Blok Zinciri Teknolojisi ile Veri Korunumu Sağlanmış Oylama Uygulaması

A Voting Application Which Provides Data Security via Blockchain Technology

Yağız Gani
Bilgisayar Mühendisliği
İhsan Doğramacı Bilkent Üniversitesi
Ankara, Türkiye
yagizgani@hotmail.com

Mustafa Mert Aşkaroğlu
Bilgisayar Mühendisliği
İhsan Doğramacı Bilkent Üniversitesi
Ankara, Türkiye
mertaskaroglu@hotmail.com

***Öz*— Dijital oylama ve seçim uygulamalarında akla gelen en önemli endişe, verinin değiştirilmesi veya verinin silinmesidir. Bu endişeler düşünüldüğünde Blok Zinciri teknolojisi güven sorununu ortadan kaldırma ve verinin korunumunun artırılması konularında kendi ispatlamış teknolojilerin başında gelmektedir. Dolayısı ile Blok Zinciri teknolojisi kullanılarak hazırlanacak bir oylama veya seçim uygulaması endişelerin minimize edilmesi konusunda yardımcı olacaktır. Bu çalışmada veri değiştirilmesinin önüne geçmeyi hedefleyen Blok Zinciri tabanlı bir oylama uygulaması geliştirilmiştir.**

Anahtar Sözcükler— Blok Zinciri, Akıllı Sözleşmeler, Onaylama Prensipleri, Kayıt Defteri, Oylama Sistemi

***Abstract*— The main concerns which are coming to mind in the digital voting and election applications are data manipulation or deletion. When these concerns are considered, Blokchain is a foremost technology which proved itself on disposal of distrust and providing security of the data. Therefore, a voting or election system which is implemented with using Blockchain technology enable us to minimize the concerns which are stated above. In this study, a blockchain based voting application which aims to prohibit data manipulation is developed.**

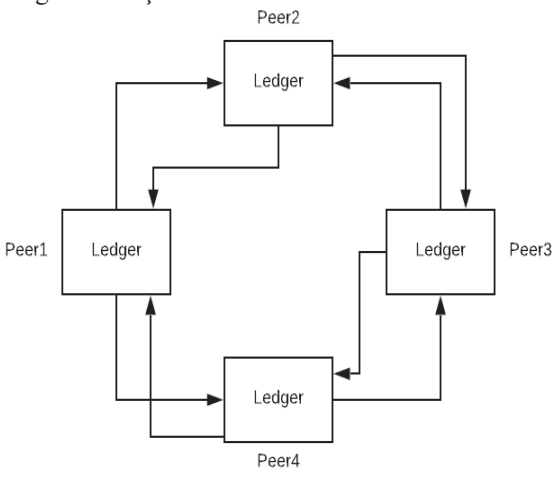
Keywords— Blockchain, Smart Contracts, Endorsement Policies, Ledger, Voting System

I. GİRİŞ

Son yıllarda insan gücüyle veya manuel olarak çalışan sistemlerin dijitalleştirilerek verimliliğinin artırılması yaygın olarak kullanılan işlemlerin başında gelmektedir. [2] Bununla beraber oylama ve seçim işlemleri için güvenli ve dijital bir platform oluşturmak, sonucun çabuk elde edilmesi, harcanan kaynak miktarının azaltılması ve paydaşların oylama sonucuna olan güveninin artırılması konusunda yardımcı olabilecek potansiyele sahiptir. Oylama ve seçim sistemlerinin dijitalleştirilmesi insan hatasının minimize edilebilmesi bakımından da gereklidir. Oylama sayım sürecine ne kadar çok insan dahil olursa oylama sonuçlarında kasıtlı veya kasıtsız olmak üzere hata oluşabilme riski de bir o kadar artmaktadır. Ancak, dijital oylama işlemlerinde de akla gelebilecek birkaç önemli endişe söz konusudur. Bunlardan en önemlileri; verinin değiştirilmesi, verinin silinmesi ve katılımcılarının hangi oyu kullandığının tespit edilebilmesi sorunudur. Bu sorunlar göz önünde bulundurulduğunda Blok Zinciri teknolojisi uygun bir çözüm olabilme potansiyeline sahiptir. Bu teknoloji 2008 yılında Satoshi Nakamato isminde bir geliştirici tarafından icat edilmiştir. Nakamato bu teknolojiyi kısa bir süre sonra icat ettiği Bitcoin'in altyapı teknolojilerinden birisi olarak kullanmıştır. Blok Zinciri Bitcoin'de üçüncü bir otorite olmadan güvenli bir şekilde ara transferinin yapılmasını sağlayan teknoloji olarak ortaya çıkmıştır**.** O günden sonra Blok Zinciri teknolojisinin popülaritesi git gide artmış ve diğer kullanım alanları araştırılmaya başlanmıştır. [3] Bu teknolojinin kullanımı finansal ve finansal olmayan alanlar üzerine ikiye ayırmak uygun olacaktır. [3] Ama tüm kullanım alanlarının ortak yönünün, bir güven sorununu ortadan kaldırmaya çalışmak olduğu aşikardır. Bu noktada Blok Zinciri teknolojisinin yapısı ve dijital oylama sistemindeki endişeleri nasıl çözebileceği diğer bölümlerde açıklanmıştır.

II. BLOK ZİNCİRİ GENEL AĞ YAPISI

Blok Zinciri tutulan işlem (transaction) kayıtların değiştirilemediği dağıtılmış bir sistem olarak tanımlanabilir. Bu kayıtlar sistemdeki her eş (peer) de bulunmakla beraber kayıt defterine (ledger) eklenecek olan işlemler, mutabakat algoritmalarına (consensus algorithms) bağlı olarak gerçekleştirilebilmektedir. [1] Dolayısıyla mutabakat algoritmaları dışında gerçekleştirilmiş her işlem, kayıtlarda tutulmasına rağmen geçersiz sayılacaktır. Bunun dışında Blok Zinciri ağ yapısı, merkezsiz (decentralized) bir sistem yapısına sahiptir. Bu nedenle blok zinciri tek bir hata noktasına bağlı olarak çökmez veya işlevsiz duruma düşmez. [1] Bunun ötesinde merkezsiz ağ yapısı, yetkilendirilme noktasının dağılmasına sebep olur. Blok Zinciri ağ yapısında yetkilendirilmiş tek bir nokta bulunmaz. Dolayısıyla sistem bir hedef ve saldırı noktası barındırmaz. Bu sayede belirli bir eşin ele geçirilmesi sistemde güvenlik zafiyeti oluşturmamaktadır. Blok Zinciri Ağı yapısı Şekil 1 de gösterilmiştir.



Şekil 1. Blok Zinciri Ağ Yapısı

978-1-7281-3964-7/19/$31.00 ©2019 IEEE

III. BLOK ZİNCİRİ TEKNOLOJİSİNİN TEMEL BİLEŞENLERİ VE OYLAMA SİSTEMİNE KATKILARI

A. Akıllı Sözleşmeler (Smart Contracts)

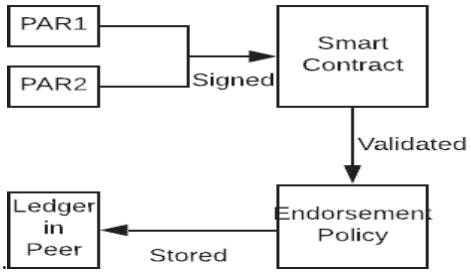
Akıllı sözleşmeler Blok Zinciri Ağında bir işlemin nasıl gerçekleşeceğinin belirtildiği kod parçalarına verilen addır. Akıllı Sözleşmeler, işlemlerin içinde gömülü olarak bulunur ve bir işlem (transaction) gerçekleştiğinde otomatik olarak aktive olurlar. [1] Akıllı Sözleşmeleri, paydaşlar arasında işlem gerçekleştirilmeden önce kararlaştırılmış prosedürler gibi düşünmek isabetli bir karar olacaktır. Buna bağlı olarak Akıllı Sözleşmeler, dijital oylama uygulamasının güvenilirliğini artırma konusunda önemli bir rol oynamaktadır. Çünkü herhangi bir seçim veya oylama etkinliğinde gerçekleştirilmesi gerekecek bazı prosedürlerin olacağı kesindir. Bu prosedürler genelde oylama işlemi gerçekleşmeden önce yazılı olarak görevlilere bildirilir ve bu prosedürlerin uygulanması görevlilerin kontrolüne bırakılmış olunur. Daha öncede belirtildiği gibi insan faktörü işin içine girdiği zaman kasıtlı veya kasıtsız olmak üzere hata riski artmaktadır. O yüzden gerekli prosedürlerin uygulanacağını garanti eden güvenilir bir sistem yaratılması bu hata olasılığının önüne geçmekte son derece yararlı olacaktır. Akıllı Sözleşmeler sayesinde prosedürler çalıştırılabilir bir kod parçası haline dönüştürülmüş olacak ve her işlem gerçekleştirildiğinde çalışarak prosedürlerin uygulanmasının garantisi verilmiş olacaktır.

B. Kayıt Defteri (Ledger)

Kayıt Defterleri Blok Zinciri ağında gerçekleşmiş olan her işlemin kaydını, değiştirilemez ve silinemez bir biçimde tutan işlem depolama yapısıdır. Kayıt Defteri kendi içinde iki çeşit yapı barındırır. Bir tanesi Durum Kaydı (World State), diğeri ise İşlem Tarihçesi (Transaction History) olarak adlandırılır. [4] Durum Kaydı bir objenin güncel değerlerini tutarken, İşlem Tarihçesi o obje ile gerçekleştirilmiş tüm işlemleri depolar. Örneğin Blok Zinciri tabanlı bir araba satış platformu düşünelim. Eğer bir arabanın boya rengi kırmızıdan maviye değiştirildiyse, Durum Kaydında objenin rengi mavi olarak kayıt altına alınır. Fakat İşlem Tarihçesine bakıldığında arabanın renginin önceden kırmızı olduğuna ve arabanın rengi kırmızıyken hangi işlemlerin yapıldığına ulaşım şansı olacaktır. Bu yapı, oylama sisteminde de büyük ölçüde önem arz etmektedir. Örnek olarak, eğer sistem üzerinde 'A' seçeneğine verilmiş bir oy, 'B' seçeneğine değiştirilmiş, yani müdahale edilmişse, bu oyun önceden 'A' seçeneğine verildiğinin takibi İşlem Tarihçesi sayesinde kolay bir şekilde yapılabilecektir. Bu yüzden Kayıt Defteri, sistem müdahalesinin tespitinde önemli bir destek sağlamaktadır.

C. Onaylama Prensipleri (Endorsement Policies)

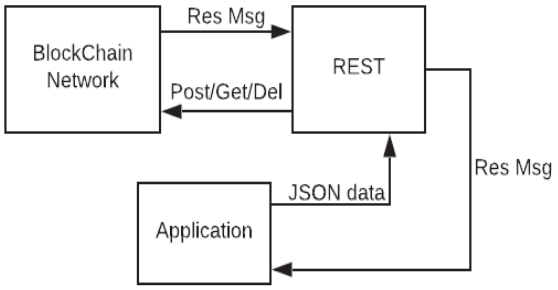
Onaylama Prensipleri bir işlemin (transaction) geçerli sayılabilmesi için ağ içinde eşlerden talep edilen yetkilendirmelere verilen isimdir. [3][4] Onaylama prensipleri zincir kodunda (chain code) önceden belirtilir ve zincir kodu aktive edildiğinde gerçekleşen işlemin geçerliliğini kontrol ederler. Örnek olarak bir zincir kodunun yazımı bittikten sonra zincir kodunu onaylayacak prensipler atanabilir. Mesela Akıllı Sözleşmelerin hangi eşin özel anahtarı ile (private key) imzalanması gerektiği bir onaylama prensibidir. Böylelikle bir işlemin gerçekleştirilmesi için hangi eşlerin (peer) anlaşması gerektiği belirtilmiş olunur. Onaylama Prensipleri doğru işlemin gerçekleşip gerçekleşmemesi ile ilgilenmezler, onların temel amacı gerçekleşen işlemin geçerli sayılıp sayılamayacağının kontrol edilmesidir. Onaylama Prensipleri, sistemde taklit işlemlerin veya taklit akıllı sözleşmelerin barındırılmasının önüne geçer, çünkü geçersiz işlemler sistemde tutulsa bile onaylanmadığı belirtilir. Bu sayede hem geçersiz işlem kayıtlarının takip edilmesinin olanağı sağlanır hem de sahte işlem girişimlerinin önüne geçilmiş olunur. Şekil 2'de bu sürecin nasıl gerçekleştiği gösterilmiştir.



Şekil 2. İşlem Kaydedilme Döngüsü

IV. SİSTEM YAPISI

Sistem, 3 ana parçadan oluşmaktadır; Android uygulaması, REST Sunucusu ve Blok Zinciri Ağı. Android uygulaması, kullanıcıdan gelen girdileri REST Sunucusuna aktarmaktadır. REST Sunucusu ise kullanıcıdan gelen girdilere göre Blok Zinciri Ağında gerekli işlemlerin oluşması için komutlar gönderir. Dolayısıyla, gelen veri REST Sunucu aracılığıyla, Blok Zinciri Ağına aktarılmaktadır. Yukarıda bahsedilen teknolojilerin özellikleri ve daha detaylı açıklamaları ilerleyen bölümlerde ilgili başlıklar altında anlatılmaktadır. Sistem yapısı Şekil 3'te gösterilmiştir.



Şekil 3. Sistem Yapısı Gösterimi

A. Blok Zinciri Ağ Kurulumu

Projede, Blok Zinciri Ağı kurmak için Hyperledger Fabric Servisi kullanılmıştır. Bu servis IBM tarafından piyasaya sürülmüş olunup, Blok Zinciri tabanlı iş uygulamaları geliştirmek isteyen yazılımcıların işini kolaylaştırmak amacıyla kullanılmaktadır. Hyperledger Fabric servisinin MacOS ve Ubuntu platform destekleri bulunmaktadır. [5] Fabric Servis'ini kullanmak için belirli teknolojilerin hali hazırda yüklü olması gerekmektedir. Bu teknolojiler Docker ve NodeJs servisleridir. [5] İlerleyen bölümlerde bu teknolojilerin ne oldukları ve ne için kullanıldıkları hakkında daha detaylı açıklamalar yer almaktadır.

- Docker: Docker programı, kurulan Blok Zinciri ağının çoklu platformlarda ortak bir şekilde çalışabilmesi için gereklidir. Fabric'i kurmadan önce, terminal yardımı ile kurulum gerçekleştirilmiştir. [5] Blok Zincirinin kurulumu ve kod yazımı sürecinde tüm çalışma bir Docker Konteyner içinde tutulmuştur. Bu sayede ortaya çıkacak olan Blok Zinciri Ağı tüm platformlarda sorunsuz bir şekilde çalışabilmektedir. [5]
- NodeJs Servisi: NodeJs servisi Blok Zinciri içerisindeki yapının değiştirilmesi ve yapının ihtiyaca göre düzenlenmesini sağlamaktadır. Fabric kurulumu yapılmadan önce NodeJs servisi yüklü bulunmak zorundadır. Terminal kullanılarak bu kurulum gerçekleştirilmiştir. [5]
- Fabric Kurulumu: Gerekli tüm öncül programlar başarı ile yüklendikten sonra, terminal kullanılarak gerekli komutun girilmesi ile Composer CLI araçları başarılı bir şekilde kurulmuştur. Bu komut aşağıda belirtilmiştir. [6]

  ```
  npm install -g composer-cli
  ```

  Composer CLI araçları, Blok Zinciri geliştiricileri için tüm önemli işlemlerin yapılmasını sağlayacak araçlardır. Fabric kurulumu da yapılarak sistem kurulumu başarı ile tamamlanmıştır. Bir sonraki bölümde, sistem tasarımı ve kod yazımı aşaması yer almaktadır. [6]

B. Blok Zinciri Ağı Yazım Süreci

Hyperledger Fabric'in yaratıcısı olan IBM firması Visual Studio Code editörünü önermektedir. [6] Bunun sebebi ise, Hyperledger Fabric kullanılarak Zincir Kodu, Modelleme ve Erişim Kontrolü yazma ortamının, Visual Studio Code'da bir eklenti olarak bulunmasıdır.

- IBM Modelleme Dili: Bu dil kullanılarak, Blok zincirinde yer alacak varlıklar (entity) ve konseptler oluşturulmaktadır. [1] Örneğin, bu çalışmada "Katılımcı" varlığı oluşturulmuş ve bu varlığın içinde yer alacak nitelikler eklenmiştir (katılımcı ismi, kimliği, adres, vs.). Örnek modelleme kodu aşağıda belirtilmiştir.

  ```
  participant Participant identified
  by participantId {
    o String participantId
    o SignupValues signupValue
    o Info info optional
    o Integer type
    o String Descr optional
  }
  ```

- Erişim Kontrol Dili: Bu kısımda bir eşin veya paydaşın, Blok Zinciri Ağında neye erişim izninin olup neye olamayacağı yazılmaktadır. [1] Örneğin normal bir kullanıcının, Yönetici kullanıcı tipinin yaptığı işlemlere erişiminin olmaması ve ilgili işlemleri (transaction) görmemesi kuralı burada belirtilmektedir.
- Zincir Kodları Yazımı: Zincir Kodları, Akıllı Sözleşmelerin yapısının oluşturulduğu kod parçalarına verilen isimdir. [1] Bu kodlar, JavaScript dili kullanılarak yazılmıştır. Örnek Zincir kodu aşağıda belirtilmiştir.

```
function increaseCoin(coinData){
    var participantRegistry = {}
    return
getParticipantRegistry('org.pollst
ar.participant.Voter').then(functi
on(registry){
        participantRegistry       =
registry;
        return
participantRegistry.get(coinData.p
articipantId);
    }).then(function(participant){
        participant.coin          =
participant.coin                  +
coinData.addition;
        return
participantRegistry.update(partici
pant);
    }).then(function(){
        // Successful update
        var         event         =
getFactory().newEvent('org.pollsta
r.participant', 'coinIncreased');
        event.participantId       =
coinData.participantId;
        emit(event);
    }).catch(function(error){
        throw new Error(error);
    });
}
```

C. Blok Zinciri Ağı Yönetimi

Ağ yönetimi işlemi birden fazla aşamadan oluşmaktadır. Sistemin düzgün çalışabilmesi için bu aşamalar eksiksiz ve doğru bir şekilde gerçekleştirilmiştir.

- Fabric Çalıştırılması: Yazılan tüm kodların başarılı bir şekilde çalışması ve sisteme yüklenmesi için öncelikle Fabric Sisteminin çalışıyor vaziyette bulunması gerekmektedir. Bunun için gerekli komutlar aşağıda belirtilmiştir. [6]

  ```
  # Start Server
  cd ~/fabric-dev-servers
  export FABRIC_VERSION=hlfv12
  ./startFabric.sh
  ```

- BNA Arşivi Yaratılması ve Yüklenmesi: Fabric'in arka planda çalışır vaziyette bulunduğundan emin olduktan sonra yazılmış olan tüm zincir kodları, model kodları, erişim kontrol kodları, onaylama prensipleri ve benzeri tüm blok zinciri ağı bileşenlerinin arşivlendiği bir dosya oluşturulması gerekmektedir. Bu dosya İş Ağı Arşivi (Business Network Archive) dosyası olup, yüklenecek olan

Blok Zinciri Ağının temel prensiplerini içermektedir. BNA dosyası yaratmak ve yüklemek için gerekli komutlar aşağıda belirtilmiştir.

```
# Create bna file
composer archive create --sourceType dir --sourceName ../ -a archive.bna
# Deploy BNA file
composer network install -a ./archive.bna -c PeerAdmin@hlfv1
```

- Admin Kartı Yaratma: Blok Zinciri Ağında 2 tip kart mevcuttur. Bunlar; Ağ Yönetici Kartı ve Eş Yönetici Kartıdır. Ağ Yönetici Kartı, Blok Zinciri Ağı oluştururken sistem tarafından otomatik olarak yaratılmaktadır. İş Ağı Yöneticisi, Blok Zinciri Ağı yüklendikten sonra, sistem yapısının ayarlanması ve güncellenmesinden sorumlu olan kişidir. [7] Eş Yönetici Kartı ise sisteme girecek tüm eşler için dinamik olarak yaratılmaktadır. Blok Zinciri yapısal olarak merkezi olmayan bir sistem olduğundan dolayı ağa yeni giriş yapan tüm eşler için kart yaratılmaktadır. Bu işlem için gerekli olan terminal komutu aşağıda verilmiştir. [6]

```
./createPeerAdminCard.sh
```

- Ağın Çalıştırılması: Tüm işlemler bittikten sonra Blok Zinciri Ağının çalıştırılması gerekmektedir. Bu işlem için gerekli terminal komutu aşağıda belirtilmiştir. [7]

```
composer network start -n networkName -c PeerAdmin@hlfv1 -V 0.0.1 -A admin -S adminpw
```

Sistem başarılı bir şekilde başlatıldıktan sonra, yaratılan Eş Kartı'nın sisteme eklenmesi gerekmektedir. Bunun için gerekli komut aşağıda verilmiştir.

```
# Use the card generated
composer card import -f admin@pollstarblockchainnetwork.card
```

Bu işlemler sonucunda sistem başarılı bir şekilde test edilebilir haldedir. Yaratılan Akıllı Sözleşmelerin (smart contract) hepsi artık sistemde hazır ve kullanılabilir hale gelmiştir. Eğer zincir kodunda herhangi bir hata oluşmuş ise, sistem hata verir ve kullanıcıyı uyarır. Buna göre kod düzenlenip yukarıdaki aşamalar tekrar uygulanarak önceden oluşturulan Blok Zinciri Ağının içine tekrar yüklenebilir.

- Sistemin Simülasyonu: Sistemin simülasyonu amacıyla kullanılan "Composer Playground" isimli bir servis bulunmaktadır. Bu servis Blok Zinciri geliştiricisine, yarattığı Akıllı Sözleşmelerin kullanımı ve simülasyonu konusunda kolaylık sağlar. [8] Örneğin, kullanıcının "Anket Oyla" adında bir akıllı sözleşme yarattığını farz edelim. Kullanıcı bu sözleşmeyi, gerekli parametreleri girerek test edebilir ve başarılı bir şekilde çalışıp çalışmadığını görebilir. Bu sayede sistemin işleyiş kontrolü sağlanmaktadır. Composer Playground servisi, kullanıcıya yukarıda bahsedilen işlemlerin kolaylıkla gerçekleştirilebilmesi için ara yüz sağlamaktadır. [8]

D. Loopback Servisi

Loopback servisi, açık kaynaklı bir NodeJS servisidir. [9] Sistemde halihazırda yüklü olan NodeJs, REST Sunucu için kullanıcı ara yüzü yaratabilmek üzere LoopBack adında bir servis sağlamaktadır. [10] Dolayısı ile Loopback kullanılarak REST Sunucu için ara yüz oluşturulmuştur. Bu sayede REST Sunucu kullanımı hızlı ve kolay hale gelmiştir. Visual Studio Code aracının konsolu kullanılarak Loopback Proje tipi oluşturulabilir ve gerekli tüm dosyalar Proje Klasörü içinde ayrı bir klasörde yaratılabilir. Loopback oluşturmak için gerekli terminal komutu aşağıda verilmiştir.

```
yo hyperledger-composer
```

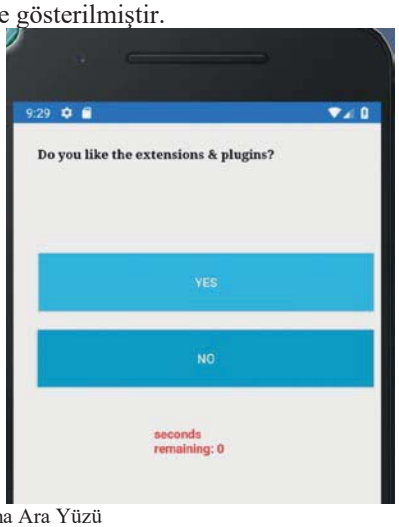
Bu klasöre erişim sağlandıktan sonra, REST Sunucusu'nu başlatmak için, başlatma komutu girilmiştir. Konsol çıktısı olarak, REST Sunucusunun URL'si yer almaktadır. [11] Bu linke tıklanarak REST Sunucusu ara yüzüne başarılı bir şekilde erişim sağlanmıştır.

E. REST Sunucusu

REST Sunucusu yardımıyla, Blok Zinciri Ağı ve Android uygulaması arasında veri alışverişi sağlanmıştır. Uygulama kısmında ise "Volley" adında Ağ Kütüphanesi kullanılarak REST Sunucu ile uygulama iletişimi sağlanmıştır. Bu iletişim, JSON veri formatı ile gerçekleştirilmiştir. Android kısmından gelen girdiler, JSON formatına dönüştürülerek REST Sunucusuna aktarılmıştır. REST içinde de gelen veriler kullanılarak Blok Zincirindeki gerekli işlemler aktive edilmiş ve işlem kayıtları oluşturulmuştur.

F. Kullanıcı Ara Yüzü

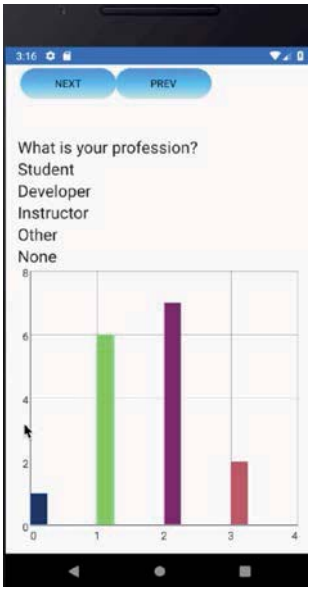
Kullanıcı ara yüzünden gelen girdiler REST Sunucusuna aktarılır. Bu girdiler Blok Zinciri Ağında bazı değişikliklerin olmasına sebep olacağı gibi sadece Durum Kaydındaki (World State) bilgilerin öğrenilmesi için de kullanılabilir. Örneğin kullanıcının oylama işlemini gerçekleştirmesi veya bir oylama sorusu yaratılması, Blok Zincirinde Durum Kaydının değişikliğini gerektirir. Bu girdiler arka planda JSON formatına dönüştürülür ve REST Sunucusuna "POST" mesajı olarak gönderilir. Oylama ara yüzü ekran görüntüsü Şekil 4'te, soru yaratma ara yüzü ekran görüntüsü de Şekil 5'te gösterilmiştir.



Şekil 4. Oylama Ara Yüzü

Şekil 5. Soru Yaratma Ara Yüzü

Bunun ötesinde kullanıcıdan gelen girdiler aynı zamanda Blok Zinciri Ağında olan verilerin gösterilmesiyle ilgili olarak da kullanılabilir. Kullanıcıdan gelen girdiler arka planda yine JSON formatına dönüştürülerek REST sunucusuna "GET" isteği şeklinde yollanır. Bu sayede Blok Zinciri Ağından istenen Durum Kaydı bilgisi uygulama ara yüzüne aktarılabilir. Bu işlemin gerçekleştiği ara yüzlerden biri olan sonuç ara yüzü ekran görüntüsü Şekil 6'da gösterilmiştir.



Şekil 6. Sonuç Ara Yüzü

V. SONUÇ

Bu çalışmada Blok Zinciri tabanlı bir dijital oylama platformu geliştirilmiştir. Bu çalışmanın temel motivasyonu, dijital bir oylama sisteminin akla getireceği endişelerin ve sorunların minimize edilmesidir. Yapılan bu çalışma sayesinde insanların daha güvenilir ve takip edilebilir bir oylama sistemine sahip olmaları ve oylama sonuçları açıklandıktan sonra oluşabilecek endişelerin ve kargaşanın önüne geçilmesi hedeflenmiştir. Blok Zinciri teknolojisinin güven sorunlarını çözmede ne kadar etkili bir teknoloji olabileceğini göstermek ve finans alanı dışındaki uygulamalara yenisi eklemek de ikincil hedefler arasındadır.

KAYNAKÇA

[1] Gaur, N., Desrosiers, L., Ramakrishna, V., Novotný, P., Baset, S. and O'Dowd, A. (2018). *Hands-on blockchain with Hyperledger*.

[2] Sciencesphere.org. (2017). [online] Available at: http://www.sciencesphere.org/ijispm/archive/ijispm-0501.pdf#page=67 [Accessed 27 May 2019].

[3] Crosby, M., Pattanayak, P., Verma, S. and Kalyanaraman, V. (2016). *BlockChain Technology: Beyond Bitcoin*. [online] Scet.berkeley.edu. Available at: http://scet.berkeley.edu/wp-content/uploads/AIR-2016-Blockchain.pdf [Accessed 27 May 2019].

[4] Hyperledger-fabric.readthedocs.io. (2019). *Smart Contracts and Chaincode — hyperledger-fabricdocs master documentation*. [online] Available at: https://hyperledger-fabric.readthedocs.io/en/release-1.4/smartcontract/smartcontract.html [Accessed 1 Jun. 2019].

[5] "Installing pre-requisites | Hyperledger Composer", *Ibm-blockchain.github.io*, 2019. [Online]. Available: https://ibm-blockchain.github.io/develop/installing/installing-prereqs.html#macos. [Accessed: 02- Jun- 2019]

[6] "Installing the development environment | Hyperledger Composer", *Ibm-blockchain.github.io*, 2019. [Online]. Available: https://ibm-blockchain.github.io/develop/installing/development-tools.html. [Accessed: 02- Jun- 2019]

[7] "Deploying Business Networks | Hyperledger Composer", *Hyperledger.github.io*, 2019. [Online]. Available: https://hyperledger.github.io/composer/v0.19/busin ess-network/bnd-deploy.html. [Accessed: 03- Jun- 2019].

[8] "Using Playground | Hyperledger Composer", *Hyperledger.github.io*, 2019.[Online]. Available: https://hyperledger.github.io/composer/v0.19/playg round/playground-index. [Accessed: 03- Jun- 2019].

[9] "LoopBack - Node.js framework", *Loopback.io*, 2019. [Online]. Available: https://loopback.io/. [Accessed: 03- Jun- 2019].

[10] "Generating a REST API | Hyperledger Composer", *Hyperledger.github.io*, 2019. [Online]. Available: https://hyperledger.github.io/composer/v0.19/integr ating/getting-started-rest-api. [Accessed: 03- Jun- 2019].

[11] "Publishing events from the REST server | Hyperledger Composer", *Hyperledger.github.io*, 2019. [Online]. Available: https://hyperledger.github.io/composer/v0.19/integr ating/publishing-events. [Accessed: 03- Jun- 2019]