Tutorial 04 to do in class – Remember to upload the repo link to Teams.

Antes de iniciar:

- Terminar los talleres no calificables anteriores.
- Este taller muestra ejemplos de creación de servicios REST en Django.

Setup

- Create a directory named todoapp.
- Inside the todoapp folder, create a new django project named *backend.*
- Create an app in the backend folder named *todo.*
- Go to */backend/settings.py* and add the following code:

Add Bold Code

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Custom apps start here
    'todo.apps.TodoConfig',;
```

- Go to */backend/models.py* and add the following code:

Replace Entire Code

```
from django.db import models
from django.contrib.auth.models import User

# Create your models here.
class ToDo(models.Model):
    title = models.CharField(max_length=100)
    memo = models.TextField(blank=True)

    # Set to current time
    created = models.DateTimeField(auto_now_add=True)
    completed = models.BooleanField(default=False)

    # User who posted
    user = models.ForeignKey(User, on_delete=models.CASCADE)

    def __str__(self):
        return self.title
```

- Go to the Terminal and run the following commands:
  ```
  python manage.py makemigrations
  python manage.py migrate
  ```

- Go to the Terminal and run the following commands:
  ```
  python manage.py createsuperuser
  ```

- Create an admin login, then run the app by executing the following command in the Terminal:

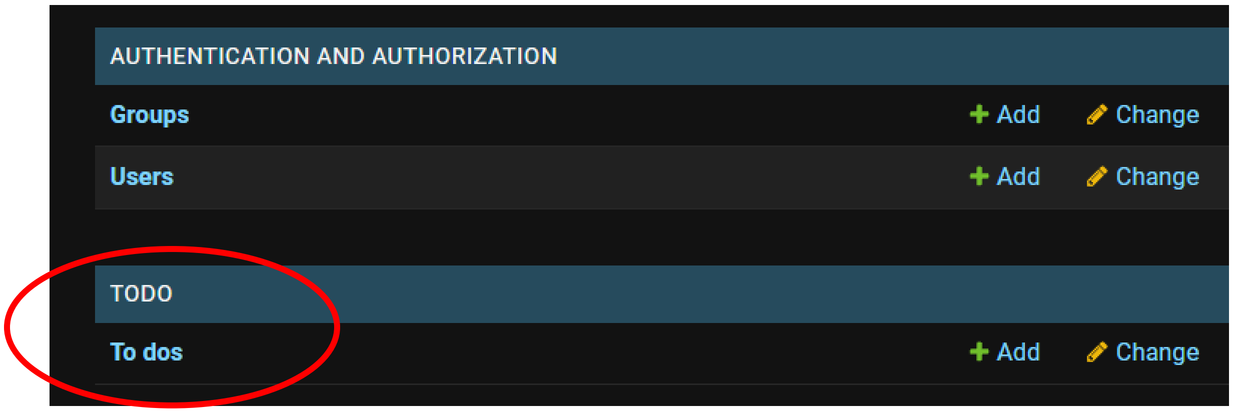
  ```
  python manage.py runserver
  ```

- Go to *todo/admin.py* and add the following code:

Replace Entire Code

```
from django.contrib import admin
from .models import ToDo

# Register your models here.
admin.site.register(ToDo)
```

- Go to http://localhost:8000/admin/ , log in with your superuser credentials. You should see the following:



- You can add a To Do by clicking +Add.

A. An API system

Django REST Framework (DRF)

- Go to the Terminal and install DRF:

```
pip install djangorestframework
```

- Go to *backend/setting.py* and add the following code:

Add Bold Code

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Custom apps start here
    'todo.apps.TodoConfig',
    # Third-party apps start here
    'rest_framework',
```

Creating an API

- Go to the Terminal and run the following command:

```
Python manage.py startapp api
```

- Go to the */backend/settings.py* and add the following code:

Add Bold Code

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Custom apps start here
    'todo.apps.TodoConfig',
    api.apps.ApiConfig',
    # Third-party apps start here
    'rest_framework',
```

API Serializers

- Create a new file in the *api* app called *serializers.py* and add the following code:

Add Entire Code

```
from rest_framework import serializers
from todo.models import ToDo

class ToDoSerializer(serializers.ModelSerializer):
    created = serializers.ReadOnlyField()
    completed = serializers.ReadOnlyField()

    class Meta:
        model = ToDo
        fields = ['id', 'title', 'memo', 'created', 'completed']
```

API Controller

- Go to *api/views.py* and add the following content:

Replace Entire Code

```
from rest_framework import generics
from .serializers import TodoSerializer
from todo.models import Todo

class TodoList(generics.ListAPIView):
    # ListAPIView requires two mandatory attributes, serializer_class and
    # queryset.
    # We specify TodoSerializer which we have earlier implemented
    serializer_class = TodoSerializer

    def get_queryset(self):
        user = self.request.user
        return Todo.objects.filter(user=user).order_by('-created')
```

API routes

- Go to *backend/urls.py* and make the following changes in **bold**.

Modify Bold Code

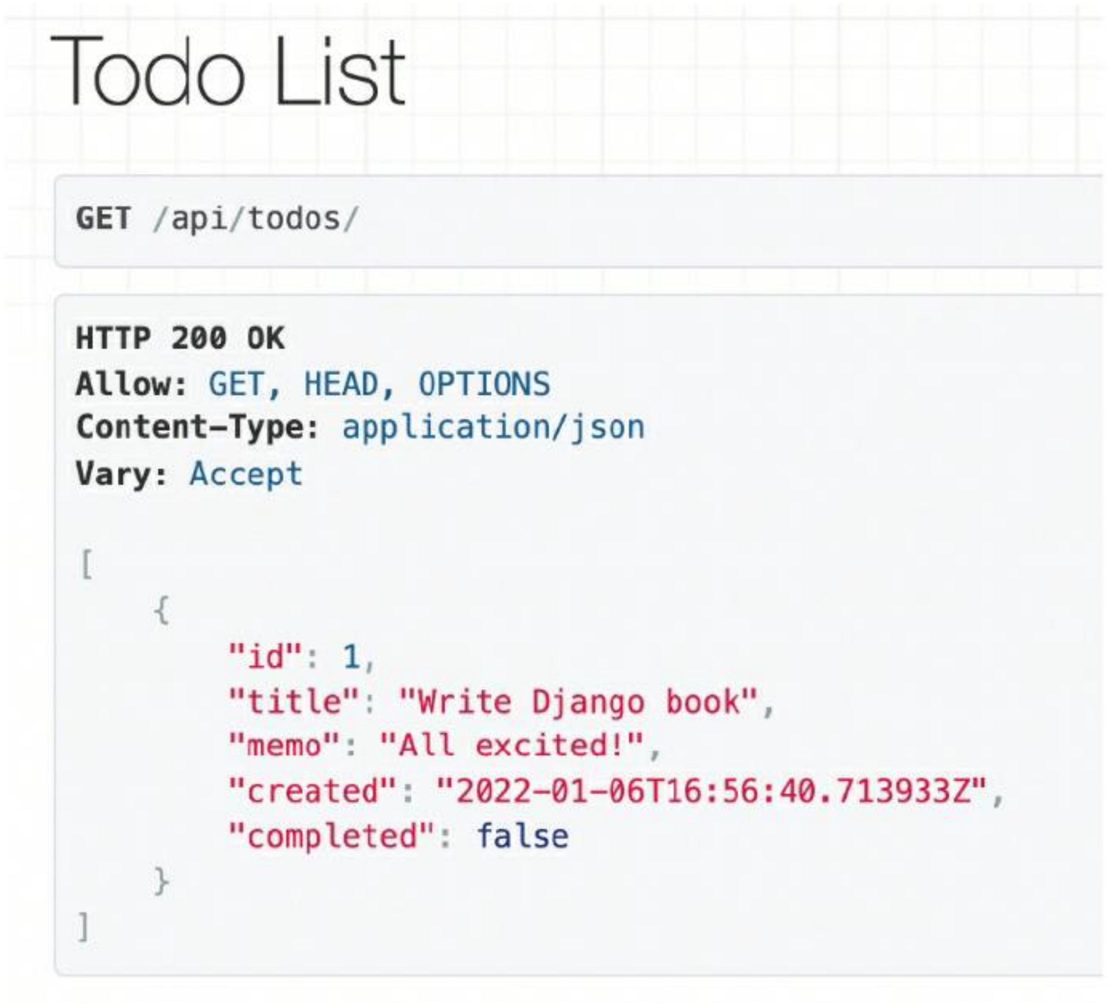
```
...
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('api.urls')) # Include urls from API.
]
```

- Create a file named *urls.py* in the *api* app and add the following code:

```
from django.urls import path
from . import views

urlpatterns = [
    path(todos/', views.ToDoList.as_view(), name='list'),
]
```

Run the application

- Go to http://127.0.0.1:8000/api/todos and it should display a screen like this:



Todo List

```
GET /api/todos/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 1,
        "title": "Write Django book",
        "memo": "All excited!",
        "created": "2022-01-06T16:56:40.713933Z",
        "completed": false
    }
]
```

Improve API Controller

- Got to api/views.py and add the following code:

Modify Bold Code

```
from rest_framework import generics
from .serializers import TodoSerializer
from todo.models import Todo

class TodoListCreate(generics.ListCreateAPIView):
    # ListAPIView requires two mandatory attributes, serializer_class and
    # queryset.
    # We specify TodoSerializer which we have earlier implemented
    serializer_class = TodoSerializer

    def get_queryset(self):
        user = self.request.user
        return Todo.objects.filter(user=user).order_by('-created')

    def perform_create(self, serializer):
        #serializer holds a django model
        serializer.save(user=self.request.user)
```

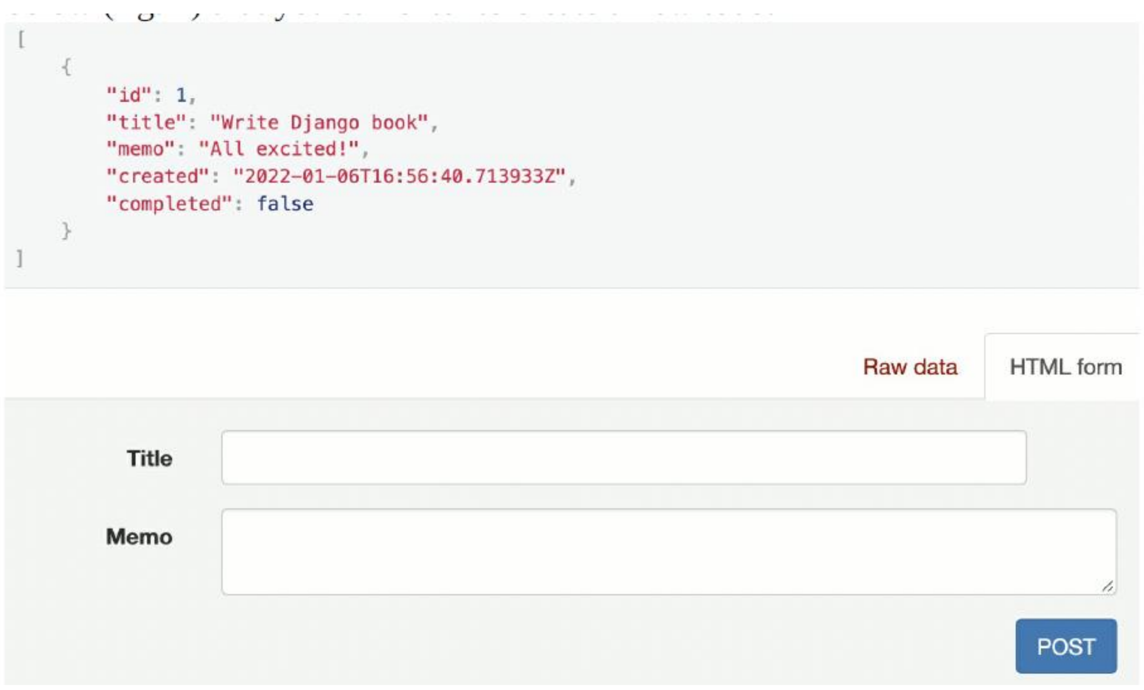
Modify API routes

Modify Bold Code

```
from django.urls import path
from . import views

urlpatterns = [
    path(todos/', views.ToDoListCreate.as_view(), name='list'),
]
```

Run the application

- Go to http://127.0.0.1:8000/api/todos and it should display a screen like this:



```
[
    {
        "id": 1,
        "title": "Write Django book",
        "memo": "All excited!",
        "created": "2022-01-06T16:56:40.713933Z",
        "completed": false
    }
]
```

- You can now create To Do lists from the API.

B. An API system with Permissions

Api Controller

- Go to *api/views.py* and add the following code:

Modify Bold Code

from rest_framework import generics, **permissions**

from .serializers import TodoSerializer

from todo.models import Todo

class TodoList**Create**(generics.List**Create**APIView):

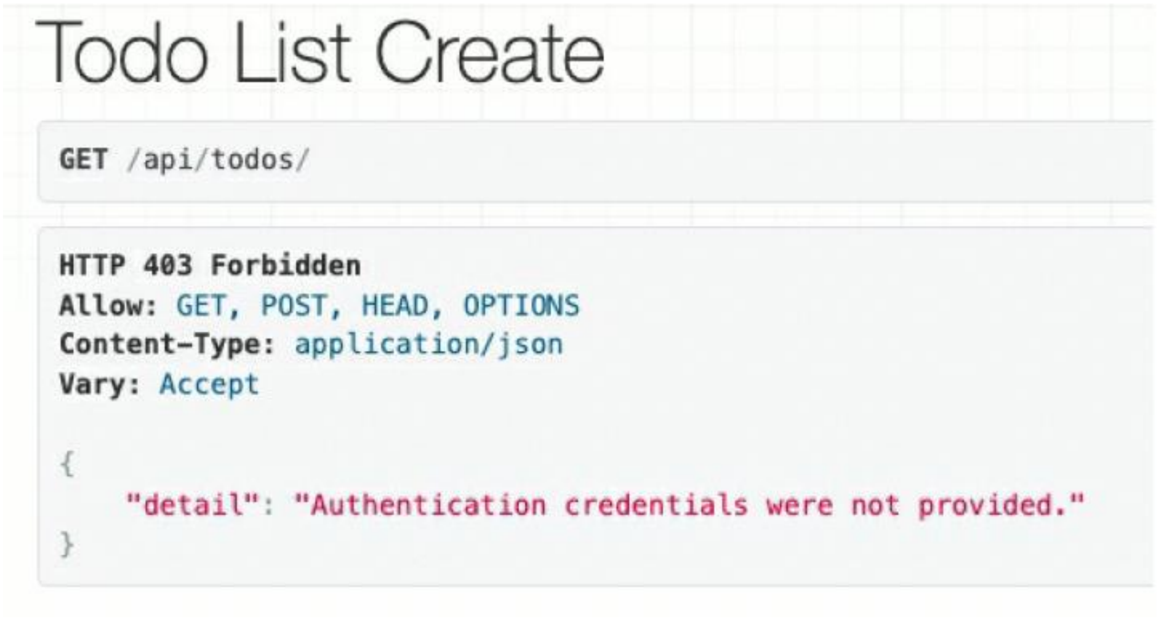
    …

    serializer_class = TodoSerializer

    permission_classes = [permissions.IsAuthenticated]

Run the application

- Go to *localhost:8000/admin* and log out of your superuser account.
- Next, go to *localhost:8000/todos* and you should see the following screen:



```
GET /api/todos/
```

```
HTTP 403 Forbidden
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "detail": "Authentication credentials were not provided."
}
```

- Before proceeding with finalizing user authentication, we will explore other operations with our API endpoint.

C. Other C.RU.D. operations with an API system

API routes

- Go to *api/urls.py/* and add the follwoing content:

Add Bold Code

```
…
urlpatterns = [
path('todos/', views.TodoListCreate.as_view()),
path('todos/<int:pk>', views.TodoRetrieveUpdateDestroy.as_view()),
]
```

API Controller

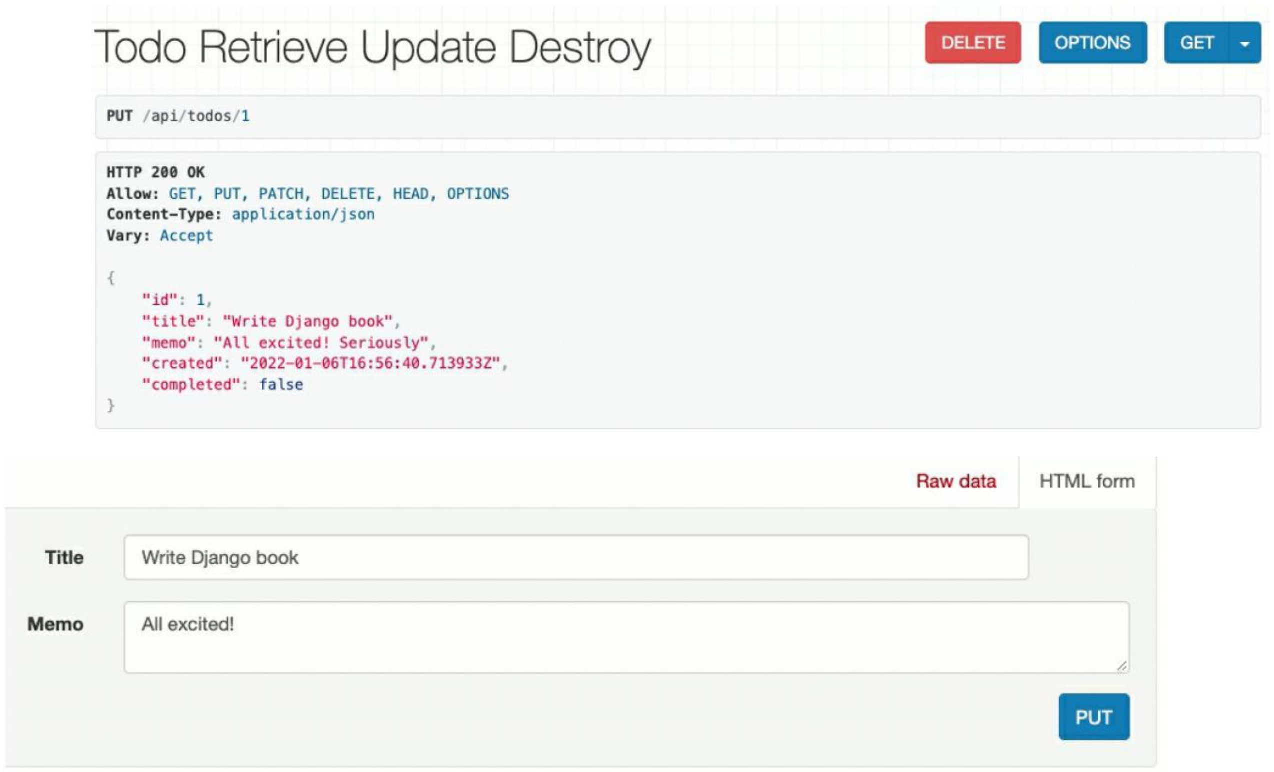
- Go to *api/views.py* and add the follwing content:

Add Bold Code

```
…
class TodoListCreate(generics.ListCreateAPIView):
…

class TodoRetrieveUpdateDestroy(generics.RetrieveUpdateDestroyAPIView):
serializer_class = TodoSerializer
permission_classes = [permissions.IsAuthenticated]
def get_queryset(self):
user = self.request.user
# user can only update, delete own posts
return Todo.objects.filter(user=user)
```

Run the application

- Go to *http://localhost:8000/api/todos/1* and you should see the following screen:



```
PUT /api/todos/1
```

```
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "title": "Write Django book",
    "memo": "All excited! Seriously",
    "created": "2022-01-06T16:56:40.713933Z",
    "completed": false
}
```

Title: Write Django book

Memo: All excited!

D. Non-C.R.U.D. operations

- We will add logic to implement a view that completes a To Do List

API routes

- Go to *apì/urls.py* and add the following code:

Add Bold Code

```
...
urlpatterns = [
path('todos/', views.TodoListCreate.as_view()),
path('todos/<int:pk>', views.TodoRetrieveUpdateDestroy.as_view()),
path('todos/<int:pk>/complete', views.TodoToggleComplete.as_view()),
]
```

API Serializer

- Go to *api/serializers.py* and add the following code:

Add Bold Code

```
from rest_framework import serializers
from todo.models import ToDo

class ToDoSerializer(serializers.ModelSerializer):
    ...

class ToDoSerializer(serializers.ModelSerializer):

    class Meta:
        model = ToDo
        fields = ['title', 'memo', 'created', 'completed']
```

API Controller

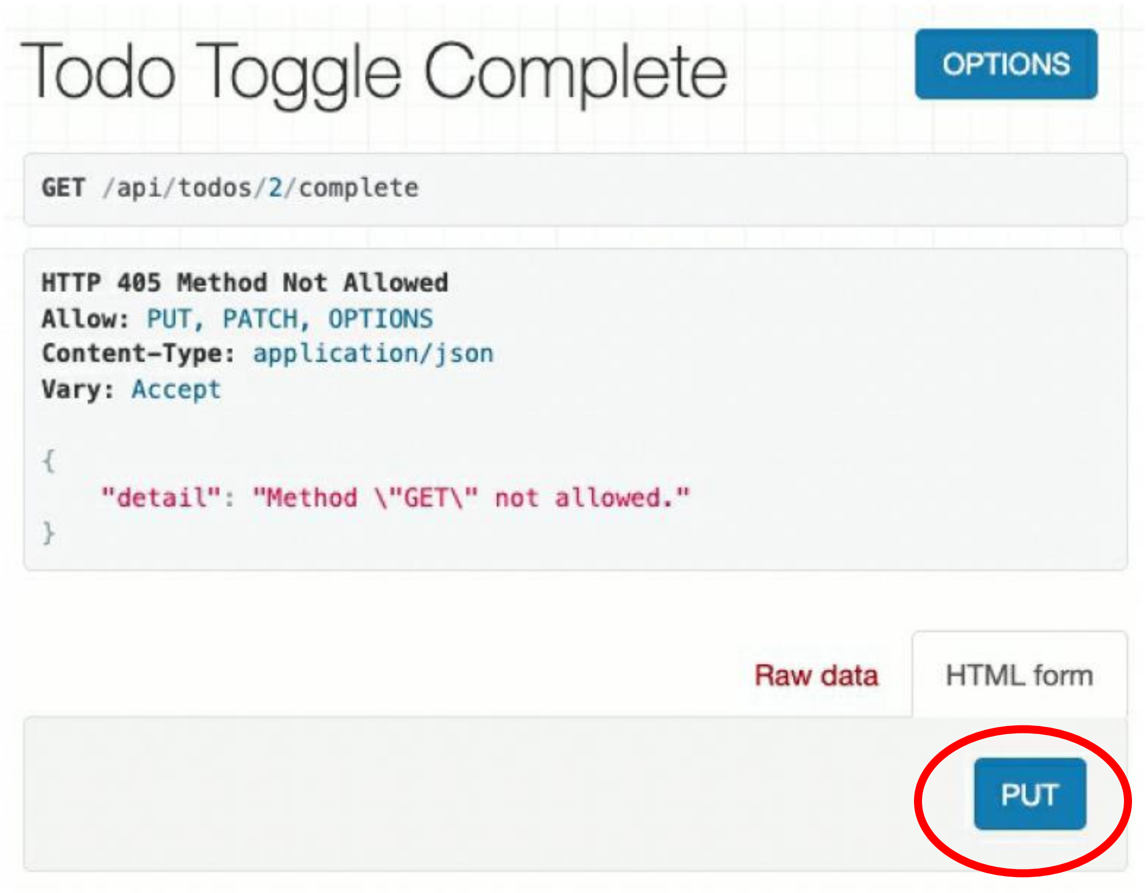
- Go to *api/views.py* and add the following code:

Add Bold Code

```
from rest_framework import generics, permissions
from .serializers import TodoSerializer, TodoToggleCompleteSerializer
from todo.models import Todo

class TodoListCreate(generics.ListCreateAPIView):
…

class TodoRetrieveUpdateDestroy(generics.RetrieveUpdateDestroyAPIView):
…

class TodoToggleComplete(generics.UpdateAPIView):
        serializer_class = TodoToggleCompleteSerializer
        permission_classes = [permissions.IsAuthenticated]

        def get_queryset(self):
                user = self.request.user
                return Todo.objects.filter(user=user)

        def perform_update(self,serializer):
                serializer.instance.completed=not(serializer.instance.completed)
                serializer.save()
```

Running the application

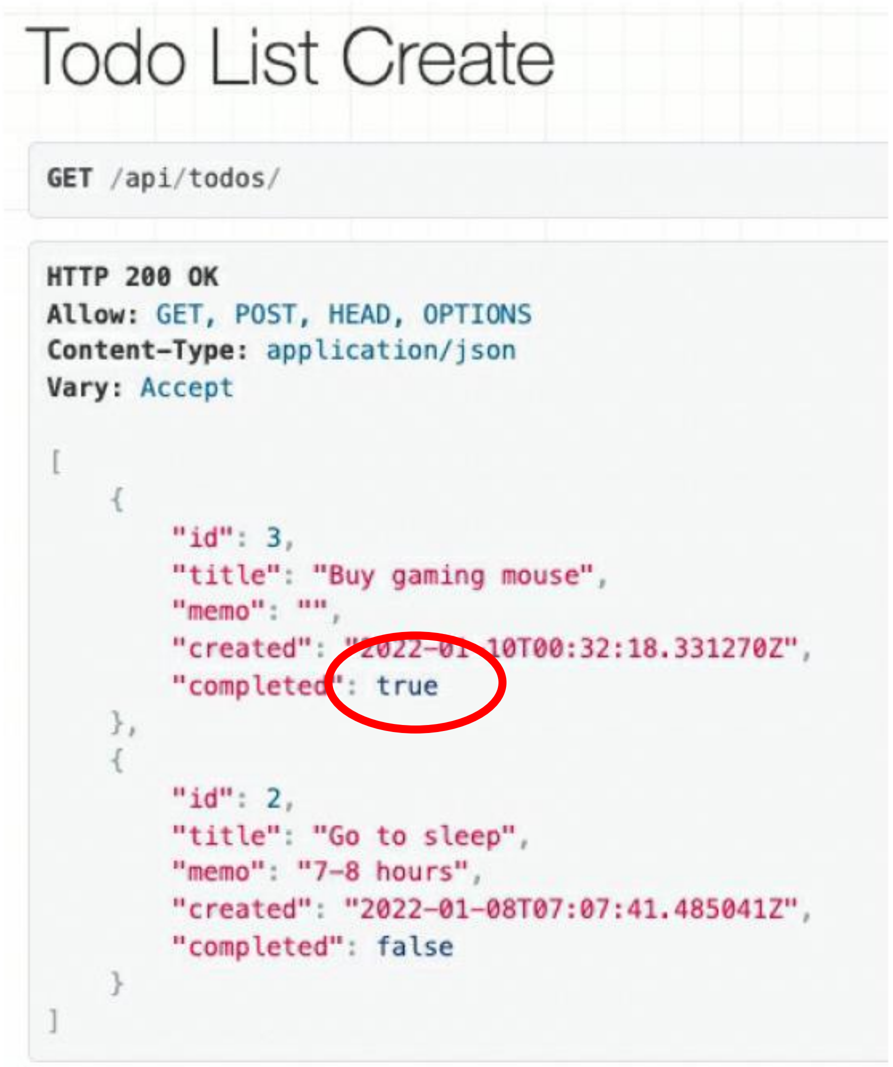
- Go to *localhost:8000/api/todos/2/complete* and click on 'PUT' on the following screen:



```
GET /api/todos/2/complete
```

```
HTTP 405 Method Not Allowed
Allow: PUT, PATCH, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "detail": "Method \"GET\" not allowed."
}
```

- Go to *localhost:8000/api/todos* and you should see the following screen:



Todo List Create

```
GET /api/todos/
```

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 3,
        "title": "Buy gaming mouse",
        "memo": "",
        "created": "2022-01-10T00:32:18.331270Z",
        "completed": true
    },
    {
        "id": 2,
        "title": "Go to sleep",
        "memo": "7-8 hours",
        "created": "2022-01-08T07:07:41.485041Z",
        "completed": false
    }
]
```

E. An API System with signup & login

Setting up authentication token

- Go to *backend/settings.py* and add the following app:

Add Bold Code

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Custom apps start here
    'todo.apps.TodoConfig',
    'api.apps.ApiConfig',
    # Third-party apps start here
    'rest_framework',
    'rest_framework.authtoken',
]
```

- Run the following command in the Terminal:

Add Bold Code

```
python manage.py migrate
```

- Go to *backend/settings.py* and add the following configuration (at the bottom of the file):

Add Bold Code

```
...
REST_FRAMEWORK = {
'DEFAULT_AUTHENTICATION_CLASSES':[
'rest_framework.authentication.TokenAuthentication',
]
```

API routes

- Next, head to *api/urls.py* and add the following code:

Add Bold Code

```
from django.urls import path
from . import views

urlpatterns = [
    path('todos/', views.ToDoListCreate.as_view(), name='todo_list'),
    path('todos/<int:pk>', views.ToDoRetrieveUpdateDestroy.as_view(), name='todo_RUD'),
    path('todos/<int:pk>/complete', views.ToDoToggleComplete.as_view()),
    path('signup/', views.signup, name='signup'),
    path('login/', views.login, name='login'),
]
```

API controller

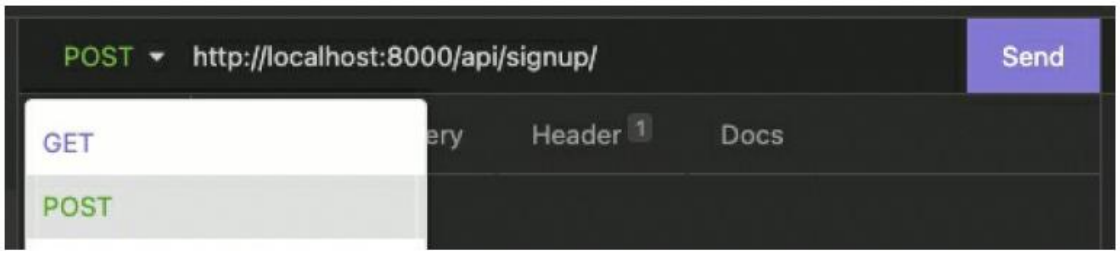
- Go to *api/views.py* and add the following code:

```
...
from todo.models import Todo
from django.db import IntegrityError
from django.contrib.auth.models import User
from rest_framework.parsers import JSONParser
from rest_framework.authtoken.models import Token
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
from django.contrib.auth import authenticate
...

class TodoToggleComplete(generics.UpdateAPIView):
...

@csrf_exempt
def signup(request):
        if request.method == 'POST':
        try:
                data = JSONParser().parse(request) # data is a dictionary
                user = User.objects.create_user(
                username=data['username'],
                password=data['password'])
                user.save()
                token = Token.objects.create(user=user)
                return JsonResponse({'token':str(token)},status=201)
        except IntegrityError:
                return JsonResponse(
                {'error':'username taken. choose another username'},
                status=400)

@csrf_exempt
def login(request):
```

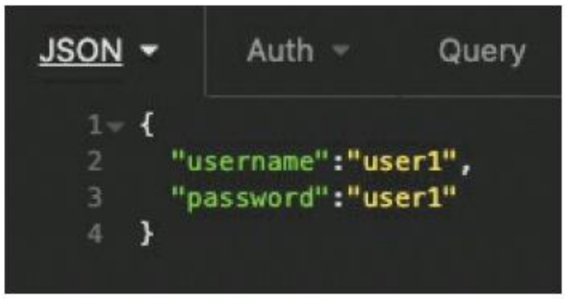
```
if request.method == 'POST':
    data = JSONParser().parse(request)
    user = authenticate(
        request,
        username=data['username'],
        password=data['password'])
    if user is None:
        return JsonResponse(
            {'error':'unable to login. check username and
        password'}, status=400)
    else: # return user token
        try:
            token = Token.objects.get(user=user)
        except: # if token not in db, create a new one
            token = Token.objects.create(user=user)
        return JsonResponse({'token':str(token)}, status=201)
```

Testing API

- You can download the Insomnia client from https://insomnia.rest then go to *http://localhost:8000/api/signup/* and do the following:

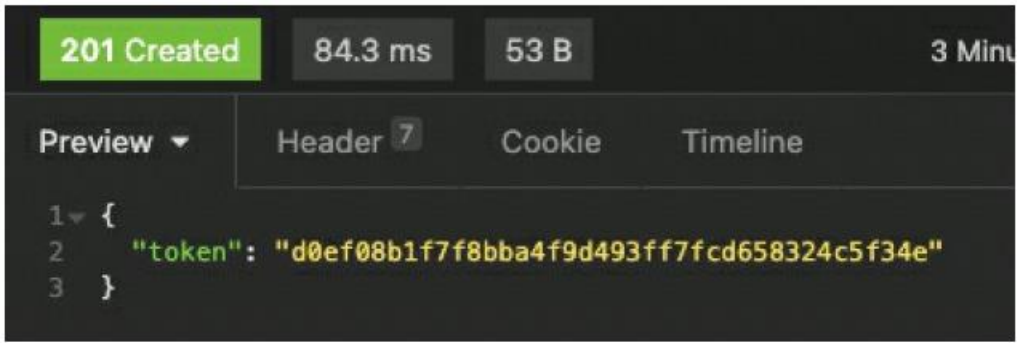


```
POST http://localhost:8000/api/signup/
```



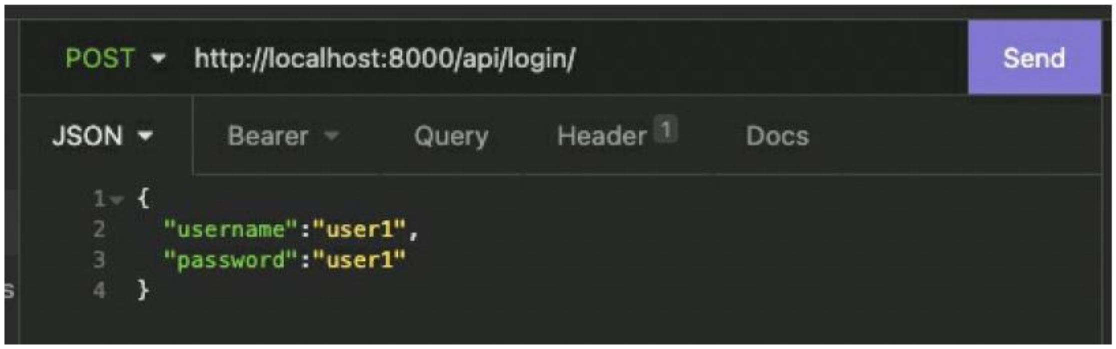
```
{
  "username":"user1",
  "password":"user1"
}
```

- When you hit send, the following should some up:



```
{
  "token": "d0ef08b1f7f8bba4f9d493ff7fcd658324c5f34e"
}
```

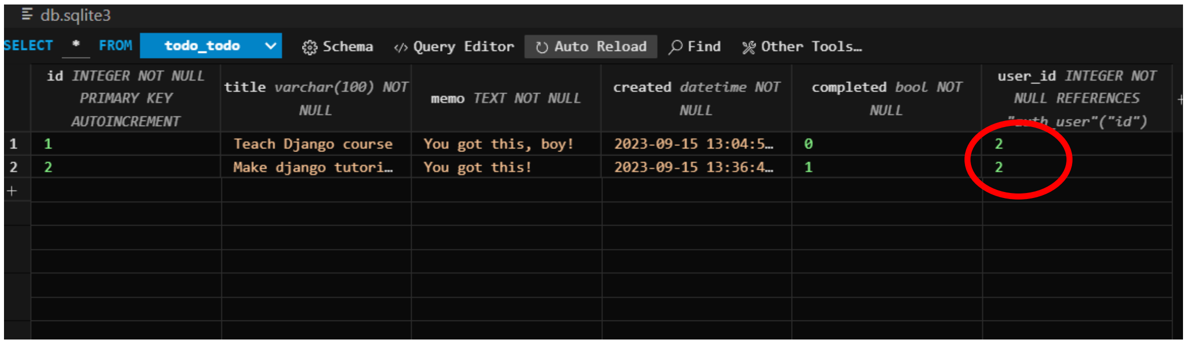
- To test the login go to Insomnia and run the following:



```
http://localhost:8000/api/login/
```

```
{
  "username":"user1",
  "password":"user1"
}
```

- After hitting send you should see the authorization token.
- Go to *db.sqlite3* and in the *todo_todo table* make the following change:

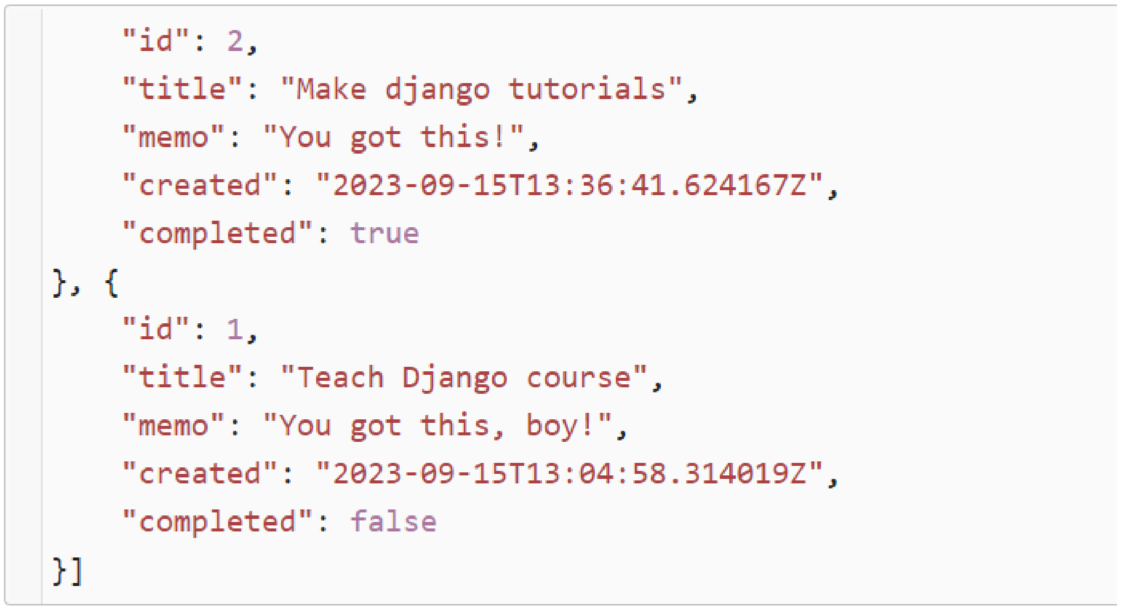


	id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT	title varchar(100) NOT NULL	memo TEXT NOT NULL	created datetime NOT NULL	completed bool NOT NULL	user_id INTEGER NOT NULL REFERENCES "auth_user"("id")
1	1	Teach Django course	You got this, boy!	2023-09-15 13:04:5…	0	2
2	2	Make django tutori…	You got this!	2023-09-15 13:36:4…	1	2

- Go to https://reqbin.com/curl and type the following command:

  curl http://127.0.0.1:8000/api/todos/ -H "Authorization: Token <**YOUR TOKEN>**"

- Click run, and you should see the following screen:



```
    "id": 2,
    "title": "Make django tutorials",
    "memo": "You got this!",
    "created": "2023-09-15T13:36:41.624167Z",
    "completed": true
}, {
    "id": 1,
    "title": "Teach Django course",
    "memo": "You got this, boy!",
    "created": "2023-09-15T13:04:58.314019Z",
    "completed": false
}]
```

Challenge

- Take you helloworld app and generate an API endpoint for it (for simplicity you can leave the authentication until the end).
- HINT: You do not need to delete your MVT controllers to do this.