**Django Notification Package**

**Django-notification paketi**: Django projelerinde kullanıcı bildirimleri oluşturmak ve yönetmek için kullanılan bir araçtır. Bu paket, kullanıcıların belirli eylemler veya olaylar hakkında bilgilendirilmesini sağlar.

**Paket kurulumu:**

| pip install django-notifications-hq |
| --- |

Ayarlar:setting.py dosyanıza ‘notifications’ uygulamasını ekleyiniz.

| INSTALLED\_APPS = [  ... 'notifications',  ...  ] |
| --- |

**django-activity-stream :**  Kullanıcı aktivitelerini izlemek ve bir zaman akışı formatında göstermek için kullanılır. Kullanıcılar veya sistem tarafından gerçekleştirilen eylemleri kaydeder ve görüntüler.

**Activity Stream Settings:**

| ACTSTREAM\_SETTINGS = {  'MANAGER': 'myapp.managers.MyActionManager',  'FETCH\_RELATIONS': True,  'USE\_JSONFIELD': True, } |
| --- |

**MANAGER**: Bu ayar, özel bir action (eylem) yöneticisi tanımlamak için kullanılır. myapp.managers.MyActionManager, aktiviteleri yönetmek için özel bir yöneticiyi işaret eder. Bu, aktiviteleri filtrelemek veya özelleştirilmiş sorgular yapmak için kullanılabilir.

**FETCH\_RELATIONS**: Bu ayar, ilişkili modellerin (örneğin, bir gönderiye yapılan yorumların) sorgular sırasında otomatik olarak getirilip getirilmeyeceğini belirler. True olarak ayarlandığında, ilişkili veriler otomatik olarak getirilir ve performans artırılabilir.

**USE\_JSONFIELD**: Bu ayar, aktivitelerin depolanmasında JSONField kullanılıp kullanılmayacağını belirler. True olarak ayarlandığında, aktiviteler JSON formatında saklanır. Bu, esneklik sağlar ve ilişkisel olmayan verilerin depolanmasını kolaylaştırır.

Bu ayarlar, Django uygulamanıza güvenlik ve kullanıcı aktivitelerini izleme gibi ek özellikler eklemenizi sağlar. django-honeypot spam ve bot saldırılarına karşı koruma sağlarken, django-activity-stream kullanıcı aktivitelerini kaydetmenizi ve izlemenizi sağlar

**django-honeypot** : Django projelerinde spam ve bot saldırılarına karşı koruma sağlamak için güçlü ve kullanımı kolay bir araçtır. Bu paket, formlara eklenen gizli alanlar sayesinde botları tespit eder ve engeller.Bir "honeypot" alanı ekleyerek, gerçek kullanıcılar tarafından doldurulmaması gereken gizli bir alan oluşturur. Bu alanın doldurulması, formun bir bot tarafından gönderildiğini işaret eder.

**HONEYPOT\_FIELD\_NAME**: Bu ayar, honeypot alanının adını belirler. Varsayılan alan adı honeypot iken, bu ayarla my\_honeypot\_field olarak değiştirilmiştir. Bu alan formda gizli olarak yer alır ve gerçek kullanıcılar tarafından doldurulması beklenmez. Eğer bu alan dolu olarak gönderilirse, bu bir spam girişimi olarak değerlendirilir.

django push notification:(Mobil) Apple Push Notification Service (APNs) ve Google Cloud Messaging (GCM) ile entegrasyon sağlar. Bildirimlerinizi hem iOS hem de Android cihazlarına gönderebilirsiniz.

pinax-notifications:(Çeşitli bildirim türleri sağlar) : Django uygulamaları için bildirim sistemleri oluşturmayı kolaylaştırır. Bu paket, kullanıcılarınıza çeşitli türde bildirimler göndermenize olanak tanır ve esnek bir yapı sunar. E-posta, SMS, web bildirimleri gibi çeşitli bildirim türlerini destekler.

**stream-django**: Stream, gerçek zamanlı akışlar ve bildirimler oluşturmayı kolaylaştıran bir kütüphanedir. stream-django, Django ile Stream hizmetini entegre etmenizi sağlar. Genellikle sosyal ağ uygulamaları ve etkinlik akışları için kullanılır.

**django-webpush**: Web push bildirimleri göndermek için kullanılır. Modern web tarayıcıları üzerinden kullanıcılarınıza push bildirimleri göndermek için uygun bir çözümdür.

**sentry-telegram**:Sentry ile Telegram botları arasında entegrasyon sağlar. Hata ve uyarıları Telegram üzerinden almak için kullanılır.

**django-herald**:Django projeleri için genel bir bildirim sistemi sağlar. E-posta, SMS ve diğer bildirim yöntemlerini destekler.

**Django NotifyX**: Django projeleri için basit ve esnek bir bildirim sistemi sunar. Bildirimleri kullanıcılar arasında iletmek için kullanılabilir.

**django-notifs**:Django uygulamalarında bildirimleri yönetmek için bir sistem sağlar. Bildirimlerin veritabanında saklanması ve yönetilmesi konusunda yardımcı olur.

**django-messages-extends**: Django'nun yerleşik mesajlaşma sistemine ek özellikler ve genişletmeler sağlar. Kullanıcılar arasında mesajlaşmayı daha esnek hale getirir.

**django-gcm**: Google Cloud Messaging (GCM) ile entegrasyon sağlar. Android cihazlarına push bildirimleri göndermek için kullanılır.

**django-stored-messages**: Kullanıcılar arasında mesajların saklanmasını ve yönetilmesini sağlar. Bu mesajlar kullanıcı oturumu kapatıldığında bile saklanabilir.

**django-knocker**:Bildirimlerin yönetimi ve gönderimi için bir sistem sağlar. Kullanıcılar arasında çeşitli bildirim türlerini destekler.

**django-sitemessage**: Web sitesindeki kullanıcılar için mesajlar ve bildirimler oluşturur. Site içi mesajlaşma sistemleri için uygundur.

**django-subscription**:Abonelik tabanlı bildirim sistemleri için kullanılır. Kullanıcıların belirli konularda bildirim almak için abone olmalarını sağlar.

**django-user-streams**:Kullanıcılar arasındaki etkileşimleri takip etmek ve bildirimler oluşturmak için kullanılır. Özellikle sosyal uygulamalarda kullanıcı etkinliklerini yönetmek için uygundur.

**django-channels**:Django'nun gerçek zamanlı web özellikleri ekleyen bir pakettir. Web soketleri ve diğer gerçek zamanlı protokoller için destek sağlar. Alpha sürümüdür, dolayısıyla bazı özelliklerde değişiklik olabilir.

**kawasemi**:Web push bildirimleri göndermek için bir çözümdür. Kullanıcılar tarayıcılarına push bildirimleri gönderebilir.

**Django-Offline Messages**:Çevrimdışı olan kullanıcılar için mesajların saklanmasını ve sonradan iletilmesini sağlar. Kullanıcıların çevrimdışıyken gönderilen mesajları almasını sağlar.

**django-scarface**:Bildirim ve mesajlaşma sistemleri için bir yapı sağlar. Kullanıcılar arasında mesajlaşmayı ve bildirimleri yönetir.

**django-rest-messaging**:Django REST Framework ile uyumlu bir bildirim sistemidir. REST API üzerinden bildirimler yönetmek ve göndermek için kullanılır.

**openwisp-notification**:OpenWISP platformunun bir parçası olarak bildirim sistemleri sağlar. OpenWISP projelerinde bildirimleri yönetmek için kullanılır.

**django\_simple\_notification**: Basit bir bildirim sistemi sağlar. Kullanıcılar arasında basit bildirimlerin yönetimi ve gönderimi için kullanılır. Genellikle az sayıda özelliğe sahip, kolay kullanımlı bir pakettir.

**Django AJAX messages**:Django uygulamanızda AJAX tabanlı mesajlaşma ve bildirimler sağlamak için kullanılır. Sayfa yeniden yüklemeden bildirimlerin gösterilmesini sağlar.

**django-heythere**: Basit bir bildirim ve mesajlaşma sistemi sağlar. E-posta ve diğer bildirim türlerini destekler ve kolay entegrasyon sunar.

**django-notifier**:Django uygulamaları için bir bildirim sistemi sağlar. Beta aşamasında olduğundan bazı özellikler eksik olabilir veya değişkenlik gösterebilir.

**universal\_notifications**:Universal bir bildirim sistemi sağlar. Birden fazla platform ve bildirim türü ile uyumlu olabilir. Genellikle geniş bir özellik setine sahiptir.

**django-user-messages**:Kullanıcılar arasında mesajlaşmayı yönetmek için kullanılır. Basit bir mesajlaşma sistemi sağlar ve kullanıcılar arasındaki iletişimi yönetir.

**django-webline-notifications**:Web uygulamaları için bildirim sistemleri sağlar. Web tabanlı bildirimler ve mesajlaşma özellikleri sunar.

**Django Infopush**: Bildirim göndermek için kullanılır, özellikle bilgi ve haber bildirimleri odaklıdır. Kullanıcılar için bilgi akışını yönetir.

**Django Courier**:Django için bildirim ve mesajlaşma sistemi sağlar. Ancak, bu paket güncel değil veya mevcut değil olabilir. (Django Vox olarak güncellenmiş olabilir.)

**Django Courier (now Django Vox)**:Django Courier, artık Django Vox olarak bilinmektedir. Bildirim sistemleri sağlar ve mevcut versiyonları Django Vox adı altında güncellenmiştir.

**django-nyt**: Bildirim sistemleri sağlar. Basit ve esnek bir yapı sunar, genellikle küçük veya orta ölçekli projeler için uygundur.

**django-signal-notifier**:Django sinyallerini kullanarak bildirimler oluşturur. Sinyal tabanlı bildirimler sağlar ve Django’nun sinyal mekanizmasını kullanır.

**Django Vox**:Eski adıyla Django Courier, güncel versiyonu olarak bilinir. Genel bildirim ve mesajlaşma sistemleri sağlar.

**realtime-feeds**:Gerçek zamanlı akışlar ve bildirimler sağlar. Genellikle anlık bildirimler ve veri akışları için kullanılır.

**Dinamik ve Statik Bildirimler Kullanımı**

Küçük basit projeler için statik bildirimler yeterli olabilrken ,kullanıcı etkileşimin yoğun olduğu projelerde dinamik bildirimler tercih edilmesi daha uygundur.

Statik bildirimlerde sunucu üzerinde ek bir yük oluşturulmaz.Kullanıcı tarayıcısında doğrudan render edilmektedir.Çok sayıda statik bildirim yönetimi zor olmaktadır .

Dinamik bildirimlerde sunucu tarafında daha fazla işlem yapılması gerekmekte bunun performans çözümlerine gidilmesi gerekmektedir .Bunu da birkaç yontemle arttırabilirz.

**Dinamik Bildirimler İçin Kullanılan Yöntemler:**

**1-WebSocket kullanımı:**

Geleneksel Http protokolü yerine çift yönlü iletişim sağlayan web socket kullanılabilir.Websocketle sürekli açık bir bağlantı kurulur ve sunucu bildirim göndermek için HTTP isteklerini beklemek zorunda kalmaz.

**Django channels** kullanımı: Django Channels django projelerinde websocket destegini eklmek için kullanabiliecegimiz bir araçtır.Bu dinamik bildirimlerinizi daha etkin ve performanslı bir şekilde yönetmeyi sağlar.

**2-Mesaj Kuyruğu Sistemleri**

**Redis:** Redis hızlı ve verimli bir yapı kullanarak gerçek zamanlı bildirimlerin işlenmedisne yardumcuır.Redisin pub/sub mekanizması ,bildiirmlerin dağıtımını hızlandırabilir.

**Celery:** Celery arka planda çalışan görevleri yönetmek için kullanılır.Bildirimlerin zamanlanmış veya gecikmeli olarak gönderilmesinin sağlayarak yük dengelemesi yapabiliriz.

**3-Ajax ve Pooling**

**Ajax:** dinamik içerikleri güncellmek içi ajax isteklerde bulunabilir Belirli aralıklara sunucudan yeni bildirim almayı sağlar.

**Long Polling:** Ajax ‘ın bir versiyonu olan Longpooling sunucu ile uzun süreli bağlantı kurarak yeni bildirim geldikçe istek gonderir.Websocketin bir alternatifi olabilir ama basit yapıda .

**4-Asynchronous Processing**

**Async Views:** Django'da asenkron görünümler (async views) kullanarak, I/O-bound işlemleri daha verimli bir şekilde yönetin.

**Asynchronous Task Queues**: Görevlerinizi asenkron olarak işlemek için Celery gibi görev kuyruğu sistemlerini kullanın.

**Statik Bildirimler İçin Kullanılan Yöntemler**

### **1-Caching (Önbellekleme)**

**Memcached veya Redis**: Sık kullanılan verileri önbelleğe alarak sunucu üzerindeki yükü azaltabilirsiniz. Bu, özellikle sık sık değişmeyen statik veri için faydalıdır.

**Django Cache Framework**: Django'nun yerleşik önbellek frameworkünü kullanarak belirli sorguları ve sonuçları önbelleğe alabilirsiniz.

### **2-Load Balancing (Yük Dengeleme)**

**Nginx veya HAProxy**: Trafiği birden fazla sunucuya dağıtarak yük dengelemesi yapabilirsiniz. Bu, sunucunuzun üzerindeki yükü azaltarak performansı artırır.

**Horizontal Scaling:** Gerektiğinde yeni sunucular ekleyerek sisteminizi yatay olarak ölçeklendirin.

### **3- Database Optimization (Veritabanı Optimizasyonu)**

**Indexing**: Veritabanı tablolarındaki sık kullanılan sütunları indeksleyerek sorgu performansını artırın.

**Query Optimization:** Veritabanı sorgularınızı optimize edin. Gereksiz karmaşık sorgulardan kaçının ve yalnızca gereken verileri sorgulayın.

**Database Sharding:** Veritabanı yükünü dağıtmak için veritabanı parçalama (sharding) tekniklerini kullanın.

### **4-Efficient Data Transfer**

**Data Compression:** JSON veya diğer veri formatlarını sıkıştırarak ağ üzerinden veri transferini hızlandırabilirsiniz.

**Minimizing Payload:** Gereksiz verileri transfer etmekten kaçının. Yalnızca gereken bilgileri gönderin.

### **5- Content Delivery Network (CDN) Kullanımı**

CDN: Bildirimlerin statik kaynaklarını (örneğin, JavaScript dosyaları, stil dosyaları) CDN aracılığıyla dağıtarak sunucu üzerindeki yükü azaltın.

**Gerçekleştirim :**

**Proje Özeti:**

Bu projede, Django ve Django Channels kullanarak bir web uygulaması için temel bir bildirim sistemi geliştirilmiştir. Sistem, kullanıcılara gerçek zamanlı bildirimler göndermek amacıyla tasarlanmıştır. Bu bildirimler, kullanıcıların sisteme olan etkileşimlerine dayanarak tetiklenir ve kullanıcılara önemli olaylar hakkında anında bilgi verir.

### **Kullanılan Teknolojiler :**

**Django:** Web uygulamasının backend'ini geliştirmek için kullanıldı.

**Django Channels:** Gerçek zamanlı Websocket bağlantıları kurmak ve yönetmek için kullanıldı.

**Celery:** Uzun süren görevleri arka planda asenkron bir şekilde çalıştırmak için kullanıldı.

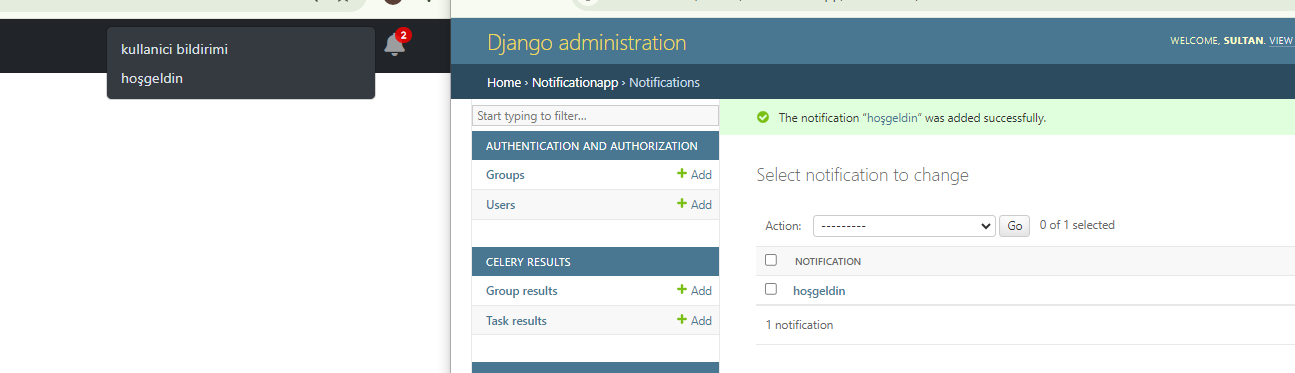
**Channel-Redis**: Bellek katmanı

Projede Django Channels ile doğrudan etkileşimi Celery görevine taşımak yerine, signal handler'ınızda Celery görevini tetikleyerek ve ardından bu Celery görevi içinde channels işlemlerini gerçekleştirerek daha modüler bir yapı oluşturuldu.

Django Channels'ı kullanarak WebSocket'e mesaj göndermek için aslında mevcut yapıyı korunabilir fakat bu işlemi Celery task'ında yapmak **uygulamamızın performansını arttırmaktadır**. İşlemi arka planda yaparak, request-response döngüsünü hızlandırıldı ve uygulama üzerindeki yük azaltılmıştır.

Channesl katmanı kullanarak mesaj gonderme işlemini Celery task olrak tanımladık asenkron kodu senkron bir şekilde çalıştırılmıştır.Single header kullanarak Celery task teteiklenmiş olup,notification modeli kaydedildiğinde,bu task arka planda çalışmış olur.Aynı zamanda, Celery aracılığıyla yönetilen bu asenkron işlemler, daha ölçeklenebilir bir yapı sağlar ve potansiyel olarak yoğun trafik durumlarında daha iyi performans gösterir.

| from django.db.models.signals import post\_save from django.dispatch import receiver from channels.layers import get\_channel\_layer from asgiref.sync import async\_to\_sync from .models import Notification from .tasks import send\_notificationgroup @receiver(post\_save, sender=Notification) def notification\_created(sender, instance, created, \*\*kwargs):  if created:  send\_notificationgroup.delay('group\_name', instance.message) |
| --- |





Basic Notification App - (channel-redis-celery) <https://github.com/sultansrzybkb/celery-redis-notification>

Social Notification App - (channel/web-socket) <https://github.com/sultansrzybkb/social-notificationApp>

**Gelecek Çalışmalar:**

### **Bildirim Tiplerine Göre Modelleme**

### Bildirim tiplerini (email, SMS, push notification vb.) modellemek ve kullanıcının hangi tür bildirimleri almak istediğini özelleştirmesine olanak tanımak için, Django model yapınızı genişletebilirsiniz. Bu, kullanıcı deneyimini kişiselleştirir ve daha etkili bir bildirim yönetimi sağlar.Bunun için pinax notification paketi kullanılabilir..

### **Kullanıcı Tercihlerine Göre Bildirim Ayarları**

Kullanıcıların hangi tür bildirimleri almak istediklerini belirleyebilecekleri ayarlar sunmak, gereksiz bildirimleri engeller ve kullanıcı memnuniyetini artırır.

Bu yöntemler, Django'da bildirim sistemlerinin daha etkili ve ölçeklenebilir bir şekilde yönetilmesine olanak tanırken, aynı zamanda sistem performansını ve kullanıcı deneyimini artırmaya yardımcı olur.