

BURSA TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ



**MAKİNE ÖĞRENMESİ, ASP.NET MVC VE FLUTTER KULLANARAK
CROSS PLATFORM
BAKIM YÖNETİM SİSTEMİ GELİŞTİRİLMESİ**

LİSANS BITİRME ÇALIŞMASI

Batuhan AVCI

Bilgisayar Mühendisliği Bölümü

TEMMUZ 2022

BURSA TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VEDOĞA BİLİMLERİ FAKÜLTESİ



**MAKİNE ÖĞRENMESİ, ASP.NET MVC VE FLUTTER KULLANARAK
CROSS PLATFORM
BAKIM YÖNETİM SİSTEMİ GELİŞTİRİLMESİ**

LİSANS BİTİRME ÇALIŞMASI

**Batuhan AVCI
18360859039**

Bilgisayar Mühendisliği Bölümü

Danışman: Dr. Öğr. Üyesi Mustafa Özgür CİNGİZ

TEMMUZ 2022

BTÜ, Mühendislik ve Doğa Bilimleri Fakültesi Bilgisayar Mühendisliği Bölümü'nün 18360859039 numaralı öğrencisi Batuhan AVCI, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "Makine Öğrenmesi, ASP.NET MVC Ve Flutter Kullanılarak Cross Platform Bakım Yönetim Sistemi Geliştirilmesi" başlıklı bitirme çalışmasını aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Danışmanı : Dr. Öğr. Üyesi Mustafa Özgür CİNGİZ
Bursa Teknik Üniversitesi

Jüri Üyeleri : Dr. Öğr. Üyesi Erdem YAVUZ
Bursa Teknik Üniversitesi

Dr. Öğr. Üyesi Seçkin YILMAZ
Bursa Teknik Üniversitesi

Savunma Tarihi : 07 Temmuz 2022

BM Bölüm Başkanı : Prof. Dr. Turgay Tugay Bilgin
Bursa Teknik Üniversitesi/...../.....

İNTİHAL BEYANI

Bu bitirme çalışmasında görsel, işitsel ve yazılı biçimde sunulan tüm bilgi ve sonuçların akademik ve etik kurallara uyularak tarafımdan elde edildiğini, bitirme çalışması içinde yer alan ancak bu çalışmaya özgü olmayan tüm sonuç ve bilgileri bitirme çalışmasında kaynak göstererek belgelediğimi, aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim.

Öğrencinin Adı Soyadı: Batuhan AVCI

İmzası :

ÖNSÖZ

Bitirme çalışması projesinde beraber çalıştığım Berk SOĞUKPINAR'a, özellikle makine öğrenmesi konusunda desteklerini esirgemeyen ve her zaman yardım etmeye çalışan Dr. Öğr. Üyesi Mustafa Özgür CİNGİZ'e, Bursa Teknik Üniversitesi'nin İMEP kapsamında anlaşmalı olduğu kurum olan Diniz Adient Oto Donanım firmasında bize mentörlük yapan Hüseyin BABUÇ ve Erdem POLAT'a ve son olarak sadece bitirme çalışmasında değil Bursa Teknik Üniversitesi'nde öğrenim gördüğüm süre boyunca her zaman öğrencilerle ilgili olduğu ve yol göstericiliği için Prof. Dr. Turgay Tugay BİLGİN'e teşekkür ediyorum.

Temmuz 2022

Batuhan Avcı

İÇİNDEKİLER

	<u>Sayfa</u>
ÖNSÖZ	iiiv
İÇİNDEKİLER	v
ÇİZELGE LİSTESİ	vi
ŞEKİL LİSTESİ	vii
ÖZET	ix
SUMMARY	11
1. GİRİŞ.....	13
1.1 Tezin Amacı	13
1.2 Literatür Araştırması	13
1.3 Hipotez	14
2. VERİTABANI TABLOLARI	15
2.1 Tabloların İncelenmesi	15
2.2 Tablolar Arasındaki İlişkiler	23
2.3 Tabloların ASP.NET İle Bağlanması	26
3. UYGULAMANIN İNCELENMESİ	29
3.1 Kullanıcı Giriş	29
3.2 İş İstekleri	31
3.3 Arıza Talepleri	38
3.4 Periyodik Bakımlar	43
3.5 Kritik Stok	46
3.6 Admin Yönetim Paneli	48
4. MAKİNE ÖĞRENMESİ	59
5. SONUÇ	70
5.1 Öneriler	70
KAYNAKLAR	71
ÖZGEÇMİŞ.....	72

ÇİZELGE LİSTESİ

	<u>Sayfa</u>
Çizelge 2.1 : BYS_Users tablosu.....	16
Çizelge 2.2 : BYS_Department tablosu.....	16
Çizelge 2.3 : BYS_Entity tablosu.....	17
Çizelge 2.4 : BYS_Status tablosu.....	17
Çizelge 2.5 : BYS_Breakdown tablosu.....	18
Çizelge 2.6 : BYS_BreakdownMaterialLog tablosu.....	18
Çizelge 2.7 : BYS_BreakdownEmployeeLog tablosu.....	19
Çizelge 2.8 : BYS_JobRequest tablosu.....	19
Çizelge 2.9: BYS_RejectJobRequest tablosu.....	20
Çizelge 2.10 : BYS_Decline tablosu.....	20
Çizelge 2.11 : BYS_Machine tablosu.....	21
Çizelge 2.12 : BYS_MachineType tablosu.....	21
Çizelge 2.13 : BYS_Material tablosu.....	22
Çizelge 2.14 : BYS_PeriodicCheck tablosu.....	22
Çizelge 2.15 : BYS_Frequency tablosu.....	23
Çizelge 2.16 : BYS_Type tablosu.....	23
Çizelge 2.17 : BYS_Situation tablosu.....	23

ŞEKİL LİSTESİ

	<u>Sayfa</u>
Şekil 2.1 : Kullanılan tablolar	15
Şekil 2.2 : SQL Server ile tabloların bağlanması	26
Şekil 2.3 : SQL Server bağlantı türünün seçilmesi.....	27
Şekil 2.4 : İstenilen tabloların eklenmesi.....	28
Şekil 3.1 : Kullanıcıdan gelen bilginin doğrulanması.	29
Şekil 3.2 : Kullanıcıya mail gönderilmesi.	29
Şekil 3.3 : Random string veri üretilmesi.	30
Şekil 3.4 : Giriş sayfasına dinamik olarak veri gönderilmesi.....	30
Şekil 3.5 : Giriş sayfası.....	31
Şekil 3.6 : Kullanıcıya gelen mail.....	31
Şekil 3.7 : İş isteklerinin veritabanından alınması.....	32
Şekil 3.8 : İş isteklerinin görüntülenmesi.	32
Şekil 3.9 : Yeni iş isteği oluşturma.	33
Şekil 3.10 : Yeni iş isteği oluşturma kodu.....	33
Şekil 3.11 : İş isteklerini onaylama ekranı.	34
Şekil 3.12 : İş isteklerini tamamlama ekranı.	34
Şekil 3.13 : Bildirimlerin veritabanından getirilmesi.	35
Şekil 3.14 : İş isteklerinin filtrelenmesi.....	36
Şekil 3.15 : İş isteklerinin Excel formatına çevrilmesi	37
Şekil 3.15 : İş isteklerinin Excel olarak çıktı alınması	37
Şekil 3.17 : Arızaların veritabanından getirilmesi.	38
Şekil 3.18 : Arızaların kullanıcıya gösterilmesi.	38
Şekil 3.19 : Arıza kaydı oluşturulması.	39
Şekil 3.20 : Malzeme isminin girilmesi.....	40
Şekil 3.21 : Sunucuya eş zamanlı istek atılması.	40
Şekil 3.22 : Sunucuda çalışan fonksiyon.	41
Şekil 3.23 : İstemcide dinamik olarak inputların isimlendirilmesi.....	41
Şekil 3.24 : Sunucu tarafında gelen malzemelerin alınması.....	41
Şekil 3.25 : Gelen tüm malzemelerin log olarak tutulması.	42
Şekil 3.26 : Arızaların Excel formatına çevrilmesi.	43
Şekil 3.27 : Arızaların Excel formatına geri döndürülmesi.....	43
Şekil 3.28 : Periyodik bakımların görüntülenme sayfası.....	44
Şekil 3.29 : Yeni periyodik bakım oluşturulması.	44
Şekil 3.30 : Periyodik bakımların kontrol edimesi.	46
Şekil 3.31 : Malzeme stoklarının gösterilmesi.	47
Şekil 3.32 : Stok bilgilerinin veritabanından getirilmesi.	47
Şekil 3.33 : Admin yönetim paneli.	48
Şekil 3.34 : Sunucu tarafında çalışan fonksiyon.....	49
Şekil 3.35 : Veritabanına sorgu atılıp istemciye gönderilmesi.	49
Şekil 3.36 : Tüm kullanıcıların gösterilmesi.....	50

Şekil 3.37 : Kullanıcı bilgilerinin gösterilmesi.....	50
Şekil 3.38 : Kullanıcı düzenleme formu.....	51
Şekil 3.39 : Kullanıcı ekleme formu.....	51
Şekil 3.40 : Tüm makinelerin görüntülenmesi	52
Şekil 3.41 : Makine bilgilerinin gösterilmesi.	52
Şekil 3.42 : Makine bilgilerinin güncellenmesi.	53
Şekil 3.43 : Yeni makine oluşturulması.....	53
Şekil 3.44 : Malzemelerin gösterilmesi.	54
Şekil 3.45 : Aranan malzemelerin veritabanından getirilmesi.....	54
Şekil 3.46 : Malzeme detaylarının gösterilmesi.	55
Şekil 3.47 : Malzeme düzenlemesinin yapılması.	55
Şekil 3.48 : Fabrika bölümlerinin gösterilmesi.	56
Şekil 3.49 : Bölüm bilgilerinin düzenlenmesi.	56
Şekil 3.50 : Tüm periyodik bakımların görüntülenmesi.....	57
Şekil 3.51 : Periyodik bakım bilgilerinin görüntülenmesi.....	57
Şekil 3.52 : Periyodik bakım bilgilerini güncelleme formu.	58
Şekil 4.1 : Tahmin değerlerini kaydetmek için SQL tablosu.	59
Şekil 4.2 : Excel dosyasına atma işleminde değişkenler.	60
Şekil 4.3 : Tek bir Excel dosyasında tüm kayıtların tutulması.	60
Şekil 4.4 : Tüm Makineler İçin Arızalanma Zamanları Farkı'nın Alınması.	61
Şekil 4.5 : CSV formatına aktarım işlemi.....	61
Şekil 4.6 : Jupyter Notebook çalıştırılma komutu.	62
Şekil 4.7 : Kullanılan kütüphaneler.	62
Şekil 4.8 : Arıza Fark Sayılarını CSV Dosyadan Okuma.....	63
Şekil 4.9 : Data Sayısı Az Olan Makinenin Tahmin Değerleri.	64
Şekil 4.10 : Arıza Kaydı 50'den Fazla Olan Makinelerin Kontrolü.....	64
Şekil 4.11 : Arıza Kaydı 50'den Fazla Olan Makinelerin Bozulma Sayıları.	65
Şekil 4.12 : Oluşturulan İki CSV Dosyasının Alınması.	66
Şekil 4.13 : Her Makinenin Tahmin Değerlerinin Dinamik Gerçekleştirilmesi.....	66
Şekil 4.14 : Arıza Tahmini.	67
Şekil 4.15 : Modelin Oluşturulması.....	67
Şekil 4.16 : Eğitim ve Test Skorları.	67
Şekil 4.17 : LSTM Modeli İle Tahmin Sonuçları.....	68
Şekil 4.18 : GRU Modeli İle Tahmin Sonuçları.	68
Şekil 4.19 : SQL Server'da Tahmin Verilerini Kaydetme.	69

MAKİNE ÖĞRENMESİ, ASP.NET MVC VE FLUTTER KULLANARAK CROSS PLATFORM BAKIM YÖNETİM SİSTEMİ GELİŞTİRİLMESİ

ÖZET

Bitirme projesi Bursa Teknik Üniversitesi'nin İMEP kapsamında anlaşmalı olduğu Diniz Adient Oto Donanım firmasında kullanılmak üzere geliştirilmiştir. Proje 4 ana kısımdan oluşmaktadır.

Birinci kısım firma bünyesindeki çalışanların iş istekleri oluşturulabilmesi ve bu iş isteklerinin oluşturan kişinin bağlı olduğu sırasıyla şef,yönetici ve firma bünyesinde bulunan bakım görevlisinin onayı ile birlikte aktarmalı olarak devam edecek şekilde geliştirilmiştir. İş istekleri bildirimleri şef, yönetici veya bakım görevlilerinin bildirimler bölümüne gelmektedir. Gerekli onayları aldıktan sonra bakım görevlisine gelen iş isteği, firmadaki kullanılan malzemeler ve adam/saat gibi bilgilerin olduğu bir form doldurulup onaylanarak kapatılabilir veya reddedilerek geri gönderileilmektedir.

İkinci kısım firma bünyesindeki makinelerin arızalarını takip etmek için oluşturulmuştur. Uygulamayı kullanan kullanıcılar, bir makinenin arızalanması durumunda ilgili makineyi ve makinenin arızasını onarabilecek bakım görevlisini seçerek arıza isteği oluşturabilmektedirler. Arıza bilgileri için tüm arızaların listelendiği bir sayfa bulunmaktadır. Bu sayfa aracılığıyla bakım görevlileri ilgili arızaaya gidererek kullanılan malzeme ve personel bilgisi ile arızayı kapatabilmektedir.

Üçüncü kısım firmada bulunan makinelerin normal arızalar dışında, arıza yapmadan periyodik bakımlarının yapılması gerekiğinde firmanın işini kolaylaştırmak için geliştirilmiştir. Günlük çalışan bir Windows Servisi yardımıyla periyodik bakım oluşturulan makineler, son bakım tarihleri ve ne zaman yapılması gibi bilgiler işlenerek, periyodik bakımı gelen makinelerin bilgileri son kullanıcıya gösterilmektedir.

Dördüncü kısım ise firmada bulunan malzemelerle ilgili kritik stok bilgisinin son kullanıcıya gösterildiği bir ekrandır. İş isteklerinde ve arızalarda kullanılan malzemeler, firma bünyesindeki malzemelerin stoklarından düşülmekte ve kullanılan her bir malzemenin log bilgileri tutulmaktadır.

Anlatılan 4 kısım haricinde sadece admin özelliğine sahip olan kullanıcıların görebileceği bir 'Admin Yönetim Paneli' bulunmaktadır. Bu panelde admin kullanıcılar, tüm kullanıcılar,makineler,malzemeler,firma bölümleri, bölümlerin şefleri ve yöneticileri hakkında ekleme,çıkarma,düzenleme ve silme gibi işlemler yapabilmektedir. Bu sayede sisteme aksaklılık çıkması durumunda panelden düzeltilmesi ve sistemin sürekliliği amaçlanmıştır.

Projenin makine öğrenmesi kısmında ise, firma bünyesindeki 2016 yılından beri kaydedilen makine arızalanması verileri kullanılmıştır. Python programlama dili ile Tensorflow/Keras kütüphaneleri kullanılarak makinelerin bir sonraki arızalanma zamanı, zaman serileri kullanarak tahmin edilmeye çalışılmıştır.

Projenin web kısmını ASP.NET MVC Framework, C# ve JavaScript programlama dilleri ile MSSQL Server veritabanı kullanarak geliştirilmiştir. Mobil kısmı ise Dart/Flutter ve .NET Core kullanılarak Berk Soğukpinar tarafından geliştirilmiştir. Makine öğrenmesi kısmında ise Python programlama dili ve Tensorflow/Keras kütüphaneleri kullanılarak ortak çalışma yapılmıştır.

Anahtar kelimeler: Tensorflow, Makine Öğrenmesi, ASP.NET, Flutter, İş İsteği, Makine Arızası

**DEVELOPMENT OF CROSS PLATFORM MAINTENANCE
MANAGEMENT SYSTEM USING MACHINE LEARNING, ASP.NET MVC
AND FLUTTER**

SUMMARY

The graduation project has been developed to be used in Diniz Adient Auto Hardware company, which Bursa Technical University has a contract with within the scope of IMEP. The project consists of 4 main parts.

The first part has been developed in such a way that the employees within the company can create job requests and these job requests will continue with the approval of the chief, the manager and the maintenance worker within the company, respectively. Job request notifications come to the notifications section of the chief, manager or maintenance workers. After obtaining the necessary approvals, the job request received by the maintenance officer can be closed by filling out a form containing information such as the materials used in the company and man/hour, or it can be rejected and sent back.

The second part was created to follow the breakdowns of the machines within the company. Users using the application can create a breakdown request by selecting the relevant machine and the maintenance person who can repair the breakdown of the machine in case of a breakdowns of a machine. For fault information, there is a page listing all faults. Through this page, maintenance personnel can go to the relevant breakdown and close the breakdown with the material and personnel information used.

The third part has been developed to facilitate the work of the company when periodic maintenance of the machines in the company is required, except for normal breakdowns, without breakdowns. With the help of a Windows Service running every day, information such as the machines for which periodic maintenance is created, the last maintenance dates and when it should be done are processed, and the information of the machines whose periodic maintenance is received is displayed to the end user.

The fourth part is a screen where the critical stock information about the materials in the company is displayed to the end user. The materials used in work requests and breakdowns are deducted from the stocks of the materials within the company and the log information of each material used is kept.

Except for the 4 sections described, there is an 'Admin Management Panel' that only users with the admin feature can see. In this panel, admin users can perform operations such as adding, removing, editing and deleting about all users, machines, materials, company departments, chiefs and managers of departments. In this way, in case of breakdown in the system, it is aimed to fix it from the panel and to ensure the continuity of the system.

In the machine learning part of the project, the machine failure data recorded in the company since 2016 was used. Using the Python programming language and Tensorflow/Keras libraries, the next failure time of the machines was tried to be estimated using time series.

I have developed the web part of the project using ASP.NET MVC Framework, C# and JavaScript programming languages and MSSQL Server database. The mobile part was developed by Berk Soğukpınar using Dart/Flutter and .NET Core. In the machine learning part, a collaborative work was carried out using the Python programming language and Tensorflow / Keras libraries.

Keywords: Tensorflow, Makine Öğrenmesi, ASP.NET, Flutter, Job Request, Machine Breakdown

1. GİRİŞ

ASP.NET Microsoft tarafından geliştirilmiş olan açık kaynaklı Web uygulama geliştirme teknolojisidir. Kullanan kişilere .NET çatısı altında büyük kolaylık sağlamaktadır. Diniz Adient Oto Donanım bünyesinde sadece web olarak değil Windows uygulama olarak da C# programlama dili ve ASP.NET Framework’ü kullanılmaktadır. Firma bünyesinde çalışan Bakım Ticket Uygulaması, Windows uygulaması olarak çalışmaktadır. Ayrıca firmanın 4 farklı lokasyonda fabrikası bulunmaktadır. Farklı lokasyonlardan şirketin bir sunucusuna bağlanma işlemleri ciddi bir bağlantı yavaşlığına sebep olmaktadır. Bunun için ASP.NET MVC kullanılarak Bakım Yönetim Sistemi’ni web uygulaması olarak geliştirilmesine karar verilmiştir.

Uygulamada ek olarak Makine Öğrenmesi kullanılarak bakım bölümü tarafından kullanılan makinelerin arızalanma zamanlarını Python'un Keras kütüphanesi ile tahmin etme işlemi yapılmıştır.

1.1 Tezin Amacı

Diniz Adient Oto Donanım firmasında hali hazırda Bakım Ticket Uygulaması olarak bir uygulama aktif olarak çalışmaktadır. Firmanın bilgi işlem müdürü Sedat Alan tarafından istenmesi sonucunda uygulamanın geliştirilmesi kararlaştırılmıştır. Uygulama hem Web hem Mobil olarak iki platformda geliştirilecektir. Diniz Adient bünyesinde Bursa, Gönen, Gebze ve Gölcük fabrikalarında kullanılacak şekilde geliştirilmesi amaçlanmıştır. Uygulamanın geliştirilmesinde makine öğrenmesi kullanarak da arızalanmış makinelerin tarihlerini Keras kütüphanesi ile kullanarak sonraki arıza zamanları tahmin edilerek makinenin bozulma zamanından önce müdahale edilerek herhangi bir üretimin yavaşlamasını veya durmasını engellemek amaçlanmıştır.

1.2 Literatür Araştırması

Diniz Adient Oto Donanım firmasında halihazırda kullanılan Bakım Ticket Uygulaması Yazılım Ekibi Takım Lideri Erdem Polat tarafından C# dili ile Windows Forms uygulaması olarak geliştirilmiştir. Uygulama ile ilgili gözlemler sonucunda eksikler tespit edilmiştir. Bunun sonucunda Bakım Yönetim Sistemi uygulamasının Web ve Mobil uygulama olarak kullanılmasının uygun olduğu kanısına varılmıştır.

- Eski Bakım Ticket uygulaması Windows Forms uygulaması olarak geliştirilmiş olduğundan UI/UX Design açısından eksik kaldığı tespit edilmiştir.
- Eski Bakım Ticket uygulamasının Diniz Adient firmasına bağlı Gönen ve Gebzedeki firmalarda kullanılması ancak her firmada ayrı veritabanı olmasının veritabanında gereksiz şışirme yaratacağı tespit edilmiştir.
- Eski Bakım Ticket uygulamasında iş isteklerinin onaylanması kısmında hiyerarşije göre Bakım personeli işi tamamladıktan sonra Bölüm Şefi ve Bölüm Müdürü'nün onayına gidecek şekilde ayarlanmış olup bu sistemin yanlış olduğu kullanıcılar tarafından dönüt alınarak saptanmıştır.

Bu eksiklerinin giderilmesi adına uygulamanın yeniden geliştirilmesi kararlaştırılmış olup eksiklere karşılık yeni Bakım Yönetim Sistemi uygulamasında;

- Mobil ve Web tarafında geliştirilmesi ile UI/UX Design açısından daha iyi bir görüntü sağlanması amaçlanarak kullanıcıların uygulamayı kullanmasının arttırılması sağlanmak istenmiştir.
- Tek bir veritabanında 4 ayrı fabrikanın verileri tutularak her fabrikanın veritabanında ayrı ayrı aynı tabloların bulunması engellenmiştir.
- İş isteği onaylama kısmında Bölüm Şefi, Bölüm Müdür'ü o işin yapılmasını onayladıktan sonra Bakım personeli tarafından gerçekleştirilmesi sağlanarak Bölüm Şefi veya Bölüm Müdürü tarafından onaylanmamış bir iş isteğinın Bakım personeli tarafından yapılmamasını sağlanmıştır.

1.3 Hipotez

Diniz Adient Oto Donanım firmasında geliştirme sürecinde web uygulama tarafından ASP.NET framework kullanılmıştır. Bakım Yönetim Sistemi uygulamasının web platformda geliştirilmesi ile farklı fabrikalardan bağlantı esnasındaki yavaşlığın azaltılması amaçlanmıştır. Makine öğrenmesi ise uygulamanın geliştirilme sürecinde kullanılması makinelerin bozulmadan önce bakımlarının yapılip üretimin durmasını veya aksamasını engellemek amacıyla kullanılmıştır.

2. VERİTABANI TABLOLARI

Projede 17 adet tablo kullanılmıştır. Tablolar öncelikle MSSQL Server'da oluşturulmuş olup daha sonra .NET ile Entity Framework kullanılarak DatabaseFirst yaklaşımı ile bağlantı yapılmacaktır. Kullanılan tablolar aşağıdaki görselde gösterilmektedir (Şekil 2.1). Tablolar arasında birebir ve bireçok olmak üzere birden çok bağlantı bulunmaktadır.

- + dbo.BYS_Breakdown
- + dbo.BYS_BreakdownEmployeeLog
- + dbo.BYS_BreakdownMaterialLog
- + dbo.BYS_Decline
- + dbo.BYS_Department
- + dbo.BYS_Entity
- + dbo.BYS_Frequency
- + dbo.BYS_JobRequest
- + dbo.BYS_Machine
- + dbo.BYS_MachineType
- + dbo.BYS_Material
- + dbo.BYS_PeriodicCheck
- + dbo.BYS_RejectJobRequest
- + dbo.BYS_Situation
- + dbo.BYS_Status
- + dbo.BYS_Type
- + dbo.BYS_Users

Şekil 2.1 : Kullanılan tablolar

2.1 Tabloların İncelemesi

Uygulamada kayıtlı olan her bir kullanıcının bilgerini tutabilmek için BYS_Users tablosu oluşturulmuştur. Bu tabloda kullanıcıların isim, soyisim, mail, kullanıcı kodu gibi kişisel bilgilerinin yanında bağlı olduğu bölümün id bilgisi de tutulmaktadır. Tabloda primary key olarak UserId kullanılmıştır. Kullanıcılar giriş yaparken firma bünyesinde kullandıkları kullanıcı adını girecekler, kullanıcı adı doğru ise maillerine bir doğrulama kodu gönderilecek ve bu kodla beraber sisteme giriş yapabileceklerdir. Kullanıcıların doğrulama kodu 'VerificationCode' adındaki sütunda tutulmaktadır. İlgili tablo çizelgede gösterilmiştir (Çizelge 2.1).

Çizelge 2.1 : BYS_Users tablosu.

Sütun İsmi	Veri Tipi
UserId	Int
FirstName	Nvarchar(MAX)
SecondName	Nvarchar(MAX)
Email	Nvarchar(MAX)
IsAdmin	Bit
IsActive	Bit
DepartmentId	Int
PhoneNumber	Nvarchar(MAX)
UserCode	Nvarchar(MAX)
Token	Nvarchar(MAX)
EntityId	Int
VerificationCode	Nvarchar(MAX)
StatusId	Int
CreatedBy	Int
CreatedAt	Datetime

Kullanıcıların bağlı oldukları bölümleri tutabilmek için BYS_Department tablosu oluşturulmuştur. Bu tabloda bölümlerin ismi, bölüm şefinin kullanıcı id'si, bölüm yöneticisinin kullanıcı id'si ve firma bünyesinde bulunan 4 adet fabrikanın hangisinde bulunduğu belirtmek için EntityId adındaki sütun bulunmaktadır. Tabloda primary key olarak DepartmentId kullanılmıştır. EntityId sütunu tüm tablolarda fabrika bilgisini göstermektedir. Aşağıdaki çizelgelde BYS_Department tablosu gösterilmektedir (Çizelge 2.2).

Çizelge 2.2 : BYS_Department tablosu.

Sütun İsmi	Veri Tipi
DepartmentId	Int
DepartmentName	Nvarchar(MAX)
DepartmentChiefId	Int
DepartmentManagerId	Int
EntityId	Int

Kullanıcıların, bölümlerin ve daha sonra bahsedilecek olan malzeme, makine, iş istekleri gibi verilerin hangi fabrikada bulunduğu belirtmek için BYS_Entity tablosu kullanılmaktadır. BYS_Entity tablosunda sadece EntityId ve EntityName bulunmaktadır. Primary key olarak EntityId kullanılmıştır. Aşağıdaki çizelgede BYS_Entity tablosu gösterilmiştir (Çizelge 2.3).

Çizelge 2.3 : BYS_Entity tablosu.

Sütun İsmi	Veri Tipi
EntityId	Int
EntityName	Nvarchar(10)

Uygulamadaki her kullanıcı 4 adet statüden birine sahip olmalıdır. Bunlar sırasıyla normal kullanıcı, şef, yönetici veya bakım görevlisidir. Her kullanıcıkda statü bilgisi id şeklinde tutulmakta ve gerekiğinde BYS_Status tablosundan çekilmektedir. Tablolarda verileri primary key'ler kullanılarak birbirlerine bağlayıp, gerekiğinde ilgili id kullanılarak bilgiyi farklı tablodan edinmek hem verilerin kaplayacağı yer açısından hemde hız açısından daha efektif olduğu için böyle bir yol izlenmiştir. Tabloda primary key olarak StatusId kullanılmış olup ilgili tablo aşağıdaki çizelgede gösterilmiştir (Çizelge 2.4).

Çizelge 2.4 : BYS_Status tablosu.

Sütun İsmi	Veri Tipi
StatusId	Int
Status	Nvarchar(MAX)

Kullanıcıların açtığı arızaları tutabilmek için BYS_Breakdown tablosu oluşturulmuştur. Bu tabloda arızayla ilgili konu,açıklama,başlangıç zamanı,bitiş zamanı gibi bilgiler tutulmaktadır. Arıza yapan makinenin bilgisini elde edebilmek için MachineId adında bir sütun, arızaya bir dosya eklenecek ise eklenen dosyanın ismini kaydetmek için BreakdownImage ve iş istekleri ile arızaların aciliyetini belli etmek BYS_Situation tablosuna bağlı olan SituationId gibi sütunlar tabloda bulunmaktadır. Ayrıca arızanın hangi bölümde ortaya çıktığına dair BYS_Department tablosuna bağlı olan DepartmentId, arızayı kimin onaracağına dair bilgiyi içeren, BYS_Users tablosuna bağlı olan BreakdownResponsibleId, bakım görevlisi arızayı tamamlandığında yapacağı açıklama ile ilgili BreakdownDoneDescription, arızada oluşan hurda sayısı CountOfScrap, arızanın hangi tipe girdiği (Mekanik,Elektrik,Yapı) ile ilgili bilgiyi tutan TypeId ve son olarak arızanın red edilip edilmediğine dair bilgiyi tutan IsDeclined adında sütunlar bulunmaktadır (Çizelge 2.5).

Çizelge 2.5 : BYS_Breakdown tablosu.

Sütun İsmi	Veri Tipi
BreakdownId	Int
Subject	Nvarchar(MAX)
Description	Nvarchar(MAX)
BreakdownRequesterId	Int
CreatedAt	Datetime
DepartmentId	Int
MachineId	Int
StartDate	Datetime
DueDate	Datetime
TypeId	Int
BreakdownResponsibleId	Int
CountOfScrap	Int
EntityId	Int
IsActive	Bit
SituationId	Int
BreakdownDoneDescription	Nvarchar(MAX)
IsDeclined	Bit
BreakdownImage	Nvarchar(MAX)

Arızaları ve iş isteklerini tamamlarken farklı malzemelerden farklı miktarlarda kullanılabilmektedir. Aynı zamanda hangi bakım personelinin kaç saat çalıştığı bilgisi de log olarak veritabanında tutulmaktadır. Kullanılan malzemelerin hangi iş isteği ve arızada, kaç tane ve hangi fabrikada kullanıldığına dair bilgileri tutmak için BYS_BreakdownMaterialLog tablosu oluşturulmuştur. Tabloda primary key olarak BreakdownMaterialLogId kullanılmıştır. Tablodaki BreakdownOrJobRequestId sütunu, iş isteğin'in veya arızanın id'sini tutmakta olup, IsJobRequest sütunu ise kullanılan malzemenin iş isteğinde mi yoksa arıza da mı kullanıldığıının anlaşılmasılığını sağlamaktadır. İlgili tablo aşağıdaki çizelgede gösterilmektedir (Çizelge 2.6).

Çizelge 2.6 : BYS_BreakdownMaterialLog tablosu.

Sütun İsmi	Veri Tipi
BreakdownMaterialLogId	Int
BreakdownOrJobRequestId	Int
UsedMaterialId	Int
UsedMaterialCount	Int
IsJobRequest	Bit

İş isteklerinde ve arızalarda hangi personelin kaç saat görev yaptığı bilgisinin BYS_BreakdownMaterialLog tablosundaki mantıkla BYS_BreakdownEmployeeLog tablosunda tutulmaktadır. Aşağıdaki çizelgede ilgili tablo gösterilmektedir (Çizelge 2.7).

Çizelge 2.7 : BYS_BreakdownEmployeeLog tablosu.

Sütun İsmi	Veri Tipi
BreakdownEmployeeLogId	Int
BreakdownOrJobRequestId	Int
UsedEmployeeId	Int
UsedEmployeeHour	Int
IsJobRequest	Bit

Kullanıcıların oluşturdukları iş isteklerini tutabilmek için BYS_JobRequest tablosu kullanılmaktadır. Tabloda primary key olarak JobRequestId kullanılmıştır. İş isteğin konusu, açıklaması, oluşturan kişi, şef, yönetici, bakım görevlisi, bulunduğu bölüm, reddedildi ise red tablosundaki id bilgisi gibi bilgiler bu tabloda tutulmaktadır. Aşağıdaki çizelgede ilgili tablo gösterilmektedir (Çizelge 2.8).

Çizelge 2.8 : BYS_JobRequest tablosu.

Sütun İsmi	Veri Tipi
JobRequestId	Int
TypeId	Int
Subject	Nvarchar(MAX)
Description	Nvarchar(MAX)
RequesterId	Int
JobDescription	Nvarchar(MAX)
StartDate	Datetime
DueDate	Datetime
HourlyRate	Float
LaborHour	Float
LaborExpense	Float
MaterialExpense	Float
SituationId	Int
ApprovalChiefId	Int
ApprovalChiefTime	Datetime
ApprovalManagerId	Int
ApprovalManagerTime	Datetime
ApprovalMaintenanceId	Int
ApprovalMaintenanceTime	Datetime
EntityId	Int
RequestDate	Datetime
IsActive	Bit
DepartmentId	Int
ApprovalRequesterTime	Datetime
JobRequestFile	Nvarchar(MAX)
RejectId	Int

İş istekleri red edildikleri zaman, neden red edildiğinin ve kimin red ettiğine dair bilgileri tutmak için BYS_RejectJobRequest tablosu kullanılmaktadır. Tabloda

primary key olarak RejectId kullanılmıştır. İlgili tablo aşağıdaki çizelgede gösterilmiştir (Çizelge 2.9).

Çizelge 2.9 : BYS_RejectJobRequest tablosu.

Sütun İsmi	Veri Tipi
RejectId	Int
RejectDescription	Nvarchar(MAX)
JobRequestId	Int
<u>RejectUserId</u>	Int

Arızaların bakım görevlileri tarafından red edilmeleri ise BYS_Decline tablosu tarafından tutulmaktadır. Tabloda primary key olarak DeclineId kullanılmış olup, red edilmenin açıklaması, iş isteklerinde de kullanılabilirlik açısından iş isteği olup olmadığı ve fabrika id'si gibi bilgiler tabloda tutulmaktadır. İlgili tablo aşağıdaki çizelgede gösterilmektedir (Çizelge 2.10).

Çizelge 2.10 : BYS_Decline tablosu.

Sütun İsmi	Veri Tipi
DeclineId	Int
DeclineDescription	Nvarchar(MAX)
IsJobRequest	Bit
JobRequestOrBreakdownId	Int
<u>EntityId</u>	Int

Firma bünyesindeki makinelerin bilgilerinin tutulması için BYS_Machine tablosu oluşturulmuştur. Tabloda primary key olarak MachineId kullanılmıştır. Tabloda makinenin ismi, numarası, oluşturulma tarihi, bölümü, periyodik bakımının zamanı ve kalan gün sayısı bilgileri bulunmaktadır. Makinelerin belirli aralıklarla periyodik bakıma girmesi gerekebilir. Bu sebeple sunucuda her gün bir kere çalışacak olan bir Windows konsol uygulaması yazılmıştır. Bu uygulama her gün, oluşturulan periyodik bakımlara göre makineleri kontrol etmektedir. Son bakım yapılan tarihlerini, periyodik bakım oluşturulurken girilen kaç günde bir bakımının yapılması gereğine dair bilgiler ile beraber, periyodik bakım yapılması gereken makineleri tespit eder ve BYS_Machine tablosundaki PeriodDayLeft kısmına kaç gün kaldığı ile ilgili gerekli bilgiyi yazmaktadır. Periyodik bakım oluşturma işlemleri daha sonra anlatılacak olup BYS_Machine tablosu aşağıdaki çizelge gösterilmektedir (Çizelge 2.11).

Çizelge 2.11 : BYS_Machine tablosu.

Sütun İsmi	Veri Tipi
MachineId	Int
MachineName	Nvarchar(MAX)
MachineNumber	Nvarchar(MAX)
CreatedById	Int
CreatedAt	Datetime
DepartmentId	Int
MachineTypeId	Int
MachinePeriodMonth	Int
IsActive	Bit
EntityId	Int
IsPeriodicTimeCame	Bit
PeriodDayLeft	Int

Makinelerin tiplerini farklı bir tabloda tutmak amacıyla BYS_MachineType tablosu oluşturulmuştur. Makine tablosundaki Id ile bağlanmak için bir adet MachineTypeId sütununa ve MachineTypeDescription adı verilen makine tipinin açıklanacağı sütuna sahiptir. İlgili tablo aşağıdaki çizelgede gösterilmektedir (Çizelge 2.12).

Çizelge 2.12 : BYS_MachineType tablosu.

Sütun İsmi	Veri Tipi
MachineTypeId	Int
MachineTypeDescription	Nvarchar(MAX)

Firma bünyesindeki malzemelerin listesini tutmak için BYS_Material tablosu kullanılmaktadır. Bu tabloda malzemenin seri numarası, referansı, oluşturulma zamanı, ismi, bulunan stok ve kritik stok gibi bilgiler tutulmaktadır. Kritik stok miktarı, bulunan stok miktarının altına indiği zaman uygulama tarafından uyarı gelmektedir. İş istekleri bitirilme esnasında veya makine arızalarının giderildiği esnada ilgili bakım görevlileri kullandıkları malzemeleri ve kaç adet kullandığı gibi bilgileri girerek firmانın stoğunda bulunan malzemeden kullanılan miktar kadar malzeme eksilmektedir. Firma bünyesinde yaklaşık 60 bin adet malzeme bulunduğuundan dolayı bakımci görevlilerin stok seçme işlemleri için dinamik bir arama sistemi web uygulamasına eklenmiş olup ilerleyen kısımlarda anlatılacaktır. İlgili tablo aşağıdaki çizelgede gösterilmektedir (Çizelge 2.13).

Çizelge 2.13 : BYS_Material tablosu.

Sütun İsmi	Veri Tipi
MachineId	Int
MachineName	Nvarchar(MAX)
MachineNumber	Nvarchar(MAX)
CreatedById	Int
CreatedAt	Datetime
DepartmentId	Int
MachineTypeId	Int
MachinePeriodMonth	Int
IsActive	Bit
EntityId	Int
IsPeriodicTimeCame	Bit
PeriodDayLeft	Int

Kullanıcıların, makine bakımları için oluşturdukları periyodik bakımlar BYS_PeriodicCheck tablosunda tutulmaktadır. Periyodik bakımla ilgili makine, son bakım tarihi, bildirim gönderilme sıklığı, bakım sıklığı, sorumlu personel gibi bilgiler bu tabloda tutulmaktadır (Çizelge 2.14).

Çizelge 2.14 : BYS_PeriodicCheck tablosu.

Sütun İsmi	Veri Tipi
PeriodicCheckId	Int
MachineId	Int
ExpirationDate	Datetime
FrequencyId	Int
FrequencyTime	Int
LastCheck	Datetime
CompanyName	Nvarchar(MAX)
CompanyMail	Nvarchar(MAX)
NotificationFrequencyDay	Int
ResponsibleEmployeeId	Int
Description	Nvarchar(MAX)
IsActive	Bit
EntityId	Int
CreatedAt	Datetime
CreatedById	Int

Periyodik bakım tablosunda bakımın ne zaman yapılacağı ‘FrequencyId’ ve ‘FrequencyTime’ adındaki iki sütun ile belli olmaktadır. FrequencyTime bize bir Integer tipinde sayı vermektedir. FrequencyId ise BYS_Frequency tablosuna bağlanmış olup tablodaki ‘Gün’, ‘Ay’ veya ‘Yıl’ verilerinden bir tanesini temsil etmektedir. Kullanıcıdan gelen verilere göre periyodik bakımın hangi aralıklarla

yapılacağı belirlenmektedir. Aşağıdaki çizelgede BYS_Frequency tablosu gösterilmektedir (Çizelge 2.15).

Çizelge 2.15 : BYS_Frequency tablosu.

Sütun İsmi	Veri Tipi
FrequencyId	Int
FrequencyName	Nvarchar(4)

Kullanıcılar iş isteklerinde ve arızalarda hangi tip olduklarının (Mekanik,Elektrik,Yapı) id bilgilerinin tutulması için BYS_Type tablosu kullanılmaktadır. Tabloda primary key olarak TypeId kullanılmıştır. Ayrıca tip ismini tutmak amacıyla TypeName sütunu bulunmaktadır. Aşağıdaki çizelgede ilgili tablo gösterilmektedir (Çizelge 2.16).

Çizelge 2.16 : BYS_Type tablosu.

Sütun İsmi	Veri Tipi
TypeId	Int
TypeName	Nvarchar(50)

Uygulamada kullanılan son tablo ise iş isteklerin ve arızalarının aciliyet durumunu belli etmek için kullanılan BYS_Situation tablosudur. Tabloda primary key olarak SituationId kullanılmış olup ‘Normal’, ‘Acil’ ve ‘Çok Acil’ olmak üzere 3 adet veriden oluşan bir tablodur. Aşağıdaki çizelgede ilgili tablo gösterilmektedir (Çizelge 2.17).

Çizelge 2.17 : BYS_Situation tablosu.

Sütun İsmi	Veri Tipi
SituationId	Int
SituationName	Nvarchar(MAX)

2.2 Tablolar Arasındaki İlişkiler

Uygulamada kullanılan tablolar arasında birebir ve bireçok olmak üzere birçok ilişki bulunmaktadır.

Her bir kullanıcı bir adet bölüme ait olabilir fakat bir bölümde birden çok kişi bulunabilir. Bundan dolayı BYS_Users ile BYS_Department arasında birebir ve bireçok ilişki bulunmaktadır.

Her bir kullanıcı bir adet fabrikada bulunabilir fakat her fabrikada birden çok kişi bulunabilmektedir. Bundan dolayı BYS_Users ile BYS_Entity arasında birebir ve bireçok ilişki bulunmaktadır.

Her bir kullanıcının bir adet statü durumu (şef,yönetici,normal,bakımcı) bulunmaktadır. Dolayısıyla BYS_Users ve BYS_Status arasında birebir ilişki bulunmaktadır.

İş istekleri redlerinin depolandığı tablo olan BYS_RejectJobRequest ise bir adet iş isteğine bağlı olabileceğiinden BYS_JobRequest tablosu ile birebir ilişkiye sahiptir. Aynı zamanda bir reddetme işlemini sadece bir kullanıcı yapabileceğinden BYS_Users ile birebir ilişkiye sahiptir.

Periyodik bakım bilgilerinin tutulduğu BYS_PeriodicCheck tablosundaki her bir periyodik bakım bir frekansa sahip olması gerektiği için BYS_Frequency tablosu ile birebir ilişkiye sahiptir aynı zamanda her periyodik bakımın bir adet sorumlu bakım görevlisi ve bir adet oluşturan kişiye sahip olabileceği için BYS_Users tablosu ile birebir aynı zamanda bakım görevlilerden her biri birçok periyodik bakımından sorumlu olabileceği ve birçok periyodik bakım oluşturabileceğiinden bireçok ilişkiye sahiptir.

Malzemelerin tutulduğu tablo olan BYS_Material tablosundaki her malzeme bir adet kullanıcı tarafından yaratılabilcecinden ve her kullanıcı birden çok malzeme yaratma imkanına sahip olduğundan BYS_Users tablosu ile birebir ve bireçok ilişkiye sahiptir. Ayrıca malzeme yalnızca bir fabrikada bulunabileceğinden ve bir fabrikada birden çok malzeme bulunabileceğinden BYS_Entity tablosu ile birebir ve bireçok ilişkiye sahiptir.

Makinelerin tutulduğu BYS_Machine tablosu, bir kullanıcı tarafından oluşturulabildiğinden ve bir kullanıcı birden çok makine oluşturabildiği için BYS_Users tablosu ile birebir ve bireçok ilişkiye sahiptir. Bir makine bir adet bölüme ait olabileceğinden ve bir bölümde birden çok makine bulunabileceğinden dolayı BYS_Department tablosu ile birebir ve bireçok ilişkiye sahiptir. Bir makine bir adet makine tipine sahip olacağından dolayı BYS_MachineType tablosu ile birebir ilişkiye sahiptir. Ayrıca bir makine bir fabrikaya ait olacağından dolayı ve bir fabrikada birden çok makine olacağından dolayı BYS_Entity tablosu ile birebir ve bireçok ilişkiye sahiptir.

İş isteklerini tutan BYS_JobRequest tablosu, her iş isteği bir adet tipe sahip olabileceğinden ve her bir tip birden çok istege ait olabileceğinden BYS_Type tablosu ile birebir ve bireçok ilişkiye sahiptir. Her iş isteğin bir adet istek açan kullanıcı, şefi, yöneticisi, bakım görevlisi olabileceğinden ve aynı kişiler birden çok iş isteğinde görev alabileceğinden dolayı BYS_Users tablosu ile birebir ve bireçok ilişkilere sahiptir. Bir iş isteği bir duruma sahip olabileceğinden fakat bir durum birden fazla iş isteğinde bulunabileceğinden dolayı BYS_Situation tablosu ile birebir ve bireçok ilişkiye sahiptir. Bir iş isteği bir fabrikada bulunabileceğinden ve bir fabrikada birden çok iş isteği bulunabileceğinden dolayı BYS_Entity tablosu ile birebir ve bireçok ilişkiye sahiptir. Bir iş isteği bir bölüme ait olabileceğinden ve bir bölümde birden fazla iş isteği bulunabileceğinden dolayı BYS_Department tablosu ile birebir ve bireçok ilişkiye sahiptir. Son olarak bir iş isteği sadece bir red edilmeye sahip olabileceğinden ve bir reddedilme bir iş isteğinde olabileceğinden dolayı BYS_RejectJobRequest tablosu ile birebir ilişkiye sahiptir.

Fabrika bölümlerinin tutulduğu BYS_Department tablosu, bir şefe ve bir yöneticiye sahip olabileceğinden ve bir şef ile bir yönetici sadece bir bölüme ait olabileceğinden BYS_Users tablosu ile arasında birebir ilişki bulunmaktadır. Bir bölüm bir fabrikada bulunabileceğinden ve bir fabrikada birden çok bölüm bulunabileceğinden BYS_Entity tablosu ile birebir ve bireçok ilişkiye sahiptir.

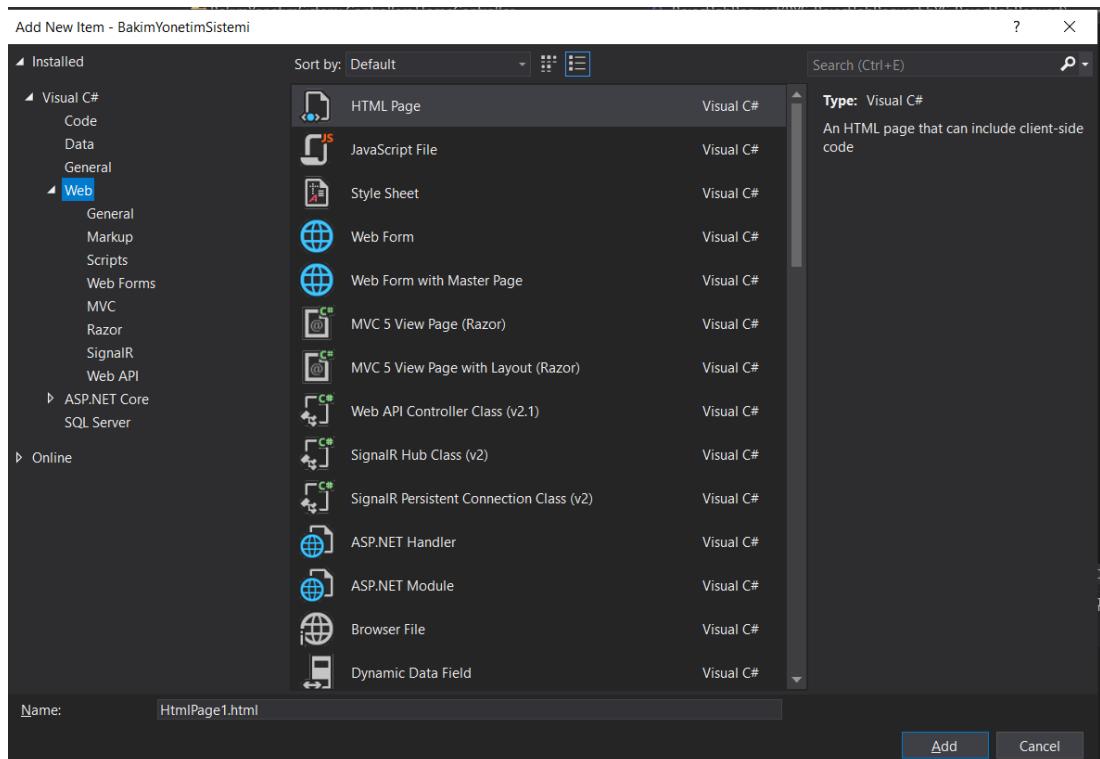
Arızada veya iş isteklerinde kullanılan malzemelerin tutulduğu BYS_BreakdownMaterialLog tablosu sadece bir iş isteği veya arızaya ait olabileceğinden BYS_Breakdown ve BYS_JobRequest tablolarıyla birebir ilişkiye sahiptir. Ancak kullanılan her malzeme teker teker eklendiği için bir iş isteği veya arıza birden çok log'a sahip olabilmektedir. Bu yüzden aralarında bireçok ilişki de bulunmaktadır. Aynı zamanda BYS_Entity tablosu ile birebir ve bireçok ilişkiye sahiptir. Ayrıca BYS_Material tablosu ile birebir ve bireçok ilişkiye sahiptir.

Arızada veya iş isteklerinde kullanılan personellerin ve çalışma saatlerinin tutulduğu BYS_BreakdownEmployeeLog tablosu BYS_BreakdownMaterialLog tablosundaki aynı sebeplerden dolayı BYS_Breakdown ve BYS_JobRequest tablolarıyla birebir ve bireçok ilişkiye sahiptir. Bir personel birden fazla logda bulunabilir ama bir log bir personele ait olabileceğinden dolayı BYS_Users tablosuyla birebir ve bireçok ilişkiye sahiptir. Aynı zamanda BYS_Entity ile birebir ve bireçok ilişkisi bulunmaktadır.

Arızaların tutulduğu BYS_Breakdown tablosu, bir adet istek yapan kullanıcıya, bir adet sorumlu kullanıcıya sahip olabileceğinden BYS_Users tablosu ile birebir ve bireçok ilişkiye sahiptir. Bir adet bölüme ait olabileceğinden ve bir bölümde birden fazla arıza bulunabileceğinden BYS_Department tablosu ile birebir ve bireçok ilişkiye sahiptir. Bir makineye ait olabileceğinden ve bir makine birden fazla arıza bulunabileceğinden BYS_Machine tablosu ile birebir ve bireçok ilişkiye sahiptir. Aynı sebeplerden dolayı da BYS_Type, BYS_Situation ve BYS_Entity tabloları ile birebir ve bireçok ilişkilere sahiptir.

2.3 Tabloların ASP.NET İle Bağlanması

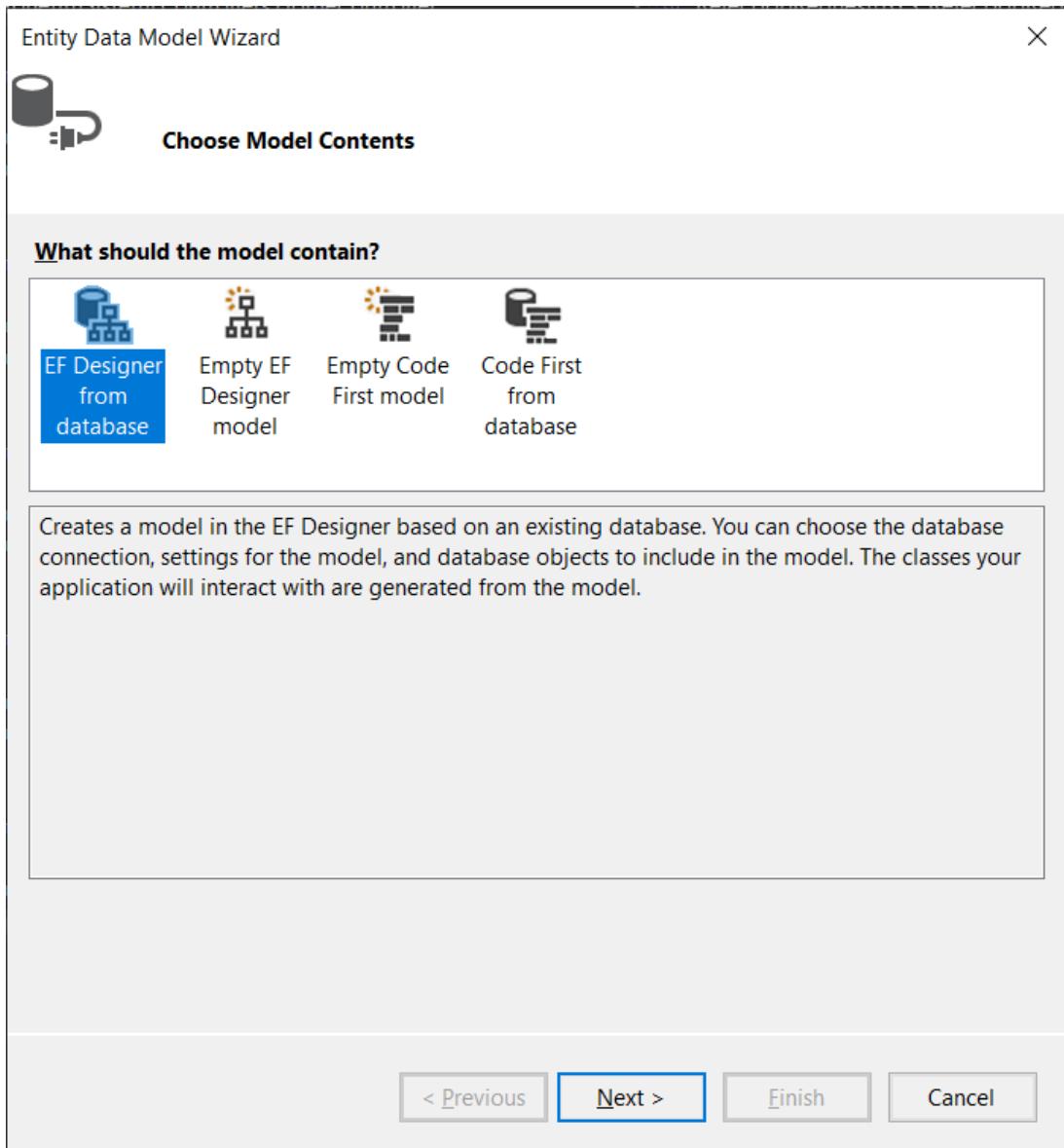
MSSQL Server'da oluşturulan tablolar ASP.NET'e Entity Framework yardımıyla bağlanacaktır. Visual Studio'da ASP.NET MVC uygulaması oluşturulduktan sonra sağ tarafta klasörler kısmında yer alan 'Models' klasörüne sağ tıklayıp önce 'Add' sekmesine daha sonra 'New Item' sekmesine tıklanır. Karşımıza aşağıdaki şekilde gösterilen pencere gelecektir (Şekil 2.2).



Şekil 2.2 : SQL Server ile tabloların bağlanması

Gelen pencereden sol kısmda yer alan 'Data' butonuna tıklayarak 'ADO.NET Entity Data Model' sekmesine tıklayıp 'Add' butonuna tıklanır. Gelen pencerede bize hangi

yaklaşım ile SQL Server ile bağlantı kurulacağını sormaktadır. Genel olarak kullanılan iki yaklaşım bulunmaktadır: Code First ve Database First yaklaşımları. Uygulamada Database First yaklaşımı kullanılmaktadır. Bağlantının sorulduğu pencerede ‘EF Designer from database’ seçeneğine tıklanarak devam ettirilir (Şekil 2.3).

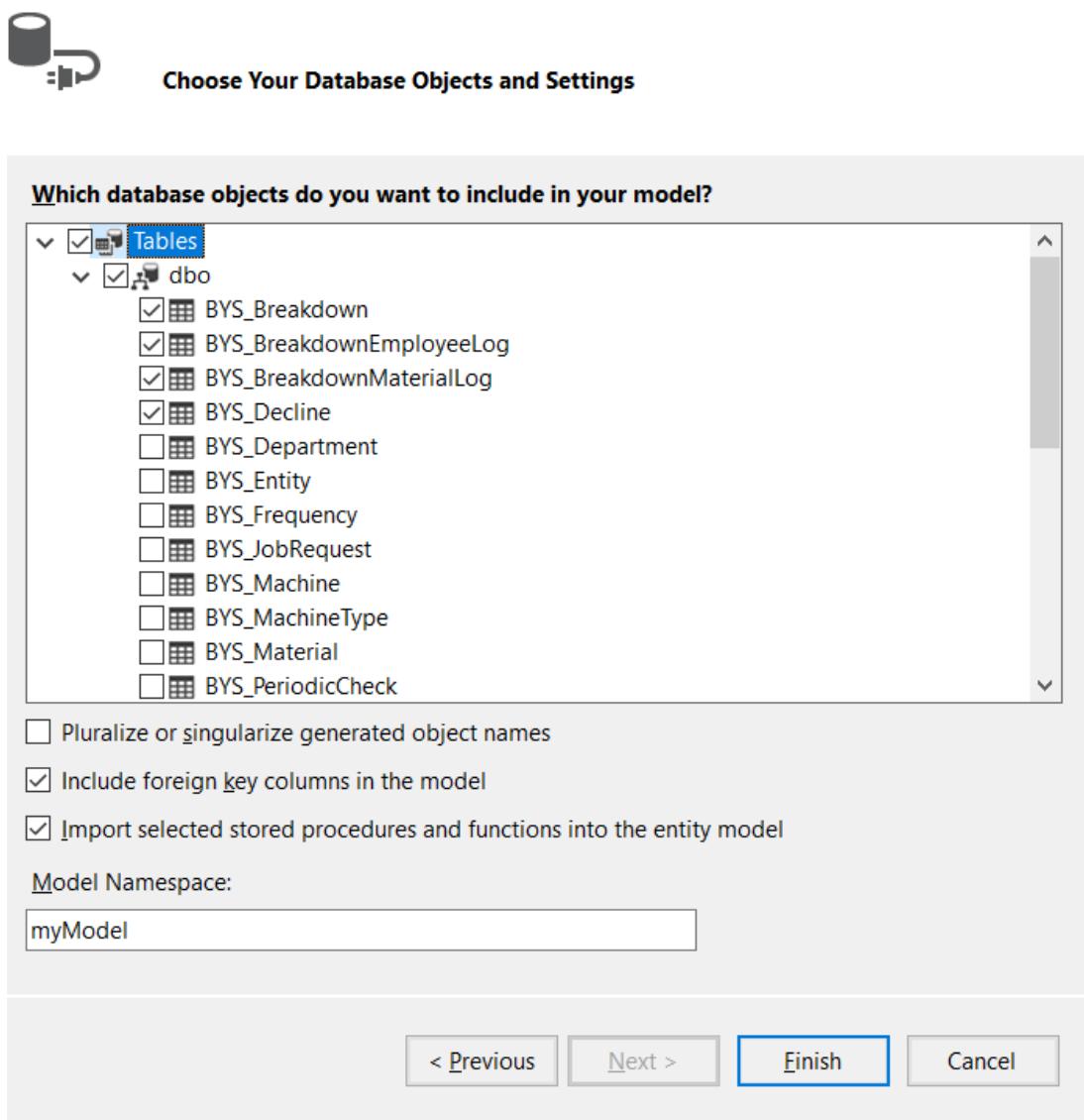


Şekil 2.3 : SQL Server bağlantı türünün seçilmesi

Next butonuna tıklanarak devam edilir. Sonraki pencerede ‘New Connection’ butonuna tıklayarak local veya uzak sunucuda bulunan SQL Server ile ilgili kullanıcı adı ve parola gibi bilgileri girerek bağlantı yapılması gerekmektedir. Bağlantı yapıldıktan sonra ASP.NET uygulamasına eklemek istenilen tablolar eklenir (Şekil 2.4) ve böylelikle kullanıma hazır hale gelmektedir.

Entity Data Model Wizard

X



Şekil 2.4 : İstenilen tabloların eklenmesi

3. UYGULAMANIN İNCELENMESİ

Projede yer alan kısımlar kullanıcı girişi, iş istekleri, arıza talepleri, periyodik bakımlar, kritik stok ve admin yönetim paneli olmak üzere 5 bölümde anlatılacaktır.

3.1 Kullanıcı girişi

Uygulamayı kullanan kullanıcılar iki aşamada giriş yapmaktadır. Öncelikle firma bünyesinde çalışan herkese verilen kullanıcı kodlarını girmelidirler. Kullanıcı kodunun doğru olup olmadığı sistem tarafından kontrol edilmektedir. Kullanıcı kodu doğru olduğunda ise her kullanıcının mailine bir doğrulama kodu gönderilmektedir. Firma bünyesinde bir mail gönderme servisi hali hazırda bulunmaktadır. Bulunan servis veritabanını her 1 dakikada bir kontrol ederek yeni eklenen satırları mail olarak gönderme işlemi yapmaktadır. Anlatılan uygulamada ise, kullanıcılara mail göndermek için ilgili mail tablosuna yeni bir satır eklenmektedir. Kullanıcının girdiği kullanıcı kodu bilgisine göre veritabanında kontrol yapan kod parçası (Şekil 3.1) ve kullanıcıya doğrulama kodunu mail gönderen kod parçası (Şekil 3.2) aşağıda gösterilmiştir.

```
string userCode = Request.Form["UserCode"];
if (userCode != null)
{
    using (Bys_Entities1 db = new Bys_Entities1())
    {
        var user = db.BYS_Users.Where(x => x.UserCode == userCode).FirstOrDefault();
```

Şekil 3.1 : Kullanıcıdan gelen bilginin doğrulanması

```
using (MAJORSKTEntities dbMskt = new MAJORSKTEntities())
{
    mailVerificationCode = new Random().Next(100000, 999999).ToString();
    MAIL mail = new MAIL();
    mail.MailFrom = "bakimyonetimsistemi@adient.com";
    mail.MailTo = user.Email;
    mail.MailSubject = "Bakım Yönetim Sistemi Doğrulama Kodu";
    mail.MailBody = "<html><head></head><body><h2>Sayın " + user.FirstName + " " +
    dbMskt.MAIL.Add(mail);
    dbMskt.SaveChanges();
}
```

Şekil 3.2 : Kullanıcıya mail gönderilmesi

Kullanıcıdan alınan kullanıcı kodu ve doğrulama kodu bilgileri doğru ise session kullanarak kullanıcının girişi sağlanmaktadır. Her fonksiyonda session kontrolü

yapmamak adına boolean değer döndüren iki adet fonksiyon oluşturulmuştur. Bu fonksiyonlar kullanıcının hangi aşamada olduğunu belirleyen session'ları kontrol etme işini yapmaktadır. İstemci ile sunucu arasındaki session bağlantısı ile bu kullanıcının hangi aşamada olduğu anlaşılmaktadır.

Random string veri üreten bir fonksiyon yazılmıştır. Bu fonksiyon ister doğrulama kodu üretiminde ister uygulamanın herhangi bir yerinde random sayıya ihtiyaç duyulduğunda kullanılmak üzere tasarlanmıştır. Aşağıdaki görselde (Şekil 3.3) ilgili kod parçası gösterilmektedir.

```
private static string RandomString(bool? isVerification)
{
    Random random = new Random();
    const string pool = "abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    var builder = new StringBuilder();
    int length = 12;
    if (isVerification == true)
    {
        length = 6;
    }

    for (var i = 0; i < length; i++)
    {
        var c = pool[random.Next(0, 55)];
        builder.Append(c);
    }

    return builder.ToString();
}
```

Şekil 3.3 : Random string veri üretilmesi

Kullanıcının hem kullanıcı kodunu hemde doğrulama kodunu girdiği sayfa aynı sayfadır. Backend tarafında girilen input tag'ının name özelliği ve başlık kısımları dinamik olarak değiştirilir (Şekil 3.4). Böylelikle ekstra sayfa kullanmadan işlem halledilmiş olur.

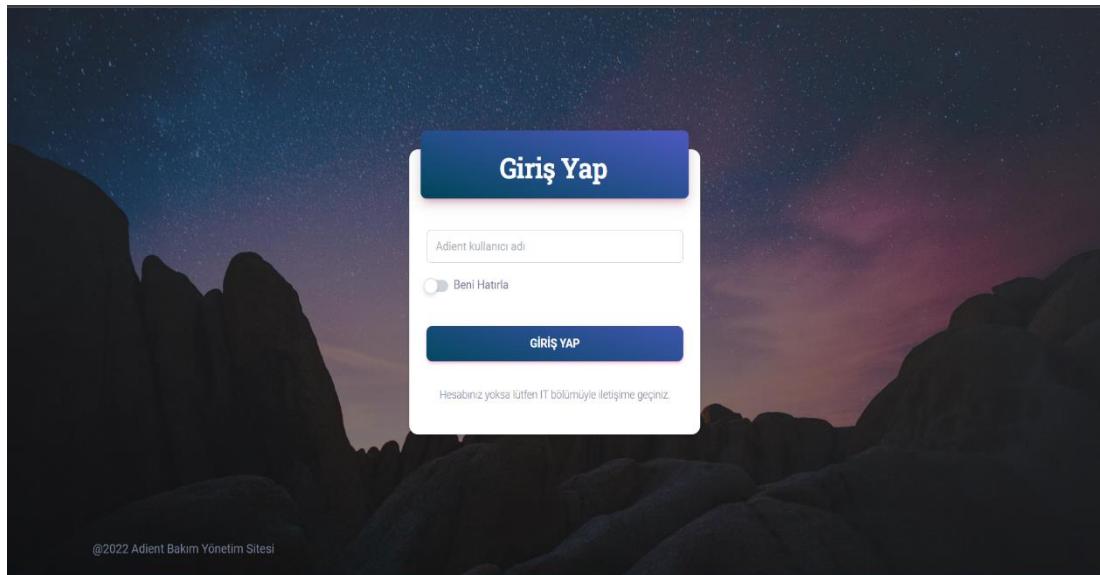
```
bool isAuthenticated = IsAuthenticated();

if (isAuthenticated == false)
{
    ViewBag.title = "Giriş başarılı lütfen mailinizdeki onay kodunu giriniz.";
    ViewBag.input = "VerificationCode";
    ViewBag.placeholder = "Doğrulama Kodu";

    return View("SignIn");
}
```

Şekil 3.4 : Giriş sayfasına dinamik olarak veri gönderilmesi

Kullanıcının giriş yaptığı sayfa aşağıdaki görselde gösterilmiştir (Şekil 3.5).



Şekil 3.5 : Giriş sayfası

Kullanıcıya gelen mail ise örnek olarak gösterilmiştir (Şekil 3.6).

Reply Reply All Forward IM
bakimyonetimsistemi@adient.com | Batuhan Avcı
Bakım Yönetim Sistemi Doğrulama Kodu
i This message was sent with High importance.

Sayın Batuhan Avcı

Doğrulama Kodunuz : 385110

Şekil 3.6 : Kullanıcıya gelen mail

3.2 İş istekleri

Kullanıcılar doğru bir şekilde giriş yaptıktan sonra, iş isteklerini görecekleri sayfaya yönlendirilmektedirler. Kullanıcılara aktif olan iş istekleri ve kullanıcının bulunduğu fabrikadaki iş istekleri gösterilmektedir. Aşağıdaki görselde veritabanındaki bulunan iş isteklerinin birebir ve bireçok ilişkiler yardımıyla id bilgilerini kullanarak farklı tablolar ile etkileşime girmesi ve ilgili bilgileri farklı tablolardan çekmesi işlemleri gösterilmektedir. Ayrıca çekilen bilgileri tabloda daha verimli bir şekilde gösterebilmek adına 'JobRequestTable' adında bir sınıf oluşturulmuştur. Çekilen bilgiler bu sınıfta bir liste olacak şekilde doldurulmakta ve istemci kısmına

döndürülmektedir (Şekil 3.7). Bu işlemlerin yanında kullanıcı Index.cshtml sayfasına yönlendirilmektedir. Sunucudan gelen veriler bir for döngüsü yardımıyla ekrana bastırılmaktadır. Kullanıcıya gelen ekran aşağıdaki şekilde (Şekil 3.8) gösterilmiştir.

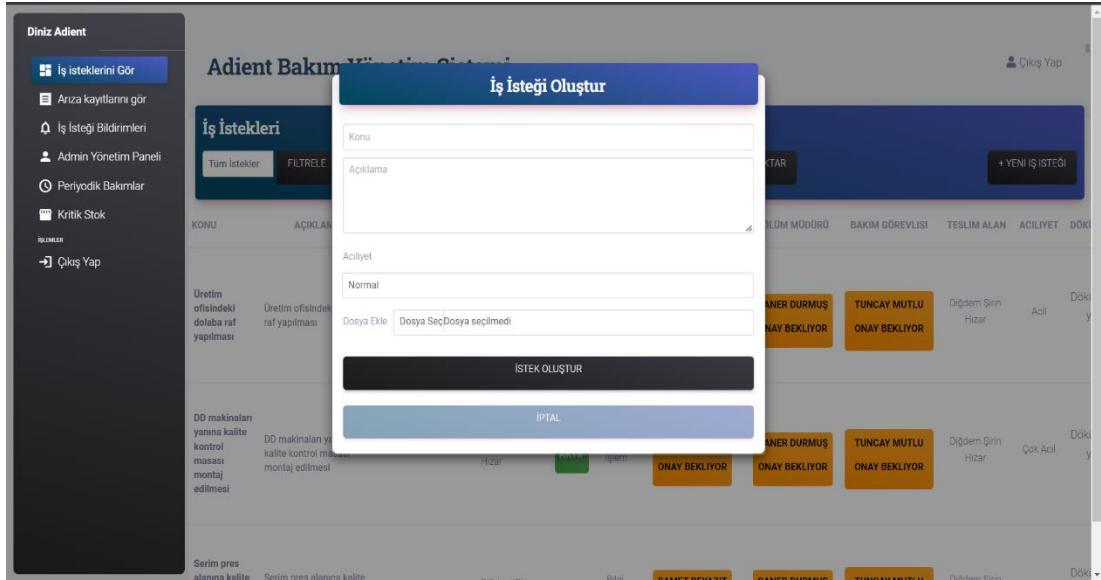
```
var allJobRequests = (from x in db.BYS_JobRequest
                      where x.EntityId == entityId
                      where x.IsActive == true
                      join s in db.BYS_Users on x.RequesterId equals s.UserId
                      join y in db.BYS_Users on x.ApprovalChiefId equals y.UserId
                      join z in db.BYS_Users on x.ApprovalManagerId equals z.UserId
                      join w in db.BYS_Users on x.ApprovalMaintenanceId equals w.UserId
                      join a in db.BYS_Situation on x.SituationId equals a.SituationId
                      join d in db.BYS_Department on x.DepartmentId equals d.DepartmentId
                      select new JobRequestTable
{
    Subject = x.Subject,
    Description = x.Description,
    RequestDate = x.RequestDate,
    RequesterName = s.FirstName + " " + s.SecondName,
    IsActive = x.IsActive == true ? "AKTİF" : "AKTİF DEĞİL",
    ApprovalChief = y.FirstName + " " + y.SecondName,
    ApprovalManager = z.FirstName + " " + z.SecondName,
    ApprovalMaintenance = w.FirstName + " " + w.SecondName,
    Situation = a.SituationName,
    Department = d.DepartmentName,
    ApprovalChiefTime = x.ApprovalChiefTime,
    ApprovalManagerTime = x.ApprovalManagerTime,
    ApprovalMaintenanceTime = x.ApprovalMaintenanceTime,
    JobRequestFile = x.JobRequestFile ?? null,
}).OrderByDescending(x=>x.RequestDate).ToList();
```

Şekil 3.7 : İş isteklerinin veritabanından alınması

Şekil 3.8 : İş isteklerinin görüntülenmesi

İş istekleri açıldığı zaman belli aşamalardan geçmesi gerekmektedir. Öncelikle bir iş isteği oluşturmak için sağ üst kısımda yer alan ‘Yeni İş İsteği’ yazan butona tıklanır

ve yeni sayfa yüklenmeden aynı sayfa üzerinde bir modal açılmaktadır. Bu modalda ilgili bölümler doldurulup form olarak sunucuya gönderilir ve böylece yeni bir iş isteği oluşturulmaktadır. Açılan form modalı aşağıda gösterilmiştir (Şekil 3.9).



Şekil 3.9 : Yeni iş isteği oluşturma

Form gönderildikten sonra sayfa yenilenmekte ve oluşturulan istek listeye eklenmektedir.

İş isteği oluşturma işleminden sonra, istek oluşturan kişinin bağlı olduğu bölüm şefinin iş isteğini onaylaması gerekmektedir. Yeni iş isteği oluşturulurken kullanılan kod parçası (Şekil 3.10) aşağıdaki görselde gösterilmektedir.

```
var userId = Convert.ToInt32(Session["userId"]);
var user = db.BYS_Users.Where(x => x.UserId == userId).FirstOrDefault();
var maintenanceId = db.BYS_Users.Where(x => x.DepartmentId == 14).FirstOrDefault();
var jobRequest = new BYS_JobRequest
{
    Subject = subject,
    Description = description,
    SituationId = situation,
    RequesterId = userId,
    ApprovalChiefId = user.BYS_Department.DepartmentChiefId != null ? user.BYS_Department.DepartmentChiefId : 115,
    ApprovalManagerId = user.BYS_Department.DepartmentManagerId != null ? user.BYS_Department.DepartmentManagerId : 115,
    ApprovalMaintenanceId = maintenanceId != null ? maintenanceId.UserId : 115,
    EntityId = user.EntityId,
    RequestDate = DateTime.Now,
    IsActive = true,
    DepartmentId = user.DepartmentId
};
```

Şekil 3.10 : Yeni iş isteği oluşturma kodu

Bölüm şefinden sonra bölüm yöneticisine istek iletilmekte ve bölüm yöneticisinin de onaylaması gerekmektedir. Solda yer alan ‘İş İsteği Bildirimleri’ sekmesinde her bir kişiye ait onay bekleyen iş istekleri görüntülenebilmektedir. Bölüm şeflerinin ve yöneticileri kendilerine gelen iş isteklerini sadece reddebilme ve onaylama yetkisine

sahiptirler. İş isteğini reddetme butonuna basıldığında karşısına bir modal çıkar ve red sebebi sorulmaktadır. Girilen açıklama iş isteklerinin red edilmelerinin logunun tutulduğu BYS_RejectJobRequest tablosunda depolanmaktadır. Red etmek yerine onaylamayı seçerler ise iş isteği şeften yöneticiye, yöneticiden ise bakım görevlisine geçmektedir. Aşağıdaki görselde şef ve yöneticilere ait iş isteklerini onaylama ve red etme ekranı gözükmektedir (Şekil 3.11).

Şekil 3.11 : İş isteklerini onaylama ekranı

Bölüm şefleri ve yöneticileri ilgili iş isteğini onayladıktan sonra istek sorumlu bakım personeline düşmektedir. Sorumlu bakım personeli iş isteğini reddetme veya iş isteğini bitirme yetkisine sahiptir. İş isteğini bitirirken kullandığı malzemeleri ve çalışan personelleri ekleyebilmektedir. Eklenilen malzemeler veritabanındaki stoktan düşürülür, log olarak tutulur. Eklenilen personeller ise daha sonra hangi personelin hangi arızada kaç saat çalıştığı ile ilgili bilgilerin edinilmesi açısından tutulmaktadır. Aşağıdaki görselde bakım görevlisinin iş isteğin son aşamasında dolduracağı form görüntülenmektedir (Şekil 3.12).

Şekil 3.12 : İş isteklerini tamamlama ekranı

İş istekleri eğer herhangi bir şef, yönetici, veya bakım görevlisinden reddedilirse iş isteği aktifliğini kaybetmektedir. Eğer ki şef, yönetici veya bakım görevlisinin tamamından onay alırsa tekrar işi oluşturan kişiye iş isteği geri dönmemektedir. Eğer isteği yapan kişi de onaylar ise iş isteği tamamen aktifliğini kaybetmekte fakat iş isteğini reddeder ise bakım görevlisine geri dönmemektedir. Böylelikle iş isteklerinin aktarım kısmını tamamlanmaktadır.

Bildirim ekranının gösterildiği kısımla ilgili, bir kişiye gelen bildirimlerin gösterilmesine dair kod parçası aşağıdaki görselde gösterilmiştir (Şekil 3.13).

```
var notificationJobRequests = (from x in db.BYS_JobRequest
                                where x.EntityId == user.EntityId
                                where x.ApprovalChiefTime != null
                                where x.ApprovalManagerTime != null
                                where x.ApprovalMaintenanceTime != null
                                where x.ApprovalRequesterTime == null
                                where x.RequesterId == user.UserId
                                where x.IsActive == true
                                join y in db.BYS_Users on x.ApprovalManagerId equals y.UserId
                                join z in db.BYS_Users on x.ApprovalChiefId equals z.UserId
                                join w in db.BYS_Users on x.ApprovalMaintenanceId equals w.UserId
                                join d in db.BYS_Situation on x.SituationId equals d.SituationId

                                select new NotificationTable
{
    Subject = x.Subject,
    Description = x.Description,
    RequestDate = x.RequestDate,
    ApprovalManager = y.FirstName + " " + y.SecondName,
    JobRequestId = x.JobRequestId,
    ApprovalChief = z.FirstName + " " + " " + z.SecondName,
    ApprovalMaintenance = w.FirstName + " " + w.SecondName,
    Situtation = d.SituationName,
    ApprovalChiefTime = x.ApprovalChiefTime,
    ApprovalManagerTime = x.ApprovalManagerTime,
    ApprovalMaintenanceTime = x.ApprovalMaintenanceTime,
}).Take(300).ToList();
```

Şekil 3.13 : Bildirimlerin veritabanından getirilmesi

Bildirimlerin veritabanından getirilme mantığı, veritabanında BYS_JobRequest tablosunda tutulan ‘ApprovalChiefTime’, ‘ApprovalManagerTime’, ‘ApprovalMaintenanceTime’ ve ‘ApprovalRequesterTime’ sütunlarına bağlıdır. Örnek vermek gerekirse bir şefe gelen bildirimler, iş istekleri tablosunda ‘ApprovalChiefId’ sütunu şefin UserId sine eşit olanlar arasından

‘ApprovalChiefTime’ sütunu NULL değere eşit olanlardır. Açıklamak gerekirse iş isteklerinin bulunduğu tabloda onaylaması gereken şef, giriş yapan kullanıcı olma durumu kontrol edilir aynı zamanda şefin daha önce onay verip vermediği de kontrol edilmektedir. Böylelikle şefin bildirimleri şef ekranına düşmüş olmaktadır. Aynı işlemler yönetici ve bakım görevlileri için de gerçekleşmektedir.

İş isteklerinin görüntülendiği sayfaya geri gelinecek olursa, iş istekleri ‘Tüm İstekler’, ‘Aktif İstekler’ ve ‘İnaktif İstekler’ olmak üzere 3 ayrı şekilde filtreleme işlemi gerçekleştirilebilmektedir. Filtreleme işleminde sunucu tarafına filtrelenmesi istenen kelime gönderilmekte ve ona göre veritabanından veriler getirilmektedir. İlgili kod parçası aşağıdaki şekilde gösterilmektedir (Şekil 3.14).

```
string filterRequest = Request.Form["filterRequest"];
int userId = Convert.ToInt32(Session["userId"]);

if (filterRequest == "availableRequests")...

if (filterRequest == "notAvailableRequests")...

if (filterRequest == "all")...
```

Şekil 3.14 : İş isteklerinin filtrelenmesi

İş istekleri aynı zamanda Excel formatında çıktı olarak da alınabilmektedir. Excel formatında işlem yapabilmek için Visual Studio Nuget Paket Yöneticisi’nden EPP+ eklentisi indirilmelidir. Yükleme işleminden sonra öncelikle bir şablon Excel dosyası oluşturmak gerekmektedir. Şablon excel dosyasının yolu bir string veriye atanmalı ve ExcelPackage sınıfından bir değişken türeterek bu değişken yardımıyla Excel dosyası yönetilmelidir. Excel formatında çıktı alabilmek için kullanıcının hangi tarihler arasındaki iş isteklerini almak istedikleri bilgisi istenmektedir. Gelen tarihlere göre veritabanına soru atılır ve gelen iş istekleri şablonun altına bir foreach döngüsü yardımıyla doldurulur. Daha sonra sunucuya kaydedilerek kullanıcıya Excel dosyası geri döndürülmektedir. Veritabanına atılan sorgudan sonra Excel dosyasının hazırlanması ile ilgili kod bloğu (Şekil 3.15) ve örnek bir Excel formatında çıktı (Şekil 3.16) aşağıdaki görsellerde gösterilmektedir.

```

ExcelPackage.LicenseContext = LicenseContext.NonCommercial;

DosyaYolu = Path.Combine(ServerDosyaYolu, "İş İstek Listesi.xlsx");

if (System.IO.File.Exists(DosyaYolu))
{
    System.IO.File.Delete(DosyaYolu);
}

var jobRequestExcelPath = "C:\\Users\\aavcib\\Desktop\\BakımYonetimSistemi\\BakımYonetimSistemi\\Uploads\\excel\\jobRequestExcel.xlsx";

using (var excelPaketi = new ExcelPackage(new FileInfo(jobRequestExcelPath)))
{
    var sayfalar = excelPaketi.Workbook.Worksheets;
    var sayfa1 = sayfalar[0];
    int Y = 3;
    var referansAdıHucresi = sayfa1.Cells[1, 5];
    referansAdıHucresi.Value = Baslangic.ToString("dd.MM.yyyy") + " - " + Bitis.ToString("dd.MM.yyyy");

    foreach (JobRequestTable jobRequestTemp in jobRequests) ...

    excelPaketi.SaveAs(new FileInfo(DosyaYolu));
}

```

Şekil 3.15 : İş isteklerinin Excel formatına çevrilmesi

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
19.06.2022 - 21.06.2022																						
2	İş İstek ID	Konu	Açıklama		İstek Tarihi	İstek Yapan Kişi	Aktiflik	Bölüm	Aciliyet	Sorumlu Şef	Şef Onay	Sorumlu Yönetici	Yönetici Onay	Sorumlu Bakıcı	Bakıcı Onay							
3	10	İş isteği	Denemesi		20.06.2022	Batuhan Avcı	AKTİF	Bilgi İşlem	Normal	Ali Türker	20.06.2022	Sedat Alan	20.06.2022	Berker Çelenk	20.06.2022							
4	241	Branda Ka Hammadde ambar branda ka	20.06.2022		Görkem BALCI	AKTİF	Bilgi İşlem	Normal	Ahmet Erdem	Serkan Sümer												
5																						
6																						
7																						
8																						
9																						
10																						
11																						
12																						
13																						
14																						
15																						
16																						
17																						
18																						
19																						
20																						
21																						
22																						
23																						
24																						
25																						
26																						
27																						
28																						
29																						

Şekil 3.16 : İş isteklerinin Excel olarak çıktı alınması

İş istekleri oluştururken kullanıcılar dosya da yükleyebilmektedirler. Dosya şirketin sunucusundaki ortak bir bölümde tutulmakta olup, istenen zamanda iş isteklerin görüntülendiği tabloda ‘Döküman’ sütunu altından ‘İndir’ butonuna tıklayarak indirilebilmektedir.

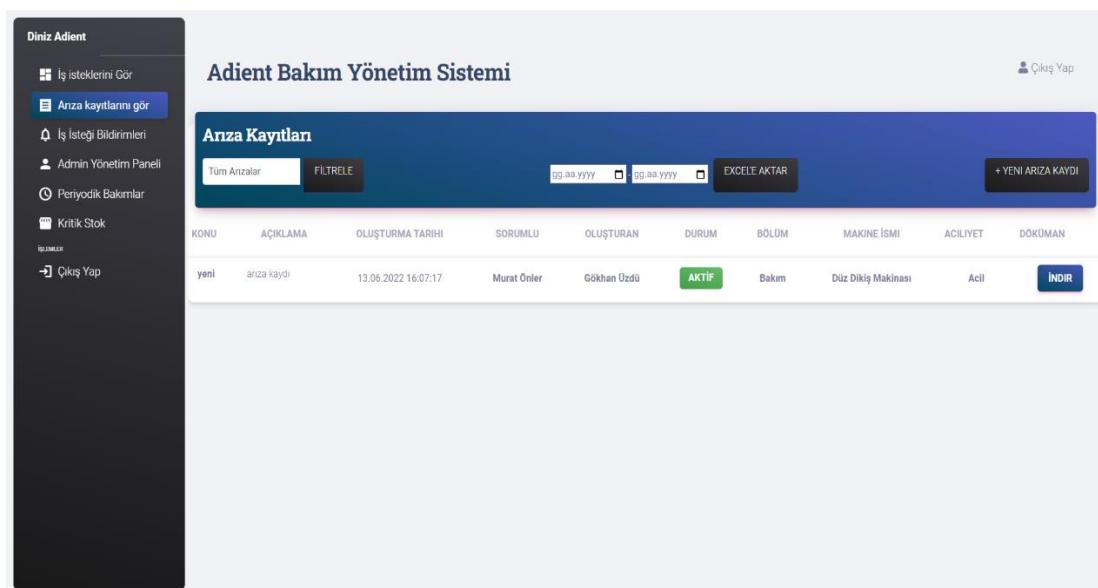
3.3 Arıza Talepleri

Kullanıcılar soldaki menüden arıza taleplerini görebilmektedirler. Arızaların iş isteklerinden farkı, bir makineye ait olmasıdır. İş istekleri her alanda her türlü iş için açılabilecek iken, arızalar sadece bir makineye ait olabilmektedir. Ayrıca arızalarda aktarma sistemi bulunmamakta, bakım görevlileri tüm arızaların görüntünlendiği sayfadan direkt olarak arıza bilgilerini görüp arızayı reddedebilir veya onaylayıp kapatabilmektedir. Soldaki menüden ‘Arıza Kayıtlarını Gör’ linkine tıklandığında arızaların görüntünlendiği sayfa açılmaktadır. Arızaların getirilmesi ile ilgili kod bloğu (Şekil 3.17) ve kullanıcıların gördüğü sayfa (Şekil 3.18) aşağıdaki görsellerde gösterilmektedir.

```
var allBreakdownList = (from x in db.BYS_Breakdown
                        where x.EntityId == entityId
                        where x.IsActive == true
                        join s in db.BYS_Users on x.BreakdownRequesterId equals s.UserId
                        join y in db.BYS_Department on x.DepartmentId equals y.DepartmentId
                        join z in db.BYS_Machine on x.MachineId equals z.MachineId
                        join w in db.BYS_Situation on x.SituationId equals w.SituationId
                        join c in db.BYS_Users on x.BreakdownResponsibleId equals c.UserId
                        select new BreakdownTable
                        {
                            BreakdownId = x.BreakdownId,
                            Subject = x.Subject,
                            Description = x.Description,
                            CreatedAt = x.CreatedAt,
                            RequesterName = s.FirstName + " " + s.SecondName,
                            IsActive = x.IsActive == true ? "AKTİF" : "AKTİF DEĞİL",
                            Department = y.DepartmentName,
                            Machine = z.MachineName,
                            Situation = w.SituationName,
                            BreakdownFile = x.BreakdownImage,
                            BreakdownResponsible = c.FirstName + " " + c.SecondName
                        }).ToList();

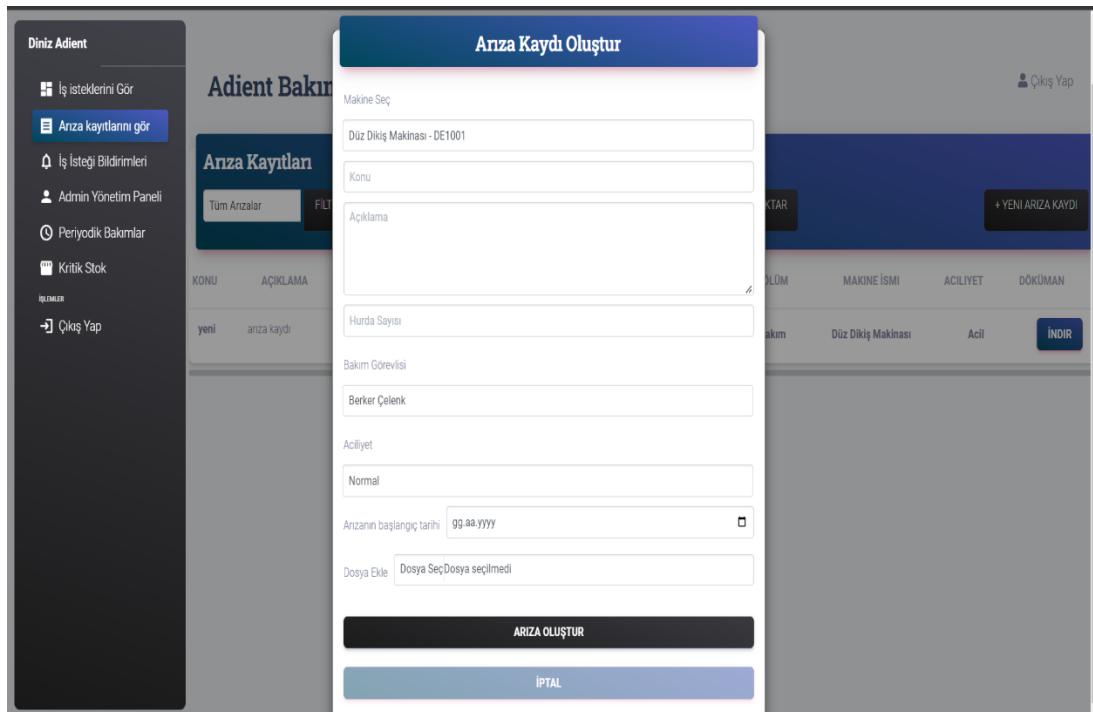
var machineTypes = db.BYS_MachineType.ToList();
var allMachines = db.BYS_Machine.ToList();
var allMaintenanceEmployee = db.BYS_Users.Where(x => x.IsActive == true && x.EntityId == user.EntityId && x.StatusId == 4).ToList();
```

Şekil 3.17 : Arızaların veritabanından getirilmesi



Şekil 3.18 : Arızaların kullanıcıya gösterilmesi

Arıza kayıtları sayfasında iken kullanıcılar sağ üstteki ‘Yeni Arıza Kaydı’ butonuna tıklayarak aynı iş isteklerindeki gibi bir modal yardımıyla yeni bir arıza kaydı oluşturabilmektedir. Fakat buradaki fark kullanıcılar arıza yapan makineyi ve sorumlu bakım görevlisini bir dropdown yardımıyla kendileri seçebilmektedir. İlgili modal aşağıdaki şekilde (Şekil 3.19) gösterilmektedir.



Şekil 3.19 : Arıza kaydı oluşturulması

Arıza kaydı oluşturulduktan sonra sayfa yenilenmekte ve ilgili arıza veritabanına eklenmektedir.

Arızaların görüntülendiği listede, giriş yapmış olan kullanıcı bakım görevlisi ise, ekstra bir sütun açılmaktadır. Bu sütun yardımıyla bakım görevlileri arızaların bilgilerine ulaşabilmektedir. Arıza bilgilerinin yanında eğer arıza aktif halde ise arıza açıklaması, kullandıkları malzeme ve çalışan personelleri girebileceği bir form bulunmaktadır. Aynı zamanda eğer arızayı red edecek ise red nedenini girebileceği ayrı bir form bulunmaktadır.

Arızada kullanılan malzemelerden yaklaşık 60 bin adet olduğu için dropdown seçenekleri burada işlememektedir. Bu sorun malzeme seçiminin dinamik bir şekilde yapılması ile çözülmüştür. Kullanıcı malzemeleri gireceği input alanına her bir harf girdiğinde sunucu tarafında olan fonksiyona istek gönderecek, sunucu tarafındaki fonksiyonda ise veritabanına girilen harf ile sorgu atıp geriye JSON olarak veri döndürecek şekilde bir sistem geliştirilmiştir. Geriye döndürülen JSON JavaScript ile işlenerek istemci

kısmında kullanıcıya gösterilmektedir. Kullanıcı gösterilen ilgili malzemeler arasında eklemek istediği malzemeye tıklayarak ekleme yapabilmektedir. Tıklamadan sonra ilgili malzemenin ismi ile kaç adet kullanıldığıyla ilgili bilgi girebileceği bir div bloğu eklenmektedir. Kullanıcı her malzeme seçtiğinde işlemler tekrarlanmakta ve seçimler tamamlandıktan sonra dinamik bir şekilde sunucuya gönderilmektedir. Aşağıdaki görsellerde malzemelerin girileceği ekran örnek olarak iki harf girilmiş şekilde (Şekil 3.20), JavaScript kullanılarak sunucuya istek atan kod bloğu (3.21) ve sunucu tarafında eş zamanlı olarak çalışan fonksiyon (Şekil 3.22) gösterilmiştir.

Şekil 3.20 : Malzeme isminin girilmesi

```
function getMaterials() {
    var searchTerm = document.getElementById("searchTerm").value;
    var url = "@Url.Action("GetMaterials", "Home")?searchTerm=" + searchTerm;
    console.log(url);
    fetch(url).then(function (response) {
        return response.json();
    }).then(function (data) {
        res = document.getElementById("result");
        res.innerHTML = '';
        let list = '';

        for (var i = 0; i < data.length; i++) {
            list += `li materialdescription="${data[i].MaterialDescription}" onclick="clickedLi(this)"`;
        }
        res.innerHTML = '<ul>' + list + '</ul>';
        return true;
        console.log(data);
    }).catch(function (err) {
        console.log("Cant fetch");
    });
}
```

Şekil 3.21 : Sunucuya eş zamanlı istek atılması

```

[HttpGet]
0 references
public ActionResult GetMaterials()
{
    using (Bys_Entities1 db = new Bys_Entities1())
    {
        var searchTerm = Request.QueryString["searchTerm"];
        var allMaterials = db.BYS_Material.Where(x => x.MaterialDescription.Contains(searchTerm)).Take(6).ToList();
        return Json(allMaterials, JsonRequestBehavior.AllowGet);
    }
}

```

Şekil 3.22 : Sunucuda çalışan fonksiyon

Arızalarda kullanılan malzemeler istemci tarafında aynı isim (UsedMaterialId) ile yanına eklenen malzeme sayısını alarak (Örn: UsedMaterialId2) eşsiz bir şekilde malzeme input'larına name özelliği eklenmiş olur. Böyle yapılmasının sebebi sunucu tarafında aynı isimde olanları bir listeye atmak ve bir for döngüsü yardımıyla kullanılan malzemeleri malzeme stoğundan düşmek ve log tablosunda kaydını tutabilmektir. Aşağıdaki görselde istemci tarafında eşsiz bir şekilde input'ları isimlendirme işlemi (Şekil 3.23), tüm malzemeleri sunucu tarafında alma işlemi (Şekil 3.24) ve bir for döngüsü yardımıyla log'lama işleminin başlangıcı (Şekil 3.25) gösterilmektedir.

```

function clickedLi(a) {
    res = document.getElementById("result");
    res.innerHTML = '';
    var materialDescription = $(a).attr('materialdescription');
    $("#selectedInput").append(` 
        <div id="ff" class="inputToTake"> <div class="input-group input-group-outline my-3">
            <input value="${materialDescription}" name="UsedMaterialId${a.id}"
                   id="materialSelect${k}" class="form-control myClassP" style="width:inherit"
                   readonly/>
        </div>
        <div class="input-group input-group-outline my-3">
            <input placeholder="Adet" name="UsedMaterialCount${a.id}" min="1" max="100"
                   required type="number" id="adet" class="form-control myClassP">
        </div></div>
    `);
}

```

Şekil 3.23 : İstemcide dinamik olarak inputların isimlendirilmesi

```

var ListUsedMaterialId = Request.Form.AllKeys.Where(x => x.Contains("UsedMaterialId")).ToList();
var ListUsedMaterialCount = Request.Form.AllKeys.Where(x => x.Contains("UsedMaterialCount")).ToList();

```

Şekil 3.24 : Sunucu tarafında gelen malzemelerin alınması

```

for (int i = 0; i < ListUsedMaterialId.Count; i++)
{
    var usedMaterialCountFromRequest = GetDataFromRequest(ListUsedMaterialCount[i]);
    string a = ListUsedMaterialId[i].Substring(14);
    int usedMaterialIdFromRequest = Convert.ToInt32(a);

    BYS_BreakdownMaterialLog bYS_BreakdownMaterialLog = new BYS_BreakdownMaterialLog();
    int variableOfTryParse = 0;
    bYS_BreakdownMaterialLog.UsedMaterialId = usedMaterialIdFromRequest;

    if (int.TryParse(usedMaterialCountFromRequest, out variableOfTryParse))
    {
        bYS_BreakdownMaterialLog.UsedMaterialCount = variableOfTryParse;
    }
}

else
{
    bYS_BreakdownMaterialLog.UsedMaterialCount = 0;
}

```

Şekil 3.25 : Gelen tüm malzemelerin log olarak tutulması

Malzeme işlemi için yapılan işlemlerin bir benzeri çalışan personeller için de yapılır ve arıza kaydı böylelikle kapanmaktadır.

Kullanıcılar iş isteklerindeki gibi arızaları filtreleyerek belirli arızaları getirebilmektedir. ‘Tüm arızalar’, ‘Devam Eden Arızalar’ ve ‘Bitmiş Arızalar’ olmak üzere sunucu tarafından bir fonksiyona form yardımıyla istedikleri filtre için bilgi göndermeye, gönderilen bilgi ile veritabanına soru atılıp ilgili arızalar getirilmektedir.

Kullanıcılar aynı zamanda belirli tarih aralığındaki arıza kayıtlarını Excel formatında çıktı olarak alabilmektedirler. Aynı iş isteklerindeki gibi öncelikle kullanıcıdan iki adet tarih alınmakta, sunucu tarafından bir fonksiyona HTTP POST metodu ile gönderilmekte ve veritabanına ilgili tarih aralığında arıza bulunup bulunmadığını dair soru atılmaktadır. Excel formatında çıktı alabilmek için iş isteklerinde olduğu gibi Epp+ eklentisi yüklenmeli ve bir adet şablon Excel dosyası oluşturulmalıdır. Şablon dosyanın yolu gerekli kütüphaneye verilir, veritabanından gelen sorgular bir for döngüsü yardımıyla Excel satırlarına işlenir. İlgili kod blogu aşağıdaki görselde gösterilmiştir (Şekil 3.26).

```

ExcelPackage.LicenseContext = LicenseContext.NonCommercial;

DosyaYolu = Path.Combine(ServerDosyaYolu, "Arıza Listesi.xlsx");

if (System.IO.File.Exists(DosyaYolu))
{
    System.IO.File.Delete(DosyaYolu);
}

var jobRequestExcelPath = entryDosyaYolu;

using (var excelPaketi = new ExcelPackage(new FileInfo(jobRequestExcelPath)))
{
    var sayfalar = excelPaketi.Workbook.Worksheets;
    var sayfa1 = sayfalar[0];
    int Y = 3;
    var referansAdiHucresi = sayfa1.Cells[1, 5];
    referansAdiHucresi.Value = Baslangic.ToString("dd.MM.yyyy") + " - " + Bitis.ToString("dd.MM.yyyy");

    foreach (BreakdownTable breakdownTemp in breakdown)...

    excelPaketi.SaveAs(new FileInfo(DosyaYolu));
}

```

Şekil 3.26 : Arızaların Excel formatına çevrilmesi

Arıza ve iş istekleri oluşturulurken, oluşturma bilgilerinin yanında dosya da gönderilmektedir. Oluşan dosya şirketin sunucusundaki bir klasörde depolanmakta ve dosyanın ismi veritabanında ilgili arızada tutulmaktadır. Dosya oluşturulurken, kullanıcı girişinde random sayılar üretmek için kullandığımız fonksiyon yardımıyla dosyanın başına random bir string veri eklenmektedir. Bunun sebebi aynı isimde dosya gönderilirse karışıklık ve olası hataları engellemektir. Veritabanındaki dosya ismiyle, sunucudaki yolda dosyanın varlığı kontrol edilir. Eğer var ise ilgili dosya kullanıcıya File olarak geri döndürülmektedir. İşlemi gerçekleştiren kod bloğu aşağıdaki görselde gösterilmiştir (Şekil 3.27).

```

string dosyaYolu = Path.Combine(ServerDosyaYolu, BreakdownFile);

if (System.IO.File.Exists(dosyaYolu))
{
    byte[] fileBytes = System.IO.File.ReadAllBytes(dosyaYolu);
    string fileName = BreakdownFile;
    return File(fileBytes, System.Net.Mime.MediaTypeNames.Application.Octet, fileName);
}

```

Şekil 3.27 : Arızaların Excel formatında geri döndürülmesi

3.4 Periyodik Bakımlar

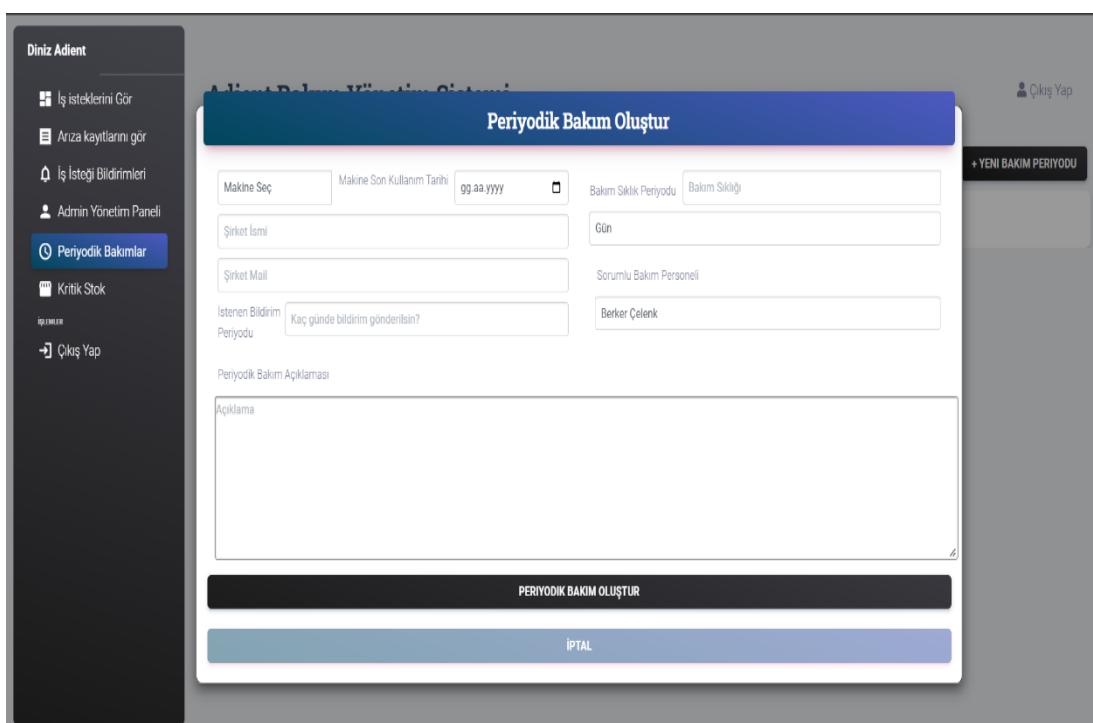
Firmada kullanılan makinelerin, arıza yapılmalarının haricinde, bakım yapılmaları gerekebilmektedir. Bu nedenle uygulama içerisine, periyodik bakımlarının yönetilebileceği bir modül eklenmiştir. Uygulama ana sayfasında sol taraftaki

menüden ‘Periyodik Bakımlar’ linkine tıklandığında, periyodik bakımı gelen makinelerin görüntülendiği ve yeni bakım periyodu oluşturulan sayfaya yönlendirme yapılmaktadır. İlgili sayfa aşağıdaki görselde gösterilmiştir (Şekil 3.28).



Şekil 3.28 : Periyodik bakımların görüntülenme sayfası

Periyodik bakımların görüntülendiği sayfada sağ üstte bulunan ‘+ Yeni Bakım Periyodu’ yazan buton yardımıyla yeni periyodik bakım oluşturulabilen modal açılmaktadır. Açılan modal aşağıdaki görselde gösterilmiştir (Şekil 3.29).



Şekil 3.29 : Yeni periyodik bakım oluşturulması

Periyodik bakımlar oluşturulurken periyodik bakımın hangi makineye uygulanacağına dair makine bilgisi, makinenin başka bir şirkete ait olması durumunda şirket ismi, mail adresi gibi bilgiler ile beraber sorumlu bakım görevlisi, bakımın yapılacaksı periyot, bildirim gönderilmesi istenen süre ve bakım ile ilgili açıklama bilgileri kullanıcidan alınmaktadır. Kullanıcıdan bakım sıklık periyodu olarak bir Integer veri alınmaktadır. Bu bilginin yanında ‘Gün’, ‘Ay’ veya ‘Yıl’ seçeneklerinden birini seçeceği dropdown bulunmaktadır. Örnek vermek gerekirse kullanıcı 3 ve ‘Ay’ seçeneğini girdiğinde makinenin periyodu 3 aylık olarak belirlenecektir. Kullanıcı formu doldurduktan sonra ‘Periyodik Bakım Oluştur’ butonuna tıklayarak periyodik bakımı oluşturabilmektedir. Periyodik bakımların kontrolü için bir adet Windows Console App oluşturulmuştur. Oluşturulan konsol uygulaması, öncelikle veritabanındaki periyodik bakımları tutan tablodan hala aktif olan tüm periyodik bakımları alarak başlamaktadır. Gelen listede dönen bir foreach döngüsü yardımıyla öncelikle periyodik bakımın son zamanı ile şu andaki zamanlaştırılır eğer zamanı geçmiş ise periyodik bakımın aktifliği ile makine tablosundaki ilgili makine bulunarak veritabanı bölümünde bahsedilen ‘IsPeriodicTimeCame’ sütunu 1’den 0’a çekilmektedir. Eğer zaman geçmemiş ise Datetime türünden bir değişken oluşturularak şu anki zaman ile son kontrol yapılan zamanlaştırılır. Son kontrol tarihi yok ise hiç kontrol yapılmadığı anlamına geldiğinden ilk oluşturulma zamanı ilelaştırılır. Böylelikle makinenin en son bakımından beri olan zaman farkı elde edilmiş olmaktadır. Elimizdeki bu bilgi ile, kullanıcının girdiği bilgilerde olan ne kadar sürede bir bakım yapılması gerektigine dair bilgi ilelaştırılır. Kullanıcının seçtiği Integer veri alınarak, eğer ay seçmiş ise 30 ile, hafta seçmiş ise 7 ile çarpılarak bir sayı elde edilmektedir. Elde edilen bu sayı ile zaman farkındaki gün sayısını karşılaştırmakta ve böylelikle makinenin periyodik bakımının gelip gelmediği anlaşılmaktadır. Eğer geldiyse makine tablosundan ilgili makine bulunur ve makinenin IsPeriodicTimeCame sütunu aktif hale getirilir. Ayrıca makine tablosunda bulunan PeriodDayLeft kısmına, kaç gün kaldığı ile ilgili bilgi yazılmaktadır. Yazılan konsol uygulaması Windows’daki görev zamanlayıcı kullanılarak her gün saat 9’da 1 kere çalışacak şekilde ayarlanmıştır. Bu sayede her gün periyodik bakımlar kontrol edilmekte ve makinelerin bakım periyotları sistem üzerinden kontrol edilebilir halde olmaktadır. Konsol uygulamasındaki foreach döngüsüne ait ekran görüntüsü aşağıdaki görselde gösterilmektedir (Şekil 3.30).

```

foreach (var item in allPeriodicChecks)
{
    if (item.ExpirationDate?.Date == DateTime.Now.Date)...

    TimeSpan? zamanFarki;

    if (item.LastCheck != null)
    {
        zamanFarki = DateTime.Now - item.LastCheck;
    }
    else
    {
        zamanFarki = DateTime.Now - item.CreatedAt;
    }

    var zamanFarkiSonKontroldenBeri = zamanFarki?.Days;
    var kontrolYapilmasiGerekenZaman = item.FrequencyTime;

    if (item.FrequencyId == 2) kontrolYapilmasiGerekenZaman *= 7;

    if (item.FrequencyId == 3) kontrolYapilmasiGerekenZaman *= 30;

    if (zamanFarkiSonKontroldenBeri < kontrolYapilmasiGerekenZaman)... 

    // Bildirim yapılması gereken if bloğu -----
    else if (zamanFarkiSonKontroldenBeri >= kontrolYapilmasiGerekenZaman)... 
}

```

Şekil 3.30 : Periyodik bakımların kontrol edimesi

3.5 Kritik Stok

Uygulamada, firma bünyesinde yer alan malzemelerin görüntülenebileceği kritik stok modülü bulunmaktadır. Bu modül sayesinde, iş isteklerinde ve arızalarda kullanılan malzemeler, kritik stok miktarı, güncel stok bilgisi, referans ve seri numarası gibi bilgiler uygulamayı kullanan kullanıcılara gösterilmektedir. Firma bünyesinde bulunan malzemelerin, güncel stok miktarları kritik stok miktarından büyük ise yeşil, eşit ise sarı, düşük ise kırmızı renk ile gösterilmekte böylelikle uygulamayı kullanan görevliler stok miktarını daha rahat kontrol edebilmektedirler. İlgili modül (Şekil 3.31) ve sunucu tarafından çalışan fonksiyona ait kod bloğu (Şekil 3.32) aşağıdaki görsellerde gösterilmiştir.

KRİTİKLIK	REFERANS	SERİ NUMARASI	BULUNAN STOK	KRİTİK STOK	MAKİNE İSMİ	OLUŞTURULMA ZAMANI
1	1003131	3131	30	10	IMBUS HAVSA BASLI CIVATA M6X5 ÇELİK	7/17/2019 12:00:00 AM
1	1003132	3132	10	3	IMBUS HAVSA BASLI CIVATA M6X50 ÇELİK	7/17/2019 12:00:00 AM
1	1003201	3201	50	10	PLASTIK EMNIYETLİ SOMUN M10	7/17/2019 12:00:00 AM
1	1004485	4485	1	1	ZAMAN ROLESİ 24-220VAC ENTES MCB8 0-600K	7/17/2019 12:00:00 AM
1	1004522	4522	30	10	MINYATÜR RÖLÉ PHOENIX-CONTAC REF:2961105 (REL-MR-24VDC/21)	7/17/2019 12:00:00 AM
1	1004557	4557	5	1	BUTON(MANTAR-YAYLI-SİYAH)Ç22 TELEMECANIQUE (011886) REF:XB4 BC21	7/17/2019 12:00:00 AM
1	1004558	4558	5	1	BUTON(MANTAR-YAYLI-YESİL)Ç40 TELEMECANIQUE (088844) REF:ZB4 BC3 + BZ 101	7/17/2019 12:00:00 AM
1	1004568	4568	10	1	BUTON (YAYLI-ISIKLI-BEYZ)Ç22 TELEMECANIQUE (088730) REF:XB4 BW3165 <Ç22-NA+NK>	7/17/2019 12:00:00 AM

Şekil 3.31 : Malzeme stoklarının gösterilmesi

```
public ActionResult CriticalStock()
{
    bool isAuthenticated = IsAuthenticated();

    if (!isAuthenticated)
    {
        ViewBag.title = "Giriş Yap";
        ViewBag.input = "UserCode";
        ViewBag.placeholder = "Adient kullanıcı ismi";

        return View("SignIn");
    }

    int userId = Convert.ToInt32(Session["userId"]);

    using (Bys_Entities1 db = new Bys_Entities1())
    {
        var entityId = db.BYS_Users.FirstOrDefault(y => y.UserId == userId).EntityId;
        var user = db.BYS_Users.FirstOrDefault(x => x.UserId == userId);

        var allMaterialsWithCriticalStocks = db.BYS_Material.Where(x => x.EntityId == entityId
        && x.MaterialCriticalStock != null && x.MaterialQuantity != null).ToList();

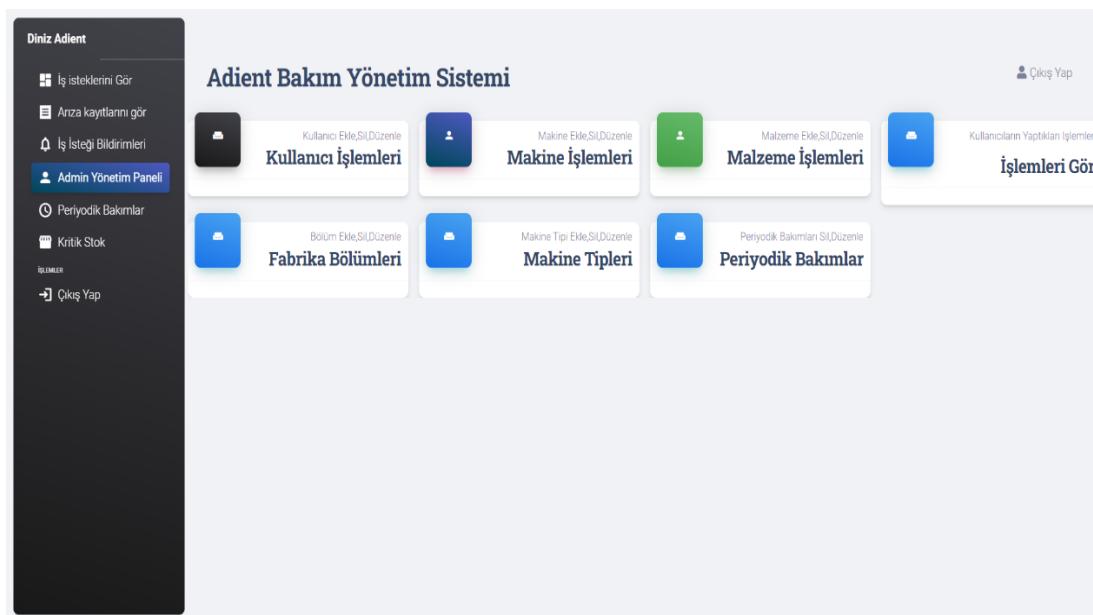
        ViewBag.allMaterialsWithCriticalStocks = allMaterialsWithCriticalStocks;
        ViewBag.activeList = "criticalStock";

        return View();
    }
}
```

Şekil 3.32 : Stok bilgilerinin veritabanından getirilmesi

3.6 Admin Yönetim Paneli

Kullanıcıların, makinelerin, malzemelerin, fabrika bölümlerinin, periyodik bakımların ve makine tiplerinin yönetilmesi için kullanılan, sistemde herhangi bir aksaklılığın olması durumunda elle düzeltme işlemi yapılabilecek bir admin yönetim paneli uygulamaya eklenmiştir. Paneye erişim izni sadece belirli yöneticilere verilir ve erişim izni verilmeyen kullancılara panel gösterilmemektedir. Uygulamadaki sol taraftaki menüden ‘Admin Yönetim Paneli’ linkine tıklandığında ilgili sayfaya (Şekil 3.33) yönlendirme yapılmaktadır.



Şekil 3.33 : Admin yönetim paneli

Panelde öncelikle HTML ve CSS kullanılarak oluşturulmuş 7 adet div bloğu bulunmaktadır. Bloklardan herhangi birine tıklandığında sunucu tarafında çalışan tek bir fonksiyona yönlendirilmektedir. Bu fonksiyona gelen parametre ile hangi işlemin yapılacağı anlaşılmakta ve Razor kullanılarak istemci tarafında kontrol edilerek ilgili veriler gösterilmektedir.

Sunucu tarafında çalışan fonksiyon, gelen parametreyi kontrol eder, parametrelerin eşleşmesi durumunda veritabanına soru atılır ve gelen veri ViewData olarak istemci kısmına gönderilir. İstemci kısmında ise Razor ile kontrol yaparak gelen veri kullanıcıya yansıtılmış olur. Bu sayede tek sayfa ve tek fonksiyon yardımıyla birden fazla işlem yapılabilmektedir. Sunucu tarafında çalışan fonksiyon (Şekil 3.34) ve veritabanına soru atılarak gelen sonucu istemci tarafına gönderen kod bloğu (Şekil 3.35) ve aşağıdaki görsellerde örnek olarak gösterilmiştir.

```

public ActionResult IndexWithTable(string dataToShow)
{
    var isAdmin = IsAdmin();

    if (isAdmin == false || dataToShow == null) return RedirectToAction("Index", "Home");

    ViewBag.activeList = "adminPanel";

    if (dataToShow == "showUsers")...
    if (dataToShow == "showMachines")...
    if (dataToShow == "showMaterials")...
    if (dataToShow == "showDepartments")...
    if (dataToShow == "showMachineTypes")...
    if (dataToShow == "showMachinePeriods")...

    return View("Index");
}

```

Şekil 3.34 : Sunucu tarafında çalışan fonksiyon

```

if (dataToShow == "showUsers")
{
    using (Bys_Entities1 db = new Bys_Entities1())
    {
        int userId = Convert.ToInt32(Session["userId"]);
        var allUsers = (from x in db.BYS_Users
                        where x.UserId != userId
                        join s in db.BYS_Department on x.DepartmentId equals s.DepartmentId
                        join y in db.BYS_Entity on x.EntityId equals y.EntityId
                        join z in db.BYS_Status on x.StatusId equals z.StatusId
                        select new UserTable
                        {
                            UserId = x.UserId,
                            FirstName = x.FirstName,
                            SecondName = x.SecondName,
                            Email = x.Email,
                            IsAdmin = x.IsAdmin == true ? "Admin" : "Normal",
                            IsActive = x.IsActive == true ? "Aktif" : "Aktif Değil",
                            DepartmentName = s.DepartmentName,
                            PhoneNumber = x.PhoneNumber,
                           UserCode = x.UserCode,
                            Entity = y.EntityName,
                            Status = z.Status,
                        }).ToList();
        ViewData["allUsers"] = allUsers;
    }
}

```

Şekil 3.35 : Veritabanına soru atılıp istemciye gönderilmesi

Admin kullanıcı tarafından tüm kullanıcılar bir tablo halinde görüntülenebilmektedir. Tabloda kullanıcıya ait isim, soyisim, kullanıcı kodu, email gibi bilgiler yer almaktadır. Ayrıca gösterilen tabloda ‘Tüm Kullanıcılar’, ‘Aktif Kullanıcılar’ ve ‘İnaktif Kullanıcılar’ olmak üzere filtreleme seçeneği de bulunmaktadır. Form yardımıyla sunucuya istek atılır ve veritabanına atılan sorgu değiştirilerek sayfa yenilenir. Bu işlemler sonucunda filtreleme işlemi gerçekleşmiş olur. İlgili tablo aşağıdaki görselde gösterilmektedir (Şekil 3.36).

#USERID	ISIM	SOYISIM	KULLANICI KODU	BÖLÜM	AKTİFLİK	EMAIL	TELEFON	FABRİKA	ADMINLİK	STATÜ
4	Berk	Soğukpinar	asogukb	Bilgi İşlem	AKTİF	berk.sogukpinar@adient.com	5554443322	Gönen	Normal	Requester
5	Erdem	Polat	apolate	Bilgi İşlem	AKTİF	erdem.polat@adient.com	3334444	Gönen	Normal	Chief
6	Sedat	Alan	aalans	Bilgi İşlem	AKTİF	sedat.alan@adient.com	123345	Gönen	Normal	Manager
9	Ali	Türker	aturkeal	Bilgi İşlem	AKTİF	ali.turker@adient.com	0539293290	Gönen	Normal	Chief
14	Hüseyin	Babuc	babuc	Bilgi İşlem	AKTİF	huseyin.babuc@adient.com		Gönen	Admin	Requester

Şekil 3.36 : Tüm kullanıcıların gösterilmesi

Tablo üzerinde gezinirken CSS yardımıyla ilgili satır renklendirilmekte ve tıklanlığında ise kullanıcı bilgilerinin bulunduğu sayfaya yönlendirilme yapılmaktadır. İlgili sayfa aşağıdaki görselde gösterilmektedir (Şekil 3.37).

Şekil 3.37 : Kullanıcı bilgilerinin gösterilmesi

Kullanıcı bilgilerinin gösterildiği ekranda kullanıcı düzenlenebilmekte veya silinebilmektedir. Kullanıcı sil butonuna tıklandığında form yardımıyla sunucuya istek atılır ve ilgili kullanıcının aktifliği false değerine çekilir. Kullanıcı düzenleme butonuna tıklandığında ise kullanıcının bilgilerinin girili olduğu bir formun olduğu sayfa görüntülenmektedir. İlgili sayfa aşağıdaki görselde gösterilmektedir (Şekil 3.38).

Şekil 3.38 : Kullanıcı düzenleme formu

İlgili form doldurularak kullanıcının güncelleme işlemi tamamlanmaktadır. Kullanıcının bulunduğu bölüm, fabrika ve statü durumu gibi bilgiler iş isteklerindeki aktarım işlemleri için özellikle önem arz etmektedir.

Kullanıcıların görüntülendiği sayfada ‘Kullanıcı Ekle’ butonuna tıklandığında kullanıcı ile ilgili bilgilerin girilebileceği formu içeren bir sayfa gösterilmektedir. Kullanıcı bilgileri girildikten sonra ‘Kullanıcı Oluştur’ butonuna tıklanarak kullanıcı oluşturma işlemi tamamlanmaktadır. İlgili sayfa aşağıdaki görselde gösterilmektedir (Şekil 3.39).

Şekil 3.39 : Kullanıcı ekleme formu

Admin yönetim paneli içerisinde ‘Makine İşlemleri’ bloğuna tıklandığında firma bünyesinde bulunan makineler görüntülenmektedir. Makineler ile ilgili makine ismi, numara, bakım periyot bilgisi gibi bilgiler bulunmaktadır. Gösterilen tabloda istenilen makineye tıklanılarak makine ile ilgili bilgilerin görüntülenmesi sağlanabilmektedir. Fabrika bünyesindeki tüm makinelerin görüntülenmesi (Şekil 3.40) ve bir makine ile ilgili bilgilerin görüntülenmesi (Şekil 3.41) ile ilgili görseller aşağıda gösterilmiştir.

MAKİNE İSMİ	NUMARA	TİPİ	OLUSTURAN	AKTİFLİK	OLUSTURULMA TARİHİ	BAKIM PERİYODU	FABRİKA
Düz Dikiş Makinası	DE1001	Dikiş	Batuhan Avcı	AKTİF	7/17/2019 12:00:00 AM	6 Ay	Gönen
Düz Dikiş Makinası	DE1002	Dikiş	Batuhan Avcı	AKTİF	7/17/2019 12:00:00 AM	6 Ay	Gönen
Düz Dikiş Makinası	DE1003	Dikiş	Batuhan Avcı	AKTİF	7/17/2019 12:00:00 AM	6 Ay	Gönen
Düz Dikiş Makinası	DE1004	Dikiş	Batuhan Avcı	AKTİF	7/17/2019 12:00:00 AM	6 Ay	Gönen
Düz Dikiş Makinası	DE1005	Dikiş	Batuhan Avcı	AKTİF	7/17/2019 12:00:00 AM	6 Ay	Gönen

Şekil 3.40 : Tüm makinelerin görüntülenmesi

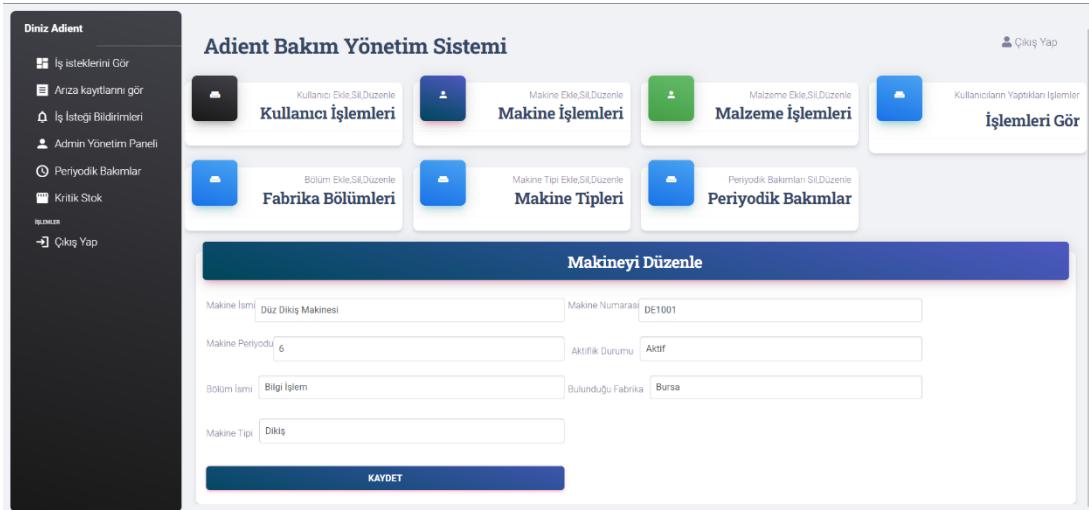
Düzeltenme Tarihi: 7/17/2019 12:00:00 AM - Batuhan Avcı

Fabrika: Gönen
Makine Bakım Periyodu: 6 Ay
Aktiflik: Aktif

Şekil 3.41 : Makine bilgilerinin gösterilmesi

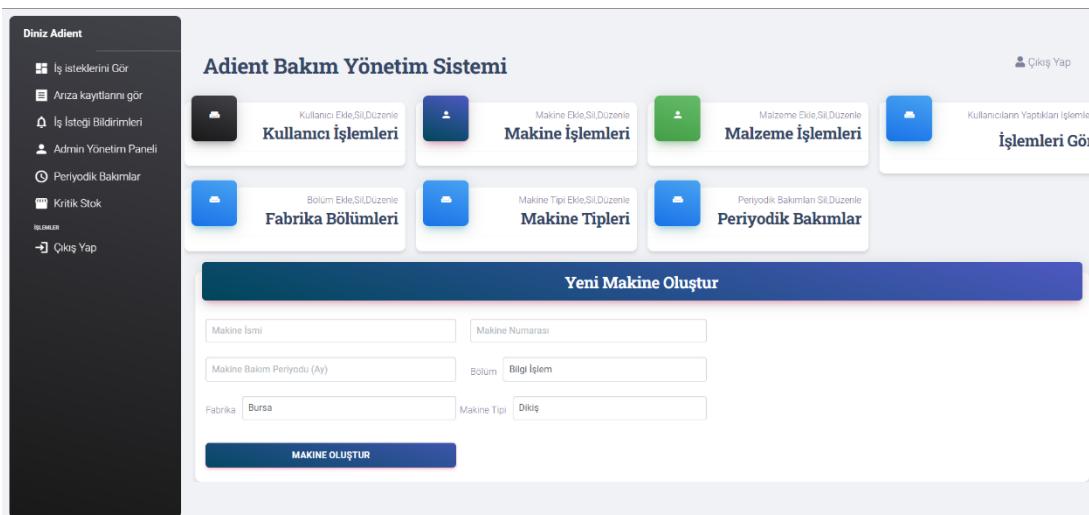
Makine bilgileri gösterilirken ‘Makineyi Sil’ butonuna tıklandığında bir form yardımıyla sunucu tarafında çalışan bir fonksiyona istek gönderilerek ilgili makinenin aktifliği false duruma çekilmektedir. ‘Makineyi Düzenle’ butonuna tıklanıldığında ise kullanıcı işlemlerinde olduğu gibi, düzenlenecek makine bilgileri ile dolu olan bir form

gösterilmektedir. Kullanıcıya gösterilen makine düzenleme sayfası (Şekil 3.42) aşağıdaki görselde gösterilmektedir.



Şekil 3.42 : Makine bilgilerinin güncellenmesi

Makinelerin gösterildiği sayfada ‘Makine Ekle’ butonuna tıklanıldığında makine ekleme formu kullanıcıya gösterilmektedir. Makine ile ilgili bilgiler girildikten sonra ‘Makine Oluştur’ butonuna tıklayarak yeni makinenin sisteme eklenmesi sağlanmaktadır (Şekil 3.43).



Şekil 3.43 : Yeni makine oluşturulması

Admin yönetim panelinde, firma içerisinde bulunan malzemeler yönetilebilmektedir. Panel üzerindeki ‘Malzeme İşlemleri’ bloğuna tıklandığında, admin kullanıcıya tüm malzemelerin listesi firma bünyesinde yaklaşık 60.000 adet malzeme bulunduğundan gösterilememektedir. Bunun yerine ilk 300 adet makine gösterilip, diğer malzemeler arasından arama yapılmasına olanak sağlayan bir sistem geliştirilmiştir. Kullanıcı

aramak istediği malzemenin malzeme ismini girerek arama yapabilmektedir. İstemci ekranında gösterilen ekran (Şekil 3.44) aşağıda gösterilmiştir.

MAZNE REFERANS	SERİ NUMARASI	İSMİ	KRİTİKLIK	AKTİFLİK	OLUŞTURAN	OLUŞTURMA ZAMANI	KRİTİK STOK	BULUNAN SAYI
1000001	1	CONTA L1 16X16X7 SKT REF:40299	Normal	AKTİF	Batuhan Avcı	7/17/2019 12:00:00 AM		
1000002	2	CONTA B2 8X22X8 SIMRİT (YAZI 069/04 HEKİ) REF:BA8X22X7EN72NBR902	Normal	AKTİF	Batuhan Avcı	7/17/2019 12:00:00 AM		
1000003	3	CONTA L3 10X19X7 SKT REF:40171	Normal	AKTİF	Batuhan Avcı	7/17/2019 12:00:00 AM		
1000004	4	CONTA BA 10X22X7 SKT REF:40247	Normal	AKTİF	Batuhan Avcı	7/17/2019 12:00:00 AM		

Şekil 3.44 : Malzemelerin gösterilmesi

Kullanıcı malzeme ismini girdiği zaman sunucu kısmında bulunan bir fonksiyona bir form yardımıyla girilen input gönderilmektedir. Sunucu tarafında, istemciden gelen malzeme ismini, veritabanında içerenleri arayacak şekilde sorgu atılmaktadır. Daha sonra gelen verilerden 100 adeti kullanıcıya geri döndürülmektedir. Veritabanına atılan soruyu içeren kod bloğu (Şekil 3.45) aşağıda gösterilmiştir.

```
allMaterials = (from x in db.BYS_Material
                join s in db.BYS_Users on x.MaterialCreatedById equals s.UserId
                join y in db.BYS_Entity on x.EntityId equals y.EntityId
                where x.MaterialDescription.Contains(searchInput)
                select new MaterialTable
                {
                    MaterialId = x.MaterialId,
                    MaterialSerialNumber = x.MaterialSerialNumber,
                    MaterialReference = x.MaterialReference,
                    MaterialCreatedBy = s.FirstName + " " + s.SecondName,
                    MaterialCreatedTime = x.MaterialCreatedTime,
                    IsActive = x.IsActive == true ? "Aktif" : "Aktif Değil",
                    MaterialDescription = x.MaterialDescription,
                    MaterialCriticalStock = x.MaterialCriticalStock,
                    MaterialQuantity = x.MaterialQuantity,
                    IsCritical = x.IsCritical == true ? "Kritik" : "Normal",
                    Entity = y.EntityName
                }).Take(100).ToList();
```

Şekil 3.45 : Aranan malzemelerin veritabanından getirilmesi

Veritabanından gelen veriler istemcide tablo ile gösterilmektedir. Bu şekilde çok sayıda malzemenin görüntülenmesi ile ilgili sıkıntı çözülmüş olmaktadır.

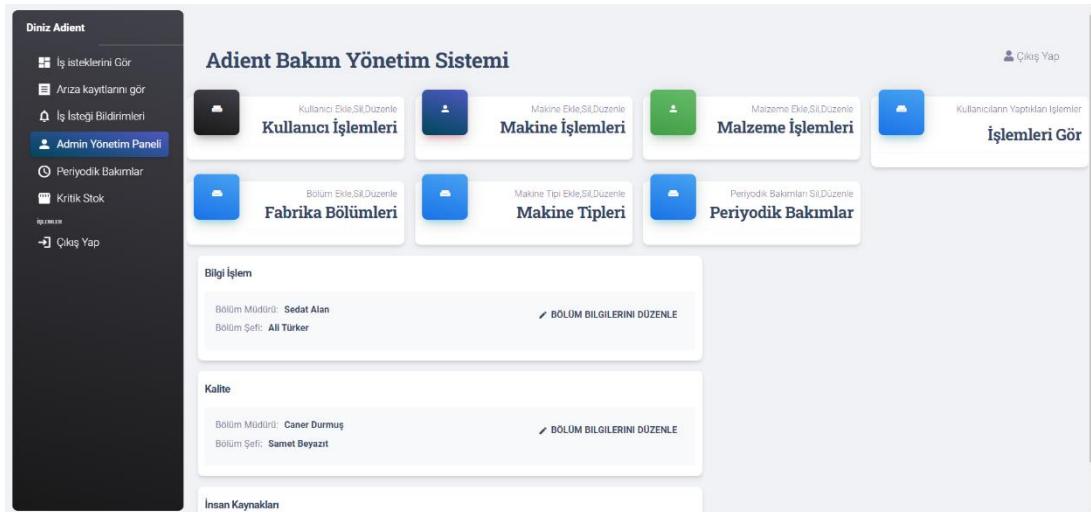
Kullanıcı malzemelerin görüntülendiği sayfadaki tabloda istediği malzemenin üstüne tıklayarak malzeme ile ilgili detayların görüntülendiği sayfaya (Şekil 3.46) yönlendirilir. Malzeme detayları sayfasında ise kullanıcılar ve makinelerde olduğu gibi malzeme bilgilerinin girili olduğu bir formu içeren düzenleme sayfasına (Şekil 3.47) yönlendirme işlemi yapılmaktadır.

Şekil 3.46 : Malzeme detaylarının gösterilmesi

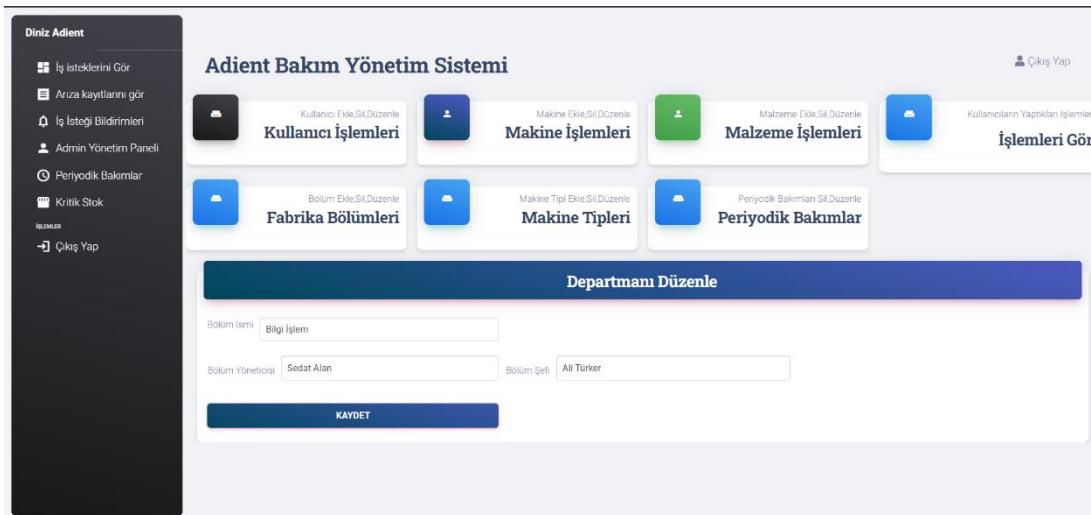
Şekil 3.47 : Malzeme düzenlemesinin yapılması

Admin kullanıcı aynı zamanda fabrika da bulunan bölümleri de görüntüleyebilmektedir. Admin yönetim panelinde ‘Fabrika Bölümleri’ bloğuna

tıklandığı zaman kullanıcıya fabrikada bulunan bölümler gösterilmektedir. Fabrikadaki bölümler admin kullanıcı tarafından düzenlenebilmektedir. Bölümlerin kullanıcıya gösterildiği ekran (Şekil 3.48) ve bölüm bilgilerinin düzenlenebildiği ekran (Şekil 3.49) aşağıdaki görsellerde gösterilmiştir.



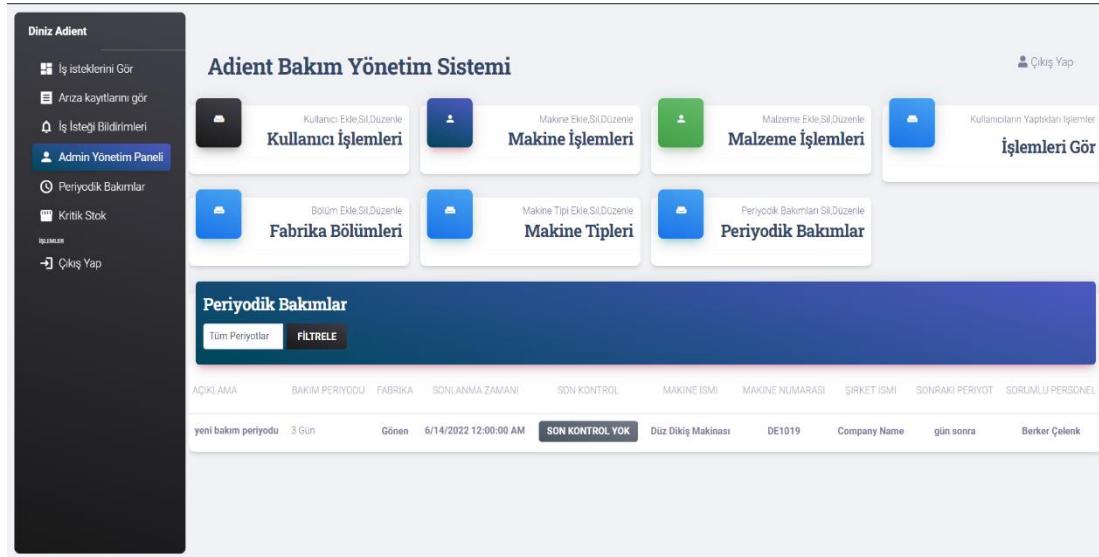
Şekil 3.48 : Fabrika bölümlerinin gösterilmesi



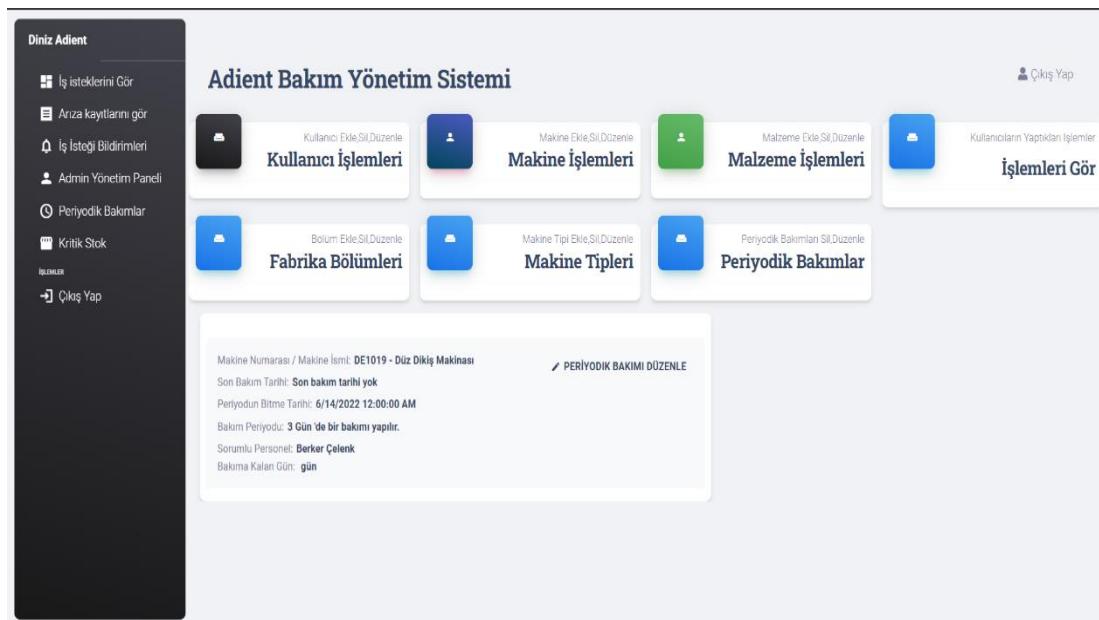
Şekil 3.49 : Bölüm bilgilerinin düzenlenmesi

Admin yönetim panelinin son kısmında ise, admin kullanıcı oluşturulmuş olan periyodik bakımları yönetebilmektedir. Panel üzerinden ‘Periyodik Bakımlar’ bloğuna tıklandığında veritabanında aktif olan tüm periyodik bakımlar kullanıcıya gösterilmektedir. Gösterilen tabloda bakım periyodu, fabrika, periyodun sonlanma zamanı, son kontrol zamanı, makine ismi gibi bilgiler yer almaktadır. Kullanıcı istediği periyodik bakımın üstüne gelerek periyodik bakım ile ilgili bilgilerin olduğu sayfaya gidebilmektedir. Tüm periyodik bakımların görüntüülendiği sayfa (Şekil 3.50) ile

periyodik bakım bilgilerinin görüntüülendiği sayfa (Şekil 3.51) aşağıdaki görsellerde gösterilmiştir.



Şekil 3.50 : Tüm periyodik bakımların görüntülenmesi



Şekil 3.51 : Periyodik bakım bilgilerinin görüntülenmesi

Admin kullanıcı periyodik bakım bilgilerinin görüntüülendiği sayfadan, periyodik bakımı düzenleme sayfasına gidebilmektedir. Diğer kullanıcı, makine ve malzeme seçeneklerinde gösterildiği gibi, seçilen periyodik bakım ile ilgili bilgilerin doldurulmuş olduğu bir form bulunan sayfa kullanıcıya gösterilmektedir. Kullanıcı formu istediği gibi güncelliyerek sunucuda çalışan bir fonksiyona POST metoduyla istek atmakta ve veritabanına ilgili periyodik bakımı güncelleme işlemi yapılmaktadır. Periyodik bakımın güncelleme formu (Şekil 3.52) aşağıdaki görselde gösterilmiştir.

Diniz Adient

- İş isteklerini Gör
- Anza kayıtlarını gör
- İş İsteği Bildirimleri
- Admin Yönetim Paneli
- Periyodik Bakımlar
- Kritik Stok
- İŞLEMLER**
- Çıkış Yap

Adient Bakım Yönetim Sistemi



Kullanıcı Ekle,Sil,Düzenle
Kullanıcı İşlemleri



Makine Ekle,Sil,Düzenle
Makine İşlemleri



Malzeme Ekle,Sil,Düzenle
Malzeme İşlemleri



Bölüm Ekle,Sil,Düzenle
Fabrika Bölgümleri



Makine Tipi Ekle,Sil,Düzenle
Makine Tipleri



Periyodik Bakım Ekle,Sil,Düzenle
Periyodik Bakımlar

Periyodu Düzenle

Periyot Açıklaması:

Şirket Maili:

Makine Periyot Zamanı: Gün

Aktiflik durumu:

Şirket İsmi:

Bildirim gönderilme süresi (gün):

Fabrika:

Şekil 3.52 : Periyodik bakım bilgilerini güncelleme formu

4. MAKİNE ÖĞRENMESİ

ASP.NET MVC uygulamasının geliştirme kısmının tamamlanmasının ardından projenin Makine öğrenmesi tarafının geliştirilmesi aşamasına geçilmiştir. Danışman Dr.Öğr.Üyesi Mustafa Özgür Cingiz ile yapılan toplantıda Bakım Yönetim Sistemi’nde kullanılan makinelerin arızalanma zamanının tahmin edilmesi hususunda gerekli görüşme yapılmış olup Danışman Dr.Öğr.Üyesi Musfata Özgür Cingiz'in verdiği kaynaklar doğrultusunda Makine Öğrenmesi ile ilgili araştırmalar yapılmıştır. Araştırmaların sonucunda Python dilinde Tensorflow'un Keras kütüphanesi ile bu işlemin gerçekleştirilemesi konusunda karar alınmıştır.

İlk olarak SQL Server'da bulunan ve 2016 yılından bu yana kaydı tutulan makinelerin bozulma zamanları ile ilgili tablonun özellikleri Şekil 4.1'de verilmiştir.

Column Name	Data Type	Allow Nulls
Bakim_Ticket_Ariza_Id	int	<input type="checkbox"/>
Bakim_Ticket_Makine_Id	int	<input checked="" type="checkbox"/>
Bakim_Ticket_Ariza_Baslangic	datetime	<input checked="" type="checkbox"/>
Bakim_Ticket_Ariza_Bitis	datetime	<input checked="" type="checkbox"/>

Şekil 4.1 : Tahmin değerlerini kaydetmek için SQL tablosu

Şekil 4.1'da verilen görselde makine öğrenmesi kısmında makinelerin bir sonraki arızalanma zamanlarının tahmin edilmesinde kullanılacak verilerin özellikleri verilmiştir. Burada önemli olan kısım Arızanın başlangıç ve bitiş tarihlerinin biliniyor olmasıdır. Python tarafında Keras kütüphanesi ile ilgili araştırmalar yapılrken modellere verilen değerlerden bir tanesinin zaman tahmin edilecek değerin ise bir sayı olması gereği öğrenildikten sonra Danışman Dr.Öğr.Üyesi Mustafa Özgür Cingiz ile yapılan görüşme sonucunda bir makinenin arıza başlangıç zamanları arasındaki farkı alarak makinenin bir sonraki arızalanma zamanının tahmin edilmesinin uygun olduğu sonucuna varıldı.

Bu görüşmenin Keras kütüphanesine verileri aktarabilmek adına C# dilinde bir .csv dosyanın oluşturulması kararına varılmış olup bu .csv uzantılı dosyanın içerisinde kayıtlı olan tüm makinelerin bozulma zamanları, MakineId ve bir makine için oluşan iki arıza arasındaki Fark Sayısı bilgisinin tutulması amaçlanmıştır. Ardından Bakim_Ticket_Ariza tablosundaki tüm verilerin önce .xlsx ardından da .csv uzantılı

dosya formatına dönüştürülebilmesi adına C#'da .NET Framework kullanılarak bir Console Application oluşturulmuştur. Burada yapılmak istenen işlem Bakim_Ticket_Ariza tablosundaki tüm verilerin Bildirim Tarihi, Fark Sayısı ve MakineId bilgisi şeklinde bir excel dosyasına yazdırılıp .csv uzantılı dosya formatına dönüştürülmesini sağlamaktır. İlk olarak bu işlemin gerçekleştirilebilmesi için veritabanı bağlantısı Entity Framework ile sağlanmış olup Bakim_Ticket_Ariza tablosu Console Application'a eklenmiştir. Bu işlemin ardından tüm kayıtlar içinde gezebilmek adına while döngüsünden faydalananmiş olup Şekil 4.2'de verilen görselde;

```

int machineCount      = 1;
String DosyaYolu     = "";
Excel.Workbook wb     = null;
Excel.Worksheet ws    = null;
int excelRowCount     = 3;
string tempPath       = Path.GetTempFileName();
int machineListIndex  = 0;
bool oneCreateXlsx    = true;

```

Şekil 4.2 : Excel dosyasına atma işleminde değişkenler

Bu işlemde hem tüm makineleri hem de makinelerin birden fazla arızalanma durumunda arıza bilgilerine erişebilmek adına while döngüsünde kullanılması adına machineCount ve machineListIndex değişkenleri oluşturulmuştur. machineCount değişkeni makine sayısını kontrol ederken machineListIndex makinelerin liste elemanlarının tümünün gezilmesi imkanını sağlayacaktır. oneCreateXlsx değişkeni bool tipinde tutulmuş olup Şekil 4.3'de verilen görselde;

```

if (oneCreateXlsx == true)
{
    DosyaYolu      = Environment.GetFolderPath(Environment.SpecialFolder.MyVideos) + "\\MakineArızalanma.xlsx";
    File.WriteAllBytes(tempPath, Properties.Resources.emptyExcel);
    Excel.Application excel = new Excel.Application();
    wb             = excel.Workbooks.Open(tempPath);
    ws             = wb.Worksheets[1];
    oneCreateXlsx  = false;
}

```

Şekil 4.3 : Tek bir Excel dosyasında tüm kayıtların tutulması

Makinelerin arıza bilgilerinin tek bir .xlsx uzantılı dosyada tutulmasını sağlamak adına while döngüsünde bir kere oluşturulmasını sağlamak için oluşturulmuş değişkendir.

Bu işlemin ardından da makinelerin arızalanma günleri arasındaki fark sayısını hesaplamak adına Şekil 4.4'de verilen görselde;

```
var differenceDayCount = machine[machineListIndex + 1].Bakım_Ticket_Ariza_Baslangic - machine[machineListIndex].Bakım_Ticket_Ariza_Baslangic;

if (differenceDayCount?.Days != 0)
{
    if (machine[machineListIndex].Bakım_Ticket_Ariza_Baslangic != null)
    {
        ws.Cells[1][excelRowCount] = ((DateTime)machine[machineListIndex].Bakım_Ticket_Ariza_Baslangic).ToString("dd.MM.yyyy");
    }

    if (differenceDayCount != null)
    {
        ws.Cells[2][excelRowCount] = differenceDayCount?.Days;
    }

    if (machine[machineListIndex].Bakım_Ticket_Makine_Id != null)
    {
        ws.Cells[3][excelRowCount] = machine[machineListIndex].Bakım_Ticket_Makine_Id;
    }
    excelRowCount++;
}
machineListIndex++;

if (machineListIndex + 1 >= machine.Count)
{
    machineCount++;
    machineListIndex = 0;
    continue;
}
```

Şekil 4.4 : Tüm Makineler İçin Arızalanma Zamanları Farkı'nın Alınması

İlk olarak bir makinenin arızalanma zamanları arasındaki farkı her bir makine için ayrı ayrı yapıp ardından bu bilgileri tek tek .xlsx uzantılı dosyaya kaydetme işlemi gösterilmiştir. Eğer o makine için tüm kayıtlar .xlsx uzantılı dosyaya eklenmişse makine sayısını kontrol eden machineCount değişkeni 1 arttırılarak diğer makineye geçilmesi sağlanmış olup machineListIndex ise makinenin arıza kayıtları içinden ilk elemanı alarak başlaması gerektiğinden dolayı sıfır eşitlenir. Bu işlemlerin ardından son olarak tüm makinelerin kayıtları .xlsx uzantılı dosyaya kaydedilir ve ardından Şekil 4.5'de verilen görseldeki kod parçasında;

```
var workbook = new Workbook(DosyaYolu);
workbook.Save(Environment.GetFolderPath(Environment.SpecialFolder.MyVideos) + "\\MakineArızalanma.csv");
```

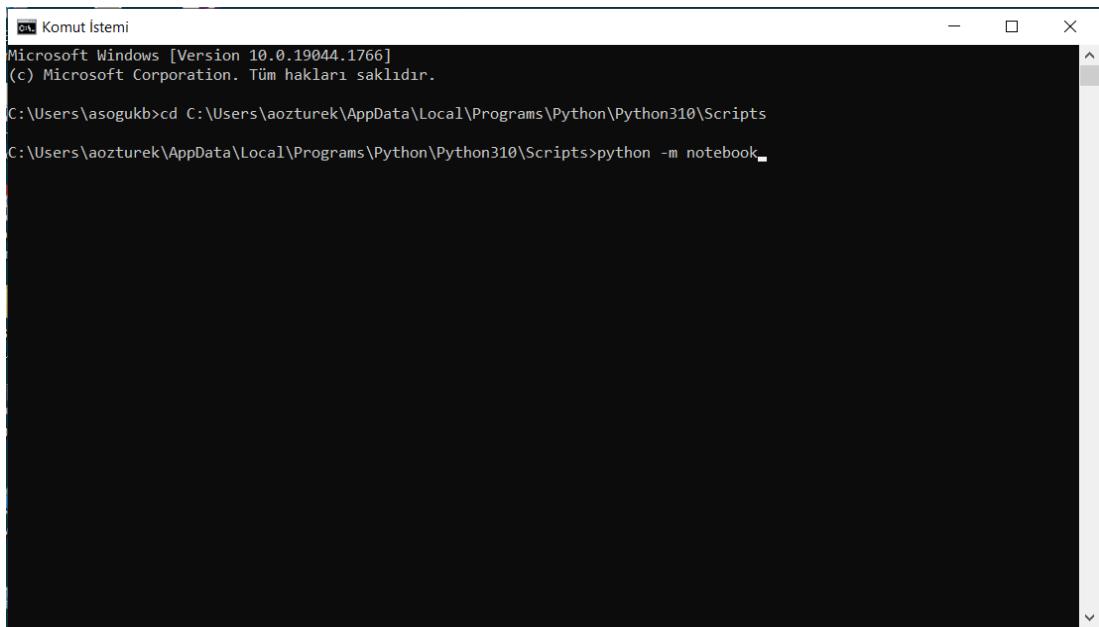
Şekil 4.5 : CSV formatına aktarım işlemi

Oluşturulmuş .xlsx uzantılı dosya döngü kırıldıktan sonra .xlsx dosyasının oluşturulduğu gibi bir defa oluşturulurak tüm kayıtların .csv uzantılı dosyaya aktarılması sağlanmıştır.

Bu işlemin ardından Python tarafından Keras kütüphanesi ile ilgili araştırmalara başlanmıştır. Danışman Dr.Öğr.Üyesi Mustafa Özgür Cingiz ile yapılan toplantı sonucunda Keras kütüphanesinin LSTM ve GRU modellerinin ayrı ayrı kontrol

edilmesi gerektiği bilgisi alındıktan sonra LSTM ve GRU modelleri ile ilgili araştırmalar yapılmış olup ilk olarak LSTM ve GRU modellerinin neden kullanıldığı anlaşılmış olup ardından iki model arasındaki farklar öğrenilmiştir. Bu öğrenimler ardından Keras kütüphanesi kullanılırken hem LSTM modeli eğitilerek hem de GRU modeli eğitilerek işlemler yaptırılıp grafikler oluşturulacak ve de sonuçlar ilerleyen kısımda gösterilecektir.

İlk olarak python tarafında projeyi geliştirebilmek adına Jupyter Notebook'a erişim sağlanması gereklidir. Şekil 4.6'daki görselde Jupyter Notebook'a giriş komutu verilmiştir.



Şekil 4.6 : Jupyter Notebook çalıştırılma komutu

Bu işlemin ardından Jupyter çalıştırılır ve uygulama geliştirme aşamasına geçilir. Python'da uygulamayı geliştirmek adına import edilmesi gereken kütüphaneler Şekil 4.7'de verilmiştir.

```
In [434]: import numpy
import matplotlib.pyplot as plt
import pandas
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM,GRU
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
```

Şekil 4.7 : Kullanılan kütüphaneler

Şekil 4.7'de verilen görselde Keras kütüphanesinin modelleri LSTM ve GRU'nun kullanılması adına import edilmesi gereklidir. MinMaxScaler ise minimum ve maksimum gözlemlenebilir değerlerin değerlerin eğitim verileri kullanılarak tahmin edilmesini sağlayacaktır. Mean Squared Error ise tahmin edilen değerin hata miktarını hesaplamaya yarar. Import işleminden sonra oluşturulan .csv uzantılı makinelerin arızalanma zamanları arasındaki fark sayısını, arızalanma zamanlarını ve MakineId bilgilerini tutan dosyanın içindeki verilerin okunması kısmına geçilir. Bu kısımda model eğitilirken kullanılacak yani tahmin edilecek veri arızalanma zamanlarındaki fark sayıları olduğundan .csv uzantılı dosyadan o verinin çekilip boş bir listeye atılması sağlanmış ve listenin elemanları makinenin arıza zamanları arasındaki fark sayısı değerleri ile doldurulmuştur. Şekil 4.8'de bu işlemi gerçekleştiren kod parçası verilmiştir.

```
In [436]: dataframe = pandas.read_csv('C:\\\\Users\\\\asogukb\\\\Videos\\\\MakineArizalanma2.csv',engine='python',sep=";")

In [437]: data_list = []

In [438]: machineId = 171

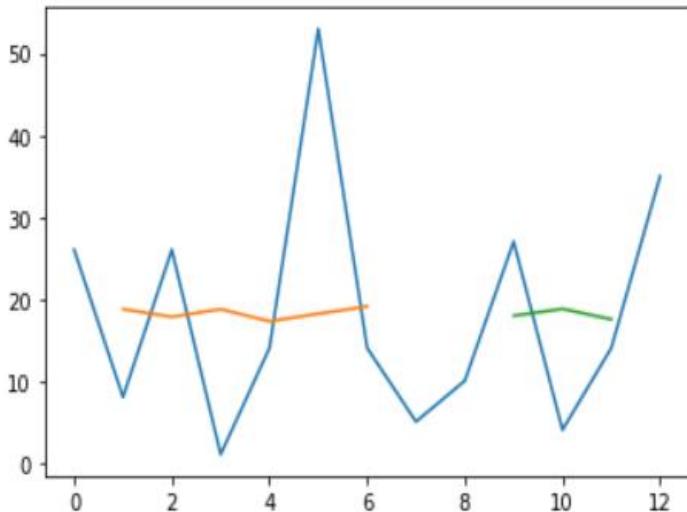
In [439]: for row in dataframe.iterrows():
    if(row[1].MakineId == machineId):
        data_list.append(row[1].FarkSayisi)

    print(data_list)
    print(len(data_list))

[1, 3, 1, 1, 1, 8, 3, 1, 24, 4, 3, 1, 8, 24, 52, 24, 34, 7, 6, 3, 3, 2, 10, 1, 2, 2, 9, 9, 11, 14, 1, 27, 42, 25, 2, 11, 1,
21, 5, 1, 17, 3, 10, 1, 7, 1, 5, 3, 2, 10, 6, 17, 3, 9, 7, 6, 3, 25, 5, 63, 5, 4, 2, 11, 5, 89, 68, 5, 113]
70
```

Şekil 4.8 : Arıza Fark Sayılarını CSV Dosyadan Okuma

Bu kısımda ilk başta yazılan kod parçasında Şekil 4.8'de görüldüğü gibi kod çalıştırılırken makinenin Id bilgisi elle girilmelidir. Ancak karşılaşılan bir sorundan ve işlemin dinamik gerçekleşmemesinden kaynaklı projede değişikliğe gidilmiştir. Yapılan çalışmalar sonucunda arızalanma sayısı 50'den az olan makinelerde model eğitilmiş ve grafik çizdirilmiş olup veri setinin eleman sayısının az olmasından kaynaklanan ve Şekil 4.9'de verilen grafik üzerinde olduğu gibi;



Şekil 4.9 : Data Sayısı Az Olan Makinenin Tahmin Değerleri

Grafik hatalı bir şekilde çizdirilmiştir. Bunun sebebi Danışman Dr.Öğr.Üyesi Mustafa Özgür Cingiz ile toplantı gerçekleştirildikten sonra veri setinin eleman sayısının az olmasından dolayı grafiğin hatalı çıkması sonucuna varılmış olup 50 adet arıza kaydı bulunmayan makineler için bu işlemin gerçekleştirilmemesine karar verilmiştir. Ayrıca bu grafiklerin de dinamik olarak çizilmesi, sonrasında kullanmak adına SQL Server'da kaydedilmesi ve arızalanma sayısı 50'den fazla olan makinelerin bulunması adına yeni bir .csv uzantılı dosya oluşturmanın mantıklı olduğu kanısına varılmıştır.

Tüm kayıtların Python tarafından kullanılmasını sağlayan .csv dosyanın oluşturulmasına benzer işlemler yapılmış olup Şekil 4.10'daki görselde;

```

if (machine.Count != 0 && machine.Count >= 50)
{
    ws.Cells[1][excelRowCount] = machine[0].Bakim_Ticket_Makine_Id;
    ws.Cells[2][excelRowCount] = machine.Count;
    excelRowCount++;
}

machineCount++;

```

Şekil 4.10 : Arıza Kaydı 50'den Fazla Olan Makinelerin Kontrolü

Bakim_Ticket_Ariza tablosundan bozulma sayıları 50'den fazla olan makinelerin excel dosyasına kaydedilmesi işlemi Şekil 4.10'daki kod parçasında verilmiştir. Bu işlemin sonucunda olan excel dosyasının görünümü Şekil 4.11'da verilmiştir.

	Makinelid	BozulmaSayısı
2		
3	9	93
4	17	454
5	59	198
6	62	56
7	71	52
8	72	54
9	73	122
10	109	76
11	110	177
12	113	99
13	114	161
14	122	448
15	123	140
16	124	649
17	125	78
18	126	162
19	128	51
20	129	76
21	130	142
22	137	140
23	139	80
24	140	87
25	161	282
26	162	179
27	163	184
28	164	221
29	165	308

Şekil 4.11 : Arıza Kaydı 50'den Fazla Olan Makinelerin Bozulma Sayıları

Şekil 4.11'daki görselde 50'den fazla arızalanan makinelerin MakineId ve arızalanma sayısı bilgileri verilmiştir. İşlemlere devam ederken diğer oluşturulan ve tüm kayıtların olduğu .csv uzantılı dosya ile 50'den fazla arızalanan makinelerin Id bilgisinin bulunduğu .csv uzantılı dosyanın bilgileri MakineId yardımı ile sadece 50'den fazla arızalanan makineler ile ilgili işlemlerin yapılmasını sağlayacak olup böylece 50'den fazla kullanılacak olan .csv dosyasının da kullanılması adına Şekil 4.12'de verilen kod parçası python tarafına eklenmiştir.

```
In [271]: dataframe = pandas.read_csv('C:\\\\Users\\\\asogukb\\\\Videos\\\\MakineArizalanma1.csv',engine='python',sep=";")
dataFrameCount = pandas.read_csv('C:\\\\Users\\\\asogukb\\\\Videos\\\\MakineArizalanmaSayisi.csv',engine='python',sep=";")

In [272]: data_list_count = []

In [273]: for row in dataFrameCount.iterrows():
    data_list_count.append(row[1].MakineId)
print(len(data_list_count))

56
```

Şekil 4.12 : Oluşturulan İki CSV Dosyasının Alınması

Şekilde verilen kod parçasında 50'den fazla arızalanma kaydı olan makinelerin MakineId bilgilerinin boş bir listenin içine atılması verilmiştir. Bu işlemin amacı 56 ayrı makine için tahmin işleminin dinamik olarak gerçekleşip tahmin değerlerinin SQL Server'da kaydedilmesini sağlamaktır. İlk oluşturulan ve tüm kayıtların bulunduğu .csv dosyasındaki MakineId ile bu listedeki MakineId bilgisi aynı olduğunda arızalanma fark gün sayıları alınacak ve model eğitilerek bir tahmin değeri elde edilecektir. İlk olarak arıza kaydı 50'den fazla olan tüm makineler için tahmini değerleri dinamik bir şekilde oluşturabilmek adına Şekil 4.13'deki kod parçası verilmiştir.

```
In [274]: for i in range(len(data_list_count)):
    data_list = []
    for row in dataframe.iterrows():
        if(row[1].MakineId == data_list_count[i]):
            data_list.append(row[1].FarkSayisi)
    data_list= numpy.array(data_list).reshape(-1,1)
    dataset = data_list.astype('float32')
    scaler = MinMaxScaler(feature_range=(0, 1))
    dataset = scaler.fit_transform(dataset)

    train_size = int(len(dataset) * 0.67)
    test_size = len(dataset) - train_size
    train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
```

Şekil 4.13 : Her Makinenin Tahmin Değerlerinin Dinamik Gerçekleştirilmesi

Şekilde görülen kod parçasında iç içe for döngüsü kullanılmış olup her bir makinenin arızalanma tahmini değerleri bulunup SQL Server'da kaydedildikten sonra diğer makineye geçilmesini sağlar. Bu kısımda oluşturulan listede arızalanma zamanlarının fark gün sayılarının değerleri olup bir sinir ağı modellemesi olacağinden MinMaxScaler kullanılarak ölçekleme yapılmış olup değerler 0-1 aralığına güncellenmiştir. Bu işlemde verilerin %67'si modeli eğitmek adına %33'ü ise test verisi olarak tahmin değerlerinin performansını görüntülemek adına hazırlanmıştır. Bu işlemin de ardından yeni bir veri seti oluşturmak adına fonksiyon yazılmış olup Şekil 4.14'de kod parçası verilmiştir.

```

def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])
    return numpy.array(dataX), numpy.array(dataY)

```

Şekil 4.14 : Arıza Tahmini

Bu fonksiyonun amacı bir sonraki arızalanma zamanını tahmin etmek adına önceki arıza kayıtlarının kaç tanesine bakılacağına göre bir değer veren fonksiyondur. Look_back değişkeni burada 1 olarak verilmiş olup belirli bir zamandaki arıza başlangıç zamanını alarak ($t+1$) zamandaki arıza kaydının ne zaman olacağını tahmin etmektir. Bu işlemin de ardından tahmin değerleri için model oluşturulur ve Şekil 4.15'de model oluşturma işleminin yapıldığı kod parçası verilmiştir.

```

model = Sequential()
model.add(GRU(4, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(trainX, trainY, epochs=100, batch_size=1, verbose=2)

```

Şekil 4.15 : Modelin Oluşturulması

Kod parçasında görüldüğü üzere GRU model oluşturulmuş olup aynı işlemler tüm makineler için LSTM modeli için de gerçekleştirilmiş olup aradaki farklar ilerleyen kısımda gösterilecektir. Burada epochs değeri 100 iterasyon olarak verilmiştir. Bu işlemin sonucunda model eğitilmiş olup sonuçlar test edilmelidir. Şekil 4.16'da örnek olarak bir makinenin test skorları verilmiştir.

```

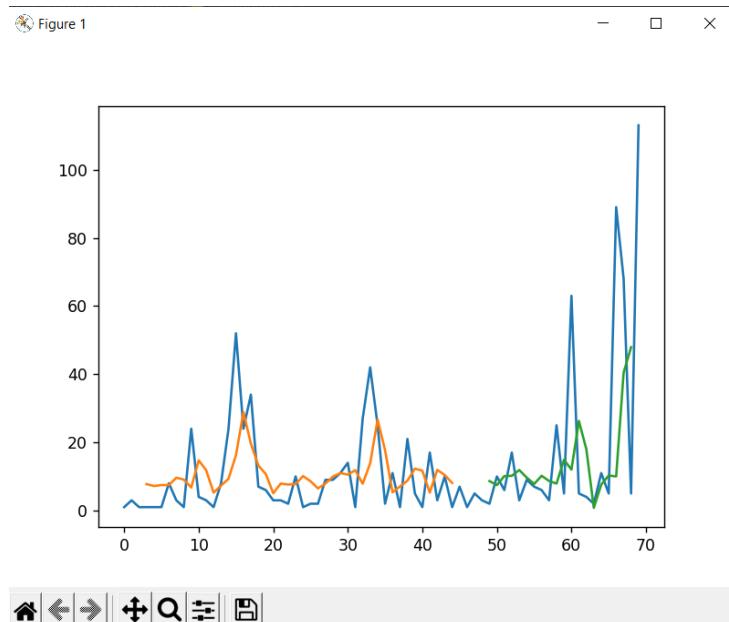
44/44 - 0s - loss: 0.0245 - 51ms/epoch - 1ms/step
Epoch 99/100
44/44 - 0s - loss: 0.0245 - 50ms/epoch - 1ms/step
Epoch 100/100
44/44 - 0s - loss: 0.0246 - 50ms/epoch - 1ms/step
2/2 [=====] - 0s 2ms/step
1/1 [=====] - 0s 18ms/step
Train Score: 12.75 RMSE
Test Score: 13.90 RMSE

```

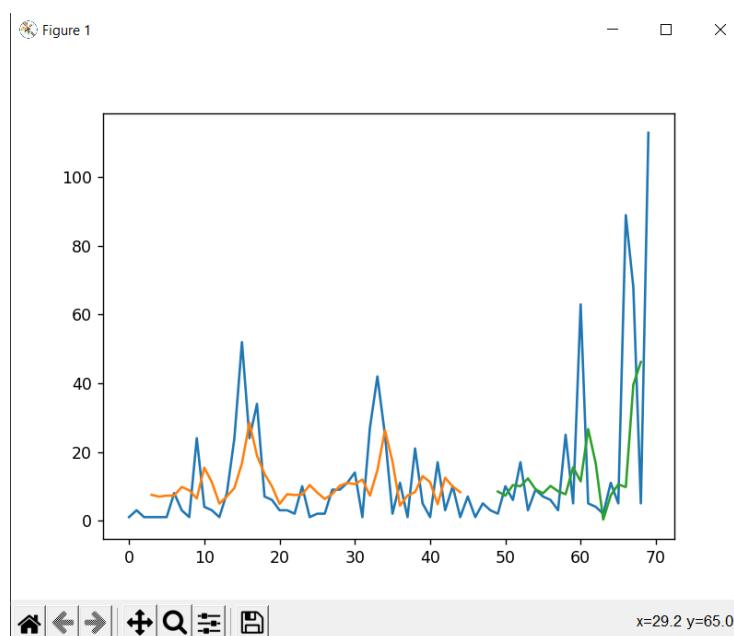
Şekil 4.16 : Eğitim ve Test Skorları

Bu görselde MakineId bilgisi 170 olan makinenin test ve eğitim skorları yazdırılmıştır. Bu işlemin de ardından tahmini verilerin üretip grafiklerin oluşturulması kısmına

geçilmiş olup tahmini değerlerin grafikleri LSTM ve GRU modelleri için aynı makinelerde karşılaşılacaktır. Şekil 4.17'de LSTM modeli ile 171.makine ve Şekil 4.18'de GRU modeli ile 171.makinenin bir sonraki arızalanma zamanlarının tahminlerinin olduğu grafikler gösterilmiştir.



Şekil 4.17 : LSTM Modeli İle Tahmin Sonuçları



Şekil 4.18 : GRU Modeli İle Tahmin Sonuçları

Görsellerde görüleceği üzere LSTM ve GRU modeli ile oluşturulan tahmin grafiklerinde fazla bir farka rastlanmamıştır. Bu işlemin de ardından tüm tahmini

değerlerin SQL Server'da oluşturulan BYS_MaintenanceTime tablosuna aktarılması
Şekil 4.19'da verilen kod parçasında gösterilmiştir.

```
conn = pyodbc.connect('Driver={SQL Server};'  
                      'Server=M5088157;'  
                      'Database=staj;'  
                      'Trusted_Connection=yes;')  
  
cursor = conn.cursor()  
for i in range(len(testPredictPlot)):  
    nonControl = numpy.isnan(testPredictPlot.item(i))  
    if(nonControl == False and testPredictPlot.item(i) > 0):  
        cursor.execute("INSERT INTO BYS_MaintenanceTime (MachineId,MaintenanceTestPredictPlot,IsLSTM) values(?, ?, ?)",  
        conn.commit()  
    continue
```

Şekil 4.19: SQL Server'da Tahmin Verilerini Kaydetme

Bu işlemin de ardından makine öğrenmesi kullanarak arızalanan makinelerin bir sonraki arızalarını tahmin etme işlemi tamamlanmış olur.

5. SONUÇ

Uygulama başarıyla tamamlanmış olup Diniz Adient Oto Donanım firmasında Gönen fabrikasının sunucusunda ASP.NET MVC olarak yayınlanmış durumdadır. Sisteme firma internet altyapısına bağlı olunan her yerden giriş yapılabilmektedir. Makine öğrenmesi kısmında ise firmada bulunan makinelerle ilgili farklı modellemeler üzerinde çalışmalar yapılmış ve hangi makinenin ne zaman arızalanacağı ile ilgili tahminde bulunulmuştur. İlgili bilgiler firma ile paylaşılmıştır.

Bakım Yönetim Sistemi’nde eski Bakım Ticket uygulamasına göre daha iyi bir görüntü, daha hızlı bağlantı ve daha kolay bir yönetim sistemine geçilmiş durumdadır. Ayrıca makine öğrenmesinde gelen verilerin artmasıyla beraber tahmin değerlerinin her geçen gün daha doğru değerlere yaklaşması beklenmektedir.

5.1 Öneriler

Bakım Yönetim Sistemi’nde makine arızalarının takibi için makine arızalandıktan sonra kullanıcıların web uygulamasına giriş yapıp, ilgili makineyi bulup yeni bir arıza talebi oluşturulması gereklidir. Bu işlem arızanın daha hızlı onarılması için hızlandırılabilir durumdadır. Firmanın Gönen’deki fabrikasında bakım sorumlusu olan Tuncay MUTLU ile olan görüşmeler sonucunda makine arızaların takibi için farklı bir çözüm ortaya çıkmıştır. Çözüme göre her bir makineye ait QR kodu makineye yapıştırılacak. Cep telefonu veya tablet ile kamera kullanılarak QR kod okutulacak ve direkt makine arızası oluşturma sayfasına ilgili makine bilgilerinin girili olduğu form ortaya çıkmış olacak. Bu çözümün ayrı bir avantajı da web uygulaması için bilgisayara gitmek zorunda kalmamak olaya yerinde müdahale etmek olarak düşünülmüştür. Bu çözüm ile beraber daha hızlı ve verimli bir şekilde makine arıza takibi yapılabilecektir.

KAYNAKLAR

Komple ASP.NET Web Geliştirme Eğitimi,

<https://www.udemy.com/course/angular-6-ve-aspnet-core-mvc-bastan-sona-proje-gelistirme/>

Introduction to ASP.NET Web Programming Using the Razor Syntax (C#), Erişim: 7 Ocak 2022, <https://docs.microsoft.com/en-us/aspnet/web-pages/overview/getting-started/introducing-razor-syntax-c>

A Complete C# Directory Tutorial, Erişim: 25 Ağustos 2019, <https://www.c-sharpcorner.com/article/directories-in-c-sharp/>

File.Delete(String) Method, <https://docs.microsoft.com/en-us/dotnet/api/system.io.file.delete?view=net-6.0>

Introduction to LINQ Contains, <https://www.educba.com/linq-contains/>

File Upload And Download Using ASP.NET MVC 5, Erişim: 5 Temmuz 2019, <https://www.c-sharpcorner.com/article/file-upload-and-download-using-asp-net-mvc-5-for-beginners/>

Tanışman, S. & Karcıoğlu, A., Uğur, A., Bulut, H., (2021). LSTM Sinir Ağı ve ARIMA Zaman Serisi Modelleri Kullanılarak Bitcoin Fiyatının Tahminlenmesi ve Yöntemlerin Karşılaştırılması, *Avrupa Bilim ve Teknoloji Dergisi*, 514-520,

Yamak, P., Li, Y., Gadosey, P.,(2019). Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial. ACAI 2019, 49-55,. doi.org/10.1145/3377713.3377722

Keras ile Tekrarlayan Sinir Ağları(RNN). Erişim: 9 Kasım 2015, <https://www.tensorflow.org/guide/keras/rnn>

Keras LSTM Layer Explained for Beginners with Example, <https://machinelearningknowledge.ai/keras-lstm-layer-explained-for-beginners-with-example/>

Keras LSTM ile Zaman Serisi Tahmini, Erişim: 7 Şubat 2018, http://ibrahimdelibasoglu.blogspot.com/2017/10/keras-lstm-ile-zaman-serisi-tahmini.html?_sm_au_=iVVtRZvH1wtrqw0FCK7BkKtcWkKM4

Predictive Analytics: Regression Analysis with LSTM,GRU and BiLSTM in TensorFlow, Erişim: 16 Temmuz 2020, <https://towardsdatascience.com/predictive-analysis-rnn-lstm-and-gru-to-predict-water-consumption-e6bb3c2b4b02>

LSTM vs GRU in Recurrent Neural Network: A Comparative Study, <https://analyticsindiamag.com/lstm-vs-gru-in-recurrent-neural-network-a-comparative-study/>

ÖZGEÇMİŞ

TARANMIŞ
VESİKALIK
FOTOĞRAF

Ad-Soyad

: Batuhan AVCI

Doğum Tarihi ve Yeri

: 01/01/1999 ESKİSEHİR

E-posta

: avcibatuhan26@gmail.com

BİTİRME ÇALIŞMASINDAN TÜRETİLEN MAKALE, BİLDİRİ VEYA SUNUMLAR:

-
-
-