DÜZCE ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ

Asp. Net MVC Core Ortamında Üniversite Web Sayfası Tasarımı

162120012 – Ali SARI

OCAK 2022

İÇİNDEKİLER DİZİNİ

KISALTMALAR

SQL	Structured Query Language
MVC	Model-View-Controller
CSRF	Cross site request forgery
EF	Entity Framework Core
ORM	Object Relational Mapping
OOP	Nesne Yönelimli Programlama
CRUD	Create – Read – Update – Delete
SPA	Single page application
WASM	WebAssembly
API	Application Programming Interface
SDK	Software Development Kit

ŞEKİLLER TABLOSU

BÖLÜM 1

GİRİŞ

Günümüzde gelişen teknoloji ile birlikte web sayfaları oldukça hızlı bir ivmeye sahip olmuşlardır. Teknolojinin geldiği bu noktada web sayfaları hayatımızın büyük bir kısmında yer kaplamaktadır. Web sayfalarının gelişimleri ve popülerleşmeleri ile insan hayatında oldukça kolaylık sağlayan bir etken haline gelmiştir. Birçok alanda olduğu gibi eğitim alanında da birçok insan web sayfalarından yardım almaktadırlar. Bu çalışmada ASP .Net Core yazılım geliştirme teknolojisi kullanılmaktadır. Asp .net Core altyapısı olarak Mvc altyapısı kullanılmaktadır. Admin panellerini kullanmak için 2 farklı sistem kullanılmakta. Düzce üniversitesi ana sayfasında olması gereken tablolar için Blazor Admin kullanılmaktadır. İkinci kısımda ise ana sayfa dışına (düzce üniversitesi sitesinin dropdown butonlarının hepsi, fakülteler ve enstitüler) kullanması gereken tablolar için Area kullanılarak yapılmıştır. Admin paneline sadece yetkili girişi sağlanmaktadır. Bu yüzden Admin paneline girmek isteyen kullanıcı veya yetkili giriş yapmadıktan sonra panele giriş yapamamaktadır. Eğer giriş yapmış ise admin paneline giriş yapabilmektedir. Giriş yapıp yapılmadığı Session'lar ile sağlanmaktadır. Tabloları Sql veri tabanında oluşturmak için ara kısım Migration için Entity Framework Core teknolojisi kullanılmaktadır. Bu yapılan çalışma responsive'dir (web sitesinin bilgisayar, mobil ve tablet cihazlardan da girildiğinde site içindeki resim, yazı gibi elementlerin ekran genişliğine göre yeniden şekillenip ekrana tam oturması ile oluşur).

BÖLÜM 2

ASP. NET MVC CORE ORTAMINDA ÜNİVERSİTE WEB SAYFASI TASARIM YAPIMI

2.1 ASP .NET Core

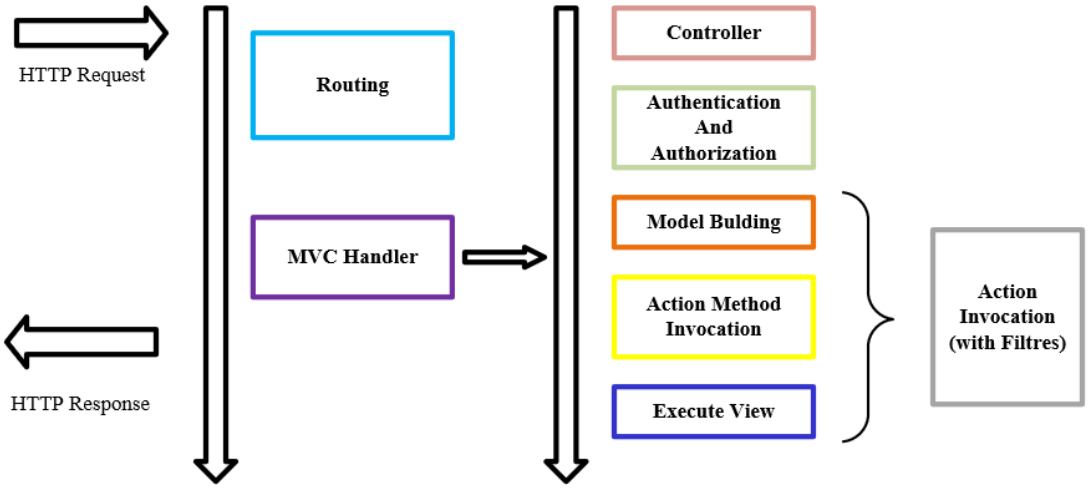
ASP .NET Core, Microsoft ve .NET topluluğu tarafından geliştirilmiş ve geliştirilmeye devam eden, Github üzerinde bulunan açık kaynaklı, çapraz platform geliştirme yapmaya olanak sağlayan bir platformdur [1]. 2016 yılında tanıtılan teknoloji, ASP.NET tabanlı uygulamalar geliştirmek için en iyi alternatiflerden birisidir. ASP .NET Core, .NET teknolojileri ile yazılım geliştirme çalışmaları yapan geliştiricilerin daha duyarlı, güvenilir ve genişletilebilir uygulamalar geliştirebilmelerini kolaylaştırmaktadır. ASP.NET Core, tüm ASP.NET altyapısının yeniden tasarlanarak, Web API ve MVC altyapıları ile birleştirilmesini sağlamaktadır. Web uygulamaları için API'lar oluşturulması bu şekilde daha kolay hale gelmiştir. Microsoft bu platformu düzenli olarak geliştirmektedir. ASP.NET Core teknolojisi sayesinde modern uygulamaları daha az efor ve maliyet ile, daha kısa sürede oluşturabilmeye olanak sağlanmıştır [2]. Asp .Net Core çapraz geliştirmeye verdiği olanak sayesinde Windows, macOS ve Linux'u desteklemektedir. Ayrıca bulut ve IoT uygulamaları oluşturulmasında kullanılabilmektedir.

2.1.1. ASP.NET Core Temel Özellikleri

- ASP.NET Core içerisinde birçok kütüphane ve optimizasyon aracı bulundurmaktadır. Bu araçlar sayesinde daha sürdürülebilir, yüksek performanslı yazılımlar oluşturabilmektedir.
- Açık kaynak kodlu esnek bir geliştirme platformudur.
- Asp .Net Cross platformdur. Yani platform bağımsızdır. Bu da Asp .Net Core kullanılarak geliştirilen uygulamaların Windows dışındaki diğer işletim sistemlerinde de uygulamaları çalıştırabilmek, diğer kullanıcıların erişimine açabilmek anlamına gelmektedir [3].

- Asp .Net Core, docker ve diğer container servisleri ile birlikte rahatlıkla yayına alınabilmektedir.
- Microsoft tarafından Github üzerinde, kaynak kodları açık bir şekilde MIT ve Apache 2 lisanslarına sahip olarak yer almaktadır.
- Asp .Net Core uygulamaları C#, F# ve Visual Basic dilleriyle geliştirile bilinmektedir.
- Asp .Net Core basit kodlama bileşenleri kullanıla bilinmektedir.
- Async/Await ile Asekron işlemler yapılabilmektedir.
- Asp .Net Core, CSRF koruması sunmaktadır [4].
- Razor Pages ile gelişmiş kodlama verimliliği sağlamaktadır.
- Çeşitli web uygulamaları oluşturma özgürlüğü sağlamaktadır.
- Asp .Net Core platformunda bakım kolaydır.
- Platform bağımsızlığı sebebiyle sadece IIS web sunucusuna ihtiyaç duymamaktadır IIS veya Apache gibi farklı sunucularda çalışmaktadır.

2.1.2. ASP .NET YAŞAM DÖNGÜSÜ SSS



Şekil 2. 1. ASP .NET YAŞAM DÖNGÜSÜ SSS

2.1.2.1. Routing

Döngüdeki ilk durağımızda, Route tablosunda tanımladığımız URL kalıplarına uygun bir istek URL'i geldi mi diye kontrol burada edilir. Gelen Url, tablomuzdaki kurallara uygun bir istekse, ilk eşleşen Route'ye göre yönlendirme sistemi isteği MVCHandler'e yönlendirir. Uygun değilse ise 404 durum kodu geriye döndürür.

2.1.2.2. MVCHandler

MVCHandler, ProcessRequest metoduyla isteğe karşılık gelen Controller'i ControllerFactory sınıfından oluşturur.

2.1.2.3. Controller Creation

- Execute metodu çağrılır,
- Execute gider ExecuteCore'u çağırır,
- ExecuteCore RouteData'dan istenen Action'un adını alır,
- ExecuteCore ActionInvoker'in InvokeAction metodunu çağırır.

2.1.2.4. Action Execution

InvokeAction ile Action göreve çağrıldığında:

- Url'den gelen istekteki parametre verilerini Model Binder sınıfı gerekli dönüştürme ve validation işlemlerinden sonra parametrelere bağlar.
- Authentication Filter çalışarak, Action'un önceden belirlenen kullanıcılara özel olup olmadığına karar verilir. Mesela Authorize Attribute'si belirlendiyse, istenmeyen kullanıcı Action'u çalıştıramaz.
- OnActionExecuting ve OnActionExecuted metodlarından önce Action Filter'lar çalıştırılır.
- Nihayetinde ExecuteResult çalıştırılır ve Action Result hazırlanır.

2.1.2.5. ExecuteResult

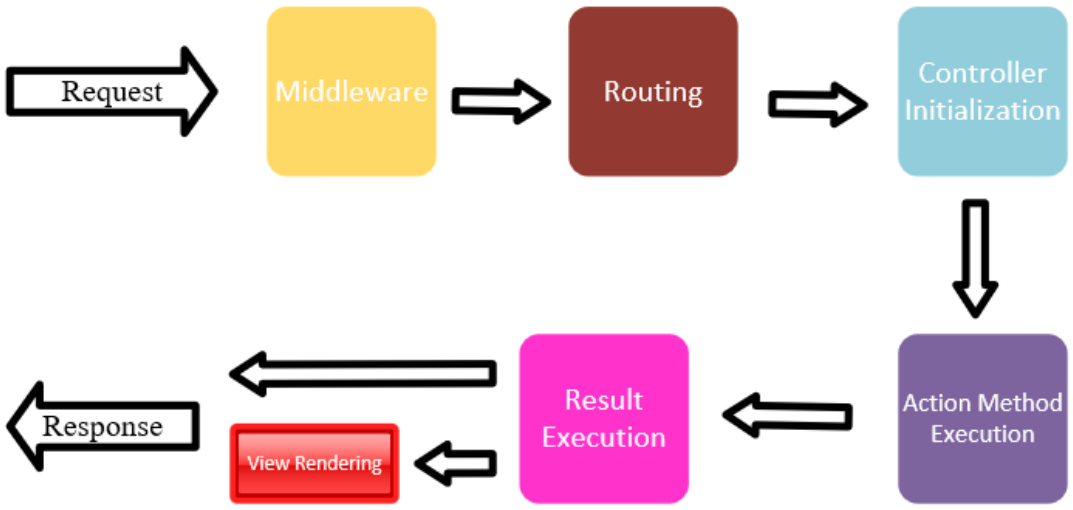
ExecuteResult çalıştığında:

- Gene ilk olarak OnResultExecuting ve OnResultExecuted metotlarından evvel belirlenen filtreler çalışır.
- İstenilen sonuç tipi View ise gerekli View Engine çağrılarak View ekrana gösterilir. Eğer başka bir şey (Json, Text, Binary vs.) ise, View Engine devreye girmeden direk olarak verilir.
- Ayrıca burası sonucu aldıktan sonra herhangi bir kodla müdahalemizin, yaşam döngüsü sonlanmadan önceki son yeridir.

2.1.2.6. View Initialization ve Rendering

ViewResultType istenen View Engine(Razor, Webform gibi) ile render edilir.Son olarak sonuç browser'a geri yollanır ve döngü sonlanır.

2.1.3. ASP .NET CORE YAŞAM DÖNGÜSÜ SSS



Şekil 2. 2. ASP .NET CORE YAŞAM DÖNGÜSÜ SSS

2.1.3.1. Middleware

Ara yazılım, istekleri ve yanıtları işlemek için bir uygulama işlem hattında bir araya gelen yazılımdır. Her Middleware requestin devamı için süreci, kendinden sonraki middleware aktarmak yada bitirmekle sorumludur.

2.1.3.2. Routing

Routing, gelen request'in en uygun olan enpoint'e eşlemesi ile ilgili sürece denir.

2.1.3.3. Controller Intialization

Burada olayı başlatan ve sonlandıran middleware EndpointMiddleware'dir. ActionEndpointFactory içerisinde CreateRequestDelegate ile oluşan delegate tetiklenir. Bununla birlikte ControllerActinInvoker devreye girer. İşte bu noktada geleneksel MVC pipeline'nını ControllerActionInvoker yönetir. Aslında ControllerActionInvoker görevi tam olarak CreateController ile birlikte devralıyor. Bunun öncesinde ki işlemler miras aldığı ResourceInvoker tarafında gerçekleşiyor.

2.1.3.4. Action Method Execution

Bir görünüm oluşturma veya istemciyi başka bir URL'ye yönlendirme gibi denetleyiciden gelen çeşitli yanıt türleri, tüm bu yanıtlar, genellikle eylem sonucu olarak bilinen IActionResult nesnesi tarafından işlenir.

2.1.3.5. Result Execution

Action methoddan dönen response objesini render etme ile ilgilenir.

2.1.4. Asp .NET Core ve .NET Framework Karşılaştırması

- ASP .NET'in önceki sürümleri ile karşılaştırıldığında istenmeyen bir çok yük ASP .NET Core'da kaldırılmıştır.

- Client-Side framework ile tam bir entagrasyon sağlanmaktadır. ASP.NET Core istemci tarafında yer alan Client-Side Framework olarak adlandırılan AngularJS, KnockoutJS, Gulp, Grunt, Bower gibi frameworkler ile rahatlıkla entegre edilebilmektedir.
- ASP.NET Core Cloud (Bulut) ortamlar için kolay yapılabilen konfigürasyonu sayesinde rahatlıkla optimize edilebilmektedir.
- ASP.NET Core uygulamaları hem .NET Framework hem de .NET Core platformlarında çalışabilmektedir. Ayrıca .NET Core'da bulunmayan ancak .NET Framework'de yer alan API'lerin de kullanımı hususunda esneklik sağlayabilmektedir.
- ASP.NET Core uygulamalarını host etmek kolaydır. Ek olarak ASP.NET Core IIS hostingi desteklemektedir.
- Bir ASP.NET Core uygulaması sıfırdan başlayarak yeniden tasarlanabilir [5].
- ASP.NET Core'da Global.asax veya uygulama başlatma etkinliği bulunmaz. Her şey startup.cs class sınıfında yapılmaktadır.
- ASP.NET Core uygulamalarında özel katman yazılımı yazmak çok kolaydır.
- Tag yardımcıları, görüntü bileşenleri daha sağlam, yeniden kullanılabilir ve server taraflı HTML yeni nesil ile uyumludur.
- Birim testleri oluşturmak ve yönetmek çok daha kolaydır.
- ASP.NET Core Identity kullanarak kimlik doğrulama ve yetkilendirmenin kolay konfigüre edilebilmesini sağlamaktadır [6].

2.2. Entity Framework Core

Entity Framework Core, Microsoft'un Entity Framework setinin en son sürümüdür. Hafif, genişletilebilir ve Microsoft'un .NET Core çerçevesinin bir parçası olarak çapraz geliştirmeyi destekleyecek şekilde tasarlanmıştır. Entity Framework'ün önceki sürümlerine göre performans iyileştirmeleri yapılmış ve kullanımı daha kolay olacak şekilde tasarlamıştır.EF Core bir nesne ilişkisel eşleyicidir (ORM). Nesne ilişkisel eşleme, bir uygulamanın programlama dilinde tanımlanan nesneler ile ilişkisel veri kaynaklarında depolanan veriler arasında eşleme yapmak için gerekli işi gerçekleştirerek, geliştiricilerin verilerle nesneye dayalı bir şekilde çalışabilmelerini sağlayan bir tekniktir [7].

Entity Framework' ün amacı yazılım geliştiricileri katı SQL sorgularından kurtararak ORM imkânı sağlamaktadır. ORM, ilişkisel veri tabanları yönetim sistemleri (Relational Database Management Systems) ile nesne yönelimli programlama (Object Oriented Programming) arasında köprü görevi yapan ve veri tabanında bulunan verileri yönetilmesini sağlayan bir modelleme tekniğidir. Kısaca Entity Framework veri tabanı elemanlarını veri tabanına bağlı olmadan nesneler halinde kullanmasını sağlamaktadır. Bunun sayesinde yazılım geliştirici katı yani uzun ve karmaşık SQL sorgularını yazmaktan kurtulur. Bunun yanında Entity Framework CRUD (Create , Read , Update , Delete) işlemleri ile uygulama ve veri tabanı arasındaki iletişimin yapılmasını sağlamaktadır [8].

2.2.1. ORM

ORM ise veri tabanı ile nesneye yönelik programlama arasındaki ilişkiyi kuran teknolojidir. Yani Entity Framework, nesne tabanlı programlamada veri tabanındaki tablolara uygun nesneler oluşturma tekniğidir.

ORM için; Veri tabanında yaratmış olunan her bir nesneye karşılık kod tarafında bir nesne oluşturan programlardır. Bu programlar code generation veya shema generation tekniği kullanarak yazılması gereken kodu otomatik üretiyor veya tam tersinde yazılan kod şablonuna uygun database şemasını oluşturmaktadır.

Günümüzde kullanılan birçok ORM aracı bulunmaktadır. Örneğin; Java tabanlı olarak Hibernate, Flex'de Athena Framework, Delphi' de ECO gibi. Entity Framework ise Microsoft tarafından geliştirilen .Net tabanlı bir ORM aracıdır [9].

2.2.1.1. ORM'nin Avantajları

- Hangi programlama dili kullanıyorsa onunla yazmaya devam edile bilinmektedir.
- Veri tabanı ile uygulama arasında soyutlama olmaktadır ve böylelikle veri tabanı tarafında değişiklik yapma zorluğu ortadan kalkmaktadır.
- Sorguların çoğu daha iyi performans göstermektedir. SQL tarafında sorgu yazma ile

kıyaslandığında ORM daha çok zaman kazandırmaktadır [10].

2.2.2. CRUD İşlemleri

2.2.2.1. Veri tabanına Kayıt Ekleme

Veri tabanına kayıt yapabilmek için öncelikle hangi tabloya kayıt eklenecekse o class türünden yeni bir nesne oluşturulmaktadır. Daha sonra bu nesneye özellik bilgileri eklenir ve context nesnesinin Add metodu kullanılarak veri tabanına eklenmektedir. Context nesnesinin SaveChanges metodu kullanılmadan yapılan değişiklikler veri tabanı üzerinde uygulanmaz [9].

2.2.2.2. Veri tabanında Var olan Kayıtları Listeleme

Veri tabanında var olan kayıtlar üzerinde işlem yapabilmek için Context nesnesinden faydalanılmaktadır. Belirli bir tablodaki tüm verileri göstermek için bir foreach döngüsü kurularak tüm veriler sırası ile ekrana getirilebilmektedir [9].

2.2.2.3. Veri tabanında Var olan Belirli Kayıtları Listeleme

Veri tabanındaki tablolarda yer alan tüm verileri listelemek yerine belirli bir özelliği olan verileri görüntülemek istenirse (Bölüm ID si 1 olan bölümde okuyan öğrenciler gibi) context nesnesi ile veri tabanına erişirken bu parametreyi de belirtilmesi gerekmektedir.

2.2.2.4. Veri Güncelleme

Veri tabanında yer alan bir kayda eriştikten sonra o kayıt üzerinde düzenleme (güncelleme) yapmak çok kolaylaşmaktadır. Örneğin Fakülte Adı “Teknoloji Fakültesi” olan kaydın bilgilerini değiştirmek istenirse yapmanız gereken Teknoloji Fakültesi’e ait bilgileri context

üzerinden gerekli nesneye aktarmaktır.

2.2.2.5. Veri Silme

Veri tabanında yer alan bir kayda eriştikten sonra o kayıt üzerinde düzenleme (silme) yapmak çok kolaylaşmaktadır. Fakülteler tablosundaki id si 1 olan kayıt silmek istenirse, o kayda ait bilgilere context üzerinden erişerek silinebilmektedir [9].

2.2.3. DbContext

DBContext, Entity Framework'ün kalbidir. Varlık sınıfları ile veri tabanı arasındaki bağlantıdır. DBContext, veri tabanını sorgulamak ve verileri varlık olarak belleğe yüklemek gibi veri tabanı etkileşimlerinden sorumludur. Ayrıca varlığa yapılan değişiklikleri izler ve değişiklikleri veri tabanında sürdürür [11].

.NET nesnelerini kullanarak verileri sorgulamak, eklemek, güncelleştirmek ve silmek için, önce modelinizde tanımlanan varlıkları ve ilişkileri bir veri tabanındaki tablolarla eşleyen bir model oluşturmanız gerekmektedir.

Bir modelinize sahip olduktan sonra uygulamanızın etkileşimde bulunduğu birincil sınıf System.Data.Entity.DbContext (genellikle bağlam sınıfı olarak adlandırılır) olur. Bir modelle ilişkili bir DbContext kullanarak şunlar yapıla bilinmektedir:

- Sorguları yazma ve yürütme
- Sorgu sonuçlarını varlık nesneleri olarak gerçekleştirme
- Bu nesnelerde yapılan değişiklikleri izleme
- Nesne değişikliklerini veri tabanında geri Sürdür
- Nesneleri bellekten UI denetimlerine bağlama

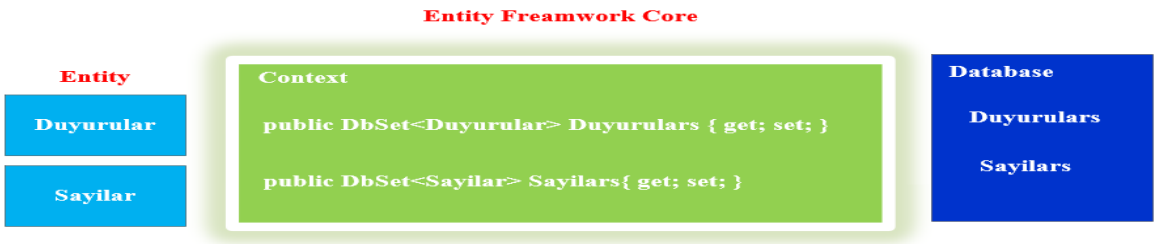
2.3. Entity Framework - Code First Migration

Veri tabanı ile Programlama dili arasında bağ kuran bir tekniktir. Projenizde veri tabanı işlemlerinizi mümkün mertebe Visual Studio tarafında kod yazarak gerçekleştirmenizi sağlayan bir yaklaşımdır. Bu yaklaşım sayesinde veri tabanı ara yüzü ile yazılımcı arasında ilişki minimize edilmektedir [14].

EF çekirdeğinde kodunuzda veri modelleri oluşturulur veya güncellenmektedir. Bu yaklaşıma Code First yaklaşımı denilmektedir. Uygulamanız büyüdükçe modelinizde birçok değişiklik yapılacaktır. Bu, veri tabanınızın modelle senkronize olmamasına neden olmaktadır.

Bu nedenle, modelde değişiklik yaparken veri tabanını güncellemeniz gerekmektedir. Ancak manuel olarak değişiklik yapmak hataya açıktır. Modelden yeni bir veri tabanı da oluşturabiliriz, ancak bu veri kaybına neden olmaktadır.

EF Core geçişleri, değişiklikleri veri tabanına aktarmayı ve ayrıca veri tabanındaki mevcut verileri korumayı kolaylaştırır. Geçişler oluşturmak ve kaldırmak için add-migration ve remove-migration gibi komutlar sağlamaktadır. Geçişler oluşturulduktan sonra, veri tabanını için update-database yazılmaktadır. Ayrıca geçiş komut dosyasını oluşturabilir ve üretim sunucusunda çalıştırılabilmektedir [15].



Şekil 2. 3. Code First - Migrations Akış Diyagramı

2.3.1. Avantajları

- Database First ve Model First ile yapılan her şey yapıla bilinmektedir ve kullanımı basittir.
- Yönetmeniz gereken ve sürekli büyüyen bir edmx modeliniz yoktur.
- Migration dosyaları sayesinde veri tabanının versiyonları kolaylıkla saklanabilir.
- Yapılan tüm değişiklikler framework tarafından saklanmakta ve istenilen versiyona veri tabanı yapısı da uyumlu şekilde dönülebilmektedir.
- Küçük model değişiklikleri herhangi bir veri kaybına neden olmaz.
- Daha fazla özelleştirme seçeneğiniz ve kod üzerinde tam denetime sahipsiniz.
- Veri tabanında manuel değişiklikler büyük olasılıkla kaybolacaktır çünkü kodunuz veri tabanını tanımlıyor.
- Varlıklarınızı istediğiniz gibi çok kolay şekilde, özgürce tasarlana bilinecektir.
- Modellemede hata oluştuğu durumda Framework bunu size hemen bildirecektir [16].

2.3.2. Dezavantajlar

- Görsel bir ara yüz olmadığı için veri tabanını koddan yönetmek sizi zorlayabilir.
- Veri tabanını oluşturabilmek için C# bilmeniz gerekmektedir.

2.3.3. Migration 3 adımlı bir işlem

- Öncelikle model oluşturur veya modellerimizde değişiklik yapmaktadır. Bu noktada yeni bir model oluşturmuş olabilirsiniz veya var olan modellerinizde değişiklik yapmış olabilirsiniz. Böyle bir işlem yaptığınızda mutlaka migration ile database tarafına değişiklik yansıtılmalıdır.
- Migration Ekleme işlemi yapılmaktadır. Database tarafına değişikliği yansıtabilmek için migration eklenmektedir.
- Migration uygulama işlemi yapılmaktadır. Migration uygulamamıza bir isimle eklenince,

henüz işlem database yansıdı demek değildir bunu database'e yansıtabilmek için migration'u database'e push etmeniz gerekmektedir [17].

2.4. Asp .Net Core MVC Area

Asp .Net Core MVC'de bilindiği üzere klasör yapısı sınırlandırılmıştır. Model dosyaları için Models, Controller dosyaları için Controllers ve View dosyaları için Views klasörleri kullanılmaktadır. Bu yapı dışında projeye ihtiyaçlarınıza yönelik sayfalar eklemek mümkün değildir.

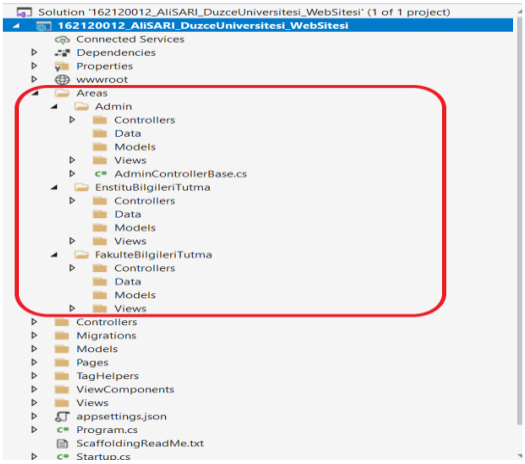
Asp .Net Core Web Form uygulamasında proje dizini içerisinde yeni bir klasör oluşturup istediğiniz ismi verebilir ve klasör içerisindeki sayfalara web tarayıcıdan kolayca erişe bilinmektedir. Bu yönüyle MVC uygulamaları web form uygulamalarından ayrılmaktadır.

Asp.net MVC' de area yapısı bu eksikliği gidermek için kullanılmaktadır. Daha net bir örnekle açıklanacaksa, geliştirilen web sitesinin içeriklerini yönetecek yönetici sayfalarını bir klasör altında toplamak gerekir ve bunun için area (admin paneli için URL: http://localhost:xxxx /admin) kullanılması gerekmektedir [18].

- Alanlar, uygulamanızda ayrı bir modül olan MVC'nin bir parçasıdır.
- Modülleri alansal olarak daldırarak bir seviye uygulaması oluştura bilinmektedir.
- Aynı ad denetleyicisini, birden çok kez görüntülüne bilinmektedir.
- Çalışan, Yönetici, Müşteri vb. alanlar oluşturula bilinmektedir.
- Uygulamanın _Layout'unu içeren MVC gibi paylaşılan bir klasör var. ve _ViewStart ve görünümler klasöründeki _ViewImport görünümleri.

Herhangi bir görünümü yürütürken, _Layout çağırılmaktadır, ancak ondan önce _ViewStart ve _ViewImport çağırılmaktadır. _ViewStart, her görünüme eklemeden her görünüm sayfasına otomatik olarak uygulanan "_Layout" yolunu içermektedir. _ViewImport, birden çok kez

eklemeyi göz ardı etmek için görünüm sayfasında kullanmak istediğiniz tüm etiket yardımcılarını ve ad alanlarını içermektedir [19].



Şekil 2. 4. Multiple Areas in ASP.NET CORE MVC Application

2.4.1. Asp .Net Core MVC Area Özellikleri

Area, web projenizin içerisinde çalışan ve bağımsız özellikleri olabilecek bir yapıdadır. Projenin ana dizininde yer alan klasör yapısının benzeri oluşturacağınız her area içerisinde de olmaktadır.

Örneğin Models, Controllers ve Views klasörleri eklediğiniz area içerisinde yer almaktadır. Ayrıca oluşturduğunuz her area içerisinde kendi web.config dosyası da yer alır. Buradaki web.config dosyası ile bu area için proje genelinden farklı ayarlar tanımlamak mümkündür.

2.5. Asp .Net Core Session

SessionState, kullanıcı uygulama üzerinde işlem yaparken veya gezinirken kullanıcı verilerini depolanmasına yarayan bir durum yönetim nesnesidir. Bir istemciden gelen istekler arasında veriyi tutabilmek için uygulama tarafında tutulmaktadır. Bilgilere ulaşmak için sessionId üzerinden hareket edilir. Kullanıcı her bir istekte, SessionID değeri sunucuya gönderilir o değere ait bilgiler sunucu üzerinden alınır ve işlemler devam etmektedir. Session'ın bilgisinin silinmesi için, kullanıcının browser'ı kapatması veya uygulama üzerinde sonlandırılması gibi işlemlerle silinmesi sağlanabilmektedir. Default olarak session'ın ne kadar süre duracağına

uygulama üzerinden ’de karar vermektedir.

Asp.Net Core, istemciye bir oturum bilgisi(sessionId) içeren bir çerez(cookie) ekler ve oturum süresince işlem yapabilmek için bu değere ihtiyaç duymaktadır. Tarayıcı cookie içerisinde yer alan SessionID ile her istekle birlikte uygulamaya göndermektedir. Uygulama üzerinde istemciye ait bilgileri alabilmek için bu değeri kullanılmaktadır [20].

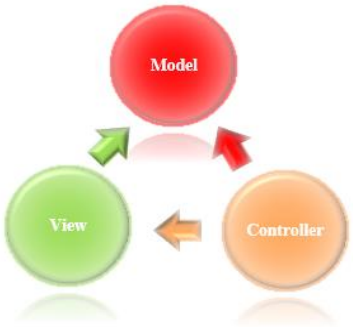
Session ile çalışırken bilinmesi gerekenler;

- SessionId değeri oturum açtığınız browser’a özgüdür. Farklı bir browser’da oturum açtığınızda size farklı bir SessionId değeri verilecektir.
- Browser (tarayıcı) kapattığınızda oturum sonlandırılacaktır.
- Oturum süresi dolmuş ise, uygulama yeni bir yeni bir oturum oluşturur.
- Uygulama açıldıktan sonra en son yapılan istekten sonra belirli bir süre oturumu tutmaya devam eder. Default olarak belirlenmiş zaman aşımı süresi 20 dakikadır. Bu değeri uygulama üzerinden değiştirile bilinmektedir.
- Oturum kapandığında üzerinde tutulan tüm veriler silinmektedir. Kalıcı depolama yöntemi değildir. Hassas bilgiler tutulmamalıdır.

2.6. Asp .Net Core MVC

MVC mimari deseni, bir uygulamayı üç ana bileşen grubu olarak ayırmaktadır: Modeller, Görünümler ve Denetleyiciler. Bu düzen, endişelerin ayrımını elde etmek için yardımcı olur. Bu düzen kullanılarak kullanıcı istekleri, kullanıcı eylemlerini gerçekleştirmek ve/veya sorguların sonuçlarını almak için Modelle çalışmakla sorumlu olan denetleyiciye yönlendirildi. Denetleyici, kullanıcıya görüntülemek için Görünümleri seçer ve gereken tüm Model verilerini sağlar.

Aşağıdaki diyagramda üç ana bileşen ve diğer bileşenlere başvurulan bileşenler aşağıda vermektedir:



Şekil 2. 5. MVC

Tek işi olan bir şeyi (model, görünüm veya denetleyici) kodlama, hata ayıklama ve test etmek daha kolay olduğundan, bu sorumlulukların bu açıklaması uygulamayı karmaşıklık açısından ölçeklendirmeye yardımcı olmaktadır. Bu üç alandan iki veya daha fazlasına yayılan bağımlılıkları olan kodu güncelleştirmek, test etmek ve hata ayıklamak daha zordur. Örneğin, kullanıcı arabirimi mantığı iş mantığından daha sık değişiklik gösterir. Sunu kodu ve iş mantığı tek bir nesnede birleştirilmişse, kullanıcı arabirimi her değiştirildiğinde iş mantığı içeren bir nesne değiştirilmelidir. Bu genellikle hatalara neden olur ve her minimal kullanıcı arabirimi değişikliğinin ardından iş mantığının yeniden test güncelleştirmesi gerekmektedir [22].

2.6.1. Model

Model, MVC'de projenin iş mantığının (business logic) oluşturulduğu bölümdür. İş mantığıyla beraber doğrulama (validation) ve veri erişim (data access) işlemleri de bu bölümde gerçekleştirilmektedir.

Model tek katmandan oluşabileceği gibi kendi içinde birden fazla katmandan da oluşabilir. İç yapılandırma projenin büyüklüğü ile yazılım geliştiricinin planlamasına kalmış bir durumdur. Eğer proje büyük çaplı ise modeli birden çok katmana ayırmak projenin yönetimi açısından faydalı olacaktır.

2.6.2 View

View, MVC'de projenin ara yüzlerinin oluşturulduğu bölümdür. Bu bölümde projenin kullanıcılara sunulacak olan HTML dosyaları yer almaktadır. Projenin geliştirildiği yazılım dillerine göre dosya uzantıları da değişebilmektedir. Projelerin büyüklüğüne göre dikkat edilmesi gereken bir nokta ise, klasörlemedir.

Eğer bir web projesi geliştiriyorsanız, projenin View'larının yer aldığı klasörlerinin hiyerarşisi, ilerleyen dönemlerde karmaşıklığa sebep olmaması için dikkatli yapılmalıdır. Kimi yazılım geliştiriciler web projelerinde HTML dosyaları ile Javascript, CSS ve resim dosyalarını aynı klasör içinde barındırmaktadır. Ufak bir ayrıntı gibi görünse de projenin ilerleyen dönemlerinde ciddi problemler oluşturmaktadır. View'ın bir görevi de kullanıcılardan alınan istekleri controller'a iletmektir.

2.6.3. Controller

Controller, MVC'de projenin iç süreçlerini kontrol eden bölümdür. Bu bölümde View ile Model arasındaki bağlantı kurulmaktadır. Kullanıcılardan gelen istekler (request) Controller'larda değerlendirilir, isteğin detayına göre hangi işlemlerin yapılacağı ve kullanıcıya hangi View'ın döneceği (response) belirtilmektedir.

2.6.4 MVC'nin Yaşam Döngüsü (Life Cycle)

MVC'nin parçaları olan Model, View ve Controller'ın ne olduğu yukarıdaki bölümde anlatılmaktadır. Şimdi bu bilgileri toparlayıp MVC'nin yaşam döngüsünü (çalışma prensibini) incelenmektedir.

Şekil 2. 6. MVC'nin Yaşam Döngüsü

2.6.4.1. HTTP Request

Her ASP.NET MVC uygulamasını görüntülemek istemeniz bir request(istek) tir. Bu istediğinizi HTTP üzerinden IIS tarafından alınmaktadır. Her yaptığınız istek Server tarafından bir yanıtla son bulması gerekmektedir.

2.6.4.2. Routing

ASP.NET MVC uygulamasını her istek yaptığınızda, yaptığınız yanıt UrlRoutingModule HTTP Module tarafından durdurulur. UrlRoutingModule bir isteği durdurduğu zaman, gelen istek RouteTable'dan hangi Controller tarafından üstleneceğine karar verilmektedir.

2.6.4.3. Controller

RouteTable'dan gelen route bilgisine göre Controller hangi Action'ı çalıştıracaksa o View çalıştırılır. View, Controller tarafından render edilmez. Controller tarafından geriye ViewResult döndürülmektedir.

2.6.4.4. ViewResult

ViewResult, View'i render etmek için aktif View Engine'i çağırılmaktadır.

2.6.4.5. ViewEngine

Bir CSHTML dosyayı oluşturduğunuzda içerisindeki script ve markuplar, Razor View Engin tarafından bazı ASP.NET API'lerini sayfalarınızı HTML'e çevirmek için kullanılmaktadır.

2.6.4.6. View

View Engine tarafından HTML'e çevirilen kodlar kullanıcıya sunulmaktadır.

2.6.4.7. Response

HTTP üzerinden View kullanıcıya gösterilmektedir [22].

2.7. ASP NET Core Tag Helper

Tag Helpers, daha okunabilir, anlaşılabilir ve kolay geliştirebilir bir View inşa etmenizi sağlayan, Asp.Net Core ile birlikte HtmlHelpers'ların yerine gelen yapılardır. HtmlHelpers ile tasarlanan sayfaların okunabilirliği çok az ve karmaşık görünen yapılardır. Fakat TagHelpers'larla beraber sayfalar artık daha sade ve okunabilir hale gelmektedir.

2.7.1 TagHelpers ve HtmlHelpers Karşılaştırılması

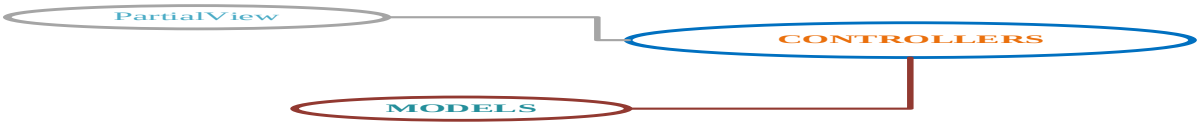
- TagHelper'lar veiw'lardaki kod maliyetini oldukça düşürmektedir.
- HtmlHelpers'ların html nesnelerinin generate edilmesini server'a yüklenmesinin getirdiği maliyeti de ortadan kaldırmaktadır.
- HtmlHelpers'lar da ki progmatik yapılanma, programlama bilmeyen tasarımcıların çalışmasını imkânsız hale getirmekteydi. TagHelpers'lar buradaki kusuru giderdi ve tasarımcılar açısından programlama bilgisine ihtiyaç duyulmaksızın çalışma yapılabilir nitelik kazandırmaktadır [23].

2.8. ASP NET Core ViewComponent

ViewComponent modüler tasarım yapılanmasını oluşturulmasını ve kullanılmasını sağlayan bir yapı olmaktadır. .Net Core mimarisi ile hayatımıza giriş yapmıştır. Esasında PartialView ile aynı amaca hizmet eder fakat aralarında teknik olarak farklar vardır.

Şimdi basit bir MVC yapısı düşünün kullanıcıdan bir istek yapıldığını düşünürsek bunu önce

Controller karşılar daha sonra bir veri taşıyacaksa buna Modelden ulaşır ve en nihayetinde View a ulaşmaktadır. Eğer birde PartialView varsa en son olarak ona ulaşır. Yani kısacası PartialView üzerinde bir veri göstermek istediğinizde en basit döngü yukarıda bahsedilen gibi olmaktadır.



Şekil 2. 7. Partial View

Peki ViewComponent bu konuda ne demektedir. ViewComponent diyor ki beni de modüler olarak parça parça kullanabilirsin fakat eğer bende bir veri göstermek istiyorsan Controller'ı hiç araya katmadan onu yormadan direkt olarak Model ile iletişime geçip veriyi gösterebilirim. İşte PartialView ve ViewComponent arasındaki en temel fark bu olmaktadır.

Şekil 2. 8. View Component

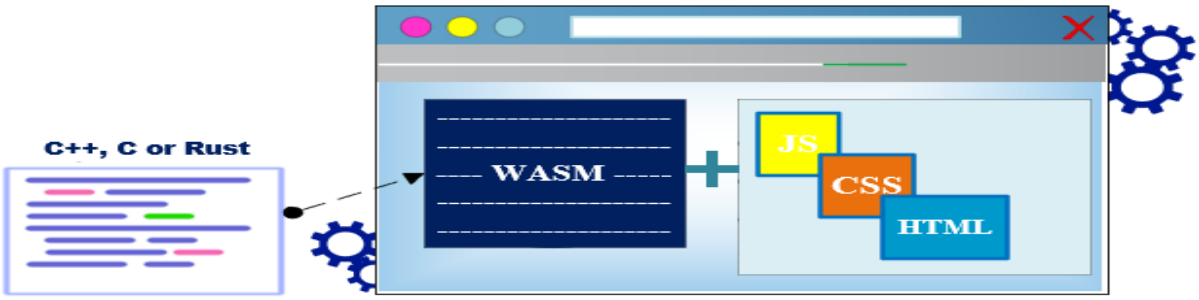
Burada iş Controller üzerinden çıkıyor ve Controller üzerindeki gereksiz yük hafiflemektedir. Bu saye de proje üzerinde yer alan Controller amacından sapmadan işlerini yapabilir hale gelmektedir. Özetle ; PartialView yapılanması ihtiyacı olan dataları Controller üzerinden elde edeceği için Contoller'daki maliyeti artırmakta ve SOLID prensiplere aykırı davranılmasına sebebiyet verebilmektedir. Ayrıca PartialView yapılsak olarak Controller üzerinden beslenmektedir. ViewComponent ihtiyacı olan dataları Controller üzerinden değil, direkt kendi cs dosyasından elde edebilmektedir. Böyle Controller üzerindeki gereksiz maliyeti ortadan kaldırmayı hedeflenmektedir [24].

2.9. Blazor

Blazor; browser üzerinde WebAssembly teknolojisini kullanılarak, ASP.NET Core üzerinde C#, Razor ve HTML kodları ile client-side uygulamalar yapabileceğiniz yeni bir .NET web framework'üdür. SPA oluşturma sürecini basitleştirerek .NET üzerinde full-stack olarak geliştirme yapılmasına yardımcı olmaktadır.

.NET geliştiricisi iseniz, ASP.NET MVC'den aşina olduğunuz Razor syntaxı ve C# programlama dilini kullanarak kolay bir şekilde Blazor uygulamaları yapıla bilinmektedir. Bunu, ASP.NET Core tabanlı Angular, ReactJS gibi bir JavaScript framework'üne benzetilmesi mümkündür. Avantajlarından büyük bir tanesi ise, bunu yaparken herhangi ek bir eklenti yüklememizin gerekmiyor olması. Günümüz modern browserları (mobil browserlar da dahil olmak üzere) Blazor'ı stabil bir şekilde çalıştırabilmektedir.

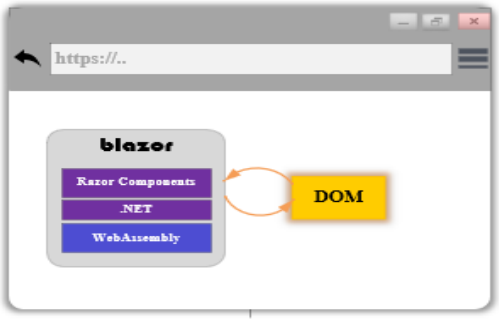
2.9.1. WebAssembly



Şekil 2. 9. WebAssembly

WASM, JavaScript dışındaki programlama dilleri ile yazılan kodun browser üzerinde çalıştırılmasını sağlayan bytecode formatıdır. Böylece klasik JavaScript koduna göre daha fazla performans alınabilmektedir. Yapabildiği şeyi yukarıdaki resimde net olarak görmek mümkün.

Blazor ise .NET alt yapısını WebAssembly üzerine konuşlandırarak, C# ile SPA türünde web uygulamaları yapmaya olanak sağlamaktadır. Blazor'ın WebAssembly ile birleştiği altyapının görseli ise şu şekilde;



Şekil 2. 10. Blazor'ın WebAssembly ile birleştiği altyapının görseli

2.9.2. Server-side ve client-side Blazor

Blazor için şu anda iki farklı model bulunmaktadır. Biri istemci tarafı, diğer ise sunucu taraflı model.

2.9.2.1. İstemci taraflı (client-side) model

İstemci taraflı (client-side) model WebAssembly üzerinde çalışmaktadır. Bir client-side Blazor uygulamasını browser üzerinde açıldığında önce C# kodları ve Razor dosyaları derlenmekte. Derlenen bu dosyalar browser tarafından indirildikten sonra sayfayı çalıştırma işlemi başlamaktadır. Bu sebeple uygulamanın boyutu, bu kısımda oldukça önem teşkil etmiş olmaktadır. Çünkü daha fazla boyut, daha yavaş bir performans demektir.

2.9.2.2. Sunucu taraflı (server-side) modelde

Sunucu taraflı (server-side) modelde, UI güncellemeleri, event yönetimi ve Javascript interop çağrıları SignalR bağlantısı üzerinden yapılmaktadır. Ayrıca .NET altyapısı WebAssembly yerine CoreCLR üzerinde çalışmakta, bu da .NET ekosistemine erişme, debug yapma gibi faydalar sağlamaktadır. Yukarıda bahsedilen derleme ve browser download işlemleri server-side modelde yoktur. Çünkü zaten derlenmiş bir çıktıyı publish almış olunmaktadır. Bu sebeple client-side modele göre daha performanslı çalışmaktadır [25].

2.9.3. Özellikleri

- En önemli özelliği open-source olması ve bu da tamamen ücretsiz olarak kullanılabileceği anlamı taşmaktadır.
- Javascript yerine C# kullanarak web uygulamaları oluşturmaya imkân sağlamaktadır.
- Component denilen, tekrar kullanılabilir yapılar oluşturula bilinmektedir. Bunu partial view gibi düşüne bilinir. Componentlere ViewBag mantığında değişkenler aktarmak mümkündür. Bunları da html attributelerle yapıla bilinmektedir.

- Visual Studio, Visual Studio Code üzerinde veya .NET ortamını sağlayan IDE'ler üzerinde kolayca geliştirme yapıla bilinmektedir.
- Bellek korumalı bir sanal ortamda çalışır ve yerel uygulamalardan neredeyse daha hızlıdır.
- Gerekli tüm SPA özellikleri, güncel bir geliştirici deneyimi için bileşenler, yönlendirme, bağımlılık enjeksiyonu gibi Blazor tarafından desteklenir.
- Zengin IntelliSense ve daha az geliştirme süresi sağlamaktadır.
- Tüm genel tarayıcılar tarafından desteklenmektedir [26].

2.9.4. Blazor Dezavantajları

2.9.4.1. Dosya Büyüklüğü

Javascript ile geliştirilmiş kodlara kıyasla Blazor ile oluşturulmuş uygulama dosyaları daha büyüktür ve diskte daha fazla yer işgal etmektedir. Bu durumda sayfanın ilk açılışında yavaşlamaya sebep oluna bilinmektedir.

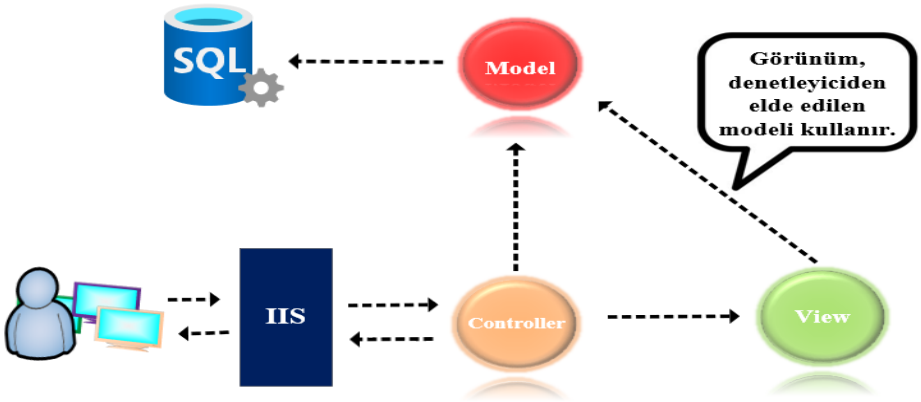
2.9.4.2. Hata Ayıklama (Debugging)

Hata Ayıklama anlamında henüz yeterince gelişmiş olduğunu söylenememektedir [26].

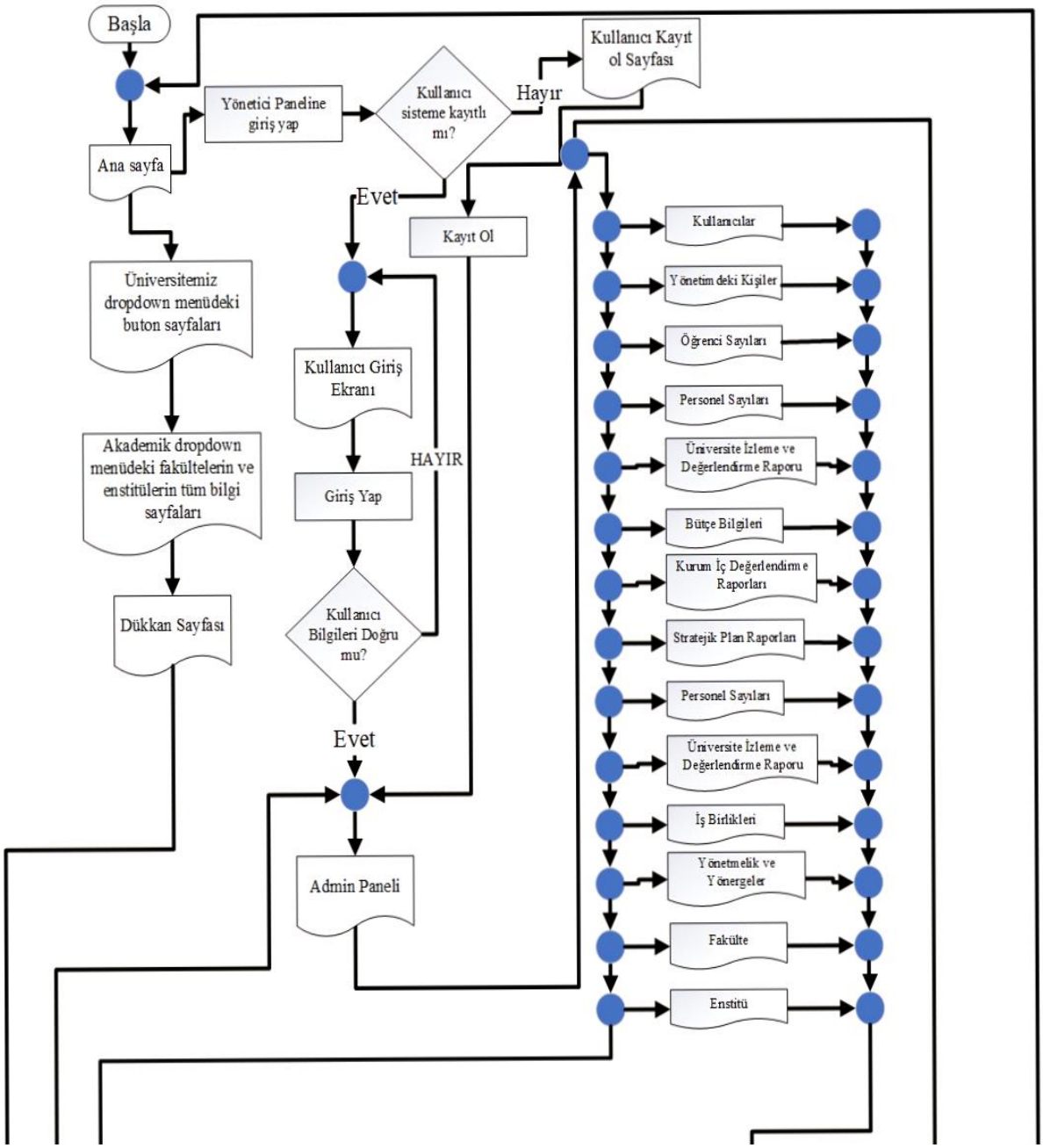
2.10. Çalışmanın Gerçekleştirilmesi

Yapılan bu Asp .Net MVC Core Ortamında Üniversite Web Sayfası Tasarımı çalışmasında veri tabanı olarak Microsoft SQL server 2018 kullanılmaktadır. Oluşturulan web sitesinde ana sayfa, üniversitemiz dropdown menü listesindeki butonlar, akademik dropdown menü'deki fakültelerin, enstitülerin bütün bilgileri ve dükkân mağazası sayfaları bulunmaktadır. Bulunan bu sayfaların bilgilerini yönetici panelinden eklenmektedir. Yönetici paneline giriş yapmak için öncelikle kayıt olunması gerekmektedir. Eğer zaten kaydınız bulunuyorsa admin giriş panelinden giriş yapılır eğer yapılan giriş doğru ise admin paneline yönlendirilmektedir. Admin panelinde Area ve Blazor admin kullanılmaktadır. Ana sayfa işlemleri için blazor admin ile

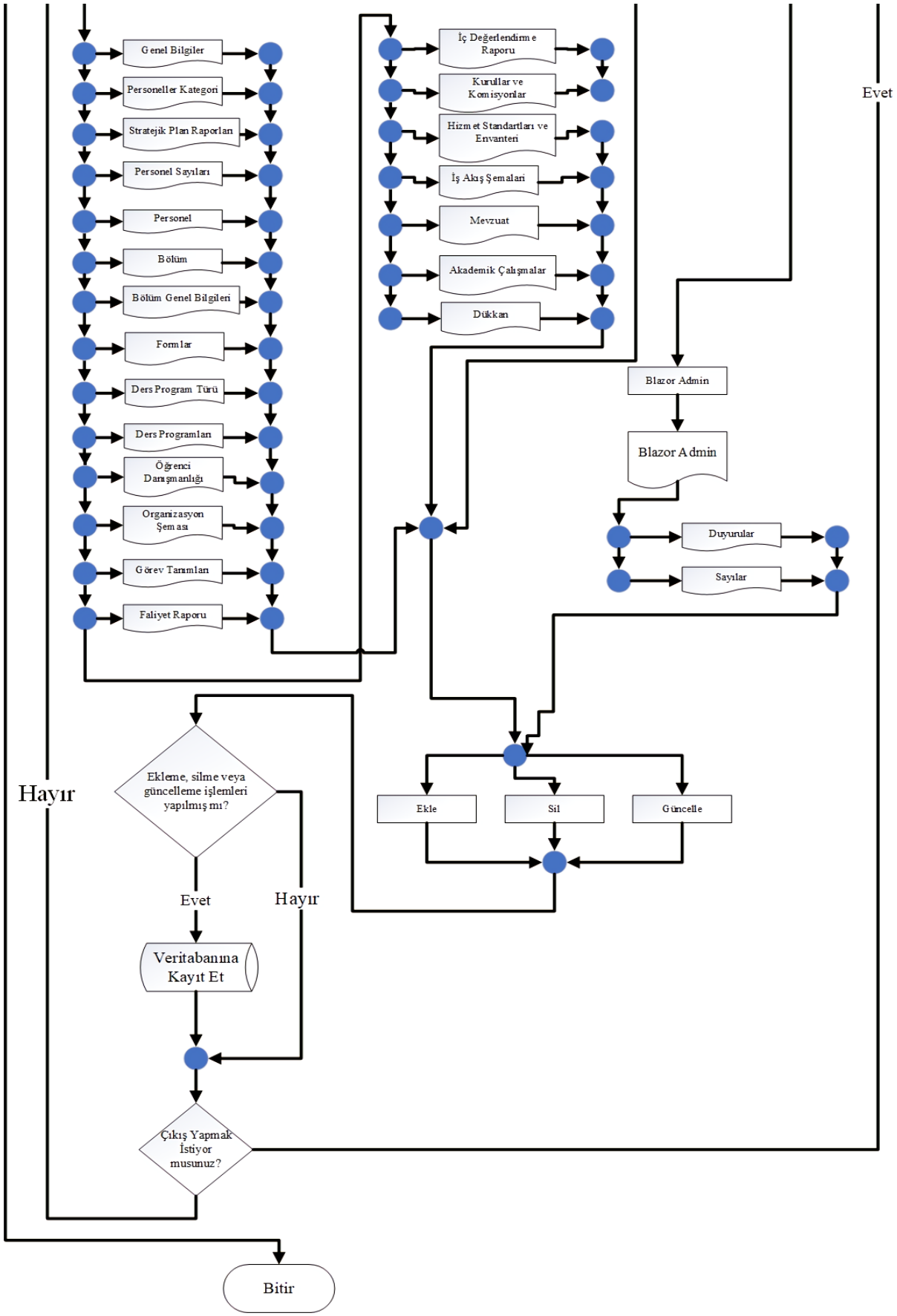
duyurular ve ana sayfada bulunan sayılar tablolarında işlem yapıla bilinmektedir. Ana sayfa dışındaki diğer sayfalar için Area kullanılmaktadır. Bu kısımlarda üyeler yönetimi, yönetimdeki kişiler, öğrenci sayıları, personel sayıları, bölüm program sayıları, değerlendirme raporları, bütçe bilgileri, kurum içi değerlendirme raporları, plan raporları, iş birlikleri, yönetmelik ve yönergeler, fotoğraf galerisi, dükkân ve akademik işlemler yani fakülte bilgileri enstitü bilgileri için işlem yapılması gereken diğer tüm tablolar vardır. Kayıt ekleme, silme, güncelleme işlemleri yaptıktan sonra admin panelinden çıkış yapıla bilinmektedir. Çıkış yapabilmek için session'lar temizlenmektedir. Admin panelinden çıkış yapıldıktan sonra Düzce Üniversitesi web sayfasında sql den çekilen bilgilere ulaşıla bilinmektedir.



Şekil 2. 11. Çalışmanın Blok Şeması



Şekil 2. 12. Çalışmanın Akış Diyagramı 1



Şekil 2. 13. Çalışmanın Akış Diyagramı 2

BÖLÜM 3

SONUÇ

Bu çalışmada Asp.Net MVC Core platformu kullanılarak Düzce Üniversitesi için web sayfası yapılmıştır. Yapılan çalışmada admin paneli için yönetici girişi bulunmaktadır. Yönetici sisteme kayıtlı ise veri tabanına veri ekleme, silme ve güncelleme işlemi yapabilmektedir. Ekleme, silme ve güncelleme işlemlerini için iki farklı sistem kullanılmıştır. Birinci sistemde her tablo, buton kullanıldığında sayfa yenilenir. İkinci sistemde ise tek sayfa (single page) blazor admin kullanılmaktadır. Admin panelinden çıkış yaptıktan sonra sitede ana sayfa, üniversitemiz dropdown menü butonları, akademik dropdown menüdeki fakülteler enstitüler ve dükkân sayfaları bilgileri için veri tabanından veri çekilerek listeleme yapılan sayfalar kullanılmaktadır.

Yapılan bu çalışmada daha fazla verim alınabilmesi için bölüm sekreterleri, gerekli kişiler ile siteyi ziyaret edenler arasında özel görüşmelerin yapılabileceği bir mesaj ekranı eklenebilir. Dükkândan alışveriş yapabilmek için ödeme işlemleri eklenebilir. Öğrencilerin ders çalışırken kullanabilecekleri ek kaynakların bulunduğu bir kütüphane sistemi geliştirilebilir.

KAYNAKÇA

[1] Web Yazılım. (2019, July 11). *Asp .NET core nedir*. TukanAjans. https://tukanajans.com/blog/web-yazilim/net-core-nedir/

[2] Bulut, C. (2019, December 22). *ASP.net core nedir yetenekleri nelerdir*. Argenova https://www.argenova.com.tr/asp-net-core-nedir-yetenekleri-nelerdir

[3] Pomelo Soft. (2021, October 24). *Net core'un temel özellikleri.* Pomelo Soft https://www.pomelosoft.com/blog/asp-net-core-nedir

[4] Arge24. (n.d.). Asp n*et core'un temel özellikleri.* Retrieved October 24,2021 from https://www.arge24.com.tr/asp-net-core-nedir/

[5] Pomelo Soft. (n.d.). *Net core bileşenleri nelerdir.* Retrieved October 24,2021 from https://www.pomelosoft.com/blog/asp-net-core-nedir

[6] Gürsoy, İ. (2019, January 09). *Asp net core ve asp.net core, hangi özelliklere sahiptir.* https://www.ismailgursoy.com.tr/net-core-ve-asp-net-core/ /

[7] Blue Mark Academy. (n.d.). *Asp net core entity freamwork core.* Retrieved October 25,2021 from https://bluemarkacademy.com/egitim/entity-framework-core/

[8] Altunel, M. (2018, November 09). *Entity framework nedir? ne işe yarar?* mehmetaltunel https://www.mehmetaltunel.com/csharp/entity-framework-nedir-ne-ise-yarar/

[9] Kurt, B. (2019, 7 June). *Entity framework nedir? Entity freamwork ne işe yarar?* BekirKurt https://bekirkurt.net/wp-content/uploads/2019/11/VTYS1-Vize-Haz%C4%B1rl%C4%B1k.pdf

[10] Gürsoy, İ. (2021, March 26). *ORM ve EF CORE Nedir?* İsmail GÜRSOY https://www.ismailgursoy.com.tr/orm-ve-ef-core-nedir/

[11] TekTutorialsHub-Logo. (n.d.). *DbContext in Entity Framework Core.* Retrieved October 29,2021 from https://www.tektutorialshub.com/entity-framework-core/ef-core-dbcontext/

[12] TekTutorialsHub-Logo. (n.d.). *DbContext in Entity Framework Core.* Retrieved October 29,2021 from https://www.tektutorialshub.com/entity-framework-core/ef-core-dbcontext/

[13] Peterfeatherstone. (n.d.). *Entity Framework Core'da DbContext nasıl kullanılır.* Retrieved October 29,2021 from https://tr.peterfeatherstone.com/950-how-to-use-the-dbcontext-in-entity-framework-cor

[14] Microsoft. (2021, November 31). *DbContext ile çalışma ve türetilmiş sınıfı tanımlama*. https://docs.microsoft.com/tr-tr/ef/ef6/fundamentals/working-with-dbcontext

[15] Yıldız, G. (2016, March 22). *Entity Framework İle Code First Yaklaşımı.* Gencayyildiz https://www.gencayyildiz.com/blog/entity-framework-ile-code-first-yaklasimi/

[16] TekTutorialsHub-Logo. (n.d.). *EF core migrations.* Retrieved October 29,2021 from https://www.tektutorialshub.com/entity-framework-core/ef-core-migrations/

[17] Doğan, K. (2019, February 03). *Code First Yaklaşımı avantajları, dezavantajları*. Medium https://medium.com/@kdrcandogan/code-first-yakla%C5%9F%C4%B1m%C4%B1-736a4ea6c85b

[18] Gürsoy, İ. (2021, April 10). *Model class ve migration nedir?* İsmail GÜRSOY https://www.ismailgursoy.com.tr/model-class-ve-migration-nedir/

[19] TeknologWeb. (2017, October 03). *Asp.net mvc area kullanımı.* Tekno Log Web https://www.teknologweb.com/asp-net-mvc-area

[20] Saini, K. (2020, October 13). *What are areas?* c-sharpcorner https://www.c-sharpcorner.com/article/working-with-areas-in-asp-net-core-3-1/

[21] Vuranok, M. (2020, June 03). *Asp.net core mvc state management bölüm.* Mshowto https://www.mshowto.org/asp-net-core-mvc-state-management-bolum-2.html

[22] SENYURT, B.S. . (2004, December 30). *Derinlemesine session kullanımı.* https://buraksenyurt.com/post/Derinlemesine-Session-Kullanc4b1mc4b1-1-bsenyurt-com-dan

[23] Smith, S. (2021, October 03). *ASP.NET Core MVC'ye Genel Bakış.* Microsoft https://docs.microsoft.com/tr-tr/ef/ef6/fundamentals/working-with-dbcontext

[24] Ayar, E.C. .(2021, March 12). *TagHelpers nedir? Nasıl kullanılır?* https://emrecanayar.wordpress.com/2021/03/12/taghelpers-nedir-nasil-kullanilir/

[25] Ayar, E.C. .(2021, April 25). *ViewComponent nedir? Nasıl oluşturulur? Nasıl kullanılır?* https://emrecanayar.wordpress.com/2021/03/12/taghelpers-nedir-nasil-kullanilir/

[26] Kocabuga, E. (n.d.). *Blazor nedir?* Retrieved December 26,2021 from https://erhankocabuga.com/blazor-tarayici-tabanli-asp-net-core-web-uygulamalari

[27] İŞLEYİCİ, B. (2019, December 12). *WebAssembly ve Blazor nedir?*. Medium https://medium.com/batech/webassembly-ve-blazor-nedir-dd5de1f60dc