**Lojistik Entegrasyon ve Takip Sistemi - Proje Raporu / Teknik Tasarım Dokümanı**

**1. İhtiyaç Duyulan Uygulama Katmanları Yorumu**

* **Sunum Katmanı (Presentation Layer - UI):**
  + **Amaç:** Kullanıcılarla doğrudan etkileşimi sağlamak. Kullanıcı girdilerini almak, iş mantığı katmanından gelen verileri kullanıcıya anlaşılır bir şekilde sunmak.
  + **Neden Gerekli:** Kullanıcıların siparişleri görüntülemesi, durumlarını güncellemesi için görsel bir arayüze ihtiyaç vardır.
  + **Proje Tipi:** Logisticsv1.Web (ASP.NET Core MVC Web Uygulaması).
* **Uygulama Katmanı (Application Layer - Business Logic / Orchestration):**
  + **Amaç:** Uygulamanın iş kurallarını, iş akışlarını ve kullanım senaryolarını (use case'leri) barındırmak. Domain katmanındaki varlıkları kullanarak işlemleri yönetmek ve Infrastructure katmanından bağımsız kalmak.
  + **Neden Gerekli:** Projenin çekirdek iş mantığının (sipariş çekme, sipariş güncelleme, teslimat bildirme gibi) merkezi ve test edilebilir bir yerde konumlanması için kritik. Bu katman, UI veya dış sistemlerin iş mantığını doğrudan değiştirmesini engeller.
  + **Proje Tipi:** Logisticsv1.Application.Services.
* **Domain Katmanı (Domain Layer - Core Business Entities):**
  + **Amaç:** Uygulamanın temel iş varlıklarını (entities), değer nesnelerini (value objects), enumarasyonları ve bu varlıklar arasındaki ilişkileri tanımlamak. İş kurallarının en saf hali burada bulunur.
  + **Neden Gerekli:** İş mantığının kalbini oluşturur ve diğer tüm katmanların bağımlı olduğu en bağımsız katmandır. Uygulamanın temel tanımını içerir.
  + **Proje Tipi:** Logisticsv1.Domain.
* **Altyapı Katmanı (Infrastructure Layer - Data Access, External Services):**
  + **Amaç:** Veritabanı erişimi (ORM), harici servis entegrasyonları (API istemcileri), dosya sistemine erişim, loglama gibi teknik detayları ve dış bağımlılıkları yönetmek. Bu katman, Application katmanında tanımlanan arayüzlerin (interfaces) somut implementasyonlarını içerir.
  + **Neden Gerekli:** İş mantığının teknik detaylardan soyutlanmasını sağlar. Veritabanı teknolojisi değişse bile Application katmanının etkilenmemesini mümkün kılar.
  + **Proje Tipi:** Logisticsv1.Infrastructure (Entity Framework Core için), Logisticsv1.ApiClients.BCompany (Harici API'ler için).
* **Middle (Arka Plan Servisleri / Workers):**
  + **Amaç:** Kullanıcı arayüzünden bağımsız olarak veya zamanlanmış şekilde çalışan işleri yürütmek. Uzun süreli işlemler, periyodik görevler, kuyruk işlemleri gibi senaryolar için kullanılır.
  + **Neden Gerekli:** Sipariş çekme ve teslimat bildirimi gibi işlemlerin sürekli veya belirli aralıklarla, bir kullanıcı etkileşimi olmadan çalışması gerektiği için bu katmana ihtiyaç duyulmuştur. Bu, ana UI uygulamasının performansını ve yanıt verebilirliğini etkilemez.
  + **Proje Tipi:** Logisticsv1.OrderFetcher.Console (Zamanlanabilir konsol uygulaması), Logisticsv1.DeliveryNotifier.Worker (.NET Worker Service).

**Yorum:** Bu katman ayrımı, projenin **temiz mimari** prensiplerine uygun olarak geliştirilmesini sağlamıştır. Her katmanın belirli sorumlulukları vardır ve bağımlılıklar dıştan içe doğru (Sunum -> Uygulama -> Domain) akmaktadır. Altyapı katmanı, uygulama katmanındaki arayüzlerin implementasyonlarını sağlayarak bağımlılık yönünü tersine çevirir (Dependency Inversion). Bu, kodun daha **bakım dostu**, **ölçeklenebilir** ve **test edilebilir** olmasını sağlar.

**2. İş Tanımları (Projenin Ana Bölüm Tanımları, Aktiviteler)**

Bu bölümde, projenin ana iş akışlarını ve ilgili aktiviteleri detaylandırın.

**A. Sipariş Yönetimi (Web Uygulaması - Logisticsv1.Web)**

* **Amaç:** Lojistik operasyonlarının merkezi yönetimini ve sipariş takibini sağlamak.
* **Aktiviteler:**
  + **Siparişleri Görüntüleme:** Sistemdeki tüm siparişlerin listesini, temel bilgileriyle (Sipariş ID, Tarih, Alıcı Adı, Statü) birlikte görüntüleme.
  + **Sipariş Detaylarını İnceleme:** Belirli bir siparişin tüm detaylarını (ürünler, miktarlar, kargo bilgileri) görüntüleme.
  + **Sipariş Durumu Güncelleme:** Siparişlerin statüsünü "Bekliyor", "İşleniyor" veya "Teslim Edildi" olarak güncelleme.
  + **Teslimat Bilgisi Girişi:** Sipariş "Teslim Edildi" statüsüne geçtiğinde, plaka numarası ve teslim eden kişi bilgisini girme.

**B. Sipariş Entegrasyonu (Konsol Uygulaması - Logisticsv1.OrderFetcher.Console)**

* **Amaç:** Harici bir kaynaktan (B Şirketi) sipariş verilerini otomatik olarak sisteme çekmek.
* **Aktiviteler:**
  + **Harici API Sorgulama:** Belirli bir tarih aralığı için (örneğin dünün siparişleri) B Şirketi'nin SOAP servisini sorgulama.
  + **Veri Doğrulama ve Dönüştürme:** Çekilen sipariş verilerini sistemin kendi formatına dönüştürme ve temel doğrulama kontrolleri yapma.
  + **Veritabanına Kayıt:** Yeni veya güncellenmiş sipariş bilgilerini yerel veritabanına ekleme/güncelleme.
  + **Loglama:** İşlem sonuçlarını ve olası hataları loglama.

**C. Teslimat Bildirimi (Worker Servisi - Logisticsv1.DeliveryNotifier.Worker)**

* **Amaç:** "Teslim Edildi" olarak işaretlenen siparişlerin teslimat durumunu harici bir API'ye otomatik olarak bildirmek.
* **Aktiviteler:**
  + **Bekleyen Teslimatları Tespit Etme:** Veritabanında "Teslim Edildi" statüsünde olan ancak henüz harici API'ye bildirimi yapılmamış siparişleri belirleme.
  + **Bildirim Verisi Hazırlama:** Teslimat bilgileri (Sipariş ID, Plaka, Teslim Eden Kişi vb.) içeren bir bildirim nesnesi oluşturma.
  + **Harici API'ye Gönderme:** Hazırlanan bildirim nesnesini B Şirketi'nin Teslimat API'sine gönderme.
  + **Bildirim Durumunu Güncelleme:** Başarılı bir bildirimin ardından, veritabanındaki ilgili teslimat kaydını "bildirildi" olarak işaretleme.
  + **Hata Yönetimi ve Yeniden Deneme:** Bildirim hatalarını ele alma ve başarısız bildirimler için yeniden deneme mekanizması (worker servisi sayesinde otomatik).
  + **Loglama:** Tüm bildirim işlemlerini ve sonuçlarını loglama.

**3. Zaman Planlaması (Gantt Diyagramı hiç çizmedim.)**

Gantt diyagramında olması gereken ana görevleri ve süreleri belirtebilirim.

**Proje Başlangıç Tarihi:** [örn: 01.01.2025] **Proje Bitiş Tarihi:** [örn: 28.02.2025]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Görev ID** | **Görev Adı** | **Süre (Gün)** | **Başlangıç Tarihi** | **Bitiş Tarihi** | **Bağımlılıklar** | **Sorumlu** |
| 1. | **Planlama ve Analiz** |  |  |  |  |  |
| 1.1 | Gereksinim Analizi | 3 | 01.01.2025 | 03.01.2025 |  | Halil |
| 1.2 | Mimari Tasarım & Katmanlama | 2 | 04.01.2025 | 05.01.2025 | 1.1 | Halil |
| 1.3 | Veritabanı Tasarımı | 2 | 06.01.2025 | 07.01.2025 | 1.2 | Halil |
| 1.4 | Use Case Diyagramları | 1 | 08.01.2025 | 08.01.2025 | 1.1 | Halil |
| 2. | **Altyapı Geliştirme** |  |  |  |  |  |
| 2.1 | Temel Repository & UoW | 3 | 09.01.2025 | 11.01.2025 | 1.3 | Halil |
| 2.2 | DbContext & Migrations | 2 | 12.01.2025 | 13.01.2025 | 2.1 | Halil |
| 2.3 | Mock API İstemcileri (BCompany) | 3 | 14.01.2025 | 16.01.2025 | 1.2 | Halil |
| 3. | **Core & Domain Geliştirme** |  |  |  |  |  |
| 3.1 | Domain Varlıkları (Entities) | 2 | 09.01.2025 | 10.01.2025 | 1.2 | Halil |
| 3.2 | DTO'lar | 1 | 11.01.2025 | 11.01.2025 | 3.1 | Halil |
| 4. | **Uygulama Servisleri Geliştirme** |  |  |  |  |  |
| 4.1 | OrderService | 4 | 17.01.2025 | 20.01.2025 | 2.1, 3.2 | Halil |
| 4.2 | OrderImportService | 3 | 21.01.2025 | 23.01.2025 | 2.3, 4.1 | Halil |
| 4.3 | DeliveryNotificationService | 3 | 24.01.2025 | 26.01.2025 | 2.3, 4.1 | Halil |
| 5. | **Sunum Katmanı Geliştirme (Web UI)** |  |  |  |  |  |
| 5.1 | Sipariş Listeleme UI | 3 | 27.01.2025 | 29.01.2025 | 4.1 | Halil |
| 5.2 | Sipariş Detay & Güncelleme UI | 4 | 30.01.2025 | 02.02.2025 | 4.1 | Halil |
| 6. | **Arka Plan Servisleri Geliştirme** |  |  |  |  |  |
| 6.1 | OrderFetcher Konsol Uyg. | 2 | 03.02.2025 | 04.02.2025 | 4.2 | Halil |
| 6.2 | DeliveryNotifier Worker Servisi | 3 | 05.02.2025 | 07.02.2025 | 4.3 | Halil |
| 7. | **Testler ve Hata Ayıklama** |  |  |  |  |  |
| 7.1 | Birim Testleri | 5 | 08.02.2025 | 12.02.2025 | Tüm Geliştirme | Halil |
| 7.2 | Entegrasyon Testleri | 4 | 13.02.2025 | 16.02.2025 | 7.1 | Halil |
| 7.3 | Kullanıcı Kabul Testleri (UAT) | 3 | 17.02.2025 | 19.02.2025 | 7.2 | Halil |
| 8. | **Dokümantasyon & Dağıtım** |  |  |  |  |  |
| 8.1 | README.md Güncelleme | 1 | 20.02.2025 | 20.02.2025 | Tüm Geliştirme | Halil |
| 8.2 | Proje Raporu Hazırlama | 3 | 21.02.2025 | 23.02.2025 | Tüm Geliştirme | Halil |
| 8.3 | GitHub'a Yükleme | 1 | 24.02.2025 | 24.02.2025 | 8.1, 8.2 | Halil |
| 8.4 | Son Kontroller | 1 | 25.02.2025 | 25.02.2025 | 8.3 | Halil |

**4. Gereksinim Analizleri (Requirements)**

**A. Fonksiyonel Gereksinimler (Functional Requirements - FR)**

* **FR-1: Sipariş Entegrasyonu**
  + FR-1.1: Sistem, B Şirketi'nin SOAP servisi üzerinden günlük (veya belirlenen aralıkta) sipariş verilerini çekebilmelidir.
  + FR-1.2: Sistem, çekilen sipariş verilerini yerel veritabanına kaydedebilmelidir.
  + FR-1.3: Sistem, zaten mevcut olan siparişleri tekrar eklememeli, ancak güncellemeleri yönetebilmelidir (şu anki implementasyonda sadece ekleme var, bu maddeyi beklentinize göre düzenleyebilirsiniz).
  + FR-1.4: Sistem, çekilen siparişin ürün detaylarını da kaydedebilmelidir.
* **FR-2: Sipariş Takibi ve Yönetimi (Web UI)**
  + FR-2.1: Kullanıcılar, sistemdeki tüm siparişleri listeleyebilmelidir.
  + FR-2.2: Kullanıcılar, her bir siparişin detaylarını (ürünler, teslimat adresleri, iletişim bilgileri vb.) görüntüleyebilmelidir.
  + FR-2.3: Kullanıcılar, bir siparişin statüsünü "Bekliyor", "İşleniyor" veya "Teslim Edildi" olarak güncelleyebilmelidir.
  + FR-2.4: Sipariş statüsü "Teslim Edildi" olarak güncellendiğinde, kullanıcı plaka numarası ve teslim eden kişinin adını girebilmelidir. Bu alanlar zorunlu olmalıdır.
  + FR-2.5: Kullanıcılar, sistemdeki verilere güvenli bir web arayüzü üzerinden erişebilmelidir (Kimlik doğrulama/Yetkilendirme eklenmediyse, "Geliştirmenin ilk aşamasında basit erişim sağlanmaktadır, güvenlik daha sonra eklenebilir" gibi bir not düşebilirsiniz).
* **FR-3: Teslimat Bildirimi**
  + FR-3.1: Sistem, "Teslim Edildi" statüsüne geçen siparişlerin teslimat bilgilerini, henüz bildirim yapılmadıysa, B Şirketi'nin Teslimat API'sine otomatik olarak gönderebilmelidir.
  + FR-3.2: Sistem, başarılı bildirimlerin ardından ilgili teslimat kaydını "bildirildi" olarak işaretleyebilmelidir.
  + FR-3.3: Sistem, bildirim hatalarını loglayabilmeli ve belirlenen periyotlarda yeniden deneme yapabilmelidir.

**B. Fonksiyonel Olmayan Gereksinimler (Non-Functional Requirements - NFR)**

* **NFR-1: Performans**
  + NFR-1.1: Web arayüzü, sipariş listeleme ve detay görüntüleme işlemlerinde 2 saniye içinde yanıt vermelidir (varsayımsal).
  + NFR-1.2: Sipariş entegrasyonu işlemi, günde X adet siparişi Y dakika içinde tamamlayabilmelidir (varsayımsal).
* **NFR-2: Güvenilirlik**
  + NFR-2.1: Sistem, harici API bağlantı hatalarını (timeout, network error) yönetebilmeli ve loglayabilmelidir.
  + NFR-2.2: Teslimat bildirimi servisi, başarısız bildirimleri otomatik olarak belirli bir süre sonra tekrar denemelidir.
* **NFR-3: Ölçeklenebilirlik**
  + NFR-3.1: Uygulama katmanları ve servisler, gelecekteki iş yükü artışını karşılayacak şekilde bağımsız olarak ölçeklendirilebilir olmalıdır (mikroservis mimarisine geçişe uygunluk).
* **NFR-4: Bakım Kolaylığı**
  + NFR-4.1: Kod tabanı, temiz mimari prensiplerine uygun olmalı ve kolayca anlaşılabilir, değiştirilebilir ve genişletilebilir olmalıdır.
  + NFR-4.2: Proje, uygun yorumlama (comments) ve dokümantasyona sahip olmalıdır.
* **NFR-5: Güvenlik**
  + NFR-5.1: Hassas verilerin saklanması ve aktarılması için uygun güvenlik önlemleri alınmalıdır (şu anki kapsamda minimum, gelecekte geliştirme alanı).
  + NFR-5.2: Veritabanı bağlantı bilgileri appsettings.json gibi konfigürasyon dosyalarında saklanmalıdır.
* **NFR-6: Kullanılabilirlik**
  + NFR-6.1: Web arayüzü, sezgisel ve kullanıcı dostu olmalıdır.
  + NFR-6.2: Durum güncellemeleri ve bildirimler için kullanıcıya geri bildirim sağlanmalıdır (örneğin başarı/hata mesajları).

**5. Use Case Diyagramları**

**Temel Aktörler:**

* **Lojistik Çalışanı:** Web arayüzünü kullanan kişi.
* **Harici Sistem (B Şirketi):** Siparişleri sağlayan veya teslimatları alan sistem.
* **Sistem:** Lojistik Entegrasyon ve Takip Sistemi'nin kendisi.

**A. Sipariş Yönetimi Use Case Diyagramı (Web UI)**

* **Aktör:** Lojistik Çalışanı
* **Use Case'ler:**
  + Siparişleri Görüntüle
    - Sipariş Detaylarını Görüntüle (Extend veya Include ilişkisiyle "Siparişleri Görüntüle"ye bağlı)
  + Sipariş Durumu Güncelle
    - Teslimat Bilgisi Gir (Extend ilişkisiyle "Sipariş Durumu Güncelle"ye bağlı, sadece "Teslim Edildi" statüsü için)

**B. Sipariş Entegrasyonu Use Case Diyagramı (Konsol Uygulaması)**

* **Aktör:** Harici Sistem (B Şirketi - siparişleri sağlayan taraf) veya Zamanlayıcı (Operatör tarafından başlatılan/zamanlanan görev)
* **Use Case'ler:**
  + Siparişleri Çek
    - Sipariş Verilerini Dönüştür (Include ilişkisi)
    - Siparişleri Veritabanına Kaydet (Include ilişkisi)
    - Log Oluştur (Include ilişkisi)

**C. Teslimat Bildirimi Use Case Diyagramı (Worker Servisi)**

* **Aktör:** Sistem (Otomatik Çalışan)
* **Use Case'ler:**
  + Bekleyen Teslimatları Bildir
    - Bildirim Verisi Hazırla (Include ilişkisi)
    - Harici API'ye Bildirim Gönder (Include ilişkisi)
    - Bildirim Durumunu Güncelle (Include ilişkisi)
    - Log Oluştur (Include ilişkisi)

**6. Proje Tipi Yorumları (2 ve 4 Numaralı Maddeler İçin)**

Bu bölümde, projenizde kullandığınız farklı proje tiplerini neden seçtiğinizi detaylandırın.

**A. Web Uygulaması (ASP.NET Core MVC - Logisticsv1.Web) Yorumu**

* **Neden Seçildi?**
  + **Hızlı Geliştirme:** ASP.NET Core MVC, web tabanlı arayüzler için Microsoft'un modern ve güçlü bir framework'üdür. Model-View-Controller deseni, UI, iş mantığı ve veri erişimini ayırarak hızlı ve düzenli geliştirme imkanı sunar.
  + **Zengin UI Desteği:** HTML, CSS, JavaScript ve Razor View Engine ile zengin ve interaktif kullanıcı arayüzleri oluşturmak kolaydır.
  + **Geniş Ekosistem ve Topluluk:** .NET ekosistemi ve geniş topluluk desteği, sorun çözme ve kaynak bulma konusunda avantaj sağlar.
  + **RESTful API Yetenekleri:** MVC ile birlikte Web API yetenekleri de gelir, bu da gelecekte mobil uygulamalar veya diğer sistemlerle entegrasyon için API katmanı sunma esnekliği sağlar.
* **Projedeki Rolü:** Kullanıcıların siparişleri görüntülemesi ve durumlarını manuel olarak güncellemesi için bir yönetim paneli sağlamaktır.

**B. Konsol Uygulaması (.NET Konsol Uygulaması - Logisticsv1.OrderFetcher.Console) Yorumu**

* **Neden Seçildi?**
  + **Zamanlanmış Görevler İçin Uygunluk:** Harici sistemlerden veri çekme gibi periyodik veya zamanlanmış görevler için basit ve hafif bir çözümdür. Windows Görev Zamanlayıcısı (Task Scheduler) veya Linux Cron gibi araçlarla kolayca zamanlanabilir.
  + **Kaynak Verimliliği:** Bir web uygulamasına veya sürekli çalışan bir servise göre daha az kaynak tüketir, sadece çalıştığı zaman aktif olur.
  + **Bağımsız Çalışma:** Kullanıcı arayüzünden tamamen bağımsız çalışabilir.
  + **Basitlik:** Karmaşık bir arayüze ihtiyaç duymayan, tek bir amaca odaklanmış görevler için en basit proje tipidir.
* **Projedeki Rolü:** B Şirketi'nden sipariş verilerini otomatik olarak çekmek ve veritabanına kaydetmek gibi belirli bir görevi yerine getirmek.

**C. Worker Servisi (.NET Worker Service - Logisticsv1.DeliveryNotifier.Worker) Yorumu**

* **Neden Seçildi?**
  + **Arka Plan Süreçleri İçin İdeal:** Sürekli arka planda çalışması gereken, kuyrukları dinleyen, periyodik kontroller yapan veya uzun süreli işlemler yürüten görevler için tasarlanmıştır.
  + **Güvenilirlik ve Kurtarma:** İşletim sistemi servisleri olarak yapılandırılabilir, bu da uygulamanın yeniden başlatıldığında otomatik olarak başlamasını ve daha kararlı çalışmasını sağlar.
  + **Bağımsızlık:** Bir web uygulamasının performansını etkilemeden kendi iş mantığını arka planda yürütür.
  + **Microservices'e Uygunluk:** Küçük, tekil sorumluluklara sahip, bağımsız dağıtılabilir servisler oluşturmak için uygun bir modeldir.
* **Projedeki Rolü:** "Teslim Edildi" olarak güncellenen siparişlerin teslimat bildirimlerini harici API'ye otomatik ve güvenilir bir şekilde göndermek.