

KOCAELİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

LİSANS TEZİ

BLOK ZİNCİRİ İLE BAĞIŞ TAKİP SİSTEMİ

TAHA UZ

KOCAELİ 2023

KOCAELİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ

BLOK ZİNCİRİ İLE BAĞIŞ TAKİP SİSTEMİ

TAHA UZ

Prof.Dr. Ahmet Sayar
Danışman, Kocaeli Üniv.

.....

Doç.Dr. Pınar ONAY DURDU
Jüri Üyesi, Kocaeli Üniv.

.....

Arş. Gör. Yılmaz DİKİLİTAŞ
Jüri Üyesi, Kocaeli Üniv.

.....

Tezin Savunulduğu Tarih: 06.06.2023

ÖNSÖZ VE TEŞEKKÜR

Bu tez çalışması, blok zinciri ile bağış takip sistemi oluşturularak sahteciliği önlemek ve güvenliği sağlamak amacıyla gerçekleştirilmiştir.

Tez çalışmamda desteğini esirgemeyen, çalışmalarına yön veren, bana güvenen ve yüreklendiren danışmanım Prof. Dr. Ahmet Sayar'a sonsuz teşekkürlerimi sunarım.

Tez çalışmamın tüm aşamalarında bilgi ve destekleriyle katkıda bulunan arkadaşım Rana Dudu Kabak'a teşekkür ediyorum.

Hayatım boyunca bana güç veren en büyük destekçilerim, her aşamada sıkıntılarımı ve mutluluklarımı paylaşan sevgili aileme teşekkürlerimi sunarım.

Haziran – 2023

Taha UZ

Bu dokümandaki tüm bilgiler, etik ve akademik kurallar çerçevesinde elde edilip sunulmuştur. Ayrıca yine bu kurallar çerçevesinde kendime ait olmayan ve kendimin üretmediği ve başka kaynaklardan elde edilen bilgiler ve materyaller (text, resim, şekil, tablo vb.) gerekli şekilde referans edilmiş ve dokümanda belirtilmiştir.

Öğrenci No: 190202013

Adı Soyadı: Taha UZ

İmza:.....

İÇİNDEKİLER

ÖNSÖZ VE TEŞEKKÜR	i
İÇİNDEKİLER	ii
ŞEKİLLER DİZİNİ	iii
TABLolar DİZİNİ	iv
SİMGELER VE KISALTMALAR DİZİNİ	v
ÖZET	vi
ABSTRACT	vii
1.GİRİŞ	1
2. BLOK ZİNCİRİ VE KULLANILACAK DİĞER TEKNOLOJİLER	3
2.1. Blok Zinciri Teknolojisi	3
2.1.1. Blok zinciri teknolojisinin avantajları	3
2.1.2. Blok zinciri teknolojisinin dezavantajları	4
2.1.3. Blok zinciri çeşitleri	4
2.1.4. Blok zinciri teknolojisinde gaz kavramı	5
2.2. Kullanılan Uzlaş Algoritması	6
2.3. React.js	8
2.4. Solidity	9
2.5. Remix IDE	10
2.6. EtherScan ve BscScan	11
3. LİTERATÜR TARAMASI	11
4. AKILLI KONTRATLAR İLE GENEL SİSTEM TASARIMI	18
4.1. Akıllı Kontratlar İçerisindeki Fonksiyonlar ve Kullanım Amaçları	20
4.1.1. Mapping fonksiyonları	20
4.1.2. Genel fonksiyonlar	21
5. BAĞIŞ TAKİP SİSTEMİ	23
5.1. Bağış Toplama ve Para Takibi Amaçlı Merkeziyetsiz Uygulama	24
5.1.1. Kripto cüzdan bağlantısı	28
5.1.2. Web arayüzünün akıllı sözleşme ile etkileşimi	30
5.2. Akıllı Kontrat Üzerinde Takip Sistemi İçin Oluşturulacak Algoritmanın Yapısı	32
5.3. Takip Sistemi Arayüzü	40
6. SONUÇLAR	42
KAYNAKLAR	44
ÖZGEÇMİŞ	46

ŞEKİLLER DİZİNİ

Şekil 2.1. PoS Algoritması	8
Şekil 3.1. Sistem Tasarımı	12
Şekil 3.2. Sistem Mimarisi	13
Şekil 3.3. Kullanıcı Transfer Geçmişi	13
Şekil 3.4. Akış Diyagramı	14
Şekil 3.5. Bağış Talebi	14
Şekil 3.6. Hesap Adresi Dönüştürme İşlemi	15
Şekil 4.1. Genel Sistem Mimarisi	20
Şekil 5.1 Transfer Geçmişi Takip Tasarımı	24
Şekil 5.2. PizzaHut Sipariş Formu	25
Şekil 5.3. Web Altyapısının Gelişimi	27
Şekil 5.4 App ile dApp Farkı	27
Şekil 5.5. Fiş Sistemi Altyapısı	40
Şekil 5.6. Kullanıcı Hesap Bilgisi	41
Şekil 5.7. Kullanıcı Transfer Geçmişi	42
Şekil 5.8. Tüm Vakıflara Ait Bağış Geçmişi	42

BLOK ZİNCİRİ İLE BAĞIŞ TAKİP SİSTEMİ

ÖZET

Günümüzde deprem, sel, orman yangınları ve gıda yardımı gibi nedenler altında para toplayan kurum, vakıf ve devlet gibi organlara aktarılan paraların takibi yapılmamasından veya görevli kişiler tarafından kontrolünün yapılmasından kaynaklı bazı insani faktörlerin devreye girmesiyle oluşan olumsuz koşullardan dolayı parayı doğru kişilere aktarılmaz.

Bu çalışmada blok zincirinin merkeziyetsiz olması yani diğer 3. parti kuruluşları ortadan kaldırarak direkt bağış ve bağışçı arasında bir para transferi gerçekleşmesi ile oluşan para aktarımları ve takibinin şeffaf olmasıyla, bu süreci sağlıklı bir biçimde gerçekleştirilmesi amaçlamaktadır. Blok zinciri üzerindeki akıllı kontrat ile yaptığımız bu sistem, bağışçılar bağışlarını bir kişiye değil, merkeziyetsiz bir sisteme sahip olan akıllı kontrat içerisine yapmaktadır ve bu sistemi şeffaf bir biçimde ağ üzerinden doğrulanıp herkese açık bir biçimde yayınlanmaktadır. Kontrat içerisinde toplanan para, akıllı kontrat içerisinde oluşturulan fiş mimarisi ile bağışlar insan faktöründen etkilenmeden ihtiyacı olan kişilere aktarılmaktadır. Bağış işlemini fiş içerisindeki işaretleme algoritması sayesinde sistem içerisine son kullanıcıya kadar takibini gerçekleştirebilmektedir. Kullanıcılar merkeziyetsiz web uygulamasından yararlanarak bağış ve bağış takip işlemlerini gerçekleştirir.

Anahtar kelimeler: Akıllı Kontrat, Bağış Sistemi, Blok Zinciri Ağı, Ethereum, Binance Smart Chain, Testnet, Metamask, Gaz, Solidity, Slippage, DApp, Ethers, Kontrat ABI, Bscscan, EtherScan, Remix IDE, Wagmi, Bağış

DONATION TRACKING SYSTEM WITH BLOCKCHAIN

ABSTRACT

Today, the money is not transferred to the right people due to the negative conditions that arise due to the fact that the funds transferred to the bodies such as institutions, foundations and the state that collect money for reasons such as earthquakes, floods, forest fires and food aid are not followed up or are controlled by the people in charge.

In this study, it is aimed to carry out this process in a healthy way by the fact that the blockchain is decentralized, that is, by eliminating other 3rd party organizations and by realizing a money transfer between the donor and the donor, the money transfers and follow-up are transparent. In this system, which we have made with the smart contract on the blockchain, the donors don't make their donations to a person, but to the smart contract, which has a decentralized system, and this system is transparently verified over the network and published publicly. The money collected in the contract and donations are transferred to the people in need without being affected by the human factor, with the voucher architecture created within the smart contract. Thanks to the marking algorithm in the receipt, the donation process can be followed up to the end user in the system. By using the decentralized web application, users perform donations and donation tracking processes.

Keywords: Smart Contract, Donation System, Blockchain Network, Ethereum, Binance Smart Chain, Testnet, Metamask, Gas, Solidity, Slippage, DApp, Ethers, Contract ABI, BSCScan, EtherScan, Remix IDE, Wagmi, Donation

1. GİRİŞ

İnternet teknolojisinin gelişmesinin etkilerinden biri de hayırseverliğin daha açık ve şeffaf hale gelmesidir. Durum bu hale gelmeden önce yardım toplama, bağış yapma gibi konularda insanlar belki daha önce başlarına geldiği için, bazen ise çevrelerinden gördüklerine dayanarak verdikleri bağışların tam olarak ihtiyaç sahiplerine gidip gitmediğinden emin olamadıkları için çekimser davranabilmektelerdi. Para aktarma işlemi gerçekleşmeden önce onay süreci şu şekilde gerçekleşir: Evinizi satmaya çalışıyorsanız ancak yardıma ihtiyacınız varsa emlakçıya gidip yardım isteyebilirsiniz. Evi küçük bir ücret karşılığında kendi müşterilerine sunmayı kabul edeceklerdir. Bu şekilde, bir seferde sadece bir kişiye teklif etmekten çok daha fazla potansiyel alıcı evinizi görebilecektir. Ancak bunlardan herhangi biri yapılmadan önce emlakçıya evinizi satması için izin vermeniz ve satmak istediğiniz evin gerçekten sahibi olduğunuzu kanıtlamanız gerekir. Mülkiyet belgeniz istenir ve ardından taraflar bir sözleşme imzalar. Bu son kısım, tıpkı onay süreci gibidir, çünkü emlakçı, alıcı ve satıcı arasında, bir akıllı sözleşmeye benzer şekilde, bir aracı görevi görür ve varlığınızı satmak için izin vermeniz gerekir. Akıllı kontratın içerisinde kullanıcının onayı (approve ve allowances) az önce anlatılan örnekteki gibi gerçekleşmektedir. React projemizin içerisinde web uygulamaları blok zinciri ağına bağlantısını sağlayan kütüphanelerden Rainbow Kit ve Wagmi modülleri yardımı ile kripto cüzdanlarını akıllı kontratımızı ile erişim sağlanması amaçlanan web arayüzüne entegre edilmektedir. Daha sonra kontrat ile web bağlantısı gerçekleştirilmesi için sistemde kullanılacak tüm akıllı kontratların adresleri ve ABI bilgileri alınması gerekecektir. ABI, tıpkı API'nin yaptığı gibi, ancak daha düşük bir düzeyde, ikili sözleşmeyle etkileşime geçmek için kullanılan yöntemleri ve yapıları tanımlar. ABI, işlev imzaları ve değişken bildirimleri gibi gerekli bilgileri, EVM'nin bu işlevi bayt kodunda çağırmak için anlayabileceği bir biçimde kodlaması için işlevi çağırını belirtir; buna ABI kodlaması denir. ABI kodlaması çoğunlukla otomatiktir ve REMIX gibi derleyiciler veya blok zinciriyle etkileşime giren cüzdanlar tarafından halledilir. Sözleşme ABI,

JSON biçiminde temsil edilir. Bu şekilde kontrat içerisindeki fonksiyonlara erişim sağlandıktan sonra para transferi işlemine geçilir. Sistem üzerindeki transfer fonksiyonları, kullanıcıların kripto para transferlerini gerçekleştirmelerini sağlar. İki tür transfer fonksiyonu bulunmaktadır.

Bağışçı ve vakıf arasında transfer: Kullanıcılar, belirli vakıf adreslerine kripto para transferi yapabilir ve transfer işlemi sırasında bağış yapacakları vakfa yüzdelik oranı belirleyebilir. Transfer işlemi, kullanıcının kimlik bilgisi ve kayıt kodu kontrolüyle gerçekleştirilir. Transfer bilgileri kaydedilerek fiş benzeri bir yapı oluşturulur.

Vakıf ve ticari firma arasındaki transfer: Bu fonksiyon, vakıfların ticari firmalardan alacakları eşyaları ihtiyaç sahiplerine ulaştırmak için kullanılır. Bağışçıların yaptığı bağışlar vakıfların cüzdanlarına aktarılmaz. Transfer işlemi, fişler aracılığıyla gerçekleştirilir. Fişlerdeki bağış miktarları, genel harcanacak miktar ile eşitlenene kadar dolaşır. Bu miktar eşitlendikten sonra fişlerin üzerine bağışçıların isimleri ve yaptıkları bağış miktarları basılır ve tüm kullanıcıların fiş arşivlerine eklenir. Vakıflar harcama yaptığında fişlerdeki miktarı günceller ve kalan para miktarını kaydetmek için yeni bir fiş oluşturur. Bu fiş, bağışın tamamlandığını göstermek amacıyla kullanılır.

Bu süreçte paraların doğru kişilere aktarılması vakıf kuruluşlarının dürüstlüğüne ve güvenilirliğine bağlıdır. Bu süreçteki dolandırıcılık ve sahte belgelendirmeler ile yapılan bağışların ihtiyaç sahiplerine ulaşamaması durumu ortaya çıkabilir. Blok zinciri üzerinde yaptığımız bu sistem ile bağışçılar bağışlarını bir kişiye değil, merkeziyetsiz bir sisteme yapmaktadır ve bu sistem, şeffaf bir biçimde ağ üzerinden (Ethereum, BSC) doğrulanıp herkese açık bir biçimde yayınlanmaktadır. Akıllı kontrat içerisinde toplanan para ve merkeziyetsiz yapı sayesinde akıllı kontrat içerisinde oluşturulan fiş mimarisi ile bağışlar insan faktöründen etkilenmeden direkt olarak ihtiyaç sahiplerine aktarılmaktadır. Kullanıcılar para transferi ve takip işlemlerini web üzerindeki DApp ile takip eder ve gerçekleştirilen bağış işlemini fiş içerisindeki işaretleme algoritması sayesinde sistem içerisine son kullanıcıya kadar takibini gerçekleştirebilir.

Çalışmanın takip eden bölümlerinde, ikinci bölümde blok zincir teknolojisi ile birlikte remix ortamı ayrıca react ve solidity program dilleri hakkında detaylar

verilecektir. Üçüncü bölümde projemizdeki kullanılması düşünülen yapıların literatürde benzer projeler ile karşılaştırılması yapılmaktadır. Dördüncü bölümde akıllı kontratların genel olarak oluşturacağı mimari hakkında detaylı bilgiler verilecektir. Beşinci bölümde takip sistemi içerisindeki web ile kontrat bağlantısı, kontratın ve arayüzün cüzdan etkileşimi, alım işlemleri gerçekleştirilirken dikkat edilmesi gereken kavramlar, etiket sisteminde kullanıcıdan alınacak bilgiler ve saklama yöntemi, oluşturulacak sistemin web arayüz tasarımı detaylandırılacaktır. Altıncı bölümde projenin genel anlamda amaçladığı hedeflerin sonuçlarından bahsedilecektir.

2. BLOK ZİNCİRİ VE KULLANILACAK DİĞER TEKNOLOJİLER

2.1. Blok Zinciri Teknolojisi

Blok zinciri en basit haliyle, blokların kriptografi ile korunarak birbirine bağlanan bir yapıdan oluşmaktadır. Her bloğun karakterini yapılan işlemlerin içerdiği veriler, zaman damgası (timestamp) ve önceki bloğun hash değeri oluşturmaktadır. Her bir blokta işlemlerin kendi içerisinde kayıtlarını içerdiği genel defter olarak düşünülebilir. Blok zinciri tek bir yerde değil, düğümler ağında depolanmaktadır ve burada her ağ düğümü bu blok zincirinin bir kopyasına sahiptir. Bu, tüm kayıtların herkese açık olduğu ve tüm ağ düğümleri tarafından kolayca doğrulanabileceği anlamına gelir, bu da bir düğümdeki blokzincirindeki herhangi verinin değiştirilmesini çok pahalı hale getirir. Blok zincirine bir blok eklendiğinde, tüm düğümler arasında fikir birliği sağlanmadan bloğun işlemlerini değiştirmek son derece zordur. Tüm bu özellikler temeli eşler arası konsensüs protokolüne dayalı tasarlanmıştır [1].

2.1.1. Blok zinciri teknolojisinin avantajları

Blok zinciri merkezi olmayan bir yapıya sahip olduğundan, güvenliği ve güvenilirliği artırır. Verilerin şeffaf bir şekilde paylaşıldığı ve her işlemin kriptografik olarak doğrulandığı bir sistem sunar. Bu, veri manipülasyonunu engeller ve güvenilirlik sağlar. Ayrıca, blok zinciri teknolojisi, araçların ortadan kaldırılmasına ve doğrudan işlemlerin gerçekleştirilmesine olanak tanır, bu da maliyetleri azaltır ve işlemleri hızlandırır. Verilerin dağıtık olarak saklanması, tek bir noktada meydana gelebilecek

arızaların veya saldırıların etkisini azaltır [2]. Blok zinciri ayrıca izlenebilirlik ve şeffaflık sağlar, kaynakların takip edilmesini iyileştirir. Tüm bu avantajlar, blok zinciri teknolojisinin birçok sektörde etkili ve yenilikçi çözümler sunmasını sağlar.

2.1.2. Blok zinciri teknolojisinin dezavantajları

Blok zinciri teknolojisinde her işlem ve veri, ağda bulunan her katılımcı ile paylaşılır. Ancak her ne kadar blok zincirinin güvenilir olduğundan bahsediyor olsak da bazı durumlarda hassas verilerin ifşa olma riski bulunmaktadır. Blok zinciri ağında bulunan zayıf halkalar bütünlük ve güvenlik sorunlarına yol açabilir [2].

Blok zincirinin başka bir dezavantajı olarak enerji tüketiminden söz edilebilir. Özellikle Proof of Work algoritmasını kullanan blok zincirleri, matematiksel problemleri çözmek için fazla miktarda enerji kullanırlar. Bu, çevreye zarar veren bir durumdur.

2.1.3. Blok zinciri çeşitleri

Blok zinciri teknolojisinde farklı ihtiyaçlara ve izin mekanizmalarına yönelik olarak farklı blok zinciri ağı çeşitleri bulunmaktadır. Bunlar özel blok zinciri, konsorsiyum blok zinciri ve açık blok zinciri olarak üç farklı sınıfta incelenebilir.

Özel blok zinciri; ağa yalnızca ağın kurucusu veya herhangi kurucu yetkisine sahip katılımcı tarafından izin verilen kullanıcılar katılım sağlayabilirler [3]. Bu tür sistemlerde ihtiyaç dahilinde merkezi otorite kuralları değiştirebilir veya işlemleri geri alabilir. Özel sistemlerin kullanılması sayesinde maliyetlerin düşürülmesi ve verimliliklerin artırılması sağlanır [4].

Konsorsiyum blok zinciri; açık (public) ve özel (private) blok zincir ağlarının birleşimi olarak tanımlanabilir. Bu tarz blok zincir sistemlerinde veriler, açık veya özel olabilir. Bu tarz ağlara IBM firması tarafından geliştirilen Hyperledger projesi örnek olarak verilebilir.

Açık blok zinciri; ağda bulunan herkes işlemi kontrol edip doğrulayabilir ve consensus sürecine dahil olabilir [2]. Bu sistem tamamen bağımsızdır ve bir merkezi otorite gerektirmez. Bu sistemlere Bitcoin ve Ethereum örnek verilebilir.

Neden açık blok zinciri diye düşünecek olursak; açık blok zinciri ağlarında kullanıcıların kimlik doğrulaması, public key cryptography kullanılarak gerçekleştirilir. Açık blok zinciri yapılanmasında, her ne kadar katılım herkese açık olsa da, kimlik doğrulama işlemi sayesinde ağdaki kullanıcıların güvenliği sağlanmaktadır [5]. Bunun dışında açık blok zincirlerindeki düğüm sayısı, kötü niyetli kişilerin ağı tehlikeye sokacak işlemlerini gerçekleştirmelerinin zorlaşmasını sağlar, çünkü diğer kullanıcılar bu işlemleri reddedecektir.

2.1.4. Blok zinciri teknolojisinde gaz kavramı

Madencilerin blok zincirini korumak ve güvenliğini sağlama almak için yaptıkları çalışmaları telafi etmek amaçlı gaz kavramı tanıtıldı. Proof of stake algoritması Eylül 2022'de kullanıma sunulduktan sonrasında gaz ücretleri ETH'yi stake etmenin ve doğrulamanın ödülü olarak kullanılmaya başlandı; bir kullanıcı ne kadar çok stake yaparsa o kadar fazla ödül almaktadır.

"Gaz limiti", bir doğrulayıcının belirli bir işlemde yapacağını tahmin ettiğiniz maksimum çalışma miktarıdır. Daha yüksek bir gaz limiti, genellikle kullanıcı yapacağı işlemin daha fazla çalışması gerektiği beklentisinde olduğu anlamına gelir. "Gaz fiyatı", yapılan işin birimi başına uygulanan fiyatıdır. Bir işlem maliyetini hesaplarken gaz fiyatıyla çarpılan gaz limitinin sonucu berirlirler. Pek çok işlem, gaz fiyatına eklenen ipuçlarını da içerir. Yani ne kadar çok öderseniz, işleminiz o kadar hızlı tamamlarsınız. Bir kullanıcı gaz limitini ne kadar düşük tahmin ederse, öncelik sıralamasında o kadar geriye atılır. İşlem ücreti (transaction fee), para havalesi için ödediğiniz ücrete benzer. Ağdaki işlemlerin doğrulanması ve işlenmesi gibi temel görevleri yerine getiren Ethereum doğrulayıcıları, etherlerini stake etme ve blokları doğrulama karşılığında yani anlaşılacak bir şekilde anlatılması gerekirse servis sağlayıcıya ağlarını kullanması için ödeme yapıyorsunuz [6].

Göz önünde bulundurulması gereken bir diğer faktör de, işlemlere yönelik arz ve talebin gaz fiyatlarını belirlemesidir; ağ sıkışıkça, gaz fiyatları yüksek olabilir. Öte yandan, fazla trafik yoksa düşük olabilir.

Gaz ücretleri, kullanıcıların ETH'lerini stake etmeleri için teşvik olarak Ethereum blok zincirinde ve ağında kullanılır. Staking, dürüst olmayan davranışları engellediği

için kara zinciri güvence altına almak için çalışır. Sahiplerine, ETH'lerini staking yapmaları karşılığında, blok zincirinin güvenliğini sağlamaya ve çalışmasına yardımcı olmaya yardımcı oldukları için bir ödül olarak küçük ödemeler verilir.

2.2. Kullanılan Uzlaş Algoritması

Blok zinciri teknolojisinde deftere kayıt edilecek bilgi konusunda çoğunluk tarafından ortak olarak kabul gören bir fikre varmak kritik bir noktadır. Ethereum gibi dağıtık sistemler olan blok zincirlerinde ağda bulunan düğümlerin ortak bir noktada anlaşabiliyor olması gerekir. Bu noktada gerekli anlaşma uzlaş algoritmaları ile sağlanır. Bu konuda blok zinciri teknolojisi, güvenilir ve etkili bir uzlaş algoritması kullanmalıdır [7]. Var olan uzlaş algoritmalarının en önemli amacı sisteme herhangi bir saldırı olması durumunda kötü niyetli kişileri engellemektir. Bunun yanında, blok zinciri ağına dahil olan tüm düğümlerin kayıt defterlerini aynı işlem sırasıyla günceller.

Blok zincirlerinde tercih edilen birden fazla uzlaş algoritması bulunmaktadır. Bu algoritmalarından en çok kullanılanları; PoW (Proof of Work), PoS (Proof of Stake), DPoS (Delegated Proof of Stake) ve Practical Byzantine Fault Tolerance olarak sıralanabilir. Bu başlık altında, Ethereum blok zincirinin de Ethereum 2.0 ile beraber kullanımına geçmiş olduğu Proof of Stake uzlaş algoritmasından bahsedilecektir. Bu geçişin gerçekleşmesinin nedeni olarak PoS uzlaş algoritması, eskiden kullanımda olan PoW mekanizmasına oranla daha kolay bulmacalar içerir ve bu nedenle blok oluşturmak çok daha az zaman alır [8]. Bununla beraber, daha az donanım ve elektrik maliyeti gerektirir. Bu sebeple çevre dostu olduğu söylenebilir.

Proof of Stake algoritması, bir önceki paragrafta da bahsedildiği üzere, PoW algoritmasının yüksek kaynak tüketimi problemine bir çözüm olması amacıyla 2012 yılında kullanılmaya başlanmıştır [9]. Bu tip uzlaş algoritmasında madenci kavramı yerine doğrulayıcılar bulunmaktadır. Bu doğrulayıcı düğümler, sahip oldukları kripto para biriminin geri ödenebilir miktarlarını yatırırlararak sisteme katkı sağlarlar. Bu şekilde bir sistemin ana amacı, doğrulayıcı düğümlerin kendi kripto para birimlerine ait geri ödenebilir payları bahis olarak kullanarak sahipliklerini ve güvenlerini kanıtlamalarını sağlamaktadır. Bununla beraber, kötü amaçlı saldırılar bu katkıda bulunulan olayları riske atar ve bu sayede sistem güvenliği ve anlaşma sağlanır [10].

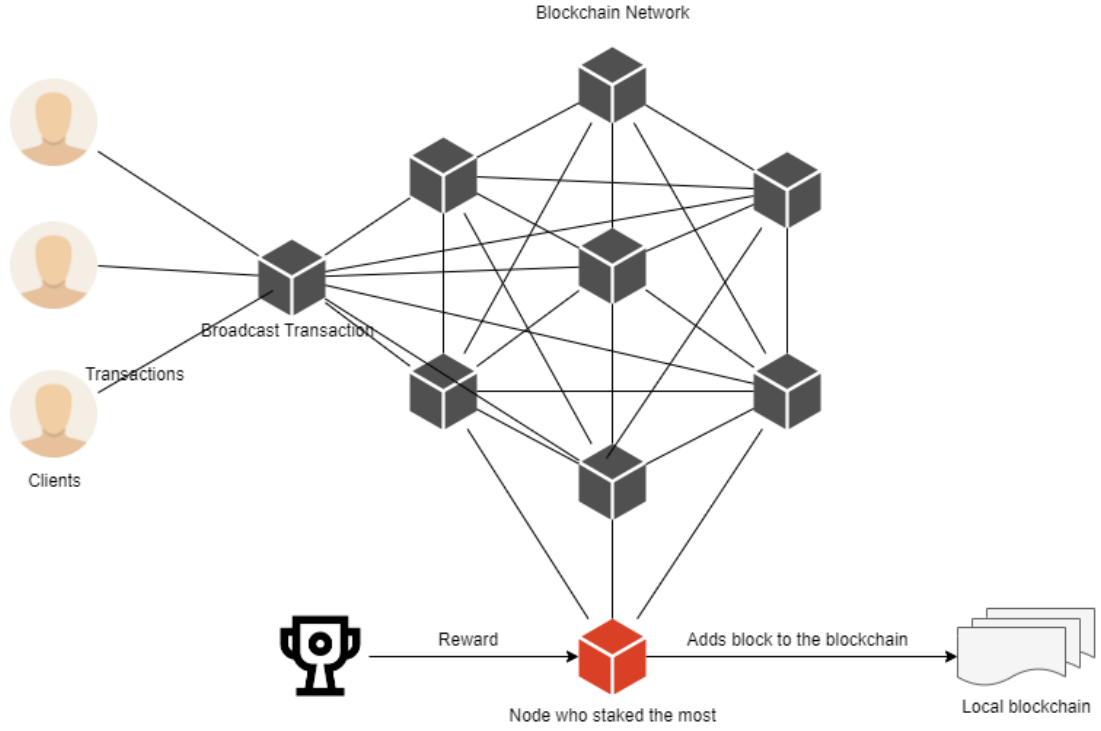
PoS algoritmasında bir sonraki bloğu oluşturan bireysel düğüm, diğer rakip düğümlere karşı ne kadar bahis yaptığına bağlı olarak seçilir [11]. Bahis genellikle, doğrulayıcının belirli bir blok zinciri ağı için ağda sahip olduğu coinlerin sayısına dayanır. Bu sistemlerde, işlem ücreti genellikle ödülüdür. Bu doğrulama işlemi sürecinin bir parçası olmak isteyen kullanıcılar bahislerini bahsedilen üzre ağda kilitlemelidir.

Ancak bu durumda ağ, en büyük bahse sahip olan tek bir düğüm tarafından domine edilecektir. Bu sorunu çözmek adına seçim sürecine başka yöntemler de eklenmiştir. Bu yöntemlerden ikisi "rastgele blok seçimi" ve "coin yaş seçimi" olarak adlandırılmaktadır [12].

Rastgele blok seçimi yönteminde, bir sonraki doğrulayıcı, karma değeri ve bahis üzerine dayalı olarak seçilir ve en yüksek bahis ve en düşük karma değerinin birleşimi olan düğüm seçilir. Ancak bu durumda genellikle düğümler bir sonraki forger'ı tahmin edebilir, nedeni ise ağ düğümlerinin sahip oldukları bahis miktarının halka açık olmasıdır.

Coin yaş seçimi yönteminde ise, bir sonraki doğrulayıcı, bahsin ne kadar süredir tutulduğu ve bahis miktarına bağlı olarak seçilir. Coin yaş, bahis yapılan madeni paraların bahiste tutulduğu gün sayısının bahis yapılan madeni para sayısı ile çarpılmasıyla hesaplanır. Bir düğüm tarafından bir blok doğrulandıktan sonra, coin yaş değeri tekrar sıfırlanır. Büyük bahislere sahip düğümler tarafından blok zincirin domine edilmesini engellemek için bir blok doğrulandıktan sonra, düğümün bir sonraki bloğu doğrulamak için belirli bir süre beklemesi gerekir.

Bir düğüm bir sonraki bloğu doğrulaması için seçildiğinde, blokta bulunan işlemlerin geçerli olup olmadığını kontrol eder ve işlemler geçerli ise düğüm tarafından imzalanır ve nihayetinde düğüm tarafından blok zincirine kayıt edilir. Blokta bulunan bu işlemlerle ilişkili olan işlem ücretleri, düğüm tarafından bir ödül olarak alınır.



Şekil 2.1. PoS Algoritması

2.3. React.js

Günümüzde front-end framework'ler ve kütüphaneler, modern web geliştirmenin önemli bir parçası haline gelmiştir. React.js, JavaScript topluluğu içinde yavaş yavaş modern web geliştirme için popüler bir framework haline gelen bir front-end kütüphanesidir [13].

React.js, Facebook tarafından geliştirilen açık kaynaklı bir JavaScript framework'ü ve kütüphanesidir. Vanilla.js'te kullanıldığından çok daha az kodla etkileşimli kullanıcı arayüzleri ve web uygulamaları oluşturmak için hızlı ve verimli bir şekilde kullanılmaktadır.

React'te, bağımsız lego blokları olarak düşünebileceğiniz yeniden kullanılabilir bileşenler oluşturarak uygulamalarınızı geliştirirsiniz. Bu bileşenler, bir araya getirildiğinde uygulamanın tüm kullanıcı arayüzünü oluşturan son bir arayüzün ayrı parçaları olarak düşünülebilir.

React'in bir uygulamadaki birincil rolü, en iyi ve en verimli oluşturma işlemini sağlayarak, bu uygulamanın görünüm katmanını bir MVC modelindeki V (View) gibi işlemektir. Tüm kullanıcı arayüzünü tek bir birim olarak ele almak yerine, React.js, geliştiricileri bu karmaşık kullanıcı arayüzlerini, tüm kullanıcı arayüzünün yapı taşlarını oluşturan tek tek yeniden kullanılabilir bileşenlere ayırmaya teşvik eder. Bunu yaparken React.js framework'ü, JavaScript'in hızını ve verimliliğini, web sayfalarını daha hızlı oluşturmak ve son derece dinamik ve yanıt veren web uygulamaları oluşturmak için DOM'u manipüle etmenin daha verimli bir yöntemiyle birleştirir.

2.4. Solidity

Solidity, akıllı sözleşmeleri uygulamak için nesne yönelimli, üst düzey bir dildir. Akıllı sözleşmeler, Ethereum durumundaki hesapların davranışını yöneten programlardır.

Solidity, Ethereum Virtual Machine'i (EVM) hedeflemek için tasarlanmış süslü parantezli bir dildir. C++, Python ve JavaScript'ten etkilenir. Sağlamlık statik olarak yazılır, diğer özelliklerin yanı sıra kalıtımı, kütüphaneler ve karmaşık tanımlamaları destekler. Solidity ile oylama, kitle fonlaması, kör müzayedeler ve çoklu imza cüzdanları gibi kullanımlar için sözleşmeler oluşturabilirsiniz.

Sözleşmeleri dağıtırken, Solidity'nin en son yayınlanan sürümünü kullanmalısınız. İstisnai durumlar dışında, yalnızca en son sürüm güvenlik düzeltmeleri alır. Ayrıca, son değişiklikler ve yeni özellikler düzenli olarak tanıtılmaktadır [14].

Solidity genel yazım kuralları, fonksiyonları ve değişken tiplerinden bahsetmek gerekirse; address türü 20 bayt veya 160 bite eşit olan bir Ethereum adresi tutabilir. Önünde 0x olan onaltılık gösterim şeklinde döndürür. Sadece bir sözleşme fonksiyonunun yürütümü esnasında depolanan değerlere bellek (Memory) değişkenleri denir. Bunlar blok zincirinde kalıcı şekilde depolanmadıkları için kullanımları çok daha ucuzdur. "block.timestamp" değişkeni ise "uint256" tipinde mevcut blok dönemi zaman damgası göstermektedir. "msg.sender" değişkeni adres tipindedir ve kontrat ile etkileşime geçen kişinin bilgileri konusunda değerler dönderir. Constructor fonksiyonları sadece sözleşme ilk dağıtılığında tek sefer

yürütülür. Kontrat içerisinde sadece tek sefer için değer ataması gerekli olan değişkenler için kullanılmaktadır [15]. Solidity’de kullanılabilecek fonksiyon çeşitleri aşağıdaki şekildedir [14]:

External fonksiyonlar sözleşme arayüzünün bir parçasıdır, bu da diğer sözleşmelerden ve işlemler yoluyla çağrılacakları anlamına gelir. External bir fonksiyon f internal olarak çağrılmaz (yani $f()$ çalışmaz, ancak $this.f()$ çalışır). External fonksiyonlar, büyük veri dizileri aldıklarında bazen daha verimlidir.

Public fonksiyonlar sözleşme arayüzünün bir parçasıdır ve internal olarak veya mesajlar yoluyla çağrılabilir. Genel durum değişkenleri için otomatik bir alıcı fonksiyonu oluşturulur.

Internal tipindeki fonksiyonlara ve durum değişkenlerine, bunu kullanmadan yalnızca internal olarak (yani mevcut sözleşme veya ondan türetilen sözleşmeler içinden) erişilebilir.

Private fonksiyonlar ve durum değişkenleri, türetilmiş sözleşmelerde değil, yalnızca tanımlandıkları sözleşme için görünür.

2.5. Remix IDE

Solidity programlama dilinde yazılan akıllı sözleşmeler için en yaygın ve güvenilen derleyici editör olan Remix IDE, Ethereum üzerinde akıllı sözleşme geliştirmek için gereken eklentilere erişim sağlayan bir geliştirme ortamına sahiptir. Bu IDE’de diğer IDE’lerde olduğu gibi kod yazabilir, derleyebilir, test edebilir ve hata ayıklayabilirsiniz.

Remix, açık kaynaklı bir IDE olup, esnek erişim imkanıyla birlikte ücretsiz olarak çoklu eklentilere ve sezgisel GUI'lere erişim sunar. Hem acemi hem de uzman geliştiriciler için faydalıdır. Acemi geliştiriciler, akıllı sözleşme geliştirmeye yönelik uzmanlık kazanmak için Remix IDE’yi kullanabilirken, uzman geliştiriciler ise akıllı sözleşmeleri test etmek ve hata ayıklamak için bu IDE’yi tercih edebilirler.

Remix IDE, Remix kütüphaneleri ve Remix eklenti motoru gibi farklı hizmetlere de sahiptir. Web tabanlı olarak veya masaüstü uygulaması olarak kullanılabilir. Ayrıca, Visual Studio Code eklentisi olarak da mevcuttur.

2.6. Etherscan ve BscScan

Etherscan [16], Ethereum blok zincirindeki işlemleri, adresleri, akıllı sözleşmeleri ve diğer ilgili verileri takip etmek için kullanılan bir Ethereum blok zinciri tarayıcısıdır. Ethereum ağının şeffaflığı sayesinde kullanıcılarına genel olarak erişilebilir bir veri tabanı sunar. Bu platform üzerinden kullanıcılar, Ethereum adreslerini arayabilir, işlemleri kontrol edebilir, akıllı sözleşmeleri inceleyebilir ve blok zinciri üzerindeki diğer etkinlikleri izleyebilirler.

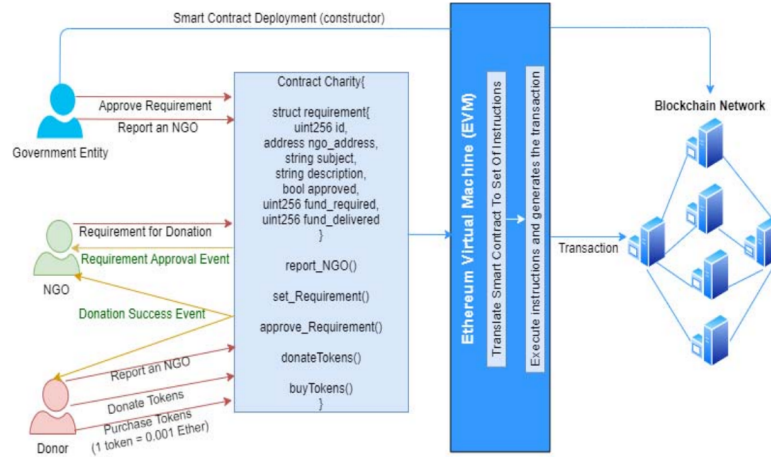
BscScan [17], tıpkı Etherscan'de olduğu gibi, Binance Smart Chain üzerindeki işlemleri, adresleri, akıllı sözleşmeler ve diğer ilgili verileri takip etmek için kullanılan bir blok zinciri tarayıcısıdır. BscScan, Etherscan'in Binance Smart Chain için özelleştirilmiş bir sürümüdür. BSC ağının şeffaflığı sayesinde kullanıcılarına genel olarak erişilebilir bir veri tabanı sunar. BscScan üzerinden kullanıcılar, BSC adreslerini arayabilir, işlemleri kontrol edebilir, akıllı sözleşmeleri inceleyebilir ve blok zinciri üzerindeki diğer etkinlikleri izleyebilirler. Ayrıca BscScan, geliştiricilere uygulamalarını BSC ile entegre etmeleri için BSC ile etkileşimde bulunan birçok hizmetin API'sini sağlamaktadır.

Etherscan ve BscScan ekosistem için önemli araçlar ve kaynaklardır. Kullanıcılara değerli bilgiler sunarlar. Bu tarayıcılar sayesinde kontratın içeriği herkes tarafından okunabilir, ayrıca tüm işlemleri gösteren arayüz sunmaktadır. Bununla birlikte Remix'te de olduğu gibi kontrat ile etkileşime girebilecek bir arayüz sağlar ve kontratın ABI bilgilerine de ulaşılabilir. Çalışmada ise kontratın herkes tarafından okunabilmesi ve kontratı ağ üzerinde yayınlayan kişinin kontratı doğrulaması amaçlı kullanılmıştır.

3. LİTERATÜR TARAMASI

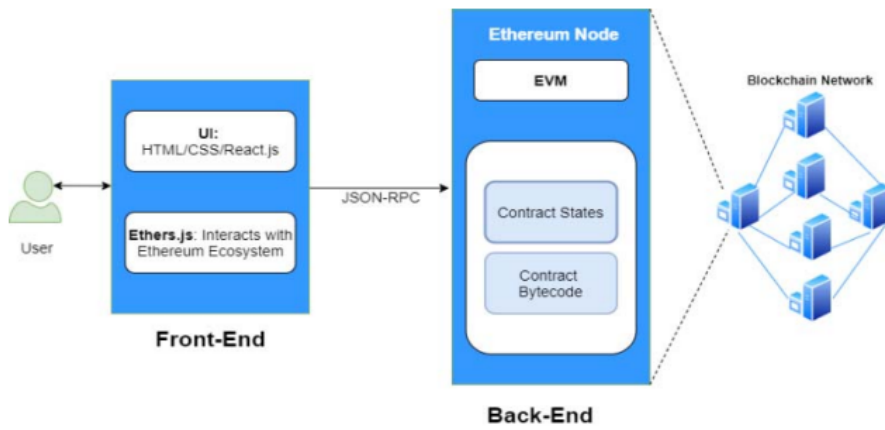
Blok Zincir Kullanan Yardım, Hayırseverlik ve Bağış Takip Sistemi adlı makalede sözleşmede kullanıcıdan cüzdan, onay ve vatandaşlık numarası bilgileri alınarak

ethereum ağı ile transfer işlemi gerçekleştirilmiştir. Buradaki yaklaşım token mantığında olup BNB ve ETH gibi para birimlerine karşılık gelen kendi ekonomik sistemi kullanmaktadır [18]. Böylelikle kendi para birimlerini BNB veya ETH gibi para birimlerine çevirerek gerçek para akışı sağlamaktadır.



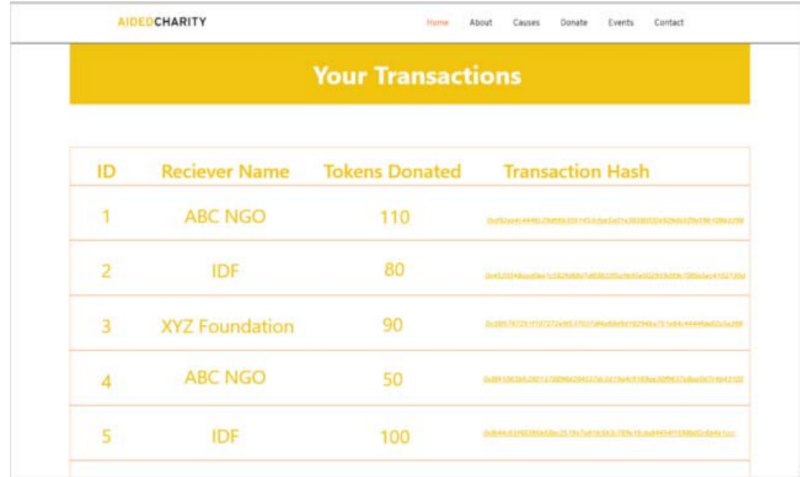
Şekil 3.1. Sistem Tasarımı [18]

Arkada gerçekleşen transfer işlemlerinin UI kısmını HTML/CSS/React.js kullanarak sözleşmede web arayüzü ile bağlantı kısımları ve kullanıcının cüzdan bağlaması, karşılıklı transfer onayı ve transfer fonksiyonlarının React içerisinde bulunan “Web3” API’ si yardımı ile gerçekleştirilmektedir.



Şekil 3.2. Sistem Mimarisi [19]

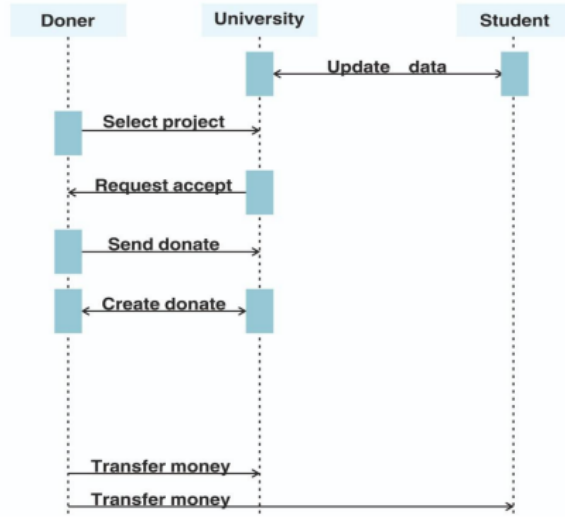
Burada BSC Testnet üzerinde gerekleřtięi gibi transfer bilgileri kullanıcıya gsterilmektedir fakat o paranın bir sonraki adımını takibi yapılmamaktadır. Buradaki senaryo sadece czdzandan czdana para aktarımı gerekleřirken iyi alıřmaktadır.



ID	Reciever Name	Tokens Donated	Transaction Hash
1	ABC NGO	110	0x050a00
2	IDF	80	0x050a00
3	XYZ Foundation	90	0x050a00
4	ABC NGO	50	0x050a00
5	IDF	100	0x050a00

řekil 3.3. Kullanıcı Transfer Gemiři [18]

niversite Baęıřları iin Blok Zinciri Tabanlı Takip Sistemi adlı bu makalede ęrencilerin 3. parti bir kuruluř olmadan direkt olarak baęıřıların niversite onayı ile ęrencilerin czdanlarına baęıř yapılması iin tasarlanmıřtır. Baęıřıların baęıř niyetlerini eřitli faktrler etkilerken [19], baęıřılar ve ilgili dięer taraflar arasındaki gven eksiklięi, insanların hayır kurumlarına baęıř yapmaktan kaınmasının temel nedenidir. niversitenin sunduęu kurallar erevesinde baęıř gerekleřir, bu kurallar niversitenin onayıyla yazılan akıllı szleřmenin iinde yer almaktadır. Burada gerekleřtirilen projenin sonucunda byk hacimli veri alabilmek iin szleřmenin optimize edilmesi tavsiye edilmiřtir.



Şekil 3.4. Akış Diyagramı [20]

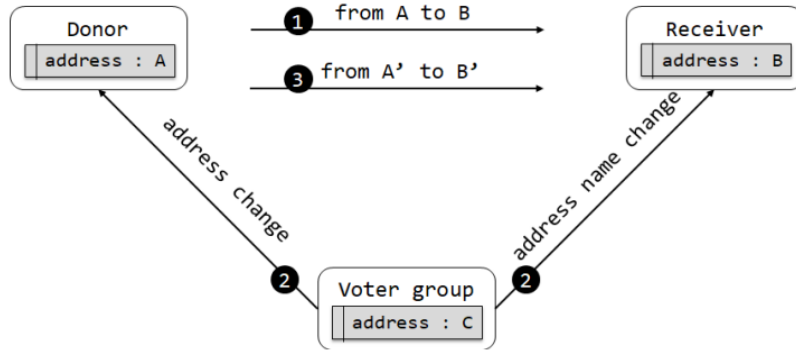
Güvenlik açığı oluşturan etkenleri saf dışı bırakması makaleyi önemli kılan noktadır. İlk olarak üniversitenin bağışçı talebini kabul veya reddettiğini belirtmek, bir bilgisayar korsanının veya bilinmeyen bir tarafın bu kapsama girmesi olasılığını önler. Diğerleri ise öğrenciye paranın ulaşmasındaki gecikme, bağışçı parasının kaybolduğunu hissedebilir bu yüzden, geliştirilen blok zinciri özel ağlara indirgeyip, ağ trafiği düşürülerek daha hızlı transfer gerçekleştirilir [20].

Islamic University

ID_student	Type of donation	Donation amount	Request for donation	
985725	Tuition fees	325\$	Accept	Refuse
199655	Student supplies	120\$	Accept	Refuse
253678	Tuition fees	389\$	Accept	Refuse

Şekil 3.5. Bağış Talebi [20]

Konuyla ilgili olarak yapılmış bir diğer çalışmada bağışçı sistemde bir özel ve bir genel anahtara sahip kullanıcıdır. Bağışçının kendine ait bir ETH adresi vardır. Bu adres akıllı sözleşmelere erişmek ve oluşturmak için kullanılır. Aynı şekilde alıcı da özel ve genel bir anahtara sahiptir. Seçmen grubu, sistemi kullanan tüm kullanıcılardan oluşur ve hesabı seçmen hesabı olarak adlandırılır.



Şekil 3.6. Hesap Adresi Dönüştürme İşlemi. [21]

Sistemde kullanılan akıllı sözleşmeler bağış sözleşmeleri olarak geçmektedir. Bunlar alıcının oluşturduğu, bağışçının implement edebildiği sözleşmelerdir. Sözleşmeler sözleşme adresini ve alıcı hesabının adresini içerir.

Bu çalışmada kullanılan sistemde Chrome’da MetaMask kullanılan bir ağ ortamı bulunur. Solidity programlama dili web tabanlı Remix IDE’de JavaScript VM ortamı olarak kullanılır [21]. Remix IDE’ nin belirtilen faydaları göz önünde bulundurularak çalışmada bu teknolojinin kullanılmasına karar verilmiştir [21].

Çalışma konumuzla ilgili bir diğer makalede, bağışçılar ve yararlanıcılar arasında güven, şeffaflık ve kronoloji sağlayan ademi merkezilikle blok zinciri tabanlı yardım (BBC) programları önerilmektedir. Bu yazıda, hayır kurumları (CP) olarak tanımlanan bağışçıların ve hayır kurumlarından yararlananlar (CB) olarak tanımlanan yararlanıcıların, hayır kurumları tarafından çeşitli bağış toplayanlar için yönetilen blok zincirine kaydoldukları bir BBC programı olan uphaar’dan bahsedilmektedir (CO) [22].

Yardım planı verileri ve CP’ye fon tahsisleri blok zincirinde defter girişleri olarak yönetildiğinden, plan sahte bağış toplayıcıları azaltmaktadır. Büyük CP ve CB kullanıcılarının güvenliğini ve gizliliğini korumak için program, pahalı dijital imzalamayı azaltan ve anahtar dağıtım merkezleri aracılığıyla sertifika oluşturmayı ortadan kaldıran sertifikasız şifreleme (CLC) yoluyla kısmi anahtar nesiller önermektedir. İmza yönleri için şema, içeriden gelen saldırılara karşı gizli anlaşmayı ortadan kaldıran ve hesaplama maliyetlerini azaltan hafif, sertifikasız bir imza şifreleme şeması (CSS) önerir. Şema, internet güvenlik protokolleri ve

uygulamalarının otomatik doğrulaması (AVISPA) aracı kullanılarak güvenlik analizi ve gayri resmi saldırı analizi ve maliyet değerlendirmeleri için değerlendirilir.

Uygulama, CP ve CB arasındaki para akışlarının ikili güvenlik ve şeffaflık sorunlarını ele alıyor. İlk olarak, blok zincir, toplanan paranın CB'yi düzeltmek için dağıtılmasına izin veren güvenilir ve değişmez kronolojik defter girişleri sağlar. CP, CB ve işlem imzalarının gizliliği ve gizliliği için planlar, geniş bir kullanıcı grubu arasındaki güvenlik kısıtlamalarını ele alan bir CLC ve CSS entegrasyonu önermektedir. Son olarak, toplanan fonlar SC aracılığıyla CB'ye gönderilir ve bağış sertifikaları CP'ye geri verilir. Ayrıca, yardım planı blok zincirinde halka açık olduğundan, CO ve sertifika yetkilileri arasındaki çarpışmalar hafifletilerek para akışının uçtan uca görünürlüğü sağlanır.

Bir diğer makalede şu soru yanıtlanmaya odaklanılmıştır: Blok zincir teknolojisini kullanarak hayır kurumu bağış süreçlerinin izlenebilirliğini ve şeffaflığını nasıl artırabiliriz? Bu soruyu cevaplamak için de ilgili tüm taraflar arasındaki güveni artırmak ve bağışların hedeflenen alıcılara ulaşmasını garanti altına almak amacıyla hayır bağışlarının hem izlenebilirliğini hem de şeffaflığını iyileştirmeyi amaçlayan blok zincir tabanlı bir bağış izlenebilirlik framework'ü önerilmektedir. Bu Framework ayrıca npo'lar ve onların hayır kurumu bağış süreçleriyle ilgili tüm belirsizlikleri ve şüpheleri ortadan kaldırmayı amaçlamaktadır. Ayrıca, yardım bağışlarının izlenmesine ilişkin uygulanabilirliğini ve verimliliğini test etmek için önerilen Framework'e dayalı olarak web tabanlı bir sistem geliştirilmiştir. Geliştirilen sistemin, tüm şüpheli araçları ve faaliyetleri ortadan kaldırmada kullanımı kolay ve etkili olduğu kanıtlanmıştır [23].

Mevcut bağış sistemleri birçok temel ve arzu edilen özellikten yoksun olduğundan, blok zincirinin yeteneklerini kullanarak tarafların güvenliğini ve güvenliğini kazanabilecek dinamik, güvenli, şeffaf ve güvenilir bir hayır kurumu bağış sistemi geliştirmeye açık bir ihtiyaç vardır. Bu makale, yardım sektöründeki literatüre ve uygulamaya aşağıdaki önemli katkıları sağlamaktadır:

Şu anda hayır kurumu bağış süreçleriyle ilgili sorunların ortadan kaldırılması açısından izlenebilirliğin önemini vurgulamak.

Bağış sürecine dahil olan tüm tarafların bilgilerini toplayan ve güvence altına alan hayır bağışları için blok zincir tabanlı bir bağış izlenebilirlik framework'ü önermek.

Mali israfın gizlenmesini veya hileli faaliyetlere katılmayı önleyecek, hayır kurumlarına yapılan bağışlarla ilgili bilgileri şeffaf ve merkezi olmayan bir şekilde saklama aracı sağlamak.

Hayır bağışlarıyla ilgili en güncel ve doğru ayrıntıların kullanılabilirliğini kolaylaştırmak amacıyla teorik kavramları pratik uygulamalarla birleştirmek.

Dağıtılmış defterler ve akıllı sözleşmeler kullanarak hem geriye hem de ileriye doğru izlenebilirliği sağlarken, aynı zamanda tüm tarafların kimliklerinin doğrulanmasını ve gizliliklerinin korunmasını sağlar.

Hayır kurumu bağış süreçlerinde yer alan taraflar arasında yüksek derecede şeffaflık ve güven elde etmek için bağışların ilgili tüm taraflar tarafından izlenebilir olması gerekmektedir. Tüm tarafların bağışçı tarafından yapıldıkları andan hedeflenen alıcılar tarafından alındıkları ana kadar hayır bağışlarını izlemelerini sağlamayı amaçlayan blok zincir tabanlı bir bağış izlenebilirliği (BBDT) çerçevesi önerilmektedir. Amaç, genel hayır kurumu bağış sürecini otomatikleştirmek ve bununla ilişkili güven ve şeffaflık düzeyini artırmaktır. Önerilen Framework'te, hayır kurumu bağış zinciri bağışçı ile başlar ve muhtaç tarafla sona erer. Yeni bir hayır kurumu bağış izlenebilirlik sistemi oluşturmak için halka açık izin verilen bir blok zinciri kullanır. Dahası, yetkili, hesap verebilir ve değiştirilemez bir sistem elde etmek için katılımcıların kimlikleri garanti edilir. Önerilen sistem, bir işleme katılan her bir taraf için genel ve özel adresler oluşturmak için bir kripto para cüzdanı kullanır. Açık anahtar, kişisel kimliklerine bir bozulma getirmeksizin net çalışma içindeki her tarafın kimliğidir [24]. Mütevelli ve muhtaç listesine dahil olanların ağa katılmaya davet edilmesi gerekir, bu da dolandırıcıların para kazanmak amacıyla ağa katılmasını engeller. Üstelik, bir bağışçı ağa katılmak istiyorsa, bir katılma isteği göndermelidir. Bir bağışçının bilgileri doğrulandıktan sonra, onlara bir doğrulama kodu gönderilecektir. Tüm taraflar tek bir merkezi olmayan ağda toplanacak ve her bir tarafın ağda kendi blok zincir hesabı olacaktır.

Ayrıca sistem, sistemin işlevlerine erişim izni verilmeden önce ilgili tüm tarafların kimliklerinin doğrulanmasını gerektirir. Hem muhtaç taraflar hem de müteveli heyetleri dava oluşturma yeteneğine sahiptir. Ancak muhtaç tarafların oluşturduğu tüm davaların bağış alabilmesi için müteveli heyetleri tarafından onaylanması gerekir. Ayrıca bağışçılar, bağış sürecinin tüm aşamalarında bağışlarının ilerlemesini takip etme olanağına sahiptir. Sistemin özellikleri, ağı başarılı bir şekilde katılan her taraf tarafından kullanılabilir. Ayrıca, müteveli heyetlerinin temel işlevleri yeni davalar oluşturmak, davaları kabul etmek veya reddetmek ve bağışları takip etmektir. Son olarak, sistem, bu dava tamamlandıktan sonra belirli bir davaya katkıda bulunan tüm bağışçıları bilgilendirir.

Önerilen Framework, halka açık bir Ethereum zinciri oluşturarak ve ardından Solidity diliyle akıllı sözleşmeler yazarak blok zincir teknolojisini kullanmaktadır. Sistem, gerekli işlevsel bağları sağlamak için birkaç farklı bileşeni entegre eder. Her bir tarafın kişisel verileri, diğer taraflarca tespit edilemeyen ve erişilemeyen bir adres değeri ile yeniden gönderilecektir. Ayrıca taraflar arasındaki etkileşimler, kimliklerinin açığa çıkmasını veya mahremiyetlerinin ihlal edilmesini önleyecek olan MetaMask adreslerine dayanacaktır. Böylece önerilen sistem, ilgili tüm taraflara tam anonimlik sağlayacaktır [25].

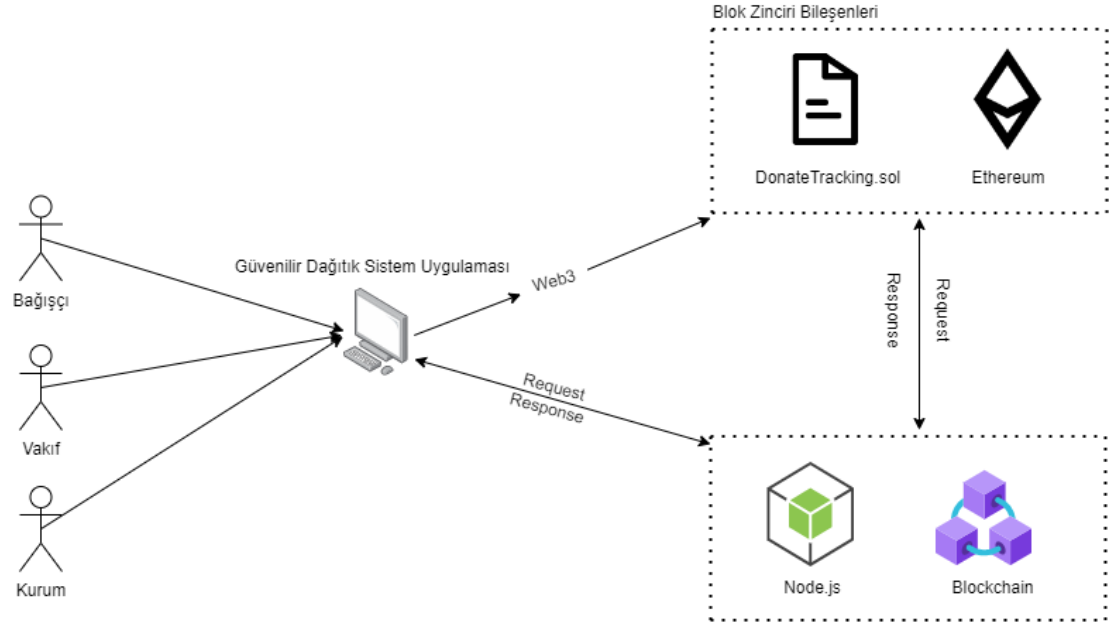
4. AKILLI KONTRATLAR İLE GENEL SİSTEM TASARIMI

Sistem altyapısı akıllı kontrat içerisinde gerçekleşmektedir. Kontrat içerisinde kullanıcıların gerçekleştirdiği transfer geçmişi, kullanıcıların sisteme kayıt olurken kontrat sahibinin kullanıcılara dağıttığı giriş kodu ve kullanıcının kimlik bilgisi saklanmaktadır. Sistemin bu kısmı veri tabanı gibi işlemektedir. Kullanıcıların sisteme tamamen erişim sağlaması için kayıt kodu kontrat sahibi tarafından tanımlanmaktadır. Kullanıcıların anonimliğini ortadan kaldırmak içinse kullanıcı ismi ve kimlik numarası talep edilmektedir. Bu bilgiler kullanıcıdan alındıktan sonra kullanıcıya özel, 'mapping' tipindeki bir değişken içerisine 'byte32' tipine çevrilmiş kimlik numaraları kaydedilir. Burada kayıt kodu ve kimlik bilgisi haricinde kullanıcının ismi, toplam bağış yapılan para ve vakıf tarafından kullanıcının yaptığı bağış üzerinden harcanan miktar bilgileri saklanmaktadır [26].

Sistem üzerinden bir cüzdandan diğerine kripto para aktarmak için iki çeşit transfer fonksiyonu vardır. Bu fonksiyonlardan ilki bağışçı ve vakıf arasında transfer sağlar. Burada sistem üzerinde yetkililer tarafından (kontrat sahibi) kaydedilen vakıf adresleri bulunmaktadır. Kullanıcı vakıf adreslerine para transferi gerçekleştirirken hangi vakfa ne kadar bağış yapacağını da kendisi yüzdelik oranı girerek transfer işlemini gerçekleştirmektedir. Daha sonra transfer işleminin kullanıcının belirli kimlik bilgisi ve kayıt kodu kontrolü sistem tarafından bakılır. Bu şartlardan geçtikten sonra bağışçı ve vakfın kimlik, kullanıcı ismi, cüzdan adresi ve bağış miktarı alınır. Sistem tarafından rastgele üretilen dokuz karakterli bir kod ile çağrılan değişken içerisine kayıt edilir. Bu yapı fiş gibi düşünülebilir. Fiş içerisine kaydedilen bu bilgiler kullanıcıların kimlik bilgileri ile oluşturulmuş fiş kodlarını saklayan bir değişken üzerine kaydedilir. Böylelikle her bir transfer için oluşturulan ve bu transfer ile etkileşim içerisine giren kullanıcıların kimlik numarası ile sadece fişlerinin saklandığı ve bunları görüntüleyebildikleri bir yapı oluşur.

İkinci olarak sistem üzerinde adresleri kaydedilen vakıf ile ticari firma arasındaki transferi gerçekleştiren bir fonksiyon vardır. Fonksiyonun genel yapısı önceki anlatılanla benzer çalışır fakat birtakım farklılıklar vardır. Bağışçıların yaptığı bağışlar vakıfların cüzdanlarına aktarılmamaktadır. İnsan faktörü ortadan kalkarak blok zincirinin merkeziyetsiz yapısı sayesinde burada vakıfların dolandırıcılık girişimlerine karşı önlem amaçlı kontrat içerisinde saklanmaktadır. Transfer fonksiyonu içerisindeki yapı fişler üzerinden yapılmaktadır. Burada vakıflar belli bir miktar ticari firmalardan ihtiyaç sahiplerine yapılacak yardım için alınacak eşyalar için gerçekleştirilecek transferlerinde kendi üzerlerine yapılan bağışlara fişler aracılığı ile yapmaktadır. Burada sırası fişler içerisindeki bağış miktarları genel harcanacak miktar ile eşitlenene kadar fişler arasında dolaşılır. Bu miktar eşitlendikten sonra diğer fiş sistemine ek olarak bağışçıların isimleri ve yaptıkları bağış miktarları fiş üzerine basılarak etkileşime geçmiş tüm kullanıcıların (bağışçılar, vakıf ve ticari firma) fiş arşivlerine eklenir. Sistem vakıfların harcama yaptığında fişler içerisindeki miktarın bir kısmını yazıp diğer kısmını harcamama durumlarında kalan para miktarını kaydetmek için fişin bir kopyasını oluşturur, ayrıca bu fiş içerisindeki tüm paranın bağışlandığına dair işaretlemek amaçlı da kullanılmaktadır.

Kontrat içerisinde kullanıcıların bilgileri daha düzenli görebilmesi amaçlı ve sadece kontrat sahibinin çağırabildiği fonksiyonlar vardır. Sistem üzerine vakıf ve ticari firmaların/ihtiyaç sahiplerinin adresleri sadece yetkili tarafından eklenmesi ve silinmesi durumu kontrat sahibine bırakılmıştır. Böylelikle kontroller sayesinde daha güvenilir bağış sağlanmaktadır.



Şekil 4.1. Genel Sistem Mimarisi

4.1. Akıllı Kontratlar İçerisindeki Fonksiyonlar ve Kullanım Amaçları

Remix IDE kullanılarak yazılan akıllı sözleşme testnet ağı üzerinden yayımlanarak, sonraki bölümlerde bahsedilecek fonksiyonların testleri gerçekleştirilmiştir. Akıllı sözleşmede kullanılması düşünülen bu fonksiyonları ilerleyen bölümlerde tek tek anlatılmıştır.

4.1.1. Mapping fonksiyonları

Solidity programlama dilinde eşlemeler (mappings), diğer dillerdeki bir hash tablosu veya sözlük gibi işlev görür. Bu, verileri anahtar-değer çiftleri şeklinde depolamak için kullanılır [27].

Bir anahtar herhangi bir yerleşik veri türü olabilir, ancak referans türleri izin verilmezken değer herhangi bir türde olabilir. Eşlemeler genellikle eşsiz Ethereum adresini ilgili değer türüyle ilişkilendirmek için kullanılır.

AddressToLabels adlı fonksiyon içerisinde; adres değişkeni ile kimlikNo, loginCode, userName, totalDonate, totalSpending bilgilerini Label” adlı bir struct içerisinden çağırarak amacıyla kullanılmaktadır.

PersonelInvoice adlı fonksiyon içerisinde; adres değişkenine kaydedilen byte32 tipinde bir dizi içerisinde kullanıcının fiş numaralarını saklamak ve istenildiğinde döndürmek amacıyla kullanılmaktadır.

LabelCodestoAddress adlı fonksiyon içerisinde; kullanıcıların login kodları üzerinden adres sorgulamak için kullanılmaktadır.

TransferCodestoAddressTrack adlı fonksiyon içerisinde; oluşturulan fiş numaraları ile kullanıcıların transfer bilgilerini göstermek amaçlı kullanılmaktadır.

IsFoundation adlı fonksiyon içerisinde; vakıflara özel fonksiyonlar çağırılırken çağırılan kişinin vakıf kimliğini doğrulamak amaçlı kullanılmaktadır.

4.1.2. Genel fonksiyonlar

Bu bölümde ise akıllı sözleşmede bulunan mapping fonksiyonları dışındaki transfer yaparken veya arayüz içerisinde bilgileri düzenli ve sıralı aktarırken kullanılan genel fonksiyonlardan bahsedilmiştir.

TransferBNB adlı fonksiyon içerisinde; para transferi yaparken kimlik numarası doğrulaması yapar ve ardından belirtilen adreslere para transferi yapar. Bağışçının girdiği miktar, gerekli kontroller yapıldıktan sonra belirlenir ve bu miktar, fiş numarası ile birlikte transfer geçmişindeki tüm bilgileri içeren bir fişe kaydedilir.

TransferFoundationBNB adlı fonksiyon içerisinde; sadece vakıfların kullanabileceği bir işlemdir. Bu işlem, normal transfer işleminden farklıdır çünkü vakıfların harcama bilgileri dışında işaretleme ve fiş mimarisi sayesinde kullanıcıların paralarını takip edebildikleri mimariyi de kapsamaktadır.

GetAllLabels adlı fonksiyon içerisinde; kullanıcı bilgilerinden sisteme kayıtlı olan login kodları ile kimlik bilgilerini liste olarak çeker.

GetAllFoundation adlı fonksiyon içerisinde; sistemde kayıtlı olan tüm vakıfların cüzdan adreslerini liste halinde döndürmektedir.

GetAllCommercialFirms adlı fonksiyon içerisinde; sisteme kayıtlı ticari kurumların adreslerini liste olarak döndürür.

GetAllTransfereHistory adlı fonksiyon içerisinde; yollanan parametre içerisindeki adresin tüm transfer geçmişini liste şeklinde döndürür.

AddFoundation adlı fonksiyon içerisinde; sisteme yeni vakıfları veya ihtiyaç sahibi bir bireyin adresini kaydetmek için kullanılır. Sadece kontrat sahibi bu fonksiyonu çağırabilir.

RemoveFoundation adlı fonksiyon içerisinde; sisteme yeni vakıfları veya ihtiyaç sahibi bir bireyin adresini silmek için kullanılır. Sadece kontrat sahibi bu fonksiyonu çağırabilir.

AddCommercialFirms adlı fonksiyon içerisinde; sistemin içerisine ticari firma adresi ekler. Sadece kontrat sahibi bu fonksiyonu çağırabilir.

RemoveCommercialFirms adlı fonksiyon içerisinde; sistemin içerisine ticari firma adresi siler. Sadece kontrat sahibi bu fonksiyonu çağırabilir.

GetFoundationCount adlı fonksiyon içerisinde; sisteme kayıtlı vakıf sayısı döndürür.

GetCommercialFirmsCount adlı fonksiyon içerisinde; sisteme kayıtlı ticari firma sayısını döndürür.

GetlabelCountPI adlı fonksiyon içerisinde; kullanıcı adreslerinin içerisinde kayıtlı olan fiş sayısını döndürür.

GetPersonelInvoice adlı fonksiyon içerisinde; kullanıcı adreslerinin içerisinde kayıtlı olan fiş kodlarını döndürür.

GenerateMyLabel adlı fonksiyon içerisinde; sistem içerisine kimlik bilgisi kaydetmek için kullanılır.

SetlabelToSystem adlı fonksiyon içerisinde; sistem içerisine kontrat sahibinin kodu ile kayıt olmak ve tüm sistemi kullanmak için kullanılır.

SetUserNameToSystem adlı fonksiyon içerisinde; sistem içerisine kullanıcıların kullanıcı adlarını byte32 tipinde kaydedip saklar.

TransferHistoryAndPersonelInvoiceSaved adlı fonksiyon içerisinde; sistemde gerçekleştirilen transfer bilgilerini (from, to, fromKimlikNo, toKimlikNo, Amount, fromUserName, toUserName) kaydeder. Fiş kodu basar ve sorgulamak için fişleri kişilerin adres bilgisine kaydedip daha sonra çağrılabilmesi için saklar.

TransferHistoryAndPersonelInvoiceSavedFoundation adlı fonksiyon içerisinde; sistemde gerçekleştirilen transfer bilgilerini (from, to, fromKimlikNo, toKimlikNo, Amount, fromUserName, toUserName) kaydeder. Fiş kodu basar ve sorgulamak için fişleri kişilerin adres bilgisine kaydedip daha sonra çağrılabilmesi için saklar. Bunu yaparken fiş ve işaret algoritmaları sonucunda fişlere yazılacak bağışçıların isimlerini de hem fişlere yazıp hem de fişler ile etkileşime geçen kişilerin adreslerine transfer bilgileri fiş arşivlerine aktarılmaktadır.

5. BAĞIŞ TAKİP SİSTEMİ

Bu başlık altında akıllı kontrat içerisindeki genel kurallardan, bununla birlikte kontrat içerisindeki fiş ve işaretleme algoritması yapısından, kontrat ile web arayüzü arasındaki etkileşimi sağlayan sistemden ve genel olarak web üzerinde oluşturulan web arayüzünden bahsedilecektir. Takip sisteminin genel yapısından bahsetmek gerekirse; akıllı kontrat içerisinde yazılan kurallar çevresinde kullanıcıların elektronik cüzdanlar arasında kontrollü transfer yapılması sağlanmaktadır. Transfer işlemleri bağışçı ve vakıf/ihtiyaç sahibi ya da ticari firma arasında gerçekleşir. İşlemler için dokuz karakterli bir kod ve fiş kesme yapısı kullanılır. Vakıflar ihtiyaç sahiplerine yardım amaçlı ticari firmalardan harcama bağışçıların parası ile gerçekleşir. Fiş içerisinde işaretleme algoritması sayesinde bağışçılara bilgilendirme yapmak amaçlı kesilen fişleri, transfer ile etkileşime girmiş tüm kullanıcıların fiş

arşivlerine eklenerek bağışçıların para takibi gerçekleştirilir. Burada kullanıcı tüm transfer işlemlerini web arayüzünde takip edebilir ve transfer işlemlerini gerçekleştirebilir.

Expenditures Of Foundations						
AHBAP Total Spending: 0.5000						
TEGV Total Spending: 0.3000						
Code	From	To	Citizen Num(From)	Citizen Num(To)	Donate Amount	Donaters
70S7Y9P50	AHBAP	MIGROS	12345678913	12345678915	0.25 BNB	TAHA[0.10] RANA[0.15]
0LDRCFOGE	AHBAP	MIGROS	12345678913	12345678915	0.05 BNB	RANA[0.05]
FV92SHJYZ	AHBAP	MIGROS	12345678913	12345678915	0.20 BNB	TAHA[0.20]
S36KTEIUS	TEGV	MIGROS	12345678914	12345678915	0.30 BNB	TAHA[0.10] RANA[0.20]

Şekil 5.1. Transfer Geçmişi Takip Sistemi Genel Tasarımı

5.1. Bağış Toplama ve Para Takibi Amaçlı Merkeziyetsiz Uygulama

Akıllı kontratların web arayüzü üzerinde bağlantılarını ve erişimini sağlamak için birtakım araçlara ihtiyaç duyulmaktadır. Genel anlamda bu tarz araçları kapsayan yapıların genelini kapsayan Web3 kavramı ile karşılaşılmaktadır.

Web3 kavramından bahsetmek gerekirse; öncelikle bu sistemi gerçek anlamda kavramak için internet tarihini ve Web3'ün kendisinden önceki modellerinden nasıl farklı olduğunu anlamak gerekmektedir. Web 1.0, genellikle statik HTML sayfalarından varolan bir koleksiyondur ve sınırlı etkileşim olanakları kullanıcılara sunmaktadır. Belli başlı tartışma forumları özel sohbet ve tartışma panoları gibi platformlara ev sahipliği yapsada, internet çoğu kişilerin aklında az sayıda etkileşimin veya finansal işlemin gerçekleştiği bir yer olarak akıllara kazınmıştır.

İnteraktiflik ve finansal işlemlerin gerçekleştiği yerlerde, güvenli ödeme yöntemi altyapısı bulunmadığından dolayı limitleri sınırlı bir yapı sunmaktaydı. Pizza Hut, müşterilerin pizza sipariş vermelerini sağlamak amacıyla web sitesi içerisinde müşteri bilgilerini içeren bir sipariş formu sunduğundan dolayı Web 1.0 alanında

yenilikçi bir şirket olarak ön plana çıktı fakat ödemelerini web sitesi üzerinden değilde şahsen yapılması gerekiyordu [28].



Pizza Hut

Welcome to Pizza Hut!

This Electronic Storefront is brought to you by **Pizza Hut and The Santa Cruz Operation**. You may click on the Pizza Hut logo on any page to submit comments regarding this service to webmaster@pizzahut.com.
Take a look at what we have to offer: [Sample Menu](#)

If you would like to order a pizza to be delivered, please provide the following information:

Name

Street Address

Voice Phone ###-###-#### (where we can reach you)

[Continue](#)

SCO
OPEN SYSTEMS SOFTWARE

Şekil 5.2. PizzaHut Sipariş Formu [28]

İnternet kendi gelişim hızını arttırmaya devam etmekteydi. Fiber optik altyapıları ve arama motoru optimizasyonları nedeniyle sosyal interaktivite, müzik, video paylaşımı ve finansal işlemler için kullanıcı istemlerinin çarpıcı biçimde artış göstermesiyle, web 2004 yılında bu taleplere karşı gelişme göstermeye başladı.

Daha fazla iletişim için bu istemler, şu anlarda internet kuruluşların ve şirketlerinin genelinin varoluşuna sebebiyet verdi. Facebook ve Twitter gibi öncü platformlar sosyal interaktiviteyi kolaylaştırarak insanların ihtiyaçlarını karşıladı; Google, insanlara büyük bir çevrimiçi bilgi havuzu içerisinde bilgilenmeleri için etkili bir imkan sağlamaktadır. Bankaların çevrimiçi hizmetlerindeki 256 bit AES gibi şifreleme standartlarının oluştuğu finansal etkileşimler ve elektronik fon transfer gibi yenilikler doğdu.

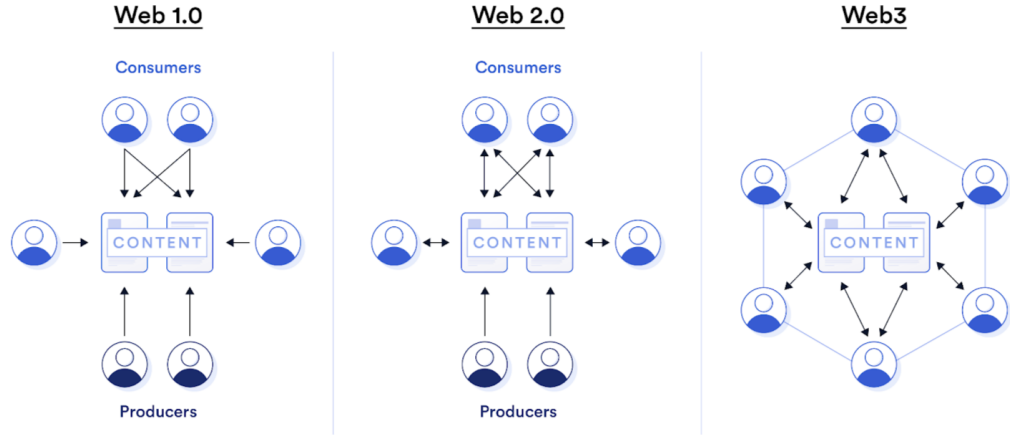
Bu yeni ve daha etkileşim imkanları sunan internet, yeni fonksiyonlar ekleyerek kullanan bireylerin web tecrübelerini genel anlamda iyileştirdi. Ancak aynı şekilde, günümüze kadar çevrimiçi hayatlarımızı tanımlamayı sürdüren pazarlıksız yaklaşım da sundu: Bahsettiğimiz yeni fonksiyonlardan ve interaktivitelere yararlanmak amacıyla insanların çoğu bilgi ve sorumluluğu kavramlarıyla taraflara bölünmüş üçüncü kişi platformlara teslim edilmesi ve merkezi varlıklara veri ve içerik sahipliği açısından ciddi anlamda güç ve aksiyon vermesi gerekiyordu.

2008'de Satoshi Nakamoto , blok zinciri teknolojisinin temellerini ve dijital para birimini ana hatlarıyla vurgulayan Bitcoin teknik açıdan inceleyen makalesi yayınladı ve Web 2.0 paradigmasını yenilemek için bir yol belirledi. Bitcoin, dijital işlemler konusunda insanların fikrini kökten değiştirdi ve internet aracılığıyla güvenilir bir üçüncü parti yazılıma ihtiyaç duymadan gerçekleştirilen ilk güvenli para alışverişi aracını kullanıcılara sundu. Satoshi, “Gerekli olan, güven yerine kriptografik kanıtlara dayalı bir elektronik ödeme sistemidir” diye de makalesinde insanlara değil teknolojiye olan güven ile merkeziyetsiz sistemin önemini vurgulamış oldu.

Akıllı sözleşmelerin keşfine kadarki süreçte merkeziyetsiz bir internet modeline ihtiyaç duyulmadı. Akıllı sözleşmeler, aslında kullanıcılar arasında doğrudan ve güvenli bir biçimde işlem yapmalarını olanak tanıyarak; şeffaf ve kriptografik aracılığıyla güvenliği sağlanarak, gerçeklerle desteklenen bu kavram yenilik olarak karşımıza çıktı.

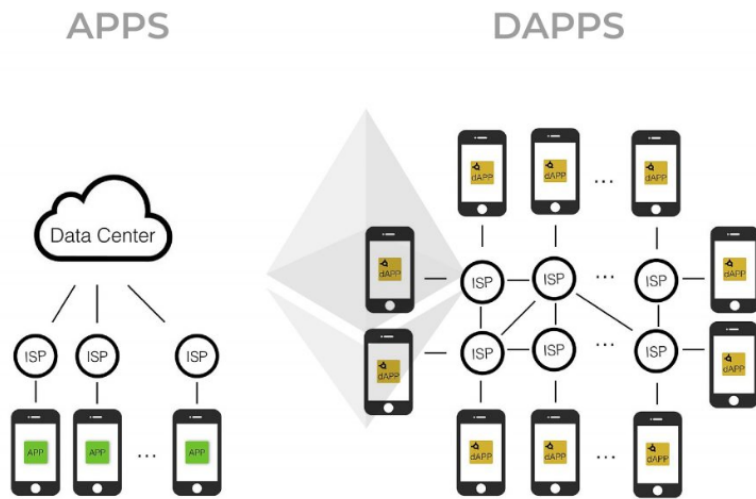
Basitçe özetlemek gerekirse, Web3 kavramı genel anlamda sözleşmeler sistemini yaratmayı bununla birlikte bireylerin ve kurumların anlaşmalara ulaşma yöntemini yenileştirmeyi hedefleyen merkeziyetsiz bir internet gösterimidir.

Web3 son kullanıcılarının, cihazlarının ve uç ağlarının cihaz üzerindeki uç zekayı geliştirilmesiyle birlikte dağıtılmasını sağlar. Protokolleri: bağlantı, içerik alışverişi, mesaj geçirme, veri paylaşımı, koordinasyon vb. için şifreli, izlenebilir, P2P, D2D, uç ağ iletişimi ve uç bulut ağ iletişim protokollerini desteklemektedir. Altyapı: verimli, enerjiye duyarlı, güvenilir, gecikmeye duyarlı P2P, D2D, uç bulut ağı, bilgi işlem, kaynak sağlama, programlama ve hizmet verme vb. gibi özellikler sunmaktadır. İletişim: heterojen uç cihazlar aracılığıyla son kullanıcılar arasında kişiselleştirilmiş, heterojen iletişim protokolleri ve ağ bağlantısı vb. sistemlerle ekosistemde güvenli, gizliliği koruyan P2P ve D2D iletişimleri sağlamaktadır. Geliştirme açısından açık kaynak, son kullanıcı, son cihaz, uç ağ odaklı DApp programlama ve etkileşimli bir iş akışı odaklı arayüz oluşturmayı destekler.



Şekil 5.3. Web Altyapısının Gelişimi [28]

DApp'ler, Web 2.0 dünyasından alışkın olduğumuz uygulamalardan ve Web 1.0 alanındaki statik HTML sayfalarından ayıran net özelliklere sahiptir. Bunlar herhangi bir birey veya kuruluş tarafından değil, blok zinciri ağlarının merkeziyetsiz altyapısı yönünden desteklenmektedir. Görünüşte basit, merkeziyetsiz programlar, merkeziyetsiz finans (DeFi) , veriye duyarlı sigorta ürünleri , oyna-kazan çevrimiçi oyunlar gibi buna benzer karmaşık, otomatik sistemler yaratmak için kullanılmaktadır [29].



Şekil 5.4. App ile dApp Farkı [29]

Genel olarak kavramlardan bahsettik. Akıllı kontrat ile blok zinciri teknolojisi yardımı ile yapılan bu bağışların insan faktörünü ortadan kaldırıp kontrat içerisinde paranın birikmesi ve kontrat içerisindeki fiş ve işaretleme algoritması yardımı ile direkt olarak ihtiyaç sahibine aktarılması bu merkeziyetsizliği ve güvenliği sağlamaktadır.

5.1.1. Kripto cüzdan bağlantısı

Kripto parablok zinciri işlemlerinin doğru şekilde onaylanması için işlemlerin bir eşsiz anahtar ile imzalanmasıyla birlikte halka açık bir anahtar kullanılarak onaylanmalıdır. Blok zinciri, son kullanılacak bloğun bilgileriyle güncellenmesi, sadece bu basamaktaki işlemlerin uygulanması sonucunda gerçekleşmektedir. Eşsiz anahtarlar tüm kişiler için emsalsizdir, kullanıcıya özeldir. Aynı şifrelerinizin önemle sakladığınız gibi saklamanız gerekir.

Eşsiz anahtarları diğer kişilerden korumak için elektronik kripto cüzdanlarında saklanması gerekmektedir. Süreçlerin gerçekleştirilmesi için eşsiz anahtarlar büyük önem taşımaktadır. Aynı biçimde eşsiz anahtarlarınızın, hasebiyle kripto paralarınızın güvenliği için elektronik kripto cüzdanları büyük önem taşır. İki çeşit elektronik kripto cüzdanı vardır: sıcak cüzdanlar, eşsiz anahtarlarınıza ulaşmanızda yardım eden internet ile bağlantılı durumdaki tarayıcı eklentileri olarak kullanılmaktadır. Laptoplar, telefon veya tarayıcı gibi cihazlarla internete erişebilmeniz gerekir. Soğuk cüzdanlar ise aynı USB gibi fiziksel donanımlar içerisinde barındırılan kripto cüzdanlardır. İnternet kaynağı olmadan yani çevrimdışı çalıştıklarından dolayı soğuk cüzdanlar diğerine göre daha güvenilirdir ve işlemlerin onaylanmasında sadece bilgisayar erişimi gereklidir. Ne yazık ki soğuk cüzdanlar, sıcak cüzdanlar kadar avantaj sağlamamaktadır.

Kripto cüzdanı MetaMask'ın önemli avantajlarından biri de açık kaynaklı yazılım olmasıdır. MetaMask 'ı kullanan müşteriler yazılım ile ilgili bilgilere ve yeniliklere aktif şekilde erişebilmekte bununla beraber bu yeniliklere etkileşime geçebilmektedir. Açık kaynaklı olmasından dolayı MetaMask tüm kullanıcılar için güvenli ortam sağlamaktadır. MetaMask gibi diğer firmaların sağladığı kripto cüzdanları açık kaynaklı değildir.

React projemizin içerisinde web uygulamaları blok zinciri ağına bağlantısını sağlayan kütüphanelerden Rainbow Kit ve Wagmi modülleri yardımı ile kripto cüzdanlarını akıllı kontratımızı ile erişim sağlanması amaçlanan web arayüzüne entegre edilmektedir. Kullanıcılar bu sayede elektronik cüzdanını arayüze bağlayarak yetkilendirme ve izin taleplerine karşılık yanıt oluşturabilecek ve akıllı kontrat içerisindeki fonksiyonlarla yetkisi olduğu süreçte etkileşimde olabilecektir.

Daha sonra blok zincir ağ konfigürasyonu için id, isim, ağ, ikon bağlantısı, ikon arka plan rengi, para biriminin özellikleri (decimal, isim, sembol), RPC url, blok tarayıcısı gibi değişkenler atanmaktadır.

```
const BSCTestnetChain = {
  id: 97,
  name: "BSC Testnet",
  network: "Binance Smart Chain Testnet",
  iconUrl:
    "https://cryptologos.cc/logos/binance-coin-bnb-logo.png",
  iconBackground: "#fff",
  nativeCurrency: {
    decimals: 18,
    name: "BNB",
    symbol: "BNB",
  },
  rpcUrls: {
    default: "https://data-seed-prebsc-1-s1.binance.org:8545",
  },
  blockExplorers: {
    default: {
      name: "BSC Testnet",
      url: "https://testnet.bscscan.com/",
    },
  },
  testnet: true,
};
```

Oluşturulan bu konfigürasyonu Wagmi modülü içerisindeki “configureChains()” fonksiyonu içerisine parametre olarak konulmasıyla Provider ve Chains değişkenleri döndürür.

```
const { provider, chains } = configureChains(
```

```
[BSCTestnetChain],  
[jsonRpcProvider({ rpc: (chain) => ({ http: chain.rpcUrls.default }) })]  
);
```

Rainbow Kit modülü içerisindeki MetaMask, Trust Wallet, Wallet Connect ve Coinbase Wallet gibi cüzdan uygulamalarının blok zincir ağ konfigürasyonları Chains değişkeni yollanarak ayarlanır.

```
const connectors = connectorsForWallets([  
  {  
    groupName: 'Recommended',  
    wallets: [  
      metaMaskWallet({ chains }),  
      trustWallet({ chains }),  
      walletConnectWallet({ chains }),  
      coinbaseWallet({ chains }),  
    ],  
  },  
]);
```

Connectors ve Provider değişkenleri Wagmi modülü içerisindeki “CreateClient()” fonksiyonu içerisine parametre olarak yollanır. Artık Wagmi içerisindeki modülü blok zinciri ağı ile bağlantısı kurulduğuna göre bağlanan tüm hesapları akıllı kontratlar ile etkileşimde bulundurabiliriz.

5.1.2. Web arayüzünün akıllı sözleşme ile etkileşimi

ABI (Application Binary Interface), bilgisayar bilimi bağlamında, genellikle işletim sistemleri ve kullanıcı programları olmak üzere iki program modülü arasındaki bir arabirimdir.

EVM (Ethereum Virtual Machine), Ethereum ağının temel bileşenidir. Akıllı sözleşmeler önceden de belirtildiği gibi EVM üzerinde yürütülen Ethereum blok zincirinde depolanan kod parçalarıdır. Solidity veya Vyper gibi üst düzey dillerde yazılan akıllı sözleşmelerin EVM yürütülebilir bayt kodunda derlenmesi gerekir; bir akıllı sözleşme dağıtıldığında, bu bayt kodu blok zincirinde depolanır ve bir adresle ilişkilendirilir. Ethereum ve EVM için bir akıllı sözleşme, yalnızca bu bayt kodu

dizisidir. Üst düzey dillerde tanımlanan işlevlere erişmek için kullanıcıların, bayt kodunun onunla çalışabilmesi için adları ve bağımsız değişkenleri bayt gösterimlerine çevirmesi gerekir. Yanıt olarak gönderilen baytları yorumlamak için, kullanıcıların daha yüksek seviyeli dillerde tanımlanan dönüş değerleri grubuna geri dönüş yapması gerekir. EVM için derlenen diller, bu dönüştürmeler hakkında katı kurallar uygular ancak bunları gerçekleştirmek için işlemlerle ilişkili kesin adların ve türlerin bilinmesi gerekir. ABI, bu adları ve türleri kesin, kolayca ayrıştırılabilir biçimde belgeler ve yöntem çağrılarını ile akıllı sözleşme işlemleri arasında keşfedilebilir ve güvenilir çeviriler yapar.

```
[{"internalType":"bytes32","name":"","type":"bytes32"}],  
"name":"tempTransferCodestoAddressTrack",  
"outputs":[{"internalType":"address",  
"name":"fromAddress","type":"address"},  
{"internalType":"address","name":"toAddress","type":"address"},  
{"internalType":"bytes32","name":"myLabelCode","type":"bytes32"},  
{"internalType":"bytes32","name":"labelCode","type":"bytes32"},  
{"internalType":"uint256","name":"donateBalance","type":"uint256"},  
{"internalType":"bytes32","name":"fromUserName","type":"bytes32"},  
{"internalType":"bytes32","name":"toUserName","type":"bytes32"},  
{"internalType":"bool","name":"donated","type":"bool"}],  
"stateMutability":"view","type":"function"},  
{"inputs":[{"internalType":"uint256[]","name":"donateRate","type":"uint256[]"}],
```

Bir kodun arayüzünün insan tarafından okunabilir bir temsili olan API'ye (Uygulama Programı Arayüzü) çok benzer. ABI, tıpkı API'nin yaptığı gibi, ancak daha düşük bir düzeyde, ikili sözleşmeyle etkileşime geçmek için kullanılan yöntemleri ve yapıları tanımlar. ABI, işlev imzaları ve değişken bildirimleri gibi gerekli bilgileri, EVM'nin bu işlevi bayt kodunda çağırmak için anlayabileceği bir biçimde kodlaması için işlevi çağırını belirtir; buna ABI kodlaması denir. ABI kodlaması çoğunlukla otomatiktir ve REMIX gibi derleyiciler veya blok zinciriyle etkileşime giren cüzdanlar tarafından halledilir. Sözleşme ABI, JSON biçiminde temsil edilir.

Genel olarak kavramlardan bahsettik. Özetlemek gerekirse; akıllı kontrat, web arayüzü ve elektronik cüzdanın birbirleri arasında etkileşime geçmesi için React içerisindeki 'Ethers' ve 'Wagmi' kütüphanelerinden yararlanılmaktadır. Wagmi ile birlikte kontratın yüklenebileceği Ethereum ve Binance Smart Chain gibi blok zinciri

ağları sisteme tanımlanır. Bu ağların RPC adresi, ağ numarası, ondalık sayı numarası, ağ ismi ve sembolü ile tanımlanmaktadır. Daha sonra kontratı ağ üzerinde yükleyen kişi tarafından doğrulanması gerekir. Tanımlanan kontratın ABI ve adresi ağ üzerinden elde edilerek Ethers içerisindeki 'Contract' fonksiyonu içerisine parametre olarak verilmektedir. Bu şekilde React içerisindeki kod ile blok zinciri ağına yüklenmiş kontrat arasında etkileşim gerçekleştirilir. Bu bilgilere ek olarak kullanıcıların kontrat ile gerçekleştireceği işlemler içerisinde cüzdanları ile kontratın arasındaki onay ve yetkilendirme işlemleri ise Ethers sayesinde gerçekleştirilmektedir. Arayüzde bazı durumlarda kullanılan ağ bilgisi, imza yetkisi ve hesap bilgisi erişilmesi gerekmektedir, bu gibi durumlarda ise Wagmi'den yararlanılır.

5.2. Akıllı Kontrat Üzerinde Takip Sistemi İçin Oluşturulacak Algoritmanın Yapısı

Etiketleme sisteminde ilk olarak; kullanıcı sistemde işlem yapmadan önce bir kod oluşturması gerekir. Bu kod isteğe göre rastgele bir biçimde 9 karakterden oluşturulabilir ya da kullanıcı kendi istediği 9 karakter ile de oluşturabilir. Akıllı kontrat içerisinde ilk olarak “bytes” türünde iki değişken oluşturulur. Kalıcı olarak ihtiyaç duyulmayan değişkenler fonksiyon içerisinde “Memory” tipinde geçici olarak tutularak fonksiyon çağrısını tamamladıklarında direkt yok olurlar. Buradaki “bytes” dizisi içerisindeki etiket sistemi için 9 karakteri saklayan değişken ve tüm harfler ile birlikte rakamları tutan değişken, işlemler bitirildikten sonra boşuna yer tutmayıp silinmesi için “Memory” tipinde saklanmaktadır.

```
bytes memory result      = '000000000';  
bytes memory characters   =  
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
```

Bir döngü içerisinde karakter dizisinde dolaşarak rasgele fonksiyonu içerisinden dönen rakamlar ile etiketimiz oluşturulur.

```
assembly {      labelCode := mload(add(result, 32))      }
```


Kod parçası yardımı ile oluşturulan “bytes” tipindeki değişken “bytes32” tipine dönüştürülerek “struct” yapısı içerisinde depolanma amacıyla yollanılır. Kullanıcı rastgele değilde kendisinin belirlediği bir karakterde etiket oluşturmak istiyorsa “string” tipinde alınan karakter dizisini “bytes(karakter_dizisi)” yardımı ile “bytes” dizisine çevirdikten daha sonra aynı işlemler uygulanarak “bytes32” ye çevrilip sistemde depolanabilir.

Kullanıcıdan alınan bilgiler bir “struct” değişkeninde depolanır. Bu yapı; yeni objeler oluşturmak için tasarlanan, ihtiyacımız için kullanılacak değişkenlerden meydana gelen ve tekrar tekrar kullanılabilir sınıflardır. Nesneye yönelik programlama dillerindeki sınıflarla benzer işleve sahiptir. Diğer dillerde class olarak isimlendirilen sınıflar Solidity’de “struct” olarak isimlendirilir.

```
struct Label{  
    bytes32 myLabelCode;  
    bytes32 labelCode;  
    bytes32 userName;  
    uint256 totalDonate;  
    uint256 totalSpending;  
}
```

Yapının içerisinde kimlik numarası, kullanıcının kayıt kodu, kullanıcı ismi, toplam bağış ve bağışının ne kadarı aktarıldığının bilgisi saklanmaktadır. Buradaki bilgiler akıllı kontrat içerisinde haritalama (mapping) sistemi yardımı ile değişkenlere aktarılmakta ya da değişkenlerin ihtiyacına göre erişilmektedir. Haritalama, diğer programlama dillerindeki sözlük (dictionary) ya da karma tablolarına (hash table) benzer özellik taşıyan, anahtar-değer (key-value) eşleşmesi yapılarak verilerin depolandığı bir veri türüdür. Bir sözcükte anahtar yalnızca bir kere kullanılabilir.

```
mapping(address => Label) public addressToLabels;  
mapping(bytes32 => address) public LabelCodestoAddress;
```

Kullanıcıların etiketleme yapısında, özel ve tek olduğundan cüzdan adresleri ya da kimlik numaraları ile haritalama sistemine anahtar olarak kullanılmaktadır. Anahtar

içerisinde saklanan bilgilerin her birine ulaşmak için “mapping” tipinde tanımlanmış “addressToLabels” değişkeni içerisine anahtar yardımıyla “.myLabelCode” kullanıcının etiket koduna ulaşılabilir.

Kullanıcıların etiketleme yapısında, özel ve tek olduğundan cüzdan adresleri haritalama sistemine anahtar olarak kullanılmaktadır. Anahtar içerisinde saklanan bilgilerin her birine ulaşmak için “mapping” tipinde tanımlanmış “addressToLabels” değişkeni içerisine anahtar yardımıyla “.userName” kullanıcının etiket koduna ulaşılabilir.

```
addressToLabels[_labelCodes.at(index)].myLabelCode,  
addressToLabels[_labelCodes.at(index)].labelCode,  
addressToLabels[_labelCodes.at(index)].userName,  
addressToLabels[_labelCodes.at(index)].totalDonate,  
addressToLabels[_labelCodes.at(index)].totalSpending
```

Etiket sisteminin oluşturulmasından sonraki adım sistem içerisine kaydetmektir. Transfer işlemi yapılması için şart koyduğumuz etiket kodu ile ancak işlemler gerçekleştirilmektedir. Sisteme kaydetmek için ise kullanıcıdan aldığımız “string” tipindeki kodu “bytes32” ye çevirip koşullarla adres ve etiket kod kontrol ediliyor. Bu kontrollerin ilkinde sisteme kayıt edilecek etiket kodunun aynı kullanıcı tarafından bir önceki kod ile aynı olup olmadığı kontrol edilmektedir. AddressToLabels değişkeni içerisinde fonksiyonu çağıran kişi adresi verilerek karşılığı aranmaktadır. Eğer bir eşitlik yok ise kontrolden geçer ancak bir eşitlik varsa sistem bir hata döndürerek işlemleri bitirir.

```
require(addressToLabels[msg.sender].myLabelCode != bytes32(0), "You must  
have a login with citizen number");
```

Etiket kodu oluşturulurken geçici olarak oluşturan kişinin adresine bir haritalama tipindeki “labelCodestoAddress” içerisinde kayıt edilir. Eğer başarılı bir biçimde kayıt olmadıysa varsayılan (0x00) değeri olarak kalmaktadır. Bir değer atanmamış değişkenler varsayılan olarak 0’a ayarlanmaktadır. Adres sıfır, adresine gönderilen ETH ile işlem yapılamaz ve

sonsuz kadar kaybolur. İkinci koşulda ise bu adresin başarılı bir biçimde kayıt edilip edilmediği kontrol edilir.

```
require(labelCodestoAddress[_result] != address(0), "Invalid referrer code.");
```

Sistem altyapısı akıllı kontrat içerisinde gerçekleşmektedir. Kontrat içerisinde kullanıcıların gerçekleştirdiği transfer geçmişi, kullanıcıların sisteme kayıt olurken kontrat sahibinin kullanıcılara dağıttığı giriş kodu ve kullanıcının kimlik bilgisi saklanmaktadır. Sistemin bu kısmı veri tabanı gibi işlemektedir. Kullanıcıların sisteme tamamen erişim sağlaması için kayıt kodu kontrat sahibi tarafından tanımlanmaktadır. Kullanıcıların anonimliğini ortadan kaldırmak için ise kullanıcı ismi ve kimlik numarası talep edilmektedir. Bu bilgiler kullanıcıdan alındıktan sonra kullanıcıya özel, 'mapping' tipindeki bir değişken içerisine 'byte32' tipine çevrilmiş kimlik numaraları kaydedilir. Burada kayıt kodu ve kimlik bilgisi haricinde kullanıcının ismi, toplam bağış yapılan para ve vakıf tarafından kullanıcının yaptığı bağış üzerinden harcanan miktar bilgileri saklanmaktadır.

```
require(addressToLabels[msg.sender].labelCode != bytes32(0), "You must have a label to donate");
```

Sistem üzerinden bir cüzdandan diğerine kripto para aktarmak için iki çeşit transfer fonksiyonu vardır. Bu fonksiyonlardan ilki bağışçı ve vakıf arasında transfer sağlar. Burada sistem üzerinde yetkililer tarafından (kontrat sahibi) kaydedilen vakıf adresleri bulunmaktadır. Kullanıcı vakıf adreslerine para transferi gerçekleştirirken hangi vakfa ne kadar bağış yapacağını da kendisi yüzdelik oranı girerek transfer işlemini gerçekleştirmektedir. Daha sonra transfer işleminin kullanıcının belirli kimlik bilgisi ve kayıt kodu kontrolü sistem tarafından bakılır. Bu şartlardan geçtikten sonra bağışçı ve vakfın kimlik, kullanıcı ismi, cüzdan adresi ve bağış miktarı alınır.

```

struct TransferHistory{
    address fromAddress;
    address toAddress;
    bytes32 myLabelCode;
    bytes32 labelCode;
    uint256 donateBalance;
    bytes32 fromUserName;
    bytes32 toUserName;
    bytes32[] userNames;
    uint256[] donateAmounts;
    bool donated;
}

```

Sistem tarafından rastgele üretilen dokuz karakterli bir kod ile çağrılan değişken içerisine kayıt edilir. Bu yapı fiş gibi düşünülebilir. Fiş içerisine kaydedilen bu bilgiler kullanıcıların kimlik bilgileri ile oluşturulmuş fiş kodlarını saklayan bir değişken üzerine kaydedilir. Böylelikle her bir transfer için oluşturulan ve bu transfer ile etkileşim içerisine giren kullanıcıların kimlik numarası ile sadece fişlerinin saklandığı ve bunları görüntüleyebildikleri bir yapı oluşur.

```

transferCodestoAddressTrack[transferCode].fromAddress = from;
transferCodestoAddressTrack[transferCode].toAddress = to;

transferCodestoAddressTrack[transferCode].myLabelCode =
addressToLabels[from].myLabelCode;

transferCodestoAddressTrack[transferCode].labelCode =
addressToLabels[to].myLabelCode;

transferCodestoAddressTrack[transferCode].donateBalance = amount;
transferCodestoAddressTrack[transferCode].fromUserName =
addressToLabels[from].userName;

transferCodestoAddressTrack[transferCode].toUserName =
addressToLabels[to].userName;

transferCodestoAddressTrack[transferCode].userNames = resultUserNames;
transferCodestoAddressTrack[transferCode].donateAmounts = amounts;

```

İkinci olarak sistem üzerinde adresleri kaydedilen vakıf ile ticari firma arasındaki transferi gerçekleştiren bir fonksiyon vardır. Fonksiyonun genel yapısı önceki

anlatılanla benzer çalışır fakat birtakım farklılıklar vardır. Bağışçıların yaptığı bağışlar vakıfların cüzdanlarına aktarılmamaktadır. İnsan faktörü ortadan kalkarak blok zincirinin merkeziyetsiz yapısı sayesinde burada vakıfların dolandırıcılık girişimlerine karşı önlem amaçlı kontrat içerisinde saklanmaktadır. Transfer fonksiyonu içerisindeki yapı fişler üzerinden yapılmaktadır. Burada vakıflar belli bir miktar ticari firmalardan ihtiyaç sahiplerine yapılacak yardım için alınacak eşyalar için gerçekleştirilecek transferlerinde kendi üzerlerine yapılan bağışlara fişler aracılığı ile yapmaktadır. Burada sırası fişler içerisindeki bağış miktarları genel harcanacak miktar ile eşitlenene kadar fişler arasında dolaşılır. Bu miktar eşitlendikten sonra diğer fiş sistemine ek olarak bağışçıların isimleri ve yaptıkları bağış miktarları fiş üzerine basılarak etkileşime geçmiş tüm kullanıcıların (bağışçılar, vakıf ve ticari firma) fiş arşivlerine eklenir.

```
personelInvoice[from].push(transferCode);
personelInvoice[to].push(transferCode);

for ( uint i = 0; i < list.length; i++ ) {
    if(list[i] != address(0)){
        personelInvoice[list[i]].push(transferCode);
    }
}
```

Sistem vakıfların harcama yaptığında fişler içerisindeki miktarın bir kısmını yazıp diğer kısmını harcamama durumlarında kalan para miktarını kaydetmek için fişin bir kopyasını oluşturur, ayrıca bu fiş içerisindeki tüm paranın bağışlandığına dair işaretlemek amaçlı da kullanılmaktadır.

```
tempTransferCodestoAddressTrack[code].donateBalance = amount - perAmount;
```

Kontrat içerisinde kullanıcıların bilgileri daha düzenli görebilmesi amaçlı ve sadece kontrat sahibinin çağırabildiği fonksiyonlar vardır. Sistem üzerine vakıf ve ticari firmaların/ihtiyaç sahiplerinin adresleri sadece yetkili tarafından eklenmesi ve silinmesi durumu kontrat sahibine bırakılmıştır. Böylelikle kontroller sayesinde daha

güvenilir bağış sağlanmaktadır. Genel fiş ve işaretleme algoritması aşağıda verilmiştir:

```
function transferFoundationBNB(address payable _to, uint amountDonate)
external onlyFoundation{

    bytes32 transferCode;

    uint256 perAmount = amountDonate;

    require(addressToLabels[msg.sender].myLabelCode != bytes32(0), "You
must have a login with citizen number");
    require(addressToLabels[msg.sender].labelCode != bytes32(0), "You must
have a label to donate");

    bytes32[] memory userNamesDonaters = new bytes32[](10);
    uint256[] memory amountDonaters = new uint256[](10);
    address[] memory addresslist = new address[](10);

    uint256 count;

    uint256 length = personelInvoice[msg.sender].length;

    for(uint256 i = 0; i < length; i++){
        bytes32 code = personelInvoice[msg.sender][i];
        if (perAmount != 0){
            if (transferCodestoAddressTrack[code].toAddress == msg.sender) {
                uint256 amount =
tempTransferCodestoAddressTrack[code].donateBalance;
                address addressF = transferCodestoAddressTrack[code].fromAddress;
                if(transferCodestoAddressTrack[code].donated == false){
                    if (amount != 0){
                        bytes32 userN =
transferCodestoAddressTrack[code].fromUserName;
                        if (perAmount >= amount){
                            perAmount -= amount;
                            userNamesDonaters[count] = userN;
                            amountDonaters[count] = amount;
                            addressToLabels[addressF].totalSpending += amount;
                            addresslist[count] = addressF;
                            count += 1;
                            transferCodestoAddressTrack[code].donated = true;
                        }else{
                            userNamesDonaters[count] = userN;
                            amountDonaters[count] = perAmount;
                        }
                    }
                }
            }
        }
    }
}
```

```

        addressToLabels[addressF].totalSpending += perAmount;
        addresslist[count] = addressF;
        count += 1;
        tempTransferCodestoAddressTrack[code].donateBalance =
amount - perAmount;
        perAmount = 0;
        //transferCodestoAddressTrack[code].donated = true;
    }
}
}
}
}

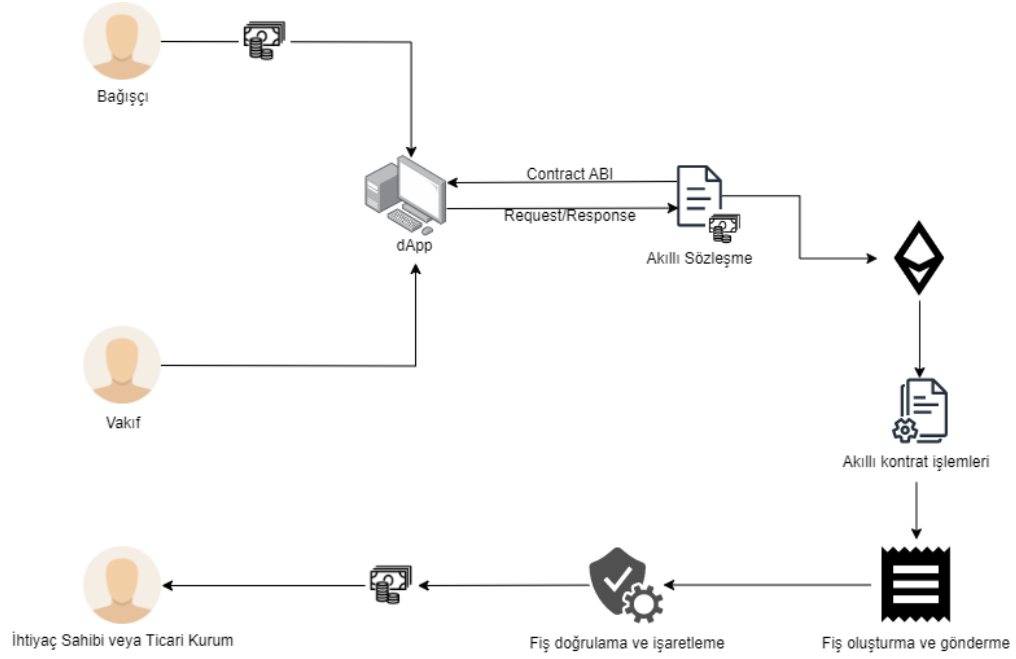
require(perAmount == 0, "Transfer Fail !");
_to.transfer(amountDonate);
transferCode = codeCreator();

transferHistoryAndPersonelInvoiceSavedFoundation(transferCode,msg.sender,_to,
amountDonate,userNamesDonaters,addresslist,amountDonaters);

    addressToLabels[msg.sender].totalSpending += amountDonate;
    addressToLabels[_to].totalSpending += amountDonate;

}

```



Şekil 5.5. Fiş Sistemi Altyapısı

5.3. Takip Sistemi Arayüzü

Arayüzde büyük önceliklerden birisi kullanıcı deneyimidir. Sade, pürüzsüz ve her yaştan kullanıcının kullanabileceği kolaylıkta bir kullanımı amaçlar. Hangi amaçla ve nasıl kullanılacağını düşünmek, unutulmaz bir tecrübe yaratmak için büyük önem taşımaktadır. Arayüzün gerçekleştirmeyi amaçladığı görevleri net olacak biçimde tanımlamak ve arayüz kullanımına yönelik motivasyonları doğru aktarmak gereklidir. Bu kriterler doğrultusunda ürün ve kullanıcı arasındaki bağ daha belirgin hale gelecektir. Bu şartlar doğrultusunda arayüz ve kullanıcı arasındaki deneyim daha anlamlı ve kullanıcının kavrayabileceği hale gelecektir. Müşterilerin bir ürünü içerisindeki arayüzün hangi amaçlar doğrultusunda kullanılacağını arkasında pek çok sebep gösterilebilir. Arayüzün kullanıcı deneyimindeki önemleri arasında; görüntüden, temsil ettiği değerlere kadar pek çok etmen içerebilir. Arayüzün kullanıcılara sağlayacakları, o ürünün ne kadar kullanışlı olduğunu vurgulamaktadır. Akıcı ve sade bir tecrübe sunmak için tüm koşullar değerlendirilir. Böylelikle, ürün veya ihtimamın tasarımı hem görsel olarak göze sade ve estetik gözükmesi hem de erişilebilirlik amacı ile optimizasyon sürecinden geçerek aktarılmaktadır.

Görsel tasarım, programlama, psikoloji gibi disiplinleri içermektedir. Kullanıcıların davranışlarını tahmin ederek duygularına yön verecek şekilde tasarımlar yapılmaktadır. Tasarımcı, kullanıcı yerine kendini koyarak hareket etmesi kilit noktalardan biridir. Potansiyel kullanıcının bedensel ve mental sınırlarına dikkat etmek ve erişilebilirlik açısından kapsamlı düşünmek bununla birlikte dikkatlerini kaybetmeyecek şekilde tasarımlar çıkartmak gerekir. Kullanıcı taharrini doğru gerçekleştirmek, prototip bir demo oluşturmak ve demoların testlerini düzgün anlamak bu alanda büyük önem taşımaktadır.

Tasarımcılar bu durumlarda kullanıcı rolüne bürünmelidirler ve kullanıcıların gerekliliklerini geliştirme sürecinin merkezine koymaları gerekmektedir. Bu nedenle, kullanıcı merkezli tasarım anlayışı benimsenmelidir.

Bu bağlamda web arayüzünde kullanıcılara hesap bilgisi, vakıf işlem takibi ve kod arama bölümleri sunulmaktadır. Hesap bilgisi sekmesinde kullanıcılar cüzdan bilgisi, kimlik numarası, toplam bağış miktarı, kayıt kodu, kullanıcı isimleri ve elektronik cüzdanındaki para miktarına tek bir yerden ulaşması amaçlandığından tablo olarak sunulmaktadır.

Your Account Info					
User Name	My Address	Citizen Number	Login Code	User BNB Balance	Total Donate
AHBAP	0x8A5508aFDFEd245E5388b887C8e86f1d976c8DAE	12345678913	12345678912	5.7673 BNB	0.5000 BNB

Şekil 5.6. Kullanıcı Hesap Bilgisi

Ayrıca kendisinin etkileşimde bulunduğu transferlerin akıllı kontrat kısmındaki fiş yapısı içerisinde transfer bilgileri de tablo biçiminde yansıtılır.

Your Transfer History						
Code	From	To	Citizen Num(From)	Citizen Num(To)	Donate Amount	Donaters
PW6Z81K51	RANA	AHBAP	1111111111	12345678913	0.20 BNB	-
UWQRID488	RANA	TEGV	1111111111	12345678914	0.20 BNB	-
70S7Y9P50	AHBAP	MIGROS	12345678913	12345678915	0.25 BNB	TAHA(0.10) RANA(0.15)
OLDRCFQGE	AHBAP	MIGROS	12345678913	12345678915	0.05 BNB	RANA(0.05)
S36KTE1US	TEGV	MIGROS	12345678914	12345678915	0.30 BNB	TAHA(0.10) RANA(0.20)

Şekil 5.7. Kullanıcı Transfer Geçmişi

Vakıf işlem takibi kısmında ise vakıflara yapılan toplam bağışların ve harcamaların fişleri tablolar halinde herkesin takip etmesi için kullanıcılara sunulmaktadır.

Donors To The Foundation						
AHBAP Total Donate: 0.5000						
TEGV Total Donate: 0.5000						
Code	From	To	Citizen Num(From)	Citizen Num(To)	Donate Amount	
GMXUQNTXO	TAHA	AHBAP	23897206614	12345678913	0.10 BNB	
PW6Z81K51	RANA	AHBAP	1111111111	12345678913	0.20 BNB	
5RW74KNFK	TAHA	AHBAP	23897206614	12345678913	0.20 BNB	
Z90GY9PX	TAHA	TEGV	23897206614	12345678914	0.10 BNB	
UWQRID488	RANA	TEGV	1111111111	12345678914	0.20 BNB	
VYSUQEAIZ	TAHA	TEGV	23897206614	12345678914	0.20 BNB	

Şekil 5.8. Tüm Vakıflara Ait Bağış Geçmişi

Kod arama kısmında ise kullanıcılara verilen fiş numaraları sayesinde kullanıcılar tablolar içerisinde aranamayacak kadar kalabalık olması durumunda özellikle bakmak istedikleri transferin bilgilerine ulaşabilmektedirler.

6. SONUÇLAR

Bu çalışma kapsamında bağış kapsamındaki tüm transferlerde para hareketlerinin kişiler (bağışçı, vakıf ve ticari firma) tarafından, akıllı kontrattan alınan verilerin kontrat ABI'leri ile Wagmi ve Ethers kütüphaneleri yardımıyla uygun bir web altyapısı ile kullanıcılara uygulama sunulmuştur. Kullanılan fiş sistemi içerisinde vakıfların sadece sistemde kendi üzerine tanımlanmış bağışları fiş üzerinden harcamaları sağlanarak bir şahsın cüzdanında paranın durmasının önüne geçilmekte

ve kontrat içerisinde harcama yaparak dolandırıcılığı önlemektedir. Bağışçıların yapılan bu bağışlarda vakıflar tarafından yaptığı bağış miktarının ne kadarı harcandığı daha sonra bilgilendirilerek paranın takibi sağlanmış olup fiş üzerindeki kimlik bilgisi, isimleri, cüzdan adresi ve bağış miktarı transfer ile etkileşime geçen tüm taraflara bilgilendirme yapılmaktadır.

Burada şunu belirtmek gerekir ki; akıllı kontrat içerisinde paranın kuralları önceden belirlenmiş ve bu kurallara herkesin bakabileceği ortamda yayınlanmış bir biçimde gerçekleştirilen bağış işlemlerinin güvenli bir biçimde ihtiyaç sahiplerine ulaşması için insan faktörünü ortadan kaldırarak hedeflenen güvenilir bir bağış takip sistemi yapısına ulaşılmıştır.

KAYNAKLAR

- [1] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor. Making smart contracts smarter. In *Proceedings of the 23rd Conference on Computer and Communications Security*, pages 254–269, 2016.
- [2] Gale P. F., Crossley P. A., Bingyin X., Yaozhong G., Cory B. J., Barker J. R. G., Fault Location Based on Travelling Waves, *Fifth International Conference on Developments in Power System Protection*, York, United Kingdom, 30 March-01 April 1993.
- [3] Iżykowski J., *Fault Location on Power Transmission Lines*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2008.
- [4] Lewis L. J., Traveling Wave Relations Applicable to Power-System Fault Locators, *Transactions of the American Institute of Electrical Engineers*, 1951, **70**(2), 1671-1680.
- [5] Aurangzeb M., Crossley P. A., Gale P., Fault Location on a Transmission Line Using High Frequency Travelling Waves Measured at a Single Line End, *IEEE Power Engineering Society Winter Meeting*, 2000, **4**, 2437-2442.
- [6] <https://www.investopedia.com/terms/g/gas-ethereum.asp>
- [7] Moritz Wendl, My Hanh Doan, Remmer Sassen, The environmental impact of cryptocurrencies using proof of work and proof of stake consensus algorithms: A systematic review, *Journal of Environmental Management*, Volume 326, Part A, 2023, 116530, ISSN 0301-4797, <https://doi.org/10.1016/j.jenvman.2022.116530>.
- [8] Buterin, Vitalik, Griffith, Virgil. (2017). Casper the Friendly Finality Gadget.
- [9] Moritz Wendl, My Hanh Doan, Remmer Sassen, The environmental impact of cryptocurrencies using proof of work and proof of stake consensus algorithms: A systematic review, *Journal of Environmental Management*, Volume 326, Part A, 2023, 116530, ISSN 0301-4797, <https://doi.org/10.1016/j.jenvman.2022.116530>.
- [10] S. M. Skh Saad and R. Z. Raja Mohd Radzi, “Comparative Review of the Blockchain Consensus Algorithm Between Proof of Stake (POS) and Delegated Proof of Stake (DPOS)”, *Int J Innov Comp*, vol. 10, no. 2, Nov. 2020.
- [11] Shifferaw, Y., Lemma, S. (2022). Limitations of proof of stake algorithm in blockchain: A review. *African Journals Online*, 21(97), 216947.
- [12] Tarek Frikha, Faten Chaabane, Nadhir Aouinti, Omar Cheikhrouhou, Nader Ben Amor, Abdelfateh Kerrouche, "Implementation of Blockchain Consensus Algorithm on Embedded Architecture", *Security and Communication Networks*, vol. 2021, Article ID 9918697, 11 pages, 2021. <https://doi.org/10.1155/2021/9918697>.
- [13] <https://blog.hubspot.com/website/react-js>
- [14] <https://medium.com/@yangnana11/solidity-function-types-4ad4e5de6d56>

- [15] <https://ethereum.org/tr/developers/docs/smart-contracts/anatomy/>
- [16] EtherScan teknolojisi dokümantasyonu. <https://etherscan> <https://etherscan.io/>
- [17] BscScan teknolojisi dokümantasyonu. <https://bscscan.com/> bscscan
- [18] A. Singh, R. Rajak, H. Mistry and P. Raut, "Aid, Charity and Donation Tracking System Using Blockchain," 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), 2020, pp. 457-462, doi: 10.1109/ICOEI48184.2020.9143001.
- [19] Hyndman, N., & McConville, D. (2016). Transparency in Reporting on Charities' Efficiency: A Framework for Analysis. *Nonprofit and Voluntary Sector Quarterly*, 45(4), 844–865.
- [20] A BLOCKCHAIN-BASED TRACKING SYSTEM FOR UNIVERSITY DONATION FERWANA, Esraa Ahmed
[<http://acikerisim.karabuk.edu.tr:8080/xmlui/handle/123456789/1506>]
- [21] Lee, Jaekyu, Aria Seo, Yeichang Kim, and Junho Jeong. 2018. "Blockchain-Based One-Off Address System to Guarantee Transparency and Privacy for a Sustainable Donation Environment" *Sustainability* 10, no. 12: 4422. <https://doi.org/10.3390/su10124422>
- [22] Saraswat, Deepti, et al. "UpHaaR: Blockchain-based charity donation scheme to handle financial irregularities." *Journal of Information Security and Applications* 68 (2022): 103245.
- [23] Almaghrabi, Abeer, and Areej Alhogail. "Blockchain-based donations traceability framework." *Journal of King Saud University-Computer and Information Sciences* (2022).
- [24] Haque, A.K.M., Rahman, M., 2020. Blockchain Technology: Methodology, Application and Security Issues. *ArXiv Preprint ArXiv:2012.13366*.
- [25] Emmadi, N., Vigneswaran, R., Kanchanapalli, S., Maddali, L., Narumanchi, H., 2018. Practical deployability of permissioned blockchains. In *International Conference on Business Information Systems* (pp. 229–243). Springer.
- [26] Wickström, J., Westerlund, M., & Pulkkis, G. (2020). Rethinking IoT Security: A Protocol Based on Blockchain Smart Contracts for Secure and Automated IoT Deployments. *ArXiv*, abs/2007.02652.
- [27] <https://www.geeksforgeeks.org/solidity-mappings/>
- [28] <https://medium.com/chainlink-community/web3-nedir-9f989e2cc473>
- [29] <https://blog.bitnovo.com/en/what-is-a-dapp/>

ÖZGEÇMİŞ

Taha Uz, 2000’de İzmir’de doğdu. Lise öğrenimi Diyarbakır Anadolu Lisesi, Rekabet Kurumu Anadolu Lisesi, Ahmet Kanatlı Anadolu Lisesi ve Yeni Çizgi Temel Lisesi’nde görmüştür. Yeni Çizgi Temel Lisesi ile öğrenimini tamamlayarak mezun oldu. 2019 yılında girdiği Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü’nde 4. sınıf olarak okumaya devam etmektedir. Şu an kendisini blok zinciri, oyun geliştirme ve web3 alanlarında geliştirmektedir.