Akademik Başvuru Sistemi

Emre Bektaş  
Kocaeli/Türkiye

221307063@kocaeli.edu.tr

Ali Balcı  
Kocaeli/Türkiye

221307098@kocaeli.edu.trVelihan Mutlu

Kocaeli/Türkiye

221307032@kocaeli.edu.tr

*Özet: Bu raporda başta NodeJS + React kullanarak ve Kocaeli Üniversitesi atama yönergesindeki isterler göz önünde bulundurularak tasarlanılan bir web uygulamasının nasıl geliştirildiği anlatılıyor. Site tasarlanırken ne gibi zorluklarla karşılaşıldığı ve bu zorlukların nasıl çözüldüğü ana konusu.*

# Problemi Belirleme

Atama yönergesi ve isterle göre bizden 4 farklı kullanıcının erişip farklı işlemler yapabildiği bir entegre web sistemi geliştirmemiz istendi. Bu sistemi geliştirmek için ihtiyacımız olan kütüphaneler üstüne konuşup backend için NodeJS, frontend için ise React kullanmaya karar verdik. İsterleri ve atama yönergesini okuyup özümsedikten sonra sistemin taslağını çıkararak işe başladık ve problemi belirleme kısmını böylece tamamlamış olduk.

# Hazırlanma Aşaması

## NodeJS ve React kütüphanelerinin nasıl çalıştığını anlamak için çalışma

Daha önce bu kütüphanelerle tecrübemiz olmadığı için projemizde kullanacağımız teknolojiler hakkında araştırmalar yaptık. Yaklaşık 10 günlük bir süreç içerisinde backend kısmıyla ilgili bilgiler edindik ve uygulamayı ayağa kaldırmaya ve basit router işlemlerini yapmaya yetecek deneyimi sağladık.

## Frontend için entegre teknoloji araştırması

Frontend için React’a entegre ChakraUI adlı bir tool kullanarak HTML/CSS yerine basit taglarla UI design edebildiğimiz bir tool kullandık. Bu tool bizi frontendi tasarlarken hem hızlandırdı hem de elverişli bir arayüz tasarlayabilmemizi sağladı.

# Projeye Başlangıç

Öncelikle kullanıcının gerçek bir birey olduğunu anlamak için Nüfus Müdürlüğünün Sağladığı api bağlantısını soap kütüphanesiyle gerçekleştirdik. Kullanıcı sisteme kayıt olurken gerçek isim, soyisim, doğum tarihi rolleriyle api’ya bir request yolluyor ve eğer nvi api’ında bu şahıs mevcutsa kayıt başarıyla gerçekleştiriliyor. Bu işlemler users.js adlı router’da gerçekleşiyor ve gerekli auth işlemleri sağlanıyor.

Kullanıcı girişi tamamlandıktan sonra sıra diğer kullanıcıları tanımlamaya geldi. Yönetici, admin ve juri kullanıcıları için Users adlı bir veritabanı modeli oluşturuldu. MongoDB fonksiyonlarıyla bir register routerı oluşturarak bu kullanıcıları Postman aracılığıyla veritabanına kaydettik. Bu sayede artık rol yönetimi daha rahat tanımlanabildi. Login işlemleri için routerlar tasarlandıktan sonra giriş işlemlerinin sonuna geldik.

## Kullanıcı yetkilendirmeleri

Gerekli yetkilendirmeleri yönetmek için JWT teknolojisini kullandık. Bu teknolojiyle her tanımlı rol sisteme login attığında kendilerini eşsiz bir token atanıyordu. Bu token JWT\_SECRET alanında saklanıp **Authorization** alanı “Bearer <token>” biçiminde okunuyordu. Her routerda tanımlı olan auth değişkeniyle kullanıcıların erişimi olmadığı alanları düzenlemesi veya değiştirmesi bu sayede engellendi.

## Backend routerları tasarlanması

* Her router’ı, dosya adının temsil ettiği kök yol altında (/applications, /criteria, /publications vb.) dinamik olarak mount ettik; böylece yeni bir modül eklerken fazladan yapılandırma yapmamıza gerek kalmadı.
* Korumalı tüm uç noktalarda **authMiddleware**’i çalıştırdık: “Bearer <JWT>” üstbilgisini doğruladıktan sonra kullanıcı kimliğini ve rolünü req.user’a ekledik, geçersiz veya eksik token’ı ise 401/403 yanıtlarıyla engelledik.
* Rol-tabanlı erişim kurallarını rota düzeyinde tanımladık: Aday’ın yalnızca kendi başvurularını görmesine, Jüri’nin atanmış başvuruları incelemesine, Yönetici / Admin’in yönetim uç noktalarına erişmesine imkân verdik.
* Dosya yüklemeleri için **multer** kullandık; her kullanıcıya özel dizinler oluşturduk ve PDF boyutlarını başvurular için 5 MB, jüri raporları için 10 MB ile sınırladık.
* /apply rotasında yüklenen her dosyanın meta verisini JSON dizisiyle eşleştirdik; eşleşme sayısı tutmazsa yüklemeyi iptal edip geçici dosyaları sildik.
* Yayın yüklerken PDF’leri önce geçici klasöre kaydedip ardından Firebase Storage’a aktardık; işlem bitince geçici dosyayı sildik ve aynı DOI’nin yinelenmesini engelledik.
* criteria uç noktalarında alan-grup kriterlerini getirip oluşturma / güncelleme işlemlerini yalnızca Admin / Yönetici rollerine açtık; güncellemelerde fieldGroup değerinin değiştirilememesini sağladık.
* PDF üretimi için /generate-pdf/atama-yonergesi rotasında **PDF-Lib** kullandık; istek gövdesinden gelen kişisel, yayın ve istatistik verilerini şablon form alanlarına yerleştirdik ve Türkçe karakter desteği için Noto Sans fontunu gömdük.
* Jüri değerlendirmesi tamamlandığında çoğunluk oyuna göre başvurunun final\_decision alanını güncelledik ve durum geçmişine “Onaylandı/Reddedildi” kaydı ekledik.

## Veritabanı modelleri

Veritabanı modellerini tasarlamak için MongoDB, arayüzü kullanabilmek için de Studio3T uygulamasını kullandık. Modelleri hazırlarken isterleri ve atama yönergesindeki kriterleri göz önünde bulundurduk.

Aşağıda model dosyalarının isimleriyle beraber ne işe yaradıkları gösterilmekte.

## Modeller

* **Application**
* Adayın, seçtiği ilana yaptığı her başvuruyu temsil ettik; durum geçmişiyle (Beklemede → Onaylandı/Reddedildi) tam izlenebilirlik sağladık.
* Jüri üyelerini, rolleri (Başkan/Üye) ve her üyenin değerlendirme-raporlarını aynı dokümanda topladık; çoklu imza sürecini sadeleştirdik.
* checkJuryEvaluationStatus() ile oy çokluğunu otomatik hesaplayıp başvuruyu doğrudan “Onaylandı” ya da “Reddedildi” durumuna taşıyoruz.
* Zaman damgalarını (created\_at, updated\_at) otomatik tuttuk; böylece raporlarda sürüm takibi sorunsuz ilerliyor.
* **ApplicationCriteria**
* Her ana alan grubu (sağlık-fen, eğitim-sosyal, hukuk-ilahiyat, güzel sanatlar) için minimum yayın, puan ve dil skoru eşiklerini tanımladık.
* createDefaultCriteria() metoduyla varsayılan eşikleri ilk kurulumda tek seferde oluşturuyoruz; elle veri ekleme derdini bitirdik.
* Kabul edilebilir yabancı dil sınavlarını ve taban puanını aynı yerde tutarak İK-taraflı doğrulamayı basitleştirdik.
* **PositionCriteria**
* Kademe bazlı (Dr. Öğr. Üyesi / Doçent / Profesör) ayrıntılı puan matrisini tanımladık; kategori → alt kriter → puan aralığı yapısıyla esnek skor hesaplamasına zemin hazırladık.
* Her alt kriterde yıl sınırı, yazar sayısı çarpanı, etki faktörü bonusu gibi kuralları sakladık; böylece puan motoru tek yerden yönetilebilir oldu.
* **ApplicationDocument**
* Adayın yüklediği her belgeyi tür, dosya yolu, boyut ve zengin metadata (etki faktörü, atıf, konferans tipi vb.) ile kaydettik; denetim sırasında tüm teknik veriye tek sorguda erişiyoruz.
* Belge bazlı puanı (calculated\_points) ön-hesaplayıp başvurudaki toplam puan dağılımına doğrudan besledik.
* **Publication**
* DOI’si benzersiz makaleleri, kategori-indeks bilgisiyle sakladık ve kayıt öncesi otomatik puan hesabı yaptık; manuel puanlama hatasını sıfırladık.
* getPublicationStats() ile A1-A2, A1-A4, ana yazar adedi, toplam puan gibi istatistikleri tek pipeline’da topluyoruz; rapor üretimini hızlandırdık.
* **Evaluation**
* Her jüri üyesi-başvuru ikilisi için tekil kayıt kısıtlaması getirdik (compound index); aynı başvuruya çifte değerlendirme engellendi.
* complete() metoduyla puan, rapor ve yorumları tek adımda kaydedip durumu “Tamamlandı” yapıyoruz; süreç izleme netleşti.
* **User**
* TC Kimlik No’yu benzersiz anahtar yaptık, şifreleri sifre\_hash alanında BCrypt ile sakladık; KVKK ve güvenlik gereksinimlerini karşıladık.
* Rol bazlı (Aday / Admin / Yönetici / Jüri) erişim yetkilerini model düzeyinde belirledik; yetkilendirme middleware’ine doğrudan besledik.

## Frontend Tasarlanması

Akademik Personel Başvuru Sistemi’nin React-tabanlı arayüzü, Chakra UI bileşen kütüphanesiyle tasarlanmış tek-sayfa bir uygulama (SPA) yapısına sahip. İstemci tarafında rol bazlı yönlendirme, JWT tokene dayalı korumalı rotalar ve zengin bileşen mantığıyla ayrılmış ekranlar sayesinde aday, jüri, yönetici ve admin rollerinin her biri kendi görevlerine odaklanan, tutarlı bir kullanıcı deneyimi elde ediyor. Uygulama; kök bileşende tema/sağlayıcı kurulumunu, “App Router” katmanında erişim kontrolünü ve her ekranda Axios ile arka-uç API’sine yapılan talepleri kapsayan net bir modüler mimari izliyor.

## Frontend source dosyaları ve işlevleri

**Main.jsx** : Uygulamanın giriş noktasıdır. React 18’in createRoot API’si ile <App/> bileşenini DOM’a bağlar, global stil ve durum yönetimi için ChakraProvider’ı sarmalar; bu sayede tüm alt bileşenler temaya ve Chakra UI bileşenlerine erişebilir.

**App.jsx:** Merkezî yönlendirme katmanıdır. react-router-dom ile tanımlanan rotalar, ProtectedRoute bileşeni aracılığıyla JWT’nin varlığını ve kullanıcının rol bilgisini kontrol ederek yetkisiz erişimleri otomatik olarak “/login” sayfasına yönlendirir. Aday, Jüri, Yönetici ve Admin ekranları ayrı ayrı korunduğu için her rol sadece kendi izin verilen sayfaları görebilir.

**Login.jsx:** Çok sekmeli (Tabs) giriş formudur. Kullanıcı TC Kimlik No ve şifresini gönderir; başarılı yanıt geldiğinde JWT ve kullanıcı verileri localStorage’a kaydedilir, kullanıcının rolüne göre uygun ekrana yönlendirilir. Rol sekmesi değiştiğinde form otomatik olarak ilgili role resetlenir; hatalı kimlik/şifre veya rol uyuşmazlığında bilgilendirici toast/alert mesajları gösterilir.

**Register.jsx:** Adayların sisteme ilk kez kayıt olması için tasarlanmıştır. Form verileri doğrulanıp arka-uca gönderildikten sonra başarı durumunda “Kayıt Başarılı” toast’ı üretir. Doğum yılı alanı dinamik olarak 1960’tan günümüze kadar oluşturulur; şifre tutarsızlıkları istemci tarafında engellenir.

**Main-page.jsx:** Genel (oturum açılmamış) ziyaretçilere aktif ilanların kart görünümlü bir listesini sunar. Sayfa ilk yüklendiğinde /api/applications/public-active-posts uç noktasından ilanları çeker; yükleme sırasında iskelet (Skeleton) kartlar gösterir, ilan yoksa bilgi alert’i verir. “Detayları Görüntüle” butonu ilanı modala taşır; “Giriş” ve “Kayıt” butonları yönlendirme işlevi görür.

**Aday-ekrani.jsx:**

Aday oturumu açıldığında görünen panodur.

* Aktif ilanları ve kullanıcının önceki başvurularını ayrı API çağrılarıyla getirir.
* Başvuru süresi denetimleri (başlamamış/bitmiş) ve “zaten başvuruldu” kontrolü içerir.
* İlan detayı modaldan görüntülenir; “Başvur” tıklandığında seçilen ilan bilgileri localStorage’a yazılır ve aday başvuru formu rotasına yönlendirilir.
* Başvuruların durumuna göre renkli Badge bileşenleri (Beklemede → turuncu, Kabul → yeşil vb.) dinamik olarak üretilir.

**Juri-ekrani.jsx**

Jüri üyelerine atanan başvuruların değerlendirme merkezidir.

* assigned-applications, application/:id ve view-document/:id uç noktalarına çağrılar yaparak başvuru, belge ve yayın verilerini yükler.
* Jüri raporu PDF’i seçilip FormData ile sunucuya gönderilir; karar (olumlu/olumsuz) ve yorumlar aynı istekte iletilir.
* Tamamlanan ve bekleyen değerlendirme sayıları üst kısımda istatistik olarak gösterilir; PDF ön izleme modalla açılır.

**Yonetici-ekrani.jsx:** Yönetici rolünün ilan-sonrası süreçleri yürüttüğü paneldir.

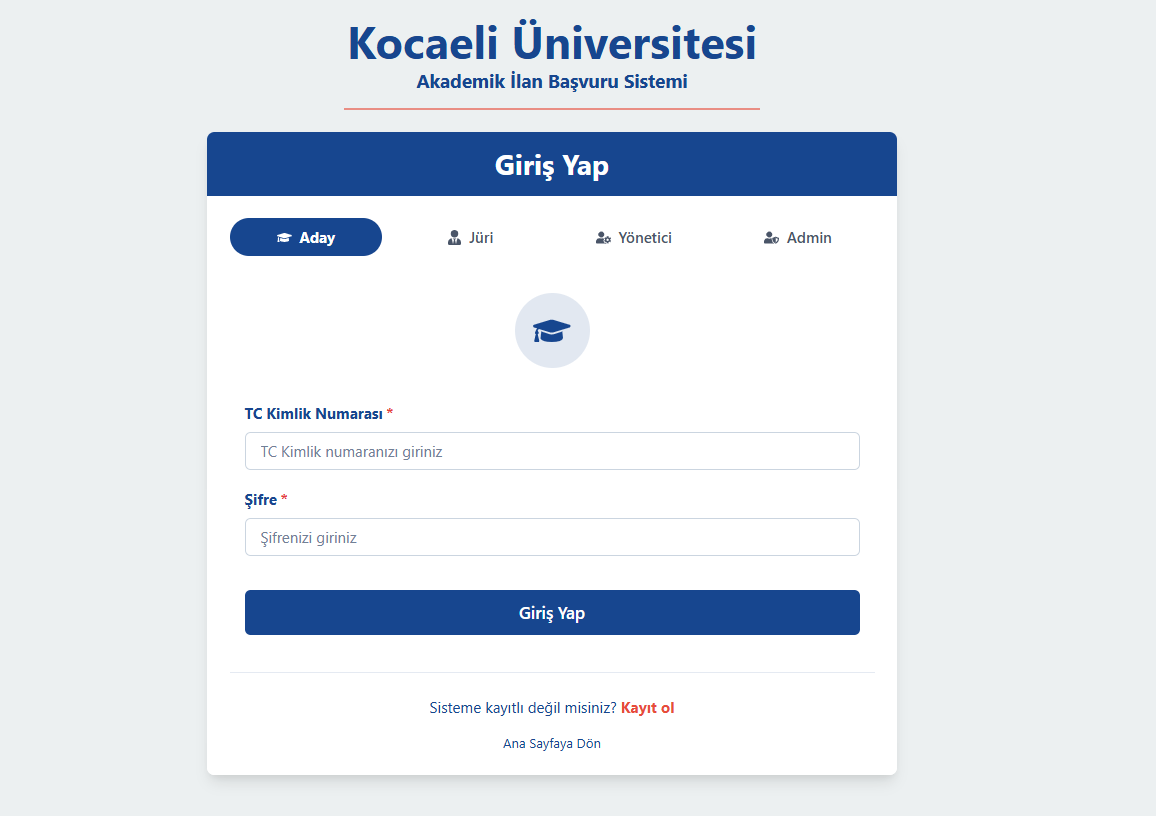
* İstatistik kutucuklarında toplam ilan, toplam başvuru, jüri sürecindeki başvuru sayıları gibi verileri gösterir.
* Atama kriterlerini (pozisyon türü, puan eşikleri vs.) oluşturma/düzenleme/silme, ilanlara ait başvurulara jüri atama ve kriterlere göre puan hesaplama akışlarını içerir.
* Çok adımlı modal ve onay diyaloglarıyla kriter/jüri yönetimi yapılır; PDF çıktısı (“Atama Yönergesi”) oluşturmak için sunucuya JSON gönderip blob yanıtını indirir.

Admin-ekrani.jsx: Sistemin en yetkili panelidir; ilan yaşam döngüsü ve kullanıcı yönetimini merkezileştirir.

* İlan oluşturma/güncelleme formu ile başlık, açıklama, akademik kademe, alan grubu, tarih aralığı ve gerekli belgeler girilir; mevcut ilanları tablo ve kart görünümlerinde listeler, durum (Açık/Kapalı/Tamamlandı) değiştirilebilir veya ilan silinebilir.
* Başvuru istatistikleri, bekleyen başvurular, kullanıcı listesi ve PDF belgesi yükleme/indirme eylemleri tek panelde toplanmıştır.
* React hooks üzerinde detaylı durum yönetimi (useState, useEffect, useDisclosure) ve geniş toast geri bildirimi ağı kullanır.

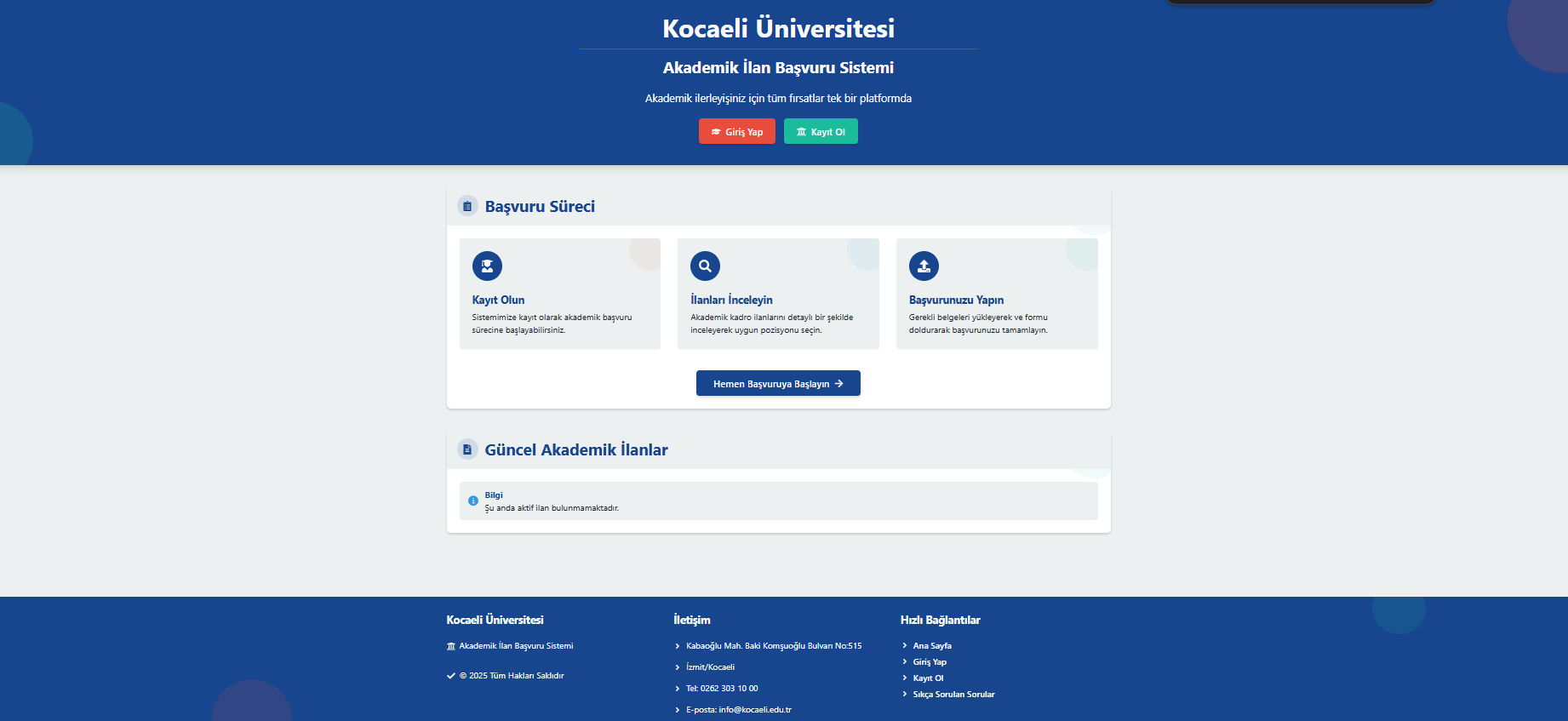
## Sistem UI’ı görüntüleri

1. Giriş ekranı:



#### Bu ekrandan 4 farklı kullanıcı girişi sağlanır. Ayrıca alttaki Kayıt Ol tuşuyla aday kullanıcı kaydı gerçekleştirilebilir.

1. Ana Ekran



b) Bu ekranda kullanıcılar mevcut ilanları görerek giriş sayfasına yönlendirilirler.

##### Kaynakça

[1] Facebook, “React: A JavaScript library for building user interfaces,” <https://react.dev/>, accessed Apr. 30, 2025.

[2] Chakra UI, “Chakra UI Documentation,” <https://chakra-ui.com/>, accessed Apr. 30, 2025.

[3] Node.js Foundation, “Node.js v20 Documentation,” <https://nodejs.org/en/docs>, accessed Apr. 30, 2025.

[4] Express Team, “Express 4.x API Reference,” <https://expressjs.com/en/4x/api.html>, accessed Apr. 30, 2025.

[5] MongoDB Inc., “MongoDB Manual (v7.0),” <https://www.mongodb.com/docs/manual/>, accessed Apr. 30, 2025.

[6] J. Jones, M. Jones, and N. Sakimura, “RFC 7519: JSON Web Token (JWT),” Internet Engineering Task Force, Apr. 2015.

[7] T.C. İçişleri Bakanlığı Nüfus ve Vatandaşlık İşleri Genel Müdürlüğü, “Kimlik Doğrulama Servisi (SOAP) Entegrasyon Kılavuzu,” Ankara, Türkiye, 2023.

[8] Google LLC, “Firebase Storage Documentation,” <https://firebase.google.com/docs/storage>, accessed Apr. 30, 2025.

[9] Ender Can NodeJS dersleri, <https://www.youtube.com/watch?v=fwY5DQDtcSU&list=PLr0qvr5J-KIOrT9rCbjEjVgz7CTsjbgA->