**ALIŞVERİŞ MERKEZİ (AVM) YÖNETİMİ OTOMASYON PROGRAMI SHOPPING CENTER MANAGEMENT AUTOMATION PROGRAM**

*BUSE YENER, NİSA AKSOY, MEHMET SEZER*

BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ

KOCAELİ ÜNİVERSİTESİ

*busee.yener@gmail.com*

BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ

KOCAELİ ÜNİVERSİTESİ

*nisaaksoy08@gmail.com*

BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ KOCAELİ ÜNİVERSİTESİ

*211307027@kocaeli.edu.tr*

# Özet

*Bu dökümanda alışveriş merkezi yönetimi otomasyon programından, bu programda yer alan ana menüde ulaşabildiğimiz haritaya, ürünlere, geliştiricilere ve ayarlara değindik. Bu uygulamayı oluştururken esas aldığımız normalizasyon tablolarını ve MVVM kalıbını da ekledik.*

# Abstract

*In this document, we mentioned the shopping center management automation program, the map that we can access in the main menu of this program, the products, the developers and the settings. We also added the normalization tables and the MVVM pattern that we used to create this application.*

# 1.Giriş

Uygulamamızı yaparken Nesne Yönelimli Programlama, XAML ve C# dilinden faydalandık. Peki NYP, XAML ve C# nedir,ne işe yarar?

Nesne Yönelimli Programlama (Object Oriented

Programming),mantıksal işlemlerden ziyade, sınıflara, nesnelere (object) ve nesneler üzerinde işlemlere dayanan bir programlama dili modelidir. Amacı, ihtiyaç duyulan programı daha küçük parçalara bölerek bu parçaları yönetilebilir ve yeniden kullanılabilir hale getirmektir. Programlar nesnelerin birbirleriyle etkileşime geçmelerini sağlamak için tasarlanır. Nesne yönelimli programlama dilleri yüksek seviyeli diller olarak adlandırılır. Bu dillere örnek olarak; Java, C++, C#, Python, PHP, JavaScript, Ruby, Perl, Smalltalk, Objective-C sayabiliriz. Backend kısmında kullandığımız C# yani diğer bir adıyla C Sharp, Microsoft tarafından geliştirilen sunucu ve gömülü sistemleri çalıştırmak için tasarlanmış programlama dilidir. C# ile farklı uygulamalar ve programlar oluşturulabilir. Masaüstü, mobil, web ve oyun geliştirmede kullanılır. .NET Framework ortamında kullanılmak üzere geliştirilmiş nesne tabanlı programlama dillerinden birisidir.

C# ile uygulama geliştirilirken kullanılan platform Visual Studio IDE’dir. Visual Studio birçok işletim sistemlerinde editör olarak kullanılmaktadır. Web tasarım süreçlerinin önemli kısımlarından biri olan Back-end geliştirmede C# .NET çerçevesi ile birlikte çok yaygın bir kullanıma sahiptir. C# dili ile oluşturulan web hizmetleri hızlı ve güvenilirdir.

Frontend kısmında kullandığımız XAML(Extensible

Application Markup Language) diğer bir adıyla Genişletilebilir Uygulama İşaretleme Dili, Microsoft’un GUI tarafını temsil eden XML türevidir. XML tabanlı çalışır ve XML’nin özelliklerini kullanır. Bildirim tabanlı XAML işaretleme dilinde görünür UI ögeleri oluşturabilir ve kısmi sınıf tanımları aracılığıyla işaretlemeye katılmış arka planda kod dosyalarını kullanarak arabirimi tanımını çalışma zamanından ayırabiliriz. Xaml Phone 8, Phone

8.1, Windows 8 ve Windows 10 Metro uygulaması veya bir web sayfasında arayüz tasarımı sağlamak için geliştirilmiştir. Xaml dosya uzantısı \*.xaml'dir.

Xaml işaretleme dili ile geliştirme yapmanızı sağlayan ortamlardan biri, Visual Studio ortamındaki WPF

Application'dır.

Alışveriş merkezi yönetimi otomasyon programını yaparken MySQL Workbench ve SQL dilini kullandık.

SQL (Structured Query Language) Türkçe karşılığı Yapılandırılmış Sorgu Dili, birçok veriyi farklı boyutlarda barındıran veritabanlarını depolamak ve yönetmek için kullanılan bir programlama dilidir. Verilere ulaşmak, tasarlamak, güncellemek, tanımlamak, değiştirmek, kaldırmak gibi işlemlerde kullanılır. Genellikle ilişkisel veritabanında tercih edilir.

MySQL yapısı ismini SQL ve Michael Widenius’un kızının adının(My) birleşiminden alır. Bu yapı ilk kez 1995 yılında kullanıma sürülmüştür. Web tasarım dünyasında kulllanılan en popüler açık kaynaklı bir ilişkisel veritabanı yönetim sistemi olan bu yapı, verileri web üzerinde hızlı bir şekilde depolamak ve işlemek isteyen kişiler için tasarlanmıştır. MySQL C/C++ programlama dillerini kullanır.

MySQL Workbench SQL dilinde yazılan sorguları çalıştırdığımız görsel bir veritabanı tasarım aracıdır. Oracle tarafından geliştirilmiştir.

# 2.İlişkisel Veritabanı Yönetim Sistemleri

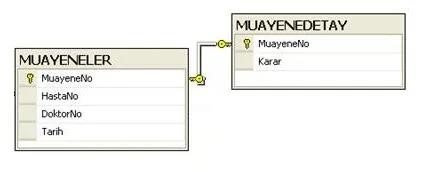
İngilizce karşılığı Relational Database Management olan İlişkisel Veritabanı Yönetim Sistemi, hızlı erişim için tasarlanan birbiriyle ilişkili veri noktalarını depolayıp düzenleyen ve bu veriler arasında bağlantı kurularak tasarlanmış bir veritabanı modelidir. Bu veritabanında, satır ve sütunların meydana getirdiği tablolar birbirlerine anahtar olarak adlandırdığımız verilerle bağlıdır. İlişkisel model aynı anda birçok tablodan veri çekilmesini sağlar. Aynı zamanda verilerin herhangi bir sorgulamada birlikte görüntülenmesini de sağlar.. İlişkisel veritabanlarına dBase, Informix, Ingres, MySQL, Oracle ve PostgreSQL örnek verilebilir

## 2.1 Varlık İlişki Modeli

Varlık İlişki Modeli ilk kez 1976 yılında ortaya atılmıştır. Bu model gerçek hayattaki nesneler ve aralarındaki ilişkilerini betimleyen bir modelleme türüdür. Genelde veritabanı oluşturmak amacıyla kullanılır.

### 2.1.1 Bire bir İlişki Modeli

Bire Bir İlişki Modeli, nadir olarak tercih edilir. İki tablo arasındaki ilişkiyi sağlamak için kullanılan bu tipin amacı sol tablodan gelen kaydın sağ tablodaki sadece bir kayıt ile eşleştirilmesidir.



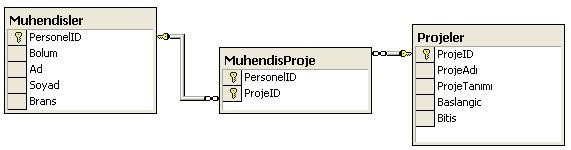
### 2.1.2 Bire çok İlişki Modeli

Bire Çok İlişki Modeli, en çok tercih edilen modeldir. Bu ilişki tipi sol tablodan gelecek bir kaydın sağ tablodaki birden fazla kayıtla eşleştirilmesidir.



### 2.1.3 Çoka çok İlişki Modeli

Çoka Çok İlişki Modeli, karmaşıklığı sebebiyle veritabanında birçok soruna yol açar. Her iki tabloda eşleşecek olan veriler ara bir tabloda tutularak üçüncü bir tabloda birleştirilir.



## 2.2 Normalizasyon

Normalizasyon diğer adıyla Ayrıştırma, veritabanlarında çok fazla satır ve sütundan oluşan bir tablonun içeriğini organize etmek, veri tekrarını en aza indirgemek ve bu tabloyu tekrarlardan arındırmak için daha az satır ve sütunlara ayırma işlemidir. 2 temel amacı vardır: Veritabanındaki tekrarları yok etmek ve Veri doğruluğunu artırmak. Veritabanlarına normal formlar olarak uygulanır. Başarılı uygulanan normalizasyon veritabanının sabit diskteki boyutunu azaltır.



### 2.2.1 1NF (1.Normal Form)

Bir veri tabanının 1NF olabilmesi için şu özellikleri karşılayabilmesi gerekir:

* Aynı tablo içinde tekrarlayan kolonlar bulunmamalıdır.
* Her kolonda yalnızca bir değer bulunabilir.
* Her satır bir eşsiz anahtarla tanımlanmalıdır.

### 2.2.2 2NF (2.Normal Form)

Bir veri tabanının 2NF olabilmesi için şu özellikleri karşılayabilmesi gerekir:

* Tablo 1NF olmalıdır.
* Anahtar olmayan değerler ile bileşik anahtarlar arasında kısmi bağımlılık durumu oluşmamalıdır. Kısmi bağımlılık durumu, anahtar olmayan herhangi bir değer bileşik bir anahtarın yalnızca bir kısmına bağlı ise oluşur.
* Herhangi bir veri alt kümesi birden çok satırda tekrarlanmamalıdır. Yeni tablolar oluşturulmalıdır.
* Ana tablo ile yeni tablolar arasında, dış anahtarlar (foreign key) kullanılarak ilişkiler tanımlanmalıdır.

### 2.2.3 3NF (3.Normal Form)

Bir veri tabanının 3NF olabilmesi için şu özellikleri karşılayabilmesi gerekir: • Veri tabanı 2NF olmalıdır.

• Anahtar olmayan hiç bir kolon bir diğerine bağlı olmamalı ya da geçişken fonksiyonel bir bağımlılığı olmamalıdır. Her kolon eşsiz anahtara tam bağımlı olmak zorundadır.

### 2.2.4 3.5NF (3.5 Boyce-codd Normal Form)

Bir veri tabanının 3.5NF olabilmesi için şu özellikleri karşılayabilmesi gerekir:

* Veri Tabanı 3NF olmalıdır.
* Her belirleyici kolon aynı zamanda bir aday anahtar olmalıdır.

### 2.2.5 4NF (4.Normal Form)

Bir veri tabanının 4NF olabilmesi için şu özellikleri karşılayabilmesi gerekir:

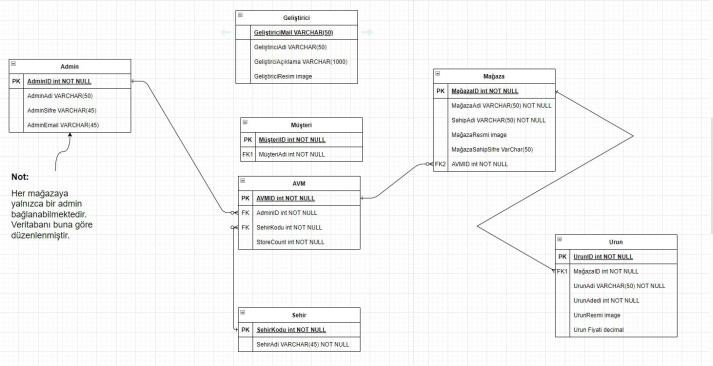
* Veri Tabanı 3NF olmalıdır.
* Çok-değerli bağımlılıkları olmamalıdır.

### 2.2.6 5NF (5.Normal Form)

Bir veri tabanının 5NF olabilmesi için şu özellikleri karşılayabilmesi gerekir:

* Veriler, projeksiyon birleştirme yöntemi ile elde edilir.
* Multivalue bağımlılıkları ortadan kaldırır.

# 3.Varlık-İlişki Diyagramı

Bir varlık-ilişki diyagramı , bir bilgi sisteminin varlıklarını ve bu varlıklar arasındaki ilişkileri grafiksel olarak gösteren bir veri modelleme tekniğidir. Varlık-İlişki diyagramı , varlık çerçevesi altyapısını temsil etmek için kullanılan kavramsal ve temsili bir veri modelidir.

Oluşturduğumuz tabloların önce özelliklerinden sonra da varlık ilişki türlerinden bahsedeceğiz.

1. olarak Admin tablomuzun özelliklerini sıralayalım:
   * AdminID adında int veri tipinde bir değişkeni ve bu değişkenin NOT NULL(geçersiz değil) özelliği bulunuyor.
   * AdminAdi adında VARCHAR(50) veri tipinde bir değişkeni bulunuyor.
   * AdminSifre adında VARCHAR(45) veri tipinde bir değişkeni bulunuyor.
   * AdminEmail adında VARCHAR(45) veri tipinde bir değişkeni bulunuyor.
2. olarak Geliştirici tablomuzun özelliklerini sıralayalım:
   * GeliştiriciMail adında VARCHAR(50) veri tipinde bir değişkeni bulunuyor.
   * GeliştiriciAdi adında VARCHAR(50) veri tipinde bir değişkeni bulunuyor.
   * GeliştiriciAçıklama adında VARCHAR(1000) veri tipinde bir değişkeni bulunuyor.
   * GeliştiriciResim adında image veri tipinde bir değişkeni bulunuyor.
3. olarak Müşteri tablomuzun özelliklerini sıralayalım:
   * MüşteriID adında int veri tipinde bir değişkeni ve bu değişkenin NOT NULL(geçersiz değil) özelliği bulunuyor.
   * MüşteriAdi adında VARCHAR(50) veri tipinde bir değişkeni ve bu değişkenin NOT NULL(geçersiz değil) özelliği bulunuyor bulunuyor.
4. olarak AVM tablomuzun özelliklerini sıralayalım:
   * AVMID adında int veri tipinde bir değişkeni ve bu değişkenin NOT NULL(geçersiz değil) özelliği bulunuyor.
   * AdminID adında int veri tipinde bir değişkeni ve bu değişkenin NOT NULL(geçersiz değil) özelliği bulunuyor.
   * SehirKodu adında int veri tipinde bir değişkeni değişkeni ve bu değişkenin NOT

NULL(geçersiz değil) özelliği bulunuyor.

* + StoreCount adında int veri tipinde bir değişkeni ve bu değişkenin NOT NULL(geçersiz değil) özelliği bulunuyor.

1. olarak Sehir tablomuzun özelliklerini sıralayalım:
   * SehirKodu adında int veri tipinde bir değişkeni ve bu değişkenin NOT NULL(geçersiz değil) özelliği bulunuyor.
   * SehirAdi adında VARCHAR(45) veri tipinde bir değişkeni ve bu değişkenin NOT

NULL(geçersiz değil) özelliği bulunuyor.

1. olarak Mağaza tablomuzun özelliklerini sıralayalım:
   * MağazaID adında int veri tipinde bir değişkeni ve bu değişkenin NOT NULL(geçersiz değil) özelliği bulunuyor.
   * MağazaAdi adında VARCHAR(50) veri tipinde bir değişkeni ve bu değişkenin NOT

NULL(geçersiz değil) özelliği bulunuyor.

* + SahipAdi adında VARCHAR(50) veri tipinde bir değişkeni değişkeni ve bu değişkenin NOT NULL(geçersiz değil) özelliği bulunuyor.
  + MağazaResmi adında image veri tipinde bir değişkeni bulunuyor.
  + MağazaSahipSifre adında VARCHAR(50) veri tipinde bir değişkeni bulunuyor.
  + AVMID adında int veri tipinde bir değişkeni ve bu değişkenin NOT NULL(geçersiz değil) özelliği bulunuyor.

1. olarak Urun tablomuzun özelliklerini sıralayalım:
   * UrunID adında int veri tipinde bir değişkeni ve bu değişkenin NOT NULL(geçersiz değil) özelliği bulunuyor.
   * MağazaID adında int veri tipinde bir değişkeni ve bu değişkenin NOT NULL(geçersiz değil) özelliği bulunuyor.
   * UrunAdi adında VARCHAR(50) veri tipinde bir değişkeni değişkeni ve bu değişkenin NOT NULL(geçersiz değil) özelliği bulunuyor.
   * UrunAdedi adında int veri tipinde bir değişkeni bu değişkenin NOT NULL(geçersiz değil) özelliği bulunuyor.
   * UrunResmi adında image veri tipinde bir değişkeni bulunuyor.
   * UrunFiyati adında decimal veri tipinde bir değişkeni bulunuyor.

## 3.1. Tablolar Arası İlişkiler

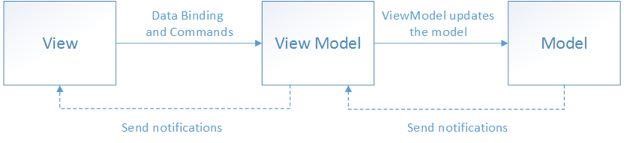
* Admin tablosu ile AVM tablosu arasında, BireÇok ilişki türü vardır.
* Sehir tablosu ile AVM tablosu arasında, BireÇok ilişki türü vardır.
* AVM tablosu ile Mağaza tablosu arasında, BireÇok ilişki türü vardır.
* Mağaza tablosu ile Urun tablosu arasında, BireÇok ilişki türü vardır.

Yukarıda da gördüğümüz gibi bire-çok ilişki türünde, bir tablodaki bir satır veri diğer tablodaki birden fazla satırlık veriye denk gelir.

# 4.MVVM Modeli

Model-View-ViewModel (MVVM) düzeni, bir uygulamanın iş ve sunum mantığını kullanıcı arabiriminden temiz bir şekilde ayırmaya yardımcı olur.

Uygulama mantığı ve kullanıcı arabirimi arasında temiz bir ayrım yapmak, çok sayıda geliştirme sorununun giderilmesine yardımcı olur ve uygulamanın testini, bakımını, gelişmesini kolaylaştırabilir. Ayrıca kodun yeniden kullanım fırsatlarını büyük ölçüde geliştirebilir. MVVM, Model, View ve ViewModel olmak üzere üç temel kısımdan oluşmaktadır. Her biri ayrı bir amaç için çalışır.



## 4.1 View (Görünüm)

Görünüm, verilerimizi son kullanıcılara aktardığımız görsel arayüzdür. Kullanıcının ekranda gördüklerinin yapısını, düzenini ve görünümünü tanımlamakla sorumludur. Son kullanıcı ile uygulama arasında köprü görevi görür. Her görünümün iş mantığı içermeyen sınırlı bir arka planda kod ile XAML'de tanımlanması idealdir.

Projemizin MVVM yapısındaki view bölümünün örneğinde ise seçilen butonun görseli ve kodlarını aşağıda görmekteyiz.



<Button Content="Mehmet Sezer"

Command="{Binding dev1Command}"

Style="{DynamicResource LightButton}"

Height="60" Margin="10"

HorizontalAlignment="Stretch"/>

## 4.2 View Model (Görünüm Modeli)

ViewModel, Model ile View'ı bağlayan kısımdır. View doğrudan ViewModel yardımıyla Model'e erişim sağlar ve bazı işlemleri gerçekleştirir. Görünüm modelinin sağladığı özellikler ve komutlar, kullanıcı arabirimi tarafından sunulacak işlevselliği tanımlar. Görünüm modeli, görünümün etkileşimlerini gerekli olan tüm model sınıflarıyla koordine eder. Görünüm modeliyle model sınıfları arasında genellikle bire çok ilişki türü vardır. Görünüm modeli, görünümdeki denetimlerin verilere doğrudan bağlanabilmesi için model sınıflarını doğrudan görünüme gösterebilir. Bu durumda model sınıflarının veri bağlamayı destekleyecek ve bildirim olaylarını değiştirecek şekilde tasarlanması gerekir.

Projemizin MVVM yapısında Developer Model'in ve View'in birleştirildiği ViewModel Sınıfında bulunan örnek kodlar aşağıda mevcuttur.

public DeveloperModel CurrentDevModel

{

get { return \_currentDevModel; }

set { \_currentDevModel = value; OnPropertyChaned();}

}

#endregion

#region RelayCommand public RelayCommand dev1Command { get; set; } public RelayCommand dev2Command { get; set; } public RelayCommand dev3Command { get; set; }

#endregion public DeveloperViewModel()

{

IDeveloperRepository devRepo = new DeveloperRepository();

CurrentDevModel =

devRepo.GetByMail("mehmetszr05@gmail.com"); dev1Command = new RelayCommand(O => {

CurrentDevModel =

devRepo.GetByMail("mehmetszr05@gmail.com"); });

dev2Command = new RelayCommand(O => {

CurrentDevModel =

devRepo.GetByMail("buseynr@gmail.com"); });

dev3Command = new RelayCommand(O => {

CurrentDevModel =

devRepo.GetByMail("nisaksy@gmail.com"); });

}

## 4.3 Model (Modelleme)

Veritabanından, web servislerinden, herhangi bir veri kaynağından gelen verilerimizi temsil etmek için, uygulama içerisinde de kullanılan bu yapı POCO veya entity sınıflarından oluşmaktadır. Veri tutarlığını ve doğruluğunu kontrol eden iş kuralları da burada yer almaktadır.Bu model WCF RIA Services ile döndürülen entityler olabileceği gibi doğrudan tanımlanmış POCO nesneleri de olabilir.Model sınıfları genellikle veri erişimini ve önbelleğe almayı kapsülleyen hizmetler veya depolarla birlikte kullanılır.

Projemizin MVVM yapısında bulunan DeveloperModel sınıfının örnek kodları aşağıda mevcuttur.

using System; using System.Windows.Media; using System.Windows.Media.Imaging; namespace MallMan.MVVM.Model

{ public class DeveloperModel

{

private string \_developerMail; private string \_developerName; private string \_developerCV; private ImageSource \_developerImage;

public string DeveloperMail

{

get { return \_developerMail; } set { \_developerMail = value; }

}

public string DeveloperName

{

get

{ return \_developerName; } set { \_developerName = value;}

}

public string DeveloperCV

{

get { return \_developerCV; } set { \_developerCV = value; }

}

public ImageSource DeveloperImage

{ get

{

if (\_developerImage == null)

{

return new BitmapImage(new Uri(@"C:\Users\mehme\source\repos\MallMan\MallMan

\ExampleImage\developerImageNotFound.jpg"));

} else return \_developerImage;

}

set { \_developerImage = value; }

}

}

}

# Projede Karşılaşılan Sorunlar

## Programlama Dili

Projemizi yaparken nesne tabanlı olmasından dolayı backend kısmında C# tercih ettik. Geçen yıl öğrenmiş olduğumuz bu dil ile farklı uygulamalar ve programlar oluşturulabiliyoruz. Ayrıca masaüstü, mobil, web ve oyun geliştirmede de kullanabiliyoruz. Frontend kısmında kullandığımız işaretleme dili XAML’dir. Microsoft’un GUI tarafını temsil eden XML türevidir. XML tabanlı çalışır ve XML’nin özelliklerini kullanır.

## Platform

C# dilini seçtiğimiz için geliştiricisi olan Visual Studio kullandık. Visual Studio birçok işletim sistemlerinde editör olarak kullanılabiliyor.

## Veri Tabanı Yönetim Sistemi

İlişkisel Veritabanı Yönetim Sistemi kullandık. Tablolar arası ilişkileri şematize ettik. Satır ve sütunlardan oluşan tablolar işimizi kolaylaştırdı.

SQL sorgularını çalıştırmak için ise MySQL Workbench kullandık.

## Mimari

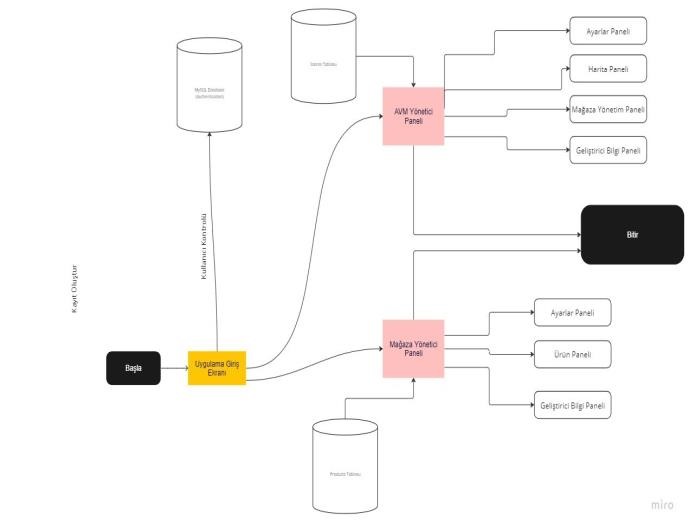
Mimari yapımızda ise MVVM Modeli kullandık. 3 kısımdan oluşması, günümüzde hem Google hem de Microsoft tarafıından desteklenmesi tercih etme sebeplerimizdendir. Seperation of concerns denilen, bir bilgisayar programını farklı bölümlere ayıran bu ilkeye uygundur. Her bölümde, bir bilgisayar kodunu etkileyen bir dizi bilgi içeren ayrı bir konuyu ele alır.

# Akış Şeması

Akış şeması, bir sürecin, işlemlerin veya bir programın adımlarını, kararlarını ve veri akışını gösteren grafiksel bir temsilidir.Bir problemin çözümünü açıklamak, bir iş sürecini analiz etmek veya bir programın mantığını göstermek için kullanılır.

## Projenin Akış Şeması

Aşağıda bulunan görselde projemizin akış şeması eklenmiştir.



# KAYNAKÇA

[1.https://www.tercihyazilim.com/blog/c-sharp-nedir](https://www.tercihyazilim.com/blog/c-sharp-nedir)

2.https://www.webtekno.com/c-sharp-nedir-c-sharpprogramlama-dili-h80441.html

3.https://tr.wikipedia.org/wiki/SQL

4.https://veriakademi.com/sql-nedir

5.https://aws.amazon.com/tr/what-is/sql/

6.https://www.oracle.com/tr/database/what-is-database/

7.https://www.oracle.com/tr/database/what-is-arelational-database/

8.https://www.natro.com/blog/mysql-server-nedir-nasilkullanilir/

9.https://azure.microsoft.com/tr-tr/resources/cloudcomputing-dictionary/what-is-a-relationaldatabase/

10.https://tr.wikipedia.org/wiki/%C4%B0li%C5%9Fkisel \_veri\_taban%C4%B1\_y%C3%B6n

etim\_sistemi

11.https://www.sqlekibi.com/guncel-icerikler/veritabani-yonetim-sistemi-dbms-ve-iliskiselveri-tabanirdbms-ne-demektir.html/

12.https://azure.microsoft.com/tr-tr/resources/cloudcomputing-dictionary/what-is-arelationaldatabase/#whatis

13.https://kevserkose.wordpress.com/2017/05/26/varlikiliski-modeli-entity-relationshipmodel/

14.http://web.hitit.edu.tr/dosyalar/materyaller/skevserkav uncu@hititedutr260220172F9Q1A2C.pdf

15.https://tr.wikipedia.org/wiki/Varl%C4%B1kba%C4%

9F%C4%B1nt%C4%B1\_modeli#:~:text=Varl%C4%B1k

%2Dba%C4%9F%C4%B1nt%C4%B1%20modeli%20(

%C4%B0ngilizce%3A,Genellikle%20veritaban%C4%B 1%20olu%C5%9Fturmak%20i%C3%A7in%20kullan% C4%B1lmaktad%C4%B1r.

16.https://medium.com/@guvengulec/veritaban%C4%B1 -i%CC%87li%C5%9Fki-tipleri7f93e3cbf3eb

17.https://www.iakademi.com/veritabaninormalizasyonu-ve-iliski-turleri/

18.https://www.dijitalders.com/icerik/13/2623/veri\_taban inda\_iliski\_turleri.html

19.https://medium.com/turuncu-internetsolutions/normalizasyon-nedir-f212541f15fb

20.http://www.baskent.edu.tr/~gmemis/courses/introducti ontodatabase/Db10.pdf

21.https://ilkaygenc.com.tr/normalizasyon-nedir-sql/

22.http://beltslib.net/veri-tabanlarindanormalizasyon.html

23.https://social.msdn.microsoft.com/Forums/trTR/47058052-1da2-4b87-a115- e97724a0110a/ansi-nedir-?forum=csharptr

24.http://www.mustafabektastepe.com/tag/ansi-sqlstandartlari/

25.http://www.mustafabektastepe.com/2012/01/01/sqlnedir-ne-degildir-tarihcesi/#more-88

26.https://www.protan.com.tr/mysql-nedir/

27.https://www.biltektasarim.com/blog/mysql-nedir

28.https://www.hostinger.web.tr/rehberler/mysql-nedir

29.https://bulutistan.com/blog/mysql-nedir/

1. https://tr.wikipedia.org/wiki/MySQL
2. [https://learn.microsoft.com/tr-tr/xamarin/xamarinforms/enterprise-application-patterns/mvvm](https://learn.microsoft.com/tr-tr/xamarin/xamarin-forms/enterprise-application-patterns/mvvm)

32.https://tr.wikipedia.org/wiki/Nesne\_y%C3%B6nelimli \_programlama

33.https://bidb.itu.edu.tr/seyirdefteri/blog/2019/02/05/obj ect-oriented-programming

34.https://academy.patika.dev/courses/oop/oop-nedir

35.http://www.baskent.edu.tr/~gmemis/courses/introducti ontodatabase/Db10.pdf

36.http://beltslib.net/veritabanlarindanormalizasyon.html

37.https://learn.microsoft.com/trtr/dotnet/desktop/wpf/xaml/?view=netdesktop-7.0

38.https://gelecegiyazanlar.turkcell.com.tr/konu/egitim/c101/xaml

39.https://www.mshowto.org/mvvm-nedir.html

40.https://devnot.com/2015/mvvme-hizli-bir-giris/

4[1.https://www.ilkayilknur.com/mvvm-model-viewviewmodel-nedir](https://www.ilkayilknur.com/mvvm-model-view-viewmodel-nedir)

[ViewList Sorting and Filtering From Documentation]

42.https://learn.microsoft.com/trtr/dotnet/api/system.win dows.data.collectionview?view=windowsdesktop-7.0

[Data Binding]

43.https://learn.microsoft.com/trtr/dotnet/desktop/wpf/dat a/?view=netdesktop-7.0

[Data Binding]

44.https://www.buraksenyurt.com/post/WPF-Veriye-

Baglanmak-(Data-Binding)-bsenyurt-com-dan

[Live Charts for GeoMap, so we can use Mall's ID]

45.https://github.com/Live-Charts/LiveCharts/blob/master/WpfView/GeoMap.cs

[Icon Stuff]

46.https://icons8.com/

[Another Icon Site]

47.https://www.flaticon.com/

[Backgrounds]

48.https://www.pexels.com/trtr/arama/desk/?size=large&orientation=landscape

[OnProperyChanged Event AKA Handling Properties]

49.https://learn.microsoft.com/trtr/dotnet/desktop/wpf/data/how-to-implement-propertychange-notification?view=netframeworkdesktop-4.8