Daha önceki videolarda kullanıcı yönetimi ile işlemler yapmış tema dosyalarını oluşturmuştuk. Django’nın kullanıcı yönetimi tema dosyaları templates/registration/ klasöründe bulunduğunu söylemiştik. Bu temaları daha önceden yaptığımız için tekrardan yapmamıza gerek yoktur.

Settings.py de login login redirect url değiştirmemiz ve temalardaki extend edilen dosyayı değiştirmemiz yeterli olacaktır.Kullanıcı giriş yaptığında tekrardan login sayfasına yönlensin. Role sayfasını oluşturduğumuzda oraya yönlendireceğiz.

{% extends 'project/layout/base.html' %}

LOGIN\_REDIRECT\_URL = '/accounts/login/'

Bu sayede kullanıcı giriş ve çıkışı, şifre değiştirme, şifre sıfırlama şu anda hazır ve çalışmaktadır.

Daha önce learning kısmında kullanmış olduğumuz tema sisteminin aynısını burada da kullanalım. Hatırlarsanız layout klasöründe base, header ve footer html kullanmıştık. Content bloğunda da ilgili view içeriğini getirmiştik. Bu sefer header ve footer bilgilerini de base içerisinde kullanalım.

base.html hazırlamış olduk. base içerisinde 3 adet bloğumuz bununmaktadır.

{% block title %}

Bu blok içerisinde yer alan bilgiler sayfanın title bilgisini oluşturmaktadır

{% endblock %}

{% block top %}

Bu blok içerisinde yer alan bilgiler sayfanın üst bölümünde yer alan kısmı oluşturmaktadır.

{% endblock %}

{% block content %}

Bu blok arasında yer alan bilgiler sayfanın content kısmını oluşturmaktadır. Her view tarafından oluşturulan çıktılar bu bölümde görüntülenecektir.

{% endblock %}

Navbarımız halen learning için oluşturulan navbarı içermektedir. Bu navbarı mevcut sistemimiz için güncellememiz gerekiyor. Giriş yapılan kullanıcı tipine göre de farklı alanlar içermeli. Ziyaretçiler, bayiler ve distribitörlere farklı navbar gözükmesi gerekmektedir. İlk etapta her kullanıcı yönetimi için kullanacağımız menüleri ekleyelim. Daha sonra viewleri oluşturdukça farklılaştırırız.

{% load static %}  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>  
 {% block title %}  
 {% endblock %}  
 </title>  
 <link href="{% static 'css/bootstrap.min.css' %}" rel="stylesheet">  
 <link href="{% static 'css/jumbotron.css' %}" rel="stylesheet">  
</head>  
<body>  
  
  
<nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">  
 <a class="navbar-brand" href="#">STOCKS PROJECT</a>  
  
 <div class="collapse navbar-collapse" id="navbarsExampleDefault">  
 <ul class="navbar-nav mr-auto">

{% if not user.is\_authenticated %}  
  
 <li class="nav-item {% if request.resolver\_match.url\_name == 'login' %}active{% endif %}">  
 <a class="nav-link" href="{% url 'login' %}">Üye Girişi</a>  
 </li>  
 {% else %}

<li class="nav-item dropdown">  
 <a class="nav-link dropdown-toggle" href="#" id="dropdown01" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">Üye İşlemleri</a>  
 <div class="dropdown-menu" aria-labelledby="dropdown01">  
 <a class="dropdown-item" href="{% url 'password\_change' %}">Şifre Değiştir</a>  
 <a class="dropdown-item" href="{% url 'logout' %}">Çıkış</a>  
 </div>  
 </li>  
 {% endif %}  
 </ul>  
 <form class="form-inline my-2 my-lg-0">  
 <input class="form-control mr-sm-2" type="text" placeholder="arama kelimesi..." aria-label="Search">  
 <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Ara</button>  
 </form>  
 </div>  
</nav>  
  
<main role="main">  
 {% block top %}  
 {% endblock %}  
  
 <div class="container">  
 {% block content %}  
 {% endblock %}  
 <hr>  
  
 </div>   
</main>  
  
<footer class="container">  
 <p>**&copy;** Created for Django Course 2019</p>  
</footer>  
  
<script src="{% static 'js/jquery-slim.min.js' %}"></script>  
<script src="{% static 'js/bootstrap.min.js' %}"></script>  
</body>  
</html>

Üye girişi yapıldığında role sayfamıza yönlendireceğimizi söylemiştik. Şimdi role sayfamızı hazırlamaya başlayalım. Projeyi oluşturmaya başladığımızda distribütör ve bayilerin yetkililerinin olduğunu ve bayi ve distribütör için personel eklenebileneceğini söylemiştik.

Bayi modelini hatırlarsak eğer kullanıcı author alanında ise Bayinin yöneticisi konumunda olacaktır. Bayiler ile Kullanıcı modeli arasında da ilişki kurmuştuk eğer ilişki tablosunda varsa bayi personeli olduğunu anlarız. Distribütörler içinde aynı durum geçerlidir.

Yani kullanıcı giriş yaptığında hangi bayi ve distribütörlerde yetkisinin olduğunu ve yetki seviyesini tespit etmemiz gerekli.

* Distribütör yöneticisi olabilir
* Bayi yöneticisi olabilir
* Distribütör personeli olabilir
* Bayi personeli olabilir
* Birden fazla bayi veya distribütörde rolü olabilir

project/views/ klasörüne role.py oluşturalım ve yetkileri tespit edecek kodumuzu yazalım.

from django.views import View  
from django.shortcuts import render  
from project.models import Dealer**,** Distributor  
from django.contrib.auth.mixins import LoginRequiredMixin  
from django.db.models import Q  
  
  
class RoleView(LoginRequiredMixin**,** View):  
  
 def get(self**,** request):  
 dealers = Dealer.objects.filter(Q(author=request.user) | Q(staff=request.user))  
 distributors = Distributor.objects.filter(Q(author=request.user) | Q(staff=request.user))  
  
 context = {  
 'dealers': dealers**,** 'distributors': distributors**,** }  
  
 return render(request=request**,** template\_name='project/accounts/role.html'**,** context=context)

\_\_init\_\_.py de tanıtalım.

from .role import RoleView

Kullanıcı yönetim sayfalarımızı accounts/ altına tanımlamıştık. Ana url dosyamıza accounts/role/ için route oluşturalım ve gelen talepleri RoleView yönlendirelim.

from django.contrib import admin  
from django.urls import path, include  
import debug\_toolbar  
from django.conf.urls.static import static  
from django.conf import settings  
from project import views  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('project/', include('project.urls')),  
 path('learning/', include('learning.urls')),  
 path('accounts/', include('django.contrib.auth.urls')),  
 path('\_\_debug\_\_/', include(debug\_toolbar.urls)),  
 path('accounts/role/', views.RoleView.as\_view(), name='role-list'),  
]  
  
if settings.DEBUG:  
 urlpatterns += static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

template dosyamızda yetkileri listeleyelim.

{% extends 'project/layout/base.html' %}  
  
{% block content %}  
 <div class="card-deck">  
 {% if distributors %}  
 {% for distributor **in** distributors %}  
 <div class="card">  
 <div class="card-body">  
 <h5 class="card-title">{{ distributor.name }}</h5>  
 </div>  
 <div class="card-footer">  
 <small class="text-muted">  
 {% if distributor.author\_id == request.user.id %}  
 Distribütör Yöneticisi  
 {% else %}  
 Distribütör Personeli  
 {% endif %}  
 </small>  
 </div>  
 </div>  
 {% endfor %}  
 {% endif %}  
  
 {% if dealers %}  
 {% for dealer **in** dealers %}  
 <div class="card">  
 <div class="card-body">  
 <h5 class="card-title">{{ dealer.name }}</h5>  
 </div>  
 <div class="card-footer">  
 <small class="text-muted">  
 {% if dealer.author\_id == request.user.id %}  
 Bayi Yöneticisi  
 {% else %}  
 Bayi Personeli  
 {% endif %}  
 </small>  
 </div>  
 </div>  
 {% endfor %}  
 {% endif %}  
 </div>  
{% endblock %}

Sayfamızı kontrol ettiğimizde üye girişi yapılmadığında login sayfasına yönlendirdiğini görüyoruz. Giriş yaptığımızda ise kullanıcının rolleri gelmiş. Şimdi rollere tıklandığında ilgili şirket rolü için açılacak sayfayı tasarlayalım. Bayiler ayrı sayfaya Distribütörler ise ayrı link oluşturalım.

Bayi yönetim sayfası -> accounts/dealer/  
Distribütör yönetim sayfası -> accounts/distributor/

Kullanıcı sayfaları için view içerisinde ayrı bir klasör oluşturalım. Viewler onun içerisinde yer alsınlar. Öncelikle eski yaptığımız role sayfasını almamız gerekli. Accounts/ klasörü oluşturalım ve dosyayı içerisine taşıyalım. Boş bir init dosyası oluşturalım ve RoleView burada tanıtalım.

Eski init dosyamızda tanıttımız RoleView modül bulamadığından sorun veriyor. Bunu değiştirelim.

from .accounts import RoleView

Şimdi dealer ve distribütör için view dosyalarımızı oluşturup, init dosyamızda tanıtalım. Öncelikle distribütörler ile başlayalım.

accounts klasörüne distributor.py dosyasını ekleyelim ve sınıfımızı oluşturalım.

from django.views import View   
from django.contrib.auth.mixins import LoginRequiredMixin  
  
  
class DistributorRoleView(LoginRequiredMixin, View):  
 pass

\_\_init\_\_.py de tanıtalım.

from .role import RoleView   
from .distributor import DistributorRoleView

view klasörünün init dosyasında tanımlamalarımızı yapalım

from .accounts import RoleView, DistributorRoleView

Ana url dosyamızda routelarımızı oluşturalım ve gelen talepleri viewlere yönlendirelim. Yönlendirme yaparken hangi distribütör için tıklandığının tespit edilebilmesi için distribütörün primary keyinide göndermemiz gerekmektedir

from django.contrib import admin  
from django.urls import path**,** include  
import debug\_toolbar  
from django.conf.urls.static import static  
from django.conf import settings  
from project import views  
  
urlpatterns = [  
 path('admin/'**,** admin.site.urls)**,** path('project/'**,** include('project.urls'))**,** path('learning/'**,** include('learning.urls'))**,** path('accounts/'**,** include('django.contrib.auth.urls'))**,** path('\_\_debug\_\_/'**,** include(debug\_toolbar.urls))**,** path('accounts/role/'**,** views.RoleView.as\_view()**,** name='role-list')**,** path('accounts/distributor/<int:pk>/'**,** views.DistributorRoleView.as\_view()**,** name=distributor-home')**,**]

role template dosyasında distributor linklerinin yönlendirmesini yapalım.

{% if distributors %}  
<div class="card-deck">  
 {% for distributor in distributors %}  
 <div class="card">  
 <div class="card-body">  
 <h5 class="card-title"><a href="{% url 'distributor-home' distributor.pk %}">{{ distributor.name }}</a></h5>  
 </div>  
 <div class="card-footer">  
 <small class="text-muted">  
 {% if distributor.author\_id == request.user.id %}  
 Distribütör Yöneticisi  
 {% else %}  
 Distribütör Personeli  
 {% endif %}  
 </small>  
 </div>  
 </div>  
 {% endfor %}  
</div>  
{% endif %}

Linke tıklandığında DistributorRoleView sınıfı çağırılacaktır. Sınıf çağırıldığında distribütör id’ye kullanıcının yetkisi var mı bakalım. Veritabanından distribütörün authoru kullanıcı olan veya personel olarak kullanıcının eklendiği kayıtları getirelim. Eğer kayıt bulamazsak role sayfasına yönlendirelim.

class DistributorRoleView(LoginRequiredMixin, View):  
  
 def get(self, request, pk=None):  
  
  
 distributor = Distributor.objects.filter((Q(author=request.user) | Q(staff=request.user)) & Q(id=pk))  
  
 context = {  
 'distributor': distributor  
 }  
  
 if not distributor:  
 return redirect('role-list')

return render(request=request, template\_name='project/accounts/distributor.html', context=context)

Şimdi bu koşullarda Distributor sayfası her çağrıldığında yetki kontrolü yapacak. Bunun yerine yetki alındığında sessiona yetkisi olduğu bilgisini ekleyebiliriz.

Django’da sessionlara request.session ile ulaşırız.

from django.views import View  
from django.contrib.auth.mixins import LoginRequiredMixin  
from project.models import Distributor  
from django.db.models import Q  
from django.shortcuts import render**,** redirect  
  
  
class DistributorRoleView(LoginRequiredMixin**,** View):  
  
 def get(self**,** request**,** pk=None):  
  
 role\_page = request.session.get('role\_page'**,** False)  
  
 if role\_page == 'distributor':  
 distributor = Distributor.objects.get(pk=pk)  
 else:  
 distributor = Distributor.objects.filter((Q(author=request.user) | Q(staff=request.user)) & Q(id=pk)).first()  
  
 if distributor:  
 request.session['role\_page'] = 'distributor'  
 request.session['role\_id'] = pk  
 request.session['role\_name'] = distributor.name  
  
 else:  
 return redirect('role-list')  
  
 context = {  
 'distributor': distributor  
 }  
  
 return render(request=request**,** template\_name='project/accounts/distributor.html'**,** context=context)

Distrübütor sayfamızın temasını oluşturalım. Sayfaya girince ilgili sayfalara yönlendirecek alanlar oluşturalım linklerini sayfaları oluşturunca veririz.

{% extends 'project/layout/base.html' %}  
{% block top %}  
 <div class="jumbotron"><h4>Distribütör Yönetimi : {{ distributor.name }}</h4></div>  
{% endblock %}  
  
{% block content %}  
 <div class="jumbotron">  
 <div class="row w-100">  
 <div class="col-md-4">  
 <div class="card border-danger p-3">  
 <div class="card border-danger text-danger p-3 my-card" >  
 <span class="fa fa-user" aria-hidden="true"></span>  
 </div>  
 <div class="text-center mt-3">  
 <h4Personel Yönetimi</h4>  
 </div>  
 </div>  
 </div>  
  
 <div class="col-md-4">  
 <div class="card border-info p-3">  
 <div class="card border-info text-info p-3 my-card">  
 <span class="fa fa-book" aria-hidden="true"></span>  
 </div>  
 <div class="text-center mt-3">  
 <h4>Ödeme Yönetimi</h4>  
 </div>  
 </div>  
 </div>  
 <div class="col-md-4">  
 <div class="card border-success p-3">  
 <div class="card border-success text-success p-3 my-card">  
 <span class="fa fa-eye" aria-hidden="true"></span>  
 </div>  
 <div class="text-center mt-3">  
 <h4>Sipariş Yönetimi</h4>  
 </div>  
 </div>  
 </div>  
  
 </div>  
 <br>  
 <div class="row w-100">  
  
 <div class="col-md-4">  
 <div class="card border-info mx-sm-1 p-3">  
 <div class="card border-info shadow text-info p-3 my-card" >  
 <span class="fa fa-book" aria-hidden="true"></span>  
 </div>  
 <div class="text-center mt-3">  
 <h4>Ürün Yönetimi</h4>  
 </div>  
 </div>  
 </div>  
 <div class="col-md-4">  
 <div class="card border-warning mx-sm-1 p-3">  
 <div class="card border-warning shadow text-warning p-3 my-card">  
 <span class="fa fa-eye" aria-hidden="true"></span>  
 </div>  
 <div class="text-center mt-3">  
 <h4>Bayi Yönetimi</h4>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
  
{% endblock %}

Aynı şeyleri bayi sayfamız içinde yapalım. accounts klasörüne dealer.py dosyasını ekleyelim ve sınıfımızı oluşturalım.

from django.views import View   
from django.contrib.auth.mixins import LoginRequiredMixin  
  
  
class DealerRoleView(LoginRequiredMixin, View):  
 pass

\_\_init\_\_.py de tanıtalım.

from .role import RoleView  
from .dealer import DealerRoleView  
from .distributor import DistributorRoleView

view klasörünün init dosyasında tanımlamalarımızı yapalım

from .accounts import RoleView, DistributorRoleView, DealerRoleView

Ana url dosyamızda routelarımızı oluşturalım ve gelen talepleri viewlere yönlendirelim. Yönlendirme yaparken hangi bayi için tıklandığının tespit edilebilmesi için bayinin primary keyinide göndermemiz gerekmektedir.

from django.contrib import admin  
from django.urls import path**,** include  
import debug\_toolbar  
from django.conf.urls.static import static  
from django.conf import settings  
from project import views  
  
urlpatterns = [  
 path('admin/'**,** admin.site.urls)**,** path('project/'**,** include('project.urls'))**,** path('learning/'**,** include('learning.urls'))**,** path('accounts/'**,** include('django.contrib.auth.urls'))**,** path('\_\_debug\_\_/'**,** include(debug\_toolbar.urls))**,** path('accounts/role/'**,** views.RoleView.as\_view()**,** name='role-list')**,** path('accounts/dealer/<int:pk>/'**,** views.DistributorRoleView.as\_view()**,** name='dealer-home')**,** path('accounts/distributor/<int:pk>/'**,** views.DistributorRoleView.as\_view()**,** name='distributor-home')**,**]

Bayi view dosyamızı oluşturalım.

from django.views import View  
from django.shortcuts import render**,** redirect  
from project.models import Dealer  
from django.contrib.auth.mixins import LoginRequiredMixin  
from django.db.models import Q  
  
  
class DealerRoleView(LoginRequiredMixin**,** View):  
  
 def get(self**,** request**,** pk=None):  
  
 role\_page = request.session.get('role\_page'**,** False)  
  
 if role\_page == 'dealer':  
 dealer = Dealer.objects.get(pk=pk)  
 else:  
 dealer = Dealer.objects.filter((Q(author=request.user) | Q(staff=request.user)) & Q(id=pk)).first()  
  
 if dealer:  
 request.session['role\_page'] = 'dealer'  
 request.session['role\_id'] = pk  
 request.session['role\_name'] = dealer.name  
  
 else:  
 return redirect('role-list')  
  
 context = {  
 'dealer': dealer  
 }  
  
 return render(request=request**,** template\_name='project/accounts/dealer.html'**,** context=context)

Bayi template dosyamızı oluşturalım

{% extends 'project/layout/base.html' %}  
  
{% block top %}  
 <div class="jumbotron"><h4>Bayi Yönetimi : {{ dealer.name }}</h4></div>  
{% endblock %}  
  
{% block content %}  
 <div class="jumbotron">  
 <div class="row w-100">  
 <div class="col-md-4">  
 <div class="card border-danger p-3">  
 <div class="card border-danger text-danger p-3 my-card" >  
 <span class="fa fa-user" aria-hidden="true"></span>  
 </div>  
 <div class="text-center mt-3">  
 <h4>Personel Yönetimi</h4>  
 </div>  
 </div>  
 </div>  
  
 <div class="col-md-4">  
 <div class="card border-info p-3">  
 <div class="card border-info text-info p-3 my-card">  
 <span class="fa fa-book" aria-hidden="true"></span>  
 </div>  
 <div class="text-center mt-3">  
 <h4>Ödeme Yönetimi</h4>  
 </div>  
 </div>  
 </div>  
 <div class="col-md-4">  
 <div class="card border-success p-3">  
 <div class="card border-success text-success p-3 my-card">  
 <span class="fa fa-eye" aria-hidden="true"></span>  
 </div>  
 <div class="text-center mt-3">  
 <h4>Sipariş Yönetimi</h4>  
 </div>  
 </div>  
 </div>  
  
 </div>  
 </div>  
  
{% endblock %}

Kullanıcılar login ve logout olduklarında hala sessionda role sayfasının bilglileri tutulmaktadır. Bunların karışmaması için login ve logout yapıldığında sessionda tuttuğumuz role\_page ve role\_id bilgilerini sıfırlayalım. Bunu iki yöntemle yaparız. Biz şu anda login ve logout için herhangi bir view yazmadık. Ya bunları manuel olarak yazmalıyız yada sinyallerden faydalanmalıyız.

Django kullanıcı yönetimi login, logout ve login başarısız olduğunda bize built in olarak üç farklı sinyal sunar. Biz sinyalleri uygulama içerisinde değil projede kullanacağımız için stocks/ kasörüne apps.py oluşturalım. Sinyalleri çağırmak için ready metodunda sinyalleri import etmemiz gerekmektedir.

*stocks/apps.py*

from django.apps import AppConfig  
  
  
class StocksConfig(AppConfig):  
 name = 'stocks' # proje adı ile aynı olmalı

Sinyallerimizi oluşturalım.

*stocks/signals.py*

from django.contrib.auth.signals import user\_logged\_in**,** user\_logged\_out, user\_login\_failed

from django.dispatch import receiver  
  
  
@receiver(user\_logged\_in)  
def user\_logged\_in(sender**,** user**,** request**,** \*\*kwargs):  
 print('giriş yapıldı')

@receiver(user\_logged\_out)  
def user\_logged\_out(sender**,** user**,** request**,** \*\*kwargs):  
 print('çıkış yapıldı')

@receiver(user\_login\_failed)  
def user\_not\_logged\_in(sender**,** credentials**,** request**,** \*\*kwargs):  
 print('giriş başarısız')

Sinyallerimizi import edeceğimiz ready metodunu yazalım.

*stocks/apps.py*

from django.apps import AppConfig  
  
  
class StocksConfig(AppConfig):  
 name = 'stocks' # proje adı ile aynı olmalı  
  
 def ready(self):  
 import stocks.signals

installed.apps ta Config dosyamızı çağıralım.

*stocks/settings.py*

INSTALLED\_APPS = [  
 ...'stocks.apps.StocksConfig'**,** ...]

Artik login, logout işlemlerinde sinyal olarak yazdığımız metodlarımz çalışacaktır. Şimdi bu metodlarda sessionları false olarak değiştirelim.

*stocks/signals.py*

from django.contrib.auth.signals import user\_logged\_in**,** user\_logged\_out  
from django.dispatch import receiver  
  
  
@receiver(user\_logged\_in)  
def user\_logged\_in(sender**,** user**,** request**,** \*\*kwargs):  
 request.session['role\_page'] = False  
 request.session['role\_id'] = False  
  
  
@receiver(user\_logged\_out)  
def user\_logged\_out(sender**,** user**,** request**,** \*\*kwargs):  
 request.session['role\_page'] = False  
 request.session['role\_id'] = False

Role sayfasından yeni role seçildiği için çağırıldığında da mevcut rolleri yani sessionları sıfırlayalım.

from django.views import View  
from django.shortcuts import render  
from project.models import Dealer**,** Distributor  
from django.contrib.auth.mixins import LoginRequiredMixin  
from django.db.models import Q  
  
  
class RoleView(LoginRequiredMixin**,** View):  
  
 def get(self**,** request):  
 request.session['role\_page'] = False  
 request.session['role\_id'] = False  
   
 dealers = Dealer.objects.filter(Q(author=request.user) | Q(staff=request.user))  
 distributors = Distributor.objects.filter(Q(author=request.user) | Q(staff=request.user))  
  
 context = {  
 'dealers': dealers**,** 'distributors': distributors**,** }  
  
 return render(request=request**,** template\_name='project/accounts/role.html'**,** context=context)

Kullanıcı giriş yaptığında role sayfasına yönlendirelim. Bu settings.py den yapıyoruz. Giriş yaparak deneyelim. Evet yönlendirmemiz tamamlandı.

LOGIN\_REDIRECT\_URL = '/accounts/role/'

Artık kullanıcı giriş yaptığında otomatik olarak role sayfasına yönlenmektedir. Kullanıcı buradan hangi rol ile devam etmek istiyorsa seçip ilgili sayfaya yönlenebilir.

Personel yönetimine tıklandığında distribütörün personel listesi açılsın. Personel listeleme sayfası hem bayi hem de distribütör için var. Biz tek view ile kullanıcı ekletelim view kısmında eğer seçilen Role bayi ise bayi personelleri, seçilen role distribütör ise distribütör personelleri.

Bunun için view/accounts içine list\_staff dosyamızı oluşturalım.

from project.models import Distributor**,** Dealer  
from django.contrib.auth.decorators import login\_required

from django.shortcuts import render

@login\_required  
def list\_staff(request):  
 role\_id = request.session.get('role\_id'**,** False)  
 role\_page = request.session.get('role\_page'**,** False)  
  
 if role\_page == 'distributor':  
 queryset = Distributor.objects.prefetch\_related('staff').get(pk=role\_id)  
 else:  
 queryset = Dealer.objects.prefetch\_related('staff').get(pk=role\_id)  
  
 context = {  
 'staff\_list': queryset.staff.all()  
 }  
  
 return render(request=request**,** template\_name='project/accounts/staff\_list.html'**,** context=context)

init dosyasında tanıtalım.

from .role import RoleView  
from .dealer import DealerRoleView  
from .distributor import DistributorRoleView  
from .user\_management import list\_staff

ana init dosyasında tanıtalım

from .accounts import RoleView**,** DealerRoleView**,** DistributorRoleView**,** list\_staff

url dosyamızı güncelleyelim

path('accounts/list\_staff/'**,** views.list\_staff**,** name='list-staff')**,**

Tema dosyamızı oluşturalım.

{% extends 'project/layout/base.html' %}  
  
{% block top %}  
 <div class="jumbotron"><h4>Personel Listele : {{ request.session.role\_name }}</h4></div>  
{% endblock %}  
{% block content %}  
  
 <div style="margin-bottom: **15**px**;**text-align: right"><a href="#" class="btn btn-primary">Personel Ekle</a></div>  
  
 <table class="table table-striped">  
 <thead>  
 <tr>  
 <th scope="col">#</th>  
 <th scope="col">Kullanıcı Adı</th>  
 <th scope="col">Adı Soyadı</th>  
 <th scope="col">Email</th>  
 <th scope="col"></th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for staff in staff\_list %}  
 <tr>  
 <th scope="row">{{ forloop.counter }}</th>  
 <td>{{ staff.username }}</td>  
 <td>{{ staff.first\_name }} {{ staff.last\_name }}</td>  
 <td>{{ staff.email }}</td>  
 <td style="text-align: right">  
 <a href="#" class="btn btn-danger btn-sm">Sil</a>  
 <a href="#" class="btn btn-warning btn-sm">Düzenle</a>  
 <a href="#" class="btn btn-primary btn-sm">Yetkiler</a>  
 </td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
  
  
{% endblock %}

Evet kontrol ettiğimizde personel listemiz geliyor. Şimdi kullanıcı eklemeyi aktif hale getirelim.

Djangonun kullanıcı oluşturmak için built-in olarak barındırdığı UserCreationForm nesnesi vardır. Bu formu kullanarak kullanıcı oluşturabiliriz.

Bunun için view/accounts içine create\_staff dosyamızı oluşturalım ve UserCreationForm import edelim. View oluşturalım

from django.contrib.auth.forms import UserCreationForm  
from django.shortcuts import redirect, render  
from django.db import transaction  
  
  
@transaction.atomic  
def create\_staff(request):  
  
 role\_id = request.session.get('role\_id', False)  
 role\_page = request.session.get('role\_page', False)  
  
 if role\_id is False or role\_page is False:  
 return redirect(staff-list')  
  
 if request.method == 'POST':  
 form = UserCreationForm(request.POST)  
 if form.is\_valid():  
 user = form.save()

if role\_page == 'distributor':  
 user.distributors.add(role\_id)  
 else:  
 user.dealers.add(role\_id)  
  
 return redirect(staff-list')  
 else:  
 form = UserCreationForm()  
  
 return render(request, 'project/accounts/create\_staff.html', {'form': form})

init dosyalarında tanıtalım

from .role import RoleView  
from .dealer import DealerRoleView  
from .distributor import DistributorRoleView  
from .user\_management import staff\_list, create\_staff

from .accounts import RoleView, DealerRoleView, DistributorRoleView, staff\_list, create\_staff

url dosyamızı güncelleyelim

path('accounts/create\_staff/'**,** views.create\_staff**,** name='create-staff')**,**

template dosyamızı oluşturalım.

{% extends 'project/layout/base.html' %}  
  
{% block content %}  
  
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Kullanıcı Ekle</button>  
 </form>  
  
  
{% endblock %}

Template dosyamızı özelleştirelim.

{% extends 'project/layout/base.html' %}  
  
{% block content %}  
  
 <form method="post">  
 {% csrf\_token %}  
 {% for field **in** form %}  
 <p>  
 {{ field.label\_tag }}<br>  
 {{ field }}  
  
 {% for error **in** field.errors %}  
 <p style="color: red">{{ error }}</p>  
 {% endfor %}  
 </p>  
 {% endfor %}  
 <button type="submit">Kullanıcı Ekle</button>  
 </form>  
  
{% endblock %}

Kullanıcı listeleme template sayfamıza ekleme llinkini ekleyelim

<div style="margin-bottom: **15**px**;**text-align: right"><a href="{% url 'create-staff' %}" class="btn btn-primary">Personel Ekle</a></div>

Ekleye tıkladığımızda kullanıcı ekleme formu geliyor. Kullanıcı oluştururken ad, soyad ve email alanlarını da ekleyelim. Bunun için yeni bir form oluşturup UserCreationForm’dan türetebiliriz. Projects klasörü içine forms.py dosyası oluşturalım ve bunun içerisinde yeni formumuzu oluşturalım.

from django.contrib.auth.forms import UserCreationForm  
from django import forms   
  
  
class StaffCreationForm(UserCreationForm):  
 first\_name = forms.CharField(label='Ad')  
 last\_name = forms.CharField(label='Soyad')  
 email = forms.EmailField(label='Email')

init dosyasında tanıtalım

from .user\_management import StaffCreationForm**,**

view dosyamızı güncelleyelim.

from django.contrib.auth.decorators import login\_required   
from django.shortcuts import render**,** redirect  
from project.forms import StaffCreationForm  
from django.db import transaction  
from stocks.helpers import check\_superuser  
  
  
@login\_required  
@transaction.atomic  
def create\_staff(request):  
  
 role\_id = request.session.get('role\_id'**,** False)  
 role\_page = request.session.get('role\_page'**,** False)  
  
 if role\_id is False or role\_page is False:  
 return redirect('role-list')  
  
 if request.method == 'POST':  
  
 form = StaffCreationForm(request.POST)  
  
 if form.is\_valid():  
 user = form.save(commit=False)  
 user.first\_name = form.cleaned\_data['first\_name']  
 user.last\_name = form.cleaned\_data['last\_name']  
 user.email = form.cleaned\_data['email']  
 user.save()  
  
 if role\_page == 'distributor':  
 user.distributors.add(role\_id)  
 else:  
 user.dealers.add(role\_id)  
  
 return redirect('list-staff')  
 else:  
 form = StaffCreationForm()  
  
 return render(request**,** 'project/accounts/create\_staff.html'**,** {'form': form})

Tema dosyamızı güncelleyelim.

{% extends 'project/layout/base.html' %}  
  
{% block top %}  
 <div class="jumbotron"><h4>Personel Ekle : {{ request.session.role\_name }}</h4></div>  
{% endblock %}  
  
{% block content %}  
 <form method="post">  
 {% csrf\_token %}  
 {% for field in form %}  
 <p>  
 {{ field.label\_tag }}<br>  
 {{ field }}  
  
 {% for error in field.errors %}  
 <p style="color: red">{{ error }}</p>  
 {% endfor %}  
 </p>  
 {% endfor %}  
 <button type="submit">Kullanıcı Ekle</button>  
 </form>  
  
{% endblock %}

Kullanıcı silme için yeni bir view oluşturalım. Bunun için view/accounts içine delete\_staff dosyamızı oluşturalım

from project.models import Distributor**,** Dealer  
from django.contrib.auth.decorators import login\_required   
from django.shortcuts import redirect  
from stocks.helpers import check\_superuser  
  
  
@login\_required  
def delete\_staff(request**,** pk=None):  
 role\_id = request.session.get('role\_id'**,** False)  
 role\_page = request.session.get('role\_page'**,** False)  
  
 if role\_page == 'distributor':  
 item = Distributor.objects.get(pk=role\_id)  
 else:  
 item = Dealer.objects.get(pk=role\_id)  
  
 item.staff.remove(pk)  
  
 return redirect('list-staff')

Kullanıcı listeleme template sayfamıza silme llinkini ekleyelim

<a href="{% url 'delete-staff' staff.pk %}" class="btn btn-danger btn-sm">Sil</a>

url dosyamızı güncelleyelim

path('accounts/delete\_staff/<int:pk>/'**,** views.delete\_staff**,** name='delete-staff')**,**

Kullanıcı düzenleme için form oluşturalım

from django.contrib.auth.models import User

class StaffUpdateForm(forms.ModelForm):  
  
 edit\_password = forms.CharField(label='Şifre'**,** widget=forms.PasswordInput()**,** required=False)  
  
 class Meta:  
 model = User  
 fields = ('first\_name'**,** 'last\_name'**,** 'edit\_password'**,** 'email')

Kullanıcı düzenleme için yeni bir view oluşturalım. Bunun için view/accounts içine edit\_staff dosyamızı oluşturalım

from django.contrib.auth.decorators import login\_required   
from django.shortcuts import render**,** redirect  
from project.forms import StaffUpdateForm  
from django.contrib.auth.models import User  
from django.contrib.auth.password\_validation import validate\_password  
from django.core.exceptions import ValidationError  
from stocks.helpers import check\_superuser  
  
  
@login\_required  
def edit\_staff(request**,** pk=None):  
 instance = User.objects.get(pk=pk)  
  
 if request.POST:  
  
 form = StaffUpdateForm(request.POST**,** instance=instance)  
  
 if form.is\_valid():  
 password = form.cleaned\_data['edit\_password']  
  
 if password:  
 try:  
 validate\_password(password**,** instance)  
  
 user = form.save(commit=False)  
 user.set\_password(password)  
 user.save()  
 return redirect('list-staff')  
  
 except ValidationError as error:  
 form.add\_error('edit\_password'**,** error)  
 else:  
 user = form.save()  
 return redirect('list-staff')  
 else:  
 form = StaffUpdateForm(instance=request.user)  
  
 return render(request**,** 'project/accounts/edit\_staff.html'**,** {'form': form})

Template dosyamızı oluşturalım

{% extends 'project/layout/base.html' %}  
  
{% block top %}  
 <div class="jumbotron"><h4>Personel Düzenle : {{ request.session.role\_name }}</h4></div>  
{% endblock %}  
  
{% block content %}  
 <form method="post">  
 {% csrf\_token %}  
 {% for field in form %}  
 <p>  
 {{ field.label\_tag }}<br>  
 {{ field }}  
  
 {% for error in field.errors %}  
 <p style="color: red">{{ error }}</p>  
 {% endfor %}  
 </p>  
 {% endfor %}  
 <button type="submit">Kullanıcı Güncelle</button>  
 </form>  
  
{% endblock %}

url dosyamızı güncelleyelim

path('accounts/edit\_staff/<int:pk>/'**,** views.edit\_staff**,** name='edit-staff')**,**

Kullanıcı listeleme template sayfamıza düzenleme llinkini ekleyelim

<a href="{% url 'edit-staff' staff.pk %}" class="btn btn-warning btn-sm">Düzenle</a>

Kullanıcı yetki düzenleme için yeni bir view oluşturalım. Bunun için view/accounts içine edit\_staff\_permissions dosyamızı oluşturalım

from django.contrib.auth.decorators import login\_required   
from django.shortcuts import render**,** redirect  
from project.forms import StaffPermissionForm  
from django.contrib.auth.models import User**,** Permission  
from stocks.helpers import check\_superuser  
  
  
@login\_required  
def edit\_permissions(request**,** pk=None):  
  
 instance = User.objects.get(pk=pk)  
  
 if request.method == 'POST':  
 form = StaffPermissionForm(request.POST)  
  
 if form.is\_valid():  
 # Product Permission  
 product\_permission = form.cleaned\_data['product\_permission']  
 product\_db\_permission = Permission.objects.get(codename='manage\_product')  
 if product\_permission:  
 instance.user\_permissions.add(product\_db\_permission)  
 else:  
 instance.user\_permissions.remove(product\_db\_permission)  
  
 # Dealer Permission  
 dealer\_permission = form.cleaned\_data['dealer\_permission']  
 dealer\_db\_permission = Permission.objects.get(codename='manage\_dealer')  
 if dealer\_permission:  
 instance.user\_permissions.add(dealer\_db\_permission)  
 else:  
 instance.user\_permissions.remove(dealer\_db\_permission)  
  
 # Order Permission  
 order\_permission = form.cleaned\_data['order\_permission']  
 order\_db\_permission = Permission.objects.get(codename='manage\_order')  
 if order\_permission:  
 instance.user\_permissions.add(order\_db\_permission)  
 else:  
 instance.user\_permissions.remove(order\_db\_permission)  
  
 # Payment Permission  
 payment\_permission = form.cleaned\_data['payment\_permission']  
 payment\_db\_permission = Permission.objects.get(codename='manage\_payment')  
 if payment\_permission:  
 instance.user\_permissions.add(payment\_db\_permission)  
 else:  
 instance.user\_permissions.remove(payment\_db\_permission)  
  
 return redirect('list-staff')  
 else:  
  
 product\_permission = instance.has\_perm('project.manage\_product')  
 dealer\_permission = instance.has\_perm('project.manage\_dealer')  
 order\_permission = instance.has\_perm('project.manage\_order')  
 payment\_permission = instance.has\_perm('project.manage\_payment')  
  
 form = StaffPermissionForm(initial={'product\_permission': product\_permission**,** 'dealer\_permission': dealer\_permission**,** 'order\_permission': order\_permission**,** 'payment\_permission': payment\_permission})  
  
 return render(request**,** 'project/accounts/edit\_permissions.html'**,** {'form': form})

Modellere için manage permissionları oluşturalım.

permissions = [('manage\_product'**,** 'Ürün yönetim izni')]

permissions = [('manage\_payment'**,** 'Ödeme yönetim izni')]

permissions = [('manage\_order'**,** 'Sipariş yönetim izni')]

permissions = [('manage\_dealer'**,** 'Bayi yönetim izni')]

Kullanıcı listeleme template sayfamıza yetki düzenleme llinkini ekleyelim

<a href="{% url 'edit-permissions' staff.pk %}" class="btn btn-primary btn-sm">Yetkiler</a>

url dosyamızı güncelleyelim

path('accounts/edit\_permissions/<int:pk>/'**,** views.edit\_permissions**,** name='edit-permissions')**,**

Yetkisi olmayan linkleri gizleyelim.

{% if request.user.is\_superuser %}

Personel yönetimi için

{% endif %}

{% if perms.project.manage\_payment or request.user.is\_superuser %}

Ödeme yönetimi için

{% endif %}

{% if perms.project.manage\_order or request.user.is\_superuser %}

Sipariş yönetimi için

{% endif %}

{% if perms.project.manage\_product or request.user.is\_superuser %}

Ürün yönetimi için

{% endif %}

{% if perms.project.manage\_dealer or request.user.is\_superuser %}

Bayi yönetimi için

{% endif %}

Daha önce oluşturduğumuz kullanıcı yönetimi alanları için super user mı diye kontrol edelim. Bunu user\_passes\_test dekoratörü ile yapabiliriz. Eğer fonksiyon true dönerse view çalıştırılır aksi taktirde login sayfasına yönlendirilir.

Stocks içerisine helpers diye bir klasör oluşturalım. Bu klasör içerisinde yardımcı fonksiyonlarımızı tutalım. Kullanıcı işlemlerine ait metodlar için user\_management dosyası oluşturalım ve içerisine metodumuzu yazalım. Eğer user superuser ise true dönecektir.

def check\_superuser(user):  
 return user.is\_superuser

init dosyasında tanıtalım

from .user\_management import check\_superuser

kullanıcı sayfası view metdouna user\_passes\_test dekoratörü ekleyelim.

from stocks.helpers import check\_superuser  
from django.contrib.auth.decorators import login\_required**,** user\_passes\_test

@login\_required  
@user\_passes\_test(check\_superuser)  
def list\_staff(request):  
 …..

Bu dekoratörümüzü tüm kullanıcı sayfaları için ekleyelim. Artık bu viewler sadece superuser tarafından çalıştırılabilecektir.

project/views/accounts/init son hali

from .role import RoleView  
from .dealer import DealerRoleView  
from .distributor import DistributorRoleView  
from .list\_staff import list\_staff  
from .create\_staff import create\_staff  
from .edit\_staff import edit\_staff  
from .edit\_staff\_permissions import edit\_permissions  
from .delete\_staff import delete\_staff

project /views/init son hali

from .accounts import RoleView**,** DealerRoleView**,** DistributorRoleView**,** list\_staff**,** create\_staff**,** delete\_staff**,** edit\_staff**,** edit\_permissions

------------ HERKES KENDİ STAFFINI GÜNCELLEYEBİLMELİ ------------