BİLECİK
ŞEYH EDEBALİ ÜNİVERSİTESİ

Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

DÜŞÜK BAŞARIMLI WEB UYGULAMASI PROBLEM ANALİZİ VE İYİLEŞTİRME ÇALIŞMASI: ÖĞRENCİ BİLGİ SİSTEMİ DERS SEÇME MODÜLÜ ÖRNEĞİ

Uğur TALAŞ
Yüksek Lisans Tezi

Tez Danışmanı
Dr. Öğr. Üyesi Salim CEYHAN

BİLECİK, 2019
Referans No: 10249856

BİLECİK
ŞEYH EDEBALİ ÜNİVERSİTESİ

Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

DÜŞÜK BAŞARIMLI WEB UYGULAMASI PROBLEM ANALİZİ VE İYİLEŞTİRME ÇALIŞMASI: ÖĞRENCİ BİLGİ SİSTEMİ DERS SEÇME MODÜLÜ ÖRNEĞİ

Uğur TALAŞ
Yüksek Lisans Tezi

Tez Danışmanı
Dr. Öğr. Üyesi Salim CEYHAN

BİLECİK, 2019

BILECIK
SEYH EDEBALI UNIVERSITY

Institute of Science and Technology
Department of Computer Engineering

PROBLEM ANALYSIS AND IMPROVEMENT STUDY OF LOW-PERFORMANCE WEB APPLICATION: AN EXAMPLE OF STUDENT INFORMATION SYSTEM COURSE SELECTION MODULE

Uğur TALAS
Master's Thesis

Thesis Advisor
Assist Prof, Salim CEYHAN

BILECIK, 2019

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS
JÜRİ ONAY FORMU

Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulunun 08/04/2019 tarih ve 21-13 sayılı kararıyla oluşturulan jüri tarafından 30/04/2019 tarihinde tez savunma sınavı yapılan Uğur TALAŞ'ın "Düşük Başarımlı Web Uygulaması Problem Analizi ve İyileştirme Çalışması: Öğrenci Bilgi Sistemi Ders Seçme Modülü Örneği" başlıklı tez çalışması Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak oy birliği ile kabul edilmiştir.

Jüri

Üye (Tez Danışmanı) : **Dr. Öğr. Üyesi Salim CEYHAN**

Üye : Prof.Dr. Resul KARA

Üye : Prof.Dr. Cihan KARAKUZU

Bilgisayar Mühendisliği Ana Bilim Dalı Başkanı:

ONAY

Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulunun

…./…./…..... tarih ve ……/…..... sayılı kararı.

İMZA/ MÜHÜR

TEŞEKKÜR

Tez çalışmamın başından sonuna kadar emeği geçen saygıdeğer hocam ve danışmanım Dr. Öğr. Üyesi Salim CEYHAN'a, yüksek lisans eğitimime devam edebilmem hususundaki desteklerinden dolayı başta daire başkanım Murat FİDAN'a, şube müdürüm Öğr. Gör. Yusuf MUŞTU'ya ve tüm BŞEÜ bilgi işlem dairesi çalışanlarına, teknik konulardaki tecrübelerinden faydalandığım Öğr. Gör. Musa TURKAN ve Öğr. Gör. Ahmet KALA'ya projenin analiz ve test aşamalarında yoğun mesai harcayan Yavuz BİÇİCİ ve Ercan İNAN'a, sunucu yapılandırmasında bana yardımcı olan Semih KARACA ve Hüseyin PARMAKSIZ'a, Öğrenci Bilgi Sistemi projesinin doğuşundan bugünlere gelmesinde büyük emeği olan öğrenci işleri daire başkanı Sezer KUYUCU ve tüm OBS çalışma ekibine, eğitim hayatım boyunca desteğini ve sevgisini her zaman hissettiğim hayat arkadaşım, eşim Fatma Nur TALAŞ'a, biricik kızım İkra TALAŞ'a, çalışmalarımda bana inanan aileme, rahmetli babam Bilal TALAŞ'a, annem Veliye TALAŞ'a, abim Mesut TALAŞ'a, kayın babam Resul TETİK'e, kayın validem Durdu TETİK'e ve kardeşim Ezgi TETİK'e, ayrıca çalışmaktan yorulduğum dönemlerde benimle vakit geçirip bana moral ve destek olan arkadaşlarım Yasin ADIGÜZEL, Semih ÇAĞLI ve Semih AKKERMAN'a teşekkür ederim.

BEYANNAME

Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kılavuzu'na uygun olarak hazırladığım bu tez çalışmasında, tez içindeki tüm verileri akademik kurallar çerçevesinde elde ettiğimi, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun olarak sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

30/04/ 2019

Uğur TALAŞ

DÜŞÜK BAŞARIMLI WEB UYGULAMASI PROBLEM ANALİZİ VE İYİLEŞTİRME ÇALIŞMASI: ÖĞRENCİ BİLGİ SİSTEMİ DERS SEÇME MODÜLÜ ÖRNEĞİ

ÖZET

Üniversitelerin tamamında, öğrencinin tüm eğitim süreçlerinin takip edildiği öğrenci bilgi otomasyon sistemleri bulunmaktadır. Ders seçme haftalarında yaşanan aşırı yoğunluk nedeniyle, birçok üniversitede sunucular gelen talep trafiğine cevap veremediğinden sistemin durmasına yol açmaktadır. Bu çalışma kapsamında, Bilecik Şeyh Edebali Üniversitesinin kullanmış olduğu öğrenci bilgi sistemi yazılımı incelenip, yazılım mimarisinde kullanılan sorgulama teknolojilerinde ve sunucu yapılandırmasında geliştirmeler yapılarak bu problemin önüne geçilmiştir. Yeni geliştirilen sistemde ASP.Net mimarisi yerine MVC mimarisi kullanılmıştır. Bu mimariyle birlikte tasarım kısmında Bootstrap kullanılmıştır. Tasarım farklı boyutlardaki ekranlarda uyumlu çalışması sağlanmıştır. Eski sistem incelendiğinde birçok hesaplama C#'ta, yani sunucu tarafında yapılıyorken sunucuya binen yükü istemcilere taşımak amacıyla hesaplamaların birçoğu JavaScript'te alınmıştır. Bu sayede yapılacak hesaplamaların her birini sunucu değil öğrencinin kullandığı bilgisayarın yapması sağlanmıştır. Bu hesaplamaları yapmak için kullanılan verilerin birbirinin sonucuna bağlı olmayan karmaşık sorguları multi-thread yöntemi ile paralel şekilde sorgulanarak işlemlerin hızlandırılması sağlanmıştır. Sistemde bir kullanıcının işlem yapması için geçen süre geliştirmeler sayesinde yaklaşık 10 kattan fazla zaman kazancı elde edilerek, IIS'te oluşan kuyrukta bekleme süreleri azaltılıp sistemin hataya düşmesi engellemiştir. Yapılan yazılımsal geliştirmelere ek olarak, uygulama tek bir sunucu üzerinde çalışırken Load Banlance işlemi yapılarak gelen client yoğunluğu dört sunucuya dağıtılmış ve sunucuların önüne talepleri karşılayan bir balancer sunucusu konulmuştur. Bu sayede tekrar benzeri bir problem ile karşılaşılırsa arkada worker olarak çalışan sunucuların sayısı arttırılıp sistemin ayakta kalması sağlanmıştır. Genel olarak, hata ve uygulanan çözüm yöntemine bakıldığında sadece öğrenci otomasyonunda ders seçme modülünde değil yoğunluğa bağlı problem yaşayan tüm otomasyon yazılımları için bu çözümlerin uygulanabilir olduğu düşünülmektedir.

Anahtar Kelimeler: MVC; Load Balance; Multi Thread; Yazılım Performansı; Otomasyon

PROBLEM ANALYSIS AND IMPROVEMENT STUDY OF LOW-PERFORMANCE WEB APPLICATION: AN EXAMPLE OF STUDENT INFORMATION SYSTEM COURSE SELECTION MODULE

ABSTRACT

All of the universities have student automation systems in which all of the student's education processes are monitored. It is known that in the course selection week at the start of the semester, these systems or underlying structure usually cannot handle the high traffic and service outages are experienced. In this study, the student information system used in Bilecik Seyh Edebali University is re-analyzed and service disruption eliminated by optimizing software architecture, SQL query structure and server configuration. MVC architecture is used instead of Web Form architecture in the re-designed system. Design of the front-end is implemented with Bootstrap and pages are designed to be responsive for different window, screen sizes. While analyzing the old system it is seen that most calculations are done in C# back-end and these calculations are moved to client side by using JavaScript. This allowed the calculations to be performed by the student computers instead of web server. Complex database queries that are not depended on each other's results for these calculations are queried in parallel with multi-threading method in order to achieve speed. Typical system delay for an operation to this eliminated service outage by reducing waiting time in the queue for IIS. In addition to software improvements, a load balancer application is employed in order to distribute the client traffic to four servers with a balancer server in front of them instead of running the software in a single server. Even if high client traffic leads to problems, this method allows the system to operate without service interruption by adding additional worker servers in the load-balancing configuration. The solutions described here for eliminating the service outage for the student automation system can be applied to all automation systems that suffer from same problem due to high volumes of traffic.

Key Words: MVC; Multi-thread; Load Balance; Software Performance; Automation

İÇİNDEKİLER

Sayfa No

TEŞEKKÜR .....
BEYANNAME .....
ÖZET ..... **I**
ABSTRACT ..... **II**
İÇİNDEKİLER ..... **III**
SİMGELER VE KISALTMALAR ..... **VIII**
ÇİZELGELER DİZİNİ ..... **V**
ŞEKİLLER DİZİNİ ..... **VI**
1 GİRİŞ **1**
1.1 Literatür Özeti 2
1.2 Çalışmanın Kapsamı 3
2 PROBLEMİN TESPİTİ VE İNCELENMESİ **4**
2.1 Yaşanan Problem 4
2.2 Mevcut Sistemin İncelenmesi 5
2.3 Sunucu yapısının incelenmesi 5
2.4 Yazılım teknolojisinin ve mimarisinin incelenmesi 6
2.5 Sistemde Tespit Edilen Aksaklılar 8
3 KULLANILAN TEKNOLOJİLER **10**
3.1 Load Balance 10
3.2 ASP.NET MVC 13
3.2.1 ASP.NET MVC yaşam döngüsü 16
3.2.2 Razor 16
3.3 Multi Thread 17
3.4 Entity Framework 20
3.5 JavaScript 22
3.6 Bootstrap 24
4 DERS SEÇME MODÜLÜNÜN HAZIRLANMASI **27**
4.1 Yazılım Mimarisi Tasarımı 28
4.2 Sunucu Tasarımı 29
4.3 Veri Tabanı İyileştirmesi 31

4.4 Use Case Diyagramları 33
4.5 Aktivite Diyagramları 35
4.5.1 Login aktivitesi 35
4.5.2 Öğrenci bilgileri getir aktivitesi 37
4.5.3 Akademik form aktivitesi 38
4.5.4 Kaydet aktivitesi 40
4.5.5 Ders seçmemiş yap aktivitesi 42
4.5.6 Öğrenci ara aktivitesi 43
4.5.7 Kayıt onay aktivitesi 44
4.6 İşletilen Kurallar 44
4.6.1 Akts sınırı kontrolü 45
4.6.2 Çakışma kontrolü 45
4.6.3 Kontenjan doluluk kontrolü 47
4.6.4 Harç hesabı 48
4.6.5 Önkoşul kontrolü 48
4.6.6 Alttan almaya zorlama kontrolü 49
4.7 Ara Yüz Tasarımı 50
4.7.1 Login 51
4.7.2 Banner 52
4.7.3 Anasayfa 53
4.7.4 Öğrenci bilgileri gösterimi 53
4.7.5 Ders gösterimi ve kontenjan gösterimi 54
4.7.6 Seçilen derslerin gösterimi 57
4.7.7 Raporlar 59
4.7.7.1 Akademik form 59
4.7.7.2 Kayıt onay raporu 61
4.8 Log İşlemleri 61
5 KARŞILAŞTIRMALI PERFORMANS ANALİZİ 64
6 SONUÇLAR 68
KAYNAKLAR 70
ÖZ GEÇMİŞ .....

ÇİZELGELER DİZİNİ

Sayfa No

Çizelge 1. Ders seçme veri hazırlanma süreleri performans tablosu 64
Çizelge 2. Kaydet aktivite tamamlanma süreleri performans tablosu 65
Çizelge 3. Tarayıcı bağlantı sayıları tablsou 65
Çizelge 4. Tarayıcı sayfa yüklenme süresi 66
Çizelge 5. Kod satır sayıları tablosu 66
Çizelge 6. Akademik form açılma süreleri performans tablosu 67

ŞEKİLLER DİZİNİ

Sayfa No

Şekil 2.1. Mevcut otomasyon sunucu yapısı (* Referans verilmeyen tüm çizimler draw io programı ile çizilmiştir). 5
Şekil 2.2. Mevcut sistemin mimarisi. 7
Şekil 3.1. Load Balance yapısı (http://tutorials.jenkov.com/images/software-architecture/load-balancing-1.png). 12
Şekil 3.2. ASP.NET bileşenleri. 13
Şekil 3.3. APS.NET MVC mimarisi (http://www.webdevelopmenthelp.net/2013/10/ 14
Şekil 3.4. Multi thread çalışma süreleri (http://www.ntu.edu.sg/home/ehchua/programming/java/j5e_multithreading.html). 18
Şekil 3.5. Çift çekirdekli bir işlemcide multi thread çalışma süreleri (http://www.ntu.edu.sg/home/ehchua/programming/java/j5e_multithreading.html). 18
Şekil 3.6. Entity framework mimarisi. 20
Şekil 3.7. İstemci ve sunucu taraflı çalışan kodlar. 22
Şekil 3.8. AJAX çalışma yapısı (http://javascript-coder.com/tutorials/re-introduction-to-ajax.phtml). 23
Şekil 3.9. Uygulamanın açıldığı ekran çözünürlük sayıları grafiği. 25
Şekil 4.1. Mevcut otomasyon yazılım mimarisi. 27
Şekil 4.2. Yeni uygulamanın yazılım mimarisi. 28
Şekil 4.3. Load balance sunucu yapısı 30
Şekil 4.4. Ders bilgilerinin tutulduğu tabloların ilişki yapısı. 31
Şekil 4.5. Ders bilgilerinin tutulduğu tablolarda yapılan ilişkisel düzenleme. 33
Şekil 4.6. Use Case diyagramı. 34
Şekil 4.7. Login aktivite diyagramı. 36
Şekil 4.8. Öğrenci bilgileri getir aktivite diyagramı. 37
Şekil 4.9. Akademik form aktivite diyagramı. 39
Şekil 4.10. Kaydet aktivite diyagramı. 41
Şekil 4.11. Ders seçmemiş yap aktivite diyagramı. 42
Şekil 4.12. Öğrenci ara aktivite diyagramı. 43

Şekil 4.13. Kayıt onay aktivite diyagramı. 44
Şekil 4.14. AKTS sınırı aşma uyarısı 45
Şekil 4.15. Devam zorunlu derslerim gösterimi. 46
Şekil 4.16. Çakışma uyarısı. 47
Şekil 4.17. Kontenjan doluluk uyarısı. 47
Şekil 4.18. Alttan almaya zorlama kuralı öğrenci uyarısı. 50
Şekil 4.19. Alttan almaya zorlama kuralı yönetici uyarısı. 50
Şekil 4.20. Login ekranı. 51
Şekil 4.21. Banner. 52
Şekil 4.22. Arama ekranı. 52
Şekil 4.23. Ana sayfa ekranı. 53
Şekil 4.24. Öğrenci bilgilerinin gösterimi. 54
Şekil 4.25. Öğrenci resmi görüntüleme ekranı. 54
Şekil 4.26. Açılan dersler ve yarı yılların gösterimi. 55
Şekil 4.27. Seçmeli grupların gösterimi. 55
Şekil 4.28. Derse verilen kontenjanların gösterimi. 56
Şekil 4.29. Seçmeli derslerde yerine işlemi 57
Şekil 4.30. Seçilen derslerin takvim görünümü. 58
Şekil 4.31. Seçilen derslerin liste gösterimi. 58
Şekil 4.32. Akademik formun görünümü. 60
Şekil 4.33. Kayıt onay raporu. 61
Şekil 4.34. Log izleme ekranı 63

SİMGELER VE KISALTMALAR

OBS	: Öğrenci Bilgi Sistemi / Öğrenci Bilgi Otomasyonu
AKTS	: Avrupa Kredi Transfer Sistemine karşılık gelen kredi
IIS	: Internet Information Services
YÖK	: Yüksek Öğretim Kurumu
JS	: Java Script
CSS	: Cascading Style Sheets
SQL	: Structured Query Language
MSSQL	: Microsoft Structured Query Language
ASP.NET	: Active Server Page
MVC	: Model View Controller
RAM	: Random Access Memory
CPU	: Central Processing Unit
UML	: Unified Modeling Language
HTML	: Hyper Text Markup Language
GB	: Gigabyte
GHZ	: Giga Hertz
DNS	: Domain Name System
HDD	: Hard Disk Drive
LINQ	: Language Integrated Query
HTTP	: Hyper Text Transfer Protocol
HTTPS	: Hyper Text Transfer Protocol Secure
SSL	: Secure Sockets Layer
UDP	: User Datagram Protocol
TCP	: Transmission Control Protocol
DB	: Data Base
URL	: Uniform Resource Locator
JSON	: JavaScript Object Notation
XML	: Extensible Markup Language
CRUD	: Create Read Update Delete
VS	: Visual Studio
ORM	: Object Relational Mapping

DLL	:	Dynamic Link Library
MS	:	Mili Saniye
SN	:	Saniye

1 GİRİŞ

Günümüzde internetin yaygınlaşmasıyla birlikte web uygulamalarının sayısı artmıştır. Web programlarının yaygınlaşmasıyla birlikte kâğıt üzerinde yapılan evrak işlemleri web uygulaması üzerinden takip edilmeye başlanmıştır. Kamu kurumlarında, endüstride, ticarette ve benzeri birçok alanda yapılan işleri yürütmek, kayıt altına almak raporlamak gibi tüm süreçleri yönetmek için web üzerinde otomasyon sistemleri geliştirilmektedir. Bu otomasyonlar sayesinde iş gücüden tasarruf ve son kullanıcıya kaliteli hizmet verilmesi sağlanmaktadır.

Günümüzde üniversitelerin tamamında öğrencilerin kayıt işlemlerinden başlayıp, mezuniyete kadar tüm süreçlerin ve verilerin işlenip tutulduğu web tabanlı öğrenci bilgi otomasyonları bulunmaktadır. Bu otomasyonlar içerisinde, öğrencilerin her eğitim döneminde alacakları derslerini belirleyip, dönemlik kayıt yeniledikleri ders seçme modülleri bulunmaktadır. Bu modül öğrenci bilgi otomasyonlarının belirli zaman aralığında en yoğun kullanılan modüllerinden biridir. Birçok üniversitede yoğun kullanılan ders seçme haftalarında sunucular gelen talep trafiğine cevap veremediğinden sistemin durması ve uygulamanın hizmet verememesi gibi problemler yaşanmaktadır.

Bu çalışmada B.Ş.E.Ü., öğrenci bilgi sistemi, ders seçme modülünün en yoğun kullanıldığı kayıt yenileme haftalarında aşırı yoğunluğa bağlı olarak sistemde yaşanan problemlere çözüm aranmıştır. Mevcut otomasyon sunucu yapılanması, yazılım teknolojisi ve yazılım mimarisi açılarından incelenmiş ve çözüm planlanmıştır. Planlanan çözüm kapsamında MVC mimarisi kullanılarak ders seçme modülü yeniden geliştirilmiştir. Yeni geliştirilen sistemde sistemi hızlandırmak adına veri sorgulamaları çok kanallı programlama (multi thread) teknolojisi ile paralel şekilde sorgulanmış ve sistemin son kullanıcıya hızlı cevap vermesi için çalışmalar yapılmıştır. Bazı hesaplamalar istemci bilgisayarlarına taşınıp server tarafında yapılan işlem adımı azaltılmıştır. Uygulama tarafında baştan sona yapılan yenilemenin yanında sunucu yapılanmasında yük dengeleme (load balance) yöntemine geçilmiştir. Bu yöntem sayesinde paralel çalışan sunucularda aynı uygulama çalıştırılarak birden fazla sunucunun ders seçme modülüne hizmet vermesi sağlanmıştır.

Problemin özüne ve çözümde uygulanan yönteme bakıldığında sadece Bilecik Şeyh Edebali Üniversitesi öğrenci bilgi sistemi özelinde değil yoğunluğa bağlı olarak

sunucuların hizmet veremediği tüm otomasyonlar için genel bir çözüm örneği teşkil etmektedir.

Yapılan çalışmalar sonucunda ders seçme modülü 2017 – 2018 eğitim yarıyılı bahar döneminde devreye alınmış olup canlı ortamda sistem test edilmiş ve sonuçlar gözlemlenmiştir.

1.1 Literatür Özeti

Literatürde bu tez çalışmasındaki konu olarak ele alınan problemi ya da benzeri problemleri direkt olarak inceleyip, problemi bütünüyle ele alan, çözüm önerisinde bulunan bir çalışma bulunamamıştır. Ancak genel olarak problemin çözülmesi için kullanılan teknolojiler özelinde çalışmalar yapılmıştır. Bu çalışmada literatürdeki bu yapılan çalışmalardan faydalanılıp bütüncül bir çözüm elde edilmiştir.

Yazılım performansı için hizmet verecek bir sistemin tüm bileşenlerini; sunucularını, ağ yapılarını, yazılım geliştirme süreçlerindeki tüm adımları ve araçları, rolleri, aktiviteleri ve etkileşimde bulunarak, sistemin bütünü performans temelli bir mimaride oluşturulmalıdır (Sevinçok ve Ünalır, 2013).

Sahip olunan sunucular arasında yükü dengeli dağıtmak, sistemi verimini ve üretkenliğini arttırırken, çalışma süresini azaltarak daha kısa zamanda daha hızlı sonuçlar elde etmeye olanak sağlamaktadır. Bu kapsamda, gelen veriyi en kısa zamanda işleyebilmek ve yüksek üretkenlik elde edebilmek için, yükü mümkün olduğunca eşit dağıtabilmek gerekmektedir (Dalabasmaz 2018).

Multi Threading birden fazla ipliğin tek bir işlem bağlamında yer almasına izin verir. Bu iplikler işleme ayrılmış olan kaynakları paylaşırlar ancak her biri bir sorumluluğu üstlenmek üzere bağımsız hareket ederler. Böylece, çok iplikli programlama modeli geliştiricilere anlık yürütmenin kullanılabilir bir soyutlamasını sağlar. Sorumlulukların iplikler arasında paylaşılması sayesinde işlem aynı anda birden fazla işi ele alabilir. Ayrıca, çok iplikleme tek bir işlemin birden fazla işlemcisi olan ya da birden fazla çekirdeği olan bilgisayar sistemlerinde paralel şekilde çalışmasına izin verir. (Alan 2011).

Dağıtık Sistemlerde (Distributed Systems) toplam işin mevcut sistemlere en iyi biçimde dağıtılması, başarımı en üst düzeye çekecektir. Yük dengeleme (load balancing) adı verilen bu işlem sistemin dağıtık tasarlanmasını sağlamaktadır. (Berka, ve Vajteršic, 2011).

1.2 Çalışmanın Kapsamı

Bu çalışma kapsamında yoğunluğa bağlı durma problemi yaşayan web uygulamalarına çözüm aranmıştır. Bu tip problem yaşayan B.Ş.E.Ü ders seçme modülü örneği üzerinde problem tespiti yapılıp, projede yeniden yazılıp mevcut sistem ile karşılaştırılarak performans analizi yapılmıştır.

Tezin birinci bölümünde giriş, literatür taraması ve gerçekleştirilen tez çalışmasının kapsamı hakkında bilgiler verilmiştir.

İkinci bölümde tez kapsamında örnek olarak ele alınan uygulamamın problemi tespit edilmiştir. Mevcut uygulamanın yapısı hakkında teknik inceleme yapılıp bu konu hakkında bilgiler verilmiştir.

Üçüncü bölümde gerçekleştirilen yeni uygulamada kullanılan teknolojiler hakkında bilgiler verilmiştir.

Dördüncü bölümde uygulamanın mimarisi ve sunucu yapılanmasından başlayarak, use case diyagramları, aktivite diyagramlarıyla birlikte projede işletilen tüm kuralların detayı ve projenin ara yüz çalışmalarına kadar tüm geliştirme süreçleri hakkında detaylı bilgi verilmiştir.

Beşinci bölümde yeni geliştirilen modül devreye alındıktan sonra yazılımın performansı ile ilgili ölçümler yapılıp eski sistemin ölçülen verileriyle kıyas edilerek karşılaştırmalı performans analizi yapılmıştır.

Altıncı bölümde yapılan çalışmadan elde edilen sonuçlar ve gelecekte yapılabilecek geliştirmeler için öneriler hakkında bilgiler verilmiştir.

2 PROBLEMİN TESPİTİ VE İNCELENMESİ

Bu çalışma kapsamında düşük başarımlı web uygulamalarındaki sorunları önlemek amacıyla ele alınan örnekte, ilk olarak sistemin durmasına sebep olan alt problemlerin tespiti yapıldı. Daha sonra, uygulamanın donanımsal olarak bulunduğu sistem ve yazılımsal yapısı aşağıda verildiği gibi incelenerek uygun çözüm yolları aranmıştır.

2.1 Yaşanan Problem

Kayıt yenileme haftalarında, ders seçme modülünün öğrenciler için kullanıma açılmasıyla birlikte birçok öğrenci aynı anda sisteme girip ders seçmek istemektedir. Sunucuya gelen istemci talep sayısı arttıkça sunucunun cevap süresi de artmaya yani sunucudaki işlemler yavaşlamaya başlar. Son kullanıcı yaptığı işlemin sonucunu daha geç görmeye başlar. Bu sırada sunucu gelen taleplere daha hızlı cevap verebilmek için işlemci kullanım düzeyini maksimuma yani %100'e kadar arttırır ve işlem yaparken her yeni gelen talebi IIS kuyruğuna ekler ve sırasıyla işler. Mevcut sistemde öğrenci sisteme girdiğinde verilerinin getirilmesi yaklaşık 1400ms sürmektedir. IIS kuyruğunda işlemi sıraya alınan son kullanıcı bekleme süresi uzadıkça sayfayı yenileyerek sunucuya ikinci, üçüncü talebi göndermektedir. Her bir öğrenci defalarca talep gönderdiğinde IIS kuyruğu sistemdeki mevcut öğrenci sayısından çok daha fazla istemci talebine maruz kalmaktadır. IIS kuyrukta biriken işleri tamamlamak için kullandığı işlemci gücünü arttırdıkça sunucu %100 işlemci performansı ile çalıştığında bir süre sonra IIS'i durdurmaktadır. IIS durduğu anda o anda sistemdeki tüm kullanıcıların işlemleri yarım kalmaktadır. IIS yeniden başladığında ise sistemde işlemini bitirememiş kullanıcılar tekrar bir anda işlem yapmaya çalışır ve tekrar IIS'teki işlem kuyruğu artmakta ve aynı durum tekrar ortaya çıkmaktadır. Böylelikle süreç kısır döngüye girer ve sistemdeki kullanıcı sayısı azalana kadar IIS duraklayarak çalışmaya devam eder. Mevcut otomasyon incelendiğinde sistemin öğrenciye açılmasıyla birlikte ortalama 4 saat boyunca sistem kararlı bir şekilde hizmet verememektedir (Yıldırımoğlu, M., 2015).

OBS sisteminde, uygulama ve veri tabanı sunucuları birbirinden ayrılmıştır. Sistem ders seçme anında canlı şekilde izlendiğinde ve sunucu log'ları incelendiğinde hataların uygulama sunucusunda olduğu görülmektedir. Sistemde problem yaşandığı esnada lokal makinadan veri tabanı sunucusuna SQL sorgusu gönderildiğinde sunucu taleplere cevap vermektedir. Uygulama sunucusu göz önüne alındığında log'larında 'IIS

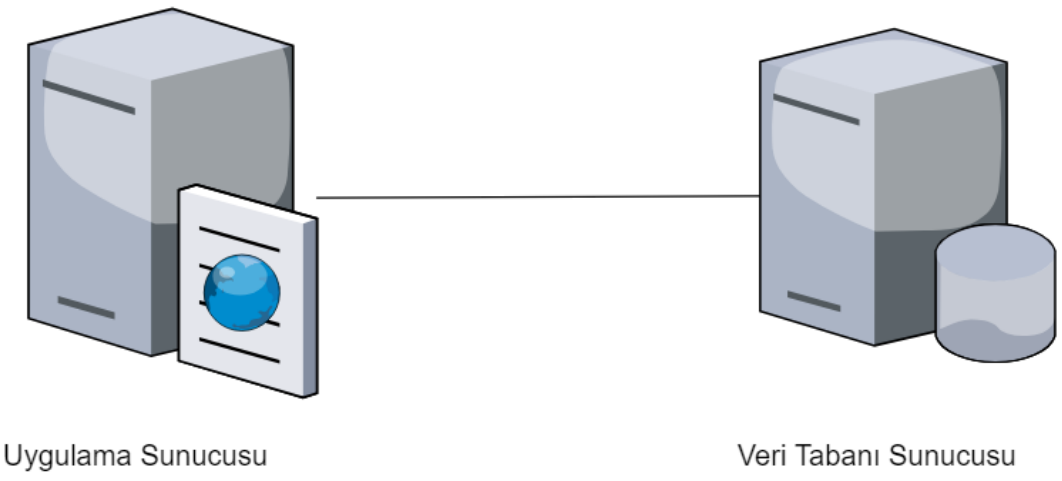
durduruldu' hatası görülmektedir. Bu inceleme sonucunda sorun yaşanan yerin uygulama sunucusu olduğu görülmüştür. Çözüm yöntemlerin de uygulama üzerine yoğunlaştırılmıştır. Veri tabanı ise mevcut haliyle yaşanan yoğunluğa dayanabildiği için o kısımda yoğun bir çalışma yapılmamıştır.

2.2 Mevcut Sistemin İncelenmesi

Bu çalışma B.Ş.E.Ü. öğrenci bilgi sistemi örneğinde olacağı için öncelikle mevcut otomasyon incelenip aksaklıklar belirlenip bu otomasyon üzerinden çözüm yöntemi geliştirilmiştir.

2.3 Sunucu yapısının incelenmesi

Mevcut otomasyonun çalıştığı sunucularda Windows işletim sistemi bulunmaktadır. Uygulama sunucusu ve veri tabanı sunucusu birbirinden ayrılmıştır. Uygulama sunucusunda otomasyon IIS üzerinde çalışmaktadır ve oturum bilgileri sunucunun state server'ında tutulmaktadır. Veri tabanı sunucusunda ise MSSQL kuruludur ve uygulama MSSQL bir veri tabanına sahiptir. Bu veri tabanına OBS haricinde başka uygulamalarda erişmektedir.



Şekil 2.1. Mevcut otomasyon sunucu yapısı (* Referans verilmeyen tüm çizimler draw io programı ile çizilmiştir).

Şekil 2.1'de görülen sunucuların ikisi de fiziksel olmayan sanal sunuculardan oluşmaktadır.

Uygulama Sunucusunun Özellikleri:

- Windows Server 2012 işletim sistemi
- intel(r) xeon(r) cpu e5-2680 v2 @ 2.80ghz 32 çekirdekli işlemci
- 32 GB Ram
- 270 GB HDD

Veri Tabanı Sunucusunun Özellikleri:

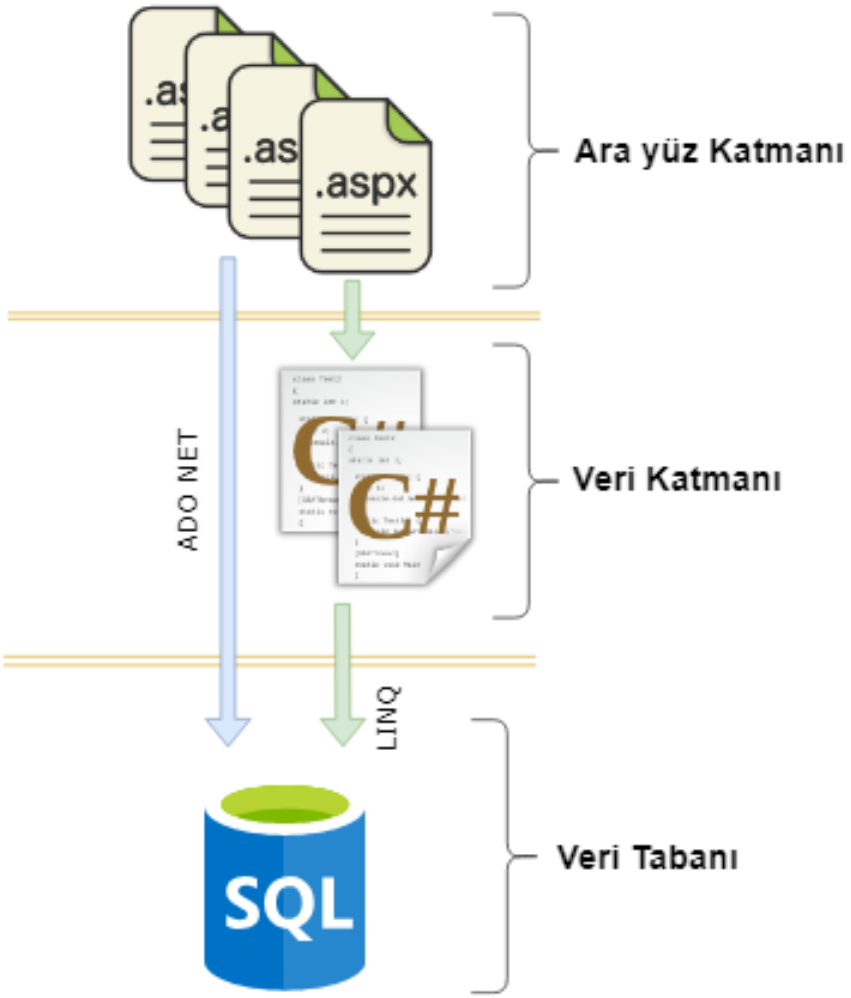
- Windows Server 2012 işletim sistemi
- intel(r) xeon(r) cpu e5-2680 v2 @ 2.80ghz 32 çekirdekli işlemci
- 64GB Ram
- 200 GB HDD

Veri tabanı sunucusu 32 çekirdek olmasına rağmen MSSQL lisansının 20 çekirdekli işlemciyi desteklemesi üzerine alındığı için, maksimum kullanabildiği çekirdek sayı 20'dir. Bu sunucu üzerinde veri tabanının yanında OBS raporlarının oluşturulduğu SQL Reporting Service'te çalışmaktadır.

Uygulama sunucusu üniversitenin kullandığı sanallaştırma ortamında verilebilen maksimum çekirdek sayısı olan 32 çekirdek ile çalışmaktadır. Bu sunucu üzerinde OBS uygulaması haricinde gece çalışan servisler, banka servisleri, bologna sayfaları gibi çok yoğun kullanılmayan farklı uygulamalarda çalıştırılmaktadır.

2.4 Yazılım teknolojisinin ve mimarisinin incelenmesi

Mevcut Otomasyon ASP.NET web form mimarisi ile geliştirilmiş bir uygulamadır. Uygulamanın backend tarafında C# dili kullanılmıştır. Frontend tarafında ise Html, Css ve Java Script dilleri kullanılmıştır.



Şekil 2.2. Mevcut sistemin mimarisi.

Şekil 2.2'de mevcut otomasyonun mimari yapısı çizilmiştir. Katmanlı mimari uygulanarak kurulan sistemde iki farklı katman üzerine yazılım inşa edilmiştir.

Veri katmanında Entity Framework teknolojisi kullanılarak veri tabanının modeli oluşturulmuştur. Bu model veri tabanındaki tablo ve ilişki yapısının birebir kopyası olan C# Class yapısından oluşan bir modeldir. Model oluşturulurken Code First yöntemi ile geliştirilmiştir. Proje genelinde Linq sorguları yazılsa da ders seçme modülünde Ado.Net üzerinden stored procedure ile sorgulamalarda yapılmaktadır.

Kullanıcı oturumları State Server üzerinde tutulmaktadır. Bu oturum açma yöntemi ile IIS durduğu durumlarda dahi kullanıcı oturumların ayakta kalmasını sağlamaktadır. Bu sistemin hata verdiği durumlarda son kullanıcıya güzel bir hizmet olsa da State Sever ekstra işlemci ve bellek kullanımına sebep olmaktadır. Yoğun

kullanımda yüksek işlemci performansına ihtiyaç olduğundan bu oturum tutma yöntemi problemimizi olumsuz yönde etkilemektedir.

Ara yüz katmanı ASP.NET mimarisiyle geliştirilen .aspx uzantılı web form sayfalarından oluşmaktadır. Bu sayfaların backend tarafında her bir sayfaya bağlı C# sınıf ve metotları bulunmaktadır. Sayfalarda ASP.NET nesneleri ve sayfaya entegre edilen Telerik kütüphanesinin nesneleri bulunmaktadır. Telerik kütüphanesi projede yoğun şekilde kullanılmaktadır. Proje ilk oluşturulurken bu kütüphane birçok kontrolü hazır olduğundan kolaylık sağlasa da proje özelleştikçe hem yazılımın geliştirme maliyetini arttırmakta hem de çalışma anında uygulama performansını olumsuz etkilemektedir.

Ders seçme modülü incelendiğinde öğrenci bilgileri ve açılan derslerin bilgileri stored procedure çalıştırılarak sorgulanmaktadır. Veriler getirildikten sonra yapılacak kontroller için ihtiyaç duyulan veriler tek tek linq sorgularıyla veri tabanından getirilmektedir. Kontrollerin birçoğu C# tarafında yapılırken bazı kontroller Java Script tarafında yapılmaktadır.

2.5 Sistemde Tespit Edilen Aksaklılar

Problemin genel anlamda uygulama tarafında yaşandığını tespit etmiştik. Mevcut sistemin incelenmesiyle birlikte uygulamadaki aksaklıklar belirlenmiştir.

Son kullanıcı sisteme girdiğinde öğrenci verilerinin hazırlanması sırasında sunucuda geçen süre yaklaşık ortalama 1450ms olarak ölçülmüştür. Son kullanıcı tarayıcısında bu verinin ekrana getirip işlem yapılabilir hale gelmesi ortalama 2sn sürmektedir. Bu ölçümler ders seçme haftasında değil günlük kullanımda yapılmıştır. Yoğun kullanım anında ise 4-5 saniyeye kadar uzayabilmektedir. Son kullanıcı bu süreyi beklemeyip tekrar tekrar sunucuya talep göndermektedir. Burada çözüm getirmemiz gereken en temel problem son kullanıcıya geri dönüşün çok yavaş olmasıdır. Son kullanıcıya hızlı bir şekilde cevap döndürülmediğinde yeni taleplerle IIS kuyruğunda yığılma oluşmaktadır. Kuyrukta oluşan bu yığılma IIS limitlerini aştığında IIS'in durmasına sebep olmaktadır. Yığılmayı önlemek için son kullanıcının bekleme süresinin düşürmesi gerekmektedir.

Son kullanıcı sisteme bağlandığında resim, css, js v.b. dosyaları istemci tarafında indirebilmek için toplamda 34 bağlantı açmaktadır. Açılan 34 bağlantı her bir kullanıcı için ayrı ayrı açılmaktadır. Sunucu yoğun kullanımda uygulamadaki istemci taleplerinin

yanında bu dosyaların indirilmesine de hizmet etmektedir. Bağlantı sayısı ile birlikte dosyaların boyutlarının küçültülmesi de bağlantı sürelerini düşürecektir.

Ders seçme modülü çok fazla kural ve kontrol içeren karmaşık bir çalışma yapısına sahiptir. Bu kuralların bir kısmı JS üzerinde yapılıyorken büyük bir çoğunluğu ise C# tarafında yapılmaktadır. C# tarafında yapılan özellikle intibak takibi ve notların açılan derslerle eşleştirilmesi sunucu tarafında zaman kaybına yol açmaktadır. Sunucu tarafındaki kontrollerin azaltılıp istemci tarafına aktarılmalıdır.

Öğrencinin notları getirilirken ve müfredatı hesaplanırken veri tabanındaki intibaklar ile ilgili recursive sorgular yazılmaktadır. Bu hesaplama her ders için yapılmaktadır. Bu hesaplamadaki recursive yapının önüne geçmek için veri tabanında düzenleme yapılmalı ve recursive olmadan müfredat ve notlara erişilmesi sağlanmalıdır.

ASP.NET Web Form ile geliştirilen sistemlerde sayfa mekanizmasının arkasında bulunan view state, performansı olumsuz etkilemektedir. Web Form'da sayfaya bir talep yapıldığında sayfadaki tüm veriler yenilenir. Sayfadaki tüm verilerin gereksiz yere tekrardan yüklenmesi performansı olumsuz etkilemektedir. Ayrıca ASP.NET mimarisi backend tarafında bir düzenleme getirmediği için karşılaşılan hatalarda yazılım onarım maliyetini arttırmaktadır. Bu sebeple hem yazılım onarım maliyetlerini düşüren hem de view state gibi hantal bir mekanizmaya sahip olmayan bir mimari tercih edilmelidir.

Son yıllarda telefon ve tablet kullanımının artmasıyla birlikte web sayfalarının farklı ekran boyutlarına ve dokunmatik ekran teknolojisine hale gelmesi önem kazanmıştır. Mevcut sistem mobil uyumluluğu yoktur. Yeni tasarlanan modülün mobil ekranlara uyumlu bir şekilde tasarlanmalıdır.

3 KULLANILAN TEKNOLOJİLER

Yaşanan problemin en temel sebebi sistemin yavaş çalışıyor olması ve bu yavaşlıktan dolayı sunucuya gelen talepleri, IIS işlem kuyruğunda bekletmesidir. İşlem kuyruğunda biriken talepler limitleri aşarak IIS'in durmasına yol açmaktadır. Bu çalışma kapsamında problemin önüne geçmek için uygulama farklı teknolojiler kullanılarak yeniden tasarlanmıştır. Köklü değişikliklerin yapıldığı ve büyük oranda fayda sağlayan yenilikler bu bölüm içerisinde ayrı ayrı başlıklar altında açıklanmıştır.

Projede Microsoft SQL server 2012 veri tabanı kullanılmaktadır. Veri tabanında stored procedure, view, trigger gibi yapılar çok fazla kullanılmamaktadır. Genellikle raporlama aracı içinde bu tür SQL yapılarından faydalanılmıştır. Raporlama aracı olarak SQL'in Reportin Service aracı kullanılmaktadır. Ancak bu araç genele hizmet eden kapsamlı bir raporlama servisi olduğu için performans olarak ders seçme modülüne uygun görülmemiştir. Ders seçme modülünde iki rapor bulunduğundan ve performans olarak daha faydalı olması için raporlar direkt olarak view üzerinde oluşturulmuştur. Raporlama haricinde SQL üzerinde köklü değişiklik yapılmamıştır sadece öğrencinin not ve ders bilgisinin getirildiği tabloların yapısında bir değişiklik yapılmıştır. Bu yapılan güncellemenin detayı 4. Bölümde yer alan veri tabanı tasarımı başlığında anlatılmıştır.

Web uygulama mimarisi olarak, Web Form'dan MVC mimarisine geçiş yapılmıştır. Mimari tarafında değişiklik yapılsa da arka tarafında çalışan C# dili kullanılmaya devam etmektedir. C#'ta performans kazancı sağlamak ve RAM'de kullanımını düşürmek amacıyla Using blokları kullanılmıştır. Using bloğu, içerisinde oluşturulan değişken ve nesneleri blok tamamlandığında bellekten siler (Algan, 2015). Özellikle entity nesneleri gibi bellekte yer kaplayan tanımlamaların yapıldığı yerlerde Using blokları kullanılarak sunucunun daha performanslı çalışması sağlanmıştır. Ayrıca C# kodları sunucu tarafında çalıştığı için mümkün olduğunca buradaki kod sayısı azaltılıp, Java Script üzerine atılarak yük istemci tarafına verilmiştir.

3.1 Yük Dengeleyici (Load Balance)

Yük dengeleyici sunucu mimarilerinde artan trafiği karşılayabilmek amacıyla bellek, işlemci gücü ve disk alanlarını arttırılmasını, kaynakları bireysel güçlendirmek yerine paralel çalışan sunucular kullanarak, sistemin dağıtık çalışmasını sağlayan yöntemdir. Load balance kısaca sunucu üzerindeki yükü dağıtma olarak tanımlanabilir.

Bu yükün dağıtılması donanımsal anlamda fayda sağladığı gibi uygulamanın çalıştığı IIS içinde fayda sağlamaktadır. IIS server yükü de uygulamaların bölünmesiyle birlikte yük dağıtımı yapılmış olacaktır. Yaşanan problemi incelediğimizde IIS üzerindeki kuyruğun uzaması ve IIS'in kendini durdurmasından bahsedilmişti Yük dengeleme ile birlikte IIS'te biriken talep kuyruğu da farklı sunucular üzerine dağıtılabilecektir. Yük dengeleme işlemi IIS ve DNS üzerinden yapıldığı gibi bu işlemi yapan özel ağ cihazları da bulunmaktadır. Bu proje kapsamında IIS üzerinden yük dengeleme işlemi yapılmıştır.

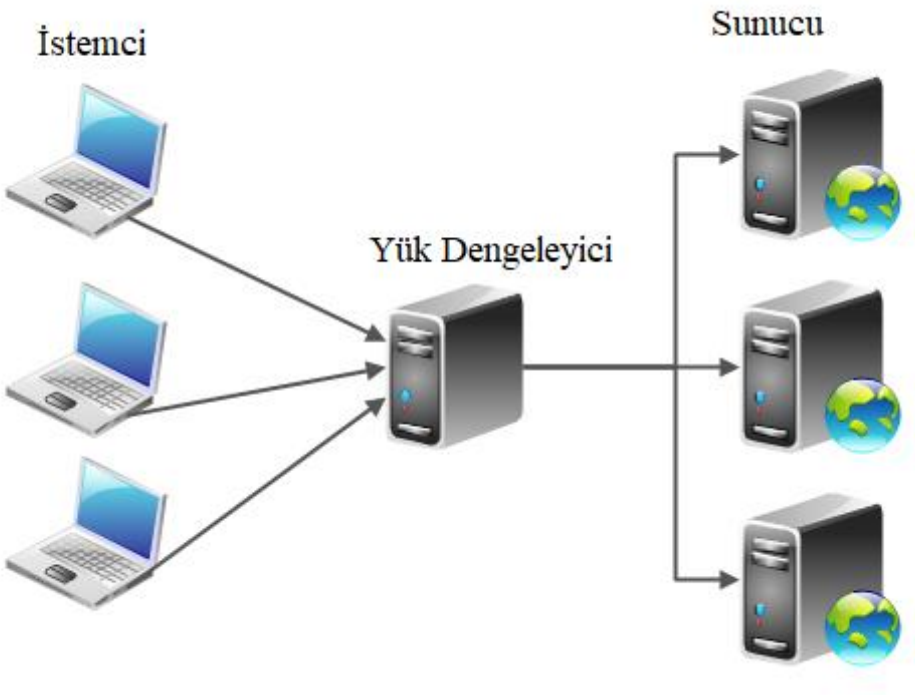
Load Balancer sunucuların paralel çalışmasından ziyade gelen trafiği dağıtmaktadır bunun sonucu olarak sunucular paralel çalışır duruma gelmektedir. Balancer dört farklı trafiği yönlendirebilmektedir.

HTTP: Standart HTTP dengelemesinde gelen istekleri HTTP tekniklerine dayanarak yönlendirilir.

HTTPS: HTTPS dengeleme işlemlerinde HTTP dengeleme ile aynı süreç işletilmektedir. Aradaki tek fark HTTPS işlemlerindeki SSL şifreleme süreçleridir. SSL ile şifrelenen istek backend'e kadar korunur. Bu çalışma kapsamında HTTPS isteklerini dağıtan bir Load Balancer yapısı kullanılacaktır

TCP: HTTP veya HTTPS kullanılmayan uygulamalar için TCP trafiğini dengeler. Örneğin, bir veri tabanı kümesine giden trafik tüm sunuculara yayılarak dengelenebilir.

UDP: Bu dengeleme load balancer üzerindeki protokol ve bağlantı noktası tanımlandıktan sonra load balancer'ın backend'deki trafiği yönlendirmek için kullanacağı protokol ve bağlantı ile eşleşmesi sağlanır.



Şekil 3.1. Yük dengeleme yapısı (http://tutorials.jenkov.com/images/software-architecture/load-balancing-1.png).

Son kullanıcı tarayıcıdan web uygulamasının adresini yazıp enter tuşuna bastığı anda DNS yönlendirmesi talebi ilk olarak Balancer sunucusuna yönlendirecektir. Balancer gelen talebi aldıktan sonra arkasında çalışan worker sunucularından ayakta olanları tespit edip önceden belirlenen algoritmaya göre worker sunucularından birine iletir ve bu noktada Balancer görevini tamamlamış olur. Son kullanıcı uygulamada işlem yapmaya devam ettiği sürece Balancer'a tekrar uğramaz ve arkada worker üzerinde çalışmaya devam eder.

Balancer yükü dağıtırken gelen talebi arkada hangi worker'a ileteceğine karar verirken farklı farklı öncelik belirleme algoritmaları kullanmaktadır. Balancer karar vermeden önce hangi sunucuların aktif çalışıyor olduğuna bakıp, ayakta olan sunucular arasında önceliklendirme yapar. En yaygın kullanılanları Round Robin ve Least Connection algoritmalarıdır.

Round Robin: Sunucuların öncelik sıralamasını balancer sunucu listesindeki sırayla dağıtır. Her bir sunucuya bir yönlendirme yaptıktan sonra başa dönüp tekrar yönlendirme işlemi yapmaya devam eder. Bu algoritmada bütün worker'lara eşit miktarda yük dağılımı gerçekleştirilir.

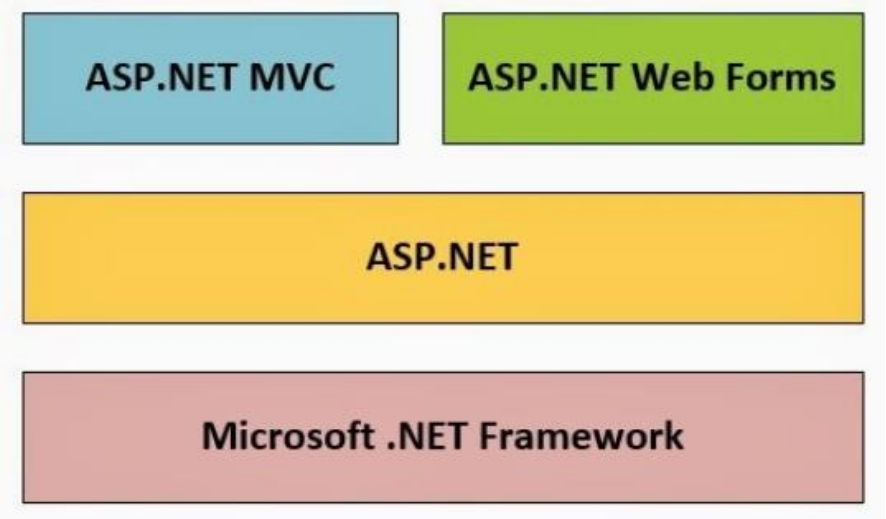
Least Connection: Least connection algoritmasında Balancer en az bağlantıya sahip sunucuyu seçer. Bu algoritma trafiğin daha uzun oturumlarla sonuçlandığı durumlarda önerilir.

Bu proje kapsamında birinci yöntem olan Round Robin kullanılmıştır. Least connection daha sağlıklı bir yapı gibi görünse de eğer arkadaki worker sunucuları eşit paylaştırılmış yükü taşıyabiliyorlarsa least connectiondaki kontrol sürelerinden tasarruf edilmiş olacaktır.

Yük dengelemenin en önemli faydalarından biride, yoğun kullanımda bir sunucuda problem yaşanırsa diğerler sunuculardaki kullanıcıların bundan etkilenmemesidir. Örneğin; 4 worker sunucuya sahip bir Load Balancer yapısında 1. worker durduğunda bundan sadece kullanıcıların %25'i etkilenecektir. Geriye kalan %75 kullanıcı çalışmasına devam edebileceklerdir.

3.2 ASP.NET MVC

ASP.NET, Microfost'un web tarafında yazılım geliştirmek için oluşturduğu platformdur. ASP.NET ilk çıktığı yıllarda backend tarafında Visual Basic ve C#'a destek verirken günümüzde sadece C#'a destek vermektedir. ASP.NET çatısı altında web form uygulamaları veya MVC uygulamaları geliştirilebilir. ASP.NET, web form ve MVC mimarilerini kapsayan bir platform olmasına rağmen ASP.NET ile web form ile sıklıkla karıştırılır (Çamoğlu, 2010).

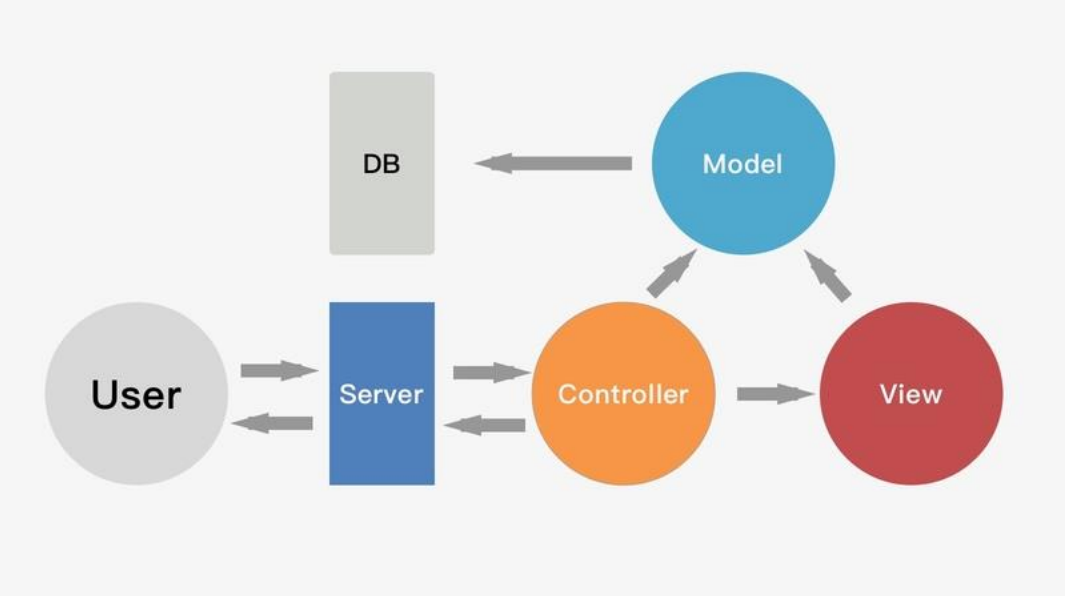


Şekil 3.2. ASP.NET bileşenleri.

MVC yazılım geliştirmede kullanılan bir mimari desendir. MVC deseni Trygve

Reenskaug tarafından 1979 yılında oluşturulmuştur (Kızmaz, 2014). Bu desen Php ve Java gibi birçok web yazılım dilleriyle birlikte kullanılmıştır. Özellikle büyük çaplı ve karmaşık kod yapısına sahip projelerde bu mimari projenin kod yasının anlaşılırlığını sağlamak adına oldukça başarılı olmuştur. MVC ile geliştirilen projelerin başarısı ve web form mimarisindeki eksikliklerin görülmesiyle birlikte, MVC'nin oluşturulmasından Microsoft 39 yıl sonra 2008 yılında MVC deseni ASP.NET platformuna entegre etmiştir.

MVC, Model, View, Controller kelimelerinin baş harflerinden oluşur ve her bir kelime MVC'nin farklı bir katmanını ifade etmektedir. Katmanlara ayrılmış bu yapı hem kod geliştirmeyi düzenli hale getirir hem de kodun okunmasını ve anlaşılırlığını kolaylaştırmaktadır.



Şekil 3.3. APS.NET MVC mimarisi (http://www.webdevelopmenthelp.net/2013/10/difference-between-asp-net-webform-and-asp-net-mvc.html).

Model: Uygulama verisinin veya durumunun saklandığı yerdir, genellikle veri tabanı veya xml/json dosyası formatındadır. Model, veri katmanını database, xml, json dosyası, vb. uygulamadan izole eder, böylece diğer katmanlarda veri katmanının detayının bilinmesine gerek kalmaz. Model katmanı sıklıkla Entity Framework, Nhibernate vb. araçlar kullanılarak oluşturulur.

View (Görünüm): İstemcinin gördüğü arayüzü içeren katmandır, genellikle Model katmanındaki verinin kullanılması ile oluşturulur. View katmanının Model ve

Controller katmanlarından ayrılması ile ara yüz değişikliklerinin uygulamanın diğer katmanlarını değiştirmeye gerek kalmadan yapılabilmesi sağlanmıştır.

View katmanında HTML5 ve CSS3 gibi tasarım teknolojilerinin en güncel versiyonlarını kullanmak mümkündür. HTML5 ve CSS3 ile masaüstü ve mobil tarayıcılarda çalışabilen uygulamalar geliştirmek çok kolaylaşmıştır

Controller (Denetleyici): İstemciden gelen isteği işlemek, model ve view katmanları arasında köprü olmak gibi görevleri yerine getirir. Controller içerisinde bir veya daha fazla action olabilir, genellikle her action sonucunda bir web sayfası üretmek için kullanılır.

ASP.NET MVC'nin Avantajları :

- Yazılımcının çıktı üzerinde tam kontrolü vardır. Bu sayede HTML'nin her alanına isteğe en uygun çıktılar ortaya konulabilir.
- Durum yönetimleri yazılım tarafından yapıldığı için web form'a göre daha çok performans sağlar.
- Yazılan kodların bağımsızlığı sayesinde her yerden erişim söz konusudur. Bu sayede tekrar kullanılabilirlik özelliği ile bir projede yazılan kodun ya da modelin diğer projelerde de rahatlıkla kullanılabilmesi sağlanır.
- Yazılan kodlar sayfaya özgü olmadığı için projenin test edilebilirliği yüksektir. Bu sayede Test Driven Development projeleri oluşturulabilir.
- JavaScript tabanlı istemci taraflı teknolojilerin kullanımını destekler.
- Arama motorları için de büyük önem taşıyan adres ve içerik uyumunu tam anlamıyla sağlar.
- İstemciye gelen HTML tamamen geliştiricinin kontrolündedir. Geri dönen HTML'in boyutu Web Form'a göre çok daha küçüktür. Bu sayede HMTL'in render edilmesi ve son kullanıcıya gösterilmesi Web Form'a göre daha performanslı çalışmaktadır.
- Her bir katmanın neyi tanımladığı ve hangi görevi üstlendiği çok nettir. Bu nedenle takım anlayışına çok uygundur. Böyle bir yapısı olduğu için de daha kolay genişletilebilir

ASP.NET MVC'nin Dezavantajları:

- Asp.NET Web Forms teknolojisine göre yenidir. Yeni olması daha az tecrübe edildiğini gösterir, bu sebeple dezavantaj denilebilir.

- Yazılımcıların, istemci taraflı ve sunucu taraflı entegrasyonunu iyi anlamaları ve sunucu taraflı teknolojileri daha iyi kullanabiliyor olmaları gerekir.
- Sürükle bırak mantığına alışmış web yazılımcıları için alışılması daha zor ve uzun olabilir.

3.2.1 ASP.NET MVC yaşam döngüsü

Web Form'da yaşam döngüsü sayfa bazlı ilerleyip tüm işlemler sayfaya yapılır ve cevap olarak kullanıcıya sayfa döner. MVC'de ise yaşam döngüsünde odak tamamen talep ve bu talebin yapıldığı controller'ın geri döndürdüğü view üzerinden gerçekleşir. Bu yaşam döngüsü ilk istek talep edilmesiyle birlikte son kullanıcı geri dönüşün oluşmasına kadar adım adım anlatılmıştır (Kızmaz, 2014).

1. Adım : HTTP Request**:** Son kullanıcı ASP.NET MVC uygulamasını görüntülemek için tarayıcıdan adresi yazıp enter tuşuna basması bir request(istek)'tir. İstek HTTP üzerinden IIS tarafından alınır ve her yapılan istek server tarafından bir yanıtla son bulur.

2. Adım : Routing: ASP.NET MVC uygulamaya her istek yapıldığında, istek url routing module tarafından durdurulur. url routing module bir isteği durdurduğunda, gelen istek Route Table'dan hangi Controller tarafından üstleneceğine karar verilir.

3. Adım : Controller: RouteTable'dan gelen yönlendirme bilgisine göre controller'daki hangi action'ı çalıştırıp geriye bir view döndürür. Bu aşamada döndürülen view render edilmez.

4. Adım : ViewResult: View'i render etmek için aktif view engine'i çağırır.

5. Adım : ViewEngine : Bir CSHTML dosyayı oluşturduğunuzda View Result içerisindeki script ve markuplar, razor view engin tarafından ASP.NET API'lerini HTML sayfaya çevirmek için kullanır.

6. Adım : View: View Engine tarafından HTML sayfaya çevrilen kodlar son kullanıcı tarayıcısına sunulur.

7. Adım : Response: Son olarak HTTP üzerinden view kullanıcıya gösterilir.

3.2.2 Razor

ASP.NET MVC Razor, Microsoft'un geliştirdiği View Engine olarak adlandırılan görüntüleme motorudur. Microsoft ilk olarak MVC 3 ile Razor'u tanıtmıştır. HTML içersinde Razor View Engine ile server tarafında çalışacak olan kodların, "@" işareti

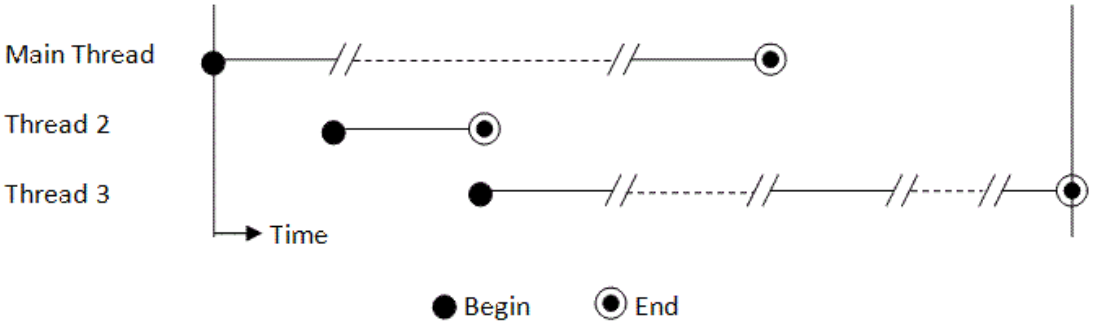
ile kullanarak oluşan söz dizimi diyebiliriz. Bu bileşen sayesinde "@" işaretini koyarak HTML içinde C# kodu yazılabilmektedir. Web Form da HTML içerisinde C# kodu yazılması mümkün değildir (Aktaş ve Sevinç, 2011).

Razor'un Avantajları:

- Razor kullanılan kod sayısını en aza indirir ve kodun akışını akıcı yapar, hızlandırır. Bir çok template söz diziliminin aksine, HTML'iniz içerisinde server bloğunu ayrıca belirtmeye gerek yoktur.
- Minimum kod gereksinime sahip olduğu için öğrenilmesi son derece kolaydır.
- Razor backend tarafında yazılan kodların benzerini frontend tarafında yazılabilmesine olanak sağlar.
- Razor geliştirmek için özel bir tool'a ihtiyaç duymaz. Notepad gibi text editörleri ile düzenleme yapılabilir. Derlenmeye ihtiyaç duymaz.
- Visual Studio 2010 ve sonrası sürümlerinde intellisense özelliği view tarafında da çalışmaktadır. Bu sayede kod yazımını kolaylaştırır.
- Razor View Engine ile Visual Studio'nun test ünitlerinin çalıştırılabilmesini sağlar.

3.3 Çok Kanallı Programlama (Multi Thread)

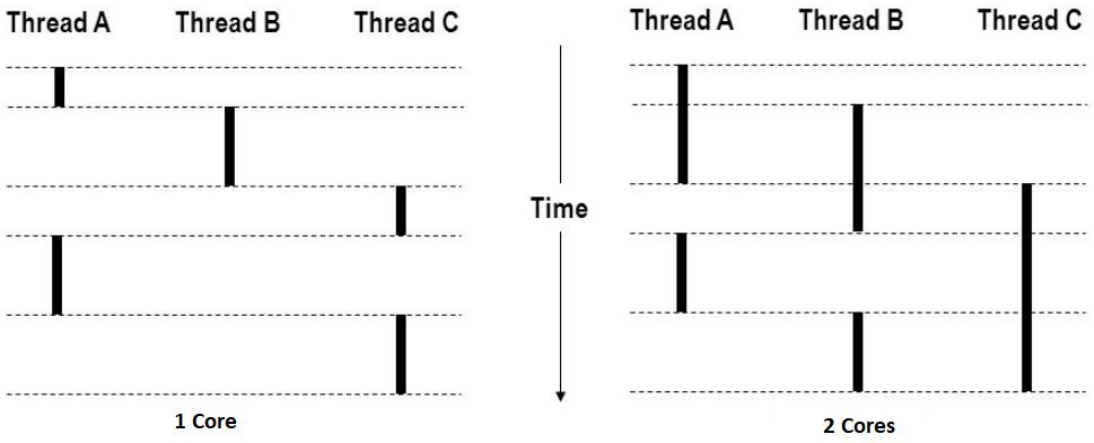
Bilgisayarda yapılan tüm işlemler geçici bellek olan RAM üzerinden çalışmaktadır. Program RAM'e yüklendiğinde işlem adını almaktadır. CPU bu işlemleri çalıştırarak bilgisayardan beklenen görevleri yerine getirmektedir. CPU'lar bu işlemleri yaparken özellikle başka bir sunucuya bağlanıp sorgu çalıştırıyorlarsa çok fazla IDLE yani bekleme süreleri oluşmaktadır. Çok kanallı programlama bekleme sürelerinde bize farklı bir kodun çalışmasına imkan sağlar. Bu sayede paralel olarak farklı kodları işlemci üzerinde koşturmuş oluruz (Aktaş ve Sevinç, 2013).



Şekil 3.4. Multi thread çalışma süreleri (http://www.ntu.edu.sg/home/ehchua/programming/java/j5e_multithreading.html).

Şekil 3.4'te 3 tane işlem multi thread olarak çalıştırılmıştır. İşlemci üzerindeki çalışma süreleri başlangıç ve bitiş aralıkları gösterilmektedir. Birinci main thread beklemeye girdiği anda ikinci tread çalışmaya başlar ve onun bitiminde ise üçüncü tread çalışmaya başlamaktadır. Üçüncü tread beklemeye geldiği anda main thread'e geri dönüp oradaki işlem devam ettirilmektedir. Akış incelendiğinde multi thread kullanılmasaydı toplam işlem süresi yaklaşık iki katına çıkacaktı. Multi thread sayesinde CPU hiç beklemeden sürekli işlem yapmış oldu.

Yukarıdaki örnekte tekil işlem yapan bir CPU üzerinde akış incelenmiştir. Birden fazla çekirdekli CPU'larda akışta çekirdek sayısına bağlı olarak işlemler paralel bir şekilde yürütülebilmektedir.



Şekil 3.5. Çift çekirdekli bir işlemcide multi thread çalışma süreleri (http://www.ntu.edu.sg/home/ehchua/programming/java/j5e_multithreading.html).

Şekil 3.5'in sol tarafındaki tek çekirdekli CPU'nun thread'leri incelendiğinde işlemlerin her biri tekil olarak çalışmaktadır. Aynı anda çalışan thread yoktur ve sadece CPU'nun IDLE süreleri değerlendirilmiştir. Şeklin sağ tarafındaki çift çekirdekli CPU'nun thread'leri incelendiğinde ise aynı anda çalışan iki tread olduğunuz görebiliyoruz. Aynı üç thread, dört çekirdekli bir CPU'da çalıştırılsaydı A, B, C thread'lerinin üçü de aynı anda çalışmaya başlayıp paralel bir şekilde yürütüleceklerdi. Bu projede 32 çekirdekli CPU'su olan sunucular kullanılacaktır. Yük dengeleme ile 4 sunucu paralel olarak uygulamayı çalıştıracaktır. Uygulama için aynı anda toplam 128 çekirdek çalıştırılacaktır. Bu oldukça yüksek bir işlem gücü sağlamaktadır.

CPU'lar tekil çekirdekten çoklu çekirdeklere geçmesiyle birlikte multi thread programlamaya olan ilgi arttı. Microsoft bu gelişmeler karşısında C# 5.0 sürümünde multi thread kod geliştirmeyi kolaylaştırmak için iki yeni komut çıkartmıştır. Task ve Task Factory C# 5.0 ile birlikte gelen komutlardır. Bu komutlar ile iki işlemi paralel yürütmek oldukça basittir. Microsoft'un geliştirdiği bu komutlar ile oluşturulmuş bir multi thread örneği aşağıdaki kod bloğunda verilmiştir (Schild ,2002).

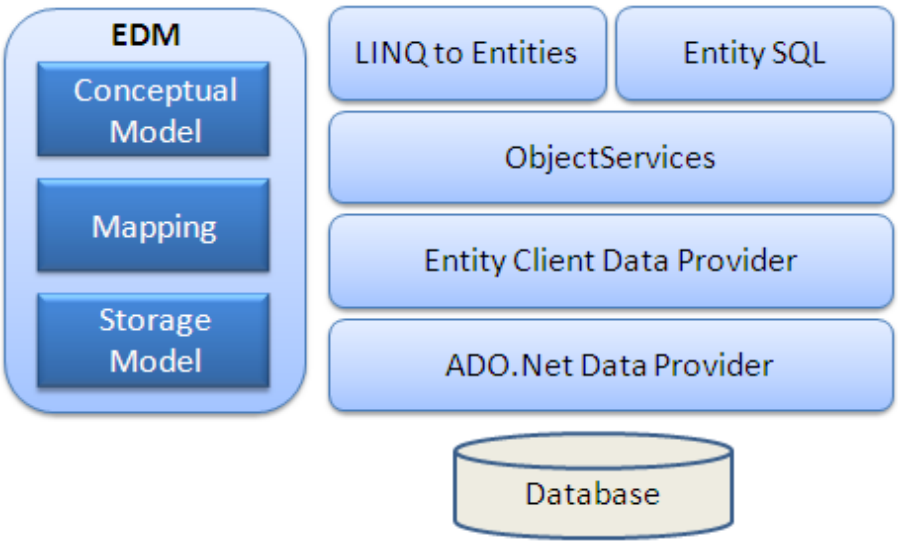
```
Task Gorev = Task.Factory.StartNew(() =>
{
                Task Islem1 = Task.Factory.StartNew(() =>
                {
 // Thread 1
 }, TaskCreationOptions.AttachedToParent);
                Task Islem2 = Task.Factory.StartNew(() =>
                {
                            // Thread 2
                }, TaskCreationOptions.AttachedToParent);
       });
 Gorev.Wait();
```

Yukarıdaki kodlar incelendiğinde yorum satırı olarak verilen Thread 1 ve Thread 2'nin olduğu bloklara istenilen kodlar yazılabilir. Bu blokların hepsi ayrı threadlar olarak işlemci üzerinde çalıştırılmaktadır. Örnekte iki thread verilmiştir ancak bunların sayısını istenilen kadar arttırılabilir. En son satırdaki Wait komutu tüm thread'lerin bitmesini beklemek için kullanılmıştır. Bu komut kullanılmadığında tüm thread'ler

asenkron olarak çalışmaya devam eder ve program alt satıra geçip ana işlemine devam eder.

3.4 Entity Framework

Entity Framework Microsoft tarafından geliştirilen .Net tabanlı bir ORM " Object Relational Mapping" aracıdır. İlk olarak 2008 yılında .NET 3.5 sürümüyle Visual Studio'ya dahil olmuştur (Şavklı, 2014). Entity Framework projenin uygulama kısmı ile veri tabanı arasındaki iletişimi sağlamaktadır. Veri tabanının bire bir kopyası C#'ta nesneler yardımı ile oluşturulur. Veri tabanında bulunan tablolar için nesneler oluşturulur veri tabanında tabloların sütunları için ise nesnelere ait özellikler oluşturulur. Oluşturulan bu nesnelere, entity ve linq sorguları ile nesneler üzerinden veri tabanı sorgulanabilir. Bu sayede veri tabanı işlemlerinin SQL sorgu cümleleri yazmadan entity model üzerinden yapılmasını sağlar. Veri tabanına yapılan CRUD (Create, Read, Update, Delete) linq sorgularını Entity Framework arka tarafta Sql sorgusuna dönüştürür. Bu işleme "Code Generating" denir. Entity mimarisi temel olarak conceptual model, mapping, storage model olmak üzere üç bölümden oluşmaktadır (Yakar, 2011).



Şekil 3.6. Entity framework mimarisi.

Conceptual Model: Bu alanda model sınıflarımız ve bu sınıfların ilişkileri yer almaktadır. Bu sınıflar veri tabanı tasarımından bağımsız olacaktır.

Storage Model: Bu alanda veri tabanı tasarım modeli yer almaktadır. Bu model

içerisinde veri tabanına ait tablolar, view'ler, stored procedure'ler ve bunlara ait ilişkiler ve key'ler yer alır.

Mapping: Model sınıfları ile tasarım modelinin arasındaki haritalama işlemlerinin bilgilerinin tutulduğu alandır.

Visual studio Entity Framework'ün iki farklı şekilde oluşturulmasına olanak sağlar. Birincisi "Database First" olarak isimlendirilen önce veri tabanı tasarımının yapılıp entity katmanın tamamı visual studio aracıyla oluşturulur. Bu yöntem kolay ve hızlı olanıdır. Bütün veri tabanın modeli birkaç dakikalık bir işlemle elde edilebilir. İkinci yöntem ise "Code First" yaklaşımıdır. Bu yöntemde ise entity modeli tamamen yazılımcı tarafından oluşturulur ve modelde yapılan değişiklikler veri tabanına yansır. Örneğin kullanıcı tablosuna doğum tarihi adında yeni bir alan eklendiğinde bu modeli veri tabanına gönderildiğinde yani migration işlemi yapıldığında veri tabanındaki kullanıcı tablosuna aynı alan eklenmektedir. Bu yaklaşımda değişiklikler ilk olarak kod tarafında yapılmaktadır. OBS sistemi içerisinde de "Code First" yaklaşımı kullanılmaktadır. OBS'de model çok büyük olduğundan "Database First" yöntemi derlenirken yavaş çalışmaktadır. Bu sebeple Code First tercih edilmiştir.

Entity Framework avantajları:

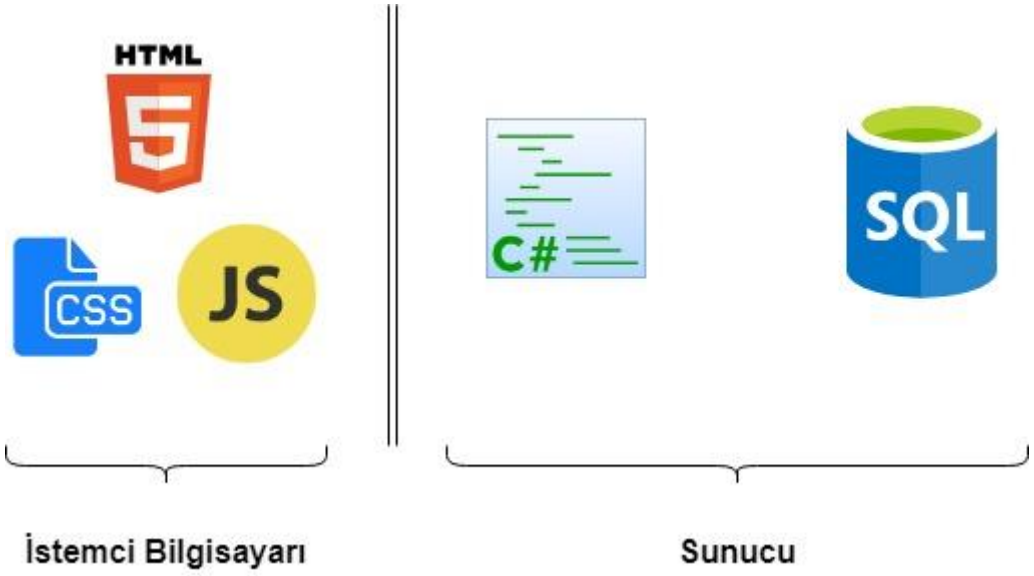
- Projeyi nesneye yönelik programlamaya uyumlu hale getirir.
- Nesneye dayalı olduğu için karmaşık sorguların yazımı SQL sorgularına göre daha kolaydır.
- Kod veri tabanını türünden bağımsız çalışır, Mssql'den Oracle'a geçildiğinde sadece model yeniden oluşturularak sorgularda değişiklik yapılmadan geçiş sağlanır.
- Yazılımcının SQL bilmesine gerek kalmadan sorgulama yapabilmesine olanak sağlar.
- Kolay test edilebilir.
- Hata tespiti yapılması kolaydır.

Entity Framework dezavantajları:

- Code Generating yapısı sorguları çevirdiği için sorgu cümlesine müdahale imkanı yoktur.
- Veri tabanı bağımsızlığı avantajdır fakat uygulama tarafındaki nesneler ile veri tabanındaki nesneler birbirine MAP edildiği için nesne bağımlılığı vardır.

3.5 JavaScript

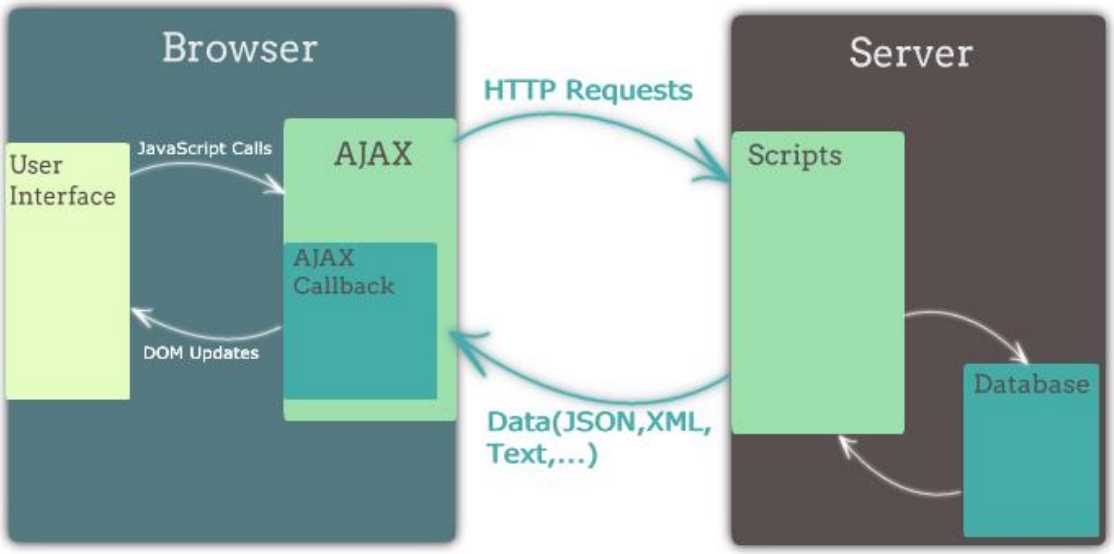
JavaScript ilk olarak Brandan Eich tarafından 1995 yılının Eylül ayında oluşturulmuş web tarayıcılarında kullanılan bir betik dilidir. Günümüzde Mozilla vakfı tarafından geliştirilmeye devam edilmektedir (Çelikbilek, 2013). Özellikle web sayfalarında HTML ve CSS ile birlikte kullanılmaktadır. Ara yüze çalışma anında müdahale ve ara yüzden sunucuya talepte bulunmak gibi birçok işlemi kolayca yapabilmemizi sağlar.



Şekil 3.7. İstemci ve sunucu taraflı çalışan kodlar.

Şekil 3.7'de görüldüğü gibi web uygulamalarında sunucu ve istemci tarafında olmak üzere kodlar iki farklı yerde çalışmaktadır. ASP.NET MVC için düşünüldüğünde C# kodları sunucuda çalışırken HTML, CSS ve JS kodları tarayıcı üzerinde yani istemci bilgisayarında çalışmaktadır. Node.Js bu çalışma yönteminin bir istisnasıdır ve Node.Js haricindeki tüm JavaScript kodları istemci tarayıcısı üzerinde çalışır. JavaScript kodlarının istemci tarafında çalışması uygulamanın dağıtık bir mimaride çalıştırılabilmesine imkan sağlamaktadır. Veri tabanından veriler çekildikten sonra gerekli hesaplamalar ve veriyi kullanıcının görmesine uygun hale getirme işlemleri JavaScript'te yapıldığında ciddi bir performans kazancı elde edilmiş olmaktadır. Bu proje kapsamında özellikle kural ve kontroller eski projede C#'ta iken JavaScript'e alınarak performans kazancı elde edilmiştir (Pekgöz, 2001).

Günümüzde JavaScript'in yaygınlaşmasıyla birlikte JS tabanlı kütüphanelerin sayısıda artmıştır.Bu kütüphanler farklı amaçlara hizmet etmektedir. John Resig 2006 yılında JavaScript kütüphanelerini daha sade ve anlaşılır bir hâle getirmek için jQuery'yi geliştirmiştir (Baltalı vd., 2011; Akdemir ,vd., 2012). Jquery özellikle ara yüzde bulunan HTML etiketlerini seçmek, içerisine veri yazmak, veri okumak ya da tasarımsal olarak bir işlem yapılabilmesini oldukça basit hale getirmektedir. Bu projede Jquery sıklıkla kullanılmıştır.



Şekil 3.8. AJAX çalışma yapısı (http://javascript-coder.com/tutorials/re-introduction-to-ajax.phtml).

Ajax, "Asynchronous JavaScript and XML" bir çok programlama dili ile entegre çalışabilen bir framework'tür (Çelik, 2013). Ajax web sayfalarında bir post işlemi yapıldığında tüm sayfanın yeniden yüklenmesinin önüne geçmek için kullanılır. Dinamik olarak sayfanın etkileşimini yapabilmek için backend ve frontend arasında köprü oluşturur. MVC büyük oranda sayfanın tekrar yüklenme probleminin önüne geçse de view'ler içerisinde ajax post işlemlerine ihtiyaç duyulmaktadır. Bu proje kapsamında istemci sunucuya bir talep yaptığında sunucudaki geçen süreyi minimuma indirmek hedeflendiğinden ajax kullanılmıştır. Ajax ile verileri istemcinin tarayıcısına alıp orada veriler işlenip ders seçmeye uygun hale getirilmiştir. Bu sayede sunucu yükü istemci bilgisayarına alınmıştır.

JavaScript Avatajları:

- Derleyiciye ihtiyacı yoktur, tarayıcı tarafından HTML ile birlikte yorumlanır.
- Hata tespiti yapmak kolaydır.
- Kullanıcı hareketi izlenebilir ve fare hareketlerine, klavye girdilerine yönelik yazılım yapılabilir.
- JavaScript birden fazla platformda, tarayıcıda çalışabilir.
- Web uygulamalarının interaktif tasarlanmasını kolaylaştır.
- Backend tarafında çalışan dillere göre daha hızlıdır.
- CDN gibi platformlarla istemciye farklı bir sunucudan hizmet verilebilir.

JavaScript Dezavantajları:

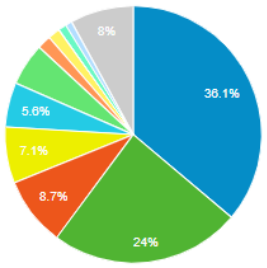
- İstemci tarafında çalışması, uygulamanın açığının olma olasılığını arttırır.
- Her tarayıcı tarafından desteklenmeyebilir, özellikle eski sürüm tarayıcılarda problem yaşanabilir.
- Javascript ile birlikte kullanılan üçüncü parti framework'ler hataya sebep olabilir veya kendine veri toplayabilir. Bu sebeple dikkatli kullanılması gerekir.

Yukarıda belirtilen dezavantajlardan en önemlisi uygulama içerisinde açıklar oluşturabilmesidir. Örneğin bir öğrenci JavaScript dosyası tarayıcıya yüklendikten sonra orada bir değişiklik yapabilir. Kuralları uygulayan bir kod satırını değiştirerek kendisine fayda sağlayabilir. Bu oldukça riskli bir durumdur. Bu tür durumların önüne geçmek için OBS içerisinde toplu olarak bu kuralları kontrol eden raporlar eklenmiştir. Bu şekilde sistemi hack'lemeye çalışan öğrenciler kolaylıkla tespit edilebilmektedir.

3.6 Bootstrap

Bootstrap açık kaynak kodlu bir CSS framework'üdür. Günümüzde tablet ve telefonun yoğun kullanılmasıyla birlikte farklı boyutlardaki ekran kullanım oranları artmıştır. Özellikle öğrencilerin yoğun kullandığı sistemlerde telefon kullanım oranının yüksek olması ekran çeşitliliğini çok fazla arttırmıştır. Bu çalışmada geliştirilen proje devreye alındıktan sonraki ekran kullanım oranları şekil 3.9'da görülmektedir. Şekilde en yoğun kullanılan 10 ekran çözünürlüğü verilirken toplamda 107 farklı boyuttaki ekrandan sisteme erişim sağlanmıştır.

1.	360x640	**4,668**	36.13%
2.	375x667	**3,105**	24.03%
3.	412x732	**1,125**	8.71%
4.	320x568	**913**	7.07%
5.	414x736	**725**	5.61%
6.	360x720	**689**	5.33%
7.	412x846	**226**	1.75%
8.	360x760	**195**	1.51%
9.	424x753	**132**	1.02%
10.	360x740	**109**	0.84%



Şekil 3.9. Uygulamanın açıldığı ekran çözünürlük kullanım oranları grafiği.

Şekil 3.9'da görülen ekran farklılıkları tasarım açısından oldukça problemli bir durumdur. Farklı ekranlarda tasarımın bozulmaması kullanışlılık açısından önemlidir. Bootstrap kütüphanesi bu problemin aşılması için geliştiricilere kolaylık sağlamaktadır. Bootstrap yapılan tasarımı otomatik olarak mobil uyumlu hale getirerek farklı ekranlarda bozulmadan görüntülenmesini sağlamaktadır.

Tasarım yaparken karşılaşılan problemlerden bir diğeri de tarayıcılar arasındaki farklılıklardır. Özellikle Css dosyaları içerisindeki bazı komutlar farklı tarayıcılarda ekrana aynı şekilde yansıtılmamaktadır. Yazılan Css dosyası içinde bu tür farklılıklar tespit edilip tarayıcıya haiz kodlama yapılması gerekmektedir. Bu işlemde her bir tarayıcı için özel kod yazmak anlamına gelmektedir. Ancak bootstrap bu farklılara çözüm olmaktadır. Chrome, Safari, Firefox, Internet Explorer, Opera v.b. yaygın olarak kullanılan tüm tarayıcılara özel komutlar bootstrap içinde hazır yazılmıştır (Balkan, 2012). Böylelikle projenin tarayıcı bağımlılığı da ortadan kaldırılmıştır.

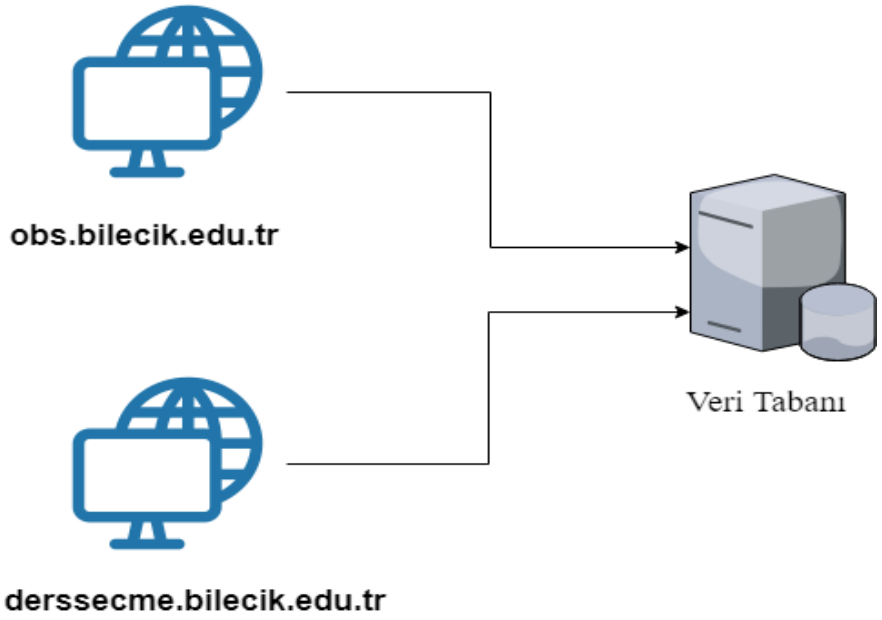
Projede Bootstrap'ın 3.3.7 versiyonu kullanılmaktadır. Bootstrap Css ve Js dosyalarının tamamını ücretsiz ve açık kaynak kodlu bir şekilde paylaşmaktadır. Css ve Js dosyaları client tarafında çalıştığından son kullanıcı sisteme bağlandığında bu dosyalar kullanıcının bilgisayarına indirilir. Buradaki her bir dosya için sunucu ve istemci arasında ayrı bir bağlantı açılır. Bu sebeple indirme işlemi ne kadar kısa sürerse performans açısından o kadar avantajlı olacaktır. İndirilecek dosya boyutunu küçültmek için bootstrap'ın "bootstrap.min.css" ve "bootstrap.min.js" versiyonları kullanılmıştır.

Bootstrap tasarım konusunda bir standart getirmiştir. Projedeki renklendirmeden, butonların boyutuna kadar tüm tasarım nesneleri bootstrap class'ları ile oluşturulmuştur.

Bu da projede bootstrap bilen bir tasarımcının kolaylıkla bir yenilik veya bir değişiklik yapmasına olanak sağlamaktadır. Projede sadece takvim görünümü bootstrap harici geliştirilmiştir. Bu konuda bootstrap'ın hazır çözümü olmadığı için projeye özel takvim tasarlanmıştır.

4 DERS SEÇME MODÜLÜNÜN HAZIRLANMASI

Yeni tasarlanan modülün mevcut sistemden farklı bir mimariyle ve farklı yazılım teknolojileriyle geliştirebilmek için mevcut sistemden ayrı çalışacak şekilde kurgulanmıştır. Bu kurgunun fayda sağlayacağı en büyük kazançlardan biride ders seçmede yaşanan yoğunluk OBS'deki diğer modülleri etkilemeyecek olmasıdır. Bu sayede OBS'de yapılan diğer tüm faaliyetlerin kararlı çalışması ile ders seçme modülünün kararlı çalışması birbirinden bağımsız ilerleyecektir. Birbirinden ayrılan bu iki yapı farklı sunucular üzerinde konumlandırılarak fiziksel olarak da bir birinden ayrılacaktır.

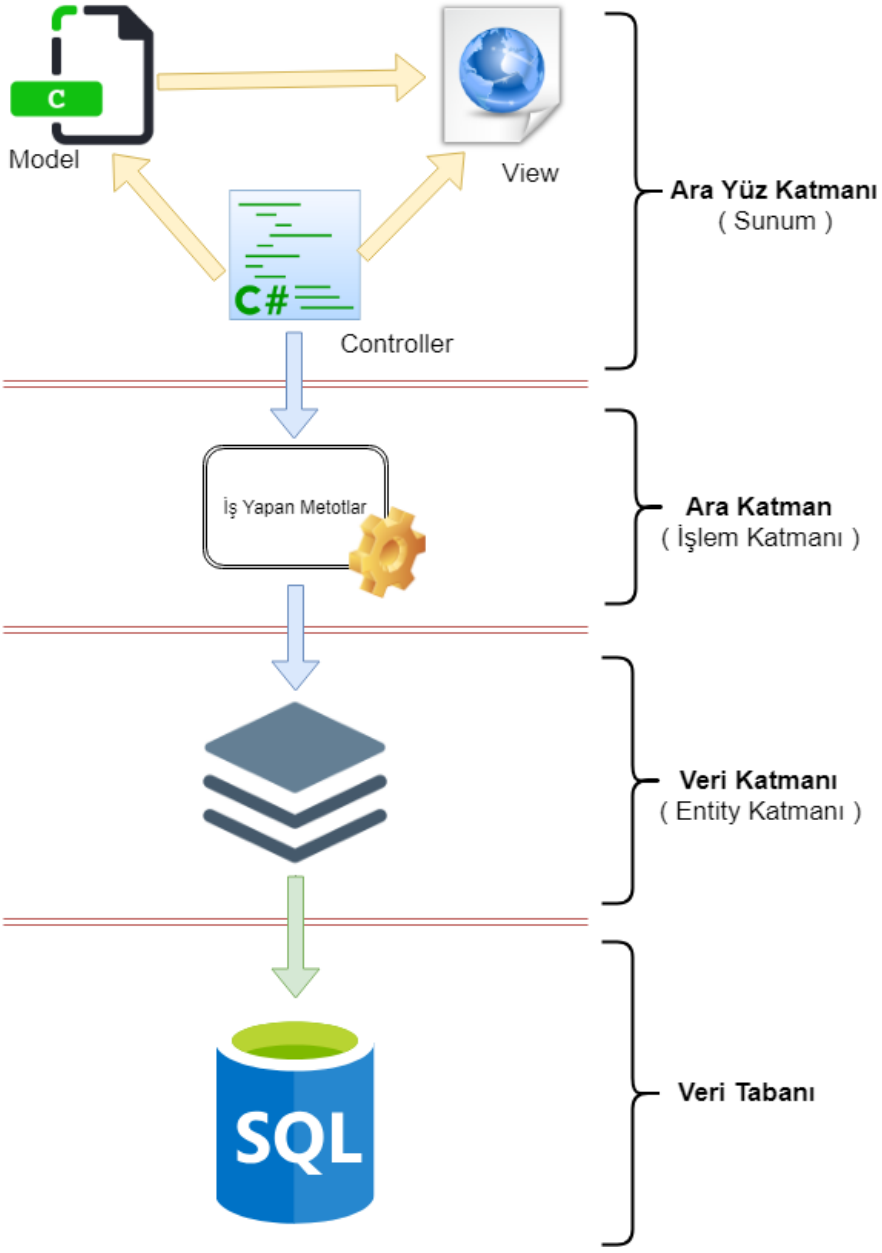


Şekil 4.1. Mevcut otomasyon yazılım mimarisi.

Şekil 4.1'de görüldüğü gibi OBS sistemi "obs.bilecik.edu.tr" alan adıyla hizmete sunulurken ders seçme modülü "derssecme.bilecik.edu.tr" alan adıyla hizmete sunulmaktadır. Ders seçme modülüne OBS içerisinden bir link ile yönlendirildiği gibi son kullanıcılar direkt olarak bu adrese ulaşabilmektedirler.

4.1 Yazılım Mimarisi Tasarımı

Yeni tasarlanan modülde genel yazılım mimarisi olarak ASP.NET MVC mimarisi kullanılmıştır. MVC mimarisi ile birlikte katmanlı mimari kullanılarak modül üç katmana ayrılmıştır. Bu katmalar şekil 4.2'de ilişkileriyle birlikte gösterilmiştir.



Şekil 4.2. Yeni uygulamanın yazılım mimarisi.

Ara yüz katmanını incelediğimizde sayfa bazlı yaklaşım olan Web Form'dan vazgeçilip, metot bazlı yaklaşım olan MVC mimarisi kullanılmıştır. Son kullanıcılar bu

katmandaki view'leri görmektedirler ve yaptıkları talepler arka tarafta bir controller'ı çalıştırmaktadır. Controller gelen talebi alır ve işlem katmanındaki ilgili metoda talepte bulunur. C#'ta yapılacak işlemlerin tamamı burada yapılır. Metot istenilen görevi gerçekleştirdikten sonra cevabı controller'a iletir. Controller gelen cevabı bir model üzerine bindirerek son kullanıcıya bu modeli view ile gösterir. Genel olarak ara yüz katmanında akış bu şekilde oluşmaktadır.

İşlem katmanı mevcut sistemde olamayan ancak yen sistemde var olan bir katmandır. Bu katman ara yüz katmanı ile veri katmanı arasında çalışır. Linq sorgularının yazıldığı ve yapılacak işlemlerin tamamlanıp ara yüz katmanına aktarılması görevini gerçekleştirir. Bu katmanın bize sağladığı en büyük kazanç, Ara yüz katmanından bağımsız projeyi yazmamızdır. Bu sayede ara yüz katmanı sadece görüntüleme yaptığımız bir katman haline gelmektedir. Ara yüz katmanı değiştirilmek istendiğinde yada ders seçme içerisinde yapılan işler başka bir projede kullanılmak istenildiğinde işlem katmanı ve veri katmanı .dll dosyaları istenilen projeye referans edilerek kolayca entegrasyon sağlanabilir. Böylelikle bu katman bize bir nevi ders seçme framework'ü oluşturmayı sağlamaktadır.

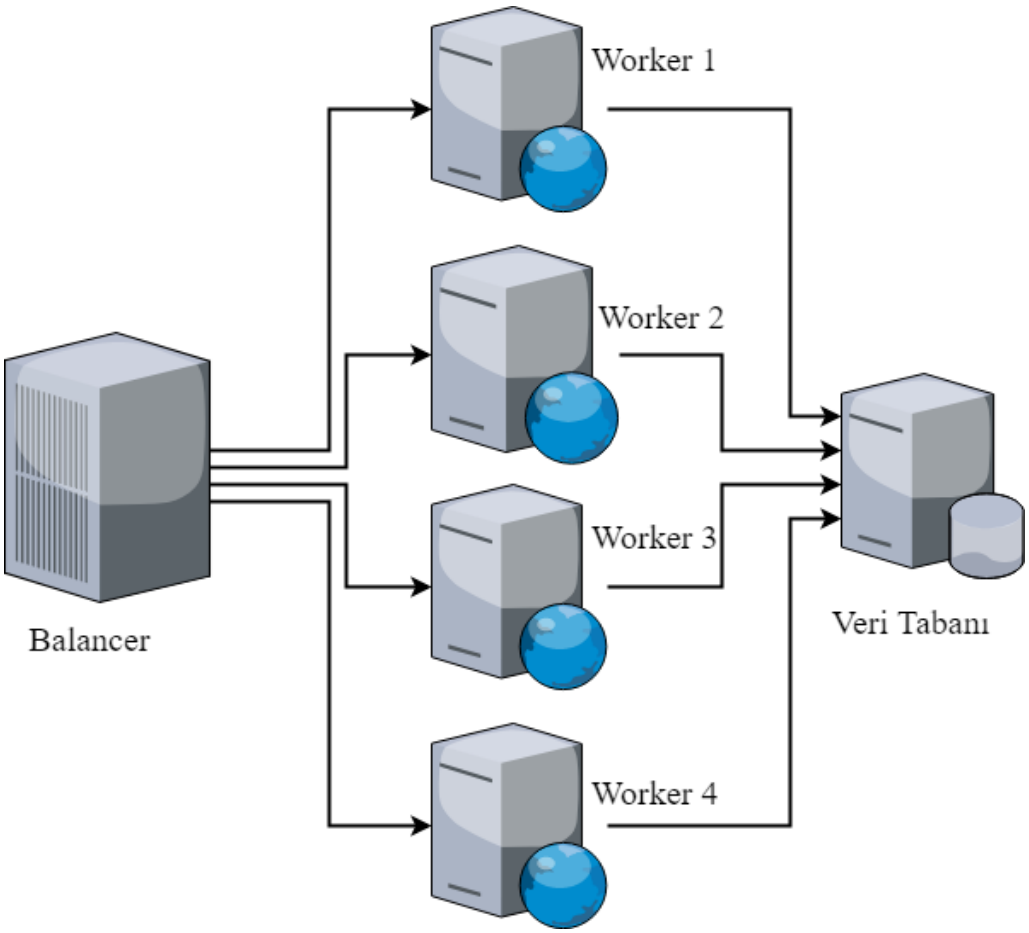
Veri katmanı, veri tabanın C# sınıfları ile modelinin oluşturulduğu katmandır. Bu katman mevcut projede de bulunmaktadır ancak burada minimal şekilde planlanmıştır. Mevcut sistemin modülünde tüm veri tabanı model olarak oluşturulmuşken yeni ders seçme modülünde sadece ders seçmede kullanılan tabloların modeli çıkartılmıştır. Mevcut sistemin modelinde 350 tablo varken yeni oluşturulan modelde 78 tablo oluşturulmuştur. Böylelikle modelin hem boyutu küçülecek hem de sorgulama esnasında yazılan Linq sorguları MSSQL sorgusuna çevrilirken daha sade sorgu cümleciği oluşturulacaktır. Model oluşturulmadan önce veri tabanı hazır olduğu için Database First ile model oluşturulmuştur.

4.2 Sunucu Tasarımı

Mevcut sistemin sunucu yapısında uygulama sunucusu tek bir sunucu üzerinde çalmaktadır. Yeni tasarlanan sunucu planlaması Load Balance teknolojisi ile kurgulanmıştır. Load Balance, birden fazla sunucunun paralel şekilde çalışmasını sağlayan teknolojidir. Ders seçme modülü birden fazla sunucuya publish yapılarak bu sunucuların aynı anda hizmet vermesi sağlanmaktadır. Örneğin bir kullanıcı 1.

Sunucuda ders seçerken başka bir kullanıcı 2. Sunucuda ders seçecektir. Sunucularda çalışan uygulama kodları birbirinin kopyası olacaktır.

Tasarlanan sunucu yapısında 1 balancer sunucusu ve 4 worker sunucusu olmak üzere toplamda 5 tane sunucu planlanmıştır. Sunucu tasarımı ve ilişkileri şekil 4.3'te görülmektedir.



Şekil 4.3. Load balance sunucu yapısı.

Load Balance teknolojisi sayesinde hem iş yükü 4'e bölünmüş olacaktır, hem de IIS kuyruğu düşünüldüğünde kullanıcının önünde biriken işlem sırası 4'e bölüneceği için son kullanıcın alacağı cevap süresi de kısalacaktır. Buradaki kullanılan sunucu sayısı öncelikle üniversitenin bu sistem için ayırdığı kaynakla doğrudan ilişkilidir. Proje için 4 sunucu kullanılması işlem sürelerinin kısalmasıyla birlikte yeterli olacağı düşünülmüştür. Ayrıca sistemde tekrar benzeri problem yaşandığında bu yeni bir sunucuya sistemin kodları atılıp ve balancer sunucusundan yönlendirme yapılarak kısa

bir süre içerisinde sisteme yeni sunucular dahil edilebilir. Bu işlem kritik zamanlarda hızlı çözüm üretilebilmesi adına pozitif etki sağlamaktadır.

4.3 Veri Tabanı İyileştirmesi

OBS projesi 350 tablodan oluşan yaklaşık 7 GB boyutundaki bir veri tabanına sahiptir. Ders Seçme modülü bu veri tabanından toplamda 78 tabloyu kullanarak çalışmaktadır. Tablo sayısı 350'den 78'e azalmış gibi görünse de projede en yoğun kullanılan ve satır sayısı en yüksek olan tabloları içerdiğinden tablo sayısındaki azalma büyük fayda sağlamamaktadır. Bu sayının azalması entity nesnesini küçülteceğinden çalışma anında uygulamanın bellek kullanımını düşürecektir. Projede veri tabanına yapılan sorguların süreleri ölçülmüştür ve sorgular üzerinde iyileştirme çalışmaları yapılmıştır. Özellikle öğrencinin notlarını açılan derslerde gösterebilmek için yapılan sorgu içerisinde bulunan recursive yapı oldukça maliyetli bir sorgudur. Burada sorguda yapılabilecek bir değişiklik olmadığından veri tabanı yapısında yani tablolar üzerinde bir değişiklik yapılması gerekmektedir.

Şekil 4.4. Ders bilgilerinin tutulduğu tabloların ilişki yapısı.

Şekil 4.4'te, yukarıda bahsedilen maliyetli sorgunun kullandığı tabloların ilişkisel görünümü verilmiştir. Tabloların isimlerinden tam olarak hangi amaca hizmet

ettiği anlaşılmadığından aşağıda hangi tablonun hangi verileri tuttuğu detaylı bir şekilde verilmiştir.

OgrenciDers: Öğrencilerin aldığı veya intibakla not getirdiği tüm verileri tutmaktadır.

AkademikPersonelDonemDers: Açılan derslerin tutulduğu bir geçiş tablosudur.

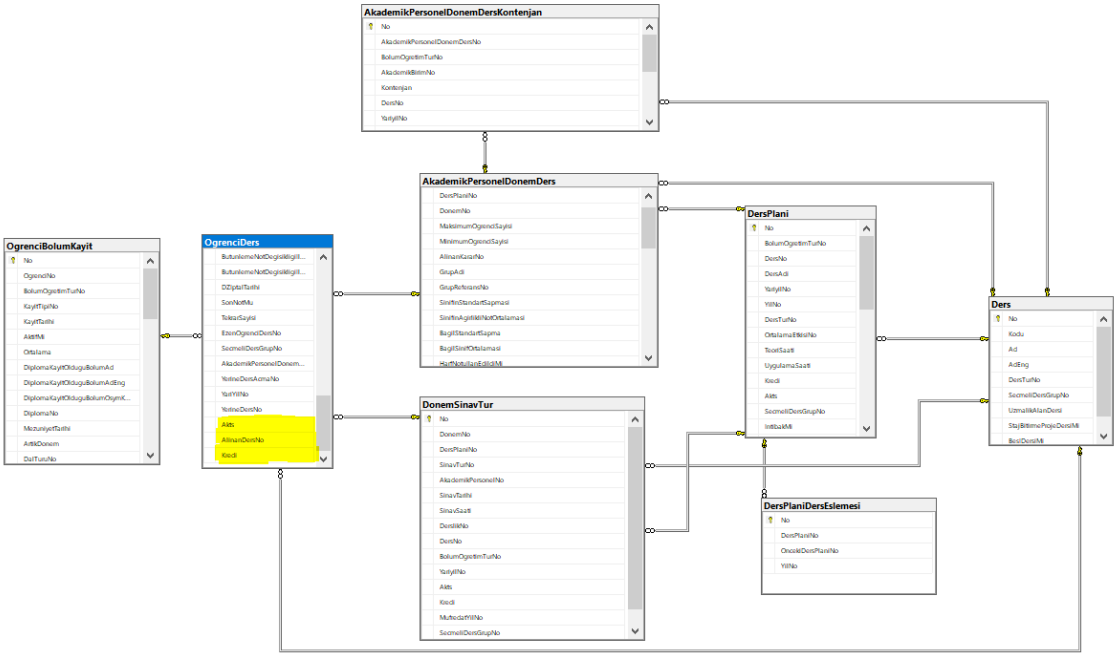
AkademikPersonelDonemDersKontenjan: Açılan derslerden verilen kontenjanların verilerini tutmaktadır.

DonemSinavTur: Öğrencilerin aldığı intibak notlarının verililerini tutmaktadır.

DersPlani: Yıl bazlı müfredat verilerinin tutulduğu tablodur.

Ders: Dersin temel verilerinin tutulduğu tablodur.

Açılan bir derste öğrencin daha önce aldığı notu bulup gösterebilmemiz için dersin değişmiş ihtimaline karşın intibak takibi yapmamız gerekmektedir. Şekil 4.4'te görülen tabloların çok yoğun kullanılan tablolar olması performansı olumsuz etkilemektedir. Örneğin lisans 4. Sınıf bir öğrenci ders seçme ekranında toplam 50 ders görüyorsa burada her bir ders için bu yoğun tablolar da gezilerek hesaplama yapılması gerekmektedir. Ayrıca notun intibaktan mı yoksa seçilen bir dersten mi geldiği bilinmediğinden iki kırılımlı bir sorgu oluşturmaktadır. Bu sebeple yazılımın performansı olumsuz etkilenmektedir. Bu problemin çözümü için veri tabanında bir takım değişiklikler yapılmıştır.



Şekil 4.5. Ders bilgilerinin tutulduğu tablolarda yapılan ilişkisel düzenleme.

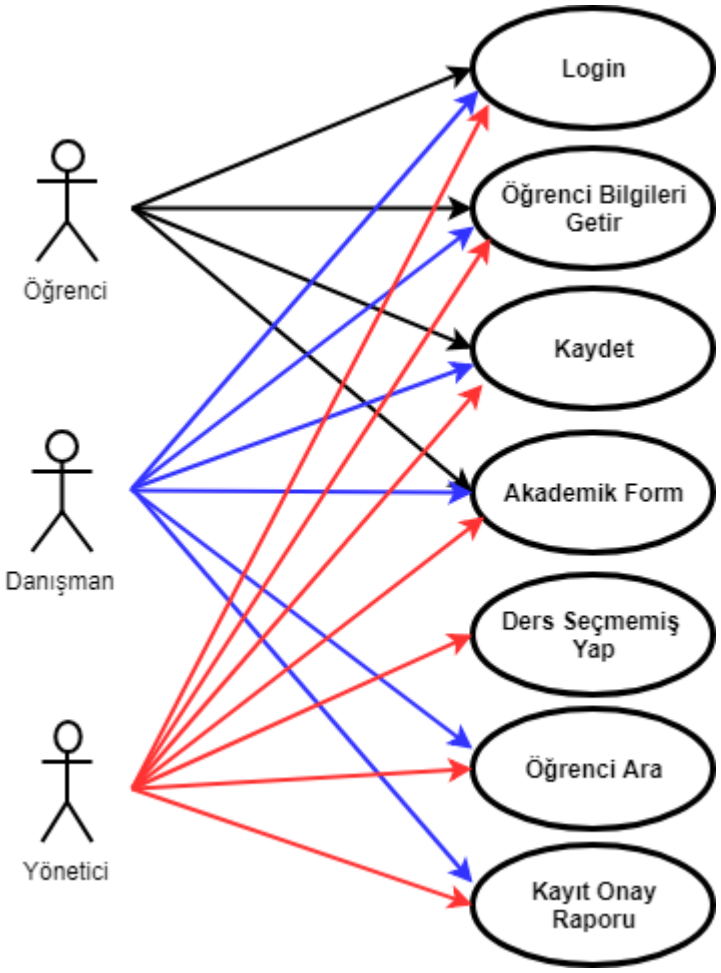
Şekil 4.5'te yapılan değişikklikler incelendiğinde."OgrenciDers" tablosu doğrudan "Ders" ve "SecmeliDersGruplari" tablolarına bağlanmıştır. Ayrıca "DersPlani" tablosunda bulunan AKTS ve Kredi bilgileri "OgrenciDers" tablosu üzerine yazılarak arada bulunan tablolar sorgudan çıkarılmıştır. Bu işlem aradan sadece 3 tablonun eksilmesi gibi görünse de yüksek oranda fayda sağlamaktadır. Arada bulunan üç tablo toplamda bir milyon civarı veriye sahiptir ve ortalama her bir öğrenci için 50 kez sorgulanmaktadır. Bu sorgulamamın haricinde kontenjan yönetimi içinde yoğun bir şekilde sorgulanan bu tablolar ders seçimi esnasında çok sık kullanılmaktadır. Yapılan çalışma sayesinde bu kullanım yoğunluğu da azaltılmıştır.

4.4 Use Case Diyagramları

Use case diyagramı, bir uygulamanın yada sistemin kimler tarafından kullanıldığını ve söz konusu kullanıcıların bu uygulama yada sistemle neler yapabildiklerini özetlemek amacıyla kullanılan bir UML diyagramıdır (Taşdelen, 2015). Bu diyagram sayesinde sistemin rolleri ve bu rollerin yetkili oldukları işlemler kolayca görülebilecektir. Use case diyagramlarını kısaca özetleyecek olursak sistemde kimler, neler yapabilir sorusunun cevabını gösteren çizimlerdir. Bu diyagramlarda işler hakkında detay bilgi verilmez. Yapılan işlemlerin sadece isimleri kullanılır. İşlemler

hakkında detaylı bilgi verilmesi için akış diyagramları ya da aktivite diyagramları gibi işlem odaklı diyagramlardan yararlanılabilir.

Sistem tasarlanırken yönetici, danışman ve öğrenci olarak üç farklı tip role sahip kullanıcı vardır. Öğrenci sadece kendi kullanıcı için işlem yapabilen kısıtlı kullanıcı rolüdür. Danışman sadece yetkili olduğu öğrencilere işlem yapabilen kısıtlı bir kullanıcı rolüdür. Yönetici tam yetkili ve tüm öğrencilere işlem yapabilen kısıtlı olmayan kullanıcı rolüdür. Aşağıdaki diyagramda bu kullanıcı tipleri ve yetkili oldukları durumlar çizilmiştir.



Şekil 4.6. Use Case diyagramı.

Yukarıdaki diyagramın dışında ortak kullanıcı tipi de oluşmaktadır. Hem öğrenci hem danışman olan kullanıcılar sistemde bulunmaktadır. Bu kullanıcı tipi için ise yönetici veya danışman rolüne sahip olan kişiler kendi öğrenciliklerini seçtikleri anda

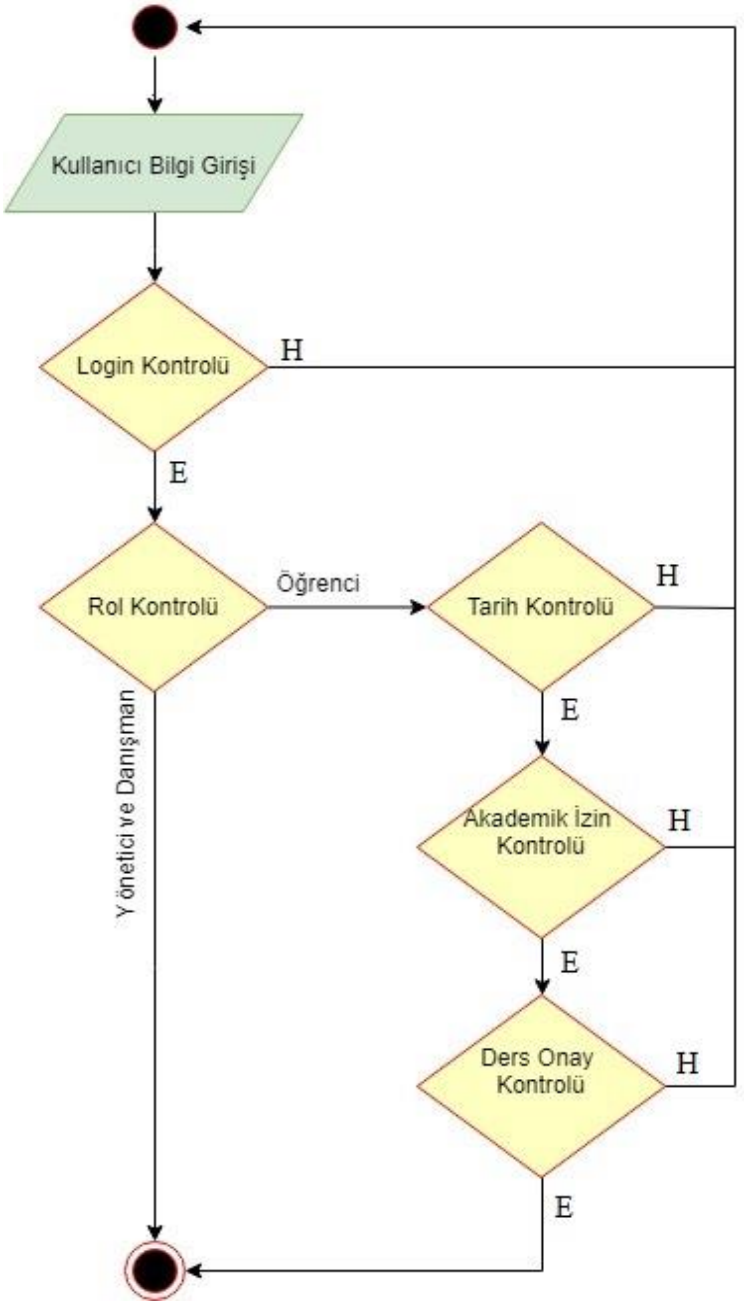
sistem öğrenci gibi çalışmakta iken farklı bir öğrenci seçtiklerinde ise danışman veya yönetici gibi davranmaktadır.

4.5 Aktivite Diyagramları

Aktivite diyagramları iş akışlarının görselleştirilmesi için kullanılır. Literatürde faaliyet diyagramı olarak da adlandırılmaktadır. Aktivite diyagramları bir bakıma use case diyagramlarının tamamlayıcısı gibidir (Taşdelen, 2015). Use case diyagramlarında geçen her bir case'in içeriğini detaylı bir şekilde ifade etmek amacıyla kullanılır. Her bir case burada bir aktivite olarak ifade edilmiştir. Aktivite diyagramları şekil olarak akış diyagramlarına benzerler ancak aralarındaki en önemli fark akış diyagramları detaylı bir şekilde her bir adımı, her bir döngüyü anlatırken aktivite diyagramı detaya girmeden yapılan işlemi özet bir şekilde ifade ederek görselleştirmemizi sağlar. Bu proje kapsamındaki aktivitelerin diyagramları ve bu aktivitelerin detaylı anlatımı bu başlık altında verilmiştir.

4.5.1 Login aktivitesi

Bu aktiviteye tüm kullanıcı rolleri erişebilmektedir. Ancak bu aktivitenin rollere göre farklı davranışları vardır. Yönetici ve danışmanlar için sadece kullanıcı adı ve şifre doğrulaması yapacaktır ancak öğrenciler için ek kontroller yapılmaktadır.

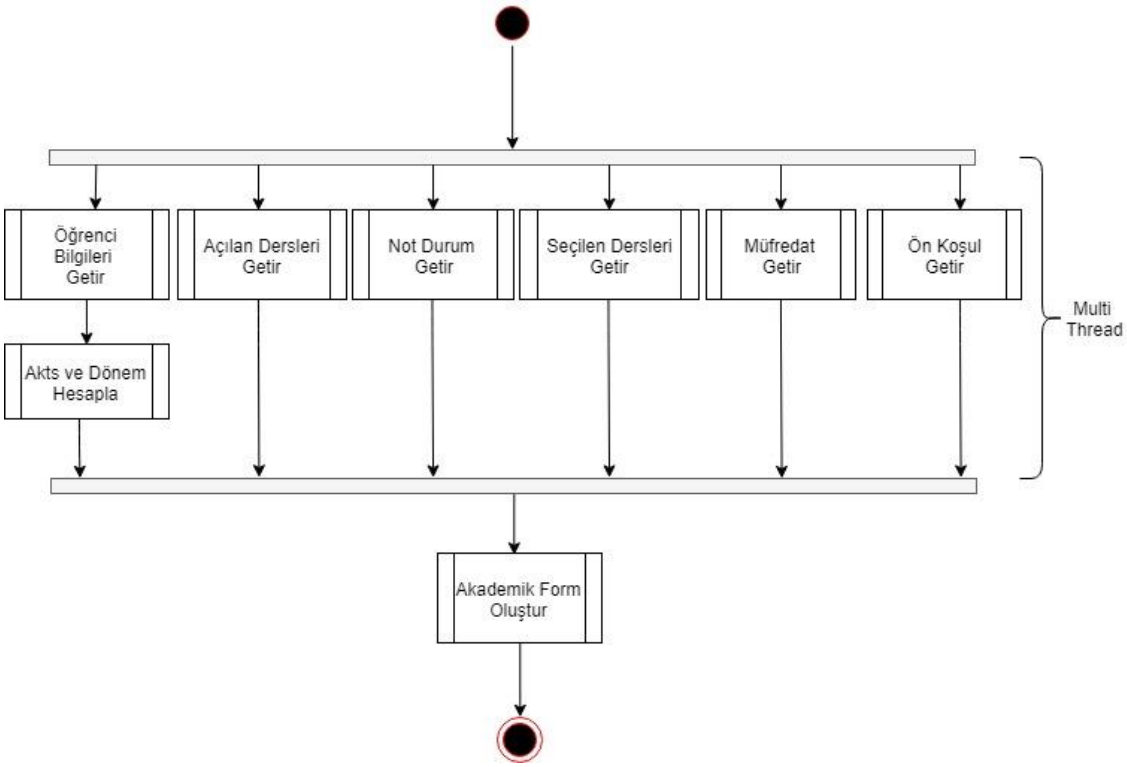


Şekil 4.7. Login aktivite diyagramı.

Login aktivite diyagramı incelendiğinde kullanıcıdan bilgiler alındıktan sonra yönetici ve danışman doğrulama sonrası direkt olarak bitiş adımına geçmektedir. Öğrenci için ise ilk olarak sisteme tanımlı kayıt yenileme haftasında olup olmadığı kontrol edilmektedir, daha sonra öğrencinin akademik izninin olup olmadığı ve ders seçmesi onaylanıp onaylanmadığı kontrolleri yapılmaktadır. Ders seçmesi onaylanan bir öğrenci sisteme girip güncelleme yapamaz. Öğrenci eğer bu kontrollerden geçerse öğrenci aktiviteyi başarılı bir şekilde tamamlayarak sisteme girişi sağlayacaktır.

4.5.2 Öğrenci bilgileri getir aktivitesi

Bu aktiviteye tüm kullanıcı rolleri erişebilmektedir. Aktivite diyagramlarının en önemli özelliklerinden biri paralel çalışan uygulamaların gösterimine uygun olmasıdır. Bu aktivitede de paralel çalışan işlem adımları bulunmaktadır. Öğrenci bilgileri sorgulanırken farklı veri kümeleri Multi Thread teknolojisi ile paralel şekilde sorgulanmıştır.



Şekil 4.8. Öğrenci bilgileri getir aktivite diyagramı.

Şekil 4.8'deki diyagramda görüldüğü üzere öğrenci bilgileri, açılan dersler, not durum, seçilen dersler, müfredat, ön koşullu dersler olmak üzere 6 veri kümesi multi thread yöntem ile sorgulanmıştır. Sistemde en büyük hız kazançlarından biri bu sayede elde edilmiştir. Buradaki veri kümeleri bir öğrencinin ders seçmesinin hazır hale gelmesi için ihtiyaç olan tüm verileri sorgulayıp getirmektedir. Ancak bu aktivite sonunda tüm istenilen veriler sorgulansa da veriler son kullanıcıya sunulmaya uygun değildir. Bu veriler Java Script'te son kullanıcıya hazır hale getirilmesi için hesaplamalar yapılacaktır. Hesaplamaların bu aktivite içinde yapılmayıp JS'de planlanmasının sebebi, veriler elde edildikten sonra sunucu tarafındaki işlemi bitirip

istemci tarafında verilerin istenen formata getirilmesini sağlamaktır. Böylelikle sisteme giren her kullanıcının verisi kendi bilgisayarı üzerinde verileri işleyecektir. Bu işlem de performans kazancı sağlayacaktır.

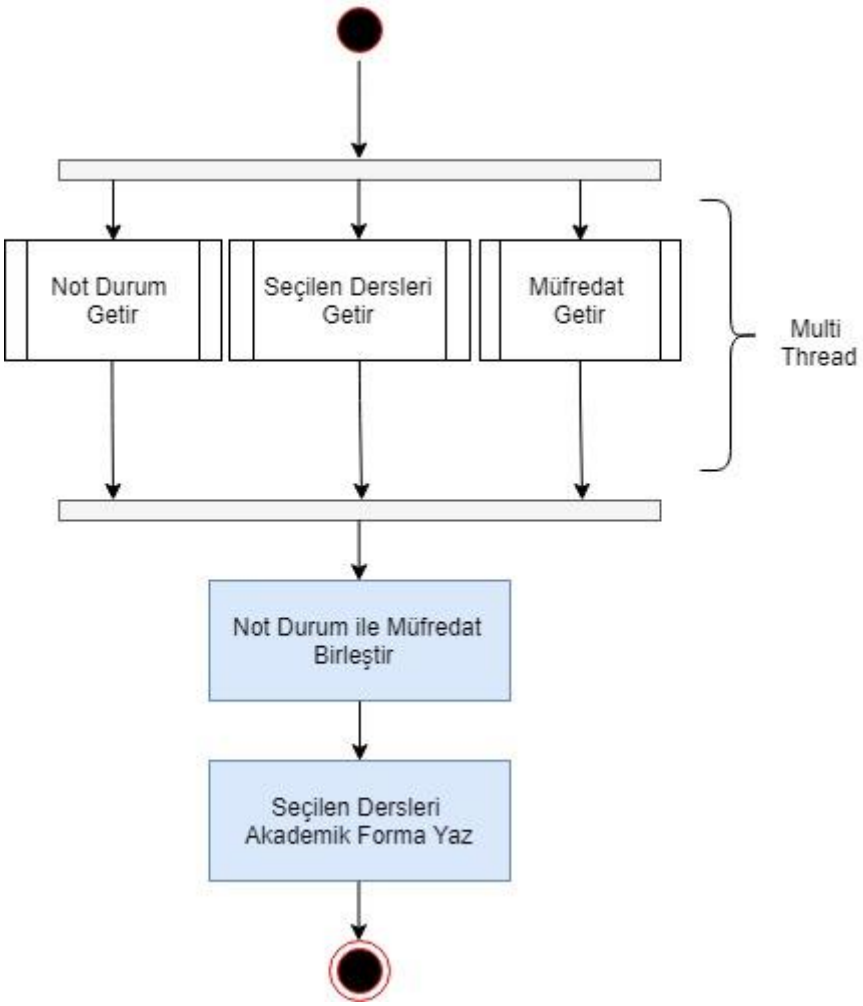
Diyagramda dikkat çeken diğer bir nokta ise akademik form işleminin paralel olarak yapılmayıp aktiviteye ekstra bir adım olarak eklenmesidir. Bu performans kaybına sebep olan bir durum gibi gözükse de akademik formun çalışma yapısı incelendiğinde performansı pozitif yönde etkilemek için getirilen bir aşama olduğu görülmektedir. Akademik form, not durum ve müfredat ders verileri ile beslenen bir formdur. Diyagram adım adım takip edildiğinde paralel sorgulamanın yapıldığı adımda not durum ve müfredat dersleri veri tabanından getirilmektedir. Akademik form içinde bu verileri tekrar sorgulamak yerine bu veriler geldikten sonraki adımda formu oluşturmak performans açısından daha avantajlı olmaktadır.

4.5.3 Akademik form aktivitesi

Akademik form öğrencinin mezuniyetini en kolay ve hızlı şekilde kontrol edilmesini sağlayan formdur. Dolayısıyla öğrencinin alması gereken dersler bu form üzerinden takip edilebilmektedir. Bu listedeki her bir derse karşılık gelen öğrencinin not aldığı derslerde listeye eklenir ve bu iki ders formda yan yana listelenmektedir. Not aldığı derslere ek olarak bu dönem seçtiği derslerde akademik forma eklenir. Böylelikle mezun durumunda olan öğrenci formdaki tüm dersleri tamamlayıp tamamlamadığı kontrol edilerek öğrencinin mezun olup olamayacağı rahatlıkla kontrol edilebilir.

Akademik form aktivitesine sistemde bulunan tüm kullanıcı rolleri erişebilmektedir. Bu aktivite 2 farklı yöntem ile tetiklenip çalışmaktadır. Sayfa açılırken 1. kez sistemin ihtiyacı olan veriler çekildiğinde çalışmaktadır. Bir önceki başlıkta öğrenci verileri getiren aktivite içerisinde buna değinilmişti. Daha sonra kaydet veya ders seçmemiş yap aktiviteleri çalıştırılmadığı takdirde mevcut form tekrar görüntülenmekte ve veri tabanına herhangi bir sorgulama yapılmayıp ilk form tekrar açılmaktadır. Ancak bu iki aktiviteden biri çalıştırılırsa formda değişiklik olabileceğinden mevcut form 2.'ye çalışmak üzere hazırlanır ve akademik form butonu tıklandığında ikinci çalışması ilkinden farklı olarak ihtiyaç duyduğu verileri tekrar çeker ve formun güncel halini oluşturur. Bu şekilde planlanmasının sebebi özellikle mezun durumunda olan öğrenciler için siteme girdiğinde birkaç kez formu açıp kapatma

ihtiyacı olabiliyor buradaki her yapılan tekrar işlem için veri tabanında oluşacak yoğunluğun önüne geçmektir.



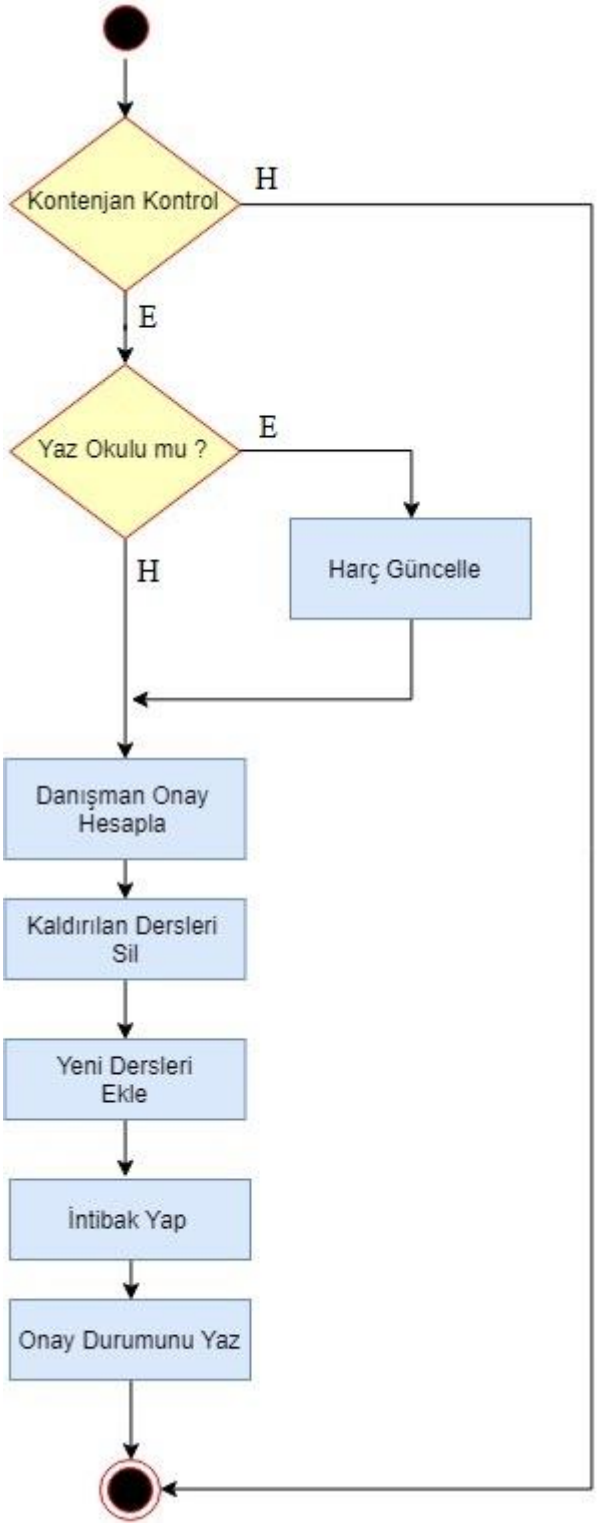
Şekil 4.9. Akademik form görüntüleme aktivite diyagramı.

Şekil 4.9'daki diyagram 2. Kez tetiklendiğinde çalışan aktiviteyi göstermektedir. İlk çalışan aktivite öğrenci bilgileri getir aktivitesi içerisinde gösterilmiştir. Burada öğrenci bilgilerini getirirken kullanılan Multi Thread teknolojisi tekrar kullanılmaktadır. Not durum, müfredat ve seçilen dersler paralel olarak sorgulanmıştır. İlk olarak müfredat forma yerleştirilip daha sonra not durum ile birleştirilmektedir. En son seçilen dersler müfredat dersleri ile eşleştirilmektedir. Veri tabanında yapılan geliştirme sayesinde buradaki eşleştirmede intibak takibi yapılmamaktadır. İntibak takibindeki recursive sorgulama adımı eksilmesiyle birlikte akademik formdaki hesaplama süresi

kısaltılmıştır. Burada hazırlanan form verisi ara yüzde derslerin başarı durumlarına göre renklendirilerek kullanıcının dikkatini çekecek şekilde sunulmuştur.

4.5.4 Kaydet aktivitesi

Bu aktiviteye tüm kullanıcı rolleri erişebilmektedir. Kaydet aktivitesi seçilen derslerin veri tabanına kaydedilmesine, harç hesaplanmasına ve yetkili kullanıcılar tarafından onay işleminin yapılmasını sağlamaktadır.



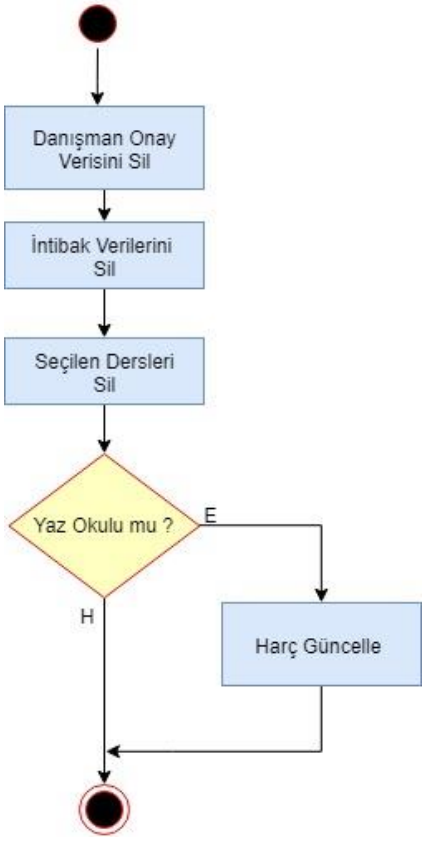
Şekil 4.10. Kaydet aktivite diyagramı.

Kaydet aktivite diyagramında ilk olarak seçilen derslerin kontenjan dolulukları kontrol edilmektedir. Eğer kontenjanı dolmuş bir ders var ise hiçbir işlem yapmayıp aktivite sonlanmaktadır. Kontenjan doluluk kontrolünden sonra yaz okulu kontrolü

yapılmaktadır. Eğer sistem yaz okulunda ise seçilen ders AKTS'sine göre harç hesaplanıp öğrenciye harç tanımlanmaktadır. Yaz okul kontrolünden sonra kayıt onay kontrolü yapılmaktadır. Aktiviteyi başlatan rol yönetici veya danışmansa ve öğrenci harcını yatırdı ise öğrencinin dersleri onaylanacaktır. Kontroller tamamlandıktan sonra işlem yapılan öğrenci daha önce ders seçtiyse önceki seçim ile şu anki seçim arasında fark var mı diye kontrol edilir ve seçilen ders listesinden çıkartılan dersler var ise silinir. Daha sonra yeni eklenen dersler veri tabanına yazılır ve seçmeli derslerde yerine işlemi yaptıysa o derslerin intibakları tanımlanır. Son olarak yukarıda hesaplanan onay durumu veri tabanına yazılıp aktivite tamamlanır.

4.5.5 Ders seçmemiş yap aktivitesi

Bu aktiviteye sadece yönetici rolüne sahip kullanıcılar erişebilmektedir. Bu aktivitenin amacı seçilen öğrencinin ilgili dönemi için ders seçme modülünün yazdığı tüm verileri silerek öğrenciyi ders seçmemiş eski haline döndürmektir.

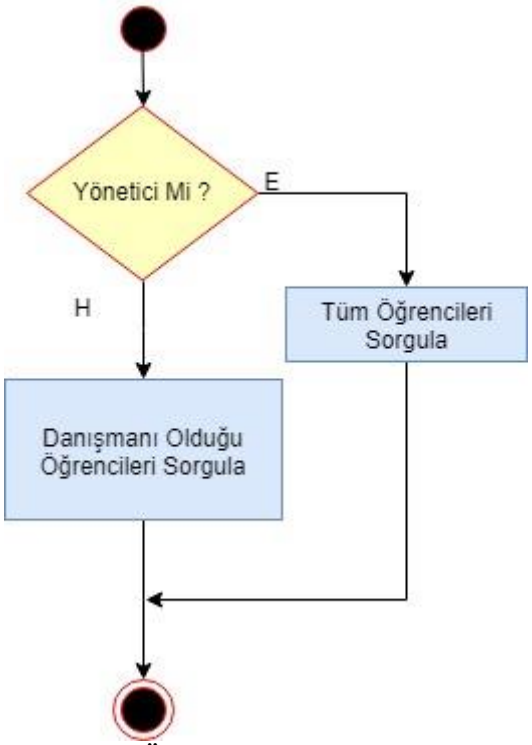


Şekil 4.11. Ders seçmemiş yap aktivite diyagramı.

Bu aktivite kaydet aktivitesinin tam tersi şekilde çalışmaktadır. Aktivite içerisinde yaz okulu kontrolü bulunmaktadır. Eğer dönem yaz okulu ise öğrencinin harcını silmektedir. Bu aktivite sonrasında öğrencinin tüm verileri silinip hiç ders seçmemiş haline dönmektedir. Aktivite tamamlandıktan sonra öğrenci üzerinde hiçbir veri kalmasa da önceki yapılan işlemler log kayıtlarında tutulmaktadır.

4.5.6 Öğrenci ara aktivitesi

Bu aktiviteye tüm kullanıcılar erişebilmektedir. Ancak öğrenciler için özel durum vardır. Öğrencilerden sadece iki farklı kaydı olan öğrenciler sorgulama yapabilmektedir. Bu sorgulama ile diğer öğrenciliklerine geçiş yapabilmektedirler. Sorgulama yaparken T.C. , ad, soy ad ve bölüm adına göre sorgulama yapılabilmektedir. Ayrıca veri girişi yapmadan sorgulandığında ise tüm öğrencileri getirmektedir.



Şekil 4.12. Öğrenci ara aktivite diyagramı.

Bu aktivitede yönetici tüm öğrencileri sorgulayıp işlem yapabilirken danışman ise sadece danışmanlık yaptığı öğrencileri listeleyebilmektedir. Yönetici veya danışmanın kendi öğrenciliği var ise kendisini de listeleyebilmektedir. Bu durumda

kendisini seçtikten sonra yaptığı tüm işlemleri sistem tarafından öğrenci rolü ile yaptırılmaktadır.

4.5.7 Kayıt onay aktivitesi

Bu aktiviteye yönetici veya danışman rolüne sahip kullanıcılar erişebilmektedir. Kayıt onay aktivitesi sonucunda rapor oluşmaktadır. Bu rapor danışman ve öğrencinin karşılıklı anlaşarak ders seçme sürecini tamamladıklarının bir sözleşmesidir. Ders seçme işlemi tamamlanıp danışman onayı yapıldıktan sonra öğrenci ve danışman birer nüsha alıp imzalayarak nihai ders seçme işleminin bu olduğunu belgelerler.

Şekil 4.13. Kayıt onay aktivite diyagramı.

Bu aktivite sadece raporlama işlemi yaptığı ve herhangi bir hesaplama işlemi yapmadığı için en sade aktivitedir. Bu aktivitenin aynısı OBS içersindede bulunmaktadır. Kullanıcılar istedikleri zaman OBS içerisinden bu rapora ulaşabilmektedirler.

4.6 İşletilen Kurallar

Ders seçme modülünde üniversitenin yönetmelik maddeleriyle belirlenmiş kurallar bulunmaktadır. Bu kulların birçoğu mevcut sistemde C#'ta yani sunucuda işletilirken yeni sistemde bu kuralları JavaScrip'te taşınıp kullanıcı bilgisayarlarında

çalışmasını sağlanmıştır. Bu işlem ile sunucu yükü istemci bilgisayarlarına taşınarak performans kazancı sağlanmıştır.

4.6.1 Akts sınırı kontrolü

Öğrencilerin her kayıt döneminde seçebilecekleri AKTS miktarı yönetmelikle sınırlandırılmıştır. Öğrenciler başarısına, kayıt tipine ve dönemine göre seçebilecekleri AKTS sınırı değişmektedir. Öğrenci bilgileri veri tabanından sorgulandıktan sonra bu sınır hesaplanmaktadır. Bu sınırlar özel durumlar hariç 30 veya 45 olarak belirlenmektedir. Yaz okulu dönemlerinde ise sınır AKTS olarak değil 4 adet ders olarak belirlenmiştir.

Öğrenci her dersi tıklandığında ders seçilen derslere eklenmeden önce araya girerek seçilen toplam AKTS ile dersin AKTS'si toplanarak sınır kontrol ediliyor eğer sınırın üzerinde çıkıldı ise dersin seçilmesine izin verilmemekte ve öğrenciye uyarı gösterilmektedir.

Şekil 4.14. AKTS sınırı aşma uyarısı.

Şekil 4.14'te AKTS sınırını aşan öğrencinin aldığı uyarının örneği görülmektedir. Bu kontrol JS'de yapılıp sunucuya yük bindirmeden istemci üzerinde çalışacak şekilde planlanmıştır.

4.6.2 Çakışma kontrolü

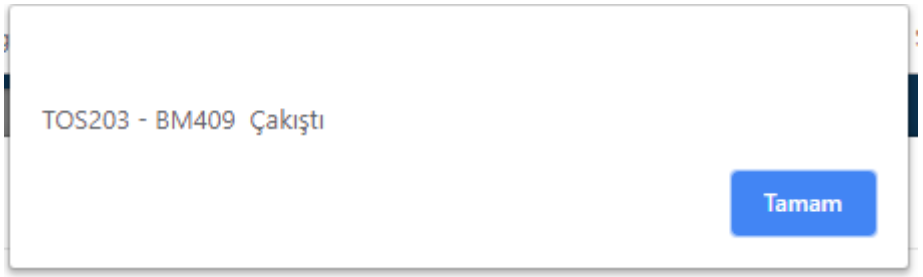
Öğrencilerin seçebileceği dersler devam zorunlu ve devam zorunlu olmayan olarak iki gruba ayrılmıştır. Devam zorunlu dersler; üzerinde not olmayan yeni seçilmiş dersler ve daha önce alınmış "DZ" notu ile kalınmış derslerdir. Öğrenci bu derslerden birini seçerse dönem içinde ilgili derse devam etmek zorundadır. Devam etmediği takdirde dersi veren akademik personel öğrenciyi devamsızlıktan bırakabilir. Devam zorunlu olmayan dersler; Öğrencinin "DZ" notu haricinde daha önce geçer ya da kalır

not aldığı tüm derslerdir. Öğrenci bu derslerden birini seçtiği takdirde devam zorunluluğu aranmayacak ve dersi veren akademik personel istese dahi öğrenciyi devamsızlıktan bırakamayacaktır. Öğrenci devam zorunlu olmayan derslerden istediği kadar aynı gün ve saatte ders alabilir ama devam zorunlu derslerden çakışan iki dersi seçemez.

Şekil 4.15. Devam zorunlu derslerim gösterimi.

Devam zorunlu olan dersler şekil 4.15'te görüldüğü gibi mavi kar tanesi sembolü ile gösterilmiştir. Kullanıcı mause'u üzerinde getirdiğinde tooltip olarak bu derslerin zorunlu olduğu açıklamada yazmaktadır. Öğrenci bu derslerden herhangi ikisini aynı gün ve aynı saatte işleniyorsa seçemez.

Bu kontrol öğrenci dersi tıkladığında ders, seçilen derslere eklenmeden önce araya girerek yapılır. Çakışma kuralına takılan öğrenci dersi seçemez öğrencinin karşısına uyarı gösterilir ve öğrencinin karşısına varsa dersin ikinci öğretimine verilen kontenjanlar listelenir eğer öğrenci buradaki kontenjanlardan seçebileceği bir ders var ise seçer ve bu seçimde de tekrar çakışma kontrolü yapılmaktadır.

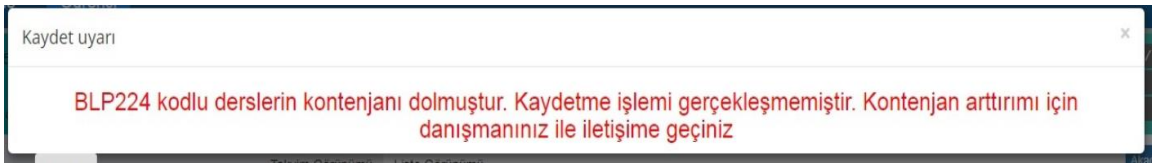


Şekil 4.16. Çakışma uyarısı.

Şekil 4.16'da çakışma uyarısı örneği gösterilmiştir. TOS203 ve BM409 kodlu dersler çakışmıştır. Bu çakışma sonrası öğrenci çakışan derslerden üst yarıyıldaki dersin varsa ikin öğretimini görebilecektir. Çakışma kontrolü de JavaScript'te yapılmıştır ve kullanıcı bilgisayarında çalışarak bize performans kazancı sağlamaktadır.

4.6.3 Kontenjan doluluk kontrolü

Üniversitedeki tüm derslerde o dersi kaç tane öğrencinin seçebileceği kontenjanla sınırlandırılmıştır. Kontenjanlar dersin açıldığı kendi bölümüne verilebildiği gibi farklı bölümlere de verilebilmektedir. Kontenjan verme yapısı oldukça esnek bir yapıya sahip bir zorunlu ders başka bir zorunlu derse, seçmeli dersin altındaki bir derse veya seçmeli grubun kendisine verilebilir. Kontenjan verilirken o dersi hangi bölümden maksimum kaç kişi seçebileceği belirlenmiş olur.



Şekil 4.17. Kontenjan doluluk uyarısı.

Bu kontrol kaydet aktivitesi içerisinde C#'ta çalışmaktadır. Kontrollerin birçoğunu JS'e taşırken bunu server tarafında bırakmamızın sebebi kontenjanın kaydet anında anlık olarak veri tabanından sorgulanması gerektiğinden kaynaklanmaktadır. Kaydet aktivitesinin en başında bu derslerin kontenjanları kontrol edilir ve eğer derste yeterli kontenjan yok ise kaydet aktivitesi bitirilir ve öğrencinin karşısına yukarıda şekilde gösterilen uyarı getirilmektedir.

4.6.4 Harç hesabı

Harcın hesaplanması YÖK'ün üniversitelere belirlediği yöntem ile yapılmaktadır. Öğrenciler güz ve bahar dönemlerinde eğer YÖK'ün belirlediği harç koşullarını sağlıyorlarsa ders seçme öncesinde öğrencilere harçlar tanımlanır. Bu dönemlerde ders seçme modülü üzerinde herhangi bir harç hesaplaması yapılmaz ancak yaz dönemlerinde hesaplama yapılacaktır. Yaz okulunda öğrencinin seçtiği AKTS ve birim ücret katsayısına bağlı olarak hesaplama işlemi yapılır ve öğrencinin harcına yazılır. Harç hesaplama formülü aşağıda gösterilmiştir.

Harç Tutarı = AKTS * Birim Katsayı Ücreti

Birim katsayı ücreti her birim tarafından ayrı olarak tanımlanmaktadır. Yukarıdaki formül her bir ders için uygulanıp toplamı alınarak öğrencinin toplam ödemesi gereken tutar hesaplanır. Öğrenciler burada hesaplanan tutarı banka üzerinden ödeyerek harçlarını yatırırlar.

4.6.5 Önkoşul kontrolü

Ön koşul; öğrencinin, ön koşul belirlenen bir derste başarılı veya o dersi alamadan, diğer koşullu dersi öğrencinin seçmesini engelleyen kuraldır. Bir ders başka bir dersle koşullandığı gibi zorunlu bir ders herhangi bir grupla yada gurubun altındaki bir ders ile yada birden fazla ders veya grupla koşullanabilmektedir.

Bu kural için tanımlanan koşullar derslerin açılan halleri üzerinden değil direkt olarak öğrencinin müfredatı üzerinden yapılmaktadır. Öğrenci tanımlanan koşulların yeni hallerini sağlarsa yada dışardan farklı bir ders getirip bu ön koşula intibak yaptırırsa öğrenci gerekli olan şartı yerine getirmiş sayılacaktır. Bu sebeple kontrol not durum üzerinden değil notların müfredat karşılıkları üzerinden ve koşullu dersler de açılan derslerden değil o derslerin müfredat karşılıkları üzerinden takip edilecektir.

Çizelge 4.1 Ön koşul örnek intibak tablosu.

2014- 2015	2015-2016	2016-2017
YBD202	İNG202	ENG202
İNG101	ENG101	ENG101

Çizelge 4.1'de ön koşullu olup intibak yapılan iki dersin örnek tablosu verilmiştir. Tablodaki YBD202 dersi sonraki yıl İNG202 dersi olmuş daha sonraki yıl ENG202 dersi olmuştur. Başka bir İNG101 dersi de ENG101 dersi olmuş ve daha sonra değişmemiş olsun. Örnekteki YBD202 dersinin İNG101 dersine koşullandığını düşünürsek. 2014 müfredatına sahip bir öğrenci için 2016-2017 senesinde ENG202 ve ENG101 dersleri açılmaktadır. Koşul verisine bakıldığında bu iki ders için herhangi bir koşul gözükmezken müfredatta tanımlanan YBD202 ve İNG101 dersleri üzerinden bu koşul takip edilmektedir. Bu sayede ders kaçıcıya intibak geçirip değişirse değişsin öğrenci bu intibakların hangisinden şartı sağlarsa sağlasın önemi yoktur. Biz bu iki dersin kökü olan müfredat hallerinden takibi yaptığımızda sorun olmayacaktır.

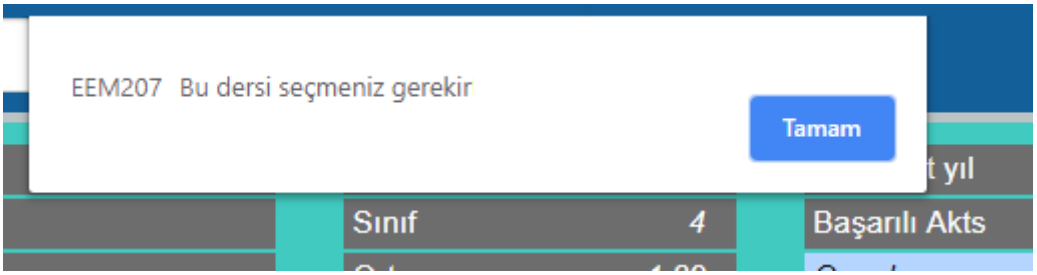
Bu kontrolün tamamı JS'de yazılmıştır. Öğrenciler kontenjanı tıkladıkları anda bu kontrol araya girerek şartları sağlayıp sağlamadığına bakar ve şartlar sağlanmıyorsa öğrenci karşısına uyarı çıkartmaktadır. Öğrenciler kendi müfredatlarına tanımlı olan tüm koşullu ders listesine OBS'de bulunan ana sayfalarından erişebilmektedirler.

4.6.6 Alttan almaya zorlama kontrolü

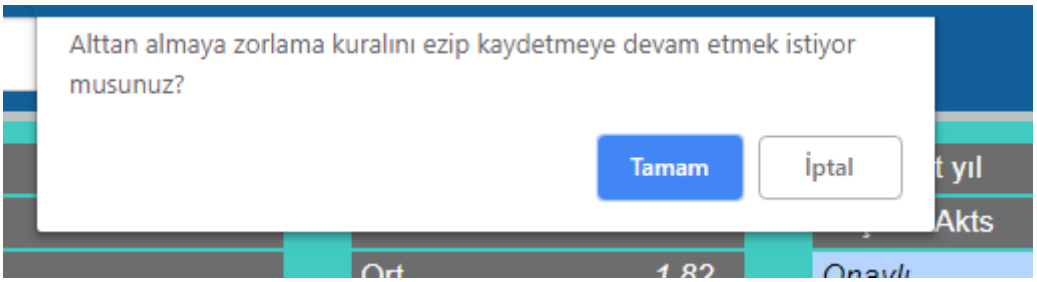
Alttan almaya zorlama kuralı üniversitenin ilgili yönetmeliğinde belirtilen, bir öğrencinin ders seçerken öncelikle en alt yarı yıldan başlayarak kendi dönemine kadar, kaldığı veya hiç almadığı zorunlu dersleri alıp ve başarma şartlarını sağlamadığı seçmeli ders gruplarından alması gereken kadar ders alıp, daha sonra kendi döneminde varsa kaldığı dersleri alıp kendi dönemindeki hiç almadığı derslerden AKTS sınırına kadar istediği kadar ders seçmelerini sağlayan kuraldır. Kuralda bahsedilen grubu başarma şartı, grubun AKTS'si kadar grup altındaki derslerden başarılı olmak ve grup altında kalır dersinin olmaması durumudur. Örneğin 5 AKTS'lik bir grup altında öğrencinin 5 AKS'lik başarılı dersi varsa ve 2 AKTS'lik te başarısız dersi varsa o grup için yeteri kadar başarılı dersi olsa dahi grup başarısız sayılmaktadır.

Bu kural sistemin en karışık algoritmaya sahip kuralıdır. Öğrenci not durumlarının alabileceği binlerce farklı kombinasyon vardır ve bu durumların hepsi alttan almaya zorlama kuralıyla doğrudan ilişkilidir. Dolayısıyla bu tüm kombinasyonlara doğru cevap veren bir algoritma geliştirilmiştir. Alttan almaya zorlama kuralı öğrenci veya danışman rolü olan kullanıcıları kısıtlarken yöneticiler için kısıtlama uyarısını verdikten sonra "devam etmek istiyor musunuz?" sorusunu sorarak yöneticiye kuralı ezme imkanı sağlamaktadır. Özel durumu olan öğrenciler için bu

kuralın kontrol edilmemesi gerekebilir bu tür durumlara yönetici marifetiyle ders seçme yaptırabilmek için bu şekilde bir tasarım yapılmıştır. Ayrıca mühendislik tamamlama, özel öğrenci, misafir öğrenci, farabi, erasmus ve mevlana kayıt tipine sahip öğrenciler alttan almaya zorlama kuralından hariç tutulmuşlardır.



Şekil 4.18. Alttan almaya zorlama kuralı öğrenci uyarısı.



Şekil 4.19. Alttan almaya zorlama kuralı yönetici uyarısı.

Bu kontrol son kullanıcı kaydet butonuna bastığı anda C#'a verileri göndermeden önce JS tarafında yapılıp, eğer kurala takılıyorsa işlemi durdurmaktadır. Karmaşık bir algoritmaya sahip olan bu kuralın JS'de yazılması performans kazancı sağlamıştır.

4.7 Ara Yüz Tasarımı

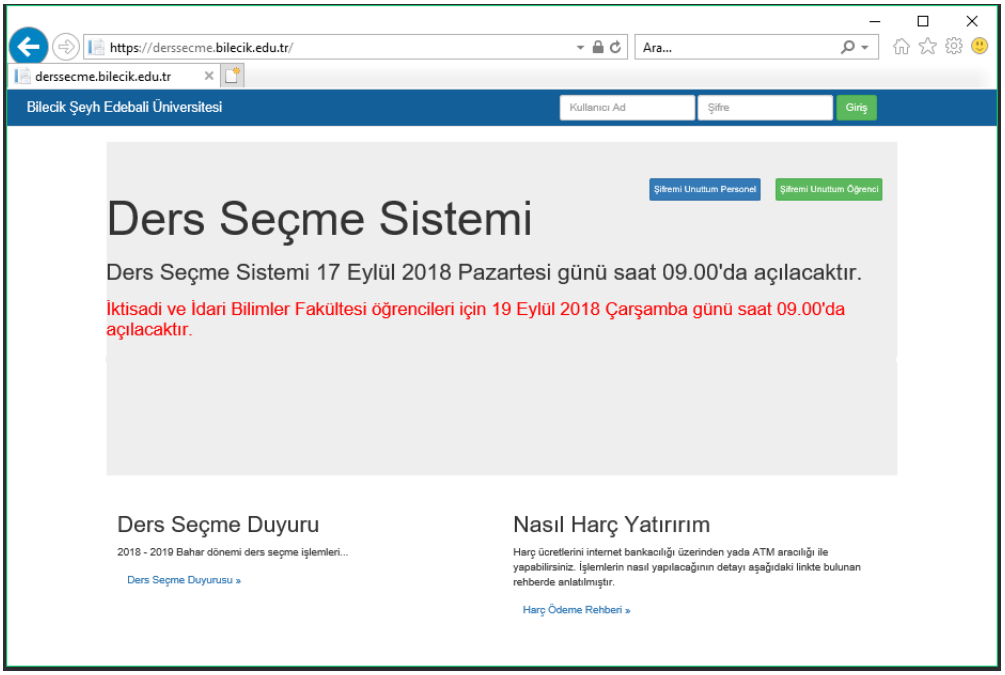
Ara yüzlerin tasarımı sistemin görsel yüzü olması ve yazılımın son kullanıcıya direkt olarak temas ediyor olması sebebiyle oldukça önemlidir. Ara yüzler tasarlanırken sade ve anlaşılır olmasına özen gösterilmiştir.

Arayüz tasarımında bootstrap tasarım kütüphanesi kullanılmıştır. Bu kütüphane sayesinde tasarım aşamasının daha hızlı yazılabilmesine olanak sağlanmıştır. Ayrıca bootstrap ile tasarım yapılan kodlar ile standart sağlanmıştır ve projeye başka bir yazılımcı müdahale etmesi kolaylaşmıştır.

Son yıllarda tablet ve telefon kullanımının artmasıyla birlikte sistemi dokunmatik ekranlar ile kullanan son kullanıcı sayısı oldukça artmıştır. Sistemin kullanım verileri de incelendiğinde özellikle telefon üzerinden ders seçen öğrenci sayısı oldukça fazladır. Bu sebeple tasarım hem farklı boyuttaki ekranlara hem de dokunmatik ekranlarda uyumlu çalışacak şekilde planlanmıştır.

4.7.1 Login

Kullanıcılar derssecme.bilecik.edu.tr adresine ilk giriş yaptıklarında karşılaştıkları ara yüzdür. Kullanıcılar sisteme giriş yapabilmeleri için bu ara yüzden kullanıcı ad ve şifrelerini girerek doğrulama yaptırıp sisteme giriş yapabilmektedirler.



Şekil 4.20. Login ekranı.

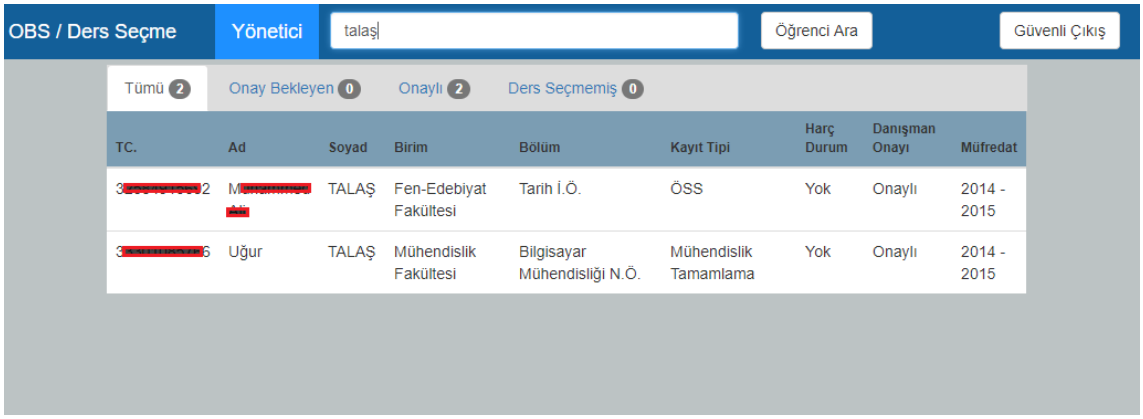
Bu ara yüzde kullanıcılar için duyuru kılavuzlar yayınlanmaktadır. Öğrenciler bu sayfadan ders seçme ile ilgili bilgilendirmeleri almaktadırlar. Ayrıca şifresini unutan personel ve öğrenciler için kullanıcı bilgilerini güncelleyebilecekleri sofra.bilecik.edu.tr'ye yönlendirme yapan iki adet buton bulunmaktadır.

4.7.2 Banner

Login yapıldıktan sonraki sayfanın en üst bölümünü oluşturan kısımdır. Banner içinde kullanıcının rolünü ve güvenli çıkış butonunu içermektedir. Yönetici ve danışmanlar içinde banner içerisinde arama çubuğu bulunmaktadır.

Şekil 4.21. Banner.

Arama çubuğunda öğrencilerin T.C, ad, soy ad, bölüm, birim bilgilerinden herhangi birini veya bir kaçını yazarak arama yaptırılabilir. Burada yazılan arama cümlesinde boşluklar bırakılarak yazılan her bir kelime ayrı ayrı istenilen verilerin arasında var mı diye sorgulama yapılmaktadır.

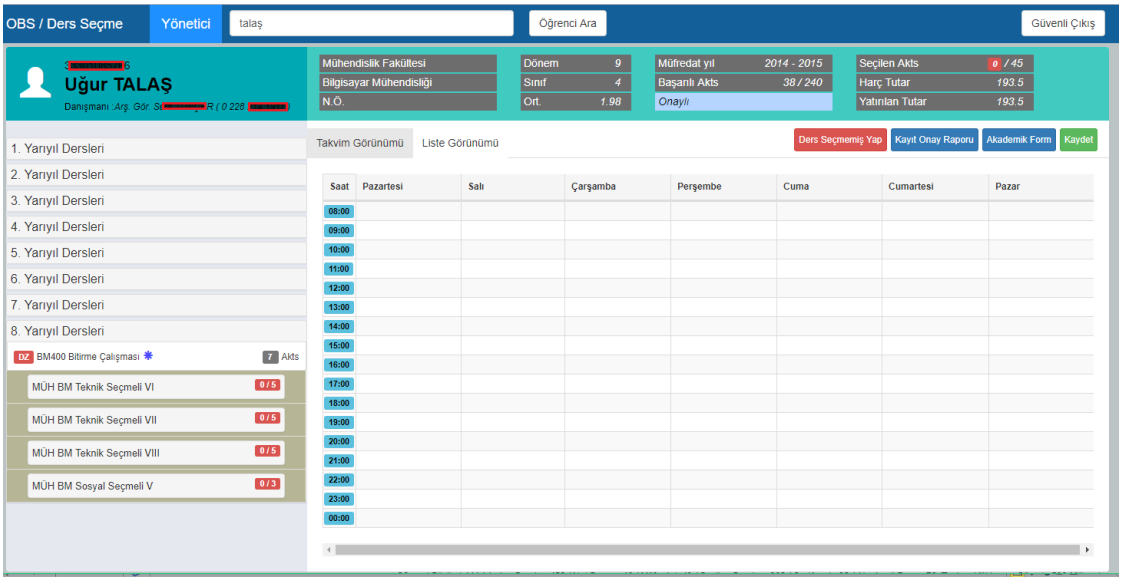


Şekil 4.22. Arama ekranı.

Arama işlemi sonucunda dönen veriler gruplanarak ekrana yansıtılmaktadır. Bu gruplama sonucunda tab menü oluşturulmuştur. Oluşturulan tab menüde tümü, onay bekleyen, onaylı ve ders seçmemiş olarak dört seçenek bulunmaktadır. Danışman buradan işlem yapması gereken öğrencileri süzebilir ve onaylama işlemini yapabilir. Bu tab menü sayesinde özellikle danışmanlar için kullanım kolaylaştırılmıştır.

4.7.3 Anasayfa

Sisteme giriş yapan kullanıcıların tamamı bu sayfaya yönlendirilmektedir. Bu ara yüz danışman ve yöneticide boş gelirken öğrenci için kendi verileriyle dolu olarak gelmektedir.

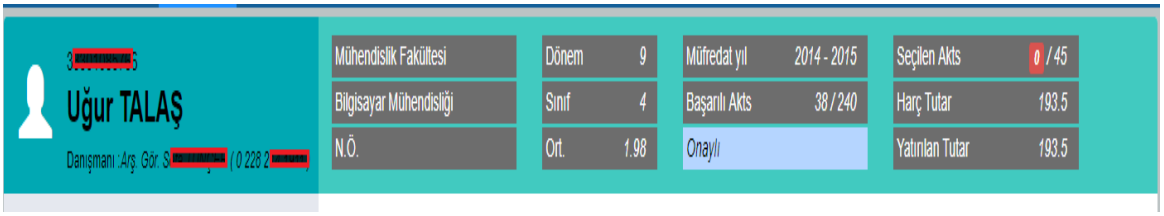


Şekil 4.23. Ana sayfa ekranı.

Şekil 4.23'te görülen ekran, ana sayfa ekranı, öğrenci bilgileri, açılan dersler, seçilen dersler olarak 3 ana bölümde tasarlanmıştır. Ayrıca ders seçmemiş yap, kayıt onay raporu akademik form ve kaydet butonları bulunmaktadır. Bu butonlar rollere göre görünürlüğü değişmektedir. Hangi rolün hangi butona erişebildiği use case diyagramlarında anlatılmıştır.

4.7.4 Öğrenci bilgileri gösterimi

Ana sayfanın üst bölümünde bulunan ve öğrenci bilgilerinin bulunduğu bölümdür. Bu bilgilerin bir kısmı öğrenciyi bir kısmı ise danışmanı ve yöneticiyi bilgilendirmek için getirilmiştir.



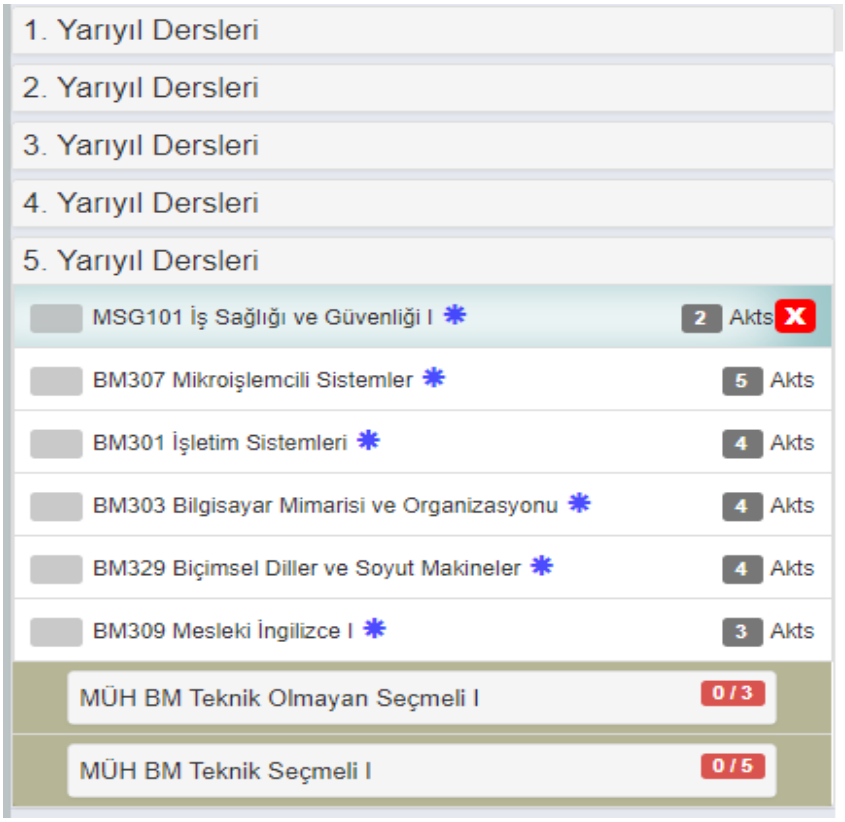
Şekil 4.24. Öğrenci bilgilerinin gösterimi.

Bu bölümde öğrencinin sabit bilgileri bulunduğu gibi sayfada yapılan işlemlerle birlikte değişen verileri de bulunmaktadır. Seçilen AKTS ve onay durum bilgisi yapılan işlemlere göre değişiklik göstermektedir. Harç tutarı verisi ise yaz okulunda seçilen dersin AKTS ve birim ücretiyle hesaplanan harcı bu alana yazılmaktadır. Bu bölüm genel olarak bilgilendirme için planlanmıştır ve herhangi işlem ve buton içermemektedir. Ayrıca danışman seçtiği öğrencinin resmini görmek istediğinde isminin yanında bulunan öğrenci ikonunu tıkladığında öğrencinin resmini bir modal içerisinde görüntüleyebilmektedir. Burada öğrenci resminin direkt olarak sayfaya basılmaması performansın olumsuz etkilenmesini önlemek amacıyla böyle planlanmıştır. Öğrenci resmi görüntüleme örneği şekil 4.25'te görülmektedir.

Şekil 4.25. Öğrenci resmi görüntüleme ekranı.

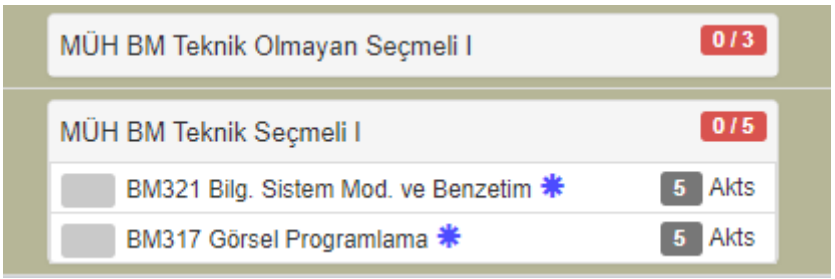
4.7.5 Ders gösterimi ve kontenjan gösterimi

Açılan derslerin listelendiği ve ekranın sol tarafında kalan bölümdür. Bu ara yüzde dersler yarıyıllara göre gruplanıp accordion menü olarak tasarlanmıştır. Bu menüde danışman ve yöneticiler tüm yarıyılları görebilirken öğrenciler güz dönemindeyse güz yarıyıllarını, baharda ise bahar yarıyıllarını görebilmektedirler.



Şekil 4.26. Açılan dersler ve yarı yılların gösterimi.

Her bir yarıyıl içerisinde zorunlu dersler ve seçmeli dersler bulunmaktadır. Zorunlu dersler üstte seçmeli dersler altta listelenmektedir. Zorunlu derslerin varsa notu, kodu adı, zorunlu olma durumu ve AKTS bilgisi gösterilmektedir. Seçilen dersler mavi dolgu ile işaretlenmektedir. Ayrıca seçilen dersin satırına sil butonu eklenmektedir. Kullanıcı isterse seçtiği dersi buradan da kaldırılabilmektedir.

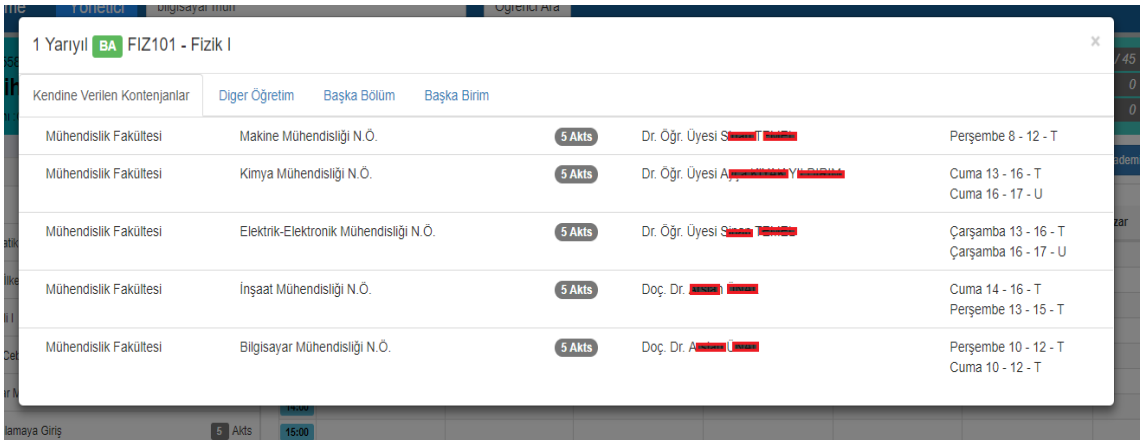


Şekil 4.27. Seçmeli grupların gösterimi.

Seçmeli ders grupları da yarıyıllar gibi acordion menü olarak planlamıştır. Müfredatta olan tüm gruplar bu şekilde listelenmektedir. Altında ders açılmadıysa

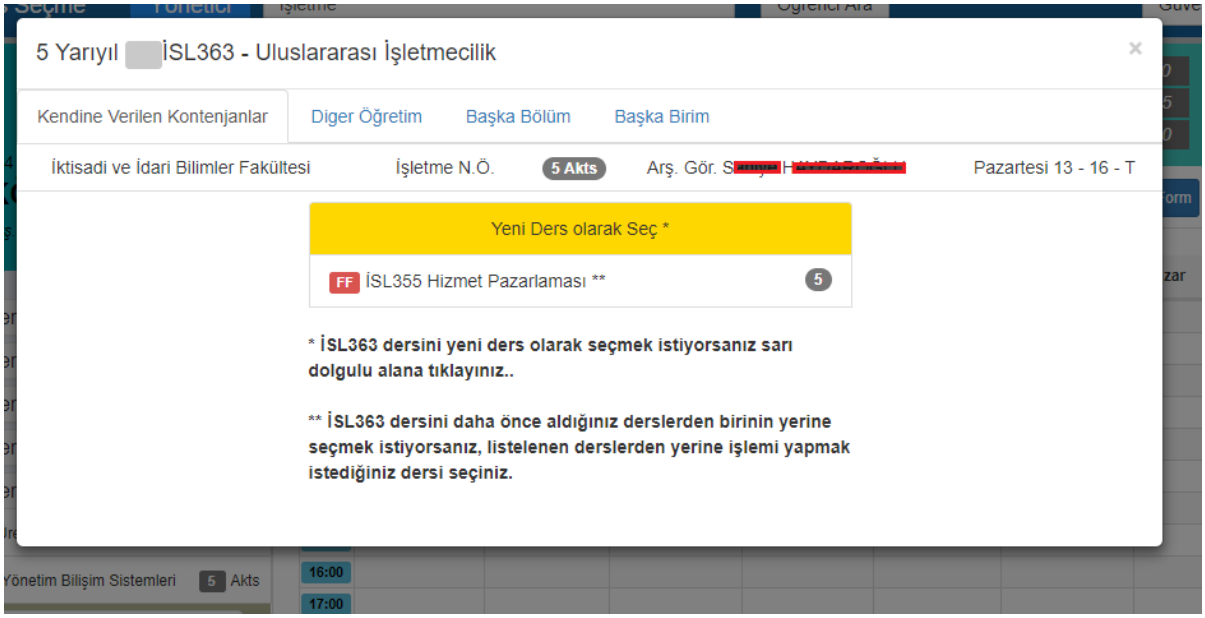
tıklandığında bunun uyarısı öğrenciye gösterilmektedir. Grup altındaki derslerin gösterimi zorunlu derslerdeki olduğu gibi planlanmıştır. Grup adının yanında gösterilen bilgi öğrencinin gruptan ne kadar AKTS başardığı ve ne kadar AKTS başarması gerektiği verisini göstermektedir. Eğer öğrenci grubu başardı ise buradaki bilgi yeşil dolguyla getirilip anlaşılırlığı kolaylaştırılmıştır.

Açılan dersler tıklandığında o derse verilen kontenjanlar bir modal içerisinde listelenmektedir. Öğrenci buradaki kontenjanlardan herhangi birini tıkladığında ders seçme işlemini tetiklemektedir.



Şekil 4.28. Derse verilen kontenjanların gösterimi.

Kontenjanların listelendiği bu modal'da dersin açıldığı birim bölüm, AKTS, dersi veren akademik personel ve dersin program bilgisi gösterilmektedir. Ayrıca içerisinde kendine verilen kontenjanlar, diğer öğretim, başka bölüm, başka birim olmak üzere dört sekmeden oluşan tab menü bulunmaktadır. Bu tab menü kullanıcı rollerine göre farklı çalışmaktadır. Öğrenciler sadece ilk sekme olan kendine verilen kontenjanları görebilmektedirler. Sadece çakışan dersleri bulunduğunda o ders için diğer öğretim sekmesi görünmektedir. Yönetici ve danışmanlar için tab menünün tamamı gösterilmektedir. Seçmeli derslerde ise bu işlemden sonra bir adım daha süreç işletilmektedir.

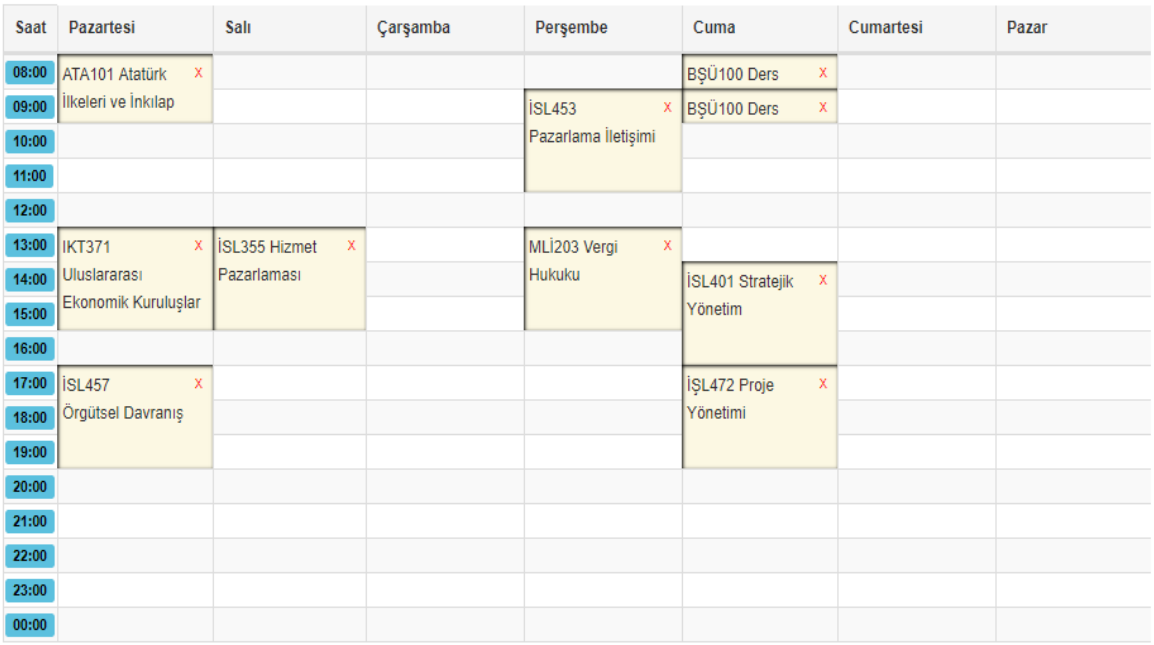


Şekil 4.29. Seçmeli derslerde yerine işlemi.

Seçmeli derslerde kontenjan tıklandıktan sonra kontenjan altında yukarıdaki resimde gösterilen menü açılmaktadır. Bu menüde öğrencinin o grup için aldığı ve üzerinde not olan dersler listelenmektedir. Öğrenci seçtiği yeni dersi bu menüde listelenen herhangi bir dersin yerine seçebilir yani o ders ile yeni dersi intibak yaptırabilir yada en üstteki sarı renkli olan yeni ders olarak seç butonunu tıklayarak intibak yapmadan dersi seçebilmektedir. Öğrenci bu gruptan herhangi bir dersten daha önce AA notu aldıysa o dersi intibak yaptıramaz veya seçtiği kontenjan FF ise yeni intibakını FF bir ders ile yapamaz. Bu kontroller JavaScript ile yapılmaktadır.

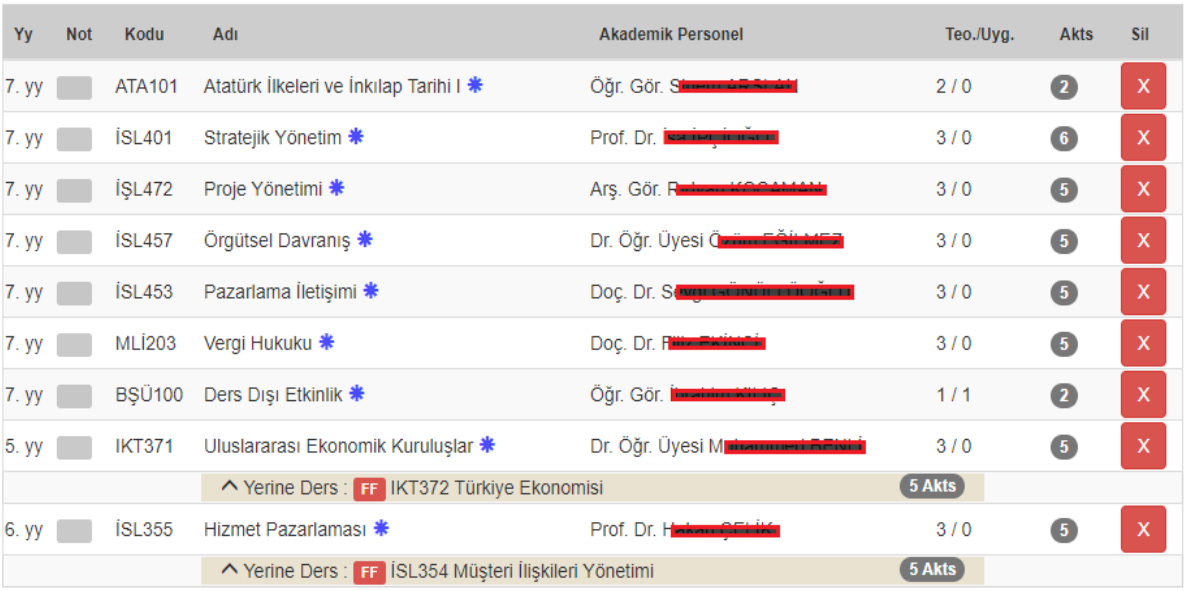
4.7.6 Seçilen derslerin gösterimi

Ara yüzde öğrencilerin seçtiği dersler takvim görünümü ve liste görünümü olarak iki farklı şekilde gösterilmektedir.



Şekil 4.30. Seçilen derslerin takvim görünümü.

Şekil 4.30'da seçilen dersler ders programı şeklinde hangi gün ve saatte işleniyorsa o yerde gösterilmiştir. Dersler 08:00'da başlayıp 23:59'da bittiği için ders olmayan saatler gösterilmemiştir. Derslerin üzerine gelindiğinde ise tooltip olarak derslerin detay bilgileri gösterilmektedir.



Şekil 4.31. Seçilen derslerin liste gösterimi.

Bu gösterim yönteminde seçilen dersler alt alta listelenmiştir. Takvim gösteriminden farklı olarak seçmeli derslerde yapılan yerine işlemi rahatça görülebilmektedir.

Kullanıcılar bu iki gösterim yöntemini tab menüden değiştirerek rahatlıkla kullanabilmektedirler. Ders silme işlemi liste ve takvim görünümündeki silme butonlarından herhangi biriyle yapılabilir.

4.7.7 Raporlar

Ders seçme modülünde akademik form ve kayıt onay raporu olmak üzere iki farklı rapor bulunmaktadır. Bu raporlar eski sistemdeki reporting servisi kullanmayıp C# tarafından aldıkları verilerin JS tarafında görselleştirilmesiyle oluşmaktadır.

4.7.7.1 Akademik form

Akademik form öğrencinin mezuniyet şartını tamamlayabilmeleri sağlamak amacıyla alması gereken dersleri belirlemek için kullanılmaktadır. Seçmeli grupların durumunu incelemek ve mezuniyet durumundaki öğrencilere rehber olması açısından çok önemlidir.

Şekil 4.32. Akademik formun görünümü.

Şekil 4.32'de görülen akademik forma dikkat edildiğinde, form iki bölümden oluşmaktadır. Sol taraf öğrencinin müfredatını yani mezun olabilmesi için tamamlaması gereken ders ve seçmeli ders gruplarını göstermektedir. Sağ taraf ise öğrencinin aldığı veya intibak yaptırdığı başarılı başarısız tüm dersleri içermektedir. Burada kontrolü kolaylaştırmak ve anlaşılırlığı arttırabilmek için satırlar üzerinde renklendirme yapılmıştır. Renklerin ifade ettiği başarı durumları aşağıda listelenmiştir.

Yeşil Renk: Başarılı ders

Kırmızı Renk: Başarısız veya hiç almadığı ders

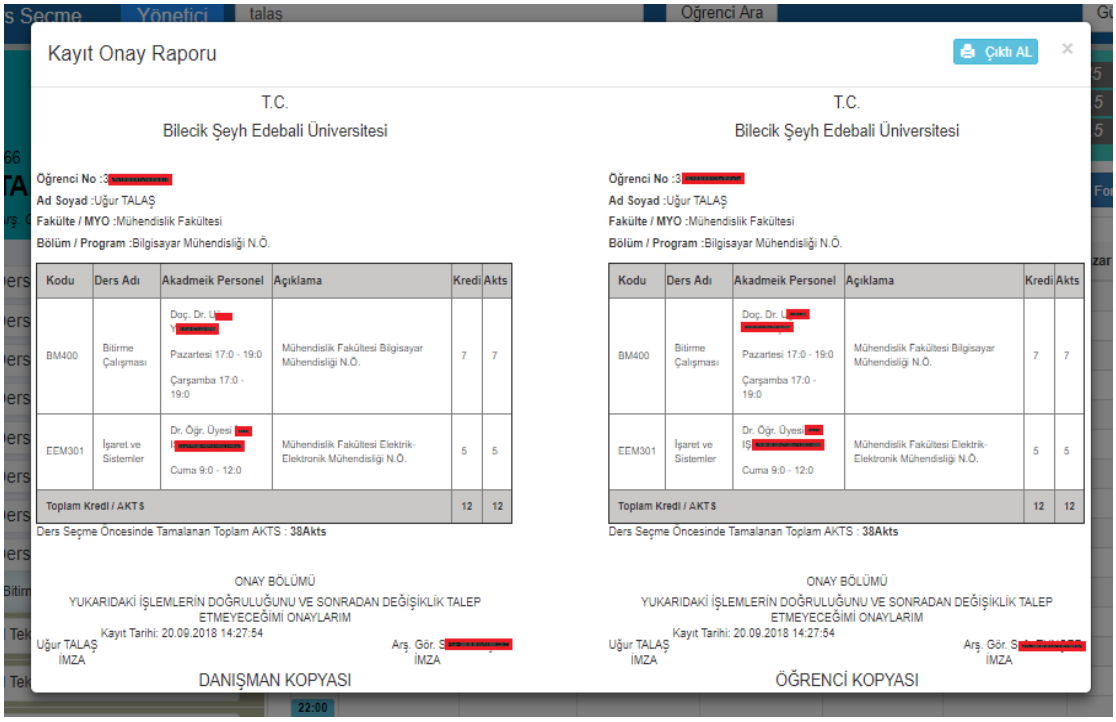
Mavi Renk: Yeni seçilen ders

Sarı Renk: Tamamlanmamış seçmeli grup

Bu renklendirme sayesinde özellikle mezuniyet kontrolü yapan kişi notların detayına bakmadan sadece renklere bakarak hızlıca yorum yapabilmektedir. Bu renklendirme kullanım kolaylığını arttıran bir yenilik olmuştur.

4.7.7.2 ***Kayıt onay raporu***

Bu rapor ders seçme işlemi tamamlandıktan sonra danışman ve öğrenci arasında nihai durumun bu olduğunu kayıt altına almak için yapılan sözleşmedir. Bu rapor iki nüsha olarak alınıp karşılıklı imzalanmaktadır.



Şekil 4.33. Kayıt onay raporu.

Raporda öğrencinin aktif dönemde seçtiği derslerin listesi bulunmaktadır. Ders listesinde dersi veren akademik personel ve kontenjanı nerden seçtiğinin detayı ve varsa intibak bilgileri de getirilmektedir. Rapor C#'ta bulunan controller'dan gelen veri JS ile görselleştirilerek oluşturulmuştur. Raporun sağ üst köşesinde bulunan çıktı al butonu ile kolayca çıktı alınabilmektedir. Bu raporun çıktı alınabilmesi için performans gözetilerek sayfaya yeni bir kütüphane eklenmeyip JavaScript'in kendi metotları ile çıktısı alınması sağlanmıştır.

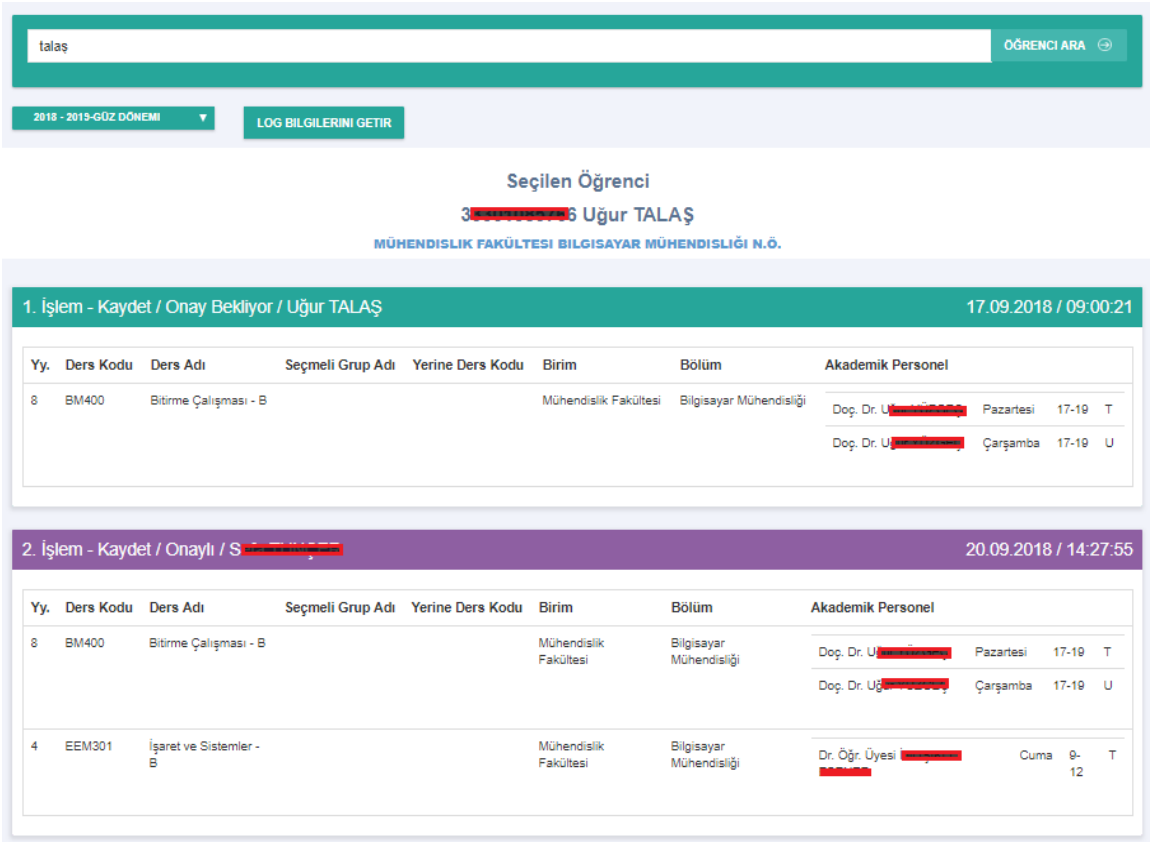
4.8 Log İşlemleri

Literatüre İngilizce adıyla giren Log kaydı, kütük olarak Türkçeye çevrilmektedir. Log; kullanıcıların veya sistemlerin yaptığı hareketleri kayıt altına almak anlamına gelmektedir. Özellikle kritik verilerin tutulduğu sistemlerde log tutmak, hangi işlemi kimin yaptığını tespit edebilmek için oldukça önemlidir. Ayrıca log tutma

işlemi, 5651 sayılı kanunun yönetmeliği ile servis sağlayıcı tarafından zorunlu hale getirilmiştir. Bu proje resmi olarak üniversitenin kayıt yenileme işlemlerinde kullanılacağı için bu kanuna uyulması zorunludur. Bu sebeple ders seçme modülünde log tutma işlemleri gerçekleştirilmiştir.

Log tutma işlemleri birçok farklı yöntemlerle ve farklı seneryolar için gerçekleştirilebilir. Genellikle web uygulamalarında iki farlı tipte log tutulmaktadır. Birincisi tüm işlemlerin kayıt altına alınmasıdır. Bu işlemde önemli, önemsiz bakılmaksızın yapılan tüm işlemler kayıt altına alınır. Ancak bu yöntemde çok fazla veri üretildiği için buradan okuma işlemi yapmak ve anlamlı veriler çıkarmak zorlaşmaktadır. İkinci yöntem ise işlem logu diye ifade edilen sistem sahibi tarafından önemli görülen kritik verilerin kayıt altına alınmasıdır. Bu yöntemde sadece kritik veriler tutulduğu için oluşacak veri kümesi birinci yönteme göre daha küçük olması bu verilerin işlenmesini kolaylaştırmaktadır.

Ders seçme modülünde işlem logu tutulmaya karar verilmiştir. Sorgulama işlemlerinin kayıt altına alınmasına ihtiyaç olmadığına karar verilmiştir. Veri tabanında değişiklik yapılan kaydet, ders seçmemiş yap ve onay işlemleri için log tutulmasına karar verilmiştir. Her bir işlemin bitiminde araya girerek işlem sonrası durumun ne olduğunu kayıt altına almaktadır. Kayıt altına alınan bu verileri OBS sistemi içerisinden izleyebilmek için log izleme sayfası yapılmıştır.



Şekil 4.34. Log izleme ekranı

Log izleme sayfasına OBS'de sadece yöneticiler erişebilmektedir. Bu sayfada öğrencinin adı, soyadı ve T.C. bilgilerinin herhangi biriyle öğrenci aratılıp seçilebilmektedir. Seçilen öğrencinin istenilen dönem için logları izlenebilmektedir. Log'lar sorgulatıldığında ise adım adım her yapılan işlem sırasıyla listelenmektedir. Her adımda öğrencinin üzerindeki dersler, işlem tarihi, işlemi kimin yaptığı ve onay durumunun ne olduğu bilgisi getirilmektedir. Bu sayede bir öğrenci için adım adım kimlerin ne işlem yapıldığı rahatça gözlemlenebilmektedir.

5 KARŞILAŞTIRMALI PERFORMANS ANALİZİ

Bir yazılımın performansı test edilirken, ya sistem çökene kadar yavaş yavaş kullanıcı sayısını arttırarak ya da bir anda birçok kullanıcı talebinde bulunularak, iki farklı şekilde yük testi yapılmaktadır. Bu testler ile sistemin dayanıklılığı ölçülmektedir (Gürbüz, 2010). Literatürdeki yük testleri, tüm sistemin çökmesini sağlayana kadar bir anlayışla yapıldığından, bu tez çalışmasında böyle bir yük testi kullanılmamıştır. Bunun yerine, geliştirilen yeni ders seçme modülü örneğinde performans kaybına yol açan modül içindeki bölümler incelenip, bazı önemli yazılımsal ve donanımsal geliştirmelerin yapıldığı noktalar için hız ölçümleri yapılmıştır. Aynı ölçümler, eski yazılımla için de yapılıp modül için karşılaştırmalı performans analizi elde edilmiştir. Bu sayede çalışmanın hangi noktada ne kadar fayda sağladığı gözlemlenebilir hale gelmiştir.

Ders seçme modülünde en maliyetli işlem öğrenci sisteme bağlandığında ders seçme için ihtiyaç duyulan tüm verilerin sorgulanıp öğrencinin ders seçimine uygun hale getirilme işlemidir. Bu işlemi yeni ve eski sistemde karşılaştırabilmek için mümkün olduğunca bir birinden farklı bölüm ve sınıflardaki öğrencilerden 30 farklı öğrenci seçilerek her biri ayrı ayrı sisteme verilerek sunucuda yani C# tarafında geçirdikleri süreler ölçülüp aritmetik ortalaması alınmıştır. Ölçümler sonucu bir öğrencinin sisteme bağlandığındaki verilerin getirilme hızı çizelge 5.1'de verilmiştir.

Çizelge 5.1. Ders seçme veri hazırlanma süreleri performans tablosu.

Ders Seçme Verilerin Hazırlanma Süresi	
Eski Ders Seçme Modülü	1478 ms
Yeni Ders Seçme Modülü	142 ms
Kazanç	Yaklaşık 10 kat hız kazancı

Kaydetme aktivitesinde öğrencinin seçtiği dersler veri tabanına yazıldığı için arka tarafta çalışan "insert" sorguları veri tabanında "select" sorgusuna göre daha maliyetli olmasından dolayı kaydet işleminde de sunucuda geçirilen süre fazladır. Ayrıca kaydetme işleminde yönetici ve danışmana haiz işlemler ve kontroller yapıldığından kaydetme işleminin performans maliyeti artmaktadır. Bu aktivitede

yapılan düzenlemeler ile elde edilen kazancı ölçebilmek içinde 30 fazlı öğrencide ders seçme işlemi yapılıp kaydet aktivitesinde geçirdiği süreler ölçülerek ortalaması alınıp çizelge 5.2'de verilmiştir.

Çizelge 5.2. Kaydet aktivite tamamlanma süreleri performans tablosu.

Kaydet Aktivitesi Tamamlanma Süresi	
Eski Ders Seçme Modülü	2215 ms
Yeni Ders Seçme Modülü	780 ms
Kazanç	Yaklaşık 3 kat hız kazancı

Ders seçme modülüne girildiğinde sayfa yüklenirken tarayıcı JS, CSS, HTML, Jpeg, Gif v.b. bileşenlerin her birini indirebilmek için ayrı ayrı bağlantı açmaktadır. IIS üzerinde maksimum bağlantı sayıları vardır. IIS maksimum bağlantı sayısı aşıldığında işlemi kuyruğa almaktadır. Bu sebeple yeni yapılan sayfada bu bağlantı sayıları düşürülmüştür. Buradan elde edilen kazancı bir hız performansı olarak nitelendirmemiz doğru olmayacaktır ancak IIS'in sağlıklı çalışması için bir kazanç sağlanmıştır. Eski ve yeni sistemdeki bağlantı sayıları çizelge 5.3'te verilmiştir.

Çizelge 5.3. Tarayıcı bağlantı sayıları tablosu.

Tarayıcı Bağlantı Sayıları	
Eski Ders Seçme Modülü	34
Yeni Ders Seçme Modülü	14
Kazanç	Bağlantı sayısında %60 azalma

Çizelge 5.4. Tarayıcı sayfa yüklenme süresi.

Tarayıcı Ham Sayfa Yüklenme Süresi	
Eski Ders Seçme Modülü	1250 ms
Yeni Ders Seçme Modülü	819 ms
Kazanç	%40'lık hız kazancı

Kodlamada yapılan optimizasyon ve MVC mimarisinin getirdiği düzenleme ile kod satır sayısında düşüş sağlanmıştır. Kod satır sayısındaki azalma backend'te oluşan dll dosyasının sunucu tarafında açılıp işlenmesini hızlandırmıştır. Ayrıca kod tarafının küçülmesi performansa direkt etki etmese de kodun anlaşılırlığı ve sürdürülebilirliğini de arttırmıştır. Kod satır sayıları çizelge 5.5'te verilmiştir.

Çizelge 5.5. Kod satır sayıları tablosu

Kod Satır Sayıları	
Eski Ders Seçme Modülü	7800 satır
Yeni Ders Seçme Modülü	3680 satır
Kazanç	Proje büyüklüğünde %50 azalma

Akademik form öğrencinin hangi dersleri seçeceğine karar verirken rehber niteliğinde olduğundan ders seçme esnasında sıklıkla kullanılmaktadır. Akademik form için de 30 farklı öğrencide ölçümler yapılmıştır. Ölçümlerde gözlemlenen performans kazancı sonucunda OBS içerisindeki akademik kaldırılıp ders seçme için tasarlanan akademik form OBS içerisine entegre edilmiştir. Ölçümlerde elde edilen sonuçlar çizelge 5.6'da verilmiştir.

Çizelge 5.6. Akademik form açılma süreleri performans tablosu.

Akademik Form Açılma Süreleri	
Eski Ders Seçme Modülü	372 ms
Yeni Ders Seçme Modülü	47 ms
Kazanç	Yaklaşık 9 kat hız kazancı

Bu çalışmada geliştirilen yazılım ölçümleri yukarıdaki performans kazanç değerlerini verse de tez çalışmasının asıl amacı ders seçme haftalarında yaşanan hizmet kesintilerinin önüne geçmekti. Yeni ders seçme modülü canlı ortama alınıp 3 ders seçme dönemi geçirildi ve IIS ayakta kalma süresi %100 olduğu gözlemlenmiştir. IIS'in ders seçme haftasında hiç durmaması yapılan çalışmanın B.Ş.E.Ü. ders seçme modülü örneğinde başarılı sonuçlar verdiğini göstermektedir.

6 SONUÇLAR

Bu tez çalışmasında kullanıcı yoğunluğuna bağlı olarak IIS'in gelen trafik yükünü karşılayamayıp durmasıyla ilgili yaşanan sistem kesintilerine çözüm aranmıştır. Bilecik Şeyh Edebali Üniversitesi ders seçme modülü örneği üzerinde sistemin donanımsal sunucu yapısı, IIS yapılandırması ve web uygulaması teknik olarak incelenmiştir. İnceleme sonucunda son kullanıcının yaptığı isteklere sunucunun verdiği cevap süresi hızlandırılıp IIS üzerinde geçen sürenin düşürülmesi gerektiği sonucuna varılmıştır. Sunucuya gelen isteklere verilen cevapların hızlandırılması için ders seçme modülü yeniden geliştirilmiştir. Bu geliştirmeler; yazılımsal teknolojik yenilikler ve uygulamanın çalıştığı sunucu ortamında yapılan iyileştirmelerdir. Yazılımsal teknolojik yenilik olarak MVC, Multi theared ve modüldeki bazı kontroller için istemci taraflı yazılım kullanılmıştır. Sunucu ortamında Load Balance kullanılarak birden fazla sunucuda paralel işlem yapılması sağlanmıştır. Yapılan iyileştirmeler sonucunda;

- Son kullanıcı modüle giriş yaptığında IIS'e açılan bağlantı sayısı %60 oranında azaltıldı.
- Öğrenci ders seçme verilerini getiren sorgu yaklaşık 10 kat hızlandırıldı.
- Ders seçme işlemi sonundaki kaydet işlemi yaklaşık 3 kat hızlandırılmıştır.
- Sayfanın tarayıcıda yüklenme süresi %40 hızlandırılmıştır.
- Kod optimizasyonu ile kod satır sayısı 7800'den 3680'e düşürüldü.

Kod satır sayısındaki azalma sayesinde yazılımsal iyileştirme taleplerinin karşılanması ve hata olması durumunda hataya müdahale süresinde fayda sağlanmıştır. Burada verilen yüzdelik ölçümlerin detayı 5. Bölümdeki performans analizinde verilmiştir. Ayrıca sistemin güncel tasarım kütüphaneleriyle tasarlanması, mobil uyumluluğu arttırmış ve eski sisteme göre daha kullanıcı dostu bir tasarım haline gelmiştir.

Bu çalışmadaki çözüm yolu incelendiğinde web uygulama odaklı iyileştirmeler yapılıp, uygulamanın sorunsuz çalışması sağlanmıştır. Ancak veri tabanında küçük bir tasarımsal değişikliğin haricinde bir iyileştirme yapılmamıştır. İleride sistemin veri tabanı gelen istek yükünü karşılayamadığı bir durumla karşılaşılırsa, veri tabanı da uygulama gibi dağıtık bir mimaride tasarlanabilir. Ayrıca günümüzde kullanımı yaygınlaşmaya başlayan No SQL teknolojilerinden de faydalanılabilir.

Bu çalışmada kullanılan Load Balance yöntemi hem kaynak yönetimi konusunda hem de yükü dağıtma konusunda büyük fayda sağlamıştır. İleride öğrenci sayısındaki artışa bağlı olarak sunucular gelen trafiği karşılayamaz duruma geldiklerinde sisteme yeni sunucu ekleyip balancer üzerinde yönlendirmeyi güncelleyerek kolay bir şekilde problemin çözülmesine zemin hazırlayan bir yapı kurgulanmıştır.

Yapılan çalışma sonucunda yeni geliştirilen sistem devreye alınıp ve 3 ders seçme döneminde gözlemlenmiştir. Sunucuların ayakta kalma sürelerine bakıldığında ders seçme haftası boyunca IIS'ler %100 çalışmıştır. Tez kapsamında hedeflenen durma probleminin önüne geçilmesi başarılı bir şekilde sağlanmıştır. Ayrıca B.Ş.E.Ü ders seçme modülü üzerinde incelenen IIS durma problemi ve bu problem karşısında uygulanan çözüm yöntemi, kullanıcı yoğunluğuna bağlı problem yaşayan tüm web uygulamaları için örnek bir çözüm yolu olarak önerilir.

KAYNAKLAR

AKDEMİR, C., (2012). *JQUERY*. Dikey eksen yayın dağıtım yazılım ve eğitim hizmetleri, İstanbul, 2-11

Aktaş, V., Sevinç, T. (2011). *ASP.NET MVC3 Razor*. Kodlab yayın dağıtım yazılım ve eğitim hizmetleri, İstanbul, 133-175.

Aktaş, V., Sevinç, T. (2013). *Her Yönüyle C# 5.0*. Kodlab yayın dağıtım yazılım ve eğitim hizmetleri, İstanbul, 865-866.

Alan, O.,(2011) *İplik Davranışlarının İlgiye Yönelik Programlama Yaklaşımı Kullanılarak Metrik Tabanlı Analizi*. Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

Algan, S. (2015). *Her yönüyle C# 6.0*. Pusula Yayıncılık, İstanbul, 244.

Balkan, M., (2012). *QUERY*. Pusula yayıncılık. İstanbul, 15.

Balkan, M., (2011). *JAVASCRIPT*. Pusula yayıncılık. İstanbul, 3-5.

Baltalı, S., (2011). *JQUERY*. Kodlab yayın dağıtım yazılım ve eğitim hizmetleri, İstanbul, 181.

Berka, T., Vajteršic, M. (2011). *Fast Information Retrieval In The Open Grid Service Architecture,* Serdica Journal of Computing, 207-236.

Çamoğlu, N. E., (2010). *10 Adımda Yazılım Geliştirme.* Kodlab yayın dağıtım yazılım ve eğitim hizmetleri. İstanbul, 8-9.

Çelik, R., (2013). *A'dan Z'ye QUERY*. Şeçkin yayıncılık. İstanbul, 53.

Çelikbilek, İ., (2013). *JavaScript Programlama*. Kodlab yayın dağıtım yazılım ve eğitim hizmetleri, İstanbul, 1-3.

Dalabasmaz, O.,(2018) *Dağıtık Akan-Veri Katarı İşleme Sistemleri İçin Bir Yük Dengeleme Algoritması*. Yüksek Lisans Tezi, Hacettepe Üniversitesi, Fen Bilimleri Enstitüsü, Ankara.

Gürbüz, A.,(2010). Yazılım Test Mühendisliği. Papatya Yayıncılık. İstanbul,

Kızmaz, V. U. (2014). *ASP.NET MVC 5*. Kodlab yayın dağıtım yazılım ve eğitim hizmetleri, İstanbul, 3-123.

Pekgöz, N., (2001). *Webmaster için JavaScript*. Pusula Yayıncılık, İstanbul, 1-9.

Schildt, H., (2002). *The Complete Referance C#*. The McGraw-Hill Companies, U.S.A, 565-586.

KAYNAKLAR (Devam Ediyor)

Sevinçok, A. V., Ünalır, M. O., Yazılım Mühendisliğinde Performans Yönetimi ve Otomatik Bellek Yönetim Durumunun İncelenmesi. 7th Turkish National Software Engineering Symposium İzmir, Turkey, September, 26, 2013.

Şavklı, N. E., (2014). *Her Yönüyle Entity Framework 5.0 ve Uygulamalar.* Pusula yayıncılık. İstanbul, 200-370.

Taşdelen, A. (2015). *UML ve Dizayn Patternlerier yönüyle C#.* Pusula Yayıncılık, İstanbul, 110-265.

Yakar, C., (2011). *LINQ.* Dikey eksen yayıncılık. İstanbul,5-11.

Yıldırımoğlu, M., (2015). *IIS 8.5*, Kodlab yayın dağıtım yazılım ve eğitim hizmetleri, İstanbul, 9-50.

http://www.bootstrap.com/, (Erişim Tarihi: 13.06.2018).

http://javascript-coder.com/tutorials/re-introduction-to-ajax.phtml, (Erişim Tarihi: 28.11.2018).

http://www.webdevelopmenthelp.net/2013/10/difference-between-asp-net-webform-and-asp-net-mvc.html, (Erişim Tarihi: 21.12.2018).

https://rubygarage.org/blog/technology-stack-for-web-development , (Erişim Tarihi: 11.01.2019).

http://www.ntu.edu.sg/home/ehchua/programming/java/j5e_multithreading.html, (Erişim Tarihi: 20.12.2018).

http://tutorials.jenkov.com/images/software-architecture/load-balancing-1.png , (Erişim Tarihi: 25.11.2018).

ÖZ GEÇMİŞ

Kişisel Bilgiler

Adı Soyadı : Uğur TALAŞ

Doğum Yeri ve Tarihi : Bilecik 19.06.1988

Eğitim Durumu

Lisans Öğrenimi : Düzce Üniversitesi Teknik Eğitim Fakültesi Bilgisayar Eğitimi Bölümü (2006 -2010)
Bilecik Şeyh Edebali Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü / Mühendislik Tamamlama (2017 -….)

Bildiği Yabancı Diller : İngilizce

İş Deneyimi

Stajlar : Bilecik Şeyh Edebali Üniversitesi Bilgi İşlem Daire Başkanlığı

Projeler : Öğrenci Bilgi Sistemi Yazılım Projesi
Üç Artı Bir Firma Yönetim Yazılım Projesi
E-posta Yönetim Sistemi Yazılım Projesi

Çalıştığı Kurumlar : Bilecik Şeyh Edebali Üniversitesi

İletişim

Adres : Bilecik Şeyh Edebali Üni. Bilgi işlem Daire Başkanlığı Yazılım Şube Müdürlüğü Oda No: D13 Merkez/ BİLECİK

E-Posta Adresi : ugur.talas@bilecik.edu.tr

Akademik Çalışmaları TALAŞ, U., CEYHAN ,S., 2018. ANALYSIS AND SOLUTION METHODS FOR SERVICE DISRUPTION IN AUTOMATION SYSTEMS, International Congress on Fundamental and Applied Sciences, 18 – 22 June, Skopje, Macedonia

Tarih:30/04/2019