



**T.C.  
SELÇUK ÜNİVERSİTESİ  
TEKNOLOJİ FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**Blok Zinciri Algoritmaları Kullanarak  
Tedarik Zinciri Otomasyonu Geliştirme  
(BATZO)**

**Abdül Gafur Kazım GÜLER - 123311014  
Kemal KOLCUOĞLU - 173311082**

**BİTİRME PROJESİ**

**Ocak - 2019  
KONYA  
Her Hakkı Saklıdır**

## **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  
Abdül Gafur Kazım GÜLER  
Kemal KOLCUOĞLU  
Tarih: 28/01/2019

## İçindekiler

|   |    |
|---|----|
| 1. Giriş .....  | 4  |
| 2. Tedarik Zinciri .....  | 4  |
| 2.1. Tedarik Zinciri Nedir? .....                                     | 4  |
| 2.2. Tedarik Zinciri Yönetimi .....                                   | 4  |
| 3. Blok Zinciri Algoritması.....                                      | 5  |
| 3.1. Blok Zinciri Nedir? .....  | 5  |
| 3.2. Blok Zinciri Algoritmasının Kullanım Alanları .....              | 6  |
| 3.3. Blok Zinciri Algoritması Kullanılma Nedenleri .....              | 7  |
| 4. Tedarik Zinciri Otomasyonu.....                                    | 8  |
| 4.1. Hazırlık Aşaması.....  | 8  |
| 4.2. Model Oluşturulması .....  | 9  |
| 4.3. Blok Zinciri Algoritmasının Tasarımı ve Gerçekleştirilmesi ..... | 15 |
| 4.4. Denetleyici ve Görünüm Oluşturulması .....                       | 17 |
| 4.5. Tümlleştirme Çalışması .....                                     | 17 |
| Kaynakça .....  | 19 |

# 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 bütününe 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 ıır 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:

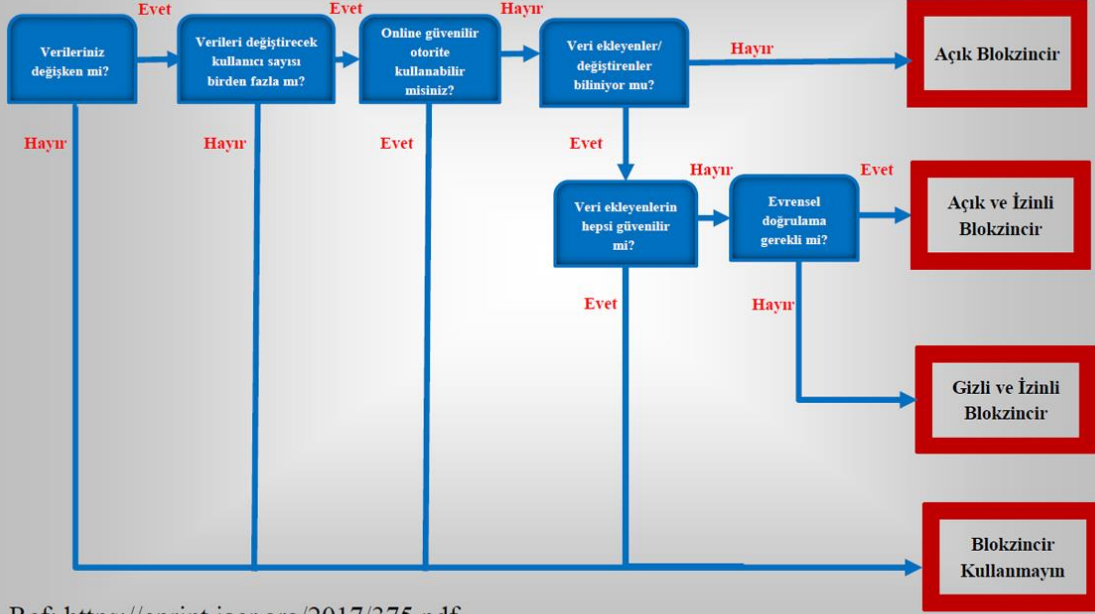
- Bankacılık
- FinTech
- Para Transferleri
- Değerli Belgelerin Yaratılması ve Saklanması
- E-Ticaret ve Ödemeler

- Hisse Senetleri ve Borsalar
- E-Noter
- Kişiden Kişiyeye 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 kişiyi aynı ya da farklı veri gruplarına ulaşabilmesi, değiştirebilmesi ve silebilmesi işlemlerinde kimin yaptığını saptamak ve ya yapılan işlemi doğrulamak için blok zinciri algoritmaları kullanılabilir. Blok zinciri algoritmasının kullanım zorunluluğu olmadığı belirtilir.

## İş Modelinizde Blokzincire İhtiyacınız Var mı?



Ref: <https://eprint.iacr.org/2017/375.pdf>

Şekil 3.1: Blok Zinciri Kullanımının Gerekliği 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 veritabanının kullanışlılığı 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. Bootstrap çatısı sayesinde ön-yüz üzerinde görsellik artırıldı 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 oluşturularak yabancı anahtar (Foreign Key) olarak yer alır. Talep tablosu içerisinde hangi ürünün talep edildiği bilgisi bulunduğu için Ürün Kategorisi ve Ürün tablolarının kayıtlarını yabancı anahtar olarak tutar. Ürün tablosunda o ürünün hangi kategoride olduğu belirtileceği için Ürün Kategorisi tablosunun değerini yabancı anahtar olarak tutar. Ayrıca ürünün üretim yeri ve eldeki stok bilgisi de tabloda barındırılır.

```
Create Database tedarikzinciri;  
  
Use tedarikzinciri; Go  
  
SET DATEFORMAT ymd
```

```

CREATE TABLE [yetki] ([YetkiID] INT IDENTITY(1,1) ,[YetkiAdi] VARCHAR(15) NULL
,[YetkiAciklama] VARCHAR(100) NULL ,CONSTRAINT [yetki_yetki_PRIMARY] PRIMARY KEY
NONCLUSTERED ([YetkiID]));

CREATE NONCLUSTERED INDEX [yetki_YetkiID] ON [yetki] ([YetkiID]);

CREATE NONCLUSTERED INDEX [yetki_YetkiID_2] ON [yetki] ([YetkiID]);

CREATE NONCLUSTERED INDEX [yetki_YetkiID_3] ON [yetki] ([YetkiID]);

CREATE NONCLUSTERED INDEX [yetki_YetkiID_4] ON [yetki] ([YetkiID]);


CREATE TABLE [il] ([IlKodu] INT NOT NULL ,[IlAdi] VARCHAR(20) NULL ,CONSTRAINT
[il_il_PRIMARY] PRIMARY KEY NONCLUSTERED ([IlKodu]));

CREATE NONCLUSTERED INDEX [il_IlKodu] ON [il] ([IlKodu]);

CREATE NONCLUSTERED INDEX [il_IlKodu_2] ON [il] ([IlKodu]);

CREATE NONCLUSTERED INDEX [il_IlKodu_3] ON [il] ([IlKodu]);

CREATE NONCLUSTERED INDEX [il_IlKodu_4] ON [il] ([IlKodu]);


CREATE TABLE [ilce] ([IlceKodu] INT NOT NULL ,[IlceAdi] VARCHAR(25) NULL ,[IlKodu] INT
NULL ,CONSTRAINT [ilce_ilce_PRIMARY] PRIMARY KEY NONCLUSTERED ([IlceKodu]),
CONSTRAINT [ilce_FK_IlKodu_Ilce] FOREIGN KEY ("IlKodu") REFERENCES "il" ( "IlKodu" ) ON
UPDATE NO ACTION ON DELETE NO ACTION);

CREATE NONCLUSTERED INDEX [ilce_IlceKodu] ON [ilce] ([IlceKodu]);

CREATE NONCLUSTERED INDEX [ilce_IX_IlKodu] ON [ilce] ([IlKodu]);

ALTER TABLE [ilce] NOCHECK CONSTRAINT [ilce_FK_IlKodu_Ilce];


CREATE TABLE [tesiskategori] ([TesisKategoriID] INT IDENTITY(1,1) ,[TesisKategoriAdi]
VARCHAR(30) NULL ,[TesisKategoriAciklama] VARCHAR(255) NULL ,CONSTRAINT
[tesiskategori_tesiskategori_PRIMARY] PRIMARY KEY NONCLUSTERED ([TesisKategoriID]));

CREATE NONCLUSTERED INDEX [tesiskategori_TesisKategoriID] ON [tesiskategori]
([TesisKategoriID]);

CREATE NONCLUSTERED INDEX [tesiskategori_TesisKategoriID_2] ON [tesiskategori]
([TesisKategoriID]);

CREATE NONCLUSTERED INDEX [tesiskategori_TesisKategoriID_3] ON [tesiskategori]
([TesisKategoriID]);

```

```
CREATE TABLE [adres] ([AdresID] INT IDENTITY(1,1) , [IlKodu] INT NULL , [IlceKodu] INT NULL , [AdresAciklama] VARCHAR(255) NULL , CONSTRAINT [adres_adres_PRIMARY] PRIMARY KEY NONCLUSTERED ([AdresID]), CONSTRAINT [adres_FK_IlKodu_Adres] FOREIGN KEY ("IlKodu") REFERENCES "il" ( "IlKodu" ) ON UPDATE NO ACTION ON DELETE NO ACTION, CONSTRAINT [adres_FK_IlceKodu_Adres] FOREIGN KEY ("IlceKodu") REFERENCES "ilce" ( "IlceKodu" ) ON UPDATE NO ACTION ON DELETE NO ACTION);
```

```
CREATE NONCLUSTERED INDEX [adres_AdresID] ON [adres] ([AdresID]);
```

```
CREATE NONCLUSTERED INDEX [adres_IX_IlKodu] ON [adres] ([IlKodu]);
```

```
CREATE NONCLUSTERED INDEX [adres_IX_IlceKodu] ON [adres] ([IlceKodu]);
```

```
CREATE NONCLUSTERED INDEX [adres_AdresID_2] ON [adres] ([AdresID]);
```

```
CREATE NONCLUSTERED INDEX [adres_AdresID_3] ON [adres] ([AdresID]);
```

```
CREATE NONCLUSTERED INDEX [adres_AdresID_4] ON [adres] ([AdresID]);
```

```
CREATE NONCLUSTERED INDEX [adres_AdresID_5] ON [adres] ([AdresID]);
```

```
CREATE NONCLUSTERED INDEX [adres_AdresID_6] ON [adres] ([AdresID]);
```

```
ALTER TABLE [adres] CHECK CONSTRAINT [adres_FK_IlKodu_Adres];
```

```
ALTER TABLE [adres] CHECK CONSTRAINT [adres_FK_IlceKodu_Adres];
```

```
CREATE TABLE [kisi] ([KisiID] INT IDENTITY(1,1) , [TCKNO] VARCHAR(11) NULL , [YetkiID] INT NULL , [Ad] VARCHAR(25) NULL , [Soyad] VARCHAR(25) NULL , [EPosta] VARCHAR(75) NULL , [Telefon] VARCHAR(25) NULL , [Etkin] SMALLINT NULL , [Guid] NVARCHAR(40) NULL , CONSTRAINT [kisi_kisi_PRIMARY] PRIMARY KEY NONCLUSTERED ([KisiID]), CONSTRAINT [kisi_FK_YetkiID_Kisi] FOREIGN KEY ("YetkiID") REFERENCES "yetki" ( "YetkiID" ) ON UPDATE NO ACTION ON DELETE NO ACTION);
```

```
CREATE NONCLUSTERED INDEX [kisi_KisiID] ON [kisi] ([KisiID]);
```

```
CREATE NONCLUSTERED INDEX [kisi_IX_YetkiID] ON [kisi] ([YetkiID]);
```

```
ALTER TABLE [kisi] NOCHECK CONSTRAINT [kisi_FK_YetkiID_Kisi];
```

```
CREATE TABLE [firma] ([FirmaID] INT IDENTITY(1,1) , [FirmaAdi] VARCHAR(255) NULL , [FirmaAciklama] VARCHAR(255) NULL , [Telefon] VARCHAR(25) NULL , [Fax] VARCHAR(25) NULL , [EPosta] VARCHAR(100) NULL , [Etkin] SMALLINT NULL , [AdresID] INT NULL , [FirmaTipi] VARCHAR(20) NULL , CONSTRAINT [firma_firma_PRIMARY] PRIMARY KEY NONCLUSTERED ([FirmaID]), CONSTRAINT [firma_FK_AdresID] FOREIGN KEY ("AdresID") REFERENCES "adres" ( "AdresID" ) ON UPDATE NO ACTION ON DELETE NO ACTION);
```

```
CREATE NONCLUSTERED INDEX [firma_FirmaID] ON [firma] ([FirmaID]);
```

```

CREATE NONCLUSTERED INDEX [firma_IX_AdresID] ON [firma] ([AdresID]);

CREATE NONCLUSTERED INDEX [firma_FirmaID_2] ON [firma] ([FirmaID]);

CREATE NONCLUSTERED INDEX [firma_FirmaID_3] ON [firma] ([FirmaID]);

ALTER TABLE [firma] NOCHECK CONSTRAINT [firma_FK_AdresID];

CREATE TABLE [uretimtesisi] ([UretimTesisID] INT IDENTITY(1,1) , [TesisKategoriID] INT
NULL , [AdresID] INT NULL , [FirmaID] INT NULL , [TesisAdi] VARCHAR(255) NULL
, [TesisAciklama] VARCHAR(255) NULL , [UretimKapasitesi] REAL NULL , CONSTRAINT
[uretimtesisi_uretimtesisi_PRIMARY] PRIMARY KEY NONCLUSTERED ([UretimTesisID]),
CONSTRAINT [uretimtesisi_FK_AdresID_UretimTesis] FOREIGN KEY ("AdresID") REFERENCES
"adres" ( "AdresID" ) ON UPDATE NO ACTION ON DELETE NO ACTION, CONSTRAINT
[uretimtesisi_FK_FirmaID_UretimTesis] FOREIGN KEY ("FirmaID") REFERENCES "firma" (
"FirmaID" ) ON UPDATE NO ACTION ON DELETE NO ACTION, CONSTRAINT
[uretimtesisi_FK_TesisKategoriID_UretimTesis] FOREIGN KEY ("TesisKategoriID")
REFERENCES "tesiskategori" ( "TesisKategoriID" ) ON UPDATE NO ACTION ON DELETE NO
ACTION);

CREATE NONCLUSTERED INDEX [uretimtesisi_UretimTesisID] ON [uretimtesisi]
([UretimTesisID]);

CREATE NONCLUSTERED INDEX [uretimtesisi_IX_TesisKategoriID] ON [uretimtesisi]
([TesisKategoriID]);

CREATE NONCLUSTERED INDEX [uretimtesisi_IX_AdresID] ON [uretimtesisi] ([AdresID]);

CREATE NONCLUSTERED INDEX [uretimtesisi_IX_FirmaID] ON [uretimtesisi] ([FirmaID]);

CREATE NONCLUSTERED INDEX [uretimtesisi_UretimTesisID_2] ON [uretimtesisi]
([UretimTesisID]);

ALTER TABLE [uretimtesisi] NOCHECK CONSTRAINT [uretimtesisi_FK_AdresID_UretimTesis];

ALTER TABLE [uretimtesisi] CHECK CONSTRAINT [uretimtesisi_FK_FirmaID_UretimTesis];

ALTER TABLE [uretimtesisi] NOCHECK CONSTRAINT
[uretimtesisi_FK_TesisKategoriID_UretimTesis];

CREATE TABLE [urunkategori] ([UrunKategoriID] INT IDENTITY(1,1) , [UrunKategoriAdi]
VARCHAR(30) NULL , [UrunKategoriAciklama] VARCHAR(255) NULL , CONSTRAINT
[urunkategori_urunkategori_PRIMARY] PRIMARY KEY NONCLUSTERED ([UrunKategoriID]);

CREATE NONCLUSTERED INDEX [urunkategori_UrunKategoriID] ON [urunkategori]
([UrunKategoriID]);

CREATE NONCLUSTERED INDEX [urunkategori_UrunKategoriID_2] ON [urunkategori]
([UrunKategoriID]);

```

```

CREATE TABLE [urun] ([UrunID] INT IDENTITY(1,1) , [UrunKategoriID] INT NULL
,[UretimTesisID] INT NULL , [UrunAdi] VARCHAR(50) NULL , [UrunAciklama] VARCHAR(255)
NULL , [UrunBirimi] VARCHAR(255) NULL , [UrunFiyati] REAL NULL , [Stok] REAL NULL
,CONSTRAINT [urun_urun_PRIMARY] PRIMARY KEY NONCLUSTERED ([UrunID]), CONSTRAINT
[urun_FK_UretimTesisID_Urun] FOREIGN KEY ("UretimTesisID") REFERENCES "uretimtesisi" (
"UretimTesisID" ) ON UPDATE NO ACTION ON DELETE NO ACTION, CONSTRAINT
[urun_FK_UrunKategoriID_Urun] FOREIGN KEY ("UrunKategoriID") REFERENCES "urunkategori" (
"UrunKategoriID" ) ON UPDATE NO ACTION ON DELETE NO ACTION);

CREATE NONCLUSTERED INDEX [urun_UrunID] ON [urun] ([UrunID]);

CREATE NONCLUSTERED INDEX [urun_IX_UrunKategoriID] ON [urun] ([UrunKategoriID]);

CREATE NONCLUSTERED INDEX [urun_IX_UretimTesisID] ON [urun] ([UretimTesisID]);

CREATE NONCLUSTERED INDEX [urun_UrunID_2] ON [urun] ([UrunID]);

ALTER TABLE [urun] NOCHECK CONSTRAINT [urun_FK_UretimTesisID_Urun];

ALTER TABLE [urun] NOCHECK CONSTRAINT [urun_FK_UrunKategoriID_Urun];


CREATE TABLE [talep] ([TalepID] INT IDENTITY(1,1) , [UrunID] INT NULL , [FirmaID] INT
NULL , [TalepMiktari] REAL NULL , [TalepTarihi] DATETIME NULL ,CONSTRAINT
[talep_talep_PRIMARY] PRIMARY KEY NONCLUSTERED ([TalepID]), CONSTRAINT
[talep_FK_FirmaID_Talep] FOREIGN KEY ("FirmaID") REFERENCES "firma" ( "FirmaID" ) ON
UPDATE NO ACTION ON DELETE NO ACTION, CONSTRAINT [talep_FK_UrunID_Talep] FOREIGN KEY
("UrunID") REFERENCES "urun" ( "UrunID" ) ON UPDATE NO ACTION ON DELETE NO ACTION);

CREATE NONCLUSTERED INDEX [talep_TalepID] ON [talep] ([TalepID]);

CREATE NONCLUSTERED INDEX [talep_IX_UrunID] ON [talep] ([UrunID]);

CREATE NONCLUSTERED INDEX [talep_IX_FirmaID] ON [talep] ([FirmaID]);

ALTER TABLE [talep] NOCHECK CONSTRAINT [talep_FK_FirmaID_Talep];

ALTER TABLE [talep] NOCHECK CONSTRAINT [talep_FK_UrunID_Talep];


CREATE TABLE [firmapersonel] ([FirmaPersonelID] INT IDENTITY(1,1) , [FirmaID] INT NULL
,[KisiID] INT NULL ,CONSTRAINT [firmapersonel_firmapersonel_PRIMARY] PRIMARY KEY
NONCLUSTERED ([FirmaPersonelID]), CONSTRAINT [firmapersonel_FK_FirmaID_FirmaPersonel]
FOREIGN KEY ("FirmaID") REFERENCES "firma" ( "FirmaID" ) ON UPDATE NO ACTION ON DELETE
NO ACTION, CONSTRAINT [firmapersonel_FK_KisiID_FirmaPersonel] FOREIGN KEY ("KisiID")
REFERENCES "kisi" ( "KisiID" ) ON UPDATE NO ACTION ON DELETE NO ACTION);

CREATE NONCLUSTERED INDEX [firmapersonel_FirmaPersonelID] ON [firmapersonel]
([FirmaPersonelID]);

CREATE NONCLUSTERED INDEX [firmapersonel_IX_FirmaID] ON [firmapersonel] ([FirmaID]);

CREATE NONCLUSTERED INDEX [firmapersonel_IX_KisiID] ON [firmapersonel] ([KisiID]);

```

```

ALTER TABLE [firmapersonel] CHECK CONSTRAINT [firmapersonel_FK_FirmaID_FirmaPersonel];

ALTER TABLE [firmapersonel] NOCHECK CONSTRAINT [firmapersonel_FK_KisiID_FirmaPersonel];


CREATE TABLE [tedarik] ([TedarikID] INT IDENTITY(1,1) ,[KisiID] INT NULL ,[TalepID]
INT NULL ,[TeslimAdresiID] INT NULL ,[CikisAdresiID] INT NULL ,[TeslimTarihi]
DATETIME NULL ,[Durum] VARCHAR(25) NULL ,[TedarikUcreti] REAL NULL ,CONSTRAINT
[tedarik_tedarik_PRIMARY] PRIMARY KEY NONCLUSTERED ([TedarikID]), CONSTRAINT
[tedarik_FK_CikisAdresiID_Tedarik] FOREIGN KEY ("CikisAdresiID") REFERENCES "adres" (
"AdresID" ) ON UPDATE NO ACTION ON DELETE NO ACTION, CONSTRAINT
[tedarik_FK_KisiID_Tedarik] FOREIGN KEY ("KisiID") REFERENCES "kisi" ( "KisiID" ) ON
UPDATE NO ACTION ON DELETE NO ACTION, CONSTRAINT [tedarik_FK_TalepID_Tedarik] FOREIGN
KEY ("TalepID") REFERENCES "talep" ( "TalepID" ) ON UPDATE NO ACTION ON DELETE NO
ACTION, CONSTRAINT [tedarik_FK_TeslimAdresiID_Tedarik] FOREIGN KEY ("TeslimAdresiID")
REFERENCES "adres" ( "AdresID" ) ON UPDATE NO ACTION ON DELETE NO ACTION);


CREATE NONCLUSTERED INDEX [tedarik_TedarikID] ON [tedarik] ([TedarikID]);

CREATE NONCLUSTERED INDEX [tedarik_IX_KisiID] ON [tedarik] ([KisiID]);

CREATE NONCLUSTERED INDEX [tedarik_IX_TeslimAdresiID] ON [tedarik]
([TeslimAdresiID]);

CREATE NONCLUSTERED INDEX [tedarik_IX_CikisAdresiID] ON [tedarik] ([CikisAdresiID]);

CREATE NONCLUSTERED INDEX [tedarik_IX_TalepID] ON [tedarik] ([TalepID]);

ALTER TABLE [tedarik] NOCHECK CONSTRAINT [tedarik_FK_CikisAdresiID_Tedarik];

ALTER TABLE [tedarik] NOCHECK CONSTRAINT [tedarik_FK_KisiID_Tedarik];

ALTER TABLE [tedarik] NOCHECK CONSTRAINT [tedarik_FK_TalepID_Tedarik];

ALTER TABLE [tedarik] NOCHECK CONSTRAINT [tedarik_FK_TeslimAdresiID_Tedarik];

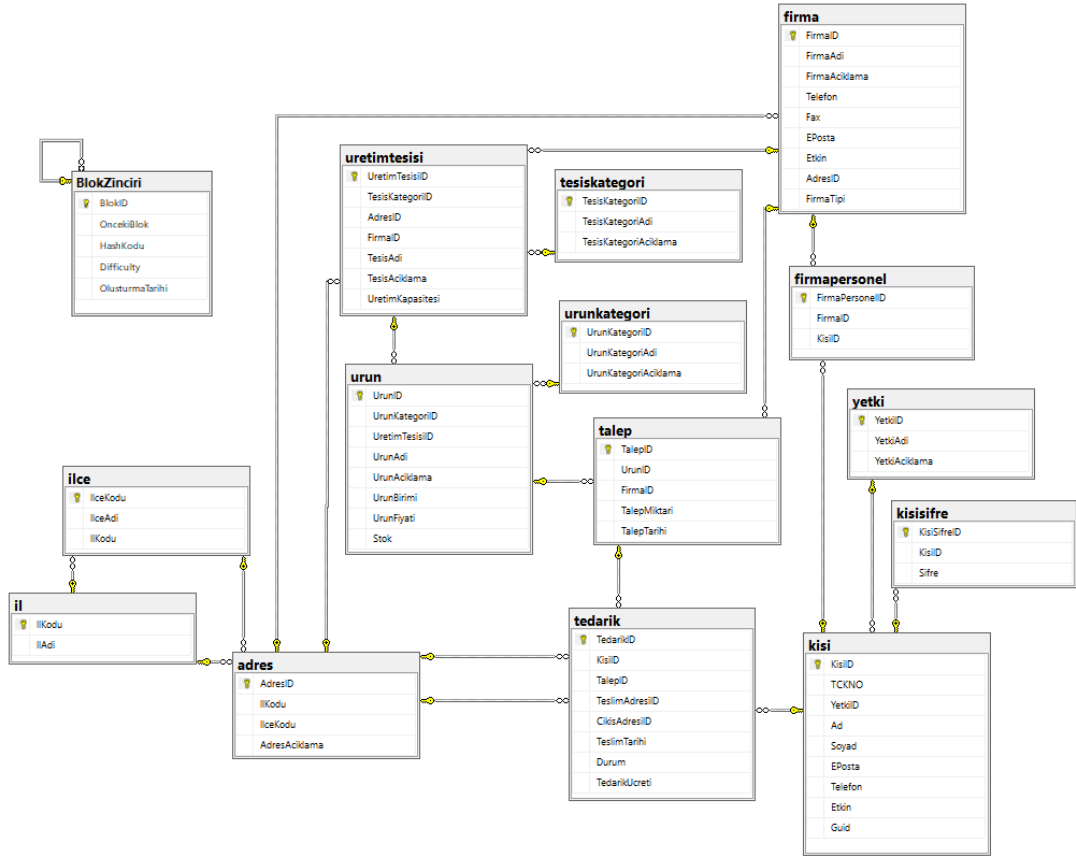

CREATE TABLE [kisisifre] ([KisiSifreID] INT IDENTITY(1,1) ,[KisiID] INT NULL ,[Sifre]
VARCHAR(15) NULL ,CONSTRAINT [kisisifre_kisisifre_PRIMARY] PRIMARY KEY NONCLUSTERED
([KisiSifreID]), CONSTRAINT [kisisifre_FK_KisiID_KisiSifre] FOREIGN KEY ("KisiID")
REFERENCES "kisi" ( "KisiID" ) ON UPDATE NO ACTION ON DELETE NO ACTION);


CREATE NONCLUSTERED INDEX [kisisifre_KisiSifreID] ON [kisisifre] ([KisiSifreID]);

CREATE NONCLUSTERED INDEX [kisisifre_IX_KisiID] ON [kisisifre] ([KisiID]);

CREATE NONCLUSTERED INDEX [kisisifre_KisiSifreID_2] ON [kisisifre] ([KisiSifreID]);

```



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",
"km1", "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 dosyalarında 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 sağlayacak 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, Blockchain 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 Blockchain with Java Part 1 [çevrimiçi], <https://medium.com/programmers-blockchain/create-simple-blockchain-java-tutorial-from-scratch-6eed3cb03fa> [Ziyaret Tarihi: 15 Haziran 2018]
- CAMPBELL, B., 2017, Blockchain 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- | QR YAZILIM          | Yazılım Geliştirici            |
| 2018- | Nutuva Organizasyon | Yazılım Geliştirici            |
| 2018  | AMEDYA A.Ş.         | Yazılım Geliştirme<br>Stajyeri |