The **REST** acronym stands for Representational State Transfer, which is an architectural design. ... Now, if you want to provide a **REST** API, the **Django REST** Framework is the best option. It make easy to expose parts of your application as a **REST** API.

Django REST framework is a powerful and flexible toolkit for building Web APIs.

Some reasons you might want to use REST framework:

* The [Web browsable API](https://restframework.herokuapp.com/) is a huge usability win for your developers.
* [Authentication policies](https://www.django-rest-framework.org/api-guide/authentication/) including packages for [OAuth1a](https://www.django-rest-framework.org/api-guide/authentication/#django-rest-framework-oauth) and [OAuth2](https://www.django-rest-framework.org/api-guide/authentication/#django-oauth-toolkit).
* [Serialization](https://www.django-rest-framework.org/api-guide/serializers/) that supports both [ORM](https://www.django-rest-framework.org/api-guide/serializers#modelserializer) and [non-ORM](https://www.django-rest-framework.org/api-guide/serializers#serializers) data sources.
* Customizable all the way down - just use [regular function-based views](https://www.django-rest-framework.org/api-guide/views#function-based-views) if you don't need the [more](https://www.django-rest-framework.org/api-guide/generic-views/) [powerful](https://www.django-rest-framework.org/api-guide/viewsets/) [features](https://www.django-rest-framework.org/api-guide/routers/).
* [Extensive documentation](https://www.django-rest-framework.org/), and [great community support](https://groups.google.com/forum/?fromgroups#!forum/django-rest-framework).
* Used and trusted by internationally recognised companies including [Mozilla](https://www.mozilla.org/en-US/about/), [Red Hat](https://www.redhat.com/), [Heroku](https://www.heroku.com/), and [Eventbrite](https://www.eventbrite.co.uk/about/)

[Requirements](https://www.django-rest-framework.org/#requirements)

REST framework requires the following:

* Python (2.7, 3.4, 3.5, 3.6, 3.7)
* Django (1.11, 2.0, 2.1)

The following packages are optional:

* [coreapi](https://pypi.org/project/coreapi/) (1.32.0+) - Schema generation support.
* [Markdown](https://pypi.org/project/Markdown/) (2.1.0+) - Markdown support for the browsable API.
* [django-filter](https://pypi.org/project/django-filter/) (1.0.1+) - Filtering support.
* [django-crispy-forms](https://github.com/maraujop/django-crispy-forms) - Improved HTML display for filtering.
* [django-guardian](https://github.com/django-guardian/django-guardian) (1.1.1+) - Object level permissions support.

## [Installation](https://www.django-rest-framework.org/#installation)

Install using pip, including any optional packages you want...

pip install djangorestframework

pip install markdown # Markdown support for the browsable API.

pip install django-filter # Filtering support

from django.conf.urls import url, include

from django.contrib.auth.models import User

from rest\_framework import routers, serializers, viewsets

# Serializers define the API representation.

class UserSerializer(serializers.HyperlinkedModelSerializer):

class Meta:

model = User

fields = ('url', 'username', 'email', 'is\_staff')

# ViewSets define the view behavior.

class UserViewSet(viewsets.ModelViewSet):

queryset = User.objects.all()

serializer\_class = UserSerializer

# Routers provide an easy way of automatically determining the URL conf.

router = routers.DefaultRouter()

router.register(r'users', UserViewSet)

# Wire up our API using automatic URL routing.

# Additionally, we include login URLs for the browsable API.

urlpatterns = [

url(r'^', include(router.urls)),

url(r'^api-auth/', include('rest\_framework.urls', namespace='rest\_framework'))

]

You can now open the API in your browser at <http://127.0.0.1:8000/>, and view your new 'users' API. If you use the login control in the top right corner you'll also be able to add, create and delete users from the system.