Avrupa Bilim ve Teknoloji Dergisi
Sayı 27, S. 658-664, Kasım 2021

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
No. 27, pp. 658-664, November 2021

Research Article

Çapraz-Platform ile Gerçek Zamanlı Bulut Veritabanı iletişimi: Bütünleşik Ev Sistemi

Ergin Tosunoğlu[1*], Ahmet Berk Üstün[2].

[1*] Bartın Üniversitesi, Fen Fakültesi, Bilgisayar Teknolojisi ve Bilişim Sistemleri Bölümü, Bartın, Türkiye, (ORCID: 0000-0002-4345-1173), ergintosunoglu@gmail.com
[2] Bartın Üniversitesi, Fen Fakültesi, Bilgisayar Teknolojisi ve Bilişim Sistemleri Bölümü, Bartın, Türkiye, (ORCID: 0000-0002-1640-4291), ustun.ab@gmail.com

(İlk Geliş Tarihi 3 Mart 2021 ve Kabul Tarihi 7 Ekim 2021)

(**DOI:** 10.31590/ejosat.890291)

ATIF/REFERENCE: Tosunoğlu, E., Üstün, A.B., (2021). Çapraz-Platform ile Gerçek Zamanlı Bulut Veri Tabanı iletişimi: Bütünleşik Ev Sistemi. *Avrupa Bilim ve Teknoloji Dergisi*, (27), 658-664.

Öz

Mobil uygulamaları daha hızlı, daha kolay ve daha ucuz geliştirebilmek için yeni yöntemler ve araçlar kullanılmaya başlanmıştır. Bu bağlamda, Çapraz Platform (Cross-Platform) ile ayrı ayrı uygulama geliştirmek yerine tek bir uygulamanın farklı işletim sistemlerine kolaylıkla uyarlanabilmesini sağlamaktadır. Xamarin köklü programlama dilleri arasında yer alan C# ile Çapraz Platform uygulamalar geliştirmemize imkân vermektedir. Ayrıca, günümüzde bulut teknolojileri, bilişim sektörüne yeni bir boyut kazandırmış, bu durum geliştiricilere kendi veri tabanı sistemlerini kurmak yerine bulut hizmetlerini veri tabanı olarak kullanmaya yönlendirmiştir. Firebase bulut veri tabanı yapısı, kullanıcılara gerçek zamanlı veri ve depolamayla iOS, Android, Windows ve hatta web tabanlı uygulamalar arasında iletişim ortamını sağlamaktadır. Özellikle Nesnelerin İnterneti (Internet of Things) uygulamalarının eşzamanlı olarak verimli bir şekilde çalışabilmesi için gerçek zamanlı veri haberleşmesi son derece hayatidir. Son yıllarda, Nesnelerin İnterneti yapılarının mobil uygulamalarla kontrolü ile farklı ölçek ve hedeflere sahip akıllı nesneleri birbirine bağlayan birçok çözüm üretilmiştir. Bu çalışmada, akıllı bir ev sistemini oluşturan farklı nesneler tek yapı haline getirilerek bütünleşik bir sistem oluşturulmuştur. Xamarin çapraz platform kullanarak tasarlanan mobil uygulama ile bulut veri tabanının gerçek zamanlı veri özelliğini kullanarak bütünleşik sisteminin kontrolünü; IOS, Android ve Windows platformlarda çalışabilecek ortak bir ara yüz tasarımı ve arka plan kod yapısı ile gerçekleştirmiştir.

Anahtar Kelimeler: Nesnelerin interneti, Bütünleşik ev sistemi, Çapraz platform, Xamarin, Firebase, Gerçek zamanlı iletişim.

Real-Time Cloud Database communication with Cross-Platform: Converged Home System

Abstract

The use of new methods and tools have been begun in order to develop mobile applications faster, easier and cheaper. Cross-Platform makes an application possible to be easily adapted to different operating systems instead of developing applications separately. Xamarin enables us to build Cross-Platform applications with the rooted programming language of C#. In addition, today's cloud technologies have brought a new dimension to the IT sector and this has directed developers to utilize cloud services as a database instead of implementing their own database systems. Firebase cloud database structure offering real-time data and storage provides users with a communication environment among iOS, Android, Windows and even web-based applications. Specifically, real-time data communication is necessary to simultaneously run the applications of the Internet of Things (IoT) in an efficient manner. In recent years, many solutions that connect smart objects having different scales and targets have been developed by controlling the Internet of Things structures with mobile applications. In this study, an integrated smart system was built by combining different objects that form a smart home system unified into a single structure. The control of the mobile application designed using Xamarin Cross-Platform and the integrated system using the real-time data feature of the cloud database has been accomplished with a joint interface design and background code structure that runs on IOS, Android and Windows operating systems.

Keywords: Internet of things, Integrated home system, Cross platform, Xamarin, Firebase, Real-time communications.

[*] Sorumlu Yazar: ergintosunoglu@gmail.com

1. Giriş

Günümüzde mobil cihaz kullanımındaki artışla beraber her alanda kullanılmaya başlayan mobil uygulamalar, kullanım alanı zenginliği ve platform çeşitliliği yönünden kullanıcı ihtiyaçlarına önemli çözümler sağlayabilmektedir. Mobil uygulamalar günlük hayatımızda haberleşmeden iletişime, nesnelerin interneti (Internet of Things) ile yaşamsal aktivitelerimizi kolaylaştırmaktan çalışmalarımızın verimliliğini arttırmaya kadar pek çok alanda karşımıza çıkmaktadır. Ancak bu uygulamaların geliştirilmesinde Android, IOS, Windows gibi farklı işletim sistemleri için farklı geliştirme ortamları ve geliştirme dilleri kullanılmaktadır. Bu platformlar birbirinden önemli ölçüde farklı olduğundan, geniş bir kullanıcı kitlesine ulaşmak isteyen yazılım geliştiricilerinin uygulamalarını her platform için ayrı ayrı geliştirmeleri gerekir. Bu durum uygulamanın geliştirme maliyetlerini artırmakta, bakım ve güncelleme ihtiyaçlarını zorlaştırmakta ve zaman kaybına neden olmaktadır.

Mobil uygulamalar önemini artırdıkça, mobil uygulamaların geliştirme yöntemleri de değişime uğramıştır. Mobil uygulamaları daha hızlı, daha kolay ve daha ucuz geliştirebilmek için yeni yöntemler ve araçlar kullanılmaya başlanmıştır. Çapraz platform (cross-platform) adı verilen bu araçlar, her işletim sistemi (IOS, Android, Blackberry ve Windows gibi) için ayrı ayrı uygulama geliştirmek yerine tek bir uygulamanın farklı işletim sistemlerine kolaylıkla uyarlanabilmesi özelliği ile çıkmışlardır (Xanthopoulos & Xinogalos, 2013). Platformlar arası geliştirme yaklaşımları, geliştiricilerin uygulamalarını bir dizi platform için tek adımda uygulamalarına izin vererek, farklı derleme yapılarından kurtulmalarına ve üretkenliğin artmasına olanak sağlamıştır (Heitkötter, Hanschke & Majchrzak, 2012).

Çapraz platform uygulama geliştirme araçlarından biri olan Xamarin; bize köklü programlama dilleri arasında yer alan C #' ile iOS, Android ve WindowsPhone için mobil uygulamaları geliştirme imkânı vermektedir. Java ve Objective-C yerine mobil uygulamalar geliştirmek için Xamarin'i seçmenin birçok avantajı bulunmakta olup; platformlar arasında kod paylaşabilmekte, C # ve .NET temel sınıf kitaplıklarının gelişmiş dil özelliklerinden yararlanarak daha verimli uygulamalar geliştirilebilmektedir (Peppers, 2015). Tek kod ile çoklu işletim sistemi (Android, iOS ve Windows Phone) için uygulama geliştirilmesine imkan vermektedir. Microsoft bünyesinde olması sebebi ile destek sistemi güçlüdür, köklü bir dil olan C# kullanıcı sayısı sayesinde, kaynak bulma ve derlenmiş kodlara ulaşma imkânı bulunmaktadır (Atkinson, 2016). Visual Studio rahat bir geliştirme ortamı sunmasının yanında geniş kütüphane yelpazesine sahiptir ve geliştiricilere kendi kütüphanelerini yazma imkânını da vermektedir (VisualStudio, 2020).

Şekil1'deki Diyagramda, platformlar arası bir Xamarin uygulamasının genel mimarisi gösterilmektedir. Xamarin, her platformda yerel kullanıcı arabirimi oluşturmanızı ve C# dili ile platformlar arasında paylaşılan uygulamalar yazmanızı sağlar. Xamarin bu işlemi yapabilmek için C# derleyicisi içerisinde Android, Windows ve IOS SDK'larını barındırır. C# ile yazılan kodlar bu SDK'lar üzerinden istenen platforma uygun hale getirilir. Xamarin'i diğer platformlardan ayıran önemli bir özelliği de; uygulamanın C# kodunun, dağıtmakta olduğunuz işletim sisteminin (Android, iOS, WindowsPhone) koduna derlenmesi yapılırken bazı optimizasyon işlemlerinden geçirilmesidir. Bu işlemin sonucu oluşturulan çıktı, geliştiricilerin yaptığı bazı temel hataları da ortadan kaldırdığı için çoğunlukla Native ortamda yazılan kodlara göre performansı daha yüksektir (Hermes, 2015).

Şekil 1. Xamarin çalışma diyagramı (Xamarin work diagram)

Mobil uygulama hangi platform için tasarlanırsa tasarlansın günümüzdeki mobil uygulamaların büyük bölümü veri tabanlarına ve örgütlenmemiş verilere giderek daha fazla bağımlı hale gelmiştir. İlişkisiz Veri Tabanı Yönetim Sistemleri (RDBMS) ile yapılandırılmamış verilerin işlenmesi oldukça zor olup, geleneksel veri tabanları ile yapılan işlemler artı maliyet oluşturmakta ve zaman kaybına neden olmaktadır (Khawas & Shah, 2018). Günümüzde bulut teknolojileri, bilişim sektörüne yeni bir boyut kazandırmıştır. Bu boyut geliştiricilere, kendi veri tabanı sistemlerini kurmak için büyük miktarda yatırım yapmak yerine ücretsiz veya karşılanabilir miktar da ödeme yaparak bulut hizmetlerini veri tabanı olarak kullanmaya yönlendirmiştir. Bulut teknolojileri sanal ortamda barındırılan uygulamaların ya da verilerin internet ile her yerden erişilmesine imkân vererek, başka yöntemler ile uzun sürelerde gerçekleşen veri işlemlerinin olabilecek en kısa sürede gerçekleştirmesini kolaylaştırmıştır (Ataç & Akleylek, 2019). Maliyet tasarrufu, daha hızlı uygulama performansı ve gerçek zamanlı veri transferi sağlamaları bulut veri tabanlarının (Cloud-Databases) mobil uygulama geliştiricileri tarafından tercih edilmesini yaygınlaştırmıştır (Al Shehri, 2013).

Google tarafından geliştiricilerin hizmetine sunulan Firebase, bulut tabanlı bir veri tabanı yapısıdır. Firebase büyük ve karmaşık yapıdaki verileri saklamak ve hızlıca sorgulamak amacıyla geliştirilmiş doküman tabanlı Nosql veri tabanıdır. Firestore esnek bir yapıya sahiptir. Veriler istenilen şemada tutulabilir. Veriler dokuman (document) adı verilen yapılarda saklanır. Dokuman topluluğuna ise kolleksiyonlar (collections) adı verilmektedir. Bu yapısı itibariyle diğer Nosql veri tabanlarına benzemektedir (FirebaseGoogle, 2020)

Şekil 2' de Firebase yapısının sağladığı hizmetler gösterilmiştir. Firebase, gerçek zamanlı veri ve depolamayla iOS, Android, Windows ve hatta web tabanlı uygulamalar oluşturmak için kullanılabilmekte ve yazılım geliştiricilerinin kullanabileceği farklı ürünlerde sunmaktadır (Stonehem, 2016). Firebase'in öne çıkan birkaç özelliği; uygulama yönetimi, kullanıcı etkinlikleri, veri depolama ve bildirim gönderimi olarak sıralanabilir.

Şekil 2. Firebase hizmetleri (Firebase services)

Firebase Authentication hizmeti, bir uygulama üzerinden kimlik doğrulamasını onaylamak için arka uç hizmetleri, kullanımı kolay SDK'lar ve anlık kullanıcı arabirimi kitaplıkları sağlar. Google, e-posta/şifre, telefon, Play Oyunlar, Facebook, Twitter, GitHub, Anonim oturum açma sağlayıcılarıyla yetkilendirilen alanlar için kullanıcılar sunucu tarafı kodu olmadan doğrulanabilmekte ve yönetilebilmektedir (Moroney, 2017a).

Firebase Cloud Messaging; Android, IOS ve web uygulamaları için mesaj ve bildirim gönderimi sağlayan platformlar arası bir bulut bildirim hizmetidir. Uygulama geliştiricilerinin, sunucu ve aygıtlar arasında iOS, Android ve web üzerinde ücretsiz olarak mesaj ve bildirim gönderip almasına olanak tanımaktadır (Moroney, 2017b).

Firebase Real-time Database; gerçek zamanlı veri tabanı işlemlerini bulut ortamında barındırılan bir veri tabanıdır. Veriler JSON olarak depolanır ve ilişkili her istemciyle sürekli olarak senkronize edilir. IOS, Android ve JavaScript SDK'larıyla herhangi bir platformlar arası uygulama geliştirildiğinde, kullanıcının talebinin büyük kısmı bir gerçek zamanlı veri tabanı örneğine dayanır ve bu örnek her yeni veriyle güncellenir. Bu özellik, geliştiricilerin bir veri tabanı geliştirme adımını atlamasına olanak tanır ve Firebase uygulamalar için arka uçta verilerin çoğunu işler (Chatterjee vd., 2018)

Günümüzde pek çok uygulama gerçek zamanlı (Real-time data) veri haberleşmesine ihtiyaç duymaktadır. Özellikle nesnelerin interneti (IoT) uygulamalarının eşzamanlıolarak çalışabilmesi için gerekli tüm veri; minimum veri kaybı ile hızlı bir şekilde iletilmelidir (Swahadika, Besari & Wibowo, 2019) IoT uygulamaları her geçen gün hayatımızın pek çok alanında kendisini göstermektedir.

Son yıllarda, ısıtma sisteminin, aydınlatmanın veya güvenlik sisteminin mobil uygulamalarla kontrolü gibi farklı ölçek ve hedeflere sahip akıllı nesneleri birbirine bağlamak için birçok çözüm ortaya çıkmıştır (Mineraud vd. 2016). Bu uygulamaların büyük çoğunluğu tek bir nesnenin kontrolü kurgusu üzere yapılmaktadır. Daha geniş bir ölçekte, birden çok sistemin tek bir uygulama ile kontrolünün gerçekleştirilmesi, bütünleşik bir IoT sağlayarak; sistemin bakımını, yeniden kalibrasyonunu kolaylaştıracak ve yeni IoT çözümlerinin gelişimine ve yönetimine fayda sağlayacaktır (Tsai vd. 2013). IoT sistemlerinde, tüm nesnelerden bilgi toplayan, onlarla iletişim halinde olan ve nesnelerin kullanılma derecesine göre sunucuyla veya doğrudan kullanıcıyla iletişim kuran bir yapı tasarlamak çok önemlidir (Dener, 2019).

Bütünleşik bir sistem tasarlamanın bazı karmaşık yönleri ve zorlukları bulunmaktadır. Her platform için ayrı kodlama yapılması ve iletişim protokollerinin ayrı ayrı tasarlanması hem zaman kaybına hem de maliyetin artmasına neden olabilmektedir. Diğer bir durum ise uygulama ile sistem arasındaki iletişimi sağlayacak yapının platformlar tarafından desteklenmesi gereksinimidir (Tao vd.,2019). IoT sistemleri için ön uç ve kullanıcı ara yüzlerinin geliştirilmesinde çapraz platform uygulamaları önemli bir rol oynayacaktır. Nihai uygulayıcılar tarafından çözümlerin kabulünde, platform farklılığının bir önemi olmayıp, istedikleri bütünleşik bir IoT yapısını her tür mobil cihazla izlemek veya kontrol edebilmektir (Umuhoza, 2017).

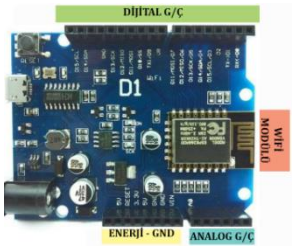
Bu çalışmada, Xamarin Çapraz-Platform kullanarak oluşturulan mobil uygulama ile bulut veri tabanının gerçek zamanlı veri özelliğini kullanarak bütünleşik bir ev sisteminin kontrolünü; IOS, Android ve Windows platformlarda çalışabilecek ortak bir ara yüz tasarımı ve arka plan kod yapısı ile gerçekleştirebilmek amaçlanmıştır. Bütünleşik bir IoT sistemin çapraz platform bir mobil uygulama ile çalışabilirliğinin sağlanmasının; her platform için ayrı ayrı ara yüz oluşturma, arka plan kod bloğu yazma ve iletişim protokolü oluşturma ihtiyacını ortadan kaldıracağı, bu sayede hem IoT hem de mobil uygulama boyutunda yeni çözümlerin gelişimine ve yönetimine fayda sağlayacağı, sistem bakımlarını ve güncellemelerini kolaylaştıracağı düşünülmektedir. Oluşturulan sistem birden fazla konutun veya iş yerinin tek bir uygulama üzerinden kontrolünede olanak sağlayaktır.

2. Sistem Yapisi (System Structure)

Mobil uygulama ile gerçek zamanlı çalışan bir sistemin izleme ve kontrol işlemlerinin tasarımı üç aşamalı olarak gerçekleştirilmiştir. Bunlar; sistemin kontrolünü ve ortam değerlerinin ölçülmesini sağlayan elektronik devre, elektronik devrenin kontrolünü ve izlenmesi sağlayan mobil uygulama ile bu iki unsur arasındaki iletişimi sağlayacak olan gerçek zamanlı Firebase veri tabanının yapılandırılmasıdır.

2.1. Elektronik Devre (Electronic Circuit)

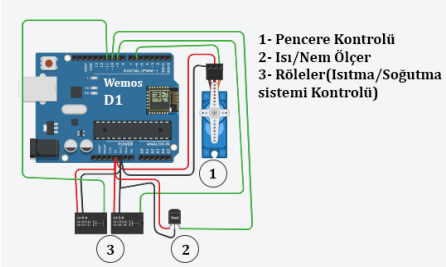
Elektronik devrenin kontrol ünitesi olarak; kendi üzerinde kablosuz haberleşme modülü bulundurması, analog ve dijital giriş/çıkış sistemlerini barındırması, düşük maliyetine göre yüksek performansa sahip olması nedeniyle Arduino-Wemos-D1 kullanılmıştır. Şekil3 ‘de Wemos-D1 genel yapısı görünmektedir.



Şekil 3. Wemos-D1 yapısı (Wemos-D1 structure)

Ortama ait sıcaklık ve nem bilgileri, dijital yapıya sahip DHT22 sensoru kullanılarak ölçülmektedir. Isıtma ve soğutma sistemlerinin aç/kapat işlemi için SDR-5VDC(10A-220V) röle kullanılmıştır. Pencerenin dereceli olarak açılasını ve

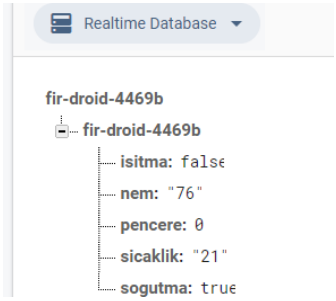
kapanmasını sağlamak için servo kullanılmıştır. Devrenin bağlantı tasarımı Şekil 4' deki gibidir.



Şekil 4. Bağlantı şeması(Connection scheme

2.2. Firebase Bulut Veri Tabanı (Firebase Cloud Database)

Verilerin hem mobil uygulamada hem de elektronik devre üzerinde anlık olarak işlenmesi için Firebase bulut veri tabanının Realtime özelliği kullanılmıştır. Çalışma için oluşturulan Firebase Realtime veri tabanı yapısı Şekil 5'de verilmiştir.



Şekil 5. Firebase Realtime veri tabanı yapısı

(Firebase realtime database structure)

Çalışma için kullanılan Firebase veri tabanı, hem mobil uygulama tarafında hem de elektronik devre tarafında anlık olarak okuma ve yazmaya açık olarak uyarlanmıştır. Bunun için Realtime özelliğinin kurallar dizini içerisinde okuma/yazma kurallarının aktif halde bulunması gerekmektedir.

Oluşturulan veri tabanı yapısı çift yönlü çalışacak şekilde tasarlanmıştır. Elektronik devreden gelen veriler, veri tabanımıza işlenerek mobil uygulama tarafından izlenmektedir. Mobil uygulama tarafından gelen veriler Firebase üzerinden elektronik devreye aktarılmaktadır. Bu yapı sayesinde internet erişiminin bulunduğu her noktadan sistemi gerçek zamanlı olarak kontrol etmek ve izlemek mümkün olmaktadır.

2.3. Mobil Uygulama (Mobile Application)

Geliştirilen mobil uygulama; Xamarin üzerinde C# programla dili kullanılarak çapraz platform olarak tasarlanmıştır. Uygulamanın Firebase veri tabanı ile bağlantı kurabilmesi ve işlem yürütebilmesi için haberleşme ve iletişim kodlarını içeren bir Firebase Kütüphanesi (Nuget) çapraz platforma eklenmiştir. Burada dikkat edilmesi gereken husus, kullanılan kütüphaneye göre veri tabanına erişim ve işlem için gerekli kod yapıları farklılık gösterebildiği gibi desteklenen platformlarda farklılık gösterebilmektedir.

Çalışmada Android, IOS ve Windows işletim sistemleri tarafından da desteklenen, Firebase REST API' sinin üzerine kurulmuş, Firebase Realtime Database için gerekli mimariyi de içeren "FirebaseDatabase.net (Version=4.0)" kütüphanesi kullanılmıştır.

Uygulamanın ilk aşamasında eklemiş olduğumuz "FirebaseDatabase.net" nuget kodları kullanılarak veri tabanı ile çapraz platform arasındaki iletişim köprüsünün kurulması sağlanmıştır.

```
DatabaseReference database = FirebaseDatabase
    .GetInstance("https://fir-          .firebaseio.com/")//Firebase Proje Adresiniz
        .GetReference("fir          ");    //Veritabanı Kodunuz
```

Şekil 6. Firebase İletişim kodu

Geliştirilen projede kullanılan iletişim kodu yapısı Şekil 6' da verilmiş olup bu kod yapısı Firebase üzerinde oluşturulan Realtime-Database projemizin altındaki veri setine ulaşmamızı sağlayacaktır.

Veri seti içerisine mobil uygulama ile veri gönderme işlemi için iki metot kullanılmıştır. Bunları senkron ve asenkron olarak sınıflandırmak mümkündür. Mobil uygulamada ısıtma ve soğutma sistemlerinin açma/kapama işlemleri bir buton yardımı ile gerçekleştiğinden eylemin gerçekleşme durumu göz önüne alınarak şekil 7'deki asenkron kod yapısı kullanılmıştır. Ancak penceremizin açılma ve kapanma eylemi bir bar ile gerçekleşmekte olup, barın anlık olarak aldığı değerlerin veri setine anlık olarak işlenmesi Şekil 8'deki senkron kod yapısı ile gerçekleştirilmiştir.

```
var myRef = database.Child("isitma");
await myRef.SetValueAsync(switch1.Checked);
```

Şekil 7. Asenkron kodlama(Asynchronous coding)

```
var myRef = database.Child("pencere");
myRef.SetValue(e.Progress);
```

Şekil 8. Senkron kodlama (Synchronous coding)

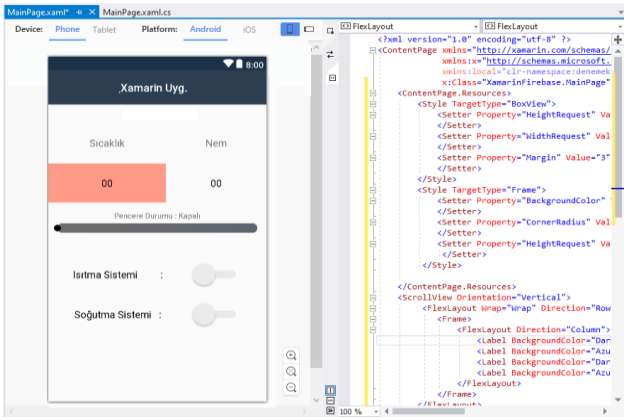
Kullanıcının mobil uygulama üzerinde bütünleşik akıllı sistem ile ilgili tüm değerleri ve anlık değişimleri takip edebilmesi için Firebase üzerindeki değişimleri izlemesi sağlanmıştır. Bunun için Şekil 9'da verilmiş olan kod yapısı kullanılmıştır.

```
var data_list = database.Child("fir-          ") // Veri dosya adı.
        .AsObservable<degerler>().AsObservableCollection<>();
return data_list;
```

Sekil 9. Veri tabanı izleme kod bloğu

(Database tracking code block)

Kullanılan ObservableCollection sınıfı kod bloğu ile Firebase veri tabanımızın üzerindeki işlemler izlenmektedir. Bu işlem sürekli olarak veri tabanına bağlı kalmamızı gerektirmeyerek arka planda çalışan kod bloğunun iş yükünü azaltmaktadır. Veri setinde bir değişiklik oluştuğunda ObservableCollection metodun oluşturduğu liste güncellenerek arka plan olaylarının tetiklenmesini ve ilgili kod yapılarının çalışmasını sağlamaktadır.



Şekil 10. Uygulama arayüzü

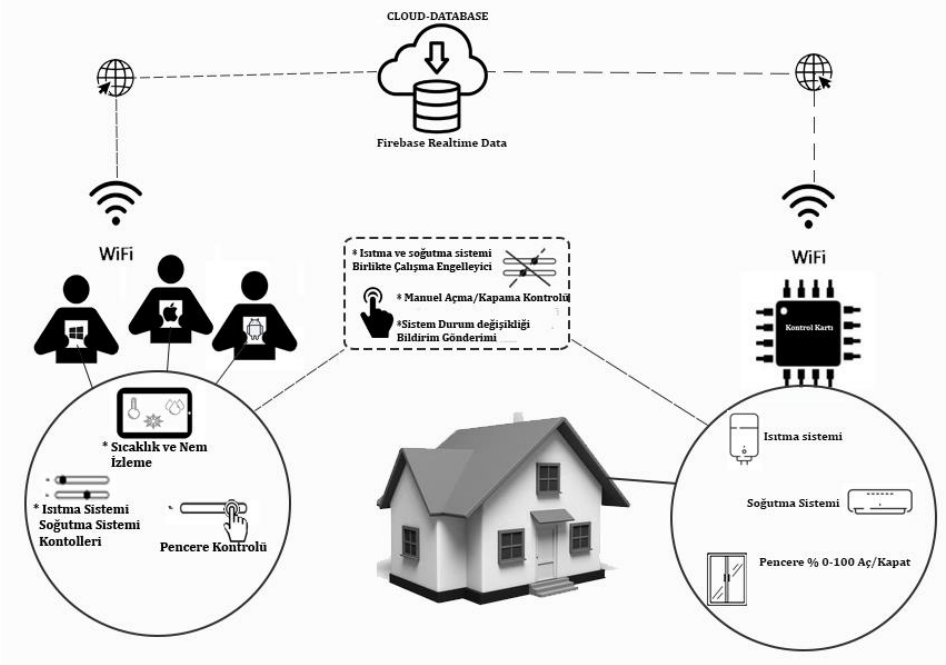
(Application interface)

Mobil uygulamanın ara yüz tasarımı Şekil 10'da gösterilmiştir. Ara yüzün tasarımında Visual Studio'nun içerdiği nesnelerin kullanılabiliyor olması Xamarin tarafında tasarımı kolaylaştırmıştır. Ayrıca Xamarin yapısında ara yüz tasarımı kodları XAML tabanlıdır. XAML, mobil uygulama geliştiricilerin kod yerine işaretleme kullanarak Xamarin. Forms uygulamalarında kullanıcı arabirimleri tasarımı için kolaylık sağlamıştır

3. Sistem Mimarisi Ve Çalişmasi (System Architecture and Its Work)

Oluşturulan IoT sistemin çalışma mimarisi Şekil 11'de verilen şemada gösterilmiştir. Mobil uygulama çapraz-platform olarak tasarlanmış olup, Android, IOS ve Windows sistemler üzerinden çalışabilmektedir. Sistemin veri aktarımları internet üzerinden gerçekleştirildiğinden, mobil uygulamanın bağlantıya eriştiği her durumda konum gözetilmeksizin kontrol ve izleme sağlanmaktadır.

Elektronik sistem, ortamın sıcaklık ve nem değerlerini ölçerek internet üzerinden Firebase veri tabanına göndermekte, aynı zamanda sistem içerisindeki ünitelerin durum bilgisini de veri tabanı üzerine işlemektedir. Mobil uygulama veri tabanındaki değişiklikleri gerçek zamanlı olarak okumakta ve bu verileri kullanıcıya bildirmektedir. Elektronik sistem mobil uygulama tarafından gönderilen işlem isteklerini anlık olarak veri tabanı üzerinden okuyarak ünitelerin kontrolünü sağlamaktadır.



Şekil 11. Sistem mimarisi (System architecture)

Kullanıcılar mobil uygulama üzerinden ortamdaki ısıtma ve soğutma sistemlerini kontrol edebilmekte, pencerelerin istenilen derecede açılmasını/kapanmasını sağlayabilmektedir. Isıtma veya soğutma sistemleri manuel olarak veya diğer bir kullanıcı tarafından açılır veya kapatılır ise sistemlerin durumundaki bu değişim veri tabanında anlık olarak işlendiğinden sistem durum değişikliği mobil uygulama tarafından algılanarak kullanıcılara ekran üzerinde bilgilendirme mesajı iletilmektedir.

Geliştirilen mobil uygulama; ısıtma veya soğutma sistemleri çalıştırılmak istendiğinde, mevcut durumlarını veri tabanı üzerinden kontrol etmektedir. Isıtma ve soğutma sistemlerinden birinin devrede olması durumunda her iki sistemin aynı anda devreye alınmasını engelleyerek, kullanıcıya çalışan sistem bilgisini ekran üzerinde bilgilendirme mesajı olarak göstermektedir.

4. Sonuç ve Tartışma (Conclusion and Discussion)

Bu çalışmada, akıllı bir ev sistemini oluşturan farklı nesneler tek yapı haline getirilerek bütünleşik bir akıllı sistem oluşturulmuş ve tasarlanan bütünleşik akıllı ev sisteminin kontrolü; Xamarin üzerinde çapraz-platform bir mobil uygulama geliştirilerek Firebase bulut veri tabanı ile gerçek zamanlı olarak gerçekleştirilmiştir. Geliştirilen uygulama çapraz-platform olarak yapılandırıldığından tek bir ara yüz tasarımı ve arka plan kod yapısı kullanılarak IOS, Android ve Windows işletim sistemleri üzerinde çalışabilirliği sağlanmıştır. Bu sayede her işletim sistemi için ayrı ayrı ara yüz tasarlama, kod bloğu yazma ve iletişim protokolü oluşturma ihtiyacı ortadan kalkmıştır. Kontrolü gerçekleştirilecek IoT sistem bütünleşik bir yapıya sahip olup sıcaklık ve nem ölçümleri, ısıtma, soğutma ve havalandırma sistemlerinin kontrolü platform farkı olmaksızın yapılabilmiştir

Mobil uygulamaların birden fazla platformda geliştirilmesi beraberinde bazı sorunları getirmekte ve bu sorunların platformlar arası farklılıklara bağlı sebeplerden kaynaklandığı görülmektedir. Bu nedenle, mobil uygulama geliştiricileri uygulamalarını farklı bir platforma taşımaya karar verdiklerinde veya diğer platformlar ile uyumluluk yakalamak istediklerinde, sorunlarla karşılaşma olasılığı oldukça yüksektir (Aljedaani vd. 2019). Çalışmada oluşturulan mobil uygulama ile platformlar arası farklılıklardan kaynaklı sorunlar Xamarin'in sağladığı Çapraz-Platform yapısı sayesinde aşılmıştır. Uygulamanın güncellenmesi ya da geliştirilmesi gibi ileriki süreçte yapılacak işlemler de mevcut çapraz platform yapısı ile gerçekleştirileceğinden platformlar arası uyum sorunlarına da çözüm sağlanmıştır.

Diğer bir husus da IoT tabanlı çözümlerin son kullanıcılar tarafından farklı akıllı telefon ve tabletlerde kullanılma istediğidir. Bu durum farklı cihazların izlemesinde ve daha fazla senaryo oluşturulmasında, akıllı cihazlar için IoT uygulamaları geliştirmeyi ve bu hizmetlerin kullanıcıya sunulmasını gerektirmektedir (Datta vd. 2015). IoT cihazlarının sınırlı yetenekleri ile sürekli artan kullanıcı talepleri arasındaki çelişki, IoT uygulamalarının geliştirilmesinin önünde büyük bir engel olmaya devam etmektedir. Farklı IoT mimarilerini birleştirmenin yanı sıra çapraz-platform uygulamaları oluşturmanın yararları günümüzde giderek önem kazanmaktadır (Ren vd. 2017). Oluşturulan akıllı sistem bütünleşik bir yapıya sahip olmasından dolayı farklı nesnelerin kullanımında ve daha fazla senaryo oluşturulmasında kolaylık sağlamakta aynı zamanda yapısı sayesinde ilave nesneler eklenmesine olanak tanımaktadır. Oluşturulan mobil uygulama çapraz-platform yapıya sahip olduğundan farklı IoT mimarilerini birleştirilmesine ve kullanıcı taleplerinin akıllı telefon ve cihazların büyük bölümünden karşılanması sağlanmaktadır.

IoT cihazlarının birlikte çalışabilirliğini sağlamak, IoT yönetiminin karmaşıklığı, akıllı sistemlerin mobil içerikle uyum sorunu ve IoT sistemlerin uyarlanabilir özellikler gerektiren dinamik yapıları nedeniyle mobil platformlar arasında son kullanıcı uygulamalarının geliştirilmesini zorlaştırmaktadır (Soursos vd 2016). IoT sistemin mobil cihazlarla uyum sorunu tasarlanan IoT sistemin ve mobil uygulamanın çapraz platform olarak tasarlanması ile giderilmiştir. Bu sayede IoT sistem dinamik yapısı etkilenmeden mobil platformlar arasında uygulama geliştirme sağlanmıştır. Xamarin sağladığı yapı sayesinde; işletim sistemleri arasındaki yapısal farklılıklardan arındırılmış bir uygulama geliştirilmesine imkân sunmuş, Firebase benzeri bulut veri tabanları ile uyumlu oluşu bütünleşik bir IoT sistem ile mobil arasında dinamik bir yapı sağlanmıştır. Bu durum son kullanıcılar tarafında farklı cihaz ve platform seçenekleri sağlayarak değişik senaryolar oluşturulabilmesini ve akıllı sistem kullanımını kolaylaştırmaktadır.

Çalışmanın bir sonraki aşamasında güvenlik sistemi ve kamera sisteminin yapıya eklenmesi sağlanabilir veya benzer bir çalışma günümüzde giderek popüler hale gelen farklı bir çapraz platform olan React Native veya Flutter ile gerçekleştirilerek çapraz-platformlar arasında karşılaştırma yapılabilir.

Kaynakça

Al Shehri, W. (2013). Cloud database database as a service. International Journal of Database Management Systems, 5(2), 1.

Aljedaani, W., Nagappan, M., Adams, B., & Godfrey, M. (2019, May). A comparison of bugs across the iOS and Android platforms of two open source cross platform browser apps. In 2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft) (pp. 76-86). IEEE.

Ataç, C., & Akleylek, S. (2019). A survey on security threats and solutions in the age of IoT. Avrupa Bilim ve Teknoloji Dergisi, (15), 36-42.

Atkinson, D.M. (2016). The Xamarin Forms Handbook. South Carolina: CreateSpace Independent Publishing.

Chatterjee, N., Chakraborty, S., Decosta, A., & Nath, A. (2018). Real-time communication application based on android using Google firebase. Int. J. Adv. Res. Comput. Sci. Manag. Stud, 6(4).

Datta, S. K., Gyrard, A., Bonnet, C., & Boudaoud, K. (2015, August). oneM2M architecture based user centric IoT application development. In 2015 3rd International Conference on Future Internet of Things and Cloud (pp. 100-107). IEEE.

Dener, M. (2019). A New Home Gateway Design and A Sensor-Based Smart Home Application Including Privacy Protection. Bilişim Teknolojileri Dergisi, 12(1), 23-32.

FirebaseGoogle. Available from: https:// firebase.google.com/products [Çevrimiçi Erişim Tarihi: 01/04/2020].

Heitkötter, H., Hanschke, S., & Majchrzak, T. A. (2012, April). Evaluating cross-platform development approaches for mobile applications. In International Conference on Web Information Systems and Technologies (pp. 120-138). Springer, Berlin, Heidelberg.

Hermes, D. (2015). Xamarin mobile application development: Cross-platform c# and xamarin. forms fundamentals. New York:Apress.

Khawas, C., & Shah, P. (2018). Application of firebase in android app development-a study. International Journal of Computer Applications, 179(46), 49-53.

Mineraud, J., Mazhelis, O., Su, X., & Tarkoma, S. (2016). A gap analysis of Internet-of-Things platforms. Computer Communications, 89, 5-16.

Moroney, L. (2017a). Definitive Guide to Firebase. California: Apress.

Moroney, L. (2017b). Firebase cloud messaging. In The Definitive Guide to Firebase (pp. 163-188). Berkeley: Apress.

Peppers, J. (2015). Xamarin Cross-platform Application Development. Birmingham: Packt Publishing Ltd.

Ren, J., Guo, H., Xu, C., & Zhang, Y. (2017). Serving at the edge: A scalable IoT architecture based on transparent computing. IEEE Network, 31(5), 96-105.

Soursos, S., Žarko, I. P., Zwickl, P., Gojmerac, I., Bianchi, G., & Carrozzo, G. (2016, June). Towards the cross-domain interoperability of IoT platforms. In 2016 European conference on networks and communications (EuCNC) (pp. 398-402). IEEE.

Stonehem, B. (2016). Google Android Firebase: Learning the Basics (Vol. 1). U.K:First Rank Publishing.

Swahadika, E., Besari, A. R. A., & Wibowo, I. K. (2019, September). Implementation of Realtime Database for IoT Home Automation and Energy Monitoring Apps based on Android. In 2019 International Electronics Symposium (IES) (pp. 170-176). IEEE.

VisualStudio. 2020; Available from: https://visualstudio.microsoft.com/tr/xamarin/ [Çevrimiçi Erişim Tarihi: 02/04/2020].

Tao, M., Zuo, J., Liu, Z., Castiglione, A., & Palmieri, F. (2018). Multi-layer cloud architectural model and ontology-based security service framework for IoT-based smart homes. Future Generation Computer Systems, 78, 1040-1051.

Tsai, C. W., Lai, C. F., Chiang, M. C., & Yang, L. T. (2013). Data mining for internet of things: A survey. IEEE Communications Surveys & Tutorials, 16(1), 77-97.

Umuhoza, E. (2017). Domain-specific modeling and code generation for cross-platform mobile and IoT-based applications.Doktora tezi, Politeknik Üniversitesi, Bilgisayar Bilimi ve Mühendisliği, Milano.

Xanthopoulos, S., & Xinogalos, S. (2013, September). A comparative analysis of cross-platform development approaches for mobile applications. In Proceedings of the 6th Balkan Conference in Informatics (pp. 213-220).