バックエンド(Django)プロジェクト作り方

1. Anacondaを開き、左下の「create」ボタンを押す
2. 適当な名前を付け、Pythonを選択する(バージョンは適当)
3. 作成されたら、▷ボタンを押し「open terminal」を選択する
4. pip install django==3.0.7を実行
5. pip install djangorestframework==3.10を実行
6. pip install djangorestframework-simplejwt==4.6.0を実行
7. pip install djoserを実行
8. pip install django-cors-headersを実行
9. バックエンドのプロジェクトフォルダを作成する(名前は何でもいい)
10. PyCharmを起動する
11. File→Openとして、先ほど作ったフォルダを選択
12. Anacondaで作ったものと紐づけるために、File→SettingとしProjectのProject Interpreterを選択
13. 歯車マークを押し、「add」を選択
14. Existing environmentにチェックし、…ボタンをクリック
15. Anacondaで作ったプロジェクト(仮想環境)を選択し、python.exe(ファイル)を選択しOKを押し問題なければまたOKを押す
16. Terminalを開く(下らへんにある)
17. django-admin startproject 適当な名前 .　とterminalに打つ
18. manage.pyを右クリックから「Run manage」を実行
19. 右上のmanageから「Edit Configuratiors」を選択
20. パラメータズのところに、「runserver」とうち、OKを押す
21. 再生ボタンを押して、run serverを起動したり停止したりできるようになる
22. Terminalから「django-admin startapp 適当な名前」を実行
23. Anacondaで作ったプロジェクトのsetting.pyを開く
24. JWTを使用できるようにと有効期限を設定するため「from datetime import timedelta」を記載
25. INSTALLED\_APPSに4つ追記する

・'rest\_framework',

・'api.apps.ApiConfig',

・'corsheaders',

・'djoser',

1. MIDDLEWAREに1つ追記する

・'corsheaders.middleware.CorsMiddleware',

1. 下記を新しく作成(React側からアクセスできるようにするため)

・CORS\_ORIGIN\_WHITELIST = [

“http://localhost:3000”]

フロントエンド側でPORTの設定をいじってない場合は3000でOK

.envファイルにPORT=8000みたいな感じでいじってたら都度変更する

1. 下記を新しく追記する(認証関係の設定)

・REST\_FRAMEWOKE = {

‘DEFAULT\_PERMISSION\_CLASSES’:[

‘rest\_framework.permissions.IsAuthenticated’,

],

‘DEFAULT\_AUTHENTICATION\_CLASSES’:[

‘rest\_framework\_simplejwt.authenticatioon.JWTAuthentication’,

],

}

1. 下記を新しく追記する

・SIMPLE\_JWT = {

‘AUTH\_HEADER\_TYPES’: (‘JWT’,),

‘ACCESS\_TOKEN\_LIFETIME’: timedelta(minutes=30),

}

1. 下記を変更する

・TIME\_ZONE = ‘Asia/Tokyo’

1. 同じフォルダ内にあるurls.pyファイルを開く
2. 下記を追加

・from django.conf.urls import include

33.urlpatternsの中に下記を追加

・path(‘api/’, include(‘api.urls’)),

・path(‘authen/’, include((‘djoser.urls.jwt’)))

1. models.pyを開き、データベースの構造を作成する

・from django.db import models

class Task(models.Model):

title = models.CharField(max\_length=50)

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

def \_\_ str\_\_(self):

return self.title

1. terminalに下記を入力して実行

・python manage.py makemigrations

1. Create ～と出たら、下記を実行

・python manage.py migrate

1. admin.pyファイルに先ほど作成したモデルを登録する
2. 下記を追加する

・from .models import Task

admin.site.register(Task)

1. adminのページを見るためには、スーパーユーザーが必要なため作成する
2. Terminalに下記を記載し実行

・python manage.py createsuperuser

1. 適当に名前などつけ、最後はy応答
2. これでadminのurlに飛ぶことができる。確認する際はrunserverを起動させ、urlにアクセスする
3. apiフォルダに「serializers.py」を作成する
4. 下記を追加する

・from rest\_framework import serializers

・from .models import Task

・from django.contrib.auth.models import User

1. 下記を追加する

・class UserSerializer(serializers.ModelSerializer):

class Meta:

model = User

fields = (‘id’, ‘username’, ‘password’)

extra\_kwargs = {‘password’: { ‘write\_only’: True, ‘required’: True}}

def create(self, validated\_data):

user = User.objects.create\_user(\*\*validated\_data)

return user

class TaskSerializer(serializers.ModelSerializer):

created\_at = serializers.DateTimeField(format=”%Y-%m-%d %H;%M”, read\_only=True)

updated\_at=serializers.DateTimeField(format=”%Y-%m-%d %H;%M”, read\_only=True)

class Meta:

model = Task

fields = (‘id’, ‘title’, ‘created\_at’, ‘updated\_at’)

1. Views.pyを開き、下記を追加する

・from rest\_framework.permissions import AllowAny

・from rest\_framework import generics

・from rest\_framework import viewsets

・from .serializers import TaskSerializer, UserSerializer

・from .models import Task

1. 下記を追加する

・class CreateUserView(generics.CreateAPIView):

serializer\_class = UserSerializer

permission\_classes = (AllowAny,)

class MyProfileView(generics.RetrieveUpdateAPIView):

serializer\_class = UserSerializer

def get\_object(self):

return self.request.user

class TaskViewSet(viewsets.ModelViewSet):

queryset = Task.objects.all()

serializer\_class = TaskSerializer

1. api直下(admin.pyがあるフォルダ)のurls.pyを開く
2. 下記を追加する

from django.urls import path

from django.conf.urls import include

from rest\_framework import routers

from api.views import TaskViewSet, CreateUserView, MyProfileView

router = routers.DefaultRouter()

router.register(‘tasks’, TaskViewSet, basename=’tasks’)

urlpatterns = [

path(‘myself/’, MyProfileView.as\_view(), name=’myself’),

path(‘register/’, CreateUserView.as\_view(), name=’register’),

path(‘’, include(router.urls)),

]

1. サーバーを起動して、URLにアクセスする