KOÜ Akademik Personel Başvuru Sistemi

Ensar Akbaş
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
ensarr.akbas@gmail.com
+90 546 230 87 07

Berkay Yüce
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
berkay.yce10@gmail.com
+90 551 971 89 07

Ahmet Ertuğrul Tuğ
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
ertugrultug54@gmail.com
+90 551 552 67 43

***Abstract*—Bu projede, üniversitelerdeki akademik personel başvuru süreçlerini dijitalleştiren ve otomatikleştiren bir web tabanlı sistem geliştirilmiştir. Sistem, dört farklı kullanıcı rolü ("Aday", "Admin", "Yönetici", "Jüri") temelinde yetkilendirme ve iş akışı sunar. Adaylar sisteme giriş yapar, başvuru belgelerini yükler ve süreci çevrim içi olarak takip eder. Yönetici ve jüri üyeleri değerlendirme ve karar süreçlerini sistem üzerinden yürütür. Sistem ayrıca "Tablo 5" puanlama mekanizmasıyla adayların akademik niteliklerini otomatik olarak hesaplar ve PDF formatında çıktılar sunar.**

Keywords*—*Akademik başvuru sistemi, kullanıcı roller, puanlama mekanizması, Rest API, JWT, React.js, Django Rest Framework, PDF üretimi.

I. GIRIŞ

Akademik personel alım süreci, üniversitelerde hâlâ büyük ölçüde manuel işlemlerle yürütülmektedir. Adayların başvuru belgelerini fiziksel olarak teslim etmesi, değerlendirme süreçlerinin e-posta ve kağıt evraklar üzerinden gerçekleştirilmesi, ilan yönetimi ve jüri raporlarının elle toplanması gibi uygulamalar hem zaman kaybına hem de çeşitli hatalara yol açmaktadır. Belgelerin eksik veya yanlış ulaşması, aday bilgilerinin kontrolünün zor olması, değerlendirme süreçlerinin takibinin yapılamaması gibi sorunlar bu süreci hem adaylar hem de idari personel açısından verimsiz hâle getirmektedir.

Bu proje, söz konusu akademik atama sürecini tamamen dijital bir yapıya taşıyarak tüm kullanıcılar için daha hızlı, şeffaf ve güvenilir bir sistem kurulmasını amaçlamaktadır. Web tabanlı geliştirilen bu sistem; adayların kimlik doğrulaması yaparak giriş yapmasını, ilanlara başvurabilmesini ve belgelerini elektronik ortamda yüklemesini sağlar. Yönetici rolündeki kullanıcılar ilan açabilir, başvuru kriterlerini belirleyebilir, jüri üyelerini sistem üzerinden atayabilir ve değerlendirme raporlarını dijital olarak toplayabilir. Jüri üyeleri ise kendilerine atanan adaylara ait belgeleri çevrim içi olarak inceleyip değerlendirmelerini sisteme yükleyebilir. Tüm bu iş akışları sistemde roller bazında yetkilendirilmiş ve merkezi olarak kontrol edilebilecek şekilde tasarlanmıştır.

II. PROBLEM TANIMI

Üniversitelerde akademik personel alım sürecinde sıkça karşılaşılan başlıca problemler; evrak eksiklikleri, elle yapılan puanlama hataları ve jüri üyeleri arasındaki koordinasyon eksikliğidir. Adaylardan istenen belgelerin eksik veya yanlış iletilmesi başvurunun reddedilmesine neden olabilirken, başvuru belgeleri üzerinde yapılan puanlamaların manuel yöntemlerle hesaplanması değerlendirme sürecinin hataya açık hâle gelmesine yol açmaktadır.

Ayrıca jüri üyeleri genellikle farklı birimlerde ve zaman dilimlerinde çalıştıkları için belgelerin paylaşımı, raporların toplanması ve değerlendirmelerin zamanında tamamlanması ciddi bir organizasyon yükü doğurmaktadır. Başvuru sürecine dair ilerleme durumunun aday tarafından şeffaf şekilde takip edilememesi de iletişim sorunlarına neden olmaktadır.

Bu proje, tüm bu sorunları ortadan kaldırmak amacıyla geliştirilmiştir. Başvuru belgelerinin sistem üzerinden dijital olarak yüklenmesi ve doğrulanması, puanlama işlemlerinin otomatikleştirilmesi, jüri değerlendirme formlarının sistem üzerinden alınması ve tüm tarafların kendi rollerine göre sisteme erişebilmesi sayesinde süreç daha verimli, izlenebilir ve hatasız bir hâle getirilmiştir.

III. KULLANILAN TEKNOLOJILER

Proje, modern web teknolojileri kullanılarak istemci-sunucu mimarisi ile geliştirilmiştir. Kullanıcı arayüzü ve sistemin mantıksal işleyişi ayrı katmanlarda ele alınmış, ölçeklenebilir ve yönetilebilir bir yapı kurulmuştur.

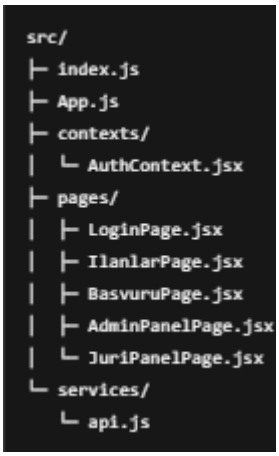
- **Backend:**
  Projede sunucu tarafı Python tabanlı Django web çatısı ile geliştirilmiştir. REST API oluşturmak için Django Rest Framework (DRF) kullanılmış ve sistemin tüm işlevleri RESTful servisler aracılığıyla erişilebilir hâle getirilmiştir. Tüm kullanıcı rolleri, başvuru akışları, ilan yönetimi ve değerlendirme işlemleri bu API katmanı üzerinden sağlanmaktadır.

A. Proje Dizini

```
akademik_sistem/
│
├── akademik_sistem/      # Proje ayarları
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
│
├── users/                # Kimlik doğrulama & profil
│   ├── models.py
│   ├── serializers.py
│   ├── views.py
│   └── urls.py
│
├── ilanlar/              # İlan yönetimi
│   ├── models.py
│   ├── serializers.py
│   ├── views.py
│   └── urls.py
│
├── basvurular/           # Aday başvuruları
│   ├── models.py
│   ├── serializers.py
│   ├── views.py
│   └── urls.py
│
├── juri/                 # Jüri atama & değerlendirme
│   ├── models.py
│   ├── serializers.py
│   ├── views.py
│   └── urls.py
│
└── manage.py
```

- **Frontend:**
  Kullanıcı arayüzü, React.js kullanılarak geliştirilmiştir. Bileşen tabanlı yapı sayesinde aday, yönetici, jüri ve admin kullanıcılarının her biri için özelleştirilmiş arayüzler tasarlanmıştır. Kullanıcı etkileşimleri API üzerinden sunucuya iletilmekte, gelen veriler dinamik olarak ekranda gösterilmektedir.

A. **Proje Dizini**



- **Veritabanı:**
  Projede verilerin kalıcı olarak saklanması için PostgreSQL kullanılmıştır. Kullanıcı bilgileri, başvuru kayıtları, ilan verileri ve yüklenen belgeler ilişkisel olarak bu veritabanında tutulmaktadır.

- **Kimlik Doğrulama:**

  Giriş ve oturum yönetimi için JSON Web Token (JWT) kullanılmıştır. Her kullanıcı girişi sonrası sunucu tarafından verilen token ile kullanıcı kimliği doğrulanmakta ve güvenli bir oturum sağlanmaktadır.

- **Dosya Yükleme ve Depolama:**
  Adaylar başvuru belgelerini sistem üzerine yükleyebilmekte ve bu belgeler Django'nun multipart dosya yükleme mekanizmasıyla sunucuya aktarılmaktadır. Belgeler, ilgili başvurularla ilişkilendirilerek saklanır.

- **PDF**
  Aday başvuru değerlendirmesi sonucu oluşturulan "Tablo 5" belgeleri, Python'un ReportLab kütüphanesi kullanılarak sistem tarafından otomatik olarak PDF formatında üretilmektedir. Bu PDF çıktılar kullanıcılar tarafından indirilebilmektedir.

IV. Sistem Mimarisi ve Modüller

Geliştirilen sistem, istemci-sunucu mimarisine sahip olup, tüm modüller arka planda Django REST API ile haberleşmektedir. Sistem, işlevsel olarak dört ana modülden oluşur ve bu modüller, kullanıcıların rollerine göre farklı arayüzler ve yetkilerle etkileşime geçer. Veriler PostgreSQL üzerinde saklanırken, istemci tarafında React.js ile bu verilere erişim sağlanır.

A. Kullanıcı Yönetimi Modülü:

Bu modül, sistemdeki dört farklı kullanıcı türünü (Aday, Admin, Yönetici, Jüri) kapsar. Her kullanıcı JWT tabanlı kimlik doğrulama ile sisteme giriş yapar. Roller arası erişim kontrolü, sunucu tarafında sağlanır. Adaylar başvuru yapabilirken, yöneticiler ilan açabilir, jüri atayabilir ve sonuçları yönetebilir.

B. İlan Yönetimi Modülü:

Yöneticiler tarafından oluşturulan akademik ilanlar bu modül aracılığıyla sisteme eklenir. Her ilan; başvuru kriterleri, başlangıç ve bitiş tarihleri ile birlikte tanımlanır. Adaylar, bu ilanları görüntüleyebilir ve başvuru işlemlerini başlatabilir. İlanlar güncellenebilir, silinebilir ve filtrelenebilir durumdadır.

C. Başvuru Yönetimi Modülü:

Adayların ilanlara yaptıkları başvurular bu modül üzerinden takip edilir. Her başvuru için aday belgelerini yükler ve sistem üzerinden belgelerin eksiksiz olup olmadığı kontrol edilir. Başvuruların durumu ("Beklemede", "Onaylandı", "Reddedildi") yöneticiler tarafından yönetilir. Başvuru bilgileri PostgreSQL veritabanında ilgili aday ve ilan ile ilişkilendirilerek saklanır.

D. Jüri Değerlendirme Modülü:

Yöneticiler ilanlara jüri üyeleri atar. Atanan jüriler, adayın yüklediği belgeleri ve sistemin oluşturduğu puanlama dosyalarını inceler. Jüri üyeleri kişisel değerlendirme raporlarını sisteme yükler ve her aday için olumlu/olumsuz karar verir. Değerlendirme tamamlandığında yönetici tüm jüri raporlarını tek noktadan görebilir.

Tüm bu modüller, backend'de REST API olarak sunulmakta, frontend tarafında ise bileşen bazlı sayfalarla etkileşimli bir kullanıcı deneyimi sağlanmaktadır. Modüller arası veri akışı net ve güvenli olacak şekilde tasarlanmıştır.

V. Veri Tabanı Yapısı ve API Katmanı

Projenin arka uç (backend) kısmı, Django ve Django Rest Framework (DRF) kullanılarak geliştirilmiştir. Tüm veritabanı yapısı, **her işlem bir model prensibiyle** ele alınmış ve sistemdeki tüm temel kavramlar (kullanıcılar, ilanlar, başvurular, jüri üyeleri, değerlendirme raporları) için ayrı modeller tanımlanmıştır.

Veritabanı Yapısı:

Bu projede kullanılan veri tabanı, ilişkisel bir yapıya sahiptir ve akademik personel başvuru sürecini dijital olarak yönetmeyi hedeflemektedir. Sistem, kullanıcılar, ilanlar, başvurular, yüklenen belgeler, jüri atamaları ve jüri raporları gibi temel varlıklardan oluşur. Bu varlıklar, aralarındaki mantıksal ilişkiler doğrultusunda modellenmiştir.

A. Kullanılan Tablolar ve İşlevleri:

- **Users: Sistemdeki tüm kullanıcıları barındırır. Kullanıcılar; aday, yönetici, jüri ve admin gibi rollerle tanımlanır. Giriş işlemleri ve yetkilendirme bu tablo üzerinden yürütülür.**

- **Announcements: Üniversite tarafından açılan akademik kadro ilanlarını içerir. İlanlar, kategori (Doçent, Profesör vb.) ve açıklama bilgileriyle birlikte tutulur.**

- **Applications: Adayların yaptığı başvurular bu tabloda yer alır. Her**

başvuru, bir ilana ve bir kullanıcıya bağlıdır. Başvurunun durumu ("Beklemede", "Onaylandı", "Reddedildi") bu tabloda izlenir.

- **Documents: Adayların başvuruları sırasında yüklediği akademik belgeleri barındırır. Belgeler türlerine göre (yayın, atıf, ödül vb.) kategorize edilebilir.**
- **Jury Assignments: Her başvuruya atanmış jüri üyelerini içerir. Bu sayede jüri üyeleri yalnızca kendi sorumlu oldukları başvuruları görüp değerlendirebilir.**
- **Jury Reports: Jüri üyelerinin değerlendirme sonrası hazırladıkları raporları içerir. Değerlendirme kararı ve rapor dosya yolu burada tutulur.**

B. İlişkiler:

- **Bir kullanıcı birden fazla başvuru yapabilir.**
- **Her başvuru yalnızca bir ilana bağlıdır.**
- **Bir başvuruya birden fazla belge yüklenebilir.**
- **Bir başvuruya birden fazla jüri üyesi atanabilir.**
- **Her jüri ataması yalnızca bir jüri raporuyla sonuçlanır.**

Bu yapı sayesinde başvuru süreci sistematik şekilde yürütülebilir, belgeler dijital ortamda saklanabilir ve jüri değerlendirme süreci şeffaf biçimde takip edilebilir.

PostgreSQL veritabanında her tablo, Django modelleri ile temsil edilir. Bu modeller arasında ilişkiler kurulmuş ve veri bütünlüğü sağlanmıştır.

API Katmanı:

Uç Nokta	Yöntem	Açıklama	Erişim
`/api/auth/register/`	POST	Yeni kullanıcı kaydı	Public
`/api/auth/login/`	POST	JWT oluşturma	Public
`/api/ilanlar/`	GET	Aktif ilanları listele	Authenticated
`/api/ilanlar/`	POST	Yeni ilan ekle (admin)	Admin only
`/api/basvurular/`	POST	Aday başvurusu gönder	Aday only
`/api/basvurular/{id}/`	PATCH	Başvuru durumu güncelle (admin)	Admin only
`/api/juri/assign/`	POST	Jüri ataması yap	Manager only
`/api/juri/{id}/report/`	POST	Değerlendirme raporu yükle (jüri)	Juror only

Django Rest Framework kullanılarak, her model için uygun bir **serializer** tanımlanmış ve bu sayede modeller REST API uç noktalarına dönüştürülmüştür. Serializer'lar, model verisinin doğrulanması, JSON formatına çevrilmesi ve istemciye (React frontend) gönderilmesi için kullanılır.

JWT Tabanlı Erişim Kontrolü (Güvenlik):

Sistemde kullanıcıların oturum yönetimi ve erişim denetimi için JWT (JSON Web Token) yöntemi kullanılmıştır. Her kullanıcı giriş yaptığında sunucudan bir JWT token alır ve bu token ile sonraki tüm API isteklerini yapar. Bu sayede kimlik doğrulama ve yetkilendirme işlemleri güvenli bir şekilde yürütülür. Ayrıca rol bazlı erişim denetimleri (örn. yalnızca yönetici ilan oluşturabilir, yalnızca jüri değerlendirme girebilir) sunucu tarafında kontrol edilir.

Bu yapı sayesinde sistem; hem modüler, hem güvenli, hem de farklı istemcilerle (web, mobil vb.) kolayca entegre olabilecek esneklikte bir API katmanına sahiptir.

VI. Frontend Arayüz Tasarımı

Sistemin kullanıcı arayüzü, **React.js** kütüphanesi kullanılarak modüler ve bileşen tabanlı bir yapıda geliştirilmiştir. Uygulama, kullanıcı rollerine göre özelleştirilmiş ekranlar sunarak hem işlevsel hem de sade bir kullanıcı deneyimi hedeflemiştir. Sistemin kullanıcı arayüzü, **React.js** ve **Vite** kullanılarak modüler ve bileşen tabanlı bir yapıda geliştirilmiştir. Uygulama, kullanıcı rollerine göre farklı ekranlar sunarak sade ve işlevsel bir deneyim sağlar. Sayfa yönlendirmeleri react-router-dom üzerinden yapılır ve **JWT token** kontrolü ile korunan alanlara erişim sağlanır.

Global durum yönetimi için **Context API** ve useReducer kullanılmış; kimlik bilgileri tüm bileşenlerde güvenli ve kolayca erişilebilir hâle getirilmiştir. Her bileşen kendi klasöründe ve stil dosyasıyla tanımlanmıştır.

API iletişimi Axios ile services/api.js dosyasında merkezi olarak yönetilir, tüm isteklere otomatik olarak token eklenir. Kullanıcı deneyimi; animasyonlar, bildirim toast'ları ve responsive tasarımla desteklenmiştir.

Modüler Yapı:

Arayüz, tekrar kullanılabilir bileşenlerden (components) oluşturulmuştur. Örneğin giriş formları, ilan kartları, belge yükleme alanları ve değerlendirme panelleri her biri ayrı bileşenler olarak tanımlanmıştır. Bu yapı sayesinde hem kod okunabilirliği artırılmış hem de bakım kolaylaştırılmıştır.

Rol Bazlı Ekranlar:

- **Aday kullanıcılar** için kimlik doğrulama, ilan görüntüleme, başvuru yapma ve belge yükleme ekranları sunulmuştur. Ayrıca adaylar başvuru durumlarını (Beklemede, Onaylandı, Reddedildi) takip edebilir.
- **Admin kullanıcılar**, sistem üzerinde ilanları görüntüleyip yönetebilecekleri bir panel arayüzüne sahiptir.
- **Yönetici kullanıcılar**, yeni ilan açma, jüri atama ve başvuru istatistiklerini görüntüleme gibi gelişmiş yetkilere sahip ekranlara yönlendirilir.
- **Jüri üyeleri**, kendilerine atanan adayların belgelerini sistem üzerinden inceleyebilir ve değerlendirme raporlarını yükleyebilir.

API Entegrasyonu – Axios:
Arayüz ile backend API arasındaki veri iletişimi için **Axios** kütüphanesi kullanılmıştır. Kullanıcıların gerçekleştirdiği işlemler (form gönderimi, belge yükleme, giriş, başvuru)

Axios aracılığıyla sunucuya gönderilmiş ve API yanıtları dinamik olarak ekrana yansıtılmıştır.

VII. Güvenlik ve Kimlik Doğrulama

Sistemde kullanıcıların güvenli bir şekilde giriş yapabilmesi ve sadece yetkili alanlara erişebilmesi için kimlik doğrulama ve yetkilendirme süreçleri özenle yapılandırılmıştır. Bu amaçla **Django Rest Framework Simple JWT** kütüphanesi kullanılarak token tabanlı oturum yönetimi uygulanmıştır.

- **Kayıt/Giriş** formları, kullanıcının rolüne göre arayüz sağlar.
- Başarılı girişte JWT, localStorage'a kaydedilir.
- Axios Authorization: Bearer <token> başlığı otomatik eklenir.
- AuthContext ile user ve role durumu yönetilir; korumalı rotalar erişim kontrolü yapar.

JWT Tabanlı Kimlik Doğrulama:

Kullanıcılar giriş yaptıktan sonra sunucu tarafından kendilerine bir **access token** ve **refresh token** verilir. Bu tokenlar, kullanıcının sistemde kim olduğunu doğrulamak için API istekleri sırasında kullanılır.

VIII. Test Süreci ve Sonuçlar

Sistemin kullanıcı arayüzü, **React.js** kütüphanesi kullanılarak modüler ve bileşen tabanlı bir yapıda geliştirilmiştir. Uygulama, kullanıcı rollerine göre özelleştirilmiş ekranlar sunarak hem işlevsel hem de sade bir kullanıcı deneyimi hedeflemiştir.

- **Birim testleri**: Serializer ve ViewSet katmanında Django test çerçevesi ile.
- **Manuel test**: Chrome/Firefox üzerinde UI işlevsellik doğrulaması.

Dağıtım:

- **Gunicorn + Nginx (backend)**
- **React statik build (frontend)**
- **Çevresel değişkenlerle esnek veritabanı yapılandırması**

IX. Karşılaşılan Zorluklar ve Çözümler

- **Rol bazlı yönlendirme**: React Router'da PrivateRoute bileşeni geliştirildi.
- **JWT senkronizasyonu**: Axios interceptor ile token yenileme mekanizması eklendi.
- **Dosya tipi validasyonu**: Multer ile yüklenen belgelerin uzantıları ve boyutları kontrol edildi.
- **PDF şablonlama**: ReportLab'da dinamik tablo oluşturma kodu optimize edildi.

X. Sonuç ve Gelecek Geliştirme Önerileri

Geliştirilen sistem, akademik personel alım süreçlerini uçtan uca otomatikleştirerek şeffaflık ve verimlilik sunmuştur. Gelecekte:

- E-posta bildirimleri
- Analitik dashboard ile istatistiksel raporlama
- **OCR entegrasyonu** ile belge doğrulama
- **Mobil uygulama eklentisi**

gibi ek özelliklerle genişletilebilir.

Kaynakça

[1] Kocaeli Üniversitesi Akademik Atama Yönergesi.

[2] Django REST Framework, "Django REST framework: Serialization & ViewSets,"

[3] React – A JavaScript library for building user interfaces, "React Documentation,"

[4] React Router, "React Router Documentation," .

[5] Axios, "Promise based HTTP client for the browser and node.js,"

[6] Vite, "Next Generation Frontend Tooling,".

[7] PostgreSQL Global Development Group, **PostgreSQL Documentation**

[8] B. Jones ve başka, "RFC 7519: JSON Web Token (JWT),"

[9] JWT(JSON Web Tokens) Nedir? Nasıl Çalışır? | Medium

[10] ReportLab, **ReportLab User Guide**