T.C.
SELÇUK ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Blok Zinciri Algoritmaları Kullanarak Tedarik Zinciri Otomasyonu Geliştirme (BATZO)

Kemal KOLCUOĞLU
173311082

ARAŞTIRMA PROJESİ

Haziran - 2018
KONYA

PROJE BİLDİRİMİ

Bu projedeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve proje yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

İmza
Kemal KOLCUOĞLU
Tarih: 19/06/2018

İçindekiler

1. Giriş

Bu rapor, Selçuk Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümü Donanım Projesi dersi kapsamında geliştirilen “Blok Zinciri Algoritmaları Kullanarak Tedarik Zinciri Otomasyonu Geliştirme” adlı yazılım hakkında bilgi sunmak için hazırlanmıştır.

2. Tedarik Zinciri

2.1. Tedarik Zinciri Nedir?

Tedarik zinciri, ürün veya hizmetlerin ürün yaşam döngü süreçlerini kapsayan ve hammaddeden yola çıkıp son müşterinin eline ulaşana kadar geçen operasyonların, bilgi akışının, fiziksel dağıtımının ve alışverişin bütününü içeren bir sistemdir.

Mal ve hizmetlerin tedarik aşamasından, üretimine ve nihai tüketiciye ulaşmasına kadar birbirini izleyen tüm halkaları kapsar. İş süreçleri açısından bakıldığında, tedarik zinciri; satış süreci, üretim, döküm yönetimi, malzeme temini, dağıtım, tedarik, satış tahmini ve müşteri hizmetleri gibi pek çok alanı içine almaktadır.

Bir ürünün ilk maddesinden başlayarak, tüketiciye ulaşması ve geri dönüşümünü de içeren tüm süreçlerde yer alan tedarikçi, üretici, dağıtıcı, perakendeci ve lojistikçilerden oluşan bir bütündür.

2.2. Tedarik Zinciri Yönetimi

Tedarik Zinciri en basit haliyle arz-talep ilişkisi yönetimidir. Bir ürün talep edilir ve talep doğrultusunda ürün üretilerek tedarik edilir. Böylece bir döngü oluşur. Ürünün üretilmesi için başka ürünlerin üretilmesi gerektiğinde de zincir oluşmaya başlar. Aynı

şekilde ürünü bir kanaldan birden çok kanala tedarik ederken oluşan durumda bir zincirdir. Bu örnekleri sayısız bir şekilde arttırabiliriz.

Arz-Talep içerisinde oluşan bu durumun yönetimi öncelikli olarak tüketicilerden alınana talep doğrultusunda gerçekleşir. Her şeyden önce bir kullanıcı vardır. Bu kullanıcı bir kişi ya da kurum olabilir. Bu kullanıcı ihtiyaçları doğrultusunda bir ürünü talep eder. Talep edilen ürünü elde etmek için bir tesis var ise işlemler o tesiste devam eder. Tesis o ürünü üreterek bir kişi ya da kuruma teslim eder. Bu ürünün talep eden tüketicilere teslim edilmesine aracı olan kişi ya da kuruma Tedarikçi denir. Tedarikçi başka bir tedarikçiye de ürünü teslim ederek zincire başka bir kişi daha eklenebilir ve bu zincire sayısız kişi eklenebilir.

Aşamalı olarak Tedarik Zinciri Yönetimi en basit haliyle şu şekilde sağlanır:

1. Tüketici ürün talep eder.
2. Talep doğrudan ya da başka kişiler/kurumlar aracılığıyla üreticiye bildirilir.
3. Üretici ürün için gerekli kaynaklara sahipse ürünü üretir ve 4. Adıma geçer, üretemez ise 1. Adımdaki tüketici yerine geçer ve 1. Adıma döner.
4. Ürünü temin etmek için tedarikçi kişiler/kurumlar ile anlaşılır.
5. Ürün teslim edilir.

3. Blok Zinciri Algoritması

3.1. Blok Zinciri Nedir?

Blok Zinciri; şifrelenmiş bir takım işlemlerin takibini sağlayan dağıtık bir veri tabanıdır. Sistemin en önemli özelliği veriler tek bir merkezde değil sistemin tüm bileşenlerinde yani tüm ağda bulunan bir defterlerde tutulmasıdır. Bu durum da bir merkeze bağlı olmaksızın işlem yapma olanağı sağlar. İşleme tabi tutulan hiçbir bilgi kaybolmaz, değiştirilemez, birden fazla yerde tutulduğu için de silinip yok edilme, arşivlerden kaybolma riski yoktur. Kayıtların birden fazla yerde tutulması güveni ve güvenliği arttıran bir unsurdur. Yani kayıt defterlerinden birisi yok olsa bile, bilgiler, ağda bulunan diğer defterlerde saklanmaya devam ettiğinden yok olmaz. Birçok işlemin bir

araya gelmesinden oluşan bloklar, çözülmesi aşırı derece zor şifreleme algoritmalarıyla, birbirine bağlanarak kaydedilirler. Bloklar kendinden önce gelen bloklara bağlı olarak bulunurlar. Yani yeni bir defter yaprağı gibi ardı sıra eklenirler. Bu durum yapılan işlemler üzerindeki güven meselesiyle bağlantılıdır. Blok zinciri oluşturan halkalardan birisinde bire değiştirildiğinde bu bilgi kendinden önceki kayıtlarla uyumsuz hale geleceği için hata verecektir, bu yüzden de değiştirmek mümkün değildir. Sistemdeki herhangi bir bilginin değiştirilebilmesi için zinciri oluşturan halkalardan ezici bir çoğunluğunun değişikliğin onaylanması gerekmektedir. Ve bu onay işlemi gerçekleşmeden hiçbir bilgi ya da varlık transferi gerçekleşmez.

3.2. Blok Zinciri Algoritmasının Kullanım Alanları

Kuşkusuz Kriptoparalar ile birlikte bankacılık sektörü ilk akla gelen alanlardan bir tanesidir. Ödeme ve para transferinde sağladığı avantajlarla en büyük devrimi bankacılık sektöründe yapacak gibi görünüyor.

Sonuçları şeffaf ve herkesçe ulaşılabilir, anket, seçim ya da her türlü kamuoyu yoklaması alanlarında kullanılabilir. Nüfus ve tapu kayıtları gibi birçok devlet kanalında kullanılabilir. Patent ve markanın yanı sıra fikrî mülkiyetin korunmasında uygulanabilir. Telif haklarının korunması ve eserlerin satışı konusunda büyük yol kat etmemize yardımcı olacaktır.

Günümüzün yine popüler kavramlarından IoT (nesnelerin interneti) ile birlikte kullanılarak hayatımızda çığır açabilir. Belki bir gün, ev kiralarken emlakçıyı bulup, kira kontratı imzalamak yerine kapının üzerindeki kare kodu telefonumuzla okutup, binlerce kişinin şahitliğinde evi kiralayabilir.

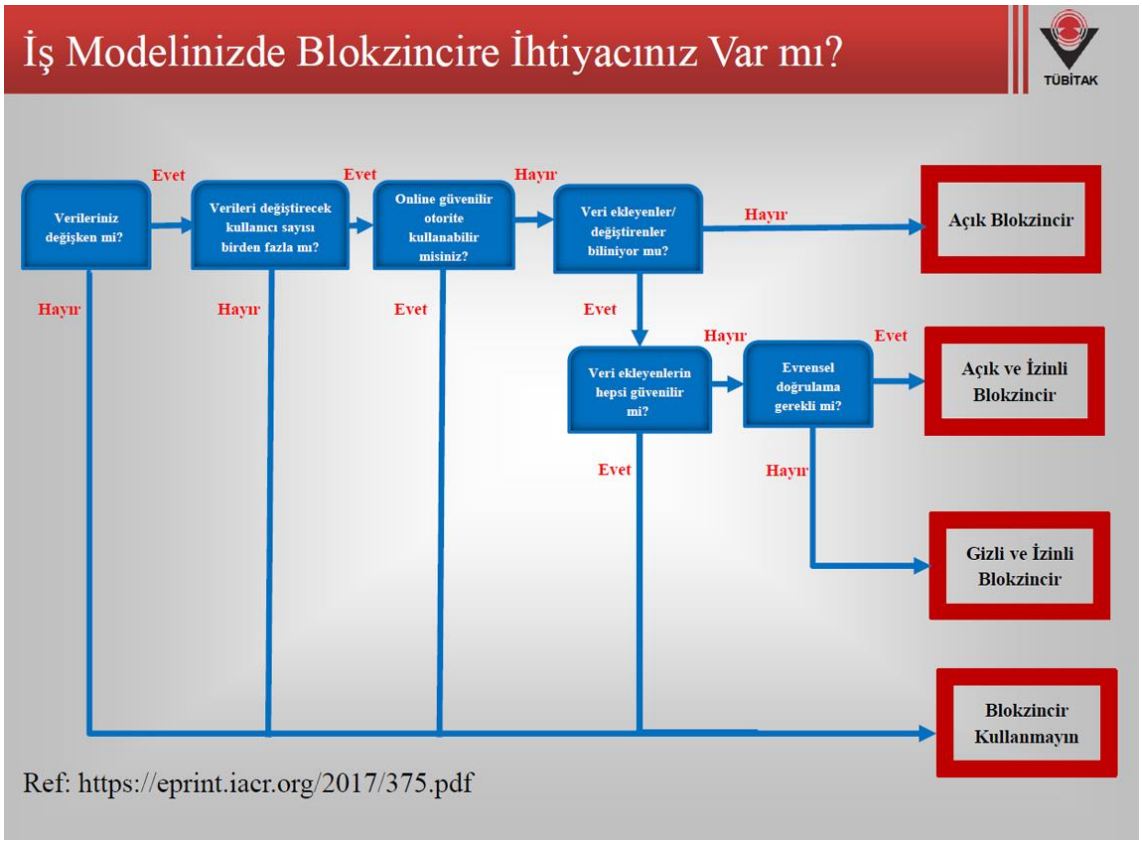
Blok zincirin genel olarak uygulama alanları şunlardır:

- Bankacılık
- FinTech
- Para Transferleri
- Değerli Belgelerin Yaratılması ve Saklanması
- E-Ticaret ve Ödemeler

- Hisse Senetleri ve Borsalar
- E-Noter
- Kişiden Kişiye Borçlanma ve Dağıtık Yapılı Kredi Sistemleri
- Bağış Sistemleri ve Mikro Ödemeler
- Bulut Bilişim ve Güvenli Bulut Depolama

3.3. Blok Zinciri Algoritması Kullanılma Nedenleri

Blok Zinciri Algoritması kullanımına kimi sistemler için oldukça gereksiz olmakla birlikte kimi sistemler için oldukça gerekli bir sistemdir. Şekil 3.1'de de gösterildiği gibi verilerin değiştirilemediği statik web sayfaları veya yazılımlar için blok zinciri kullanımı oldukça gereksizdir. Zaten verilerin eklenmesinin denetimini sağlamak için geliştirilmiş olan bu sistemde değişmeyen veri kontrol gerektirmez. Aynı şekilde veri işlemlerinin tek bir kişi gerçekleştiriyorsa ki bu kişiler genellikle webmaster ya da geliştiricinin ta kendisi olduğu için verilerin denetimi güvenli bir kişi tarafından sağlanır ve bu sayede blok zincirine gerek duyulmaz. Bu projedeki gibi birden fazla kişini aynı ya da farklı veri gruplarına ulaşabilmesi, değiştirebilmesi ve silebilmesi işlemlerinde kimin yaptığını saptamak ve ya yapılan işlemi doğrulatmak için blok zinciri algoritmaları kullanılabilir. Blok zinciri algoritmasının kullanım zorunluluğu olmadığı belirtilir.



Şekil 3.1: Blok Zinciri Kullanımının Gerekliliği Diyagramı

4. Tedarik Zinciri Otomasyonu

4.1. Hazırlık Aşaması

Projenin hazırlık aşaması ilk olarak projenin araştırılmasıyla başladı. Tedarik Zinciri Otomasyonlarında Blok Zinciri Algoritmasının kullanımının gerekliliği sorgulandı.

NET Framework içerisinde yer alan ASP.NET MVC web çatısı kullanılarak proje gerçekleştirildi. ASP.NET MVC web çatısının kullanmış olduğu mimari olan Model-Görünüm-Denetleyici mimarisi hızlı geliştirme yapabilme, çabuk öğrenebilme ve C# programlama dilinin artıları göz önüne alınarak bu projenin bu çatı üzerinde gerçekleştirilmesi kararlaştırıldı. Veritabanı olarak MS SQL Server kullanıldı. .NET

Framework ile geliştirme yapılırken MS SQL veritabanı ile olan uyumu ve MS SQL Server veritabanın kullanışlığı göz önüne alınarak bu veritabanı kullanıldı. Ayrıca veritabanı dağıtımı (Replication/ Distribution) imkanı sağlaması tercih edilmesindeki bir diğer nedenlerdendir. Ön-Uç (Front-End/Client Side) geliştirmelerinde temel araçlar olan HTML ve CSS kullanıldı. Boostrap çatısı sayesinde ön-yüz üzerinde görsellik arttırıldı ve sayfa boyutlandırması her cihaza uyumlu hale getirildi.

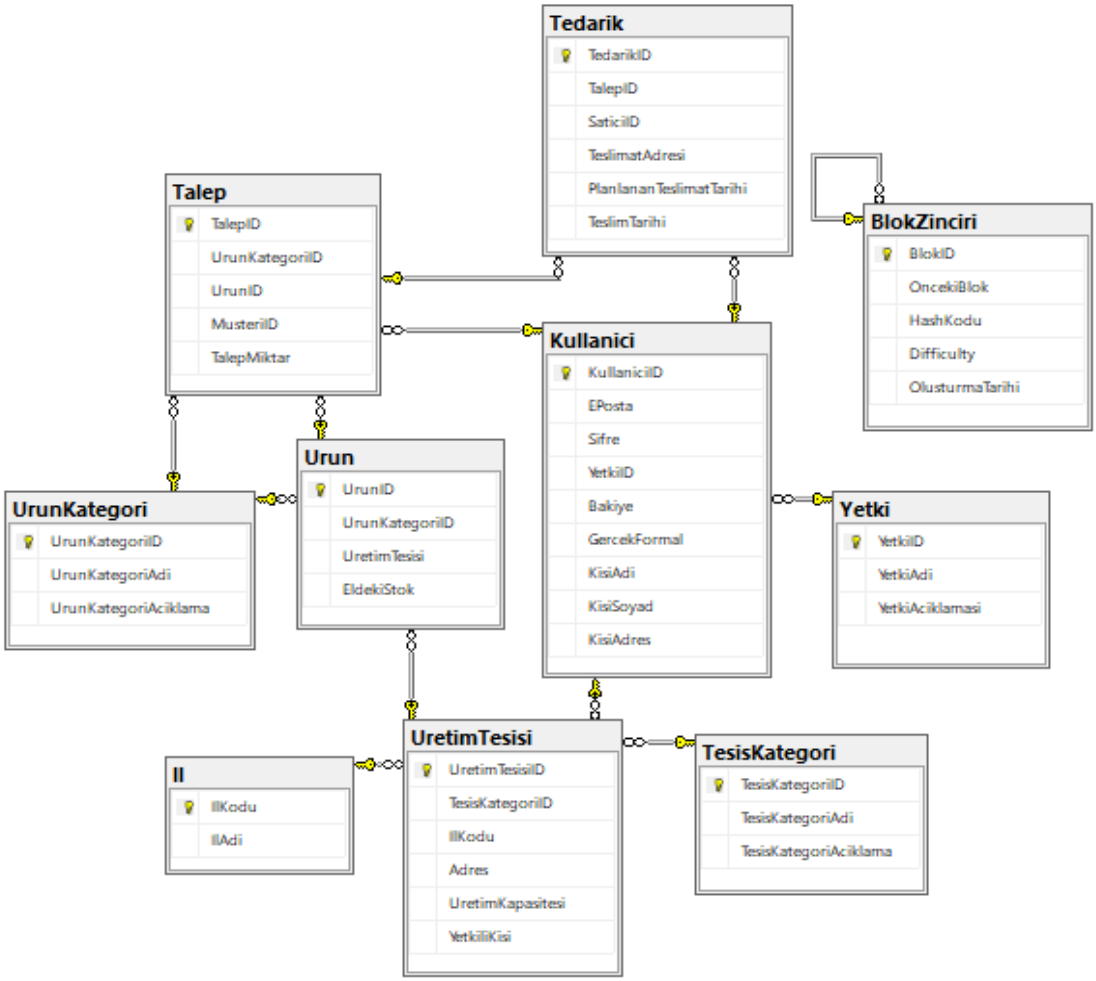
4.2. Model Oluşturulması

Tedarik Zinciri Süreci Talep ve Tedarik arasında süren bir süreç ve bu süreçte rol alan her bir kişi/kurum için model oluşturulmuştur. Projenin ilk sürümü için çok fazla detay içermeyen ve temel ihtiyaçları karşılamaya yetecek bir model oluşturuldu. Oluşturulan bu model geliştirilmeye uygun olup ileri sürümler için bir temel olacaktır. İlk olarak Kullanıcılar oluşturuldu. Kullanıcılar gerçek ya da tüzel kişiler olabilecek şekilde modellendi. Kullanıcıların eşsiz anahtar değerleri (Unique/Primary Key) T.C. Kimlik No. olarak belirlendi. Kullanıcıların yetkilerini sınırlayarak sistemin güvenliği ve yönetiminin kolaylaşması için Yetki adında bir tablo oluşturularak kullanıcılara bu yetkiler çerçevesinde erişim izi verilmesi sağlandı. Sistemdeki geri kalan diğer bütün modeller Kullanıcı modeli üzerinde şekilleniyor. Zaten Bölüm 2.2'de Tedarik Zincirinin nasıl yönetildiği ve nasıl işlediği açıklandığı için burada o bölümü referans alarak modelleme oluşturuldu. Kullanıcı Talep tablosuna kayıt oluşturarak yabancıl anahtar (Foreign Key) olarak yer alır. Talep tablosu içerisinde hangi ürünün talep edildiği bilgisi bulunduğu için Ürün Kategorisi ve Ürün tablolarının kayıtlarını yabancıl anahtar olarak tutar. Ürün tablosunda o ürünün hangi kategoride olduğu belirtileceği için Ürün Kategorisi tablosunun değerini yabancıl anahtar olarak tutar. Ayrıca ürünün üretim yeri ve eldeki stok bilgisi de tabloda barındırılır.

```
Create Database TedarikZinciri

Use TedarikZinciri

Create Table Il
(
	IlKodu int Primary Key,
	IlAdi nvarchar(15) Unique
```

```
);

Create Table Yetki
(
	YetkiID int Primary Key Identity(1,1),
	YetkiAdi nvarchar(10) Unique not null,
	YetkiAciklamasi nvarchar(255) not null
);

Create Table Kullanici
(
	KullaniciID int Primary Key Identity(1,1),
	EPosta nvarchar(50) Unique not null,
	Sifre nvarchar(25),
	YetkiID int Foreign Key References Yetki(YetkiID),
	Bakiye money not null,
	GercekFormal bit not null,
	KisiAdi nvarchar(25),
	KisiSoyad nvarchar(25),
	KisiAdres nvarchar(255)
);

Create Table TesisKategori
(
	TesisKategoriID int Primary Key Identity(1,1),
	TesisKategoriAdi nvarchar(50) Unique,
	TesisKategoriAciklama nvarchar(255) not null
);

Create Table UrunKategori
(
	UrunKategoriID int Primary Key Identity(1,1),
	UrunKategoriAdi nvarchar(25) Unique not null,
	UrunKategoriAciklama nvarchar(255) not null
);

Create Table UretimTesisi
(
	UretimTesisiID int Primary Key Identity(1,1),
	TesisKategoriID int Foreign Key References TesisKategori(TesisKategoriID),
	IlKodu int Foreign Key References Il(IlKodu),
	Adres nvarchar(255) not null,
	UretimKapasitesi decimal not null,
	YetkiliKisi int Foreign Key References Kullanici(KullaniciID),
);

Create Table Urun
(
	UrunID int Primary Key Identity(1,1),
	UrunKategoriID int Foreign Key References UrunKategori(UrunKategoriID),
	UretimTesisi int Foreign Key References UretimTesisi(UretimTesisiID),
	EldekiStok int not null
);

Create Table Talep
(
	TalepID int Primary Key Identity(1,1),
	UrunKategoriID int Foreign Key References UrunKategori(UrunKategoriID),
	UrunID int Foreign Key References Urun(UrunID),
	MusteriID int Foreign Key References Kullanici(KullaniciID),
	TalepMiktar int not null,
);
```

```
Create Table Tedarik
(
	TedarikID int Primary Key Identity(1,1),
	TalepID int Foreign Key References Talep(TalepID),
	SaticiID int Foreign Key References Kullanici(KullaniciID),
	TeslimatAdresi nvarchar(255) not null,
	PlanlananTeslimatTarihi datetime not null,
	TeslimTarihi datetime not null
);

Create Table BlokZinciri
(
	BlokID int Primary Key Identity(1,1),
	OncekiBlok int Foreign Key References BlokZinciri(BlokID),
	HashKodu nvarchar(64) not null,
	Difficulty int not null,
	OlusturmaTarihi datetime not null
);
```



Şekil 4.1: Veritabanı Varlık-İlişki Şeması

4.3. Blok Zinciri Algoritmasının Tasarımı ve Gerçekleştirilmesi

Blok Zinciri Algoritması geliştirilirken olabildiğince temel bir sistem kuruldu, karmaşık işlemlerden kaçınıldı. “SHA256” şifreleme algoritması kullanıldı hash kodu elde edildi. Hash kodu içerisine blok zinciri prensibindeki “Difficulty” özelliği için imza niteliği taşıyabilecek üç harften oluşan dokuz farklı sözcükten bir tanesi içerisine gömülecek şekilde tasarlandı. SHA256 algoritması .NET içerisinde gömülü olarak gelen sınıflar üzerinden kullanıldı, şifreleme algoritması için ayrıca bir kod yazılmadı.

Blok Zinciri Algoritmasının C# Programlama Dili ile gerçekleştirilmiş hali aşağıda verilmiştir.

```
public class Blok
{
   public String hash;
   private String data;
   String[] sign;
   SHA256CryptoServiceProvider sha;
   public Blok(List<String> datas, String previousHash, int diff)
   {
      this.data = String.Join(",", datas.ToArray());
      this.sha = new SHA256CryptoServiceProvider();
      this.sign = new String[9] { "KKY", "KKy", "Kky", "kky", "KML", "Kml",
"kml", "1104", "1997" };
      this.hash = CalculateHash(diff);
   }

   public String CalculateHash(int diff)
   {
      sha.ComputeHash(Encoding.Unicode.GetBytes(this.data));
      String calculatedhash = Convert.ToBase64String(sha.Hash);
      calculatedhash = calculatedhash.Insert(2, sign[diff]);
      return calculatedhash;
   }
}
public static class NoobChain
{
   static Blok blok;
   static Random rnd = new Random();
   static int difficulty = rnd.Next(0, 8);
   static TedarikZinciriDB tde = new TedarikZinciriDB();
   static BlokZinciri lastChain = tde.BlokZinciri.ToList().LastOrDefault();
   static BlokZinciri addBlock;
   public static void Olustur(List<String> veriler)
   {
      if (lastChain == null)
      {
         blok = new Blok(veriler, "0", difficulty);
         addBlock = new BlokZinciri
         {
            OncekiBlok = null,
            HashKodu = blok.hash,
            Difficulty = difficulty,
            OlusturmaTarihi = DateTime.Now
         };
```

```
            }
            else
            {
                blok = new Blok(veriler, lastChain.HashKodu, difficulty);
                addBlock = new BlokZinciri
                {
                    OncekiBlok = Convert.ToInt32(lastChain.BlokID),
                    HashKodu = blok.hash,
                    Difficulty = difficulty,
                    OlusturmaTarihi = DateTime.Now
                };
            }
            tde.BlokZinciri.Add(addBlock);
            tde.SaveChanges();
        }
    }
```

4.4. Denetleyici ve Görünüm Oluşturulması

Daha önce de bahsedildiği gibi MVC mimarisi kullanılan bir web çatısı kullanıldığı için adım adım bu mimarinin parçaları oluşturulur. Model oluşumu Bölüm 4.2'de anlatılmıştır. Denetleyici (Controller) sınıflar MVC mimarisi içerisinde kullanıcının isteklerine geri dönüş sağlayan ve gerekli yönlendirme (routing) işlemlerini gerçekleştiren yapıdır. Verilerin görüntülenmesini sağlayan bilgilerin aktarılması ya da verilerin düzeltimindeki arka planda çalışan kodlar bu sınıflar içerisinde yer alır. Görünüm (View) ASP.NET MVC içerisinde .cshtml uzantılı dosyalara yazılır. Bu dosyaların içerisine hem C# kodları hem de HTML, CSS, JavaScript vb. ön-uç kodları yazılabilmektedir. Bu dosyalar kullanıcının karşısına gelen sayfaların kodlarıdır.

Bu proje kapsamında denetleyici kodlarının büyük kısmını ASP.NET MVC web çatısının "Auto-Generate Code" özelliği kullanarak oluşturulmuştur. Auto-Generate Code özelliği ile Görünüm dosyalarıda kendiliğinden oluşmaktadır. Modelimizde yer alan her bir tablo için denetleyiciler ve buna bağlı en temel dört özelliğin (Oluşturma, Düzenleme, Görüntüleme ve Silme) Görünüm dosyaları projeye eklenmiştir.

4.5. Tümleştirme Çalışması

Veritabanı ve Blok Zinciri üreten sınıflarının gerçekleştirilmesiyle artık tümleştirme başlatılır. Oluşturulan Denetleyici sınıflarının içerisine o modele ait oluşturulan nesnenin niteliklerini bir List<String> veri yapısına tanımlayan ve Blok

Zinciri sınıfın bu List veri yapısını işlemesi sağlanacak bir metot denetleyicinin içerisine eklenir. Bu metotun kaynak kodları aşağıda verilmiştir.

```
private List<String> BilgileriListeyeAtma(Kullanici kullanici)
{
   List<String> bilgiler = new List<String>(9)
   {
      kullanici.KullaniciID.ToString(),
      kullanici.YetkiID.ToString(),
      kullanici.EPosta,
      kullanici.Sifre,
      kullanici.Bakiye.ToString(),
      kullanici.KisiAdi,
      kullanici.KisiSoyad,
      kullanici.KisiAdres,
      kullanici.GercekFormal.ToString()
   };
   return bilgiler;
}
```

Görünüm dosyaları içerisine Blok Zinciri ile alakalı herhangi bir ekleme yapılmamıştır. Bu işlem sadece Arka-Uç (Back-End/Server Side) kısmında gerçekleşmektedir.

Kaynakça

- TÜBİTAK, Blokzinciri [çevrimiçi], http://blokzincir.bilgem.tubitak.gov.tr/blok-zincir.html [Ziyaret Tarihi: 15 Haziran 2018]
- Logo Yazılım, 2017, Tedarik Zinciri ve Tedarik Zinciri Yönetimi Nedir? Temel Evreleri ve Faydaları Nelerdir? [çevrimiçi], https://blog.logo.com.tr/tedarik-zinciri-ve-tedarik-zinciri-yonetimi-nedir-temel-evreleri-ve-faydalari-nelerdir/ [Ziyaret Tarihi: 15 Haziran 2018]
- Lojistik Dünyası, 2011, Tedarik Zinciri Nedir? [çevrimiçi], http://www.lojistikdunyasi.net/tedarik-zinciri-nedir.html [Ziyaret Tarihi: 15 Haziran 2018]
- KOCAPINAR, Z. G., 2014, Tedarik Zinciri Nedir? [çevrimiçi], http://zaferkocapinar.com.tr/makaleler/8/Tedarik-Zinciri-Nedir/default.aspx [Ziyaret Tarihi: 15 Haziran 2018]
- KARAMAN, K., 2012, Tedarik Zinciri Yönetim Süreçleri [çevrimiçi], https://tedarikzinciri.wordpress.com/2012/01/16/tedarik-zinciri-yonetimi-surecleri/ [Ziyaret Tarihi: 15 Haziran 2018]
- AGARWAL, S., 2004, Cryptography in .NET [çevrimiçi], https://www.c-sharpcorner.com/article/cryptography-in-net/ [Ziyaret Tarihi: 15 Haziran 2018]
- Anonim, 2018, Blokchain Basics [çevrimiçi], https://www.investinblockchain.com/category/blockchain-basics/ [Ziyaret Tarihi: 15 Haziran 2018]
- Anonim, 2017, Configure Distribution [çevrimiçi], https://docs.microsoft.com/en-us/sql/relational-databases/replication/configure-distribution?view=sql-server-2017 [Ziyaret Tarihi: 15 Haziran 2018]
- Kass , 2017, Creating Your First Blokchain with Java Part 1 [çevrimiçi], https://medium.com/programmers-blockchain/create-simple-blockchain-java-tutorial-from-scratch-6eeed3cb03fa [Ziyaret Tarihi: 15 Haziran 2018]
- CAMPBELL, B., 2017, Blokchain by Example in SQL Server [çevrimiçi] , https://medium.com/@benjaminsky/blockchain-by-example-in-sql-server-8376b410128 [Ziyaret Tarihi: 15 Haziran 2018]
- Anonim, Blockchain Nedir? [çevrimiçi], https://www.kriptom.com/blockchain-nedir-sozluk/ [Ziyaret Tarihi: 15 Haziran 2018]

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	Kemal KOLCUOĞLU
Uyruğu	**:**	T. C.
Doğum Yeri ve Tarihi	**:**	KARAMAN – 11/04/1997
Telefon	**:**	05318128687
e-mail	**:**	kemal.kolcuoglu@outlook.com.tr

EĞİTİM

Derece		**Adı, İlçe, İl**	**Bitirme Yılı**
Lise	:	Hasan Çolak Anadolu Lisesi, Alanya, ANTALYA	2015
Üniversite	:	Selçuk Üniversitesi, Selçuklu, KONYA	-

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görevi**
2018	AMEDYA A.Ş.	Yazılım Geliştirme Stajyeri