**Integrating Vonage API with Django Application**

[[](https://medium.com/@benarmys4?source=post_page-----79fa9e7e5c94--------------------------------)](https://medium.com/@benarmys4?source=post_page-----79fa9e7e5c94--------------------------------)

[[AWS Tip](https://awstip.com/?source=post_page-----79fa9e7e5c94--------------------------------)](https://awstip.com/?source=post_page-----79fa9e7e5c94--------------------------------)

[Benjamin Agic](https://medium.com/@benarmys4?source=post_page-----79fa9e7e5c94--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F4c4538469ec6&operation=register&redirect=https%3A%2F%2Fawstip.com%2Fintegrating-vonage-api-with-django-application-79fa9e7e5c94&user=Benjamin+Agic&userId=4c4538469ec6&source=post_page-4c4538469ec6----79fa9e7e5c94---------------------post_header-----------)

Published in

[AWS Tip](https://awstip.com/?source=post_page-----79fa9e7e5c94--------------------------------)

·

9 min read

·

Nov 16

10



Photo by [Firmbee.com](https://unsplash.com/@firmbee?utm_content=creditCopyText&utm_medium=referral&utm_source=unsplash) on [Unsplash](https://unsplash.com/photos/SpVHcbuKi6E?utm_content=creditCopyText&utm_medium=referral&utm_source=unsplash" \t "_blank)

**Introduction**

If you are seeking guidance on the integration of a messaging service with Django, you have arrived at the right place. Over the past year, I have utilized Twilio as a messaging service with satisfactory results. However, prompted by specific project requirements in an ongoing endeavor, I have opted to transition to a different service provider, Vonage.

Given the scarcity of information regarding Django projects utilizing Vonage, I encountered challenges during this transition. Consequently, I endeavor to share my experiences through this article, aspiring to provide valuable insights for individuals who are thinking about the integration of Vonage into their projects.

**Prerequisites**

*I assume you already know the basics of Django so I wont waste too much time on explaining the models, urls and settings files.*

So before we begin there are couple of things you need to make sure to have in place. First you can use any IDE you want, I will be using Visual Code for this example. Once you decide on your IDE the following things you will need are:

1. Latest [python](https://www.python.org/downloads/) package
2. Django framework by runing **pip install django**
3. DRF by running **pip install restframework**
4. Vonage package, run **pip install vonage**
5. Download [Ngrok](https://ngrok.com/download" \t "_blank) for domain staging.

Once you got all teqhnical requirements go ahead and setup a Django project, you can follow this tutorial here if you are not familiar on how to do it: [Django tutorial](https://docs.djangoproject.com/en/4.2/intro/tutorial01/).

If you want to follow along you can download this project template I already made in Django which gives you the basic setup, so I dont have to write it all here in this article: <https://github.com/FindingBen/VonageApp>

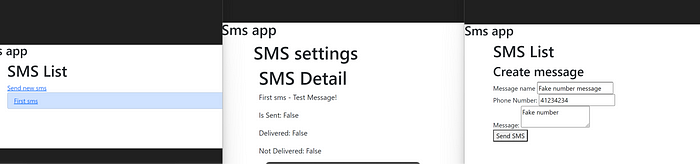
There is a requirements.txt file in there **run pip install -r requirements.txt**. Now that you got these steps lets move on the continuation of implementation.

**Django project walkthrough**

Before we move to Vonage lets quickly take a look at the template structure. This project has Sms app where I defined two views, **get\_sms** and **get\_sms\_detail**which gets us a list of sms’s and when we click on one of them it gets us a detailed view. Also there is a Sms model with following fields:

**unique\_tracking\_id**, **name**, **phone\_number**, **text\_body**, **is\_sent**, **delivered**, **not\_delivered**

I leverage pre-existing and configured Django templates, where the foundational template allows other templates to inherit essential HTML code and specific code blocks. To swiftly initiate the project and make sure u got all dependencies, execute “pip install -r requirements.txt”. In the ‘urls’ file within the SMS app, the paths are defined. Upon navigation to these paths, the rendered output is as follows:



1st one: Sms list , 2nd one: sms detail and 3rd one Create/Send sms

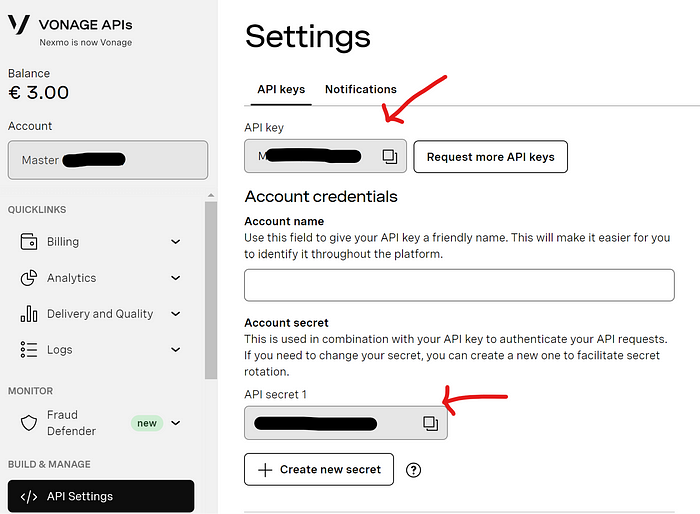
Now that our app is ready we can move on to message sending.

**Vonage configuration**

In order to use Vonage go to their official website and create an account: <https://ui.idp.vonage.com/ui/auth/registration>

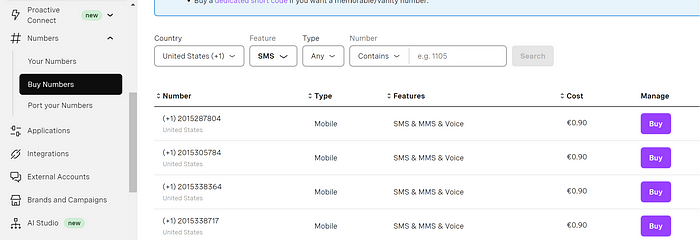
Once you completed the registration, verify your email and you are good to go. After that you need to top up some credits into your account in order to use a API service, the price per sending of sms varies from country to country, you can read more about it here:<https://www.vonage.com/communications-apis/sms/pricing/>

Once you set up your credits, go to api settings page and you can find your client information there, you will need an api key and api secret.



Vonage dashboard for developers

Once you find your keys, store that somewhere safley. Then you will need to buy a registered phone number which will allow you to send messages:



Page where you can buy numbers, the cost is 0.90 per number and it will come as invoice monthly.

So once you got that we are finally ready to move to using this and sending the message with it.

**Creating a sending view**

In our views files of Sms app, where we already established send\_sms view, insert the updated code:

import vonage  
import uuid  
  
def send\_sms(request):  
 status = None  
  
 if request.method == 'POST':  
 phone\_number = request.POST.get('phone\_number')  
 message = request.POST.get('message')  
 name = request.POST.get('name')  
 # Use your Vonage API key and secret  
 client = vonage.Client(  
 key='Your key', secret='your secret')  
 sms = vonage.Sms(client)  
 unique\_tracking\_id = uuid.uuid4()  
 response = sms.send\_message({  
 'from': 'your registered number',  
 'to': phone\_number,  
 'text': message,  
 })  
  
 if response["messages"][0]["status"] == "0":  
 sms\_object = Sms.objects.create(  
 unique\_tracking\_id=unique\_tracking\_id,  
 name='Your Name',  
 phone\_number=phone\_number,  
 text\_body=message,  
 is\_sent=True,  
 delivered=False,  
 not\_delivered=False,  
 )  
 sms\_object.save()  
  
 status = 'Message sent successfully!'  
 else:  
 status = f"Message failed with error: {response['messages'][0]['error-text']}"  
  
 return render(request, 'index.html', {'status': status})

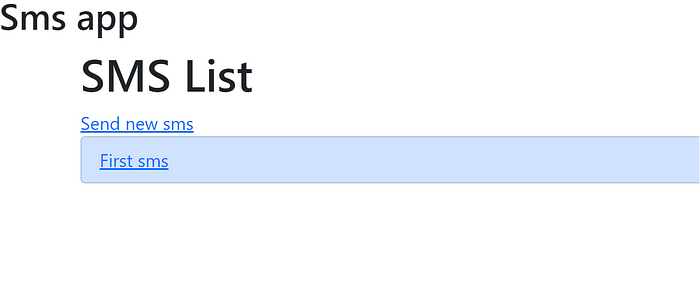
This makes sure that if the request is POST, then it proceeds with process. We configure our client with key and secret that you can get from Vonage dashboard and use sms.send\_messages function to send a text message. The ‘from’ parameter needs to contain the number that you bought from vonage.

If the response is successfull then we create a Sms object with the following fields(unique\_tracking\_id,name, phone\_number,text\_body,is\_sent,delivered,not\_delivered) which corresponds to our model of course.

*The unique\_tracking\_id will come in handy later.*

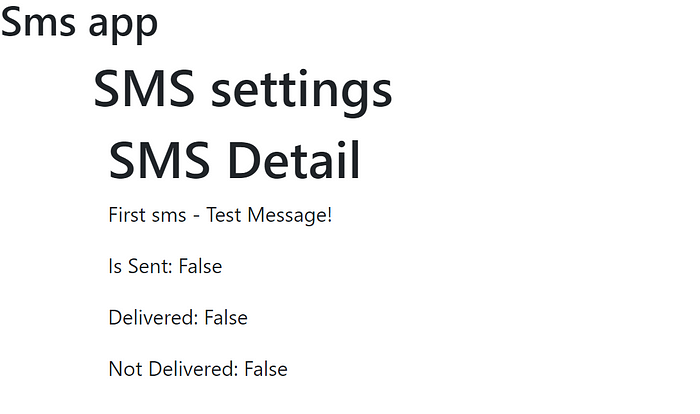
And then at the end we simply return request.

This should allow you now to send a text message to your phone number, make sure that number you insert is valid one. If we go to our create message template and insert parameters, we should be able to send the message and see it on our sms list page:



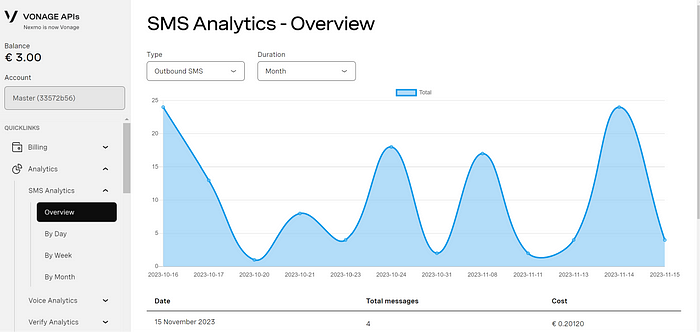
From the smsList template view

Now if we enter into the First sms we can see the following:



From sms template view

Now how do we know that our sms message actually got delivered to the recipient? We do get confirmation that message is indeed sent and if we go to our vonage dashboard analytics we can see the send results:



Vonage Sms Analytics dashboard

So we can see how messages got distributed, I got a big activity here as am using this service for a month now.

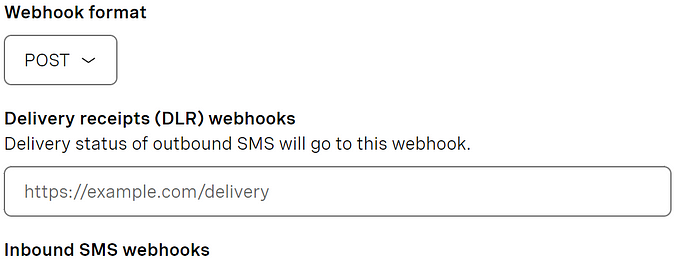
*But how do we verify if it actually got delivered to recipient?*

This is where webhook comes in! It allows us to recieve some information once the message got dispatched and we can then use that information to update our message details and see if it got delivered or not.

**Webhook configuration on Vonage**

First what is a webhook? From redhat.com : A webhook is an HTTP-based callback function that allows lightweight, event-driven communication between 2 [application programming interfaces (APIs)](https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces).

So there are two ways this can be done, either buy a staging domain and put the url of your webhook there and configure it in Django. Then go to your api settings and find **Delivery receipts (DLR) webhooks.**



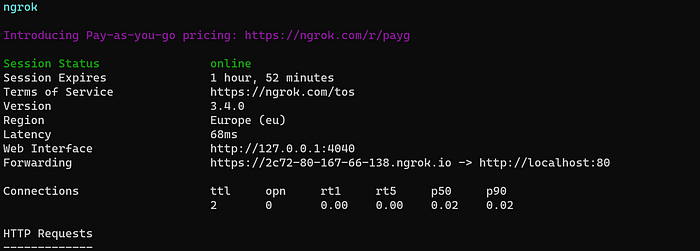
From Api settings page

Simple insert your url and you should be good to go.

But in this case for demonstration purposes we will do this locally. You would need a Ngrok which serves sort of like a terminal where we will configure our webhook and it will act as our staging domain. And then we will paste it in that delivery input in Vonage.

**Ngrok setup**

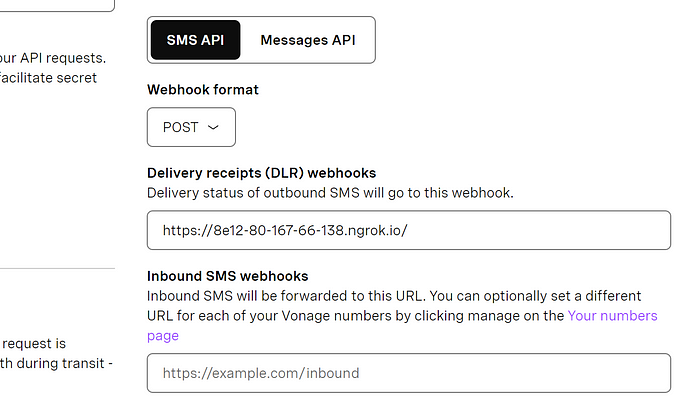
Once you download and install ngrok from here: <https://ngrok.com/download> open it by running **ngrok.exe http 80 in your**terminaland you should see something like this:



Ngrok

*Be sure that your port matches in the cmd, so for example if your port is 4444 then put that in : ngrok.exr http 4444.*

Now you can see the Forwarding part has a generated host, this is our staging domain, we will use this and paste it to webhook input in Vonage:



Now we need to get our webhook view setup in our views file so we can refer the rest of the urls to our webhook.

Remember how I said we will need our unique\_tracking\_id field later? Well that is now, we use the uuid to create sort of primary key for our Sms object, which we store in client-ref field that is comming from Vonage. Its one of many fields you can use when you configure your sending options, you can read more about it here: [https://developer.vonage.com/en/messaging/sms/guides/delivery-receipts?source=messaging](https://developer.vonage.com/en/messaging/sms/guides/delivery-receipts)

Now in our views.py file we create a following view:

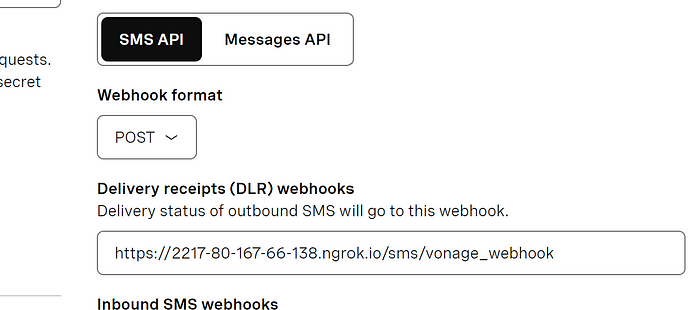
from django.shortcuts import get\_object\_or\_404, render  
from django.http import JsonResponse  
from rest\_framework.decorators import api\_view  
from rest\_framework.response import Response  
from .models import Sms  
from django.views.decorators.csrf import csrf\_protect  
from .serializers import SmsSerializer  
import vonage  
import uuid  
  
@csrf\_exempt  
@api\_view(['POST'])  
def vonage\_webhook(request):  
 data = request.data  
  
 sms\_object = Sms.objects.get(unique\_tracking\_id=data['client-ref'])  
  
 if data['status'] == 'delivered':  
 sms\_object.delivered = True  
 elif data['status'] == 'failed':  
 sms\_object.not\_delivered = True  
 elif data['status'] == 'rejected':  
 sms\_object.not\_delivered = True  
  
 sms\_object.save()  
  
 print('Received Vonage Delivery Receipt:')  
  
 # Respond with a success message  
 return JsonResponse({'message': 'Delivery receipt received successfully'}, status=200)

In the very begining we defined csrf\_exempt so we allow this view to be accessible since the request is not coming from our frontend client, but from webhook in Vonage. Then we simple retrieve the message object with the unique\_tracking\_id we defined earlier as client-ref. Then we use a if else statements to access the ‘status’ in data parameters. If the status is delivered we change our sms object parameter ‘delivered’ to True, if not then we change the ‘not\_delivered’ to True.

Now we need to create a urls path which we will append then to our ngrok host:

from django.urls import path  
from . import views  
  
urlpatterns = [  
 path('sms/', views.get\_sms\_list, name='sms-list'),  
 path('send-sms/', views.send\_sms, name='send-sms'),  
 path('sms/<int:id>/', views.get\_sms\_detail, name='sms-detail'),  
 path('vonage\_webhook', views.vonage\_webhook),  
]

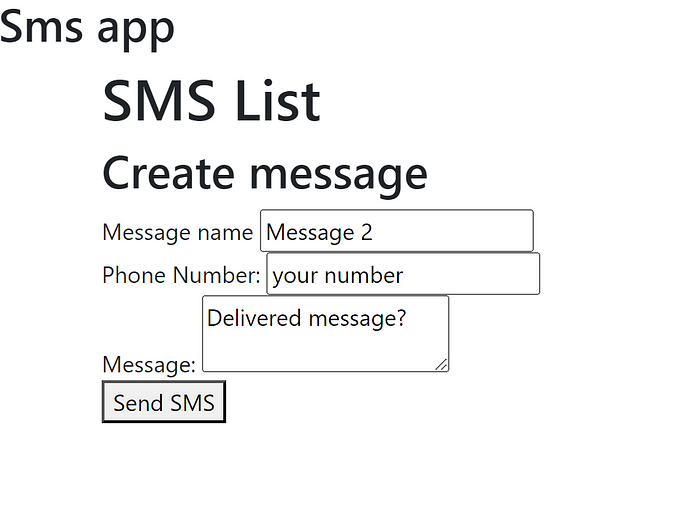
We add the path vonage\_webhook to our ngrok host which we got earlier from ngrok client and paste the full url to vonage webhook delivery input field:



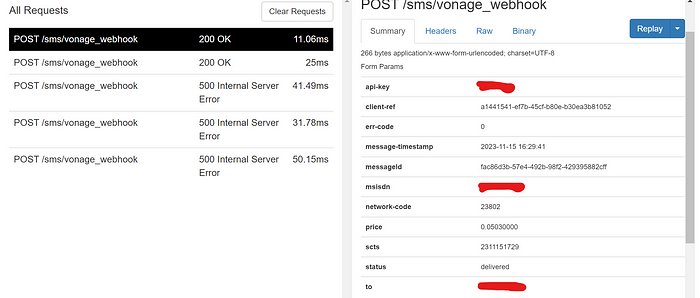
Our new webhook url which will interact with with vonage

Now go ahead and run <http://localhost:4040/inspect/http> , this will allow you to see the requests from your webhook url. Before we start testing it there is one last thing we have to do, in your settings.py file go to your ALLOWED\_HOSTS and put ‘\*’ value between the brackets. This is so that we allow ngrok requests to our django app, typically you would allow any host but for this purpose its fine, you can also just paste the ngrok url into the bracket instead of ‘\*’.

We go back to our message creation:

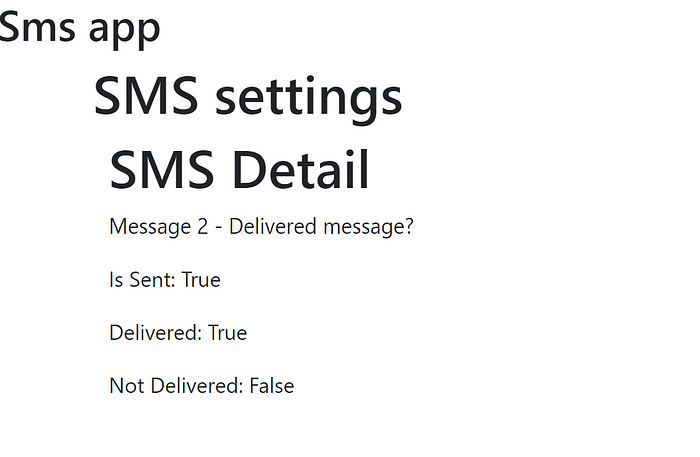


Now if we go ahead and send a sms message we get the following logs from our ngrok host:

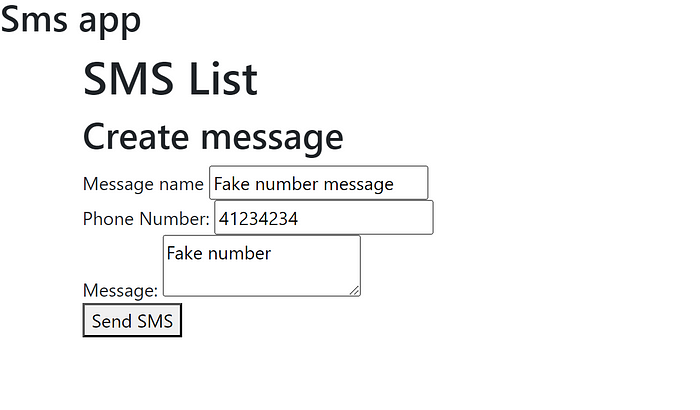


From our localhost:4040

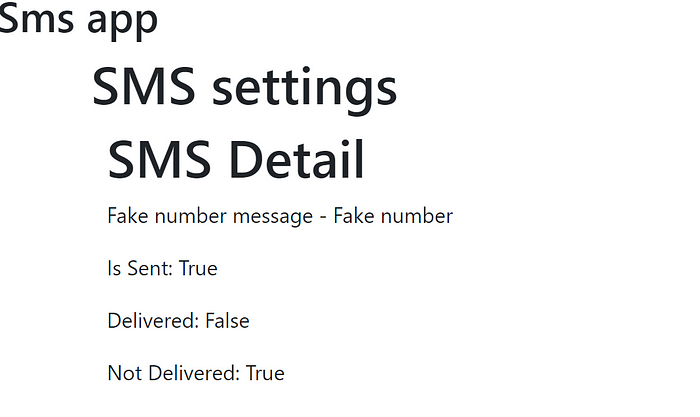
We can see that status is delivered which means message got delivered successfully, now we can use this information, catch it with out webhook function and update our sms object.



You can see that delivered field is True. Now if we set some fake number just for sake of results and testing like so:



After we hit send, we get the sms object, which indicates that message is indeed sent but if we get inside of that sms object we have this:



So we can see that its not been delivered!

**Conclusion**

The general setup of it is pretty simple I would say, this tutorial in particular focuses only on Vonage setup. Right now you can use only one phone, but that can be changed, you can make a list of phones you want to send text messages and just loop the sending function with it.

Overall I think its a great service and the documentation for it is also easy to follow.

Let me know down in comments if you found it interesting or helpfull and of course if you have any other feedback on it!