



Web Sitesi

ÖZET

Müzişyenlerin sosyalleşebilecekleri, şarkının tab, akor gibi detayları takip edebilecekleri bir platform

Hazırlayan

Engin Karataş

Bilgisayar Mühendisliği Öğrencisi

Gate Of Musicians

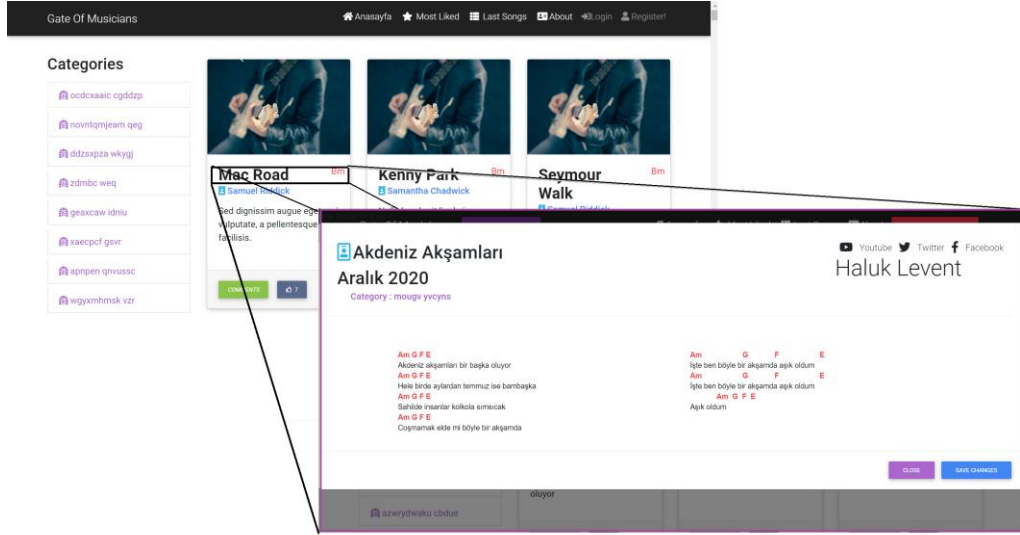
Abstract

İnternetin yaygınlaşmasıyla beraber müzisyenlerde interneti etkin bir şekilde kullanıp araştırmalarını yapmaktadırlar. İlgilendikleri şarkıların müziksel dizgelerini bulmak için bir çok siteye bir çok istekte bulunurlar. Bu sitelere erişim için çeşitli arama motorları tarafından sürekli olarak tıklanmak zorundadır. Dolayısıyla müzisyenlere oldukça zaman harcatmaktadır. Gate Of Musicians tüm müzisyenlerin herhangi müzik türünden aradıkları şarkıyı hızlıca bulmalarını ve diğer müzisyenlerle ortak bir noktada buluşmalarını amaçlar.

İçindekiler

1.Hazırlık	1
1.1 Amaç	1
1.2 Toplulukta bulunan ihtiyaç ve gerklilikler	2
1.3 Neyi daha iyi yapıyor?	3
2.Bilinmesi gerekenler	2
2.1 Akor Nedir ?	2
2.2 EntityFramework Nedir?	4
2.3 EntityFramework Codefirst Nedir?	4
2.4 MVC Nedir?	4
2.5 Asp.Net Nedir?	4
2.6 Asp.Net MVC Nedir?	4
2.7 Katmanlı mimari Nedir?	4
2.8 Projede Dikkate Edilecek Kavramlar	5
2.9 CRUD işlemleri nedir?	5
2.10 Bootstarp Nedir?	
3. Giriş:	6
3.1 GELİŞTİRME SÜRECİ	6
3.2 Admin	6
3.3 Kullanıcı	7
4. ASP.NET MVC5 PROJE DETAYLARI	9
4.1 Proje içinde trigger, procedure ve fonksyon oluşturulması	9
4.2 Entity Framework hazır stored procedure kullanımı	12
4.3 Örnek verilerin veritabanına insert edilmesi	12
4.4 Database Context Classı	15
4.5 Web.config	16
4.6 Entities	16
4.7 Bilgilerin sayfaya gönderilmesi	17

4.8 Bilgilerin sayfadan okunması	18
4.9 Site Yayınla (Publish) Ve Migrations	19
4.8 Site Yayınla	19
4.8 Migrations	20



1. Hazırlık:

1.1 Amaç:

- Müzisyenlerin akorlarını takip edebildiği, sosyalleşebilecekleri ve çaldıkları şarkı detaylarını kolayca Takip edebileceği bir platform oluşturmaktır.
- Ekran boyutu maksimum boyutta sunarak son kullanıcı olan müzisyenin kolayca akor okumasını sağlamak
- O esnada sahnede olan müzisyenlerin sitemizden akor veya tab okumasını sağlamak.

1.2 Toplulukta bulunan ihtiyaç ve gereklilikler:

- Bir şarkı yayınlandığında, şarkıyı çalmak isteyen, akorlarını merak eden kullanıcılar internet tarayıcısında arama yapmaktadır. Fakat şarkı henüz site adminleri tarafından farketilmediği, şarkının akorları internette olmadığı için son kullanıcının bu talebi karşılanamamaktadır. Bu sorunu GateOfMusicians sitesine üye olan herkese akor ve şarkı girme bilgisini vererek çözüme ulaştırılmak istenmektedir
- Yazıları okumakta zorlanan, ritim kaçıran müzisyenlerin etkin şekilde ekrana odaklanma zorluğu
- Kullanıcının farklı cihazlardan, istekte bulunacağı sitelerde cihaza göre ekran boy oranını ayarlaması yapılmamaktadır. Güncel yazılım teknolojilerini kullanarak bu sorun çözümlenmek istenmektedir.

- Herhangi internet kullanıcısının(müzişyenin) istediği şarkılara birden çok akor dizisiyle koleksiyon oluşturmak, bunları diğer kullanıcılar ile paylaşmak. Paylaşım yapan müzişyen ayrıca ödül sisteminden yararlanılması sağlanır

• 1.3Neyi daha iyi yapıyor ?

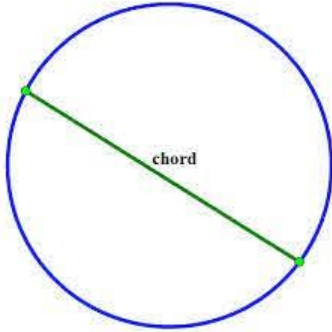
- Milyonlarca müzik türüne ait akorların girilmesi durumunda internetin olduğu her yerden erişme imkanı
- Kullanıcıların ödüllendirildiği sosyal medya sistemi.
- Responsive tasarımla her cihazda ekran en boy oranını koruyor
- En, boy oranını olabilecek maximum ölçeklerde kullanarak müzişyenlerin odaklanmasını arttırıyor.

2.Bilinmesi gerekenler:

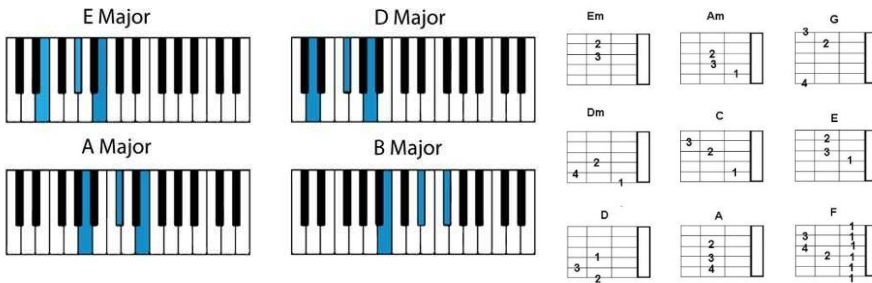
2.1Akor Nedir ?

Akor herhangi iki sesin aynı anda tınlatılmasıdır. Örneğin : La sesiyle Mi sesinin Aynı anda tınlatılmasıdır

Aşağıdaki resim geometride akoru temsil eder.



Aşağıdaki şekil piano`da akor gösterimlerini temsil eder. Aşağıdaki resim gitarda akor gösterimlerini temsil eder.



2.2 EntityFramework Nedir ?

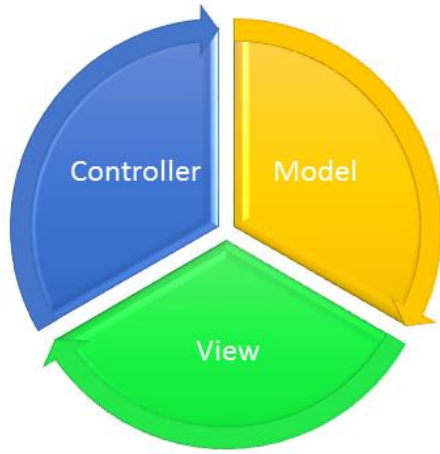
- Entity Framework, programlama dilinin içerisinde model oluşturarak veya var olan databasedeki modelleri kullanarak, databasemizi ilişkisel tablolarımızı yönetmemizi sağlayan bir modelleme-kullanma aracıdır. Bu tür modelleme araçları ORM araçları olarakta bilinmektedir.

2.3 EntityFramework Codefirst Nedir ?

- Kod yazarak veritabanımızdaki attributelere karar verebildiğimiz, Database tablolarımızı classlar kullanarak, kod yazarak anlatabildiğimiz bir yapıdır.
- İstersek bu yapıyı Veritabanı oluşturmak için, verileri veritabanından çekmek için kullanabiliriz.

2.4 MVC Nedir ?

- MVC, Model , View , Controller kelimelerinin baş harflerinden oluşur ve her kelime MVC'nin farklı bir katmanını ifade eder



2.5 Asp.Net Nedir ?

- ASP veya ASP. NET, web ve masaüstü uygulamaları oluşturmak için kullanılan bir framework'tur. Visual Studio'da geliştirilir.

2.6 Asp.Net MVC Nedir ?

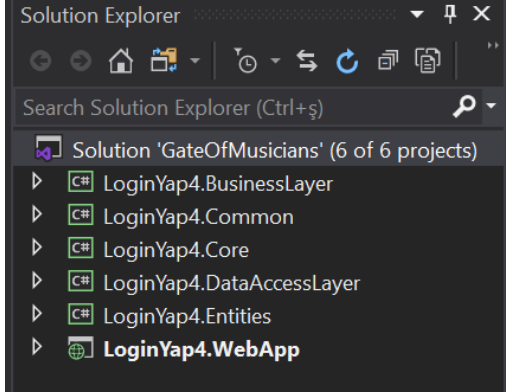
- ASP.NET MVC, MVC katmanını ASP.NET'e eklemek için Microsoft'un geliştirdiği framework'tür.

2.7 Katmanlı mimari Nedir ?

- Güvenlik, proje yönetimi, merkezi sistem gibi bir çok özellik sunan mimaridir
- Katmanlı Mimari cross platform projelerde kullanılır
- Farklı UI (User Interface)ler geldiğinde kullanıcıların oturum açmaları, database işlemleri sürekli belirli katmanlar tarafından işlenir.
- Projemizde, Mobil veya IOS kullanıcıları için ekstra oturum açma mekanizmaları oluşturmak projenin tutarlılığı açısından büyük kayıp oluşturabilir. Çünkü bir kullanıcının kaydı tek yerde olmalıdır.

2.8 Projede Dikkate Edilecek Kavramlar:

- Her kullanıcı ancak ve ancak oturum açtığı anda kendi bilgilerini görebilir ve düzenleyebilir.
- Kullanıcıların yaptığı her işlem veritabanında log bilgisi şeklinde tutulmalıdır.
- Kullanıcı, giriş yapmaya sadece şarkı ekleme, yorum yapma gibi işlemlerde ihtiyaç duymalıdır. Her kullanıcının akor okumasına izin verilerek site trafiği yüksek tutulmalıdır.



Katmanlar ProjeAdı . KatmanAdı Şeklinde olmalıdır kullanılan katmanlar 20yi aşmaktadır fakat biz bu projemiz için 6 katman oluşturduk.

Not:LoginYap4 GateOfMusicians olmalıdır

2.9 CRUD işlemleri nedir ?

CRUD işlemleri CREATE –READ – UPDATE – DELETE işlemlerinin kısaltmasıdır. Veritabanlarındaki bilgileri yönetmemizi sağlayan metodlardır.

2.10 Bootstarp Nedir ?

Bootstrap açık kaynak kodlu, web sayfaları veya uygulamaları geliştirmek için kullanılabilecek araçlar bütünü ve önyüz çatısı. Bootstrap, web sayfaları veya uygulamalarında kullanılabilecek, HTML ve CSS tabanlı tasarım şablonlarını içerir.

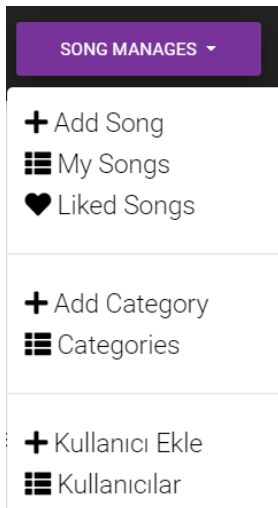
3. Giriş:

3.1 GELİŞTİRME SÜRECİ

- Projemizi katmanlı mimaride geliştirerek, gelen talepler doğrultusunda değiştirilmesi istenen kısımları hızlıca, maliyeti arttırmadan kullanıcıya sunabilmeliyiz. Katmanlı projeler fazladan iş yükü olarak görülsede proje büyüdükçe öneminin arttığı bilinmektedir. Katmanlı mimariler ile geliştirme yapıyorsak class ve dosyaları düzenli kullanmak önemlidir.
- İş yükünden, yük olarak gözükmemelidir. Bununla birlikte tekrar eden her türlü yapıdan kaçınılmalıdır. Projenin detaylarında tekrar eden yapılar gördükçe yeni classlar Interfaceler, katmanlar eklenmelidir fakat bunu pratik ve mantık çerçevesinde yapmalıyız.

3.2 Admin:

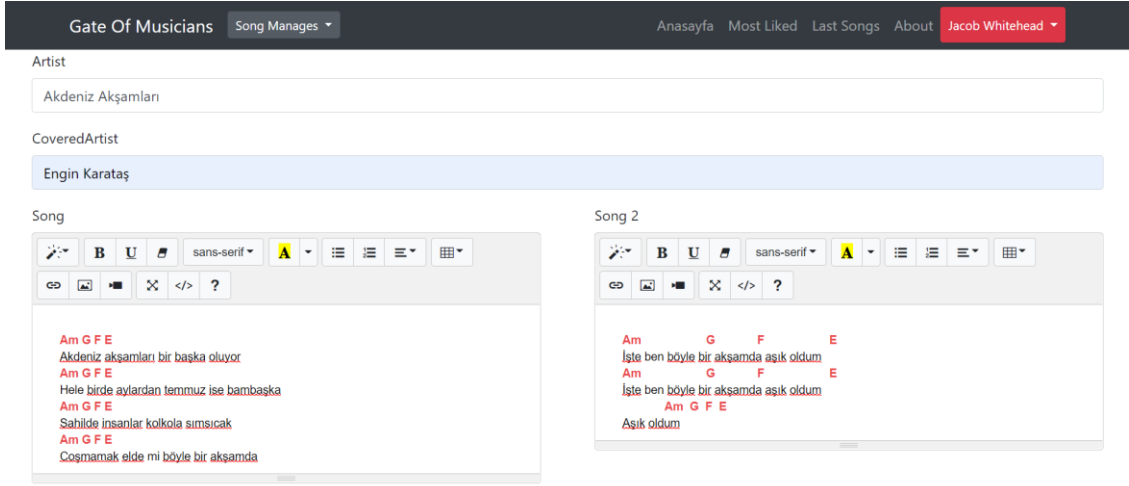
- Kategori, şarkı, kullanıcı işlemlerinin tümünü yönetebilmelidir.
- Kullanıcıları istendiği zaman silinmeli veya engellenebilmelidir.
- Yetkili admin kullanıcının erişebildiği, düzenleyebildiği içerikler tüm site kullanıcılarından fazla olmalı ve gerektiğinde site kullanıcılarını silebilmelidir.



3.3 Kullanıcı:

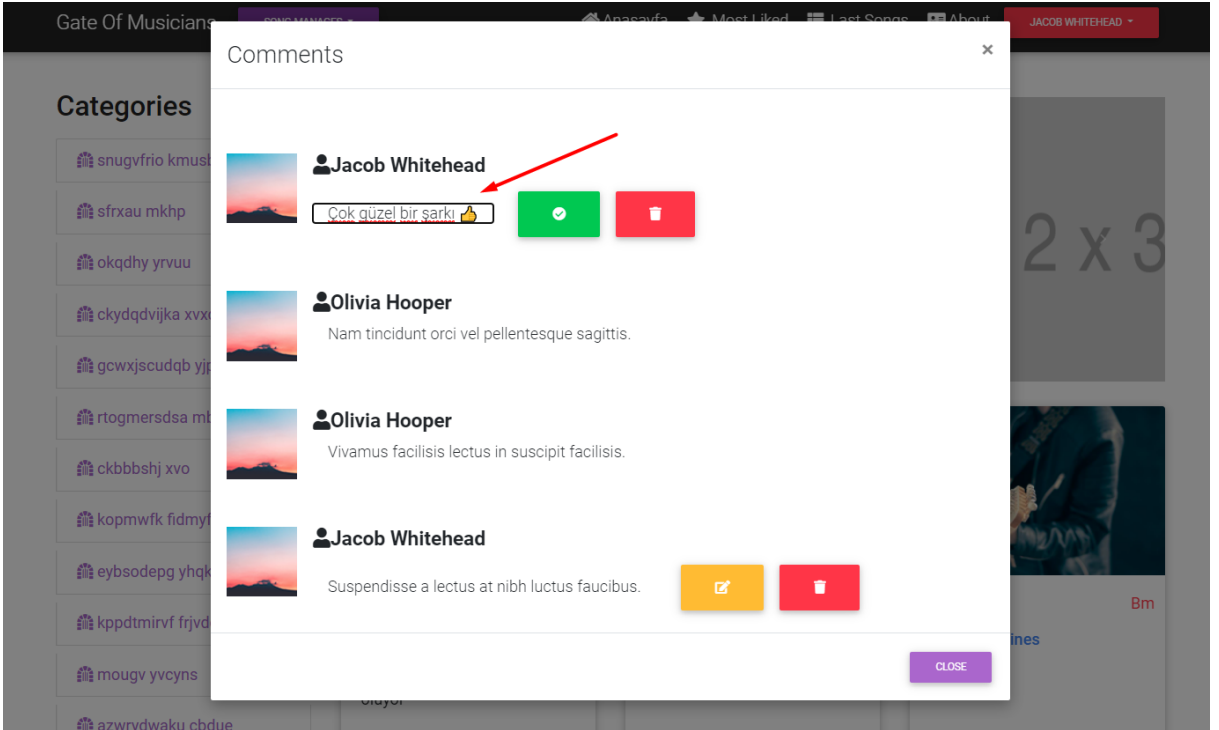
- Şarkıları bulabilmek, görüntüleyebilmeli ve istediği takdirde oturum açıp şarkı yükleyebilmelidir. Oluşturduğu şarkıları görüntüleyebilmeli, mevcut şarkılarını silebilmelidir.

Şarkı yükleme ekranı



- Sitede bulunan şarkıları beğenebilmelidir. Site kullanıcısı istediği şarkıya yorum yapabilmeli, istediği zaman yorumunu görüntüleyebilmeli, silebilmelidir.

Yorumlar ekranı



- Diğer kullanıcıların yorumlarını görebilmelidir
- Profilini düzenleyebilmelidir.

Profi bilgileri ekranı

Gate Of Musicians

SONG MANAGES

Anasayfa


★ Most Liked

Last Songs

About

ENGİN KARATAŞ

Profiliniz



Engin Karataş

Username : enginkaratas123

E-Mail : enginkaratas99@gmail.com

More...

- Profil resmini ve bilgilerini ekleyip güncelleyebilmelidir.

Profil düzenleme ekranı

Gate Of Musicians

SONG MANAGES


Anasayfa

★ Most Liked


Last Songs

About

ENGİN KARATAŞ



Choose File No file chosen

 Lütfen jpg, jpeg ya da png formatı kullanın.

Name

Engin

Surname

Karataş

Username

enginkaratas123

Password

...

E-Mail

enginkaratas99@gmail.com

SAVE

CANCEL

- Gerekli zaman oturumunu kapatabilmelidir.

4. ASP.NET MVC5 PROJE DETAYLARI:

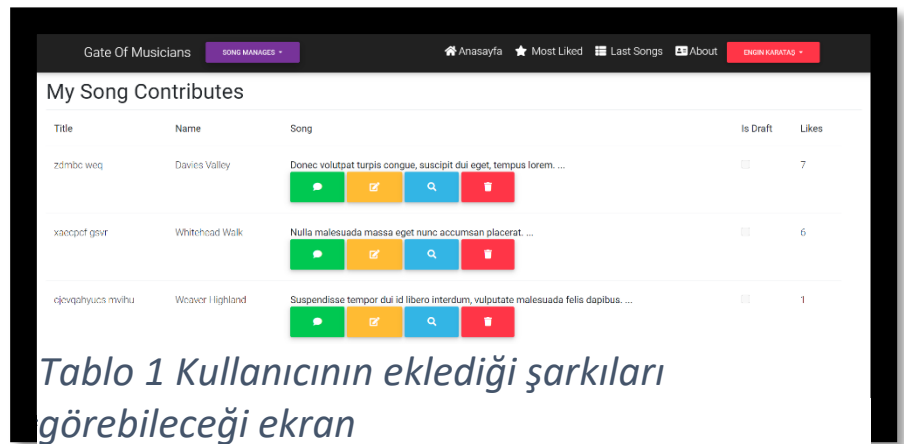
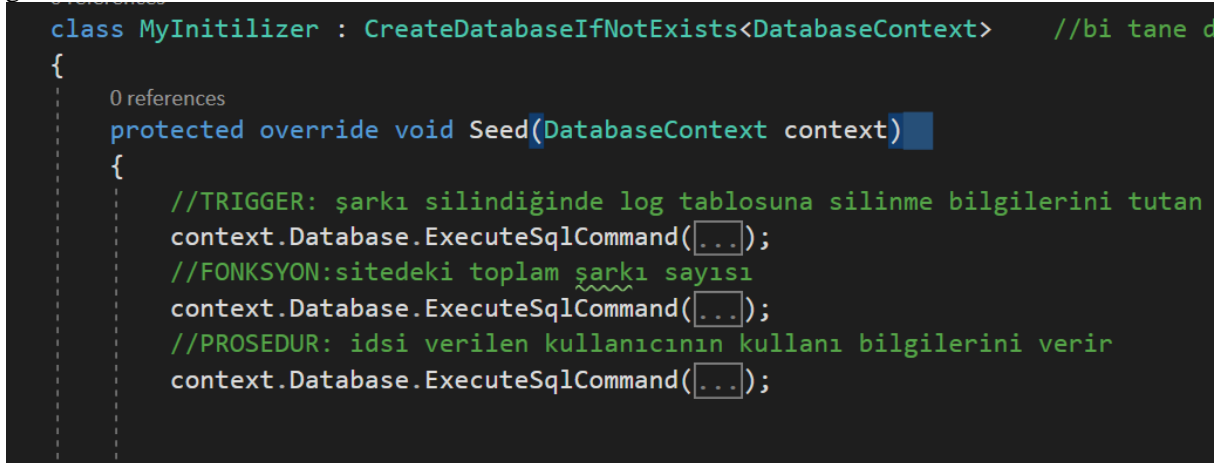
4.1 Proje içinde trigger, procedure ve fonksyon oluřturulması

Entity Framework Codefirst yaklaşımının sunduđu işlemlerden biriside SQL sorgularını c# kodlarıyla oluřturma. MSSQL DBMS arayüzünde yapılabilen her işlem codefirstten yapılabilmektedir.



Codefirst kısmında CreateDatabaseIfNotExists classının seed() metodunun ezdiğimizde gelen DbContext parametresiyle yapılabilmektedir. Burada CreateDatabaseIfNotExists veritabanı oluřturma aşamasında veri ekleme, prosedür, trigger, view, function ekleme, veya crud işlemlerinin gerçekleştirilebildiđi, T-SQL kodlarını yazabildiğimiz bir alan sunar.

Sql kodları yazarak veritabanı oluřtuđunda oluřacak kodlar Seed metodunun içerisinde gözükmektedir.



Tablo 1 Kullanıcının eklediđi şarkıları görebileceđi ekran

Örneğin fonksiyon oluşturan sql komutları:

```
//FONKSYON:sitedeki toplam şarkı sayısı
context.Database.ExecuteSqlCommand(@"
create function UserCount()
returns int
as
begin
declare @counter int;
--
select @counter = Count(Id) from Song
return @counter;
end
");

//PROSEDUR: idsi verilen kullanıcının kullanıcı bilgilerini verir
context.Database.ExecuteSqlCommand(@"
create procedure GetUsernameById
@id int
AS
Begin
select Name, Surname, Username from SiteUser where Id = @id
END
");

protected override void Seed(DatabaseContext context)
{
    //TRIGGER: şarkı silindiğinde log tablosuna silinme bilgilerini tutan
    context.Database.ExecuteSqlCommand(@"
create trigger TG_SongDelete on Song after delete
as
begin
declare @song_name nvarchar(50)
declare @viewed_number int
declare @olusturma_tarih Date
declare @aciklama nvarchar(50)
set @aciklama = 'Bir şarkı silindi';
select @song_name = Name from deleted
select @viewed_number = ViewedNumber from deleted
select @olusturma_tarih = CreatedOn from deleted
insert into dbo.Logs values (@song_name,@olusturma_tarih,@viewed_number,@aciklama)
end
");
```

C# içinde sql tablolarını, bilgilerini çekmek istenilirse, çekmek istenilen bilginin sql tarafında gözüken sütununun veri tiplerini bulunduran bir classa ihtiyaç duymaktadır.

Aşağıda usercount ve userbilgi sınıfları görünmektedir. Properiileri gelen colon adlarıyla ve tipleriyle aynı olmalıdır

```
0 references
public class usercount
{
    0 references
    public int SarkiSayisi { get; set; }
}
3 references
public class userbilgi
{
    1 reference
    public string Name { get; set; }
    1 reference
    public string Surname { get; set; }
    1 reference
    public string Username { get; set; }
}
Save (Ctrl+S)
```

Not: EF uygulamalarında etkin ölçüde Liste yapısı kullanılır.

Geriye dönen değer list türündedir ve etkilenen satır sayısı kadar liste boyutu dinamik olarak artabilmektedir.

```
public class ExecuteCommand
{
    DbContext db;

    1 reference
    public ExecuteCommand()
    {
        db = new DbContext();
    }
    1 reference
    public List<usercount> UserCount()
    {
        return db.Database.SqlQuery<usercount>("select dbo.UserCount() as SarkiSayisi").ToList();
    }

    1 reference
    public List<userbilgi> UserBilgi(int? id)
    {
        return db.Database.SqlQuery<userbilgi>(@"
        execute GetUsernameById "+id).ToList();
    }
}
```

Gelen bilgileri tek bir sayfa üzerinde görmek istiyorsak(projede Home/Index2 dizininde örneği bulunmaktadır.) her iki fonksiyonun tipini tutacak bir classa daha ihtiyacımız vardır. Bu classımız

ViewModel classımız olsun

```
using LoginYap4.BusinessLayer;
using System.Collections.Generic;

namespace LoginYap4.WebApp.Models
{
    1 reference
    public class ViewModel
    {
        1 reference
        public List<usercount> sqlFonksyonKisiSayisi { get; set; }
        1 reference
        public List<userbilgi> procedureGetUserInfo { get; set; }
    }
}
```

MVC5 sayfaya iki model göndermek TempData veri türü(obje tutar) kullanarak yapılabilmektedir. Aynı zamanda classlar yardımıyla tutulabilir

Sayfaya görüntülenebilmesi için controllere gönderilir.

```
public ActionResult Index2()
{
    ExecuteCommand ex = new ExecuteCommand();
    Random rand = new Random();

    var usercount = ex.UserCount();
    var user = ex.UserBilgi(rand.Next(1,20));

    ViewModel vm = new ViewModel();
    vm.sqlFonksyonKisiSayisi = usercount;
    vm.procedureGetUserInfo = user;

    return View(vm);
}
```

Home/Index2 sayfasına gelen bilgileri gönderelim

gelen bilgiler clasta tutulur

class bilgisi sayfaya post edilir

4.2Entity Framework hazır stored procedure kullanımı

Bir Veritabanı işlemlerini gerçekleştirilen classta(DatabaseContext modeli) OnModelCreating metodunu ezelebilmekteyiz. Bu metod DatabaseContext sınıfı oluşturulduğunda çalışacak metoddur. Şekildeki DbModelBuilder parametresi bu method çağırıldığında bir çok işlemi gerçekleştirilebilmesini sağlar.

```

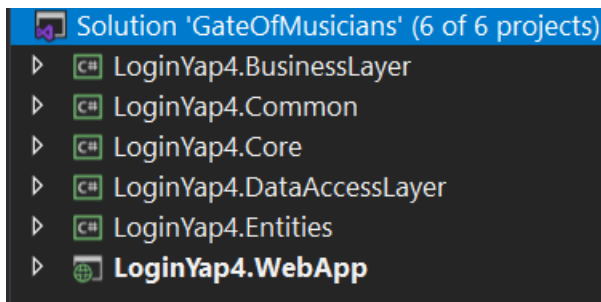
public class DatabaseContext : DbContext
{
    0 references
    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Song>()
            .MapToStoredProcedures(config =>
            {
                config.Insert(i => i.HasName("SP_SongInserted"));
                config.Update(u =>
                {
                    u.HasName("SP_SongUpdated");
                    u.Parameter(p => p.Id, "SongId");
                });
                config.Delete(d => d.HasName("SP_SongDeleted"));
            });
    }
}

```

Yukarıda bu metodu ederek EntityFramework'un sağlamış olduğu üç adet(delete insert update) otomatik prosedürlerinin MSSQL de oluşturulması için gereken kod gösterilmektedir. C# tarafında oluşacak bu otomatik prosedürler, veritabanında ilgili <tablo_adı> içerisindeki tablo_adı(entitie)'na otomatik prosedürler oluşturur. Bu prosedürler veritabanı oluşunca oluşacaktır.

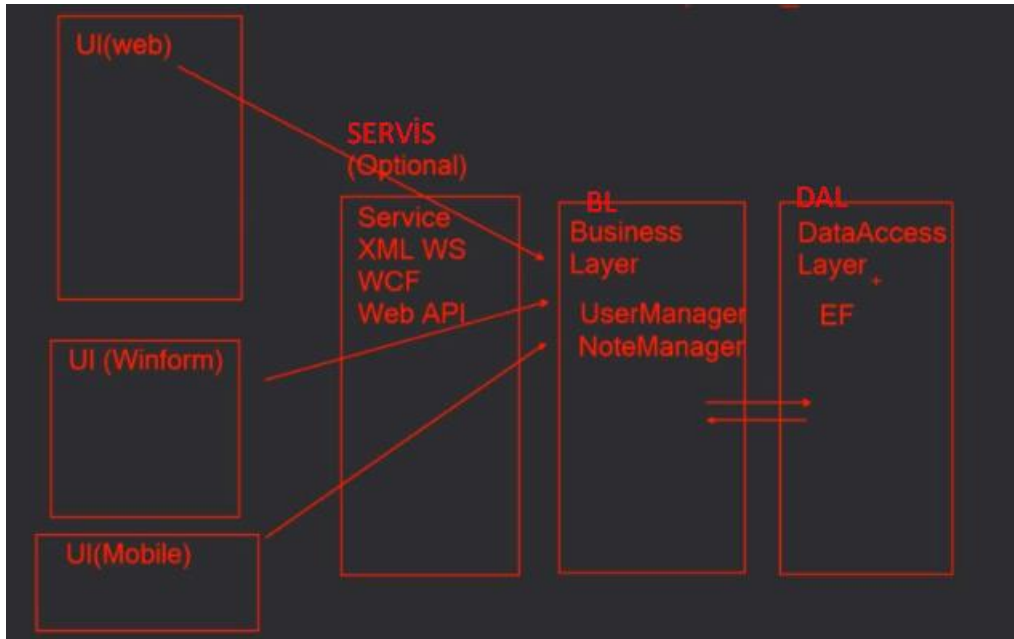
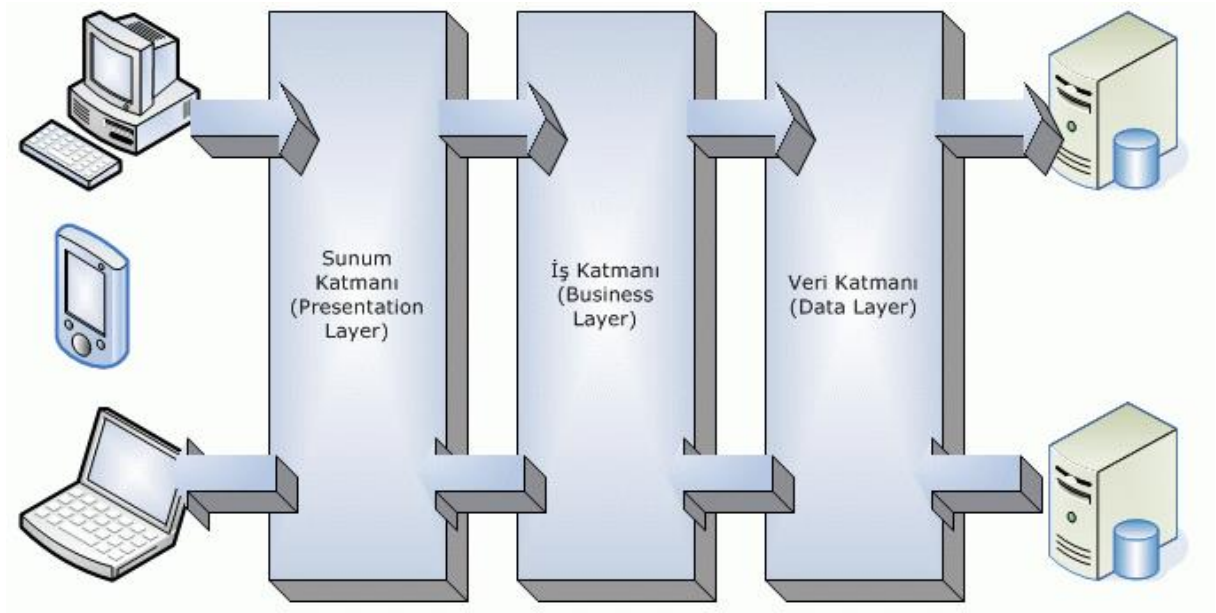
Not: Hazırda olan veritabanı için yukarıdaki işlemler gerçekleşmeyecektir. Kodun çalışmasını gerçekleştirmek için veritabanını silip tekrar yüklenmelidir.

4.2 Proje katmanlarına genel bakış



Hedeflediğimiz kullanıcılar bir şeyden memnun kalmadıklarında, katmanlı mimarilerle aksiyon almak daha kolaydır. Update edilmesi gereken bilginin deploy edilme maliyeti azalmış olur. Katmanlı mimaride güvenlik ön plandadır. Referans gösterilen katmanlar haricinde erişim mümkün olmaz. Proje karmaşıklığı azalır. Şirketteki değişen yazılımcıların projeye girmesini anlamasını oldukça kolaylaştırır.

Destekleyici görseller:



4.3 Örnek verilerin veritabanına insert edilmesi

Aşağıda gördüğünüz kod bloğu fakedata adlı classın(ilk sürümünden itibaren desteği yok) farklı veriler üretmesi için kullanılmıştır. Üretilen veriler ilgili modellere yazılarak veritabanına yüklenmektedir. Ayrıca aşağıdaki kodlar Seed() metodunun içerisine yazılmıştır.

Seed metodu hakkında önemli bir detay: Proje bittikten sonra eğer migration seneryolarını kullanılırsa, oluşturulan migrationsların “configuration class” içerisindeki seed metoduna atılmalıdır.

```

// Adding fake categories..
for (int i = 0; i < 17; i++)
{
    Category cat = new Category()
    {
        Title = FakeData.TextData.GetAlphabetical(FakeData.NumberData.GetNumber(5, 12)) + " " + FakeData.
        Description = FakeData.PlaceData.GetAddress(),
        CreatedOn = DateTime.Now,
        ModifiedOn = DateTime.Now,
        ModifiedUsername = "enginkaratas"
    };

    context.Categories.Add(cat);

    // Adding fake Songs..
    for (int k = 0; k < FakeData.NumberData.GetNumber(5, 9); k++)
    {
        SiteUser owner = userlist[FakeData.NumberData.GetNumber(0, userlist.Count - 1)];

        Song song = new Song()
        {
            Name = FakeData.PlaceData.GetStreetName(),
            ViewedNumber = FakeData.NumberData.GetNumber(22, 328),
            RateNumber = FakeData.NumberData.GetNumber(0, 5),
            Writer = "engin",
            CoveredArtist = "owner",
            Picture = "song.png",
            Artist = "owner",
            Text = FakeData.TextData.GetSentences(FakeData.NumberData.GetNumber(25, 50)),
            IsDraft = false,
        }
    }
}

```

4.4 Database Context Classı(Data Acces Layer içerisinde)

Bu class ile temel database bağlantımızı yapmakta ve tablolarımızı oluşturmaktayız.

```

7 references
public class DatabaseContext : DbContext
{
    0 references
    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Song>().
        .MapToStoredProcedures(config =>{..});

        modelBuilder.Entity<Log>().ToTable("Logs");
        modelBuilder.Entity<SiteUser>().ToTable("SiteUser");
        modelBuilder.Entity<Category>().ToTable("Category");
        modelBuilder.Entity<Liked>().ToTable("Liked");
        modelBuilder.Entity<Chord>().ToTable("Chord");
        modelBuilder.Entity<Comment>().ToTable("Comment");
        modelBuilder.Entity<Song>().ToTable("Song");
    }

    0 references
    public DbSet<Log> Logs { get; set; }
    10 references
    public DbSet<SiteUser> SiteUsers { get; set; }
    2 references
    public DbSet<Category> Categories { get; set; }
    0 references
    public DbSet<Liked> Likes { get; set; }
    2 references
    public DbSet<Chord> Chords { get; set; }
    0 references
}

```


4.5 Web.config(Proje ile aynı konumda)

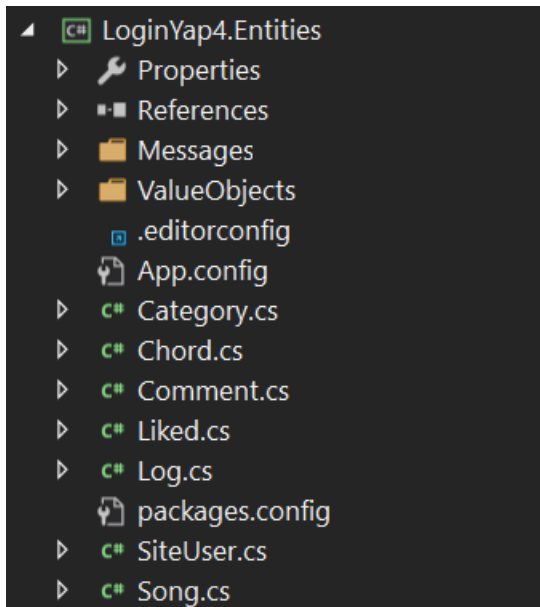
Web config çeşitli konfigrasyon ayarları, initilizerlerin tanımlanması gibi bir çok işlevi vardır. Burada connectionStrings ismindeki attribute oluşacak bağlantı yolunu belirtir. Bu bağlantı yolu belirtilmez ise Proje_Adı.Context_Adı şeklinde bir database oluşur.

```
Web.config DatabaseContext.cs MyInitializer.cs*
1 <?xml version="1.0" ?>
2 <!--
3 For more information on how to configure your ASP.NET application, please visit
4 https://go.microsoft.com/fwlink/?linkid=301888
5 -->
6 <configuration>
7 <configSections>
8
9 <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?linkid=237468 -->
10 <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b7
11 </configSections>
12 <appSettings>
13 <add key="webpages:Version" value="3.0.0.0" />
14 <add key="webpages:Enabled" value="false" />
15 <add key="ClientValidationEnabled" value="true" />
16 <add key="UnobtrusiveJavaScriptEnabled" value="true" />
17 <add key="MailHost" value="smtp.gmail.com" />
18 <add key="MailPort" value="587" />
19 <add key="MailUser" value="engineerings@gmail.com" />
20 <add key="MailPass" value="bzycg_112" />
21 <add key="SiteRootUri" value="http://localhost:47612" />
22 <add key="vs:EnableBrowserLink" value="false" />
23 </appSettings>
24 <connectionStrings>
25 <add name="DatabaseContext" connectionString="Server=(localdb)\mssqllocaldb;Database=GateOfMusicianDB;Integrated Security=SSPI;" providerName="System.Data.SqlClient" />
26 <!--<add name="DatabaseContext" connectionString="Data Source=89.252.141.89;Initial Catalog=GateOfMusicianDB;User Id=sonaovski;Password=Exocraft*-0;MultipleActiveResults
27 </connectionStrings>
```

4.6 Entities(Entities Katmanı)

Bu katman içerisinde varlık-ilşkileri belirtilir, Veritabanına yazılacak modellerin barındırıldığı katmandır.

Entities katmanı aşağıdadır.



Song tablosunun classını(model) görülmektedir.

```
namespace LoginYap4.Entities
{
    [Table("Songs")]
    27 references
    public class Song : MyEntityBase //Songs == Notes
    {
        [Required, StringLength(150)]
        4 references
        public string Name { get; set; }

        [Required, AllowHtml, StringLength(10000), DisplayName("Song")]
        4 references
        public string Text { get; set; }

        [AllowHtml, StringLength(30000), DisplayName("Song 2")]
        0 references
        public string Text2 { get; set; }

        [AllowHtml, StringLength(100), DisplayName("Salt Text For Clear Display")]
        2 references
        public string TextSalt { get; set; }
        2 references
        public string Writer { get; set; }
    }
}
```

4.7 Bilgilerin sayfaya gönderilmesi

HomeController içerisinde Index adlı Action(fonksyon) çalıştırılırsa SongManager isimli tablodaki ListQueryable() adlı hazırladığımız fonksyona LinQ sorguları gönderdik . ToList() Metoduyla geriye dönüş tipinin List olduğunu söyledik. Böylece sayfaya Linq sonucunda dönen bilgiler gönderilmiş oldu.

Bilgilerin sayfaya ulaşması için controller yapısı:

```
namespace LoginYap4.WebApp.Controllers
{
    0 references
    public class HomeController : Controller
    {
        private SongManager songManager = new SongManager();
        private CategoryManager categoryManager = new CategoryManager();
        private SiteUserManager siteUserManager = new SiteUserManager();

        //private CategoryManager categoryManager = new CategoryManager();

        // GET: Home
        0 references
        public ActionResult Index()
        {
            //LoginYap4.BusinessLayer.te db = new te();// for test(te)create db
            return View(songManager.ListQueryable().Where(x => x.IsDraft == false).OrderByDescending(x => x.ModifiedOn).ToList());
        }
    }
}
```

4.8 Bilgilerin sayfadan okunması

Sayfada bir model kullanılmadan önce sayfaya gönderilen bilgilerin türü ilk satırda olduğu gibi belirtilmelidir. `IEnumerable List` sınıfını içerisinde barındırdığından bu interface'yi çağırmanız durumunda list de kullanılabilir olacaktır.

Model tanımlarken küçük harf ile başlamak zorunludur.

Home/Index sayfası

```
1  @model IEnumerable<LoginYap4.Entities.Category>
2
3  @{
4      Layout = null;
5  }
6
7  <!DOCTYPE html>
8
9  <html>
10 <head>
11     <meta name="viewport" content="width=device-width" />
12     <title>Index</title>
13 </head>
14 <body>
15     <p>
16         @Html.ActionLink("Create New", "Create")
17     </p>
18     <table class="table">
19         <tr>
20             <th>
21                 @Html.DisplayNameFor(model => model.Title)
22             </th>
23             <th>
24                 @Html.DisplayNameFor(model => model.Description)
25             </th>
```

Aşağıda, gelen model içerisindeki bilgilere Model yazarak erişilebilmektedir.

```
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Title)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Description)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.CreatedOn)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.ModifiedOn)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.ModifiedUsername)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.Id }) |
            @Html.ActionLink("Details", "Details", new { id=item.Id }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.Id })
        </td>
    </tr>
}
```

4.9 Site Yayınlama (Publish) Ve Migrations

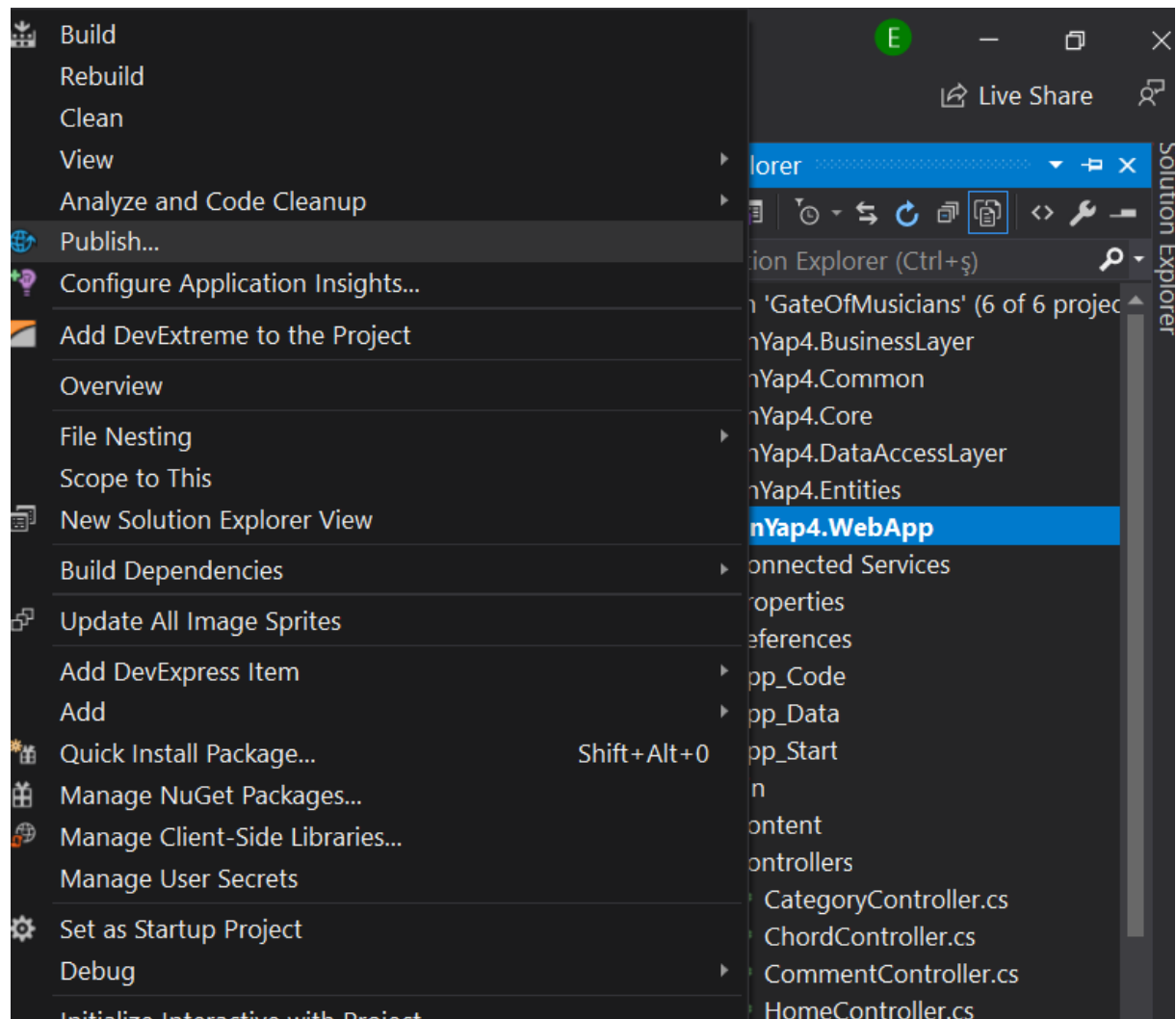
4.9.1 Site Yayınlama

Site yayınlama PHP vb. yorumalyıcı dillerden daha meşakkatli bir süreçtir. Aldığınız hostingin bazı özelliklere kapalı oluşu, Mssql databasesini yüklemenize engel olabilir. Bu yüzden MVC5te migrations senaryoları vardır.

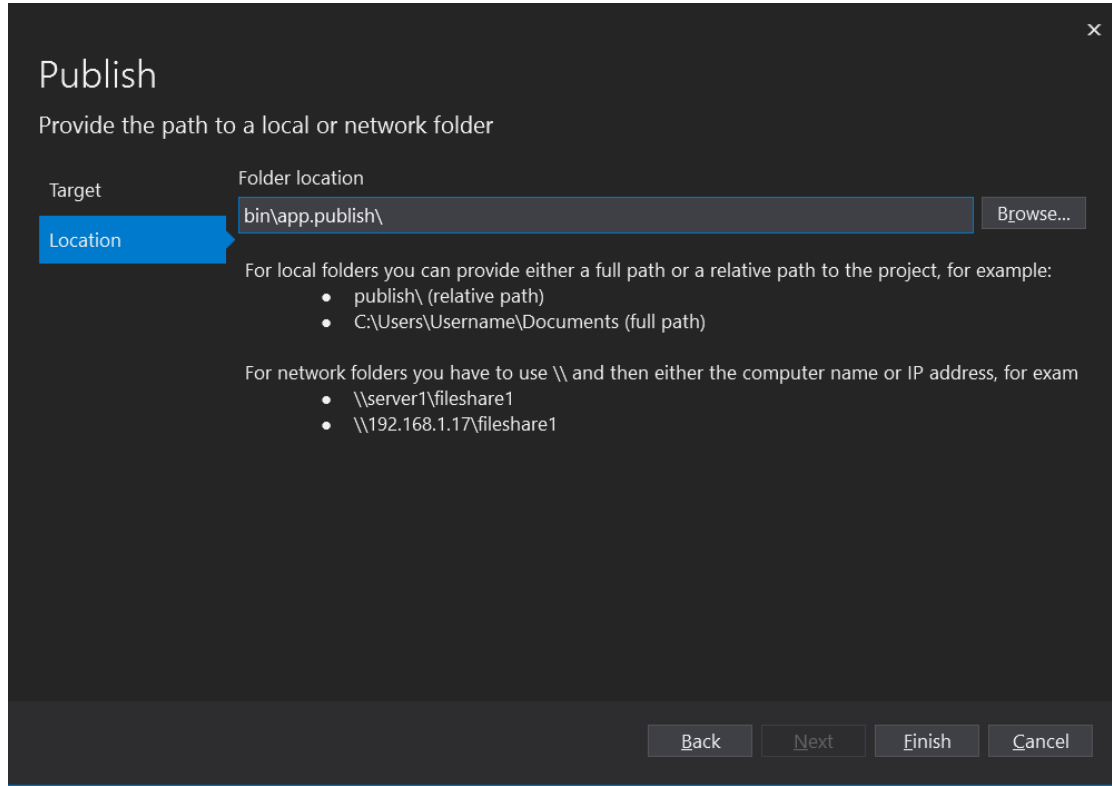
Site Publish Etmek

WebApp adlı UserInterface katmanımıza sağ tıklayıp publishi seçiyoruz.

Not: Diğer katmanlar dll olarak oluşturulmaktadır. İlerleyen görsellerde oluşturulacak ftp dosyası içerisinde bin klasöründe .dll'leri görebilmekteyiz.

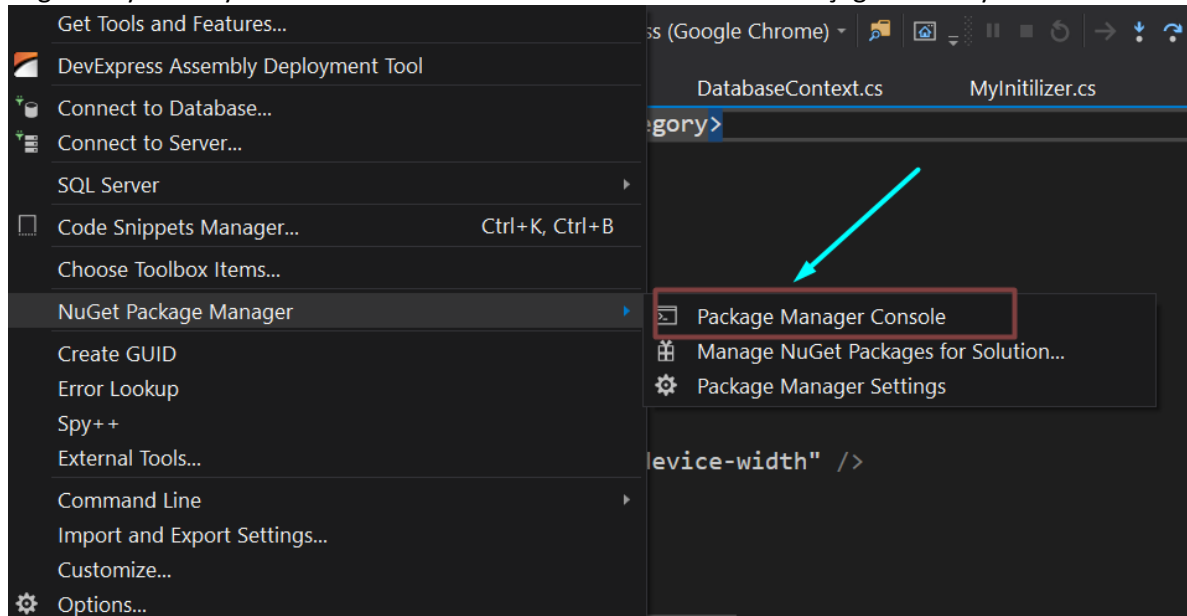


Aşağıdaki lokasyon projenin publish(release version) edilmesi istenilen yolu belirtir. Finish dediğimizde, klasör içerisinde çıkan dosyalar, ftp serverda httpdocs içerisine yüklenmelidir.



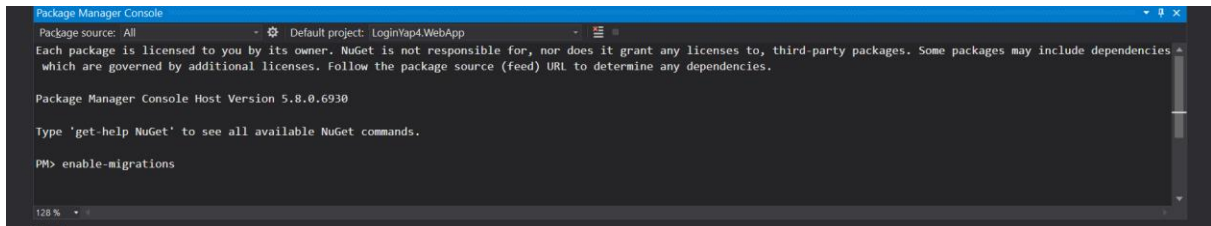
4.9.3 Migrations

Migrations, veritabanında kullanıcı, verileri oluşmaya başladığında, silinmemesi gereken kayıtlar oluşur. Ve yeni bir kolon vb bilgiler eklenmesi gerektiği zaman veritabanımızı en baştan oluşturamayacağımız için migrations senaryoları devreye girer. FTP ye release dosyalarını attıktan ve databasemizi localde çalıştığımız database adıyla aynı yaptıktan sonra, connection string, sunucudaki veritabanına bağlanmaya ayarlanmalıdır. Bu adımdan sonra aşağıdaki yol izlenmelidir.



Vetitabanını güncelleyebilmek için:

- 4 Buraya **enable-migrations** yazarak migrationsları aktif etmemiz gerekir.(dataaccess layerin bulunduğu katman ayar tuşuna basarak tanıtılmalıdır). Bu işlem Migrations adında bir klasör oluşturur ve adı configuration.cs adında bir class ekler. Bu dosyanın içinde seed() metoduna, MyInitalizer clasındaki seed() metodu içerisindeki bilgiler kopyalanmalıdır çünkü veritabanı oluşturmak ve güncellemek için migrationslar kullanılırken buradaki seed metodu çalışır. Eğer buradaki seed metodu doldurulmazsa, FakeData adındaki classımız çalışmayacak, tablo içerisindeki bilgiler doldurulmayacaktır.
- 5 **add migration migration_Adı** ile migration oluşturulmalıdır.Bu adımdan sonra classlarımız ve veritabanımız kontrol edilir. Migrationsların oluşması için bir karşılaştırma yapılır. Eğer veritabanında oluşturulan modele ait kolonlar yoksa, bu oluşturulan migrations classı içerisinde değişiklikler algılanır ve eklenmesi gereken özellikler **Up()** fonksiyonunda belirli olur.
- 6 **Update-Database** yazdığımızda ise bu veritabanının değişiklikleri onaylaması, ve oluşturulan migrationsun veritabanına atılması onaylanır.



```
Package Manager Console
Package source: All - Default project: LoginApp4.WebApp
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages. Some packages may include dependencies which are governed by additional licenses. Follow the package source (feed) URL to determine any dependencies.
Package Manager Console Host Version 5.8.0.6930
Type 'get-help NuGet' to see all available NuGet commands.
PM> enable-migrations
```

KAYNAKLAR:

Murat Başeren – Udemy – Yazılımcıların Yükselişi

<https://stackoverflow.com/>

<https://docs.microsoft.com/>