



**EGE ÜNİVERSİTESİ**

**YÜKSEK LİSANS TEZİ**

**NESNELERİN İNTERNETİ KAPSAMINDA  
MAĞAZALAR İÇİN AKILLI VE GÜVENLİ BİR  
MOBİL ÖDEME SİSTEMİNİN GELİŞTİRİLMESİ**

**Oğuz ASLAN**

**Tez Danışmanı: Prof. Dr. Levent TOKER**

**Bilgisayar Mühendisliği Anabilim Dalı**

**Sunuş Tarihi: 07.08.2018**

**Bornova-İZMİR  
2018**

**EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ**

**(YÜKSEK LİSANS TEZİ)**

**NESNELERİN İNTERNETİ KAPSAMINDA  
MAĞAZALAR İÇİN AKILLI VE GÜVENLİ BİR  
MOBİL ÖDEME SİSTEMİNİN GELİŞTİRİLMESİ**

**Oğuz ASLAN**

**Tez Danışmanı: Prof. Dr. Levent TOKER**

**Bilgisayar Mühendisliği Anabilim Dalı**

**Sunuş Tarihi: 07.08.2018**

**Bornova-İZMİR**

**2018**



Oğuz Aslan tarafından yüksek lisans tezi olarak sunulan “Nesnelerin İnterneti Kapsamında Mağazalar İçin Akıllı ve Güvenli Bir Mobil Ödeme Sisteminin Geliştirilmesi” başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile EÜ Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 7/8/2018 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

**Jüri Üyeleri**

**Jüri Başkanı**

**Prof.Dr. Levent Toker**

**Raportör Üye**

**Prof.Dr. Aybars Uğur**

**Üye**

**Doç.Dr. Murat Komesli**

**İmza**

.....  
.....  
.....



## EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

### ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi / Doktora Tezi olarak sunduğum “Nesnelerin interneti kapsamında mağazalar için akıllı ve güvenli bir mobil ödeme sisteminin geliştirilmesi” başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

07.08/2018

İmzası

Adı-Soyadı

Eğret ASLAN



**ÖZET****NESNELERİN İNTERNETİ KAPSAMINDA MAĞAZALAR İÇİN  
AKILLI VE GÜVENLİ BİR MOBİL ÖDEME SİSTEMİNİN  
GELİŞTİRİLMESİ**

ASLAN, Oğuz

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Prof. Dr. Levent TOKER

Ağustos 2018, 130 sayfa

Mobil ödeme sistemleri günümüzde ve gelecekte birçok yeniliğe imza atacaktır. Bununla birlikte mobil ödeme sistemleri konusunda yeni trendler ve alternatifler araştırılmaktadır.

Bu tezin amacı geleceğin teknolojisi haline gelmesi beklenen Nesnelerin İnterneti ve bu kapsamda kullanılan RFID teknolojisini araştırmak ve sürekli gelişmekte olan mobil teknolojiler ile Nesnelerin İnterneti kavramlarının birleştirilmesiyle mağazalarda ödeme yaparken yaşanan zorluklara karşı güvenli ve mobil yeni bir ödeme sistemi önerisi sunulmaktadır.

RFID kullanılarak yapılan ödeme sistemleri günümüzde ve gelecekte sıklıkla kullanılmaya başlayacaktır. Bu tezin mağazalarda halen kullanılmakta olan barcode'lu sistemlerden RFID'li ödeme sistemlerine geçiş aşamasını gerçekleştirmesi planlanmaktadır.

Müşterilerin mağazalara girmeden önce kasadaki kuyruğa bakarak ilgili mağazaya girdiği ve buna göre ürün satın aldığı günümüz mağazacılık sektörü için geliştirilmekte olan bu proje sayesinde alıcıların ürün satın almaya teşvikinin yanı sıra güncel teknolojilerin kullanılması ile popülerliğin de artması hedeflenmektedir.

**Anahtar sözcükler:** Nesnelerin interneti, RFID, Mobil Ödeme

**ABSTRACT****DEVELOPING A SMART AND SECURE MOBILE PAYMENT  
SYSTEM FOR THE SCOPE OF INTERNET OF THINGS**

ASLAN, Oğuz

MSc in Computer Eng.

Supervisor: Professor Dr. Levent TOKER

August 2018, 130 pages

Mobile payment systems will take today and author of many innovations in the future. However, new trends and alternatives are being investigated in mobile payment systems.

The purpose of this thesis is to provide mobile and secure payment processing solutions against experienced difficulties when you pay in stores by combining the concept of the Internet of Things and continually evolving mobile technologies and Internet of Things are expected to become the future of technology.

Payment systems use RFID technologies will start to be used frequently present and future. This thesis is planned to perform the transition to the payment system with RFID with barcode currently used in stores.

Now, customers make a point of the intensity of the queue in front of the cashier's desk at the store before entering the store. If the intensity of the queue in front of the cashier's desk at the store is excessive, customers do not prefer enter the store. Through this new method is being developed, customers are encouraged to shopping and it aims to increase the popularity of the store with the use of current technologies.

**Keywords:** Internet of things, RFID, Mobile Payment

## TEŞEKKÜR

Bu çalışma süresince gerekli verilerin sağlanmasında kolaylık gösteren Ege Üniversitesi Kütüphanesi çalışanlarına, özellikle kıymetli görüşlerinden yararlandığım ve yakın ilgisini esirgemeyen sayın Prof. Dr. Levent TOKER'e, tezin biçimlenmesinde değerli katkılarını aldığım arkadaşlarıma teşekkürü bir borç bilirim.



## İÇİNDEKİLER

### Sayfa

|                                       |      |
|---------------------------------------|------|
| ÖZET .....                            | ii   |
| ABSTRACT .....                        | ii   |
| TEŞEKKÜR .....                        | iii  |
| İÇİNDEKİLER .....                     | v    |
| ŞEKİLLER DİZİNİ .....                 | xiii |
| ÇİZELGELER DİZİNİ .....               | xix  |
| KISALTMALAR DİZİNİ .....              | xxi  |
| 1. GİRİŞ .....                        | 1    |
| 2. ÖNCEKİ ÇALIŞMALAR .....            | 2    |
| 2.1 Günümüz Ödeme Teknolojileri ..... | 2    |
| 2.1.1 Kartlı ödeme sistemleri.....    | 2    |
| 2.1.2 Mobil cüzdanlar .....           | 3    |
| 2.1.2.1 İş Bankası ParaKod .....      | 3    |
| 2.1.2.2 MobilExpress.....             | 3    |
| 2.1.2.3 Turkcell Cüzdan.....          | 3    |
| 2.1.2.4 iGaranti .....                | 4    |
| 2.1.3 Online cüzdanlar .....          | 4    |

## İÇİNDEKİLER (devam)

### Sayfa

|                                                      |    |
|------------------------------------------------------|----|
| 2.1.3.1 BKM Express .....                            | 4  |
| 2.1.3.2 iPara .....                                  | 4  |
| 2.1.3.3 PayPal .....                                 | 5  |
| 2.1.4 SMS ile ödeme sistemleri .....                 | 5  |
| 2.1.4.1 Mikro Ödeme.....                             | 5  |
| 2.1.4.2 PayToGo .....                                | 5  |
| 2.1.5 JetKasa.....                                   | 5  |
| 2.1.5 Near Field Communication (NFC) ile ödeme ..... | 6  |
| 3. PROJENİN TANIMLANMASI .....                       | 7  |
| 3.1 Akıllı Mobil Ödeme Sistemi.....                  | 7  |
| 3.1 Önerilen Yöntem .....                            | 8  |
| 3.2 Akış Şeması .....                                | 9  |
| 4. PROJE MATERYALİ VE YÖNTEMLER .....                | 11 |
| 4.1 Veritabanı Yönetimi .....                        | 11 |
| 4.1.1 Analiz ve tabloların oluşturulması.....        | 11 |
| 4.1.2 Structured Query Language (SQL).....           | 18 |
| 4.1.3 MSSQL Database .....                           | 19 |

## İÇİNDEKİLER (devam)

### Sayfa

|                                                            |    |
|------------------------------------------------------------|----|
| 4.1.4 Microsoft SQL Server Management Studio (MSSMS) ..... | 20 |
| 4.1.5 Aqua Data Studio.....                                | 20 |
| 4.2 Sunucu Web Servis Yazılım Geliştirme Yönetimi .....    | 21 |
| 4.2.1 Microsoft Visual Studio .....                        | 21 |
| 4.2.2 Yazılım geliştirme süreci .....                      | 22 |
| 4.2.2.1 Kullanılan yazılım geliştirme yaklaşımları.....    | 23 |
| 4.2.2.2 Kullanılan Design Pattern'ler .....                | 26 |
| 4.2.2.3 Data Transfer Objects .....                        | 30 |
| 4.2.2.4 Garbage Collector .....                            | 30 |
| 4.2.2.5 Exception Handling .....                           | 32 |
| 4.2.2.6 Kullanılan arabirimler.....                        | 33 |
| 4.2.2.7 Kullanılan yazılım geliştirme teknolojileri.....   | 36 |
| 4.2.2.8 Servisler .....                                    | 37 |
| 4.2.2.9 Ağ protokolleri.....                               | 40 |
| 4.2.2.10 Nuget Package Manager .....                       | 43 |
| 4.2.3 Servis yazılımı katmanları .....                     | 44 |
| 4.2.3.1 Infrastructure Layer (Data Katmanı) .....          | 44 |

## İÇİNDEKİLER (devam)

|                                                    | <u>Sayfa</u> |
|----------------------------------------------------|--------------|
| 4.2.3.3 Entity Framework .....                     | 49           |
| 4.2.4 Domain Layer .....                           | 50           |
| 4.2.4.1 DBContext İşlemleri (CRUD İşlemleri) ..... | 50           |
| 4.2.4.2 Authentication Service.....                | 51           |
| 4.2.4.3 User Service.....                          | 53           |
| 4.2.4.4 File Service .....                         | 54           |
| 4.2.4.5 Message Service .....                      | 55           |
| 4.2.4.6 Security Service .....                     | 57           |
| 4.2.4.7 Card Service.....                          | 58           |
| 4.2.4.8 Cart Service .....                         | 60           |
| 4.2.4.9 Product Service .....                      | 61           |
| 4.2.5 Presentation Layer (Service Layer) .....     | 62           |
| 4.2.5.1 HTTPContext sınıfı .....                   | 62           |
| 4.2.5.2 HTTPRequestMessage sınıfı .....            | 62           |
| 4.2.5.3 HTTPResponseMessage sınıfı.....            | 63           |
| 4.2.5.4 Url Routing .....                          | 64           |
| 4.2.5.5 Exception Service .....                    | 67           |

## İÇİNDEKİLER (devam)

|                                                      | <u>Sayfa</u> |
|------------------------------------------------------|--------------|
| 4.2.5.6 JavaScript Object Notation (JSON).....       | 68           |
| 5. AKILLI MOBİL ÖDEME SİSTEMİ .....                  | 70           |
| 5.1 Servisin Sunucuya Kurulumu .....                 | 70           |
| 5.1.1 Servisin kuruluma hazır hale getirilmesi ..... | 70           |
| 5.1.2 Kullanılan işletim sistemi .....               | 72           |
| 5.1.3 Internet Information Services (IIS).....       | 72           |
| 5.1.4 Servisin sunucuya kurulması .....              | 73           |
| 5.2 Mobil Uygulama Yazılım Geliştirme Yönetimi ..... | 79           |
| 5.2.1 Hybrid mobil uygulama nedir? .....             | 80           |
| 5.2.2 Visual Studio Code .....                       | 81           |
| 5.2.3 Mobil uygulamada kullanılan teknolojiler ..... | 82           |
| 5.2.3.1 Git .....                                    | 82           |
| 5.2.3.2 Node JS.....                                 | 83           |
| 5.2.3.3 Node Package Modules (npm) .....             | 83           |
| 5.2.3.4 Bower.....                                   | 85           |
| 5.2.3.1 Cordova (Phonegap) .....                     | 85           |
| 5.2.3.2 IONIC Framework.....                         | 86           |

## İÇİNDEKİLER (devam)

### Sayfa

|                                                                 |     |
|-----------------------------------------------------------------|-----|
| 5.2.3.3 Angular JS .....                                        | 88  |
| 5.2.3.4 Telerik Kendo UI.....                                   | 89  |
| 5.2.3.5 JQuery.....                                             | 90  |
| 5.2.3.6 Asynchronous JavaScript and XML (AJAX).....             | 90  |
| 5.2.3.7 Font Awesome Icons .....                                | 93  |
| 5.2.3.8 Hyper Text Markup Language 5 (HTML5).....               | 94  |
| 5.2.3.9 Cascading Style Sheets 3 (CSS3) .....                   | 95  |
| 5.2.3.10 Mobil uygulama ile native özelliklerin kullanımı ..... | 96  |
| 5.2.3.11 Bootstrap.....                                         | 100 |
| 5.3 Mobil uygulamanın gerçekleştirilmesi .....                  | 100 |
| 5.3.1 Mobil uygulama projesinin oluşturulması .....             | 100 |
| 5.3.2 Mobil uygulamanın cep telefonuna kurulumu.....            | 101 |
| 5.3.2.1 IOS işletim sistemine kurulumu .....                    | 101 |
| 5.3.2.2 Android işletim sistemine kurulumu .....                | 122 |
| 5.4 Radio Frequency Identification (RFID).....                  | 123 |
| 5.4.1 RFID Reader özellikleri.....                              | 124 |
| 5.4.2 RFID Label özellikleri.....                               | 124 |

**İÇİNDEKİLER (devam)**Sayfa

|                                                     |     |
|-----------------------------------------------------|-----|
| 5.4.3 Ülkemizde RFID Reader kullanım alanları. .... | 124 |
| 5.5 Quick Response Code (QRCode) .....              | 125 |
| 5.5.1 QRCode Generator .....                        | 125 |
| 5.5.2 Ülkemizde QRCode kullanım alanları.....       | 125 |
| 6. SONUÇ.....                                       | 126 |
| KAYNAK DİZİNİ.....                                  | 128 |



## ŞEKİLLER DİZİNİ

| <u>Şekil</u>                                                                 | <u>Sayfa</u> |
|------------------------------------------------------------------------------|--------------|
| 3.1. Akıllı mobil ödeme sistemi .....                                        | 10           |
| 4.1 Microsoft SQL Server Management Studio kullanıcı arayüzü .....           | 20           |
| 4.2 Aqua Data Studio kullanıcı arayüzü.....                                  | 21           |
| 4.3 Microsoft Visual Studio IDE arayüzü .....                                | 22           |
| 4.4 Singleton Design Pattern 1. yöntemi.....                                 | 26           |
| 4.5 Singleton Design Pattern 2. yöntemi.....                                 | 27           |
| 4.6 Singleton Design Pattern 3. yöntemi.....                                 | 27           |
| 4.7 Model View Controller (MVC) Pattern ve katmanları.....                   | 29           |
| 4.8 Projede kullanılan bir Exception Handling örneği .....                   | 32           |
| 4.9 Projede kullanılan IDisposable arabirim örneği .....                     | 33           |
| 4.10 Projede kullanılan ICloneable arabirim örneği.....                      | 34           |
| 4.11 Projede kullanılan serialization örneği .....                           | 35           |
| 4.12 Web servis mimarisi örneği.....                                         | 38           |
| 4.13 Representational State Transfer (REST) mimarisi .....                   | 40           |
| 4.14 “Akıllı mobil ödeme sistemi” için kullanılan örnek bir POST isteği..... | 42           |
| 4.15 Nuget Package Manager erişimi.....                                      | 43           |
| 4.16 Nuget Package Manager ile 3rd-party bir kütüphanenin eklenmesi .....    | 44           |

## ŞEKİLLER DİZİNİ (devam)

| <u>Şekil</u>                                                       | <u>Sayfa</u> |
|--------------------------------------------------------------------|--------------|
| 4.17 Data katmanı oluşturma sihirbazı açılışı .....                | 45           |
| 4.18 Data katmanı oluşturma sihirbazı .....                        | 46           |
| 4.19 Database First yaklaşımı oluşturma sihirbazı açılışı .....    | 46           |
| 4.20 Database First yaklaşımı Data Model oluşturma sihirbazı ..... | 47           |
| 4.21 Data Model örneği .....                                       | 48           |
| 4.22 Projede gerçekleştirilen Entity Data Model diyagramı .....    | 49           |
| 4.23 Örnek bir Authentication Service metodu .....                 | 52           |
| 4.24 Örnek bir User Service metodu .....                           | 53           |
| 4.25 File Service dosya yükleme işlemi .....                       | 54           |
| 4.26 Message Service tür dağıtımı ve gönderimi işlemleri .....     | 55           |
| 4.27 SMS Service sınıfı ve SendMessage metodu .....                | 56           |
| 4.28 Mail Service sınıfı ve SendMessage metodu .....               | 57           |
| 4.29 Kullanıcı şifrelerinde kullanılan Encrption işlemi .....      | 58           |
| 4.30 Kullanıcı şifrelerinde kullanılan UTF-8 Coding işlemi .....   | 58           |
| 4.31 Örnek bir Card Service metodu .....                           | 59           |
| 4.32 Örnek bir Cart Service metodu .....                           | 60           |
| 4.33 Örnek bir Product Service metodu .....                        | 61           |

## ŞEKİLLER DİZİNİ (devam)

| <u>Şekil</u>                                                               | <u>Sayfa</u> |
|----------------------------------------------------------------------------|--------------|
| 4.34 Örnek bir HttpRequestMessage Controller metodu .....                  | 63           |
| 4.35 Örnek bir HttpResponseMessage ile Response üretimi .....              | 64           |
| 4.36 Örnek bir Url Routing kullanımı .....                                 | 65           |
| 4.37 ViewModel kullanılarak View'a veri gönderme .....                     | 66           |
| 4.38 ViewModel ile gelen verinin View'da kullanılması .....                | 67           |
| 4.39 Exception Service için kullanılan metotlar.....                       | 68           |
| 4.40 Projede kullanılan bir JSON örneği.....                               | 68           |
| 5.1 Web servisin Publish edilmesi adımı .....                              | 71           |
| 5.2 Web servisin Publish edilmesi.....                                     | 71           |
| 5.3 Internet Information Services (IIS) kullanıcı arayüzü.....             | 72           |
| 5.4 Programlar ve özellikler arayüzü.....                                  | 73           |
| 5.5 Windows özellikleri aç/kapat penceresi .....                           | 74           |
| 5.6 IIS Web yönetimi araçları ayarları .....                               | 75           |
| 5.7 IIS durum ve ranılama ve genel HTTP özellikleri ayarları .....         | 76           |
| 5.8 Güvenlik, performans ve uygulama geliştirme özellikleri ayarları ..... | 77           |
| 5.9 IIS'te uygulama ekle sihirbazı açılması .....                          | 78           |
| 5.10 IIS uygulama ekle sihirbazı .....                                     | 79           |

## ŞEKİLLER DİZİNİ (devam)

| <u>Şekil</u>                                                            | <u>Sayfa</u> |
|-------------------------------------------------------------------------|--------------|
| 5.11 Microsoft Visual Studio Code IDE arayüzü.....                      | 82           |
| 5.12 Node Package Modules (npm)'in bilgisayara kurulumu.....            | 84           |
| 5.13 Mobil uygulamada IONIC Framework kullanımı örneği.....             | 87           |
| 5.14 IONIC Nifty Modal ile network hatası gösterimi.....                | 88           |
| 5.15 Projede Angular JS Directive kullanımı.....                        | 89           |
| 5.16 AJAX ile gerçekleştirilen Login Request işlemi.....                | 92           |
| 5.17 Mobil uygulamada kullanılan Font Awesome Icon'larına örnek.....    | 94           |
| 5.18 Mobil uygulamada HTML5 ile geliştirilen Login ekranı.....          | 95           |
| 5.19 Mobil uygulamada CSS3 ile geliştirilen sepet bilgileri ekranı..... | 96           |
| 5.20 Mobil uygulamada ayarlar menüsü .....                              | 97           |
| 5.21 Mobil uygulamada ActionSheet kullanımı.....                        | 98           |
| 5.22 Mobil uygulamada ActionSheet uygulanması.....                      | 99           |
| 5.23 Mobil uygulama projesinin oluşturulması.....                       | 101          |
| 5.24 Apple Developer Program login ekranı.....                          | 102          |
| 5.25 Apple Developer Program hesabı doğrulama işlemi.....               | 102          |
| 5.26 Join The Apple Developer Program ekranı .....                      | 103          |
| 5.27 Apple Developer Program satın alma adımı .....                     | 103          |

## ŞEKİLLER DİZİNİ (devam)

| <u>Şekil</u>                                                             | <u>Sayfa</u> |
|--------------------------------------------------------------------------|--------------|
| 5.28 Start Your Enrollment adımı .....                                   | 104          |
| 5.29 Apple Developer Program paket seçenekleri .....                     | 104          |
| 5.30 Apple Developer Program ödeme ekranı .....                          | 106          |
| 5.31 Apple Developer Program sertifika oluşturma menüsü .....            | 106          |
| 5.32 Apple Developer Program App ID menüsü .....                         | 107          |
| 5.33 Apple Developer Program yeni App ID oluşturma işlemi .....          | 108          |
| 5.34 IOS İşletim sistemine sahip mobil cihazın iTunes'a bağlanması ..... | 109          |
| 5.35 IOS işletim sistemine sahip mobil cihazdan UDID eldesi .....        | 109          |
| 5.36 Apple Developer Program'a cihaz tanımlanması.....                   | 110          |
| 5.37 Provisioning Profile ekranı.....                                    | 111          |
| 5.38 Provisioning Profile seçenekleri.....                               | 112          |
| 5.39 Provisioning Profile konfigürasyonu.....                            | 112          |
| 5.40 Provisioning Profile cihaz eşleştirilmesi .....                     | 113          |
| 5.41 Provisioning Profile isimlendirme işlemi.....                       | 113          |
| 5.42 Provisioning Profile oluşturulması .....                            | 114          |
| 5.43 IONIC projesi platform listesi .....                                | 115          |
| 5.44 IONIC projesi IOS platform Build işlemi .....                       | 115          |

## ŞEKİLLER DİZİNİ (devam)

| <u>Şekil</u>                                                            | <u>Sayfa</u> |
|-------------------------------------------------------------------------|--------------|
| 5.45 Mobil uygulamanın XCode ile açılması.....                          | 116          |
| 5.46 XCode ile Apple Developer Program’a giriş .....                    | 117          |
| 5.47 XCode ile Apple Developer Program sertifikalarının yüklenmesi..... | 117          |
| 5.48 XCode projesinde Apple Developer Program Team ayarı.....           | 118          |
| 5.49 XCode Projesinde Apple Developer Program Build Settings ayarları.  | 119          |
| 5.50 XCode ile mobil uygulamanın paketlenmesi .....                     | 120          |
| 5.51 iTunes ile uygulamaların görüntülenmesi .....                      | 120          |
| 5.52 Mobil uygulama dosyasının elde edilmesi .....                      | 121          |
| 5.53 Mobil uygulamanın cihaza kurulumu.....                             | 121          |
| 5.54 IONIC projesi Android Platform Build işlemi .....                  | 122          |
| 5.55 Mobil uygulamanın Android Studio ile açılması .....                | 123          |

## ÇİZELGELER DİZİNİ

Çizelge

Sayfa

|     |                                                               |     |
|-----|---------------------------------------------------------------|-----|
| 5.1 | Radio Frequency Identification (RFID) Reader özellikleri..... | 124 |
|-----|---------------------------------------------------------------|-----|





**KISALTMALAR DİZİNİ**Kısaltmalar

|       |                                        |
|-------|----------------------------------------|
| AJAX  | Asynchronous JavaScript and XML        |
| API   | Application Programming Interface      |
| BKM   | Bankalararası Kart Merkezi             |
| CSS   | Cascading Style Sheets                 |
| DOM   | Document Object Model                  |
| DDD   | Domain Driven Design                   |
| DTO   | Data Transfer Object                   |
| FCL   | Framework Class Library                |
| HTML  | Hypertext Markup Language              |
| HTTP  | Hyper Text Transfer Protocol           |
| IIS   | Internet Information Services          |
| IDE   | Integrated Development Environment     |
| JSON  | Javascript Object Notation             |
| MD5   | Message Digest 5                       |
| MSSMS | Microsoft SQL Server Management Studio |
| MVC   | Model View Controller                  |
| NPM   | Node Package Modules                   |

**KISALTMALAR DİZİNİ (devam)**Kısaltmalar

|        |                                  |
|--------|----------------------------------|
| ORM    | Object Relational Model          |
| QRCode | Quick Response Code              |
| REST   | Representational State Transfer  |
| RFID   | Radio Frequency Identification   |
| SMTP   | Simple Mail Transfer Protocol    |
| SOAP   | Simple Object Access Protocol    |
| SQL    | Structured Query Language        |
| TDD    | Test Driven Design               |
| TFD    | Test First Development           |
| UoW    | Unit of Works                    |
| UTF8   | Unicode Transformation Format    |
| XML    | Extensible Markup Language       |
| WCF    | Windows Communication Foundation |

## 1. GİRİŞ

İnternetin gelişimiyle birlikte haberleşme altyapısının da paralel olarak artması dünyada büyük bir ağ yapısının kurulmasına olanak sağlamıştır. İlerleyen yıllarda bu ağ yapısını oluşturan bileşenlerin kat ve kat artması öngörülmektedir. 2030 yılına gelindiğinde, günlük olarak kullanılan birçok nesnenin ve elektronik cihazların bu büyük ağa ekleneceği düşünülmektedir (Ashton, K, 2012).

Günümüz mağazacılık sektöründe kullanılan ve kullanılması düşünülen birçok ödeme yöntemi mevcuttur. Bu yöntemler bir takım sorunlara çözüm olsa da bazı sorunlara hala çözüm olamamıştır. Bu konudaki en büyük sorunlardan bir tanesi de müşterilerin mağazalarda ödeme esnasında kasada beklemesidir.

Müşteriler, artık mağazaya gitmeden önce mağazada bulunan kasadaki yoğunluğa bakmaktadır. Eğer kasada çok müşteri sıra bekliyorsa müşteri, mağazaya girmeyi tercih etmemektedir. Mağazacılık sektöründe müşteriler artık daha çok bu düşünceyle alışveriş yapmaktadır. Bu tezde müşterilerin daha hızlı alışveriş yapabilmelerine olanak sağlayan yeni bir yöntemden bahsedilecektir. Bu sayede müşteriler ürün satın almaya teşvik edilmektedir.

## 2. ÖNCEKİ ÇALIŞMALAR

Bu bölümde mevcut ödeme teknolojileri ile ilgili bir takım bilgilere değinilmektedir. Geliştirilen proje ortaya çıkmadan önce mevcut ödeme teknolojileri üzerinde geniş çaplı bir araştırma yapılmıştır. Yapılan araştırma sonrasında müşterilerin özellikle ihtiyaç duyabilecekleri kısımlar üzerinde durulmuştur. Bu bağlamda elde edilen bilgiler doğrultusunda günümüz ödeme teknolojileri ve içerikleri aşağıdaki gibidir.

### 2.1 Günümüz Ödeme Teknolojileri

Günümüzde kullanılan birçok ödeme sistemi mevcuttur. Tüm bu ödeme teknolojileri belirli amaçlara hizmet vermek için tasarlanmıştır (European Payment Council, 2012; European Payment Council, 2016). Yeni ödeme teknolojilerinin kullanılmaya başlaması bir takım sorunları da beraberinde getirmiştir (BMO Bank of Montreal vd., 2013; Türkiye Cumhuriyet Merkez Bankası, 2017). Bu bağlamda sonraki alt bölümlerde ödeme sistemlerinde kullanılan teknolojiler ve bu teknolojilere ilişkin bir takım sorunlara değinilmiştir.

#### 2.1.1 Kartlı ödeme sistemleri

Kartlı ödeme sistemleri olarak kredi kartları ve banka kartları gibi ödeme sistemleri kullanılmaktadır (Treasury Alliance Group, 2014). Önemli güvenlik açıklarını önlemek için aşağıda verilen bir takım önlemler söz konusu olmuştur.

- Kasiyer tarafından satın alınacak olan ürünler önce barcode okuyucu ile sisteme tanıtılır. Daha sonra ise ödenecek toplam tutar belirlenerek kredi kartı ile müşteri tarafından ödeme gerçekleştirilir.
- İlk üretildiğinde şifresiz olarak sadece kart ile ödemeye imkân tanıyordu.
- Güvenlik açısından dolayı ilk başta kimlik ile ödemeye daha sonra da şifre ile ödemeye geçildi.
- Kartlı ödeme ile yapılan mağazalarda alışverişin yoğun olduğu dönemlerde kasa önünde kuyruklar oluşmaktadır.

### 2.1.2 Mobil Cüzdanlar

Mobil cüzdanlar, ödeme sistemlerine alternatif olarak üretilmiştir (European Payment Council, 2014). Ülkemizde kullanılan bir takım mobil cüzdan örnekleri mevcuttur.

#### 2.1.2.1 İş Bankası ParaKod

İş Bankası tarafından geliştirilen bir mobil cüzdan örneğidir (İş Bankası, 2017).

- Kredi kartı veya banka kartı taşımaya gerek kalmadan QRCode kullanarak ödeme yapılmasına imkân sunar.
- Sadece İş Bankası kredi kartı veya bankamatik kartını desteklemektedir.
- Müşteri tarafından seçilen ürünler kasada okutulduktan sonra POS cihazlarına yüklenen bir yazılım sayesinde bir QRCode üretilir.
- Müşteri, İş Bankası ParaKod mobil uygulaması sayesinde POS cihazında oluşturulan QRCode, cep telefonu kamerası vasıtasıyla mobil uygulama tarafından okunur. QRCode vasıtasıyla elde edilen ödeme bilgileri üzerinden mobil uygulama vasıtasıyla online tahsilat gerçekleşir.

#### 2.1.2.2 MobilExpress

İş Bankası ParaKod'un anlaşmalı bankalar dahilinde (Finansbank, Akbank, Garanti Bankası, Yapı Kredi Bankası v.b.) daha kapsamlı olarak sağladığı bir ödeme yöntemidir (MobilExpress, 2017).

#### 2.1.2.3 Turkcell Cüzdan

Turkcell'in müşterilerine hizmet vermek için hazırladığı bir ödeme sistemidir (Turkcell, 2017; Webrazzi, 2017).

- Birden fazla kart tanımlanmasına imkân sunmaktadır.

- Bu uygulama dahilinde Turkcell iletişim merkezlerinden Turkcell Cüzdan'a para yüklenebilmektedir.

#### **2.1.2.4 iGaranti**

iGaranti ile ParaKod gibi benzer yöntemle QRCode ile ödeme sağlanabilmektedir (Garanti Bankası, 2017).

#### **2.1.3 Online cüzdanlar**

##### **2.1.3.1 BKM Express**

BKM Express kullanan mobil uygulamalar mevcuttur (Bankalararası Kart Merkezi, 2017).

- Bu aşamada ödeme yapılırken müşteri temsilcisi satın almak istenen ürünleri gözden geçirerek toplam tutarı hesaplar ve mağaza mobil uygulamasına ödenecek tutarı girer. Ödenecek tutara ilişkin bir QRCode üretilir.
- Müşteri kendi cep telefonuna install ettiği mobil uygulaması vasıtasıyla cep telefonu kamerasından QRCode'u okutur.
- Ödeme bilgilerine ilişkin müşterinin tanımlı cep telefonuna onay SMS'i gönderilir. SMS ile gelen onay kodu, müşterinin cep telefonundaki mobil uygulamasına girilir ve ödeme gerçekleştirilir. Burada da satış danışmanları ile etkileşim söz konusu olacağından sırada beklemek olasıdır.

##### **2.1.3.2 iPara**

Mağaza içi ödemedan ziyade e-ticarette kullanılan bir ödeme yöntemidir (Aypara Ödeme Kuruluşu, 2017).

- Kendi sahip olduğu sanal POS'lar üzerinden tahsilat yapılır. Bu sayede ürün müşterinin eline ulaştıktan sonra, ürünü inceleyip onay verince ödeme ilişkin para mağaza sahibinin hesabına geçer.

- Mobil Express’de olduğu gibi kullanıcılar kart bilgilerini buraya kaydedebilir ve kayıt edilen kartlar vasıtasıyla alışveriş yapabilir.

### **2.1.3.3 PayPal**

Mobil cüzdan özelliği de mevcuttur. Kart bilgilerini kaydetmeye olanak tanımaktadır (Paypal, 2017).

### **2.1.4 SMS ile ödeme sistemleri**

#### **2.1.4.1 Mikro Ödeme**

- Anlaşmalı mobil operatörlere yapılan ön ödeme sonucu kullanılabilen bir ödeme yöntemidir (Mikro Ödeme, 2017; Wikipedia, 2017).
- Ön ödeme dışında, anlaşmalı faturalı hatta sahip olan kullanıcılar için de ödeme işlemi faturaya yansıtılabilmektedir.

#### **2.1.4.2 PayToGo**

Bir diğer SMS tabanlı mobil ödeme sistemidir. Anlaşması Turkcell ve Avea ile sınırlıdır.

### **2.1.5 JetKasa**

Daha çok gıda sektöründe kullanımı yaygındır. Migros, Kipa ve Carrefour gibi gıda ağırlıklı mağazalarda kullanılmaktadır. Seçilen ürünlerin hızlı bir şekilde ödemesinin gerçekleştirilebilmesi için tasarlanmıştır (Hürriyet, 2017).

Ödeme işlemi aşağıdaki şekilde gerçekleştirilir.

- Seçilen ürünler alışveriş sepetine eklendikten sonra ödeme işlemi için JetKasa’ya gidilir.
- Ödeme işlemi nakit veya kredi kartı ile yapılabilir.
- Her ne kadar hızlı ödeme amaçlanmış olsa da alışveriş açısından yoğun olan bir mağaza için JetKasa kuyruğunda sıra beklemek olasıdır.

### **2.1.6 Near Field Communication (NFC) ile ödeme**

Tasarlanan “akıllı mobil ödeme sistemi”ne alternatif olarak Near Field Communication (NFC) Mobil Ödeme Sistemleri örnek olarak gösterilebilir ancak bu yöntem, önerilen yöntemi içermemektedir (Gökhan, Y. vd, 2018; Minihold, R., 2011). Tasarlanan yeni sistemin, Near Field Communication (NFC) Mobil Ödeme Sistemleri’ne göre daha yeni bir teknoloji kullanılması, Nesnelerin İnterneti ve ödeme konusundaki hız açısından avantajları bulunmaktadır. Near Field Communication (NFC) Mobil Ödeme Sistemi kullanıldığında kasada sıra beklemeye neden olma olasılığı vardır.



### 3. PROJENİN TANIMLANMASI

Proje, yazılım ve donanım kısmı olmak üzere iki bölümden oluşmaktadır. Önce donanımsal birimler hakkında genel bir araştırma yapılarak temel seviyede bilgi sahibi olundu. Bu bilgi kazanımlarından sonra tasarım fikirleri üzerinden seçilen bir yöntem doğrultusunda projenin gerçekleşmesi sağlandı.

Gerçekleştirilen kısımlar adım adım sağlanarak test işlemine tabii tutuldu. Test işlemine tabi tutulan tüm parçalar uygulama aşamasında birleştirilerek bir bütün haline getirildi ve projenin taslağı oluşturuldu.

Projede yapılan her işlem sonrası gerçekleştirilen aşama bilgileri doküman haline getirilmiş ve bunlar daha sonra proje bitiminde derlenip toplanarak proje raporu oluşturulmuştur.

Ayrıca tüm kullanıcıların projeyi gerçekleştirebilmeleri için kullanılan donanımsal cihaz ve parçaların mali değerleri göz önünde bulundurulmuş ve buna bağlı olarak ucuz ve kaliteli elemanlar seçilmeye özen gösterilmiştir.

Uygulamanın gerçekleşmesi aşamasında sistem mühendislik yaklaşımıyla irdelenmiş ve karşılaşılan problemlere en uygun çözüm bulunarak uygulamaya dökülmüştür. Proje için yapılan her hamle itina ile irdelenmiş ve benzer diğer çalışmalar göz önünde bulundurularak ölçeklenebilirlik sağlanmıştır.

Yapılan tasarım güncel olarak fabrika şartlarında maliyet olarak daha da düşürülebilir ve kullanımı kolay uygulanabilirlik açısından verimli bir cihaz haline gelebilir. Bunun için öncelikle projenin fabrikasyon süreci aşamalarından geçmesi gerekmekte ve bu aşamalarda bir takım testlere tabi tutulması gerekmektedir.

Fabrikasyon aşamasına geçebilmesi için sadece parçaları hazır olarak temin edip monte eden bir süreç yerine, parçaları da kendi üreten ve bunları birleştirerek ürünü elde eden yaklaşımla da verimli sonuçlar elde edilebilecektir.

#### 3.1 Akıllı Mobil Ödeme Sistemi

Günümüzde kullanılan bir takım ödeme sistemlerinden bahsedilmiştir. Bu bağlamda ödeme sistemleri kartla, kartsız bir şekilde SMS ile veya mobil uygulama vasıtasıyla yapılabilmektedir (Chaix, L., Torre, D., 2011). Günümüz

teknolojileri kullanılarak tüm bu mağaza içindeki ödeme sistemlerinde, ödeme esnasında kasada sıra bekleme sorununa tam olarak bir çözüm üretilmemiştir.

Bu projede değinilen “akıllı mobil ödeme sistemi” sayesinde alışveriş esnasında müşterinin kasada ödeme için sırada geçirdiği zaman reyonlara dağıtılabilecektir. Bu sayede müşteri, kasa kuyruğunda beklemekten sıkılmayacaktır.

### 3.1 Önerilen Yöntem

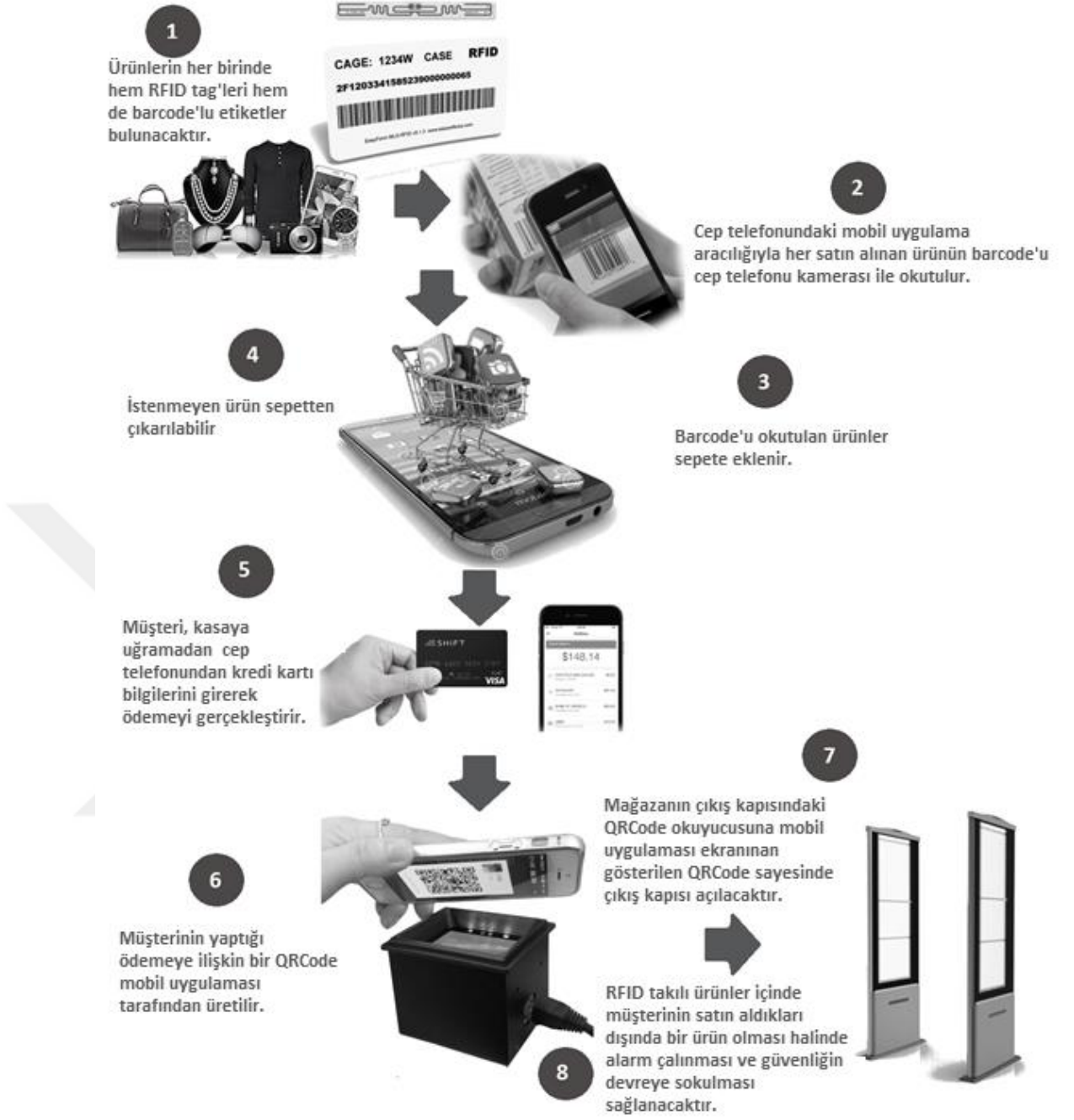
Mağaza kasasında uzun kuyrukları engellemek amacıyla geliştirilen “akıllı mobil ödeme sistemi” aşağıdaki şekilde tasarlanmıştır.

- ✓ Mağazadaki ürünlerin her birinde hem RFID tag’leri hem de barcode’lu etiketler bulunacaktır (Lotlikar, T. vd, 2013).
- ✓ Mağazaya giren müşteri, alışveriş sırasında, cep telefonundaki mobil uygulama aracılığıyla her satın aldığı ürünün barcode’unu cep telefonu kamerası ile okutarak ürünü sepete yerleştirir.
- ✓ Ürünler istenildiği kadar mobil uygulama sepetine eklenip çıkartılabilir.
- ✓ Ürünler mobil uygulama ile sanal sepete eklenirken aynı zamanda fiziksel sepete de poşetlenerek eklenmiş oluyor.
- ✓ Müşteri, mobil uygulaması vasıtasıyla online olarak hiç kasaya uğramadan ve en önemlisi sıra beklemeden cep telefonundan kredi kartı bilgilerini girerek ödemeyi gerçekleştirecektir. Bilhassa çok müşterili tekstil firmalarında kasada sıra beklemek, müşterilerin alışverişe olan ilgisini azaltmaktadır.
- ✓ Barcode ile kameraya okutulan ürünlerin mobil uygulama vasıtası ile ödeme işlemi gerçekleştirildikten sonra satın alınan ürünlerin RFID tag ID’leri ve ürün sayıları, alışveriş yapılan firmanın sunucuları tarafından ödeme yapan müşterinin mobil uygulaması ile konuşturulacaktır (Weinstein, R., 2005).

- ✓ Fatura, kurumun sunucusu tarafından e-deftere işlenecek, e-fatura olarak da e-mail ile müşteriye gönderilecektir.
- ✓ Mobili install eden her cep telefonu kullanıcısına e-mail girme zorunluluğu öngörülecek ve kullanıcı bu e-maili de doğrulayacaktır.
- ✓ Bir kereye mahsus login edildiği sırada hem cep telefonu numarası doğrulaması hem de e-mail doğrulaması yapılacaktır.
- ✓ Bu işlemler tamamlandıktan sonra müşterinin ödemesine ilişkin bir QRCode mobil uygulaması tarafından üretilecektir.
- ✓ Üretilen bu QRCode mağaza için güvenlik açığına sebebiyet vermemesi açısından çıkış kapısı için anahtar görevi görecek.
- ✓ Mağazanın çıkış kapısındaki QRCode okuyucusuna mobil uygulaması ekranından gösterilen QRCode sayesinde çıkış kapısı açılacaktır. Çıkış kapısı tasarımı sinemalarda kullanılan Barcode/QRCode okuyuculu kapılar şeklinde olabilir.
- ✓ Çıkış kapısı RFID Reader ile entegre olacaktır. RFID takılı ürünler içinde müşterinin satın aldıkları dışında bir ürün olması halinde alarm çalınması ve güvenliğin devreye sokulması sağlanacaktır (Mehrerdi, Y., 2014).

### 3.2 Akış Şeması

Akıllı Ödeme Sistemi için yukarıda bahsedilen yöntem Şekil 3.1’de gösterilmiştir.



Şekil 3.1: Akıllı mobil ödeme sistemi.

## 4. PROJE MATERYALİ VE YÖNTEMLER

Projede tasarlanan “akıllı mobil ödeme sistemi” yöntemi kapsamında kullanılması planlanan donanımsal parçalar ve yazılımsal teknolojiler en temel işlem den başlayarak projenin en uç noktasına doğru olacak şekilde aşağıdaki gibi maddeler halinde bahsedilmiştir.

### 4.1 Veritabanı Yönetimi

Verilerin sağlıklı bir şekilde saklanması ve istenildiği zaman yetkili kullanıcılar tarafından erişilebilmesi için kullanılan bir takım teknolojiler mevcuttur. Bu bağlamda ilişkisel veritabanı, 1970 yılında Edgar Frank Codd tarafından önerilmiştir.

#### 4.1.1 Analiz ve tabloların oluşturulması

Projenin analiz aşamasında Event Storming domain modellemeye uygun olarak tasarımlar ve kurgu inşa edilmeye çalışılmıştır.

Proje hem donanım hem de yazılım olmak üzere iki kısımdan oluşmaktadır. Bu bağlamda analiz ve tabloların oluşturulması kısmı yazılım kısmının gerçekleştirilmesinde devreye girmektedir. Projede gerçekleştirilmesi planlanan mobil uygulama sayesinde kullanıcılar interaktif olarak veritabanına erişimleri mümkün olacaktır.

Veritabanına erişimde her kullanıcı sadece kendine ait olan verileri ekleme, değiştirebilme ve silme yetkisinde olması gerektiğinden veritabanı tablolarının oluşturulması buna uygun olarak sağlanmıştır.

Projede kullanılan tabloların listesi ve kolonları aşağıdaki gibidir:

- ✓ **Kod:** Tüm genel bilgilerin tutulduğu tablodur. Sistem içerisinde kullanılması gereken gizli kod tanımları ve müşteri tarafında gösterilen statik bilgiler bu tabloda tutulmaktadır.

**Kolon isimleri:** ID, KodGrup, KodNo, Aciklama, Aktif.

- ✓ **Kullanici:** Kullanıcı bilgilerinin tutulduğu tablodur. Mobil uygulama üzerinden sisteme kayıt olan her bir kullanıcının bilgisi bu tabloda tutulur.

**Kolon isimleri:** ID, KullaniciAdi, Sifre, Ad, Soyad, Durum, FotoAdi, FotoYolu, FotoUzanti, Aktif, GuncellemeTarihi.

- ❖ **Durum Kolonu:** Bu kolonda sistemi kullanan kullanıcıların sistemdeki durumlarının bilgisi barındırılmaktadır. Bir kullanıcı bu kolona göre “Pasif”, “Sms onayı var, mail onayı yok”, “Sms onayı yok, mail onayı var” ve “Aktif” durumlarından herhangi birine sahip olabilir.

- ✓ **OturumKayit:** Sistemde kayıtlı olan mevcut kullanıcılar için mobil uygulama üzerinden yapılan her başarılı giriş denemesi sonucu bu tabloda birer kayıt oluşur. Kullanıcının giriş yaptığı bilgisi bu tablodan anlaşılır. Aynı zamanda bu Authorization için kullanılan Token Based Authentication bilgileri de bu tabloda tutulmaktadır. Mobil uygulama vasıtasıyla sisteme login olan her bir kullanıcı için bir token üretilir. Üretilen bu Token OturumKayit tablosunda tutulur. Artık mobil uygulamaya giriş yapan kullanıcı, ona tahsis edilen Token üzerinden tüm işlemlerini gerçekleştirebilir. Her sorgulama yapmak için tekrardan kullanıcı adı ve şifre bilgisini girmesine gerek yoktur.

**Kolon isimleri:** ID, KullaniciID, OturumNo, PlatformTuru, PlatformID, OturumTarihi, Kultur.

- ❖ **KullaniciID Kolonu:** Sistemi kullanan her bir kullanıcı için tahsis edilen unique bir değerdir.
- ❖ **OturumNo Kolonu:** Projede, Token Based Authentication kullanıldığı için sisteme login olan her bir kullanıcı için bir token üretilir. Üretilen bu token bilgisi “OturumNo” kolonunda saklanır. Token içeriği her bir login işlemine ait olan unique bir değerdir.
- ❖ **PlatformTuru Kolonu:** Mobil uygulamaya Android mi yoksa IOS işletim sistemine sahip bir cihaz vasıtasıyla mı giriş

yapıldığının bilgisini tutmaktadır. Bu bilginin tutulmasının nedeni, sistem eğer mobil uygulama kullanıcılarına bir şekilde “Notification” gönderme ihtiyacı duyarsa bildirim gönderimini sağlayabilmesi içindir. Android ve IOS işletim sistemine sahip cihazlar için bildirim gönderme işlemi farklı teknolojilerle sağlanmaktadır. Bu şekilde olan kullanıcıların ayırt edilebilmesi ve ona göre yazılımın yönlendirilebilmesi için bu bilginin saklanması gerekmektedir.

❖ **PlatformID Kolonu:** “PlatformTuru” kısmında bahsedildiği gibi mobil uygulamayı kullanan kullanıcılara bildirim gönderilebilmesi için tutulması gereken bir diğer bilgi de her bir cihazın Unique Identifier bilgisidir. Bu bilgi mobil cihazda çalışan işletim sistemine göre değişiklikler gösterebilmektedir. Bir kullanıcıya “Notification” gönderebilmek için o cihazın Unique Identifier bilgisinin bilinmesi gerekmektedir. Aksi durumda mobil uygulamanın üzerinde koştugu cihaza hiçbir şekilde bildirim gönderebilmek mümkün değildir.

✓ **OturumKayıtTarihce:** Sistemde kayıtlı olan her bir kullanıcı sisteme giriş yaptıktan sonra bir takım işlemleri gerçekleştirebilir. Bir kullanıcı sistemde işlemlerini gerçekleştirdikten sonra sistemden çıkış yapmak istediğinde OturumKayıt tablosunda bulunan login bilgileri oradan silinerek OturumKayıtTarihce adı verilen bu tabloya aktarılır. Bu sayede sistemde aktif olarak login olan kullanıcılar OturumKayıt tablosunda; sistemden logout olan kullanıcı kayıtları da OturumKayıtTarihce tablosunda tutulmuş olur. Tabloda bulunan “PlatformTuru” kolonu mobil uygulamayı kullanan kullanıcının Android mi yoksa IOS işletim sistemine sahip bir cihaz kullandığı bilgisini tutmaktadır.

**Kolon isimleri:** ID, OturumKayıtID, KullanıcıID, OturumNo, PlatformTuru, PlatformID, OturumTarihi, OturumBitisTarihi, Kultur.

✓ **Authentication:** Sisteme kayıt olan her bir kullanıcı için bir takım doğrulama işlemleri söz konusudur. Bir kullanıcı sisteme kayıt olduğunda, kayıt esnasında ondan alınan cep telefonu ve e-mail bilgileri doğrultusunda doğrulama işlemine tabii tutulur. Cep telefonu

numarasına gönderilen doğrulama kodu bilgisi ile e-mail'ine gönderilen doğrulama linki uzantı bilgileri bu tabloda tutulur. Kullanıcı sms veya e-mail doğrulamasını gerçekleştirme aşamasında hangi doğrulama kodlarına sahip olduğu bilgisi bu tablodan elde edilir ve kıyaslama yine bu tabloda bulunan veriler vasıtasıyla sağlanır. Tabloda bulunan “DogrulamaTuru” kolonu kullanıcının hangi tür doğrulamasına ait verinin barındığı bilgisini tutmaktadır.

**Kolon isimleri:** ID, KullaniciID, DogrulamaTuru, Aciklama, Aktif, GuncellemeTarihi.

- **DogrulamaTuru Kolonu:** Sisteme kayıt olan kullanıcıların vermiş oldukları cep telefonu ve e-mail bilgilerinin doğruluğunun kontrol edilebilmesi için kullanılan yöntem için tutulması gereken bilgilerin türünü belirten kısımdır. Doğrulama işleminin SMS doğrulaması mı yoksa e-mail doğrulaması mı olduğu bilgisini tutar.

- ✓ **AdresIletisim:** Sisteme kayıt olan her bir kullanıcı için ev adresi, iş adresi, ev telefonu, iş telefonu, cep telefonu ve e-mail bilgilerinin tutulduğu tablodur.

**Kolon isimleri:** ID, AdresIletisimTuru, KullaniciID, Aciklama, Aktif, Durum, GuncellemeTarihi.

- **AdresIletisimTuru Kolonu:** Kullanıcının adres ve iletişim bilgilerinin barındıran bu tabloda ne tür bilgilerin bulunduğu bilgisini tutmaktadır. Bunlar ev adresi, iş adresi, ev telefonu, iş telefonu, cep telefonu ve e-mail bilgilerinden oluşmaktadır.

- ✓ **KullaniciKart:** Mobil uygulama vasıtasıyla sepete eklemiş oldukları ürünleri hızlı bir şekilde satın alabilmek için kullanacakları kredi kartı bilgilerini barındıran tablodur.

**Kolon isimleri:** ID, KartNo, Adi, Aktif, GuncellemeTarihi, KullaniciID.

- ✓ **SistemParam:** “Akıllı mobil ödeme sistemi” kapsamında sistemin ihtiyaç duyacağı tüm sistem ayarları bu tabloda tutulmaktadır. Örneğin bu sistemde dosya işlemleri kullanılmaktadır. Kullanıcıların mağazalardan satın almak istedikleri her ürünün resimleri veritabanında tutulmamaktadır. Tüm bu resimler farklı bir disk ortamında tutulmaktadır. Veritabanında o resimlerin dosya isimleri tutulmaktadır. Bu sayede veritabanı resim boyutlarından dolayı şişmeyecektir. Sadece içerisinde ham veri barındırmış olacaktır. Bu aşamada kullanıcı eğer bir ürüne ait resimleri görmek isterse sistem dosya işlemleri servisini devreye sokacaktır. Dosya işlemleri servisinin servis url, kullanıcı adı ve şifre bilgisi “SistemParam” tablosunda tutulmaktadır. Aynı zamanda farklı bir örnek verecek olursak ürünlerin tutulduğu disk yolu bilgisi de yine ”SistemParam” tablosunda tutulmaktadır. Bu sayede herhangi bir aksaklık oluşması durumunda dosyalar mevcut sunucudan farklı bir sunucuya taşınabilir ve bu dosyaların disk yolları değiştirilebilir olmuş olacaktır. Bu parametrik yapı sayesinde yazılıma müdahale etmeden dosya değişiklikleri söz konusu olabilecektir.

**Kolon isimleri:** ID, ParamGrup, ParamNo, Aciklama, Aktif, GuncellemeTarihi.

- ✓ **Kategori:** “Akıllı mobil ödeme sistemi”ni kullanan şirkete ait satacağı ürünlerin kategorilerinin ağaç yapısı şeklinde tutulabileceği tablodur. Bu sayede bir kategori istenildiği derece kadar alt kategorilere bölünebilir olmuştur.

**Kolon isimleri:** ID, Aciklama, UstKategoriID, KategoriResimAdi, KategoriResimYolu, KategoriResimUzanti, Aktif, GuncellemeTarihi.

- **UstKategoriID Kolonu:** Bu kolon sayesinde “akıllı mobil ödeme sistemi”nin sahip olduğu şirket kategorileri, istediği dereceye kadar tanımlanabilir olacaktır. “UstKategoriID” kolonu kendisinin bağlı olduğu bir üst kategorinin Id bilgisini barındırmaktadır. Bu kolon sayesinde tüm kategoriler için bir Tree yapısı oluşturulabilmektedir.

- ✓ **Marka:** “Akıllı mobil ödeme sistemi”ni kullanan şirketlerin satış yaptıkları ürünlerin tüm marka bilgilerini tutabilecekleri tablodur. Bir müşteri mobil uygulama ile hangi ürünleri satmak isterse bu tabloya sadece o markalara ait bilgileri bu tabloya aktarması yeterli olacaktır. Bu sayede müşteriler, A markası ürünlerini “akıllı mobil ödeme sistemi” vasıtasıyla satın alabilirken, B markası ürünlerini ise satın alamayacaktır. Hangi markaların ve bu markalara ait hangi ürünlerin “akıllı mobil ödeme sistemi” üzerinden satışının yapılabileceği kararı, “akıllı mobil ödeme sistemi”ne sahip olan şirketin kararında olmuş olacaktır.

**Kolon isimleri:** ID, Aciklama, MarkaResimAdi, MarkaResimYolu, MarkaResimUzanti, Aktif, GuncellemeTarihi.

- ✓ **Urun:** “Akıllı mobil ödeme sistemi” vasıtasıyla satışı yapılacak tüm ürün bilgileri bu tabloda tutulmaktadır. Ürünün hangi kategori ve hangi markaya ait olduğu bilgileri de bu tabloda tutulmaktadır.

**Kolon isimleri:** ID, Aciklama, KategoriID, MarkaID, Aktif, GuncellemeTarihi.

- ✓ **UrunResim:** Mobil uygulama üzerinden satışı gerçekleştirilecek tüm ürünlerin birden çok resmi olabilir. Bu nedenle “Urun” tablosundan ayrı olacak şekilde yeni bir tabloda bu bilgilerin tutulması sağlanmıştır. Ürünlere ait olan resimler bu tabloda fiziksel olarak tutulmamaktadır. Ürünler veritabanının bulunduğu sunucuda veya bir başka sunucuda diskte tutulmaktadır. Sistemin bu tabloyu kullanarak bir ürüne ait resimlere erişebilmesi, tabloda bulunan “DosyaAdi”, “DosyaYolu” ve “DosyaUzanti” kolonları sayesinde gerçekleşmektedir. Bu kolonlarda bulunan bilgiler, sunucu ortamındaki diskte bulunan resimlerin dosya isimlerini barındırmaktadır.

**Kolon isimleri:** ID, Aciklama, DosyaAdi, DosyaYolu, DosyaUzanti, UrunID, Aktif, GuncellemeTarihi.

- ✓ **UrunKod:** “Akıllı mobil ödeme sistemi” kullanan bir firma satış yapacağı bir mağazasında bulunan tüm ürünlerin stok bilgilerinin tutulduğu tablodur. Bu tabloda, ürünlere ait Barcode ve Rfid

numaralarının yanı sıra, ürüne ait satış durumları ve fiyat bilgileri de tutulmaktadır.

**Kolon isimleri:** ID, Barcode, Rfid, UrunID, SatisDurumu, Fiyat, Aktif, GuncellemeTarihi.

- **Barcode Kolonu:** Her bir ürünün kendisine ait bir satış kartı bulunmaktadır. Bu kartta satış esnasında kasanın ürünü tanıyabilmesi için kullanılan barcode bilgileri tutulmaktadır. Hangi ürünün hangi barcode ile ilişkilendirildiği bilgisi bu kolonda tutulmaktadır.
- **Rfid Kolonu:** “Akıllı mobil ödeme sistemi”nde ödeme işlemi gerçekleştikten sonra mağazadan fiziksel olarak poşetlenmiş ürünlerle birlikte çıkış yapılacağı için olası çalma durumlarına önlem olması için satışı yapılmayan herhangi bir ürünün veya ürünlerin mağaza dışarısına çıkmaması gerekmektedir. Bu bağlamda Rfid kolonunda tutulan bu bilgi mağazanın çıkış yapılması esnasında kullanılacak ve satışı gerçekleşmeyen bir ürünün tespiti için kullanılacaktır.
- **SatisDurumu Kolonu:** Bu tabloda tutulan stoktaki tüm ürünlerin hangilerinin satıldığı, hangilerinin henüz satılmadığı bilgisini tutmaktadır.
- ✓ **KullaniciSepet:** Mobil uygulamayı kullanan her bir kullanıcının oluşturmuş olduğu sepet bilgisi bu tabloda tutulmaktadır. Bu tabloya göre bir kullanıcının birden fazla sepeti olabilir ancak, bir kullanıcının ödemesini yapmadığı aktif konumda olan sadece bir sepeti olabilir.

**Kolon isimleri:** ID, KullaniciID, QRCode, SepetDurum, KartNo, SepetSMSOnayKodu, Adi, Adres, Email, PhoneNumber, Aktif, GuncellemeTarihi.

- **QRCode Kolonu:** Kullanıcı, mobil uygulama vasıtasıyla sepete eklediği ürünleri satın alması durumunda çıkış kapısında kullanabileceği, sadece bu satışa ait üretilen QRCode bilgisinin tutulduğu kolondur.

- **SepetDurum Kolonu:** Mobil uygulama kullanıcısının sahip olduğu sepetin satışının gerçekleşip gerçekleşmediği bilgisini tutmaktadır.
- **SepetSMSOnayKodu Kolonu:** Kullanıcı tarafından mobil uygulama vasıtasıyla ödenen ürünler için, ödemeye ilişkin kullanıcıya gönderilen SMS onay bilgisinin tutulduğu kolondur.
- ✓ **KullaniciSepetUrun:** Mobil uygulamayı kullanan kullanıcının, kendi sepetine ait hangi ürünlerin bulunduğu bilgisinin tutulduğu tablodur. Bu tablo sayesinde kullanıcı sepete eklemiş olduğu ürünleri görebilir. Sepetine ürün ekleyebilir veya sepetinden ürün çıkartabilir olacaktır.

**Kolon isimleri:** ID, KullaniciSepetID, UrunKodID, Aktif, GuncellemeTarihi.

- **KullaniciSepetID Kolonu:** Hangi ürünlerin hangi sepete ait olduğu bilgisi bu kolonda tutulmaktadır.
- **UrunKodID Kolonu:** Kullanıcının sepetinde hangi ürünün olduğu bilgisi bu kolon sayesinde öğrenilebilmektedir.
- ✓ **HataLog:** “Akıllı mobil ödeme sistemi” içerisinde sistemsel veya yazılımsal herhangi bir sorun oluşması durumunda tablolara veri ekleme, tablolardaki veriyi düzenleme veya tablodan veri silme esnasında oluşabilecek herhangi bir sorun için bu tabloya veri yazılmaktadır. Bu sayede sistemde, aktif kullanım esnasında oluşabilecek herhangi bir sorun olması durumunda bu tablodan sorunun neden oluştuğu bilgisi araştırılabilir.

**Kolon isimleri:** ID, Exception, InnerException, Tarih.

#### 4.1.2 Structured Query Language (SQL)

Türkçe'deki karşılığı “Yapısal Sorgulama Dili” olarak geçmektedir. Veritabanında bulunan tablolara erişebilmek için bir takım sorgular oluşturulması

gerekmektedir. Bu sorgular için kullanılan dillerden bir tanesidir. Structured Query Language (SQL) özelliklerinden birkaçı aşağıdaki gibidir:

- Bir sorgulama dilidir, programlama dili değildir.
- Yapısal bir sorgulama dilidir.
- Veritabanında bulunan verileri okumak için kullanılır.
- Ekleme, silme, güncelleme ve okuma işlemleri bu sorgulama dili vasıtasıyla gerçekleştirilebilir.

Projede veritabanı ile ilgili olan tüm kısımlarda Structured Query Language (SQL) kullanılmıştır. SQL tamamıyla “akıllı mobil ödeme sistemi” servis tarafında kullanılmıştır. Mobil uygulama hiçbir şekilde doğrudan veritabanına bağlanamamaktadır. Kullanıcılar, mobil uygulama vasıtasıyla, servis aracılığıyla veritabanına yetkileri dahilinde işlemler yapabilmektedir.

#### **4.1.3 MSSQL Database**

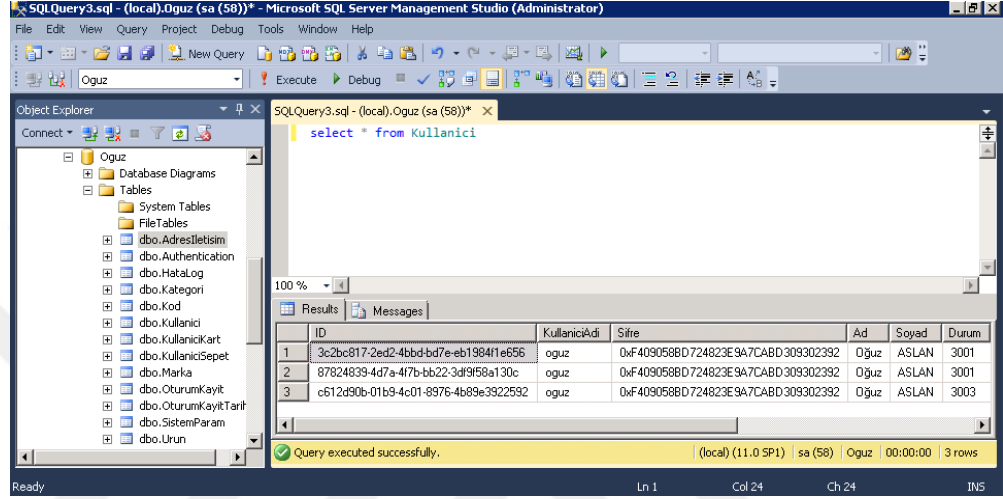
Microsoft’un sunmuş olduğu, açılımı Microsoft Structured Query Language olan MSSQL veritabanı ilişkisel verilerin depolandığı bir alandır (Mistry, R., Misner, S., 2014).

Projede, kullanıcı verileri başta olmak üzere kişisel veriler (doğum tarihi, mail, telefon vs.), authentication verileri, müşterilerin satın alacağı ürünler ve bu ürünlere ilişkin yapılan ödeme bilgileri, mağazadaki ürünler ve bu ürünlerin fiyat bilgileri (ürün fotoğrafları, barcode ve RFID numaraları, detayları vs.) gibi bilgilerin depolanacağı alan olarak kullanılmıştır.

Daha öncesinde yapısal bir sorgulama dili olan Structured Query Language (SQL)’den bahsedilmişti. SQL yapısal programlama dili birçok teknoloji firması tarafından aktif olarak kullanılmaktadır. Projede veritabanı olarak Microsoft’un ürünü tercih edilmiştir.

#### 4.1.4 Microsoft SQL Server Management Studio (MSSMS)

Adından da anlaşılacağı gibi Microsoft SQL Server Management Studio (MSSMS), Microsoft'un geliştiricilere sağladığı bir üründür. Bu ürün sayesinde veritabanı oluşturmak mümkündür. Ayrıca bu uygulama sayesinde oluşturulan veritabanından Structured Query Language (SQL) kullanarak işlemler yapabilmek mümkündür.



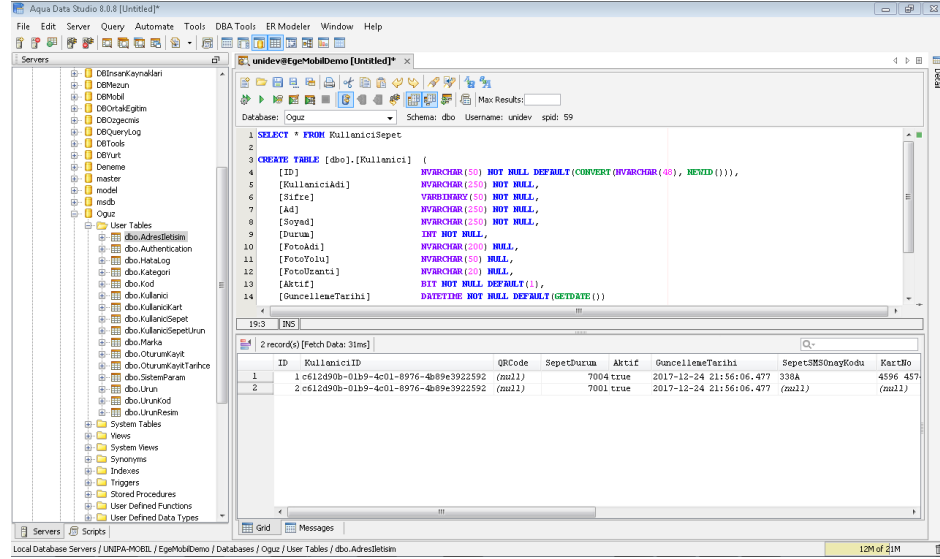
Şekil 4.1: Microsoft SQL Server Management Studio (MSSMS) kullanıcı arayüzü.

Projede Database tool olarak Microsoft SQL Server Management Studio (MSSMS) kullanılmıştır. Sunucu üzerine kurulan Microsoft SQL Server Management Studio (MSSMS) sayesinde “akıllı mobil ödeme sistemi” için oluşturulan veritabanı işlemleri buradan yönetilebilir olmaktadır.

#### 4.1.5 Aqua Data Studio

Aqua Data Studio, Microsoft SQL Server Management Studio (MSSMS) gibi veritabanı işlemlerinde kullanılan Structured Query Language (SQL) sorgulama dilini kullanmayı sağlayan bir arayüz sunmuştur. Aqua, Microsoft SQL Server Management Studio (MSSMS) gibi bir database sunmamaktadır.

Aqua, bu proje için daha çok uzak sunucuda bulunan veritabanına bağlanıp SQL sorgularını çalıştırmak ve database işlemlerini veritabanının bulunduğu uzak sunucuya bağlanmaya gerek kalmadan çözümlemeyi sağlamak için kullanılmıştır. Bu bağlamda Şekil 4.2’de Aqua Data Studio arayüzü gösterilmiştir.



Şekil 4.2: Aqua Data Studio kullanıcı arayüzü.

“Akıllı mobil ödeme sistemi” kapsamında Microsoft SQL Server Management Studio (MSSMS) sunucu üzerinde kurulmuş ve bir sefere mahsus Database oluşturma işlemi buradan gerçekleştirilmiştir. Geri kalan analiz ve tabloların oluşturulması kısmında Aqua Data Fold üzerinden veritabanının bulunduğu sunucuya database “ConnectionString” sayesinde erişim sağlanarak işlemler gerçekleştirilmiştir.

## 4.2 Sunucu Web Servis Yazılım Geliştirme Yönetimi

Bu bölümde servis tarafında geliştirilen web servis yazılımının tüm özelliklerinden, kullanılan tüm teknolojilerden ve metodolojilerden, yazılım geliştirme süreçlerinden ve proje kapsamında gerçekleştirilen “akıllı mobil ödeme sistemi” için gerekli olan tüm servis yöntemlerinden konu başlıkları şeklinde bahsedilmiştir.

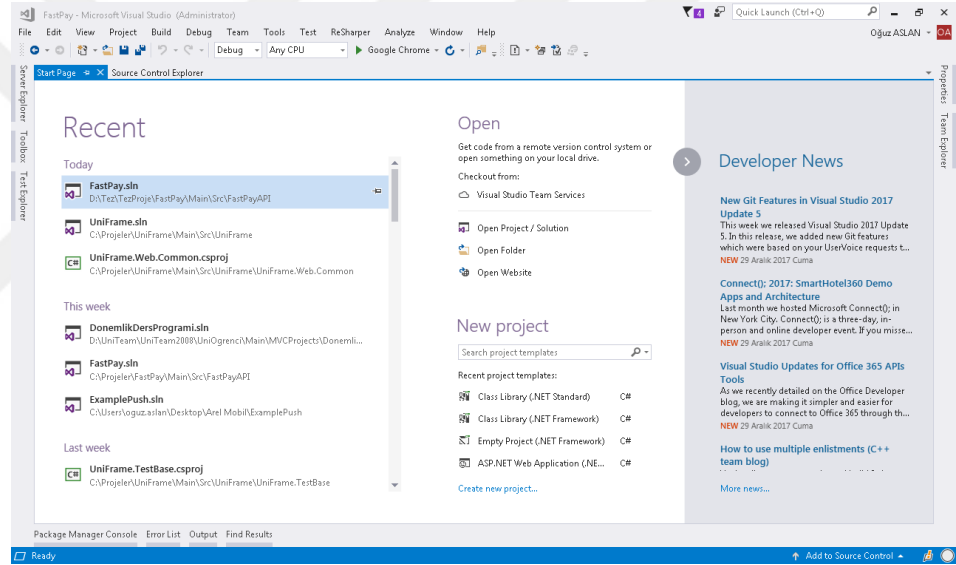
### 4.2.1 Microsoft Visual Studio

Microsoft Visual Studio, Microsoft tarafından geliştirilmiştir. Tümüleşik geliştirme ortamı olarak kullanılmaktadır. Bu tür yazılım geliştirme ortamları Integrated Development Environment (IDE) olarak adlandırılır ve Microsoft Visual Studio da bunlardan biridir.

Geliştiricilerin daha kolay yazılım ürünü ortaya koyabilmeleri için, geliştirme sürecini organize edebilen ve bir çok ek özellikler barındıran entegre bir yazılım geliştirme aracıdır.

Microsoft Visual Studio sayesinde geliştiricilere sağlanan bu kolaylıklar değerlendirildiğinde yazılım geliştirme sürecinin çok daha kısa sürede sonuçlandığı geliştiriciler tarafından belirtilmektedir.

Microsoft, kendi geliştirmiş olduğu teknolojileri yine kendi ürünü olan Microsoft Visual Studio IDE'si aracılığıyla geliştiricilere sunmaktadır. Bu sayede hem kendi ürünlerini pazarlamış oluyor hem de yazılım geliştirme süreçlerine katkıda bulunmuş oluyor. Bu bağlamda Şekil 4.3'de Microsoft Visual Studio arayüzü gösterilmiştir.



Şekil 4.3: Microsoft Visual Studio IDE arayüzü.

#### 4.2.2 Yazılım geliştirme süreci

“Akıllı mobil ödeme sistemi” projesinde dışarıdaki sistemlerin bir takım yöntemler kullanarak veritabanındaki bilgileri sorgulayabilmesi için web servis hazırlanmıştır. Web servis hazırlanırken Microsoft tarafından geliştirilen bir programlama dili olan C# kullanılmıştır.

Web servis geliştirilirken bir takım tasarım desenleri kullanılmıştır. Projede bunların en temelinde MVC Pattern gelmektedir. Açılımı Model View Controller

olan MVC ile bir problemin çözümüne ilişkin yapılan yaklaşımlar cevap bulmaktadır.

#### 4.2.2.1 Kullanılan yazılım geliştirme yaklaşımları

Bu bölümde geliştirilen projede kullanılan temel yazılım geliştirme yaklaşımlarından bahsedilecektir. Domain Driven Design (DDD) ve Test Driven Design (TDD) bunların başında gelmektedir.

- **Domain Driven Design (DDD)**

Geliştirilecek bir projenin boyunu ne kadar büyükse, o probleme veya o hizmete ait çözüm üretme süreci de o kadar sorunlu olmaktadır. Karmaşık ve büyük yazılım projeleri geliştirirken bir takım temel problemlerle karşılaşmaktadır. Domain Driven Design (DDD) bu temel problemlere çözümler üretir.

Domain Driven Design (DDD), Domain adını verdiği tüm süreçlerin veya tüm olayların birer domaini vardır ve her bir konunun tüm süreç ile ilgili işlemleri kendi Domain içerisinde gerçekleştirilir yaklaşımı ön plandadır.

Bir yazılım ürünü ortaya koyulurken konusunda uzman olan kişilere Domain Driven Design (DDD) yaklaşımında Domain Expert adı verilmektedir. Domain Expert'ler işlemi yapan en uzman kişiler olarak değerlendirilirler.

Yazılım projesi olarak tasarlanacak hizmeti, Domain Expert denen kişiler detaylıca anlatır. Geliştiriciler, bir probleme giden çözüm yollarını sınıflar, metotlar, polymorphism'ler, inheritance'ler, pattern'ler olarak görmektedirler. Aslında gerçek dünyadaki kişiler için bunlar hiçbir anlam ifade etmemektedir. Bu bağlamda Domain Expert'lerle aynı dili konuşabilmek gerekmektedir.

Domain Driven Design (DDD) yaklaşımında bu ortak dile Ubiquitous Language denilmektedir. Bu dil sayesinde Domain Expert'ler ile geliştiriciler aynı olayı farklı pencerelerden değil de aynı pencereden görmeye başlıyor. Her iki taraf da bu dili kullanarak etkileşimi sağlarsa bir problemin çok daha kısa sürede ve etkin bir biçimde anlaşılmasını sağlamak çok daha kolay olacaktır görüşünü savunmaktadır.

Domain Design Pattern (DDD), yazılımı modelleme sırasında katmanlı bir altyapı oluşturmayı hedefler. Bunlar, “Infrastructure Layer”, “Domain Layer”, “Application Layer” ve “Presentation Layer” olmak üzere dört katmandan oluşur.

Burada Domain Layer, sistemin en temel katmanını ifade etmektedir. Tüm domain objeleri, business rule’lar gibi domain ile ilgili tüm işlemler, işlevler ve kurallar burada tanımlanmaktadır. Domain Layer ayrıca diğer tüm katmanlardan bağımsız olmalıdır. Burada Loosely Coupled durumu önem arz etmektedir. Domain katmanı işin çözüldüğü kısım olarak bilinmelidir. Diğer tüm katmanlar Domain Layer’a ihtiyaç duyacaktır denebilir. Yani sistemin merkezinde Domain Layer, etrafında ise diğer katmanlar olarak düşünülebilir.

Kompleks bir sistemin tek seferde sorunsuz bir şekilde modellenmesi neredeyse imkansız gözükmemektedir. Bir proje modellendikten sonra değişime ve geliştirilmeye açık olması gerekmektedir. Bu bağlamda Refactoring önemli bir kavram olarak karşımıza çıkmaktadır. Tam bu aşamada bir projenin modellenmesinde net olunamayan kısımlarda Domain Driven Design (DDD) bir takım çözüm önerileri sunmaktadır.

Domain Driven Design (DDD) yaklaşımına göre bir problemin çözümünde Refactoring işlemleri sürekli devam edebilir durumda olmalıdır. Belirsiz olan her durum günü geldiğinde modellenebilir olmalıdır. Burada kodun öyle bir modellenmesi gerekmektedir ki farklı kişiler farklı zamanlarda kodun tamamı elden geçirilmeden bir takım düzenlemelerin veya eklemelerin yapılmasıyla sistemin tamamen bozulmaması gerekmektedir. Bu aşamada Domain Driven Design (DDD)’in üç adet öneri yöntemi bulunmaktadır. Bunlar “Constraint”, “Process” ve “Specification”dur.

Sonuç olarak Domain Driven Design (DDD) her projede kesinlikle kullanılması gereken bir yaklaşım değildir. Projenin boyutu, karmaşıklığı ve proje kapsamı burada önem arz etmektedir. Her projede Domain Driven Design (DDD) kullanılmamalıdır. Bu yaklaşım her projede kullanılabilecek bir kalıp değildir. Sadece belli karmaşıklığa, belli boyutta olan projelerde bir takım çözüm önerileri getiren teorik bir yaklaşımdır.

“Akıllı mobil ödeme sistemi” projesinde Domain Driven Design (DDD) yaklaşımı kısmen kullanılabilmektedir.

- **Test Driven Design (TDD)**

Literatürdeki diğer adı Test First Development (TFD) olarak geçmektedir. Extreme Programming (XP) development süreci oluşturucusu olan Kent Beck tarafından ortaya atılan bir metodolojidir. Günümüzde kullanılan birçok Agile ve Extreme Programming (XP) modern yazılım geliştirme süreçlerinin kodlanması bakımından bel kemiği vazifesi görmektedir.

Test Driven Design (TDD) denince akla şirkette çalışan ve sürekli uygulamaları test eden bir kişi akla gelmemelidir. Test Driven Design (TDD) demek yazılımı geliştirirken test işlemlerini bitirmiş olmak demektir. Kısaca yazılım geliştirme işlemlerinden bir aşamayı gerçekleştirirken önce test işlemleri yazılımsal olarak gerçekleştirilmekte, ardından ise bu test işlemlerinin çalıştırılacağı asıl işlemler yazılımsal olarak gerçekleştirilmektedir. Yaklaşım tam olarak bunu ifade etmektedir.

Test Driven Design (TDD)'e göre yazılım geliştirilmesinde temel olarak aşağıdaki adımlar izlenmektedir:

- Asıl kod yazılmadan önce test metodu yazılır.
- Test metodunu çalıştırılır ve test metodunun başarısız olduğunu görülür. Zaten asıl metot yazılmadığı için her türlü başarısız olacaktır.
- Test metodunda yazılan kodu başarılı bir şekilde sağlayan en basit asıl kod yazılır.
- Asıl kod Refactoring yapılır.
- Tekrar test işlemine tabi tutulur.

Test Driven Design (TDD) yaklaşımında yukarıda verilen yöntemler ne kadar basit olarak nitelendirilse de kodun okunabilirliği ve kalitesi bakımından oldukça önemli bir gelişimi ifade etmektedir. Bu sayede kodun kalitesi kat ve kat artmaktadır.

Herhangi bir projede basit bir olayı bile debug işlemine tabi tutmadan başarılı sonuç üretip üretmediği kolaylıkla Test Driven Design (TDD) yaklaşımına

göre test edilebilir olmaktadır. Genel hatlarıyla bir projenin ya da modülün beklenen kabaca tüm test senaryoları yazıldığında, koda yapılan dışarıdan bir müdahale sonrası hiçbir kod okuması yapılmadan sadece test metotlarının çalıştırılması sonucu projenin veya modülün ne kadar sorunsuz çalıştığı hakkında gelen bir izlenim sunabilmektedir.

Bu projede Test Driven Design (TDD) yaklaşımı aktif olarak kullanılmış ve yazılımın her bir metodu bu metodolojiye göre tasarlanmıştır. Test Driven Design (TDD) yaklaşımı benimsenmiş bu projenin Refactoring işlemi bu sayede çok daha kolay olabilmektedir.

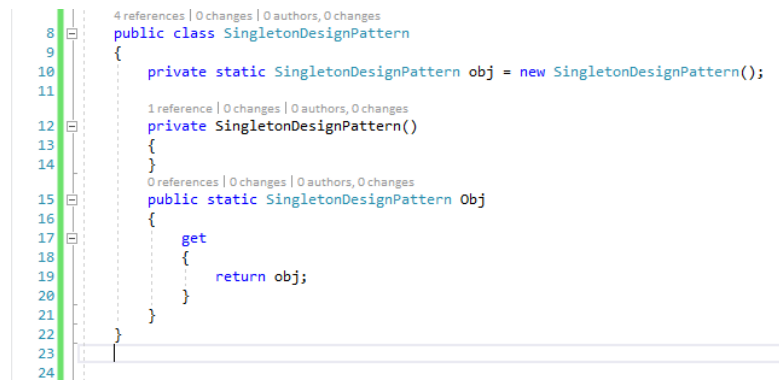
#### 4.2.2.2 Kullanılan Design Pattern'ler

Bu bölümde tez yazımında kullanılacak kağıdın ve yazıların niteliği, sayfa düzeni, satır aralıkları, sayfaların numaralandırılması açıklanmıştır. Projede, Singleton, Unit of Works ve Model View Controller (MVC) Pattern kullanılmıştır.

- **Singleton**

Singleton Design Pattern programın tüm yaşam süresi boyunca belirli bir objeden sadece bir instance'sinin olacağını garanti etmeye dayanır. Eğer bu object oluşturulmuşsa bu pattern sayesinde tekrar çağırım esnasında yeniden oluşturma işlemi yerine hali hazırda oluşturulan object referans edilerek oluşturulan tek object üzerinden hizmet sağlamayı hedeflemektedir.

Üç farklı Singleton Design Pattern kullanım örneği vardır. Bunlar, Şekil 4.4, Şekil 4.5 ve Şekil 4.6'da gösterilmiştir.



```

8 | 4 references | 0 changes | 0 authors, 0 changes
9 | public class SingletonDesignPattern
10 | {
11 |     private static SingletonDesignPattern obj = new SingletonDesignPattern();
12 |
13 |     1 reference | 0 changes | 0 authors, 0 changes
14 |     private SingletonDesignPattern()
15 |     {
16 |     }
17 |     0 references | 0 changes | 0 authors, 0 changes
18 |     public static SingletonDesignPattern Obj
19 |     {
20 |         get
21 |         {
22 |             return obj;
23 |         }
24 |     }

```

Şekil 4.4: Singleton Design Pattern 1. yöntemi.

```

10 public class SingletonDesignPattern : ISingletonDesignPattern
11 {
12     private static ISingletonDesignPattern obj = new SingletonDesignPattern();
13
14     private SingletonDesignPattern()
15     {
16     }
17     public static ISingletonDesignPattern Nesne()
18     {
19         return obj;
20     }
21 }

```

Şekil 4.5: Singleton Design Pattern 2. yöntemi.

Birinci ve ikinci yöntem arasında tek fark vardır. Fark, sadece iki sınıfın object'e erişim yöntemidir. İlk yöntemde nesneye property özelliği üzerinden erişim sağlanabiliyorken, ikinci yöntemde ise nesneye bir metod üzerinden erişim sağlanabilmektedir. Her iki erişim aracı da static olarak tanımlanmaktadır. Birinci yöntemde object "SingletonDesignPattern obj = SingletonDesignPattern.Obj;", ikinci yöntemde ise "SingletonDesignPattern obj = SingletonDesignPattern.Nesne()" olarak tanımlanırsa ilgili object zaten doğrudan oluşturulmuş olarak iletilecektir.

Bu sınıfı daha kullanışlı ve dinamik bir hale getirebilmek için oluşturulacak object'in sadece istenildiği durumda oluşturulması sağlanabilir. Bunun gerçekleştirilmesi durumu da üçüncü versiyonda mümkündür. Üçüncü sürüm Şekil 4.6'da gösterilmiştir.

```

9 public class Singleton : ISingletonDesignPattern
10 {
11     private static ISingletonDesignPattern obj;
12
13     private Singleton()
14     {
15     }
16     public static ISingletonDesignPattern Nesne()
17     {
18         if (obj == null)
19             obj = new SingletonDesignPattern();
20
21         return obj;
22     }
23 }

```

Şekil 4.6: Singleton Design Pattern 3. yöntemi.

Gerçekleştirilen projede ağırlıklı olarak üçüncü yöntem kullanılmaktadır.

- **Unit of Works**

Unit of Work (UoW) Design Pattern'in temel olarak anlamı, veritabanında birden fazla işlem gerektiren bir sürecin tek bir bağlantı üzerinden yürütülmesi ve bu işlemlerin tamamlanana kadar bir hafızada tutularak olası olumsuzluk durumlarında erişilebilir olması durumunu ifade etmektedir.

Özellikle kritik işlemler gerektiren kısımlarda kullanımı yaygındır. Yazılım karşılığı olarak "Transaction" işleminin Design Pattern'e büründürülmüş hali denebilir.

Örneğin, bir banka işlemi gerçekleştirilmek istenmektedir. Bir banka hesabından diğer banka hesabına para transferi gerçekleştirilecektir. Bu durumda yapılması gereken işlemler basittir. Transfer edilmesi gereken miktarın gönderici hesabından düşülmesi ve aynı miktarın da alıcı hesabına eklenmesi gerekmektedir.

Tam bu aşamada gönderici hesabından transfer edilecek miktar düşüldükten sonra olası sistemsel veya yazılımsal bir sorun neticesinde transfer edilecek kişinin hesabına para miktarı ilave edilemedi. Çözüm olarak, transfer eden kişinin hesabındaki paranın bir şekilde geri iade edilmesi gerekmektedir. Burada hafızada tutulması gereken bilgiler hangi kişinin hangi hesap numarasından ne kadarlık bir para miktarı düşüldü. Bu bilgi eğer bir şekilde saklanmamış olursa sadece gönderici hesabından para düşülmüş olacak ve ayrıca alıcı hesabına da para aktarılmamış olacaktı. Transfer edilecek para havada kalacak ve yanlış bir işlem söz konusu olacaktır.

Verilen örneğe göre birden fazla işlem gerektiren bir Transaction'un sorunsuz bir şekilde tamamlanması veya işlemin başında, ortalarında veya sonlarına doğru olası herhangi bir sorun neticesinde tüm işlemlerin geri alınabilmesini sağlayan yapıdır.

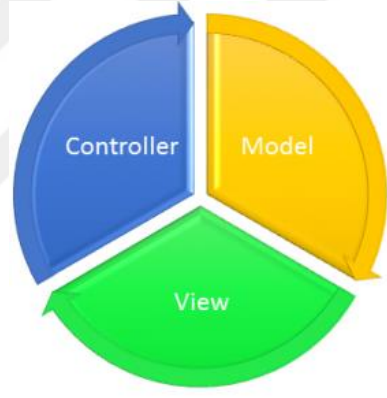
Projede, Unit of Work (UoW) kavramı ağırlıklı olarak ürünlerin satın alınma esnasında kullanıcının hesabından para çekilmesinden sonra eğer satın alınan ürünler "satın alındı" durumuna getirilemezse kişiler ürünleri satın alamama durumlarına düşebilir. Bu durumda kişilerin hesaplarından alınan paraların iade edilmesi gerekmektedir. Unit of Work (UoW) kavramı verilen

örneğe göre kullanımı söz konusu olmaktadır. Projede bunun gibi birkaç yerde daha Unit of Work (UoW) kavramı etkin olarak kullanılmaktadır.

- **Model View Controller (MVC)**

Model View Controller (MVC), özellikle web uygulaması geliştirmede piyasada etkin rol oynamaktadır. Günümüzde Model View Controller (MVC) denince Microsoft'un geliştirdiği ASP .NET MVC ilk olarak akla gelmektedir. Microsoft'un yanı sıra Model View Controller (MVC) pattern'i kullanan birçok teknoloji devî mevcuttur.

Model View Controller (MVC) pattern'inde her bir kelime aslında bir katmana karşılık gelmektedir. Her bir katmanın da kendine has özellikleri mevcuttur. Şekil 4.7'de Model View Controller (MVC) pattern'i katmanları gösterilmektedir.



Şekil 4.7: Model View Controller (MVC) Pattern ve katmanları.

Model View Controller (MVC) pattern'indeki Model katmanı, uygulama verisinin veya o veriye ait durum bilgilerinin tutulduğu kısımdır. Model katmanı daha çok veri katmanını yazılımın kullandığı uygulamadan izole etmektedir. Böylece diğer katmanlarda işlem yapılırken veri katmanının nerede olduğunun bilinmesine gerek kalmamaktadır.

Model katmanı olarak yine Microsoft'un bir ürünü olan EntityFramework kullanılabilir. Bu teknolojinin alternatifi olarak ise NHibernate ve özellikle günümüzde çok yeni ve güçlü bir Object Relational Model (ORM) tool olan Dapper örnek olarak gösterilebilir.

Model View Controller (MVC) pattern'deki View katmanı, istemci tarafında görüntülenmesi istenen tüm arayüzleri içeren kısımdır. Genellikle Model katmanında olan veriyi kullanıcılara göstermek amacıyla geliştirilmektedir. Aynı zamanda kullanıcılardan alınması gereken veriler için de kullanılmaktadır.

View katmanının diğer katmanlardan ayırmak demek, arayüzde herhangi bir değişiklik yapılması gerektiğinde diğer katmanlara (Model veya Controller) müdahale etmeden değişikliklerin yapılabilmesine olanak sağlamasıdır.

Model View Controller (MVC) pattern'deki Controller katmanı, kullanıcılardan veya sistemlerden (bunların tümü client olarak değerlendirilebilir) gelen tüm istekleri işleyen, yöneten veya sonuçlandıran kısımdır. Bu katman komuta merkezi olarak da adlandırılabilir. Tüm Business Logic (BL) işlemlerinin gerçekleştirildiği katmandır. Model ve View arasında köprü görevi görür.

Controller katmanı örneğin View katmanından almış olduğu verileri Model katmanına ileterek modellemeler yapabilirken aynı şekilde, Object Relational Model (ORM) kullanarak veritabanından almış olduğu bir veri setini Model katmanı vasıtasıyla modelleyerek View katmanına kullanıcılara sunması için gönderebilir.

Sonuç olarak Model View Controller (MVC) pattern günümüzde birçok programlama dilinde temel olarak kullanılmaktadır. Java, C#, PHP, Angular JS bu pattern'i kullanan teknolojilere örnek olarak gösterilebilir.

#### **4.2.2.3 Data Transfer Objects**

Data Transfer Object (DTO), yazılımda katmanlar arasında veri taşımak için kullanılan bir Data Container'dir. Bunlara transfer nesneleri de denir. Data Transfer Object (DTO), yalnızca verileri iletmek için kullanılır ve herhangi bir iş mantığı içermez. Yazılım dilinde basitçe setter'ları ve getter'ları vardır.

#### **4.2.2.4 Garbage Collector**

Yazılımda sunucu tarafında gelen bir süreç isteği sonrasında bu süreç başlatılırken bellekteki boş bir alan bu süreç için tahsis edilir. Buna Managed Heap denir. Managed Heap içerisinde bu süreçle ilişkin tutulan verilerin kullandığı bellek alanlarının adresleri yer almaktadır.

Sunucu tarafında gelen bu süreç isteği sonrasında oluşturulan nesneler, belirli bir zaman sonra uygulama tarafından kullanılmadıklarında atıl durumda kalırlar ve bellekten temizlenirler. Sunucu tarafında bu bellek verilerinin temizlenmesini yöneten mekanizma Garbage Collector'dur.

Uygulamanın sunucudaki runtime esnasında yazılım tarafından çalıştırılan her bir “new” operatörü için Managed Heap denilen belleğe gidilerek, süreç için ayrılan belleğin dolu olup olmadığı kontrolü yapılır. Eğer ayrılan bu bellek dolu ise Garbage Collector devreye girer.

Bu aşamada, uygulama processlerinin ihtiyaç duymadığı ve kullanılmayan bellek alanları veya objeler Managed Heap denilen bellekten silinir. Zaten yazılımcıların sıkça karşılaştığı bir sorun olan “OutOfMemoryException”, Managed Heap belleğin temizlenecek atıl durumda kalan objelerinin olmamasından dolayı bir hatadır.

Managed Heap'de belleği gösteren ilk adres, uygulamanın başlangıç adreslerini ifade etmektedir. Garbage Collector'un yaptığı Managed Heap'de bulunan tüm başlangıç noktaları arasındaki tüm objeleri incelemek ve kullanılmayanları temizlemektedir. Garbage Collector, Managed Heap'de bulunan tüm alanları potansiyel silinebilir object olarak değerlendirmektedir.

Garbage Collector, arka planda çizge teorilerine dayanan bir algoritma kullanır ve herhangi bir nesne Managed Heap'de iki kere yer alıyorsa, kullandığı çizge teorilerine dayanan algoritma sayesinde bunu teke düşürebilmektedir.

Tüm Managed Heap dolaşıldıktan sonra, elde edilen linear graph içerisinde her bir object'den sadece bir adet olması durumunu garanti eder. Bu işlem tamamlandıktan sonra tüm Managed Heap object'leri yeniden yerleştirilir ve her object'e yeni bir adres atanır. Bu sayede Managed Heap içerisinde yazılım tarafından kullanılan aynı nesnelerin çoklaması problemi çözümlenmiş olur.

“Akıllı mobil ödeme sistemi” sunucu web servis yazılımı Internet Information Services (IIS) üzerinde tanımlanarak çalıştırılması planlandığı için Internet Information Services (IIS)'in kullandığı Garbage Collector da bu sayede kullanılmış olacaktır.

#### 4.2.2.5 Exception Handling

Yazılım sektöründe gerçekleştirilen bazı yazılımlar, yazılımın gerçekleştirme aşamasında hiçbir sorun üretmezken proje, “production” denilen gerçek dünyada yerini aldığı anda bir takım sorunlar üretebilir.

Tam bu aşamada, Exception Handling devreye girmektedir. Sistemin tahmin edilebilir hata üretmeleri durumunda, .NET Framework’un kendisi tarafından sağlanan ”System.Exception” sınıfı kullanılabilir. Ancak, tahmin edilemeyen ve sadece yazılıma veya projeye özgün ortaya çıkabilecek bir hata oluşumu durumunda ise yazılımcının kendi hata sınıfını yazmak durumunda kalacaktır.

Tahmin edilebilen veya edilemeyen tüm hataları yakalayabilmek ve onarım sağlayabilmek için yazılım tarafında kullanılması tavsiye edilen birkaç yöntem mevcuttur. Bunlardan en popüler olanı “try-catch bloğu” denilen yazılım syntax’ının kullanılmasıdır.

Bu sayede “try” ve “catch” blokları arasında oluşabilecek olası bir hatanın meydana gelmesi durumunda, “catch” bloğu sayesinde oluşabilecek hatanın maksimum tahribatını minimuma indirmek mümkündür. “Try” içerisinde oluşan herhangi bir hata sonrasında yazılım “catch” bloğu içerisinde devam edeceği için bu işlemler Handle edilebilir olmaktadır.

“Akıllı mobil ödeme sistemi”nde kullanılan örnek bir Exception Handling örneği Şekil 4.8’de verilmiştir.



Şekil 4.8: Projede kullanılan bir Exception Handling örneği.

#### 4.2.2.6 Kullanılan arabirimler

Tam olarak arabirimler, soyut sınıfları ifade etmektedir. Yani, bir arabirimin içerdiği fonksiyonlar soyut denen kavramlardan olmalıdır. Arabirimler, kullanıldığı yere veya projeye göre ihtiyaç doğrultusunda hizmet verebilir olmalıdır.

.NET Framework tarafından ortaya sürülen kullanışlı birçok arabirim söz konusudur. “Akıllı mobil ödeme sistemi” projesi içerisinde bir takım arabirim kullanımları söz konusu olmuştur. Projede en çok kullanılan arabirimler aşağıdaki gibidir.

- **IDisposable arabirimi**

Uygulamada bir process’in kullandığı her bir bellek alanı Garbage Collector başlığı altında bahsedildiği gibi, Managed Heap içerisinde adresler şeklinde saklanmaktadır. “Akıllı mobil ödeme sistemi”nde “new” ile instance’lanan her bir nesne veya Data Transfer Object, kullanımlarının ardından Garbage Collector devreye alınarak Managed Heap’den manuel olarak temizlenmesi sağlanır ki Garbage Collector otomatik olarak devreye girene kadar bellek optimizasyonu korunmuş olsun. Tüm projede Garbage Collector ile optimum bellek kullanımı minimum seviyede sağlanmış bulunmaktadır.

“Akıllı mobil ödeme sistemi”nde kullanılan örnek bir IDisposable arabirimi kullanımı Şekil 4.9’daki gibidir.

```

8 public class VerificationModel : ICloneable, IDisposable
9 {
10     #region Properties
11
12     /// <summary>
13     /// </summary>
14     5 references | 0 exceptions
15     public string VerificationCode { get; set; }
16
17     /// <summary>
18     /// </summary>
19     6 references | 0 exceptions
20     public VerificationType VerificationType { get; set; }
21
22     #region IDisposable Members
23
24     25 references | 0 exceptions
25     public void Dispose()
26     {
27         VerificationCode = null;
28         VerificationType = VerificationType.Default;
29         GC.SuppressFinalize(this);
30     }
31
32     #endregion

```

Şekil 4.9: Projede kullanılan IDisposable arabirim örneği.

- **ICloneable arabirimi**

Güncel yazılım dillerinde değişkenler iki farklı şekilde tutulmaktadır. Bunlar değer tipi ve referans tipi değişkenlerdir. Değer tipi değişkenler, belleğin “stack” denilen bölgesinde tutulur. Referans tipi değişkenler ise “heap” bölgesinde tutulmaktadır.

“Akıllı mobil ödeme sistemi” geliştirme aşamasında kullanılan değişkenler referans tipli değişkenler olduğundan dolayı, değişkenlerin birbirlerine atanması veya yeni bir değişkene aktarılması durumunda o değişkenin sahip olduğu veriler, veri olarak diğer değişken üzerine aktarılmaz. Sadece o değişkenin sahip olduğu verilerin tutulduğu bellek bölgesinin adres bilgileri aktarılır. Bu demek oluyor ki, verilerin atanmış olduğu değişkene yeni bir değer ataması yapıldığı zaman birinci değişkenin de referans tipi değişken kullanıldığından dolayı atanan değer olarak güncellenmektedir.

Bu aşamada “akıllı mobil ödeme sistemi”, Data Transfer Object (DTO)’ların tümü olmak üzere kullanılan tüm referans tipi değişkenlerde ICloneable Arabirimi kullanılmış ve yukarıda bahsedildiği şekilde verinin “runtime” esnasında öngörülemeyen değişimi önlenmiş olmaktadır.

“Akıllı mobil ödeme sistemi”nde kullanılan örnek bir ICloneable arabirimi kullanımı Şekil 4.10’daki gibidir.

```

7 public class LogoutModelOut : ICloneable, IDisposable
8 {
9     #region Properties
10
11     /// <summary>
12     6 references | 0 exceptions
13     public bool IsLoggedOut { get; set; }
14
15     /// <summary>
16     6 references | 0 exceptions
17     public string LogoutErrorMessage { get; set; }
18
19     #endregion
20
21     #region ICloneable Members
22
23     /// <summary>
24     29 references | 0 exceptions
25     public object Clone()
26     {
27         return new LogoutModelOut
28         {
29             IsLoggedOut = IsLoggedOut,
30             LogoutErrorMessage = LogoutErrorMessage
31         };
32     }
33
34     #endregion
35 }

```

Şekil 4.10: Projede kullanılan ICloneable arabirim örneği.

- **Serialization**

Serialization işlemi, bir veri veya veri kümesinin belirli standartları olan ve kolayca erişilebilir durumdaki bir forma dönüştürme işlemidir.

“Akıllı mobil ödeme sistemi” içerisinde kullanılan mobil uygulamaya gönderilen tüm veriler veritabanından Structured Query Language (SQL) kullanılarak sorgulandıktan sonra bu veriler Domain Driven Design (DDD) yaklaşımında kullanılan Domain Rule’lardan geçirildikten sonra Data Transfer Object (DTO) denilen transfer nesnelerine iz düşürülür.

Daha sonra elde edilen bu Data Transfer Object (DTO)’ler sayesinde veri, bir takım kurallar işleminden geçerek en üst katman olan Application Layer’a kadar gelir. En son aşamada veri Serialization işlemine tabi tutularak Hyper Text Transfer Protocol (HTTP) üzerinden mobil uygulamaya gönderilir.

Hyper Text Transfer Protocol (HTTP) üzerinden mobil uygulamaya gönderilen her veri veya veri seti Serialization işlemine tabi tutularak Javascript Object Notation (JSON) veriler haline dönüştürülür.

“Akıllı mobil ödeme sistemi”nde kullanılan örnek bir Serialization kullanımı Şekil 4.11’deki gibidir.

```

66 public static class JsonSerializer<TType> where TType : class
67 {
68     // Usage : JsonSerializer<MusicInfo>.Serialize(musicInfo);
69     /// <summary>
70     /// Serializes an object to JSON
71     /// </summary>
72     0 references | 0 exceptions
73     public static string Serialize(TType instance)
74     {
75         var serializer = new DataContractJsonSerializer(typeof(TType));
76         using (var stream = new MemoryStream())
77         {
78             serializer.WriteObject(stream, instance);
79             return Encoding.Default.GetString(stream.ToArray());
80         }
81     }
82     /// <summary> DeSerializes an object from JSON
83     0 references | 0 exceptions
84     public static TType DeSerialize(string json)
85     {
86         using (var stream = new MemoryStream(Encoding.Default.GetBytes(json)))
87         {
88             var serializer = new DataContractJsonSerializer(typeof(TType));
89             return serializer.ReadObject(stream) as TType;
90         }
91     }
92 }
93

```

Şekil 4.11: Projede kullanılan Serialization örneği.

#### 4.2.2.7 Kullanılan yazılım geliştirme teknolojileri

Proje modüllerinden biri olan web servis geliştirilmesinde kullanılan bir takım teknolojiler, frameworkler mevcuttur. Kullanılan bu teknolojiler aşağıdaki gibidir.

- **.NET Framework**

Framework kavramı denildiğinde akla öncelikle, içerisinde Application Programming Interface (API)'lerin ve programların ihtiyaç halinde kullanabilecekleri ve sıkça kullanılabilen yazılım kütüphanelerini barındıran yazılım paketleri gelmektedir.

.NET Framework, içerisinde programların sıklıkla ihtiyaç duyabileceği Framework Class Library (FCL) şeklinde kütüphaneler barındırmaktadır. Bu kütüphane metotları sayesinde sıklıkla ihtiyaç duyulan işlevler için tekrar tekrar kod yazılması gerekmiyor. Sıklıkla kullanılacak tüm işlevlerin tek bir pakette toplanması olarak da nitelendirilebilir.

“Akıllı mobil ödeme sistemi” web servis geliştirmesi esnasında .NET Framework kütüphanesine ait birçok işlev, yöntem ve metot aktif olarak kullanılmıştır.

- **Microsoft ASP .NET MVC**

Daha önceden bahsedildiği gibi Model View Controller (MVC) Pattern yazılım geliştirme süreçlerinde etkin bir rol oynamaktadır. Katmanların birbirlerine bağımlılığını azaltan Loosely Coupled prensibine uygun bir yöntemdir. Microsoft da bu Pattern'i kullanarak Microsoft ASP .NET MVC teknolojisini piyasaya sürmüştür.

Model View Controller (MVC) Pattern birçok programlama dilinde ve framework'lerde kullanılmış ve günümüze kadar olgunlaşmış bir şekilde gelmiş bir Design Pattern'dir. Bu şekilde geliştirilen bir projenin çok daha hızlı çalıştığı, kolay test edilebilir olduğu ve tekrar kullanılabilir parçalar haline getirilebilmesi durumunu kanıtlamıştır.

Sunucu tarafı yazılımı olan web servis geliştirilmesinde Microsoft ASP .NET MVC kullanılmıştır.

- **Microsoft ASP .NET MVC Web API**

Microsoft .NET Framework'ün yeni bir sürümü ile piyasaya çıkan MVC Web API ile Representational State Transfer (REST) servisler hazırlamak mümkündür (Uğurlu, T., Zeitler, A., 2013). ASP.NET MVC Web API, client ile server arasında Hyper Text Transfer Protocol (HTTP)'lerini kullanarak haberleşmeyi sağlayan bir yapıdır. Bu yapı sayesinde mobil uygulama ile veritabanı rahatlıkla haberleştirilebilmekte ve veri transferleri sağlanabilmektedir (Microsoft, 2017).

#### **4.2.2.8 Servisler**

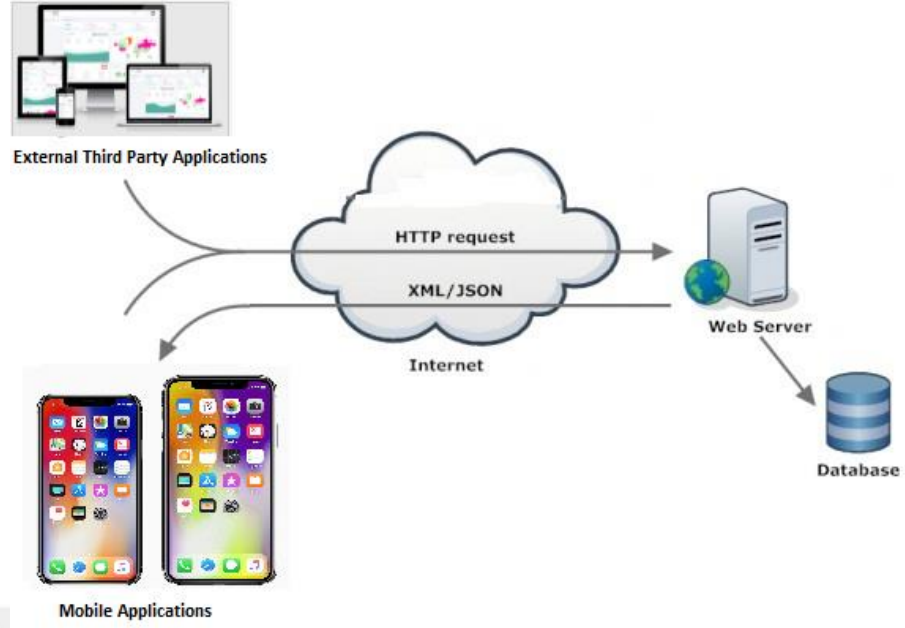
Web servisler, Hyper Text Transfer Protocol (HTTP) protokolü ile bir takım veri taşıma teknolojilerini kullanarak iki cihaz arasındaki haberleşmeyi sağlamak amacıyla geliştirilmiştir. Bu veri taşıma teknolojilerine Extensible Markup Language (XML) ve JavaScript Object Notation (JSON) örnek olarak gösterilebilir.

Web servisler cihazlararası haberleşmeyi sağlayan yazılım parçalarıdır. Buradaki asıl amaç platformdan bağımsız olacak şekilde sistemlerin haberleşebilmelerini sağlayabilmektir.

Web servis teknolojisinde servisin hangi platformda ya da dilde geliştirildiğinin bir önemi yoktur. Örneğin PHP'de yazılmış bir web servise, farklı bir sunucuda çalışan ve .NET ile geliştirilmiş bir yazılım kolaylıkla erişebilir. Burada tek ortak nokta tüm programlama dillerinin ortak olarak kullandığı bir protokol olan Hyper Text Transfer Protocol olmasındandır.

Tüm programlama dilleri, Hyper Text Transfer Protocol standardını temel olarak kabul etmiştir. Bu sayede veri, sistemler arası haberleşme programlama dilinden bağımsız olarak tümünde ortak olan Hyper Text Transfer Protocol (HTTP) üzerinden kolaylıkla sağlanabilmektedir.

İstemcilerin veriye erişebilmeleri açısından Web servisin önemini Şekil 4.12 anlatmaktadır.



Şekil 4.12: Web servis mimarisi örneği.

- **Simple Object Access Protocol (SOAP) servisler**

Simple Object Access Protocol (SOAP) servisler, dağıtık sistemler arasında veri transferi yapmak amacıyla kullanılan bir servis yöntemidir. Sistemler arası haberleşmede etkim bir rol oynamaktadır. Hyper Text Transfer Protocol (HTTP), verileri bünyesinde barındırırken kendisinin kabul edemediği sıra dışı nesneleri taşıyamamaktadır. Bu aşamada Simple Object Access Protocol (SOAP), Hyper Text Transfer Procotol (HTTP)'nin bu açığı kapatmaktadır.

Simple Object Access Protocol (SOAP) servislerin olumsuz yanlarından bir tanesi veri gönderiminin sadece tek yönlü olmasıdır. Projede ilk etapta Simple Object Access Protocol (SOAP) servisi araştırılmış olsa da bir takım olumsuzluklarından ve Representational State Transfer (REST) servislerin platform bağımsızlığı özelliklerin dolayı tercih edilmemiştir.

- **Representational State Transfer (REST) servisler**

Representational State Transfer (REST), “client-server” yaklaşımını rahatlıkla kullanabilen ve bu yönde iletişimini gerçekleştiren bir mimaridir. Hyper Text Transfer Protocol (HTTP) ile aynı doğrultuda geliştirilmiş olmasıyla birlikte herkesin bildiği World Wide Web altyapısını da kullanmaktadır.

Representational State Transfer (REST) mimarisini kullanan tüm servisler genelde RESTful servisler olarak adlandırılmaktadır. Buradaki en önemli nokta servis tabanlı diğer mimarilerden birkaçı olan Simple Object Access Protocol (SOAP) ve Windows Communication Foundation (WCF)'in aksine veri transferi daha hızlı ve adapte edilmesi daha kolaydır.

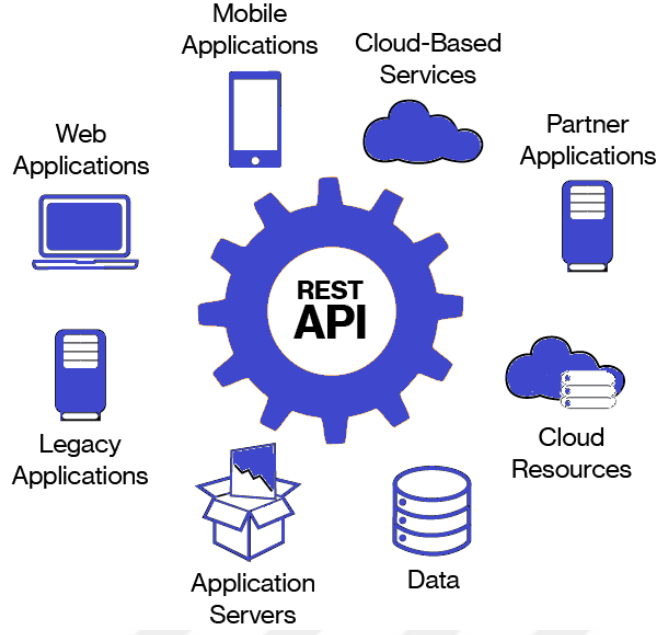
Kullanımları diğer servis teknolojilerine göre çok daha basittir. Aynı zamanda esnetilebilirlerdir ve yetenekleri diğer web servislere nazaran daha fazladır.

Bir takım kurallara sahip olsa da Simple Object Access Protocol (SOAP) gibi çok katı kuralları olmaması onu cazip kılan en önemli özelliklerden biridir. Representational State Transfer (REST) servisler Hyper Text Transfer Protocol (HTTP) üzerinde çalışmaktadır. Sistemler arası veri transferleri, Extensible Markup Language (XML) veya JavaScript Object Notation (JSON) türündeki nesneler aracılığıyla sağlanmaktadır.

Representational State Transfer (REST) servisler, “Stateless” olarak çalışmaktadır. Bu demek oluyor ki client tarafında, haberleşmek istediği RESTful servis ile ilgili herhangi bir durum bilgisi barındırmasına gerek yoktur. Bunun anlamı her bir veri transferi esnasında veri ile birlikte detaylı başlık bilgileri ve client’a özel bilgiler taşınmaz. Sadece ham veri iletilmektedir. Bu şekilde bir yapının kullanılabilir olması, network trafiğini azaltmakla beraber, network üzerinde taşınan verinin boyutunun küçülmesiyle birlikte verinin çok daha hızlı iletilmesine olanak sağlamaktadır.

Herhangi bir RESTful servisi çağırabilmek için, RESTful servisin bulunduğu ve çalışır halde olduğu servis url bilgisi basit bir sorgu için yeterli olacaktır.

RESTful servisler, Hyper Text Transfer Protocol’ü kullandığından Hyper Text Transfer Protocol istekleri olan GET, POST, PUT, DELETE metotları kullanılarak işlemlerini gerçekleştirir. Bu bağlamda Representational State Transfer (REST) mimarisi Şekil 4.13’te gösterilmiştir.



Şekil 4.13: Representational State Transfer (REST) mimarisi.

“Akıllı mobil ödeme sistemi” web servisi tamamıyla Representational State Transfer (REST) mimarisi üzerinde kurgulanmıştır. Bu demek oluyor ki bu servis sadece bir mobil uygulama tarafından kullanılmayabilir.

Herhangi bir sistem diğer sistemlere herhangi bir şekilde istekte bulunma özelliği var ise gerçekleştirilen bu servis ile haberleşebilir, veri transferinde ve web servisin sunduğu tüm işlevleri yerine getirebilir.

#### 4.2.2.9 Ağ protokolleri

Representational State Transfer (REST) web servis mimarisi ve projede kullanılan diğer yazılım teknolojileri Hyper Text Transfer Protocol (HTTP) ve HTTP Request’ler (GET, POST, PUT, DELETE) gibi bir takım ağ protokollerini kullanmıştır. Dolayısıyla projenin geliştirilme aşamasında yoğun olarak kullanılan ağ protokolleri aşağıdaki gibidir.

- **Hyper Text Transfer Protocol (HTTP)**

Hyper Text Transfer Protocol (HTTP), sistemler arası veri iletişimine kendince kurallar koyan ve benzer veri haberleşmesi yöntemlerinin tümünde kendi belirlemiş olduğu kuralların kullanılmasını isteyen bir protokoldür.

İnternet tarayıcısı üzerinden bir adres girildiğinde, ekranda url'ye ait bilgileri görüntüleyebilmek ve client sistemin ekranına verileri getirebilmek için Hyper Text Transfer Protocol (HTTP) üzerinden istek yapılır. Sonuç da yine aynı protokol kullanılarak iletilir.

Mobil uygulama ile firma verilerinin tutulduğu veritabanının konuşturulabilmesi için web servis geliştirilmesi söz konusu olmuştur. Bu bağlamda mobil uygulama ile web servisin konuşabilmesi işlemi Hyper Text Transfer Protocol (HTTP) üzerinden sağlanacaktır.

- **HTTP Request'ler (GET, POST, PUT, DELETE)**

Herhangi bir tarayıcı üzerinden bir web sayfası görüntülenmek istendiğinde, görüntülenmesi istenen web sayfası sunucusuna bir request gönderilir. Yapılan request, sunucu tarafından değerlendirilerek geri dönmesi için bir response üretir.

“Akıllı mobil ödeme sistemi”nde kullanılan bir takım HTTP Request metotları aşağıdaki gibidir:

- ✓ **GET:** Get metodu ile basit istekler sunucudan talep edilebilir. Bu isteklere parametre verebilmek için url kısmında “Query String” denilen parametrik yapılar eklenebilir. Sadece ASCII karakterler parametre olarak gönderilebilmektedir.
- ✓ **POST:** Veri güncellenmesi veya yeni bir veri oluşturulması ihtiyacı durumlarında kullanılması önerilen yöntemdir. Bu yönetime göre karmaşık istekler veya form gönderimleri sağlanabilmektedir. Sınırsız sayıda parametre gönderimini desteklemektedir. POST ile sunucuya gönderilecek veriler gönderim esnasında formun gövde kısmına yerleştirilir ve bilgiler sunucuya o şekilde gönderilir. ASCII karakterlerin yanında binary data da gönderilebilir.
- ✓ **PUT:** Bu metot ile sunucu tarafında saklanan bir verinin güncelleme işlemi yapılabilir. Güncellenecek verinin ne olduğunu sunucu tarafın bilebilmesi için request oluşturulmadan önce güncellenecek verinin ID bilgisi request içerisine eklenmesi gerekir.

- ✓ **DELETE:** Bu metot ile sunucu tarafında saklanan bir verinin silinmesi işlemi yapılabilir. Silinecek verinin ne olduğunu sunucu tarafın bilebilmesi için request oluşturulmadan önce silinecek verinin ID bilgisi request içerisine eklenmesi gerekir.

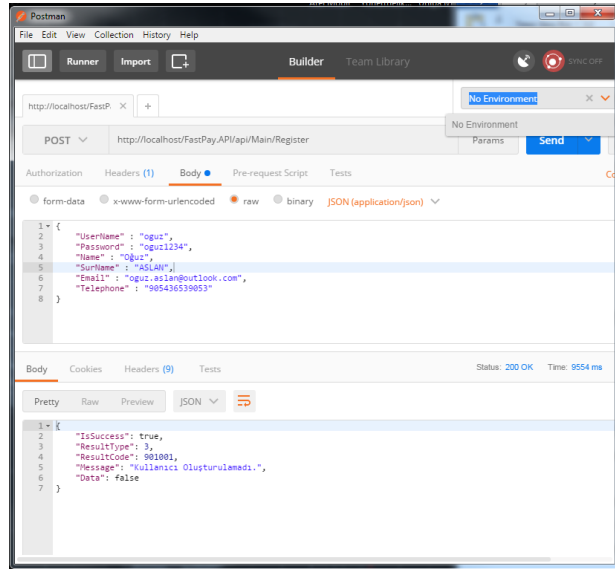
- **Postman**

Yazılım geliştirme işlemleri, “akıllı mobil ödeme sistemi”nde olduğu gibi iki farklı şekilde olmaktadır. Mobil uygulama geliştirme işlemleri daha çok “front-end” olarak nitelendirilirken, web servis ve altyapı ile ilgili çalışmalar “back-end” olarak nitelendirilmektedir.

Eğer yazılım geliştirme işleminde web servis geliştirmesi yapılıyorsa, bu web servisi kullanacak sistemlerden gelen requestler oluşturulmalıdır. Tüm sistemler üzerinden yapılacak request test işlemleri için bir takım araçlar mevcuttur. Bu araçlardan bir tanesi Postman’dır.

“Akıllı mobil ödeme sistemi”nde yazılan web servis metotlarının test edilebilmesi açısından test request’leri oluşturabilmek için Postman kullanılmıştır.

“Akıllı mobil ödeme sistemi” web servis tarafında yazılan bir servis metodunu test edebilmek mümkündür. Bu bağlamda Postman üzerinden gösterilen bir test request örneği Şekil 4.14’te gösterilmiştir.

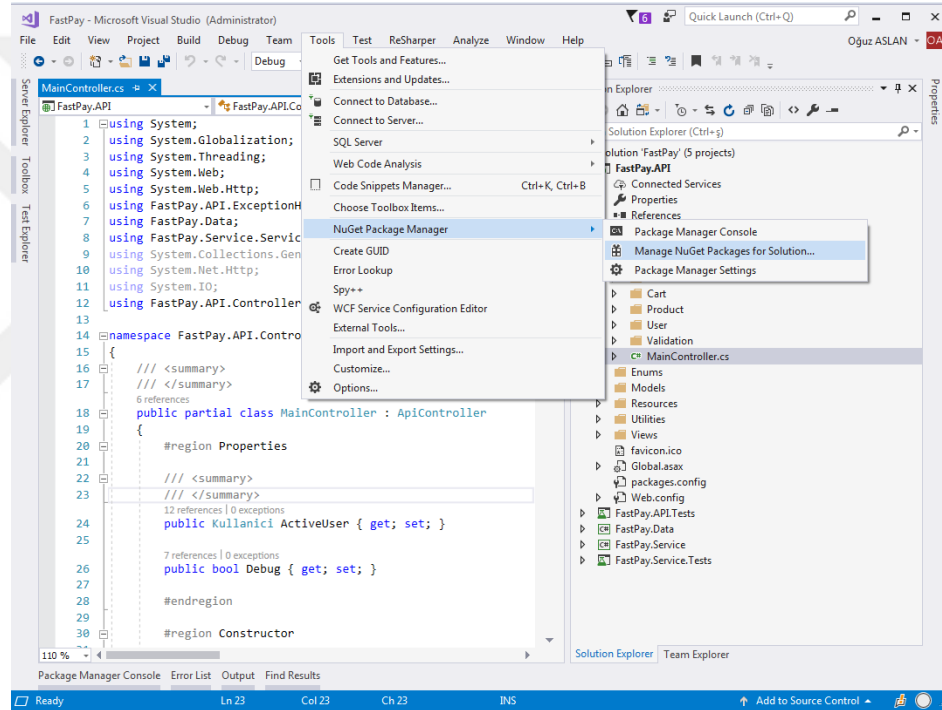


Şekil 4.14: “Akıllı mobil ödeme sistemi” için kullanılan örnek bir POST isteği.

#### 4.2.2.10 Nuget Package Manager

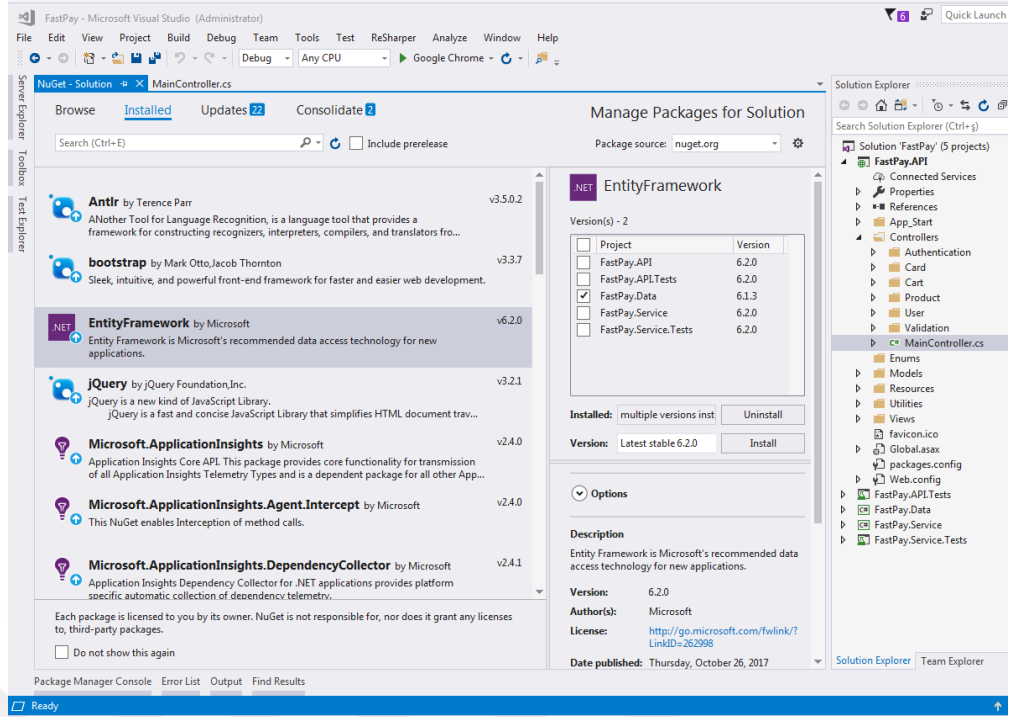
Bir proje geliştirirken ihtiyaç duyulan 3rd-party paket ve kütüphanelerin indirilip projeye eklenebileceği ve bu paket veya kütüphanelerin kolayca yönetilebileceği bir araçtır. Nuget Package Manager, Microsoft tarafından Visual Studio’ya eklenerek hizmet vermektedir.

Microsoft Visual Studio üzerinden Nuget Package Manager kullanarak bir projeye 3rd-party bir kütüphane ekleyebilmek için Şekil 4.15’te gösterildiği gibi “Tools” sekmesinden “Nuget Package Manager” seçilir. Buradan da “Manage Nuget Packages for Solution” seçilerek ilgili ekrana erişilebilir.



Şekil 4.15: Nuget Package Manager erişimi.

Nuget Package Manager’e eriştikten sonra geliştirilen projeler için hangi kütüphanelerin ekleneceği bilgileri araştırılabilmektedir. Örneğin Entity Framework kütüphanelerini projeye ekleyebilmek için Şekil 4.16’da görüldüğü üzere Entity Framework kütüphanesi seçilir. Daha sonra Entity Framework kütüphanesini açılan Solution’daki hangi projelere ekleneceğine karar verilir. En sonunda da Install butonuna basılarak ilgili kütüphane istenen projelere eklenmiş olur.



Şekil 4.16: Nuget Package Manager ile 3rd-party bir kütüphanenin eklenmesi.

### 4.2.3 Servis yazılımı katmanları

Projede Domain Driven Design (DDD) yaklaşımı uygulamaya çalışıldığından web servis oluşturulmasında aşağıdaki belirtilen Infrastructure Layer, Domain Layer ve Presentation Layer katmanları kullanılmıştır.

#### 4.2.3.1 Infrastructure Layer (Data Katmanı)

Bu bölümde Infrastructure Layer kapsamında kullanılan teknolojiler ve yazılım işlemlerinden bahsedilmiştir.

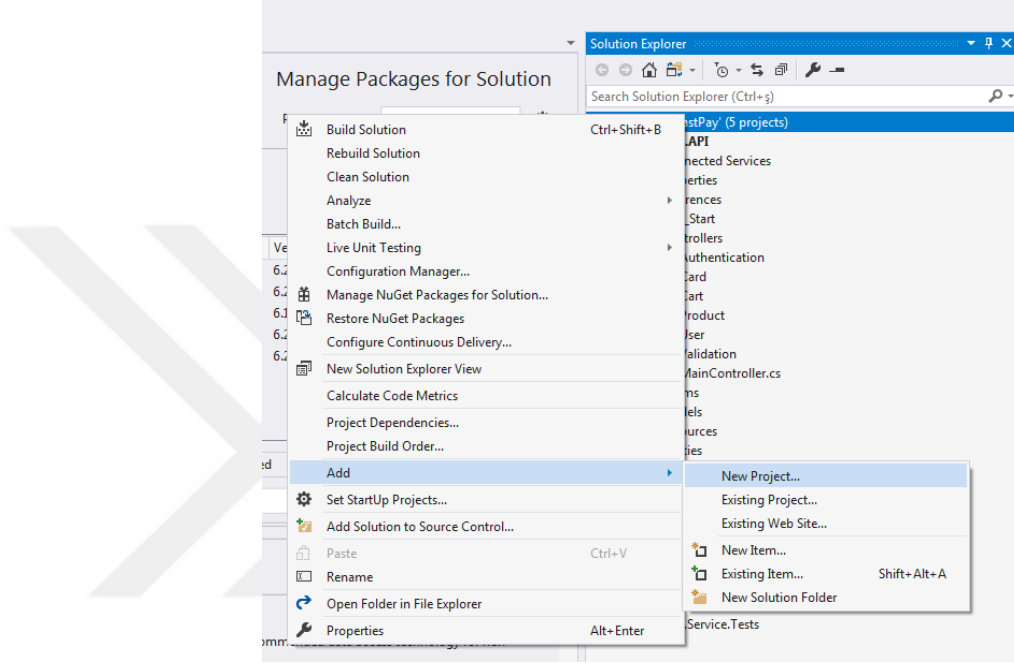
- **Microsoft MVC Database First yaklaşımı**

Microsoft'un sunduğu ASP .NET MVC teknolojisinde veritabanı işlemleri üç farklı yaklaşımla gerçekleştirilebilmektedir. Bunlar:

- ✓ Database First
- ✓ Model First
- ✓ Code First'tür.

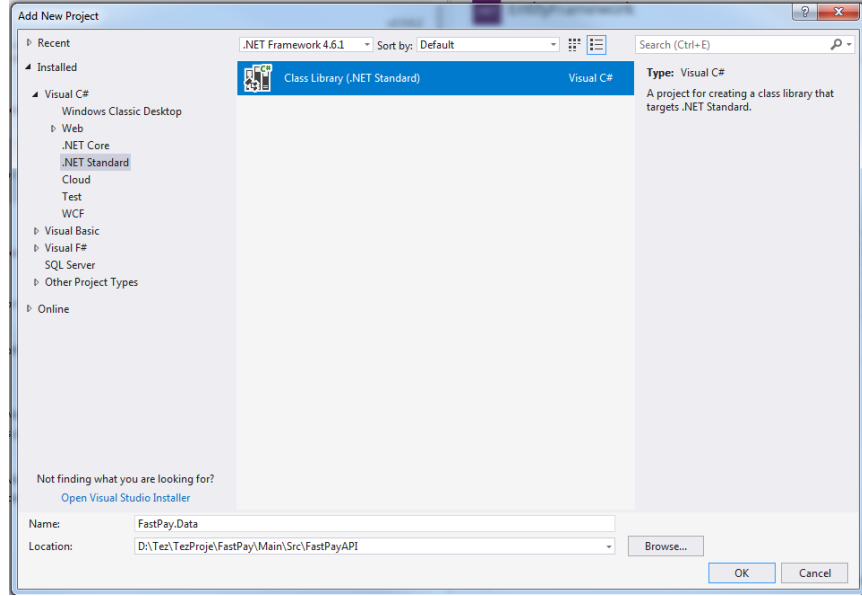
Gerçekleştirilen projede Database First yaklaşımı kullanılmıştır. Projede Database First yaklaşımı Şekil 4.17’deki gibi gerçekleştirilmiştir.

Oluşturulan projede veritabanı işlemlerini gerçekleştirebilmek için bir data katmanı oluşturulmuştur. Data katmanı projesi Microsoft Visual Studio üzerinden ana Solution’a sağ tıklanarak “Add” ve daha sonrasında da “New Project” seçilerek proje oluşturma sihirbazına erişilebilir.



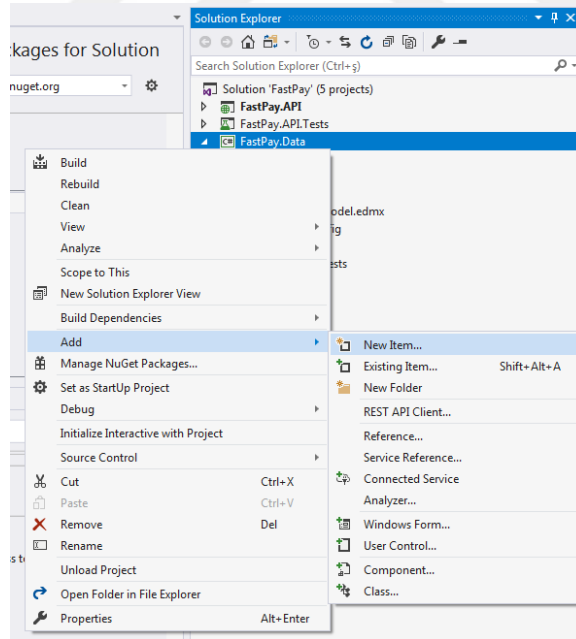
Şekil 4.17: Data katmanı oluşturma sihirbazı açılışı.

Şekil 4.17’de verilen işlemler tamamlandıktan sonra Şekil 4.18’deki gibi bir sihirbazın açılması gerekmektedir. Şekilde gösterildiği gibi sol taraftaki menüden “.NET Standard” seçilir. Ardından, orta kısımda da Class Library (.NET Standard) seçilir. Son olarak da sihirbazın aşağısında bulunan “Name” kısmında projenin adı verildikten sonra “OK” butonuna basılır ve data model oluşturabilmek için gerçekleştirilmesi planlanan data katmanı oluşturulmuş olur.



Şekil 4.18: Data katmanı oluşturma sihirbazı.

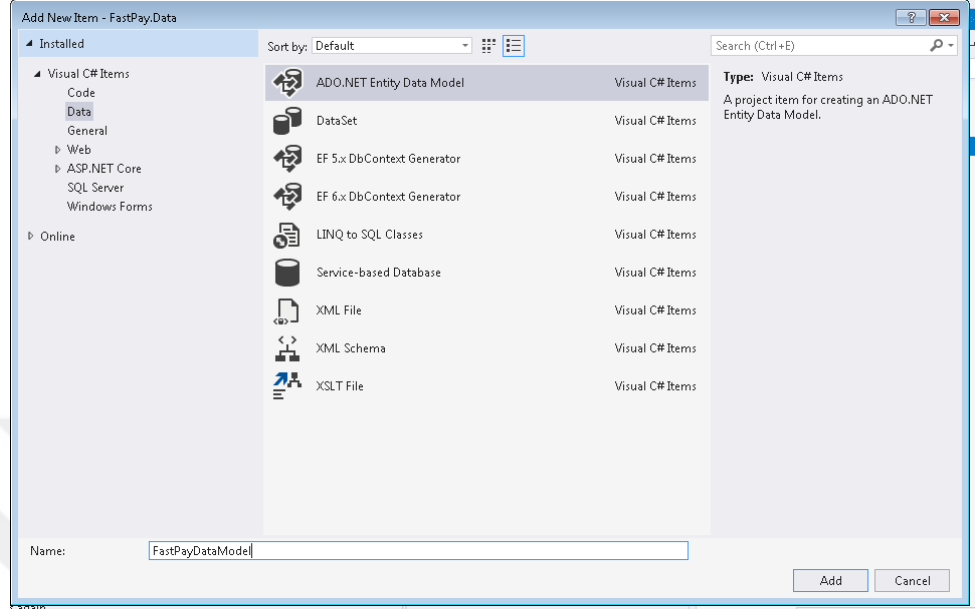
Verilen işlemler gerçekleştirildikten sonra Database First yaklaşımını gerçekleştirebilmek için oluşturulan projeye sağ tıklanarak sırasıyla “Add” ve “New Item” seçenekleri seçilir. Bu işlem Şekil 4.19’daki gibi olmalıdır.



Şekil 4.19: Database First yaklaşımı oluşturma sihirbazı açılışı.

ASP .NET MVC Database First yaklaşımını uygulayabilmek için yukarıda verilen adımlar izlendikten sonra Şekil 4.20’de görüldüğü gibi bir sihirbazın açılması gerekmektedir.

Şekil 4.20’de görüldüğü gibi sol taraftaki menüden “Data” seçeneği seçilir ve ardından “ADO.NET Entity Data Model” seçilip “OK” butonuna basılarak Data Model oluşturulmuş olur.



Şekil 4.20: Database First yaklaşımı Data Model oluşturma sihirbazı.

- **Data Model yaklaşımı**

Projede kullanılan Data Model yaklaşımı, veritabanında bulunan her bir tablo için yazılım tarafında o tabloya karşılık gelen bir model oluşturulması yaklaşımı söz konusudur. Her bir tablo için oluşturulan model içeriğinde, eşleştirildiği tablonun tüm kolon bilgileri bulunmaktadır. “Akıllı mobil ödeme sistemi”nde kullanılan tablolara ilişkin örnek bir Data Model Şekil 4.21’deki gibidir.

```

10 namespace FastPay.Data
11 {
12     using System;
13     using System.Collections.Generic;
14
15     6 references
16     public partial class Authentication
17     {
18         0 references | 0 exceptions
19         public int ID { get; set; }
20         1 reference | 0 exceptions
21         public string KullaniciID { get; set; }
22         2 references | 0 exceptions
23         public int DogrulamaTuru { get; set; }
24         2 references | 0 exceptions
25         public string Aciklama { get; set; }
26         1 reference | 0 exceptions
27         public bool Aktif { get; set; }
28         1 reference | 0 exceptions
29         public System.DateTime GuncellemeTarihi { get; set; }
30
31         0 references | 0 exceptions
32         public virtual Kod Kod { get; set; }
33         2 references | 0 exceptions
34         public virtual Kullanici Kullanici { get; set; }
35     }
36 }

```

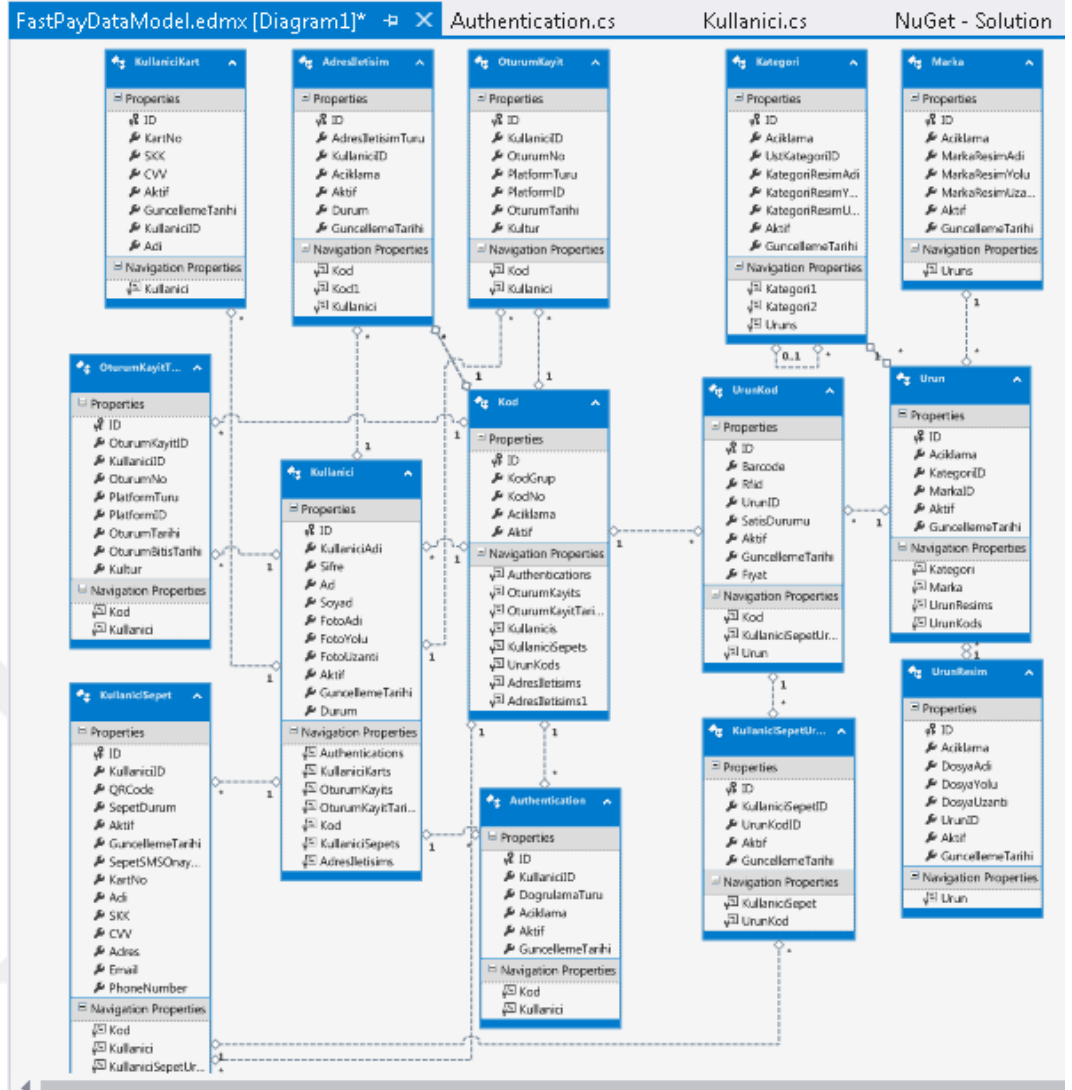
Şekil 4.21: Data Model örneği.

- **Data Model diyagramı**

Data Model yaklaşımında bahsedildiği üzere, veritabanındaki her bir tabloya ait kolonun Data Model tarafında bir property karşılığı var demektir. Buna göre Visual Studio ASP .NET MVC Database First yaklaşımın sunmuş olduğu Data Model diyagramları söz konusudur. Data Model diyagramları, data modelleri daha kolay yönetebilmek ve gerçekleştirilen analiz sonucu oluşan tablolara üst bir görünüm kazandırarak şekillendirmek amaçlanmaktadır.

Entity Data model ayrıca, tüm tablolar arasındaki “Relation” denilen tablolar arası ilişkiyi de görsel olarak sunmaktadır. Buna göre, hangi tablonun birbiri ile ilişkili olduğu gözlemlenebilmektedir. Aynı zamanda bir tablo ile diğer tablo arasında “1-1”, “1-n”, “n-1” ilişkileri de takip edilebilmektedir.

“Akıllı mobil ödeme sistemi” projesi için gerçekleştirilen Data Model diyagramı Şekil 4.22’deki gibidir.



Şekil 4.22: Projede gerçekleştirilen Entity Data Model diyagramı.

#### 4.2.3.3 Entity Framework

Entity Framework, bir yazılım projesinde Data Layer ile veritabanı arasında veri alışverişini sağlayan bir Object Relational Model (ORM) aracıdır. Entity Framework sayesinde klasik ADO .NET üzerinde Structured Query Language (SQL) sorguları yazarak verileri “DataTable” üzerinden almak yerine model tabanlı bir yaklaşım sağlar.

Entity Framework’un sağladığı model tabanlı yaklaşım sayesinde, veritabanındaki tablolar arasındaki ilişkiler de Lazy Loading özelliği ile Entity Framework modelleri üzerinde konumlandırılabilir. Bunun anlamı, bir tablodaki herhangi bir verinin modele doldurulmasının ardından yazılımcı o tabloyla ilişkili başka bir tabloda bulunan veriye erişmek istediğinde sadece

yapması gereken ilk eriştiği tablonun entity'sinde zaten önceden tanımlı olan diğer tabloya ait bağlantı sağlayan property'i kullanmasıdır.

Lazy Loading özelliği sayesinde, Entity Framework sürekli veritabanı sorgusu yapmayı ön planda engellemektedir. Aslında framework, arka planda bu işlemi yapmaktadır ancak, yazılımcı bununla uğraşmamaktadır. Bu özellik ile yazılımcının kod geliştirme süreci kısalmaktadır.

Microsoft ASP .NET MVC Web API servis methodları ile veritabanındaki verilere erişebilmeyi sağlayan, yine Microsoft'un sunmuş olduğu bir framework'tür (Dykstra T., Anderson R., 2014). Entity Framework sayesinde veritabanına bağlantı ve veriye erişim kolayca sağlanabilecektir.

#### **4.2.4 Domain Layer**

Domain Driven Design (DDD) yaklaşımına göre bir projede bulunan tüm olaylar birer domain olarak adlandırılır ve tüm bu olaylara ilişkin süreçler kendi domain içerisinde gerçekleştirilir. Bu bağlamda “akıllı mobil ödeme sistemi” projesinde gerçekleştirilen tüm süreçler ve süreç kurallarını kapsayan “Domain Rules” kısmı bu katman içerisinde gerçekleşmektedir. Yazılımın tüm iş parçacıkları, tüm iş kuralları ve karar mekanizmaları bu katmanda sağlanır. Geliştirilen bir projenin kalbi Domain Layer'dir denebilir.

Aşağıda maddeler halinde verilen konular Domain Layer katmanı içerisinde gerçekleştirilmektedir. Service olarak nitelendirilen başlıklar Domain Driven Design (DDD) yaklaşımında bahsedilen birer Domain'e karşılık gelmektedir. Her bir domainin işlemleri analizsel olarak da yazılımsal olarak da kendi içlerinde gerçekleşmektedir.

##### **4.2.4.1 DbContext İşlemleri (CRUD işlemleri)**

DbContext, Infrastructure Layer'ı kullanarak veritabanı üzerinde yapılması gereken CRUD (CREATE, READ, UPDATE, DELETE) işlemleri denen emirleri gerçekleştirir. Domain Layer'da bulunan tüm veritabanı işlemleri DbContext üzerinden gerçekleştirilmektedir.

Tüm veritabanı işlemlerinin DbContext üzerinden gerçekleştirilmesinin avantajlarından biri veritabanına erişimin tek bir kanal aracılığıyla yapılmasından

dolayı bu işlemlerin hangi kullanıcılar tarafından, ne zaman veya hangi Internet Protocol (IP) üzerinden gerçekleştirdikleri bilgisi tek bir yönetim bloğu sayesinde kontrol edilebilir özelliği sağlamaktadır.

Bunun yanı sıra, tüm veritabanı işlemlerinin yine DBContext üzerinden sağlanıyor olması, tüm CRUD (CREATE, READ, UPDATE, DELETE) işlemlerinin gerçekleştirilmesi sırasında olası herhangi bir sorunla karşılaşılması durumunda yazılımın ihtiyaç duyabileceği herhangi bir notification işlemini bu aşamada başlatılabilir olmasıdır.

#### 4.2.4.2 Authentication Service

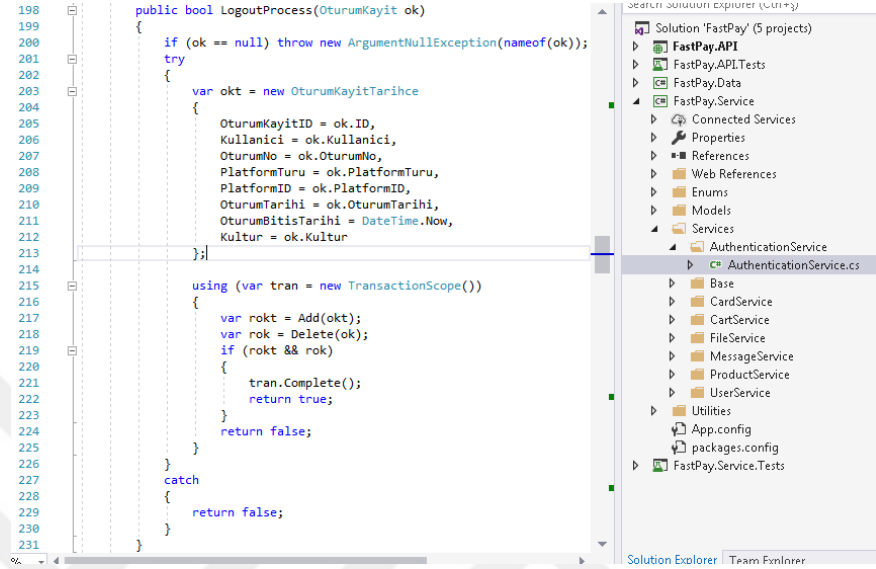
Authentication Service, gerçekleştirilen web servisini kullanacak tüm kullanıcılar için Domain Layer katmanında Authentication işlemlerini gerçekleştireceği kısımdır. Bu kısım DBContext üzerinden Entity Framework kullanarak Authentication ile ilgili tüm veritabanı tablolarına erişerek projenin ihtiyaç duyduğu tüm CRUD (CREATE, READ, UPDATE, DELETE) işlemlerini gerçekleştirmektedir.

Domain Layer içerisinde kullanılan örnek bir Authentication Service metodu Şekil 4.23'teki gibidir. “LogoutProcess” metodu, sisteme daha önceden giriş yapan kullanıcının mobil uygulama üzerinden “Logout” butonuna bastığında Domain Layer tarafında çalışan kod parçasıdır.

Metoda göre OturumKayıt türünde bir model verisini parametre olarak alan metod, kullanıcıyı logout etmeden önce kullanıcının login bilgilerini OturumKayıtTarihce adında bir Entity Data Model'e atayarak history kaydı tutmaktadır. Bu sayede sistemden logout olan kullanıcı bilgilerine veya önceden sisteme login olan kullanıcı bilgilerine erişim kolaylıkla sağlanabilir olacaktır. “TransactionScope()” kullanılmasındaki asıl amaç, kullanıcı sistemden logout edilmeden önce tarihçe kaydının history bilgileri veritabanına aşağıdaki şekilde yazılmış olsun.

Kullanıcı logout edilmeden önce sistemden kaynaklanan herhangi bir sorun oluşması durumunda servisin hizmet veremediği bir senaryo olsun. TransactionScope tam olarak bu aşamada önem arz etmektedir. Şekilden de görüleceği üzere, TransactionScope sınıfına ait “Completed()” metodu çağırılana kadar oluşan herhangi bir sorunda sistem otomatik olarak o aşamaya kadar

gerçekleştirilen düzenlemeleri, yani veri tabanı işlemlerini geri almaktadır. Bunun anlamı, birden fazla adımı olan bir süreçte oluşabilecek herhangi bir sorunda sistemin geriye sarmasını sağlayarak oluşabilecek veri kayıplarını veya olmaması gereken veri girişlerinin önüne geçmektir.



Şekil 4.23: Örnek bir Authentication Service metodu.

- **Token Based Authentication**

Token Based Authentication, bir kullanıcının sisteme bir kereye mahsus olmak üzere login olduktan sonra, o kişi için oluşturulan ve bulunması zor olan anahtar değeri denilen bir token üretilir. Bu token değeri, belirli süre ile kullanıcıya tahsis edilebilir. Kullanıcıya tahsis edilen token değeri sayesinde kullanıcı, diğer tüm işlemlerini eğer belirtilmiş bir süre var ise o süre içerisinde rahatlıkla tekrar kullanıcı adı ve şifresini girmeden gerçekleştirebilir.

Token Based Authentication'un en önemli avantajlarından biri, sürekli kullanıcı adı ve şifre bilgilerinin network üzerinde paketler halinde dolaşmasının önüne geçmesidir. Bu sayede paketleri izleyen bir kişi, sadece anlamsız bir token değeri elde edebilecektir. Elde edilen token değeri de belirli bir süre sonra expired olacağı için güvenlik anlamında önemli bir rol oynamaktadır.

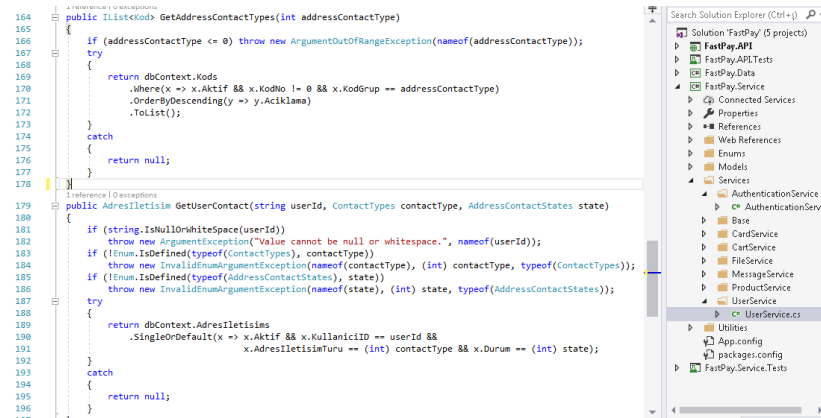
“Akıllı mobil ödeme sistemi”nde Authentication yöntemi olarak Token Based Authentication kullanılmıştır. Kullanıcı sisteme login olduktan sonra geri kalan tüm işlemlerini ona tahsis edilen token üzerinden sağlamaktadır.

#### 4.2.4.3 User Service

User Service, gerçekleştirilen web servisini kullanacak tüm kullanıcılar için Domain Layer katmanında kullanıcı işlemlerinden adres ve iletişim bilgileri işlemlerini gerçekleştireceği kısımdır. Bu kısım DbContext üzerinden Entity Framework kullanarak adres ve iletişim bilgileri ile ilgili tüm veritabanı tablolarına erişerek projenin ihtiyaç duyduğu tüm CRUD (CREATE, READ, UPDATE, DELETE) işlemlerini gerçekleştirmektedir.

Domain Layer içerisinde kullanılan örnek bir User Service metodu Şekil 24'teki gibidir. “GetAddressContactTypes” metodu, sistemde barındırılması planlanan adres ve iletişim türlerini veritabanından alarak “Kod List” türündeki Entity Model’e doldurması işlemidir. Bu metot ev adresi, iş adresi, ev telefonu, cep telefonu, iş telefonu ve e-mail bilgilerini parametrik olarak veritabanından almaktadır. Sistemin ihtiyacı dahilinde yeni bir adres iletişim türü tanımlanması istendiğinde yazılıma müdahale etmeden veritabanına eklenecek bir kayıt ile devreye sokulması mümkün olacaktır.

Aynı şekilde “GetUserContact” metodu ise, sistemde kayıtlı olan bir kullanıcının, metoda parametre olarak verilen iletişim türlerine ait verileri veritabanından alarak “AdresIletisim” türündeki Entity Model’e doldurması işlemidir. Bu metot işlemi Domain Layer’da gerçekleşmektedir. Bu metottan elde edilen veriler, daha sonra Presentation Layer’a gönderilmektedir. Presentation Layer da bu bilgiler arasından kullanıcıya iletilmesi gereken bilgileri filtreleyerek, kullandığı Web API teknoloji sayesinde filtrelenen bilgileri, Hyper Text Transfer Protocol (HTTP) üzerinden Javascript Object Notation (JSON) objelerine dönüştürerek göndermektedir.



Şekil 4.24: Örnek bir User Service metodu

#### 4.2.4.4 File Service

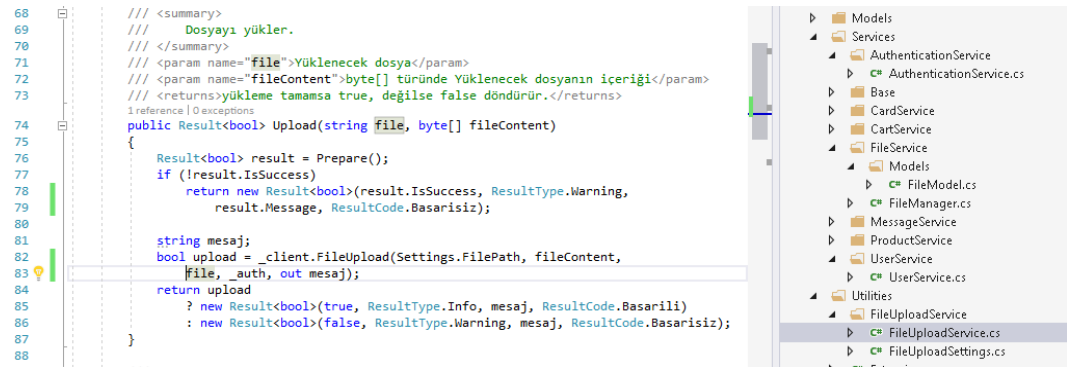
File Service, adından da anlaşılacağı üzere dosya işlemlerinin gerçekleştirildiği domaindir. Bu domain servisinin amacı herhangi bir dosyanın veritabanında tutulmasından ziyade sunucu üzerinde bir diskte tutulmasını sağlayan kısımdır.

Application Layer tarafından gelen bir dosya Domain Layer'a ulaştığında, ilgili dosyanın yükleneceği sunucu bilgileri ve sunucu üzerindeki disk yolu ayarları bu katman aracılığıyla veritabanı etkileşimleriyle sağlanmaktadır. Application Layer'dan gelen dosya, ona verilecek bir Unique Identifier sayesinde, dosyanın fiziksel olarak veritabanında tutulmasından ziyade diskte network üzerinden yüklenmesini gerçekleştirir.

Veritabanında, sadece ilgili dosyaya tahsis edilen Unique Identifier değeri tutulur. Daha sonraki bir süreçte, ihtiyaç duyulması halinde ilgili dosyanın sunucudan alınıp Application Layer'a gönderilebilmesi için, yine Unique Identifier kullanılır.

Application Layer da, mobil uygulamanın ilgili dosyayı görüntüleyebilmesi için dosyayı, gerekli formata dönüştürerek HTTP üzerinden Javascript Object Notation (JSON) formatına dönüştürerek mobil uygulamaya gönderir.

Şekil 4.25'deki kod parçasında Application Layer tarafından Domain Layer'a gelen bir dosyanın nasıl diske yüklendiği yazılımsal olarak gösterilmektedir.



Şekil 4.25: File Service dosya yükleme işlemi.

#### 4.2.4.5 Message Service

Message Service, Domain Layer’da gerçekleştirilen herhangi bir olay sırasında tetiklenmesi gereken bir durum esnasında farklı türlerde notification gönderimini sağlayan domain servsidir.

Bu servis, SMS Service ve Mail Service işlemlerini tek elden gerçekleştirebilmeyi sağlamaktadır. Notification işlemi sırasında yazılımsal olarak Sadece SMS Service kullanıldığı varsayılın. Belirli bir zaman sonra Mail Service işlemi de dahil olması gerekliliği ortaya çıktığında, yazılım sürecini tamamen yenilemek yerine Message Service katmanı oluşturularak mesaj türlerinden bağımsız olan bir yapı oluşturulmuş olur.

SMS veya Mail Service’lerinden herhangi birini kullanan diğer katmanlar, aslında sadece bu katmanı kullanarak, arka planda ne gönderdiklerini bilmeden işlemleri gerçekleştirirler. Bu şekilde bağımsızlık sağlanır ve kısmi olarak Dependency Injection gerçekleştirilmiş olur.

Message Service kullanılırken öncelikle “CreateMessageService” metodu işleme alınır ki arka tarafta hangi servisi kullanması gerektiğine karar verebilsin. Bu işlem tamamlandıktan sonra “Send” metodu sayesinde sms veya e-mail gönderimi kolaylıkla sağlanmış olur. Bu işlem ile ilgili Send metodu içeriği Şekil 4.26’da gösterilmiştir.

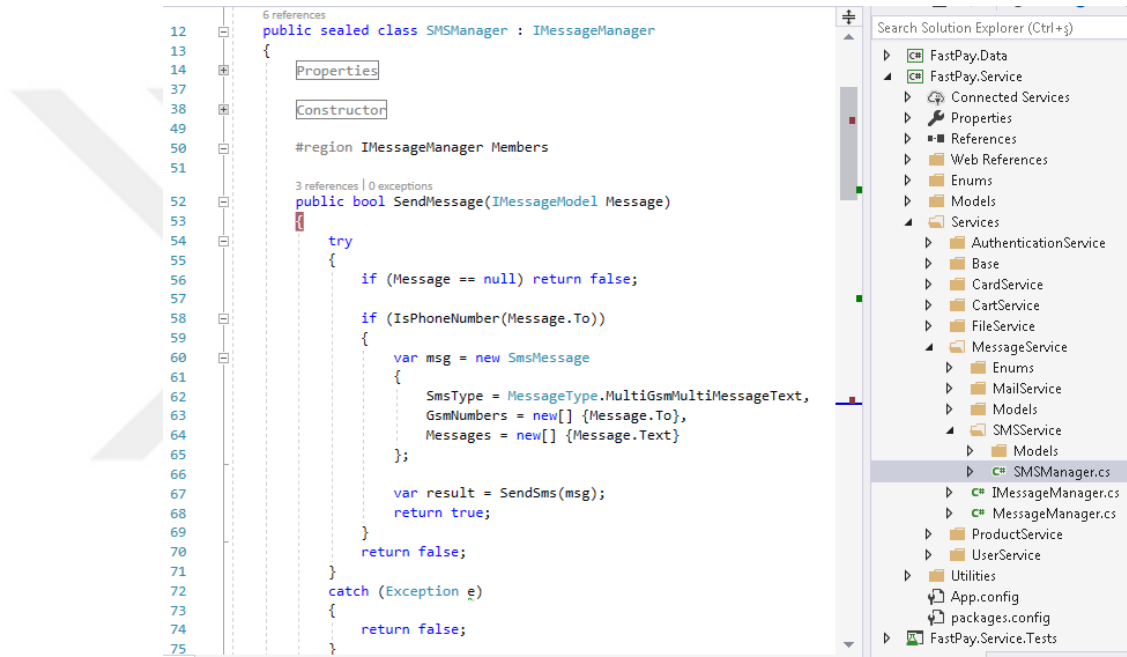


Şekil 4.26: Message Service tür dağıtımı ve gönderimi işlemleri.

- **SMS Message Service**

Daha önceden bahsedildiği gibi MessageManager sınıfı tarafından kullanılan SMS Message Service, IMessageManager interface'inden türetilmiştir. Bunun nedeni MessageManager'i kullanan diğer tüm mesaj gönderim seçeneklerini tek bir yordam ile yönetebilmektedir.

Şekil 4.27'de görüldüğü gibi IMessageManager interface'inden türeyen MessageManager sınıfı için, miras aldığı interface'in bir metodu olan "SendMessage()" işleminin SMS gönderimi olarak gerçekleştirilmektedir.

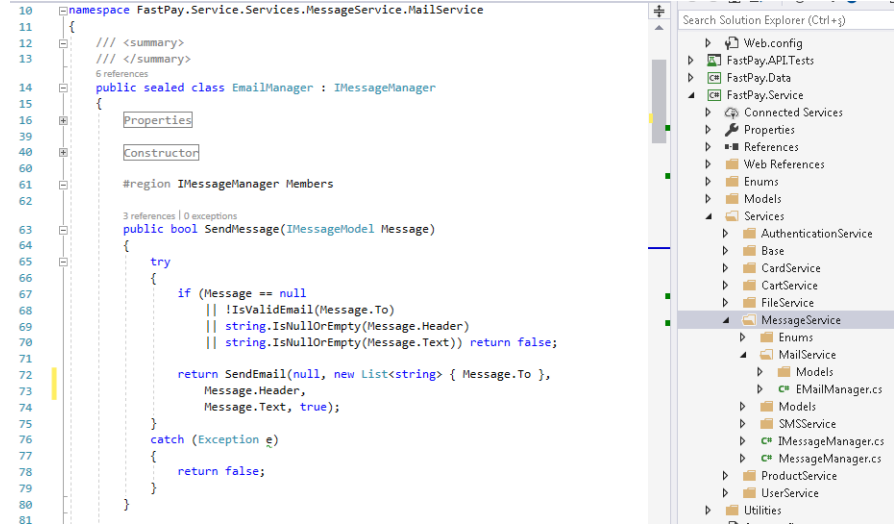


Şekil 4.27: SMS Service sınıfı ve SendMessage metodu.

- **Mail Message Service**

Daha önceden bahsedildiği gibi MessageManager sınıfı tarafından kullanılan Mail Message Service, IMessageManager interface'inden türetilmiştir. Bunun nedeni MessageManager'i kullanan diğer tüm mesaj gönderim seçeneklerini tek bir yordam ile yönetebilmektedir.

Şekil 4.28'de görüldüğü gibi IMessageManager interface'inden türeyen MailManager sınıfı için, miras aldığı interface'in bir metodu olan "SendMessage()" işleminin e-mail gönderimi olarak gerçekleştirilmektedir.



Şekil 4.28: Mail Service sınıfı ve SendMessages metodu.

#### 4.2.4.6 Security Service

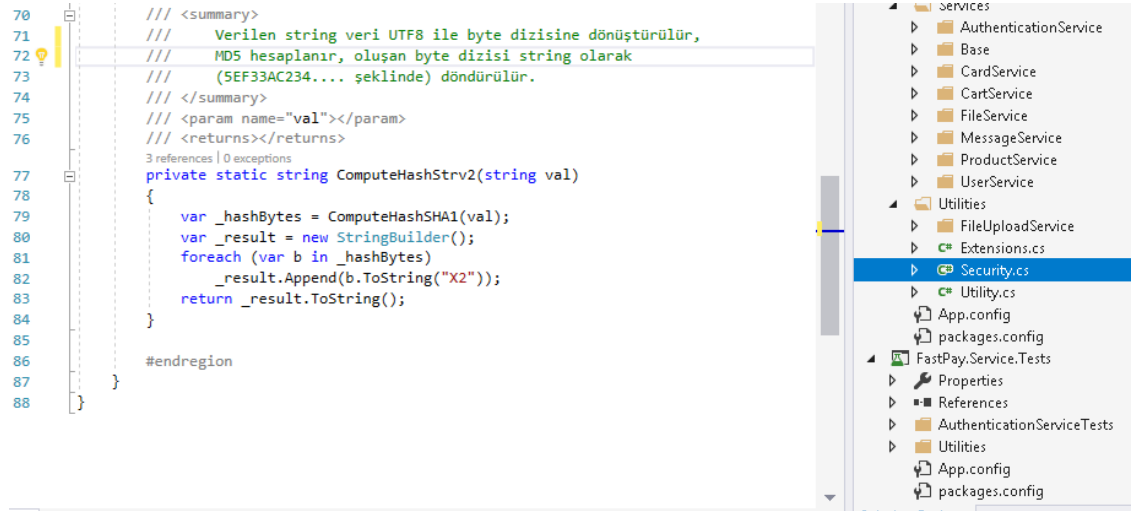
Security Service, genel olarak sisteme ilk kez kayıt olan kullanıcıların şifre bilgilerini şifreleme işlemini gerçekleştirmektedir. Ayrıca, Security Service, sadece “akıllı mobil ödeme sistemi” yöneticilerine verilebilen süper şifre işlemlerini de gerçekleştirmektedir.

- **Message Digest 5 (MD5) Encryption**

Message Digest 5 (MD5) algoritması, kendisine şifrelemesi için verilen değer kendine has denilen “finger print” özelliğinin oluşturulması anlamına gelmektedir. Temelinde “hash” metotları vardır. Veritabanı yönetimlerinde sıklıkla kullanılan bir yöntemdir. Message Digest 5 (MD5), Profesör Rivest tarafından geliştirilmiştir.

Message Digest 5 (MD5) algoritması tek yönlü olarak çalışmaktadır. Bunun anlamı şifreli metin oluşturulduktan sonra elde edilen metin kullanılarak asıl metin elde edilemez.

“Akıllı mobil ödeme sistemi” projesinde şifre bilgilerini veritabanında encrypt’leyerek saklanmasını sağlar. Projede kullanılan Message Digest 5 (MD5) yönteminin bir kısmı Şekil 4.29’daki gibidir.



Şekil 4.29: Kullanıcı şifrelerinde kullanılan Encription işlemi.

- **Unicode Transformation Format (UTF8) Coding**

Unicode Transformation Format (UTF8), Unicode dönüşüm biçimidir. 8 bitlik bir işlem dizisinden oluşur. Rob Pike ve Ken Thompson geliştirmiştir. ASCII tablosunda bulunan tüm karakterler, Unicode Transformation Format (UTF8) tarafında 8 bit ile ifade edilebilmektedir.

“Akıllı mobil ödeme sistemi” için gerçekleştirilen bir şifre bilgisini encrypt’leyebilmek için kullanılan UTF-8 Coding bloğu Şekil 4.30’daki gibidir.

```

32  /// <summary>
33  /// </summary>
34  /// <param name="val"></param>
35  /// <returns></returns>
36  3 references | 0 exceptions
37  public static byte[] ComputeHash(string val)
38  {
39      var _md5 = new MD5CryptoServiceProvider();
40      return _md5.ComputeHash(Encoding.UTF8.GetBytes(val));
41  }

```

Şekil 4.30: Kullanıcı şifrelerinde kullanılan UTF-8 Coding işlemi.

#### 4.2.4.7 Card Service

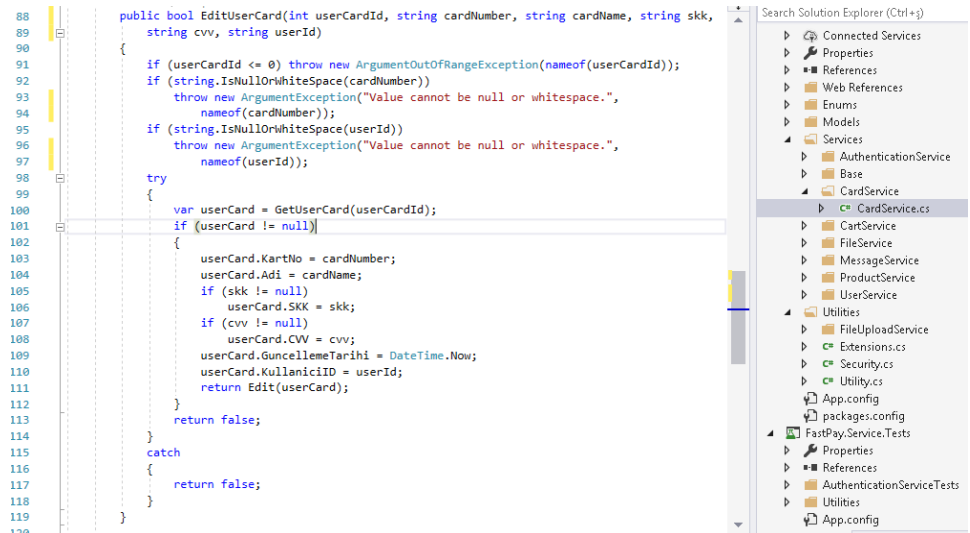
Card Service, gerçekleştirilen web servisini kullanacak tüm kullanıcılar için Domain Layer katmanında kullanıcıların kart işlemlerini gerçekleştireceği kısımdır. Bu kısım DbContext üzerinden Entity Framework kullanarak

kullanıcının kart ile ilgili tüm veritabanı tablolarına erişerek projenin ihtiyaç duyduğu tüm CRUD (CREATE, READ, UPDATE, DELETE) işlemlerini gerçekleştirmektedir.

Domain Layer içerisinde kullanılan örnek bir Card Service metodu Şekil 4.31'deki gibidir. “EditUserCard” metodu, sistemde kayıtlı olan bir kullanıcının kart bilgileri üzerinde değişiklik yapmaya imkan sunar. Metoda parametre olarak verilen “userCardId” ile bir kullanıcının verilen ID değerine ait kart bilgisi güncellenebilmektedir.

Öncelikle parametre olarak verilen “userCardId” ile veritabanından o karta ait tüm bilgiler veritabanından alınarak Entity Data Model’e doldurulur. Metoda gelen diğer parametreler sayesinde değiştirilen tüm alanlar mevcut Entity Data Model üzerinden güncellenir. Son olarak da güncellenmiş Entity Data Model nesnesi veri tabanına geri kaydedilir. Bu sayede kullanıcının mobil uygulama vasıtasıyla düzenlemiş olduğu kart bilgileri başarıyla kaydedilmiş olur.

Bu metod işlemi Domain Layer’da gerçekleşmektedir. Bu metottan elde edilen, sağlıklı güncelleme için “true” veya herhangi bir olumsuz sonuçlanan güncelleme durumu için “false” verisi, daha sonra Presentation Layer’a gönderilmektedir. Presentation Layer da bu bilgiyi kullandığı Web API teknoloji sayesinde, HTTP üzerinden Javascript Object Notation (JSON) objesine dönüştürerek mobil uygulamaya göndermektedir.



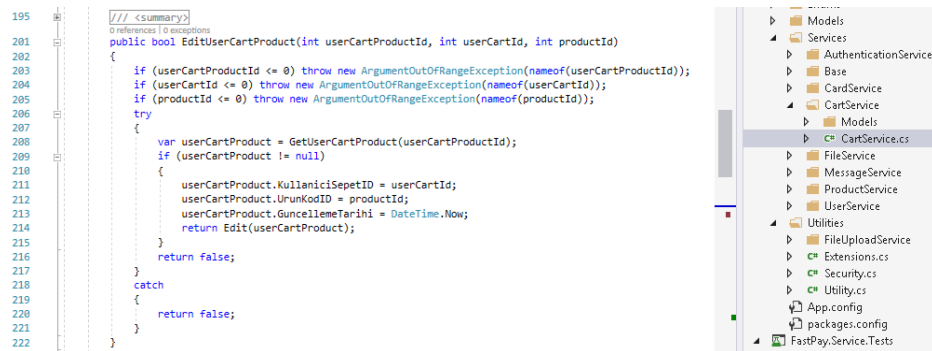
Şekil 4.31: Örnek bir Card Service metodu.

#### 4.2.4.8 Cart Service

Cart Service, gerçekleştirilen web servisini kullanacak tüm kullanıcılar için Domain Layer katmanında kullanıcıların sepet işlemlerini gerçekleştireceği kısımdır. Bu kısım DbContext üzerinden Entity Framework sepet bilgileri ve sepet içeriğinde bulunan tüm ürün bilgileri ile ilgili tüm veritabanı tablolarına erişerek projenin ihtiyaç duyduğu tüm CRUD (CREATE, READ, UPDATE, DELETE) işlemlerini gerçekleştirmektedir.

Domain Layer içerisinde kullanılan örnek bir Cart Service metodu Şekil 4.32'deki gibidir. “EditUserCartProduct” metodu, sistemde kayıtlı olan bir kullanıcının sepet içerisinde bulunan ürün bilgileri üzerinde değişiklik yapmaya imkan sunar. Metoda parametre olarak verilen “userCartProductId” ile bir kullanıcının sepetinde bulunan verilen ID değerine ait bir ürün bilgisi güncellenebilmektedir. Öncelikle parametre olarak verilen “userCartProductId” ile veritabanından o ürüne ait tüm bilgiler veritabanından alınarak Entity Data Model’e doldurulur. Metoda gelen diğer parametreler sayesinde değiştirilen tüm alanlar mevcut Entity Data Model üzerinden güncellenir. Son olarak da güncellenmiş Entity Data Model nesnesi veri tabanına geri kaydedilir. Bu sayede kullanıcının mobil uygulama vasıtasıyla düzenlemiş olduğu sepetindeki ürünleri başarıyla değiştirmiş olur.

Bu metot işlemi Domain Layer’da gerçekleşmektedir. Bu metottan elde edilen, sağlıklı güncelleme için “true” veya herhangi bir olumsuz sonuçlanan güncelleme durumu için “false” verisi, daha sonra Presentation Layer’a gönderilmektedir. Presentation Layer da bu bilgiyi kullandığı Web API teknoloji sayesinde, Hyper Text Transfer Protocol (HTTP) üzerinden Javascript Object Notation (JSON) objesine dönüştürerek mobil uygulamaya göndermektedir.



Şekil 4.32: Örnek bir Cart Service metodu.

#### 4.2.4.9 Product Service

Product Service, gerçekleştirilen web servisini kullanacak tüm kullanıcılar için Domain Layer katmanında kullanıcıların kategori, marka ve ürün işlemlerini gerçekleştireceği kısımdır. Bu kısım DbContext üzerinden Entity Framework kullanarak kategori, marka ve ürün bilgileri ile ilgili tüm veritabanı tablolarına erişerek projenin ihtiyaç duyduğu tüm CRUD (CREATE, READ, UPDATE, DELETE) işlemlerini gerçekleştirmektedir.

Domain Layer içerisinde kullanılan örnek bir Product Service metodu Şekil 4.33'teki gibidir. “SaveProductImage” metodu, sistemde kayıtlı olan bir ürünün resim bilgilerini veritabanına eklemeye imkan sunar. Metoda parametre olarak verilen “ProductImageDto” türündeki Data Transfer Object (DTO) ile bir ürüne ait resim bilgilerinden dosyanın adı, dosyanın uzantısı (jpeg, jpg, png, bitmap vs.) ve dosyanın sunucuya yüklendiği ve o dosya için tahsis edilen Unique Identifier bilgisi parametre olarak gelmektedir. Data Transfer Object (DTO) tarafından gelen tüm veriler yeni bir Entity Data Model oluşturularak içeriğine set edilir. Son olarak da Entity Data Model nesnesi veri tabanına kaydedilir. Bu sayede kullanıcının mobil uygulama vasıtasıyla eklemiş olduğu ürüne ait resim bilgileri başarıyla kaydedilmiş olur.

Bu metot işlemi Domain Layer’da gerçekleşmektedir. Bu metottan elde edilen, sağlıklı güncelleme için “true” veya herhangi bir olumsuz sonuçlanan güncelleme durumu için “false” verisi, daha sonra Presentation Layer’a gönderilmektedir. Presentation Layer da bu bilgiyi kullandığı Web API teknoloji sayesinde, Hyper Text Transfer Protocol (HTTP) üzerinden Javascript Object Notation (JSON) objesine dönüştürerek mobil uygulamaya göndermektedir.



Şekil 4.33: Örnek bir Product Service metodu.

### 4.2.5 Presentation Layer (Service Layer)

Domain Driven Design (DDD) yaklaşımına uygun olacak şekilde Domain Layer'in üst tarafına Presentation Layer konmuştur. Presentation Layer, mobil uygulamadan gelen HTTPRequest işlemlerini harmanlayarak HTTPResponse şeklinde geri dönmeyi sağlamaktadır.

Mobil uygulamadan gelen istekler Presentation Layer'a gelir ve bu bilgiler bir takım doğrulamalardan sonra Domain Layer'a geçer. Domain Layer'da ise isteğine amacına uygun işlemler yürütülür. Bu işlemler tamamlandıktan sonra Domain Layer cevabı Application Layer'a geri döner. Son olarak da Domain Layer'dan Application Layer'a gelen veri Javascript Object Notation (JSON) formatına dönüştürülerek Microsoft Web API teknolojisi sayesinde HTTPResponse üzerinden mobil uygulamaya gönderilir.

#### 4.2.5.1 HTTPContext sınıfı

Microsoft'un kullanıcılarına sunduğu ASP .NET, içerisinde barındırdığı HTTPContext sınıfı sayesinde Request, Response, Server ve Session gibi önemli özelliklere erişebilmeyi sağlamaktadır. Geliştirilen "akıllı mobil ödeme sistemi", mobil uygulaması ile sunucu web servis yazılımı arasında haberleşebilmesi için kullandığı yöntemdir.

#### 4.2.5.2 HTTPRequestMessage sınıfı

Mobil uygulama veya Hyper Text Transfer Protocol (HTTP) üzerinden request işleminde bulunabilen herhangi bir sistemden gelen requestlere, göndermiş olduğu parametrelere ve ayrıca bu işlem ile yapılması gereken tüm özelliklere ASP .NET MVC Web API teknolojisi kullanılarak, Controller metotları içerisinde HTTPRequestMessage sınıfı sayesinde erişilebilmektedir. Yapılan bir request aslında, MVC Web API tarafında, bir Controller metodunu tetiklemektedir. Bu işlem parametrelili veya parametresiz olabilmektedir.

"Akıllı mobil ödeme sistemi" için genelde parametrik istekler söz konusu olmuştur. Mobil uygulama tarafında kullanıcılara hizmet vermesi amaçlandığından dolayı bir kullanıcı diğer bir kullanıcının verilerine müdahale edememesi için her controller metodu Credential işlemine tabi tutulmaktadır. Her

kullanıcı, login olduktan sonra ona tahsis edilen token sayesinde diğer metotları kullanabilmektedir.

Doğru token'a sahip olmayan bir requeste hiçbir şekilde cevap verilmemektedir. HTTPRequestMessage sınıfı, doğru token bilgisine sahip olan istekleri, hedef olan controller metoduna taşır.

“Akıllı mobil ödeme sistemi” için gerçekleştirilen ve HTTPRequestMessage sınıfını kullanan bir controller metodu Şekil 4.34’te verilmiştir.



Şekil 4.34: Örnek bir HTTPRequestMessage Controller metodu.

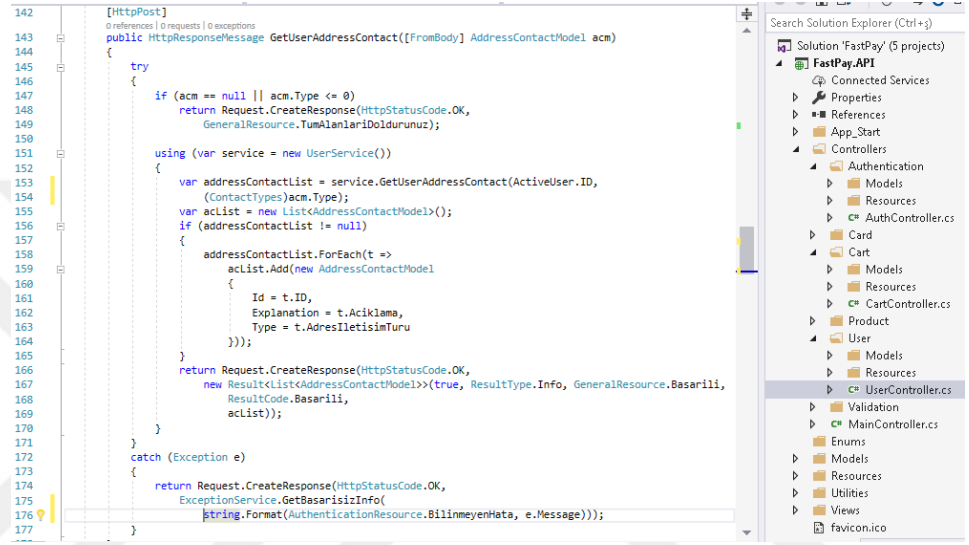
#### 4.2.5.3 HTTPResponseMessage sınıfı

HTTPRequestMessage sayesinde, daha önceden bahsedildiği gibi bir controller metoduna Hyper Text Transfer Protocol (HTTP) üzerinden yapılan bir request işleminde, Business Logic denilen ve sadece çağırılan metoda özgü işlemleri yerine getiren iş kuralları işlemleri tamamlandıktan sonra, isteği yapan sisteme bir şekilde cevap dönülmesi gerekmektedir. Bu işlemin gerçekleştirilebilmesi için HTTPResponseMessage sınıfı kullanılmaktadır.

Gelen request için, HTTPResponseMessage sınıfı türünde bir response oluşturabilmek için, HTTPRequestMessage sınıfı ile gelen isteğin özellikleri kullanılarak bir response üretilebilir. Bu işlemi gerçekleştirebilmek için HTTPRequestMessage sınıfı özelliklerinden biri olan “CreateResponse()” metodu kullanılabilir.

“Akıllı mobil ödeme sistemi” için geliştirilen web servis içerisinde, her bir metoda gelen request işlemi geriye HttpResponseMessage türünde bir nesne geri dönmektedir. Bu sınıfı üretebilmek için de HttpRequestMessage sınıfı özelliği olan “CreateResponse()” metodu kullanılmıştır.

“Akıllı mobil ödeme sistemi” için gerçekleştirilen web serviste, Şekil 4.35’de bir response işleminin nasıl üretildiğine dair örnek bir gösterim söz konusudur.



Şekil 4.35: Örnek bir HttpResponseMessage ile Response üretimi.

#### 4.2.5.4 Url Routing

Url Routing ASP .NET MVC ‘nin önemli özelliklerinden biridir. MVC sayfaları çağırılırken, sayfaya sahip olan Controller’in devreye alınabilmesi için sıklıkla kullanılan Url Routing yöntemi sayesinde, ActionResult oluşturmak mümkündür.

URL Routing kullanmak çok basittir. Oluşturmak istediğiniz Url Configuration işlemini, metodun üst tarafında tanımlamak yeterlidir. MVC Controller metodu üst tarafında köşeli parantezler içerisinde tanımlanmaktadır.

“Akıllı mobil ödeme sistemi” için gerekli işlemlerden bir tanesi de sisteme kayıt olan kullanıcının vermiş olduğu e-mail bilgisinin doğrulanması işlemidir. “Akıllı mobil ödeme sistemi” için kullanıcı mobil uygulamadan sisteme kayıt olurken kendisinden e-mail bilgisi istenmektedir. E-mail bilgisinin istenme

sebeplerinden biri de kullanıcı ürün satın alıp ödeme işlemini gerçekleştirildikten sonra kendisine satın aldığı ürünlere dair e-fatura gönderimini sağlayabilmek.

Kullanıcıya e-faturasını sağlıklı bir şekilde gönderebilmek için, kullanıcı sisteme kayıt olurken e-mail bilgisini girmesi gerekmektedir. Kullanıcının girmiş olduğu e-mail adresine login olduktan hemen sonra bir mail gönderilir. Hesabını doğrulaması için gönderilen e-mail içerisinde bir doğrulama url'si vardır. Kullanıcı, e-mailini açtıktan sonra doğrulama url'sine tıklayarak “akıllı mobil ödeme sistemi”ndeki hesabını doğrulayabilmektedir.

Kullanıcıya gönderilen e-mail içerisinde oluşturulan bu url'ye tıklandığında Url Routing devreye girmektedir ve Şekil 4.36'daki Controller metodu tetiklenmektedir. Controller metodu üst tarafında tanımlanan “ValidateUserWithEmail” ifadesi e-mail içerisindeki linkte de bulunmaktadır. Controller metot eşleşimi ve routing işlemi bu şekilde sağlanmaktadır.

Aynı zamanda her kullanıcı için Globally Unique Identifier (GUID) denen bir token de her bir e-mail doğrulamasına özgün kullanıcı için tahsis edilmektedir. Kullanımda görüldüğü gibi “verificationCode” ifadesi de verilen controller meduna parametre olarak gönderilmektedir. Bu sayede hangi kullanıcı e-mail adresini doğrulamak istediği bilinebilmektedir.



Şekil 4.36: Örnek bir Url Routing kullanımı.

“Akıllı mobil ödeme sistemi” Url Routing işlemi kullanıldıktan sonra kullanıcının e-mailini doğrulayabilmesi için birtakım işlemler gerçekleştirilmektedir. Bu işlemler gerçekleştirildikten sonra kullanıcı, linke tıkladıktan sonra açılacak sayfada olumlu ya da olumsuz birşeyler görmek

isteyecektir. E-mail doğrulama işlemi için gerekli olan durum bilgisi kullanıcıya Microsoft ASP .NET MVC’de bulunan View sayesinde gösterilmiştir.

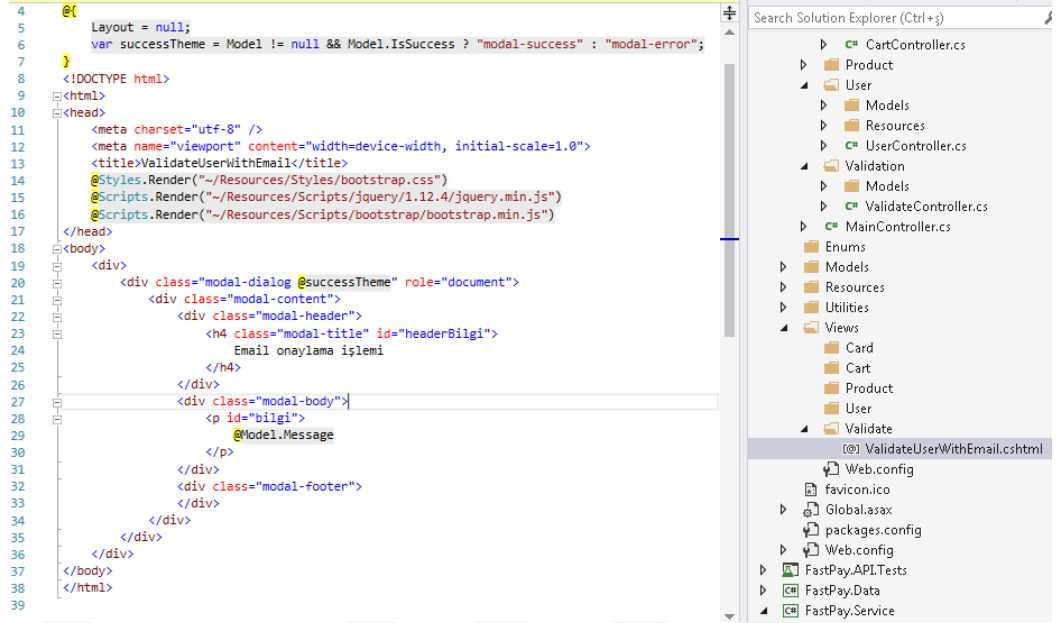
Kullanıcı, linke tıkladıktan sonra oluşturulacak olan View sayfasına aşağıdaki verilen Controller metodu içerisinde durum bilgisi gönderilmesi gerekmektedir. Bu işlem için MVC’de View’a controller tarafından veril gönderilebilmesi için sıkça kullanılan yöntemlerden biri olan ViewModel kullanılmıştır. Şekil 4.37’de görüldüğü gibi e-mail doğrulama işlemi için hazırlanmış olan “ValidateViewModel” kullanılmıştır. “ValidaViewModel” sınıfı içerisinde e-mail doğrulama işleminin sonuç bilgisini “IsSuccess” ve sonuç mesajını da “Message” şeklinde saklayabilmektedir.



Şekil 4.37: Controller üzerinden ViewModel kullanılarak View’a veri gönderme.

“ValidateViewModel” ile veri View tarafında gönderilmiştir. Controller tarafından gönderilen “ValidateViewModel”, View tarafında “Model” ifadesi ile karşılanmaktadır. ASP .NET MVC’nin Razor render teknolojisi sayesinde “@” anahtar sözcüğü ile HTML tag’leri arasına dinamik olarak veri konabilmektedir.

Şekil 4.38’deki gibi oluşturulan sonuç View sayfası, içerisinde Bootstrap, JQuery ve Razor teknolojilerini barındırmaktadır. Bu sayede kullanıcı e-mailine gelen linke tıkladığında rahatlıkla sonuç sayfasını görebilmektedir. Razor teknolojisi sayesinde ise Contoller tarafında oluşturulan “ValidateViewModel” içeriği, kullanıcıya gösterilebilmektedir.



Şekil 4.38: Controller üzerinden ViewModel ile gelen verinin View’da kullanılması.

#### 4.2.5.5 Exception Service

Domain Driven Design (DDD) yaklaşımının bir katmanı olan Presentation Layer’da kullanılan ASP .NET MVC Web API servis teknolojisi içerisinde kullanılmak üzere bir Exception Service sınıfı oluşturulmuştur. Bu servisin amacı dışarıdan gelen istekleri yönetebilmektedir.

Diğer sistemlerden yapılan isteklere web servisin cevap verebilmesi için daha önceden bahsedildiği gibi devrede olan bir token ile istekte bulunmalarıdır. Kullanımda olan bir token ile istekte bulunan bir sisteme cevap verilmektedir.

Ancak, güncel olmayan veya kullanım dışı bir token ile veya hiç token olmadan gerçekleştirilen bir request’e asla cevap verilmemektedir. Bu şekilde gerçekleştirilen request işlemlerinde, gelen request’in ilgili controller metoduna ulaşamaması için “CreateException” adında bir metot yazılmıştır. Bu metot, doğru bir token’a sahip olmayan isteği herhangi bir kullanımı veya karşılığı olmayan bir tarafa yönlendirmekte ve bu sayede, bu şekilde gelen istekler bertaraf etmektedir.

Bir diğer Exception Service metodu ise “GetBasarisizInfo”dur. Bu metodun amacı controller metotları içerisinde herhangi bir olumsuzluk oluşması

durumunda, request'te bulunan sisteme bir cevap dönmektir. Hata mesajı da içermektedir. Metot içeriği Şekil 4.39'daki gibidir.



Şekil 4.39: Exception Service için kullanılan metotlar.

#### 4.2.5.6 JavaScript Object Notation (JSON)

“Akıllı mobil ödeme sistemi” projesinde veri taşıma teknolojisi olarak JavaScript Object Notation (JSON) tercih edilmiştir. Bu tercihin nedeni JavaScript Object Notation (JSON) ile Extensible Markup Language (XML) aynı büyüklükte bir veri ile tasarlandığında JavaScript Object Notation (JSON)’un daha az boyutlu olmasıdır (Droettboom, M., 2015).

JavaScript Object Notation (JSON) ile harmanlanan bir veri, Extensible Markup Language (XML)’a göre daha az boyutta olmaktadır. Bundan dolayı tercih sebebi olmaktadır. JavaScript Object Notation (JSON)’a Şekil 4.40’daki veri seti örnek olarak gösterilebilir.

```

1 {
2     "UserName" : "oguz",
3     "Password" : "oguz1234",
4     "Name" : "Oğuz",
5     "SurName" : "ASLAN",
6     "Email" : "oguz.aslan@outlook.com",
7     "Telephone" : "905436539053"
8 }

```

Şekil 4.40 : Projede kullanılan bir JSON örneği.

Microsoft SQL veritabanındaki verilere, Microsoft ASP .NET MVC Web API servis methodları çağırılarak erişildiğinde elde edilen verilerin mobil uygulamaya aktarılabilmesi için kullanılması tercih edilen bir veri formatıdır.



## 5. AKILLI MOBİL ÖDEME SİSTEMİ

### 5.1 Servisin Sunucuya Kurulumu

Geliştirilen bir web servisin dış dünyaya hizmet verebilmesi için, dış dünya ile bağlantısı olan ve static IP adresine sahip bir ortama kurulması gerekmektedir. Bu ortam farklı işletim sistemlerine sahip bir makine olabilir. Bu ortamın da web servisini algılayabilmesi için bir takım yazılımsal araçların da varlığı gerekmektedir.

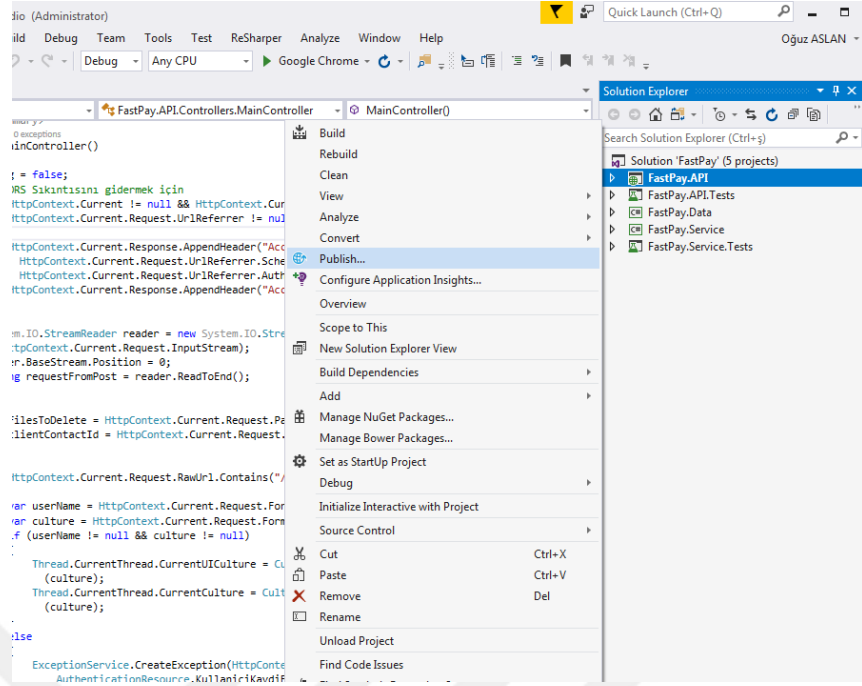
Node JS, Internet Information Services (IIS) ve Kestrel, bir web servisin herhangi bir sunucuya kurulabilmesi için gerekli olan araçlardan birkaçıdır. “Akıllı mobil ödeme sistemi” için tasarlanan mimaride aşağıdaki durumlar söz konusudur.

#### 5.1.1 Servisin kurulumu hazır hale getirilmesi

“Akıllı mobil ödeme sistemi” için geliştirilen web servisin sunucuya kurulabilmesi için öncelikle bir takım işlemlerin yapılması gerekmektedir. Web servis kurulumdan önce kurulumu hazır hale getirilmesi yani sunucunun geliştirilen web servisi tanıyabilmesi önemlidir.

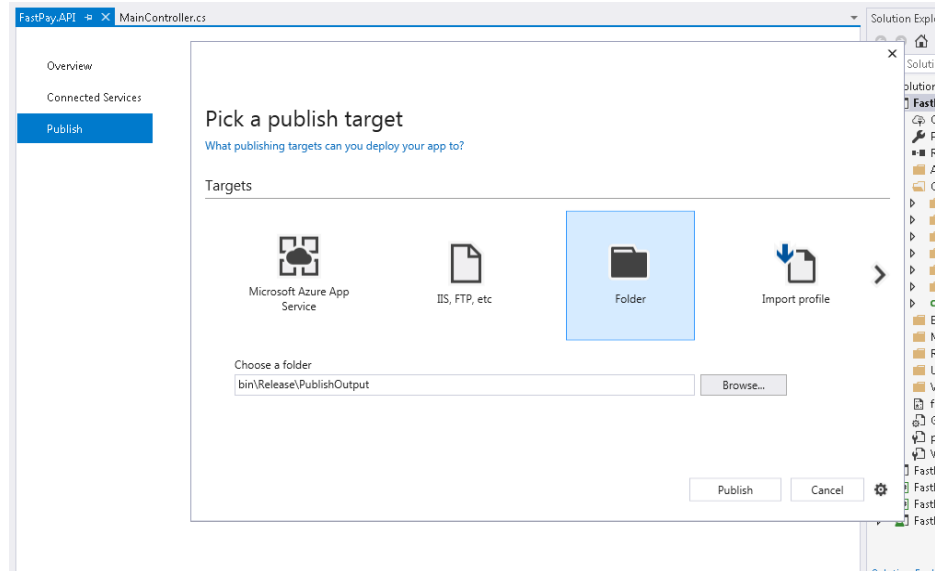
Web servisin sunucuda çalışabilmesi için Visual Studio tarafından “Publish” edilmesi gerekmektedir. Bilindiği üzere geliştirilen web servisi çok katmanlı bir şekilde tasarlanmıştır. Web servis, Domain Driven Design (DDD) yaklaşımının bir parçası olan Presentation Layer katmanında bulunmaktadır. Dolayısıyla, Presentation Layer katmanında bulunan web servisin “Publish” edilmesi gerekmektedir. Web servisin “Publish” işlemine tabii tutulduğunda, kendisine bağlı olan diğer kütüphanelerin tümü de bu işleme dahil edilmiş olacaktır.

Projede web servisin bulunduğu projeye sağ tıklayarak Şekil 5.1’de görüldüğü gibi çıkan menüden “Publish”e tıklanması gerekmektedir.



Şekil 5.1 : Web servisin Publish edilmesi adımı.

Web servisin bulunduğu projeye sağ tıklayıp çıkan menüden “Publish” dindikten sonra “Publish Sihirbazı” açılır. Şekil 5.2’de görüldüğü gibi “Target” seçeneği olarak “Folder” seçilir. Ardından “Publish” işleminden sonra oluşacak dosyaların yerleştirileceği yer seçimi yapılır. Tüm bu işlemler tamamlandıktan sonra “Publish” butonuna basılarak işlem gerçekleştirilmiş olur.



Şekil 5.2 : Web servisin Publish edilmesi.

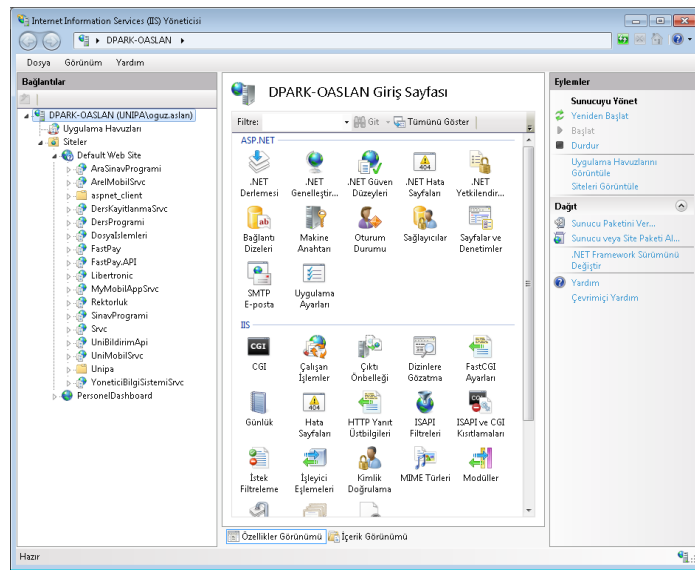
### 5.1.2 Kullanılan işletim sistemi

Web servisin üzerinde çalışacağı sistemin bir işletim sisteminin olması gerekmektedir. Bu bağlamda Microsoft Windows Server 2012 R2 işletim sistemi tercih edilmiştir. Web servis alt tarafta .NET Framework’ün 4.6.1 sürümünü kullandığı için Windows 2008 sürümlü işletim sistemleri tercih edilmemiştir. Konu ile ilgili yapılan araştırmalar sonucu Windows Server işletim sistemleri içerisinde 2012 R2 sürümü, 2008 sürümlerine göre .NET Framework 4.6.1’i daha stabil çalıştırmaktadır. Bu nedenle tercih edilen işletim sistemi Microsoft Windows Server 2012 R2’dir.

### 5.1.3 Internet Information Services (IIS)

Windows Server 2012 R2 işletim sistemi tercih edildikten sonra, bu işletim sisteminde web servisin çalışabilmesi için web sunucusu olarak bir aracın sunucuda çalışıyor olması gerekmektedir. Bu bağlamda, Microsoft’un kendi işletim sistemlerinde sunduğu bir hizmet olan Internet Information Services (IIS) kullanılmaktadır.

Internet Information Services (IIS), bir web sayfasını veya servisini çalıştırabilmek için kullanılan bir web sunucudur. Internet Information Services (IIS), Microsoft’un kullanıcılarına sunduğu bir yazılımdır. Internet Information Services (IIS) arayüzü Şekil 5.3’teki gibidir. Bu uygulamayı çalıştırabilmek için “Başlat → Çalıştır” dedikten sonra açılan pencereye “inetmgr” yazmak yeterlidir.



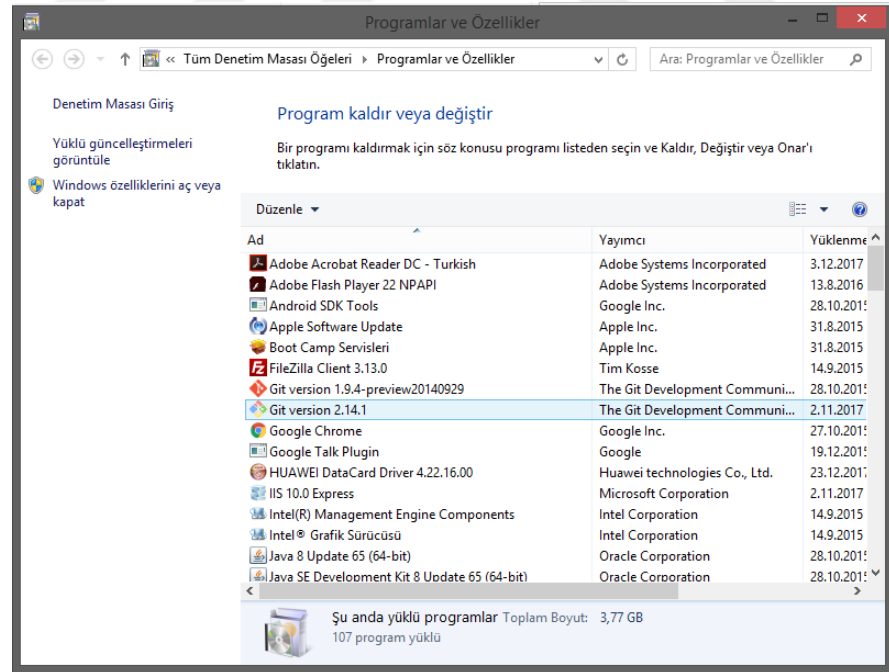
Şekil 5.3: Internet Information Services (IIS) kullanıcı arayüzü.

Internet Information Services (IIS), bir web sayfasının veya web servisin hangi .NET Framework üzerinde çalışacağı ve buna ilişkin düzenlemelerin yapılması, ilgili web servis veya sayfa için sunucu port atamaları işlemlerinin yapılması, SMTP e-posta ayarlarının sağlanması, birden fazla servisin veya sayfanın aynı port üzerinden hizmet verebilir olması düzenlemeleri, MIME type düzenlemeleri ile hangi dosya tiplerinin web sunucu tarafından kabul edilebilir olduğunun belirlenmesi ayarları ve .NET Framework yapılandırma ayarlarının düzenlenmesi gibi birçok hizmeti bir arada bulundurmaktadır.

#### 5.1.4 Servisin sunucuya kurulması

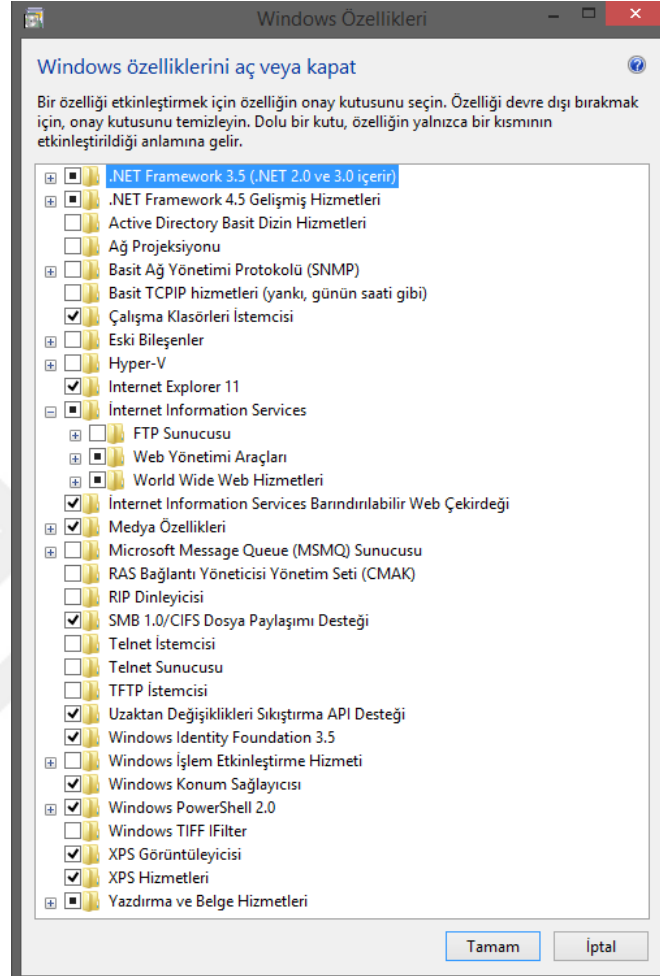
Bir web servisin sunucuya kurulması için öncelikle sunucu üzerinde Internet Information Services (IIS)’in kurulu olması gerekmektedir.

Eğer sunucuya ilk defa web servis kuruluyorsa Microsoft’un sunduğu Internet Information Services (IIS) yazılımını devreye sokmak gerekmektedir. Bunun için öncelikle “Başlat → Programlar ve Özellikler” arayüzü açılır. Açılan pencerenin sol tarafında bulunan “Windows özelliklerini aç veya kapat” seçeneği Şekil 5.4’te görüldüğü gibi seçilir.



Şekil 5.4: Programlar ve özellikler arayüzü.

Açılan bu pencerenin sol tarafında bulunan “Windows özelliklerini aç veya kapat” seçeneği seçildikten sonra, Windows özelliklerini aktif veya pasif kontrollerinin yapılabileceği yeni bir pencere açılmaktadır. Açılan pencere Şekil 5.5’teki gibidir.

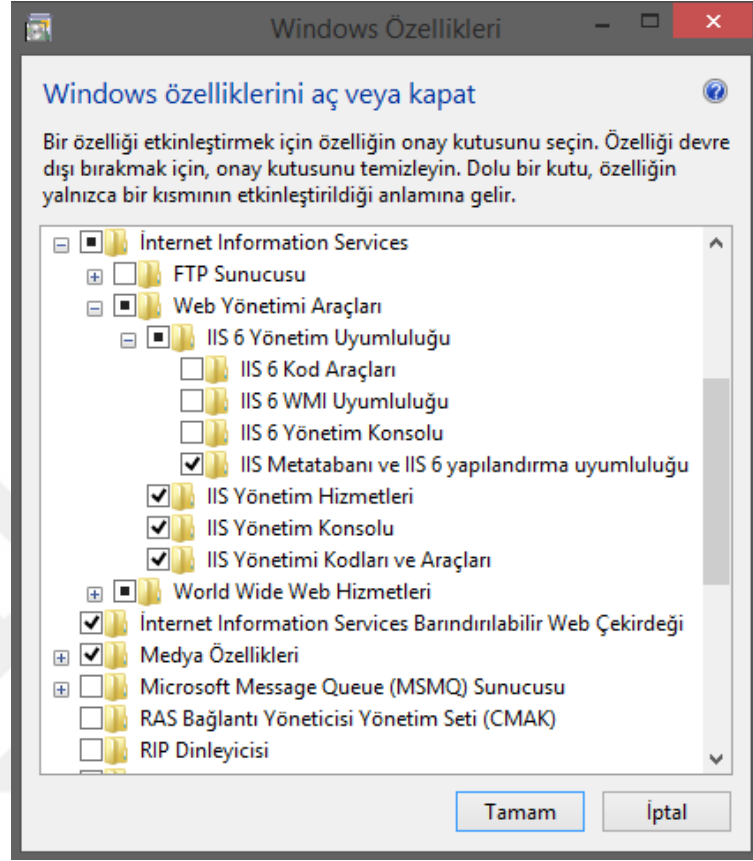


Şekil 5.5: Windows özellikleri aç/kapat penceresi.

Windows özelliklerini aç/kapat penceresi içerisinde görüldüğü gibi Internet Information Services (IIS) özelliği görülmektedir. Web serverin sağlıklı bir şekilde çalışabilmesi için bir takım özel ayarların yapılması gerekmektedir. Bu bağlamda Internet Information Services (IIS) seçeneği altında birtakım ayarların aktif edilmesi gerekmektedir.

Internet Information Services (IIS) altında bulunan “Web Yönetim Araçları” seçeneği altındaki “IIS Yönetim Hizmetleri”, “IIS Yönetim Konsolu” ve “IIS Yönetim Kodları ve Araçları” alanlarının seçili olması gerekmektedir. Ayrıca “Web Yönetim Araçları” altında “IIS 6 Yönetim Uyumluluğu” seçeneği vardır.

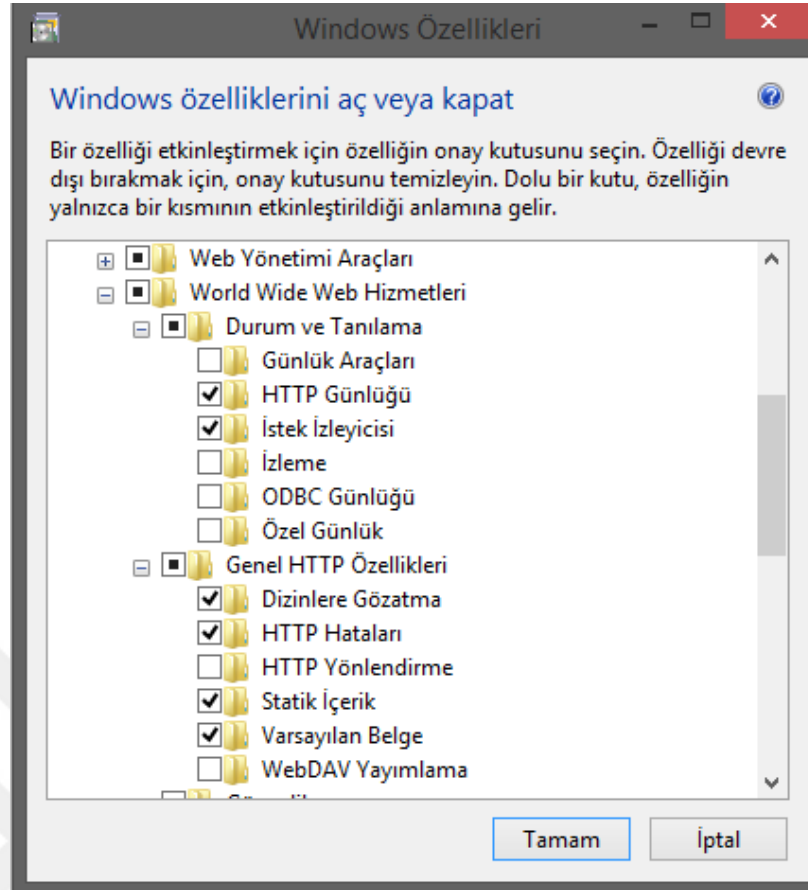
Bu seçeneğin altında bulunan “IIS Metatabanı ve IIS 6 yapılandırma uyumluluğu” seçeneğinin de seçili olması gerekmektedir. Bu işlemler Şekil 5.6’da gösterilmiştir.



Şekil 5.6: IIS Web yönetimi araçları ayarları.

Bu ayarlar tamamlandıktan sonra yine, Internet Information Services (IIS) altında bulunan ve Şekil 5.7’de de görüldüğü gibi “World Wide Web Hizmetleri” seçeneği altında “Durum ve Tanılama” seçeneği vardır. “Durum ve Tanılama” seçeneği altında bulunan “HTTP Günlüğü” ve “İstek İzleyicisi” seçeneklerinin seçili olması gerekmektedir.

Internet Information Services (IIS) altında bulunan ve aşağıdaki şekilde de görüldüğü gibi “World Wide Web Hizmetleri” seçeneği altında “Genel HTTP Özellikleri” seçeneği vardır. “Genel HTTP Özellikleri” seçeneği altında bulunan “Dizinlere Gözetleme”, “HTTP Hataları”, “Statik İçerik”, “Varsayılan Belge” seçeneklerinin de seçili olması gerekmektedir.



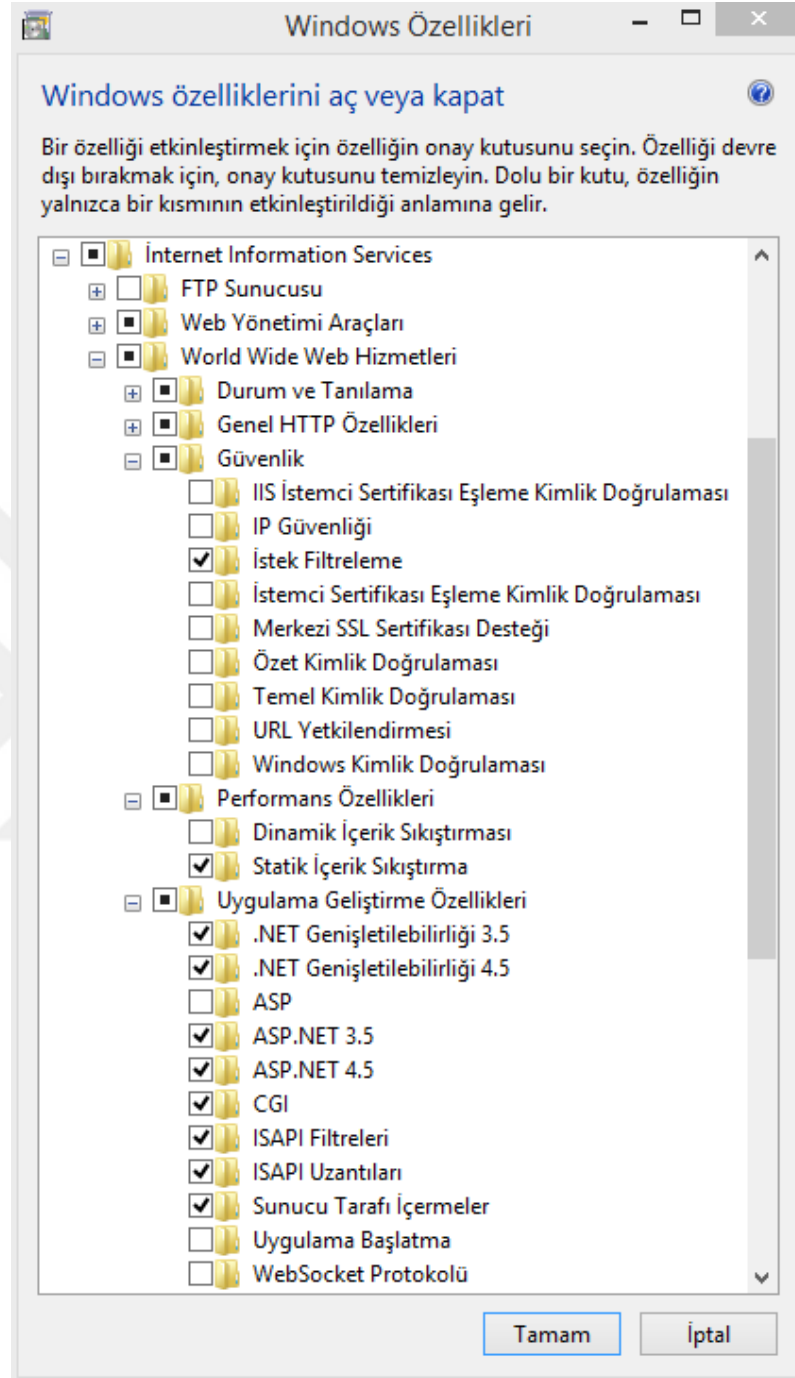
Şekil 5.7: IIS durum ve tanılama ve genel HTTP özellikleri ayarları.

Şekil 5.7’deki verilen ayarlar tamamlandıktan sonra yine, Internet Information Services (IIS) altında bulunan ve aşağıdaki şekilde de görüldüğü gibi “World Wide Web Hizmetleri” seçeneği altında “Güvenlik” seçeneği vardır. “Güvenlik” seçeneği altında bulunan “İstek Filtreleme” seçeneğinin seçili olması gerekmektedir.

Internet Information Services (IIS) altında bulunan ve Şekil 5.8’de de görüldüğü gibi “World Wide Web Hizmetleri” seçeneği altında “Performans Özellikleri” seçeneği vardır. “Performans Özellikleri” seçeneği altında bulunan “Statik İçerik Sıkıştırma” seçeneğinin seçili olması gerekmektedir.

Son olarak da, Internet Information Services (IIS) altında bulunan ve Şekil 4.48’de de görüldüğü gibi “World Wide Web Hizmetleri” seçeneği altında “Uygulama Geliştirme Özellikleri” seçeneği vardır. “Uygulama Geliştirme Özellikleri” seçeneği altında bulunan “.NET Genişletilebilirliği 3.5”, “.NET Genişletilebilirliği 4.5”, “ASP .NET 3.5”, “ASP .NET 4.5”, “CGI”, “ISAPI

Filtreleri”, “ISAPI Uzantıları”, “Sunucu Tarafı İçermeler” seçeneklerinin de seçili olması gerekmektedir.



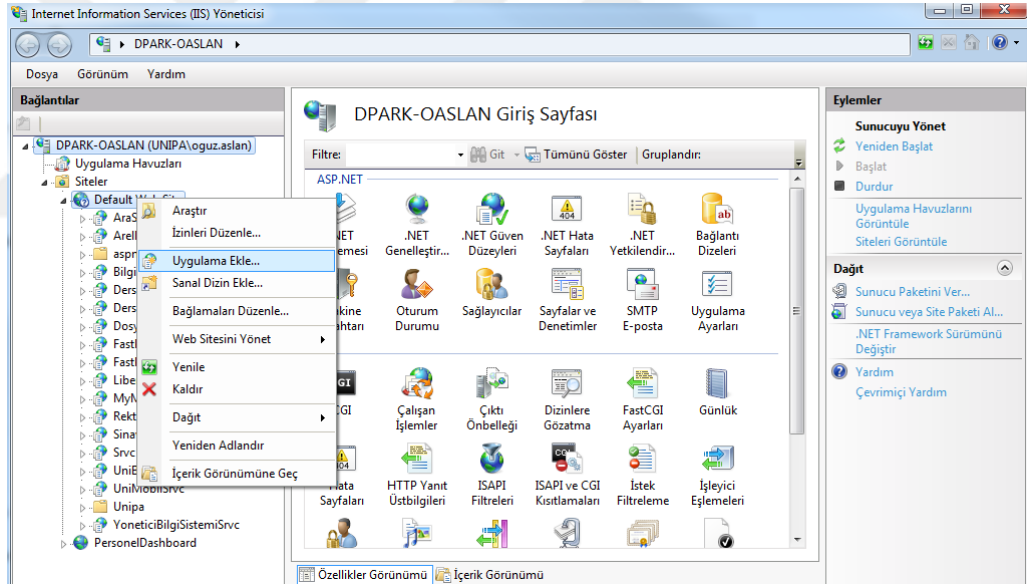
Şekil 5.8: Güvenlik, performans ve uygulama geliştirme özellikleri ayarları.

Şekil 5.8’de verilen Internet Information Services (IIS) ayarları sağlandıktan sonra, Internet Information Services (IIS) üzerine web servis kurulumu yapılabilir demektir. Windows Server işletim sistemi, masaüstünde bulunan “Başlat →

Çalıştır” adımları izlendikten sonra açılan pencereye “inetmgr” yazıldığında Internet Information Services (IIS) arayüzü açılmaktadır.

Bir web servisin kurulma hazır hale getirilmesi adımlarının nasıl olması gerektiği ve bir web servisin nasıl “publish” yapıldığından daha önceden bahsedilmişti. Internet Information Services (IIS) arayüzü açıldıktan sonra, Microsoft Visual Studio IDE’si kullanılarak “publish” edilen dosyalar, sunucu üzerinde uygun görülen bir dizine kopyalanır.

Bu işlemler gerçekleştirildikten sonra Şekil 5.9’da görüldüğü gibi Internet Information Services (IIS) üzerinde sol taraftaki menüde bulunan “Default Web Site” üzerine sağ tıklanarak “Uygulama Ekle” seçeneği seçilir ve “Uygulama Ekle” sihirbazı başlatılır. “Default Web Site”, Internet Information Services (IIS) ‘in ilk kurulumundan gelen bir özellik olarak varsayılanda 80 portu üzerinden hizmet vermektedir.

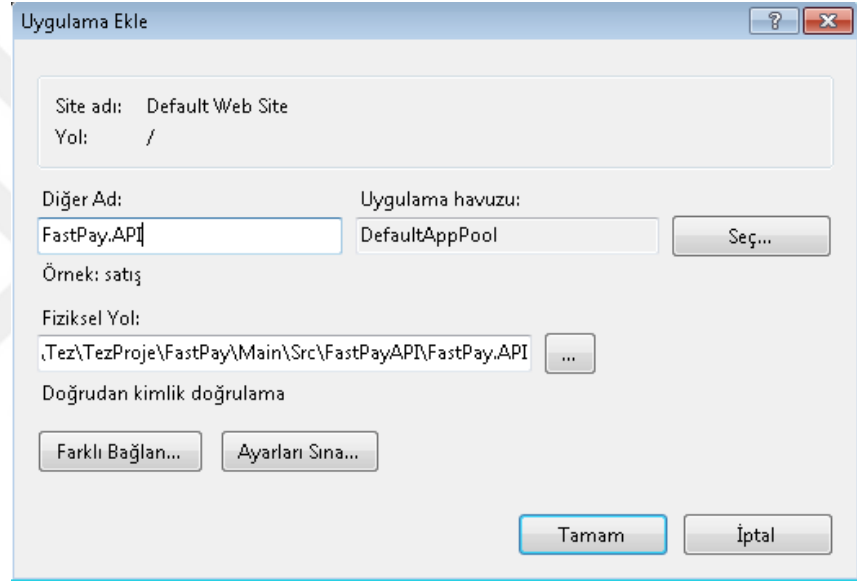


Şekil 5.9: IIS’te uygulama ekle sihirbazı açılması.

“Uygulama ekle” sihirbazı açıldıktan sonra, Diğer adı kısmına web servis için uygun görülen bir isim verilir. Bu isim daha sonra web servisi çağırarak olan diğer sistemlere tanımlanması gereken url bilgisi içerisinde olacaktır. İsim tanımlandıktan sonra, ilgili web servisin .NET Framework versiyonuna göre uygun bir uygulama havuzu seçilir. Geliştirilen “akıllı mobil ödeme sistemi” web servisi .NET Framework 4.6.1 üzerinde geliştirildiği için uygulama havuzu .NET 4.5 olacak şekilde seçilmiştir.

Internet Information Services (IIS) uygulama havuzları istenildiği kadar tanımlanabilir ve her web servis için özel bir uygulama havuzu da tanımlanabilir. Bu projede ihtiyaç duyulan uygulama havuzunun, .NET 4.5 Framework çalıştırabilir olması kontrolü olduğu için bu şekilde gerçekleştirilmiştir. “DefaultAppPool” olarak seçilen uygulama havuzu varsayılan olarak .NET 4.5 konfigürasyona uygun olarak ayarlanmıştır.

Son olarak da kuruluma hazır hale getirilmiş dosyalar, daha önceden kopyalandığı dizine ait yol bilgisi Şekil 5.10’de görüldüğü gibi “Fiziksel Yol” kısmında ayarlanır ve “Tamam” butonuna basılır. Artık web servis, diğer sistemler tarafından kullanılabilir durumdadır.



Şekil 5.10: IIS uygulama ekle sihirbazı.

İstendiği takdirde, aynı menüde yer alan “Siteler” kısmına sağ tıklanarak “Web Sitesi Ekle” seçeneği seçilerek farklı bir port üzerinden yeni bir site tanımlanabilir. Yeni bir site tanımlandıktan sonra, o site üzerine sağ tıklanır ve “Uygulama Ekle” seçeneği seçilir. Aynı adımlar sağlandıktan sonra web servis, artık belirtilen site üzerindeki tanımlanan port üzerinden hizmet vermeye başlar.

## 5.2 Mobil Uygulama Yazılım Geliştirme Yönetimi

Bu bölümde, istemci tarafında geliştirilen mobil uygulama yazılımının tüm özelliklerinden, kullanılan tüm teknolojilerden ve metodolojilerden, yazılım geliştirme süreçlerinden ve proje kapsamında gerçekleştirilen “akıllı mobil ödeme

sistemi” için gerekli olan tüm mobil uygulama yöntemlerinden konu başlıkları şeklinde bahsedilmiştir.

### 5.2.1 Hybrid mobil uygulama nedir?

Bilindiği üzere, farklı işletim sistemlerine sahip mobil cihazlar kendi işletim sistemlerini meydana getiren yazılım teknolojileri üzerinde çalışabilmektedir. Bu bağlamda, farklı işletim sistemlerinde çalışan mobil uygulamalar, çalıştığı işletim sistemine özgü olan yazılım dili ile geliştirilmesi gerekmektedir. Buna, native mobil uygulama geliştirme denir.

Native mobil uygulama geliştirme teknolojisi ile kastedilmek istenen, mobil uygulamada çalışan işletim sisteminin kullandığı programlama dili ile aynı teknolojiyi kullanarak mobil uygulama geliştirmektir. Örneğin, Android işletim sistemine sahip bir mobil cihaz içerisinde, mobil uygulama çalıştırabilmek için Java ile yazılmış olması gerekmektedir. Benzer şekilde, IOS işletim sistemine sahip bir mobil cihaz içerisinde de Objective-C ile yazılmış bir mobil uygulama çalışabilmektedir.

Hybrid mobil uygulama, mobil cihazın işletim sisteminden bağımsız bir şekilde tek bir kod ile geliştirilmiş bir mobil uygulamanın farklı işletim sistemlerinde çalışabilir olmasıdır. Hybrid mobil uygulama geliştirme teknolojisi sayesinde, farklı işletim sistemlerine sahip mobil cihazlarda, o işletim sistemi dilinde bir yazılım geliştirilmemektedir. Geliştirilen tek bir mobil uygulama, farklı işletim sistemlerinde çalışabilir durumdadır. Android, IOS veya Windows Phone işletim sistemine sahip cihazlarda, tek bir mobil uygulama geliştirilmesiyle, tüm platformlarda çalışabilir bir mobil uygulama sunulmuş olmaktadır (Manuel, P., vd, 2012).

Hybrid mobil uygulama, native mobil uygulama geliştirmenin aksine zaman ve maddi açıdan önemli avantajlara sahiptir. Geliştirilmesi planlanan bir mobil uygulamanın native mi yoksa hybrid mi geliştirilmesi daha uygun olmalı tartışması söz konusu olmaktadır. Burada önemli olan geliştirilecek mobil uygulamadan beklentinin nasıl olduğudur.

Native olarak geliştirilmiş bir mobil uygulama kendi işletim dili ile programlandığı için üzerinde çalıştığı cihazın donanım özelliklerini daha performanslı kullanabilmektedir. Hybrid bir mobil uygulama, birtakım

teknolojilerle yine üzerinde çalıştığı cihazın donanım özelliklerini kullanabilmektedir ancak, native olarak geliştirilen uygulamaya göre daha az performanslı olmaktadır. Bu tam anlamıyla kişinin mobil uygulamadan beklentisi ile ilgilidir. Bu performans farkını, mobil uygulamayı kullanan kişi fark etmemektedir veya çok az fark etmektedir.

Bahsedildiği üzere, mobil uygulama geliştirme teknolojileri native ve hybrid olmak üzere ikiye ayrılmaktadır. Burada, tasarlanan sistemin mobil uygulama donanımına olan bağımlılığına göre teknoloji tercihi yapılabilmektedir. Eğer tasarlanan mobil uygulama, cep telefonun kamera, titreşim, gyrosensor, geolocation, mikrofon, kulaklık v.b. gibi donanım özelliklerini çok fazla kullanmıyorsa hybrid mobil uygulama teknolojileri tercih edilebilir.

Hybrid mobil uygulama teknolojileri gelende HTML5, CSS3 ve Javascript tabanlı teknolojilerden oluşmaktadır. Hybrid bir mobil uygulamanın cep telefonunun donanım özelliklerine erişebilmesinde kullanılan bir takım yöntemler mevcuttur. Bu teknolojilere Phonegap/Cordova, Xamarin, Titanium ve Corona örnek olarak gösterilebilir. HTML5, CSS3 ve Javascript tabanlı mobil uygulama geliştirme teknolojilerine/framework'lerine örnek olarak IONIC Framework, Angular JS, Framework7, Kendo UI, JQuery, AJAX ve Bootstrap örnek verilebilir (Redda, Y., 2012).

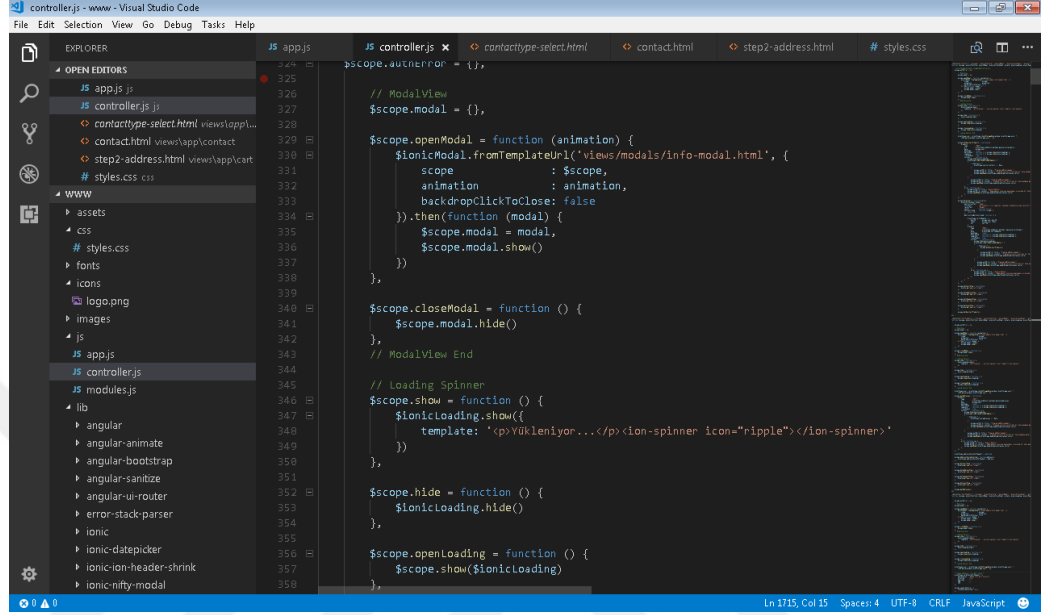
### 5.2.2 Visual Studio Code

Microsoft Visual Studio Code, Microsoft tarafından geliştirilmiştir. Tümüleşik geliştirme ortamı olarak kullanılmaktadır. Bu tür yazılım geliştirme ortamları Integrated Development Environment (IDE) olarak adlandırılır ve Microsoft Visual Studio Code da bunlardan biridir.

Geliştiricilerin daha kolay yazılım ürünü ortaya koyabilmeleri için, geliştirme sürecini organize edebilen ve bir çok ek özellikler barındıran entegre bir yazılım geliştirme aracıdır.

Microsoft Visual Studio Code sayesinde, geliştiricilere sağlanan bu kolaylıklar değerlendirildiğinde yazılım geliştirme sürecinin çok daha kısa sürede sonuçlandığı geliştiriciler tarafından belirtilmektedir.

Microsoft, kendi geliştirmiş olduğu teknolojileri yine kendi ürünü olan Microsoft Visual Studio Code IDE'si aracılığıyla geliştiricilere sunmaktadır. Bu sayede hem kendi ürünlerini pazarlamış hem de yazılım geliştirme süreçlerine katkıda bulunmuş oluyor. Proje geliştirme aşamasında Visual Studio Code'un sunduğu geliştirme arayüzü Şekil 5.11'de verilmiştir.



Şekil 5.11: Microsoft Visual Studio Code IDE arayüzü.

### 5.2.3 Mobil uygulamada kullanılan teknolojiler

Bu bölümde, geliştirilen mobil uygulama projesinde kullanılan temel yazılım geliştirme yaklaşımlarından bahsedilecektir. Hybrid Mobile Application Development bunların başında gelmektedir.

#### 5.2.3.1 Git

Yazılımı gerçekleştirilen bir projenin fiziksel olarak bilgisayarlarda durması, yazılımcılar için sakıncalı bir durumdur. Yazılımın bulunduğu bilgisayarın diskine herhangi bir sorun olması durumunda, geliştirilen tüm yazılım bir anda yok olabilir. Bu durumlara karşı, kodların fiziksel olarak bilgisayarlarda saklanması yerine bulut ortamında saklanabilmesi mümkündür. Ayrıca, birden fazla kişinin çalıştığı bir yazılım projesinde, farklı kişiler tarafından müdahale edilen kodların birleştirilmesi sorunu büyüktür.

Git, yazılan projelerin internet ortamında farklı versiyonlarda yedeklerinin olabileceği ve istenildiği tarihe ait kodlara bir anda geri gidilebileceği bir versiyon kontrol sistemidir. Hangi kod, kimin tarafından ne zaman yazılmış sorusunun cevabı da kolaylıkla git ile bulunabilmektedir. Projenin ilk gününden itibaren yapılan her kod gönderimi esnasında yedeklemeler alınmaktadır ve bu sayede istenildiği tarihe proje geri döndürülebilmektedir.

“Akıllı mobil ödeme sistemi” projesinde yazılım kodlar, git versiyon kontrol sistemi üzerinde tutulmuştur.

Git versiyon kontrol sistemi, “akıllı mobil ödeme sistemi” projesi için ayrıca Node JS kurulumunun da sağlanması için kullanılmıştır.

### **5.2.3.2 Node JS**

Bilindiği üzere Javascript, client side geliştirilen ve yazılımı client side koşan bir betik dildir. Client side derken Javascript internet tarayıcısı üzerinde çalışmaktadır. Node JS, Javascript ile sunucu taraflı uygulamalar geliştirmeye imkan sağlamaktadır. Kısaca Node JS, bir Javascript runtime platformudur denebilir. Node JS, Javascript kodunu makine koduna çevirme işini yürütmektedir. Bu sayede makine diline dönüştürülen kodlar sunucu taraflı çalışabilmektedir.

Javascript, asenkron çalışma mantığına çok uygun bir dil olduğu için makine koduna dönüşen Javascript kodları çok performanslı bir şekilde çalışabilmektedir. Node JS bu bağlamda kaynak yönetimini de çok efektif kullanabilmektedir.

Çok hızlı cevap verebilme özelliğine sahip olduğundan dolayı Reel Time uygulamalar için çok uygundur. Maliyet ve response time açısından Node JS, kendisine rakip diğer teknolojilere göre çok daha iyidir denebilir.

### **5.2.3.3 Node Package Modules (npm)**

“Akıllı mobil ödeme sistemi” projesi için bir mobil uygulama geliştirilmesi planlanmıştır. Mobil uygulama geliştirme süresince, bir takım eklentilere ihtiyaç duymuştur. Daha önceden bahsedildiği gibi Microsoft’un kullanıcılarına sunmuş

olduğu bir hizmet olan Nuget Package Manager sayesinde web servis yazılımına ek kütüphaneler ekleyebilmek mümkün olmuştur.

Aynı şekilde, geliştirilen mobil uygulamaya da ek kütüphaneler, plugin'ler ekleyebilmek için Node Package Modules (npm) kullanımı söz konusu olmuştur.

Node Package Modules (npm)'in bilgisayara kurulumu Node JS konsolu üzerinden gerçekleştirilmiştir. Şekil 5.12'de görüldüğü gibi Node Package Modules (npm)'in bilgisayar kurulumu konsol üzerinden “npm install npm@latest -g” komutu çalıştırılarak gerçekleştirilmiştir.

```

C:\WINDOWS\system32\cmd.exe

+-- read-installed@4.0.3
+-- util-extend@1.0.3
+-- retry@0.10.1
+-- sorted-object@2.0.1
+-- sorted-union-stream@2.1.3
+-- from2@1.3.0
+-- readable-stream@1.1.14
+-- core-util-is@1.0.2
+-- isarray@0.0.1
+-- string_decoder@0.10.31
+-- stream-iterate@1.2.0
+-- stream-shift@1.0.0
+-- text-table@0.2.0
+-- validate-npm-package-license@3.0.1
+-- spdx-correct@1.0.2
+-- spdx-license-ids@1.2.2
+-- spdx-expression-parse@1.0.4
+-- validate-npm-package-name@3.0.0
+-- builtins@1.0.3
+-- wrappy@1.0.2
+-- npm-lifecycle@2.0.0
+-- byline@5.0.0
+-- resolve-from@4.0.0
+-- npm-package-arg@6.0.0
+-- npm-packlist@1.1.10
+-- ignore-walk@3.0.1
+-- minimatch@3.0.4
+-- brace-expansion@1.1.8
+-- balanced-match@1.0.0
+-- concat-map@0.0.1
+-- npm-bundled@1.0.3
+-- npm-profile@2.0.5
+-- make-fetch-happen@2.5.0
+-- agentkeepalive@3.3.0
+-- humanize-ms@1.2.1
+-- ms@2.0.0
+-- cacache@9.3.0
+-- y18n@3.2.1
+-- http-cache-semantics@3.8.0
+-- http-proxy-agent@2.0.0
+-- agent-base@4.1.1
+-- es6-promisify@5.0.0
+-- es6-promise@4.1.1

```

Şekil 5.12: Node Package Modules (npm)'in bilgisayara kurulumu.

Node Package Modules (npm) kullanılarak aşağıdaki paketler, yanlarında verilen komutlarla “Akıllı mobil ödeme sistemi” projesi kapsamında geliştirilen mobil uygulamaya eklenmiştir:

- **Cordova (Phonegap):** npm install -g cordova
- **IONIC Framework:** npm install -g ionic

- **Bower:** npm install -g bower

Node Package Modules (npm) kullanılarak projeye eklenen yukarıda verilen paket içeriklerine ait detaylara, daha sonraki bölümlerde bahsedilmiştir.

#### 5.2.3.4 Bower

Bower, Twitter tarafından geliştirilen bir paket yöneticisidir. Bower sayesinde, geliştirilen front-end bir projeye ihtiyaç duyulan kütüphanelerin eklenmesini sağlamaktadır. “Akıllı mobil ödeme sistemi” projesi de bir front-end proje olduğundan dolayı bu projede de bir takım kütüphaneler ihtiyaç duyulmuştur. Bu kütüphanelerin bir kısmı Node Package Modules (npm) ile bir kısmı da Bower ile projeye eklenmiştir.

“Akıllı mobil ödeme sistemi” projesi kapsamında geliştirilen mobil uygulamaya Bower kullanılarak aşağıdaki paketler eklenmiştir.

- **JQuery:** bower install jquery
- **Angular JS:** bower install angular. Ayrıca npm ile de “npm install angular” komutu ile kurulabilir.
- **IONIC Nifty Modal:** bower install ionic-nifty-modal
- **Font Awesome:** bower install components-font-awesome –save

Bower kullanılarak projeye eklenen yukarıda verilen paket içeriklerine ait detaylara, daha sonraki bölümlerde bahsedilmiştir.

#### 5.2.3.1 Cordova (Phonegap)

Cordova, hybrid mobil uygulamalar geliştirmeyi sağlayan bir framework’tür (Phonegap, 2014). Cordova kullanılarak geliştirilmiş bir hybrid mobil uygulaması, native özellikler kazandırılmış bir web sayfasına benzer.

Geliştirilen cordova framework sayesinde, mobil cihazın donanım özellikleri hybrid uygulama tarafından kullanılabilir. Örnek olarak accelerometer, geolocation, kamera, network, storage, rehber, mikrofon, kulaklık

v.b. özellikler cordova framework sayesinde hybrid mobil uygulama sayesinde kullanılabilir.

Cordova, her bir donanım özelliğine erişebilmek için framework'ü alt parçalara ayırmıştır. Örneğin, medya özelliklerine erişebilmesi için “cordova-plugin-media” şeklinde framework'ü plugin'lere bölmüştür. Hybrid mobil uygulama için hangi plugin'ler kullanılacaksa, mobil uygulama projesine sadece kullanılacak plugin'ler eklenmektedir. Tüm framework kütüphanesi tamamıyla eklenmemektedir. Geliştirici, hangi donanım özelliğine erişmek, onu kullanmak istiyorsa ilgili donanım özelliğine ait cordova plugin'ini yüklemesi yeterli olmaktadır.

“Akıllı mobil ödeme sistemi” kapsamında geliştirilen mobil uygulama Cordova kullanılarak geliştirilmiştir.

### 5.2.3.2 IONIC Framework

IONIC Framework, Cordova üzerine geliştirilmiş bir hybrid mobil uygulama geliştirme framework'udur (IONIC Framework, 2017). Mobil uygulamada kullanılması planlanan donanım özelliklerinin kullanımı IONIC Framework üzerinden sağlanmaktadır. IONIC Framework de alt tarafta Cordova'yı kullandığı için dolaylı olarak Cordova kullanılarak donanım özelliklerine erişilmiş olur.

IONIC Framework, kendi çatısı altında birden fazla teknoloji barındırmaktadır. Daha önceden bahsedildiği gibi, altyapısında Cordova kullanmaktadır. Diğer bir yandan da Angular JS kütüphanesi üzerine inşa edilmiştir. IONIC Framework'de bulunan kod geliştirme mantığı da Angular JS'deki mantığın aynısıdır. Yazılım şekli çok benzerdir. Bunlara ek olarak IONIC Framework kullanarak bir uygulama geliştirmek isteniyorsa temel olarak HTML, Javascript ve Angular JS bilmek yeterlidir.

IONIC Framework, birçok native özelliğe destek verebilmektedir. Barkod okuma, bluetooth, parmak izi okuma gibi birçok native özelliği kullanabilmektedir.

“Akıllı mobil ödeme sistemi” kapsamında geliştirilen mobil uygulama, IONIC Framework kullanılarak geliştirilmiştir. Şekil 5.13'te görüldüğü gibi mobil

uygulamada IONIC Framework ile Angular JS birleştirilmiş yapısı görülmektedir. Belirtilen controller içerisinde IONIC özellikleri tanımlanmıştır.

```

1  angular.module("your_app_name.controllers", ['ionic', 'ngCordova'])
2
3  .controller("AuthCtrl",["$scope", "$ionicPlatform", "$ionicModal", "$state",
4    "$ionicActionSheet", "$ionicLoading", "$rootScope",
5    function ($scope, $ionicPlatform, $ionicModal, $state, $ionicActionSheet,
6      $ionicLoading, $rootScope) {
7
8    $rootScope.checkStorageData(window.localStorage.user) && ($state.go("app.home")),
9
10   // ModalView
11   $scope.modal = {},
12
13   $scope.openModal = function (animation) { ...
14   },
15
16   $scope.closeModal = function () { ...
17   },
18   // ModalView End
19
20   // Loading Spinner
21   $scope.show = function () { ...
22   },
23
24   $scope.hide = function () { ...
25   },
26
27   $scope.openLoading = function () { ...
28   },
29
30   $scope.closeLoading = function () { ...
31   },
32   // Loading Spinner End
33
34   // Login
35   $scope.user = { ...
36   },
37   $scope.authError = {},
38   $scope.doLogin = function () { ...
39   },
40
41   $scope.doStart = function () { ...
42   }
43 })

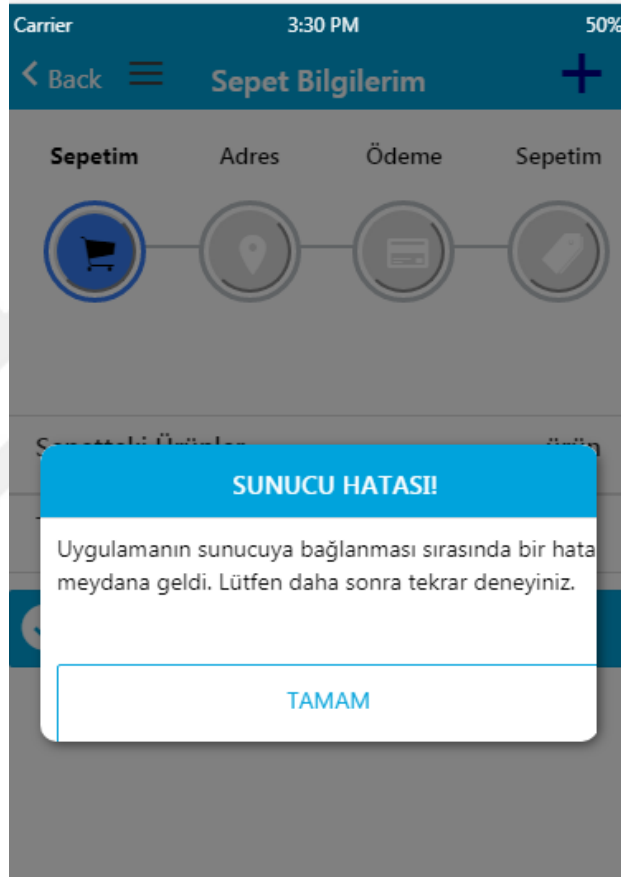
```

Şekil 5.13: Mobil uygulamada IONIC Framework kullanımı örneği.

- **IONIC Nifty Modal**

IONIC Nifty Modal, IONIC Modal box denilen pop-up penceresini çalıştırmak için kullanılacak ek CSS sınıfları grubudur. IONIC, sunmuş olduğu modal view dışında, modelin açılışı için de ekstra davranışlar sergileyebilecek CSS sınıfları sunabilmektedir. Varsayılan olarak, animasyonunda “slide-in-up” CSS class’ını kullanmaktadır. Bunun anlamı mobil uygulama içerisinde herhangi bir modal tetiklemesi söz konusu olduğunda açılacak olan modal view için animasyon “slide-in-up” olarak açılmaktadır. Dahil edilen sınıflar IONIC Nifty Modal ve bounce.js kaynaklıdır. Animasyonlar sadece modal ile sınırlı değildir ve uygulamanızdaki diğer bileşenler için yeniden kullanılabilir.

“Akıllı mobil ödeme sistemi” projesi için geliştirilen mobil uygulama içerisinde birçok yerde IONIC Nifty Modal kullanımı söz konusu olmuştur. Web servisten veri talep etme esnasında oluşabilecek sunucu veya istemci taraflı bir hatayı yakalayıp IONIC Nifty Modal ile bilgilendirme penceresi bu teknoloji sayesinde gerçekleştirilmektedir. Ayrıca network tabanlı oluşabilecek herhangi bir sorunu da yakalayıp IONIC Nifty Modal içerisinde göstermektedir. Bu bağlamda mobil uygulama kullanılan örnek bir network hatası gösterimi Şekil 5.14’te gösterildiği gibidir.



Şekil 5.14: IONIC Nifty Modal ile network hatası gösterimi.

### 5.2.3.3 Angular JS

Angular JS, Model View Controller (MVC) desenini aktif bir şekilde kullanan bir Javascript framework’üdür (Wahlin, D., 2014). Google tarafından desteklenmektedir. Angular JS, client side uygulamalar için çok kullanışlı bir teknolojidir. Sunmuş olduğu yöntemler sayesinde JQuery gibi kütüphanelere ihtiyaç duymamaktadır. Single Page App denilen uygulamaları geliştirmede etkindir.

Benzer kütüphanelerden farkı olarak çift yönlü çalışabilmesi en büyük avantajlarından biridir. Çift yönlü çalışabilme mantığında Model View-View-ViewModel (MVVM) kullanmaktadır.

Angular JS, ayrıca Hypertext Markup Language (HTML) tag'lerinin geliştirilmesine olanak sağlar. Sunmuş olduğu “directive” yöntemleri sayesinde özelleştirilmiş HTML tag'larını kullanmaya olanak sağlar. Şekil 5.15'teki örnekte gösterildiği gibi geliştirilen mobil uygulama içerisinde html tag'leri geliştirilerek “tabs” adında farklı sekmelerden oluşabilecek bir arayüz yapısı geliştirilmiştir.

```

306 .directive("listeleTabs", function() {
307     return {
308         restrict: "E",
309         transclude: !0,
310         scope: {},
311         controller: ["$scope", function(n) {
312             var e = n.tabs = [];
313             n.select = function(t) {
314                 angular.forEach(e, function(n) {
315                     n.selected = !1
316                 }),
317                 t.selected = !0,
318                 n.$emit("listele-tabs-changed", t)
319             },
320             this.addTab = function(t) {
321                 0 === e.length && (
322                     n.select(t)
323                 ),
324                 e.push(t)
325             }
326         }],
327         templateUrl: "views/templates/common/listeletab/listele-tabs.html"
328     }
329 })

```

Şekil 5.15: Projede Angular JS Directive kullanımı.

#### 5.2.3.4 Telerik Kendo UI

Telerik tarafından geliştirilen Kendo UI, bir HTML framework'üdür. Ayrıca kullandığı teknolojisi sayesinde .NET Framework gibi sunucu tarafı çalışan teknolojilere kullandığı Javascript ve HTML teknolojileri sayesinde destek olmaktadır (Adams, J, 2013).

Kendo UI, geliştiriciler için arayüzlerde kullanılmak üzere kullanışlı komponentler sunmaktadır. Sunduğu komponentler sayesinde arayüz geliştirilmesi kolay hale gelmekte ve bu tarz ihtiyaçlara cevap vermektedir.

Sunduğu bileşenlere ListView, DatePicker ve DateTimePicker, Calendar, Tooltip, TreeList ve Grid örnek olarak gösterilebilir. Bir uygulamanın

arayüzünde kullanılabilecek birçok bileşene sahiptir. Kullandığı client-side ve server-side teknolojileri sayesinde aynı bileşenler hem client-side hem de server-side kullanılabilmektedir. Arayüze veri taşıma işlemleri olarak Modal-View-ViewModel (MVVM) yapısını etkin olarak kullanmaktadır.

Geliştirilen mobil uygulamada gerekli görülen yerlerde Telerik tarafından geliştirilen Kendo UI bileşenleri kullanılmıştır.

#### **5.2.3.5 JQuery**

Temelinde Javascript barındıran bir kütüphanedir. Kullanıcı deneyimini artırıcı birçok özelliği içinde barındırmaktadır. HTML veya CSS ile geliştirilen bir uygulamanın bu teknolojilerin gerçekleştiremediği kısımlarda birçok işlevi sayesinde ihtiyaçları kolaylıkla karşılayabilen bir framework'tür (JQuery Community Experts, 2010).

JQuery yazılım tabiriyle “lightweight” bir framework'tür. Bunun anlamı daha az kod yazarak çok daha fazla iş sağlayabilmektedir. JQuery geliştiricilere bu kolaylığı sunmaktadır. Satırlarca Javascript kod ile gerçekleştirilebilecek bir olayı JQuery bir veya birkaç satır kod ile kolaylıkla gerçekleştirebilmektedir. JQuery aynı zamanda Asynchronous JavaScript and XML (AJAX) request'lerini ve Document Object Modal (DOM) nesnelerini değiştirmek gibi Javascript'te karmaşık olan görülen bu kısımların kullanımını kolaylaştırmaktadır.

“Akıllı mobil ödeme sistemi” projesinde geliştirilen melez mobil uygulama içerisinde Document Object Modal (DOM) yapılarından tutun, Asynchronous JavaScript and XML (AJAX) request'lere kadar birçok yerde kullanılmıştır.

#### **5.2.3.6 Asynchronous JavaScript and XML (AJAX)**

Asynchronous JavaScript and XML (AJAX), temelde bir programlama dili değildir. JQuery ve JavaScript temelini kullanan client-side çalışan betik bir dildir. Web sunucusundan bir veri talep etmek için kullanılmaktadır. İsim olarak yanıltıcı bir anlam taşıyabilir. Bu nedenle, Asynchronous JavaScript and XML (AJAX) uygulamaları verileri taşımak için Extensible Markup Language (XML) kullanabilir. Ancak, verileri Javascript Object Notation (JSON) olarak da kolaylıkla taşıyabilmek için kullanılan bir yöntemdir (Batra, S, 2003).

Geliştirilen mobil uygulamanın, ilgili verileri (kullanıcı bilgileri, ürün bilgileri, ödeme bilgileri vs.) elde edebilmesi için Microsoft ASP. NET MVC Web API servislerine erişebilmesini sağlayacak olan bir yöntemdir. Veriler, REST servsinden POST methodu ile alınmakta ve her servis methoduna erişmede mobil uygulamaya login olan her kullanıcı için kimlik doğrulaması yapılmaktadır.

Geliştirilen “akıllı mobil ödeme sistemi” hybrid mobil uygulaması ile server tarafında ASP .NET MVC Web API ile geliştirilen web servis ile haberleşebilmesi için kullanılmıştır. Mobil uygulamada kullanıcı, kendi kullanıcı adı ve şifresiyle web servise nasıl request’te bulunduğu Şekil 4.56’da görülmektedir. “user” adında oluşturulan Angular object ile “UserName” ve “Password” parametreleri arayüzden alınmıştır. Daha sonra kullanıcı arayüzde bulunan “Giriş” butonuna basmasıyla “\$scope.doLogin” metodu tetiklenmektedir. Bu tetiklenme sonrasında Asynchronous JavaScript and XML (AJAX) işlemi çalışmaktadır.

Asynchronous JavaScript and XML (AJAX) çağrısında Şekil 5.16’da gösterilen ve IONIC Framework&AngulaJS tarafında tutulan “\$rootScope.endPoints.methods.login” sayesinde bağlanacağı web servis url bilgisi elde edilmektedir.

AJAX metot içinde gösterilen “data” parametresi ile de arayüzden alınan kullanıcı adı ve şifresi “\$scope.user” parametresi ile web servise istekte bulunmadan önce parametre olarak verilmektedir. Ayrıca web servis ile haberleşme esnasında kullanılacak veri taşıma teknolojisinin de “JSON” olduğu gözükmemektedir.

```

47 // Login
48 $scope.user = {
49   Username   : "",
50   Password   : "",
51   PlatformID : "denmeplatformid",
52   PlatformType : "2",
53   Culture    : "tr-TR"
54 },
55 $scope.authError = {},
56 $scope.doLogin = function () {
57   $scope.authError = {},
58   $.ajax({
59     type      : "POST",
60     url       : $rootScope.endPoints.methods.login,
61     data      : $scope.user,
62     dataType  : "json",
63     beforeSend : function () { $scope.show($ionicLoading) },
64     complete  : function () { $scope.hide($ionicLoading) },
65     crossDomain : true,
66     success   : function (r) {
67       $scope.hide($ionicLoading),
68       $rootScope.isNullUndefinedOrEmpty(r) ?
69       (
70         r.IsSuccess ? (
71           $rootScope.user = r.Data,
72           window.localStorage.user = JSON.stringify(r.Data),
73           $state.go("app.home")
74         ) : (
75           $scope.authError.title = "Program başlatılamadı!",
76           $scope.authError.content = "Program verileri alınırken bir hata meydana geldi. Lütfen daha sonra tekrar deneyiniz.",
77           $scope.openModal($rootScope.modalAnimationClasses.jelly)
78         )
79       ) : (
80         $scope.authError.title = "Program başlatılamadı!",
81         $scope.authError.content = "Program verileri alınırken bir hata meydana geldi. Lütfen daha sonra tekrar deneyiniz.",
82         $scope.openModal($rootScope.modalAnimationClasses.jelly)
83       )
84     },
85     error: function (e) {
86       $scope.authError.title = "Sunucu Hatası",
87       $scope.authError.content = "Uygulamanın sunucuya bağlanması sırasında bir hata meydana geldi. Lütfen daha sonra tekrar deneyiniz.",
88       $scope.openModal($rootScope.modalAnimationClasses.jelly)
89       //angular.isObject($scope.loginError) && ()
90     }
91   })
92 },
93 },
94 },
95 }

```

Şekil 5.16: AJAX ile gerçekleştirilen Login Request işlemi.

Web servis tarafında request gerçekleştirildikten sonra sunucu isteği işler ve geriye response olarak dönmektedir. Bu aşamada Asynchronous JavaScript and XML (AJAX) teknolojisi metod içerisinde barındırdığı “success” metodu sayesinde response bilgisi yakalanabilmektedir.

Sunucu tarafında çalışan web servis isteğe cevap döndüğünde bu istek, network üzerinden talepte bulunan Asynchronous JavaScript and XML (AJAX) metodu “success” kısmını tetiklemektedir. “success” kısmı tetiklendikten sonra web servis tarafından dönen veri aşağıdaki şekilde de görüldüğü gibi Javascript Object Notation (JSON) formatında “r” parametresi ile yakalanabilmiştir.

Sunucudan dönen JSON veriye göre, kullanıcının girmiş olduğu kullanıcı adı ve şifre bilgisinin doğruluğu kontrol edilir. Kullanıcı, eğer doğru bilgileri girmişse IONIC Framework’ün sunduğu routing teknolojisi olan “\$state.go(‘app.home’)” ile kullanıcı ana sayfaya yönlendirilir. Eğer kullanıcının girmiş olduğu bilgiler hatalı ise, bu bilgileri tekrar girmesi için bilgilendirilir ve içeriye alınmaz.

Ayrıca web servis tarafında, daha önceden de bahsedildiği gibi eğer kullanıcı adı ve şifresi doğruysa, kullanıcının ana sayfa içerisinde gerçekleştirmek istediği işlemleri sağlayabilmesi için ona girmiş olduğu kimlik bilgilerine özgü token üretilir. Bu token, ana sayfa içerisinde kullanıcının, diğer işlevleri gerçekleştirebilmesi için web servis tarafında kullanılacak bir bilgi olarak LocalStorage içerisinde saklanacaktır.

Ana sayfa içerisinden, herhangi bir istek oluşturulması gerektiğinde LocalStorage içerisinde saklanan ve içerisinde token bilgisinin de barındığı “user” bilgisi bu sayede kullanılabilir ve erişilebilir olacaktır.

### 5.2.3.7 Font Awesome Icons

Font Awesome Icons, web ve mobil uygulamalarda sıklıkla kullanılan icon’lardan oluşan bir fonttur. Bir mobil uygulamanın ihtiyaç duyabileceği birçok iconu bünyesinde barındırmaktadır.

Font Awesome icon’larını kullanabilmek için yapılması gereken, Font Awesome’nin geliştiricilere verdiği “fontawesome.css” CSS kütüphanesini projeye eklemektir. Bu işlem tamamlandıktan sonra, projede kullanılacak herhangi bir icon için “<i class=""></i>” tag’leri arasına ilgili icon’un css class’ını eklemek yeterli olacaktır. Bir icon, farklı boyutlarda kullanılmak istenirse, Font Awesome’nin sunmuş olduğu birden çok boyut içeren css class’ları kullanılabilir.

“Akıllı mobil ödeme sistemi” projesinde geliştirilen mobil uygulamanın tüm icon’ları Font Awesome icon’larıdır. Giriş ekranındaki icon’lar, ana menü içerisindeki menü icon’ları ve sepet bilgilerinin bulunduğu icon’lar bunlara örnek olarak gösterilebilir. Şekil 5.17’de gösterilen ana menü, geliştirilen mobil uygulamada kullanılan Font Awesome icon’larına örnek olarak gösterilebilir.



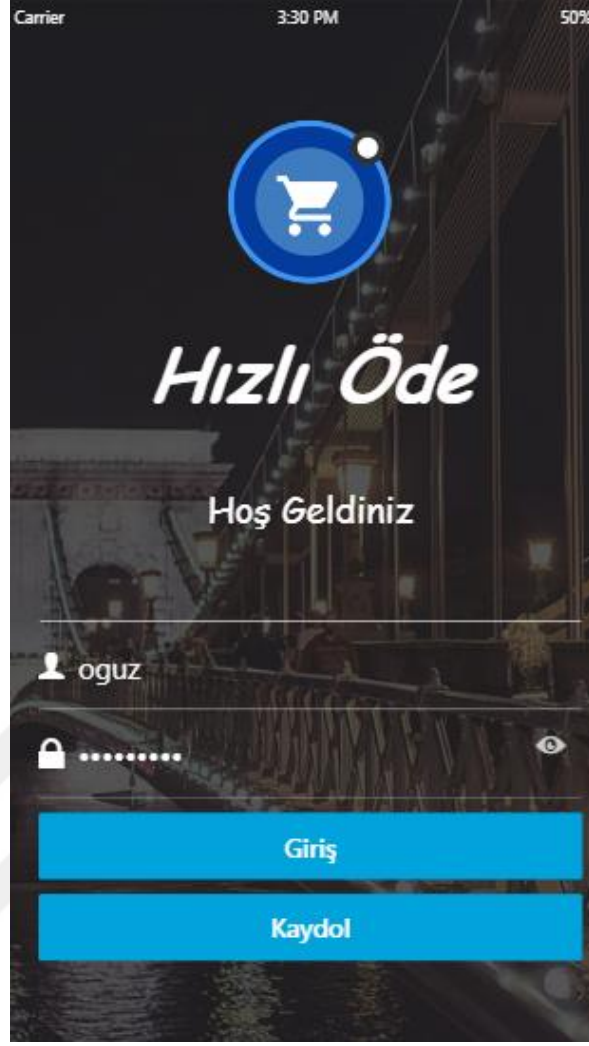
Şekil 5.17: Mobil uygulamada kullanılan Font Awesome Icon'larına örnek.

### 5.2.3.8 Hyper Text Markup Language 5 (HTML5)

Hyper Text Markup Language (HTML), Tim Berners-Lee tarafından 80'li yıllarda geliştirilmiştir. Hyper Text Markup Language (HTML) adı 90'lı yıllarda konmuştur.

Tasarlanan “akıllı mobil ödeme sistemi” için geliştirilen mobil uygulama, hybrid mobil uygulama teknolojileri ile geliştirildiği için ağırlıklı olarak HTML5 kullanımı söz konusu olmuştur.

Mobil uygulama içerisinde uygulamayı açan kullanıcıların sisteme login olabilmeleri için tasarlanan uygulama arayüzü de HTML5 kullanılarak geliştirilmiştir. Bu bağlamda sisteme giriş arayüzü olan login ekranı Şekil 5.18'deki gibi tasarlanmıştır.



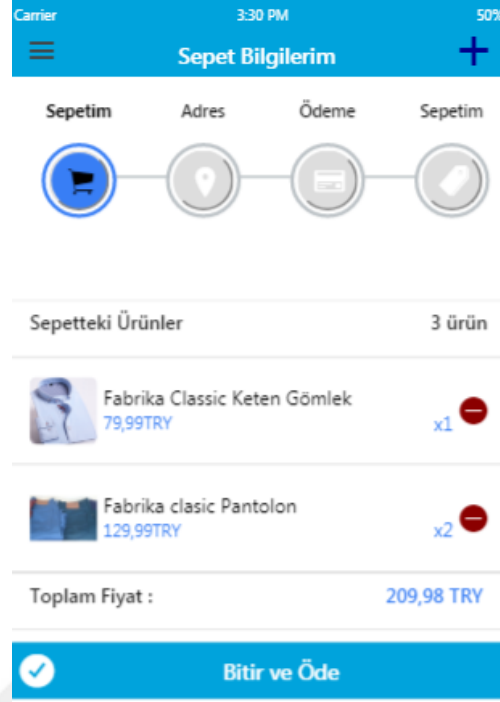
Şekil 5.18: Mobil uygulamada HTML5 ile geliştirilen Login ekranı.

### 5.2.3.9 Cascading Style Sheets 3 (CSS3)

Tasarlanan “akıllı mobil ödeme sistemi” için geliştirilen mobil uygulama, hybrid mobil uygulama teknolojileri ile geliştirildiği için ağırlıklı olarak CSS3 kullanımı söz konusu olmuştur.

Mobil uygulama içerisinde uygulamayı açan kullanıcıların sisteme login olduktan sonra açılan ana sayfada bulunan menü içerisinde, seçilen “Sepet Bilgilerim” ekranı için tasarlanan uygulama arayüzü de CSS3 kullanılarak geliştirilmiştir.

Bu bağlamda “Sepet Bilgilerim” arayüzü Şekil 5.19’deki gibi tasarlanmıştır.



Şekil 5.19: Mobil uygulamada CSS3 ile geliştirilen sepet bilgileri ekranı.

#### 5.2.3.10 Mobil uygulama ile native özelliklerin kullanımı

Mobil cihazın, kamera, titreşim, mikrofon, kulaklık, telefon rehberi, geolocation veya gyrosensor gibi özellikleri native özellikler olarak adlandırılmaktadır.

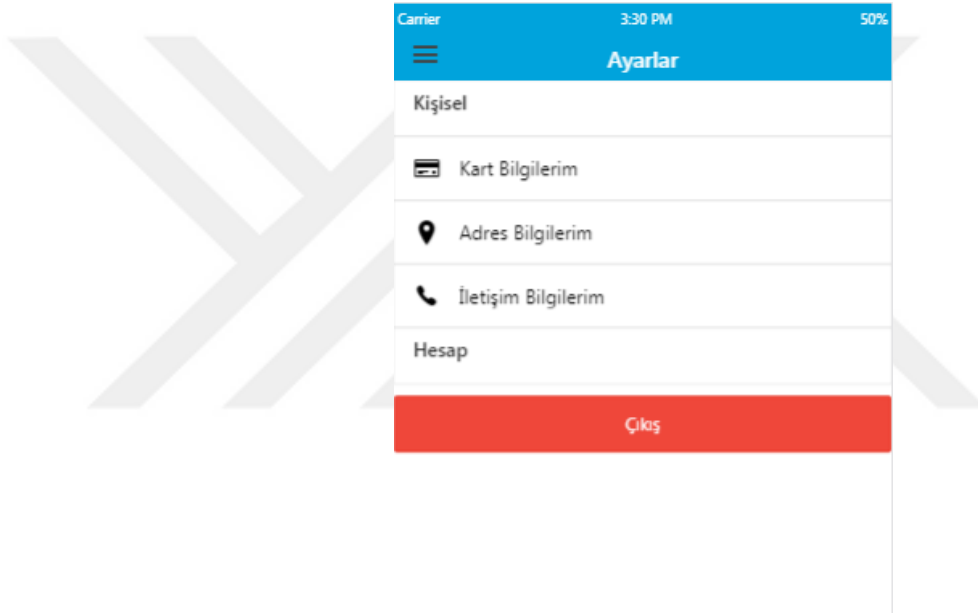
Geliştirilen “akıllı mobil ödeme sistemi” projesindeki mobil uygulama içerisinde native özellik olarak media ve kamera kullanılmıştır. Kamera kullanımı ile ilgili daha sonraki kısımlarda bilgi verilmiştir.

- **ActionSheet**

ActionSheet, mobil uygulama geliştirmede kullanılan native özelliklerden biridir. Mobil uygulama arayüzünde ihtiyaç halinde, uygulamanın belirli yerlerinde kullanıcıya bir takım seçimler sunmak gerekebilir. Bunun dışında bazı olayları işleme almadan önce kullanıcıdan onay alınması da gerekebilir. Bu gibi durumlarda ActionSheet kullanımı söz konusu olmaktadır.

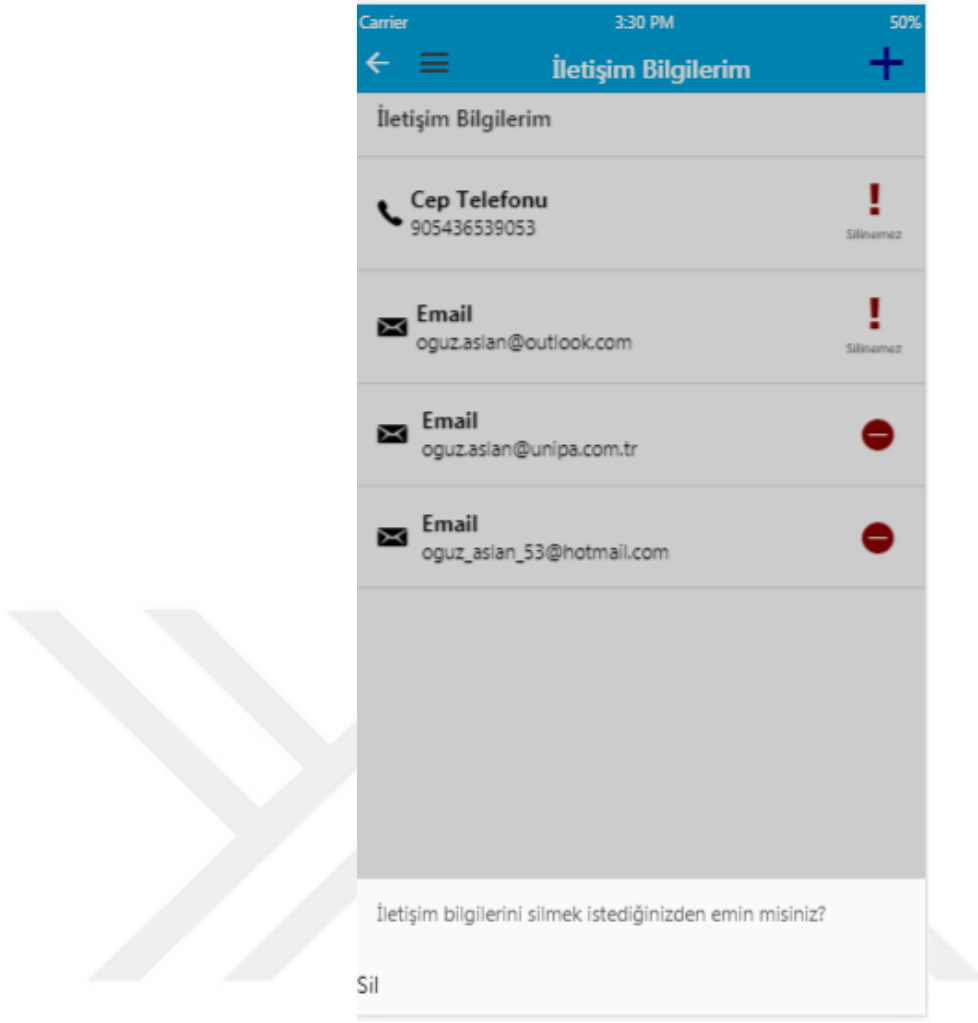
ActionSheet, mobil uygulamanın açık olan ekranında kullanıcı karşısına popup çıkarmaktadır. Popup içerisinde de kullanıcıdan seçilmesi istenen menüler doldurulmaktadır. Örneğin kullanıcı, ürünleri mobil uygulama vasıtasıyla sepete eklemek istiyor. Kullanıcı, sepete ekle butonuna bastığında sepete ekleyeceği ürünün renk seçeneği ActionSheet kullanılarak kullanıcıdan alınabilir.

“Akıllı mobil ödeme sistemi” projesi için geliştirilen mobil uygulama içerisinde ActionSheet kullanımı söz konusudur. Bu bağlamda mobil uygulamaya login olan kullanıcı ana sayfa içerisinde bulunan yan menüden “Ayarlar” işlemini seçer. Bu işlem tamamlandıktan sonra Şekil 5.20’deki menü seçenekleri karşısına çıkmaktadır.



Şekil 5.20: Mobil uygulamada ayarlar menüsü.

Ayarlar menüsü içerisinde “İletişim Bilgilerim” seçeneğini seçtikten sonra kullanıcının eklemiş olduğu iletişim bilgileri karşısına çıkar. İletişim bilgilerinden herhangi birini silmek isteyen kullanıcı Şekil 5.21’de görüldüğü gibi iletişim bilgisinin sağ tarafında bulunan çıkar butonuna basar. Bu buton, web servis aracılığıyla veritabanında bulunan kişinin seçili iletişim bilgisini sileceği için, kullanıcıdan silme işlemi öncesinde onay alınması uygun görülmüştür.



Şekil 5.21: Mobil uygulamada ActionSheet kullanımı.

Kullanıcı silmek istediği iletişim bilgisinin sağ tarafında bulunan çıkar butonuna bastığında, kullanıcının karşısına “Silmek istediğinizden emin misiniz?” sorusu yöneltilir. Evet ve vazgeç seçenekleri ActionSheet yardımıyla kullanıcıya sunulur.

Mobil uygulamada ActionSheet kullanımı, kullanıcıdan iletişim bilgilerini silmek istemesi söz konusu olduğunda onay alınması durumu için kullanılmıştır. Mobil uygulama içerisinde verilen örnek gibi birçok yerde kullanımı söz konusudur.

ActionSheet’in mobil uygulamada kullanımı Şekil 5.22’deki, gibi gerçekleştirilmiştir.

```

1728     $scope.deleteContact = function(contact){
1729         contact.State !== 0001 && (
1730             $ionicActionSheet.show({
1731                 destructiveText : "Sil",
1732                 titleText       : "İletişim bilgilerini silmek istediğinizden emin misiniz?",
1733                 cancelText      : "Vazgeç",
1734                 cancel          : function () { },
1735                 buttonClicked   : function (scope) {
1736                     return !0
1737                 },
1738                 destructiveButtonClicked: function () {
1739
1740                     $rootScope.contactToDelete = {
1741                         UserID      : $scope.user.UserID,
1742                         Token       : $scope.user.Token,
1743                         Id          : contact.Id,
1744                         Explanation : contact.Explanation,
1745                         Type        : contact.Type
1746                     },
1747                     $.ajax({
1748                         type        : "POST",
1749                         url         : $rootScope.endPoints.methods.deleteusercontact,
1750                         data        : $rootScope.contactToDelete,
1751                         dataType     : "json",
1752                         beforeSend   : function () { $scope.show($ionicLoading) },
1753                         complete    : function () { $scope.hide($ionicLoading) },
1754                         crossDomain : true,
1755                         success     : function (r) {
1756                             $scope.hide($ionicLoading),
1757                             $rootScope.isNullUndefinedOrEmpty(r) ?
1758                             (
1759                                 r.IsSuccess ? (
1760                                     $scope.getContacts()

```

Şekil 5.22: Mobil uygulamada ActionSheet uygulanması.

- **Kamera ile barcode okuma**

“Akıllı mobil ödeme sistemi” kapsamında geliştirilen mobil uygulamada, mağaza içerisinde bulunan ürünlerin barcode’larını okuyabilmek amacıyla, mobil uygulama içerisine mobil cihazın kamerasını kullanan bir eklenti eklenmesi uygun görülmüştür. Bu bağlamda, mobil cihazının kamerasına erişen mobil uygulama ile raflardan satın alınması planlanan ürünlerin barcode’ları okutulabilecek ve bu sayede mağaza sunucularında tutulan ürün bilgilerine erişilebilecektir. Ürün bilgilerine erişebilen mobil uygulama ile de müşterilerin hangi ürünlere ne kadar ödemesi gerektiği bilgisi oluşturulacaktır.

Geliştirilen mobil uygulama sayesinde, kamera vasıtasıyla ürünleri barcode’ları okutulur ve fiziksel anlamda da sepete yerleştirilir. Sepete yerleştirilen ürünlere ait bilgiler, mobil uygulama arayüzü vasıtasıyla da görüntülenebilir. Bu sayede hangi ürünlerin sepete eklendiği bilgisine erişilmiş olur.

### 5.2.3.11 Bootstrap

Bootstrap, geliştiricilerin ücretsiz olarak erişebileceği bir CSS framework'tür. Ayrıca, open source olarak geliştirilmiştir. Farklı ekran büyüklüklerine sahip cihazlarda, tasarlanan arayüzün güzel gözükmelerini sağlamaktadır. Neredeyse her bir mobil cihazın ekran büyüklüğünün farklı olduğu varsayılırsa, bootstrap kullanmak çok doğru bir tercih olabilmektedir. Bootstrap sayesinde geliştirici, geliştirmiş olduğu arayüzü tüm ekran boyutlarına uygun olarak her bir ekran için teker teker geliştirmek zorunda kalmamaktadır.

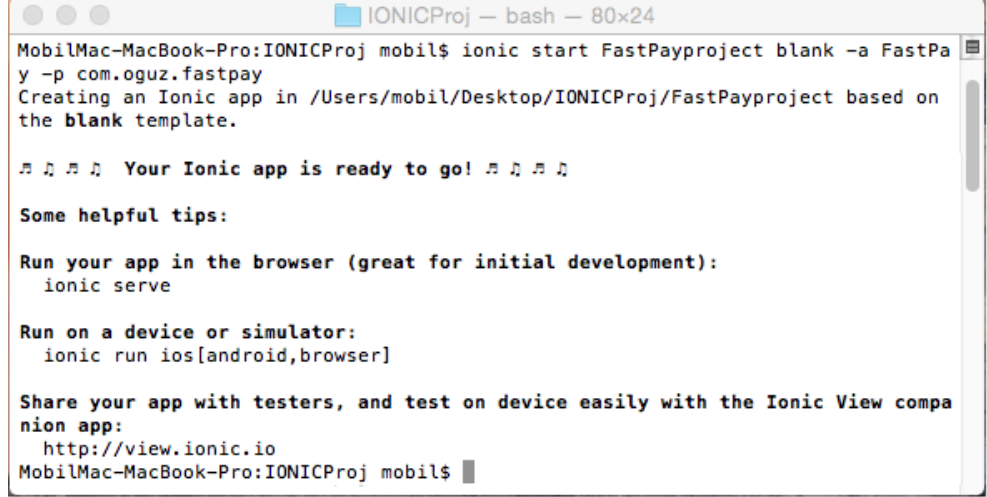
Geliştirilen mobil uygulama, hybrid mobil uygulama teknolojilerini içerdiği için bünyesinde HTML5 ve CSS3 kodlarını barındırmaktadır. Bu sayede, Bootstrap'ın da CSS framework'ü olduğu varsayıldığında, geliştirilen mobil uygulamada Bootstrap kullanımı yoğun bir şekilde söz konusu olmuştur. Bu sayede, geliştirilen mobil uygulama içindeki tüm ekranlar, farklı ekran boyutlarına sahip mobil cihazlar için teker teker herbirine uygunlu hale getirmek için arayüzler tekrar tasarlanmamıştır. Bootstrap kullanılarak geliştirilen arayüzler, farklı ekran büyüklüklerine sahip olan mobil cihazları destekler nitelikte olmuştur.

## 5.3 Mobil Uygulamanın Gerçekleştirilmesi

Bu bölümde, hybrid mobil uygulama teknolojileri kullanılarak geliştirilen mobil uygulamanın nasıl oluşturulduğu, IONIC Modal, IONIC Popup, IONIC Spinner gibi mobil uygulamada kullanılan arayüz bileşenleri konularından bahsedilmiştir. Ayrıca mobil uygulama içerisinde bilgilerin saklanmasını sağlayan LocalStorage kavramından ve mobil uygulamanın IOS ve Android işletim sistemine sahip mobil cihazlara kurulumlarının nasıl olacağından bahsedilmiştir.

### 5.3.1 Mobil uygulama projesinin oluşturulması

Daha önceden de bahsedildiği gibi mobil uygulama projesinde hybrid mobil uygulama teknolojileri kullanılmıştır. Hybrid mobil uygulama projesinde IONIC Framework üzerinde geliştirmeler yapılmıştır. Bu bağlamda, “akıllı mobil ödeme sistemi” için geliştirilen mobil uygulama projesinin oluşturulması için aşağıdaki şekilde görüldüğü gibi IONIC kontrol komutları çalıştırılmıştır. Şekil 5.23'teki gösterilen komut çalıştırılarak hybrid mobil uygulama projesi geliştirmeye hazır hale getirilmiştir.



```

IONICProj — bash — 80x24
MobilMac-MacBook-Pro:IONICProj mobil$ ionic start FastPayproject blank -a FastPay
y -p com.oguz.fastpay
Creating an Ionic app in /Users/mobil/Desktop/IONICProj/FastPayproject based on
the blank template.

🎵 🎵 🎵 🎵 Your Ionic app is ready to go! 🎵 🎵 🎵 🎵

Some helpful tips:

Run your app in the browser (great for initial development):
  ionic serve

Run on a device or simulator:
  ionic run ios[android,browser]

Share your app with testers, and test on device easily with the Ionic View compa
nion app:
  http://view.ionic.io
MobilMac-MacBook-Pro:IONICProj mobil$

```

Şekil 5.23: Mobil uygulama projesinin oluşturulması.

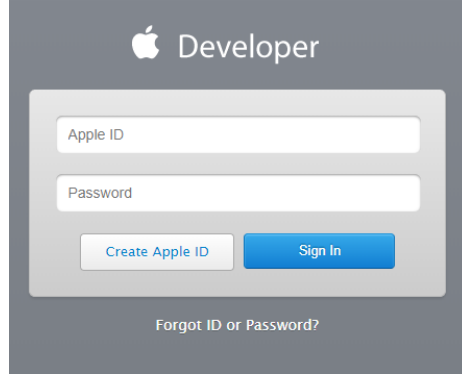
### 5.3.2 Mobil uygulamanın cep telefonuna kurulumu

Bu bölümde, farklı mobil işletim sistemlerine sahip cihazlara geliştirilen mobil uygulamanın hangi aşamalar tamamlandıktan sonra kurulduğu bilgisi anlatılmıştır. Özellikle dünyada çok kullanılan Android ve IOS işletim sistemlerine sahip cihazlara, “akıllı mobil ödeme sistemi” kapsamında geliştirilen mobil uygulamanın kurulumunun nasıl yapıldığı bilgisini içermektedir.

#### 5.3.2.1 IOS işletim sistemine kurulumu

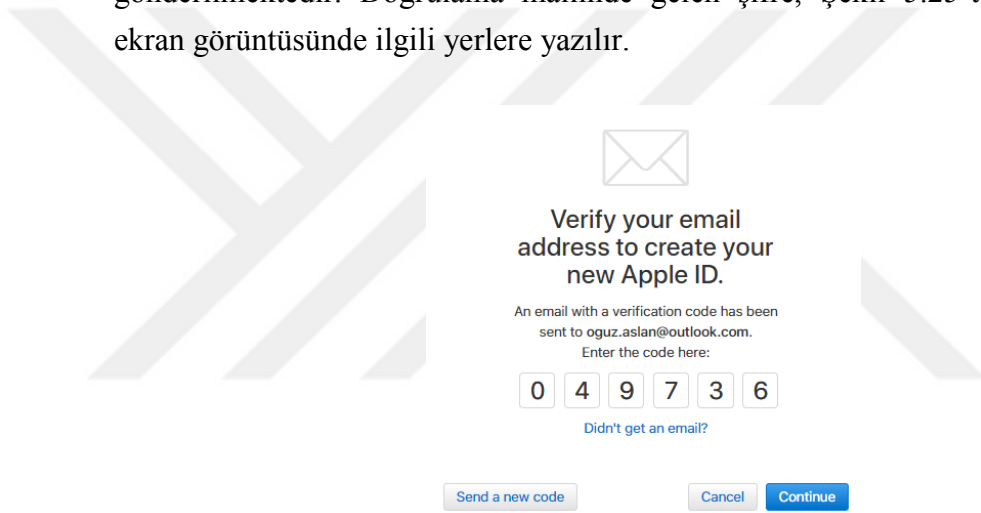
Geliştirilen mobil uygulamanın IOS işletim sistemine sahip mobil cihaza yüklenebilmesi için öncelikle Apple Developer Program hesabın satın alınmış olması gerekmektedir. Aksi taktirde IOS işletim sistemine sahip mobil cihaza, geliştirilen mobil uygulama yüklenememektedir.

Apple Developer hesabı satın almak için öncelikle Apple ID oluşturulması gerekmektedir. Şekil 5.24’te görüldüğü gibi <https://developer.apple.com/account> adresine giriş yaptıktan sonra aşağıdaki şekilde de görüldüğü gibi “Create Apple ID” butonuna basılır. Ardından ekrana çıkan form bilgileri dolurularak “continue” butonuna basılır.



Şekil 5.24: Apple Developer Program login ekranı.

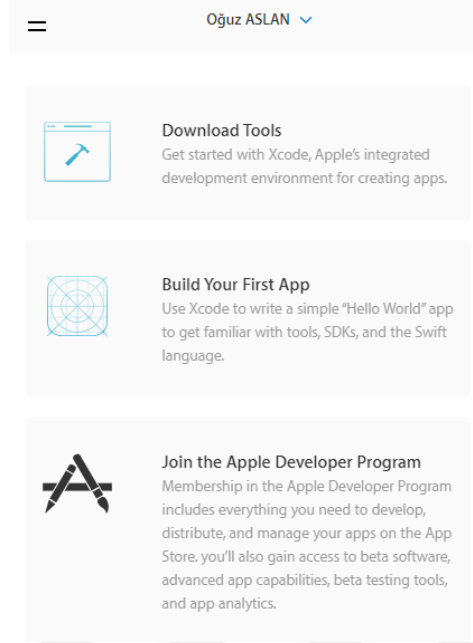
Tüm bu adımlar tamamlandıktan sonra, e-mail adresine bir doğrulama maili gönderilmektedir. Doğrulama mailinde gelen şifre, Şekil 5.25'teki gibi çıkan ekran görüntüsünde ilgili yerlere yazılır.



Şekil 5.25: Apple Developer Program hesabı doğrulama işlemi.

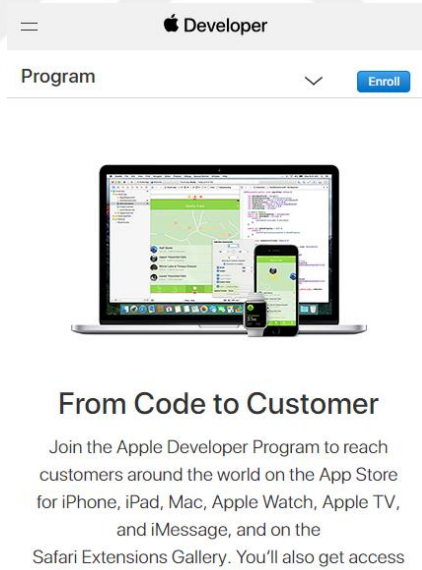
Doğrulama kodu girildikten sonra “Continue” butonuna basılır. Kullanıcı doğrulama işlemi bu şekilde tamamlandıktan sonra, belirlenen kullanıcı adı ve şifre ile <https://developer.apple.com/account> adresinden giriş yapılır. İlk giriş yapıldığından dolayı, yasal anlaşma metni ekranda çıkmaktadır. Yasal anlaşma sözleşmesini kabul ettikten sonra “Submit” butonuna basılır. Bu işlemlerden sonra Apple ID elde edilmiş olur.

Apple ID hesabına giriş yaptıktan sonra, market ve mobil uygulama işlemlerini gerçekleştirebilmek için Apple Developer Program hesabının satın alınması gerekmektedir. Apple Developer Program hesabını satın alabilmek için, Şekil 5.26'da gösterildiği gibi “Join the Apple Developer Program” butonuna tıklamak gerekmektedir.



Şekil 5.26: Join The Apple Developer Program ekranı.

Bu işlemleri tamamladıktan sonra Şekil 5.27’de görülen ekran açılmaktadır. Bu ekranın sağ üst tarafında bulunan “Enroll” butonuna basılması gerekmektedir.



Şekil 5.27: Apple Developer Program satın alma adımı.

“Enroll” butonuna basıldıktan sonra Şekil 5.28’deki gibi bir ekran açılmaktadır. Şekil 5.28’de “Start Your Enrollment” butonuna basılması gerekmektedir.

DBAs, fictitious businesses, trade names, or branches.

**Program** ▼ Enroll

**Legal Binding Authority**

As the person enrolling your organization in the Apple Developer Program, you must have the legal authority to bind your organization to legal agreements. You must be the organization's owner/founder, executive team member, senior project lead, or have legal authority granted to you by a senior employee.

Start Your Enrollment

The Apple Developer Program is 99 USD per membership year or in local currency where available. For more details, see our [enrollment support page](#).

Apple Developer | Apple Developer Program > Before You Enroll

Şekil 5.28: Start Your Enrollement adımı.

“Start Your Enrollment” butonuna basıldıktan sonra Şekil 5.29’daki gibi bir ekran açılmaktadır. Bu ekrandan da hesabı ne için alındığı bilgisi sorulmaktadır. Individual, Company ve Government olarak üç seçenek sunmaktadırlar.

Developer Program Enrollment ▼

**Entity Type**

I develop apps as

Individual / Sole Proprietor / Single Person Busin✓



**Please note:** In order for your company name to be listed as the seller on the App Store, your company must be recognized as a legal entity in your country. Sole proprietors and single person companies located in countries where they are not recognized as legal entities will be enrolled and be listed for download under the enrollee's personal legal name, if selected for distribution by Apple.

Cancel Continue

Şekil 5.29: Apple Developer Program paket seçenekleri.

Apple Developer Program hesabı satın alırken kullanıcıların karşısına çıkan “Individual” ve “Company” seçenekleri arasındaki farklar aşağıdaki gibidir:

- **Individual:** Mobil uygulamanın geliştirilme esnasında, AppStore’da yayınlanmadan önce 100 adet mobil cihaza kadar test amaçlı mobil uygulama yükleme yapılmasına izin vermektedir. Eğer daha fazla test amaçlı mobil cihaz ihtiyacı varsa bu program o kullanıcılar için uygun değildir.
- **Company:** Şirket hesapları için, bireysel seçeneğine göre daha ayrıntılıdır. Ayrıca “Individual” seçeneğine göre onay süreci daha uzundur. Bir grup çalışan için, alınacak olan hesap türüdür denebilir. Daha çok bir takımın Developer Apple Program’ı kullanabilmesine olanak sağlar. Bu programda, 500 ya da daha fazla çalışanı olan bir hedef kitle öngörülmektedir. Orta ve büyük çaplı şirketler için AppStore’da yayınlanması istenmeyen mobil uygulamalarda bu hesap türü seçilerek markette mobil uygulamanın görülmemesi sağlanarak çalışanların mobil uygulamayı indirip kullanabilmelerine imkan sunmaktadır. Örneğin, büyük çaplı kurumsal bir firma olsun. Bu şirketin, çalışanları için bir mobil uygulama ürettiği varsayalım. Bu tür bir uygulamayı firma, App Store’a koymak istemeyecektir. Bu şekilde “Company” türündeki hesapla App Store’a mobil uygulama konmadan, şirket bünyesinde kullanılacak uygulamaların dağıtımı kolaylıkla sağlanabilmektedir.

Geliştirilen mobil uygulama “Individual” türünde olduğu varsayıldığından, “Individual” seçeneği seçilir ve “Continue” butonuna basılır. Daha sonra bir form ekranı açılmaktadır. Buradaki bilgiler de doldurulup “Continue” butonuna basılır. Bu işlemten sonra birkaç doğrulama adımları daha açılmaktadır. Bu işlemler de tamamlandıktan sonra “Purchase” butonuna basılır ve Şekil 5.30’deki ekran açılır.

② Ödeme ve Faturalandırma Ayrıntıları

**Fatura İrtibatı**

Unvan

Oğuz  ASLAN

Telefon Numarası (ör. XXX1234567)

Diğer Telefon (isteğe bağlı)

oguz.aslan@outlook.com

**Ödeme Yöntemi**

VISA  

Kredi Kartı Numarası  Güvenlik Kodu

Son Kullanma Tarihi Ay  Yıl

**Fatura Adresi**

Şirket Adı (isteğe bağlı)

Adres 1

Adres 2 (isteğe bağlı)

Mahalle

İl  İlçe

Posta Kodu

**Vergi Bilgileri**

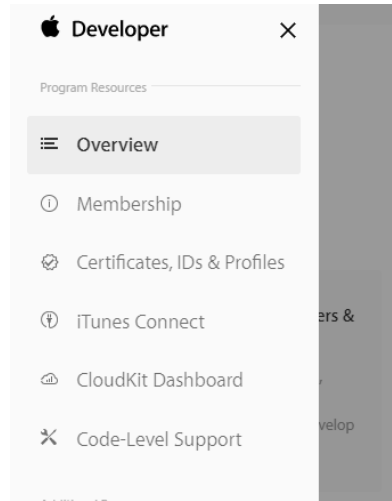
TCKN

[Hemen Sohbet Edin](#) [Devam](#)

Şekil 5.30: Apple Developer Program ödeme ekranı.

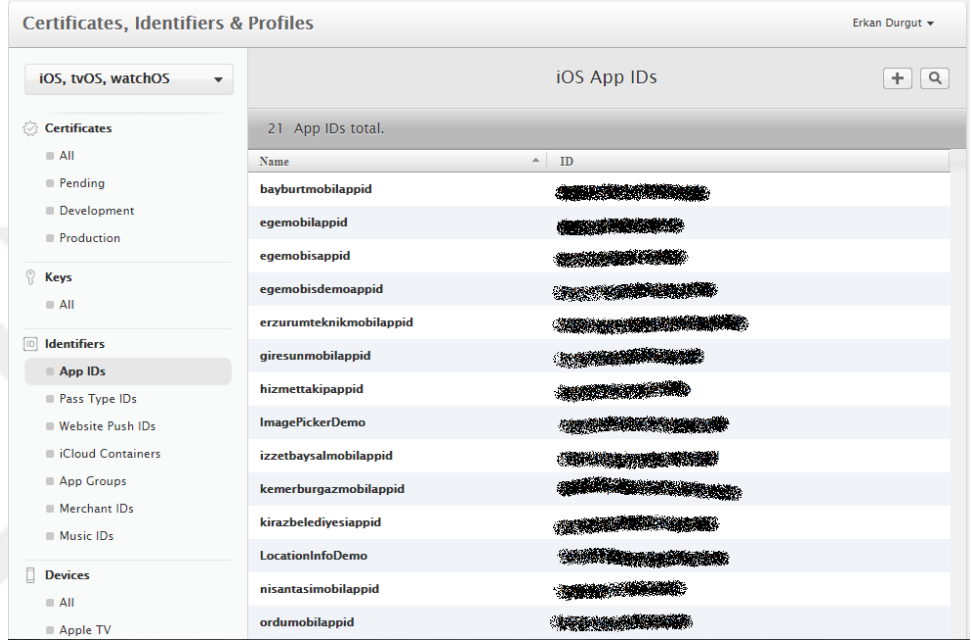
Tüm bu işlemler tamamlandıktan sonra, ödeme işlemi tamamlanmış ve Apple Developer Program hesabı satın alınmış olur.

Apple Developer Program hesabı satın alındıktan sonra, geliştirilen mobil uygulama için Developer Apple hesabı arayüzünden, mobil uygulama için bir “Identifier” oluşturulması gerekmektedir. Apple Developer Program hesabına giriş yapıldıktan sonra sol taraftaki menüden Şekil 5.31’de görüldüğü gibi “Certificates, IDs & Profiles” seçeneği seçilir.



Şekil 5.31: Apple Developer Program sertifika oluşturma menüsü.

Şekil 5.31’de belirtilen işlem tamamlandıktan sonra “Certificates, IDs & Profiles” ekranı açılmaktadır. Bu ekrandan, Şekil 5.32’de görüldüğü gibi soldaki menüden “Identifiers” başlığı altında bulunan “AppIDs” seçeneği seçilir. “AppIDs” seçeneği seçildikten sonra ilk defa Id alımı yapılıyorsa ekran boş gelecektir. Daha önce farklı mobil uygulamalar için Id değerleri oluşturulmuşsa dolu gelecektir. Yeni bir Id oluşturmak için Şekil 5.32’de görüldüğü gibi sağ üst köşede bulunan “+” butonuna basılır.



Şekil 5.32: Apple Developer Program App ID menüsü.

Açılan ekrandan geliştirilen mobil uygulamaya özel olarak verilmesi istenen “Name” alanı Şekil 5.33’te görüldüğü gibi doldurulur. “Name” kısmı doldurulduktan sonra varsayılan olarak seçilen “Explicit App ID” seçeneği olduğu gibi kalır. Bu seçeneğin altında bulunan “Bundle ID” alanına tamamen özgün olacak şekilde bir Id değeri girilir.

“Bundle ID” değeri tüm mobil uygulamalar için alınan Identifier işlemindeki parametrelerden biridir ve özgün bir değere sahip olmalıdır. Apple “Bundle ID” oluşturma ile alakalı olarak tersine domain mantığı kullanılmasını öngörmektedir. Geliştirilen mobil uygulamada gerçekleştirilen “Bundle ID” değeri Şekil 5.33’de de görüldüğü gibi “com.oguzaslan.fastpay” olacak şekilde eklenmiştir. Bu işlemler tamamlandıktan sonra “Continue” butonuna basılır.

## App ID Description

Name: FastPayAppID

You cannot use special characters such as @, &amp;, %, \*, ", "

## App ID Prefix

Value: com.oguzaslan.fastpay

## App ID Suffix

## ● Explicit App ID

If you plan to incorporate app services such as Game Center, In-App Purchase, Data Protection, and iCloud, or want a provisioning profile unique to a single app, you must register an explicit App ID for your app.

To create an explicit App ID, enter a unique string in the Bundle ID field. This string should match the Bundle ID of your app.

Bundle ID: com.oguzaslan.fastpay

We recommend using a reverse-domain name style string (i.e., com.domainname.appname). It cannot contain an asterisk (\*).

Cancel

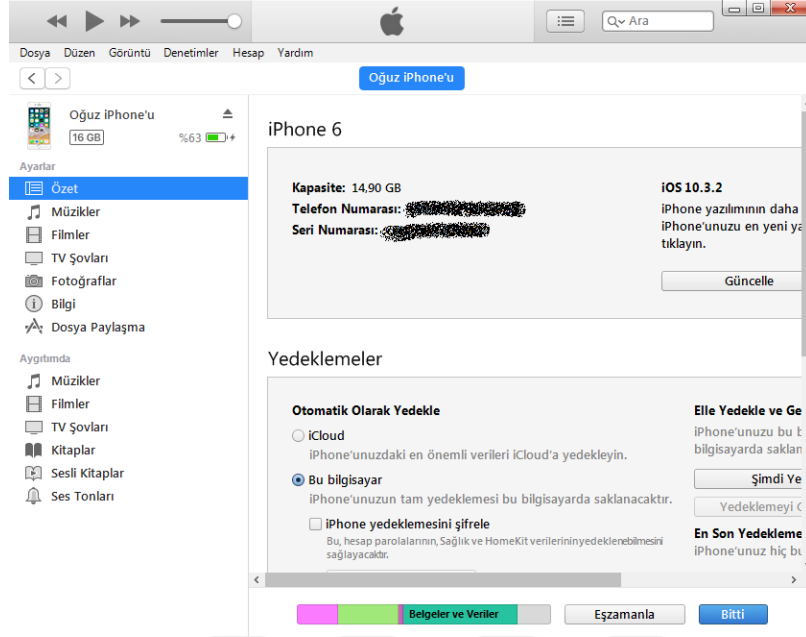
Continue

Şekil 5.33: Apple Developer Program yeni App ID oluşturma işlemi.

“Continue” butonuna basıldıktan sonra “Generate” butonuna basılır ve App ID değeri geliştirilen mobil uygulama için oluşturulmuş olur. Bu işlemleri tamamladıktan sonra, sol taraftaki menüden Devices menüsü altında bulunan All butonuna basılır.

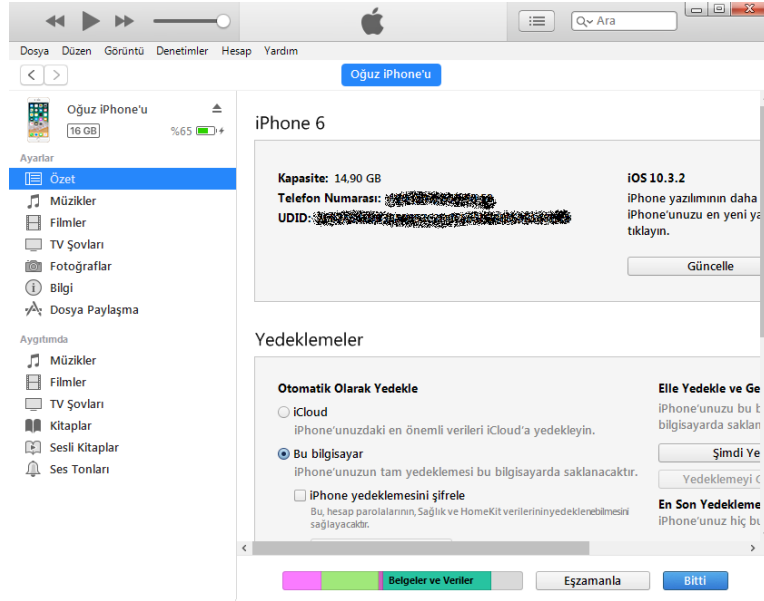
Devices menüsünün anlamı, geliştirilen mobil uygulamanın Apple Developer Program kapsamında “Development” olarak belirlenen mobil cihaza yüklenmesini sağlamaktır. “Akıllı mobil ödeme sistemi” için geliştirilen mobil uygulamanın App Store’a yayınlanmadan mobil cihaza yüklenmesini sağlamak için, mobil uygulamanın yükleneceği cihazın bu kısımda tanımlanması gerekmektedir.

Apple Developer Program içerisine bir cihaz tanımlayabilmek için, belirlenen IOS işletim sistemine sahip cihazın UDID’sini elde etmek gerekmektedir. IOS işletim sistemine sahip bir cihazın UDID değerini elde edebilmek için öncelikle mobil cihazın iTunes ile bilgisayar bağlamak gerekmektedir. Mobil cihaz USB ile bilgisayara bağlanır ve Şekil 5.34’te görüldüğü gibi iTunes uygulamasında bilgisayara bağlanan mobil cihaz görüntülenir.



Şekil 5.34: IOS İşletim sistemine sahip mobil cihazın iTunes'a bağlanması.

iTunes üzerinden IOS işletim sistemine sahip mobil cihazın UDID değerini öğrenebilmek için yukarıdaki şekilde görülen “Seri Numarası” yazan yere birkaç kere tıklamak yeterlidir. Ekrandan UDID değeri görünene kadar bu işlem tekrar edilir. Sonuç olarak Şekil 5.35’te görülen 40 karakterden oluşan UDID değeri elde edilmiş olur.



Şekil 5.35: IOS işletim sistemine sahip mobil cihazdan UDID eldesi.

UDID değeri elde edildikten sonra Apple Developer Program sayfasına geri dönülür ve “Devices” menüsü altında bulunan “All” seçeneği seçilir. Apple Developer Program’a yeni bir cihaz tanımlamak için sağ üst köşede bulunan “+” butonuna basılır. Bu işlemten sonra, daha önceden tanımlanması istenen mobil cihaz için elde edilen UDID değeri tanımlanacak bir isim ile birlikte buraya yazılarak Şekil 5.36’da görüldüğü gibi “Continue” butonuna basılır.

**Registering a New Device or Multiple Devices**

**Pre-Release Software Reminder**  
You may only share Apple pre-release software with employees, contractors, and members of your organization who are registered as Apple developers and have a demonstrable need to know or use Apple software to develop and test applications on your behalf.

Unauthorized distribution of Apple confidential information (including pre-release software) is prohibited and may result in the termination of your Apple Developer Program. It may also subject you to civil and criminal liability.

☒ **Register Device**  
Name your device and enter its Unique Device Identifier (UDID).

Name:

UDID:

☐ **Register Multiple Devices**  
Upload a file containing the devices you wish to register. Please note that a maximum of 100 devices can be included in your file and it may take a few minutes to process.  
[Download sample files](#)

Şekil 5.36: Apple Developer Program’a cihaz tanımlanması.

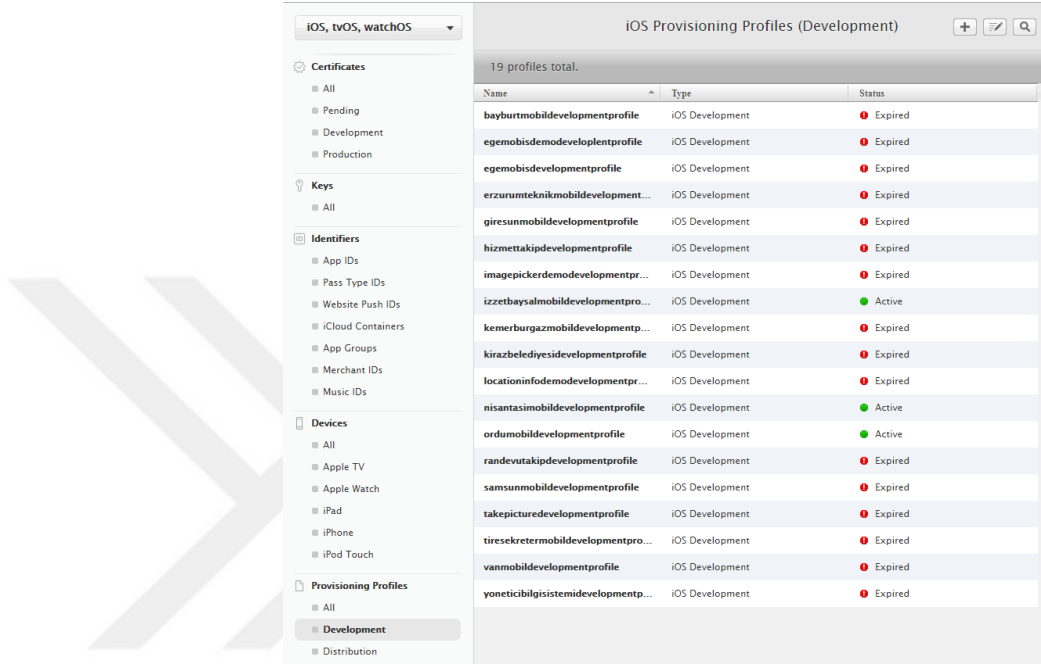
Mobil uygulamanın yükleneceği IOS işletim sistemine ait cihaz Apple Developer Program hesabına tanımlandıktan sonra Swift ile mobil uygulama geliştirme tool’u olan XCode’da kullanılmak üzere bir Development Provisioning Profile oluşturulması gerekmektedir.

Provision Profile anlamı, Apple Developer Program hesabına tanımlanan bir App ID’yi ve tanımlanan mobil cihazları eşleştirerek Development veya Distribution sertifikası oluşturma işlemidir. Eğer geliştirilecek mobil uygulama ilk etapta App Store marketine yüklenmeyecekse Development Provisionig Profile oluşturmak yeterlidir.

“Akıllı mobil ödeme sistemi” kapsamında geliştirilen mobil uygulamanın IOS işletim sistemine sahip mobil bir cihaza yüklenmesi aşamalarından biri olan Provisioning Profile oluşturma işlemi için, Apple Developer Program hesabına

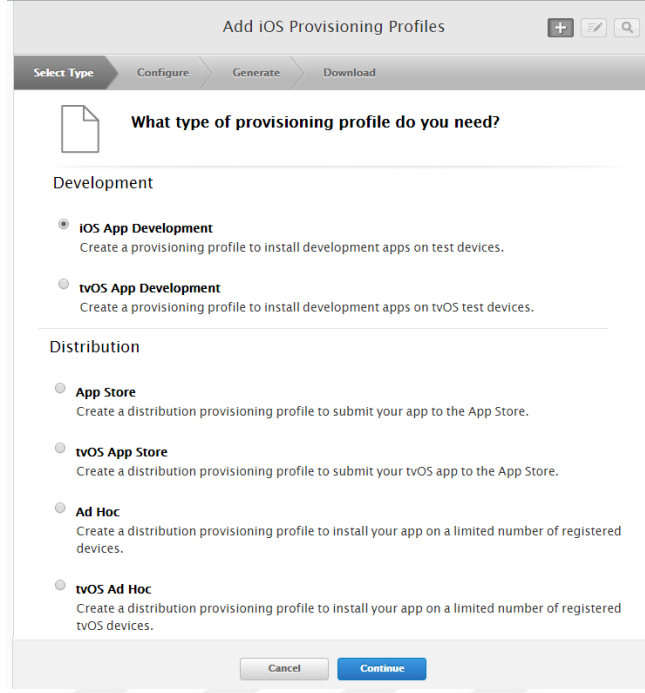
giriş yaptıktan sonra, sol tarafta “Provisioning Profiles” başlığı altında bulunan “Development” seçeneği seçilir.

Şekil 5.37’de görüldüğü gibi iOS Provisioning Profile bilgileri Development için listelenir. Yeni bir Development Provisioning Profile eklemek için sağ üst tarafta bulunan “+” butonuna basılır.



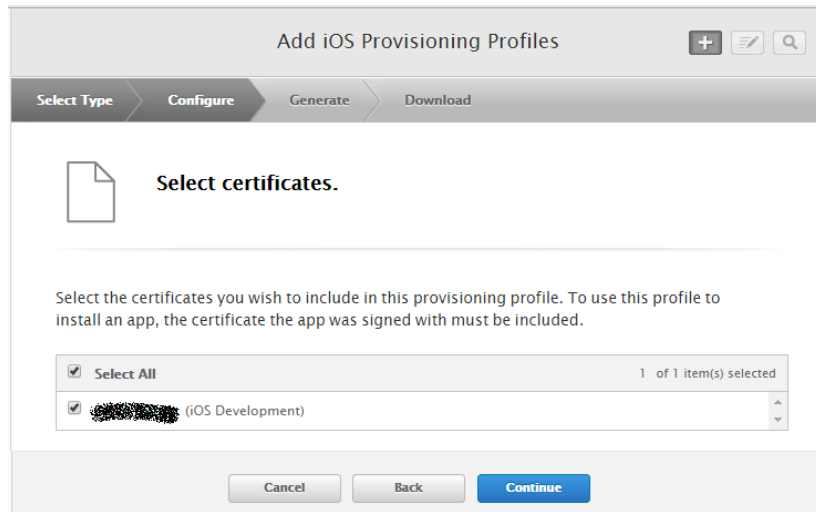
Şekil 5.37: Provisioning Profile ekranı.

Development için iOS Provisioning Profile ekranı sağ üst köşesindeki “+” butonuna basıldıktan sonra Şekil 5.38’deki ekran açılır. Bu ekranda, geliştirme aşamasında olduğu için ve App Store markete şimdilik bir yükleme yapılmayacağı için “Development” başlığı altında bulunan “iOS App Development” seçeneği seçilir. Buradaki temel amaç mobil cihaza yükleme yapılacağı için “Distribution” seçeneği altından herhangi bir işlem uygulanmaz. Bu işlem tamamlandıktan sonra “Continue” butonuna basılır.



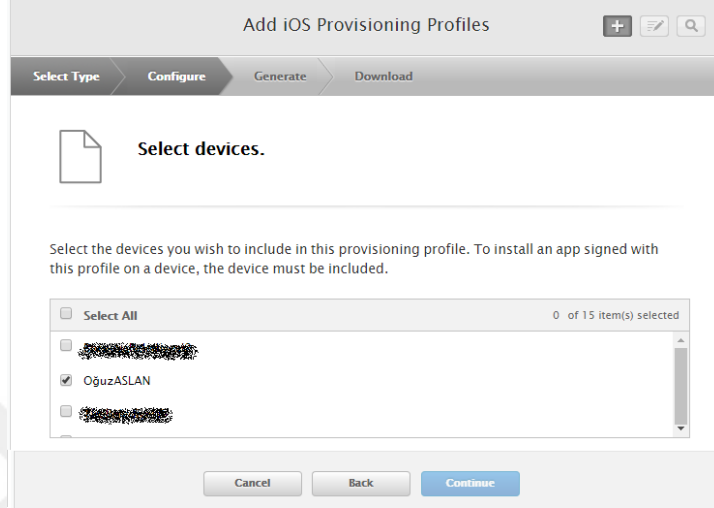
Şekil 5.38: Provisioning Profile seçenekleri.

Bu işlem tamamlandıktan sonra “Configure” adımına geçilir. “Configure” adımı, daha önceden tanımlaması yapılan App ID değeri buradan seçilir. Geliştirilecek mobil uygulama için tanımlanan App ID ile Provisioning Profile ilişkilendirmesi işlemi buradan yapılmaktadır. Daha önceden tanımlanan App ID değeri Şekil 5.39’da görüldüğü gibi seçilir ve “Continue” butonuna basılır. Daha sonra Development için tanımlanan “Certificate” bu kısımdan seçilir. Development sertifikası seçildikten sonra “Next” butonuna basılır. Bu işlem Şekil 5.39’da gösterilmiştir.



Şekil 5.39: Provisioning Profile konfigürasyonu.

Daha önceden, mobil uygulamanın yükleneceği cihazın tanımı Apple Developer Program hesabına tanımlanmıştı. Provisioning Profile oluşturulurken tanımlanan cihazın da buradan eşleştirilmesi gerekmektedir. Bu işlemi yapabilmek için açılan ekranda tanımlanan cihazlardan, yüklenmesine izin verilecek cihaz seçilir ve Şekil 5.40’de görüldüğü gibi “Continue” butonuna basılır.



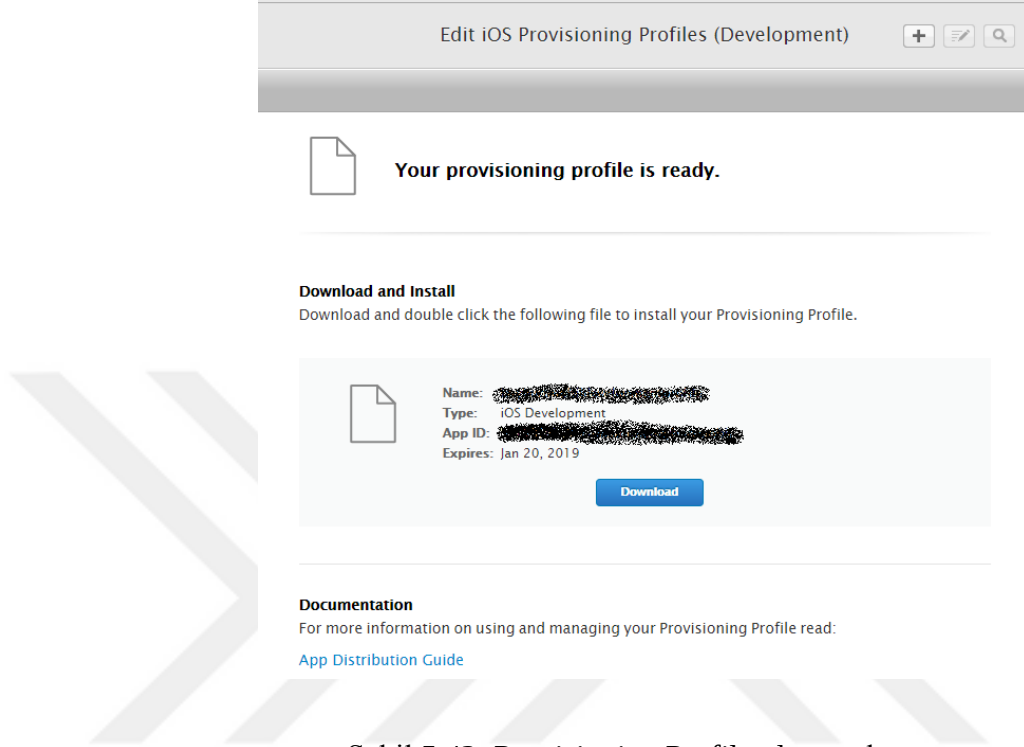
Şekil 5.40: Provisioning Profile cihaz eşleştirilmesi.

Şekil 5.40’deki işlem tamamlandıktan sonra “Generate” adımına geçilir ve aşağıdaki ekran açılır. Açılan ekranda tanımlanacak olan Development Provisioning Profile için bir isim verilmesi istenmektedir. Geliştirilen mobil uygulama için Developement Provisioning Profile tanımlamasına uygun olduğu düşünülen “fastpayDevelopmentProfile” ismi verilir ve “Continue” butonuna basılır. Bu işlem Şekil 5.41’de gösterilmiştir.



Şekil 5.41: Provisioning Profile isimlendirme işlemi.

Tüm bu işlemler tamamlandıktan sonra Development Provisioning Profile tanımlanmış olur. Oluşturulan Development Provisioning Profile sertifikası indirilmek istenirse Şekil 5.42’de olduğu gibi “Download” butonuna basılarak işlem gerçekleştirilebilir.

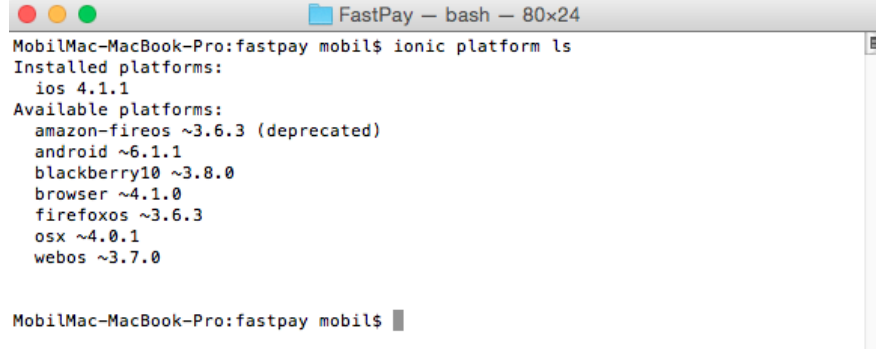


Şekil 5.42: Provisioning Profile oluşturulması.

Tüm bahsedilen sertifikalar oluşturulduktan sonra artık, “akıllı mobil ödeme sistemi” kapsamında geliştirilen mobil uygulama, IOS işletim sistemine sahip cihaza yüklenebilir demektir.

Apple, kendi işletim sistemine sahip mobil cihazlara kurulumda bir takım kısıtlamalar getirmektedir. Bu kısıtlamalardan bir tanesi de Mac işletim sistemine sahip bir makine olmadan, mobil cihaza kurulumun gerçekleştirilememesidir. Bu bağlamda, geliştirilen mobil uygulamanın öncelikle yukarıda tanımlı sertifikalarda bir uygulama hale getirilerek paketlenmesi gerekmektedir. Bu işlem için, Mac işletim sistemine kurulan XCode development IDE’si kullanılmıştır.

Hybrid mobil uygulama projesine IOS platform eklenmesi işlemi gerçekleştirilir. Bu işlemi gerçekleştirmek için proje dizini MacBook bilgisayarda bulunan terminal ile açılır. Daha sonrasında, hybrid projeye IOS platform eklemek için “ionic platform add ios” komutu çalıştırılır. Yüklenen platformlar listesi Şekil 5.43’te görüldüğü gibi “ionic platform ls” komutu ile gözlemlenebilir.



```

MobilMac-MacBook-Pro:fastpay mobil$ ionic platform ls
Installed platforms:
  ios 4.1.1
Available platforms:
  amazon-fireos ~3.6.3 (deprecated)
  android ~6.1.1
  blackberry10 ~3.8.0
  browser ~4.1.0
  firefoxos ~3.6.3
  osx ~4.0.1
  webos ~3.7.0

MobilMac-MacBook-Pro:fastpay mobil$

```

Şekil 5.43: IONIC projesi platform listesi.

Geliştirilen mobil uygulama projesi IONIC kontrol komutları kullanılarak Şekil 5.44’te görüldüğü gibi derlenir. Hybrid mobil uygulama projesinde, uygulamayı derleyebilmek için IONIC kontrol komutlarından “ionic build ios” kullanılır. Komut çalıştırıldıktan sonra başarılı bir derlemenin gerçekleştirdiğini anlayabilmek için terminalde “Build Succeeded” yazısının görüntülenmesi gerekmektedir.



```

lator.platform/Developer/usr/bin:/Applications/Xcode.app/Contents/Developer/usr/
bin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/share/dotnet"
/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolcha
in/usr/bin/dsymutil /Users/mobil/desktop/fastpay/fastpay/platforms/ios/build/emu
lator/FastPay.app/FastPay -o /Users/mobil/desktop/fastpay/fastpay/platforms/ios/
build/emulator/FastPay.app.dSYM

Touch /Users/mobil/desktop/fastpay/fastpay/platforms/ios/build/emulator/FastPay.
app
cd /Users/mobil/Desktop/FastPay/FastPay/platforms/ios
export PATH="/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimu
lator.platform/Developer/usr/bin:/Applications/Xcode.app/Contents/Developer/usr/
bin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/share/dotnet"
/usr/bin/touch -c /Users/mobil/desktop/fastpay/fastpay/platforms/ios/build/e
mulator/FastPay.app

** BUILD SUCCEEDED **

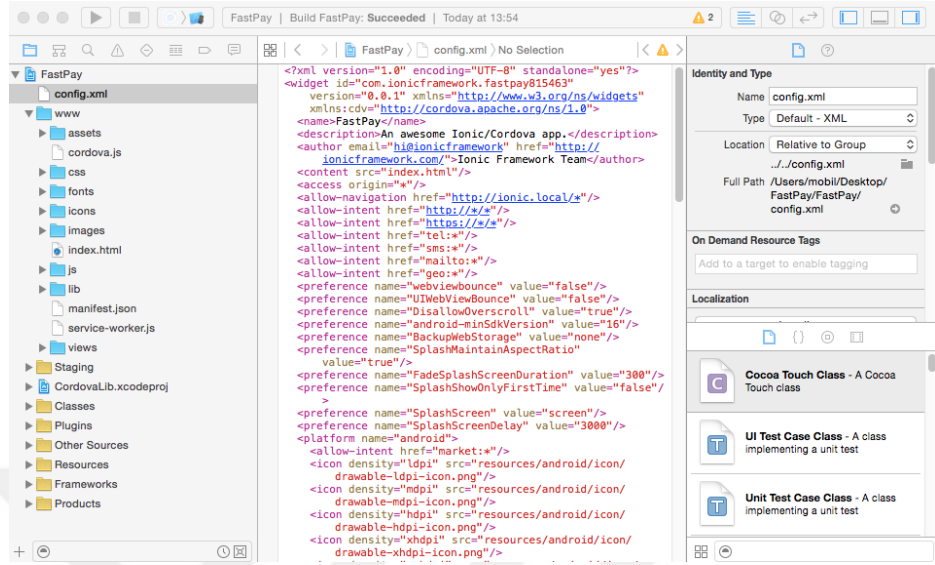
MobilMac-MacBook-Pro:fastpay mobil$

```

Şekil 5.44: IONIC projesi IOS platform Build işlemi.

Tüm bu işlemler tamamlandıktan sonra hybrid mobil uygulama projesi dosyaları içerisinde native IOS projesi oluşturulmuş olur. Proje ana dizininde bulunan “platforms” klasörü içinde gelinir. Burada hybrid mobil uygulama için eklenen platformlar gözlenebilir. Buradan “ios” klasörü açılır ve native IOS projesi elde edilir. Geliştirilen mobil uygulama için Şekil 5.45’de görülen “FastPay.xcodeproj” dosyası XCode IOS native yazılım geliştirme IDE’si ile

açılarak proje gözlemlenir. Proje açıldıktan sonra Şekil 5.45’de görüldüğü gibi olması gerekmektedir.

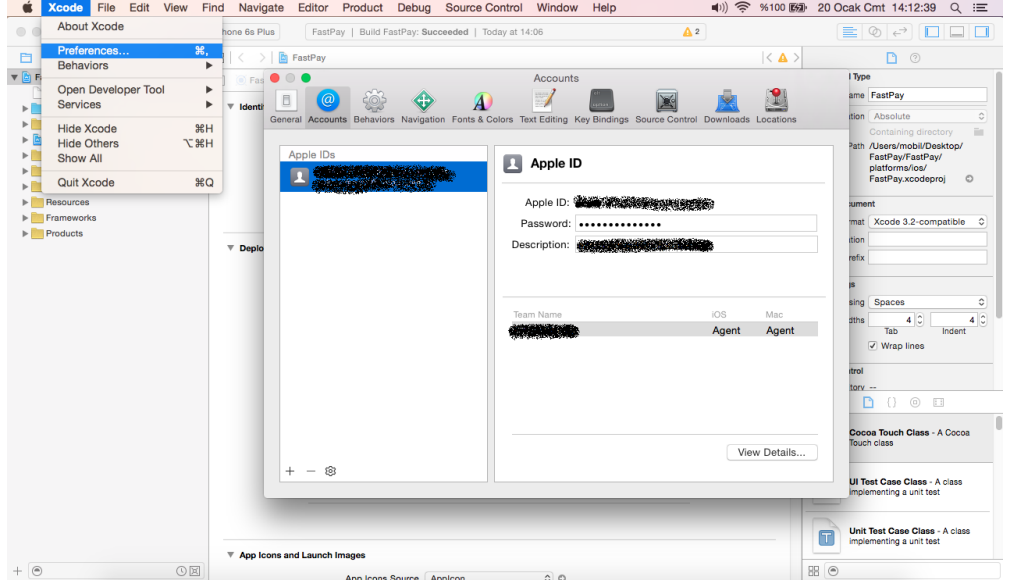


Şekil 5.45: Mobil uygulamanın XCode ile açılması.

XCode projesi açıldıktan sonra da üst menülerde bulunan “Product” menüsü içerisinde “Build” seçildiğinde de başarılı bir derleme gerçekleştirmek için “Build Succeeded” yazısının görülmesi gerekmektedir.

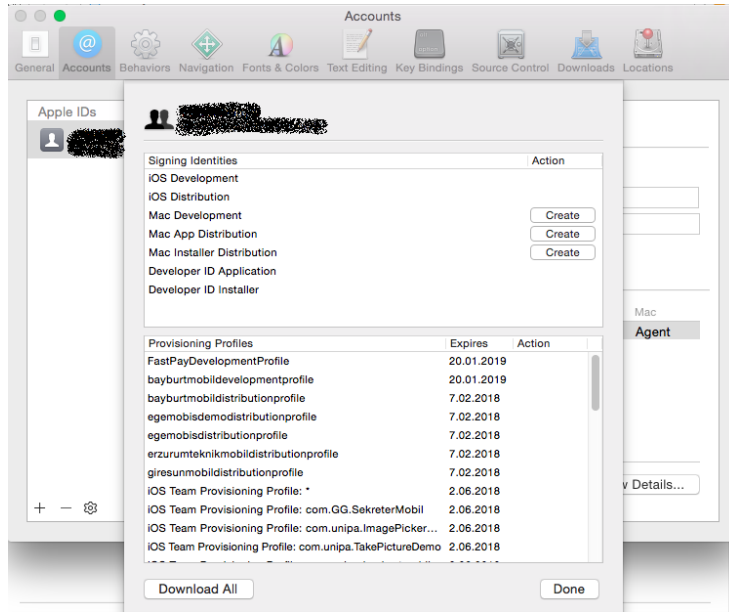
Tüm sertifika işlemlerinin proje ile eşlenebilmesi için öncelikle, Apple Developer Program hesabı ile XCode senkronizasyonu sağlamak gerekmektedir. Bu işlemi yapabilmek için XCode’da üst tarafta bulunan “XCode” menüsünden “Preferences” seçeneğinin seçilmesi gerekmektedir.

Şekil 5.46’da görüldüğü gibi öncelikle daha önceden satın alma işlemi aşamaları anlatılan Apple ID için, Apple Developer Program hesabı ile login olmak gerekmektedir.



Şekil 5.46: XCode ile Apple Developer Program’a giriş.

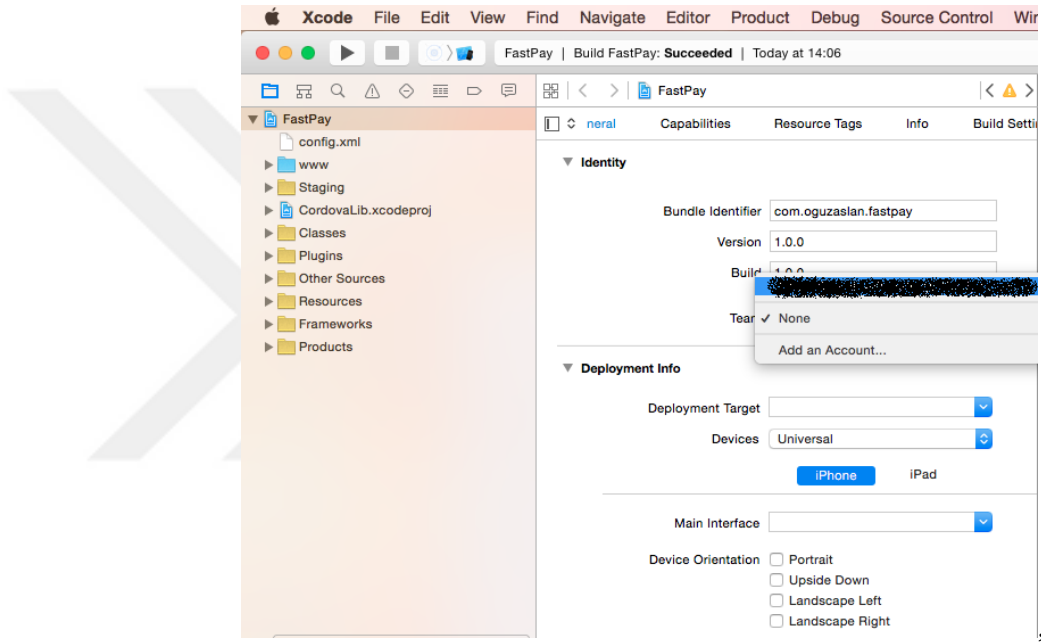
XCode üzerinden Apple Developer Program hesabına login olduktan sonra, Şekil 4.86’da görüldüğü gibi pencerenin sağ alt tarafında bulunan “View Details” butonuna basılması gerekmektedir. Bu butona basıldıktan sonra Şekil 5.47’de görüldüğü gibi bir pencere açılmaktadır. Bu pencerede Apple Developer Program web sayfasında tanımlanan sertifikaların gözükmesi gerekmektedir. Senkronizasyon için, pencerenin sol alt tarafında bulunan “Download All” butonuna basılması gerekmektedir.



Şekil 5.47: XCode ile Apple Developer Program sertifikalarının yüklenmesi.

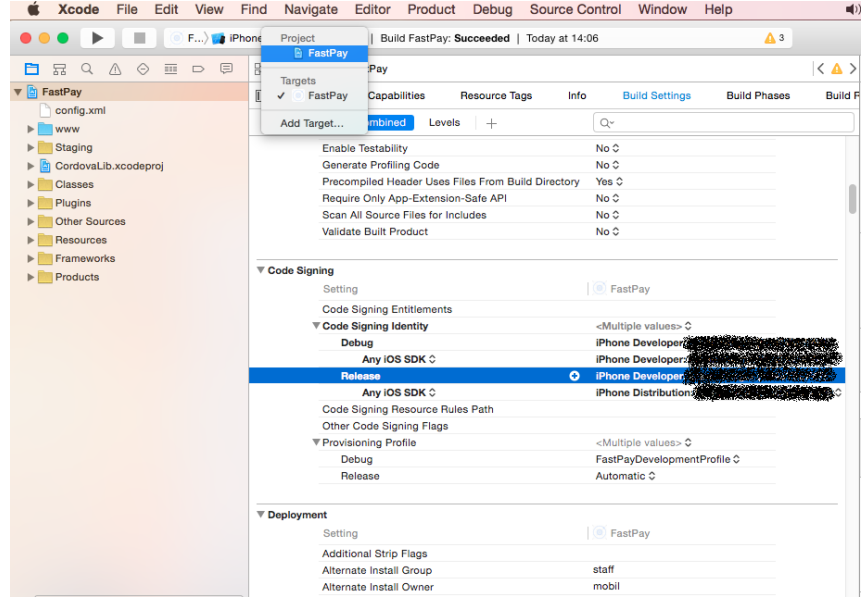
Senkronizasyon sağlandıktan sonra bu pencereler kapatılır. Geliştirilen mobil uygulamanın, IOS işletim sistemine sahip cihaza yüklenebilmesi için, proje ile daha önceden Developer Apple Program hesabı üzerinden yapılan App ID ve Provisioning Profile sertifikalarının proje ile eşleştirilmesi gerekmektedir.

Bu işlemi yapabilmek için, Şekil 5.48’de görüldüğü gibi projenin ana dizini Project Explorer’den seçilmelidir. Projenin ana dizini seçildikten sonra orta kısımda çıkan “General” sekmesi ayarları yapılandırılmalıdır. Şekil 5.48’de de görüldüğü gibi “General” sekmesi altında bulunan “Team” seçeneği, Apple Developer Program hesabı ile eşleştirilir.



Şekil 5.48: XCode projesinde Apple Developer Program Team ayarı.

“Team” eşleştirmesinden ardından “Build Settings” sekmesinde bulunan ayarların sağlanması gerekmektedir. Bu işlemlerin yapılabilmesi için Şekil 5.49’da görüldüğü gibi, “Build Settings” sekmesi altında bulunan “Code Signing” başlığı altındaki “Code Signing Identity” ve “Provisioning Profile” ayarlarının Apple Developer Program’da tanımlanan sertifikalarla yapılandırılmış olması gerekmektedir. Aynı “Build Settings” ayarı hem “Project” hem de “Targets” için yapılmış olması gerekmektedir.



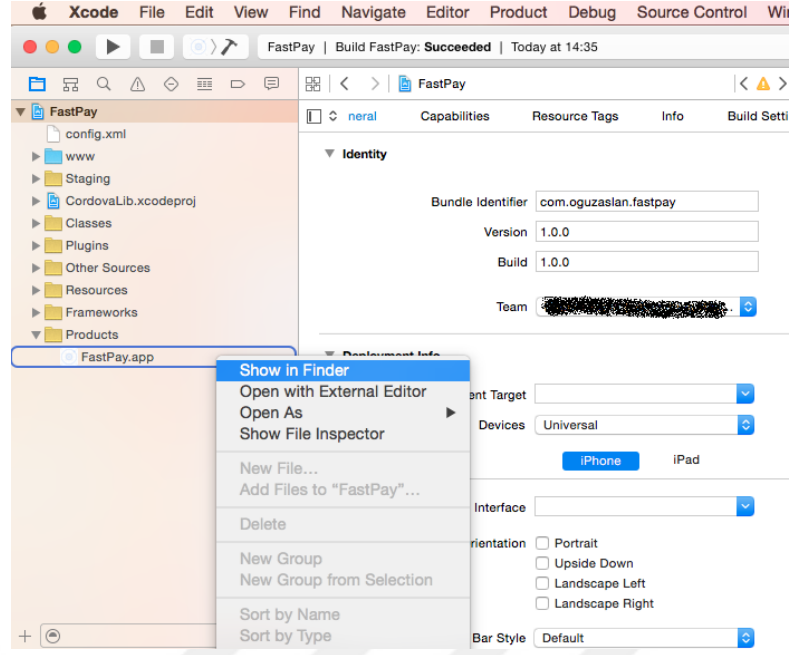
Şekil 5.49: XCode Projesinde Apple Developer Program Build Settings ayarları.

Apple Developer Program sertifikaları ile mobil uygulama projesi eşleştirildikten sonra, mobil uygulamaya native olarak XCode üzerinden paketlenmeye hazır hale getirilmiştir.

IOS işletim sistemine sahip mobil cihaza kurulumun gerçekleştirilebilmesi için, XCode üzerinden proje “Generic IOS Device” platformu için “build” edilir.

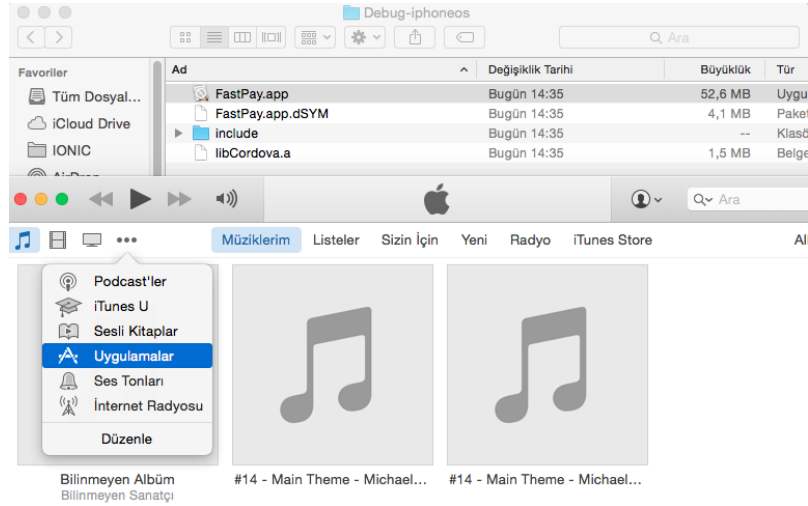
Bu işlem tamamlandıktan sonra Şekil 5.50’da görüldüğü gibi Project Explorer’da bulunan proje dosyalarından “Products” klasörü altında paketlenmiş mobil uygulama gözükmemektedir.

İlgili dosyaya sağ tıklayıp “Show in Finder” seçeneği seçilir ve ilgili mobil uygulama dosyası elde edilmiş olur.



Şekil 5.50: XCode ile mobil uygulamanın paketlenmesi.

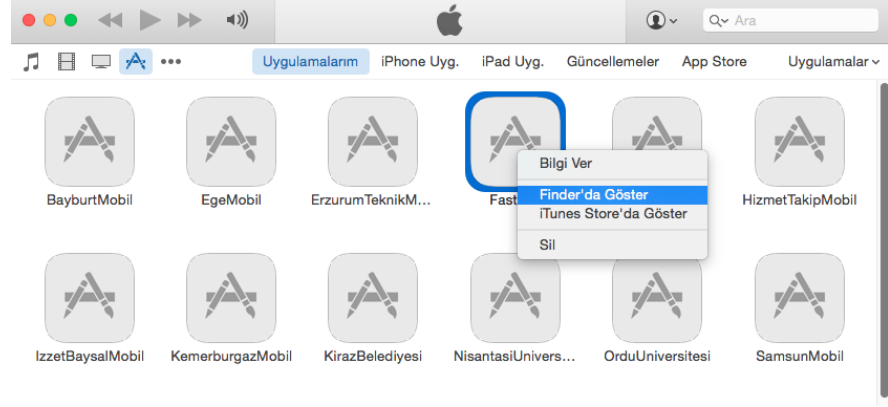
Mobil uygulama dosyası elde edildikten sonra “.app” uzantılı dosyanın elde edilmesi gerekmektedir. Bu işlemi gerçekleştirebilmek için iTunes açılır. Şekil 5.51’de gösterildiği gibi “Application” sekmesi açılır ve uygulamaların olduğu ekrana gelinir.



Şekil 5.51: iTunes ile uygulamaların görüntülenmesi.

XCode’den elde edilen mobil uygulama projesi iTunes’un bu ekranına sürüklenip bırakılır. Mobil uygulama iTunes uygulamalar içerisine yüklendikten sonra Şekil 5.52’de gösterildiği gibi, yüklenen mobil uygulamaya sağ tıklanarak

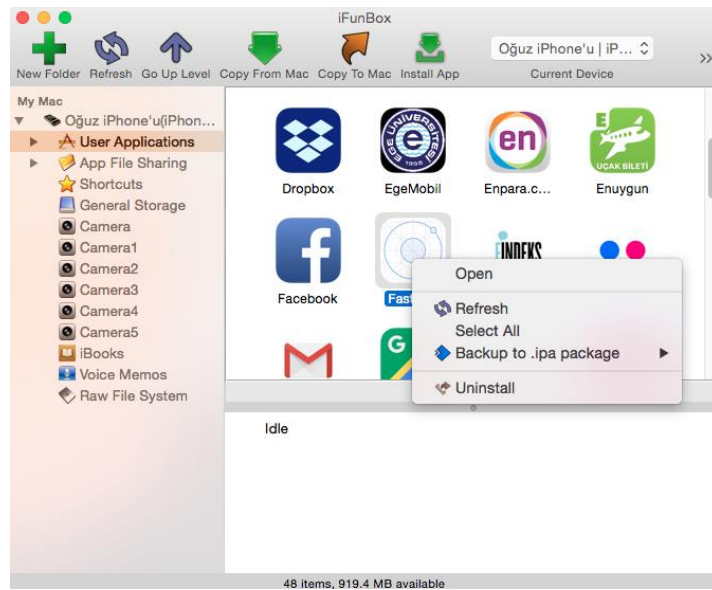
“Finder’da Göster” seçeneği seçilir. Bu işlem tamamlanınca bir pencere açılır ve IOS işletim sistemine sahip mobil cihaza yüklenecek “.ipa” uzantılı dosya elde edilmiş olur.



Şekil 5.52: Mobil uygulama dosyasının elde edilmesi.

Tüm bu adımlar tamamlandıktan sonra “.ipa” uzantılı uygulama elde edildikten sonra iFunBox uygulaması indirilir ve bilgisayara kurulur. Mobil uygulama yüklenmesi istenen IOS işletim sistemine sahip IOS cihaz, USB ile bilgisayara bağlanır.

iFunBox açılarak Şekil 5.53’te görüldüğü gibi pencerenin üst tarafında bulunan “Install App” butonuna basılarak “.ipa” uzantılı dosya seçilir ve ardından “Open” butonuna basılır. Bu sayede mobil uygulama, IOS işletim sistemine sahip mobil cihaza yüklenmiş olur.

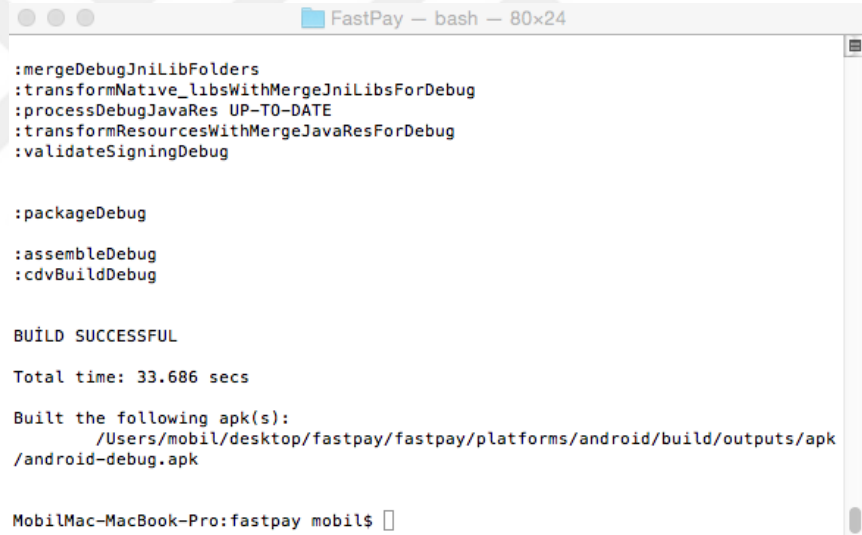


Şekil 5.53: Mobil uygulamanın cihaza kurulumu.

### 5.3.2.2 Android işletim sistemine kurulumu

Hybrid mobil uygulama projesine Android platform eklenmesi işlemi gerçekleştirilir. Bu işlemi gerçekleştirmek için proje dizini MacBook bilgisayarda bulunan terminal ile açılır. Daha sonrasında hybrid projeye Android platform eklemek için “ionic platform add android” komutu çalıştırılır. Yüklenen platformlar listesi daha önceden bahsedildiği şekilde “ionic platform ls” komutu ile gözlemlenebilir.

Geliştirilen mobil uygulama projesi IONIC kontrol komutları kullanılarak Şekil 5.54’de görüldüğü gibi Android işletim sistemine sahip cihaz için derlenir. Hybrid mobil uygulama projesinde, uygulamayı derleyebilmek için IONIC kontrol komutlarından “ionic build android” kullanılır. Komut çalıştırdıktan sonra başarılı bir derlemenin gerçekleştirdiğini anlayabilmek için terminalde “Build Succeeded” yazısının görüntülenmesi gerekmektedir.



```
FastPay - bash - 80x24

:mergeDebugJniLibFolders
:transformNative_libsWithMergeJniLibsForDebug
:processDebugJavaRes UP-TO-DATE
:transformResourcesWithMergeJavaResForDebug
:validateSigningDebug

:packageDebug

:assembleDebug
:cdvBuildDebug

BUILD SUCCESSFUL

Total time: 33.686 secs

Built the following apk(s):
/Users/mobil/desktop/fastpay/fastpay/platforms/android/build/outputs/apk
/android-debug.apk

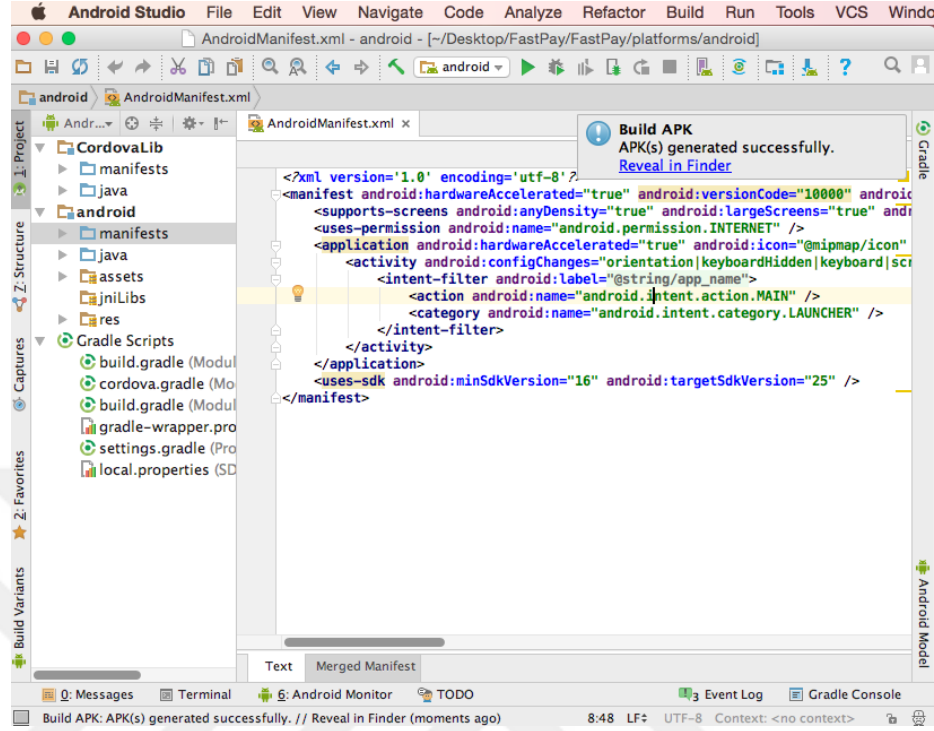
MobilMac-MacBook-Pro:fastpay mobil$
```

Şekil 5.54: IONIC projesi Android Platform Build işlemi.

Tüm bu işlemler tamamlandıktan sonra hybrid mobil uygulama projesi dosyaları içerisinde native Android projesi oluşturulmuş olur. Proje ana dizininde bulunan “platforms” klasörü içinde gelinir.

Hybrid mobil uygulama için eklenen platformlar gözlenebilir. Buradan “android” klasörü açılır ve native Android projesi elde edilir. Geliştirilen mobil uygulama Android projesi Şekil 5.55’te görülen Android Studio native yazılım

geliştirme IDE'si ile açılarak proje gözlemlenir. Proje açıldıktan sonra Şekil 5.55'te görüldüğü gibi olması gerekmektedir.



Şekil 5.55: Mobil uygulamanın Android Studio ile açılması.

Android projesi açıldıktan sonra da üst menülerde bulunan “Build” menüsü içerisinde “Build APK” seçildiğinde de başarılı bir derleme gerçekleştirmek için “APK(s) Generated Successfully” yazısının görülmesi gerekmektedir.

Bu işlem tamamlandıktan sonra “.apk” uzantılı mobil uygulama dosyasına, yukarıdaki şekilde görüldüğü gibi sağ üst tarafta bulunan “Reveal in Finder” seçeneği seçilerek erişilebilir.

Daha sonra da elde edilen bu dosya Android işletim sistemine sahip mobil cihaza atılarak kurulumu kolaylıkla gerçekleştirilebilir.

## 5.4 Radio Frequency Identification (RFID)

Bu bölümde, Radio Frequency Identification (RFID) Reader ve RFID Label özelliklerinden ve ayrıca ülkemizde RFID Reader kullanım alanlarından bahsedilecektir.

### 5.4.1 RFID Reader özellikleri

“Akıllı mobil ödeme sistemi” kapsamında geliştirilen projede kullanılan RFID Reader özellikleri, çalışma frekans aralıkları, okuma ve yazma mesafeleri, operasyon modları ve haberleşme port’ları gibi bilgiler Çizelge 5.1’de verilmiştir.

Çizelge 5.1: Radio Frequency Identification (RFID) Reader özellikleri.

| Özellik      | Açıklama                                                                                                           |
|--------------|--------------------------------------------------------------------------------------------------------------------|
| Marka/model  | YongKaiDa Integrative long range 5m card reader 8dbi Antenna RS232/RS485/Wiegand ISO 18000-6C (EPCGEN2) UHF Reader |
| Frekans      | 920Mhz~925Mhz                                                                                                      |
| Anten        | 8dbi Circular Polarization Antenna                                                                                 |
| Okuma        | Max. 6m                                                                                                            |
| Yazma        | Max. 1m                                                                                                            |
| Operasyon    | Master-slave mode, timer mode                                                                                      |
| Haberleşme   | RS-232                                                                                                             |
| Boyut        | 227MM*227MM*60MM, 2.2 Kg                                                                                           |
| Güç adaptörü | AC 110~220V/DC +12V                                                                                                |

### 5.4.2 RFID Label özellikleri

“Akıllı mobil ödeme sistemi” kapsamında geliştirilen projede kullanılan RFID Label özellikleri, Radio Frequency Identification (RFID) Reader’ın okuma aralıklarına, frekans özelliklerine uygun olarak seçilmiştir. Ayrıca kullanılan RFID Label’lar etiket şeklinde olup, mağaza içerisinde satılması planlanan ürünlerin etiketlerine yapıştırılabilir özelliktedir.

### 5.4.3 Ülkemizde RFID Reader kullanım alanları

Teknolojinin giderek gelişmesiyle birlikte ülkemizde RFID kullanımı yoğun bir şekilde artmaktadır. Bu bağlamda, akıllı park uygulamaları, elektronik olarak tanınan plaka işlemleri ile gerçekleştirilen akıllı trafik yönetimi, hasta takip sistemleri, Hızlı Geçiş Sistemleri (HGS), depo yönetimi ve depolarda ürün stok

takip sistemleri gibi birçok projede kullanılan Radio Frequency Identification (RFID) Reader'ın ülkemizdeki kullanım alanlarının başında gelmektedir.

### **5.5 Quick Response Code (QRCode)**

Bu bölümde, QrCode kavramı, QRCode Generator ve “akıllı mobil ödeme sistemi” kapsamında geliştirilen mobil uygulamada QRCode'un nasıl kullanıldığı açıklanmıştır.

#### **5.5.1 QRCode Generator**

QRCode, Quich Response Code'un kısaltılmışıdır. İlk önce otomotiv endüstrisi için Japonya'da tasarlanmış matris barkodun veya iki boyutlu barkodun markasıdır. Barkod, bağlı olduğu ürün hakkında bilgi içeren, makine tarafından okunabilen bir optik etikettir (Narayanan, A., 2012).

QRCode, verileri etkili bir şekilde depolamak için dört standart kodlama modunu (sayısal, alfanümerik, bayt / ikili ve kanji) kullanır. Ekstra uzantı metotlar da kullanılabilir. Bunun için bir takım teknolojiler sunulmaktadır. “Akıllı mobil ödeme sistemi” kapsamında geliştirilen mobil uygulama içerisinde yapılan ödemelere ilişkin QRCode teknolojisi kullanılması planlanmıştır.

Geliştirilen web servis içerisinde müşterilerin sepetlerinde bulunan ürünleri ödemeleri aşamasında QR kod üretimi söz konusu olmaktadır. Üretilen QRCode içerisine müşterinin ödediği ürünler için unique bir Identifier bilgisi saklanması planlanmıştır. Bu sayede mağazanın çıkış kapısından geçiş yapabilmeleri öngörülmüştür.

#### **5.5.2 Ülkemizde QRCode kullanım alanları**

Son yıllarda ülkemizde QRCode kullanımı gittikçe artmıştır. Gelecekte de QRCode kullanımının farklı teknolojilerde ve yöntemlerde yoğun bir şekilde kullanılacağı düşünülmektedir.

Ülkemizdeki kullanım alanları yönünden Findeks gibi Kredi Kayıt Bürosu, sanayi, bilimsel araştırma, gıda, tarım, özellikle bilişim, ulaşım ve haberleşme gibi birçok sektörde kullanılmaktadır.

## 6. SONUÇ

Nesnelerin İnterneti (IoT) gelişimine katkı sağlayacağı düşünülen bu çalışma ile internet ortamına aktarılan nesnelerin sayısının artış göstermesi ve insanların yeni teknolojilerle yaşamlarının kolaylaşacağı düşüncesi öngörülmektedir.

IoT'nin gelişimine katkı sağlayan birçok araştırma yapılmaktadır. IPv6'ya geçişle birlikte internete çok daha fazla nesnenin bağlanabileceği düşünülmektedir. Bu sayede IoT kavramı internet ağ yapısının incelenmesinde ve yeni internet ağ tasarımları oluşturmada etkin rol oynayacaktır.

Bu çalışma, IoT kapsamında RFID etiketleri ve güncel mobil uygulama geliştirme teknolojileri kullanılarak günlük yaşantıda son kullanıcıların yaptıkları alışveriş işlemlerine kolaylıklar sunmaktadır.

Bu yöntem kullanılarak son kullanıcıdan toplanan veriler üzerinden Fraud Detection (Sahtecilik Tespiti) işlemleri gerçekleştirilebilir. Bu sayede son kullanıcıların satın aldığı ürün eğilimlerine göre bir takım sahtecilik tespiti işlemleri uygulanabilir (Smart Card Alliance Payments Council, 2014). Hatta şahıs, ürün, lokasyon ve fiyat bazlı potansiyel risk algoritmaları uygulanabilir.

“Akıllı mobil ödeme sistemi” sayesinde üretilen veriler üzerinden Machine Learning kullanılarak kullanıcıların alışveriş alışkanlıkları üzerinden son kullanıcılara kampanyalar veya alışveriş önerileri sunmak mümkündür.

Müşterilerin daha kolay ve hızlı alışveriş yapabilmelerine olanak sağlayan bu yöntem sayesinde müşteriler ürün satın almaya teşvik edilmektedir. Bu yöntem, sıra beklemeden kolaylıkla alışveriş yapılabilmesini düşünülmektedir. Bu sayede sektörel olarak bu yöntemi kullanmak isteyen ticari kuruluşlar için de mağazada, gelir seviyesinin de artacağı öngörülmektedir.

“Akıllı mobil ödeme sistemi” kapsamında geliştirilen bu projede, bir müşterinin minimum sürede mağazadan alışverişini tamamlayıp mağazadan çıkışı esas alınmıştır. Bu bağlamda geliştirilen yöntemde farklı ödeme teknikleri söz konusu olabilmektedir.

Ödemeye ilişkin üretilen QRCode teknolojisi yerine bir takım farklı teknolojiler kullanılması söz konusu olabilmektedir. Aynı şekilde geliştirilen mobil uygulama teknolojileri yönünden de native uygulama geliştirme teknolojileri gibi farklı teknolojiler kullanımı olabilir.

Geliştirilen web servis tarafında da bir takım farklı yazılım geliştirme dilleri kullanılabilir. Benzer şekilde web servisin kurulu olduğu sunucu ve web servisin üzerinde çalıştığı Internet Information Services (IIS) için de farklı kullanımlar olabilir.



## KAYNAKLAR DİZİNİ

**Adams J.**, 2013, Learning Kendo UI Web Development. Packt.

**Ashton K.**, 2012, “That ‘Internet of Things’ Thing”. <http://www.rfidjournal.com/articles/view?4986> (Erişim Tarihi: 28.05.2018).

**Aypara Ödeme Kuruluşu**, “iPara”, <https://www.ipara.com/> (Erişim tarihi: 26.12.2017).

**Bankalararası Kart Merkezi**, “BKM Express”, <https://bkmexpress.com.tr/> (Erişim tarihi: 26.12.2017).

**Batra S.**, 2003, AJAX, Asynchronous Javascript and XML, University of Applied Science and Technology Information Technology and Systems Management.

**BMO Bank of Montreal, CIBC, National Bank of Canada, RBC Royal Bank, Scotiabank, TD Bank Group**, 2013, Payments Security White Paper, BMO Bank of Montreal.

**Chaix L, Torre D.**, 2011, Four Models for Mobile Payments, University Nice Sophia-Antipolis.

**Droettboom M.**, 2015, Understanding JSON Schema. Space Telescope Science Institute.

**Dykstra T, Anderson R.**, 2014, Getting Started with Entity Framework 6 Code First using MVC 5. Microsoft.

**European Payment Council**, 2012, White Paper Mobile Payments, Brussels, Belgium, EPC492-09-12.

**European Payment Council**, 2014, White Paper Mobile Wallet Payments, Brussels, Belgium, EPC492-09-16.

**European Payment Council**, 2016, White Paper Mobile Payments, Brussels, Belgium, EPC492-09-16.

**Garanti Bankası**, “iGaranti”, <http://www.igaranti.com.tr/ig> (Erişim tarihi: 26.12.2017).

**Gökhan Y, Ahmet M, Figen Ö, Ahmet Ç.**, 2014, “NFC Tabanlı Akıllı Alışveriş Sistemi”. <http://docplayer.biz.tr/18352431-Nfc-tabanlı-akilli-alisveris-sistemi.html> (Erişim Tarihi: 28.05.2018).

**Hürriyet**, “Kasiyersiz Kasa”, <http://www.hurriyet.com.tr/marketlerde-kasiyersiz-asa-donemi-12291438> (Erişim tarihi: 26.12.2017).

## KAYNAKLAR DİZİNİ (devam)

- IONIC Framework**, “IONIC Framework”, <http://ionicframework.com/docs/> (Erişim tarihi: 26.12.2017).
- İş Bankası**, “ParaKod”, [www.parakod.com.tr](http://www.parakod.com.tr) (Erişim tarihi: 26.12.2017).
- JQuery Community Experts**, 2010, JQuery Cookbook, O'REILLY.
- Lotlikar T, Kankapurkar R, Parekar A, Mohite A.**, 2013, Comparative Study of Barcode, QR-code and RFID System, International Journal of Computer Technology and Applications.
- Manuel P, Singh I, Cicchetti A.**, 2012, Comparison of Cross-Platform Mobile Development Tools, International Conference on Intelligence In Next Generation Networks.
- Mehrjerdi Y.**, 2014, RFID: A Bibliographical Literature Review with Future Research Directions, International Journal of Industrial Engineering & Production Research.
- Microsoft**, “ASP.NET Web API 2: HTTP Message Lifecycle”, <http://www.asp.net/media/4071077/aspnet-web-api-poster.pdf> (Erişim tarihi: 26.12.2017).
- Mikro Ödeme**, “Mikro Ödeme”, <http://www.mikro-odeme.com/tr/> (Erişim tarihi: 26.12.2017).
- Minihold R.**, 2011, Near Field Communication (NFC) Technology and Measurements, Rodhe&Schwarz.
- Mistry R, Misner S.**, 2014, Introducing Microsoft SQL Server 2014 Technical Overview. A Division of Microsoft Corporation One Microsoft Way Redmond. Washington 98052-6399.
- MobilExpress**, “MobilExpress”, <https://www.mobilexpress.com.tr/Default.aspx#mobilexpress-nedir> (Erişim tarihi: 26.12.2017).
- Moore R.**, 2014, PhoneGap. Alpha Software.
- Narayanan A.**, 2012, QR Codes and Security Solutions, International Journal of Computer Science and Telecommunications.
- Paypal**, “Paypal”, <https://www.paypal.com/tr/home> (Erişim tarihi: 26.12.2017).
- Redda Y.**, 2012, Cross Platform Mobile Applications Development, Norwegian University of Science and Technology.

## KAYNAKLAR DİZİNİ (devam)

- Smart Card Alliance Payments Council**, 2014, Technologies for Payment Fraud Prevention: EMV, Encryption and Tokenization, Brussels, Belgium, EPC492-09-14.
- Treasury Alliance Group**, 2014, Fundamentals of Payment System, Treasury Alliance Group.
- Turkcell**, “Turkcell Cüzdan”, <http://www.turkcell.com.tr/servisler/turkcell-cuzdan> (Erişim tarihi: 26.12.2017).
- Türkiye Cumhuriyet Merkez Bankası**, “Ödeme Sistemleri”, <http://www.tcmb.gov.tr/wps/wcm/connect/a28ff0bd-1e9a-42be-8f7d-3249a1b3cf1c/OdemeSistemleri.pdf?MOD=AJPERES> (Erişim tarihi: 26.12.2017).
- Uğurlu T, Zeitler A.**, 2013, Pro ASP .NET Web API, HTTP Web Services in ASP .NET. Apress.
- Wahlin D.**, 2014, Angular JS in 60 Minutes. Google.
- Webrazzi**, “Turkcell Cüzdan”, <http://webrazzi.com/2012/10/04/turkcell-cuzdan> (Erişim tarihi: 26.12.2017).
- Weinstein R.**, 2005, RFID: A Technical Overview and Its Application to the Enterprise, IEEE IT Professional.
- Wikipedia**, “PrePay”, [https://en.wikipedia.org/wiki/Prepay\\_mobile\\_phone](https://en.wikipedia.org/wiki/Prepay_mobile_phone) (Erişim tarihi: 26.12.2017).