

# İÇİNDEKİLER

	Sayfa
KISALTMA LİSTESİ	iv
ŞEKİL LİSTESİ	v
ÖNSÖZ	vi
ÖZET	vii
ABSTRACT	viii
1. GİRİŞ	1
1.1 Motivasyon	1
1.2 Tezin Amacı ve Araştırma Soruları	1
1.3 Literatürdeki Diğer Çalışmalar	2
1.4 Tezin Yapısı	3
2. LİTERATÜR BİLGİLERİ	4
2.1 Kriptografi	4
2.1.1 Özetleme fonksiyonları, açık ve özel anahtar şifrelemesi	4
2.1.2 Dijital imza	5
2.2 Block Zinciri Teknolojisi	7
2.2.1 Block zinciri tanımı	7
2.2.2 Block zincirinin sunduğu özellikler	8
2.2.2.1 Dağıtık veri tabanı	8
2.2.2.2 Uçtan uça iletişim	8
2.2.2.3 Şeffaflık	9
2.2.2.4 Kayıtların değiştirilemez olması	9
2.2.3 Block zinciri uygulamaları	9
2.3 Bir Blok Zinciri Uygulaması Ethereum	10
2.3.1 Akıllı sözleşmeler ve Solidity	11
2.3.1.1 Akıllı sözleşme nedir?	12
2.3.1.2 Solidity nedir?	13
2.3.2 Ethereum ve Bitcoin arasındaki farklar	13
3. KULLANILAN TEKNOLOJİ VE ARAÇLAR	15
3.1 Truffle	15
3.2 Ganache	15
3.3 Web3.js	15
3.4 Metamask	15
3.5 JavaScript	15
3.6 NodeJS	16
3.7 NPM	16
3.8 Geliştirme Araçları	16

3.8.1	Sublime text	16
3.8.2	iTerm2	16
3.9	Github	16
4.	UYGULAMANIN GELİŞTİRİLMESİ	17
4.1	Truffle Dizin Yapısı	17
4.2	Akıllı Sözleşmenin Yazılması	18
4.3	Web3 ve Sözleşmeleri Örneklenmesi	20
4.4	Uygulamanın Arayüzünün Yazılması	21
4.4.1	Html dosyalarının oluşturulması	21
4.4.2	JavaScript dosyasının oluşturulması	23
4.5	Metamask Üzerinde Hesap Oluşturma	24
5.	KURULUM	27
5.1	Yerel Sunucuda Kurulum	27
6.	SONUÇ	29
6.1	Uygulamayı Diğer Alanlara Genişletme	29
KAYNAKLAR		38

## **KISALTMA LİSTESİ**

EVM	Ethereum Virtual Machine, Ethereum Sanal Makinesi
SHA	Secure Hash Algorithm, Güvenli Özetleme Algoritması
RSA	Ron Rivest, Adi Shamir ,Leonard Adleman
CPU	Central Processing Unit, Merkezi İşlem Birimi
ASIC	Application Specific Integrated Circuit, Uygulamaya Entegre Edilmiş Devreler
RPC	Remote Procedure Call, Uzaktan Yordam Çağrısı
API	Application Programming Interface, Uygulama Programlama Arayüzü
HTML	Hypertext Markup Language, Hiper Metin İşaretleme Dili

## ŞEKİL LİSTESİ

Şekil 2.1 Veri şifrelemenin sınıflandırılması.	4
Şekil 2.2 Merkle ağaçları ve blok zinciri yapısı arasındaki bağlantı	6
Şekil 2.3 Blok Zincir Mimarisi	8
Şekil 2.4 EVM'nin yapısı Kaynak: Edureka	10
Şekil 2.5 Ethereum blok yapısı	11
Şekil 2.5 Akıllı Sözleşme Mantiğı	12
Şekil 4.1 Truffle'ın indirilmesi	17
Şekil 4.2 Truffle için Boş Proje Yapısının İndirilmesi	17
Şekil 4.3 Adayın Struct Yapısı	19
Şekil 4.4 Mapping Yapısı	19
Şekil 4.5 Oy verme şartları	19
Şekil 4.6 Blok zinciri ile Akıllı Sözleşmenin İletişim Kurulmasının Sağlanması	20
Şekil 4.7 EVM için bytecode oluşturulması	20
Şekil 4.8 Yazılan sözleşmenin taşınması(migrate) işlemi	21
Şekil 4.9 Tarayıcıda Metamask olup olmadığının kontrolü	21
Şekil 4.10 Oy kullanma sayfası	22
Şekil 4.11 Admin portal	22
Şekil 4.12 Web3 ve Truffle contract kütüphanelerinin projeye dahil edilmesi	23
Şekil 4.13 Akıllı sözleşmedeki fonksiyonlarla etkileşim kurulması	23
Şekil 4.14 Web3 örneğinin oluşturulması ve gas miktarının belirlenmesi	24
Şekil 4.15 Metamask adres bilgilerinin girilmesi	25
Şekil 4.16 Metamask'a hesap eklenmesi	26
Şekil 4.17 Ganache test hesapları	26
Şekil 5.1 Git üzerinden projenin kopyalanması	27
Şekil 5.2 Sunucunun çalıştırılması	27
Şekil 5.3 Yazılan sözleşmenin taşınması(migrate) işlemi	28
Şekil 5.4 Sunucuya tarayıcı üzerinden oy verme sayfasına erişim	28
Şekil 5.5 Sunucuya tarayıcı üzerinden admin portalına erişim	28

## ÖNSÖZ

Lisans eğitimim boyunca her zaman tecrübesi ve bilgi birikimiyle kendisinden çok şeyler öğrendiğim ayrıca bu çalışmanın hazırlanmasında, yorumları ve eleştirileriyle katkıda bulunan danışman hocam Öğr. Gör. Yunus Özen'e ve tez konusunun belirlenmesinden tezin son aşamasına gelene kadar bana yol gösteren ve değerli vakitlerini ayırarak bana destek olan değerli dostum Sefa İrken'e teşekkürlerimi sunarım. Ayrıca tezimin başından sonuna kadar desteklerini esirgemeyen, en zor günlerimde yanımda olan aileme teşekkür ederim.

Bu tezin tamamlanmasında eksikliklerimi gidermek için çalışmalarından, makalelerinden, örnek kodlarından yararlandığım dünyanın dört bir yanında yaşayan sayısız geliştiriciye teşekkürü bir borç bilirim.

## ÖZET

Günümüzde birçok ülkede yapılan oylamalarda oy veren kişinin kimliğini doğrulamak, atılan oyların güvenli bir şekilde gerekli kurumlara iletilmesi ve kullanılan oyların takibini sağlayarak seçimi kazananı belirleyecek şeffaflığı ve doğruluğu sağlamakta sıkıntılar yaşanmaktadır. Seçim güvenliği ve güvenilirliği konusunda geleneksel birçok önlem alınmaktadır. Bu önlemler hem maliyetli hem karmaşık hem de tartışmaya açık oluyorlar. Bu sorunlardan dolayı blok zinciri teknolojisi çok uygun bir aday olarak karşımıza çıkıyor.

Blok zinciri, daha demokratik veya katılımcı bir karar alma mekanizmasına sahip yeni yönetim sistemlerinin geliştirilmesine ve herhangi bir insan müdahalesi olmaksızın bir bilgisayar ağı üzerinde çalışabilen merkezi olmayan (özerk) örgütlere olanak sağlamaktadır. Bu da yukarıda bahsedilen güvenlik sorunlarını ortadan kaldırmaktadır.

Birçok uygulamada bu teknoloji denenmeye başlamıştır. Bu çalışmada; blok zincir teknolojisinin oy kullanma uygulaması konusunda şu ana kadar yapılan belli başlı çalışmalar incelenerek yapılmıştır. Solidity programlama dilinde geliştirilmiş akıllı sözleşme sistemi ile birlikte izin verilen bir blok zinciri mimarisi ve ortak kriptografik araçlar önerilmiştir ve kullanılmıştır. Devlet yönetiminde blok zinciri uygulamalarının potansiyel yararları ve bu uygulamaların olası etkilerine değinilmiştir.

**Anahtar Kelimeler:** Blok zinciri, Oylama Sistemi, Akıllı Sözleşme, Kriptografi, Uçtan Uca Ağ

## **ABSTRACT**

Nowadays, there are problems in confirming the identity of the person who votes in the polls in many countries, communicating the votes cast to the necessary institutions in a safe way and ensuring the transparency and correctness that will determine the winner by following the used votes. Many traditional measures are taken in regards to election safety and reliability. These measures are both costly and complex and open to debate. Because of these problems, block-chain technology is a very suitable candidate.

The block-chain allows the development of new management systems with a more democratic or participatory decision-making mechanism and the possibility of decentralized (autonomous) organizations that can work on a computer network without any human intervention. This, in turn, removes the security issues mentioned above.

In many applications this technology has begun to be tried. In this study; block chain technology has been carried out by examining the main studies on voting application so far. A permitted peer-to-peer block-chaining architecture and common cryptographic tools have been proposed and used in conjunction with the intelligent contract system developed in the Solidity programming language. The potential benefits of block chain implementations in government administration and the possible effects of these applications have been addressed.

**Keywords:** Blockchain, Voting System, Smart Contract, Cryptography, Peer to Peer Network

## 1. GİRİŞ

### 1.1 Motivasyon

Günümüzde dünyanın birçok yerinde bir seçim belirli aşamalardan geçerek gerçekleşir. Öncelikle seçim günü halk oy sandığına gider, sandık görevlisi kimlik kontrolünü yapar ve ilgili pusulayı vererek kabine girip seçmenin oyunu kullanmasını ister. Oy kullandıktan sonra kabinden çıkıp oylar sandığa atılır ve oy diğer oy pusulaları ile karışır. Böylece oy verme işlemi gizliliğe uygun yapılmış olur. Bundan sonrasında seçim görevlileri oyları sayarlar. Halk, bu noktada güvenmek zorunda olduğu bir süreçten geçer. Güvenmek zorunda olduğumuz bu süreçte önemli iki nokta vardır. Bu noktalardan biri geçerli ve geçersiz oylarla ilgili doğru kararın verilmesi ki sandık görevlileri bu konuda çoğu kez fikir ayrılığına düşebiliyor. Bir diğer önemli nokta ise seçim için kullanılan bilgisayarların ve yazılımlarının güvenliği. Örneğin Yüksek Seçim Kurulu'nun kullandığı SEÇSİS sisteminde sayım değerlerinin sisteme yanlış girilmesi veya kötü niyetli bir şekilde tek tuşla değiştirilebilmesi seçim için güvenilirliği ortadan kaldırır. Çünkü sonuçlar tek bir merkezde tutulur ve o sonuçlar değiştirilirse aksini ispatlamak zorlaşır. Buradan yola çıkılırsa mevcut sistemlerin toplamının bir birleşimi ve daha az hata eğilimli ve güvene dayalı bir sistemin geliştirilmesi gerekmektedir.

Kamu ve özel kuruluşların dijitalleşmesi geliştikçe, müşteriler ve vatandaşlar yeni güvenlik açıklarına maruz kalmaktadır. Örneğin kimliklerin veya kişisel bilgilerin hackerlar tarafından çalınması olasılığı için endişeleniyoruz. Geleneksel bir veri tabanı sistemi bu gereksinimleri yalnızca kısmen yerine getirir ve bu nedenle alternatif teknolojiler araştırılmalıdır. 2008 yılında, Bitcoin ve Blok Zinciri teknolojisinin ortaya çıkması ile değişmez, kriptografik olarak güvenli ve dağıtık bir veritabanı sistemi temeli atılmıştır. (Nakamoto, 2008). Blok zinciri ayrıca, daha demokratik veya katılımcı bir karar alma mekanizmasına sahip yeni yönetim sistemlerinin geliştirilmesine ve herhangi bir insan müdahalesi olmaksızın bir bilgisayar ağı üzerinde çalışabilen merkezi olmayan (özerk) örgütlere olanak sağlamaktadır

### 1.2 Tezin Amacı ve Araştırma Soruları

Bu lisans tezinin amacı, birbirine güvenmeyen taraflar arasında güvenli bir şekilde oy kullanmak için blok zinciri teknolojisinin ve akıllı sözleşmelerin nasıl kullanılabileceğini



göstermektedir. Araştırmanın sonuçları birçok kullanım örneğine açık bir şekilde uygulanabilir olacaktır. Araştırma hedefi üç araştırma sorusuna ayrılabilir:

**AS1: Güvenilir oylama nasıl elde edilir?**

Bu soruyu cevaplamak için bir literatür taraması yapılacak, ayrıca mevcut çerçeveler ve veri depolama teknolojileri (blok zinciri) için bir çalışma yürütülecektir. İkinci bölüm, farklı teknolojileri, uygulamanın gereklilikleri ile karşılaştırarak yapılacaktır. Bu araştırma sorusu blok zinciri uygulamasının bir bütün olarak mimarisine odaklanmış ve bir güvenlikten değerlendirmeye çalışılmıştır.

**AS2: Seçim ve benzeri oylamaların maliyetleri nasıl azaltılır?**

Bu araştırma sorusunun cevabı sonuç bölümüne açıklanacaktır.

**AS3: Blok zinciri teknolojisi ile hangi uygulamalar geliştirildi?**

Blok zinciri çoğunlukla Bitcoin kripto para ile ilişkisi için bilinir. Bitcoin, para birimi işlemlerinde Blok zinciri teknolojisini kullanır. Bununla birlikte, Bitcoin kripto para blok zinciri teknolojisini kullanan tek çözüm değildir. Bu nedenle blok zinciri teknolojisini kullanarak geliştirilen güncel uygulamaları bulmak önemlidir. Diğer uygulamaları belirlemek, blok zincirini kullanmanın diğer yollarını ve yollarını anlamaya yardımcı olabilir. Bölüm 1.3 ve 2.2.3' te blok zinciri teknolojisinin farklı uygulamaları ile ilgili araştırma sonuçları incelenebilir.

### **1.3 Literatürdeki Diğer Çalışmalar**

Uygun bir yapı kazandırılması halinde anonim olarak yapılan seçimler ve oylamalar için pekala blok zinciri teknolojisi kullanılabilir. Seçimlere şeffaflık kazandırabilecek blok zinciri sayesinde sahte oy kullanımının önüne geçilebilir ve katılımcılar kendi oylarını kontrol edebilirler.

Oylar blok zinciri üzerinde olacağı için devletler ve seçmenlerin denetim iznine sahip olacaklar; oyların hiçbirinin değiştirilmediğine, geri döndürülmediğine, kaldırılmadığına ve gayri meşru oyların eklenmediğine emin olacaklar.

- Blok zincir oy verme şirketi olan Follow My Vote ([www.followmyvote.com](http://www.followmyvote.com)) paydaşları ve üyeleri ile uçtan uca blok zincir oylama çözümü alfa sürümünü piyasaya sürdü.

- Güney Kore hükümeti, Blocko şirketi (www.blocko.io) Blok zinciri üzerinde yeni bir oy sistemi oluşturulmasını destekliyor. Blocko'nun Coystack platformunu, oylama projeleri ve seçim sonuçlarını almak blok zinciri sistemi kurmak için çalışıyor. Bu program sayesinde vatandaşlar kendi fikirlerini önerecek, karar vericiler ve yerel hükümetler en iyi ve kazanan fikir / projeleri finanse edecek.
- Ethereum Stack Exchange kullanıcıları için bir soru – cevap ve oylama sitesidir. Platform merkezi olmayan bir uygulama platformu ve akıllı sözleşmeleri etkin hale getiren bir blok zinciridir.
- BitCongress merkezi olmayan, P2P (eşler arası), açık kaynaklı bir oylama sistemi protokolüdür. Herkesin kendi spesifik ihtiyacı için kullanabileceği veya kullanabileceği bir açık kaynak oylama protokolü olarak çalışabilir. BitCongress biraz da dahil olmak üzere şifreleme teknolojilerinin (Bitcoin, BitMessage, BitTorrent, Proof of existence / Varoluş Kanıtı ve Reddit) karışımıdır.(Çağlar O., 2018).

#### 1.4 Tezin Yapısı

Tezin konusunun aydınlatılması, konunun önemi veya neden araştırıldığını açıklayan giriş bölümü

Blok zinciri teknolojisinin tanımı ve özelliklerinin açıklanması. Ethereum, Solidity ve akıllı sözleşmelerin tanımı. Ethereum ve Bitcoin arasındaki farkların anlatılması

Tezde anlatılan yazılımın geliştirilmesinde kullanılan programlama dilleri, araçlar ve kütüphanelerin neler olduğu, kısa tarihi.

Yazılımın geliştirme süreci

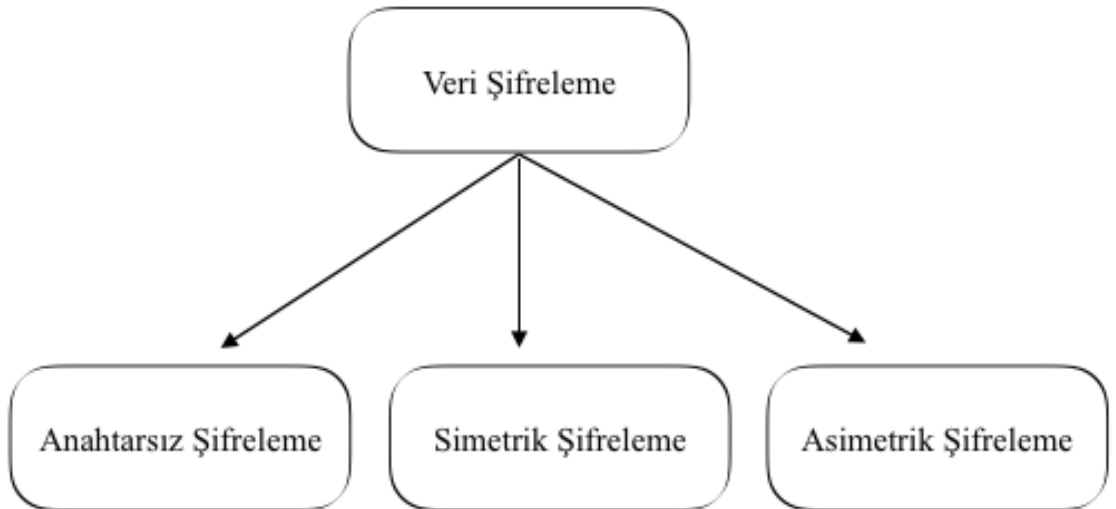
Geliştirilen yazılımın kurulumu

## 2. LİTERATÜR BİLGİLERİ

Bu bölümde, kriptografi, blok zinciri teknolojisi ve akıllı sözleşmeler gibi ilgili konular hakkında bazı temel kavramlara kısa bir giriş yapılacak ve açıklanacaktır. Bundan sonra, mevcut ve hali hazırda muhafaza edilmiş, ilgili blok zinciri platformlarının bir açıklaması verilecektir.

### 2.1 Kriptografi

Kriptografi, verileri istenmeyen alıcılar için faydasız hale getirmek amacıyla çeşitli teknikler kullanarak verilerin dönüştürülmesini sağlar. Bu bağlamda faydasızlık, iki temel eylemin engellenmesi anlamına gelmektedir; veriden bilgi çıkarıp yanlış verileri eklemek veya verileri değiştirmek. (Alfred J. Menezes, 1996).



### Şekil 2.1 Veri şifrelemenin sınıflandırılması.

Veri şifreleme üç dalda sınıflandırılabilir. Şekil 2.1’ de gösterildiği gibi anahtarsız, simetrik ve asimetrik şifreleme. Anahtarsız şifreleme, bir mesajı şifrelemek için anahtar kullanmayan işlevlerdir. Örneğin özetleme fonksiyonları. Simetrik şifreleme, bir mesajı şifrelemek ve şifreyi çözmek için aynı anahtarı kullanırken; asimetrik anahtar şifrelemede farklı anahtarlar kullanılır. Bu anahtarlar açık(public) ve kapalı- özel(private) anahtar olarak geçmektedir.

#### 2.1.1 Özetleme fonksiyonları, açık ve özel anahtar şifrelemesi

Bu tez bağlamında blok zinciri ve gizlilik anlayışını en çok ilgilendiren anahtarsız şifrelemede kullanılan özetleme fonksiyonlarıdır. Özetleme fonksiyonu, kendisine verilen bir değerden benzersiz uzunluğa sahip bir çıktı üreten fonksiyonlardır. Asıl dosyalar yerine verileri özet haline getirip daha sonra karşılaştırmak daha kolaydır. Eğer aynı çıktıyı üreten iki farklı girdi bulunursa bu duruma çatışma denir ve özetleme fonksiyonu kırılmış olur. Bir özetleme fonksiyonundan beklenenler:

- Aynı girdi ile sürekli aynı çıktı oluşur ve farklı girdilerin çıktıları birbiriyle aynı olamaz.
- Çıktılar belirli uzunluklarda anahtar üretebilmelidir.
- Şifreleme algoritmalarından farklı olarak tek yönlü olmalıdır. Yani fonksiyonlar tarafından üretilen anahtarlardan fonksiyonlara verilen değer elde edilmemelidir.
- Çatışmalara olanak vermemelidir.

En güvenli kabul edilen, yani tersine çevrilmesi en zor olan özetleme fonksiyonu SHA-3’tür. İdeal özetleme fonksiyonunun tüm kriterlerini yerine getirir. Özetleme fonksiyonları, bilgisayarların tanımlama, karşılaştırma veya başka verilere karşı yürüttüğü hesaplamalarda uygundur. Bu fonksiyonlar blok zincirlerinin yapısının daha iyi anlaşılması için gerekli bir temeldir.

Simetrik anahtar şifreleme, bir mesajın göndericisinin ve alıcının mesajı şifrelemek ve şifresini çözmek için kullanılan tek bir ortak anahtarı paylaştığı bir şifreleme sistemidir. Bu şifreleme algoritması daha basit ve hızlıdır, ancak ana dezavantajı güvenli anahtar dağıtımının zorluğudur. En popüler simetrik anahtar sistemi Veri Şifreleme Standardı’dır(DES).

Asimetrik anahtar şifreleme, verileri şifrelemek ve şifresini çözmek için açık ve kapalı

anahtarlar kullanır. Şifrelemeyi yapan anahtara açık anahtar, şifreyi çözen anahtar ise özel anahtardır. Açık anahtarlar herkese dağıtılabilir. Ama hangi anahtarın kime ait olduğundan emin olunmalıdır. Bu yüzden sertifikalar kullanılmaktadır. Sertifika açık anahtar ile sahibinin kimliği arasındaki bağlantının belgesidir. Özel anahtar ise sadece şifreyi çözecek kullanıcıda bulunur, açık anahtar ise gizli değildir. En popüler asimetrik anahtar sistemi RSA'dir ve günümüzde yaygın olarak kullanılmaktadır.

### **2.1.2 Dijital imza**

Kriptografi kullanarak kimlik doğrulama ve reddetme gerçekleştirmek için dijital imzalar kullanılır. Başka bir deyişle, belirli bir kişinin / aygıtın belirli bir mesajı göndermesini sağlamak için, tıpkı mektupların eskiden gönderenin eliyle imzalanması gibi, dijital olarak imzalanması gerekir. Dijital imzaların oluşturulmasında ve doğrulanmasında dijital sertifikalar kullanılır. Günümüzde yüksek güvenlik gereksinimini karşılamak için kullanılan dijital imza kullanıcılara bilgi bütünlüğü, kimlik doğrulama ve inkar edilemezlik sunmayı hedefler.

Bilgi Bütünlüğü: Verinin izinsiz olarak üzerinde oynama yapılmasını engeller.

Kimlik Doğrulama: Verinin ve gönderenin geçerli olmasını sağlar.

İnkar Edilemezlik: Elektronik ortamda yapılan işlemlerin inkar edilmesini önler.



Şekil 2.2 Merkle ağaçları ve blok zinciri yapısı arasındaki bağlantı

Blok zinciri teknolojisinin anlaşılması için gerekli olan şey Merkle ağaçları kavramıdır. Merkle ağaçları, kriptografi ve bilgisayar bilimlerinde, özet ağacı ya da Merkle ağacında her yaprak düğümü veri bloğunun özet değerini, her yaprak olmayan düğüm ise kendi alt düğümlerinin kriptografik özet değerlerini içerir.

Merkle ağacı büyük veri yapılarının verimli ve güvenli bir şekilde doğrulanmasını sağlar. Merkle ağaçları, özet listeleri ve özet zincirlerinin genelleştirilmiş halidir.(Merkle Ağaçları, Vikipedi) Şekil 2.2’de gösterildiği gibi herhangi bir blok, kendisinden önceki ve sonraki bloklara özet algoritması ile bağlanmıştır. Genellikle, özetleme fonksiyonları için SHA-2 gibi bir kriptografik özet işlevi kullanılır. Verinin değiştirilmemesi garanti altına alınır ve blok zinciri, bitcoin ve Ethereum uçtan uca ağları gibi sistemlerin yapısında kullanılır.

## 2.2 Block zinciri teknolojisi

Genel olarak kabul edilen blok zincirleme teknolojisine tek bir resmi tanımı yoktur. Bir çok tanım başlangıç noktası olarak Bitcoin’i kullanır ve onun ilk uygulaması olan kripto para

birimi ile blok zinciri teknolojisini açıklar. Alternatif tanımlara gelirsek, Ethereum'un kurucusu Vitalik Buterin tarafından yapılan bir tane mevcuttur:

*Bir blok zinciri, herkesin programlarını yükleyebildiği ve programların kendiliğinden yürütülmesi için bırakabileceği, her programın mevcut ve önceki tüm durumlarının her zaman herkesin görebileceği ve üzerinde çalışan programların çok güçlü kripto ekonomik olarak güvence altına alındığı bir bilgisayardır. Zincir, tam olarak blok zinciri protokolünün belirttiği şekilde yürütülmeye devam edecektir..* (Buterin, V. 2015). Buterin tarafından yapılan blok zinciri tanımı çok titiz veya teknik değildir ve kesinlikle Bitcoin tanımı ile aynı değildir, ama blok zinciri sistemlerinin birçok özelliğini içermeyi başarır. Yukarıdaki tanım, blok zincirinin neredeyse tüm mevcut uygulamalarını kapsayan çok geniş bir tanedir. Bu nedenle, okuyucular tarafından iyi anlaşılması için daha ayrıntılı bir açıklama aşağıda verilecektir. İlk olarak, bölüm 2.2.1'de oluşturulan blok zincirleme teknolojileri açıklanacaktır.

### **2.2.1 Blok zinciri tanımı**

Günümüzde bilgisayar sistemleri merkezi sistemlerden uzaklaşıp dağıtık sistemlere geçmektedir. Uçtan uca(P2P) dosya paylaşımı protokolü olan BitTorrent veya güvenli ve anonim ağ katmanı olan I2P, bunun güzel örneklerindendir. Fakat bu mimarinin en başarılı uygulamalarından biri Bitcoin ile ortaya çıktı. Blok zinciri teknolojisi, Satoshi Nakamoto'nun 2008 yılında yayınlanan Bitcoin: Eşler Arası Elektronik Nakit Sistemi makalesinde kripto para olarak tanımlanan Bitcoin'in arkasında olan bir teknoloji bileşenini, kriptografik olarak birbirine zincirlenmiş bir dizi veri bloğu olarak tanımlanmaktadır. (Nakamoto, 2008). Yani Bitcoin üretimi ve Bitcoin ile alakalı yapılan işlemlerin kaydı Blok Zinciri adı verilen temel bir teknoloji ile tutulmaktadır.

Genel bir tanım olarak *"Blok zincirinin temelini, merkezi olmayan ve güvenilir yöntemlerle topluca tutulan güvenilir bir veri tabanının teknik bir planı" olduğunu belirtmektedir.*" (Tian, 2016). Blok zinciri teknolojisi, günümüzün problemlerinden olan, merkezi sistem yapısını dağıtık hale getirerek bu sistemlerin daha verimli çalışmasını sağlamaktadır. Blok zincir modeli iki temel kavramdan oluşmuştur.

Kayıtlar: Her bir blok yapısının üzerine oluşturulduğu içerik bilgisidir. Bilgiden kasıt; müşteri kaydı, para transferi vb. olarak düşünülebilir.

**Bloklar:** Kayıtlar birleştirilip blokların içerisine yazılır. Blok zincirindeki bir blok içerisinde temel olarak; kendinden önceki bloğun özet değeri, kendi özet değeri ve gerçekleşen işlemler listesi bulunur. Bir bloğun oluşturulması anında özetleme fonksiyonları ve dijital imza kullanılır.



Şekil 2.3 Blok Zincir Mimarisi

Şekil 2.3’de de gösterildiği gibi blok zinciri Oluşum(Genesis) bloğu ile başlar ve her blok bir öncekine içinde bulunduğu değer ile bağlanır. Sistem bu şekilde sonsuza kadar uzanır.

## 2.2.2 Block zincirinin sunduğu özellikler

Bu bölümde blok zincirlerinin önemli özellikleri açıklanacaktır.

### 2.2.2.1 Dağıtık Veri Tabanı

Günümüzdeki yaklaşımlarda tek merkezli veya çok merkezli ağ yapısı kullanılmaktadır. Blok zincirlerinde ise dağıtık(distributed) ağa bağlanan tüm kullanıcılara açık bir yapı üzerinde tüm bilgisayarlarda bir birinin kopyası tutulmaktadır.Bu şekilde aracı bir kuruma ihtiyaç gereksinimi ortadan kalkmakta ve aracılardan getirdiği işlem karmaşıklığı ve ekstra maliyetleri ortadan kaldırmaktadır. Ayrıca merkezi olan ağ yapılarında gerçekleşen sorunları(saldırıları vb.) etkisiz hale getirmektedir.

### 2.2.2.2 Uçtan Uça İletişim

Taraflar arasında iletişim kurulması için herhangi bir merkezi yapı kullanılması yerine bireysel düğümler bilgileri eşler arası bir ağda birbirlerine doğrudan iletir ve depolar.



(Nakamoto, 2008). Ayrıca bloklar arasındaki fikir birliği(consensus) nedeniyle herhangi bir merkeze ihtiyaç duyulmaz

### **2.2.2.3 Şeffaflık**

Blok zinciri teknolojisinin en çekici yönlerinden biri, sağlayabileceği gizlilik derecesidir. Bununla birlikte, bu durum gizlilik ve şeffaflığın etkili bir şekilde nasıl bir arada var olabileceği konusunda kafa karışıklığına yol açmaktadır. Bu kısımda, blok zincirinin iki kavramı kullanıcıların yararına nasıl dengelediğini inceleyeceğiz.

Bir blok zincirinin şeffaflığı, her bir açık adresinin sahiplerinin ve işlemlerinin görüntülemeye açık olmasından kaynaklanmaktadır. Yani şeffaflıktan maksat bir blok zinciri ağına bir düğüm eklendiğinde sistemdeki daha önceden şimdiye kadar gerçekleşmiş tüm işlemleri(transaction) görebilir. Bir araştırmacının kullanılması ve kullanıcının genel adresi ile donatılmış olan işletmelerin sahip oldukları işlemleri ve gerçekleştirdikleri işlemleri görmek mümkündür. Bu büyüklükteki şeffaflık, özellikle büyük işletmelere ilişkin olarak, daha önce finansal sistemlerde mevcut değildi ve bugüne kadar mevcut olmayan bir dereceye kadar hesap verebilirlik katıyor.

### **2.2.2.4 Kayıtların değiştirilemez olması**

Bir blok zinciri değişmez(immutable) olacak şekilde tasarlanmıştır. Blok zinciri, oluşum bloğundan başlayarak son bloğa kadar olan tüm işlemleri depolar. Bu zincirlerin ve blok kimliklerinin arkasında yatan güçlü bir kriptografik ve ileri matematik yapısından dolayı zinciri bozmak veya içindeki bilgiyi değiştirmek şuanki teknoloji ile imkansızla yakındır. Eğer bir bloktaki bir bilgi değiştirilirse zincir yapısından dolayı bu tüm düğümler içinde görülebilir hale gelecektir.

### **2.2.3 Block zinciri uygulamaları**

İlk popüler uygulaması olan sanal para (bitcoin) ile tanınan blok zinciri teknolojisinin birçok farklı alanda kullanılabileceği keşfedilmeye başlanmıştır. Teknolojinin zaman içinde geliştirilmesi ve tanınmasına paralel gelecekte uygulama alanlarının çok çeşitleneceğini söylenebilir. Aşağıda blok zinciri teknolojisinin uygulamalarına ilişkin bir liste verilmiştir (Usta ve Doğanekin, 2017).

- Bankacılık

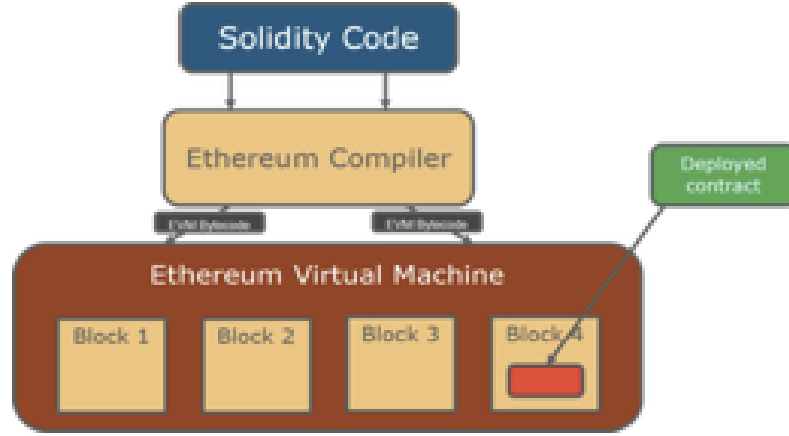
- Para Transferleri
- Değerli Belgelerin Yaratılması ve Saklanması
- E-Ticaret ve Ödemeler
- Hisse Senetleri ve Borsalar
- E-Noter
- Kişiden Kişiyne Borçlanma ve Dağıtık Yapılı Kredi Sistemleri
- Bağış Sistemleri ve Mikro Ödemeler

### 2.3 Bir blok zinciri uygulaması Ethereum

Ethereum, ilk defa Kuzey Amerika Bitcoin Konferansında Vitalik Buterin tarafından tanıtılmıştır. Dışarıdan bakıldığında altcoin gibi görünsede aslında diğer altcoinlere oranla çok daha fazla geniş avantaj ve etki alanına sahiptir.

En basit haliyle Ethereum, geliştiricilerin merkezi olmayan uygulamaları oluşturmasını ve dağıtmasını sağlayan blok zinciri teknolojisine dayanan açık bir yazılım platformudur. Ethereum'un cazibesi, geliştiricilerin akıllı sözleşmeler oluşturmaya olanak verecek şekilde oluşturulmuş olmasıdır. Akıllı sözleşmeler, belirli koşullar sağlandığında görevleri otomatik olarak yürüten komut dosyalarıdır. Şekil 2.4'deki görüldüğü gibi ortam sadece bir sanal makine biçimindeki blok zincirinde mevcut olduğundan, akıllı sözleşmeler tamamen ağdan, dosya sisteminden veya düğüm makinelerindeki diğer işlemlerden yalıtılmıştır. Ethereum ile ilgili bilinmesi gereken bazı terimler şunlardır.

Ethereum Sanal Makinesi (EVM): Ethereum kullanıcıların kendi operasyonlarını oluşturmalarını sağlar. Ethereum Sanal Makinesi (EVM) bunu mümkün kılar. Ethereum'un çalışma ortamı olarak, EVM akıllı sözleşmeler yürütür.

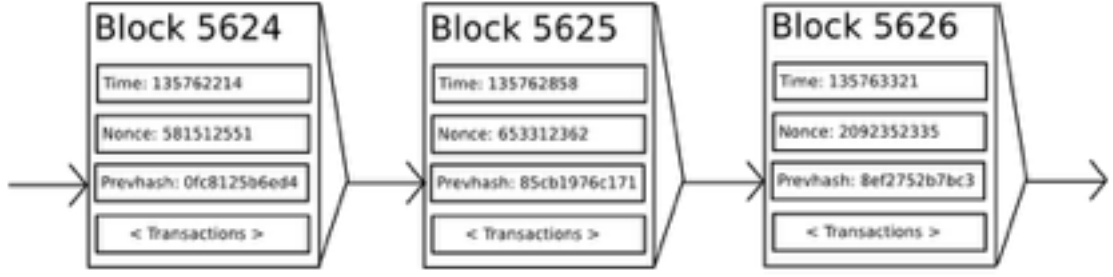


Şekil 2.4 EVM'nin yapısı Kaynak: Edureka

Eter(Ether, ETH): Ethereum ağındaki işlemlerde ödeme yapmak için kullanılan kripto para biriminin adıdır. Şu anda 1 ETH'nin değeri 529,97 dolar civarındadır. Genel işlemler ve hizmetler için ödeme yapmanın bir parçası olan Eter, aynı zamanda, EVM'de hesaplama yapmak için kullanılan Gas'ı bulmak için de kullanılır.

Gas Limit: Her sözleşmeyi çalıştırılabilmesi için belirli bir miktarda eter ödenmesi gerekiyor. Ne kadar eter ödenmesi gerektiği çalıştırılacak olan uygulamanın boyutuna ve sözleşme içinde saklamak istediğimiz veriye göre değişiyor. Örneğin, bir uygulama çalıştırılıyor ve doğal olarak bunun bir maliyeti oluyor. Bu maliyet madencilere ödeniyor. Böylelikle aslında madencinin CPU'sundan belli bir miktar çalıştırma ücreti ödenmiş oluyor. Böylelikle madenciler para kazanmış oluyor.

Madencilik: Eter madenciliği, ETH'nin ağ işlemlerinin onaylanması yoluyla gerçekleştirilmesidir. Daha spesifik olarak madencilik, Ethereum blok zincirindeki tüm aktiviteyi teyit etmek için gerçekleşen işlemlerin onaylanmasına katılımdır. Bu her platformda yapılabileceği gibi, kişisel bilgisayarların yanı sıra özel tasarlanmış platformlar için de kullanılabilir. Madencilikteki zorluk, ETH'nin toplanmasıyla tüketilen elektriğe harcanandan daha fazla para üretmektir. Bitcoin'in madencilik sistemleri, işlemci gücünüze ya da elinizdeki sistem sayısına bağlı olarak değiştiği için biraz adaletsiz durum ortaya çıkartır. Ethereum'da ise ekran kartlarıyla eşitlikçi ASIC denilen bir sistemle üreticiler arasındaki dengenin korunması sağlanır



Şekil 2.5 Ethereum blok yapısı

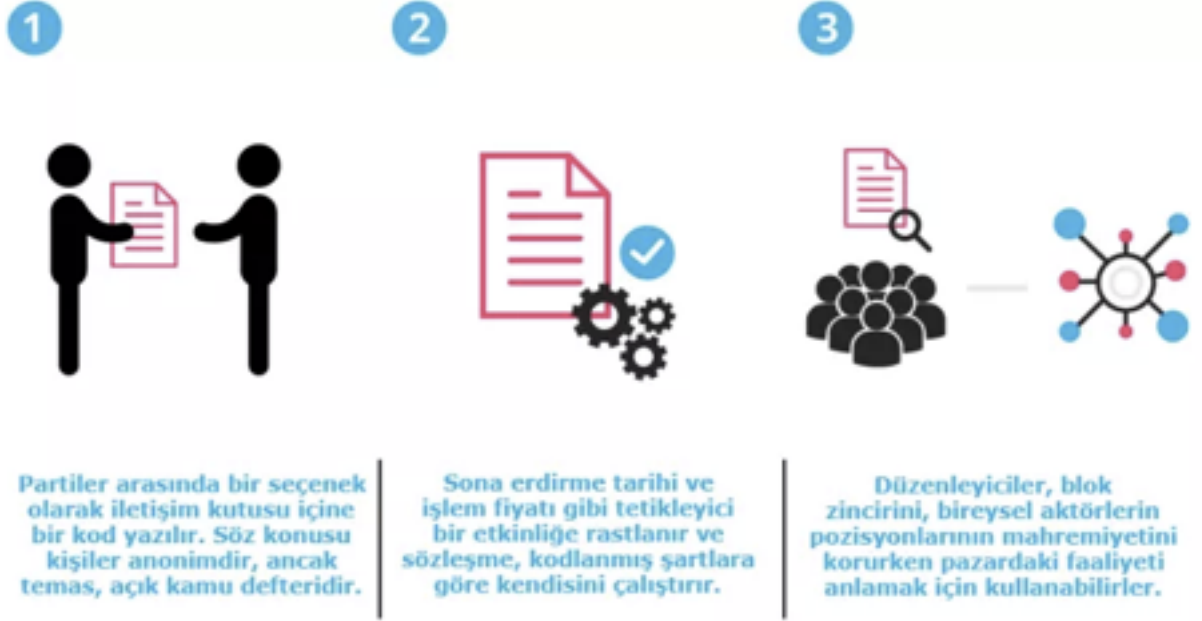
### 2.3.1 Akıllı sözleşmeler ve Solidity

Akıllı sözleşmeler ya da İngilizce ifadeyle smart contracts, ifadesi ve kavramı Nick Szabo tarafından 1996'da POS (satış noktası) gibi elektronik işlem yöntemlerinin işlevselliğini dijital ortama genişletme vizyonu ile önerildi. Akıllı sözleşmeler, araçların hizmetlerinden kaçınırken, mülk, hisse veya herhangi bir değeri şeffaf, anlaşmazlıklardan uzak bir şekilde paylaşmanıza yardımcı olur. Bu bölümde akıllı sözleşmelerden ve bir sözleşme oluşturmak için nasıl bir yol izlemeli ve neleri kullanmamız gerektiğini öğreneceğiz.

#### 2.3.1.1 Akıllı sözleşme nedir?

Akıllı sözleşmeler, belirli koşullar yerine getirildiğinde otomatik olarak yürüten kendi kendine çalışan bir bilgisayar programıdır. Akıllı sözleşmelerle, yabancılardan anlaşmazlık olmadan şeffaf bir şekilde, değerli her şeyi paylaşabilirsiniz.

Bu sözleşmeler, blok zinciri tabanlı bir satış makinesi olarak düşünülebilir. Akıllı sözleşmeler, önceden yapılandırılmış kurallara göre kod yürütmek için bir yakıt olarak eter kullanır.



Şekil 2.5 Akıllı Sözleşme Mantığı

Vitalik Buterin akıllı sözleşmeleri şu şekilde tanımlar; *program bu kodu çalıştırır ve bir noktada bir koşul otomatik olarak geçerliliğini belirler ve varlığın bir kişiye gitmesi gerekip gerekmediğini veya diğer kişiye geri dönüp gideceğini ya da bazı bileşimlerin derhal iade gerektirip gerektirmeyeceğini belirler.* Bu arada, merkezi olmayan yapı da belgeyi depolar, çoğaltır ve belgeye belirli bir güvenlik ve değişmezlik kazandırır. İşleri basit ve verimli hale getirmek için blok zinciri üzerinde bir sözleşme yazılarak birçok çözüm otomatik hale getirebilir. Akıllı sözleşmelerin avantajları şöyledir;

- Aracıları ortadan kaldırır.
- Aracılar olmadığı için maliyeti azaltır.

Sözleşmelerin birçok kullanım alanı vardır. Ethereum'un geliştiricilerinden Gavin Wood akıllı sözleşmelerin geleceğini şu sözler ile tanımlıyor. *Akıllı sözleşmelerin kendisinin potansiyeline gelince, sağlıktan otomobile, gayrimenkul ve hukuka kadar etkileyebileceği endüstri alanlarının sonu yoktur. Liste uzayıp gidiyor. Akıllı sözleşmelerin toplumun yönünü değiştirme potansiyeli büyük önem taşır. Bu, her türlü sosyal değişiklik için teknik bir temel sağlayacak*

*bir şey ve bunu heyecan verici buluyorum.*(Gavin Wood, Akıllı sözleşmelerin geleceği).

### 2.3.1.2 Solidity nedir?

Solidity, sözleşmeye dayalı, yüksek seviyeli bir programlama dilidir. JavaScript'e benzer bir söz dizimine sahiptir. Ethereum Sanal Makineyi (EVM) hedeflemek için tasarlanmıştır. Çalışma zamanının karşısı olarak derleme zamanındaki kısıtlamaları doğrulama ve zorlama işlemini gerçekleştiren statik olarak yazılan komut dosyası dilidir.

Solidity de bir sözleşme yazıldıktan sonra, EVM bayt koduna derlenir ve daha sonra belirli bir Ethereum adresinde çalıştırılır. Ancak, Ethereum'daki akıllı sözleşmeleri dağıtmak ve etkileşimde bulunmak için, bir web API.3'sinin yanında özel bir JavaScript RPC kütüphanesi kullanılır. Akıllı programlama sözleşmesi, Ethereum ve Solidity ile başladığı için, hala geliştirme aşamasındadır.

Modülerlik: Akıllı sözleşmeler mümkün olduğunca küçük ve basit tutulmalıdır. Yerel değişkenler ve fonksiyonların uzunluğu, sözleşmelerin mümkün olduğunca okunabilir olmasını sağlamak için sınırlandırılmalıdır. Sözleşmeler ne kadar modüler olursa, akıllı sözleşmeler sistemini geliştirmek o kadar kolay olur.

Kontroller-Etkiler: Fonksiyonlar algoritmanın ilk adımında ön koşul kontrolleri gerçekleştirmelidir. Daha sonra, ikinci adım olarak, durum değişkenine yapılan değişiklikler yapılmalıdır. Son olarak diğer sözleşmelerle etkileşimler meydana gelmelidir.

### 2.3.2 Ethereum ve Bitcoin arasındaki farklar

Günümüzün Bitcoin, Ethereum, kripto para ve blok zinciri teknolojilerini çevreliyor. Bu alanda geliştirilen çok fazla para var ve yakın zamanda da yavaşlayacak gibi görünmüyor. Ethereum ve Bitcoin arasındaki farkları şöyle sıralayabiliriz;

Bitcoin de bir bloğun zincire eklenme süresi 10 dakikadır, Ethereum da ise bu süre sadece 15 saniyedir. Bu da Ethereum da işlemlerin daha hızlı netileneceğini gösterir.

Bitcoin de madencilik işlemleri, madencinin bilgisayarının işlemci gücüne göre değişir. Ethereum'da ise ekran kartlarıyla eşitlikçi ASIC denilen bir sistemle üreticiler arasındaki dengenin korunması sağlanır

İki kripto para arasındaki temel fark, Ethereum'un programlanabilir olmasıdır. Blok Zinciri teknolojisi, üretilen paradan (bilgiden) daha fazlasını gerektirir. Bitcoin'in yazılımsal tabanı

değişikliklerin uygulanması için çok yavaş kalır. Birçok yatırımcı ilk kripto para birimi imajı yüzünden Bitcoin'e yönelmiştir.(Ethereum vs. Bitcoin, 2018)

Ethereum, her operasyona veya blok zincirindeki depolama kullanımına gas olarak bilinen bir maliyet atar. Bitcoin işlem maliyetleri, büyüklüklerine göre belirlenir.

### **3. KULLANILAN TEKNOLOJİ VE ARAÇLAR**

#### **3.1 Truffle**

Truffle, Ethereum için popüler bir test geliştirme çerçevesidir. Truffle, blok zinciri uygulamalarını hızla oluşturmak, derlemek, dağıtmak ve test etmek için bir ortam sağlar. Gelişimi kolaylaştırır. Hızlı geliştirme yapmak için sözleşmelerinizi otomatik olarak test eder.

Hem JavaScript hem de Solidity sözleşmeleriniz için otomatik testler yazıp, sözleşmeleri hızlı bir şekilde geliştirir.

### 3.2 Ganache

Truffle Suite'in bir parçası olan Ganache, sözleşmelerinizi ve işlemlerinizi öne ve merkeze yerleştirerek dapp geliştirmeyi basitleştirir. Ganache'yi kullanarak uygulamanın blok zincirini nasıl etkilediğini hızlı bir şekilde görülebilir ve hesaplanabilir, bakiyeler, sözleşme oluşturmaları ve gas maliyetleri gibi ayrıntılar anlaşılabilir. Ayrıca Ganache'nin gelişmiş madencilik kontrollerini ihtiyaçlarınıza daha iyi uyacak şekilde ayarlayabilir. Ganache, Windows, Mac ve Linux için indirebilir.

### 3.3 Web3.js

Web3, Blok zinciri ile etkileşime girmeye ve akıllı sözleşmelere yapılan çağrılar da içeren bir Javascript API'sidir. Bu API, geliştiricilerin uygulamalarının içeriğine odaklanmalarını sağlayan Ethereum İstemcileri ile iletişimi özetler.

### 3.4 Metamask

MetaMask, eğer bir Dapp ( merkezileştirilmemiş uygulama ) oluşturuyorsanız ve insanların kullanmasını istersiniz MetaMask desteği şarttır. Yani akıllı sözleşmelerle etkileşimde bulunabilmek için, derledikleri, dağıtıldıklarından ve istemci tarafı JavaScript'teki web3 ile etkileşimde bulunmalıyız. Bu küçük chrome uzantısı kullanıcıların Dapp ile ne kadar kolay etkileşime girdiğini büyük ölçüde geliştirir.

### 3.5 JavaScript

JavaScript, yaygın olarak web tarayıcılarında kullanılmakta olan bir betik dilidir. JavaScript ile yazılan istemci tarafı betikler sayesinde tarayıcının kullanıcıyla etkileşimde bulunması, tarayıcının kontrol edilmesi, asenkron bir şekilde sunucu ile iletişime geçilmesi ve web sayfası içeriğinin değiştirilmesi gibi işlevler sağlanır.(Geleceği Yazarlar, 2016).

### 3.6 NodeJS

Node.js, açık kaynaklı, sunu tarafında çalışan ve ağ bağlantılı uygulamalar için geliştirilmiş



bir çalışma ortamıdır. Node.js uygulamaları genelde istemci taraflı betik dili olan JavaScript kullanılarak geliştirilir. En önemli avantajı JavaScript'in sağladığı bloklamayan G/Ç imkânıyla yüksek ölçeklenebilirliği ve yüksek veri aktarabilmesidir. Bu teknolojiler sık sık gerçek zamanlı Web uygulamalarında tercih edilmekle beraber kullanım alanı popülaritesiyle orantılı olarak genişlemiştir.(Node.js, 2016).

### **3.7 NPM**

NPM yani Node Paket Yöneticisi (Node Packet Manager), Node.js üzerinde çalışan modüllere erişim olacağı sağlayan bir paket yöneticisidir. Npmjs.org adresinde bulunan modüllerin, geliştiricinin kendi uygulamasına ekleyebilmesini sağlar.

### **3.8 Geliştirme Araçları**

#### **3.8.1 Sublime text**

Sublime Text, içinde birçok programlama dili arayüzü barındıran, çapraz platform bir kaynak kod düzenleme ve metin editörüdür. Arayüzü Vim'den ilham alınarak tasarlanmıştır. Sublime Text açık kaynaklı ya da özgür bir yazılım değildir. Buna rağmen genişleme paketlerinin pek çoğu özgür yazılım lisansı ile dağıtılmakta ve Sublime Text kullanıcılarının oluşturduğu topluluk tarafından geliştirilmektedir.(Sublime Text, 2016).

#### **3.8.2 iTerm2**

iTerm2, macOS için bir GPL lisanslı terminal emülatörüdür. iTerm2, pencere saydamlığı, tam ekran modu, standart klavye kısayolları gibi işletim sistemi özelliklerini destekler. Diğer özellikler arasında özelleştirilebilir profiller ve geçmiş terminal giriş / çıkışını anında yeniden oynatma yer alır.(iTerm2, 2015).

### **3.9 Github**

Github, Git üzerine kurulu internet bazlı depolama sistemidir. Kullanıcıların Git depolarını uzak sunucuda tutabilmelerine sağlar. Versiyon kontrolünü, kaynak kodu yönetimi gibi özelliklerin yanı sıra, aynı zamanda statik internet sayfalarını yayınlamaya da olmak sağlar. Statik internet sayfalarını yayınlayabilmesi, Github tercih edilmesinin en büyük sebebidir.

#### 4. UYGULAMANIN GELİŞTİRİLMESİ

Bir blok zincirinde, dağıtılan işlemlerin (oyların) daimi bir kaydı olduğundan, her oy, seçmen kimliğini ortaya çıkarmadan tam olarak nerede ve ne zaman olduğu gibi geri dönülemez bir şekilde izlenebilir. Buna ek olarak, geçmiş oylar değiştirilemezken, saldırıya uğratılamaz. Çünkü her işlem ağdaki her düğüm tarafından doğrulanır. Ve herhangi bir dış veya iç saldırgan, kaydı değiştirmek için düğümlerin % 51'inin kontrolüne sahip olmalıdır. Saldırgan, gerçek kimlikleriyle oy kullanmayı başarmış olsa bile, uçtan uca oylama sistemleri, seçmenlerin sisteme doğru oyların girip girmediğini doğrulamasına izin vererek sistemi son derece güvenli hale getirebilir. Yani seçmenler oylarını kontrol edebildiği için sistem güvenlidir.

Bu bölümde yapılan uygulama kurulum aşamasından başlanarak detaylı bir şekilde anlatılacaktır. Bir kullanıcının kimliğini girebileceği ve bir aday için oy kullanabileceği ve seçim için aday ve seçim tarihinin eklenebileceği bir uygulama olacaktır. Ayrıca aday başına oy sayısını sayan buton olacaktır. Bu şekilde bir uygulamada akıllı sözleşmeler oluşturma ve etkileşim sürecine odaklanabileceğiz.

##### 4.1 Truffle Dizin Yapısı

Uygulamanın yapılmasına öncelikle truffle indirilerek başlanmalıdır. Çünkü truffle geliştiriciye boş bir proje yapısı imkanı verir. Truffle indirmek için Node.js ve Npm'in bilgisayarlarda yüklü olması gerekir.

- Truffle'ın kurulumu için gerekli komut Şekil 4.1'de gösterildiği gibi “npm install -g truffle”'dir.

```
nazlican@nazlicans-Air ~/Desktop$ npm install -g truffle
/Users/nazlican/.npm/versions/node/v7.7.2/bin/truffle -> /Users/nazlican/.npm/versions/node/v7.7.2/lib/node_modules/truffle/build/cli.bundled.js
/Users/nazlican/.npm/versions/node/v7.7.2/lib
└─ truffle@4.1.11
```

Şekil 4.1 Truffle'ın indirilmesi

Truffle komutlarını kullanmak için boş bir proje yapısının GitHub'dan indirilmesi gerekir. Şekil 4.2'deki komut çalıştırılarak iskelet yapıda bir proje indirilir. Bu depo sadece Truffle Box'ın iskeletidir. (Github, 2018).

```

nazlican@nazlicans-Air ~/Desktop/truffle$ git clone https://github.com/tko22/truffle-webpack-boilerplate
Cloning into 'truffle-webpack-boilerplate'...
remote: Counting objects: 32, done.
remote: Total 32 (delta 0), reused 0 (delta 0), pack-reused 32
Unpacking objects: 100% (32/32), done.

```

Şekil 4.2 Truffle için Boş Proje Yapısının İndirilmesi

Truffle indirildikten sonra dizin yapısı şunları içermelidir:

- contracts / - Tüm sözleşmeleri tutan klasör. Burada dikkat edilmesi gereken nokta Migrations.sol dosya silinmemelidir. Çünkü buradaki kodlar akıllı sözleşmemizi güncellememizi sağlayan kısımdır.
- migrations / - Akıllı sözleşmelerin Blok zincirine dağıtılmasına yardımcı olan dosyaları tutan klasördür.
- src / - uygulama için HTML / CSS ve Javascript dosyalarını tutar.
- truffle.js - Truffle yapılandırma dosyası
- build / - Sözleşmeleri derleyene kadar bu klasör ortaya çıkmaz. Bu klasör artifactleri tutar, bu nedenle bu dosyalardan herhangi biri değiştirilmemelidir. Build klasörü sözleşmenizin işlevini ve mimarisini tanımlar. Blok zincirinde akıllı sözleşmeler ile nasıl etkileşimde bulunacağı konusunda Truffle sözleşmeleri ve web3 bilgilerini verir.

#### 4.2 Akıllı Sözleşmenin Yazılması

Her türlü uygulama için, akıllı sözleşmel olabildiğince basit olmalıdır. Çünkü bölüm 2.3’de anlatıldığı gibi yapılan her hesaplama / işlem için ödeme(gas) yapılması gerekir. Akıllı sözleşmelerin Blok zincirinde sonsuza dek kalacağı unutulmamalıdır. Yani, ne kadar karmaşık olursa, bir hata yapmak o kadar kolay olur.

Akıllı sözleşme şunları içerecektir:

- Durum Değişkenleri: Blok zincirinde kalıcı olarak saklanan değerleri tutan değişkenlerdir. Liste değişkenleri ile aday ve adayların sayısını tutmak için durum değişkenlerini kullanacağız.
- Fonksiyonlar: Fonksiyonlar akıllı sözleşmelerin yürütücüleridir. Blok zinciri ile etkileşim kurmak için kullanılır ve iç ve dış olarak farklı görünürlük seviyelerine sahiptirler. Bir

değişkenin değerini / durumunu değiştirmek istediğinizde, bir işlemin gerçekleşmesi gerektiğini - maliyetin Ether olduğu unutulmamalıdır.

- Struct Tipleri: Bu C'deki struct yapısına çok benzer. Struct, birden fazla değişken tutmanıza olanak tanır ve birden çok özneliği olan şeyler için kullanılır. Uygulamada adaylar sadece isimlerini, partilerini ve oy sayılarını alacak, ancak onlara daha fazla nitelik eklenebilir.
- Mapping: Bunları anahtar / değer çiftine sahip olduğu C#'daki dictionary kavramına benzetebiliriz. Uygulamada iki mapping kullandık.

Burada listelenmeyen birkaç tür daha vardır. Ancak bu dört yapı akıllı bir sözleşmenin genellikle kullanacağı yapıların çoğunu kapsamaktadır.

```
pragma solidity ^0.4.18;

contract Voting {
    struct Candidate {
        uint id;
        string name;
        string party;
        uint voteCount;
    }
}
```

Şekil 4.3 Adayın Struct Yapısı

Temel olarak, “Candidate” isimli adayı tanımlayan bir Struct var. Struct burada birden fazla değişkene sahip olan türleri bir arada tutmak için kullanıldı.

```
1 mapping (uint => Candidate) public candidates;
2 mapping (address => bool) public voters;
```

Şekil 4.4 Mapping Yapısı

“candidates”(seçmenleri) ve “voters”(adayları) takip etmek için, onları tamsayı ve adres endeksli oldukları ayrı haritalara koyuldu. Aday veya Seçmen’in indeks / anahtar değeri ile onların kimliğini tanımlanabilir. Bu onlara erişecek fonksiyonların tek yoludur. Ayrıca, onları endekslememize yardımcı olacak olan “candidates” ve “voters” sayısını da takip ediyoruz. Bir kullanıcı oy verdiğinde, mapping yeni bir Voter(seçmen) yapısı oluşturulur ve eklenir.

```

function vote(uint candidateID) public {
    require((votingStart <= now) && (votingEnd > now));
    require(candidateID > 0 && candidateID <= countCandidates);
    //daha önce oy kullanmamış olmalı
    require(!voters[msg.sender]);
    voters[msg.sender] = true;
    candidates[candidateID].voteCount ++;
}

```

Şekil 4.5 Oy verme şartları

Şekil 4.5’te sırayla;

- Seçimin belirlenen tarihler arasında yapılması sağlanır,
- Geçerli bir aday olmalıdır
- Oy kullanan kişinin daha önce oy kullanmamış olması gerekir.
- Seçmenin oy durumu değiştirilir.
- Adayın oy sayısını güncellenir.

#### 4.3 Web3 ve sözleşmeleri örneklenmesi

Akıllı Sözleşme tamamlandığında, şimdi test ortamındaki blok zincirini çalıştırılmalı ve bu sözleşmeyi blok zincirine gönderilmesi(deploy) gerekir. Ayrıca sözleşmeyle iletişim kurulması için Web3.js kullanılmalıdır.

```

1 var Voting = artifacts.require("Voting")
2
3 module.exports = function(deployer) {
4     deployer.deploy(Voting)
5 }
6

```

Şekil 4.6 Blok zinciri ile Akıllı Sözleşmenin İletişim Kurulmasının Sağlanması

Test blok zincirine başlamadan önce, geçiş yaptığınızda Voting akıllı sözleşmesini dahil etmesini söyleyen klasör/contracts içinde 2\_deploy\_contracts.js adlı bir dosya oluşturulması gerekir. Bu kısım bitirildikten sonra Ethereum blok zincirini geliştirmeye başlamak için komut satırına gidip “compile” komutu çalıştırılır. Solidity, derlenmiş bir dil olduğu için

EVM'nin yürütülmesi için önce bytecode'a derlenmelidir.

```
truffle(develop)> compile
Compiling ./contracts/Migrations.sol...
```

Şekil 4.7 EVM için bytecode oluşturulması

Bu komut çalıştırdıktan sonra proje dizininde build/ klasörü oluşur. Bu klasör, Truffle'ın iç işleyişi için kritik olan artifactleri tutar, bu yüzden onlara bu klasöre dokunulmamalıdır.

Ardından sözleşmeyi taşımalıyız(migrate). Taşıma, uygulamanızın sözleşmesinin durumunun değiştirilmesine yardımcı olan bir Truffle komut dosyasıdır. Oluşturulan sözleşmen blok zincirindeki belirli bir adrese yerleştirilir, bu nedenle değişiklik yapıldığında sözleşme farklı bir adreste olacaktır. Taşıma(migrate) bunu yapmanıza yardımcı olur ve ayrıca verileri taşımanıza yardımcı olur.

```
saving artifacts...
nazlican@nazlicans-MacBook-Air ~/Desktop/nazlican-block/truffle-webpack-boilerplate % truffle deploy --reset --network development
Compiling ./contracts/Voting.sol...
Writing artifacts to ./build/contracts
Using network 'development'.
Running migration: 1_initial_migration.js
  Replacing Voting...
  ... 0x279bcd7eb6cace51d9c7aad97e843f60a6667466ff984e2be458041dff76a6fa
  Voting: 0x0d19254c6d53bc84149eef19c964c157b1f943e0
Saving artifacts...
```

Şekil 4.8 Yazılan sözleşmenin taşınması(migrate) işlemi

Yani Şekil 4.8'de gösterildiği üzere sözleşmede yapılan değişiklikler “migrate” komutu ile gönderilir. Bu komuttan sonra eğer test ortamında çalışmıyor ise(projede test ortamı kullanılıyor.) yazılan sözleşme sonsuza kadar blok zincirine eklenmiş olur.

Akıllı sözleşmeyi blok zincirine dağıttıktan(deploy) sonra, uygulama başladığında tarayıcıda bir web3.0 örneğini Javascript ile kurmamız gerekecek. Böylece, bir sonraki kod parçası js/app.js klasörünün altına yazılacaktır.

```
1
2 window.addEventListener("load", function() {
3   if (typeof web3 !== "undefined") {
4     console.warn("Using web3 detected from external source like Metamask")
5     window.web3 = new Web3(web3.currentProvider)
6   } else {
7     console.warn("No web3 detected. Falling back to http://localhost:9545. You should remove this fallback when you deploy live, as it's inherently insecure.")
8     window.web3 = new Web3(new Web3.providers.HttpProvider("http://127.0.0.1:9545"))
9   }
10  window.App.eventStart()
11 })
```

Şekil 4.9 Tarayıcıda Metamask olup olmadığının kontrolü

Şekil 4.9'deki kod parçası app.js dosyası içindedir ve iskelet projede kendiliğinden gelir. Uygulama başladığında tarayıcıda Metamask olup olmadığını kontrol eder. Eğer Metamask kullanmak tercih edilmiyorsa Truffle geliştirme blok zinciri olan localhost:9545'e çağrı yapılır. Bu uygulamada Ganache kullanıldığı için localhost:7545 olarak değiştirildi.

#### 4.4 Uygulamanın Arayüzünün Yazılması

##### 4.4.1 Html dosyalarının oluşturulması

Bu kısımda yapılması gereken, uygulama için arayüz yazmaktır. Bu, herhangi bir web uygulamasının — HTML, CSS ve Javascript'in (web3 örneğini oluşturarak çoktan Javascript yazıldı.) temellerini içerir. Öncelikle, HTML dosyaları oluşturulur.

```

1  </head>
2  <body>
3    <div id="head" class="text-center">
4      <h1>Ethereum Voting Dapp</h1>
5      <p> Welcome for Voting</p>
6      <p>Voting Dates: <span id="dates"></span></p>
7      <hr/>
8      <br/>
9    </div>
10   <div id="candidate" class="container">
11     <table class="table text-center">
12       <thead>
13         <tr>
14           <th>Name</th>
15           <th>Party</th>
16           <th>Total Vote</th>
17         </tr>
18       </thead>
19       <tbody id="boxCandidate"></tbody>
20     </table>
21     <hr/>
22     <br/>
23   </div>
24   <div id="vote" class="container text-center">
25     <p>Please select one of the candidates and click the vote button.</p>
26     <button id="voteButton" class="btn btn-primary" onclick="App.vote()" disabled>Vote</button>
27     <div id="msg"></div>
28     <hr/>
29     <div id="vote-box"></div>
30   </div>
31   <hr/>
32   <div id="account">
33     <p id="accountAddress" class="text-center"></p>
34   </div>

```

Şekil 4.10 Oy kullanma sayfası

Uygulamada iki tane Html dosyası bulunmaktadır. Şekil 4.10 aday seçip oy verme işlemini görülebildiği basit bir sayfadır.

```

1  </head>
2  <body>
3    <!--Başlık -->
4    <div id="head" class="text-center">
5      <h1>Ethereum Voting Dapp</h1>
6      <hr/>
7      <br/>
8    </div>
9
10   <div class="container">
11     <legend>Add Candidate</legend>
12     <table class="table text-center">
13       <tr>
14         <th>Name </th><td> <input id="name" type="text" name="name" placeholder ="Candidates name"></td>
15         <th>Party</th><td><input id="party" type="text" name="party" placeholder ="Candidates party"></td>
16       </tr>
17     </table>
18     <input id="addCandidate" type="submit" name="submit" value="Add Candidate">
19     <p id="Aday"></p>
20   </div>
21
22   <div class="container">
23     <legend>Define Voting Dates</legend>
24     <table class="table text-center">
25       <tr>
26         <th>Start date</th><td><input id="startDate" type="date" name=""></td>
27         <th>End date</th><td> <input id="endDate" type="date" name="" ></td>
28       </tr>
29     </table>
30     <input id="addDate" type="submit" name="submit" value="Define Dates">
31     <p id="Aday"></p>
32   </div>

```

Şekil 4.11 Admin portal

Şekil 4.11’da gösterilen Html sayfasında seçim için aday ve tarih eklenir.Bu da oldukça basit bir Html sayfasıdır.

#### 4.4.2 JavaScript dosyasının oluşturulması

Şimdi, sadece Sözleşme ile etkileşimde bulunmalı ve oylama ve her aday için oy sayısını sayma işlevlerini yerine getirmeliyiz.

İlk olarak, web3 ve Truffle Contracts dahil olmak üzere gerekli kütüphaneleri ve web paketlerini içeri aktarıyoruz. Blok zinciri ile etkileşimde bulunmak için web3’ün üzerine inşa edilen Truffle Sözleşmelerini kullanacağız.

```

1  import "../css/style.css"
2
3  import { default as Web3} from "web3"
4  import { default as contract } from "truffle-contract"
5
6  import votingArtifacts from "../../build/contracts/Voting.json"
7

```

Şekil 4.12 Web3 ve Truffle contract kütüphanelerinin projeye dahil edilmesi

Blok zinciri ile gerçekten etkileşimde bulunmak için “deployed” işlevi kullanarak Truffle Sözleşmesinin bir örneğini oluşturmalıyız. Bu, sırayla, akıllı sözleşmeden fonksiyonları



çağırarak için kullanacağınız dönüş değeri olarak örnekle temin edilecektir. Bu fonksiyonlarla etkileşim kurmanın iki yolu vardır: işlemler(transaction) ve çağrılar(call). Bir işlem, bir yazım işlemidir ve tüm ağa yayınlanacak ve madenciler tarafından işlenecektir (ve böylece, EterE mal olur.). Durum değişkenini değiştiriyorsanız, blok taşının durumunu değiştireceğinden bir işlem yapmanız gerekir. Bir çağrı, bir işlemi simüle eden, ancak durumdaki değişikliği iptal eden bir okuma işlemidir. Böylece, Eter'e mal olmayacaktır. Bu, “getter” işlevlerini çağırarak için çok uygundur. Uygulamadaki akıllı sözleşmede üç adet “getter” fonksiyonu bulunmaktadır.

Truffle Contract ile bir işlem yapmak için, örnek olarak, gönderilmiş fonksiyon tarafından döndürülen örnekle instance.functionName (param1, param2) yazarsınız. Bu işlem, işlem verileriyle dönüş değeri olarak bir temin edilir. Böylece, akıllı sözleşme fonksiyonu bir değer dönerse, ancak aynı fonksiyona sahip bir işlem gerçekleştirirseniz, bu değeri döndürmez.

```

13 VotingContract.deployed().then(function(instance){
14   instance.getCountCandidates().then(function(countCandidates){
15

```

Şekil 4.13 Akıllı sözleşmedeki fonksiyonlarla etkileşim kurulması

Javascript dosyasında 2 adet fonksiyon bulunmaktadır. Bu fonksiyonların görevleri şöyledir:

- App.eventStart(): Bu fonksiyon, bir web3 örneğini oluşturduktan hemen sonra çağrılır. Truffle Contract'ın işe yaraması için, sağlayıcıyı oluşturulmuş web3 örneğine ayarlamalı ve varsayılanları ayarlanmalıdır. (kullanılan hesap ve işlem yapmak için ödemek istediğiniz gas miktarı gibi).

```

window.App = {
  eventStart: function() {
    VotingContract.setProvider(window.web3.currentProvider)
    VotingContract.defaults({from: window.web3.eth.accounts[0], gas: 6654755})
  }
}

```

Şekil 4.14 Web3 örneğinin oluşturulması ve gas miktarının belirlenmesi

Geliştirme modunda olduğumuzdan, herhangi bir miktarda gas ve herhangi bir hesap kullanabiliriz. Üretim sırasında, MetaMask tarafından sağlanan hesabı alır ve gerçek paraya sahip olacağından kullanabileceğiniz en küçük miktarda gas'ı bulmaya çalışırız.

Her şey hazır olduğunda, şimdi her bir adayın oy kullanabilmesi için onay kutularını

göstereceğiz. Bunu yapmak için sözleşmenin bir örneğini oluşturmali ve “Candidate” bilgilerini almalıyız. Herhangi bir aday yoksa, onları oluşturacağız. Bir kullanıcının bir aday için oy kullanabilmesi için, belirli bir adayın kimliğini(ad soyadı parti adı) belirtmek gerekir.

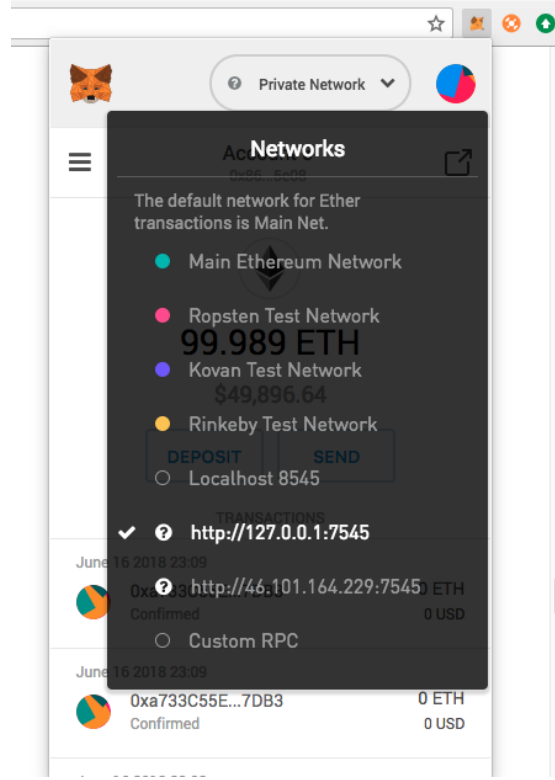
- App.vote(): Bu işlev, hangi onay kutusunu tıkladığına ve kimlik özniteliğine bağlı olarak belirli bir aday için oy kullanacaktır. Birincisi, kullanıcının kendi kullanıcı kimliğini girip girmediğini, onların kimliğini belirleyip belirlemediğini kontrol edilir. Eğer yapmazlarsa, bunu yapmalarını isteyen bir mesaj gösterilir. İkincisi, kullanıcının bir aday için oy kullanıp kullanmadığını kontrol edip, tıklanan en az bir onay kutusu olup olmadığını kontrol edilir. Onay kutularının hiçbiri tıklanmadıysa, bir aday için oy kullanmalarını belirten bir mesaj görüntülenir. Onay kutularından biri tıklanırsa, söz konusu onay kutusunun id niteliği alınır. Bu da adayın kimliğine bağlıdır ve aday için oy kullanmak için bu kullanılır. İşlem tamamlandıktan sonra, “Oy Verilmiş” mesajı görüntülenir.

Ethereum üzerinde oluşturulan uygulamalar, bir arka servis hizmeti çağrısı yapan düzenli bir uygulamaya oldukça benzer. Buradaki en zor kısım sağlam ve eksiksiz bir akıllı sözleşme yazmaktır.

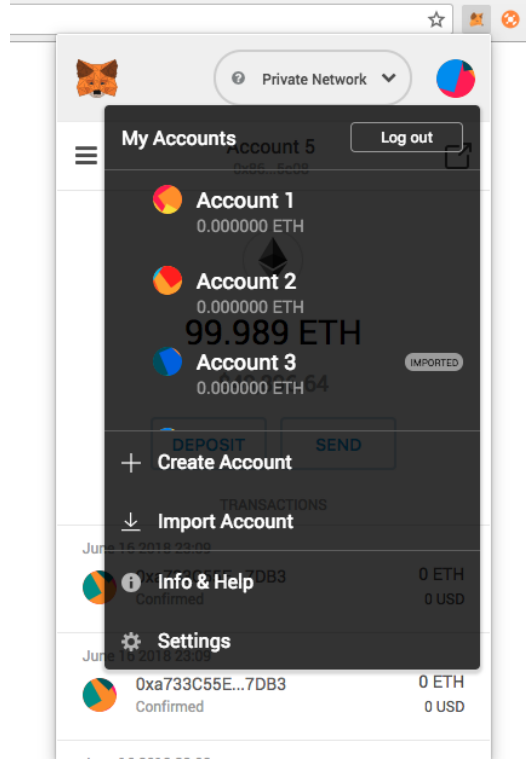
#### 4.5 Metamask Üzerinde Hesap Oluşturma

Bölüm 3.4’de Metamask’ın ne olduğu anlatılmıştır. Metamask en basit anlamı ile bu uygulamada oy veren kişinin cüzdanı konumundadır. Uygulamayı tarayıcıda görüntülemek için öncelikle projenin bulunduğu dizine gidip “npm run dev” komutu çalıştırılır. Bu komut projedeki Html, JavaScript dosyalarını derlemeye yarar. Bu komut başarılı bir şekilde çalıştırıldıktan sonra <http://localhost:8080/> adresinden proje görüntülenebilir. Seçmenlerin oy kullanabilmesi için öncelikle Metamask eklentisinin chrome üzerinde kurulu olması gerekir ve seçmenlerin burada hesap oluşturmaları gerekir.

Öncelikle Metamask kurulumundan sonra Ethereum ağına adres bilgileri girilmelidir.

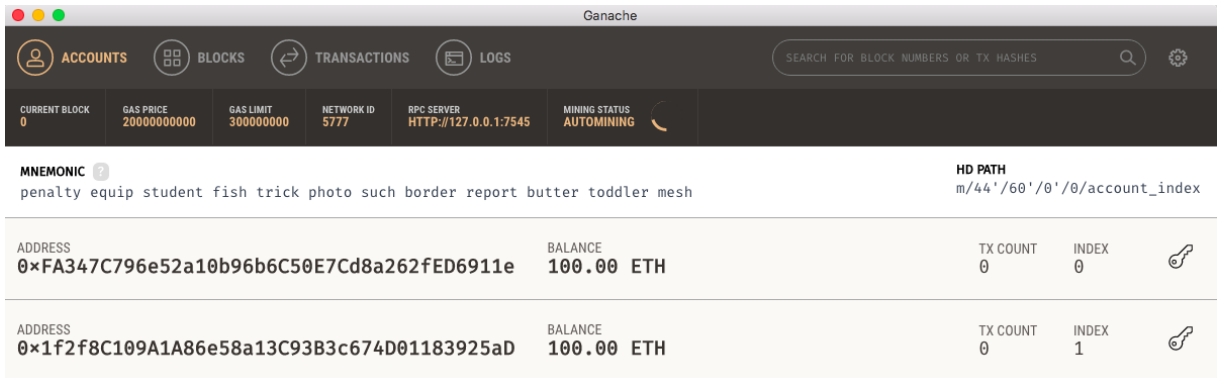


Şekil 4.15 Metamask adres bilgilerinin girilmesi



Şekil 4.16 Metamask'a hesap eklenmesi

Şekil 4.16'te gösterildiği gibi Ethereum ağı için hesap eklenmesi gerekiyor. Bu işlemi de Şekil 4.17'da uygulamanın test edilebilmesi için Ganache tarafından verilen test hesapları sayesinde yapılır. Ekleme işlemi için Şekil 4.16'de "import account" kısmına tıklayıp private key girilmesi gerekiyor. Böylece oylama web sayfası hiç bir zaman sizin kimliğinizden haberdar olmuyor.



Şekil 4.17 Ganache test hesapları

## 5. KURULUM

### 5.1 Yerel Sunucuda Kurulum

Bu çalışmada anlatılan uygulamayı yerel sunucuda çalıştırmak için öncelikle aşağıda belirtilen yazılımların yüklü olduğu varsayılmıştır.

Truffle 4.1.11

NodeJs 7.7.2

NPM 4.1.2

Git 2.13.1

Öncelikle komut satırı üzerinden aşağıdaki kodlar çalıştırılır.

```
nazlican@nazlican-Air ~/Desktop > git clone git@github.com:naz46/blockchain-voting-dapp.git
Cloning into 'blockchain-voting-dapp'...
Enter passphrase for key '/Users/nazlican/.ssh/id_rsa':
remote: Counting objects: 21, done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 21 (delta 1), reused 21 (delta 1), pack-reused 0
Receiving objects: 100% (21/21), 5.92 KiB | 0 bytes/s, done.
Resolving deltas: 100% (1/1), done.
```

Şekil 5.1 Git üzerinden projenin kopyalanması

Git üzerinden klonlama işlemi tamamlandığında oluşan klasörün içine girilerek komut satırından “npm install” komutu çalıştırılır. Bu komut sunucu kodlarının çalışması için gerekli olan package.json dosyasında belirtilmiş JS kütüphanelerini indirir.

Ana dizinde bulunan “truffle.js” dosyasında ayar yapmak gerekir. Uygulamada Ganache kullanıldığı için onun port numarası ile gelir. Eğer başka bir ortam kullanılacaksa onun portu yazılarak değiştirilmelidir.

Sunucuyu çalıştırmadan önce yeni bir konsol penceresi açıp projede src/ klasörüne gidip aşağıda gösterildiği gibi “npm run dev” komutu çalıştırılmalıdır.

```

webpack: Compiled successfully.
^C
nazlican@nazlicans-MacBook-Air ~/Desktop/nazlican-block/truffle-webpack-boilerplate/src master • ? npm run dev
> ethereum-voting-dapp@1.0.0 dev /Users/nazlican/Desktop/nazlican-block/truffle-webpack-boilerplate
> webpack-dev-server

Project is running at http://localhost:8080/
webpack output is served from /
Hash: fc3e1ef2152d22e93656
Version: webpack 3.11.0
Time: 2723ms

```

Şekil 5.2 Sunucunun çalıştırılması

Diğer konsol penceresinde ise akıllı sözleşmelerin sunucuya gönderilmesi(deploy) için aşağıdaki komut çalıştırılır.

```

saving artifacts...
nazlican@nazlicans-MacBook-Air ~/Desktop/nazlican-block/truffle-webpack-boilerplate master • ? truffle deploy --reset --network development
Compiling ./contracts/Voting.sol...
Writing artifacts to ./build/contracts

Using network 'development'.

Running migration: 1_initial_migration.js
  Replacing Voting...
    ... 0x279bcd7eb6cace51d9c7aad97e843f60a6667466ff984e2be458041dff76a6fa
  Voting: 0x0d19254c6d53bc84149eef19c964c157b1f943e0
Saving artifacts...

```

Şekil 5.3 Yazılan sözleşmenin taşınması(migrate) işlemi

Sunucu ayarları tamamlandıktan sonra tarayıcıda görüntülenebilir. Bunun için <http://localhost:8080/> adresinden erişmek mümkündür. Bu adrese erişildiğinde oylama sayfası ile karşılaşılır.

**Ethereum Voting Dapp**  
Welcome for Voting  
Voting Dates: Sat Jun 16 2018 - Sun Jun 17 2018

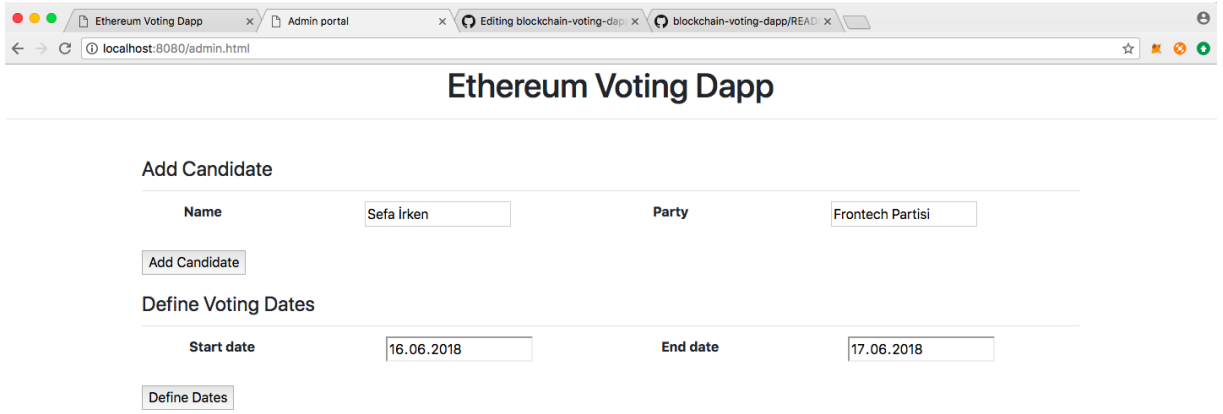
Name	Party	Total Vote
<input type="checkbox"/> Nazlıcan Öztürk	Yalova Üniversitesi Partisi	0
<input type="checkbox"/> Sefa İrken	Frontech Partisi	1

Please select one of the candidates and click the vote button.

[Vote](#)

Your Account: 0x8686a8b990d1f15c0810477c1e92a90c8c165e08

Şekil 5.4 Sunucuya tarayıcı üzerinden oy verme sayfasına erişim



The screenshot shows a web browser window with the title "Ethereum Voting Dapp". The address bar shows "localhost:8080/admin.html". The page has a header with the title "Ethereum Voting Dapp". Below the header, there are two main sections: "Add Candidate" and "Define Voting Dates".

**Add Candidate**

Name	Party
Sefa İrken	Frontech Partisi

**Define Voting Dates**

Start date	End date
16.06.2018	17.06.2018

Şekil 5. 5 Sunucuya tarayıcı üzerinden admin portalına erişim

## 6. SONUÇ

Bu tezde, tamamen merkezi olmayan bir şekilde elektronik oylama planı işlevi görecektir bir uygulama yapılmıştır. Oy veren kişinin kimliğini doğrulamak, atılan oyların güvenli bir şekilde gerekli kurumlara iletilmesi ve kullanılan oyların takibini sağlayarak seçimi kazananı belirleyecek şeffaflığı ve doğruluğu sağlamakta güvenli bir uçtan uca(peer to peer) ağı erişmek için, Solidity programlama dilinde geliştirilmiş akıllı sözleşme sistemi ile birlikte izin verilen bir blok zinciri mimarisi ve ortak kriptografik araçlar önerilmiştir ve kullanılmıştır. Ortaya çıkan uygulama gerekli şekillerde test edilip ve bir oylama sisteminin önemli tüm gerekliliklerini sağladığı görülmüştür.

**AS1: Güvenilir oylama nasıl elde edilir?**

**AS2: Seçim ve benzeri oylamaların maliyetleri nasıl azaltılır?**

**AS3: Blok zinciri teknolojisi ile hangi uygulamalar geliştirildi?**

Araştırma sorusu 1'in cevabı Bölüm 2.2.2'de açıklanmıştır. Araştırma sorusu 3'ün cevabı ise Bölüm 1.3 ve 2.2.3'te blok zinciri teknolojisinin farklı uygulamaları ile ilgili araştırma sonuçları verilmiştir. Araştırma sorusu 3 gelecek olunursa seçimler tamamen dijital bir ortamda yapıldığı için seçim için oy pusulalarının basılmasına, oy sandıklarının kurulmasına vb. gibi olaylara gereği ortadan kaldırmaktadır.

### 6.1 Uygulamayı Diğer Alanlara Genişletme

Bu tezin amacı, oylama sistemi gibi çok spesifik bir sorunu çözmektir. Diğer alanlarda olabilecek olası çapraz etkilerin farkına varamamaktadır. Bu, kullanıcıların bilgileri kayıt altına almasına ve son derece kontrollü bir şekilde farklı ortaklarla paylaşmasına izin veren bir uygulamadır. Uygulamayı normal merkezi bir sunucudan farklı kılan şey, değişmez izlenebilirliğin olması ve daha da önemlisi, müşterilerin kişisel bilgilerin şirket tarafından kullanımını sorgulama gücüne ve hakkına sahip olabilmeleridir. Bilginin satılması bugün çoğu sistemde olduğundan çok daha açık bir seviyede mutabakatlı olmalıydı. Diğer bir uygulama alanı, bilgi paylaşımının zorunlu olduğu ancak gizli veya hassas verilerle uğraşırken çok zor olduğu büyük bir şirket ortamında olabilir.



## KAYNAKLAR

- Alfred J. Menezes, S. A. V., Paul C. van Oorschot. (1996). Handbook of applied cryptography (5th ed.). CRC Press. Retrieved from [cacr.uwaterloo.ca/hac](http://cacr.uwaterloo.ca/hac)
- Buterin, V. (2015). Visions, Part 1: The Value of Blockchain Technology. <https://blog.ethereum.org/2015/04/13/visions-part-1-the-value-of-blockchain-technology/>
- Çağlar O., (2018), Sektörlere etkisi ve fırsatlar. <http://obcaglar.com/tag/decentralized/>
- Ethereum vs. Bitcoin, (2018). Ethereum ve Bitcoin arasındaki farklar. <http://www.webtekno.com/ethereum-ile-bitcoin-arasindaki-farklar-h38580.html>
- Geleceği Yazanlar, (2016). JavaScript Nedir? <https://gelecegiyazanlar.turkcell.com.tr/konu/web-programlama/egitim/301-javascript/javascript-nedir>
- iTerm2, (2015) "Features". iTerm2.com. Retrieved 2015-08-26.
- Merkle Ağaçları, Wikipedi. [tr.wikipedia.org/wiki/Merkle\\_ağacı](http://tr.wikipedia.org/wiki/Merkle_ağacı)
- GitHub, (2018) Boilerplate. <https://github.com/tko22/truffle-webpack-boilerplate>
- Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Online Multimedia. Retrieved from <https://bitcoin.org/bitcoin.pdf>
- Node.js, (2016). <https://tr.wikipedia.org/wiki/Node.js>
- Tian, F., (2016), “An agri-food supply chain traceability system for China based on RFID & blockchain technology”. Service Systems and Service Management (ICSSSM), 13th International Conference on. IEEE, 2016.
- Sublime Text, (2016). [https://tr.wikipedia.org/wiki/Sublime\\_Text](https://tr.wikipedia.org/wiki/Sublime_Text)
- Usta, A., Doğantekin, S., (2017), “Blockchain 101”. Kapital Medya Hizmetleri A.Ş., ISBN: 978-605-4584-97-0, İstanbul
- Wood G. (2017), Akıllı sözleşmelerin geleceği. [https://bitcoinlerim.com/smart-contracts-akilli-sozlesmeler-nedir/#Akilli\\_Sozlesmeler\\_harika](https://bitcoinlerim.com/smart-contracts-akilli-sozlesmeler-nedir/#Akilli_Sozlesmeler_harika)









**YALOVA ÜNİVERSİTESİ  
MÜHENDİSLİK FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**- BİTİRME TEZİ -**

**BLOK ZİNCİRİ TEKNOLOJİSİ ÜZERİNDE  
AKILLI SÖZLEŞMELER İLE OYLAMA  
SİSTEMİ**

**Nazlıcan ÖZTÜRK**

**Bitirme Tezi Danışmanı: Öğr. Gör. Dr. Yunus ÖZEN**

**YALOVA, 2018**

**YALOVA ÜNİVERSİTESİ  
MÜHENDİSLİK FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

# **BLOK ZİNCİRİ TEKNOLOJİSİ ÜZERİNDE AKILLI SÖZLEŞMELER İLE OYLAMA SİSTEMİ**

**Nazlıcan ÖZTÜRK  
140101020**

**1. Bitirme Tezi Danışmanı :Öğr. Gör. Dr. Yunus ÖZEN**

**2. Jüri Üyesi**

**:Dr. Öğr. Üyesi Osman Hilmi KOÇAL**

**3. Jüri Üyesi**

**:Dr. Öğr. Üyesi Adem TUNCER**

**Bitirme Tezinin Dönemi: 2017 – 2018 Bahar Yarıyılı**