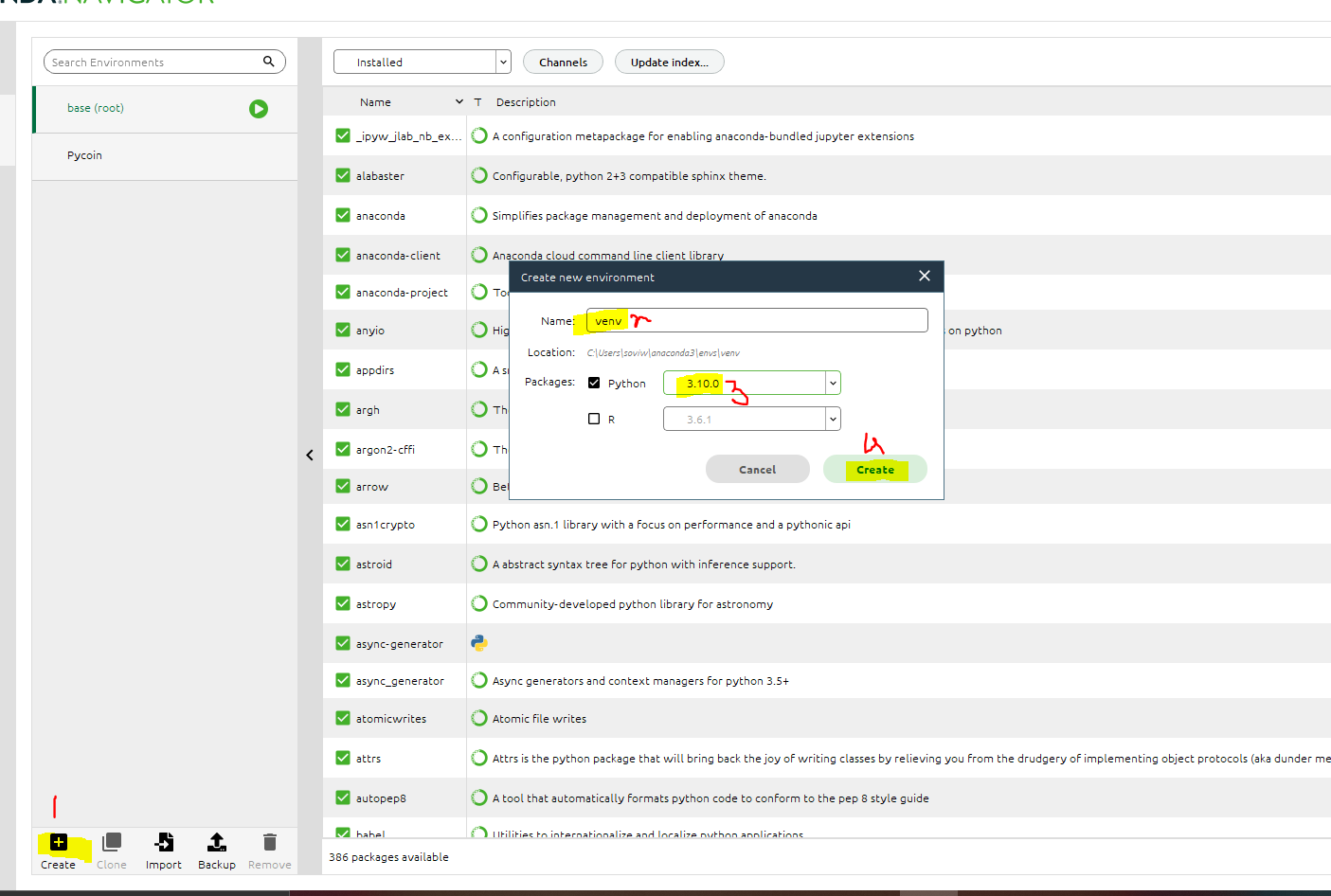
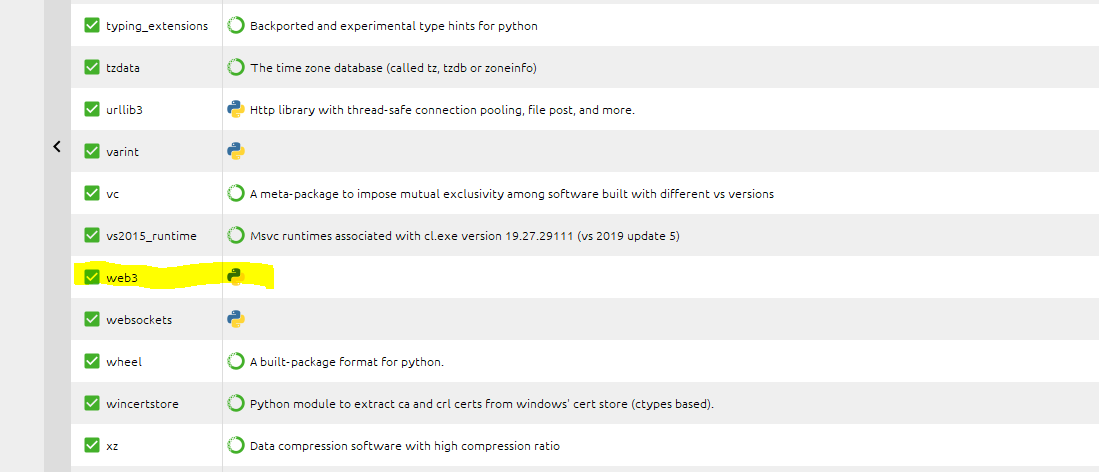
Django is a web framework similar to React/Angular but is intertwined with Python.

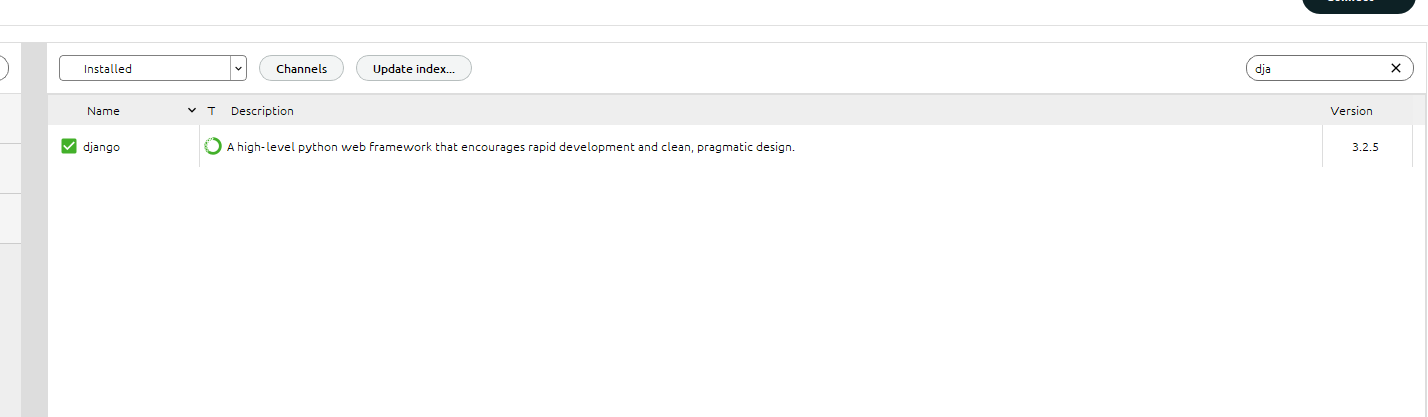
First we create an environment in Anaconda



Web3 installed:

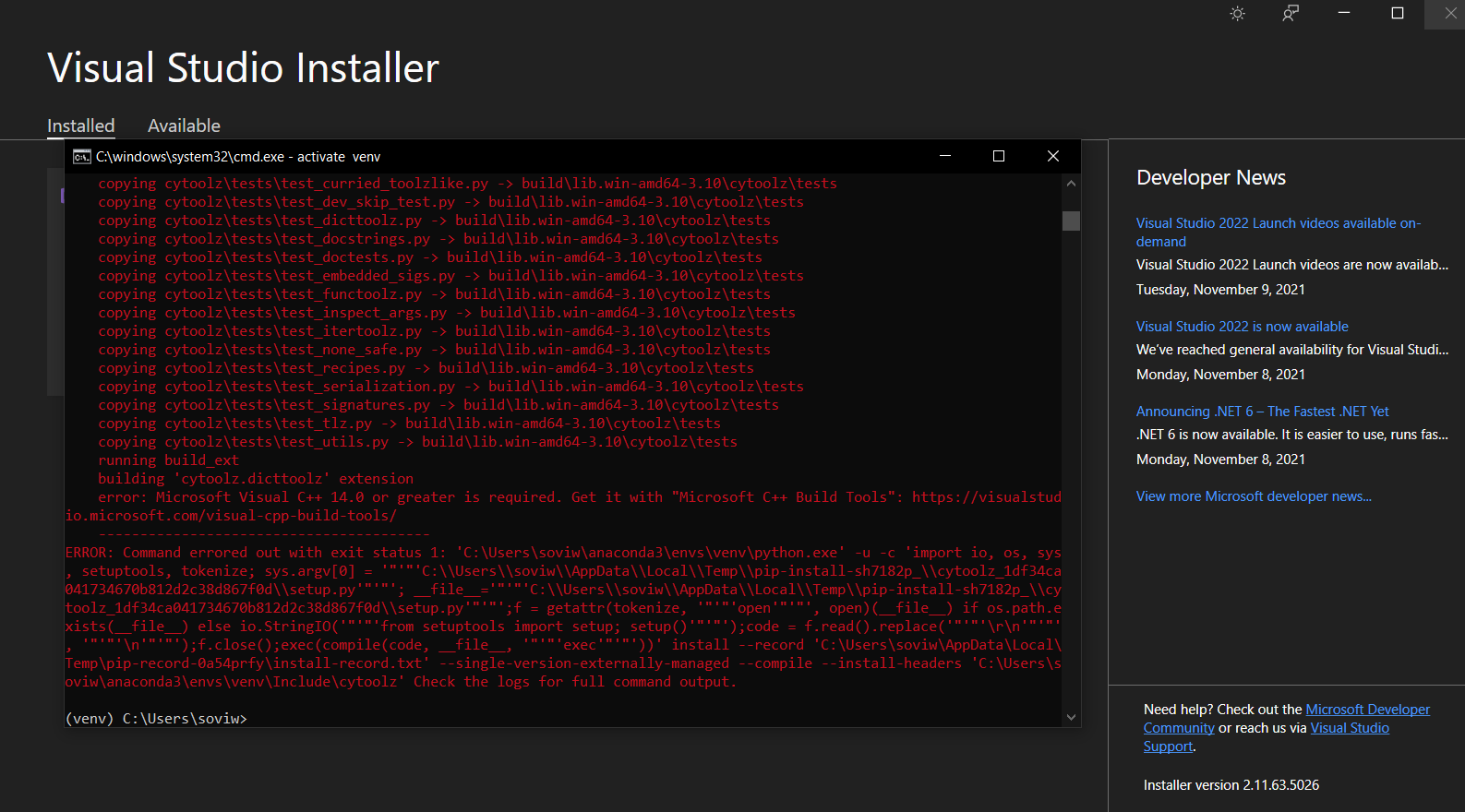


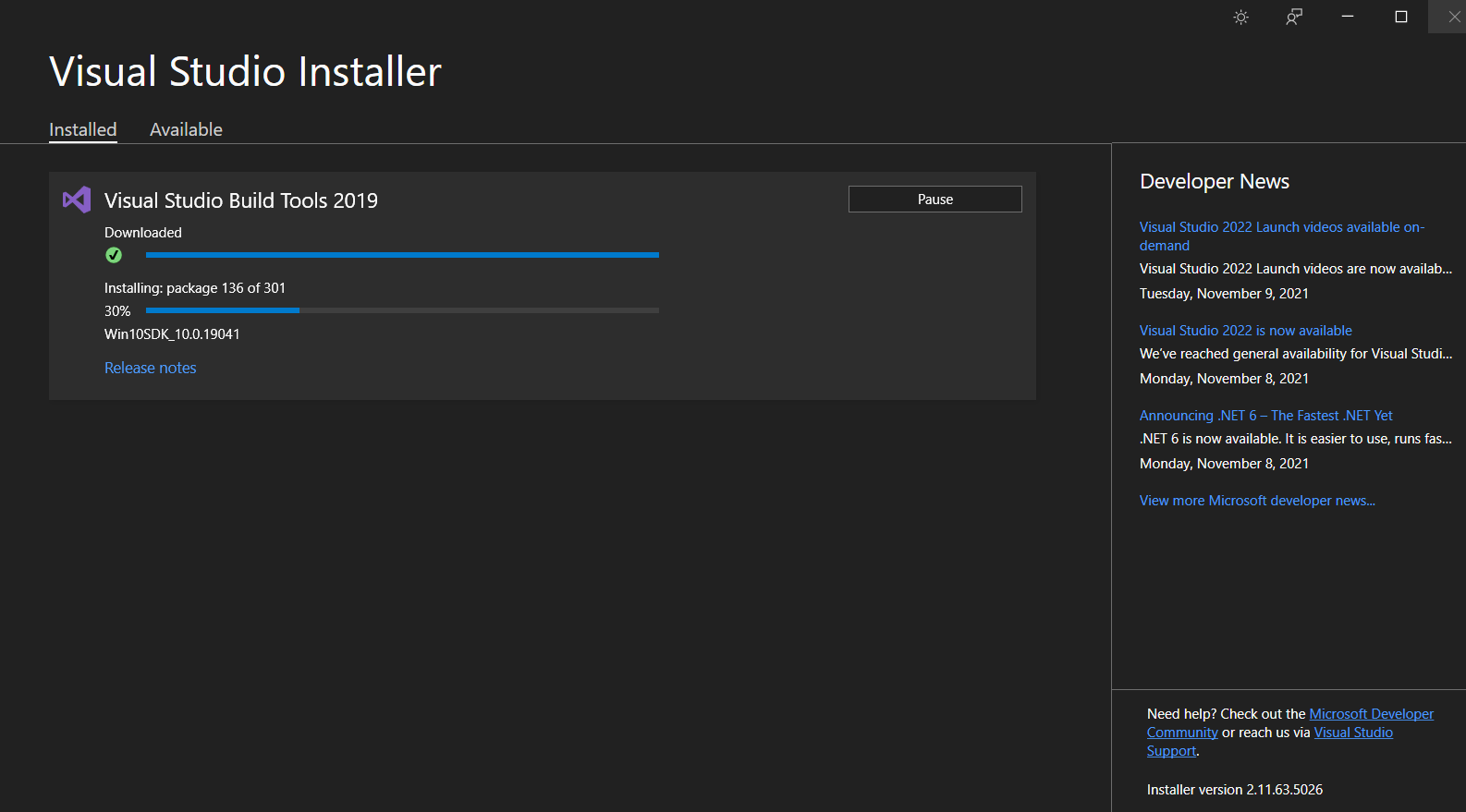
Install Django:



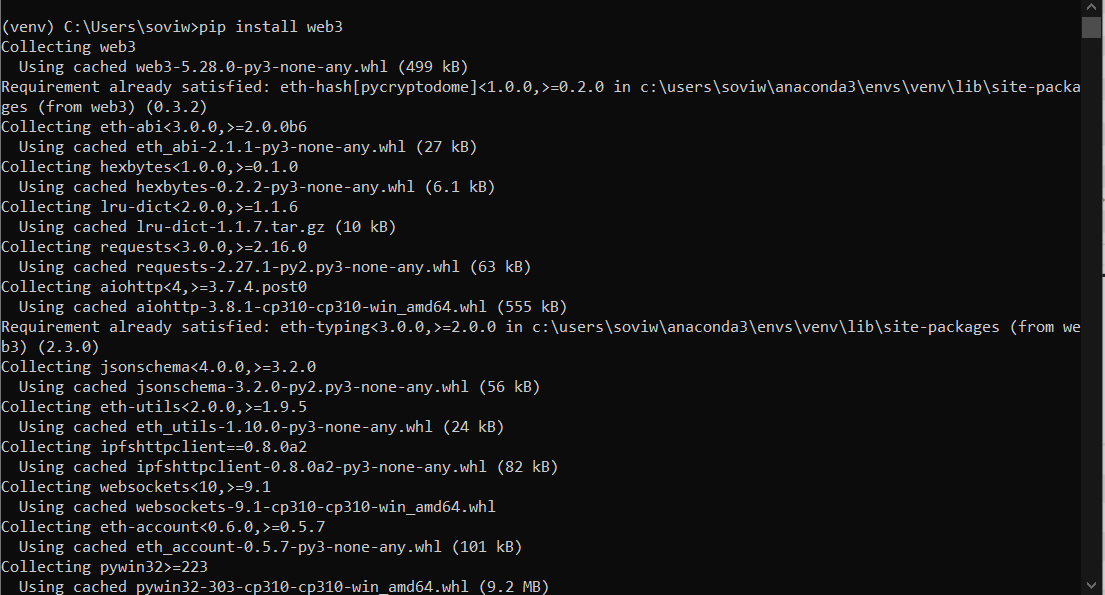
Install web 3:

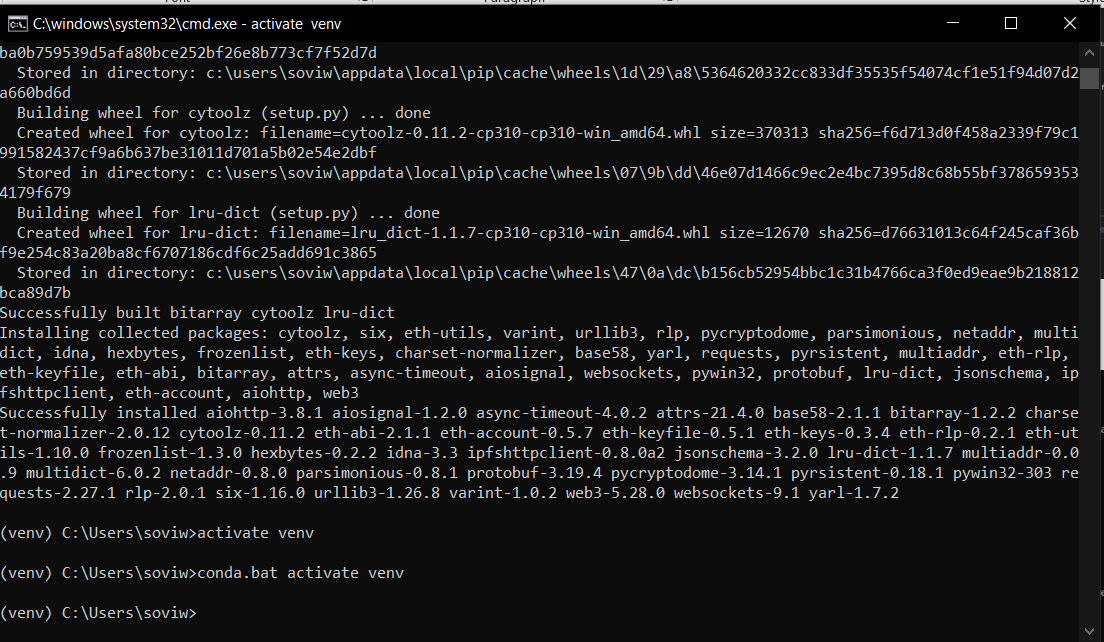
conda install -c conda-forge web3





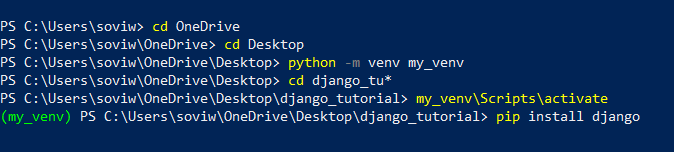
After restart:



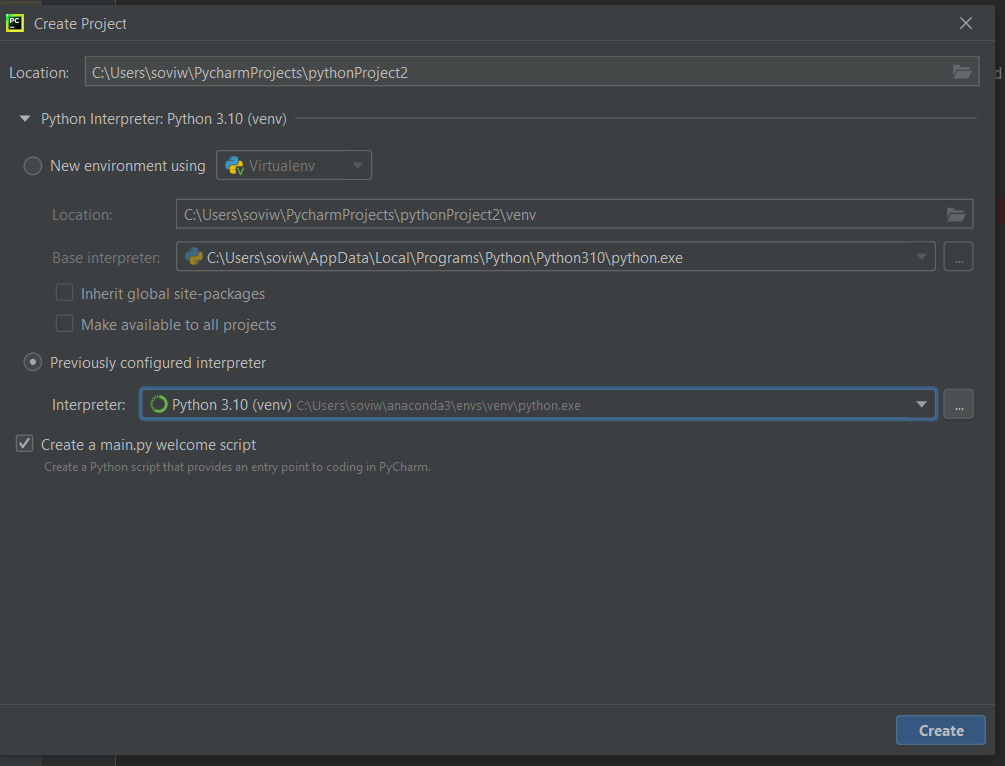


Now we go to our tutorial:

Install Django:



Or we can use below after Conda:



Now follow the steps:

Directory of C:\Users\soviw\PycharmProjects

08-03-2022 12:35 <DIR> .

08-03-2022 12:35 <DIR> ..

08-03-2022 12:33 <DIR> django\_tutorials\_2

08-03-2022 12:36 <DIR> Django\_tutorial\_2

24-11-2021 11:32 <DIR> pythonProject

20-01-2022 01:33 <DIR> pythonProject1

08-03-2022 11:47 <DIR> tutorial

0 File(s) 0 bytes

7 Dir(s) 174,719,778,816 bytes free

(venv) C:\Users\soviw\PycharmProjects>cd Django\_tutorial\_2

(venv) C:\Users\soviw\PycharmProjects\Django\_tutorial\_2>activate venv

(venv) C:\Users\soviw\PycharmProjects\Django\_tutorial\_2>conda.bat activate venv

(venv) C:\Users\soviw\PycharmProjects\Django\_tutorial\_2>django-admin startproject first ,

CommandError: Destination directory 'C:\Users\soviw\PycharmProjects\Django\_tutorial\_2\,' does not exist, please create it first.

(venv) C:\Users\soviw\PycharmProjects\Django\_tutorial\_2>django-admin --version

3.2.5

(venv) C:\Users\soviw\PycharmProjects\Django\_tutorial\_2>django-admin startproject first,

CommandError: 'first,' is not a valid project name. Please make sure the name is a valid identifier.

(venv) C:\Users\soviw\PycharmProjects\Django\_tutorial\_2>django-admin startproject first

(venv) C:\Users\soviw\PycharmProjects\Django\_tutorial\_2>python manage.py runserver

python: can't open file 'C:\\Users\\soviw\\PycharmProjects\\Django\_tutorial\_2\\manage.py': [Errno 2] No such file or directory

(venv) C:\Users\soviw\PycharmProjects\Django\_tutorial\_2>cd firts

The system cannot find the path specified.

(venv) C:\Users\soviw\PycharmProjects\Django\_tutorial\_2>cd first

(venv) C:\Users\soviw\PycharmProjects\Django\_tutorial\_2\first>python manage.py runserver

Watching for file changes with StatReloader

Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.

Run 'python manage.py migrate' to apply them.

March 08, 2022 - 13:02:11

Django version 3.2.5, using settings 'first.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

[08/Mar/2022 13:02:27] "GET / HTTP/1.1" 200 10697

[08/Mar/2022 13:02:28] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423

[08/Mar/2022 13:02:29] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP/1.1" 200 85876

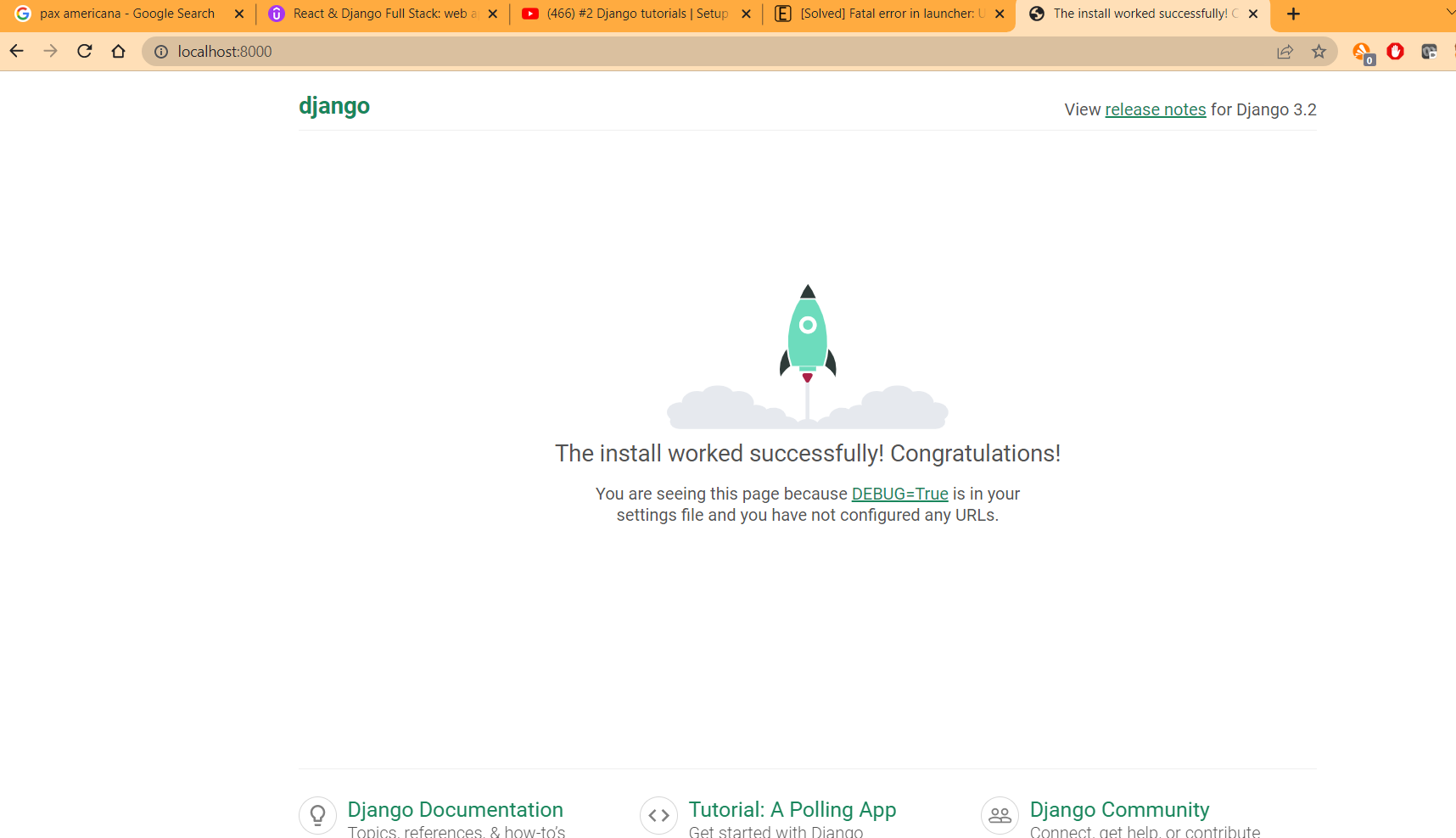
[08/Mar/2022 13:02:29] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.1" 200 86184

[08/Mar/2022 13:02:29] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1.1" 200 85692

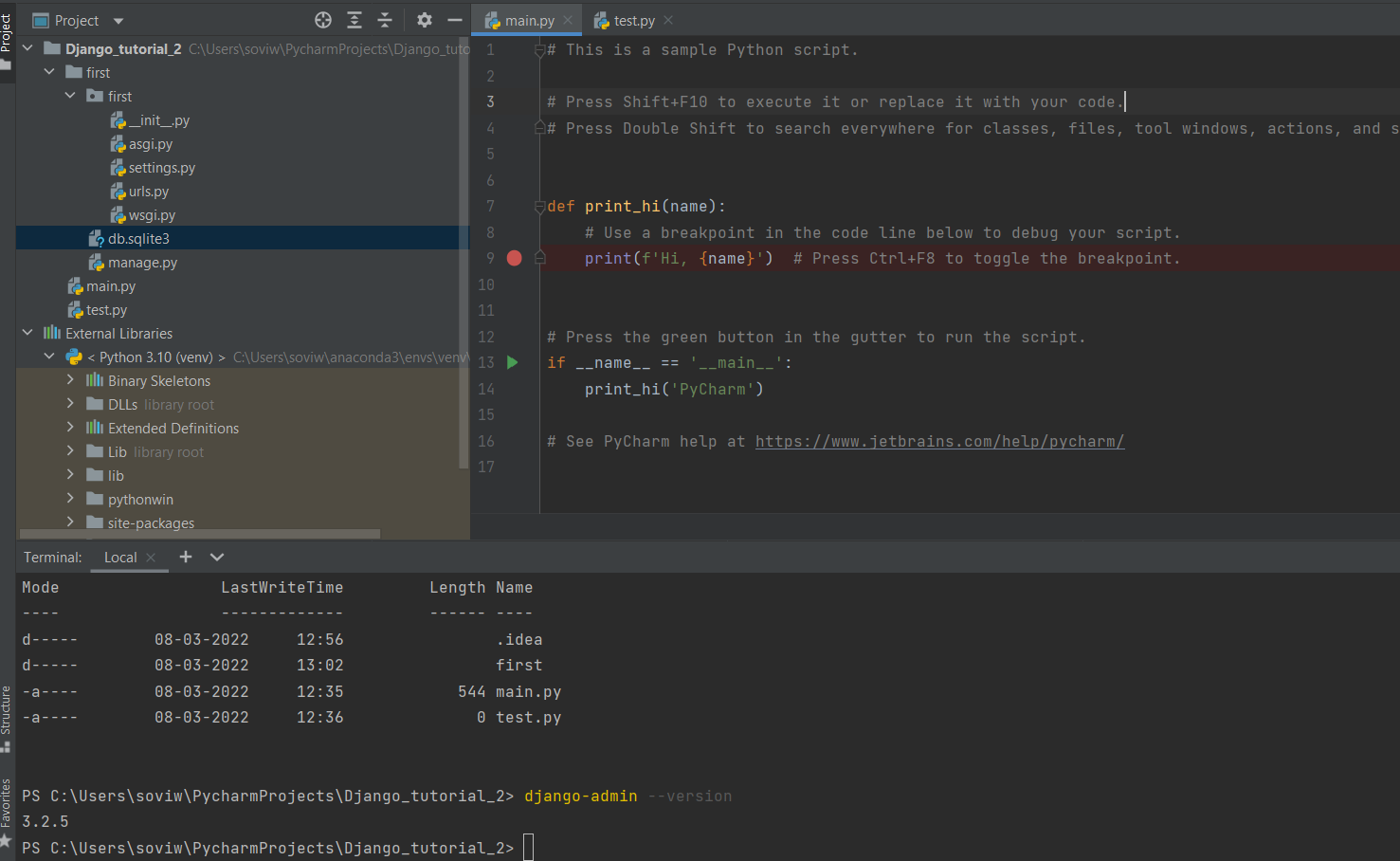
Not Found: /favicon.ico

[08/Mar/2022 13:02:29] "GET /favicon.ico HTTP/1.1" 404 2109

O/P:



Enviornment is also created:



PS C:\Users\soviw\PycharmProjects\Django\_tutorial\_2> django-admin --version

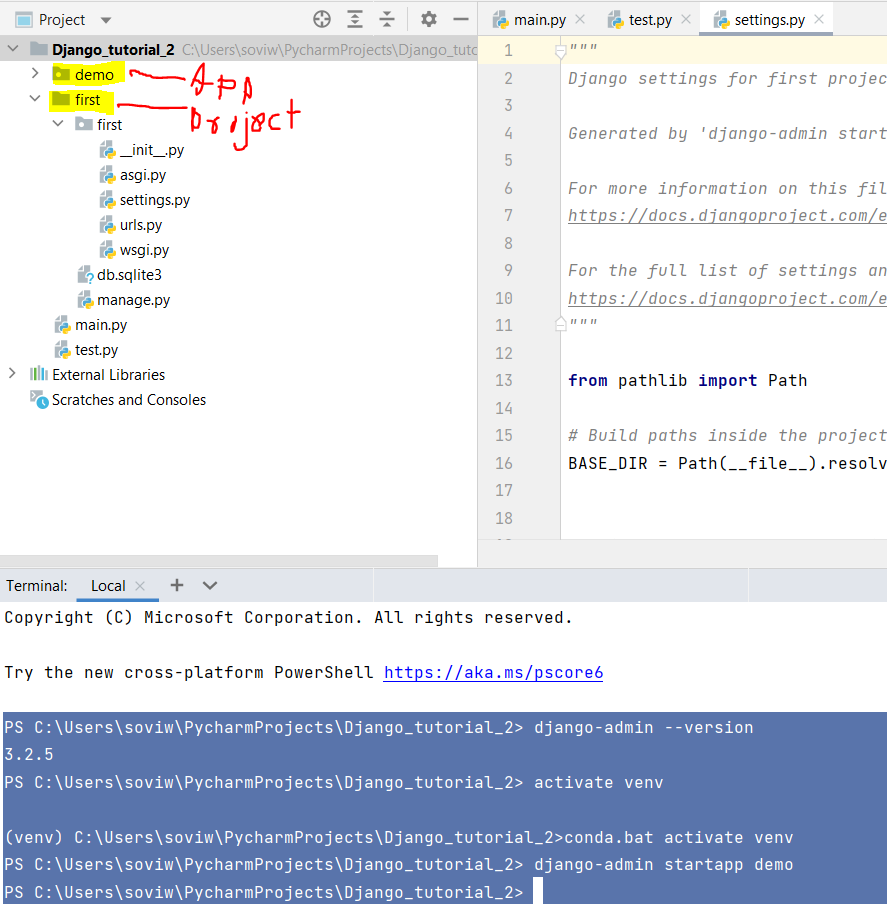
3.2.5

PS C:\Users\soviw\PycharmProjects\Django\_tutorial\_2> activate venv

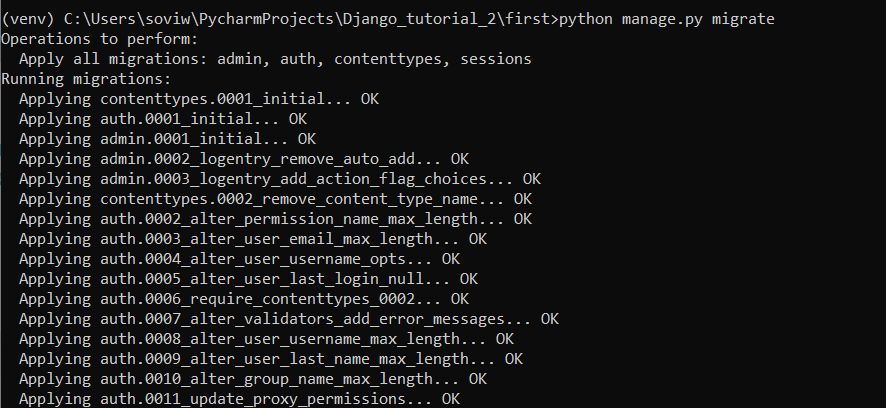
(venv) C:\Users\soviw\PycharmProjects\Django\_tutorial\_2>conda.bat activate venv

PS C:\Users\soviw\PycharmProjects\Django\_tutorial\_2> django-admin startapp demo

PS C:\Users\soviw\PycharmProjects\Django\_tutorial\_2>



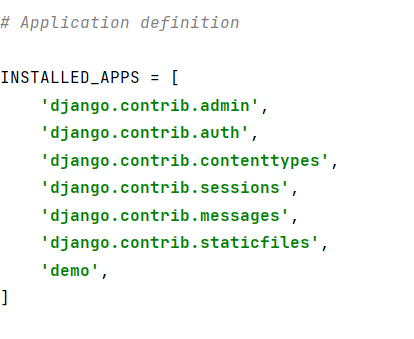
Migrations:



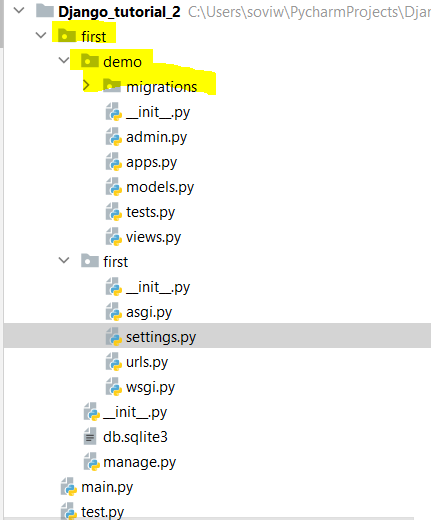
In migration folder we go to model.py:

**from** django.db **import** models  
  
*# Create your models here.***class** Book(models.Model):  
 title=models.CharField(max\_length=36)*#CharField is simple text. Max lenght 36*

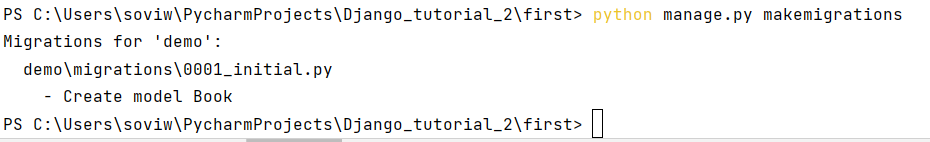
Goto settings.py:

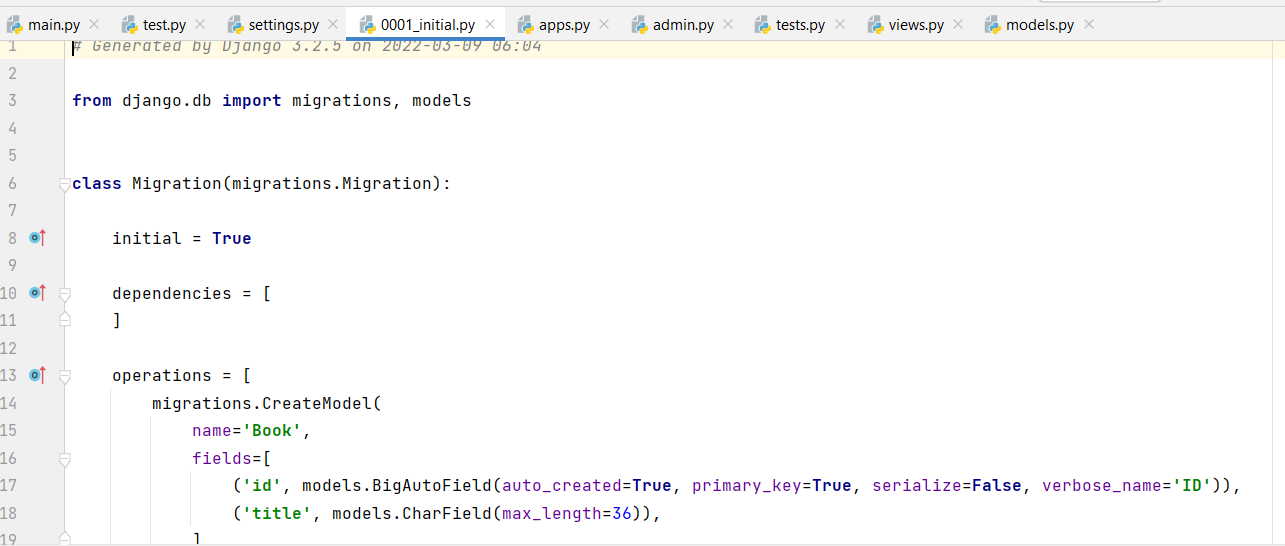


Refractor the demo app in the first project:

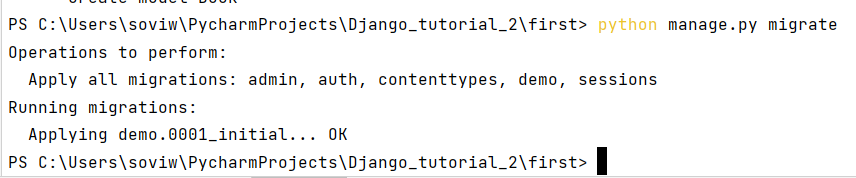


Now if we makemigrations:



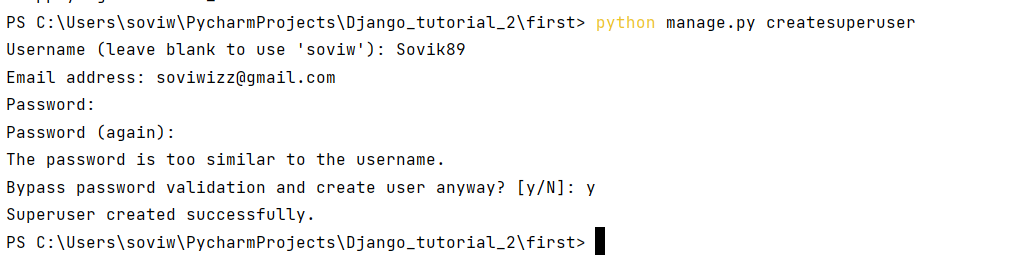


To migrate to the database:



User and admin:

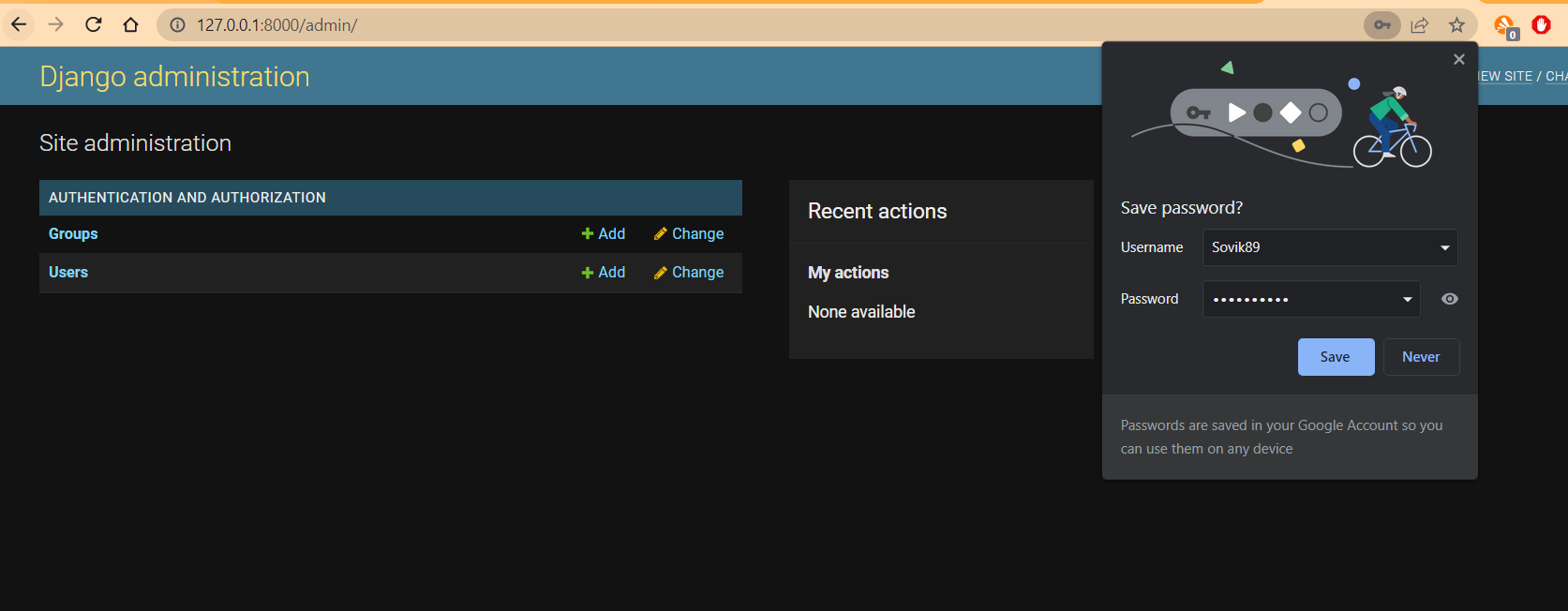
Create superuser:



After wich if I run the command:python manage.py runserver

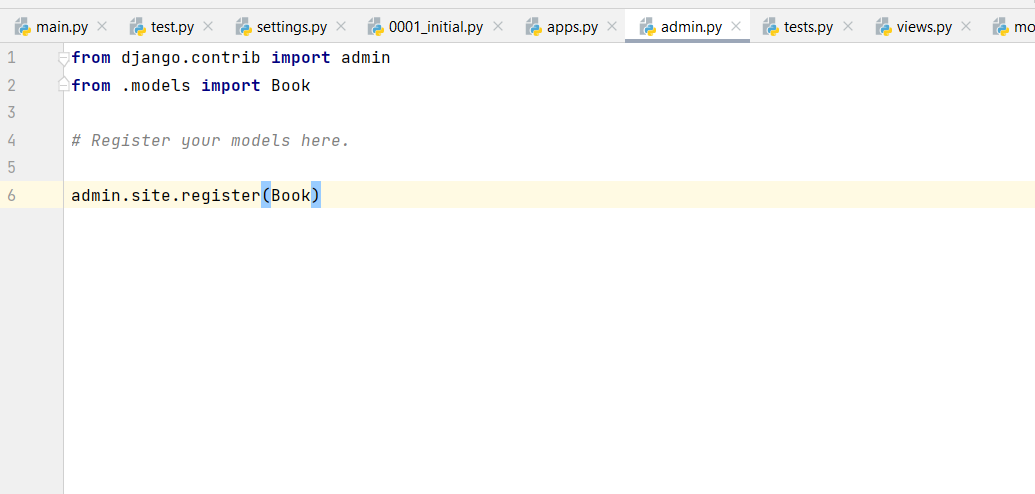
We go to localhost:8000/admin

And give the above credentials

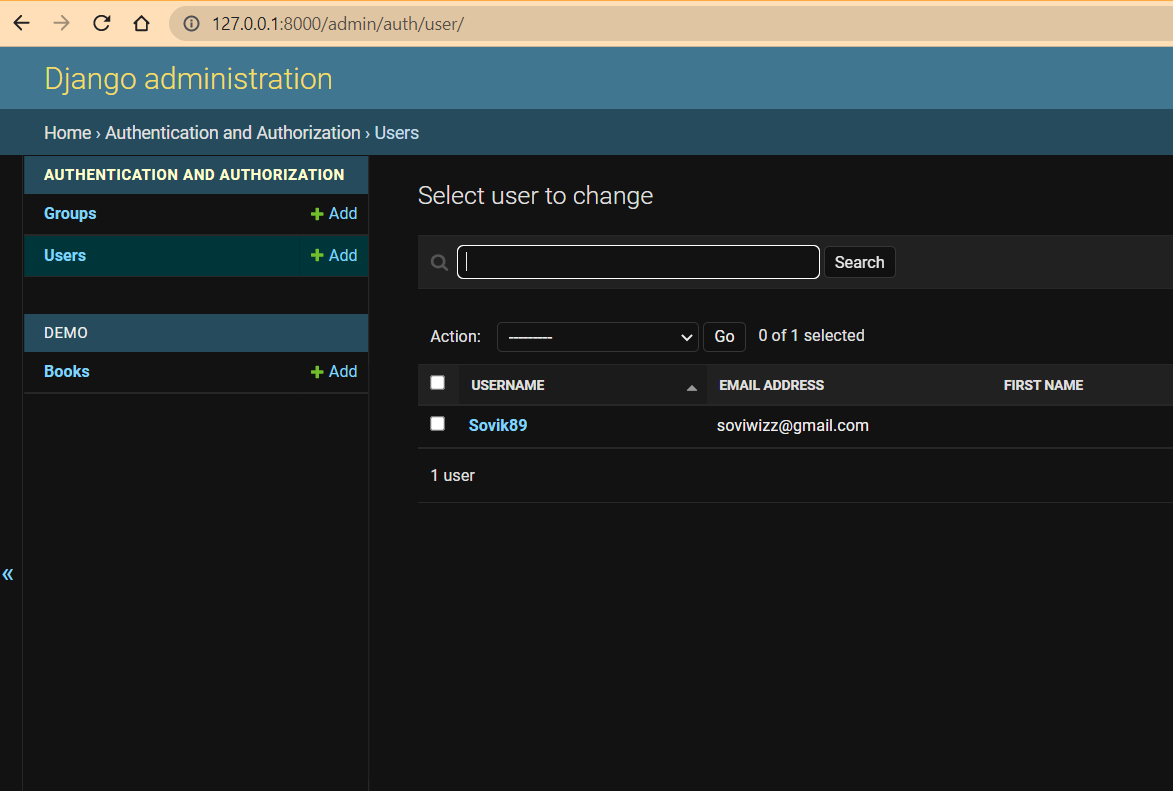


Now Book is not there in the admin panel:

For this we need to add in the admin,py:

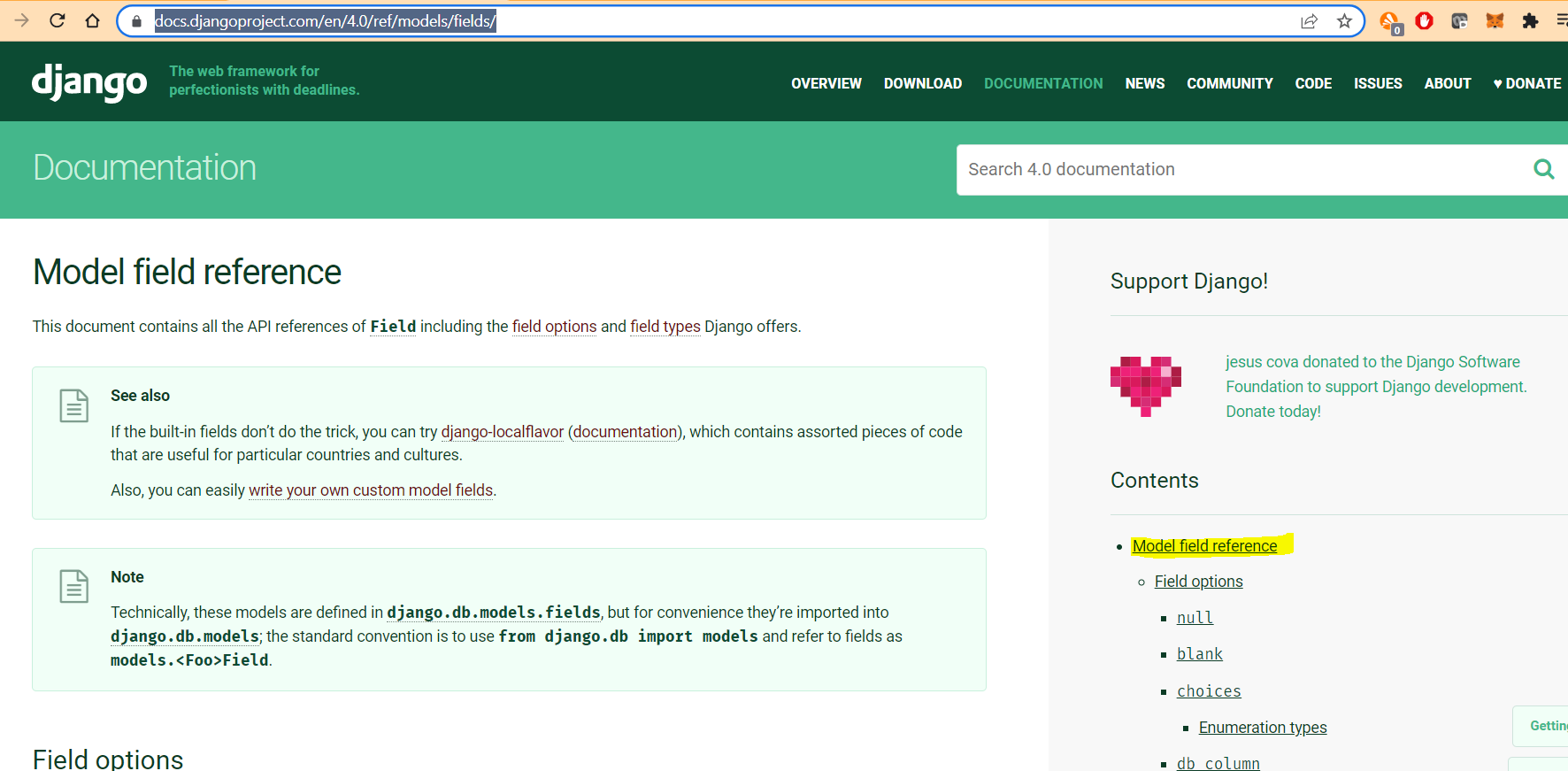


Now if we re-run the server we get:



Field Option:

<https://docs.djangoproject.com/en/4.0/ref/models/fields/>



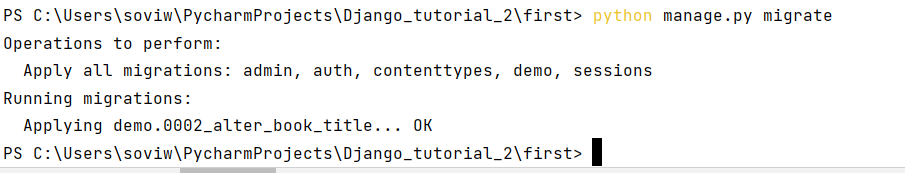
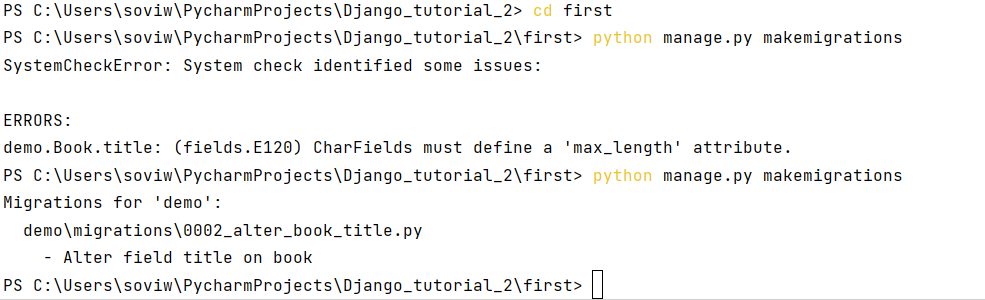
Options:

[Field options](https://docs.djangoproject.com/en/4.0/ref/models/fields/#field-options)

* [null](https://docs.djangoproject.com/en/4.0/ref/models/fields/#null)
* [blank](https://docs.djangoproject.com/en/4.0/ref/models/fields/#blank)
* [choices](https://docs.djangoproject.com/en/4.0/ref/models/fields/#choices)
  + [Enumeration types](https://docs.djangoproject.com/en/4.0/ref/models/fields/#enumeration-types)
* [db\_column](https://docs.djangoproject.com/en/4.0/ref/models/fields/#db-column)
* [db\_index](https://docs.djangoproject.com/en/4.0/ref/models/fields/#db-index)
* [db\_tablespace](https://docs.djangoproject.com/en/4.0/ref/models/fields/#db-tablespace)
* [default](https://docs.djangoproject.com/en/4.0/ref/models/fields/#default)
* [editable](https://docs.djangoproject.com/en/4.0/ref/models/fields/#editable)
* [error\_messages](https://docs.djangoproject.com/en/4.0/ref/models/fields/#error-messages)
* [help\_text](https://docs.djangoproject.com/en/4.0/ref/models/fields/#help-text)
* [primary\_key](https://docs.djangoproject.com/en/4.0/ref/models/fields/#primary-key)
* [unique](https://docs.djangoproject.com/en/4.0/ref/models/fields/#unique)
* [unique\_for\_date](https://docs.djangoproject.com/en/4.0/ref/models/fields/#unique-for-date)
* [unique\_for\_month](https://docs.djangoproject.com/en/4.0/ref/models/fields/#unique-for-month)
* [unique\_for\_year](https://docs.djangoproject.com/en/4.0/ref/models/fields/#unique-for-year)
* [verbose\_name](https://docs.djangoproject.com/en/4.0/ref/models/fields/#verbose-name)
* [validators](https://docs.djangoproject.com/en/4.0/ref/models/fields/#validators)

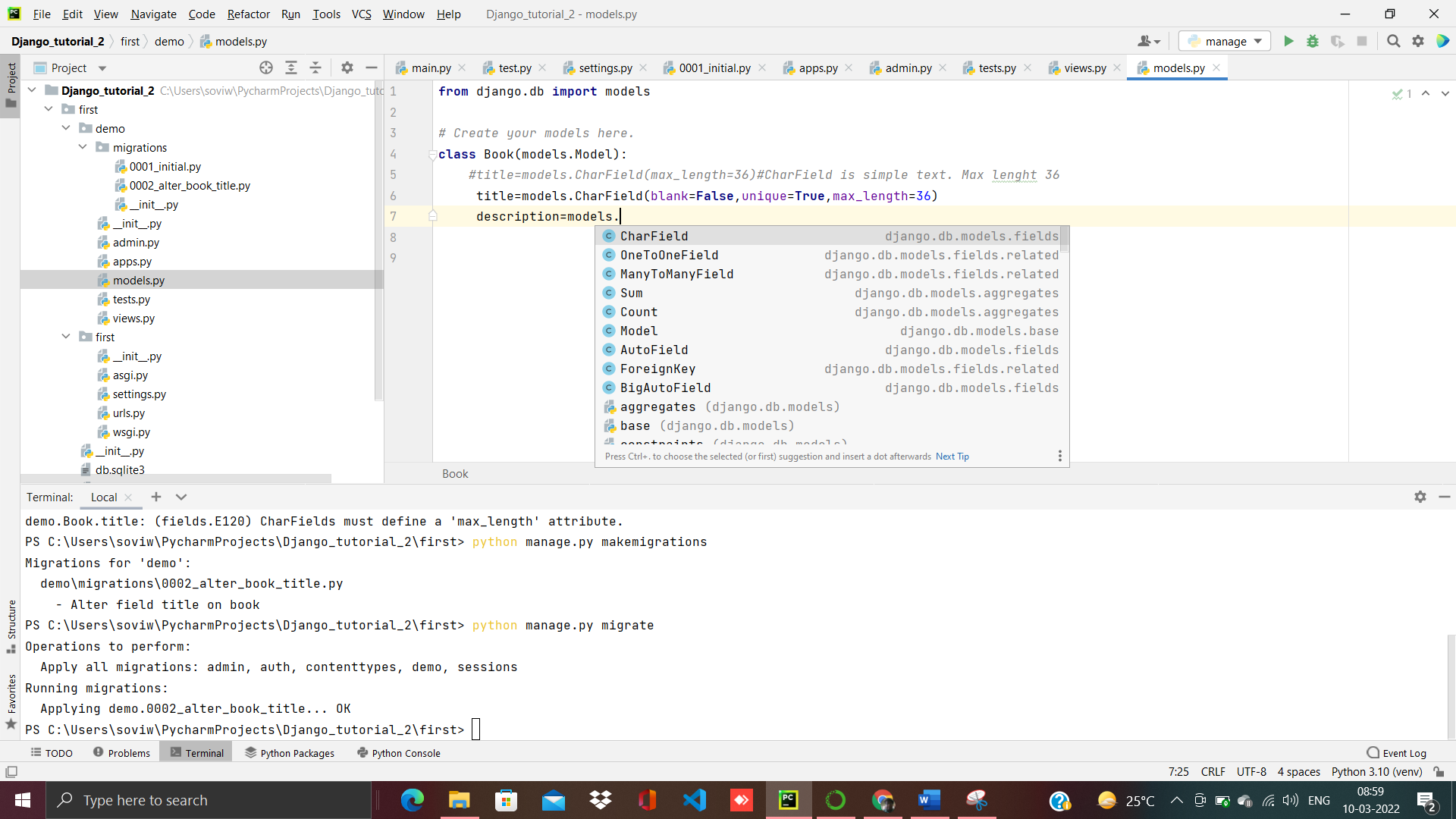
eg:

**from** django.db **import** models  
  
*# Create your models here.***class** Book(models.Model):  
 *#title=models.CharField(max\_length=36)#CharField is simple text. Max lenght 36* title=models.CharField(blank=**False**,unique=**True**,max\_length=36)



Field Type:

* + [Field types](https://docs.djangoproject.com/en/4.0/ref/models/fields/#field-types)
    - [AutoField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#autofield)
    - [BigAutoField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#bigautofield)
    - [BigIntegerField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#bigintegerfield)
    - [BinaryField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#binaryfield)
    - [BooleanField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#booleanfield)
    - [CharField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#charfield)
    - [DateField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#datefield)
    - [DateTimeField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#datetimefield)
    - [DecimalField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#decimalfield)
    - [DurationField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#durationfield)
    - [EmailField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#emailfield)
    - [FileField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#filefield)
      * [FileField and FieldFile](https://docs.djangoproject.com/en/4.0/ref/models/fields/#filefield-and-fieldfile)
    - [FilePathField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#filepathfield)
    - [FloatField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#floatfield)
    - [GenericIPAddressField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#genericipaddressfield)
    - [ImageField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#imagefield)
    - [IntegerField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#integerfield)
    - [JSONField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#jsonfield)
    - [PositiveBigIntegerField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#positivebigintegerfield)
    - [PositiveIntegerField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#positiveintegerfield)
    - [PositiveSmallIntegerField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#positivesmallintegerfield)
    - [SlugField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#slugfield)
    - [SmallAutoField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#smallautofield)
    - [SmallIntegerField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#smallintegerfield)
    - [TextField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#textfield)
    - [TimeField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#timefield)
    - [URLField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#urlfield)
    - [UUIDField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#uuidfield)
  + [Relationship fields](https://docs.djangoproject.com/en/4.0/ref/models/fields/#module-django.db.models.fields.related)
    - [ForeignKey](https://docs.djangoproject.com/en/4.0/ref/models/fields/#foreignkey)
      * [Database Representation](https://docs.djangoproject.com/en/4.0/ref/models/fields/#database-representation)
      * [Arguments](https://docs.djangoproject.com/en/4.0/ref/models/fields/#arguments)
    - [ManyToManyField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#manytomanyfield)
      * [Database Representation](https://docs.djangoproject.com/en/4.0/ref/models/fields/#id1)
      * [Arguments](https://docs.djangoproject.com/en/4.0/ref/models/fields/#manytomany-arguments)
    - [OneToOneField](https://docs.djangoproject.com/en/4.0/ref/models/fields/#onetoonefield)
  + [Field API reference](https://docs.djangoproject.com/en/4.0/ref/models/fields/#field-api-reference)
* [Field attribute reference](https://docs.djangoproject.com/en/4.0/ref/models/fields/#field-attribute-reference)
  + [Attributes for fields](https://docs.djangoproject.com/en/4.0/ref/models/fields/#attributes-for-fields)
  + [Attributes for fields with relations](https://docs.djangoproject.com/en/4.0/ref/models/fields/#attributes-for-fields-with-relations)

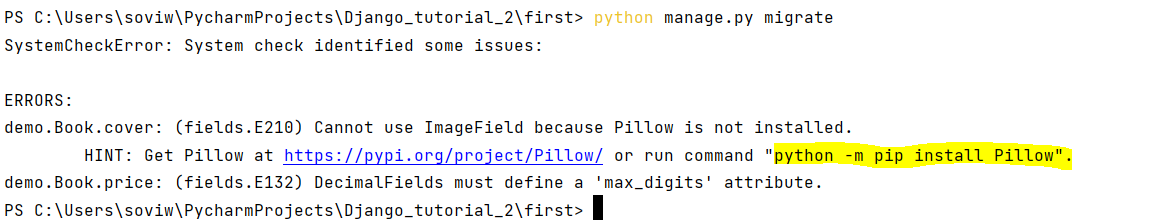


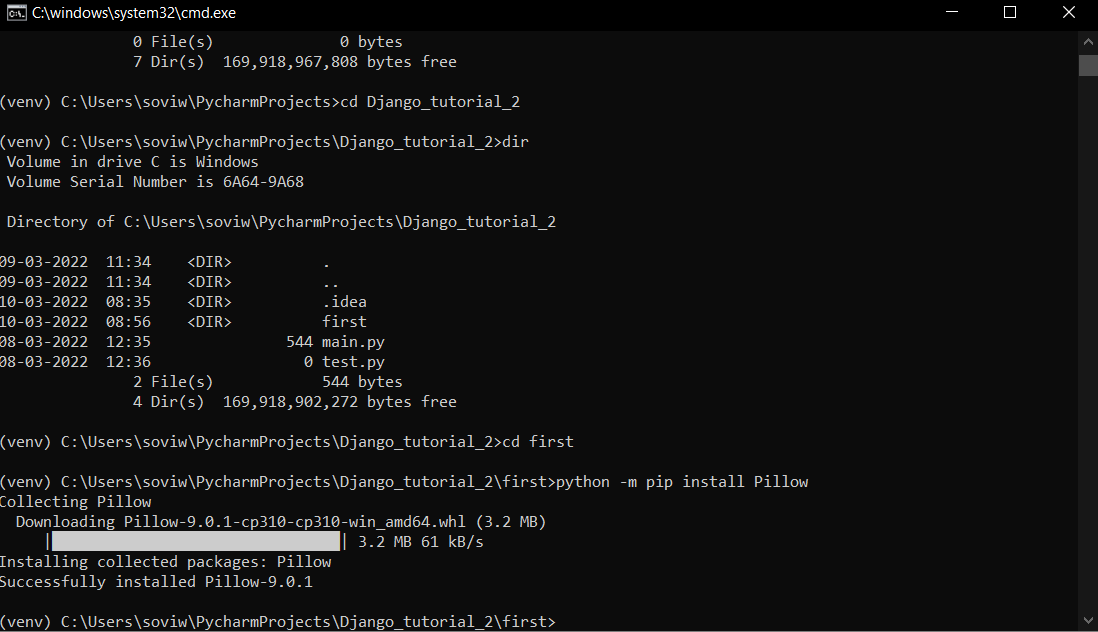
The pic shows different field types

Now we add some new Field types:

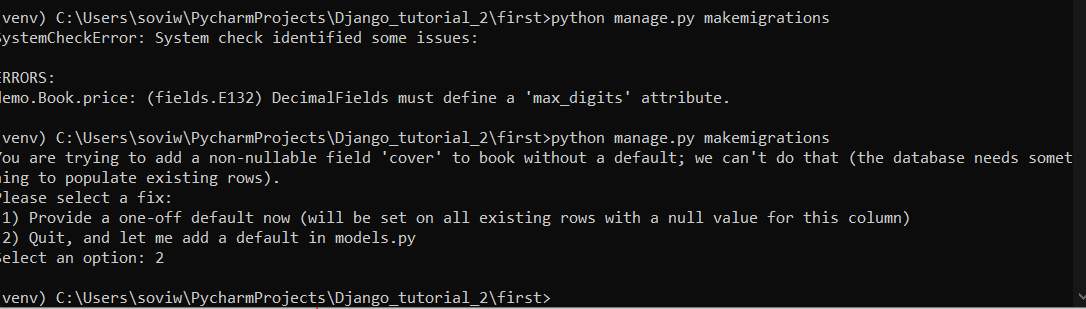


Now we get an error:





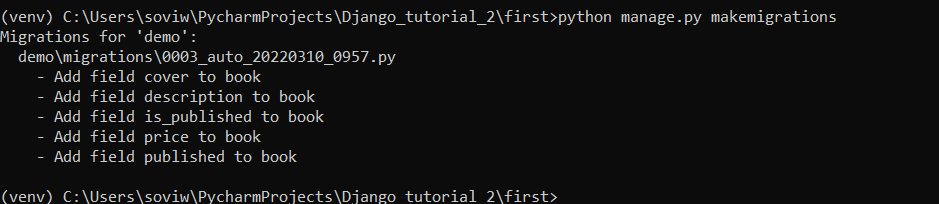
Now activate,but we get below errors:

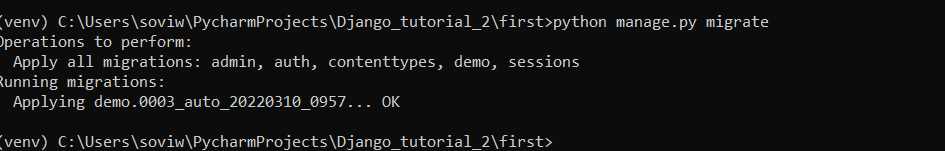


Hence we add max\_digits in Decimal Field and we add blanks in cover(same goes for date)

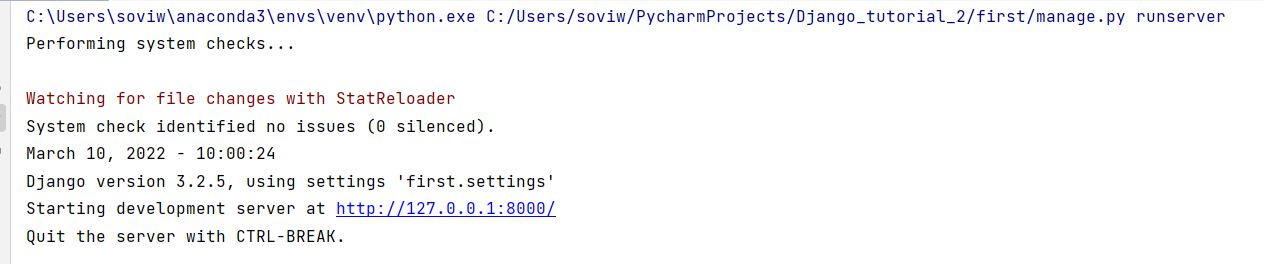


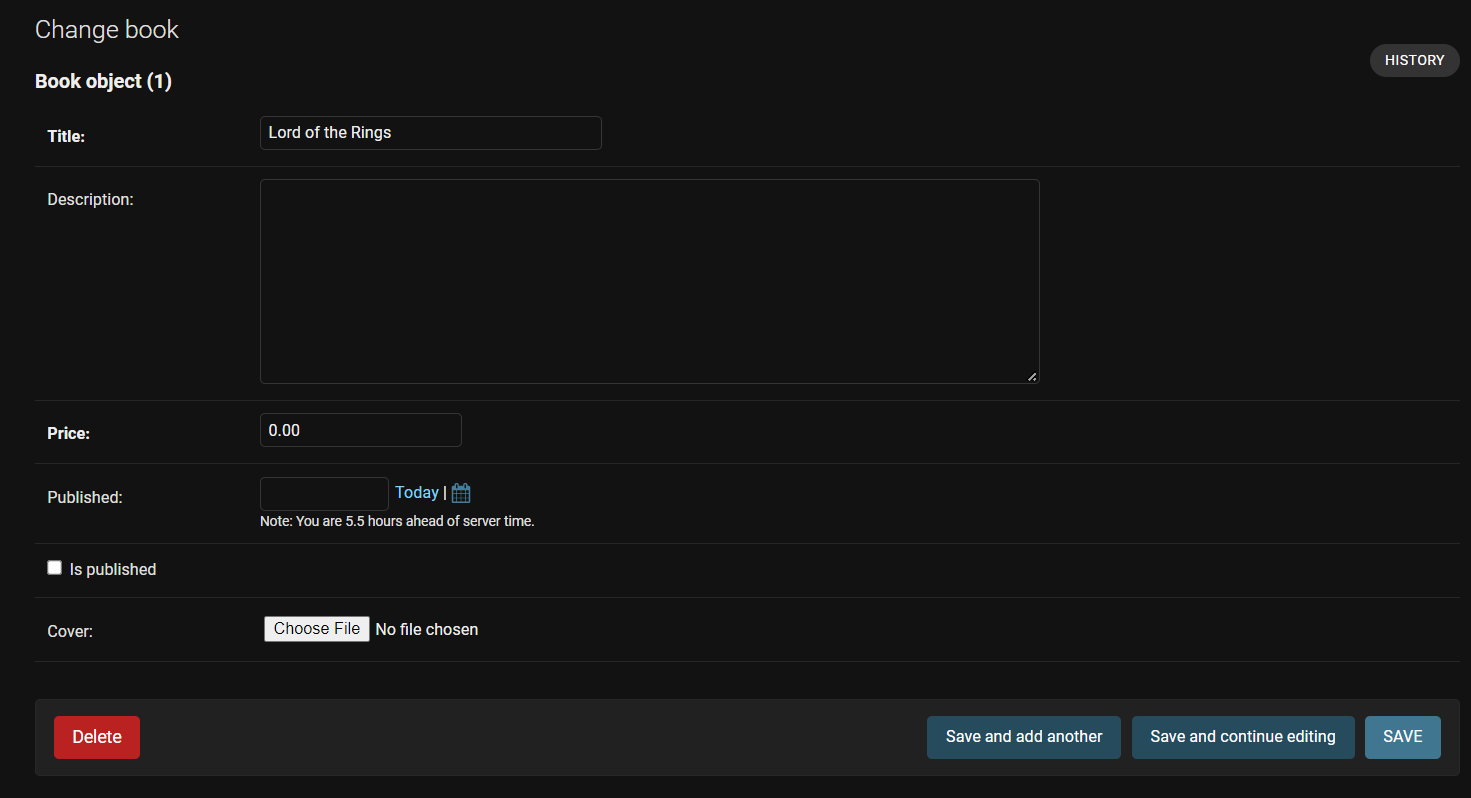
O/P:





Now we run the server

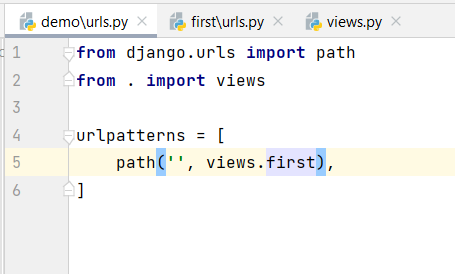




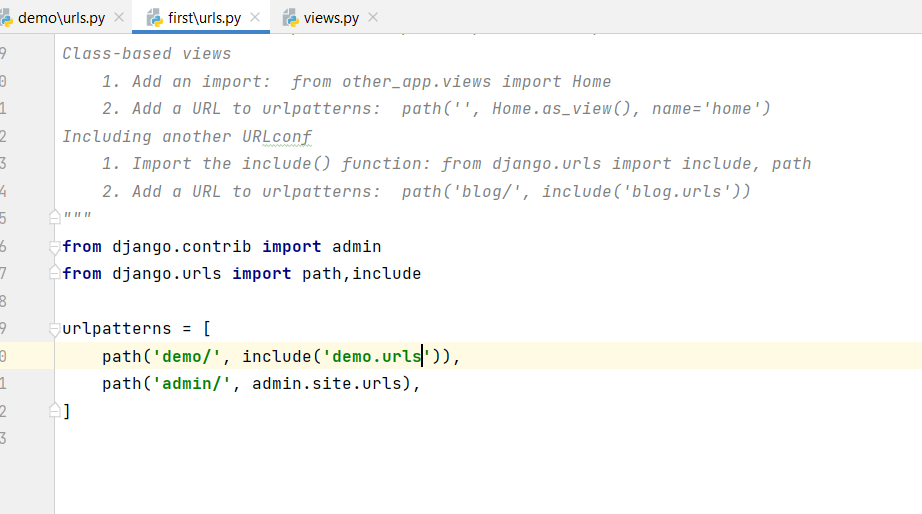
Now we study urls:

Now we create a urls.py in the demo folder.

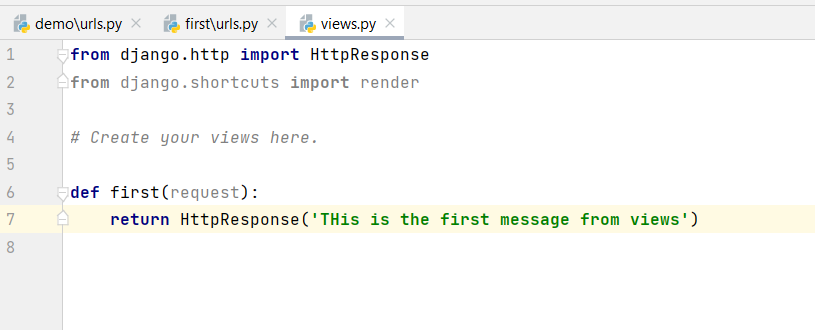
Demo/urls.py:



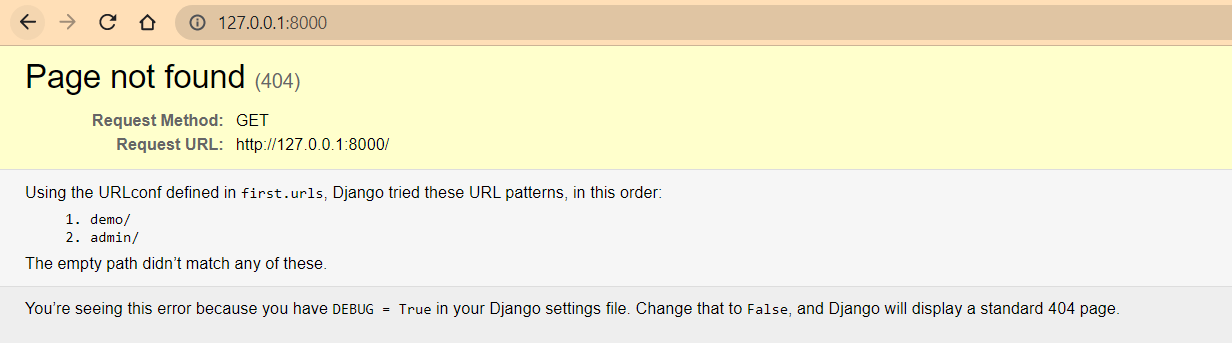
First/urls.py:

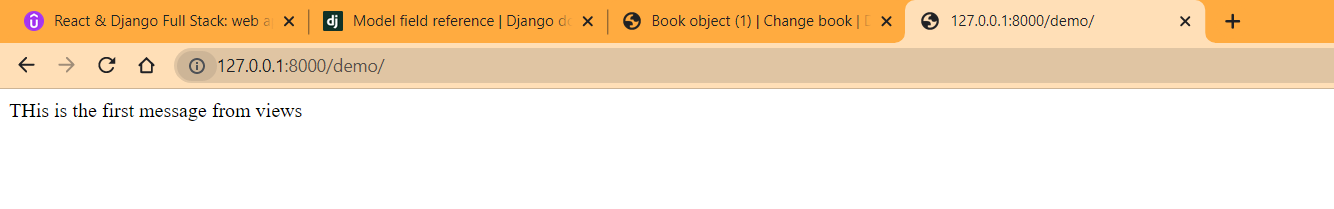


Demo/views.py:



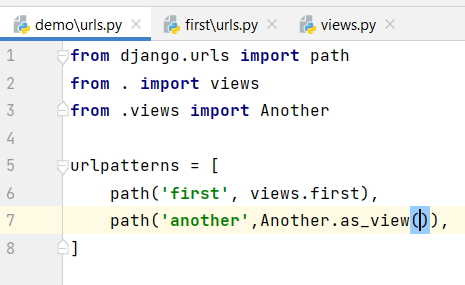
O/P:



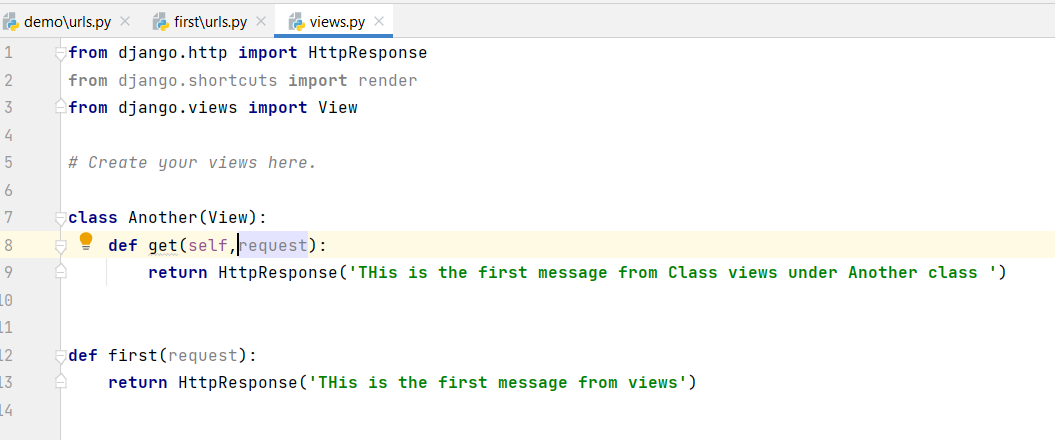


Class Views:

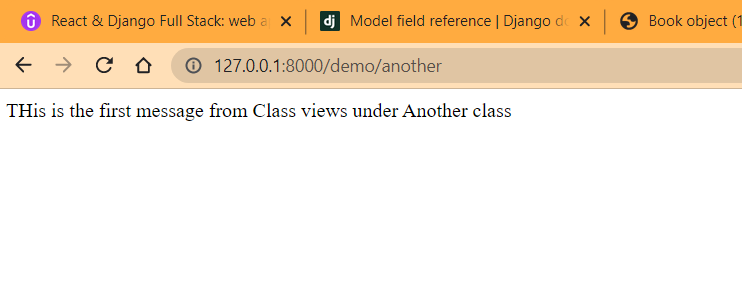
Demo/urls.py:



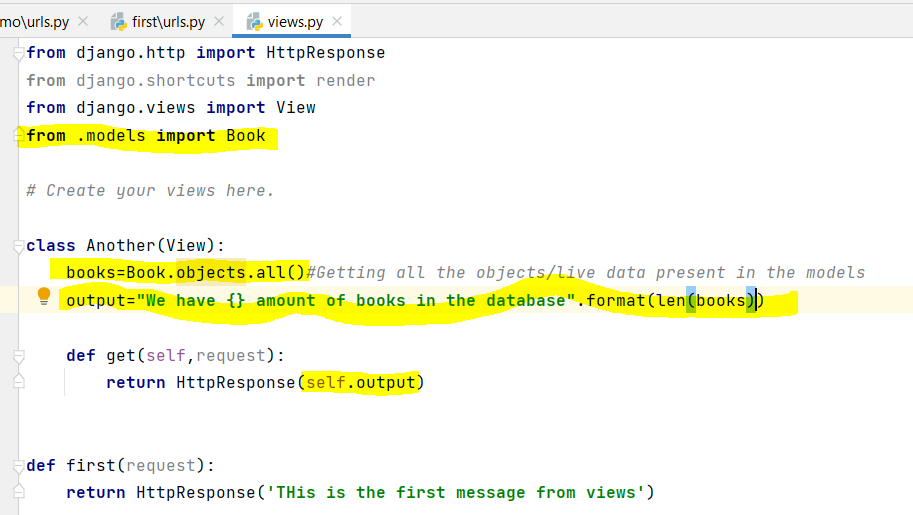
Demo/views.py:



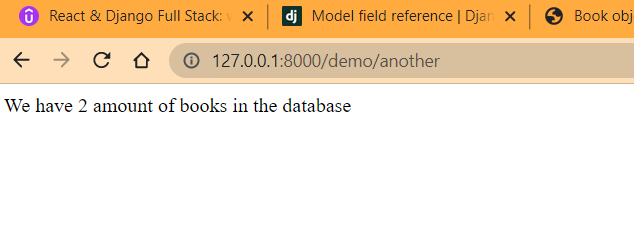
O/P:



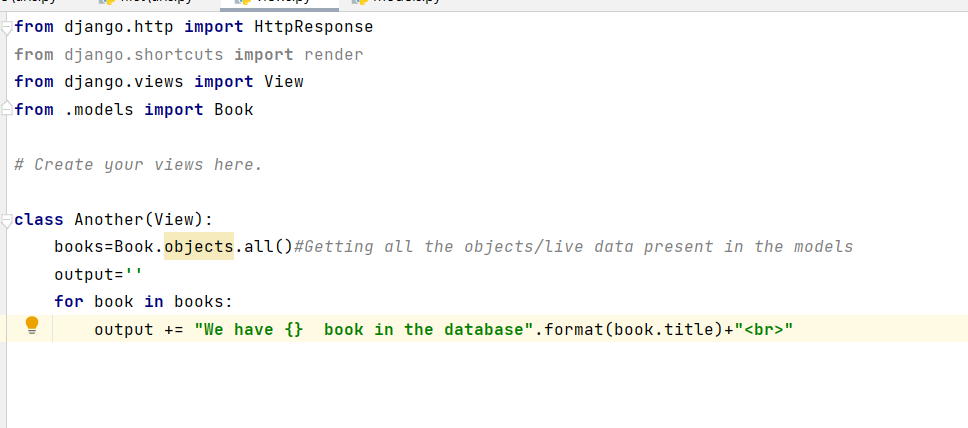
Model object views



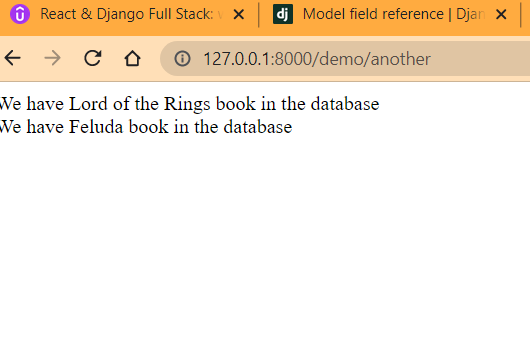
O/P:



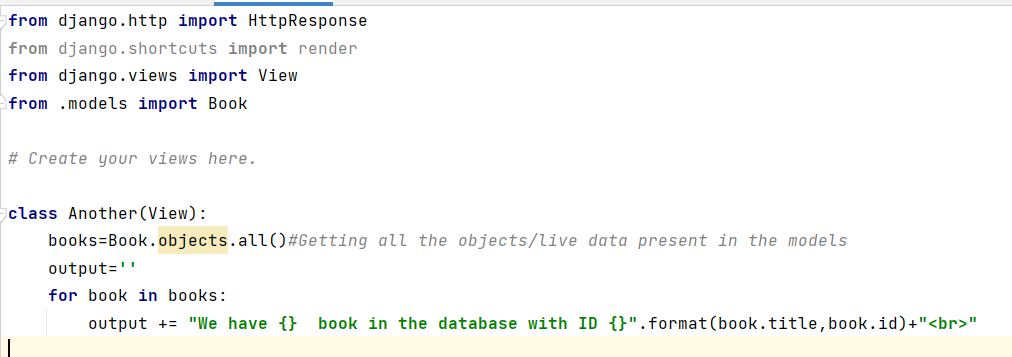
We can also check the books by title:

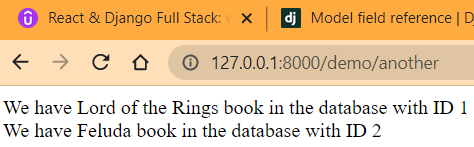


O/P:

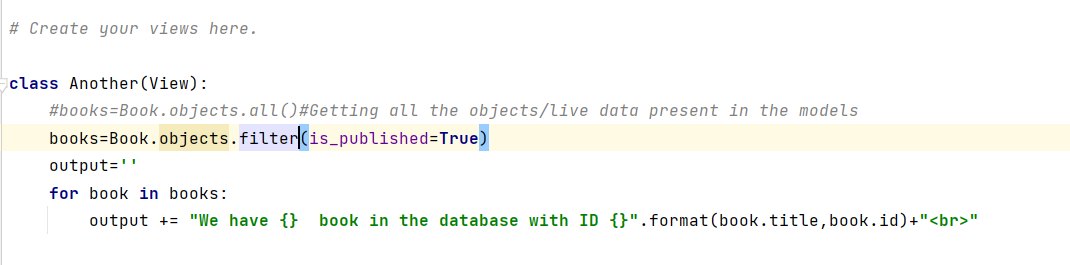


Now if we check the ID we can do so since in Django id of fields in a Model is already added.

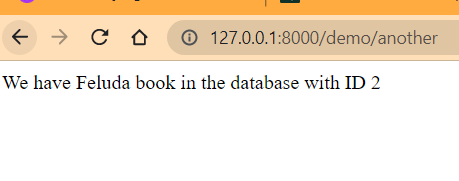


O/P:  


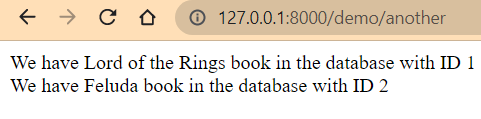
We can check for filtered projection of objects as well.



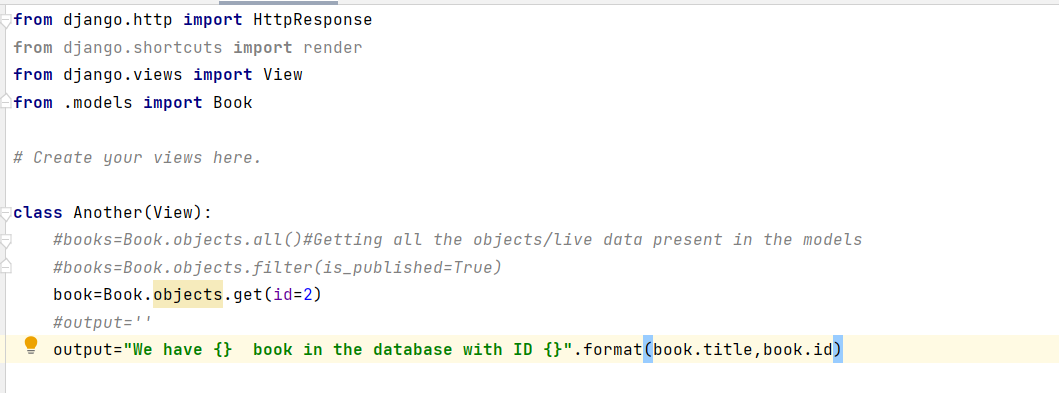
O/P:



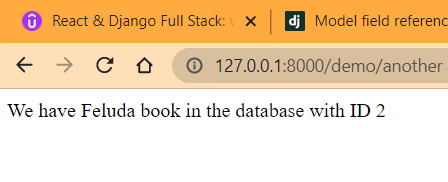
If we changed the is\_published option to True in Book 1 and restart the server:



We want to get a Tuple respective to a single field query:



O/P:

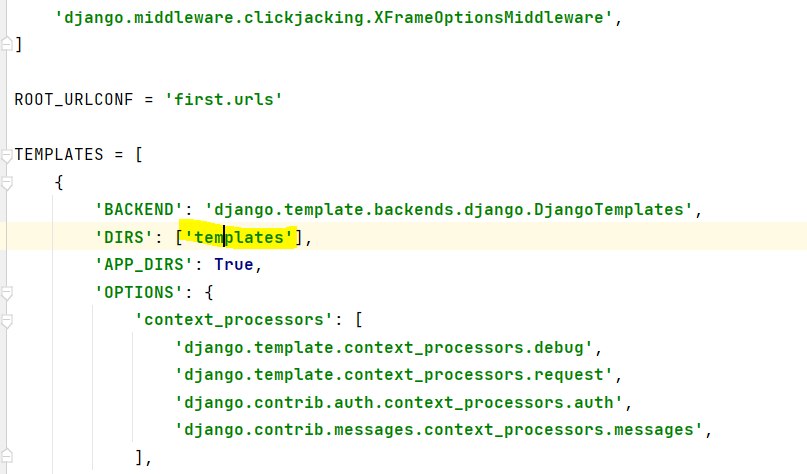


Template:

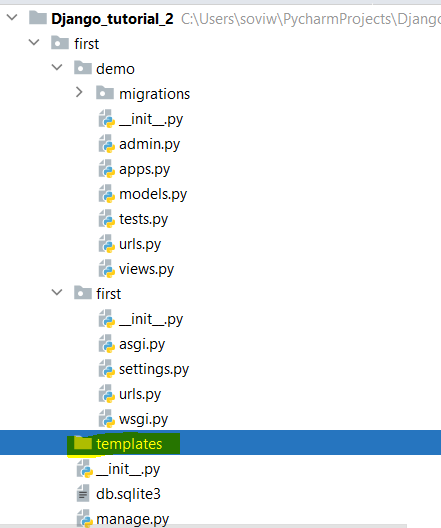
This is something we will not use in our tutorial but we will understand how we can display pages with profound ease.

So we go to the settings.py. in Templates List we add :

The directory ”DIRS”.

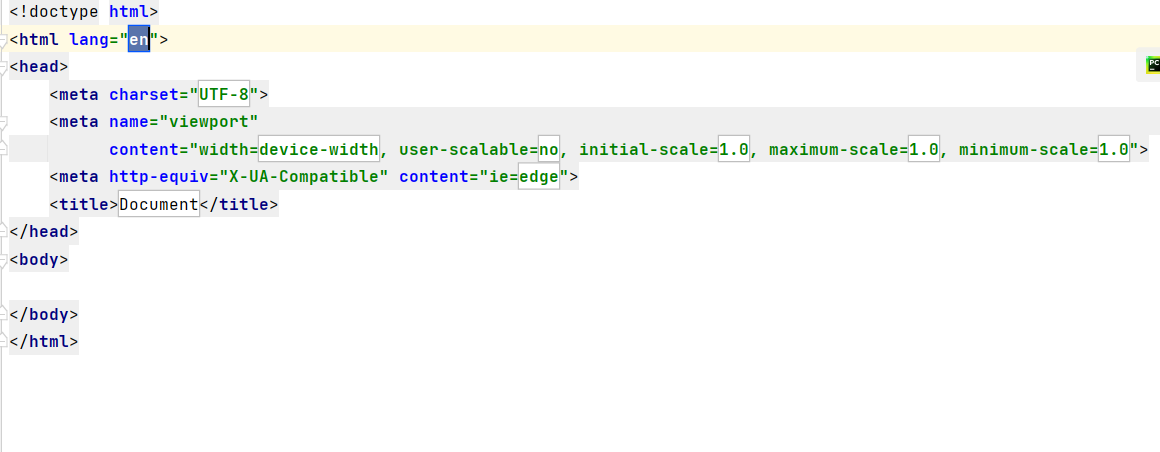


We create the new template directory in the first directory of the project



Create a file fisrt\_temp.html

Inside the file we write html:5 -> Tab. We get the below html template by default



Now we change accordingly:

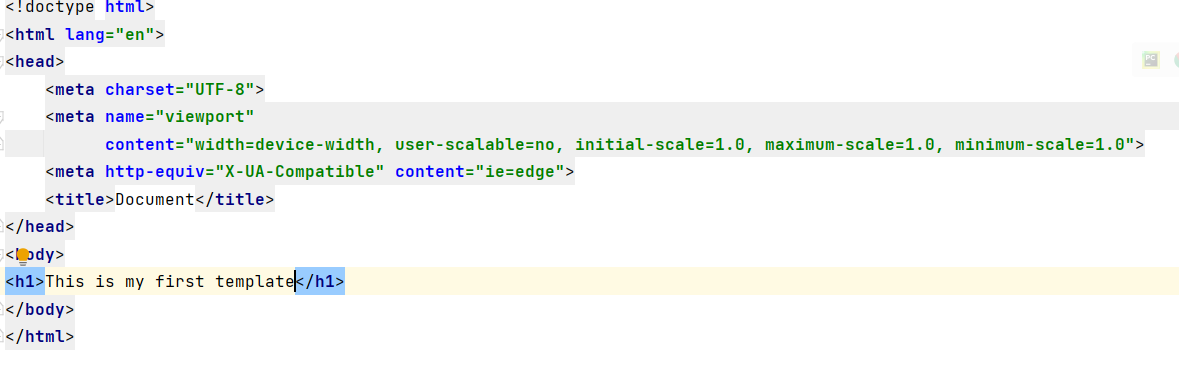
Demo/urls.py:



Views.py:

**from** django.http **import** HttpResponse  
**from** django.shortcuts **import** render  
**from** django.views **import** View  
**from** .models **import** Book  
  
*# Create your views here.  
  
# class Another(View):  
# #books=Book.objects.all()#Getting all the objects/live data present in the models  
# #books=Book.objects.filter(is\_published=True)  
# book=Book.objects.get(id=2)  
# #output=''  
# output="We have {} book in the database with ID {}".format(book.title,book.id)  
#  
# # for book in books:  
# # output += "We have {} book in the database with ID {}".format(book.title,book.id)+"<br>"  
#  
#  
#  
# def get(self,request):  
# return HttpResponse(self.output)***def** first(request):  
 *#return HttpResponse('THis is the first message from views')* **return** render(request,**'first\_temp.html'**)

first\_temp.html:



O/P:



Dynamic template:

Passing dynamic data to the template

Views.py:

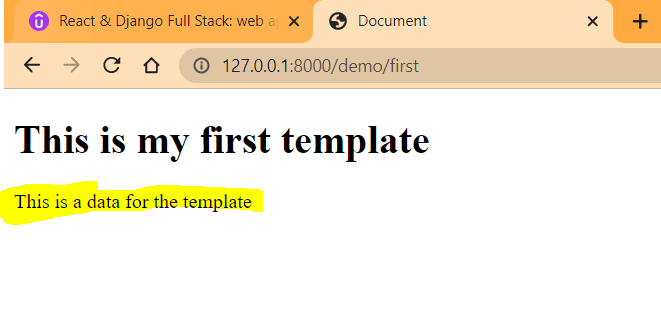
**from** django.http **import** HttpResponse  
**from** django.shortcuts **import** render  
**from** django.views **import** View  
**from** .models **import** Book  
  
*# Create your views here.  
  
# class Another(View):  
# #books=Book.objects.all()#Getting all the objects/live data present in the models  
# #books=Book.objects.filter(is\_published=True)  
# book=Book.objects.get(id=2)  
# #output=''  
# output="We have {} book in the database with ID {}".format(book.title,book.id)  
#  
# # for book in books:  
# # output += "We have {} book in the database with ID {}".format(book.title,book.id)+"<br>"  
#  
#  
#  
# def get(self,request):  
# return HttpResponse(self.output)***def** first(request):  
 *#return HttpResponse('THis is the first message from views')* **return** render(request,**'first\_temp.html'**,{**'data'**:**'This is a data for the template'**})

added the data to be passed in render

First\_temp.html:



O/P:



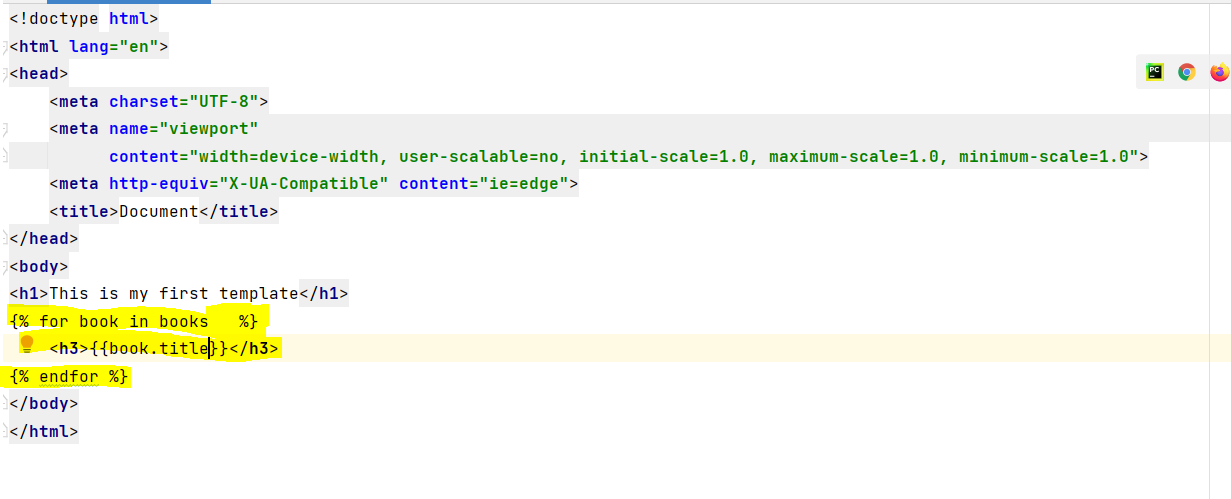
We are now passing dynamic data from view.

Now lets check if we can take our books to the html page:

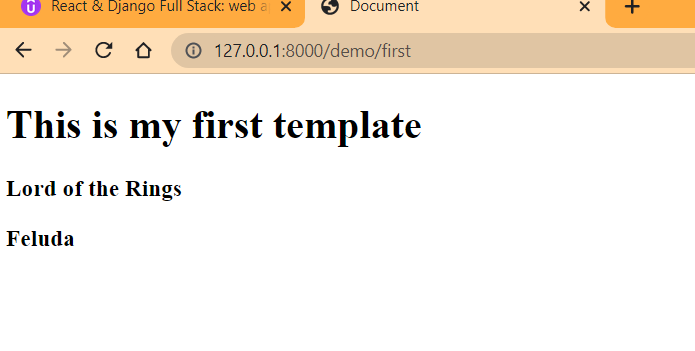
In views.py:

**def** first(request):  
 *#return HttpResponse('THis is the first message from views')* books=Book.objects.all()  
 **return** render(request,**'first\_temp.html'**,{**'books'**:books})

in first\_temp.html:



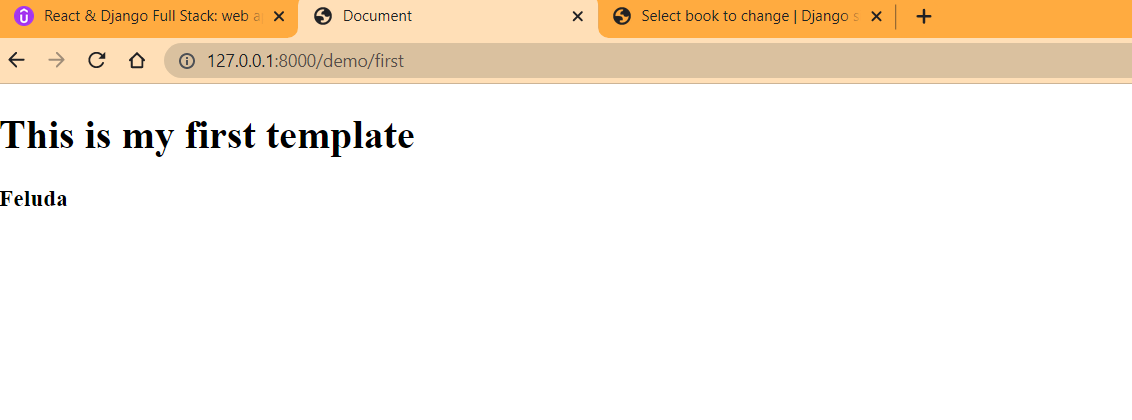
O/P:



Now we add a little complexity checking if the books are published:

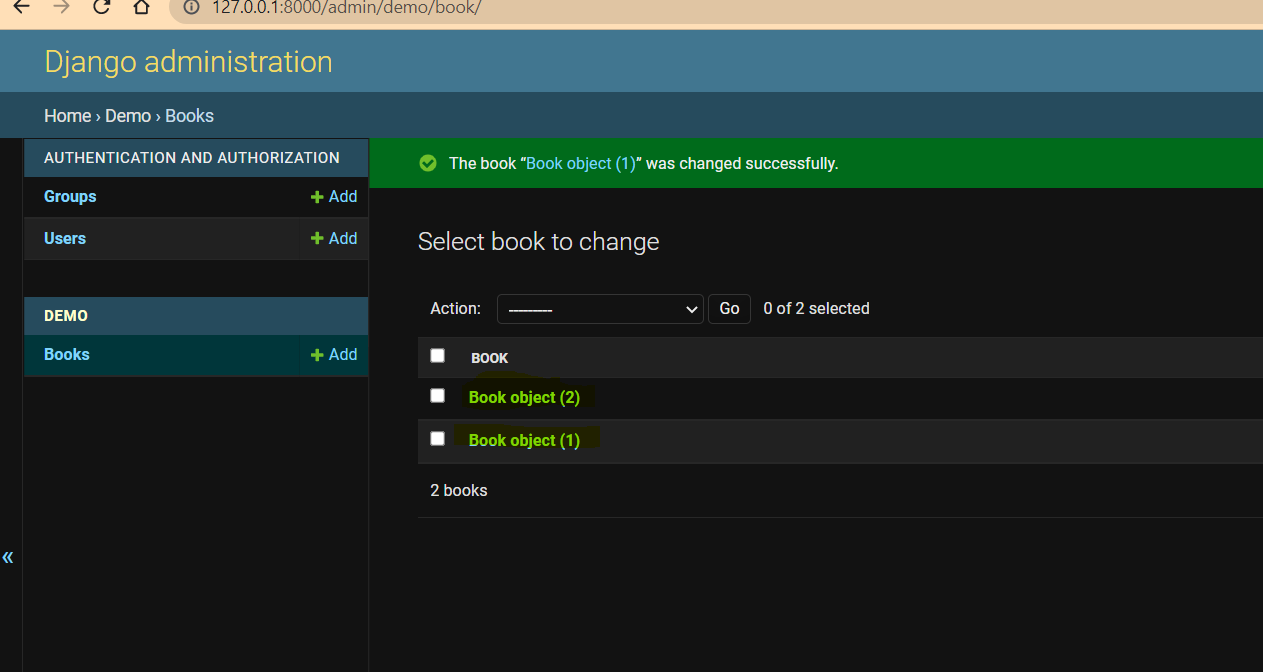


O/P:



Admin Customization:

Currently we have the Book objects as below which is not human readable:

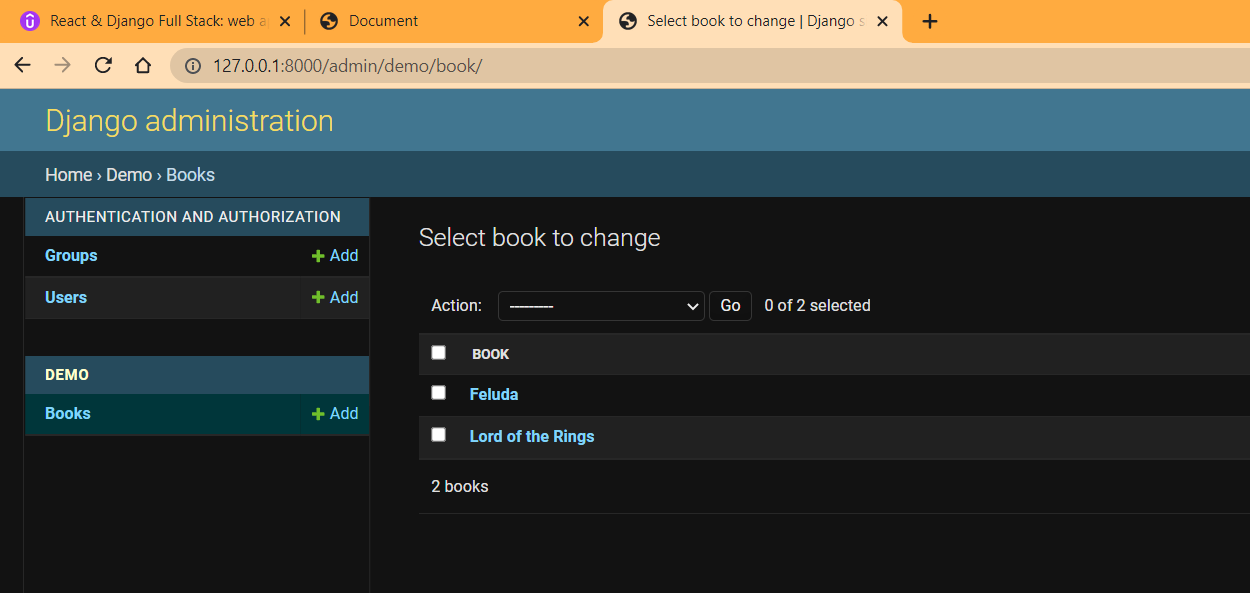


We can do customization:

We use a dunder \_\_str\_\_(): for this

In models.py:

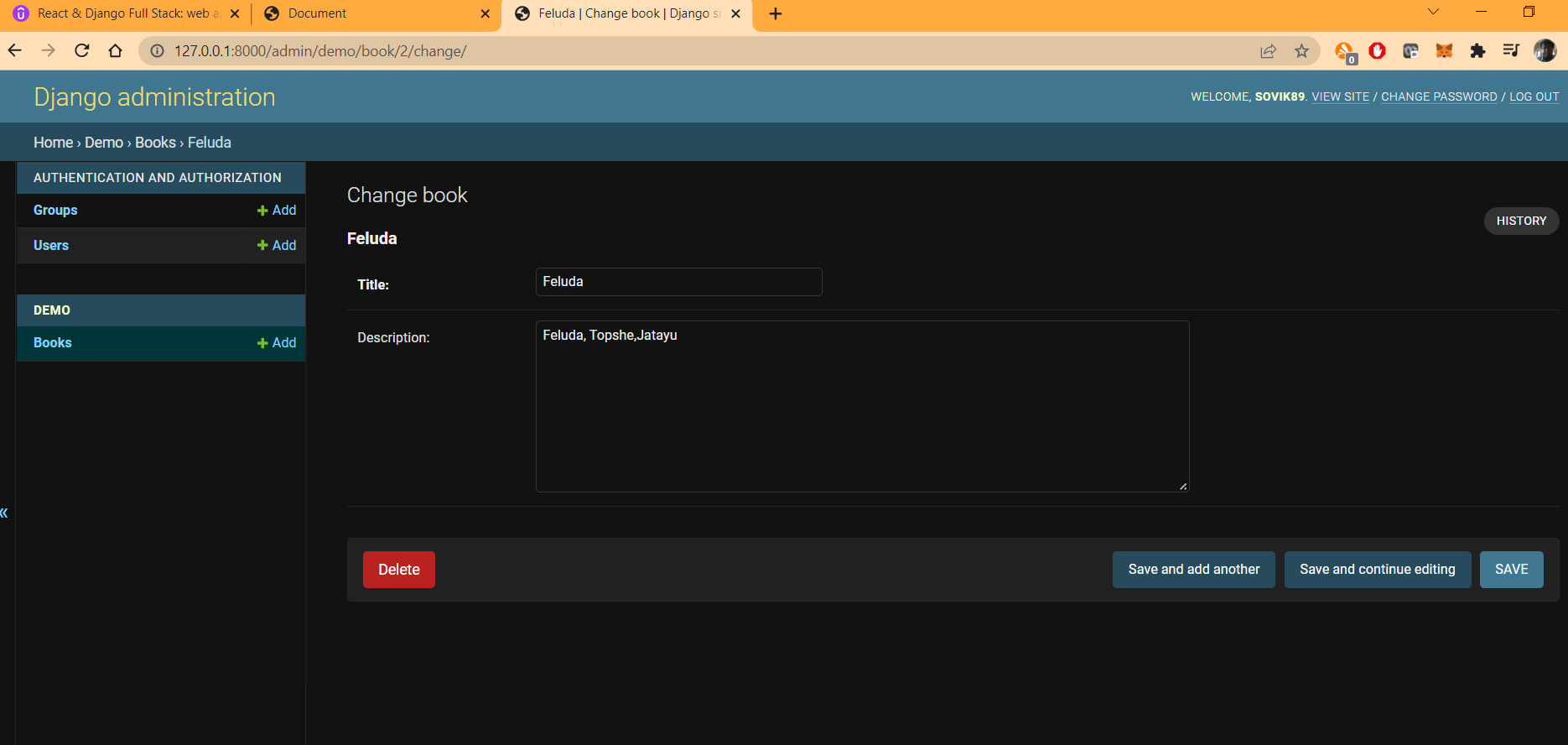


O/P:  
  


Admin.py:

**from** django.contrib **import** admin  
**from** .models **import** Book  
  
  
*# Register your models here.  
  
#admin.site.register(Book)  
  
  
#Below process actually gives more control over the class***@admin.register(Book)  
class BookAdmin(admin.ModelAdmin):  
 fields=['title','description']**

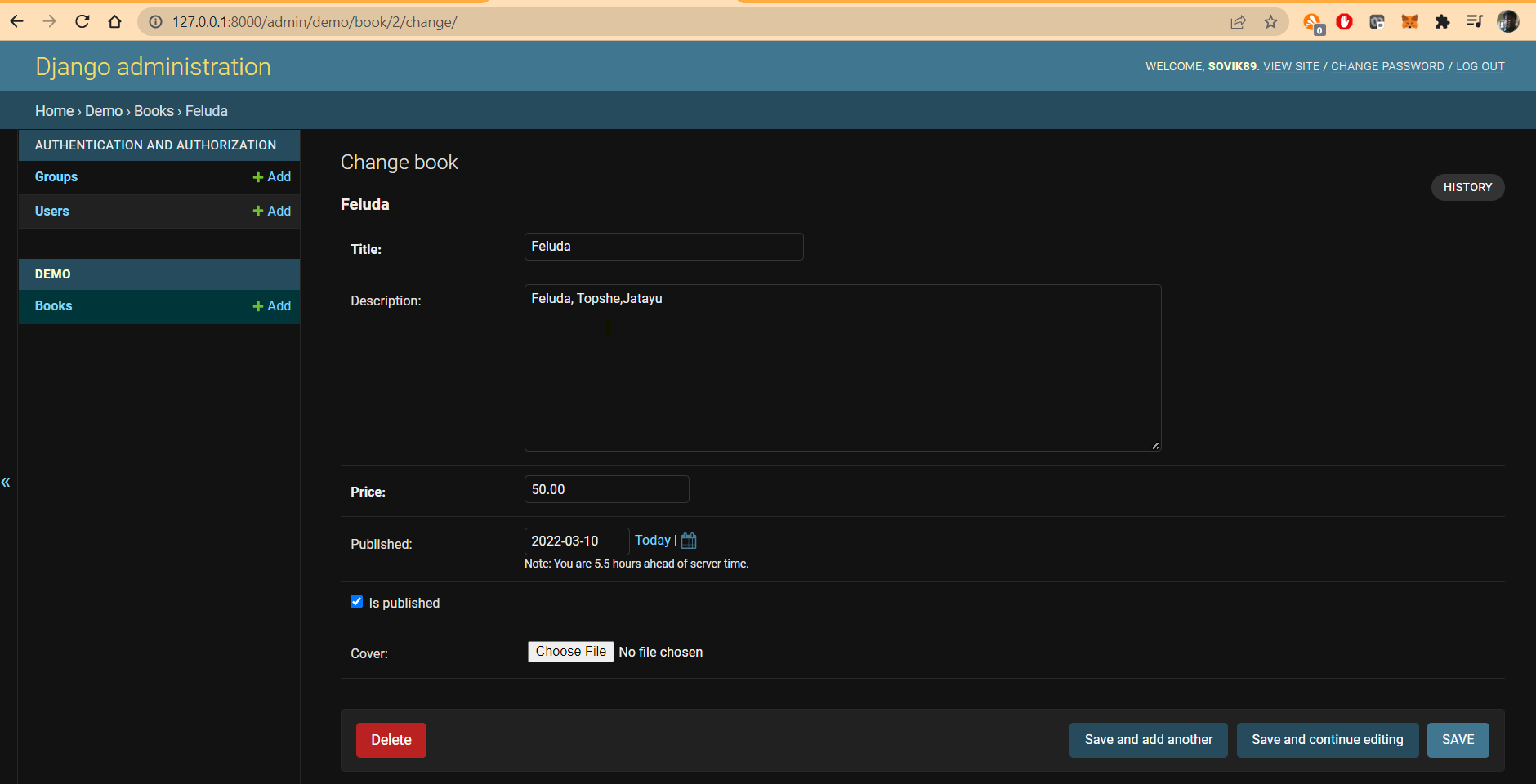
O/P:



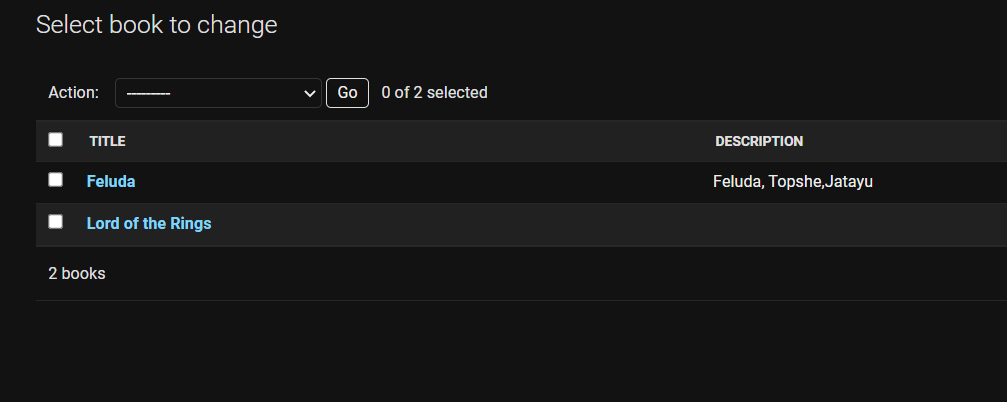
If we use built in variable list\_display. We get:



O/P:



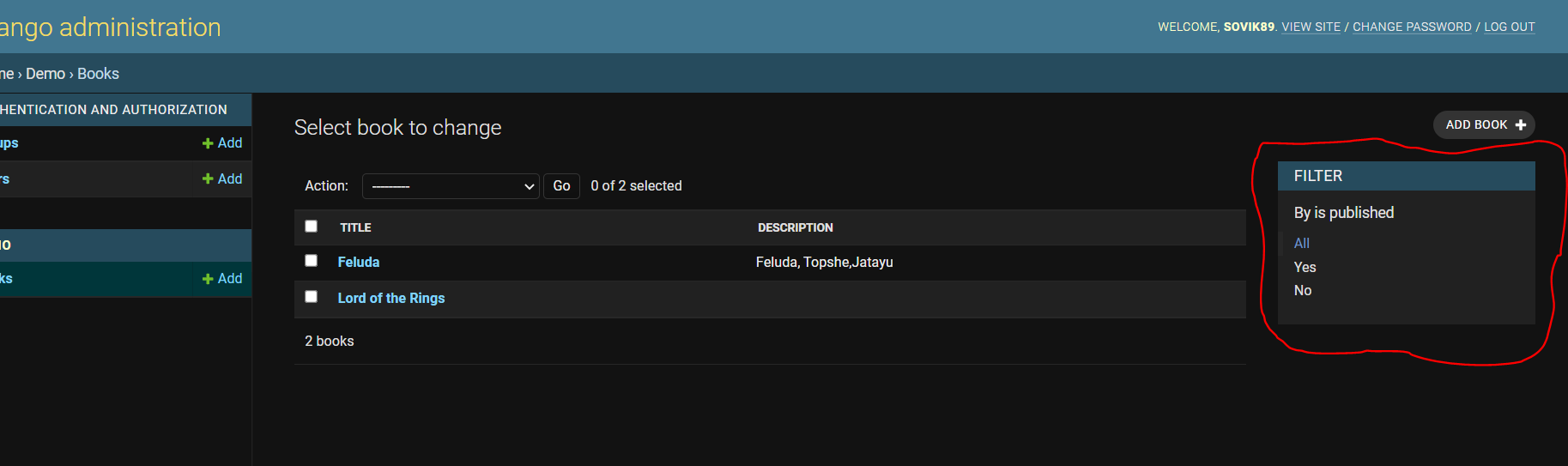
But we get description added in the Book list which was not present earlier:



We add list\_filter:

list\_filter = [**'is\_published'**]

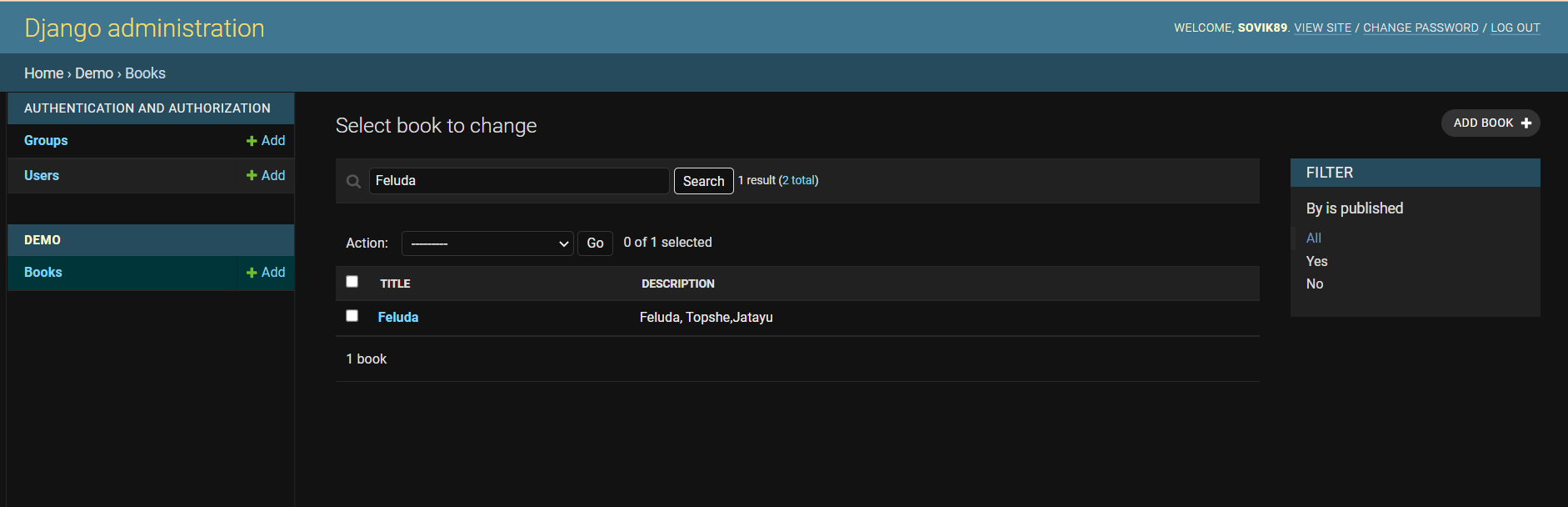
O/P:



Search by title:

search\_fields = [**'title'**]

O/P:



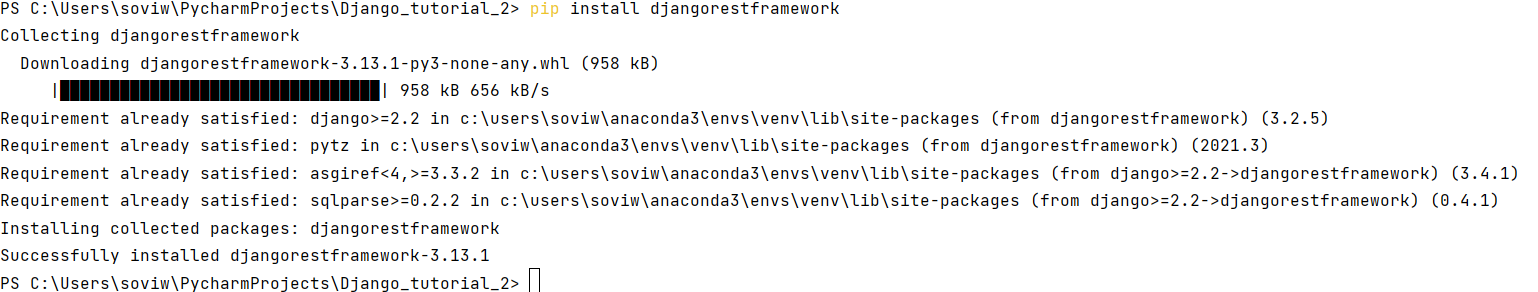
REST:

<https://www.django-rest-framework.org/>

Representational State Transfer:

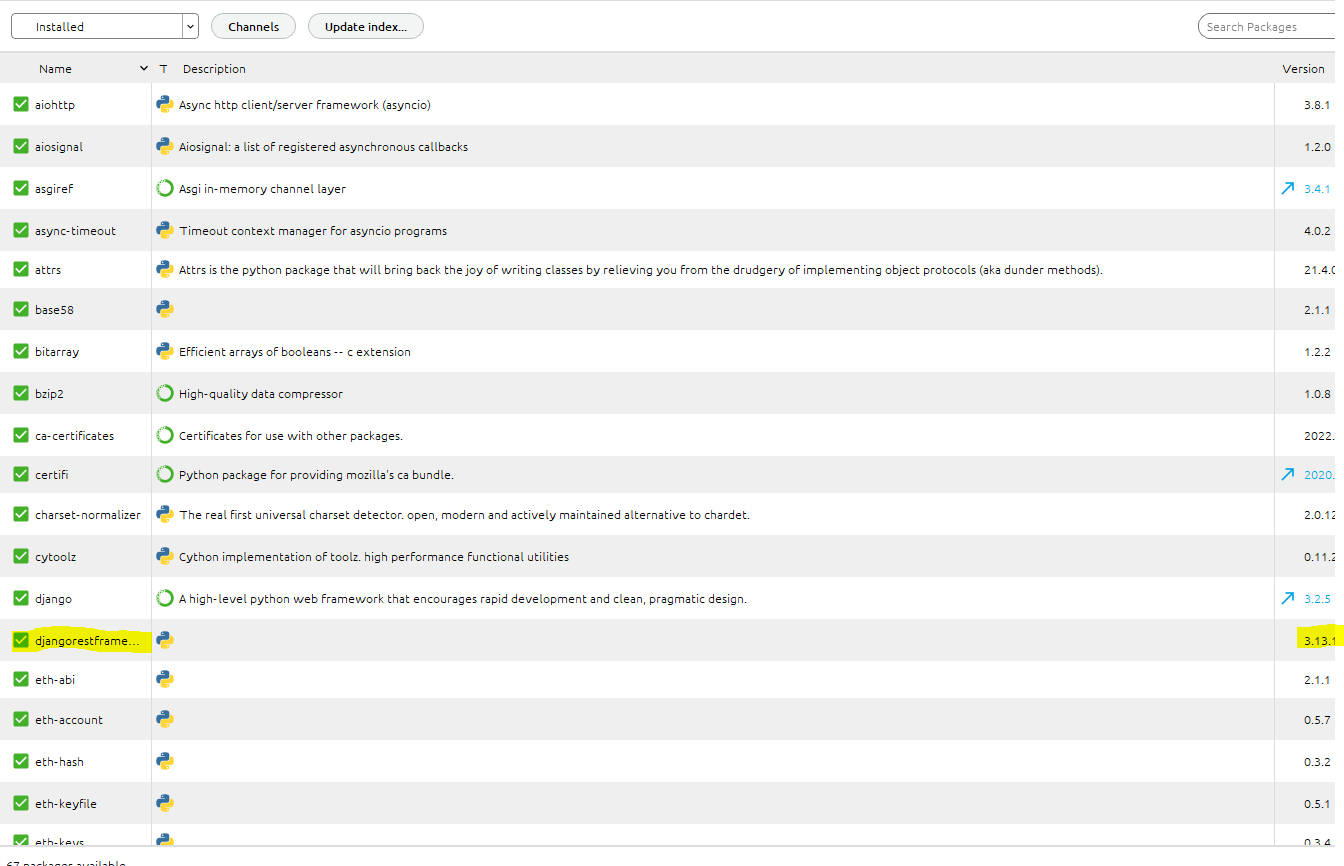
So first we have to install djangorestframework:

We install Django restframework:



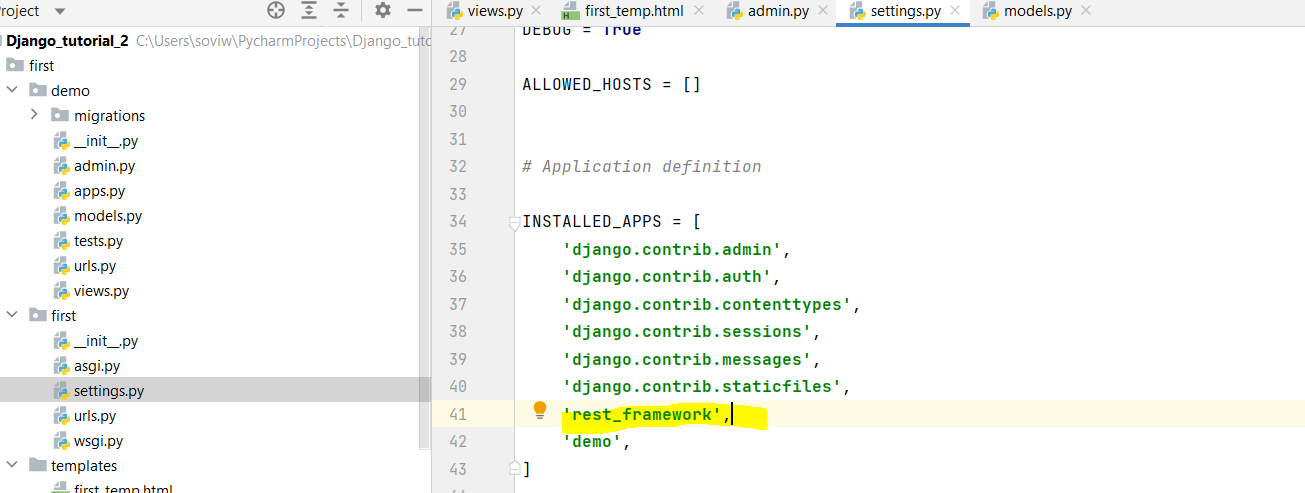
And activate:

Check from conda:

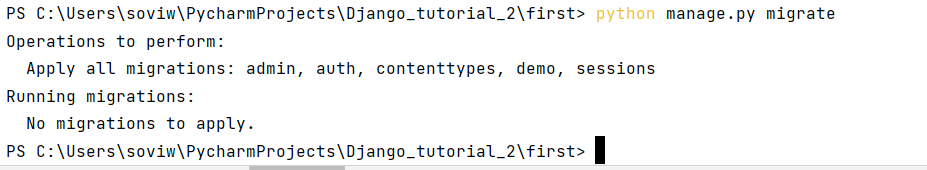


Now we add it into our apps:

In settings.py:



Just to make sure migrate once:



We serve our data with serializers:

We add a new file in the demo folder “serializers.py”:

**from** rest\_framework **import** serializers  
**from** .models **import** Book  
  
**class** BookSerializer(serializers.ModelSerializer):  
 **class** Meta:  
 model=Book  
 fields=[**'title'**,**'description'**]

Flow:

Model->Serializers🡪View🡪Urls

View.py:

*# from django.http import HttpResponse***from** django.shortcuts **import** render  
*# from django.views import View***from** .models **import** Book  
**from** rest\_framework **import** viewsets  
**from** .serializers **import** BookSerializer  
  
*# Create your views here.  
  
# class Another(View):  
# #books=Book.objects.all()#Getting all the objects/live data present in the models  
# #books=Book.objects.filter(is\_published=True)  
# book=Book.objects.get(id=2)  
# #output=''  
# output="We have {} book in the database with ID {}".format(book.title,book.id)  
#  
# # for book in books:  
# # output += "We have {} book in the database with ID {}".format(book.title,book.id)+"<br>"  
#  
#  
#  
# def get(self,request):  
# return HttpResponse(self.output)***def** first(request):  
*# #return HttpResponse('THis is the first message from views')* books=Book.objects.all()  
 **return** render(request,**'first\_temp.html'**,{**'books'**:books})  
  
  
**class** BookViewSet(viewsets.ModelViewSet):  
 serializer\_class = BookSerializer  
 queryset = Book.objects.all()

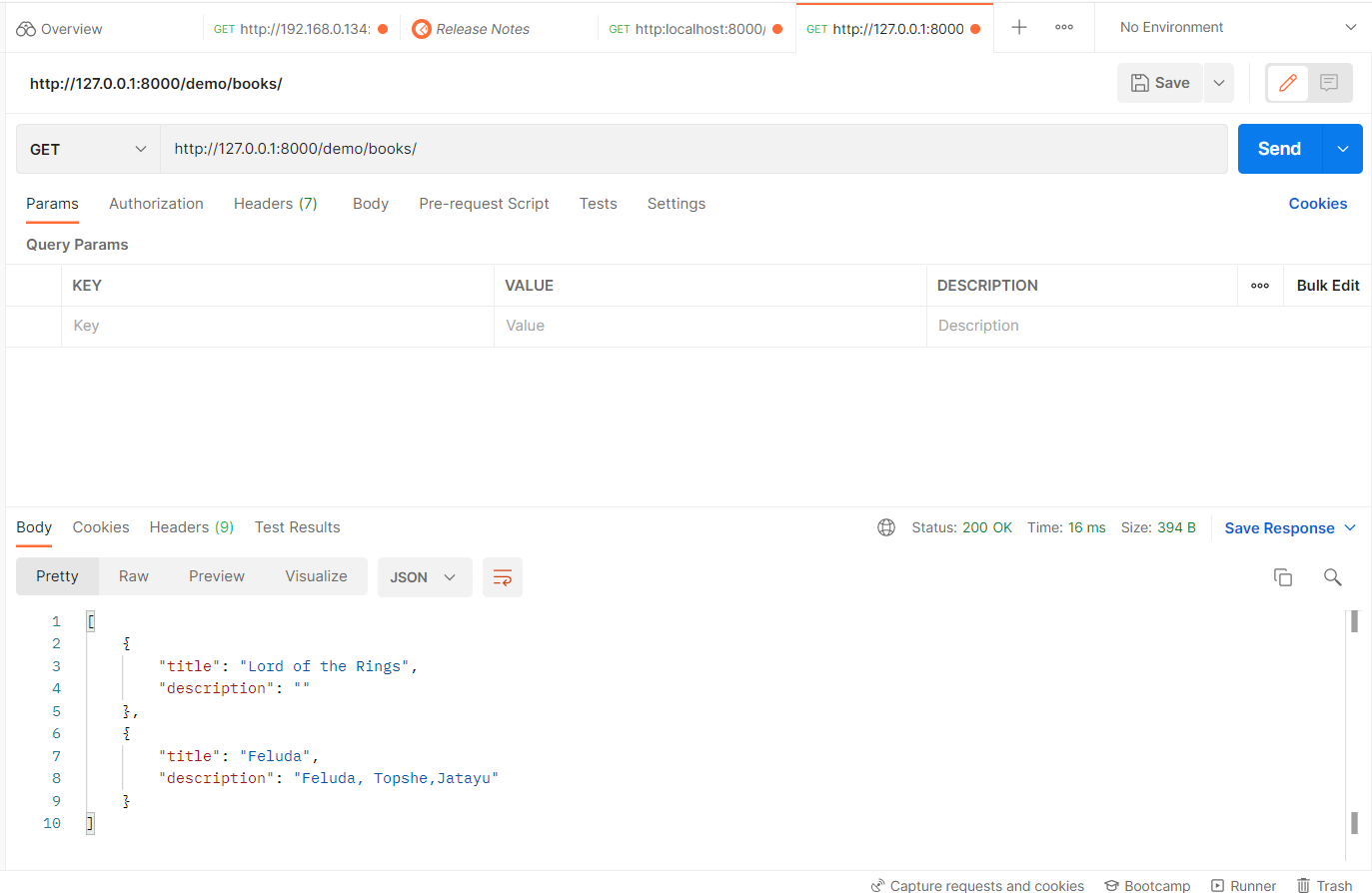
Urls.py:

**from** django.urls **import** path,include  
**from** . **import** views  
*#from .views import Another***from** rest\_framework **import** routers  
**from** .views **import** BookViewSet  
  
router=routers.DefaultRouter()  
router.register(**'books'**,BookViewSet)  
  
urlpatterns = [  
 path(**'first'**, views.first),  
 *#path('another',Another.as\_view()),* path(**''**,include(router.urls))  
  
]

O/P:

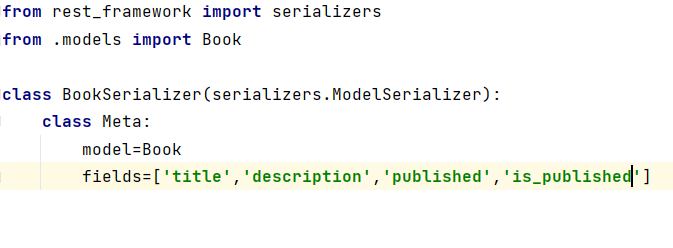


Now we use Postman for Api:

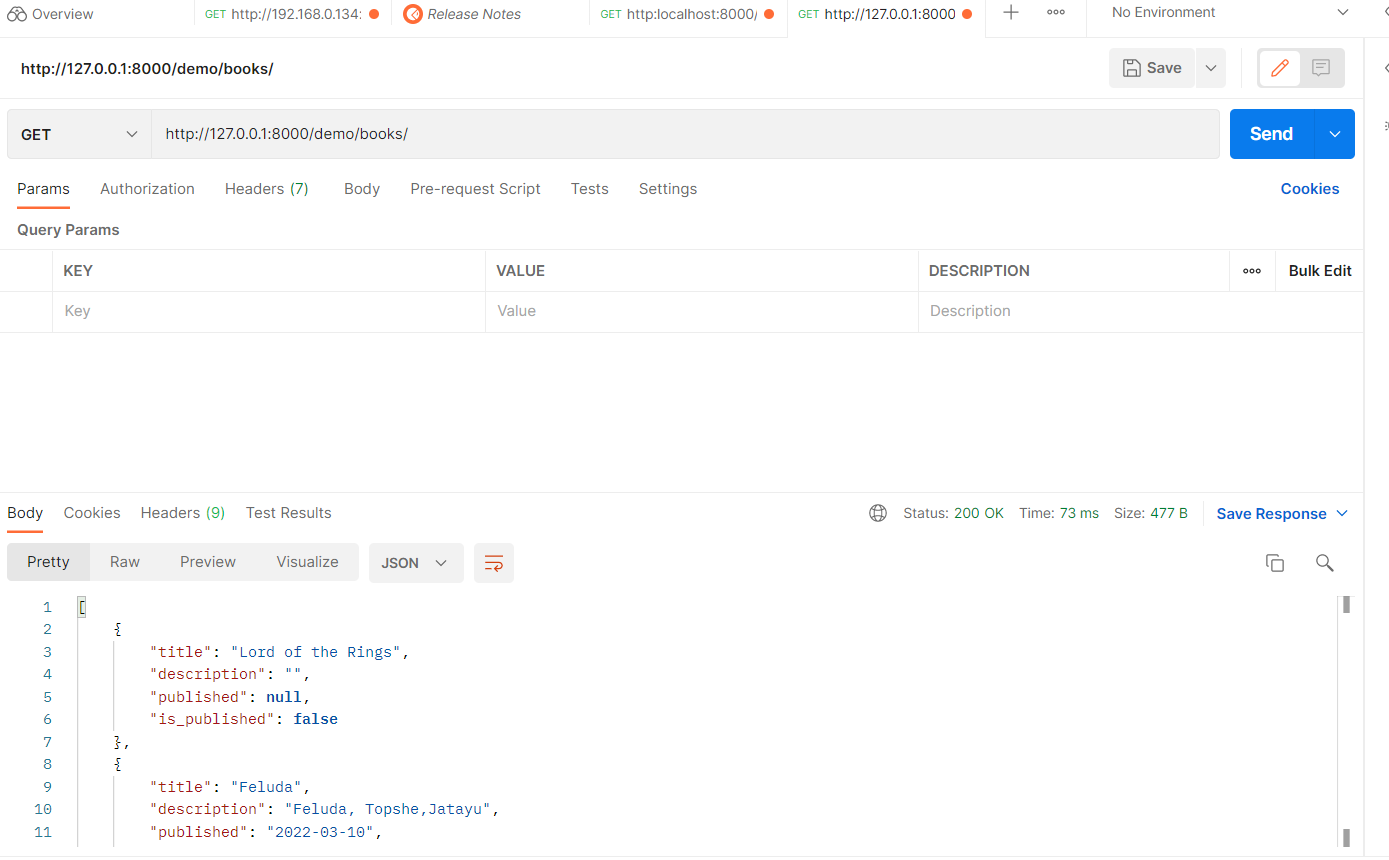


Now if we add published and is\_published in the serializers.py:

Serializers.py:

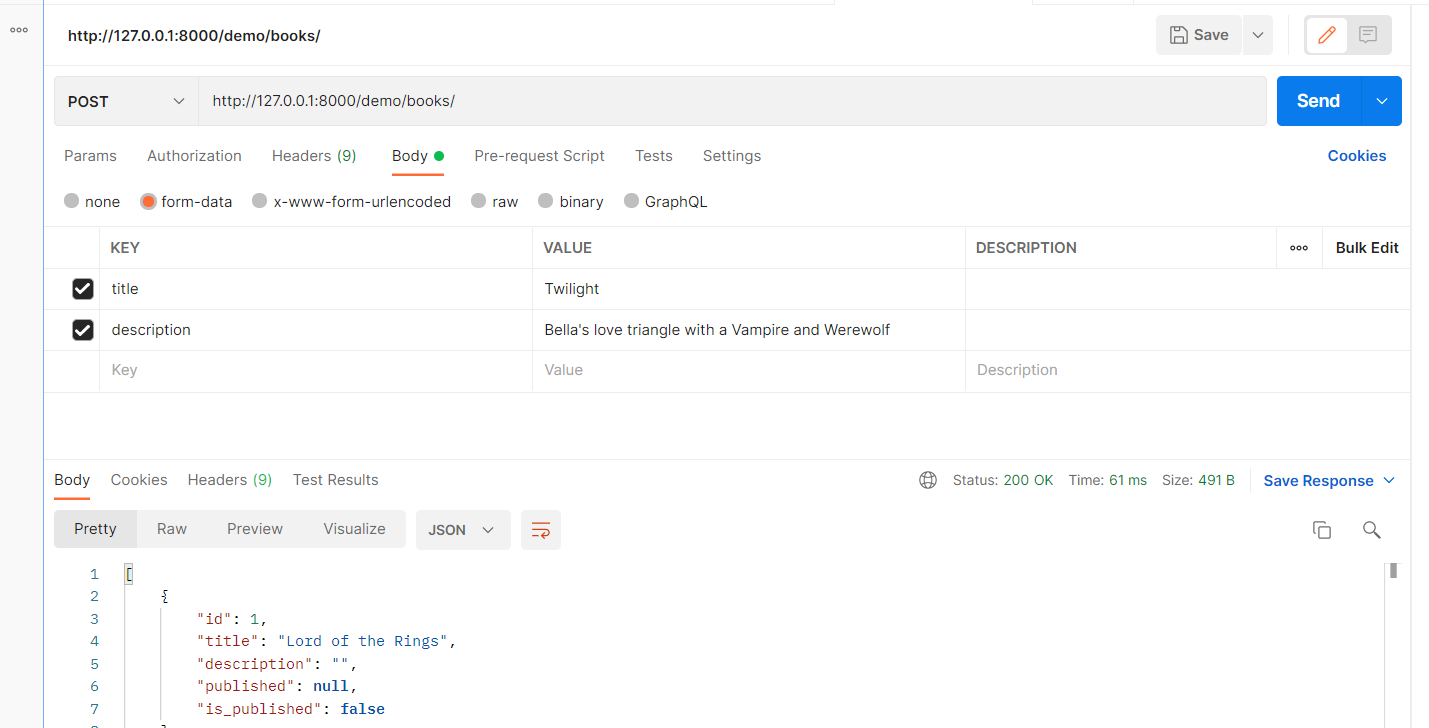


O/P:

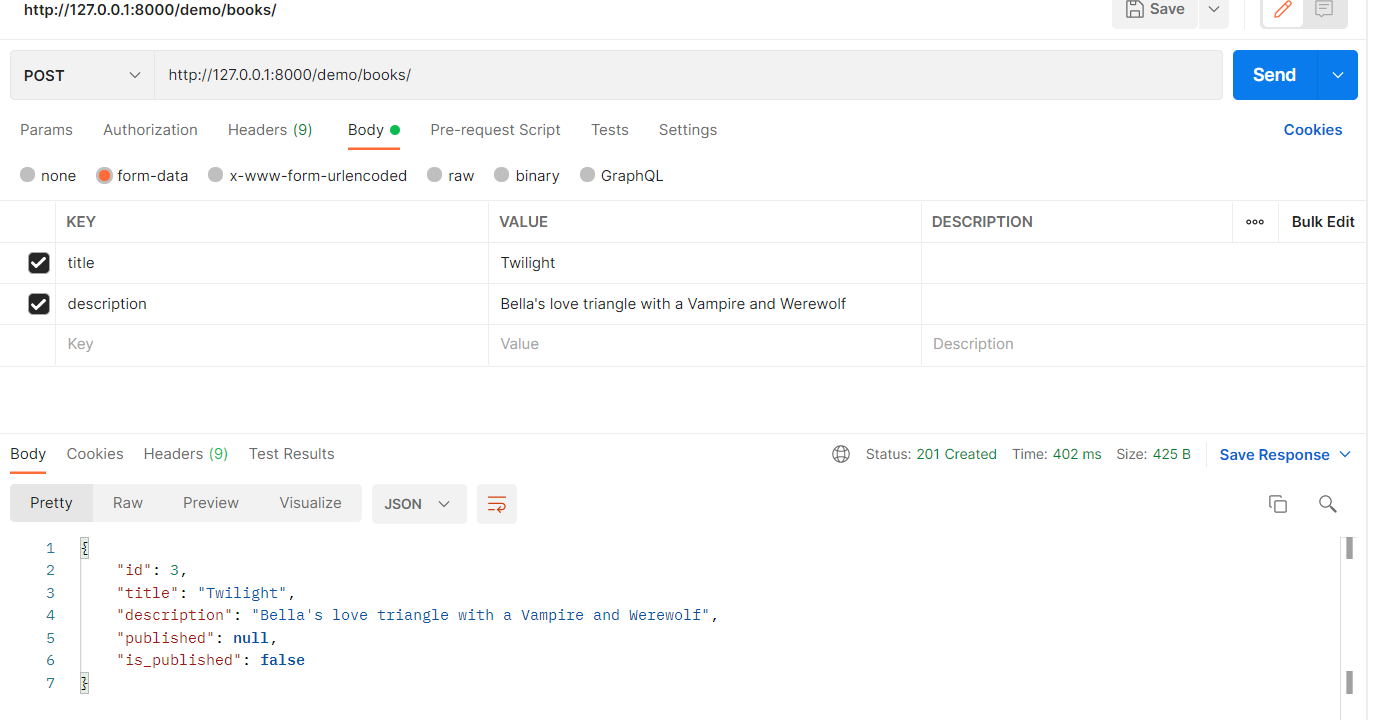


Now lets check if we can post a book:

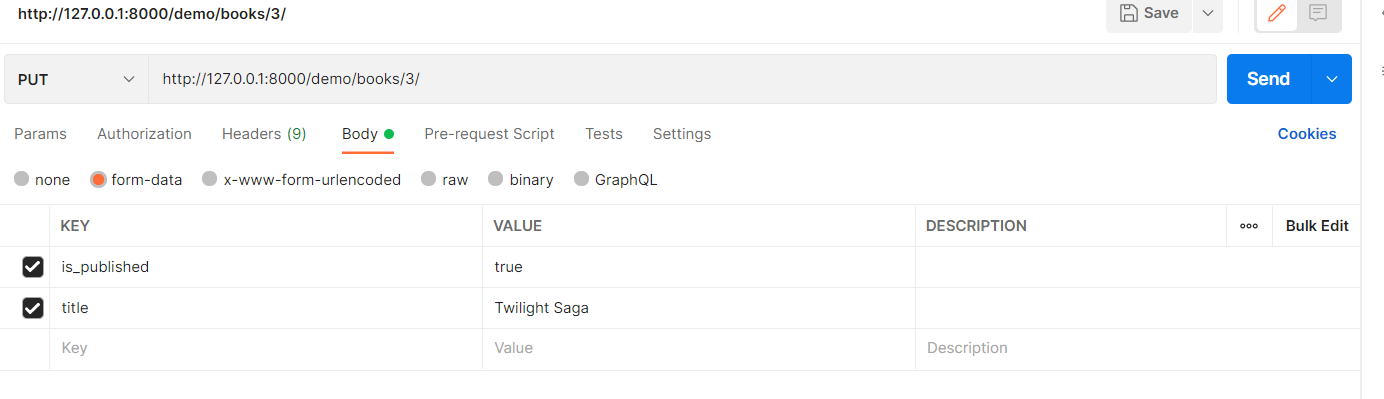
Post:



O/P:



Now we use PUT method(Update):

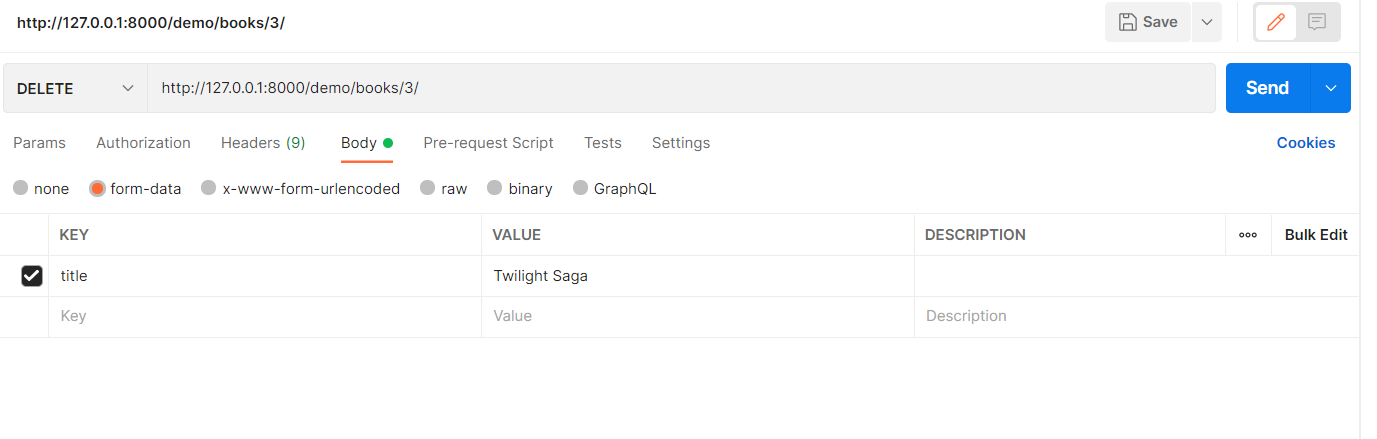


We have to use title as this is a required field:

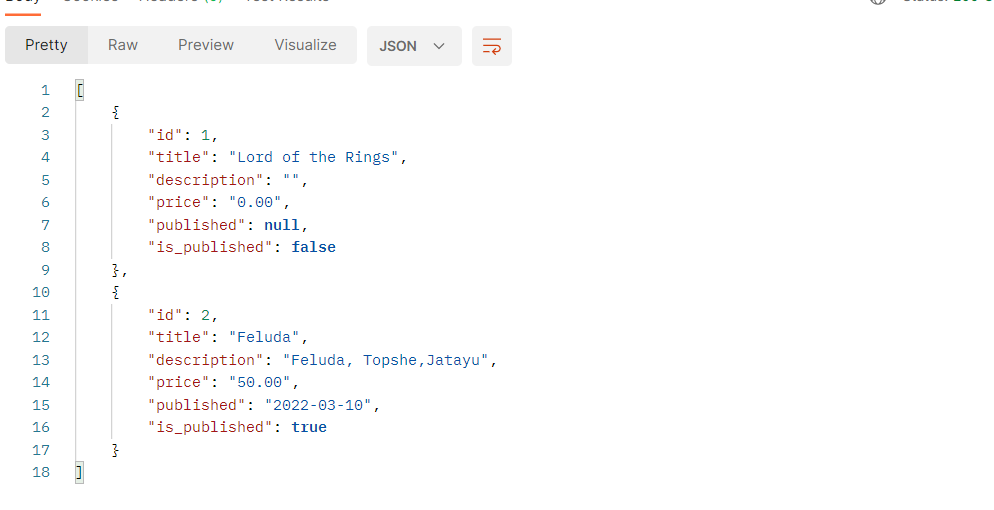
O/P:



DELETE:



O/P:



3rd book is deleted

We have GET,GET/{Specific},POST,PUT,DELETE

Token:

Creating Token is essential for keeping our application in check.

Go to settings.py:

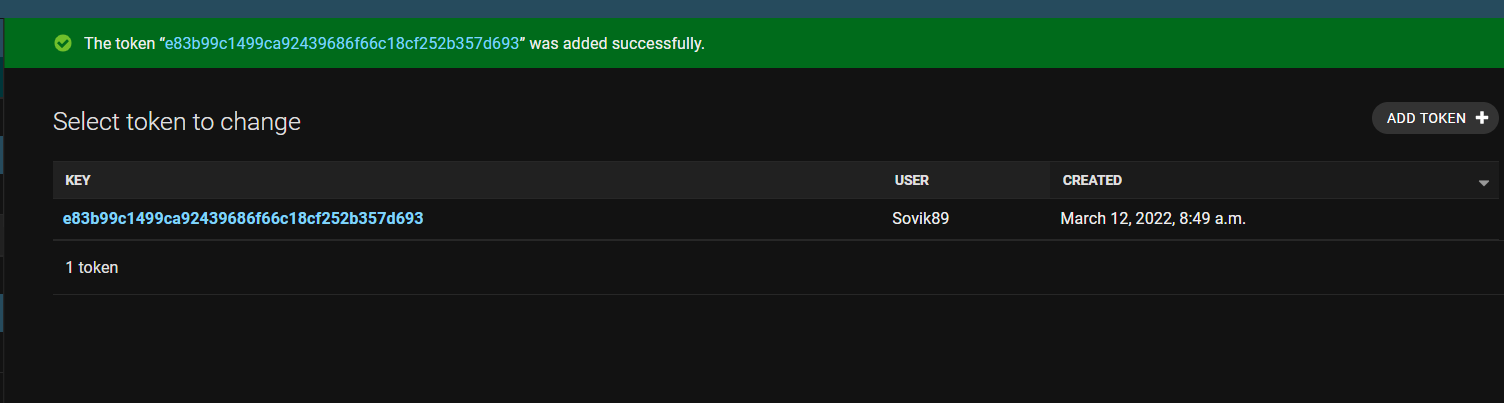
INSTALLED\_APPS = [  
 **'django.contrib.admin'**,  
 **'django.contrib.auth'**,  
 **'django.contrib.contenttypes'**,  
 **'django.contrib.sessions'**,  
 **'django.contrib.messages'**,  
 **'django.contrib.staticfiles'**,  
 **'rest\_framework'**,  
 **'rest\_framework.authtoken'**,  
 **'demo'**,  
]

Now we migrate and then we run the server.

O/P:



Now we add a token for super user:

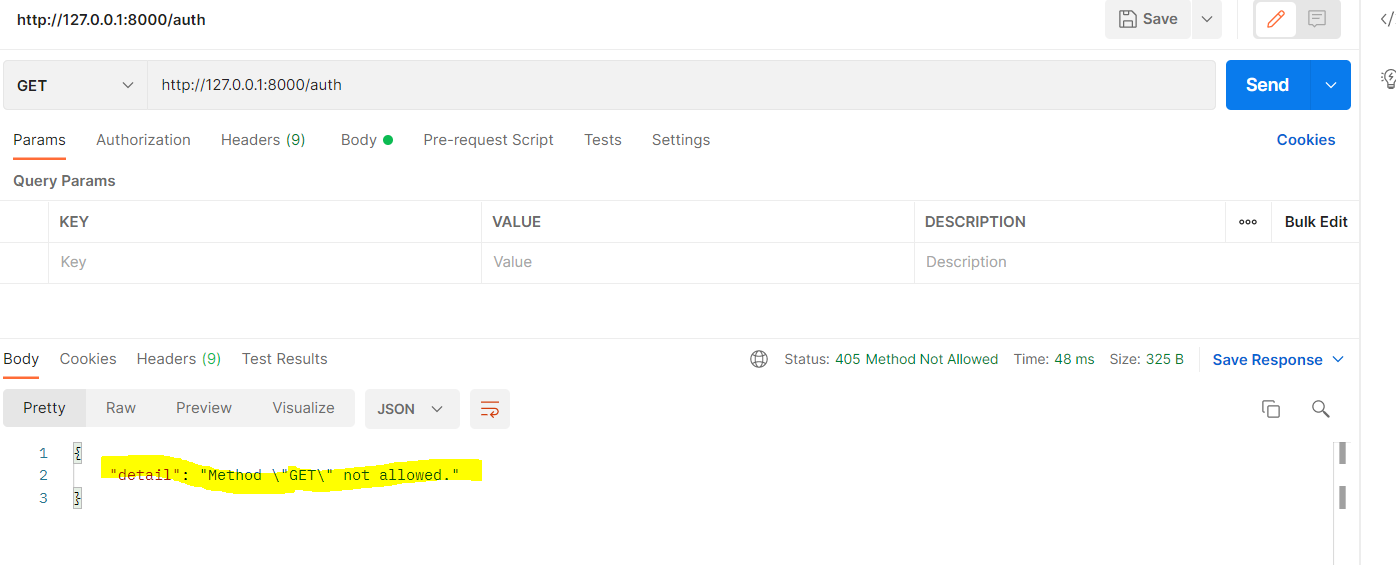


We can check this out from post man:

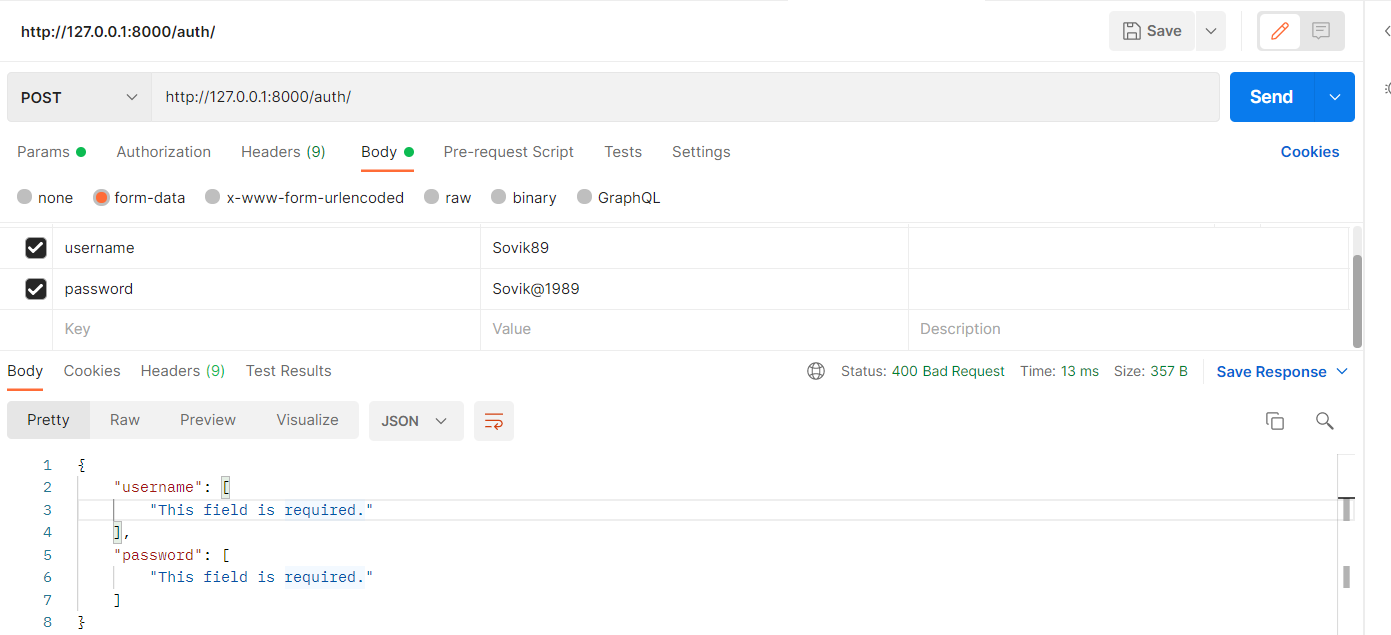
Now we create endpoint for tokens. But we create it in our project in the first/urls.py:



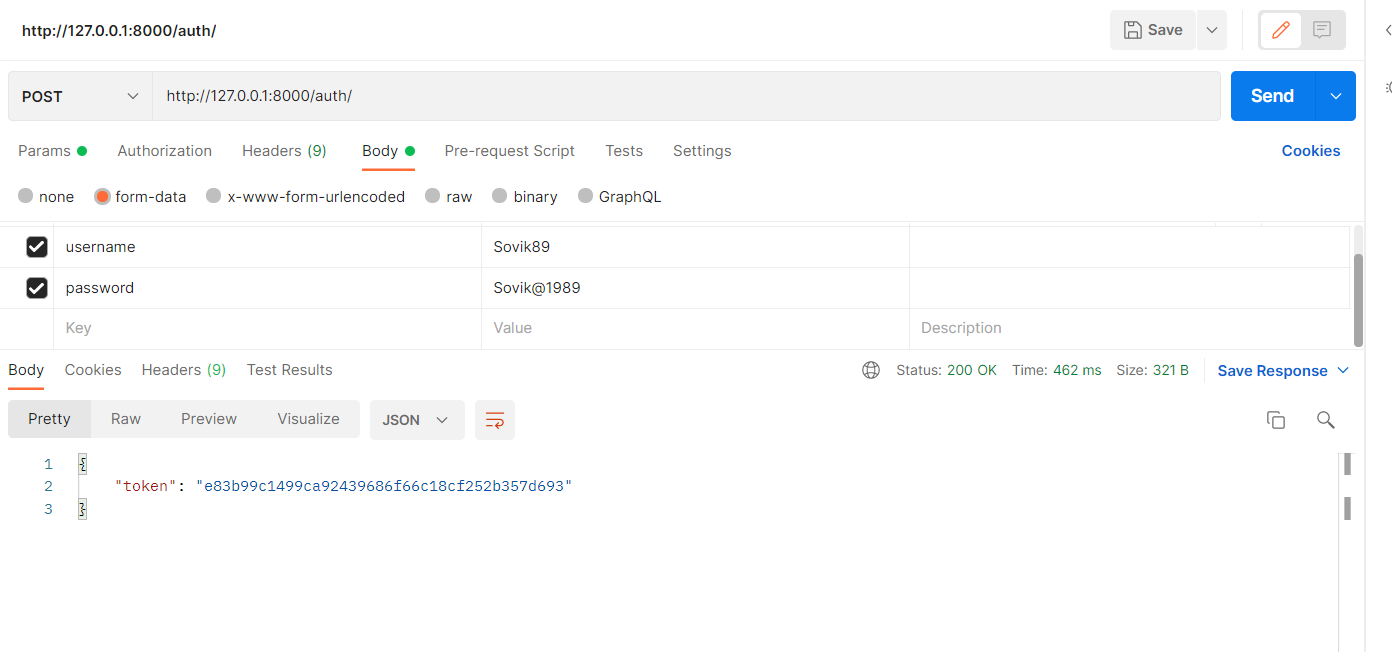
We run the server and we open postman:



GET method is not allowed. We use POST:



O/P:



Permission:

Now in views.py:

*# from django.http import HttpResponse***from** django.shortcuts **import** render  
*# from django.views import View***from** .models **import** Book  
**from** rest\_framework **import** viewsets  
**from** .serializers **import** BookSerializer  
**from** rest\_framework.authentication **import** TokenAuthentication  
**from** rest\_framework.permissions **import** IsAuthenticated  
  
*# Create your views here.  
  
# class Another(View):  
# #books=Book.objects.all()#Getting all the objects/live data present in the models  
# #books=Book.objects.filter(is\_published=True)  
# book=Book.objects.get(id=2)  
# #output=''  
# output="We have {} book in the database with ID {}".format(book.title,book.id)  
#  
# # for book in books:  
# # output += "We have {} book in the database with ID {}".format(book.title,book.id)+"<br>"  
#  
#  
#  
# def get(self,request):  
# return HttpResponse(self.output)***def** first(request):  
*# #return HttpResponse('THis is the first message from views')* books=Book.objects.all()  
 **return** render(request,**'first\_temp.html'**,{**'books'**:books})  
  
  
**class** BookViewSet(viewsets.ModelViewSet):  
 serializer\_class = BookSerializer  
 queryset = Book.objects.all()  
 authentication\_classes = (TokenAuthentication,)  
 permission\_classes = (IsAuthenticated,)

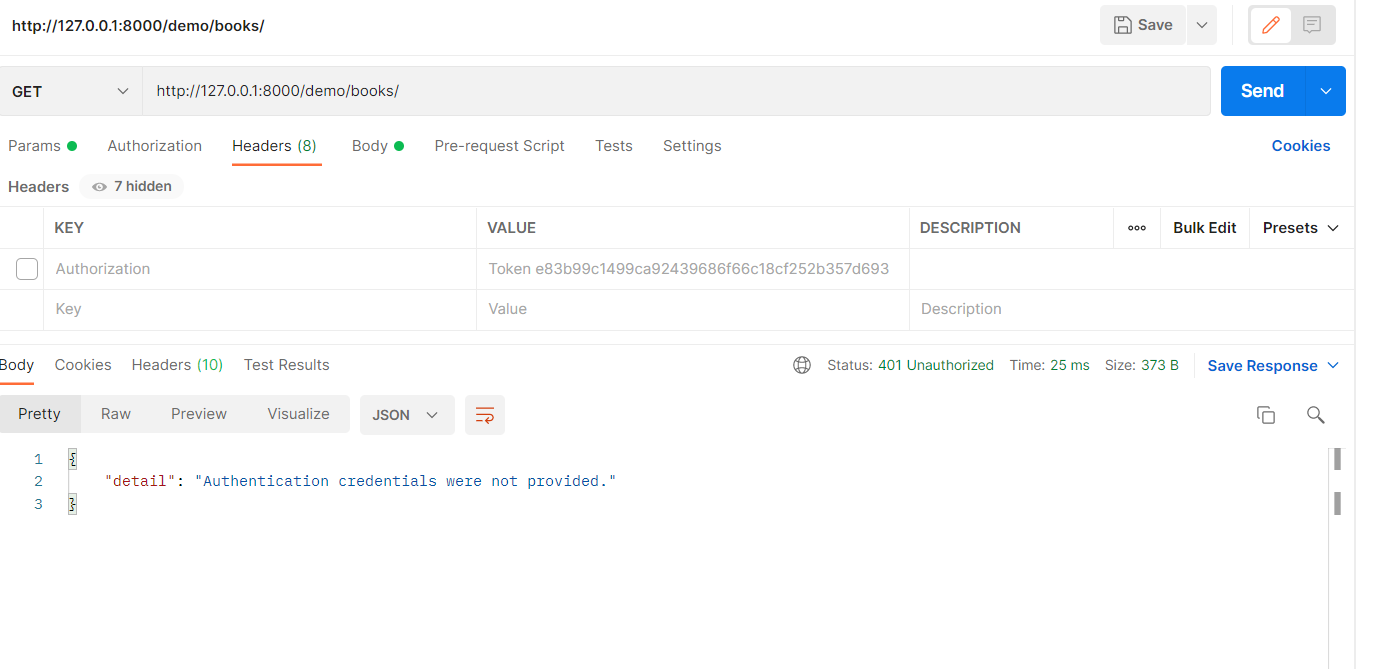
Settings.py:

REST\_FRAMEWORK = {  
 **'DEFAULT\_PERMISSION\_CLASSES'**:(  
 **'rest\_framework.permissions.AllowAny'**,  
 )  
}

The value could also be ‘rest\_framework.permissions.IsAuthenticated’, then every user endpoint could have been permissioned we want to provide autonomy to the specific endpoints.

Now we check normal GET process for /demo/books endpoint:

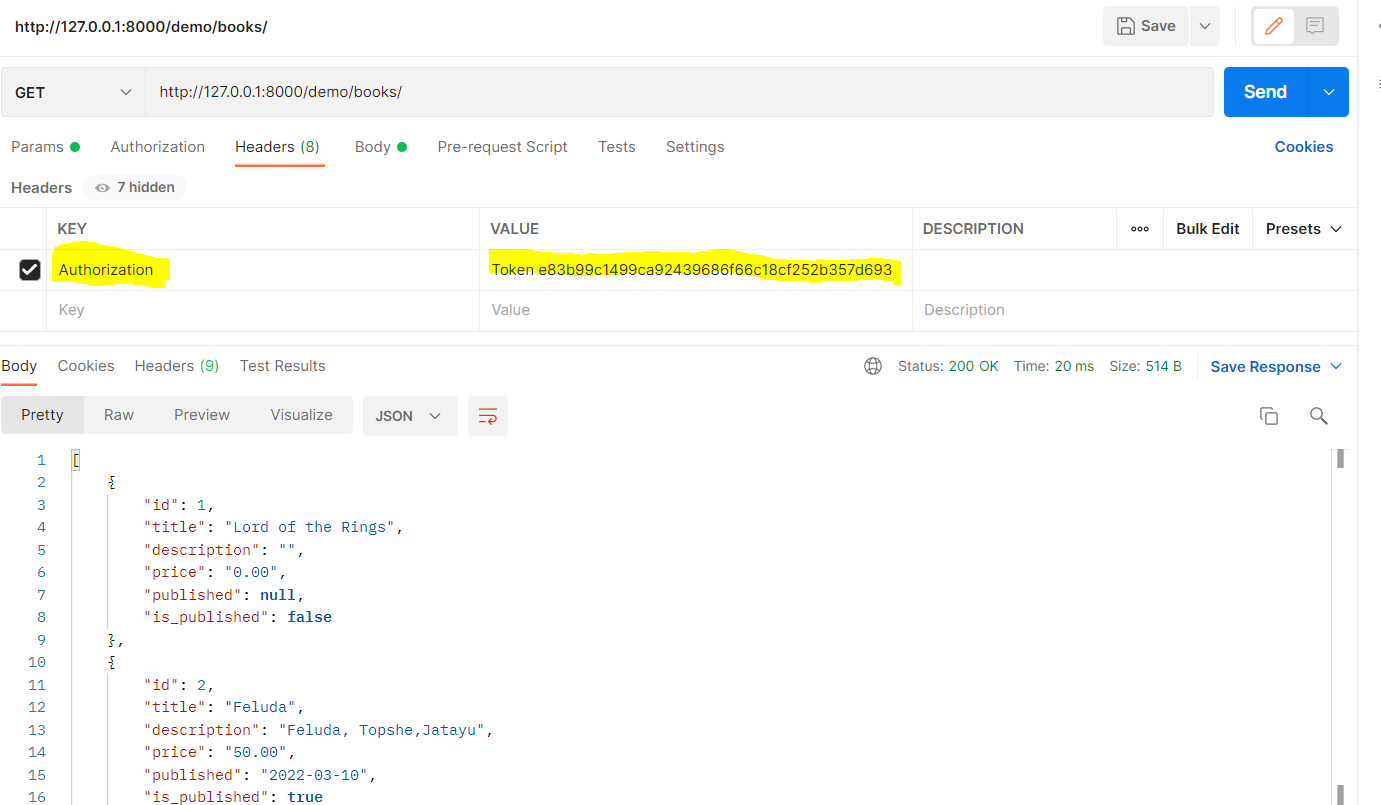
Here we are not providing any header value:



Now when we provide the header value:

Header Value:

Authorization: Token<space><Token\_value>



Relationship:

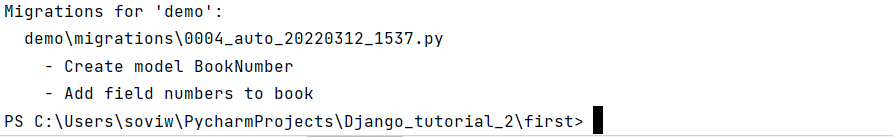
1-to-1:

We create a Book Number with ISBN number which is unique to a book:

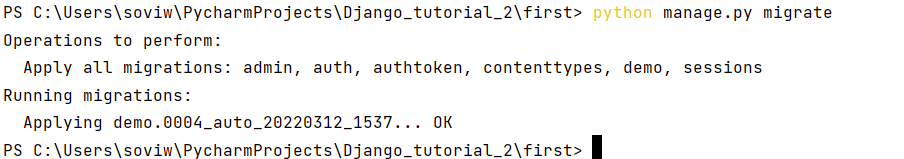
Now in Model.py:

**from** django.db **import** models  
  
*# Create your models here.***class** BookNumber(models.Model):  
 isbn\_10=models.CharField(max\_length=10)  
 isbn\_13=models.CharField(max\_length=13)  
  
  
**class** Book(models.Model):  
 *#title=models.CharField(max\_length=36)#CharField is simple text. Max lenght 36* title=models.CharField(blank=**False**,unique=**True**,max\_length=36)  
 description=models.TextField(max\_length=256,blank=**True**)  
 price=models.DecimalField(default=0,max\_digits=10,decimal\_places=2)*# can have max\_digit param, you can also use FloatFields  
 #published=models.DateField(auto\_now=True,auto\_now\_add=True)* published = models.DateField(blank=**True**,null=**True**,default=**None**)  
 is\_published=models.BooleanField(default=**False**)  
 cover=models.ImageField(upload\_to=**'covers/'**,blank=**True**)  
 numbers=models.OneToOneField(BookNumber,null=**True**,blank=**True**,on\_delete=models.CASCADE)  
  
  
 **def** \_\_str\_\_(self):  
 **return** self.title

Now make migration:



Now we migrate:

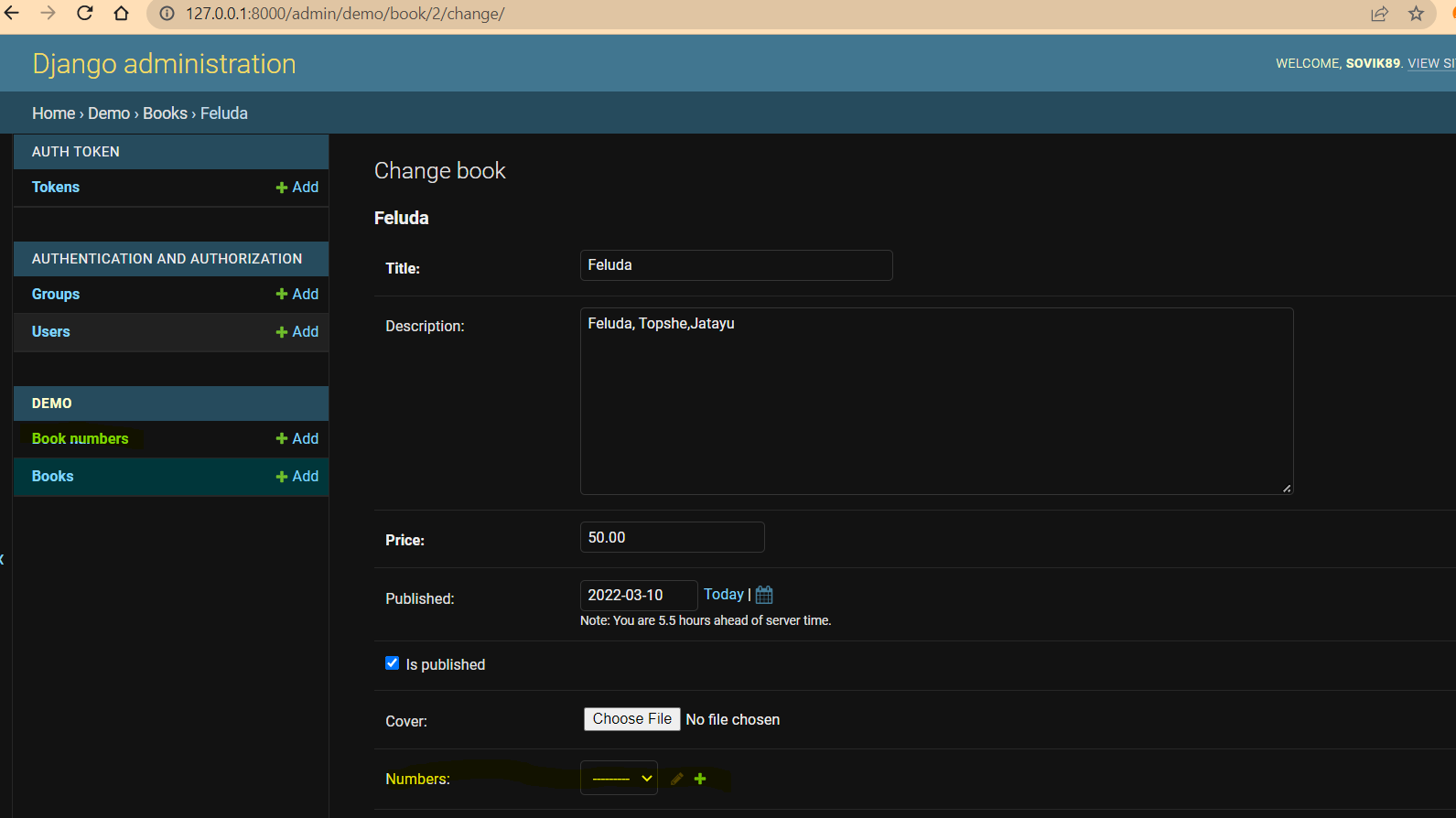


Now we run the server and check from admin panel: But nothing is there since the new model is not registered in admin.py:

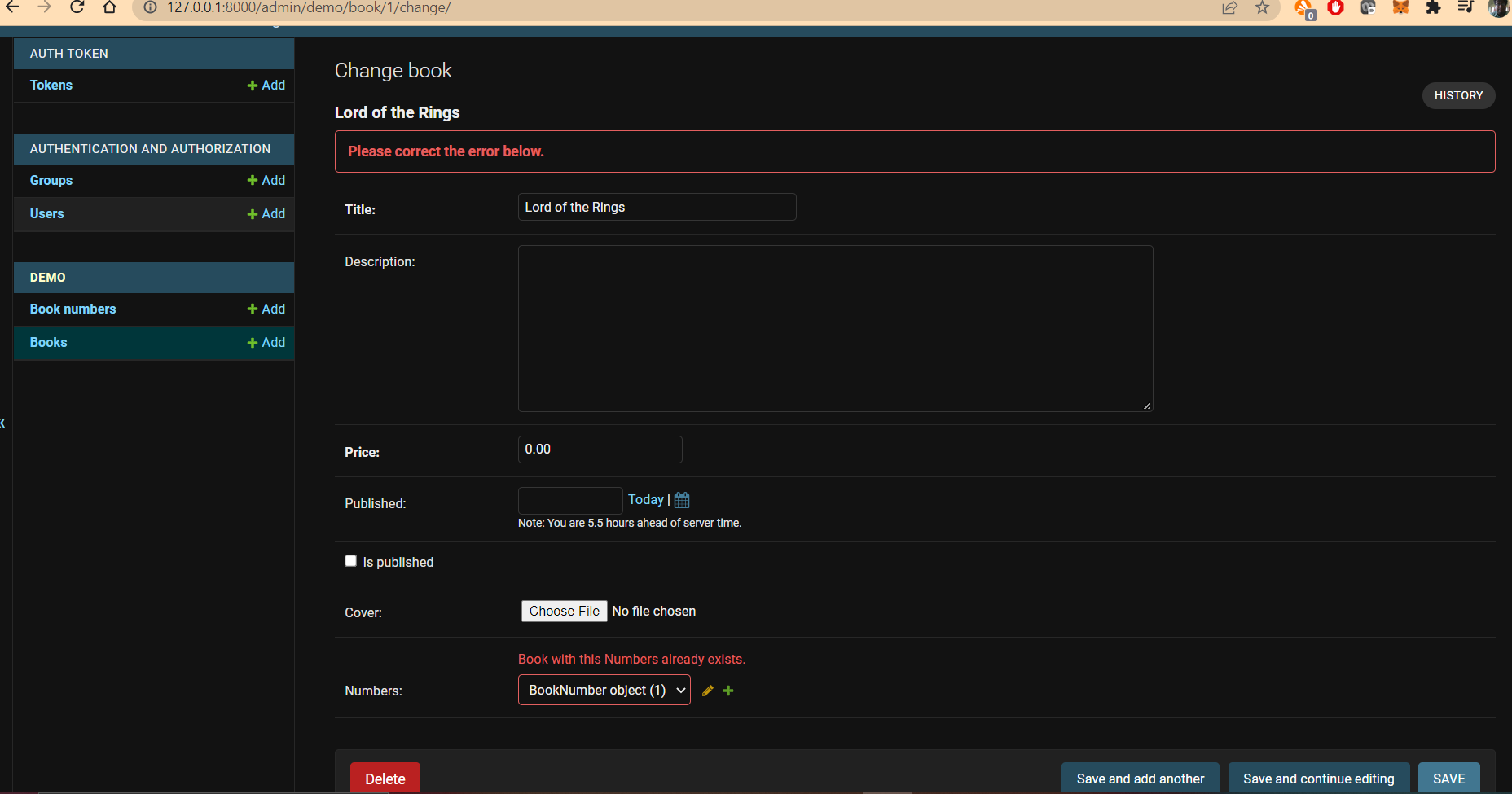
Admin.py:

admin.site.register(BookNumber)

Now we restart in the admin panel: We can find the book number and the Add sign from Book(highlighted):



We saved with Book1 and if we try to save it with Book2 again we get the below error:

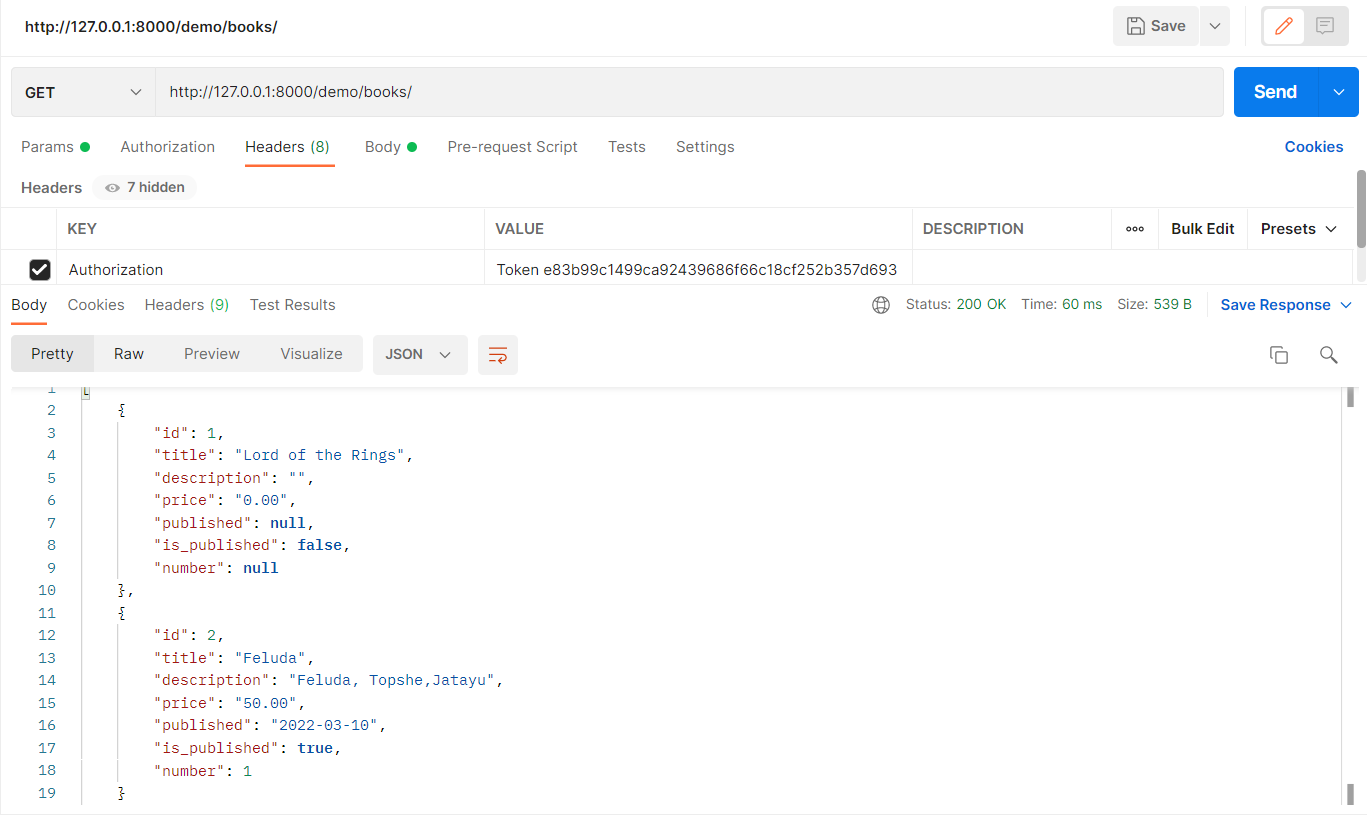


Now we go to Postman:

For this we need to change the Serializer.py:

**from** rest\_framework **import** serializers  
**from** .models **import** Book  
  
**class** BookSerializer(serializers.ModelSerializer):  
 **class** Meta:  
 model=Book  
 fields=[**'id'**,**'title'**,**'description'**,**'price'**,**'published'**,**'is\_published'**,**'number'**]

O/P:

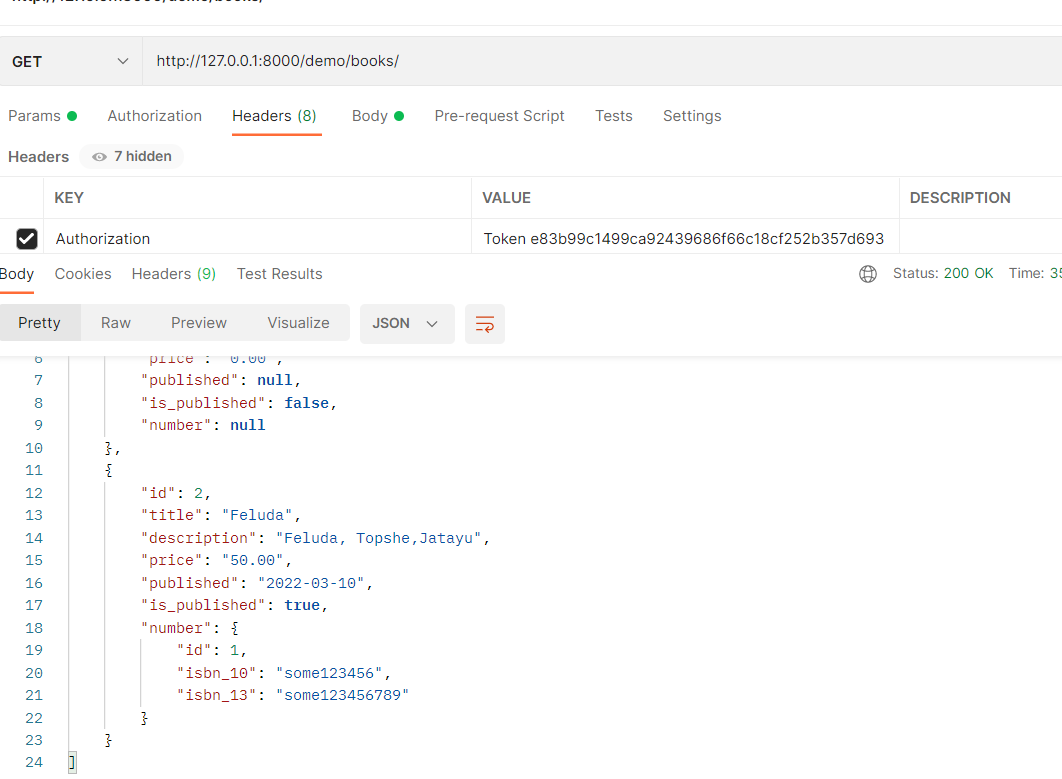


But we want to see the ISBN:

So we go back to the Serialiser.py:

**from** rest\_framework **import** serializers  
**from** .models **import** Book,BookNumber  
  
  
  
**class** BookNumberSerializer(serializers.ModelSerializer):  
 **class** Meta:  
 model=BookNumber  
 fields=[**'id'**,**'isbn\_10'**,**'isbn\_13'**,]  
  
**class** BookSerializer(serializers.ModelSerializer):  
 number=BookNumberSerializer(many=**False**)  
 **class** Meta:  
 model=Book  
 fields=[**'id'**,**'title'**,**'description'**,**'price'**,**'published'**,**'is\_published'**,**'number'**]

O/P:



One-to-Many:

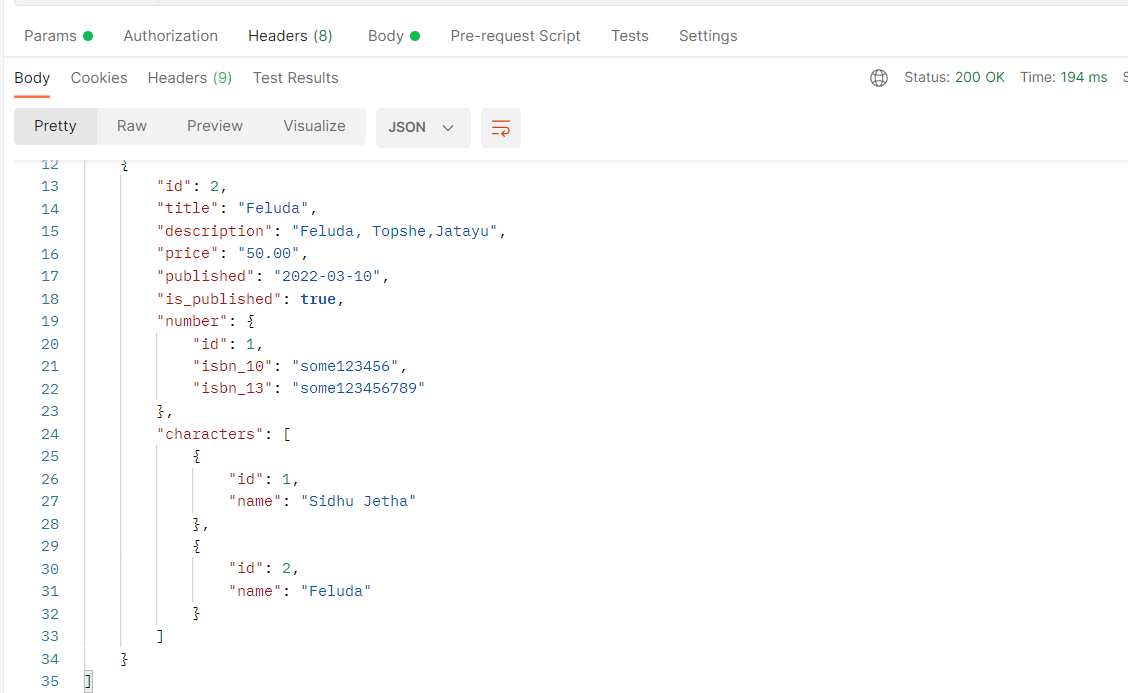
We use character as a model. So we go to model.py

**class** Character(models.Model):  
 name=models.CharField(max\_length=30)  
 book=models.ForeignKey(Book,on\_delete=models.CASCADE,related\_name=**'characters'**)

In serializer.py:

**from** rest\_framework **import** serializers  
**from** .models **import** Book,BookNumber,Character  
  
  
  
**class** BookNumberSerializer(serializers.ModelSerializer):  
 **class** Meta:  
 model=BookNumber  
 fields=[**'id'**,**'isbn\_10'**,**'isbn\_13'**]  
  
  
**class** CharacterSerializer(serializers.ModelSerializer):  
 **class** Meta:  
 model=Character  
 fields=[**'id'**,**'name'**]  
  
**class** BookSerializer(serializers.ModelSerializer):  
 number=BookNumberSerializer(many=**False**)  
 characters=CharacterSerializer(many=**True**)  
 **class** Meta:  
 model=Book  
 fields=[**'id'**,**'title'**,**'description'**,**'price'**,**'published'**,**'is\_published'**,**'number'**,**'characters'**]

O/P:



Many-to-Many:

We use Authors as many to many relation with Book:

Model.py:

**class** Authors(models.Model):  
 name = models.CharField(max\_length=30)  
 surname = models.CharField(max\_length=30)  
 *# Many to Many* books = models.ManyToManyField(Book, related\_name=**'authors'**)

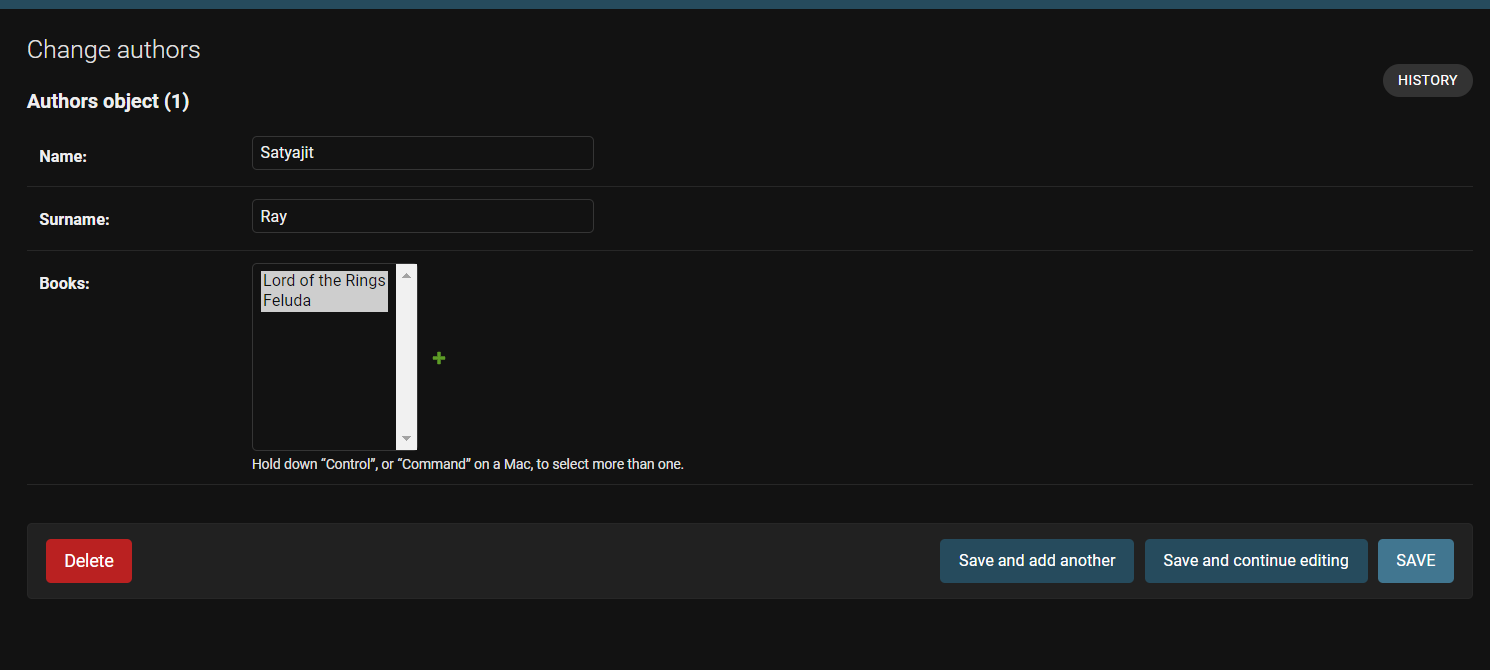
Admin.py:

**from** django.contrib **import** admin  
**from** .models **import** Book,BookNumber,Character,Authors  
  
  
*# Register your models here.  
  
#admin.site.register(Book)  
  
  
#Below process actually gives more control over the class*@admin.register(Book)  
**class** BookAdmin(admin.ModelAdmin):  
 *#fields=['title','description']* list\_display = [**'title'**,**'description'**]  
 list\_filter = [**'is\_published'**]  
 search\_fields = [**'title'**]  
  
  
admin.site.register(BookNumber)  
admin.site.register(Character)  
admin.site.register(Authors)

Serializer.py:

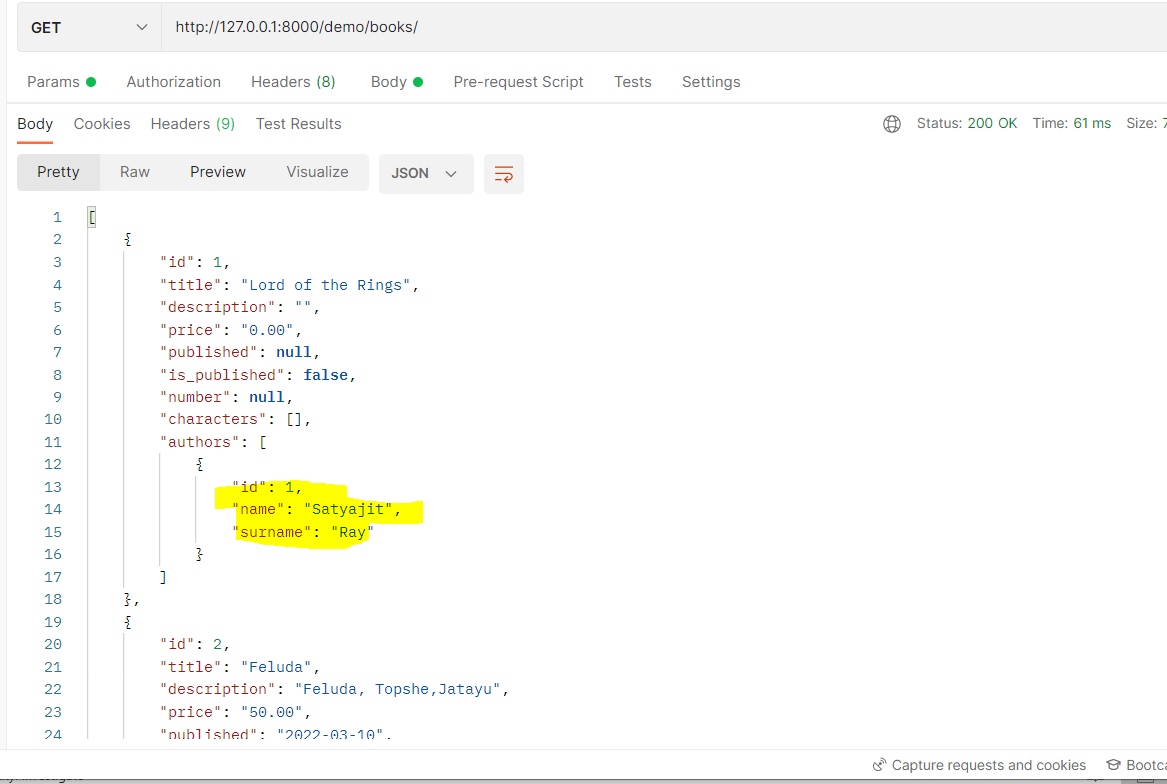
**class** AuthorsSerializer(serializers.ModelSerializer):  
 **class** Meta:  
 model=Authors  
 fields=[**'id'**,**'name'**,**'surname'**]  
  
  
**class** BookSerializer(serializers.ModelSerializer):  
 number=BookNumberSerializer(many=**False**)  
 characters=CharacterSerializer(many=**True**)  
 authors=AuthorsSerializer(many=**True**)  
 **class** Meta:  
 model=Book  
 fields=[**'id'**,**'title'**,**'description'**,**'price'**,**'published'**,**'is\_published'**,**'number'**,**'characters'**,**'authors'**]

Migrate and run server:



We can select multiple books.

O/P:



This author in both the books.