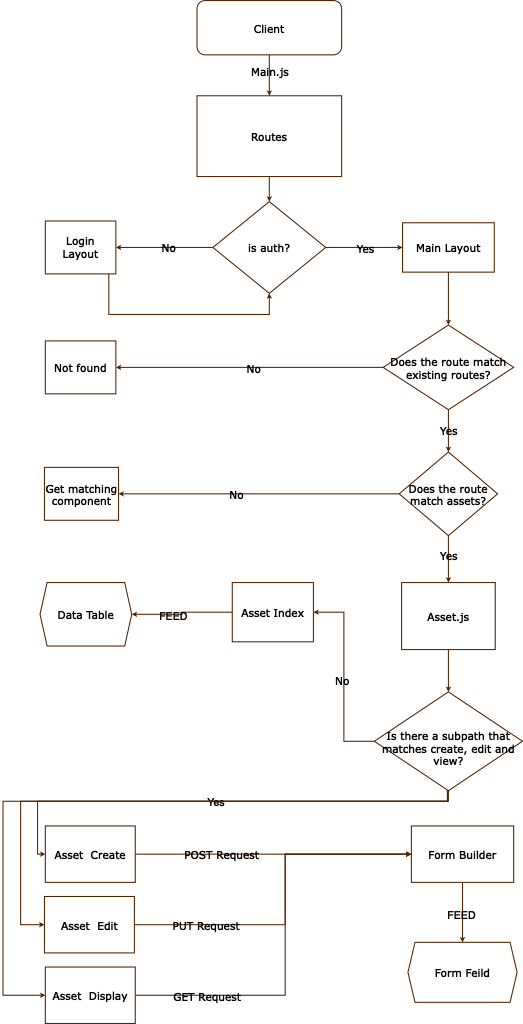
# Projelerde Kullanılacak Kütüphaneler

Bu bölümde projelerde yaygın olarak kullanılacak bazı kütüphaneler ve kullanım amaçlarına dair özet bilgiler yer almaktadır.

|  |  |
| --- | --- |
| **Kütüphane/Framework Adı** | **Kullanım Amacı** |
| **react** | Tüm frontend projelerde kullanılacak UI kütüphanesi |
| **react-router-dom** | Projelerdeki routing işlemleri için kullanılacaktır |
| **react-redux** | Global state management için kullanılacak kütüphane |
| **@reduxjs/toolkit** | Global state management için kullanılacak yardımcı kütüphane |
| **@tanstack/react-query** | HTTP isteklerini yönetmek için kullanılacaktır |
| **react-data-table— component** | Veri tabloları için kullanılacaktır |
| **formik** | Güncel projelerde form işlemleri için kullanılacaktır |
| **axios** | HTTP isteklerini yönetmek için kullanılacaktır |
| **@mui/material** | Hazır React componentleri sunmaktadır |
| **@radix-ui** | Hazır React componentleri sunmaktadır |
| **tailwindcss** | CSS kütüphanesi |
| **react-icons** | Birçok ikon kütüphanesini bir araya getirmiş olan react kütüphanesi |

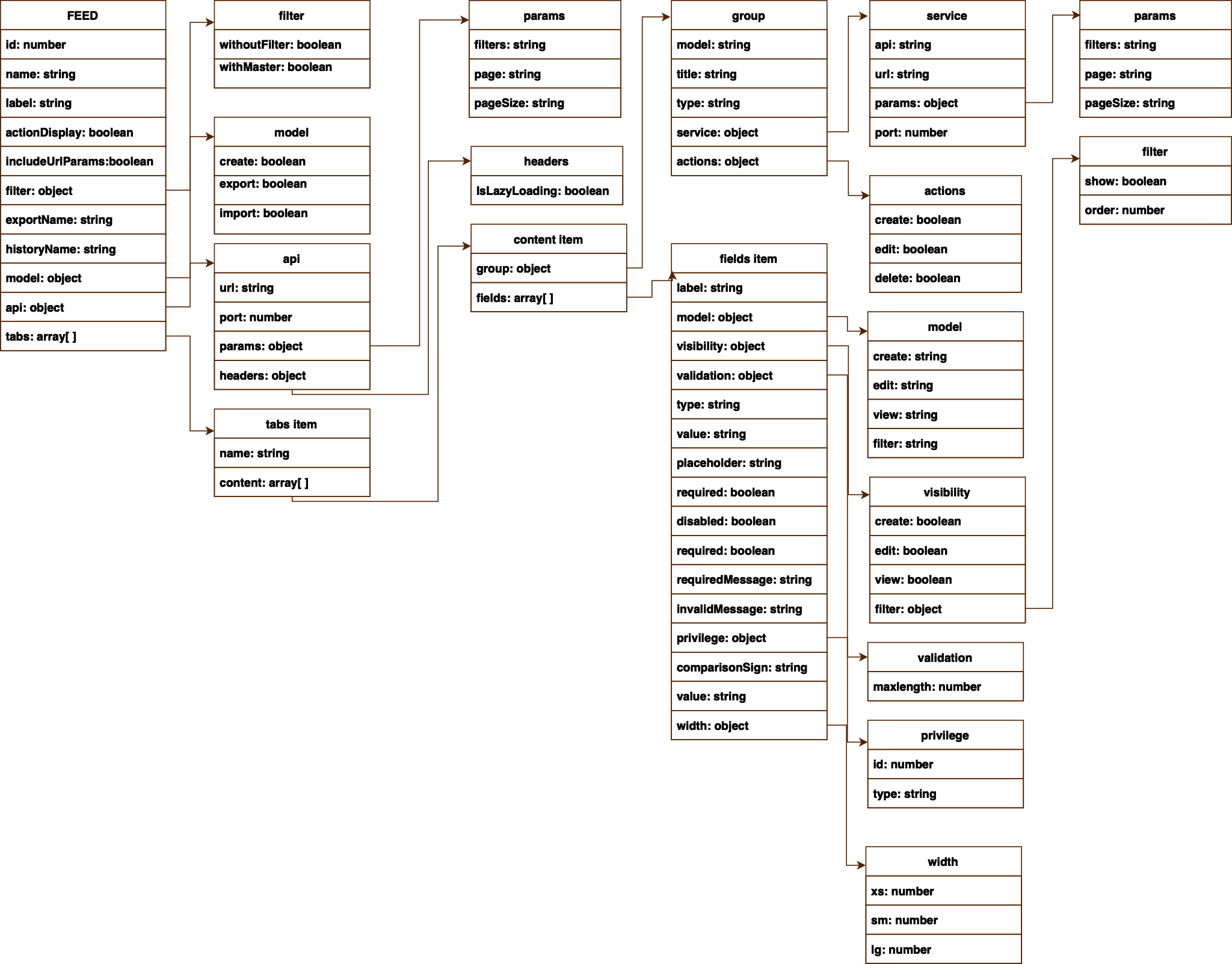
# Projenin mimarisi

Bu bölümde projelerde yaygın olarak kullanılacak örnek bir mimari yapıya ait diyagrama yer verilmiştir:



# Temel Feed yapısı

Bu bölümde temel feed yapısı ve sık kullanılacak key’ler hakkında bilgi verilmiştir:



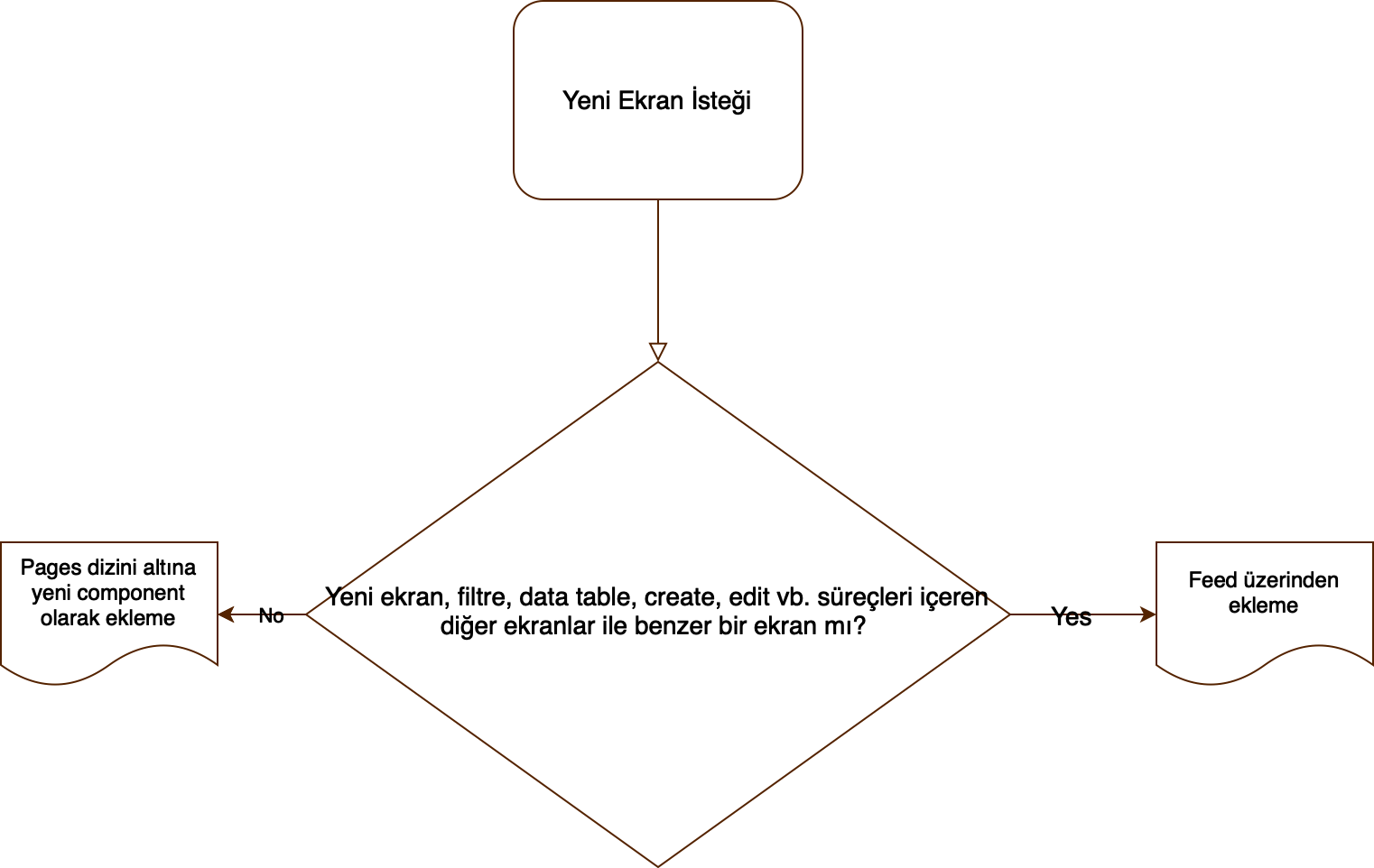
|  |  |
| --- | --- |
| **Key** | **İşlev** |
| name | Sayfanın Route tanımıdır ayrıca uıniqe olmak zorundadır |
| label | Sayfanın ekranda görünecek olan adı |
| actionDisplay | Data table’da backend’den gelen EntityPermissions’a göre işlemler kolonunun gösterilmesi |
| includeUrlParams | Ekran açıldığında ilk atılan sorguda url’deki parametlerin dahil edilmesi |
| filter | Ekran açıldığında default atılacak olan sorgu için tanımlamalar |
| exportName | AssetIndex’te iken ilgili ekrandaki verilerin export edilmesi durumu için |
| historyName | Tarihçe için gidilmesi gereken adres tanımı |
| model | Ekranın AssetEdit,Create ve View durumu için tanımlamalar |
| api | Api bilgilerinin tanımlanması için |
| tabs | Her ekran en az bir tab’dan oluşmaktadır, daha fazla varsa array içerisine eklenir. |
| withoutFilter | Index modunda iken filtreleme işlemlerinde filtre zorunluluğu durumu, true ise ekran ilk açıldığında filtre sorgusu atar |
| withMaster | Index modunda filtre zorunluluğu durumu, false ise master identifier ile istek atılabilir |
| model->create | Create Ekranına geçiş izni, false ise buton görünmez |
| model->export | Index modunda ekrandan rapor alma |
| model->import | Index modunda excel aktarma durumu, true ise excel import işlemi yapılabilir |
| api->url | Endpoint tanımı |
| api->port | Varsa endpointin çalıştığı port |
| api->params | Filtre sorgularına default eklenecek parametreler |
| api-headers | Sorgularda readers içerisine eklenmek istenen parametreler |
| tabs->name | Tab’lı yapıda Tab ismi, Tab adı “Genel” ise ekranda görünmez, diğer isimler görünür |
| tabs->content | Tab içeriğinin tutulduğu obje |
| params->filters | Filtre sorgularında Parametreye eklenecek filtreler |
| params->page | Filtre sorgularında page değeri |
| params->pageSize | Filtre sorgularında default pageSize |
| headers->isLazyLoadig | Filtre sorgularında isLazyLoading değişken durumu |
| content->group | Bir Tab içerisinde birden fazla content olabilir. Her content için varsa eğer özel Group tanımlamaları yapılır, örneğin aynı ekranda bir grup için farklı bir endpointe istek atmak yada grup ismi belirlemek gibi. Spesifik bir durum yoksa null olarak eklenir |
| content->fields | Eklenecek fieldların tutulduğu yer, Dizi olarak tutar. |
| group->model | Gösterilcek yada düzenlenecek veriler nested bir object içinde ise tanımlanır. Örneğin body:{  ornekNestedObject:{ Deger1:null, Deger2:null  }  }  gibi, bu durumda model ismi ornekNestedObject olur, fieldlar da bu obje altındaki değerler ile eşleşir |
| group->title | Grubun gösterilecek ismi |
| group->type | Grup tipi, tablo componenti yada form tipinde olabilir. |
| group->service | Varsa eğer gruba özel servis tanımının yapıldığı yer, yapılmazsa parenttan beslenir. |
| group->actions | Grubun create, edit, delete gibi işlemleri için izin tanımlaması |
| field->label | Ekranda gösterilecek field ismi |
| field->model | field’ın create, edit, view gibi durumlarda denk geldiği değer adının tanımlandığı yer |
| field->visibility | field’ın create, edit, view gibi durumlarda görünüp görünmemesi durumu. Örnek olarak bir field sadece create ekranında görünmesi isteniyorsa create değeri true olur, diğer değerler false olarak tanımlanır. |
| field->validation | Validasyon tanımlamalar |
| field->type | field Tipi, örnek olarak, selecet, text, checkbox gibi |
| field->value | Varsa eğer field’ın default değer tanımı |
| field->placeHolder | field’a ati place holder değeri |
| field->required | field'ın zorunlu olma durumu |
| field->disabled | field'ın default olarak disable olma durumu |
| field->requiredMessage | Zorunlu olduğu durumlarda doldurulmadığında gösterilecek mesaj |
| field->invalidMessage | Geçersiz değer durumlarında gösterilecek olan mesaj |
| field->privilege | field’a ait rol tanımı |
| field->comparisionSign | field filtrede tanımlı bir field ise eğer filtreleme durumunda sorguya eklenecek değer ile karşılaştırma için eklenen işaret. Örneğin değer  == 0 yada değer >= 0 gibi |
| field->width | field'ın ekrandaki width değer tanımlaması |
| field->model->create | Create ekranında field’ın body içerisinde denk geleceği key tanımı |
| field->model->edit | Edit ekranında field’ın body içerisinde denk geleceği key tanımı |
| field->model->view | View datatable gösterilecek olan değerin key tanımı |
| field->model->filter | Index’te iken filtreleme durumunda field’ın denk geleceği key |
| field->visibility->create | Create ekranında gösterilmesi izni |
| field->visibility->edit | Edit ekranında gösterilmesi izni |
| field->visibility->view | Index ekranında gösterilmesi izni |
| field->visibility->filter | Index ekranında filtrede gösterilmesi tanımı |
| field->visibility->filter->show | Index ekranında filtrede gösterilmesi izni |
| field->visibility->filter->order | Index ekranında filtrede gösterileceği konum |
| field->validation->maxLength | Maksimum uzunluk validasyonu |
| field->privilege->id | Tanımlanacak olan Role ait Id değeri |
| field->privilege->type | Role dair izin, örneğin field’ın gösterilmesi yada düzenlenebilmesi gibi |
| field->width->xs|sm|lg | Responsive durumlarda field’a özel width tanımı |
| group->service->url | Gruba özel servis varsa endpointi tanımlaması |
| group->service->params | Ekran açıldığda atılacak sorgunun alacağı değer, Örneğin PersonId değeri ya da ContactId değeri gibi denk geleceği değer tanımı |
| group->service->port | Servis port tanımlaması |
| group->actions->create | Gruba ait create izni |
| group->actions->edit | Gruba ait edit izni |
| group->actions->delete | Gruba ait delete izni |
| group->service->Params->filter | group->service->params’da açıklanan filter tanımı |
| group->service->params->page | Default sorguya eklecek page değeri |
| group->service->params-  >pageSize | Default sorguya eklecek pageSize değeri |

# UI Component Listesi

Bu bölümde projelerde yaygın olarak kullanılacak componentler ve feed’teki karşılıkları verilmiştir.

|  |  |
| --- | --- |
| **Component Adı** | **Feed Karşılığı** |
| FileInput | file |
| ImageFile | imageFile |
| CheckBoxInput | checkbox |
| RadioInput | radio |
| SelectInput | select |
| AsyncSelectInput | asyncSelect |
| Select2Input | select2 |
| TextTaggedInput | textTagged |
| TextTaggedSuggestInput | textTaggedSuggest |
| TextAreaInput | textarea |
| StepInput | step |
| ActionInput | action |
| PasswordInput | password |
| DateTimeInput | date |
| DateTimeInput | datetime |
| NumberInput | number |
| TinyEditor | tinyeditor |
| DummyInput | dummy |

# Sıfırdan bir ekran nasıl eklenir?

Projelerde yaygın kullanılan mimariye göre yeni bir ekran eklerken aşağıdaki gibi bir karar mekanizması olur.

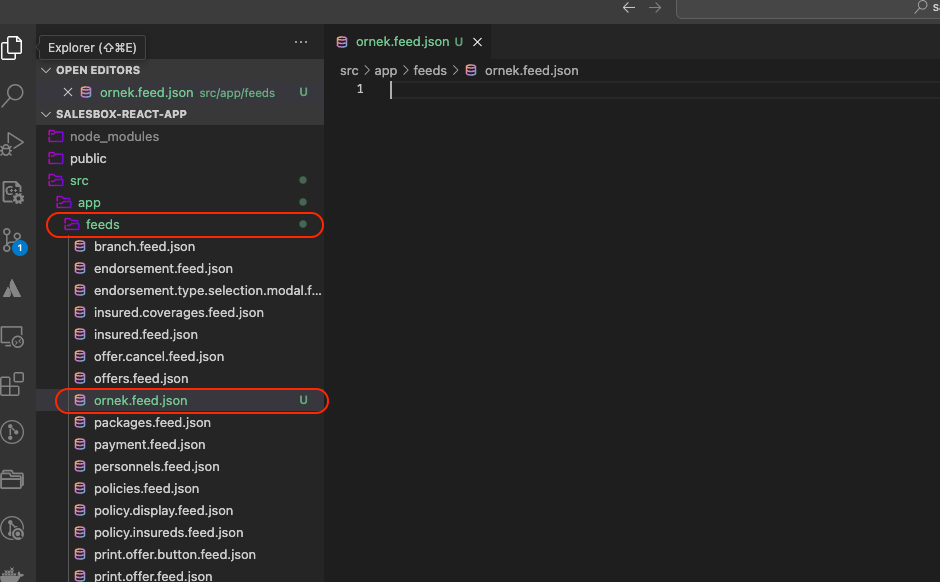
Ekran klasik feed yapısına uygunsa feed dizini altına yeni bir json uzantılı dosya olarak eklenir ve yukarıda anlatılan feed özelliklerine göre düzenlenir. Son olarak ekran sidebar’da görüntülenecek bir ekran ise sidebar.js üzerine id’si yazılarak eklenir. Ekran diğer feed’ler içerisinde kullanılacak bir ekran ise yapıya uygun olacak şekilde entitiyName key’i ile referans verilerek tanımlanır.

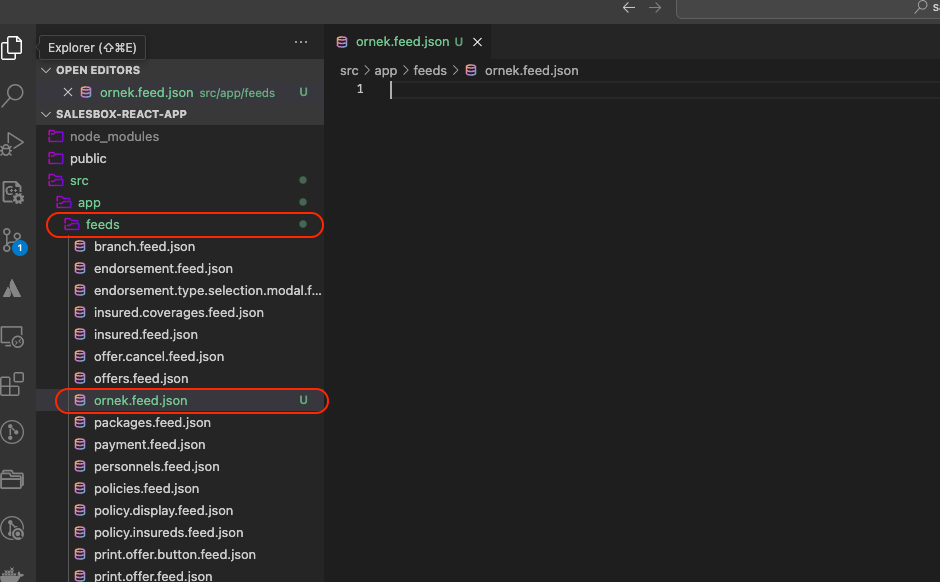
Ekran feed yapısına uygun olmayan spesifik özellikler barındıran bir ekran olacaksa eğer, pages dizini altında yeni bir component oluşturularak eklenir. Eklenen ekran routes altında handle edilerek kullanılabilir.

## Feed üzerinden ekran ekleme örneği

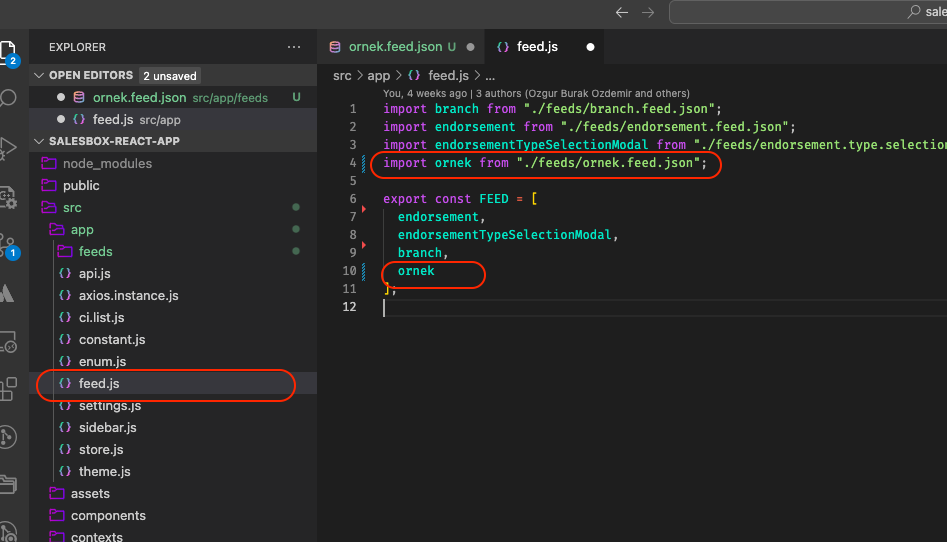
Bu bölümde feed.js üzerinden örnek bir ekran ekleyeceğiz.

1.Feeds klasörü altına ornek.feed.json tanımlaması yapılır. Ardından yeni feed dosyası içerisine gerekli feed tanımlamları yapılır.

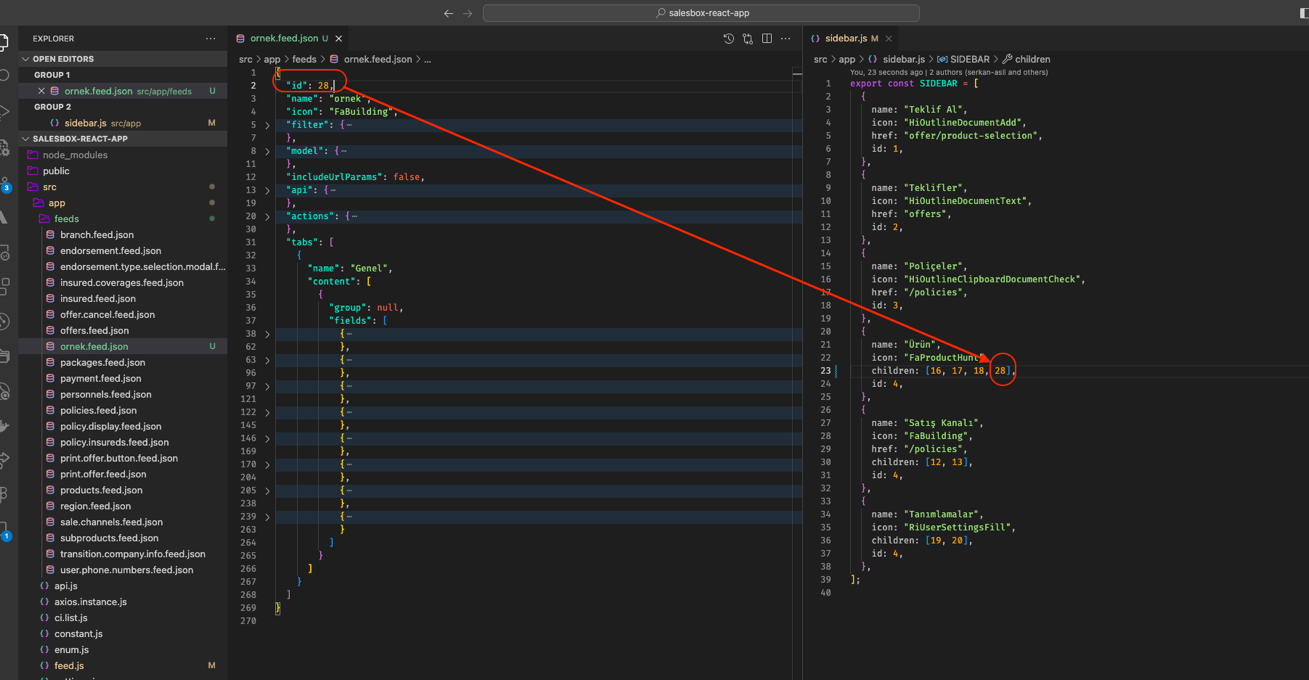




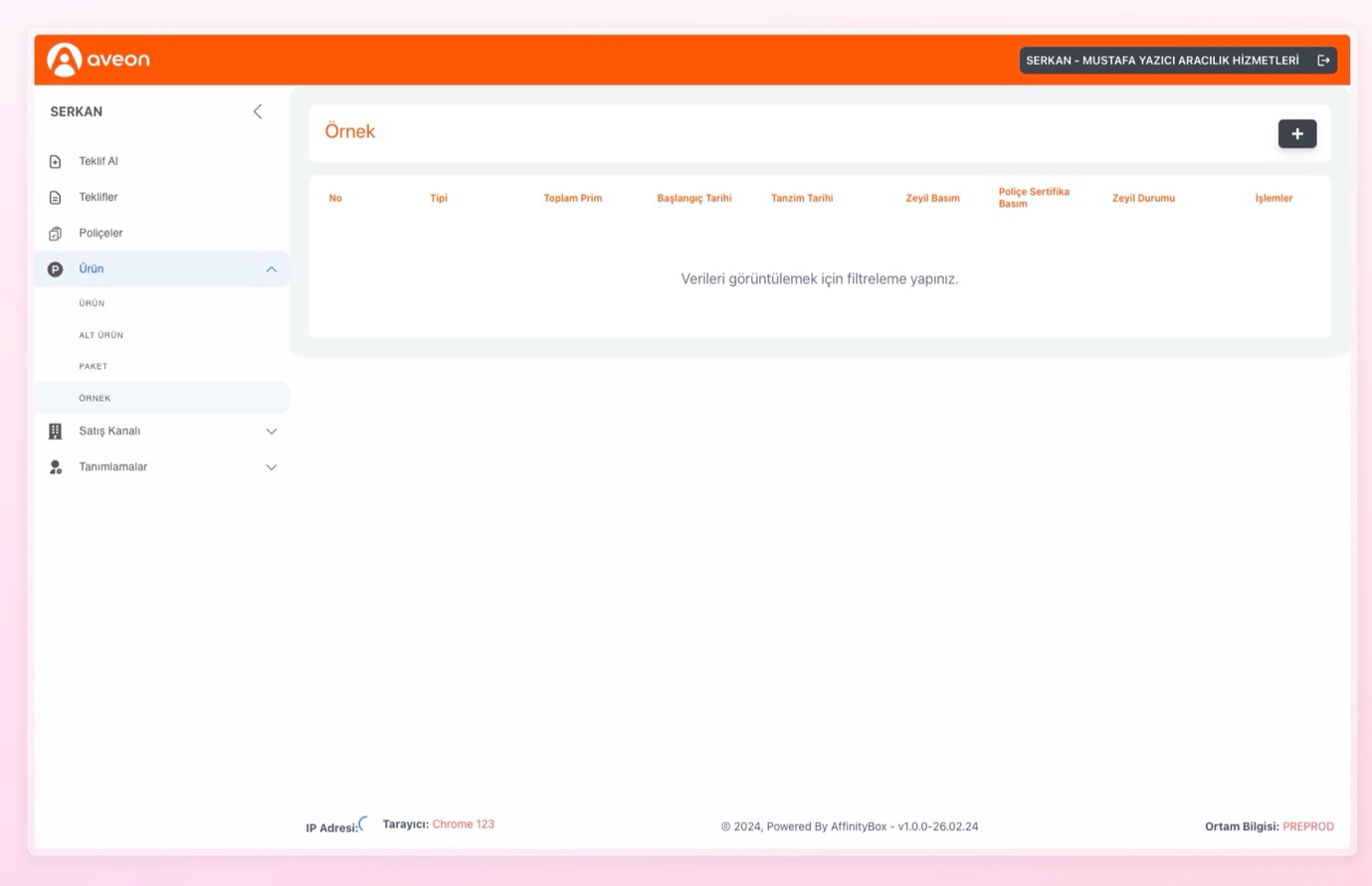
2. Yeni eklenen örnek feed dosyası feed.js içerisinde import edilir.



3. Sidebar.js içerisinde örnek feed’in id’si ile görüntülenmesinin istendiği yere eklenir.



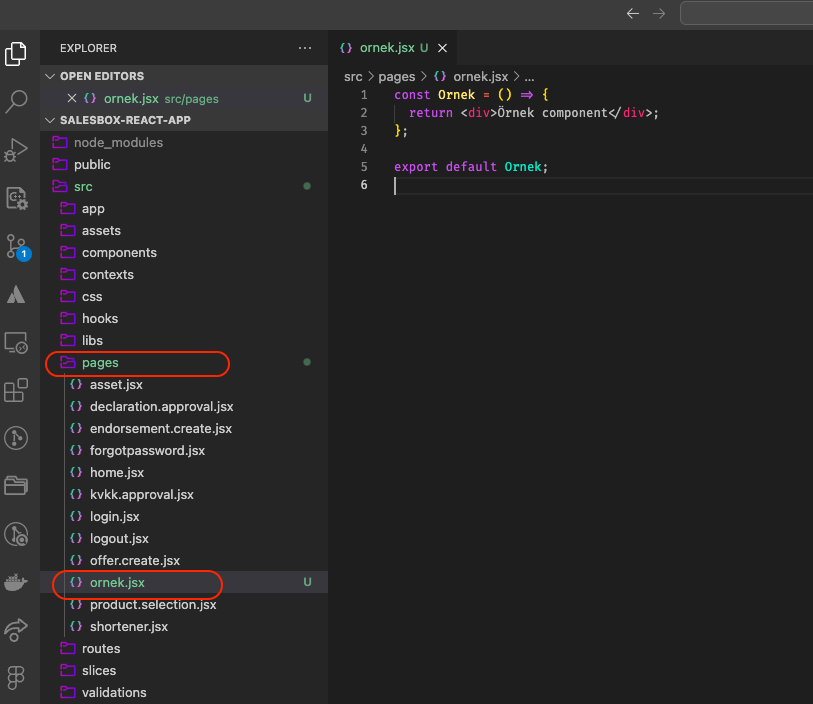
4. Sonuç



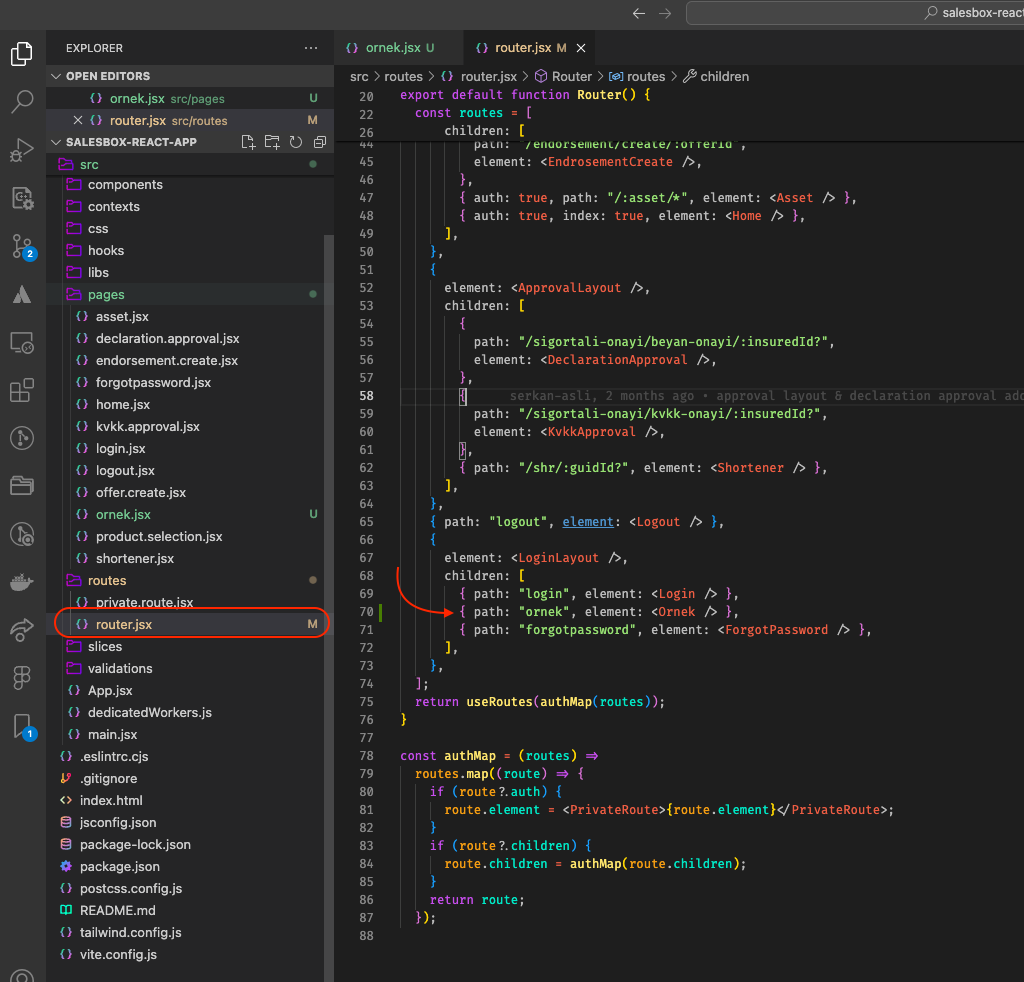
## Pages dizini altına custom ekran ekleme örneği

Bu bölümde pages dizini altına custom örnek bir ekran ekleyeceğiz.

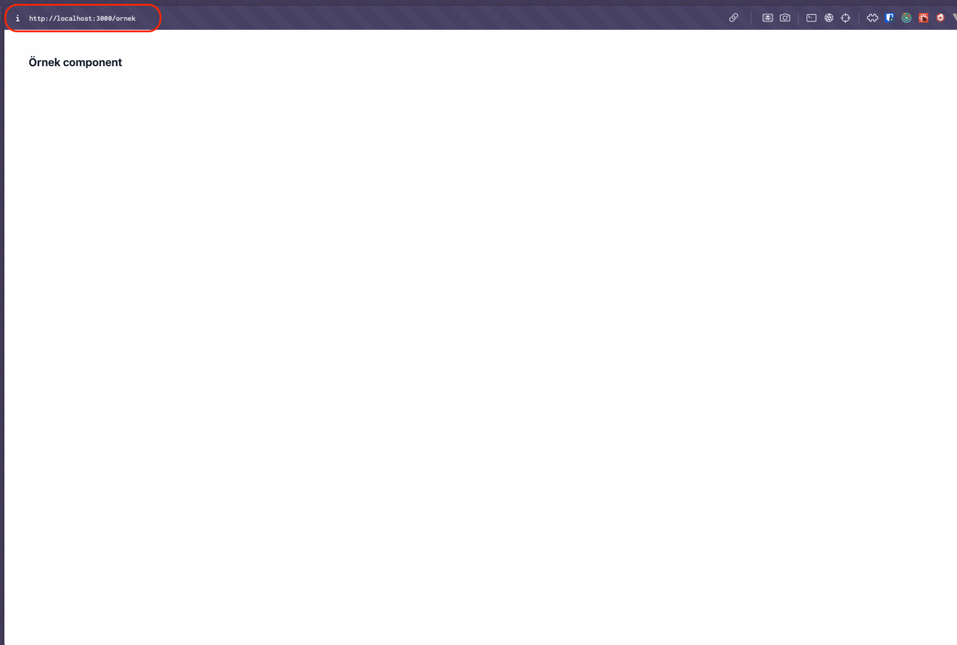
1. Pages dizini altına örnek component oluşturulur.



2.Yeni eklenen component router.jsx içerisinde istenilen layout altına eklenir.

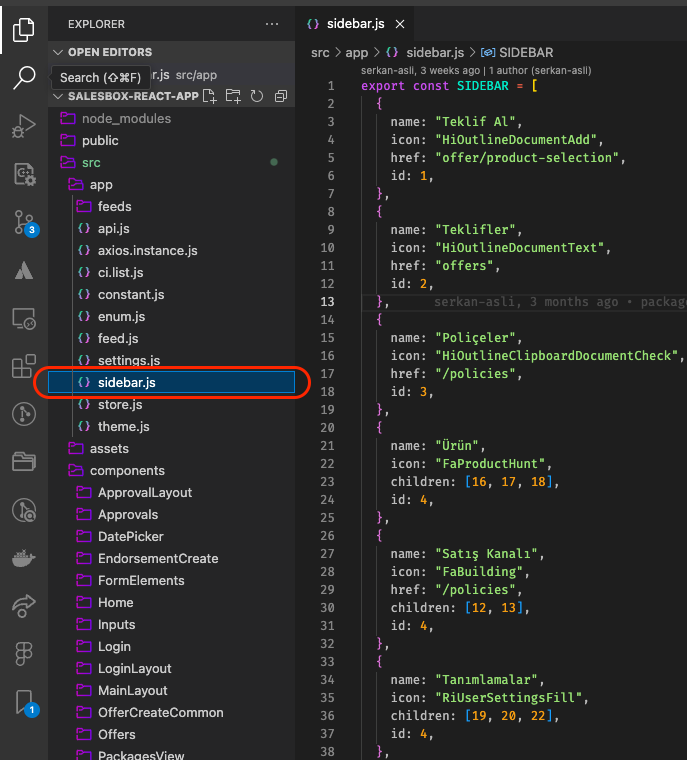
****

3.Sonuç

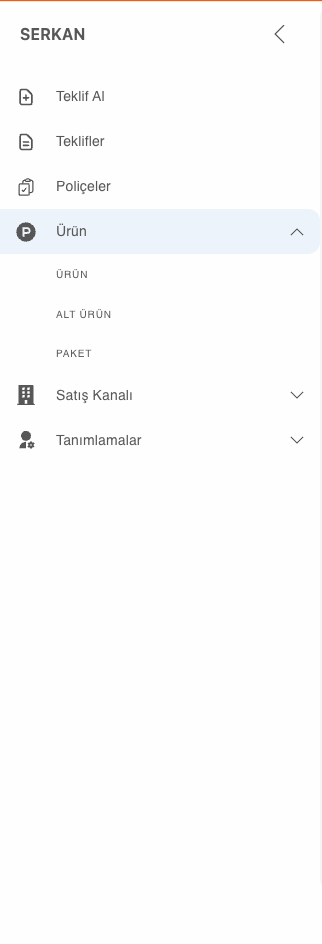


## Feed Üzerinden Eklenen Ekranın Sidebar’a Eklenmesi

Feed üzerinden yeni eklenen bir ekranı sidebar’a eklemek için app dizini altındaki sidebar.js dosyasına gidilir. Yeni ekran sidebar’da direkt olarak görünecekse href değişkenine yeni eklenen feed’in name değeri yazılır. Yeni ekran bir grup altında olacaksa (aşağıdaki örnekteki gibi) children dizisi içerisine yeni eklenen feed’in id’si yazılır.



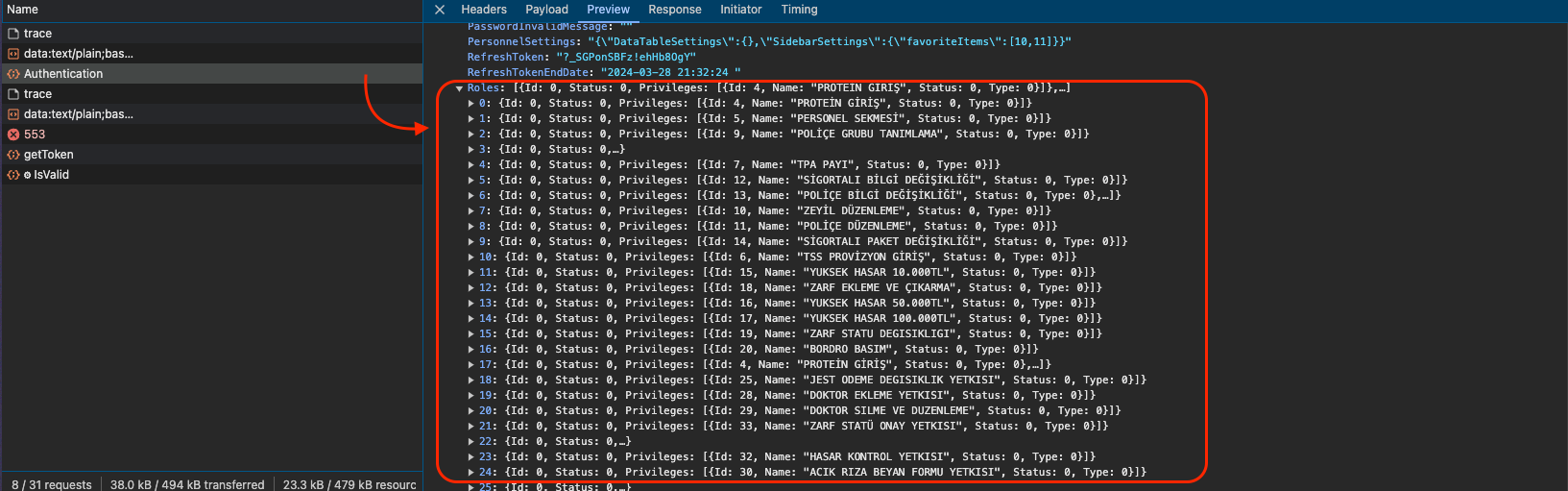
Örnek sidebar:



# UI Yetki Mekanizması

Bu bölümde authorization işlemleri ve frontend tarafında nasıl ele alındığı hakkında bilgiler verilmektedir.

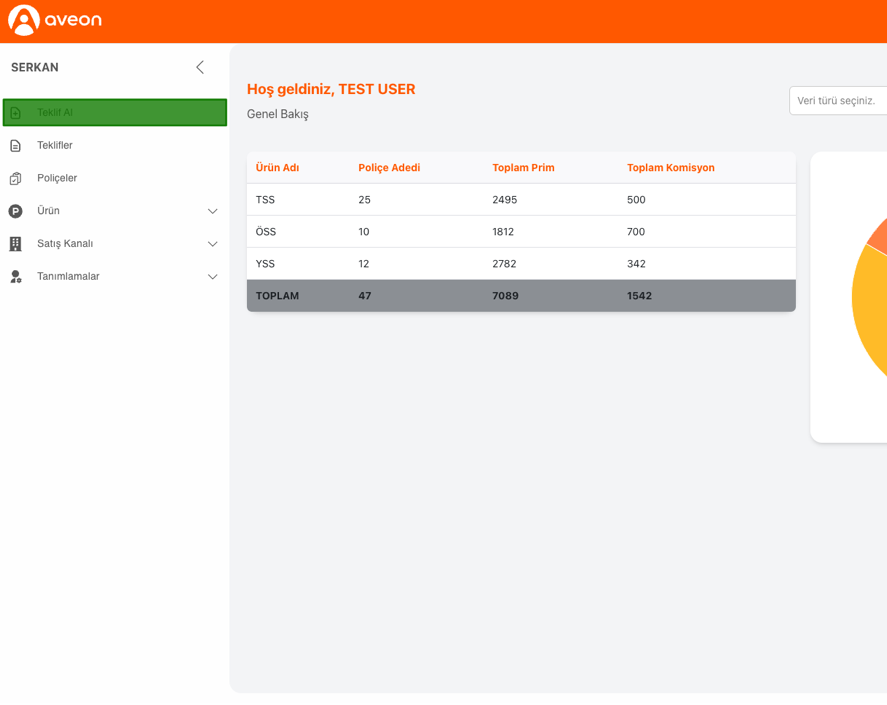
Rol ve yetki bilgileri kullanıcı login olduğu zaman backend tarafından response içerisinde veya JWT içerisinde gelmektedir. Aşağıdaki örnek görselde response içerisinde gönderilmiştir.



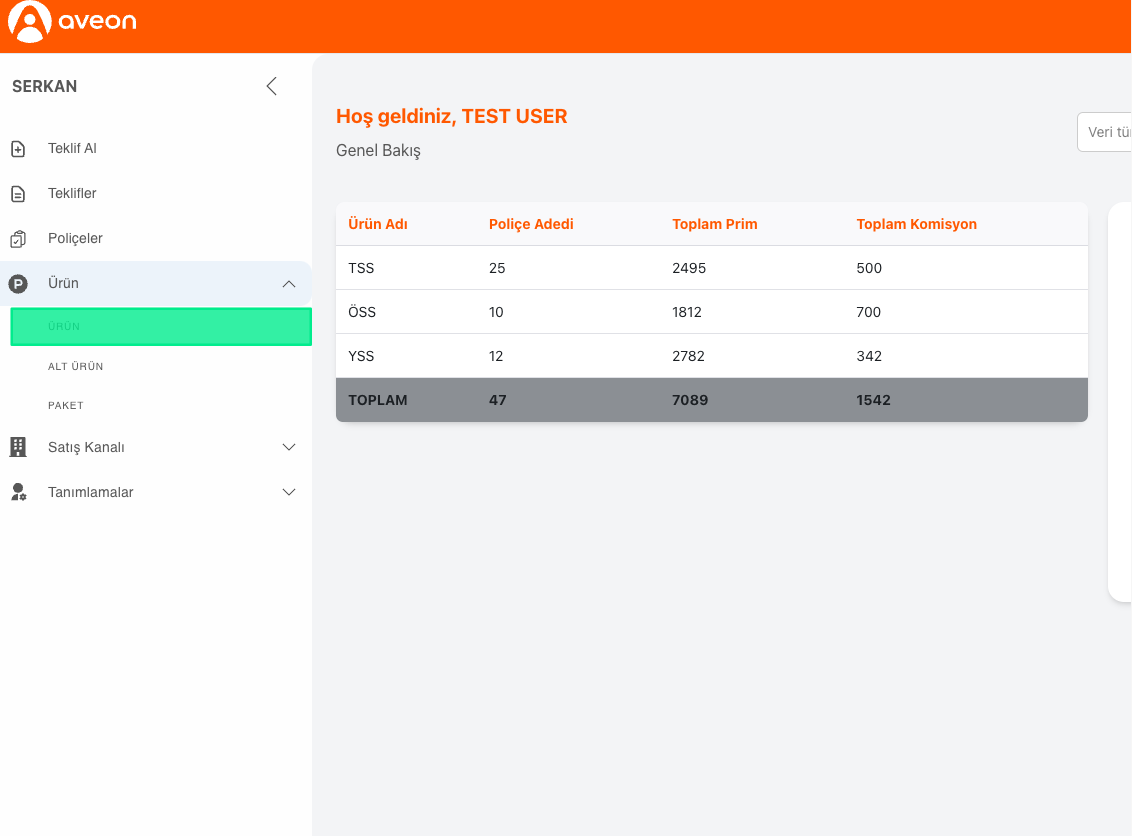
Yetki kontrolleri *checkPrivilegeInRoles* ve *checkPrivilegesInRoles* metodları yardımı ile kontrol edilir.

Yetkiler spesifik olarak bir component içerisinde kullanılabilir bunun yanı sıra feed ve sidebar.js üzerinden yapılan genel kullanım alanları da vardır. Genel kullanım alanları şöyle:

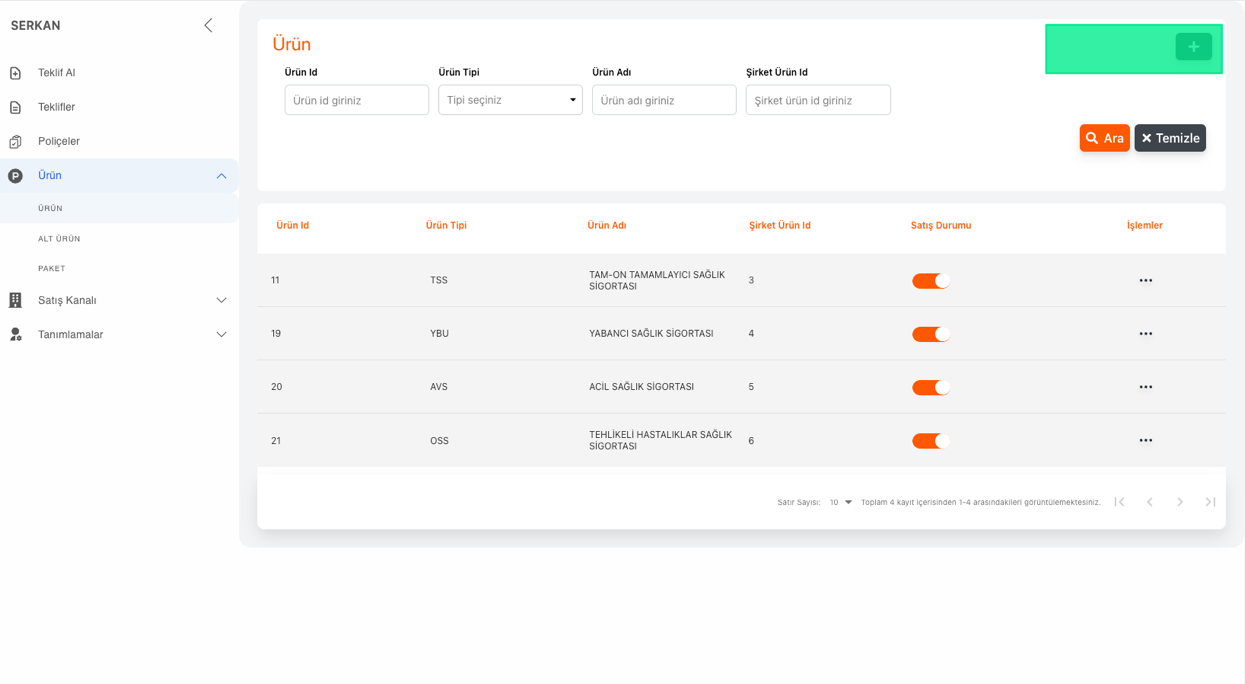
1. Sidebar’da yer alan ana menünün gizlenmesi.



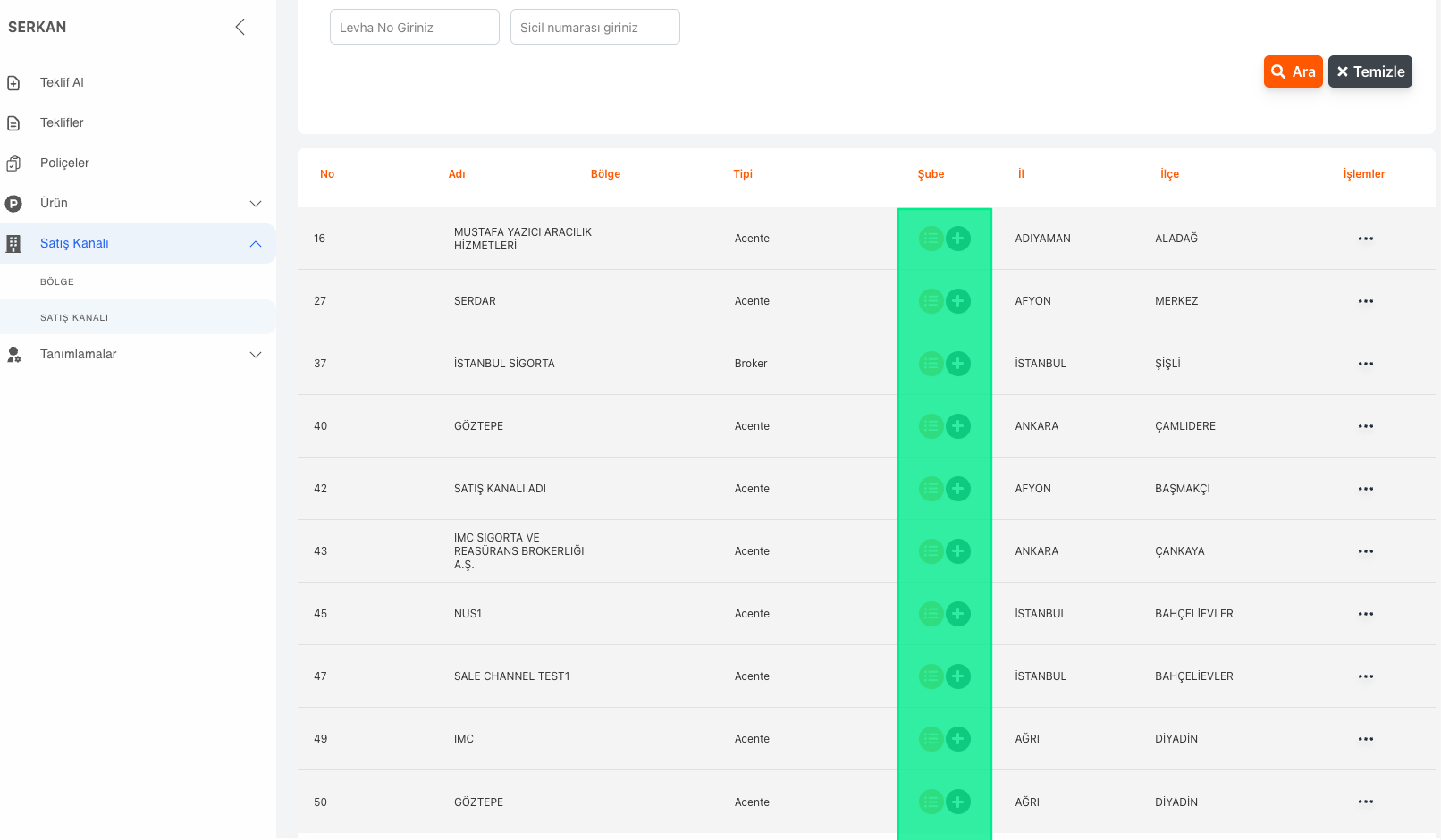
1. Sidebar’da yer alan alt menünün gizlenmesi.



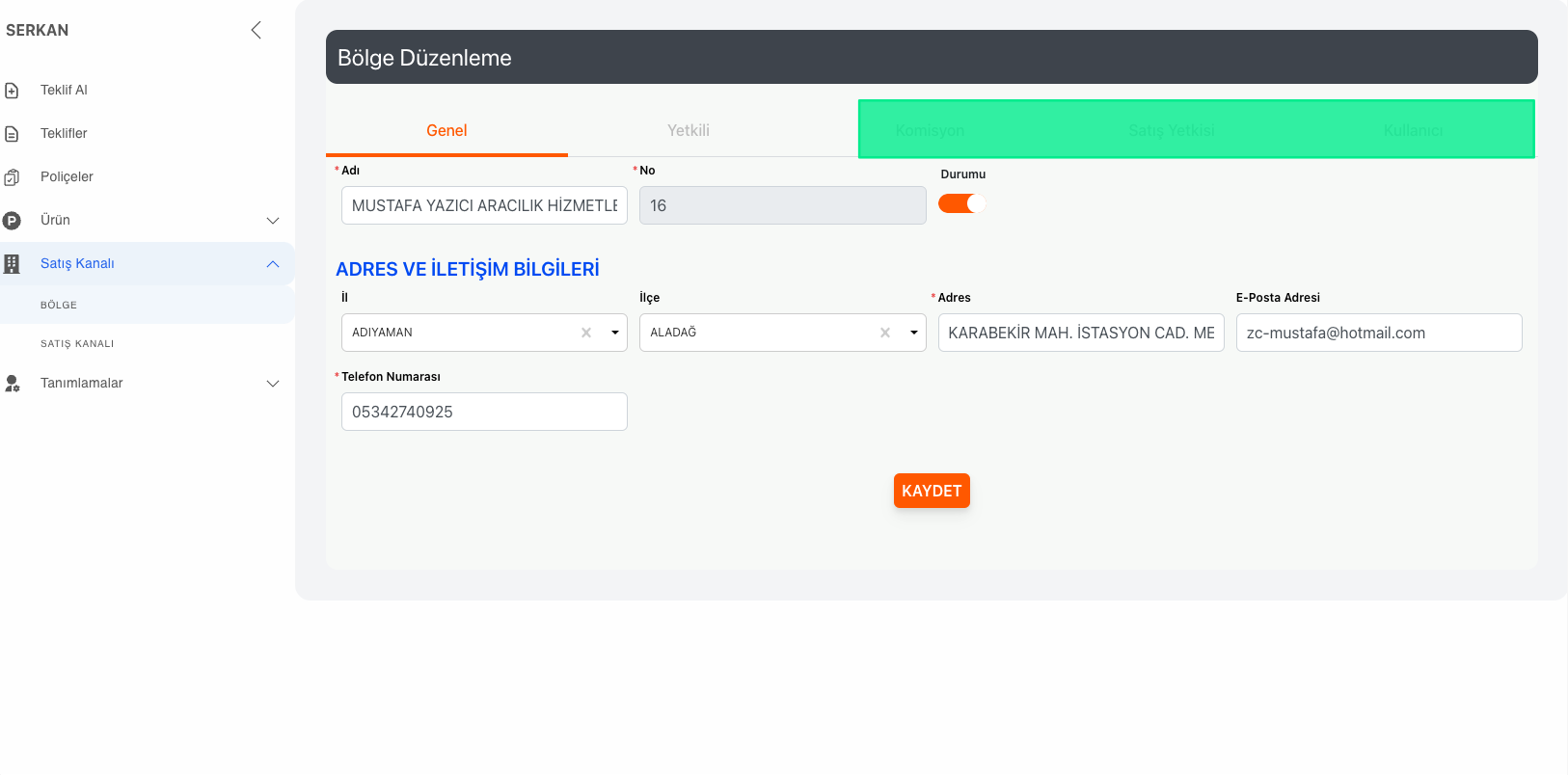
1. Entity bazlı generate edilen Create,Edit,Delete,Export,Import butonlarının yetkiye bağlanması.



1. Custom olarak eklenen butonların yetki bazlı gizlenmesi ya da disable edilmesi.



1. Entity içerisinde yer alan tab'ların yetki bazlı olarak gizlenmesi ya da disable edilmesi.



1. Entity içerisindeki bir alanın create, edit, list sayfalarında gizlenmesi ya da disable edilmesi.

