T.C

İSTANBUL AYDIN ÜNİVERSİTESİ

ANADOLU BİL MESLEK YÜKSEKOKULU

BİLGİSAYAR PROGRAMCILIĞI PROGRAMI

STAJ YERİ YORUMLAMA UYGULAMASI

Ön Lisans Bitirme Projesi

OĞUZ ÇÖREKÇİOĞLU

İSTANBUL, 2018

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
ANADOLU BİL MESLEK YÜKSEKOKULU

BİLGİSAYAR PROGRAMCILIĞI PROGRAMI

STAJ YERİ YORUMLAMA UYGULAMASI

Ön Lisans Bitirme Projesi

OĞUZ ÇÖREKÇİOĞLU

Danışman

ÖGR. GÖR. HAKAN BURAK EMEKLİ

İSTANBUL, 2018

ÖZET

STAJ YERİ YORUMLAMA

Oğuz Çörekçioğlu

Bilgisayar Programcılığı Programı

Danışman: Öğr. Gör. Hakan Burak EMEKLİ

Haziran 2018

ÖZET

İlgili bölümde okuyan öğrenci staj yerini bulur veya bulmadan önceki son adımda yani karar verme aşamasında staj yeri hakkında daha önceden orada staj yapmış kişilerin deneyimlerini öğrenmek ve gideceği kurumun kendi istediği hedeflerine uygun kurum mu olacağıyla ilgili mobil uygulamayı kullanarak şirket ya da kurum hakkındaki yorumları okur. Aynı zamanda kendi staj yaptığı yada çalıştığı kurum hakkındaki yorumlarını da sisteme dahil edebilir arzu ederse sonradan silebilir.

Anahtar Sözcükler: Mobil Uygulama, Staj, Öğrenci, Kurum, Çalışılan Kurum, C#, Xamarin, Xamarin.Android, Xamarin.IOS.

İÇİNDEKİLER

İÇİNDEKİLER i

TABLO LİSTESİ ii

ŞEKİL LİSTESİ ii

KISALTMALAR iv

SİMGELER v

1. GİRİŞ 6

1.1. Projenin Amacı 6

1.2. Kullanılan Teknoloji Ve Programlar 7

1.2.1. Microsoft Visual Studio 2017 7

1.2.2. Microsoft Azure 7

1.2.3. C# ve .NET Kütüphanesi 8

1.2.4. Microsoft SQL Server ve Management Studio 10

1.2.5. Xamarin ve Xamarin.Android 11

1.2.5.1. Derleme (Compilation) 16

2. VERİTABANI TASARIMI 18

2.1. Veritabanının Oluşturulması 18

2.2. Microsoft Azure Ve Veritabanı Hizmeti 22

2.3. Microsoft Azure'a Local'de Bulunan Veritabanını Göç Ettirme 25

2.3.1. Microsoft Azure User Oluşturma ve Veritabanı Güvenliği 27

3. PROJE 29

3.1. Projenin Oluşturulması ve İlk Aşama Bağlantı Nesnesinin Yazılması 30

3.2. Projemizin İlk Sayfası Layout ve Activity Sınıfı 31

3.3. Fragmentlerle Çalışma ve Projemizin Başka Sayfasına Geçiş 34

3.3.1. Fragmentlere Animasyon Verme 38

3.4. Kurumların Listesini Görme ve Yorumlarını Okuma 39

3.5. Kendi Yorumlarımızı Görme Ve Silme 45

3.6. Yeni Bir Yorum Oluşturma Ve Veritabanına Ekleme 47

4. SONUÇ 49

KAYNAKÇA 51

TABLO LİSTESİ

ŞEKİL LİSTESİ

Şekil 1 - Azure Hizmetleri 8

Şekil 2 - .Net Derleme Ve Çalıştırma 10

Şekil 3 - Örnek Arayüz XAML kodu 13

Şekil 4 - Örnek XAML kodu çıktısı 14

Şekil 5 - Xamarin Android veya IOS Seçiminde 15
Şekil 6 - Xamarin.Forms Seçiminde 15
Şekil 7 - Xamarin Derleme 16
Şekil 8 - Xamarin Yeni Proje Açımı 17
Şekil 9 - Veritabanı Diyagramı 21
Şekil 10 - Azure Portal 22
Şekil 11 - Azure Database Server Oluşturulması 23
Şekil 12 - Azure Database Oluşturulması 24
Şekil 13 - SQL Veritabanı Hizmeti Genel Görünüm 25
Şekil 14 - Data Migration Assistant 26
Şekil 15 - Microsoft Azure Çalışma Mantığı 28
Şekil 16 - Projemizin Ilk Activity'i 31
Şekil 17 - Giriş Axml Layout'u 32
Şekil 18 - Giriş. Axml Kodları 33
Şekil 19 - DialogFragment Class'ı 35
Şekil 20 - Dialog Giriş Layout'u 35
Şekil 21 - Fragment Kodları 37
Şekil 22 - Style Klasöründen Bir Görünüm 39
Şekil 23 - Staj Class'ı Veritabanınıdaki Tabloyu Temsilen 40
Şekil 24 - listeguncelle Methodumuz 40
Şekil 25 - Oncreate Metodu Görünüm 41
Şekil 26 - ItemClick Event'i 41
Şekil 27 - ActionBar'ın Oluşturulması 42
Şekil 28 – ActionBarXml 42

Şekil 29 - Case Search..... 43
Şekil 30 - Case Refresh..... 43
Şekil 31 - ListView ve ActionBar..... 44
Şekil 32 - Yorum Sayfası..... 44
Şekil 33 – YorumGorSil..... 45
Şekil 34 - LongItemClick ve AlertDialog..... 46
Şekil 35 - Thread1..... 46
Şekil 36 – ScrollView..... 47
Şekil 37 - ScrollView Çıktısı..... 48
Şekil 38 - Kayıt Oluştur Butonu..... 48
Şekil 39 - Sql Sorgusu ve ToastMakeText..... 49
Şekil 40 - Desteklenen Cihazlar..... 50

KISALTMALAR

MSSQL: Microsoft SQL Server olarak açılmaktadır. Microsoft'un Veritabanı hizmeti ürünüdür.

ADO.NET: Active Data Object Olarak açılmaktadır. Veritabanı ile C# dilini konuşturmak için kullanılan teknolojilerden birisidir. Ado.Net bir programlama dili değildir.

LINQ: Language Integrated Query olarak açılmaktadır, dil ile bütünleşik sorgular yazmamızı sağlar tam haliyle .Net dillerinde SQL benzeri sorgulama becerisi ekleyen bir Microsoft .Net Framework bileşenidir (Wikipedia, 2017).

T-SQL: Transact Sql yani T-SQL Microsoft Sql Server veritabanı için kullanılan sorgulama diline verilen isimdir.

IDE: Integrated Development Environment olarak açılmaktadır, bilgisayar programcılarının hızlı ve rahat bir şekilde yazılım geliştirebilmesini amaçlayan, geliştirme sürecini organize edebilen birçok araç ile birlikte geliştirme sürecinin verimli kullanılabilmesine katkıda bulunan araçların tamamını içerisinde barındıran bir yazılım türüdür (Wikipedia, 2016).

RDBMS: Relational Database Management System yani ilişkisel veritabanı sistemi olarak geçer.

API: Application Programming Interface yani Kullanıcı programı arayüzü anlamına gelmektedir. Bir uygulamaya ait yeteneklerin başka bir uygulamada da kullanılabilmesi için, yeteneklerini paylaşan uygulamanın sağladığı arayüzdür.

SİMGELER

1. GİRİŞ

Bilgisayar Teknolojisi insan hayatında çok önemli bir konumdadır. Ve insan hayatını kolaylaştırmak için birçok yazılım vardır, giderek artan teknolojinin hızlı gelişmesi ile hayatımıza küçük bilgisayarlar diyebileceğimiz akıllı cep telefonları girmiştir ve gününüzde kişisel bilgisayarı (laptop, masaüstü vb.) olmayan kişilerin bile akıllı telefon adı verdikleri cep telefonlarını kullanmaktadırlar ve buda mobil tabanlı uygulamalar geliştirilmesi ihtiyacının ortaya çıkmasını sağlamıştır, bizde mobil tabanlı sistemlerden birisi olan Linux tabanlı özgür işletim sistemi olan Android işletim sistemine projemin kapsamı doğrultusunda staj yeri yorumlaması uygulaması geliştirdim ve adına StajBilgisi dedim.

1.1. Projenin Amacı

Projede amaç insanların mobil tabanlı staj bilgisi uygulamasını kullanarak herhangi bir işyerine staja girmeden önce veya bir site veya kurum aracılığıyla çalışılacak bir kurum bulduğunda o kuruma gitmeden önce kendisini için doğru yeri mi seçtiğini yapacağı işi daha önceden yapmış aynı mekânda çalışmış bir kişinin ilgili kurumla ilgili yorumlarını okumak veya arzu ederse kendi deneyimlerini sisteme dahil etmek bu şekilde daha fazla kişiye yardımcı olmak.

1.2. Kullanılan Teknoloji Ve Programlar

Aşağıda projemizde kullanılan teknoloji ve programların açıklamalarının ayrıntılı bilgileri verilmiştir.

1.2.1. Microsoft Visual Studio 2017

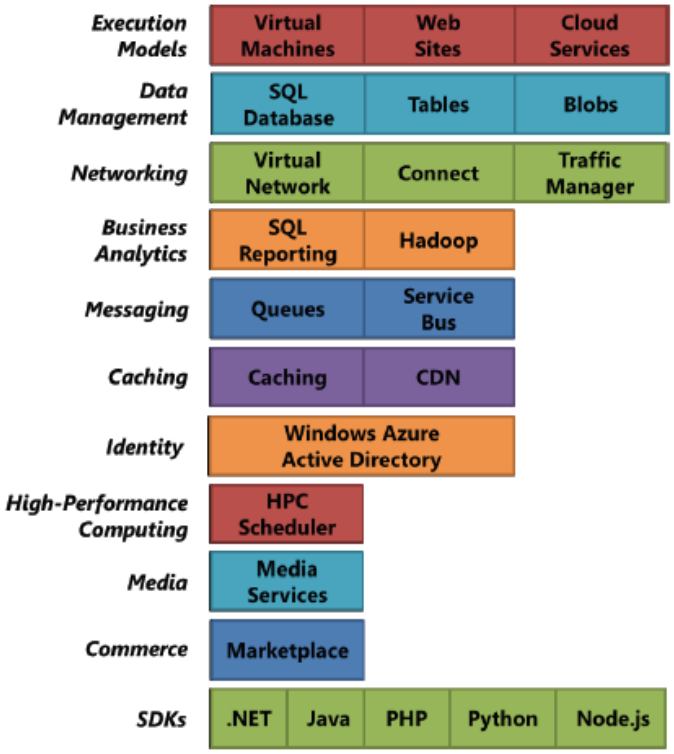
Microsoft Visual Studio, Microsoft tarafından geliştirilen bir tümleşik geliştirme ortamıdır (IDE). Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework tarafından desteklenen tüm platformlar için yönetilen kod ile birlikte yerel kod ve Windows Forms uygulamaları, Web Siteleri, Web Uygulamaları ve Web Servisleri ile birlikte konsol ve grafiksel kullanıcı arayüzü uygulamaları geliştirmek için kullanılır.

Visual Studio, değişik programlama dillerini destekler, bu da kod editörü ve hata ayıklayıcısının neredeyse tüm programlama dillerini desteklemesini sağlamaktadır. Dahili diller C/C++ (Görsel Yoluyla C++), C#(Visual C#) ile, F#, ve daha eklenebilecek birçok dil örneğin: Python içermektedir.

1.2.2. Microsoft Azure

Şu şekilde açıklanabilir, Microsoft'un bilişim sektöründe sunduğu, bulut bilişim hizmetleri sağlayan ve Windows Server işletim sisteminin özelleştirilmiş bir ürünüdür. İnternet Uygulamaları, Veri Depolama Merkezi, Sanallaştırma Platformu

gibi IT sektörünün kullanabileceği teknolojilerin bulunduğu platformlar Microsoft Azure altyapısında mevcuttur. Microsoft Azure teknolojisinin test ortamı içermesi ve ölçeklenebilir olması da kurumların bu teknolojiyi tercih etme sebeplerinden birisidir.



Şekil 1 - Azure Hizmetleri

Yukarıda kullanılabilecek bazı Azure Hizmetleri Gösterilmiştir.

1.2.3. C# ve .NET Kütüphanesi

C# yazılım sektörü içerisinde en sık kullanılan iki yazılım olan C ve C++ etkileşimi ile türetilmiştir. Ayrıca C#, ortak platformlarda taşınabilir bir (portable language) programlama dili olan Java ile pek çok açıdan benzerlik taşımaktadır. En büyük özelliği ise .Net Framework platformu için hazırlanmış tamamen nesne

yönelimli bir yazılım dilidir. Yani nesneler önceden sınıflar Halinde Yazılıdır. Programcıya sadece o nesneyi sürüklemek ve sonrasında nesneyi amaca uygun çalıştıracak kod satırlarını yazmak kalır.

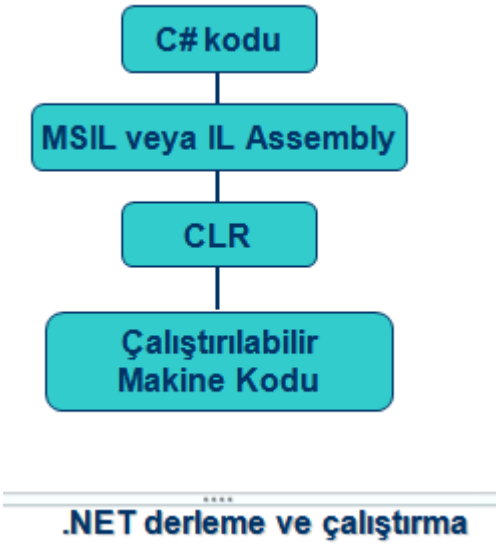
Microsoft tarafından geliştirilen C#, C++ ve Visual Basic dillerinde yer alan tutarsızlıkları kaldırmak için geliştirilmiş bir dil olmasına rağmen kısa süre içerisinde nesne yönelimli dillerin içerisinde en gelişmiş programlama dillerinden biri olmayı başarmıştır.

C# ve .Net Framework bazı kişiler tarafından tek bir kavram olarak algılanmaktadır. Fakat bu kavram birbirinden tamamen farklı amaçlar için geliştirilmiştir. Aslında C# dili Microsoft tarafından .Net Platformu için kod geliştirmek amaçlı tasarlanmış ve C# içerisindeki tüm kütüphaneler .Net Platformu içerisinde tanımlanmış kütüphanelerdir.

.Net Platformu da Java diline benzer bir çalışma mantığı izleyerek kodları çalışabilir hale getirmektedir.

.Net platformunda kod ilk önce Microsoft Intermediate Language(Microsoft Ara Dili) olarak isimlendirilmiş dosya haline dönüştürülür. Bu ara dilde saklanan

dosyalar çalışmak istenildiğinde ise CLR adı verilen sistem MSIL kodlarını çalıştırır.



Şekil 2 - .Net Derleme Ve Çalıştırma

Ortak dil çalışma zamanı CLR sisteminin temel görevi ise C# dilini taşınabilir kılmak ve diğer diller ile güvenli bir şekilde çalışmayı sağlayan sistemdir. CLR .Net Platformuna ait bir kod çalıştığı zaman JIT(Just In Time) derleyici aktif hale getirir. Aktif hale gelen jit, MSIL kodlarını yerel kod yapısına göre çalıştırarak ortak platform yapısı sağlanmış olur (Bozkurt, 2011).

1.2.4. Microsoft SQL Server ve Management Studio

SQL Server Versiyon 7.0'dan önce "kod tabanı" Sybase SQL Server tarafından Microsoft'a satılmıştır. Günümüzde Microsoft tarafında geliştirilmekte olan ilişkisel veritabanı yönetim sistemidir. İngilizce olarak RDBMS olarak geçer. Birde SSMS yani SQL Server Management Studio Mevcuttur bu ise Local'de veya uzaktaki veritabanı motoruna bağlanarak veritabanları oluşturmamızı, tabloları yönetmemizi gerekli CRUD işlemlerini gerçekleştirmemizi kolaylaştırır.

1.2.5. Xamarin ve Xamarin.Android

Xamarin en basit tanımı ile; C# dilinin kolaylıklarını kullanarak IOS, Android ve UWP için hızlı bir şekilde uygulamalar geliştirmenizi sağlayan platformdur. Yani C# ile yazarız IOS, Android ve UWP' ye kolay bir şekilde native bir uygulama yayınlarız. Bunu Yaparken Objective-C, Swift yada Java bilmemize gerek olmaz. Xamarin ile geliştirdiğimiz uygulamaların native olarak deploy edilebilmesi, code re-use'nun yani tekrar kod kullanılabilirliğinin maksimumda olması, güçlü oyun kütüphaneleri sunması ve Objective–C, Java, Swift ile yapılabilen her şeyin Xamarin 'de C# ile yapılabiliyor olması Xamarin 'in en büyük avantajlarındandır.

Bunlara ilave olarak Xamarin ile ;

- Mevcut Objective-C ve Java kodumuzu Xamarin ile kullanabilir.
- IOS Watch App ve Android Wear App geliştirebilir.
- IOS Api ve Android Api'ye tamamen erişebilirsiniz.

Xamarin 2 temel yaklaşım sunmaktadır. Bunlar;

- Xamarin.Forms
- Xamarin.Android ve Xamarin.IOS (Daha önceden kullanılan Xamarin Native ve Xamarin.Traditional isimleri ile de karşılaşabilirsiniz)

Xamarin.Forms: Tüm platformlar (IOS, Android, UWP) için kolay bir şekilde native arayüz oluşturmamızı sağlayan ve yazdığınız kodun yine tüm platformlar arasında paylaşılmasını sağlayan cross-platform yani çoklu platform, platform bağımsız yaklaşımdır. Xamarin.Forms ile yeni bir proje yarattığınız zaman, Solution içerisinde aşağıdaki projeler otomatik olarak eklenir.

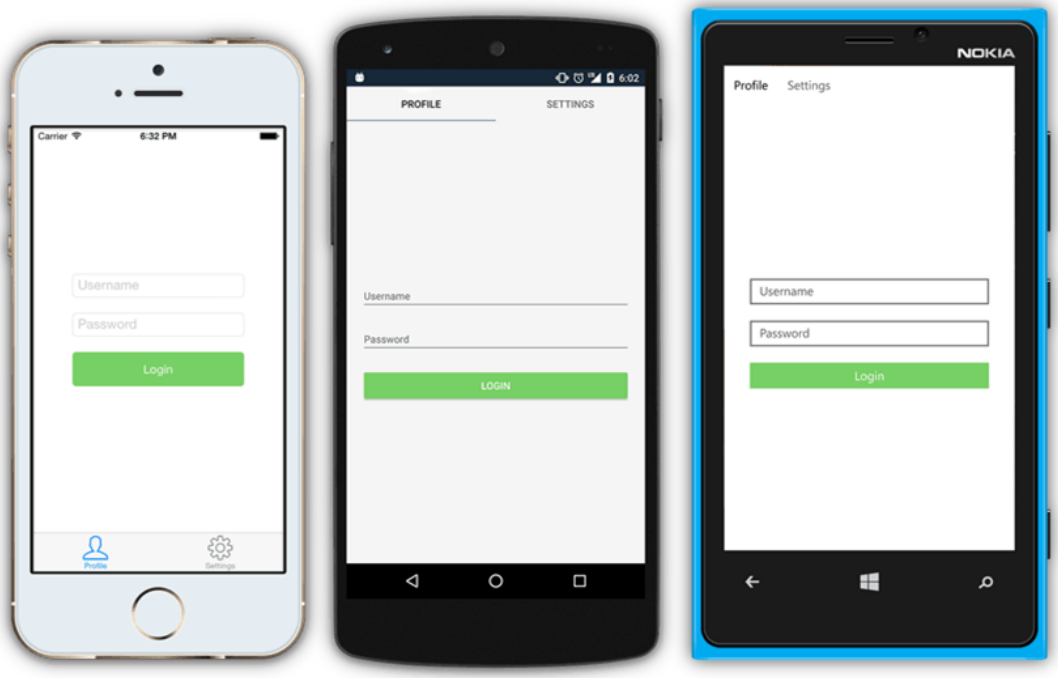
- 1 adet Code Sharing projesi (Shared Project yada Portable Class Library[PCL]),
- 1 adet IOS projesi,
- 1 adet Android projesi,
- 1 adet UWP Projesi (UWP yüklü ise)

Xamarin.Forms ile, uygulama arayüzünü CodeSharing projesi içerisinde XAML veya C# kullanarak tasarlarsanız ve bu tasarladığınız arayüz tüm platformlar için kullanılır. Bu sayede IOS için ayrı Android için ayrı Windows için ayrı arayüzler tasarlamak zorunda kalmazsınız. Yine aynı şekilde, Code Sharing projesi içerisinde yazdığınız kodlar tüm platformlar için paylaşılır. Tekrar tekrar yazmak zorunda kalmazsınız.

```
<?xml version="1.0" encoding="UTF-8"?>
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="MyApp.MainPage">
    <TabbedPage.Children>
        <ContentPage Title="Profile" Icon="Profile.png">
            <StackLayout Spacing="20" Padding="20"
                         VerticalOptions="Center">
                <Entry Placeholder="Username"
                       Text="{Binding Username}"/>
                <Entry Placeholder="Password"
                       Text="{Binding Password}"
                       IsPassword="true"/>
                <Button Text="Login" TextColor="White"
                        BackgroundColor="#77D065"
                        Command="{Binding LoginCommand}"/>
            </StackLayout>
        </ContentPage>
        <ContentPage Title="Settings" Icon="Settings.png">
            <!-- Settings -->
        </ContentPage>
    </TabbedPage.Children>
</TabbedPage>
```

Şekil 3 - Örnek Arayüz XAML kodu

Code Sharing projesi içerisinde yukarıdaki gibi yazılmış bir XAML kodunun IOS, Android ve Windows Phone görüntüsü aşağıdakine benzer olacaktır.



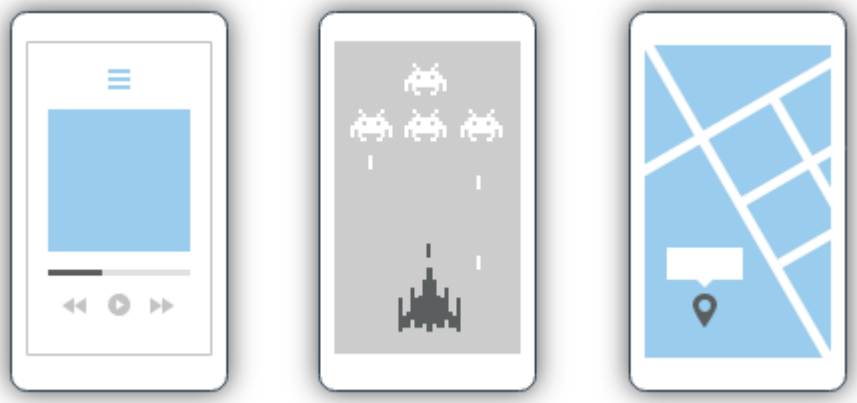
Şekil 4 - Örnek XAML kodu çıktısı

Xamarin.Android ve Xamarin.IOS: Platforma özel uygulama geliştirmenizi sağlayan yaklaşımdır. Her bir platform için ayrı proje oluşturulur. Yani Android için ayrı, IOS için ayrı, Windows için ayrı projeler açarak ayrı ayrı geliştirme yaparsınız.

Peki Hangi Yaklaşımı Tercih Etmeliyiz ?

Eğer :

- Native davranış gerektiren etkileşimlere sahip uygulama geliştirecekseniz,
- Platforma özel Apilerden bolca kullanacaksanız,
- Projenizde platforma özel arayüz, kod paylaşımından daha önemli ise, seçiminiz Xamarin.Android ve Xamarin.IOS' dan yana olmalıdır.



Şekil 5 - Xamarin Android veya IOS Seçiminde

Eğer:

- Platforma özel işlevsellik az ise,
- XAML diline aşinaysanız,
- Projenizde kod paylaşımı platforma özel arayüzden daha önemli ise,

seçiminiz Xamarin.Forms'dan yana olmalıdır.

Şekil 6 - Xamarin.Forms Seçiminde

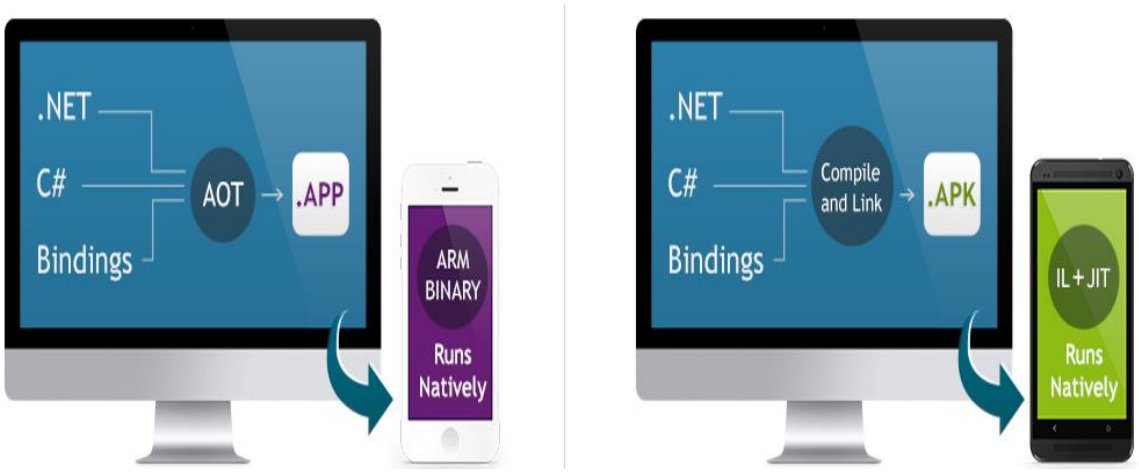
1.2.5.1. Derleme (Compilation)

Yazmış olduğumuz C# kodu her platform için farklı yollarla native uygulamaya dönüştürülmektedir.

IOS Sürecinde: C# kodu ARM assembly diline derlenir. Apple'ın AOT (Ahead of Time) derleyicisi kullanılır. Uygulamanın boyutunun düşük olması için kullanılmayan class'lar derleme sürecine dahil edilmezler.

Android Sürecinde: C# kodu IL'e derlenir ve MonoVM ve JIT'leme ile paketlenir. IOS sürecinde olduğu gibi Android'de de uygulama boyutunun düşük olması için kullanılmayan class'lar derleme sürecine dahil edilmezler.

Windows Sürecinde: C# kodu IL'e derlenir ve built-in runtime (yerleşik derleme) tarafından çalıştırılır.



Şekil 7 - Xamarin Derleme

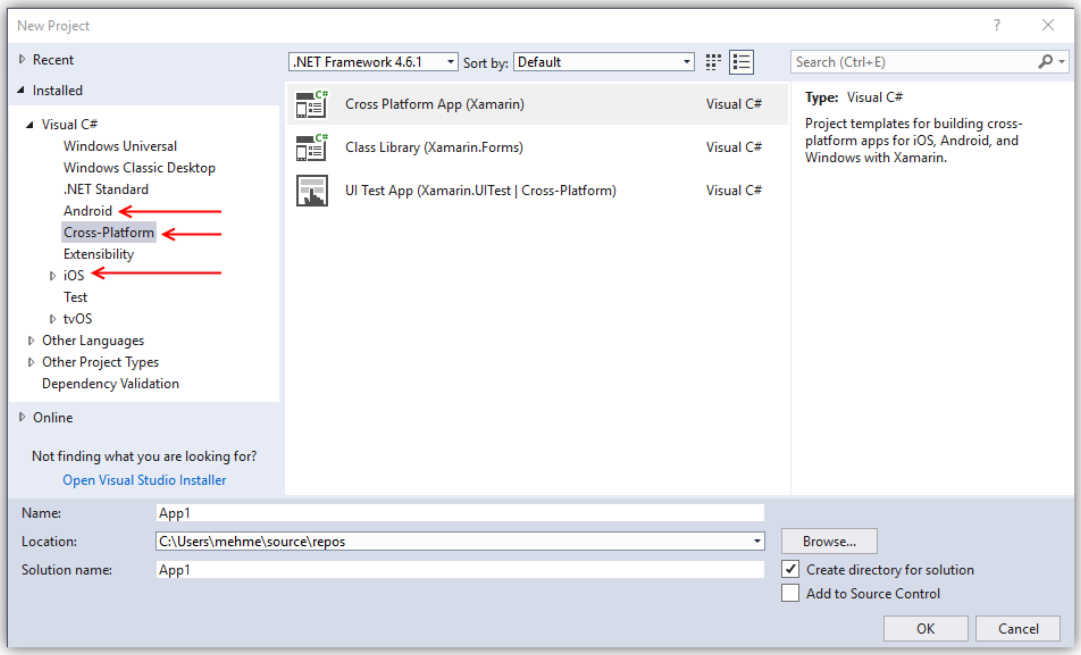
Xamarin ile uygulama geliştirmek için aşağıdaki IDE'leri kullanabilirsiniz;

- Visual Studio
- Visual Studio for Mac
- Xamarin Studio for Mac

- Xamarin Studio for Windows (Visual Studio 2017 zaten Xamarin Studio'nun tüm fonksiyonalitesini barındırdığı için Xamarin Studio for Windows desteklenmemektedir.) (Yıldız, 2017)

Biz projemizi geliştirirken Visual Studio 2017 idesini kullandık. Ayrıca Visual Studio'yu indirirken Installer ekranında Mobile & Gaming başlığının altından Mobile development with .NET seçeneğini işaretleyip Install butonuna tıklamamız yeterlidir. Bir Xamarin Projesinin açımı için :

Kurulum Sonrası Visual Studio üzerinden New -> Project seçtiğiniz zaman gelen pencerede Android, Cross-Platform ve IOS seçeneklerini gördüyseniz kurum başarıyla gerçekleşmiş ve istediğiniz proje tipini seçerek yazılım geliştirmeye başlayabilirsiniz demektir.



Şekil 8 - Xamarin Yeni Proje Açımı

2. VERİTABANI TASARIMI

2.1. Veritabanının Oluşturulması

Veritabanının oluşturulması için :

```
create database staj
```

T-SQL dili kullanılarak Management Studio ile çalıştırılmıştır. Daha sonra tabloları oluşturmak için aşağıdaki kodlar kullanılmıştır :

Üye Tablosu İçin :

```
USE [staj]
GO


CREATE TABLE [dbo].[tbl_uye](
	[uye_id] [int] IDENTITY(1,1) primary key NOT NULL,
	[kullanici_ad] [nvarchar](75) NOT NULL,
	[kullanici_sifre] [nvarchar](75) NOT NULL,
	[uye_ad] [nvarchar](75) NULL,
	[uye_soyad] [nvarchar](75) NULL,
	[uye_cinsiyet] [nvarchar](50) NULL,
	[uye_il] [nvarchar](75) NULL,
	[uye_ilce] [nvarchar](75) NULL,
	[uye_bolum] [nvarchar](200) NULL,
)
```

Burada use staj keyword'u staj database'ine bu tabloyu kaydedeceğimizi göstermektedir, kolon adları uye_id, kullanici_ad, şeklinde devam etmektedir. Identity(1,1) ilgili tabloya kayıt yapılacağı zaman birer birer sayının artması şeklinde olduğunu göstermektedir, primary key ise o kolonun tekil olacağını garanti ettirmektedir, not null o alanın boş geçilemeyeceğini, nvarchar(75) unicode string karakterler girilebileceğini, int ise -2 milyar ile +2 milyar arasında değer alınabileceğini ifade etmektedir.

İçerik Tablosu İçin :

```
USE [staj]
GO


CREATE TABLE [dbo].[tbl_icerik](
	[yorum_id] [int] IDENTITY(1,1) primary key NOT NULL,
	[kurum_ad] [nvarchar](500) NOT NULL,
	[kurum_departman] [nvarchar](500) NOT NULL,
	[kurum_calisan] [int] NOT NULL,
	[kurum_gorev] [nvarchar](500) NOT NULL,
	[kurum_adres] [nvarchar](600) NOT NULL,
	[kurum_yorum] [nvarchar](1000) NOT NULL,
	[uye_id] [int] NOT NULL,
	[tarih] [datetime] NULL,
 )
```

Burada use staj keyword'u staj database'ine bu tabloyu kaydedeceğimizi göstermektedir, kolon adları yorum_id, kurum_ad, şeklinde devam etmektedir. Identity(1,1) ilgili tabloya kayıt yapılacağı zaman birer birer sayının artması şeklinde olduğunu göstermektedir, primary key ise o kolonun tekil olacağını garanti ettirmektedir, not null o alanın boş geçilemeyeceğini null boş geçilebileceğini, datetime .Net zaman tipini, int ise -2 milyar ile +2 milyar arasında değer alınabileceğini ifade etmektedir, ayrıca buradaki uye_id kolonu tbl_uye tablosundaki uye_id'nin foreign key'idir, yani uye tablosunda yer almayan bir kullanıcı bu tabloya kayıt yapamaz anlamında gelmektedir.

İller Tablosu İçin :

```
USE [staj]
GO

CREATE TABLE [dbo].[iller](
	[il_no] [int] primary key NOT NULL,
	[isim] [varchar](50) NULL,
)
```

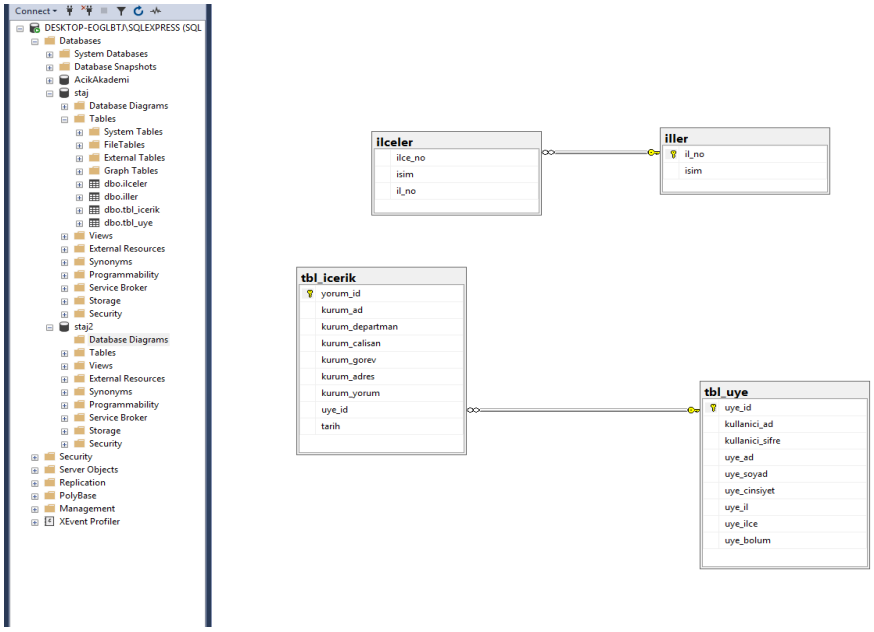
Burada use staj keyword'u staj database'ine bu tabloyu kaydedeceğimizi göstermektedir, kolon adları il_no, isim şeklindedir. İl no primary key'dir. Çünkü her il numarası tekildir.

İlçeler Tablosu İçin :

```
USE [staj]
GO


CREATE TABLE [dbo].[ilceler](
	[ilce_no] [int] NOT NULL,
	[isim] [varchar](50) NULL,
	[il_no] [int] NOT NULL
)
```

Burada use staj keyword'u staj database'ine bu tabloyu kaydedeceğimizi göstermektedir, kolon adları ilce_no, isim, il_no şeklindedir. Ayrıca burada il_no foreign key'dir ve referansı iller tablosunda il_no'dan gelmektedir. Buradaki amaç her ilin kaç tane ilçe 'si olduğunu bulabilmek içindir, veritabanları yazılım aşaması düşünülmeden ayrı bir mantıkla tasarlanmaktadır.

Veritabanımızın şeması :



Şekil 9 - Veritabanı Diyagramı

Görüldüğü üzere veritabanı diyagramında oluşturduğumuz veritabanları arasındaki ilişkiler ve bitmiş hali gözükmektedir fakat bu oluşturduğumuz veritabanı kendi bilgisayarımızdadır fakat bizim projemizde ise mobil tabanlı sistemler taşınabilir sistemler olduğu için internet ve bulut teknolojileri ile veritabanımızın herkese hizmet vermesi gerekmektedir bunun için Microsoft'un Azure Platformu kullanılmıştır.

Ayrıca Zamanı kullanıcıdan almayıp hangi tarihte veri girildiğini doğru belirlemek adına bir adet tablo bazlı trigger da yazılmıştır :

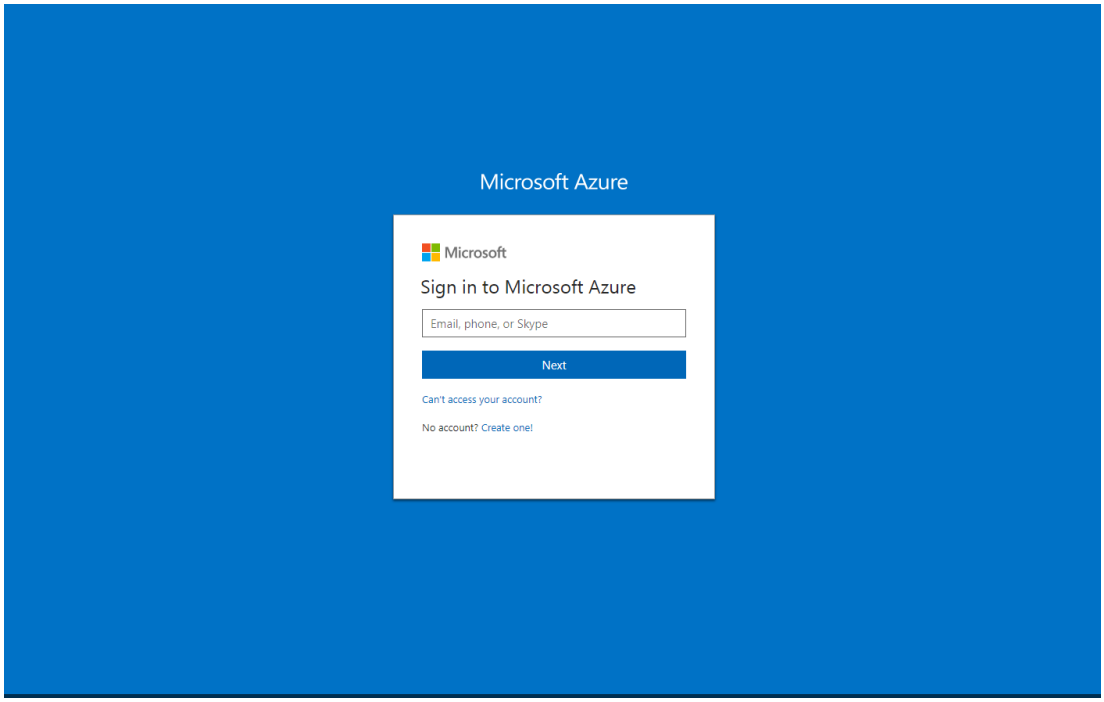
```
USE [staj]
GO
/****** Object:  Trigger [dbo].[tarihekle]    Script Date: 03/06/18 9:11:42 PM
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER trigger [dbo].[tarihekle] on [dbo].[tbl_icerik]
for insert
as
begin
```

```
declare @tarih datetime,@sonID int
select @tarih=GETDATE(),@sonID=IDENT_CURRENT('tbl_icerik') from inserted
update tbl_icerik set tbl_icerik.tarih=@tarih where yorum_id=@sonID
end
```

Bu trigger tbl_icerik tablosuna her insert işlemi gerçekleştikten sonra getdate() metodu yardımıyla o anki zamanı ilgili tablonun yeni eklenen verisine eklemektedir.

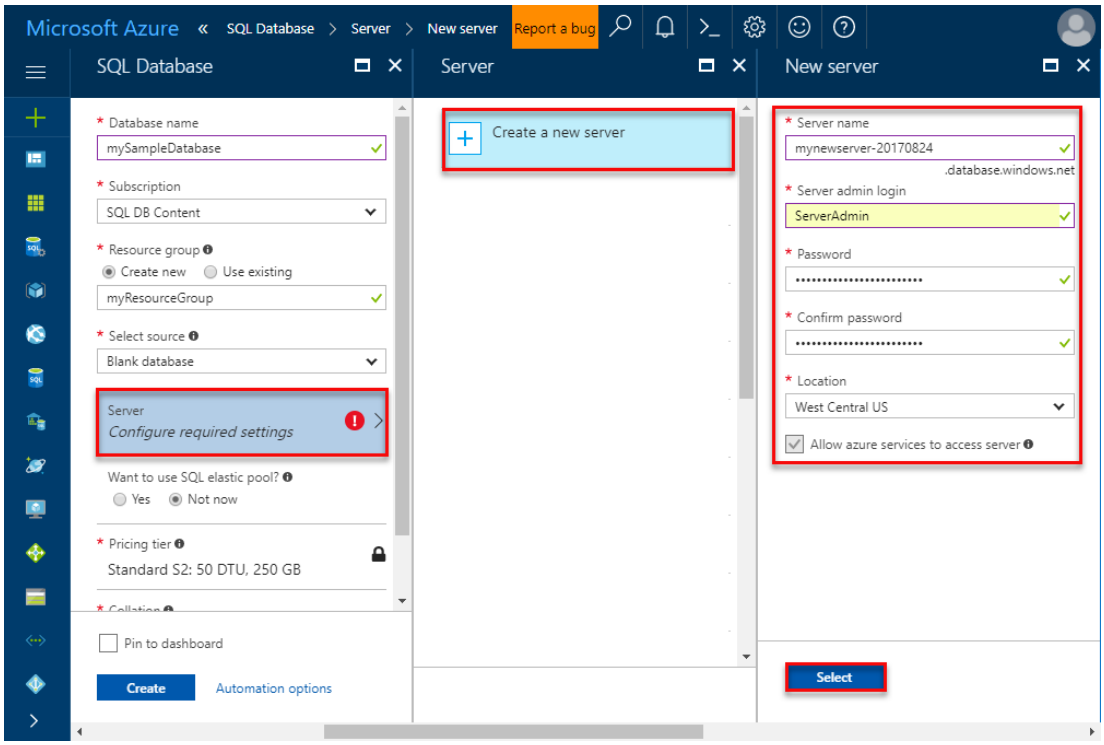
2.2. Microsoft Azure Ve Veritabanı Hizmeti

Microsoft Azure genellikle kuruluşların iş zorluklarını aşmasına yardımcı olacak şekilde tasarlanmış, sayısı her geçen gün artan bulut hizmetlerinden oluşur. Özellikle geliştiriciler için küresel bir ağda uygulama oluşturma, yönetme ve dağıtma özgürlüğü sunar (Microsoft, 2018).



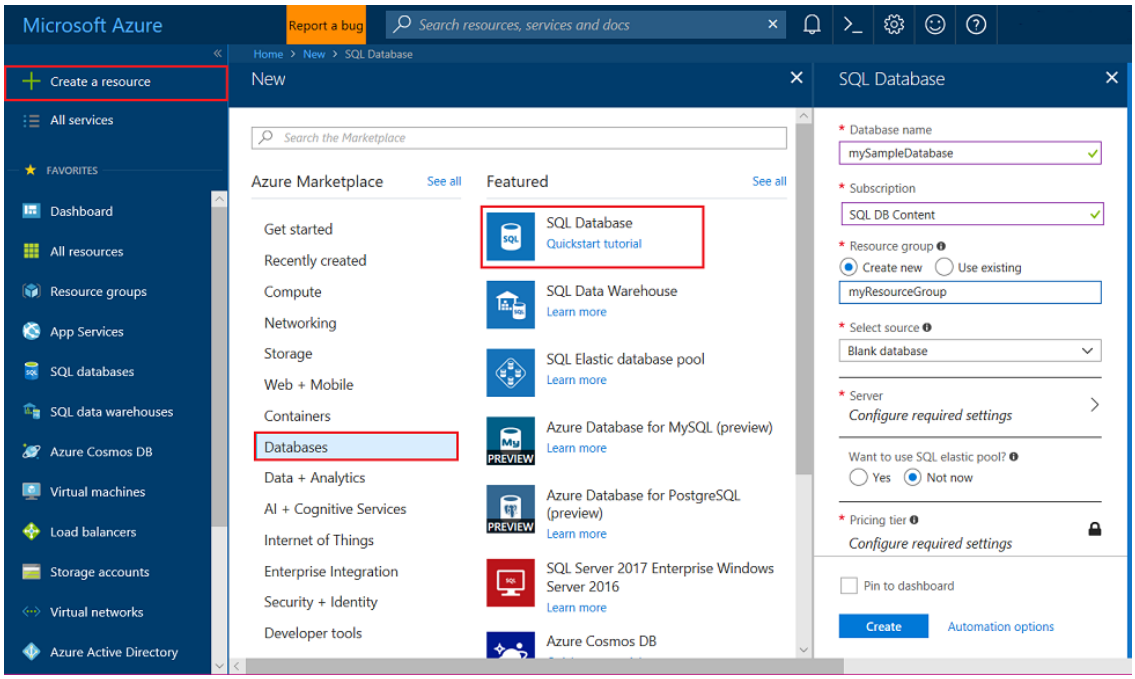
Şekil 10 - Azure Portal

Yukarıda Azure Hizmetlerine erişim için Azure Portal 'dan bir Görünüm Verilmiştir. Bizim projemizde ise Azure hizmetlerinden Azure SQL Veritabanı hizmeti kullanılmıştır. Kısacası SQL Veritabanı, Microsoft Azure 'da yer alan ve ilişkisel veri, JSON, uzamsal ve XML gibi yapıları destekleyen çok amaçlı ilişkisel veritabanı yönetilen hizmetidir. SQL Veritabanı birden fazla hizmet düzeyinde kesinti süresi olmadan dinamik kararlılık, yerleşik zekâ iyileştirmesi, global düzeyde ölçeklenebilirlik ve kullanılabilirlik ile gelişmiş güvenlik seçeneklerine sahip tahmin edilebilir performansı neredeyse sıfır yönetim gereksinimiyle sunar. SQL Veritabanı şu anda dünyanın farklı yerlerindeki toplam 38 veri merkezini kullanmaktadır ve düzenli aralıklarla eklenen yeni veri merkezleri, veritabanınızı bulunduğunuz konuma yakın bir veri merkezinde çalıştırmanızı sağlar.



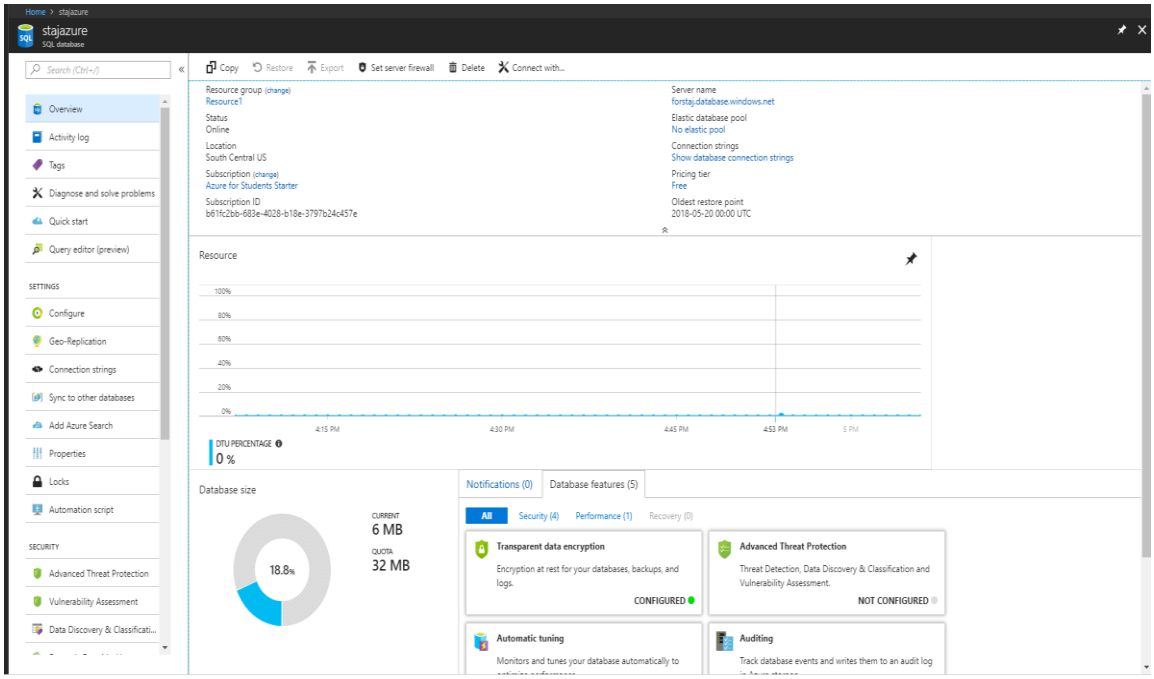
Şekil 11 - Azure Database Server Oluşturulması

Yukarıda Database oluşturmadan önce ilgili server oluşturulması ve ödeme planının seçilmesi aşamaları gözükmektedir.



Şekil 12 - Azure Database Oluşturulması

Yukarıda ise SQL Server'dan sonra Veritabanı oluşturulması işlemi gösterilmektedir.

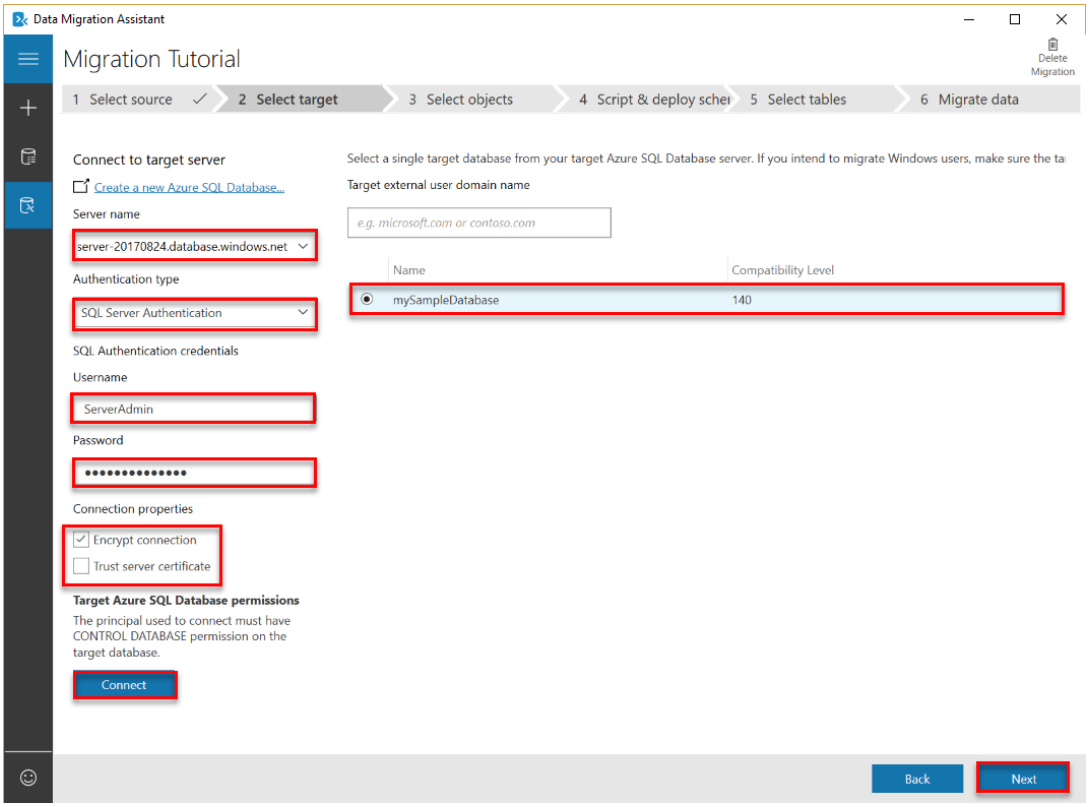


Şekil 13 - SQL Veritabanı Hizmeti Genel Görünüm

Görüldüğü üzere bizi karşılayan veritabanı sayfasında Veritabanımızın verdiği errorlar, kullanım yüzdesi kalan kullanım kapasitesi etkinleştirilen servis ve daha birçok şey gözükmektedir.

2.3. Microsoft Azure'a Local'de Bulunan Veritabanını Göç Ettirme

Biz Veritabanımızı hızlıca oluşturmak ve ilk başta bazı testleri hızlı yapmak için lokalde oluşturduk fakat sistemimiz her yerden kullanılacağı için oluşturduğumuz veritabanını yine Microsoft'un bir tool'u yardımıyla Azure Veritabanımıza Migration işlemini gerçekleştirdik bunun için, Microsoft Data Migration Assistant'ı kullandık aşağıda görsel öğeye yer verilmiştir.



Şekil 14 - Data Migration Assistant

Burada lokaldeki veritabanı ve istenirse diyagramı seçilerek gönderilecek veritabanı yani target server'ının ilgili username ve password'u girildikten sonra, internet hızına bağlı olarak yaklaşık 1, 2 dakikada migration işlemi tamamlanmıştır. Lokaldeki staj veritabanımız uzaktaki stajazure adlı veritabanına kopyalanmıştır.

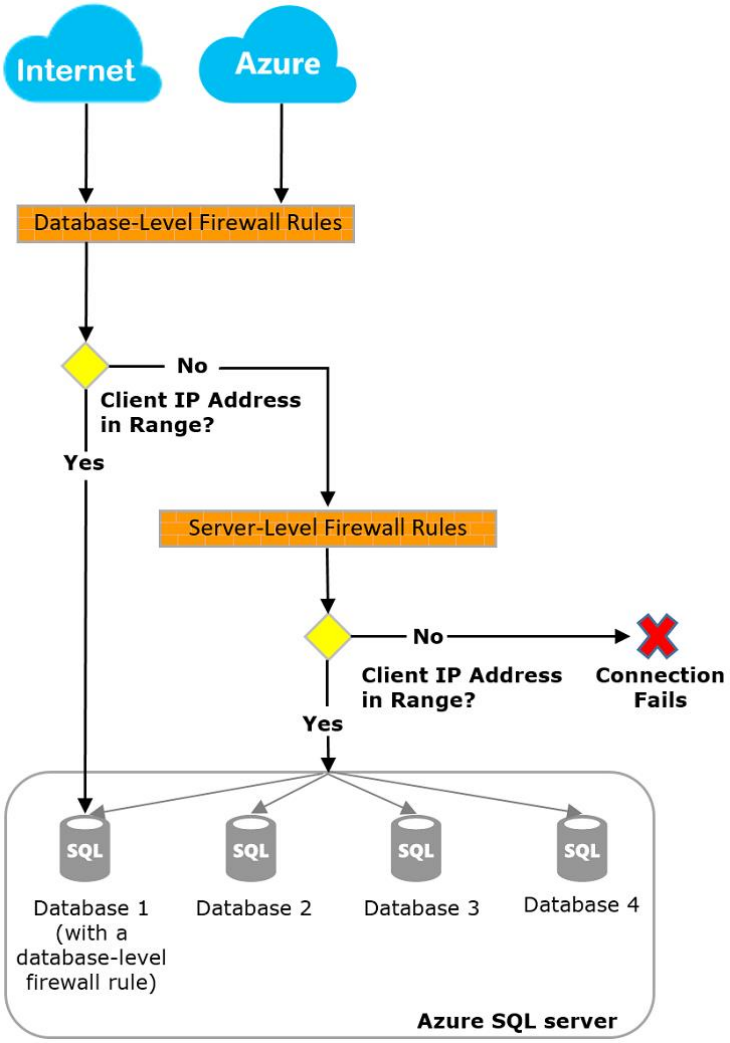
Ayrıca uygulamalarımızda kullanmamız için Azure bize 4 adet connection string'i vermektedir. Bunlar Ado.Net, JDBC, ODBC, PHP olarak ayrılmaktadır biz uygulamamızda daha çok Ado.Net bağlantı yöntemini bildiğim için ve veritabanıyla haberleşmede bu teknolojiyi kullandığım için bu string'i tercih ettim :

Server=tcp:forstaj.database.windows.net,1433;Initial Catalog=stajazure;Persist Security Info=False;User

ID={your_username};Password={your_password};MultipleActiveResultSets=False;
Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;

Yukarıda bağlantı türü tcp/ip şeklinde bağlanılacak server seçili katalog yani database, kullanıcı adı girilmesi gereken alan id, kullanıcı şifresinin girilmesi gereken alan Password yazan kısım, Encrypt şifrelemenin kullanıp kullanılmamasını, Connection Timeout ise kaç saniye içerisinde response yapılmazsa bağlantının kesileceği hakkında bilgi vermektedir.

2.3.1. Microsoft Azure User Oluşturma ve Veritabanı Güvenliği



Şekil 15 - Microsoft Azure Çalışma Mantığı

Yukarıda Sql Azure'a bağlanmak isteyen bir admin kullanıcısı veya sql user'ın aşması gereken aşamalar gözükmektedir. İlk başta database firewall rule'a bakılmaktadır. Bizde database'e bağlanmak için geçerli bir user ve password oluşturduk :

```
create user oguzDblU with password='oguzzZ99x5';     --user ekledik !
alter role db_datareader add member oguzDblU;
alter role db_datawriter add member oguzDblU;
```

ve Ayrıca mobilden bağlantı kuracağımız ve kuracağımızı bağlantının ip'si lokasyona göre değişeceği için firewall seviyesinde ip role'u ekledik:

```
EXECUTE sp_set_database_firewall_rule N'Example DB Rule','0.0.0.4','0.0.0.4';
```

Yukarıda name'i example olan bir ip range'i ekleyen store procedure çalıştırıldığı görülmektedir.

Bu şekilde ilgili veritabanımıza ulaşabilir ve gerekli read ve write işlemlerini uygulamamız bazında gerçekleştirebilriz. Ayrıca gerekli görülürse uygulama tarafında gerekli criptology yöntemleri kullanılarak veritabanına gönderilen bilgiler şifreli bir şekilde gönderilip yazılıp örneğin :

```
System.Object
   System.Security.Cryptography.HashAlgorithm
      System.Security.Cryptography.MD5
```

Sınıfları kullanılarak MD5 formatında ve okunabilir, biz projemizde gerekli görmediğimiz için bu tür bir şifreleme işlemi gerçekleştirmedik.

3. PROJE

Bu aşamamızda projemizin geliştirilmesi aşamasındaki yazılan kodların açıklaması ve diğer arayüz nesneleri kodları açıklanmıştır.

3.1. Projenin Oluşturulması ve İlk Aşama Bağlantı Nesnesinin Yazılması

İlk başta projemiz Visual Studio idesi açılarak New Project kısmından Android ve Blank Android projesi açılmış adına ise StajBilgisi denilmiştir. Sonra ilk olarak Bağlantı class'ımız oluşturulmuştur.

```
public class BaglantiSingleton //Adeta bir SingletonNesne olarak modelliyoruz
!
    {
        private static SqlConnection baglanti;


        public static SqlConnection Baglanti
        {
            get
            {
                if(baglanti==null)
                    baglanti= new
SqlConnection("Server=tcp:forstaj.database.windows.net,1433; Initial
Catalog=stajazure; Persist Security Info=False;User ID =oguzDblU;
Password=oguzzZ99x5; MultipleActiveResultSets=False; Encrypt=False; Connection
Timeout=60;");

                return baglanti;
            }
        }

    }
```

Yukarıda örnek bağlantı kodlarımız görülmektedir. Burada class'ımızın içerisinde bir adet static SqlConnection nesnesi tanımlanmaktadır static nesneler çalışma zamanında 1 kere yaratılır ayrıca biz bir Baglanti adında Metot yazarak her baglanti nesnesi istenildiğinde eğer yaratılmamışsa yaratıp onu return ettirerek projemizin ilgili yerlerinde kullanıyoruz.

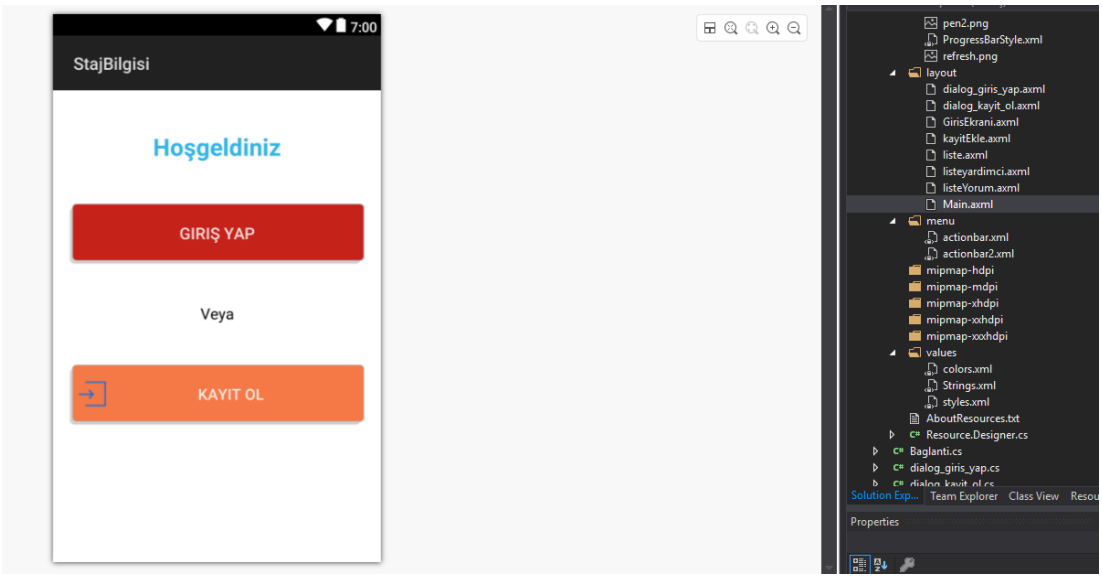
3.2. Projemizin İlk Sayfası Layout ve Activity Sınıfı

Her uygulamanın launcher olarak tanımlı bir Activity sınıfı olmalıdır. Bu şekilde tanımlanmış bir Activity, uygulama ilk açıldığında otomatik olarak harekete geçer ve kullanıcının karşısına çıkan ilk ekran olur. Activity dosyaları ilk çalıştıklarında onCreate metodu devreye girer. Bu metotta genel olarak setContentView metodu çalıştırılarak bir layout dosyasından ekran tasarımı yüklenir. Eğer ekran ilk oluştuğunda tanımlanması gereken başka değişkenler ve aksiyonlar varsa, onlar da onCreate metodu içinde gerçekleştirilebilirler.

```
namespace StajBilgisi
{
    [Activity(Label = "StajBilgisi", MainLauncher = true,Icon ="@drawable/pen2")]
    0 references
    public class MainActivity : Activity
    {
        SqlConnection con = BaglantiSingleton.Baglanti;


        private Button mBtnKaydol;
        private Button mBtnGiris;
        private ProgressBar mProgressBar;
        5 references
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);

            // Ana gösterilecek view'i aşağıda yazıyoruz !
            SetContentView(Resource.Layout.Main);

            //Aşağıda ise ana Main layout'umuzu ilgilendiren objeleri handle ettik !
            mBtnKaydol = FindViewById<Button>(Resource.Id.btnKayıtOl);
            mBtnGiris = FindViewById<Button>(Resource.Id.btnGiris);
            mProgressBar = FindViewById<ProgressBar>(Resource.Id.progressBar1);

            mBtnKaydol.Click += MBtnKaydol_Click;
            mBtnGiris.Click += MBtnGiris_Click;
        }
```

Şekil 16 - Projemizin Ilk Activity'i

Yukarıda projemizin ilk MainActivity class'ı görülmektedir, bu class Activity class'ından inherit edilmiştir ve projemizin çalışan ilk activitiy'sidir. Ayrıca ilk layout'umuzun Main adlı Layout olduğu gözükmektedir.



Şekil 17 - Giriş Axml Layout'u

Android uygulamalarında ekran tasarımları res(Resources) klasörü altında bulunan layout dosyaları ile belirlenir. Bu dosyalar normal xml formatında hazırlanan dosyalardır fakat bu xamarin'de axml formatında olmaktadır. Xml ile herhangi bir farklılığı yoktur.

Bir ekranın görünümü genelde iki farklı yerleşim tipi kullanılarak belirlenir. Bunlar RelativeLayout ve LinearLayout olarak ikiye ayrılır. LinearLayout yaklaşımı kullanıldığında öğeler sırayla ekrana dizilirler ve ekrandaki yerleşimleri ekranın en tepesinden en altına doğru sırayla gerçekleşir. Öğelerin uzunluk ve genişlikleri android:layout_width ve android:layout_height özellikleriyle belirlenir. Burada fill_parent (ana öğe ne kadar genişse hepsini doldurur) ve wrap_content (ana öğe içerisinde yer alan metin veya resim kadar yer kaplar) değerleri kullanılabileceği

gibi, pixel cinsinden uzunluk da verilebilir. Eğer pixel vererek ebat belirteceksek, px birimi yerine dp birimini kullanmamız faydalı olacaktır. Android cihazlarda farklı ebatlarda ekranlar olduğundan, px cinsinden verilen ekranların tasarımı her cihazda farklı oluşturulacaktır. Dp birimi ise ekran boyutuna göre ölçeklendirme yapmaktadır ve farklı cihazlarda benzer görünümler elde etmemizi sağlar (Turkcell, 2015).

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/background_light"
    android:weightSum="100"
    android:minWidth="25px"
    android:minHeight="25px">
    <TextView
        android:text="Hoşgeldiniz"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:layout_weight="20"
        android:id="@+id/txtHosgeldiniz"
        android:textColor="@android:color/holo_blue_light"
        android:gravity="center"
        android:textStyle="bold"
        android:textSize="28dp" />
    <Button
        android:text="Giriş Yap"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="15"
        android:id="@+id/btnGiris"
        android:background="@drawable/buttongirisstyle"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:textSize="19dp" />
    <TextView
        android:text="Veya"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="50dp"
```

Şekil 18 - Giriş. Axml Kodları

Android:id özelliği her öğeye bir tanımlayıcı verilmesini sağlar ve öğelere kaynak kod içinden erişmemize ve müdahale etmemize yardımcı olur. Burada yatılan

id değerleri Resource.Designer class'ında otomatik olarak oluşturulur ve bunlara kaynak kod içerisinden findViewById metodu kullanarak erişebiliriz.

Layout dosyalarında bir öğeye **id** atanırken **@+id** yazımı kullanılır. Bu şekilde öğelere kendi belirlediğimiz isimleri verebiliriz. **@id** ise layout dosyasında önceden tanımlanmış bir öğeye referans vermek için kullanılır. **@drawable**, **drawable** klasörüne atılan resim dosyalarına referans verir. Bu şekilde bir resim yerleştirmek ya da arka plan belirlemek istiyorsak kaynak belirtmek için **@drawable** yazımını kullanmamız gerekir. Bir düğmeye ya da metin öğesine **string.xml** dosyasında belirttiğimiz bir yazıyı koymak istiyorsak **@string** yazımıyla bu dosyada bulunan değerlere erişebiliriz. **@android** ile işletim sisteminde tanımlanmış değerlere erişebiliriz.

```
mBtnKaydol = FindViewById<Button>(Resource.Id.btnKayıtOl);
```

Yukarıda kayıtol butonunun id'sine erişimimiz görülmektedir.

3.3. Fragmentlerle Çalışma ve Projemizin Başka Sayfasına Geçiş

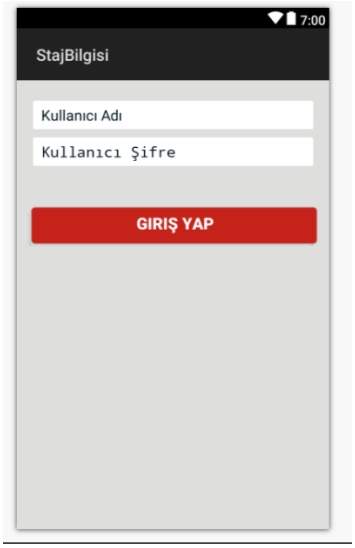
Kısacası fragmentleri tanıtmamız gerekirse Android'de çok bölmeli (multi-pane) dinamik arayüzler oluşturmak istediğimizde, Activity davaranışlarını ve arayüz bileşenlerini (liste, düğme vs.) Activity'nize girip çıkabilen modüller halinde tutmalısınız. Bu modülleri fragment sınıfı ile oluşturuyoruz. Fragmentler, yaşam döngülerini yönetebileceğiniz, özel layoutları tanımlanabilen iç içe geçmiş Activity'ler gibi davranır.

Biz burada kendi fragment class'ımızı yazarak bu işlemi gerçekleştirdik:

```
class dialog_giris_yap : DialogFragment
{
    private EditText mEdtKullaniciAdi;
    private EditText mEdtKullaniciSifre;
    private Button mBtnGirisYap;
    public event EventHandler<girisYapEventArgs> mGirisYapTamam; //Event'in kolaylığından faydalanıyoruz !

    1 reference
    public override View OnCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
    {
        base.OnCreateView(inflater, container, savedInstanceState);

        var view = inflater.Inflate(Resource.Layout.dialog_giris_yap, container, false); //Ilgili layout'u çalıştırmak için yazıyoruz ve alt
          tarafta return ettiriyoruz !
```

Şekil 19 - DialogFragment Class'ı

Yukarıda dialog_giris_yap class'ımız DialogFragment'den inherit edilerek yaratılmıştır ve fragmentlerde aslında birer layout'u temsil ettikleri için bir view yaratılıp ilgili layout'un temsili olan id'si atanmıştır.



Şekil 20 - Dialog Giriş Layout'u

Yukarıda görüldüğü üzere 2 tane edittext ve bir adet button control'u atanarak Source tarafında şekillendirilmiştir ve hepsine id verilmiştir. Şöyle demek gerekirse RelativeLayout görsel öğeleri diğer öğelere göre referans alarak dizer.

RelativeLayout tasarımında ilk eklenen öğe ekranın en tepesinde yer alır. Daha sonra eklenen öğeler android:layout_below (verilen ID'nin altında), android:layout_toLeftOf (verilen ID'nin solunda) ve android:layout_toRightOf (verilen ID'nin sağında) şeklinde sıralanır. Bu şekilde oluşturulan bütün öğeleri bir öğeyi referans alarak dizmemiz mümkündür. Yine aynı şekilde öğelerin ebatları ise android:layout_height ve android:layout_width metotlarıyla belirlenir.

```
public class girisYapEventArgs : EventArgs
    {
        private string mKullaniciAdi;
        private string mKullaniciSifre;

        public string KullaniciAdi { get=>mKullaniciAdi;
set=>mKullaniciAdi=value; } //C# 7.0 Prop kullanımı.
        public string KullaniciSifre { get=>mKullaniciSifre;
set=>mKullaniciSifre=value; }

        public girisYapEventArgs(string kullaniciadi,string kullanicisifre) :
base() //Bilindiği üzere javada super class ve base class diye ayrılan ayrımı
görüyoruz burada !
        {
            KullaniciAdi = kullaniciadi;
            KullaniciSifre = kullanicisifre;
        }

           }
```

Ayrıca yukarıda görüldüğü üzere fragment class'ımızda bir EventArgs oluşturulmuş böylece içerisine properties'ler yazılarak bu Event Sayesinde Fragment gelen kullanıcı adı ve şifre tutulmuş ve mainActivity'de kullanılmıştır. Aşağıda MainActivity'de kullanılan fragment kodları gözükmektedir :

```
private void MBtnGiris_Click(object sender, EventArgs e)
{
    FragmentTransaction transaction = FragmentManager.BeginTransaction();
    dialog_giris_yap giris_dialog = new dialog_giris_yap(); //Ilgili Class'ımızı kullanıyoruz !
    giris_dialog.Show(transaction, "Dialog Fragment2"); //Dialog'u ekranda göstermemizi sağlar !

    giris_dialog.mGirisYapTamam += Giris_dialog_mGirisYapTamam; //Hani event yazmıştıkya onun sayesinde butona tıklandığında ilgili propla
      değerini alabiliyoruz !
}

1 reference
private void Giris_dialog_mGirisYapTamam(object sender, girisYapEventArgs e)
{
    //Buraları yapmadan önce istersek System.Sql.Client manage packet'ini kullanabiliriz.
    try
    {
        con.Open();
        SqlCommand cmd = new SqlCommand("select uye_id from tbl_uye where kullanici_ad='" + e.KullaniciAdi.ToString() + "'and
          kullanici_sifre='" + e.KullaniciSifre.ToString() + " '", con);
        SqlDataAdapter sda = new SqlDataAdapter("select COUNT(*) from tbl_uye where kullanici_ad='" + e.KullaniciAdi.ToString() + "'and
          kullanici_sifre='" + e.KullaniciSifre.ToString() + " '", con);
        SqlDataAdapter ss = new SqlDataAdapter(cmd);
        DataTable dt = new DataTable();
        DataTable ds = new DataTable();
```

Şekil 21 - Fragment Kodları

Evet Yukarıda Fragmentin işlevi sona erdiğinde mGirisYapTamam Eventine gelen kullanıcı adı ve şifresi alınarak veritabanında gerekli check işlemleri yaptırılıp ilgili kullanıcı adı ve şifre mevcut ise sisteme girişi ve bir sonraki sonraki sayfaya geçişi sağlanmaktadır.

```
var intent = new Intent(this, typeof(GirisActivity));
                    var id = Convert.ToInt32(ds.Rows[0][0]);
                    intent.PutExtra("id", id);

                    StartActivity(intent); //Eğer açtığımız sayfaya
verilerimizi göndermek istiyorsak bir intent yapısı kullanmamız gerekiyor
diyebiliriz !

                    dt.Clear();
                    ds.Clear();
                    Finish();
```

Yukarıdaki kodlarda ise yeni bir intent oluşturulup bir sonraki geçilicek Activity class'ı GirisActivity şeklinde verilmiştir ve Intent.PutExtra kısmında ise veritabanında giriş yapan kullanıcının id'si alınıp bir sonraki Activity'e

gönderilmiştir. Kısacası Her Activity yeni arayüzler gösterirken belirli bir görevi yerine getirir, android uygulamalarında bir activity'i diğerine geçirmek için Intent sınıfını kullanırız ve yukarıdaki Kodlarda Finish() metodu ile ilgili Activity kill edilmiştir ve geri tuşu ile birdaha dönüş yoktur.

3.3.1. Fragmentlere Animasyon Verme

Birinci yol kendimiz bir style tanımlayıp bunu kullanabiliriz veya Android'in kendisinin bazı animasyonları olmaktadır. İlkini seçersek :

Kendimiz bir klasör açıp içerisine .xml tipinde dosyalar yazmamız gerekmektedir örneğin:

```
<?xml version="1.0" encoding="utf-8" ?>
<translate xmlns:android="http://schemas.android.com/apk/res/android"
           android:fromXDelta="0" android:toXDelta="-100%p"
           android:duration="500"/>
```

Yukarıda toXDelta fragment 'in gideceği kısmı, Ayrıca duration kısmı animasyonun kaç ms süreceğini belirtmektedir. Aynı şekilde bunun gibi başka birkaç tane .xml dosyası yazıp ekrana giriş ve çıkışlarını fragmentlerin ayarlayabiliyoruz.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<resources>
  <style name="dialog_animation">
    <item name="android:windowEnterAnimation">@anim/slide_down</item>
    <item name="android:windowExitAnimation">@anim/slide_left</item>
    <item name="android:windowSoftInputMode">adjustResize</item>
  </style>
  <style name="dialog_animation2">
    <item name="android:windowEnterAnimation">@anim/slide_up</item>
    <item name="android:windowExitAnimation">@anim/slide_right</item>
  </style>
  <style name="MyCustomTheme1" parent="android:Theme.Material.Light">
    <!-- Override the app bar color -->
    <item name="android:colorPrimary">@color/my_blue</item>
    <!-- Override the color of UI controls -->
    <item name="android:colorAccent">@color/my_purple</item>
    <item name="android:statusBarColor">@color/my_gray</item>
    <item name="android:navigationBarColor">@color/my_gray2</item>
  </style>
</resources>
```

Şekil 22 - Style Klasöründen Bir Görünüm

Yukarıda tanımlanmış xml dosyalarını baz alan style'lar gözükmektedir ve bunların name'ini

```
Dialog.Window.Attributes.WindowAnimations = Resource.Style.dialog_animation2;
//Görüldüğü üzere 2.Animasyonumuzu da set etmiş oluyoruz !
```

Bu şekilde fragment class'ımızın içerisinde kullandık.

3.4. Kurumların Listesini Görme ve Yorumlarını Okuma

Kurumların Listesini Görmek için İkinci Activity'mizden btnFirmalarıListele button'una bastıktan sonra listeActivity Activity'miz çalışmaktadır. Burada daha önce oluşturduğumuz stajmodel class'ından faydalanıyoruz.

```
namespace StajBilgisi
{
    16 references
    class stajmodel
    {
        5 references
        public int Yorumid { get; set; }
        7 references
        public string Kurumad { get; set; }
        6 references
        public string KurumDepartman { get; set; }
        6 references
        public string CalisanSayisi { get; set; }
        6 references
        public string KurumGorev { get; set; }
        6 references
        public string KurumAdres { get; set; }
        6 references
        public string Yorum { get; set; }
        7 references
        public DateTime YorumTarih { get; set; }
    }
}
```

Şekil 23 - Staj Class'ı Veritabanınıdaki Tabloyu Temsilen

Burada veritabanındaki tbl_icerik tablosunun bir temsili sayılabilecek stajmodel class'ımız bulunmaktadır. Bu class'ımızı listeActivity'de şu şekilde dolduruyoruz:

```
private void listeguncelle() //Oluşturduğumuz sjatmodel nesnesini ana ListView'e atamak için burayı yazdık !
{
    mStajmodels = new List<stajmodel>();
    SqlCommand cmd = new SqlCommand("select yorum_id,kurum_ad,kurum_departman,kurum_calisan,kurum_gorev,kurum_adres,kurum_yorum,tarih from
      tbl_icerik", con);
    con.Open(); //Aslında kontrol ettirmemiz gerekiyor !
    SqlDataReader dr = cmd.ExecuteReader();
    while(dr.Read())
    {
        mStajmodels.Add(new stajmodel() { Yorumid = (int)dr[0], Kurumad = (string)dr[1], KurumDepartman = (string)dr[2], CalisanSayisi = dr
          [3].ToString(),KurumGorev=(string)dr[4],KurumAdres=(string)dr[5],Yorum=(string)dr[6],YorumTarih=(DateTime)dr[7]});
    }
    dr.Close();
    con.Close();

    mAdapter = new listeAdapter(this, Resource.Layout.listeyardimci, mStajmodels); //listeyardimci layout'u bu şekilde bizim datamızı
      gösteriyor !
    mListView.Adapter = mAdapter; //Burasıda alıyor listeyardımcıyı liste.axml'de gosteriyor !
}
```

Şekil 24 - listeguncelle Methodumuz

Yukarıdaki kodlarımızda class'ımız verilerle doldurduk ve bir listeAdapter kullanarak listeyardimci layout'unu kullanarak datamızı gösterdik, ayrıca burada

dikkat edilmesi gereken listeguncelle() metodunun listeActivity class'ında gereken yerlerde kullanıldığıdır.

```
protected override void OnCreate(Bundle savedInstanceState)
{
    base.OnCreate(savedInstanceState);
    SetContentView(Resource.Layout.liste);
    mListView = FindViewById<ListView>(Resource.Id.listView); //Bu
    mArama = FindViewById<EditText>(Resource.Id.edtSearch); //Bu a
    mContainer = FindViewById<LinearLayout>(Resource.Id.llContaine
      kullanacağız !

    mArama.Alpha = 0; //Bu EditText'in invisible olmasını sağlar !
    mListView.ItemClick += MListView_ItemClick; //ItemClick olduğu
    mArama.TextChanged += MArama_TextChanged; //Bunu linQ kullanar
    listeguncelle();
```

Şekil 25 - Oncreate Metodu Görünüm

Görüldüğü üzere listeActivity'de sayfa ilk açıldığında gelen listview'in dolu olması için listeguncelle metodu ile ilk veriler veritabanından çekilmiştir. Buradaki başka bir event'imiz ise ListView'in ItemClick Event'idir, bu event sayesinde class'ımızdaki itemların birer position'ı vardır, bundan yararlanarak yeni bir activity açıp aşağıdaki yorum, adres, tarih bilgisi gönderilmektedir.

```
1 reference
private void MListView_ItemClick(object sender, AdapterView.ItemClickEventArgs e)
{
    var position = e.Position; //Item'ın class'da ilgili pozisyonun alıyor e.Position buda ItemClickEventArgs

    var gelenStajLine = mStajmodels[position];
    string Yorum = (gelenStajLine.Yorum).ToString();
    DateTime YorumTarih = (gelenStajLine.YorumTarih); //Veritabanında Datetime tipinde tutulduğu için burada
      gerekiyor.
    string Adres = (gelenStajLine.KurumAdres).ToString();
    string Tarih = (YorumTarih.Day, YorumTarih.Month, YorumTarih.Year).ToString();
    var intent = new Intent(this, typeof(listeYorumActivity));
    intent.PutExtra("Yorum", Yorum);
    intent.PutExtra("Adres", Adres);
    intent.PutExtra("Tarih", Tarih);
    StartActivity(intent);
    OverridePendingTransition(Android.Resource.Animation.FadeIn, Android.Resource.Animation.FadeOut);
}
```

Şekil 26 - ItemClick Event'i

Ayrıca bu Activitymizde liste layout'u temsil edilirken daha öncede belirttiğim gibi dinamik bir oluşum vardır. Örneğin ActionBar'ımız OnCreateOptionsMenu override edilerek oluşturulmuştur. Ayrıca burada yer alan EditText'imiz sayesinde listview itemlarımız içerisinde arama işlemini Linq aracılığıyla gerçekleştirmekteyiz vede search icon'una basıldığında mArama EditText'inin gözükmesi için bir animasyon işlemi gerçekleştiriyoruz:

```
1 reference
public override bool OnCreateOptionsMenu(IMenu menu) //Aynısı javada da var androidle uğraşanlar bilir. ActionBar visible hale geldi !
{
    MenuInflater.Inflate(Resource.Menu.actionbar, menu); //ActionBar'ın gözükmesi için handle ettik !
    return base.OnCreateOptionsMenu(menu);
    //Ama burası hala search icon'ını çalıştırmaz başka bir methodu da override etmemiz gerekir !
}
1 reference
```

Şekil 27 - ActionBar'ın Oluşturulması

Yukarıda Menümüzü oluşturmak için IMenu interface'inden yararlandık. Aşağıda ise yukarıda görülen actionbar'ın sırası ve png'leri koymamız için yazılan .xml dosyası gözükmektedir :

actionbar.xml | liste.axml | listeActivity.cs | slide_left.xml | dialog_giris_yap.cs | MainAc

```
1  <menu xmlns:android="http://schemas.android.com/apk/res/android">
2
3    <item android:id="@+id/search"
4          android:showAsAction="always"
5          android:title="search"
6          android:icon="@drawable/ic_action_search"/>
7    <item android:id="@+id/refreshh"
8            android:showAsAction="always"
9            android:title="Refresh"
10           android:icon="@drawable/refresh"/>
11 </menu>
12
```

Şekil 28 – ActionBarXml

Çıktısı Şu Şekildedir :

Ayrıca buradaki herbir icon'a tıkladığında çalışan kodlar :

```
public override bool OnOptionsItemSelected(IMenuItem item) //Ilgılı action itemları kullanmak içi
{
    switch (item.ItemId)
    {
        case Resource.Id.search: //Search'ın tıklandığı case olay durum ....
```

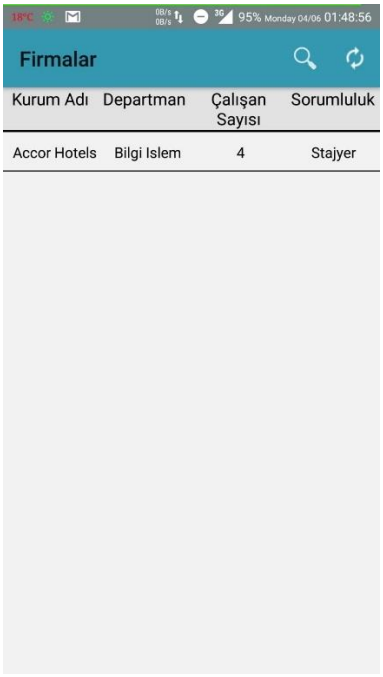
Şekil 29 - Case Search

Yukarıda switch case mantığı kullanılmıştır ve arama icon'una tıklandığında çalışan kodlar yer almaktadır, aşağıda ise refresh icon'ına basıldığında çalışan kodlar şu şekildedir:

```
 default:
     return base.OnOptionsItemSelected(item); //Return ile birşey döndürdüğümüz için break eklemiyoruz !

 case Resource.Id.refreshh: //Buda actionbar'da refresh simgesine basıldığında gerçekleşen olay !
     listeguncelle();

     return true;
//Switch burada bitiyor dikkat !
```

Şekil 30 - Case Refresh

Yukarıda ise refresh icon'ına basıldığında çalışan kodlar gözükmektedir. Son Görünüm şu şekildedir:



Şekil 31 - ListView ve ActionBar

Yukarıda Son Görünümümüz yer almaktadır. Ayrıca ilgili Item'a tıklandığında ise şu yorum sayfası gözükmektedir:

Şekil 32 - Yorum Sayfası

Buradaki Arama EditText'ine ve Yapılan Animasyon İşlemlerine sunum sırasında değinilecektir.

3.5. Kendi Yorumlarımızı Görme Ve Silme

Kendi Yorumlarımızı görüp silmek için ListeYorumGorSilActivity'sine gidilmiştir, burada yapılan işlem listeActivity'le neredeyse aynıdır fakat çekilen veriler, itemler bizim kendi yorumlarımızdır bunları veritabanında çekmek için giriş yapılan üyenin id'sine ihtiyaç vardır bizde bunu intent.PutExtra deyip gönderdiğimiz ve veritabanındaki yorum yapılan verileri ilişkili şekilde tuttuğumuz için şu şekilde verileri çekiyoruz:

```
4 references
private void listeguncelle() //Oluşturduğumuz sjatmodel nesnesini ana ListView'e atamak için burayı yazdık !
{
    if (con.State == ConnectionState.Closed)
    {
        mStajmodels = new List<stajmodel>();
        SqlCommand cmd = new SqlCommand("select yorum_id,kurum_ad,kurum_departman,kurum_calisan,kurum_gorev,kurum_adres,kurum_yorum,tarih from
          tbl_icerik " + "where uye_id='" + gelenidUye + "'", con);
        con.Open(); //Aslında kontrol ettirmemiz gerekiyor !
        SqlDataReader dr = cmd.ExecuteReader();
        while (dr.Read())
        {
            mStajmodels.Add(new stajmodel() { Yorumid = (int)dr[0], Kurumad = (string)dr[1], KurumDepartman = (string)dr[2], CalisanSayisi = dr
              [3].ToString(), KurumGorev = (string)dr[4], KurumAdres = (string)dr[5], Yorum = (string)dr[6], YorumTarih = (DateTime)dr[7] });
        }
        dr.Close();
        con.Close();
```

Şekil 33 – YorumGorSil

Yukarıda görüldüğü üzere where uye_id= gelenidUye şeklinde bir sorgu ile kendimize ait olan verileri çekeriz. Ayrıca burada yorumlarımızı silmek için yine class'ımızdaki position'dan yararlanılmış ve gerçek veritabanındaki verilerimiz

silinmiştir bunun içinde hem görsel açıdan hemde hızlı olması açısında ItemLongClickEvent'inden faydalanarak işlemlerimizi gerçekleştirdik:

```
private void MListView_ItemLongClick(object sender, AdapterView.ItemLongClickEventArgs e)
{
    var position = e.Position;
    var gelenStajLine = mStajmodels[position];
    var kurumAd = gelenStajLine.Kurumad.ToString();
    var kurumTarih = gelenStajLine.YorumTarih.ToShortDateString().ToString();

    AlertDialog.Builder alert = new AlertDialog.Builder(this); //Ekrana Display Alert Yaptırmamız için AlertDialog Class'ını kullanıyoruz !
    alert.SetTitle("Silmeyi Onayla");
    alert.SetMessage(kurumAd+" Adlı "+kurumTarih+" tarihli kaydınız silinecektir, emin misiniz ?");
    alert.SetPositiveButton("Evet", (senderAlert, args) => {
        try
        {
```

Şekil 34 - LongItemClick ve AlertDialog

Yukarıda görüldüğü üzere ListView'ımız bir event'i olan ItemLongClick'den faydalandık kullanıcıya görsel olarak bir dialog içinde sorumuzu sorduk ve silme işlemini onayladığında ilgili Yorum'un Yorum id'sinden yararlanarak ilgili kayıtı sildik. Yine buradaki Activity'mizde ek olarak bir thread işlemi gerçekleştirerek kullanıcıya verisinin silindiği görsel olarak gösterdik.

```
Thread thread = new Thread(ActLikeRequest); //Yine thread kullandık !
thread.Start();
```

Şekil 35 - Thread1

Silme işlemi gerçekleştiğinde thread çalışmaya başlasın ve 1200ms sonra ListView'imizi tekrardan listeguncelle metodundan yararlanarak görsel olarak güncellemektedir.

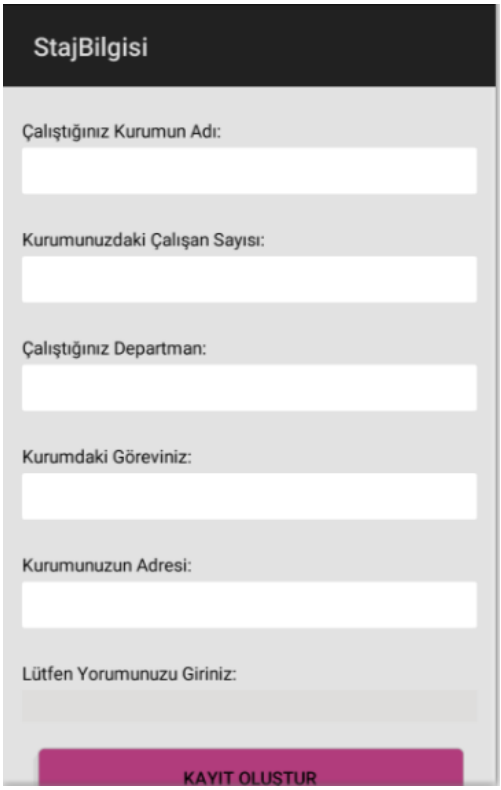
3.6. Yeni Bir Yorum Oluşturma Ve Veritabanına Ekleme

Yeni bir yorum oluşturmak için tasarladığımız görsel sayfanın kodları:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#E2E2E2">
<!--Hepsi görüldüğü üzere ScrollView içinde ScrollView'inde 1 tane parent'ı olabilir.-->
    <ScrollView
        android:minWidth="25px"
        android:minHeight="25px"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/scrollView1">
        <TableLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:stretchColumns="1">
            <TextView
                android:text="Çalıştığınız Kurumun Adı:"
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:textColor="#000"
                android:layout_marginLeft="15dp"
                android:layout_marginTop="25dp"
                android:id="@+id/textView1" />
            <EditText
                android:inputType="text"
```

Şekil 36 – ScrollView

Yukarıda kodlarımızda dikkatimiz çeken nokta sayfa controllerinin bir ScrollView Tag'i içerisinde yer aldığıdır, bunun nedeni sayfamızda çok fazla TextView, EditText gibi nesneler olursa telefonumuzun ekranının boyu yetmeyeceği için internet sitelerindeki gibi kaydırılabilir sayfa yapısı kullanılmıştır.



Şekil 37 - ScrollView Çıktısı

Yukarıda ScrollView sayfa yapısı içerisinde Xamarin Controllerimiz gözükmektedir. Bu sayfayı temsil eden kayitEkleActivity'mizin Kayıt Oluştur butonuna basıldığında şu kodlar çalışmaktadır:

```
1 reference
private void BtnKayitEkle_Click(object sender, EventArgs e)
{



    try
    {
        if (con.State == ConnectionState.Closed)
        {
            con.Open();
            if (mKurumAd.Text == "" || mKurumDepartman.Text == "" || mKurumCalisanSayisi.Text == "" || mKurumGorev.Text == "" ||
              mKurumAdres.Text == "" || mKurumYorum.Text == "")
            {
                Toast.MakeText(this, "Bütün formu eksiksiz doldurunuz ! ", ToastLength.Short).Show();
            }
            else
```

Şekil 38 - Kayıt Oluştur Butonu

İlk başta öncelikle EditText'lerin içinin boş olup olmadığı kontrol edilmektedir ve boş değilse kayıt işlemi gerekli Sql Sorguları kullanılarak gerçekleştirikmektedir.

```
string query = "insert into tbl_icerik
  (kurum_ad,kurum_departman,kurum_calisan,kurum_gorev,kurum_adres,kurum_yorum,uye_id,tarih)" + "values
  (@kurumad,@kurumdepartman,@kurumcalisan,@kurumgorev,@kurumadres,@kurumyorum,@uyeid,@tarih)";
SqlCommand cmd = new SqlCommand(query, con);

cmd.Parameters.Add("@kurumad", SqlDbType.NVarChar, 500).Value = mKurumAd.Text.ToString();
cmd.Parameters.Add("@kurumdepartman", SqlDbType.NVarChar, 500).Value = mKurumDepartman.Text.ToString();
cmd.Parameters.Add("@kurumcalisan", SqlDbType.Int).Value = Convert.ToInt32(mKurumCalisanSayisi.Text);
cmd.Parameters.Add("@kurumgorev", SqlDbType.NVarChar, 500).Value = mKurumGorev.Text.ToString();
cmd.Parameters.Add("@kurumadres", SqlDbType.NVarChar, 600).Value = mKurumAdres.Text.ToString();
cmd.Parameters.Add("@kurumyorum", SqlDbType.NVarChar, 1000).Value = mKurumYorum.Text.ToString();
cmd.Parameters.Add("@uyeid", SqlDbType.Int).Value = Convert.ToInt32(Intent.GetIntExtra("idno", 0));
cmd.Parameters.Add("@tarih", SqlDbType.DateTime).Value = zaman;
int check = cmd.ExecuteNonQuery();
if (check != 0)
{
    Toast.MakeText(this, "Kayıt Başarılıdır...", ToastLength.Long).Show();
}
```

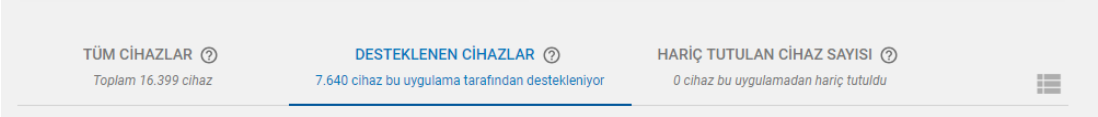
Şekil 39 - Sql Sorgusu ve ToastMakeText

Yukarıdaki Sql Sorguları çalıştırılarak doğru veri türü doğru şekilde veritabanına aktarılmıştır ve Toast Class'ıyla Javadan aşina olduğumuz kullanıcıya yapılan kayıt ekleme işleminin başarılı olup olmadığı görsel şekilde bildirilmiştir.

4. SONUÇ

Projemizin geliştirme süresi boyunca gerekli dökümantasyonlar incelenip, birçok web sitesinden yararlanılmıştır ve bunlar kaynak bölümünde belirtilmiştir, ayrıca yapılan testler ile Google Play aracılığıyla 16.399 cihazdan 7.640 cihazın desteklediği gözükmektedir, buda Xamarin'in ne kadar Native bir dil olduğunu ve özellikle Xamarin.Forms'un şirketler tarafından sıkça kullanılmaya başlanması tek

katmanda kod yazıp hem IOS hemde Android tarafında çalıştırılmasının önünü açmıştır.



Şekil 40 - Desteklenen Cihazlar

KAYNAKÇA

Bozkurt, S., 2011. *C Nedir ve C Temelleri Nedir.* [Çevrimiçi]

Kullanılabilir: http://www.teknokoliker.com/2011/11/c-nedir-c-temelleri-nelerdir.html

[Erişildi: 28 Mayıs 2018].

Microsoft, 2018. *Microsoft Azure.* [Çevrimiçi]

Kullanılabilir: https://azure.microsoft.com/tr-tr/overview/what-is-azure/

[Erişildi: 27 05 2018].

Turkcell, 2015. *Turkcell Geleceği Yazanlar.* [Çevrimiçi]

Kullanılabilir: https://gelecegiyazanlar.turkcell.com.tr/konu/android/egitim/android-201/layout

[Erişildi: 03 Haziran 2018].

Wikipedia, 2016. *Wikipedia Tümleşik Geliştirme Ortamı.* [Çevrimiçi]

Kullanılabilir: https://tr.wikipedia.org/wiki/Tümleşik_geliştirme_ortamı

[Erişildi: 27 Mayıs 2018].

Wikipedia, 2017. *Wikipedi.* [Çevrimiçi]

Kullanılabilir: https://tr.wikipedia.org/wiki/Language_Integrated_Query

[Erişildi: 27 Mayıs 2018].

Yıldız, M., 2017. *Xamarin Nedir.* [Çevrimiçi]

Kullanılabilir: http://mehmetx.com/Makaleler/Xamarin-Nedir-Xamarin-Serisi-Bolum-1/174

[Erişildi: 1 Haziran 2018].