AHMET YESEVİ ÜNİVERSİTESİ
BİLİŞİM TEKNOLOJİLERİ VE MÜHENDİSLİK FAKÜLTESİ
YÖNETİM BİLİŞİM SİSTEMLERİ YÜKSEK LİSANS
DÖNEM PROJESİ

Tedarik Zincirlerindeki İletişim ve Güvenlik Sorunlarının Blok Zincir ve Akıllı Sözleşmeler Kullanılarak Aşılması

HAZIRLAYAN
Cihan COŞGUN
172172014

DANIŞMAN
Prof. Dr. Adem KURT

İstanbul / Mayıs 2018

ETİK İLKELERE UYGUNLUK BEYANI

Dönem proje yazma sürecinde bilimsel ve etik ilkelere uyduğumu, yararlandığım tüm kaynakları kaynak gösterme ilkelerine uygun olarak kaynakçada belirttiğimi ve bu bölümler dışındaki tüm ifadelerin şahsıma ait olduğunu beyan ederim.

Cihan COŞGUN

Tedarik Zincirlerindeki İletişim ve Güvenlik Sorunlarının
Blok Zincir ve Akıllı Sözleşmeler Kullanılarak Aşılması

Cihan COŞGUN

AHMET YESEVİ ÜNİVERSİTESİ
BİLİŞİM TEKNOLOJİLERİ VE MÜHENDİSLİK FAKÜLTESİ
YÖNETİM BİLİŞİM SİSTEMLERİ YÜKSEK LİSANS
DÖNEM PROJESİ

ÖZET

İşletmelerin ürün ve malzeme tedariki sırasında karşılaştığı en büyük sorunlardan birisi tedarikçi ile tedarik eden arasında doğru ve güvenli iletişim kurulması olarak tanımlanmaktadır. İşte bu iletişim ve güvenlik sorunun çözümü için projemizde Blokzincir (Blockchain) teknolojisini kullanacağız.

Anahtar Kelimeler: Tedarik Zincirinde Blokzincir Kullanımı, Tedarik Zinciri, Blokzincir, Hyperledger, Fabric

Danışman: Prof. Dr. Adem KURT

To Solve Communication and Security Problems in Supply Chains
With Block Chain and Smart Contracts

Cihan COŞGUN

AHMET YESEVİ UNIVERSITY
IT TECHNOLOGIES AND ENGINEERING FACILITY
MANAGEMENT INFORMATION SYSTEMS
MASTER DEGREE

ABSTRACT

One of the biggest problems faced by enterprises during product and material procurement is defined as establishing accurate and secure communication between supplier and supplier. We will use Blockchain technology in our project to solve this communication and security problem.

Keywords: Supplychain Problems, Supplychain, Blockchain, Hyperledger, Fabric
Advisor: Prof. Dr. Adem KURT

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

BÖLÜM I

GİRİŞ

1.1. Problem

İşletmeler bilindiği üzere mal veya hizmet üreterek kâr elde ederler, işletmeler mal veya hizmet üretimini gerçekleştirirken sürekli bir şekilde kendiside mal veya hizmet tedarik eder. İşletmelerin en önemli işlerinden biride mal veya hizmet tedarik etmektir.

Tedarik zinciri, bir işletmenin ürettiği mal veya hizmeti son kullanıcıya teslim edene kadar, işletme içinden ve dışından gerçekleştirdiği mal, hizmet alımı, taşımacılık vb. faaliyet süreçlerinin tümüne verilen addır.

Tedarik sürecinde, alım kararlarına yönelik profesyonel olmayan bir yaklaşım, maliyet tasarrufu fırsatlarının kolaylıkla gözden kaçmasına yol açabilir ve bu da önemli düzeyde finansal kayıplara neden olabilir (Arjan J.van Weele, 2014:Önsöz).

Tedarik maliyetlerinde yapılan her harcama, işletmenin kâr ve zarar tablosuna doğrudan katkıda bulunur ancak bunun tersi de doğru olabilir. Tedarik sürecinde, tedarikçi ile ürün tedariki için talepte bulunan ticari işletme arasındaki iletişimden kaynaklı oluşabilecek en küçük gecikme, yanlış ürün veya hizmet tedariki işletmeye telafisi zor masraflara veya itibar kayıplarına yol açabilmektedir.

1.2. Araştırmanın Amacı

Tedarikçi ve tedarik eden işletme arasında oluşan iletişim problemlerin önüne geçmek için işletmeler arasında güvenli ve hızlı bir iletişim alt yapısı kurulmalıdır, böylece işletmeler tedarik sürecini daha hızlı ve güvenli biçimde tamamlayabileceklerdir.

Bu kapsamda projenin konusu olan işletmeler arasındaki güvenli iletişimi sağlamak için Blok Zincir (Block Chain) teknolojisi tam bir biçilmiş kaftandır. Blok Zincir teknolojisi işletmeler arasında güvenli bir iletişim alt yapısı ve ortak bir veritabanı kurulmasını sağlamaktadır. Projemiz bu teknolojinin tedarik zincirine uyarlanması üzerine bir çalışma yapmayı amaçlamaktadır.

1.3. Araştırmanın Önemi

Ticari işletmelerde tedarik zinciri alt yapısı genellikle kurulmamıştır ve bu nedenle tedarik sürecinde sıklıkla iletişim sorunları meydana gelmektedir. İşte bu nedenle tedarik zincirlerinde güvenli bir iletişim alt yapısı kurulması ve bu konuda araştırma yapılması büyük önem arz etmektedir.

Biz bu araştırmamızda 2008 yılından itibaren teknoloji dünyasında sıkça kullanılan güvenli iletişim ve bilgi paylaşım sistemi olan “Blok Zincir”, İngilizce adı ile “Block Chain” teknolojisini kullanacağız. Böylelikle şifrelenmiş güvenli bir alt iletişim alt yapısı sağlayabileceğiz.

1.4. Sayıltılar

Projemizde bir adet market ve marketin ürün tedarik ettiği bir adet tedarikçi kurum olduğu ve iki kurum arasında herhangi bir iletişim alt yapısı olmadığı varsayılacaktır.

1.5. Sınırlılıklar

Projemizde kullandığımız teknolojiyi gerçek ortamda test edemeyeceğimiz için sanal bir ortam oluşturduk.

1.6. Tanımlar

Blok Zincir (Blockchain)

Veri tabanı kayıt işlemlerinin doğrulanmış ve güvenli bir şekilde bir ağ üzerinde depolandığı ve paydaşlara dağıtıldığı hesap defterini veya veri tabanını güvenli bir şekilde yönetme imkânı tanyan sisteme verilen addır. Tüm paydaşların kendi kopyasına sahip olduğu bir düğüm ağı tarafından yönetilir. Her bir düğüm, zincir ağına bağlı bir bilgisayar veya sunucudur.

Tedarik Zinciri (Supplychain)

Bir işletmenin bir ürün veya hizmet satışı sırasında, ürünün geçtiği tüm aşamalar yani yarı mamül tedariği, hizmet tedariği, taşımacılık vb.. tedarik zincirinin birer parçasıdır ve bu işlemler zincirinin tümüne verilen addır.

Hyperledger Fabric

Linux vakfı tarafından geliştirilen ve daha sonra IBM'in dahil olması ile yaygın kullanım alanı bulan blokzincir teknolojisine verilen addır.

BÖLÜM II

KAVRAMSAL ÇERÇEVE

Yapılan kavramsal araştırmalar neticesinde, güvenli iletişim problemi genel olarak tüm kurumlar arasında haberleşmenin gerekli olduğu tüm alanlarda karşımıza çıkmaktadır. Genellikle bankalar arası haberleşme alt yapılarındaki bu güvenli iletişim sorununun çözümü olarak blok zincir teknolojisinin kullanıldığını görüyoruz.

Ülkemizde, bankacılık alanında önemli isimlerden biri olan Akbank, Ripple'ın blok zincir ağına dâhil olduğunu ve blok zincir teknolojisini uluslararası para transferi işlemlerinde kullanacağını 2017 Nisan ayında açıkladı (Demirel, 2017:1).

Bunun yanında, 1990 yılında 13 kamu ve özel Türk bankasının ortaklığıyla kurulan Bankalar arası Kart Merkezi (BKM) blok zincir teknolojisini birden çok alanda ve bazı teknoloji ortaklarıyla deniyor. Dijital kimlik, akıllı sözleşmeler (Smart Contracts) ve

dağıtık kayıt defteri (Distributed Ledger) konularında blok zincir temelli denemeler yapan BKM, bu kapsamda “keklik” adında bir şifrelenmiş para birimi oluşturmuş.

Blok zincir teknolojisinin önde gelen şirketlerinden olan BlockEx şirketi, dünyanın en büyük tedarik zincirlerinden birini işleten OpenText şirketi ile birlikte tedarik zinciri finansmanını daha etkili hale getirmek için blok zinciri teknolojisini kullanmak için yeni bir ortaklık kurduklarını açıkladı (Lojiport, 2017:1).

İLGİLİ ARAŞTIRMALAR

Tedarik zincirinde blok zincir teknolojisi kullanımı ile ilgili daha önce yapılan araştırmalar şu şekildedir;

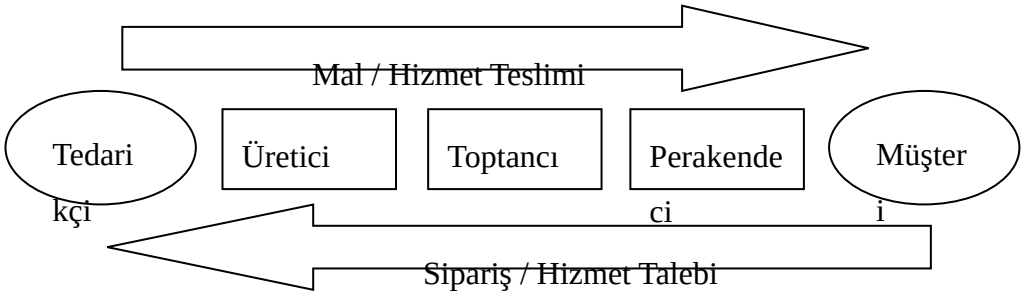
2016 yılında Loughborough Üniversitesi’nde A. Saveen Abeyratne ve Radmehr P. Monfared isimli araştırma görevlileri, “Üretim Alanında Hazır Blok Zincir ve Dağıtık Defter Teknolojisi Kullanımı” başlıklı bir araştırma yapmıştır, yaptıkları araştırmada üretim yapan işletmelerde tedarik zinciri aşamalarını dağıtık defter teknolojisini kullanarak birbirine bağlamayı amaçlamaktadırlar (Abeyratne, 2016:1).

BÖLÜM IV

1.7. Tedarik Süreci Aşamaları

Tedarik süreci, son kullanıcının işletmeden mal veya hizmet talep etmesi ile başlar, daha sonra işletme ürün veya hizmeti sağlamak için ihtiyaç duyduğu mamül, yarı mamül veya hizmetleri işletme içerisinden veya dışarıdan temin eder. Daha sonra tedarik ettiği mamül, yarı mamül veya hizmetleri, ürün veya hizmet üretiminde kullanır ve ürettiği ürünü veya hizmeti son kullanıcıya ulaştırır. Tedarik aşamasında, tedarikçi ile son kullanıcı arasındaki aracı sayısı kadar fazla ise o kadar karmaşık ve büyük bir tedarik zinciri oluşur aynı oranda tedarik süreci aşamaları da artar. Bu yüzden tedarik zinciri yönetimi çok dikkatli ve hataları en aza indirecek şekilde tasarlanmalıdır.

Şekil 4.1.1:Tedarik Zinciri

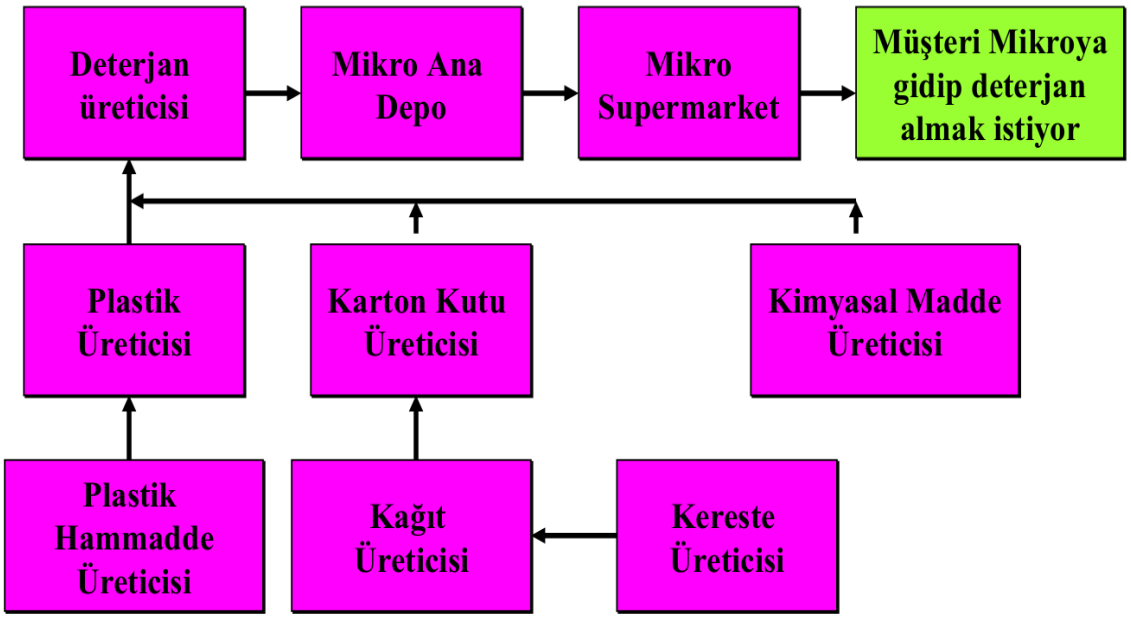


1.8. Tedarik Zinciri Yönetimi

İşletmeler envanter yönetimine, dağıtımına, depolamaya ve taşımacılığa yıllık bütçelerinden çok büyük meblağlar harcamaktalar. Bu maliyetleri ve süreçleri yönetmek için işletmelerde mutlaka tedarik zinciri profesyonelleri olması gereklidir. Tedarik zinciri oluşturmak ve sistemi kurmak her işletmenin ortak amacıdır ve bu oldukça zor bir iştir.

Tedarik zinciri kurulumu şu üç aşamadan meydana gelmektedir;

1. **Tedarik zinciri stratejisi ve tasarımı**
2. **Tedarik zinciri planlaması**
3. **Tedarik zinciri uygulaması**



Şekil 4.2.2 : Tedarik Zinciri Örneği

1.9. Tedarik Zinciri Yönetiminin Önemi

Günümüzün ekonomik şartları, sürekli değişen yapısı ve azalan üretim koşullarında tedarik zincirinde başarılı olan firmaların ayakta kalacağı ve bu konuda eski alışkanlıklarını sürdüren firmaların derin yaralar alacağı; belki kredi ile satın alınan stoklarının kolay kolay erimeyeceğinin fark edileceği ve satın alma – satış sürecinin kısaltılmasının öneminin derinden anlaşılacağı bir döneme girmiş bulunmaktayız. Üretim sürecinden bağımsız gibi gözüken ancak üretim, stok maliyetleri ile personel verimliliğinde en önemli rolü üstlenen tedarik zincirinin yalınlaştırılması ile satın alma ile oluşan giderin satış ile geri kazanılma sürecinin en aza indirilmesi ve süreç hızının arttırılması artık üretici firmalar için bir zorunluluktur (Yardımcı, 2017:1).

Yukarıda bahsedildiği gibi, işletmelerin tedarik zinciri yönetimindeki başarısı, işletmin hem finansal gücünü koruması ve güçlendirmesi hemde ürünleri son kullanıcılara eriştirmedeki başarısı sayesinde piyasadaki itibarı için son deece önemli bir yer tutmaktadır.

Yanlış yönetilen tearik zincirrleri fazla malzeme stoğu oluşmasına veya eksik/yanlış malzeme tedariğine neden olabilir bu gibi durumların işletmeler için ciddi sonuçları olabilir.

Örneğin; Aşşağıdaki tabloda bir üretim firmasında ürünün son kullanıcıya teslim edilmesne kadar olan süreçte işletmeye maaliyetlerinin yüzdesel olarak gösterimi belirtilmiştir.

Kâr	%4
Lojistik Maliyeti	%21
Pazarlama Maliyeti	%27
Üretim Maliyeti	%48

Tablo 4.3.1 : Örnek Maliyet Tablosu

1.10. Tedarik Zinciri Stratejisi ve Tasarımı

Tedarik zincirinin yapısının belirlenmesi ve her aşamada hangi süreçleri gerçekleştirileceği net olarak kararlara bağlanmalıdır. Stratejik olarak tedarik zinciri karları, üretim / depolama tesislerin yerleri ve kapasiteleri, üretilecek ve depolanacak ürünlerin belirlenmesi, hangi nakliye ve taşıma yollarının kullanılacağının belirlenmesi, kullanılacak bilgi işlem sistemlerinin belirlenmesi, gibi önemli konularda tedarik zincirinin stratejik hedeflerine ve planlarına en uygun şekilde tasarlanmalıdır.

İşte bu tasarım kararlarındaki en küçük hata uzun vadede işletmeye büyük zararlar verebilir, bu tip yanlış kararlardan geri dönüş işlemleri de yine işletmeye büyük maddi külfet getirecektir.

1.11. Tedarik Zinciri Planlaması

Tedarik zinciri tasarımının ilk aşaması doğru ve güvenilir bir planlama yapılması ile başlar, tedarik zinciri planlaması yapılırken, hangi ürün veya hizmetlerin, hangi tedarikçilerden tedarik edileceği, stokların hangi aşamalarda takviye edileceğinin planlanması, taşeron üreticilerin veya tedarikçilerin belirlenmesi, yedeklemeler. stok ve envanter politikaları planlanmalı, bu planlama yapılırken elbette planlamacının gelecekte ki rekabet koşullarına, döviz kurlarındaki değişikliklere ve taleplerde oluşabilecek belirsizliklere dikkat etmesi gerekmektedir.

1.12. Tedarik Zinciri Analizi

Tedarik zinciri planlaması yapıldıktan sonra yapılan planlamaların firmanın tedarik zinciri gereksinimlerini karşılayıp karşılayamadığına, mevcut piyasa koşullarına uyumluluğu, ekonomik ve verimli olup olmadığı gibi kriterlere göre analizlerinin yapılması gerekmektedir. Analiz sürecinde belirlenen eksiklikler veya hatalar tespit edilerek tekrar

planlama aşamasına geri dönülmeli ve bu döngü mükemmel planı yapana kadar tekrarlanmalıdır.

1.13. Tedarik Zinciri Uygulaması

Tedarik zinciri uygulaması, yapılan planlama ve analizler sonucunda belirlenen en iyi planlamanın gerçekleştirilmesidir. Tedarik zinciri uygulaması sadece müşteri siparişleri ile başlayan bir süreç zinciri değildir aynı zamanda firmanın belirli periyodik süreçlerde (aylık, haftalık, günlük) stok kontrol işlemleri, sayım işlemleri, üretim işlemleri gibi işlemleri de belirlenen plana uygun şekilde uygulanması süreçleridir.
Müşteri siparişleri alındıktan sonra eldeki stok miktarları kontrol edilmeli ve eksik miktarlar için ilgili tedarikçi firmalara sipariş geçilmeli veya üretim süreci başlatılmalı ve eğer eksik malzeme veya hizmetler varsa gerekli çalışmalar yapılmalıdır.

1.14. Tedarik Zincirinde Kullanılan Yaklaşımlar

Tedarik zinciri tasarımı ve yönetiminde belirli yaklaşımlar söz konusudur, bu yaklaşımlar şu şekilde sıralanabilir;

1. **Döngü yaklaşımı:** tedarik zincirindeki süreçler döngülere ayrılır. Her bir döngü tedarik zinciri kademelerinin arasında bulunur.

   1.1. Müşteri Sipariş Döngüsü

   1.1.1. Müşteri siparişinin alınması ve yerine getirilmesi ile ilgili tüm işlemler
   1.1.2. Müşteri varış
   1.1.3. Müşteri sipariş girişi
   1.1.4. Müşteri siparişinin karşılanması
   1.1.5. Malın müşteriye teslimi

   1.2. Yerine Koyma Döngüsü

   1.2.1. Perakendecinin stoklarını tamamlamak için gerekli olan tüm işlemler (şimdi müşteri perakendecidir)
   1.2.2. Perakendeci sipariş ihtiyacının doğması
   1.2.3. Siparişin girilmesi
   1.2.4. Sipariş karşılanması
   1.2.5. Siparişin perakendeciye varışı

   1.3. Üretim Döngüsü

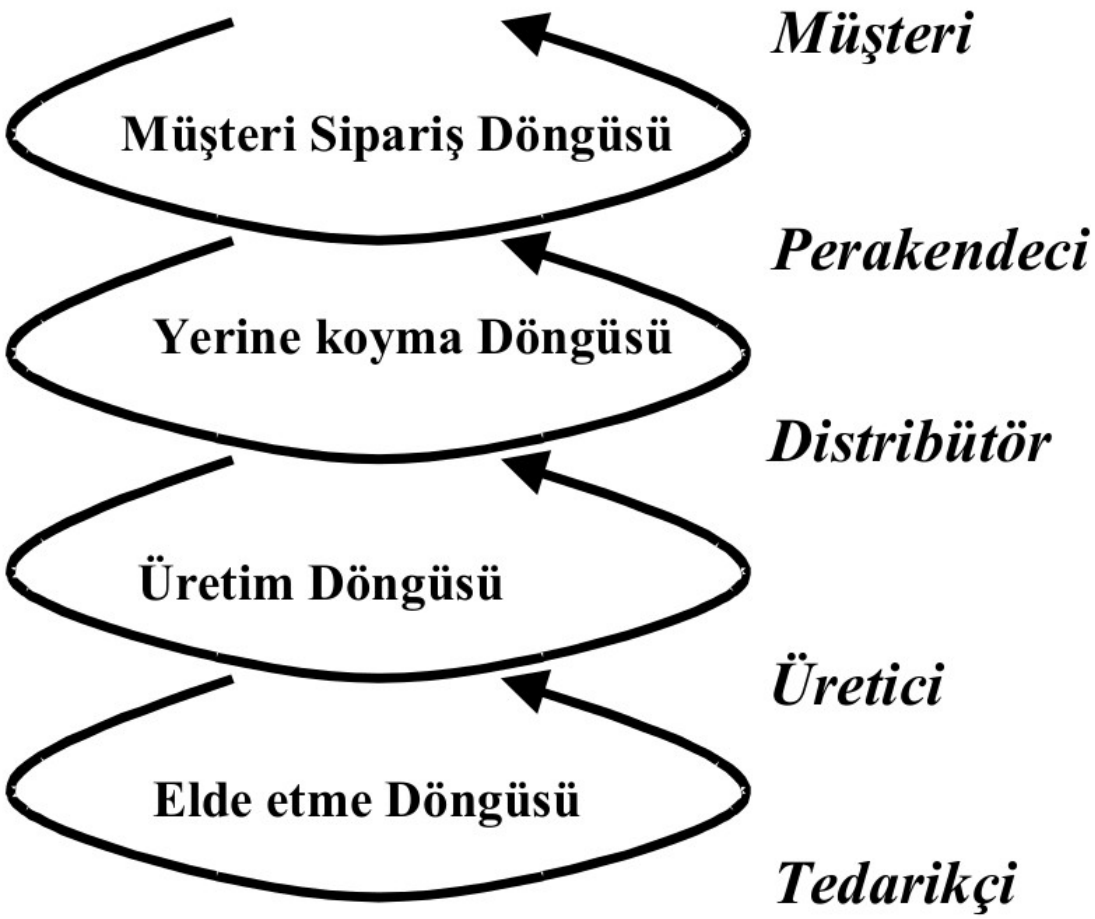
1.3.1. Distribütörün (veya perakendecinin) stoklarını tamamlamak için gerekli olan tüm işlemler.

1.3.2. Siparişin gelmesi (distribütörden / perakendeciden yada müşteriden)

1.3.3. Üretim ve Yükleme

1.3.4. Malın distribütör, perakendeci veya müşteri tarafından teslim alınması.

1.4. Tedarik Döngüsü



Şekil 4.8.3 : Tedarik Zinciri Döngü Yaklaşımı

2. **İtme/Çekme yaklaşımı:** Tedarik zincirinin süreçleri iki kategoriye ayrılır:

2.1. Müşteri siparişi ile başlatılan süreçler (İtme),

2.2. Müşteri siparişi beklentisiyle başlatılan süreçler (Çekme)

BÖLÜM V

1.15. Malzeme Kaynak Planlanması (MRP)

Bilindiği üzere iyi bir tedarik zinciri yönetimi ancak iyi bir tedarik zinciri tasarımı ile mümkündür bu konuda ise teknolojik imkânlardan ve yeniliklerden faydalanmak hayati önem arz etmektedir. İşte bu noktada uzun zamandır aktif olarak, tedarik zincirinde başarı sağlamış işletmeler tarafından kullanılmakta olan MRP yazılımları hayat kurtarıcı olarak karşımıza çıkmaktadır.

Türkçe karşılığı "Malzeme Kaynak Planlanması" olan (MRP, Material Resource Planning) uygulamaları üretim süreçlerini yönetmek için kullanılan bir üretim planlama, zamanlama ve envanter kontrol sistemidir. Çoğu MRP sistemi yazılım tabanlıdır, ancak MRP'yi elle de yapmak mümkündür.

Bir MRP sistemi üç şeyi amaçlamaktadır:

- Malzemelerin üretim için mevcut olması ve müşterilere teslim edilmek üzere ürünlerin mevcut olduğundan emin olmak
- Mağazadaki mümkün olan en düşük malzeme ve ürün seviyesini korumak
- Üretim faaliyetlerini, teslimat programlarını ve satın alma faaliyetlerini planlamak

1.16. Kurumsal Kaynak Planlaması (ERP)

Kurumsal Kaynak Planlaması (Enterprise Resource Planning-ERP) işletmelerin, farklı birim ve departmanlarındaki iş süreçlerini (muhasebe, finans, insan kaynakları, üretim, pazarlama, satın alma, lojistik, satış, envanter yönetimi, stok kontrol ve ambar yönetimi, müşteri ilişkileri yönetimi, proje yönetimi gibi) tek bir program ile işletmeye özel ve tek veri tabanı ile bütünleştiren, tüm departman ve birimlerin bu bütünleşik platformu kullanarak ilgili veriye ulaşarak sürece dahil etmesini ve bütünleşik veri altyapısının gerekli şekilde güncellenmesini sağlayan bir bilgi sistemidir (Aktaş, 2010:5).

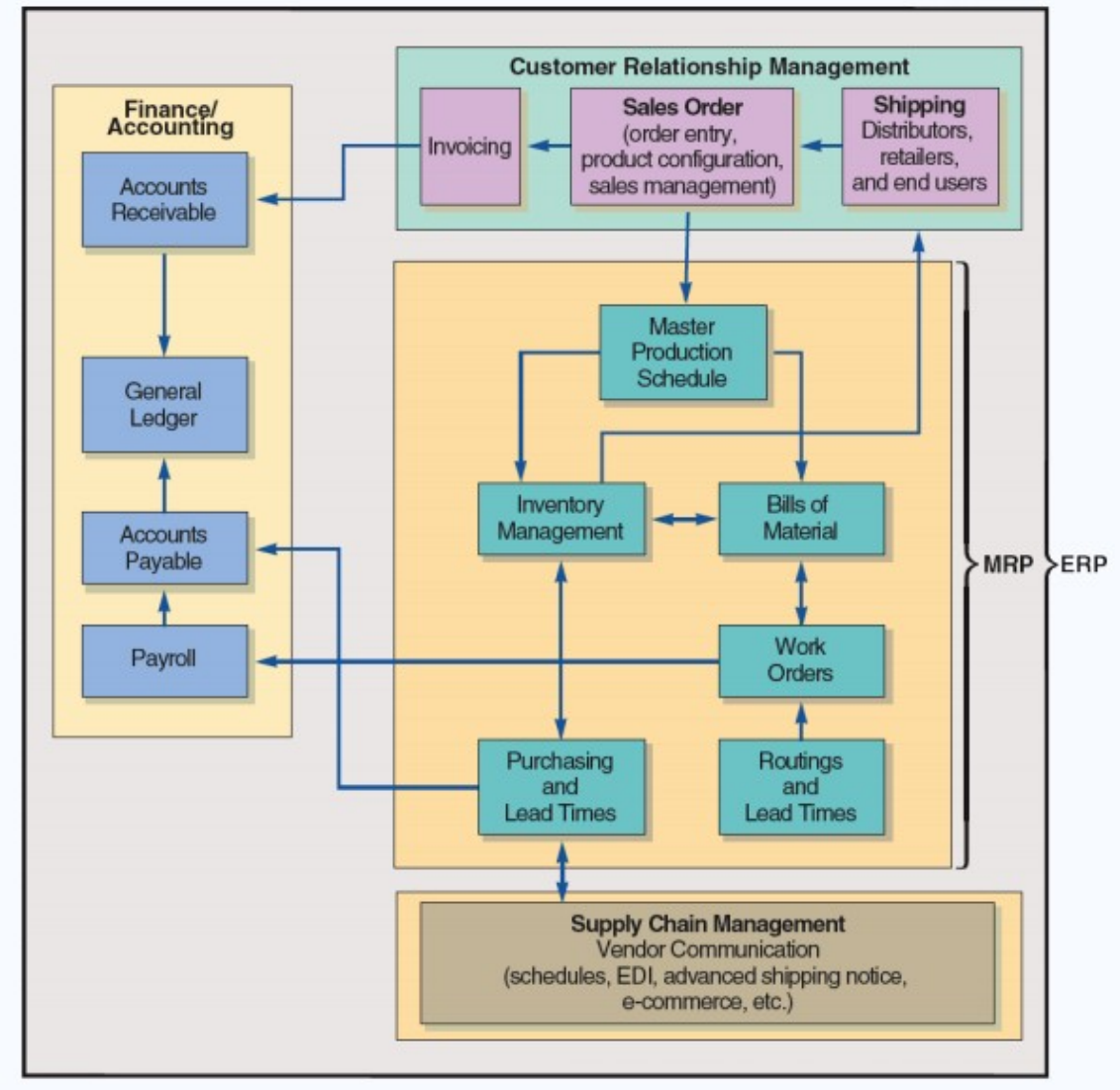
ERP sistemleri, işletmelerin bilgi paylaşımlarını kolaylaştırmak ve operasyonel verimliliği arttırmak için kurulan ve işletme genelindeki ortak kaynaklara erişimi hızlandırarak iş süreçlerini standardize eden bir

araçtır. Sistemin çalışmasını sağlayan en önemli unsur entegrasyondur. Entegrasyon tüm birimlerin operasyon süreçlerinde iş birliği sağlayarak bölüm ve Departmanl1ar arasındaki uyumu arttırmaktadır.

1.17. ERP – MRP İlişkisi

ERP uygulamaları ile yürütülen sipariş alma ve satışa çevirme süreci sonucunda oluşacak olan ürün ihtiyacını karşılamak için MRP uygulaması devreye girmektedir.

Satış sürecinin tamamlanabilmesi için ürünün başarılı bir şekilde üretilip satın alan müşteriye ulaştırılabilmesi gerekmektedir.



Şekil 5.3.4 : ERP ve MRP Kapsamı (Maines, 2017:1)

BÖLÜM VI

1.18. Tedarikçi İle İletişimin Önemi ve Güvenlik Sorunları

Buraya kadar anlatılanlar işletmenin kendi içerisinde yaşadığı sorunlar ve çözümleri ile ilgili süreçlerdi. Ancak siparişin tedarikçilere doğru, eksiksiz ve tam zamanında iletilmesi satış ve tedarik sürecinin tamamlanması için hayati önem taşımaktadır.

Örneğin bir araç galerisinden, bir müşteri araç satın almak istedi ve firmaya araç siparişi geçti. İşte bu aşamada firmanın ERP sistemi devreye girip satış sürecini başlatır, satış sürecinde eğer araç firmanın envanterinde yok ise tedarikçi firma ile iletişime geçilir ve tedarikçi firmadan araç sipariş edilir.

Varsayalım bu firma araç tedariği için herhangi bir yazılım kullanmıyor ve tedarik sürecini telefonla veya başka bir iletişim kanalı kullanarak sağlıyor olsun. İşte bu aşamada bir yanlış anlaşılma sonucu müşterinin istediği araç yerine farklı model bir araç gelmesi durumunda, siparişin yeniden düzenlenmesi, ürün değişimi gibi bir sürü sürece en baştan başlanacak ve ürünün zamanında teslim edilememesi ile birlikte aşağıdaki sorunlar yaşanacaktır;

1. **Ürünün zamanında teslim edilememesi**
2. **Müşterinin gözünde firmanın değerinin düşmesi**
3. **Firma için tedarik maliyetlerinin ikiye katlanması**
4. **Sözleşme şartları gereği firmanın ürünü zamanında teslim edememesinden kaynaklı üründe indirime gidilmesi ve firmanın satıştan zarar etmesi**

İşte bu sorunların yaşanmaması için firmalar ile tedarikçi firmalar arasında bir iletişim ve yazılım alt yapısı kurulmalıdır.

1.19. Tedarik Sürecinde Firma ile Tedarikçi Arasında İletişim

Yukarıdaki sorunların oluşmaması için firma ile tedarikçi firma arasında iletişim alt yapısı kurulmalı, bu iletişim alt yapısını kurarken kesinlikle uygun yazılımlar kullanılmalı. Böylelikle firmanın kendi yazılımları ile tedarikçi firmanın kendi içerisinde kullandığı yazılımlar birbiri ile otomatik olarak doğru, güvenli ve ortak bir alt yapıyı kullanacakları

için insan kaynaklı bir iletişim sorunu oluşmayacağından ürünlerin zamanında ve eksiksiz tedariki sağlanacaktır.

1.20. Tedarikçi Firma ile Firma Arasında Güven Sorunu

Tedarikçi firma ile firma arasında kurulan bir yazılım alt yapısında verilerin hangi tarafta tutulacağı bir güven sorunu teşkil edebilir, herhangi bir nedenle oluşabilecek olan bir anlaşmazlık neticesinde iki firmada birbirini suçlayabilir ve konu içinden çıkılmaz bir hukuki duruma dönüşebilir.

Bu nedenle iki firmada da aynı verilerin birer kopyası kesinlikle bulunmalıdır, ancak bu iki veritabanının da birbiri ile tıpatıp aynı olmalıdır ve verilerin herhangi bir şekilde değiştirilmesinin mümkün olmaması gereklidir.

BÖLÜM VII

1.21. Tedarik Zincirinde Blok Zincir Teknolojisi

Yukarıda bahsettiğimiz güven ve iletişim ortamının başarılı şekilde sağlanabilmesi için blok zincir teknolojisi adeta bir biçilmiş kaftan gibi karşımıza çıkmaktadır.

1.22. Blok Zincir (Block Chain) Nedir ?

Bir blok zinciri, dağıtık bir hesap defteri veritabanını ya da bir hesap hareketi kaydını temsil eden veri yapısıdır. Her hesap hareketi, gerçekliğini ve değiştirilemezliğini korumak için dijital olarak imzalanır ve kimse bu kayda müdahale edemez çünkü eğer herhangi bir kayda müdahale edilirse verinin tamamı bozulacaktır. Böylece hem hesap defterinin kendisi hem de içindeki hesap hareketlerinin güvenilir olduğu varsayılır.

Blok zincirin bir diğer avantajı da verilerin tek bir paydaşta değil, tüm paydaşlarda birer kopya halinde saklanıyor oluşudur, böylelikle her paydaş kendindeki verinin orijinal ve değiştirilemez olduğunu bilir ve paydaşlar arasındaki güven ortamı tesis edilir.

1.23. Blockchain Tarihçesi

2012’den beri kullanılan, herkes tarafından desteklenen ve kimsenin kontrolünde olmayan dijital para birimi olan Bitcoin dijital para biriminin sistemini bugüne dek ayakta tutan sistem blok zinciri (Blockchain) teknolojisidir. Son zamanlarda, teknoloji endüstrisinden pek çok farklı kuruluş bu teknolojiye ilgi göstermeye başladı. Ancak blockchain teknolojisi teknoloji piyasalarında Bitcoin dijital para birimi ile gündeme geldi.

1.24. Bitcoin Nedir ?

Bitcoin, yaratıcı bir ödeme ağı ve yeni bir para birimidir. Bitcoin eşler arası teknolojiyi kullanarak merkez otorite veya banka olmadan çalışır. İşlemlerin yönetimi ve bitcoinlerin dağıtımı toplu olarak ağ tarafından idare edilir. Bitcoin açık kaynaklıdır; tasarımı halka açıktır, kimse Bitcoin'e sahip değildir ve onu kontrol edemez, herkes katılabilir. Bitcoin kendine has birçok özelliği sayesinde diğer ödeme yollarıyla yapılamayacak çok farklı ödemelerin üstesinden gelebilir. (Nakamoto, 2011:1)

Mevcut bankacılık sisteminde önce herhangi bir kişi, kurum veya banka bir başka kişi, kurum veya bankaya para göndermek için ortak ve merkezi bir bankaya para göndermek zorundadır ve buda aracılık sağlayan bu merkezi bankaya belirli ve yüksek ücretlerin ödenmesini zorunlu hale getirmektedir. İşte bu aşamada Bitcoin çözümü ise herhangi bir merkezi kuruluşa olan ihtiyacı ortadan kaldırmayı amaçlamaktadır. Böylelikle bir kişi, kurum veya banka sistemdeki eşlerden herhangi birine bir transfer geçrekleştirdiğinde eşler arasındaki bağlantıdan dolayı tüm eşler belirli bir süre sonra o işlemi kendi veritabanlarına işleyecek ve böylelikle paranın transferi ve transfer işleminin güvenliği sağlanmış olacaktır.

Bitcoin avantajları şu üç başlıkta açıklanabilir;

1. Anında eşler arası işlemler
2. Dünya çapında ödemeler
3. Bedava veya çok ucuz gönderim ücreti

Şekil 7.4.5: Bitcoin

1.25. Blockchain Avantajları

Blok zinciri teknolojisinin sağladığı üç kolaylık bu sistemin tercih edilmesinde etkili oluyor:

1. Bir otoriteye ve aracıya ihtiyaç duyulmaması hem maliyetleri düşürüyor hem de işlemleri hızlandırıyor.
2. İşlemlerin pek çok farklı nokta tarafından kontrol ediliyor olması, sistemde sahtekârlık yapılması ihtimalini azaltıyor.
3. Blok zinciri, bir varlığın hangi kaynaktan çıkıp hangi kişilerin elinden geçip nereye ulaştığını takip etmek için ideal bir platform.

1.26. Blockchain Dezavantajları

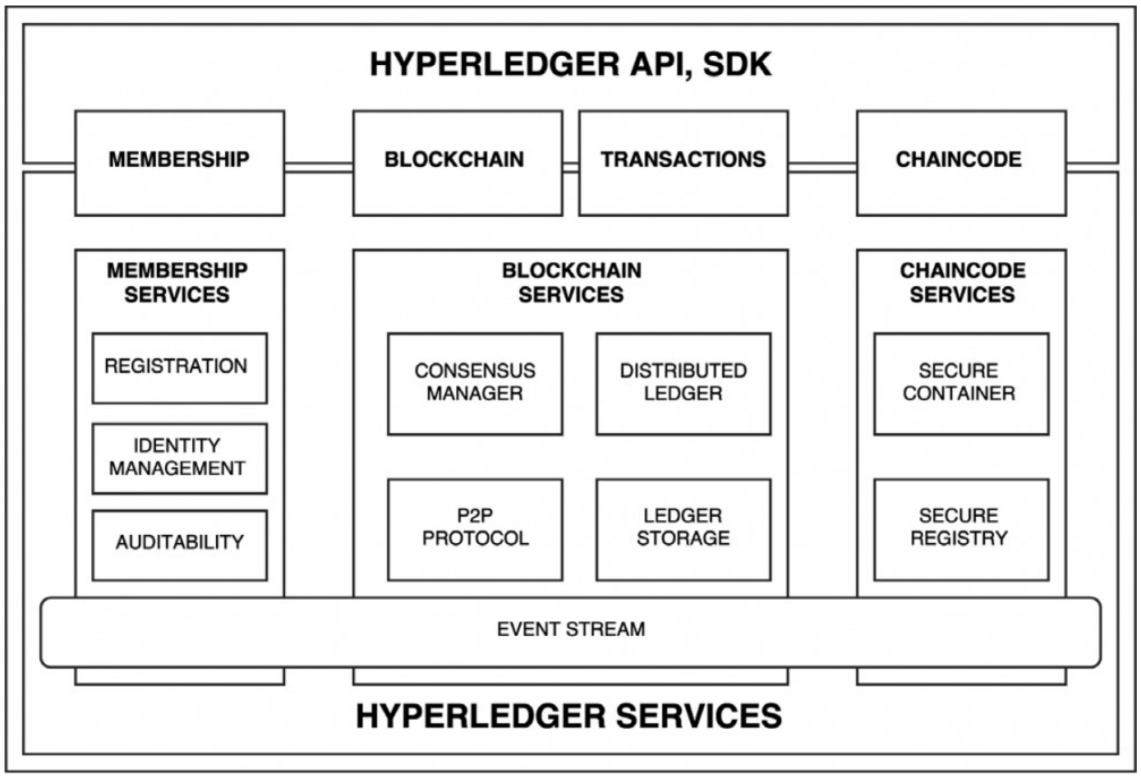
Blockchain sisteminin en büyük dezavantajlarından biri ise uygulamasının zor oluşudur. Çünkü bu açık kaynaklı sistem pek çok farklı yazılım şirketleri veya grupları tarafından, farklı hedefler için, farklı şekilde geliştirilmiştir. Bu nedenle bir standardı yoktur. Bu probleme çözüm bulmak amacıyla, aralarında IBM, Cisco, Fujitsu gibi büyük teknoloji firmalarının ve J.P. Morgan, Accenture gibi finans kuruluşlarının bulunduğu bir grup şirket, Hyperledger adlı bir sistem kurmayı amaçlıyor. Açık kaynaklı farklı blok zinciri uygulamalarını tek çatı altında toplamak amacıyla bir konsorsiyum kuran şirketler, sektörler arası para aktarımını sağlayacak dev bir altyapı oluşturmak istiyor. Bu altyapı ile internet dünyasında, finans alanında farkındalık yaratılabileceği düşünülüyor.

BÖLÜM VIII

1.27. Hyperledger Nedir ?

Hyperledger Linux Vakfı tarafından 2015 yılınında geliştirilmeye başlanan açık kaynak kodlu bir Blockchain platformudur. Hyperledger tek bir Blockchain yapısı oluşturmak yerine kendi içerisinde farklı alt projeler oluşturulmasına imkan sağlamaktadır.

Hyperledger, mimarisi iki ana kısımdan oluşmaktadır, bunlar Hyperledger servis katmanı ve bu servisleri dış dünyanın kullanımına açan Hyperledger API/ SDK katmanıdır (API: Application Programming Interface – Uygulama Programlama Arayüzü, SDK: Software Development Kit – Yazılım Geliştirme Kiti). (HyperLedger, 2015:1)



Şekil 8.1.6: Hyperledger Mimarisi

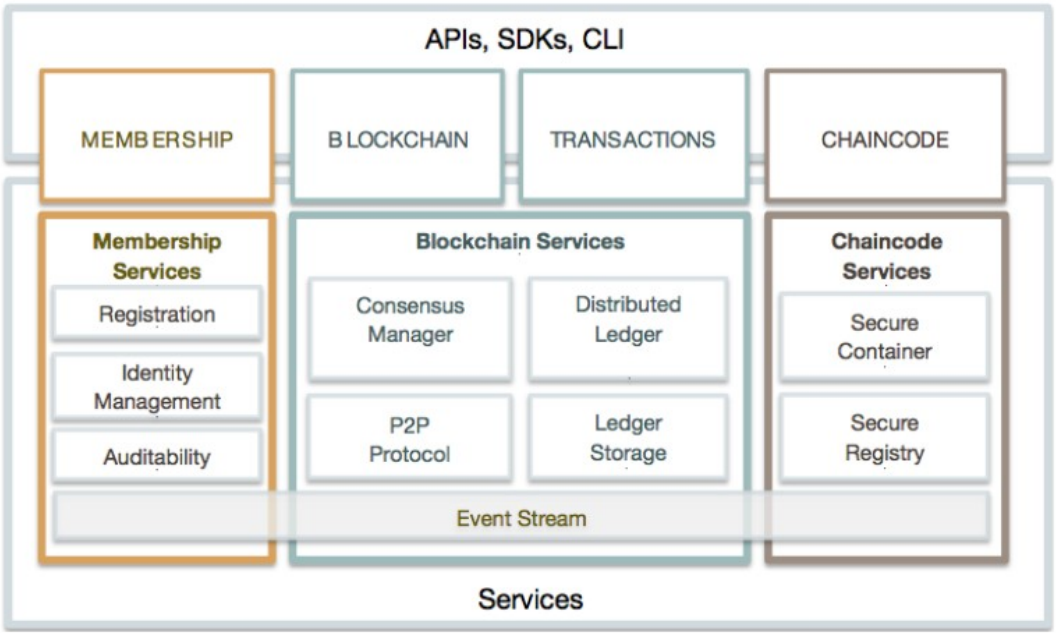
Hyperledger servis katmanı üç ana kategoride değerlendirilmektedir,

1. **Blockchain Servisleri**
2. **ChainCode (Akıllı Sözleşmeler)**
3. **Üyelik Servisleri**

Üyelik servisleri kimlik, gizlilik gibi konularda hizmet verirken Blockchain servisleri sahip olduğu P2P protokolü ile birlikte içerdiği blockchain ve mutabakat yapısını yönetmektedir. Chaincode servisleri, Hyperledger mimarisi içerisinde akıllı sözleşmelerin yönetim ve işletimini sağlamaktadır. Ayrıca alt seviyede bulunan bir haberleşme katmanı ile birlikte servis katmanı içerisinde olay güdümlü (event driven), çift yönlü etkileşim sağlanabilmektedir. Her ne kadar bu kavramlar ve yapılar kulağa oldukça karışık gelse de Hyperledger iş dünyasının ihtiyaç duyduğu temel unsurları bünyesinde sağladığı için bu gün pek çok Blockchain projesinde kullanılmaktadır. Günümüzde Bankalararası Kart Merkezi'nin gerçekleştirdiği kavram kanıtlama çalışması olan özel BBN Blockchain projesinde de Hyperledger platformu kullanılmıştır.

Hyperledger kapsamındaki projelerin en bilinenlerinden olan Fabric projesi IBM ve Digital Asset tarafından Hyperledger bünyesinde düzenlenen ilk Hackathon kapsamında önerilip hayat geçmiştir. Fabric projesinin en önemli özelliklerinden bir tanesi modüler mimarisidir,

bu sayede mutabakat, üyelik servisleri gibi Blockchain modülleri ihtiyaçlara göre tak-çalıştır (plug-and-play) felsefesi ile değiştirilebilmektedir.



Şekil 8.1.7 : Hyperledger Fabric Mimarisi (Keinzler, 2015:1)

BÖLÜM IX

1.28. Problemin Tanımı

Tedarikçi işletme ile tedarik eden işletme arasında doğru ve güvenli iletişim alt yapısı kurulması tedarik işletmenin eksik ve kesintili ürün üretimine neden olacaktır ve buda işletmenin hem itibarına hem de finansal durumuna büyük zararlar vermektedir.

1.29. Çözüm

İşletmeler arasında kurulacak olan özel bir Blokzincir alt yapısı ile işletmeler birbiri ile kesintisiz, güvenli ve doğru bir iletişim alt yapısına sahip olacaktır, böylelikle tedarikçi işletme tedarik ettiği malzemeleri diğer işletmeye sorunsuz olarak iletecek, tedarik eden işletme de tedarik ettiği malzemeleri sorunsuz olarak stoğuna alabileceği gibi ürünlerini de zamanında ve eksiksiz bir şekilde müşterilerine ulaştırabilecektir.

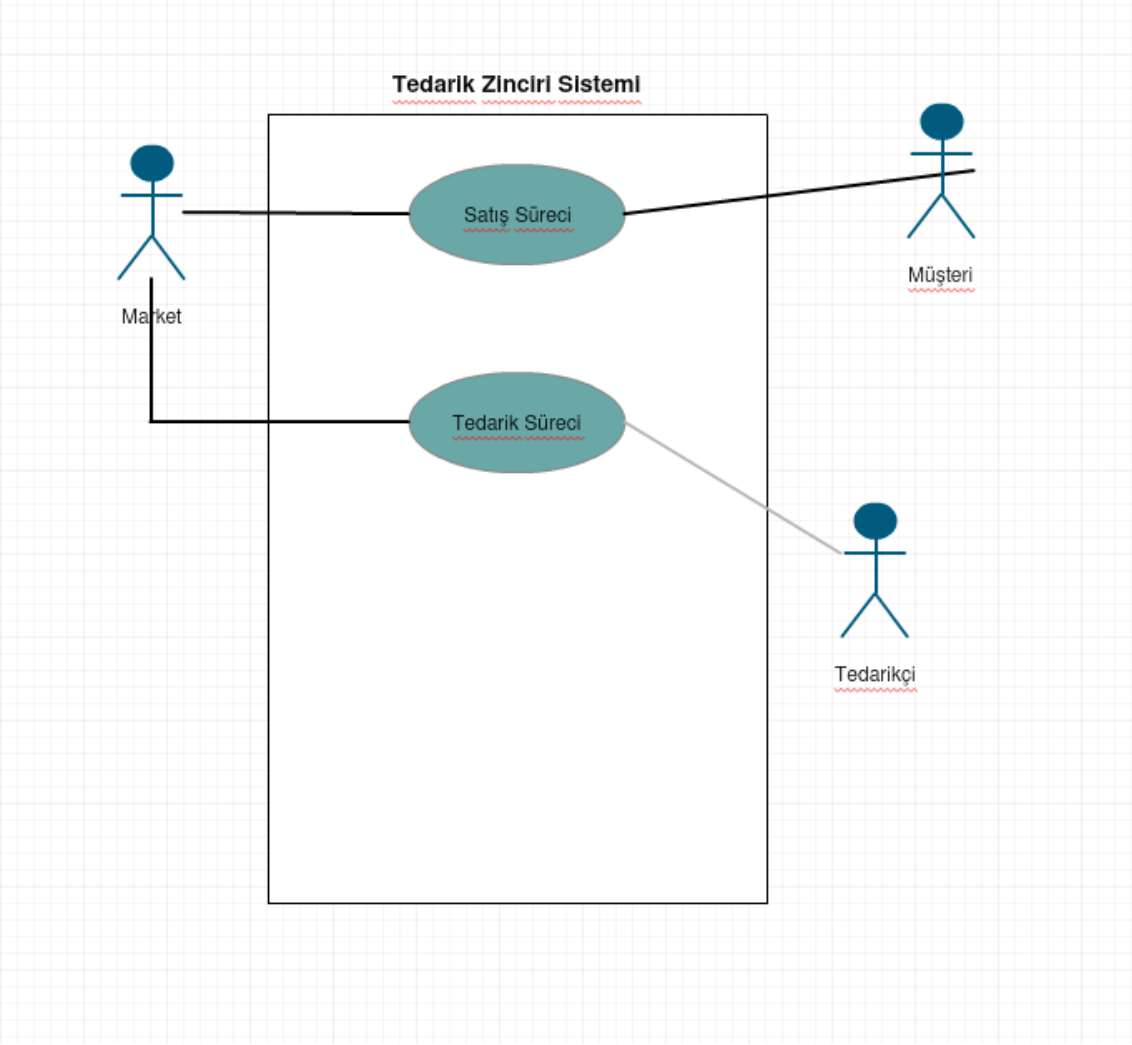
1.30. Proje Senaryosu

Projede bir market ve o markete ürün tedarik eden bir tedarikçi firma olduğu varsayılacaktır, marketin kullandığı ERP otomasyonu ürün stoklarında kritik miktarların altına düşen ürünleri tedarikçi işletmeye, kuracağımız Blokzincir alt yapısı üzerinden sipariş geçecektir. Tedarikçi firma siparişleri Blokzincir alt yapısı üzerinden sorgulayacak ve kendi ERP otomasyonlarındaki sipariş ekranlarında görecek ve siparişi hazırlayarak sipariş durumunu güncelleyecektir. Market, güncellenen sipariş bilgilerini anında sorgulayabilecek ve mal kabulünü gerçekleştirdikten sonra siparişin durumunu “kabul edildi” olarak güncelleyebilecektir. Böylelikle projemizde bahsedilen iletişim ve güvenlik sorunu çözülmüş olacaktır.

1.31. Market ERP Uygulaması

Market ERP uygulaması aşağıdaki modülleri içerecektir;

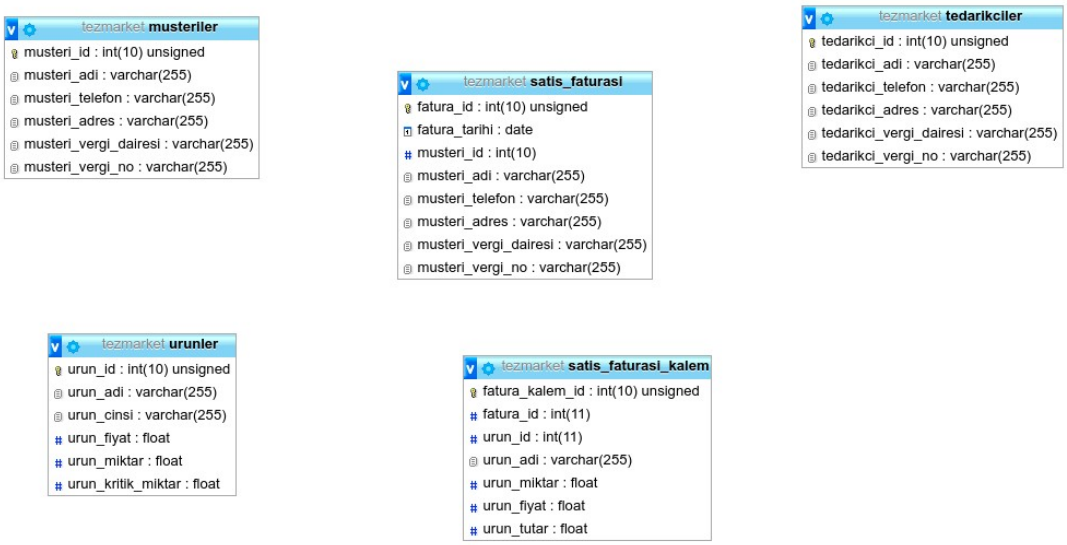
1. **Ürünler**
2. **Müşteriler**
3. **Tedarikçiler**
4. **Satış**
5. **Sipariş**



Şekil 9.4.8 : Tedarik Zinciri Use Case Diagramı

Market Veritabanı

Market veritabanı olarak MySQL veritabanı kullanılacaktır.



Şekil 9.4.1.9 : Market ERP Veritabanı Şeması

Veritabanı Oluşturma Kodu

CREATE DATABASE tezmarket CHARACTER SET utf8 COLLATE utf8_general_ci

Ürünler Tablosu Kodu

CREATE TABLE urunler (urun_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY, urun_adi VARCHAR(255), urun_cinsi VARCHAR(255), urun_fiyat FLOAT, urun_miktar FLOAT, urun_kritik_miktar FLOAT)

Müşteriler Tablosu Kodu

CREATE TABLE musteriler (musteri_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY, musteri_adi VARCHAR(255), musteri_telefon VARCHAR(255), musteri_adres VARCHAR(255), musteri_vergi_dairesi VARCHAR(255), musteri_vergi_no VARCHAR(255))

Tedarikçiler Tablosu Kodu

CREATE TABLE tedarikciler (tedarikci_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY, tedarikci_adi VARCHAR(255), tedarikci_telefon VARCHAR(255), tedarikci_adres VARCHAR(255), tedarikci_vergi_dairesi VARCHAR(255), tedarikci_vergi_no VARCHAR(255))

Satış Faturası Tablosu Kodu

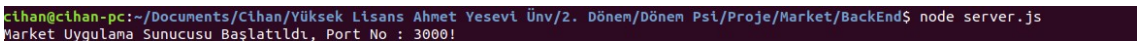
CREATE TABLE satis_faturasi (fatura_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY, fatura_tarihi DATE, musteri_id INT, musteri_adi VARCHAR(255), musteri_telefon VARCHAR(255), musteri_adres VARCHAR(255), musteri_vergi_dairesi VARCHAR(255), musteri_vergi_no VARCHAR(255))

Satış Faturası Kalem Tablosu Kodu

CREATE TABLE satis_faturasi_kalem (fatura_kalem_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY, fatura_id INT, urun_id INT, urun_adi VARCHAR(255), urun_miktar FLOAT, urun_fiyat FLOAT, urun_tutar FLOAT)

Market Arka Uç (BackEnd) Uygulaması

Market arka uç uygulaması NodeJS teknolojisi kullanılarak yazılmıştır. Market ERP uygulaması için arka plandaki kodları işletmek ve Blokzincir işlemlerini gerçekleştirecektir.

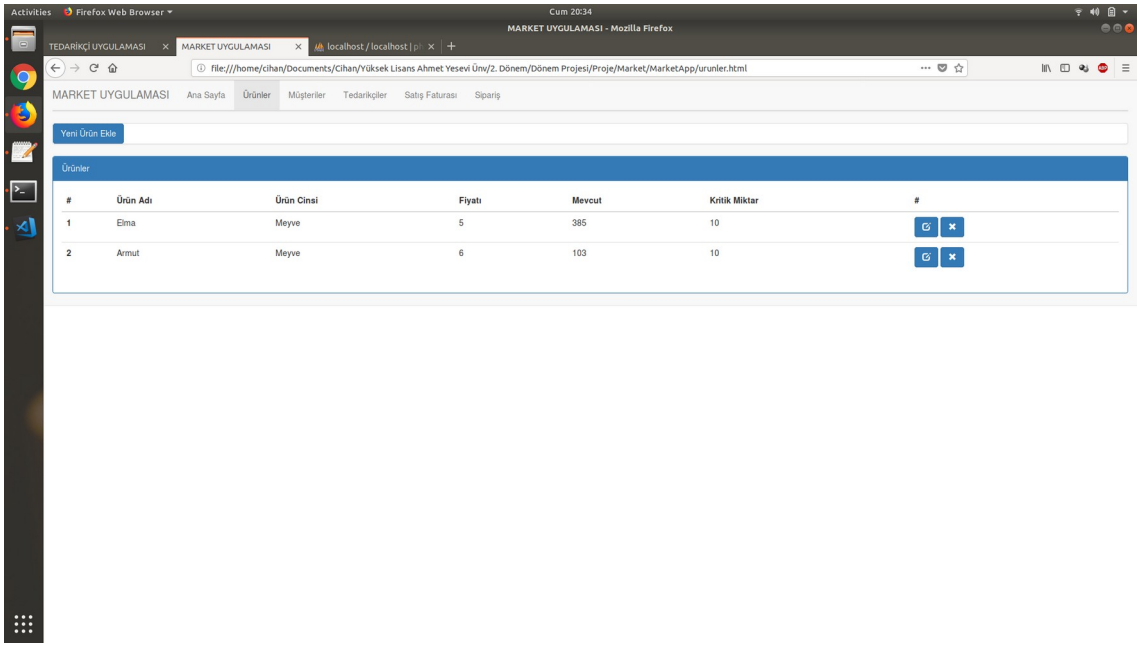


Şekil 9.4.2.10 : Market ERP Arka Uç

Ekranlar

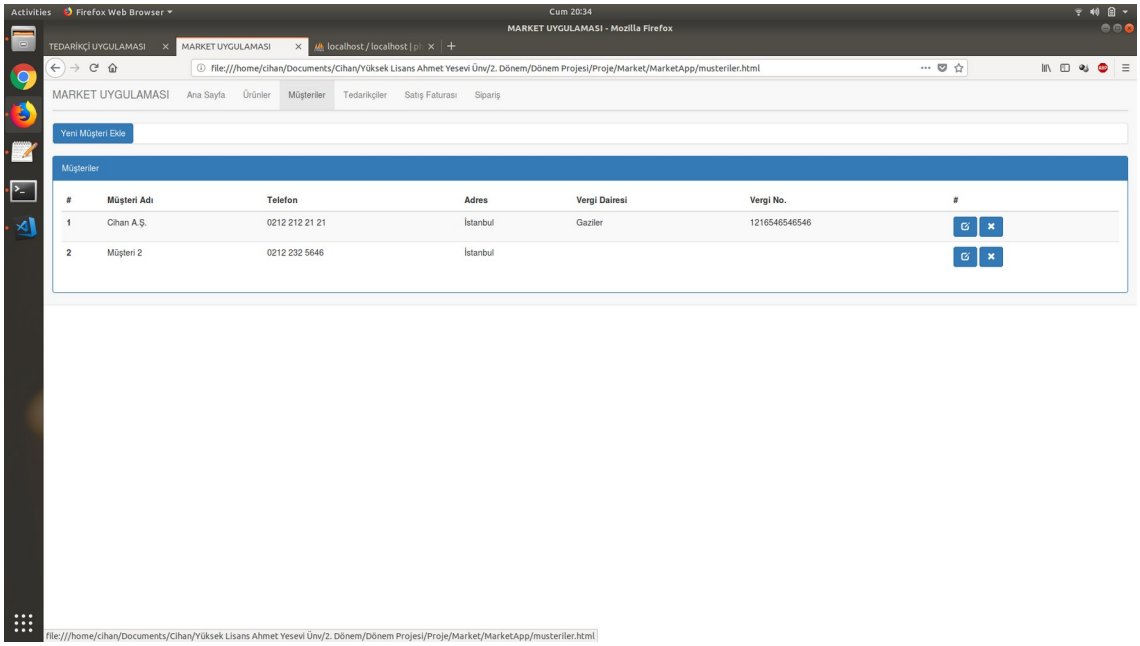
Market ERP uygulaması Bootstrap Framework, JQuery Framework, Angular JS Frameworkleri kullanılarak yazılmıştır.

Ürün Ekranı : Ürünlerin tanımlandığı ve kritik duruma düşen ürünlerin tespit edildiği ekrandır.



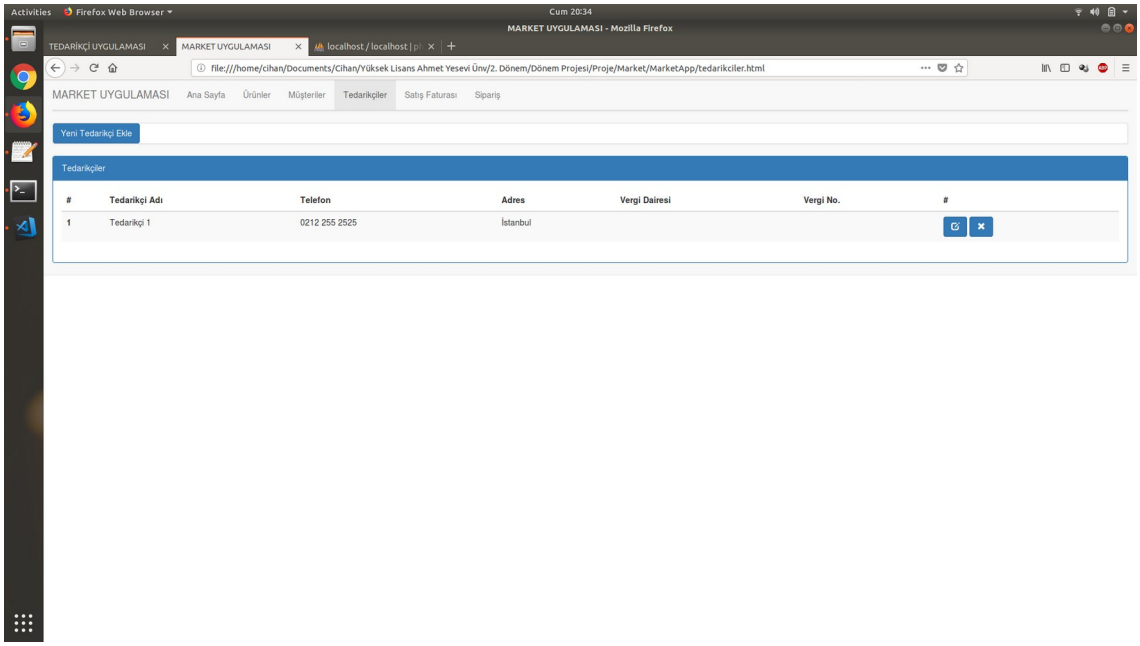
Şekil 9.4.3.11 : Market ERP Ürünler Ekranı

Müşteriler Ekranı: Müşterilerin tanımlandığı ve takip edildiği ekrandır.



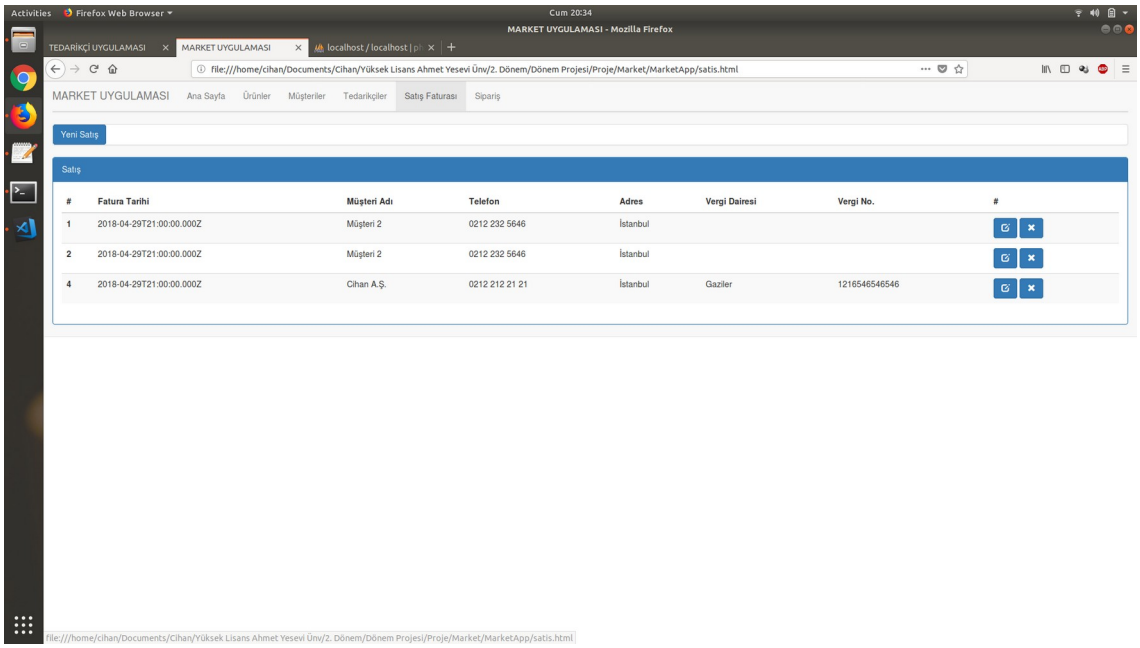
Şekil 9.4.3.12 : Market ERP Müşteriler Ekranı

Tedarikçiler Ekranı : Tedarikçilerin tanımlandığı ve takip edildiği ekrandır.

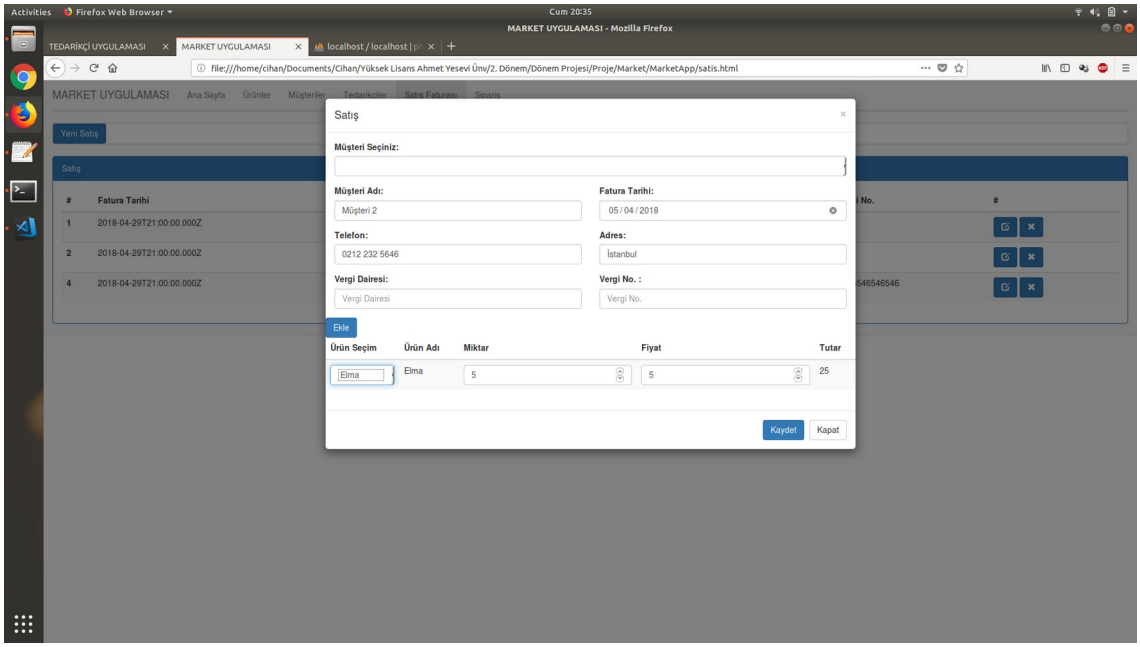


Şekil 9.4.3.13 : Market ERP Tedarikçiler Ekranı

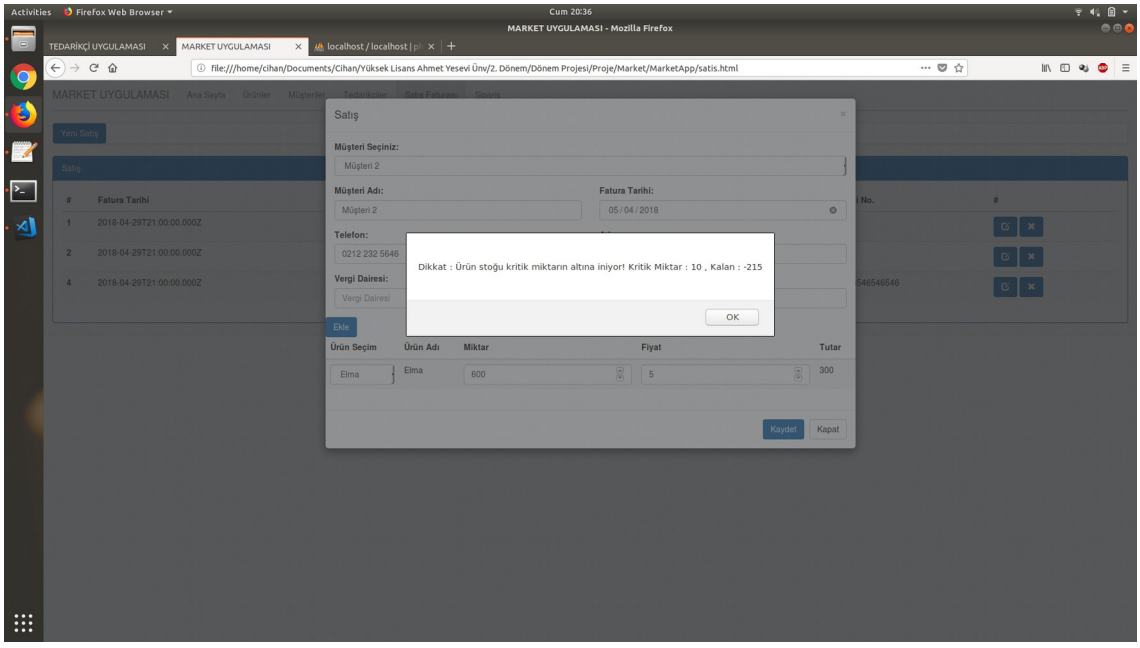
Satış Ekranı : Ürün satış işlemlerinin gerçekleştirildiği ekrandır. Satış esnasında kritik miktarın altına düşen ürünler için kullanıcıya uyarı gösterilir.



Şekil 9.4.3.14 : Market ERP Satış Ekranı

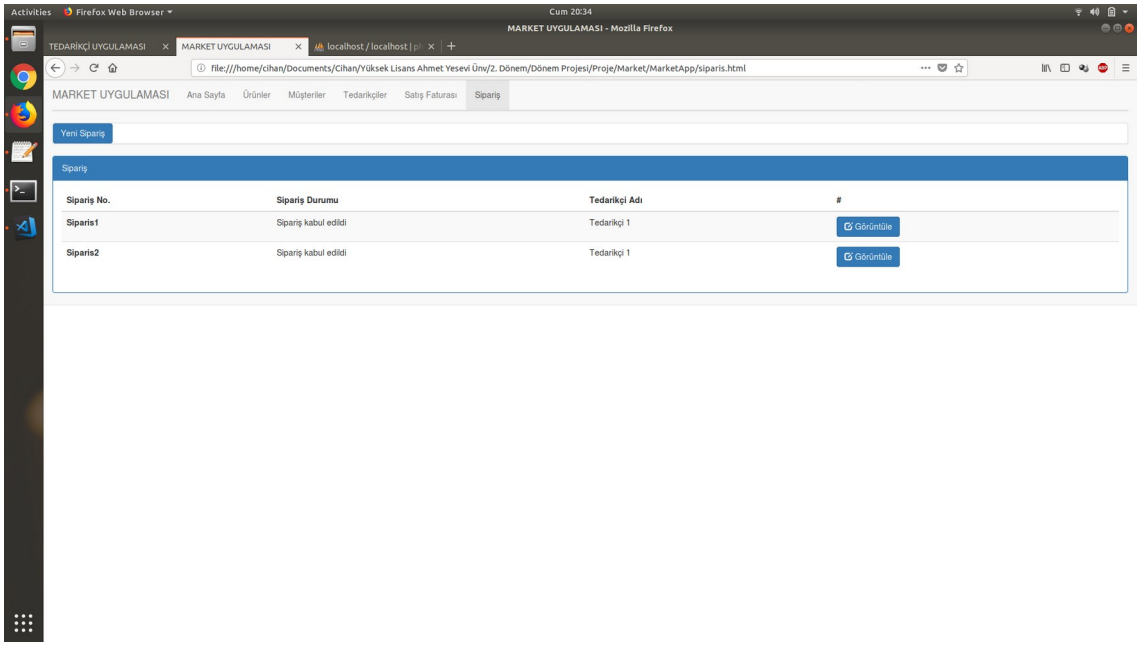


Şekil 9.4.3.15 : Market ERP Satış Ekranı 2

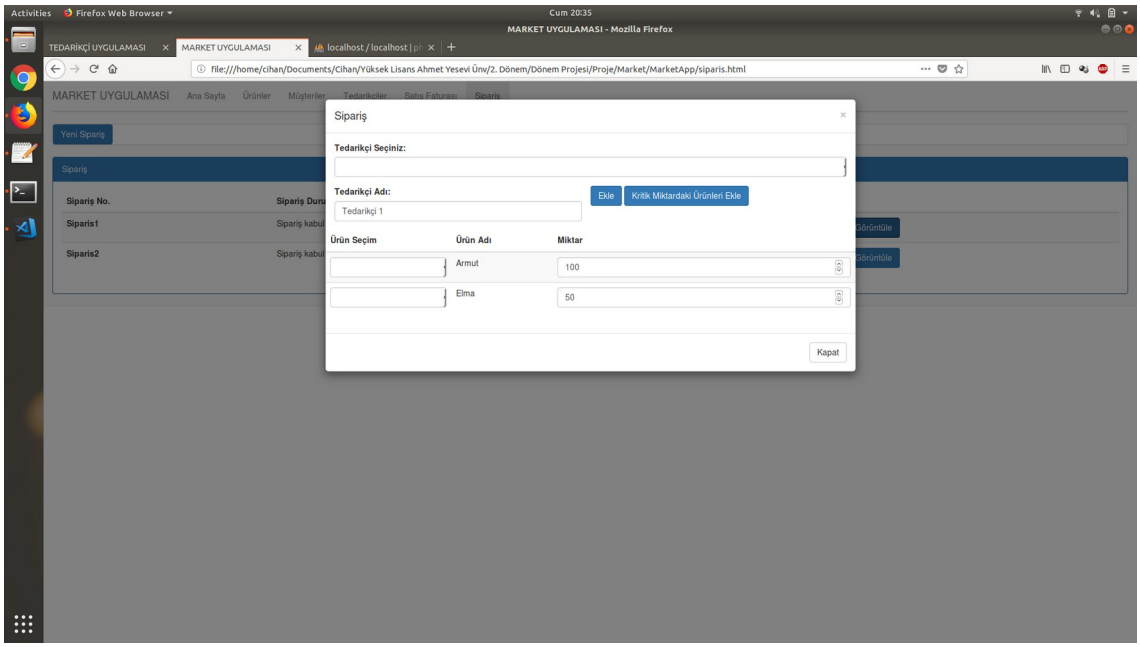


Şekil 9.4.3.16 : Market ERP Satış Ekranı Kritik Ürün Uyarısı

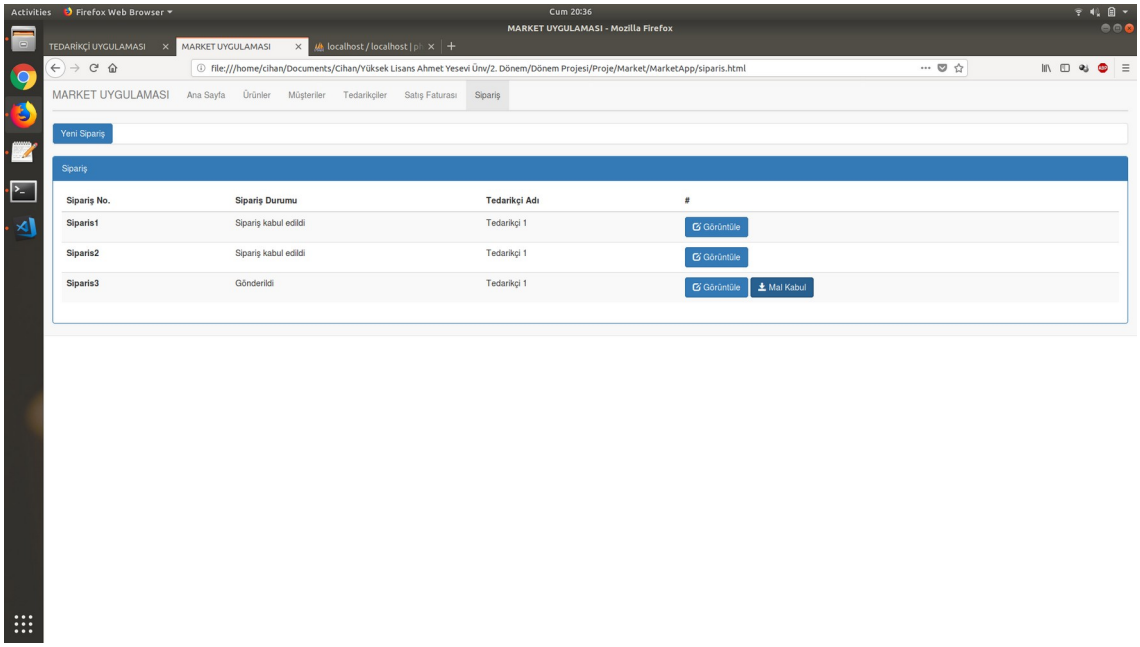
Sipariş Ekranı : Kritik stok miktarına ulaşan ürünlerin tedarikçiden sipariş edildiği ve takip edildiği ekrandır.



Şekil 9.4.3.17 : Market ERP Sipariş Ekranı



Şekil 9.4.3.18 : Market ERP Sipariş Ekranı 2



Şekil 9.4.3.19 : Market ERP Sipariş - Mal Kabul Ekranı

1.32. Tedarikçi Uygulaması

Tedarikçi uygulaması tedarikçi firmanın sipariş bilgilerini görmesi ve sipariş gönderim onayını verebilmesi için tasarlanmıştır.

Tedarikçi Arka Uç (BackEnd) Uygulaması

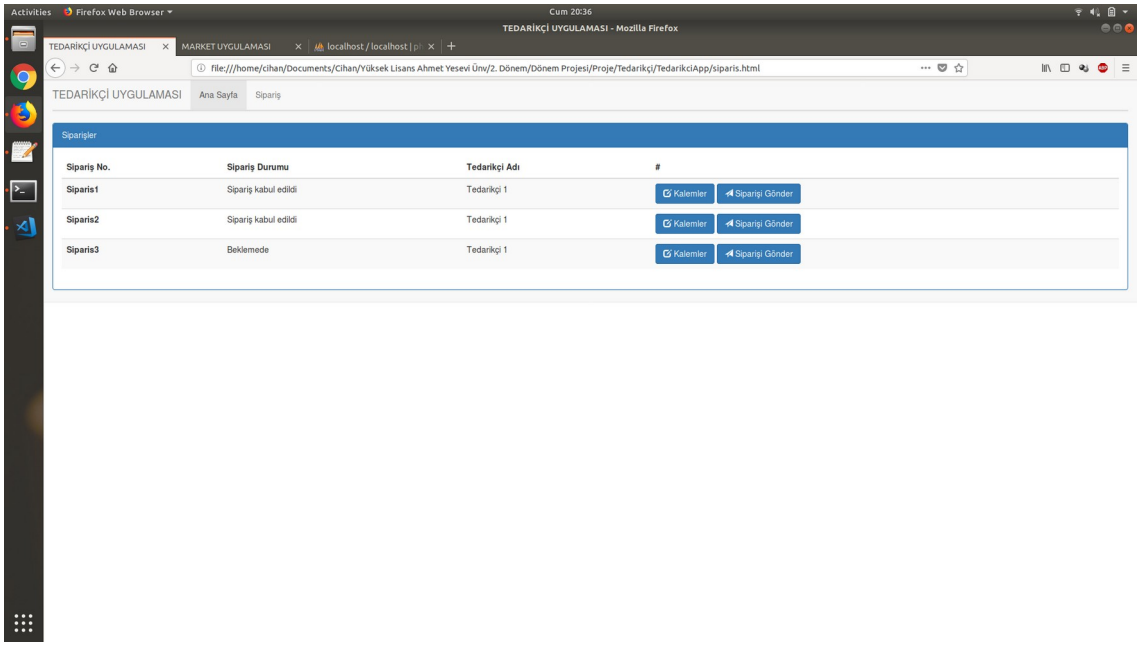
Tedarikçi uygulaması NodeJS platformu kullanılarak geliştirilmiştir. Blokzincir ile haberleşen ve gerekli arka plan ilerinin yapılmasını sağlayan uygulamadır.

```
cihan@cihan-pc:~/Documents/Cihan/Yüksek Lisans Ahmet Yesevi Ünv/2. Dönem/Dönem Projesi/Proje/Tedarikçi/BackEnd$ node server.js
Tedarikçi Uygulama Sunucusu Başlatıldı, Port No : 3001!
```

Şekil 9.5.1.20 : Tedarikçi Uygulaması Arka Uç

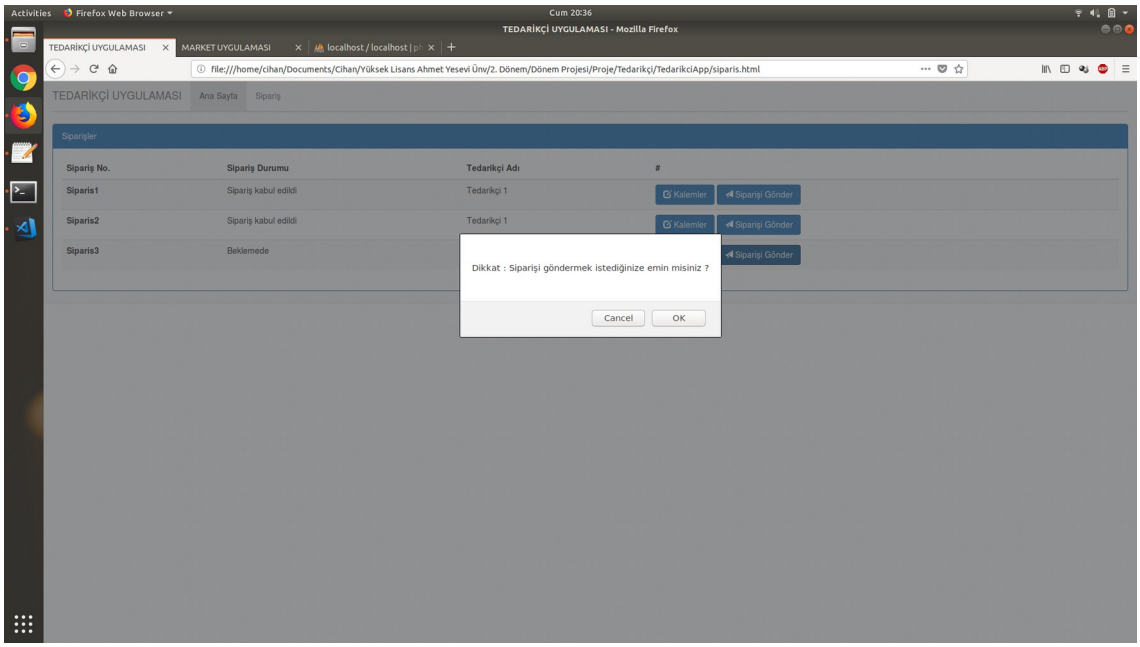
Tedarikçi Uygulaması Ön Yüz (FrontEnd)

Bootstrap, jquery, angularjs frameworkleri kullanılarak geliştirilmiştir.



Şekil 9.5.2.21 : Tedarikçi Uygulaması Sipariş Takip Ekranı

Siparişlerin takip edildiği ve gönderim durum bilgisinin Blokzincir'e iletildiği ekranlardır.



Şekil 9.5.2.22 : Tedarikçi Uygulaması Sipariş Gönderim Onayı

1.33. Market ERP ve Tedarikçi Uygulamaları İçin Gereksinimler

Uygulama Geliştirme Ortamı (IDE)

Visual Studio Code uygulama geliştirme ortamı kullanılmıştır.

Uygulama Geliştirme Platformu

NodeJS Uygulama geliştirme platformu kullanılmıştır.

Ön Yüz (FrontEnd) Framworkleri

Bootstrap, JQuery, AngularJS frameworkleri kullanılmıştır.

BÖLÜM X

1.34. Blokzincir (Blockchain) Uygulaması

Blokzincir uygulaması için Linux Foundation ve IBM tarafından geliştirilen Hyperledger Fabric alt yapısı kullanılmıştır.

1.35. Hyperledger Fabric Gereksinimleri

1. **Docker 1.14 :**
   Kurulum : sudo apt install docker
2. **Docker Compose 1.14 :**
   Kurulum : sudo apt install docker-compose
3. **GO 1.9.x :**
   Kurulum : sudo apt-get install golang-go
4. **Node JS 7.x :**
   Kurulum : sudo apt-get install nodejs
5. **NPM :**
   Kurulum : sudo apt-get install npm
6. **Python 3 :**
   Kurulum : sudo apt-get install python
7. **CURL :**
   Kurulum : sudo apt-get install curl

1.36. Fabric Binary Dosyaları

Kurulum : curl -sSL https://goo.gl/6wtTN5 | bash -s 1.1.0-preview

Yukarıdaki kod satırı çalıştırıldığında çalıştırıldığı klasöre proje için gerekli olan gerekli olan dosyalar indirilecektir.

1.37. Fabric Kurulum

```
cd /home/cihan/hyperledger
mkdir fabricca
cd fabricca
mkdir src/
mkdir src/github.com
mkdir src/github.com/hyperledger

cd ~/hyperledger/fabricca/src/github.com/hyperledger
git clone https://github.com/hyperledger/fabric-ca.git
cd ~/hyperledger/fabricca/src/github.com/hyperledger/fabric-ca

make fabric-ca-server
make fabric-ca-client

cd /home/cihan/hyperledger
mkdir server
mkdir client

nano ~/hyperledger/myexport
#bu satırları ekliyoruz;
export GOPATH=~/hyperledger/fabricca
export PATH=$PATH:~/hyperledger/fabricca/src/github.com/hyperledger/fabric-ca/bin
export FABRIC_CA_HOME=~/hyperledger/fabricca/src/github.com/hyperledger/fabric-ca
export FABRIC_CA_SERVER_HOME=~/hyperledger/server
export FABRIC_CA_CLIENT_HOME=~/hyperledger/client


source ~/hyperledger/myexport
```

cd /home/cihan/hyperledger/server

fabric-ca-server init -b "admin:adminpw"

1.38. Fabric Ayar Dosyaları

source ~/hyperledger/myexport

cd ~/hyperledger/server

#sunucu oluşturma

fabric-ca-server init -b "admin:adminpw"

#sunucu başlatma

fabric-ca-server start -b "admin:adminpw"

#market tarafı istemci yapılandırması

source ~/hyperledger/myexport

cd /home/cihan/hyperledger/client

#kullanıcı girişi ve sertifika dosyalarının çekilmesi ;

fabric-ca-client enroll -u "http://admin:adminpw@localhost:7054"

fabric-ca-client register -u "http://localhost:7054" --id.name "marketsatinalma" --id.secret "1234" --id.type "client" --id.affiliation "market.satinalma"

#id.name – Kullanıcının adı.

#id.secret – Kullanıcının kayıt parolası.

#id.affiliation – Kullanıcının bağlı olduğu kuruluş ve departman. Temel olarak kullanıcı izinlerini ayırmak için kullanılır.

#id.type – Kullanıcının türü. Normalde “client” kullanıyoruz, ancak bir kullanıcı peer, doğrulayıcı, denetçi, ca vb. olabilir.

fabric-ca-client enroll -u "http://marketsatinalma:1234@localhost:7054"

#tedarikçi tarafı istemci yapılandırması

cd /home/cihan/hyperledger/client

source ~/hyperledger/myexport

mkdir /home/cihan/hyperledger/tedarikciclient
cd /home/cihan/hyperledger/tedarikciclient
export FABRIC_CA_CLIENT_HOME=~/hyperledger/tedarikciclient
fabric-ca-client enroll -u "http://admin:adminpw@localhost:7054"
fabric-ca-client -u "http://localhost:7054" affiliation add tedarikci1.siparis
fabric-ca-client register -u "http://localhost:7054" --id.name "tedarikci1siparis" --id.secret "1234" --id.type "client" --id.affiliation "tedarikci1.siparis"

fabric-ca-client enroll -u "http://tedarikci1siparis:1234@localhost:7054"

#Böylece, kullanıcının başarıyla kaydettirilmesiyle, çıktıları belirtilen dizinde #public/private key'leri alabiliriz.

#Tebrikler! Hyperledger Fabric Client kurulumu tamamlanmış oldu! Kendi Hyperledger kurulumumuzu başarıyla tamamladık!

1.39. Kripto Oluşturucu (Crypto Generator)

Verilerimizin güvenliği için kriptografik materyali (x509 sertifikaları ve imzalama anahtarları) oluşturmak için cryptogen aracını kullanacağız. Bu sertifikalar kimliklerin temsilcisidir ve tarafların iletişim kurdukça ve imzaladıkça oturum açma / doğrulama kimlik doğrulamasına olanak tanır.

1.40. Kriptografi Nasıl Çalışıyor ?

Cryptogen, ağ topolojisini içeren ve hem Kuruluşlar hem de bu Organizasyonlara ait bileşenler için bir dizi sertifika ve anahtar üretmemizi sağlayan bir dosya - crypto-config.yaml - tüketir. Her Organizasyon, belirli bileşenleri (emsalleri ve sipariş verenleri) bu Org'a bağlayan benzersiz bir kök sertifikası (ca-cert) sağlar. Her bir Kuruluşa benzersiz bir CA sertifikası atayarak, katılımcı bir Üyenin kendi Sertifika Yetkilisini kullanacağı tipik bir ağı taklit ediyoruz. Hyperledger Fabric içindeki işlemler ve iletişim, bir işletmenin özel anahtarı (anahtar deposu) tarafından imzalanır ve daha sonra bir ortak anahtar (işaretçiler) aracılığıyla doğrulanır.

Bu dosyada bir sayı değişkeni göreceksiniz. Bunu, Kurum başına düşen akran sayısını belirtmek için kullanıyoruz; bizim durumumuzda Org başına iki akran var. Şu anda x.509

sertifikalarının ve ortak anahtar altyapısının minutialarını keşfetmeyeceğiz. İlgilendiyseniz, bu konuları kendi zamanınızda değerlendirebilirsiniz.

Aracı çalıştırmadan önce, crypto-config.yaml dosyasından snippet'e hızlıca göz atalım. OrdererOrgs başlığı altındaki “Name”, “Domain” ve “Specs” parametrelerine özellikle dikkat edin:

```
OrdererOrgs:
  # ---------------------------------------------------------------------------
  # Orderer
  # ---------------------------------------------------------------------------
  - Name: Orderer
    Domain: tedarikzinciri.com

    Specs:
      - Hostname: orderer

# ---------------------------------------------------------------------------
# ---------------------------------------------------------------------------
PeerOrgs:
  # ---------------------------------------------------------------------------
  # market
  # ---------------------------------------------------------------------------
  - Name: market
    Domain: market.tedarikzinciri.com

    # ---------------------------------------------------------------------------
    # "CA"
    CA:
       Hostname: ca #  ca.market.tedarikzinciri.com



    Template:
      Count: 2
      # Start: 5
```

```
    # Hostname: {{.Prefix}}{{.Index}} # default
    SANS:
      - "localhost"

  # ---------------------------------------------------------------------------
  # "Users"
  # ---------------------------------------------------------------------------
  # Count: The number of user accounts _in addition_ to Admin
  # ---------------------------------------------------------------------------
  Users:
    Count: 1

# ---------------------------------------------------------------------------
# tedarikci1: See "market" for full specification
# ---------------------------------------------------------------------------
- Name: tedarikci1
  Domain: tedarikci1.tedarikzinciri.com
  CA:
     Hostname: ca #  ca.market.tedarikzinciri.com

  Template:
    Count: 2
    SANS:
      - "localhost"
  Users:
    Count: 1
```

Bir ağ varlığı için adlandırma kuralı şöyledir: “{{.Hostname}}. {{. Domain}}”. Bu nedenle, sipariş düğümümüzü bir referans noktası olarak kullanarak, bir MSP ID'sinin Orderer'a bağlı olan orderer.tedarikzinciri.com adlı bir sipariş düğümüyle kalıyoruz. Bu dosya, tanımlar ve sözdizimi hakkında kapsamlı belgeler içerir. MSP hakkında daha derin bir dalış için Üyelik Servis Sağlayıcılarına (MSP) da başvurabilirsiniz.

Kriptojen aracını çalıştırdıktan sonra, oluşturulan sertifikalar ve anahtarlar kripto-config başlıklı bir klasöre kaydedilir.

Oluşturma :

export FABRIC_CFG_PATH=$PWD

cryptogen generate --config=cryptogen.yaml

1.41. Yapılandırma İşlem Üreticisi (Configuration Transaction Generator)

Configtxgen aracı dört yapılandırma dosyası oluşturmak için kullanılır:

- Düzenleyici genesis bloğu,
- Kanal yapılandırma işlemi,
- Her iki paydaş (peer) için çapa akran işlemi.

Sipariş bloğu, sipariş servisi için Genesis Bloğu'dur ve kanal yapılandırma işlem dosyası, kanal oluşturma zamanında siparişe yayınlanır. Anchor peer işlemleri, adından da anlaşılacağı gibi, bu Org'un her bir Org'daki Anchor Peer'ını belirler.

1.42. Nasıl Çalışır ?

Configtxgen, örnek ağ için tanımları içeren bir dosya - configtx.yaml - tüketir. İki üye vardır - her biri iki eş düğümünü yöneten ve koruyan bir adet Orderer Org (OrdererOrg) ve iki adet Peer Orgs (Market ve Tedarikci1). Bu dosya aynı zamanda iki Peer Orgs'dan oluşan bir konsorsiyum TedarikZinciriConsortium'u da belirtir. Bu dosyanın en üstündeki "Profiller" bölümüne özellikle dikkat edin. İki benzersiz başlığa sahip olduğumuzu fark edeceksiniz. Bir sipariş yaratma bloğu - TwoOrgsOrdererGenesis - ve kanalımız için bir tane - TwoOrgsChannel.

Bu başlıklar önemlidir, çünkü eserlerimizi oluşturduğumuzda bunları argüman olarak göreceğiz.

Not :

TedarikZinciriConsortium'umuzun sistem düzeyinde profilde tanımlandığına ve daha sonra kanal düzeyindeki profilimize referans verildiğine dikkat edin. Kanallar bir konsorsiyumun kapsamı içerisinde mevcut olup, tüm konsorsiyumlar ağın kapsamı içerisinde tanımlanmalıdır.

Bu dosya ayrıca dikkat çeken iki ek özellik içerir. İlk olarak, her bir Peer Org (peer0.market.tedarikzinciri.com & peer0.tedarikci1.tedarikzinciri.com) için çapa akranlarını belirtiyoruz. İkincisi, her üye için MSP dizininin yerini işaret ederek, sırayla

her bir Org için kök sertifikalarını düzenleyici kod satırında saklamamıza izin veriyoruz. Bu kritik bir kavramdır. Artık, sipariş servisiyle iletişim kuran herhangi bir ağ varlığının dijital imzasının doğrulanmış olabilir.

Configx dosyamız :

Profiles:

```
    TwoOrgsOrdererGenesis:
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
        Consortiums:
            TedarikZinciriConsortium:
                Organizations:
                    - *market
                    - *tedarikci1
    TwoOrgsChannel:
        Consortium: TedarikZinciriConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *market
                - *tedarikci1


Organizations:

    - &OrdererOrg
        Name: OrdererMSP

        ID: OrdererMSP

        MSPDir: crypto-config/ordererOrganizations/tedarikzinciri.com/msp
```

```
    - &market
        Name: marketMSP

        ID: marketMSP

        MSPDir: crypto-config/peerOrganizations/market.tedarikzinciri.com/msp

        AnchorPeers:
            - Host: peer0.market.tedarikzinciri.com
              Port: 7051

    - &tedarikci1
        Name: tedarikci1MSP

        ID: tedarikci1MSP

        MSPDir: crypto-config/peerOrganizations/tedarikci1.tedarikzinciri.com/msp

        AnchorPeers:
            - Host: peer0.tedarikci1.tedarikzinciri.com
              Port: 7051

Orderer: &OrdererDefaults

    OrdererType: solo

    Addresses:
        - orderer.tedarikzinciri.com:7050

    BatchTimeout: 2s

    BatchSize:

        MaxMessageCount: 10
```

```
        AbsoluteMaxBytes: 98 MB

        PreferredMaxBytes: 512 KB

    Kafka:
        Brokers:
            - 127.0.0.1:9092

    Organizations:

Application: &ApplicationDefaults

    Organizations:
```

####################

Blok Oluşturma kodları :

configtxgen -profile TwoOrgsOrdererGenesis -outputBlock genesis.block

Kanal Oluşturma :

configtxgen -profile TwoOrgsChannel -outputCreateChannelTx channel.tx -channelID mychannel

1.43. Docker Ayar Dosyası

```
version: '2'

services:

  ca.market.tedarikzinciri.com:
    image: hyperledger/fabric-ca
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca-market
```

```
      - FABRIC_CA_SERVER_CA_CERTFILE=/etc/hyperledger/fabric-ca-server-config/
ca.market.tedarikzinciri.com-cert.pem
       - FABRIC_CA_SERVER_CA_KEYFILE=/etc/hyperledger/fabric-ca-server-config/
3856267cd6b1190087c6cf5bd8191d940ce906e1cdf3ac55214dfdcd339f20ce_sk
     - FABRIC_CA_SERVER_TLS_ENABLED=true
      - FABRIC_CA_SERVER_TLS_CERTFILE=/etc/hyperledger/fabric-ca-server-config/
ca.market.tedarikzinciri.com-cert.pem
       - FABRIC_CA_SERVER_TLS_KEYFILE=/etc/hyperledger/fabric-ca-server-config/
3856267cd6b1190087c6cf5bd8191d940ce906e1cdf3ac55214dfdcd339f20ce_sk
   ports:
    - "7054:7054"
   command: sh -c 'fabric-ca-server start -b admin:adminpw -d'
   volumes:
            -  ./channel/crypto-config/peerOrganizations/market.tedarikzinciri.com/ca/:/etc/
hyperledger/fabric-ca-server-config
   container_name: ca_peermarket

 ca.tedarikci1.tedarikzinciri.com:
   image: hyperledger/fabric-ca
   environment:
    - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
    - FABRIC_CA_SERVER_CA_NAME=ca-tedarikci1
      - FABRIC_CA_SERVER_CA_CERTFILE=/etc/hyperledger/fabric-ca-server-config/
ca.tedarikci1.tedarikzinciri.com-cert.pem
       - FABRIC_CA_SERVER_CA_KEYFILE=/etc/hyperledger/fabric-ca-server-config/
e1b7faaca158fec0bfa59036d45a17d391593b1a597d69050c1615f4297c3dc7_sk
     - FABRIC_CA_SERVER_TLS_ENABLED=true
      - FABRIC_CA_SERVER_TLS_CERTFILE=/etc/hyperledger/fabric-ca-server-config/
ca.tedarikci1.tedarikzinciri.com-cert.pem
       - FABRIC_CA_SERVER_TLS_KEYFILE=/etc/hyperledger/fabric-ca-server-config/
e1b7faaca158fec0bfa59036d45a17d391593b1a597d69050c1615f4297c3dc7_sk
   ports:
    - "8054:7054"
   command: sh -c 'fabric-ca-server start -b admin:adminpw -d'
```

```
  volumes:
        - ./channel/crypto-config/peerOrganizations/tedarikci1.tedarikzinciri.com/ca/:/etc/
hyperledger/fabric-ca-server-config
  container_name: ca_peertedarikci1

 orderer.tedarikzinciri.com:
  container_name: orderer.tedarikzinciri.com
  image: hyperledger/fabric-orderer
  environment:
    - ORDERER_GENERAL_LOGLEVEL=debug
    - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
    - ORDERER_GENERAL_GENESISMETHOD=file
    - ORDERER_GENERAL_GENESISFILE=/etc/hyperledger/configtx/genesis.block
    - ORDERER_GENERAL_LOCALMSPID=OrdererMSP
    - ORDERER_GENERAL_LOCALMSPDIR=/etc/hyperledger/crypto/orderer/msp
    - ORDERER_GENERAL_TLS_ENABLED=true
      - ORDERER_GENERAL_TLS_PRIVATEKEY=/etc/hyperledger/crypto/orderer/tls/
server.key
      - ORDERER_GENERAL_TLS_CERTIFICATE=/etc/hyperledger/crypto/orderer/tls/
server.crt
                                                                                -
ORDERER_GENERAL_TLS_ROOTCAS=[/etc/hyperledger/crypto/orderer/tls/ca.crt, /etc/
hyperledger/crypto/peermarket/tls/ca.crt, /etc/hyperledger/crypto/peertedarikci1/tls/ca.crt]
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric/orderers
  command: orderer
  ports:
    - 7050:7050
  volumes:
      - ./channel:/etc/hyperledger/configtx
            - ./channel/crypto-config/ordererOrganizations/tedarikzinciri.com/orderers/
orderer.tedarikzinciri.com/:/etc/hyperledger/crypto/orderer
            - ./channel/crypto-config/peerOrganizations/market.tedarikzinciri.com/peers/
peer0.market.tedarikzinciri.com/:/etc/hyperledger/crypto/peermarket
```

- ./channel/crypto-config/peerOrganizations/tedarikci1.tedarikzinciri.com/peers/peer0.tedarikci1.tedarikzinciri.com/:/etc/hyperledger/crypto/peertedarikci1

```
  peer0.market.tedarikzinciri.com:
    container_name: peer0.market.tedarikzinciri.com
    extends:
      file:   base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer0.market.tedarikzinciri.com
      - CORE_PEER_LOCALMSPID=marketMSP
      - CORE_PEER_ADDRESS=peer0.market.tedarikzinciri.com:7051
    ports:
      - 7051:7051
      - 7053:7053
    volumes:
```

- ./channel/crypto-config/peerOrganizations/market.tedarikzinciri.com/peers/peer0.market.tedarikzinciri.com/:/etc/hyperledger/crypto/peer

```
    depends_on:
      - orderer.tedarikzinciri.com

  peer1.market.tedarikzinciri.com:
    container_name: peer1.market.tedarikzinciri.com
    extends:
      file:   base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer1.market.tedarikzinciri.com
      - CORE_PEER_LOCALMSPID=marketMSP
      - CORE_PEER_ADDRESS=peer1.market.tedarikzinciri.com:7051
    ports:
      - 7056:7051
      - 7058:7053
    volumes:
```

```
            - ./channel/crypto-config/peerOrganizations/market.tedarikzinciri.com/peers/
peer1.market.tedarikzinciri.com/:/etc/hyperledger/crypto/peer
    depends_on:
      - orderer.tedarikzinciri.com

  peer0.tedarikci1.tedarikzinciri.com:
    container_name: peer0.tedarikci1.tedarikzinciri.com
    extends:
      file:   base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer0.tedarikci1.tedarikzinciri.com
      - CORE_PEER_LOCALMSPID=tedarikci1MSP
      - CORE_PEER_ADDRESS=peer0.tedarikci1.tedarikzinciri.com:7051
    ports:
      - 8051:7051
      - 8053:7053
    volumes:
            - ./channel/crypto-config/peerOrganizations/tedarikci1.tedarikzinciri.com/peers/
peer0.tedarikci1.tedarikzinciri.com/:/etc/hyperledger/crypto/peer
    depends_on:
      - orderer.tedarikzinciri.com

  peer1.tedarikci1.tedarikzinciri.com:
    container_name: peer1.tedarikci1.tedarikzinciri.com
    extends:
      file:   base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer1.tedarikci1.tedarikzinciri.com
      - CORE_PEER_LOCALMSPID=tedarikci1MSP
      - CORE_PEER_ADDRESS=peer1.tedarikci1.tedarikzinciri.com:7051
    ports:
      - 8056:7051
```

```
    - 8058:7053
  volumes:
        - ./channel/crypto-config/peerOrganizations/tedarikci1.tedarikzinciri.com/peers/
peer1.tedarikci1.tedarikzinciri.com/:/etc/hyperledger/crypto/peer
  depends_on:
    - orderer.tedarikzinciri.com
```

Fabric Docker Compose aracı kullanarak yukarıdaki dosyada bulunan parametrelere göre fabric sunucusunu başlatacağız :

docker-compose -f artifacts/docker-compose.yaml up

1.44. Blokcincir Zincirkodu (ChainCode - SmartContract)

Mycc.js Dosyası içeriği :

```
'use strict';
const shim = require('fabric-shim');
const util = require('util');
let Chaincode = class {
 async Init(stub) {
  console.info('========== Instantiated tedarikzinciri chaincode ==========');
  return shim.success();
 }
 async Invoke(stub) {
  let ret = stub.getFunctionAndParameters();
  console.info(ret);
  let method = this[ret.fcn];
  if (!method) {
   console.error('no function of name:' + ret.fcn + ' found');
   throw new Error('Received unknown function ' + ret.fcn + ' invocation');
  }
  try {
   let payload = await method(stub, ret.params);
   return shim.success(payload);
```

```
    } catch (err) {
      console.log(err);
      return shim.error(err);
    }
  }

  async querySiparis(stub, args) {
    if (args.length != 1) {
          throw new Error('Incorrect number of arguments. Expecting SiparisNumber ex: Siparis01');
    }
    let SiparisNumber = args[0];

        let SiparisAsBytes = await stub.getState(SiparisNumber); //get the Siparis from chaincode state
    if (!SiparisAsBytes || SiparisAsBytes.toString().length <= 0) {
      throw new Error(SiparisNumber + ' does not exist: ');
    }
    console.log(SiparisAsBytes.toString());
    return SiparisAsBytes;
  }

  async initLedger(stub, args) {
    console.info('============= START : Initialize Ledger ===========');
    let Siparisler = [];
    Siparisler.push({
      tedarikci_adi: 'Tedarikçi 1',
      durum: 'Beklemede',
      kalemler: [{ urun_id: 1, urun_adi: 'Elma', urun_miktar: 1 }]
    });

    Siparisler.push({
      tedarikci_adi: 'Tedarikçi 1',
      durum: 'Beklemede',
```

```
    kalemler: [{ urun_id: 2, urun_adi: 'Armut', urun_miktar: 1 }]
  });

  for (let i = 0; i < Siparisler.length; i++) {
    Siparisler[i].docType = 'Siparis';
    await stub.putState('Siparis' + i, Buffer.from(JSON.stringify(Siparisler[i])));
    console.info('Added <--> ', Siparisler[i]);
  }
  console.info('============= END : Initialize Ledger ===========');
}

async createSiparis(stub, args) {
  //args : Beklenen : ["Siparis1","Tedarikçi1","Elma:5,Armut:2"]
  console.info('============= START : Create Siparis ===========');
  if (args.length != 3) {
    throw new Error('Incorrect number of arguments. Expecting 3');
  }
  var siparis_kalemler_str = args[2].split(',');//Elma:5,Armut:2 gibi bir veri alacak
  var siparis_kalemler = [];
  for (let index = 0; index < siparis_kalemler_str.length; index++) {
    const element = siparis_kalemler_str[index];
    const element_split = element.split(':');
    siparis_kalemler.push({ urun_adi: element_split[0], urun_miktar: element_split[1] });
  }

  var Siparis = {
    docType: 'Siparis',
    tedarikci_adi: args[1],
    durum: "Beklemede",
    kalemler: siparis_kalemler
  };

  await stub.putState(args[0], Buffer.from(JSON.stringify(Siparis)));
  console.info('============= END : Create Siparis ===========');
```

```
  }

  async queryAllSiparisler(stub, args) {

    let startKey = 'Siparis0';
    let endKey = 'Siparis999';

    let iterator = await stub.getStateByRange(startKey, endKey);

    let allResults = [];
    while (true) {
      let res = await iterator.next();

      if (res.value && res.value.value.toString()) {
        let jsonRes = {};
        console.log(res.value.value.toString('utf8'));

        jsonRes.Key = res.value.key;
        try {
          jsonRes.Record = JSON.parse(res.value.value.toString('utf8'));
        } catch (err) {
          console.log(err);
          jsonRes.Record = res.value.value.toString('utf8');
        }
        allResults.push(jsonRes);
      }
      if (res.done) {
        console.log('end of data');
        await iterator.close();
        console.info(allResults);
        return Buffer.from(JSON.stringify(allResults));
      }
    }
  }
```

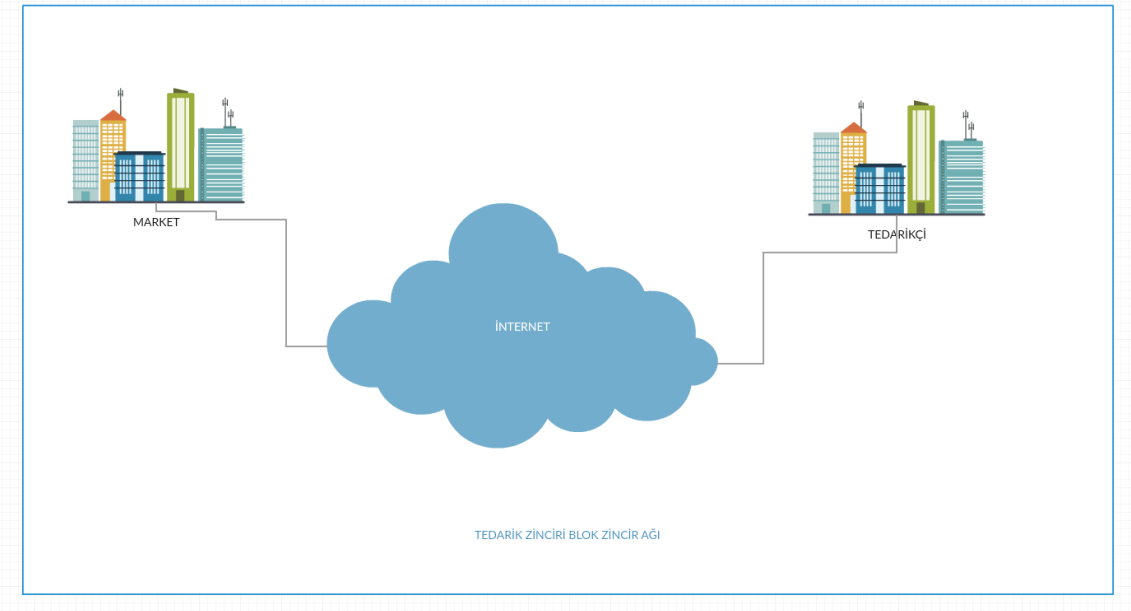
```
  async changeSiparisStatus(stub, args) {
    console.info('============= START : changeSiparisStatus ===========');
    if (args.length != 2) {
      throw new Error('Incorrect number of arguments. Expecting 2');
    }

    let SiparisAsBytes = await stub.getState(args[0]);
    let Siparis = JSON.parse(SiparisAsBytes);
    Siparis.durum = args[1];

    await stub.putState(args[0], Buffer.from(JSON.stringify(Siparis)));
    console.info('============= END : changeSiparisStatus ===========');
  }
};

shim.start(new Chaincode());
```

1.45. Blokzincir arka uç uygulaması

NodeJS alt yapısı kullanarak blokzincirimiz için bir RESTFULL servis uygulaması geliştirdik.

Servis başlatma : node app

1.46. Blokzincir Ağ Şeması



Şekil 10.13.23 : Blokzincir Ağ Şeması

BÖLÜM XI

SONUÇ

Tedarik zinciri ile market arasında bir blok zincir alt yapısı kuruldu ve verilerin güvenli ve değiştirilemeyeceği kanıtlanmış olarak iki kurum arasında dağıtıldığı tespit edildi. Böylelikle marketin tedarikçiye gönderdiği sipariş bilgileri ve kalemleri doğru şekilde tedarikçiye iletildi ve aynı zamanda tedarikçi paydaş tarafındaki veritabanına senkron ve kriptolu bir şekilde sipariş verileri yazıldı. Tedarikçinin markete göndermiş olduğu siparişin durum bilgisi de yine aynı şekilde markete iletildi ve market paydaş veritabanına senkron ve kriptolu şekilde sipariş durum verileri yazıldı. Bu sayede güvenli ve doğru bir iletişim alt yapısının kurulduğu görüldü ve tedarik zincirinde hiçbir aksaklık olmadan güvenli iletişim sağlanmış oldu.

Kaynakça

Abeyratne, S. A. (2016). Blockchain ready manufacturing supply chain using distributed ledger. S. A. Abeyratne içinde, *Blockchain ready manufacturing supply chain using distributed ledger* (s. 1). İngiltere: DSpace.

Aktaş, R. (2010). Kurumsal Kaynak Planlaması. Y. A. Doç.Dr. Rafet Aktaş içinde, *Kurumsal Kaynak Planlaması* (s. 5). Gazi Kitabevi.

Arjan J.van Weele, Ç. :. (2014). Satın Alma ve Tedarik Zinciri Yönetimi. A. J. Weele içinde, *Satın Alma ve Tedarik Zinciri Yönetimi* (s. 5). Literatür Yayıncılık.

Demirel, F. (2017, 08 07). *Türkiye'de blockchain teknolojisi üzerinde çalışan şirketler*. 05 10, 2018 tarihinde Webrazzi: https://webrazzi.com/2017/08/07/turkiyede-blockchain-kullanan-sirketler/ adresinden alındı

HyperLedger. (2015, 12 04). *HyperLedger*. 04 04, 2018 tarihinde HyperLedger: https://www.hyperledger.org/ adresinden alındı

Keinzler, R. (2015, 12 12). *Architecture of the Hyperledger Blockchain Fabric - Christian Cachin - IBM Research Zurich*. 04 04, 2018 tarihinde Architecture of the Hyperledger Blockchain Fabric - Christian Cachin - IBM Research Zurich: https://www.slideshare.net/ormium/architecture-of-the-hyperledger-blockchain-fabric-christian-cachin-ibm-research-zurich adresinden alındı

Lojiport. (2017, 11 15). *Tedarik zincirinde dev birleşme*. 05 09, 2018 tarihinde Lojiport: https://www.lojiport.com/tedarik-zincirinde-dev-birlesme-99959h.htm adresinden alındı

Maines, J. (2017, 12 01). *Inventory Management And Mrp Erp*. 04 04, 2018 tarihinde https://www.slideshare.net/Joanmaines/inventory-management-and-mrp-erp adresinden alındı

Nakamoto, S. (2011, 12 05). *Bitcoin*. 04 04, 2018 tarihinde Bitcoin: https://bitcoin.org/tr/ adresinden alındı

Yardımcı, K. (2017, 12 04). *Günümüz Ekonomisinde Tedarik Zinciri ve Önemi*. 04 04, 2018 tarihinde lean.org.tr: https://lean.org.tr/gunumuz-ekonomisinde-tedarik-zinciri-ve-onemi/ adresinden alındı