

1. GİRİŞ

Teknolojinin gelişmesiyle klasik yöntemlerle gerçekleştirilen evrak işlemleri dijital dönüşüm sayesinde elektronik ortama taşınabilmektedir. Bu dönüşüm sonucunda süreçler daha hızlı, daha esnek ve daha güvenilir şekilde yönetilebilmektedir. Dijital ortamda tutulan verilere dünyanın her yerinden ulaşılabilmesi ve bu verilerin kolay yedeklenebilir olmasından dolayı dünyadaki tüm kamu ve özel sektör kurumları dijital dönüşümü var gücüyle gerçekleştirmektedir. Bunun yanı sıra elektronik sistemler tarafından yapılan kontroller sayesinde insan hatası en aza indirilmekte ve bilgisayarların sahip olduğu yüksek işlem gücü sayesinde zamandan tasarruf edilmektedir.

“Nesnelerin İnterneti (IoT)” konsepti dijital dönüşümün önemli bir parçasıdır. Bu konsept sayesinde bilgiye erişim dünya çapında gerçekleştirilebilmektedir [2]. Bir kurum, bir bölge içerisinde ve hatta dünya üzerinde dağılmış olan küçük ölçekli sistemler, “Nesnelerin İnterneti” sayesinde kendi aralarında haberleşebilir ve daha yüksek işlem gücüne sahip olan sunucular ile iletişim kurabilir [3]. Kurumlar, bu esnek mimari sayesinde ihtiyaç duyduğu ortam ve kullanıcı etkileşimli operasyonların tamamında istenilen ölçekte sistem kurgulayabilmektedir.

Gerçekleştirilmiş olan bu projede; eğitim-öğretim kurumlarında öğrencilerin derslere olan devamının kontrolünü sağlayan ve bir çizelge üzerinde gerçekleştirilen yoklama uygulamasının dijital dönüşümü “Nesnelerin İnterneti” konsepti sayesinde gerçekleştirilmiştir.

2. KULLANILAN TEKNOLOJİLER VE PROGRAMLAR

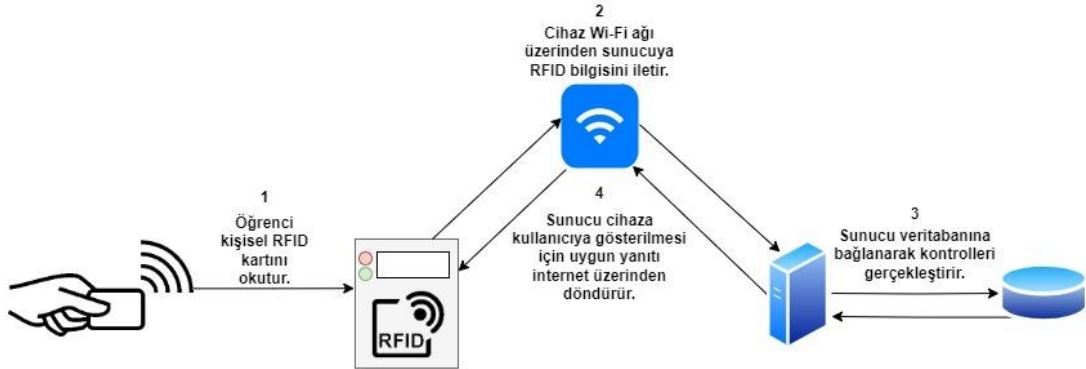
Donanım ve yazılım çözümlerinin geliştirilmesi sırasında kullanılan teknolojiler, programlar, kütüphaneler ve geliştirme ortamlarına kullanım amaçları ile Tablo 2.1’de yer verilmiştir.

<i>Teknoloji</i>	<i>Kullanım Amacı</i>
Github	Projenin geliştirilmesi aşamasında uzak sunucuda yedeklenmesi ve kaynak kodlarının yönetilmesi
GIMP	OLED için logoların yeniden boyutlandırılması ve transparan hale getirilmesi
Java 11	Sunucu taraflı yazılımın geliştirilmesi
PHPMyAdmin	MySQL veritabanı sunucusunun grafiksel arayüz ile yönetilmesi
MySQL	Depolanması gereken tüm kayıtlar için veritabanı yönetim sistemi
Apache Tomcat	Panellerden ve cihazlardan gelen HTTP isteklerinin kabul edilmesi
Postman	API uç noktalarının test edilmesi
IntelliJ Idea	Sunucu taraflı yazılım için geliştirme ortamı olarak kullanılması
Spring Boot	Yazılım iskeleti olarak kullanılması
Spring Security	Kullanıcıların oturumlandırılması ve yetkilendirilmesi
Hibernate ORM	Nesne-İlişkisel Eşleme yöntemiyle verilerin veritabanında nesneler olarak tutulması
Thymeleaf	HTML sayfalarının dinamikleştirilmesi
HTML, CSS, Bootstrap	Panellerin ön yüz tasarımının gerçekleştirilmesi
REST API	Sunucu tarafındaki API uç noktalarının oluşturulması
Arduino IDE	Gömülü yazılım geliştirme için geliştirme ortamı
Fritzing	Cihazlara ait devre tasarımı için devre tasarım programı olarak kullanılması
image2cpp	OLED ekranda resim gösterebilmek için resimlerin ByteArray’ye dönüştürülmesi

Tablo 2.1: Kullanılan teknolojiler ve kullanım amaçları

3. ELEKTRONİK YOKLAMA SÜRECİ

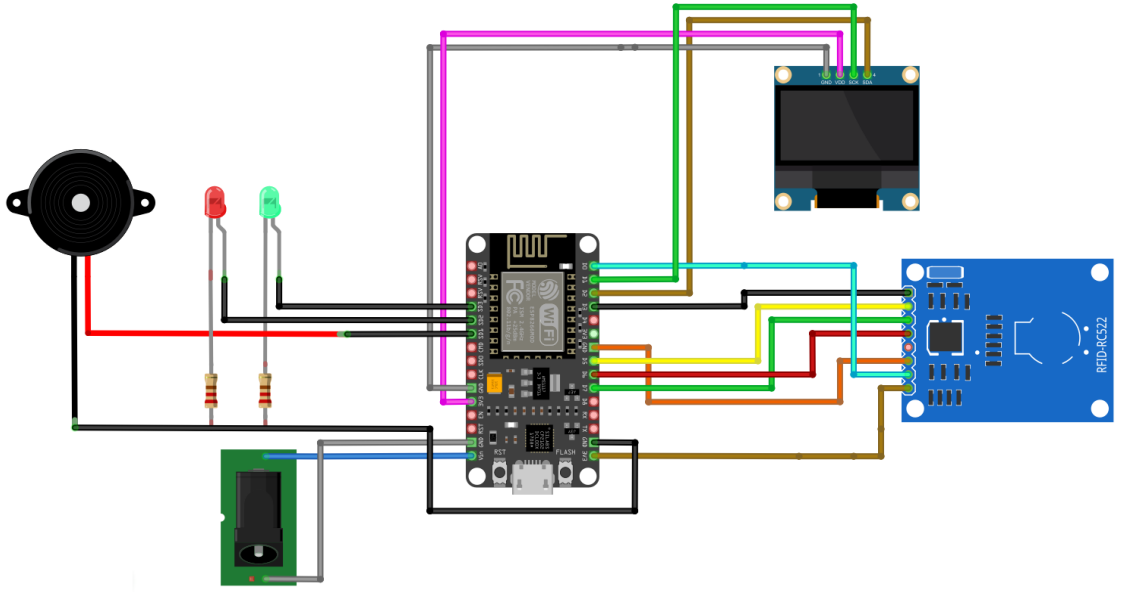
Bir derse katılan öğrencinin sistem tarafından tanınabilmesi ve bu öğrenci ile ilgili kaydın oluşturulabilmesi için Şekil 3.1’de gösterildiği gibi elektronik yoklama süreci kurgulanmıştır ve nasıl gerçekleştirildiği adım adım gösterilmiştir. 1 numaralı adımda; derse katılan öğrenci sistemde kendisine tanımlanan RFID kartını sınıfta sabit olarak bulunan cihaza okutmaktadır. Okutulan RFID kartına ait değerler kurumun Wi-Fi ağına bağlı olan cihaz tarafından sunucuya API servisleri üzerinden gönderilmektedir. Bu gönderme işlemi sırasında parametre olarak cihazın hangi sınıfta olduğunu tanımlayan bir token de gönderilmektedir. Bu sayede sunucu, hangi sınıftaki cihazın yoklama isteğinde bulunduğunu anlayabilmektedir. Sunucu cihazdan gelen RFID bilgisi ile öğrencinin kurumda kayıtlı olup olmadığı, ilgili dersi alıp almadığı ve o sınıfta ders işlenip işlenmediği gibi kompleks kontrolleri veritabanına bağlanarak gerçekleştirmektedir. Gerçekleştirilen bu kontroller sonucunda eğer herhangi bir olumsuz durum yoksa sunucu tarafından cihaza “Yoklama Alındı!” mesajı yanıt olarak gönderilmekte ve öğrenciye OLED ekranda gösterilmektedir. Eğer olumsuz bir durum varsa o duruma ait mesaj aynı şekilde cihaza gönderilerek ekranda gösterilmektedir.



Şekil 3.1: Elektronik yoklama sürecini gösterir akış diyagramı

4. DONANIM KISMININ GERÇEKLEŞTİRİLMESİ

Öğrencilerin kullanacak olduğu RFID kartlarının okunabilmesi, bu kartlardan okunan değerlerin sunucuya internet üzerinden iletilebilmesi ve öğrenciye bilgilendirme mesajlarının gösterilebilmesi için bir donanım çözümü geliştirilmiştir. Bu donanım çözümü için geliştirilen devre diyagramına Şekil 4.1’de yer verilmiştir.



Şekil 4.1: Donanım çözümü için geliştirilen devreye ait diyagram

Devrede mikrodenetleyici olarak üzerinde ESP8266 Wi-Fi modülünü bulundurması açısından NodeMCU kartı kullanılmıştır. NodeMCU geliştirme kartı, üzerinde CH340 çipini barındırmaktadır. Az alan kaplaması, projede kullanılacak olan çevre birimleri ile haberleşebilmek için yeterli pin sayısına sahip olması ve UART, SPI, I2C gibi haberleşme protokollerini desteklemesinden dolayı bu kartın projede kullanılması uygun görülmüştür. Kart 5V voltaj gerilimi ile beslenmiştir ve tüm çevre birimleri çalışabilmeleri için gerekli olan enerjiyi bu kart üzerinden almaktadır.

Devrede öğrenciye sesli ve ışıklı geri bildirim sunulabilmesi için Buzzer, bir adet 5mm kırmızı LED, bir adet 5mm yeşil LED kullanılmıştır. Sunucudan olumsuz bir yanıt geldiği zaman Buzzer uzun bir ses çıkarmakta ve kırmızı LED yanmaktadır. Öğrencinin yoklaması başarılı bir şekilde alındığı takdirde Buzzer iki kısa “bip” sesi çıkarmakta ve yeşil LED yanmaktadır. Bu 3 donanım, geliştirme kartına dijital pinler üzerinden bağlanmıştır. LED’lerin aşırı akımdan korunması için LED’lerin GND bacağına 330 Ω (Ohm) değerinde dirençler eklenmiştir.

Geliştirme kartının portatif bir şekilde beslenebilmesi için devreye DC Power Barrel Jack eklenmiştir. Bu güç girişi geliştirme kartının Vin ve GND pinlerine bağlanmıştır. Geliştirme kartının beslenmesi için tavsiye edilen voltaj gerilimi 5V-3.3V aralığında olduğundan dolayı bu girişten 5V gerilimi uygulanmaktadır.

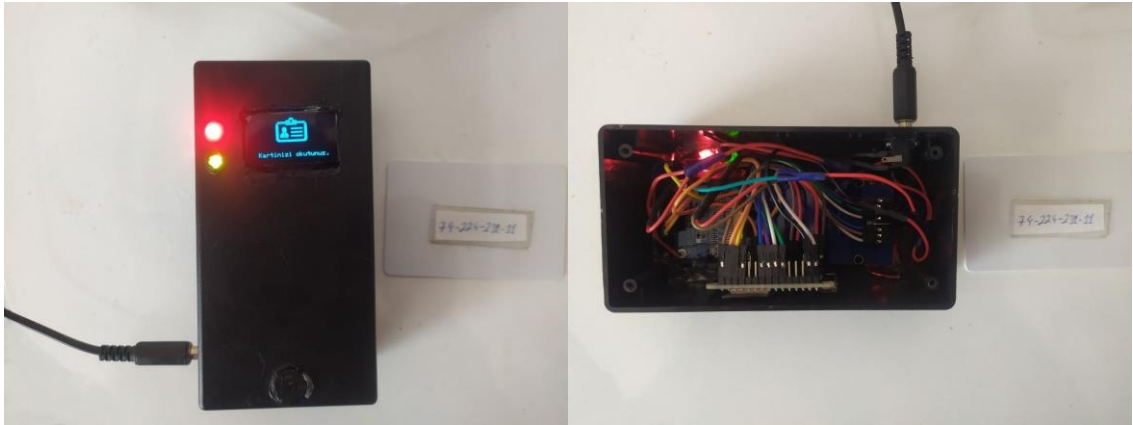
RFID kart modülü olarak RC522 isimli kit kullanılmıştır. Bu kitle bir adet RFID kart okuyucu ve iki adet RFID kartı bulunmaktadır. Bu kartların biri kredi kartı formunda diğeri anahtarlık formundadır. Her kartın kendine özgü bir hexadecimal değeri bulunmaktadır. RFID kart okuyucunun üzerindeki mikrodenetleyici ile haberleşilebilmesi için board üzerinde SS, SCK,

MOSI, MISO, RST isimli pinler bulundurmaktadır. SS, SCK, MOSI, MISO pinleri SPI haberleşmesi için kullanılmıştır. Bu modülün 3.3V voltaj gerilimi ile beslenmesi gerektiğinden, modülün enerji ihtiyacı geliştirme kartında bulunan 3.3V pininden karşılanmıştır.

Öğrenciye okunabilir geri bildirim sunulabilmesi için OLED ekran kullanılmıştır. Bu ekran üzerinde öğrenciye logolar ve sunucudan dönen yanıtlar gösterilmektedir. Mikrodenetleyici kart ile ekranın haberleşmesi için I2C protokolü kullanılmıştır ve ekran 3.3V voltaj gerilimi ile beslenmiştir.

5. DONANIMIN KUTULANMASI

Geliştirilen donanım; toz, sıvı teması, kısa devre gibi dış etkenlerden korunması için kutulanmıştır. Maliyetin düşük olması ve arzu edildiği gibi formuna müdahale edilebilmesi açısından plastik kutu kullanılması tercih edilmiştir. Bunun için Altinkaya firmasına ait PR-120 isimli kutu tercih edilmiştir. Kutu plastik malzemeden üretilmiş olup 68x130x44mm ölçülerindedir. Kutu, gereksiz alan kaplamayacak şekilde devrenin tamamını muhafaza etmiştir. Kullanıcı ile etkileşimi sağlayacak olan LED, Buzzer ve OLED ekran, kutudan delikler açılarak dışarıya çıkartılmıştır. Devrenin adaptörden beslenebilmesi için DC Barrel Jack kutunun yan tarafından dışarıya çıkarılmıştır. Kutu içerisinde kalacak olan NodeMCU ve RFID kart okuyucu çift taraflı bant ile sabitlenmiştir. Plastik kutunun malzemesinin yeterli derecede ince olmasından dolayı RFID kart okuyucu sorunsuz bir şekilde RFID kartlarının değerlerini okuyabilmektedir. Fritzing üzerinde gerçekleştirilmesi planlanan devre başarıyla hayata geçirilmiştir. Donanımın son durumu Şekil 5.1 görselinde gösterilmiştir.



Şekil 5.1: Donanım çözümü olarak geliştirilen cihaz

6. GÖMÜLÜ YAZILIMIN GERÇEKLEŞTİRİLMESİ

Geliştirme kartının çevre birimlerini sürebilmesi ve sunucu ile haberleşebilmesi için gömülü yazılım geliştirilmiştir. Bu yazılım Arduino IDE üzerinde Arduino mimarisini kullanarak geliştirilmiştir.

Yazılımda OLED ekranın sürülebilmesi, yazı ve logoların gösterilebilmesi için Adafruit GFX, Adafruit SH110X ve Wire kütüphanelerinden yararlanılmıştır. NodeMCU geliştirme kartı üzerinde bulunan ESP8266 WiFi modülünün kullanılabilmesi için ESP8266WiFi kütüphanesinden yararlanılmıştır. Sunucu tarafından cihaza geri döndürülen JSON yanıtlarının ayrıştırılabilmesi (parse) için ArduinoJson kütüphanesi kullanılmıştır. RFID kart okuyucu ile haberleşilebilmesi için RC522 modülü için geliştirilmiş olan MFRC522 kütüphanesi kullanılmıştır.

OLED ekranda öğrenciye logoların gösterilebilmesi için görseller birtakım işlemlerden geçirilmiştir. Öncelikle logolar GIMP fotoğraf düzenleme aracıyla transparan forma dönüştürülmüştür. Ardından <https://javi.github.io/image2cpp/> adresindeki image-byteArray dönüştürücü aracı kullanılarak bitmap dizileri oluşturulmuştur. Bu sayede logolar öğrenciye OLED ekranda başarıyla gösterilmiştir. Sunucudan olumsuz bir yanıt geldiği takdirde öğrenciye bir “X” logosu, olumlu bir yanıt geldiğinde “✓” logosu ve sunucudan gelen mesaj öğrenciye gösterilmektedir.

NodeMCU mikrodenetleyicisinin sunucuya bağlanabilmesi için üzerinde bulunan ESP8266 Wi-Fi modülünün internet bağlantısı gerçekleştirilmiştir. Bunun için ESP8266WiFi kütüphanesinden yararlanılmıştır. Cihaz, kaynak kodunda belirtilen Wi-Fi ağına bağlantı gerçekleştirdikten sonra yoklama isteklerini uzak sunucunun IP adresine veya domain adresine HTTP protokolü ile göndermektedir.

7. SUNUCU YAZILIMININ GERÇEKLEŞTİRİLMESİ

Sınıflarda bulunan cihazlardan gelen isteklerin işlenebilmesi, bir bölüme ait dersler, öğrenciler ve akademisyenler ile ilgili veritabanı işlemlerinin gerçekleştirilebilmesi, öğrencilerin devamsızlıklarını panel üzerinden takip edebilmesi ve sistem yöneticisinin gerekli işlemleri panel üzerinden gerçekleştirebilmesi için sunucu tarafı bir yazılıma ihtiyaç duyulmuştur. Bu yazılım Java dilinde Spring Boot Framework’ü kullanılarak geliştirilmiştir. Panellerin ön yüz (front-end) kısmı HTML,CSS,Bootstrap kullanılarak geliştirilmiştir.

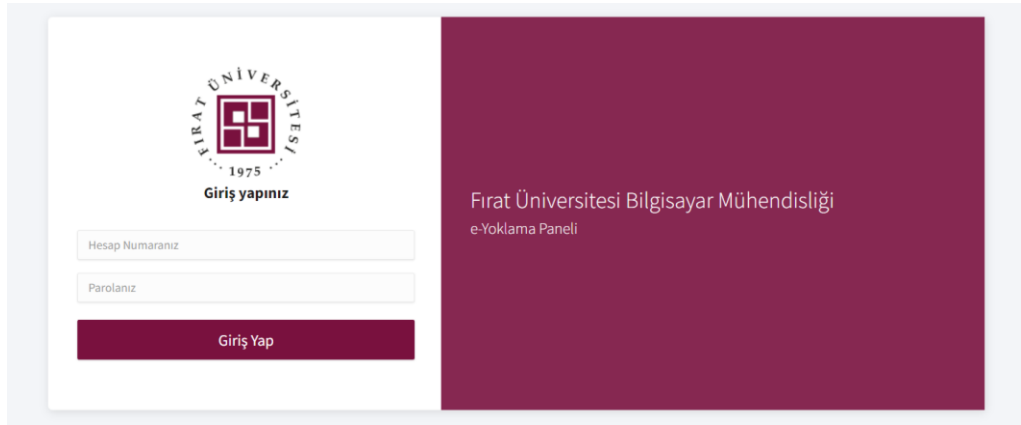
Sistem iki adet panel içermektedir. Bir panel öğrenciler için diğer panel sistem yöneticisi için geliştirilmiştir. Bu panellere yalnızca yetkisi olanların erişebilmesi için bir adet de giriş paneli oluşturulmuştur. Giriş kontrolleri Spring Security ile yapılan yapılandırmalar sayesinde gerçekleştirilmektedir. Bu güvenlik yapılandırmalarına Şekil 7.1’de yer verilmiştir. Giriş panelinin son durumuna Şekil 7.2’de yer verilmiştir.

```

Aziz Can HAMASOGLU *
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
        .httpBasic() // ===== API CONFIGURATION
        .and() HttpSecurity
            .authorizeRequests() ExpressionInterceptUrlRegistry
            .antMatchers(HttpMethod.GET, ...antPatterns: "/api").hasAuthority("ADMIN")
        .and() HttpSecurity
            .csrf().disable()
        .authorizeRequests() // ===== VIEW CONFIGURATION
        .antMatchers( ...antPatterns: "/", "/assets/**") AuthorizedUrl
        .permitAll() ExpressionInterceptUrlRegistry
        .antMatchers( ...antPatterns: "/ogrenci-panel") AuthorizedUrl
        .hasAuthority("USER") ExpressionInterceptUrlRegistry
        .antMatchers( ...antPatterns: "/home") AuthorizedUrl
        .hasAuthority("ADMIN") ExpressionInterceptUrlRegistry
        .anyRequest() AuthorizedUrl
        .authenticated() ExpressionInterceptUrlRegistry
        .and() HttpSecurity
            .formLogin() FormLoginConfigurer<HttpSecurity>
            .loginPage("/Login")
            .usernameParameter("username")
            .passwordParameter("password")
            .loginProcessingUrl("/process-login")
            .successHandler(successHandler)
            .permitAll()
        .and() HttpSecurity
            .logout() LogoutConfigurer<HttpSecurity>
            .logoutSuccessUrl("/")
            .permitAll();
}

```

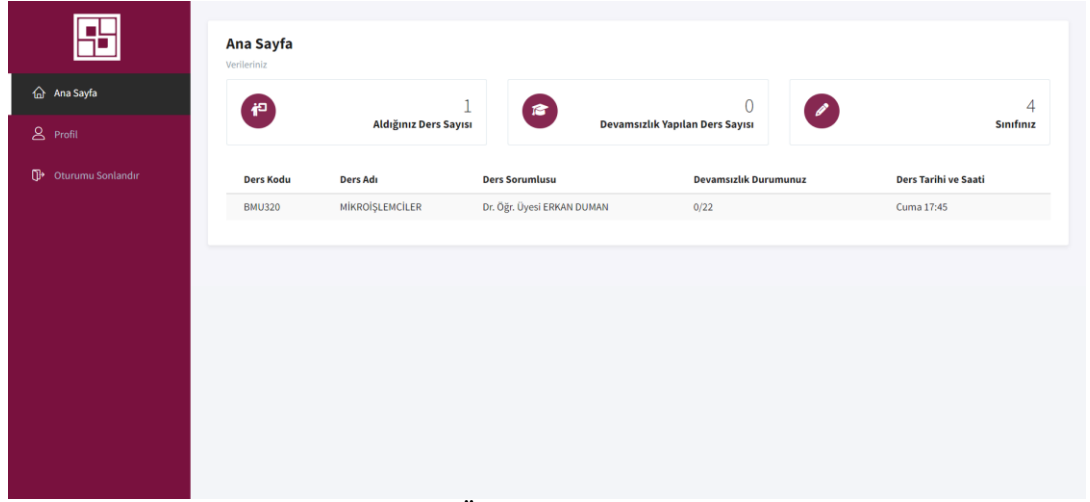
Şekil 7.1: Giriş paneli için yapılan Spring Security yapılandırmaları



Şekil 7.2: Tarayıcı üzerinden erişilen e-Yoklama sistemine giriş paneli

Öğrencilerin devamsızlıklarını takip edebileceği panele Şekil 7.3 görselinde yer verilmiştir. Bu panelde öğrencinin almış olduğu dersler otomatik olarak listelenmekte ve öğrencinin kayıtlı olduğu her derse ait devamsızlık bilgisi veritabanından çekilerek

gösterilmektedir. “Profil” sayfasında öğrenciye ait TCKN, sınıf, ad-soyad, doğum tarihi, doğum yeri bilgileri yine veritabanından çekilerek gösterilmektedir.



Şekil 7.3: Öğrenci paneli görünümü

Sistemde ADMIN (Yönetici) rolünde tanımlı bir kişi bulunmaktadır. Bu kişi sistemde tutulan kayıtlara ve sistem yapısına müdahale edebilen tek kullanıcıdır. Bu kullanıcı sisteme öğrenci ve akademisyenleri ekleyebilir, silebilir, sorgulayabilir ve kişilere ait bilgileri güncelleyebilir. Bunun yanında yoklaması alınmış bir derste “YOK” yazılmış bir öğrenciyi “VAR” yazabilir veya tam tersini yapabilir. Ayrıca sisteme yeni dersler, sınıflar tanımlayabilir veya bu sınıflara öğrencilerin kartlarını okuyacak olan cihazları tanımlayabilir. Spring Boot tarafında her panel için bir Controller sınıfı oluşturulmuş ve her Controller sınıfı içerisinde ekleme, güncelleme, silme, sorgulama ve diğer spesifik işlemler için fonksiyonlar tanımlanmıştır. Bu fonksiyonlara belirli URL adresleri ve uygun parametreler ile ulaşılabilir. Örnek olması açısından Şekil 7.4’de bir öğrenciye ait bilgileri güncelleyen fonksiyona ait kaynak kod verilmiştir.



```
// Var olan bir öğrencinin yeni bilgilerinin update edilmesi
no usages  Aziz Can HAMASOGLU
@PostMapping("/ogrenciGuncelleUpdate/{ogrenciNo}")
public String ogrenciGuncelleUpdate(Ogrenci ogrenci, @PathVariable String ogrenciNo, String parola){
    Ogrenci ogr=ogrenciRepo.findByOgrenciNo(ogrenciNo);
    User user=userRepo.findByUsername(ogrenciNo);
    if(ogrenci.getAd()!=null)
        ogr.setAd(ogrenci.getAd());
    if (ogrenci.getSoyad()!=null)
        ogr.setSoyad(ogrenci.getSoyad());
    if(ogrenci.getDogumTarihi()!=null)
        ogr.setDogumTarihi(ogrenci.getDogumTarihi());

    if(parola!=null){
        user.setPassword(new BCryptPasswordEncoder().encode(parola));
    }
    if (ogrenci.getRfidKodu()!=null)
        ogr.setRfidKodu(ogrenci.getRfidKodu());
    ogrenciRepo.save(ogr);
    userRepo.save(user);
    return "redirect:/ogrenci-islemleri";
}
```

Şekil 7.4: Bir öğrenciye ait bilgileri güncelleyen fonksiyon

Yönetici panelinde bulunan ana sayfanın görünümüne Şekil 7.5’de, “Öğrenci İşlemleri” paneline Şekil 7.6’de, “Yoklama İşlemleri” paneline Şekil 7.7’de, “Ders İşlemleri” paneline Şekil 7.8’de, “Sınıf İşlemleri” paneline Şekil 7.9’de, “Cihaz İşlemleri” paneline Şekil 7.10’da, “Akademisyen İşlemleri” paneline Şekil 7.11’da yer verilmiştir.

Şekil 7.5: Yönetici paneli ana sayfası



[Ana Sayfa](#)

[Öğrenci İşlemleri](#)

[Yoklama İşlemleri](#)

[Ders İşlemleri](#)

[Sınıf İşlemleri](#)

[Cihaz İşlemleri](#)

[Akademisyen İşlemleri](#)

[Oturumu Sonlandır](#)

Sınıf Ekle

Sınıf eklemek için aşağıdaki formu doldurup onaylayınız.


OnaylaTemizle

Sınıf Sil

Sınıf silmek için sınıf seçip onaylayınız.

OnaylaTemizle

Şekil 7.9: “Sınıf İşlemleri” paneli



[Ana Sayfa](#)

[Öğrenci İşlemleri](#)

[Yoklama İşlemleri](#)

[Ders İşlemleri](#)

[Sınıf İşlemleri](#)

[Cihaz İşlemleri](#)

[Akademisyen İşlemleri](#)

[Oturumu Sonlandır](#)

Cihaz Ekle

Cihaz eklemek için aşağıdaki formu doldurup onaylayınız.

OnaylaTemizle

Cihaz Sil

Cihaz silmek için listeden cihaz seçip onaylayınız.


OnaylaTemizle

Cihaz Sınıfını Güncelle

Cihaz sınıfını güncellemek için aşağıdaki formu doldurup onaylayınız.

OnaylaTemizle

Şekil 7.10: “Cihaz İşlemleri” paneli



[Ana Sayfa](#)

[Öğrenci İşlemleri](#)

[Yoklama İşlemleri](#)

[Ders İşlemleri](#)

[Sınıf İşlemleri](#)

[Cihaz İşlemleri](#)

[Akademisyen İşlemleri](#)

[Oturumu Sonlandır](#)

Akademisyen Ekle

Akademisyen eklemek için aşağıdaki formu doldurup onaylayınız.

☐ Erkek☐ Kadın

OnaylaTemizle

Ünvan Güncelle

Ünvanını güncellemek istediğiniz akademisyeni "Personel Numarası" ile sorgulayınız.

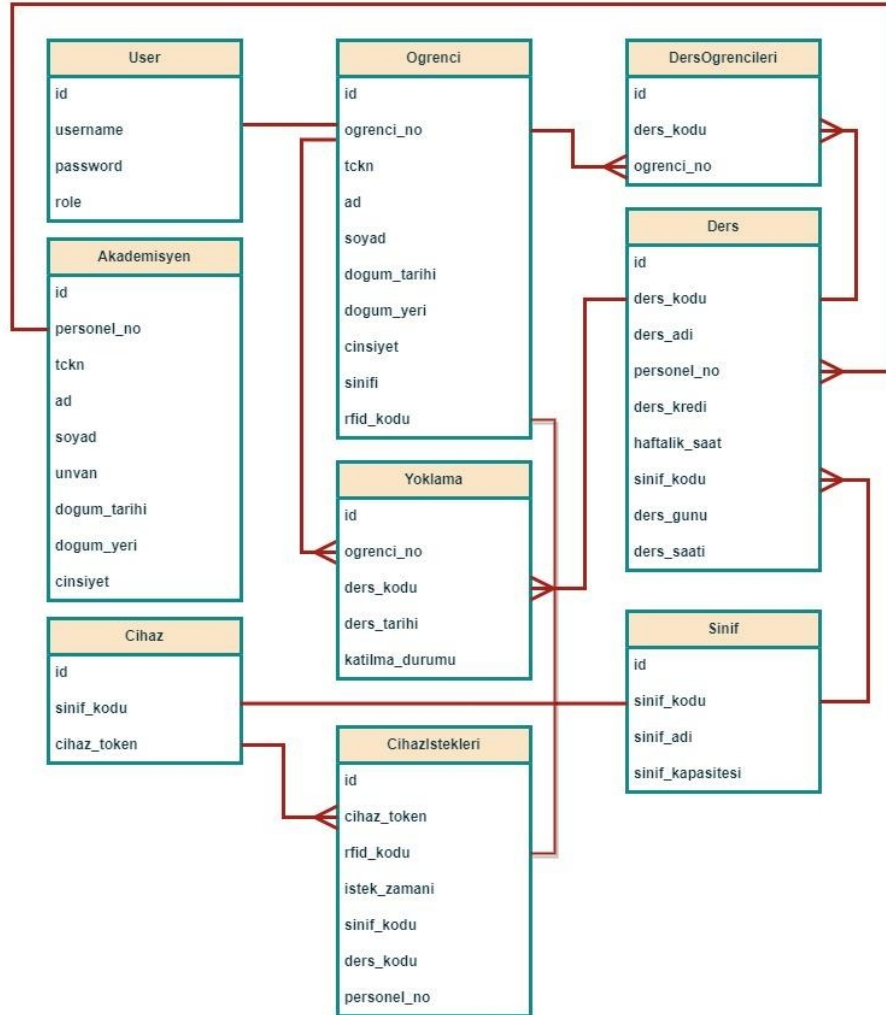
OnaylaTemizle

Şekil 7.11: “Akademisyen İşlemleri” paneli

8. VERİTABANI MİMARİSİNİN OLUŞTURULMASI

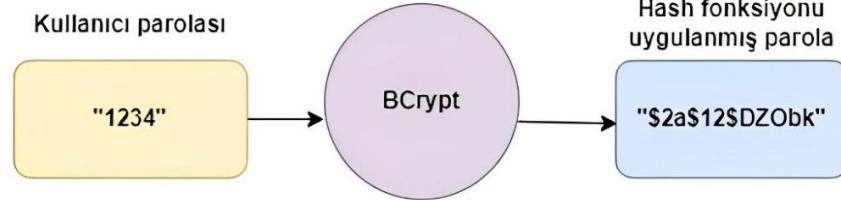
Tüm kayıtların veritabanında düzenli bir şekilde saklanabilmesi için veritabanı mimarisi geliştirilmiştir. Bu mimariye Şekil 8.1’de yer verilmiştir. User tablosu öğrencilerin ve sistem yöneticisinin panellere girişinde parola ve rol kontrollerinin gerçekleştirildiği tablodur. Öğrenciler USER rolü ile tanımlı olduğundan yönetici paneline erişimlerine izin verilmemektedir. Spring Security yapılandırmasında yalnızca ADMIN rolüne sahip olanların yönetici paneline erişmesine izin verileceği tanımlandığından öğrenciler yönetici paneline erişememektedir.

Oğrenci ve Akademisyen tablolarında öğrencilerin ve akademisyenlerin kişisel bilgileri tutulmaktadır. Yoklama tablosunda işlenen her derse ait yoklama kayıtları tutulmaktadır. DersOğrencileri tablosu, bir derse kayıtlı olan öğrencilerin bulunabilmesi için oluşturulmuştur. Cihaz tablosu her sınıfta olan cihazın sisteme kayıtlı olup olmadığını sorgulamak için kullanılır.



Şekil 8.1: Veritabanı mimarisi

User tablosuna yeni bir kayıt eklendiği zaman kayıt işleminde önce kullanıcının girmiş olduğu parola BCrypt algoritması ile hash fonksiyonu uygulanarak saklanmaktadır. Bundan dolayı veritabanında tutulan parolalar kötü niyetli kişiler tarafından çalınsa dahi hiçbir anlam ifade etmeyecektir. BCrypt algoritmasına ait fonksiyonlar Spring Boot tarafından varsayılan olarak sağlanmaktadır.



Şekil 8.2: Parolanın şifrlenmesini gösterir diyagram

9. VERİTABANI İŞLEVLERİNİN TANIMLANMASI

Nesne ilişkisel eşleme (ORM) yapısı sayesinde sunucu taraflı yazılım geliştirmesinde daha az SQL kodu yazarak ve nesne yönelimli programlama (OOP) yaklaşımına daha uygun olacak şekilde geliştirme gerçekleştirilmiştir [4]. Spring Boot üzerinde Hibernate ile ORM yapısının oluşturulması için Entity ve Repository katmanları oluşturulmuştur. Entity katmanında her tabloyu niteleyecek birer sınıf (Class) oluşturulmuştur. Bu sınıflar içerisinde tablonun sütunlarını niteleyecek olan alan adları (field) oluşturulmuştur. Bu sınıflara verilen @Column anotasyonu ile tabloda oluşturulacak olan sütunun adı verilmiştir. Örnek olması açısından Şekil 9.1’de DersOgrencileri tablosuna ait Entity sınıfının kaynak kodu verilmiştir.

```

11 usages Aziz Can HAMASOGLU
@Entity
@Table(name="DersOgrencileri")
public class DersOgrencileri {

    2 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    2 usages
    @Column(name = "ders_kodu")
    private String dersKodu;

    2 usages
    @Column(name="ogrenci_no")
    private String ogrenciNo;

    Aziz Can HAMASOGLU
    public Long getId() { return id; }

    Aziz Can HAMASOGLU
    public void setId(Long id) { this.id = id; }

    Aziz Can HAMASOGLU
    public String getDersKodu() { return dersKodu; }

```

Şekil 9.1: DersOgrencileri tablosuna ait Entity sınıfı

Veritabanı işlevlerinin tanımlanması için Repository katmanı oluşturulmuştur ve bu katman içerisinde her Entity sınıfı için birer arayüz (interface) oluşturulmuştur. Bu arayüzler JpaRepository arayüzünü kalıtım alarak bu arayüze ait fonksiyonları kullanabilmektedir. Bu fonksiyonlar save(), delete(), findAll() gibi hazır fonksiyonlardır. Bunun yanı sıra @Query anotasyonu ile daha özelleşmiş veritabanı işlevleri oluşturulmuştur. Örnek olması açısından DersOgrencileri sınıfına ait Repository arayüzüne ait kaynak kodun bir kısmına Şekil 9.2’de yer verilmiştir.

```

4 usages Aziz Can HAMASOGLU *
public interface DersOgrencileriRepository extends JpaRepository<DersOgrencileri,Long> {

    1 usage Aziz Can HAMASOGLU
    @Modifying
    @Transactional
    @Query("delete from DersOgrencileri do " +
        "where do.ogrenciNo=:ogrenciNo " +
        "and do.dersKodu=:dersKodu")
    void deleteByOgrenciNo(String ogrenciNo,String dersKodu);

    1 usage Aziz Can HAMASOGLU
    @Query("select count(*) from DersOgrencileri do " +
        "where do.ogrenciNo=:ogrenciNo")
    public String getCountByOgrenciNo(String ogrenciNo);

```

Şekil 9.2: DersOgrencileri tablosuna ait Repository arayüzü

10. PLANLI YOKLAMA GÖREVİNİN OLUŞTURULMASI

Bir ders başladıktan belli bir zaman sonra yoklama alma işleminin kapatılabilmesi için sunucu tarafında zaman planlamalı bir görev atanmıştır. Örneğin; ders başlama saatinden 15 dakika sonra öğrencinin yok yazılması. Bunun için kurumdaki her dersin (xx:00, xx:15, xx:30, xx:45) zamanlarında başladığı varsayılmıştır. Örnek bir senaryo düşünülecek olursa; Mikroişlemciler dersi 14:15’de başlamaktadır. Sunucuda zaman tetiklemeli bir görev tanımlandığı için saat 14:15 olduğunda dersin ilk 15 dakikasının dolup dolmadığına bakılacaktır. Dersin ilk 15 dakikası dolmadığından bir sonraki tetiklemenin gerçekleşeceği saat olan 14:30’da dersin yoklaması kapatılacaktır. Mikroişlemciler dersinin işlendiği sınıfta bulunan cihazdan 14:15-14:30 arasında gelen istekler “VAR” olarak işaretlenerek veritabanına kaydedilmektedir. Bu yaklaşım sayesinde sunucu sürekli olarak yeni bir dersin başlayıp başlamadığını kontrol etmek zorunda kalmayacak, gereksiz işlem yükü oluşmayacak ve veritabanı meşgul edilmemiş olacaktır.

11. SONUÇ

Yıllardır klasik olarak gerçekleştirilen işlemlerin dijital olarak dönüştürülmesi sayesinde bu işlemler daha hızlı, daha esnek ve daha az takip edilmesini gerektirecek şekilde gerçekleştirilebilmektedir. Dijital dönüşüm sayesinde insan üzerinde oluşan iş yükü kayda değer biçimde azaltılabilmekte ve işlemler otomatize edilebilmektedir. Geliştirilmiş olan bu proje sayesinde imza taklidi ve yoklama çizelgesinin kaybolması/kullanılamaz hale gelmesi/yanlış satıra imza atılması gibi insan kaynaklı olumsuz durumların önüne geçilmiş ve ders sorumlusu olan kişi üzerindeki iş yükü azaltılmıştır.