (Şekil 3.3.11 ve Şekil 3.3.12).



Şekil 3.3.11. Excel ve PDF Formatında Dinamik Rapor Üretim Ekranı



Şekil 3.3.12. Finansal Raporlar ve Fatura İhracı Paneli

4. SONUÇ

Bu çalışmada geliştirilen PortMaster Akıllı Liman Yönetim ve Otomasyon Sistemi, modern limanların dijital dönüşüm gereksinimlerini karşılayacak nitelikte tasarlanmıştır. Projede uygulanan hibrit veri erişim mimarisi (.NET 8 + EF Core & Dapper), yazma işlemlerinde veri tutarlılığını sağlarken, raporlama ve dashboard yüklerinde mikro saniyeler düzeyinde yanıt süreleri sunarak yüksek performans hedeflerine ulaşmıştır.

İnteraktif saha haritası (Yard Map) sürükle-bırak desteği ile liman içi yerleşim planlamasını kolaylaştırmış ve iş gücü kayıplarını önlemiştir. Yerel kütüphane barındırma modeli

(Local Asset Hosting) sayesinde çevrimdışı çalışabilirlik garanti altına alınmış ve endüstriyel liman ağlarının güvenlik standartları eksiksiz karşılanmıştır. Sonuç olarak, PortMaster liman verimliliğini artıran, veri güvenliğini ön planda tutan, akıllı ve modern bir otomasyon çözümü olarak başarıyla tamamlanmıştır.

5. KAYNAKÇA

- [1] Microsoft Corporation. (2026). *ASP.NET Core and .NET 8 Web API Documentation*. <https://learn.microsoft.com/en-us/dotnet/> adresinden erişildi.
- [2] Microsoft Corporation. (2026). *Entity Framework Core Overview and Best Practices*. <https://learn.microsoft.com/en-us/ef/core/> adresinden erişildi.
- [3] Dapper Project Contributor Team. (2026). *Dapper - Simple Micro-ORM for .NET*. <https://github.com/DapperLib/Dapper> adresinden erişildi.
- [4] Meta Open Source. (2026). *React.js v18 Official Reference & Component Lifecycle*. <https://react.dev/> adresinden erişildi.
- [5] Tailwind Labs Inc. (2026). *Tailwind CSS - A Utility-First CSS Framework*. <https://tailwindcss.com/> adresinden erişildi.
- [6] SoftITo Bilişim Akademisi. (2026). *Bilişim Güvenliği ve Yazılım Geliştirme Akademisi Eğitim Materyalleri*. softITo Yazılım Bilişim Akademisi, İstanbul.