Installing virtual environment for separation

pip install virtualenv

virtualenv env

env\scripts\activate

pip install Django

django-admin startproject notepad

move env folder into notepad folder

Command to start Server

python manage.py runserver

Command to create api app

python manage.py startapp api

go to settings.py to connect api to main app notepad

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'api.apps.AppConfig',

]

Api

Views.py

from django.shortcuts import render

from django.http import HttpResponse, JsonResponse

# Create your views here.

def getRoutes(request):

    routes = [

        {

            'Endpoint': '/notes/',

            'method': 'GET',

            'body': None,

            'description': 'Returns an array of notes'

        },

        {

            'Endpoint': '/notes/id',

            'method': 'GET',

            'body': None,

            'description': 'Returns a single note object'

        },

        {

            'Endpoint': '/notes/create/',

            'method': 'POST',

            'body': {'body': ""},

            'description': 'Creates new note with data sent in post request'

        },

        {

            'Endpoint': '/notes/id/update/',

            'method': 'PUT',

            'body': {'body': ""},

            'description': 'Creates an existing note with data sent in post request'

        },

        {

            'Endpoint': '/notes/id/delete/',

            'method': 'DELETE',

            'body': None,

            'description': 'Deletes and exiting note'

        },

    ]

    return JsonResponse(routes, safe=False)

urls.py

urlpatterns = [

    path('', views.getRoutes, name="routes"),

]

Connect urls file in urls.py

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', include('api.urls')),

]

after writing code in models.py

python manage.py makemigrations

python manage.py migrate

create user to access admin panel

> python manage.py createsuperuser

Username (leave blank to use 'hp'): toff

Email address: toffcodes07@gmail.com

Password: 123toff#

You need to import models and register them in admins.py (api) to use them in admin a

from django.contrib import admin

# Register your models here.

from .models import Note

admin.site.register(Note)

Django REST Framework

[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

...or clone the project from github.

git clone https://github.com/encode/django-rest-framework

Add 'rest\_framework' to your INSTALLED\_APPS setting.

INSTALLED\_APPS = [

...

'rest\_framework',

]

If you're intending to use the browsable API you'll probably also want to add REST framework's login and logout views. Add the following to your root urls.py file.

urlpatterns = [

...

path('api-auth/', include('rest\_framework.urls'))

]

Note that the URL path can be whatever you want.

For Styling API Response

from django.shortcuts import render

# from django.http import JsonResponse

from rest\_framework.response import Response

from rest\_framework.decorators import api\_view

# Create your views here.

@api\_view(['GET'])

def getRoutes(request):

    routes = [

        {

            'Endpoint': '/notes/',

            'method': 'GET',

            'body': None,

            'description': 'Returns an array of notes'

        },

        {

            'Endpoint': '/notes/id',

            'method': 'GET',

            'body': None,

            'description': 'Returns a single note object'

        },

        {

            'Endpoint': '/notes/create/',

            'method': 'POST',

            'body': {'body': ""},

            'description': 'Creates new note with data sent in post request'

        },

        {

            'Endpoint': '/notes/id/update/',

            'method': 'PUT',

            'body': {'body': ""},

            'description': 'Creates an existing note with data sent in post request'

        },

        {

            'Endpoint': '/notes/id/delete/',

            'method': 'DELETE',

            'body': None,

            'description': 'Deletes and exiting note'

        },

    ]

    return Response(routes)

This will Format API response

# Get Routes

**GET** /

**HTTP 200 OK**

**Allow:** GET, OPTIONS

**Content-Type:** application/json

**Vary:** Accept

[

{

"Endpoint": "/notes/",

"method": "GET",

"body": null,

"description": "Returns an array of notes"

},

{

"Endpoint": "/notes/id",

"method": "GET",

"body": null,

"description": "Returns a single note object"

},

{

"Endpoint": "/notes/create/",

"method": "POST",

"body": {

"body": ""

},

"description": "Creates new note with data sent in post request"

},

{

"Endpoint": "/notes/id/update/",

"method": "PUT",

"body": {

"body": ""

},

"description": "Creates an existing note with data sent in post request"

},

{

"Endpoint": "/notes/id/delete/",

"method": "DELETE",

"body": null,

"description": "Deletes and exiting note"

}

]

In order to fetch notes

Views.py

# this will fetch notes written in models

@api\_view(['GET'])

def getNotes(request):

    return Response('NOTES')

urls.py

from django.urls import path

from . import views

urlpatterns = [

    path('', views.getRoutes, name="routes"),

    path('notes/', views.getNotes, name="notes")

]

Serialise data means take python objects and turn it into JSON format

Create serializers.py

from dataclasses import field

from rest\_framework.serializers import ModelSerializer

from .models import Note

class NoteSerializer(ModelSerializer):

    class Meta:

        model = Note

        fields = '\_\_all\_\_'

        # fields=['body','updated']

Views.py

from tkinter import NO

from django.shortcuts import render

# from django.http import JsonResponse

from rest\_framework.response import Response

from rest\_framework.decorators import api\_view

from .models import Note

from .serializers import NoteSerializer

# Create your views here.

@api\_view(['GET'])

def getRoutes(request):

    routes = [

        {

            'Endpoint': '/notes/',

            'method': 'GET',

            'body': None,

            'description': 'Returns an array of notes'

        },

        {

            'Endpoint': '/notes/id',

            'method': 'GET',

            'body': None,

            'description': 'Returns a single note object'

        },

        {

            'Endpoint': '/notes/create/',

            'method': 'POST',

            'body': {'body': ""},

            'description': 'Creates new note with data sent in post request'

        },

        {

            'Endpoint': '/notes/id/update/',

            'method': 'PUT',

            'body': {'body': ""},

            'description': 'Creates an existing note with data sent in post request'

        },

        {

            'Endpoint': '/notes/id/delete/',

            'method': 'DELETE',

            'body': None,

            'description': 'Deletes and exiting note'

        },

    ]

    return Response(routes)

# this will fetch notes written in models

@api\_view(['GET'])

def getNotes(request):

    notes = Note.objects.all()  # query set

    serializer = NoteSerializer(notes, many=True)

    return Response(serializer.data)

127.0.0.1:8000/notes/

# Get Notes

**GET** /notes/

**HTTP 200 OK**

**Allow:** GET, OPTIONS

**Content-Type:** application/json

**Vary:** Accept

[

{

"id": 1,

"body": "Note#1\r\n-one\r\n-two\r\n-three",

"updated": "2022-06-26T14:05:52.971692Z",

"created": "2022-06-26T14:05:52.971692Z"

},

{

"id": 2,

"body": "Second Note",

"updated": "2022-06-26T14:06:13.188998Z",

"created": "2022-06-26T14:06:13.188998Z"

}

]

Getting Single Note

Views.py

# this will fetch single note written in models

@api\_view(['GET'])

def getNote(request, pk):

    notes = Note.objects.get(id=pk)  # query set

    serializer = NoteSerializer(notes, many=False)

    return Response(serializer.data)

urls.py

from django.urls import path

from . import views

urlpatterns = [

    path('', views.getRoutes, name="routes"),

    path('notes/', views.getNotes, name="notes"),

    path('notes/<str:pk>', views.getNote, name="note")

]

127.0.0.1:8000/notes/1

# Get Note

**GET** /notes/1

**HTTP 200 OK**

**Allow:** OPTIONS, GET

**Content-Type:** application/json

**Vary:** Accept

{

"id": 1,

"body": "Note#1\r\n-one\r\n-two\r\n-three",

"updated": "2022-06-26T14:05:52.971692Z",

"created": "2022-06-26T14:05:52.971692Z"

}

**Front end React Part**

npx create-react-app frontend

build front end accordingly

NotesListPage.js

import React, { useEffect, useState } from "react";

const NotesListPage = () => {

  const [notes, setNotes] = useState([]);

  useEffect(() => {

    getNotes();

  }, []);

  const getNotes = async () => {

    const response = await fetch("http://127.0.0.1:8000/api/notes/");

    const data = await response.json();

    setNotes(data);

    console.log(data);

  };

  return <div>NotesListPage</div>;

};

export default NotesListPage;

Cors error

Access to fetch at 'http://127.0.0.1:8000/api/notes/' from origin 'http://localhost:3000' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.

Github package to fix this error

[adamchainz/django-cors-headers: Django app for handling the server headers required for Cross-Origin Resource Sharing (CORS) (github.com)](https://github.com/adamchainz/django-cors-headers)

## Setup

Install from **pip**:

python -m pip install django-cors-headers

and then add it to your installed apps:

INSTALLED\_APPS = [

...,

"corsheaders",

...,

]

Make sure you add the trailing comma or you might get a ModuleNotFoundError (see [this blog post](https://adamj.eu/tech/2020/06/29/why-does-python-raise-modulenotfounderror-when-modifying-installed-apps/)).

You will also need to add a middleware class to listen in on responses:

MIDDLEWARE = [

...,

"corsheaders.middleware.CorsMiddleware",

"django.middleware.common.CommonMiddleware",

...,

]

CorsMiddleware should be placed as high as possible, especially before any middleware that can generate responses such as Django's CommonMiddleware or Whitenoise's WhiteNoiseMiddleware. If it is not before, it will not be able to add the CORS headers to these responses.

Also if you are using CORS\_REPLACE\_HTTPS\_REFERER it should be placed before Django's CsrfViewMiddleware (see more below).

**Configuration**

Configure the middleware's behaviour in your Django settings. You must set at least one of three following settings:

* CORS\_ALLOWED\_ORIGINS
* CORS\_ALLOWED\_ORIGIN\_REGEXES
* CORS\_ALLOW\_ALL\_ORIGINS

### CORS\_ALLOWED\_ORIGINS: Sequence[str]

A list of origins that are authorized to make cross-site HTTP requests. The origins in this setting will be allowed, and the requesting origin will be echoed back to the client in the [Access-Control-Allow-Origin header](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Origin). Defaults to [].

An Origin is defined by [the CORS RFC Section 3.2](https://tools.ietf.org/html/rfc6454#section-3.2) as a URI scheme + hostname + port, or one of the special values 'null' or 'file://'. Default ports (HTTPS = 443, HTTP = 80) are optional.

The special value null is sent by the browser in ["privacy-sensitive contexts"](https://tools.ietf.org/html/rfc6454#section-6), such as when the client is running from a file:// domain. The special value file:// is sent accidentally by some versions of Chrome on Android as per [this bug](https://bugs.chromium.org/p/chromium/issues/detail?id=991107).

Example:

CORS\_ALLOWED\_ORIGINS = [

"https://example.com",

"https://sub.example.com",

"http://localhost:8080",

"http://127.0.0.1:9000",

]

Previously this setting was called CORS\_ORIGIN\_WHITELIST, which still works as an alias, with the new name taking precedence.

### CORS\_ALLOWED\_ORIGIN\_REGEXES: Sequence[str | Pattern[str]]

A list of strings representing regexes that match Origins that are authorized to make cross-site HTTP requests. Defaults to []. Useful when CORS\_ALLOWED\_ORIGINS is impractical, such as when you have a large number of subdomains.

Example:

CORS\_ALLOWED\_ORIGIN\_REGEXES = [

r"^https://\w+\.example\.com$",

]

Previously this setting was called CORS\_ORIGIN\_REGEX\_WHITELIST, which still works as an alias, with the new name taking precedence.

### 

### CORS\_ALLOW\_ALL\_ORIGINS: bool

If True, all origins will be allowed. Other settings restricting allowed origins will be ignored. Defaults to False.

Setting this to True can be dangerous, as it allows any website to make cross-origin requests to yours. Generally you'll want to restrict the list of allowed origins with CORS\_ALLOWED\_ORIGINS or CORS\_ALLOWED\_ORIGIN\_REGEXES.

Previously this setting was called CORS\_ORIGIN\_ALLOW\_ALL, which still works as an alias, with the new name taking precedence.

CORS\_ALLOW\_ALL\_ORIGINS = True

Install React Router Dom

npm install react-router-dom

proxy url

just want to specify “/api/notes/”

Uncaught (in promise) SyntaxError: Unexpected token < in JSON at position 0

package.json

{

  "name": "frontend",

  "proxy": "http://127.0.0.1:8000",

  "version": "0.1.0",

  "private": true,

….

…

Now Restart server

React router v6 changes

[A guide to using React Router v6 in React apps - LogRocket Blog](https://blog.logrocket.com/react-router-v6/)

import { BrowserRouter as Router, Routes } from 'react-router-dom';

import "./App.css";

import Header from "./components/Header";

import NotesListPage from "./pages/NotesListPage";

import { BrowserRouter as Router, Routes, Route } from "react-router-dom";

import NotePage from "./pages/NotePage";

function App() {

  return (

    <Router>

      <div className="App">

        <Header></Header>

        <Routes>

          <Route path="/" element={<NotesListPage></NotesListPage>}></Route>

          <Route path="note/:id" element={<NotePage></NotePage>}></Route>

        </Routes>

      </div>

    </Router>

  );

}

export default App;

**Fetch Data**

**NoteListPage.js**

import React, { useEffect, useState } from "react";

import ListItem from "../components/ListItem";

const NotesListPage = () => {

  const [notes, setNotes] = useState([]);

  useEffect(() => {

    getNotes();

  }, []);

  const getNotes = async () => {

    const response = await fetch("/api/notes/");

    const data = await response.json();

    setNotes(data);

    // console.log(data);

  };

  const renderNotes = notes.map((note, index) => (

    <ListItem key={index} note={note}></ListItem>

  ));

  return (

    <div>

      <div className="notes-list">{renderNotes}</div>

    </div>

  );

};

export default NotesListPage;

**NotePage.js**

import React, { useEffect, useState } from "react";

import { useParams } from "react-router-dom";

const NotePage = ({ match }) => {

  const params = useParams();

  const noteId = params.id;

  const [note, setNote] = useState(null);

  useEffect(() => {

    getNote();

  }, [noteId]);

  const getNote = async () => {

    const response = await fetch(`/api/notes/${noteId}`);

    const data = await response.json();

    setNote(data);

  };

  return (

    <div>

      <p>{note?.body}</p>

    </div>

  );

};

export default NotePage;

ListItem.js

import React from "react";

import { Link } from "react-router-dom";

const ListItem = ({ note }) => {

  return (

    <Link to={`/note/${note.id}`}>

      <h3>{note.body}</h3>

    </Link>

  );

};

export default ListItem;

Some updates, I had problem at [1:20:30](https://www.youtube.com/watch?v=tYKRAXIio28&t=4830s), in react-router-dom version 6 there are a few difference now: \* A <Route> must be used as the child of <Routes> element If you get this error Matched leaf route at location "/" does not have an element. This means it will render an with a null value by default resulting in an "empty" page \* In V6, you can't use the component prop anymore. It was replaced in favor of element : it should be like this: <Route path="/" element={<Home />}></Route>

Show less

15

REPLY

3 REPLIES

[Ayesha Rian Tegally](https://www.youtube.com/channel/UCUdysknie2sNgx-gPzSo8jg)

### [Ayesha Rian Tegally](https://www.youtube.com/channel/UCUdysknie2sNgx-gPzSo8jg)

[1 month ago](https://www.youtube.com/watch?v=tYKRAXIio28&lc=UgzfrIrT5Fnyff5a2-94AaABAg.9UtkaD2_4Ju9aSY3LJjbuU)

How did you do it to pass the params to the NotePage.js please?

REPLY

[Traaash](https://www.youtube.com/channel/UCnQvHln5eBNiAkUlhhWnmLQ)

### [Traaash](https://www.youtube.com/channel/UCnQvHln5eBNiAkUlhhWnmLQ)

[1 month ago](https://www.youtube.com/watch?v=tYKRAXIio28&lc=UgzfrIrT5Fnyff5a2-94AaABAg.9UtkaD2_4Ju9ajtDTzOdbB)

[@Ayesha Rian Tegally](https://www.youtube.com/channel/UCUdysknie2sNgx-gPzSo8jg) If you don't have the fix yet, you need to modify

import React from "react";

import { useParams } from "react-router-dom";

const NotePage = ({ match }) => {

  const params = useParams();

  return (

    <div>

      <h1>Single Note {params.id} </h1>

    </div>

  );

};

export default NotePage;

The route itself should look like this:

<Route path="/note/:id" element = {<NotePage />}></Route> which should be wrapped in <Routes> </Routes> of course.

**Update Functionality**

Views.py

from tkinter import NO

from urllib import response

from django.shortcuts import render

# from django.http import JsonResponse

from rest\_framework.response import Response

from rest\_framework.decorators import api\_view

from .models import Note

from .serializers import NoteSerializer

# Create your views here.

@api\_view(['GET'])

def getRoutes(request):

    routes = [

        {

            'Endpoint': '/notes/',

            'method': 'GET',

            'body': None,

            'description': 'Returns an array of notes'

        },

        {

            'Endpoint': '/notes/id',

            'method': 'GET',

            'body': None,

            'description': 'Returns a single note object'

        },

        {

            'Endpoint': '/notes/create/',

            'method': 'POST',

            'body': {'body': ""},

            'description': 'Creates new note with data sent in post request'

        },

        {

            'Endpoint': '/notes/id/update/',

            'method': 'PUT',

            'body': {'body': ""},

            'description': 'Creates an existing note with data sent in post request'

        },

        {

            'Endpoint': '/notes/id/delete/',

            'method': 'DELETE',

            'body': None,

            'description': 'Deletes and exiting note'

        },

    ]

    return Response(routes)

# this will fetch notes written in models

@api\_view(['GET'])

def getNotes(request):

   notes = Note.objects.all().order\_by('-updated')  # reordering latest query set

    serializer = NoteSerializer(notes, many=True)

    return Response(serializer.data)

# this will fetch single note written in models

@api\_view(['GET'])

def getNote(request, pk):

    notes = Note.objects.get(id=pk)  # query set

    serializer = NoteSerializer(notes, many=False)

    return Response(serializer.data)

@api\_view(['PUT'])

def updateNote(request, pk):

    data = request.data  # request.body or request.POST() if not using REST F/W

    note = Note.objects.get(id=pk)

    serializer = NoteSerializer(instance=note, data=data)

    if serializer.is\_valid():

        serializer.save()

    return Response(serializer.data)

urls.py

from django.urls import path

from . import views

urlpatterns = [

    path('', views.getRoutes, name="routes"),

    path('notes/', views.getNotes, name="notes"),

    path('notes/<str:pk>/update/', views.updateNote, name="update-note"),

    path('notes/<str:pk>/', views.getNote, name="note")

]

**Front End of Update Functionality**

import React, { useEffect, useState } from "react";

import { useNavigate, useParams } from "react-router-dom";

import { ReactComponent as ArrowLeft } from "../assets/arrow-left.svg";

const NotePage = ({ match, history }) => {

  const params = useParams();

  const noteId = params.id;

  const navigate = useNavigate();

  const [note, setNote] = useState(null);

  const getNote = async () => {

    const response = await fetch(`/api/notes/${noteId}`);

    const data = await response.json();

    setNote(data);

  };

  const updateNote = async () => {

    fetch(`/api/notes/${noteId}/update/`, {

      method: "PUT",

      headers: {

        "Content-Type": "application/json",

      },

      body: JSON.stringify(note),

    });

  };

  const handleSubmit = () => {

    updateNote();

    navigate("/");

  };

  useEffect(() => {

    getNote();

  }, [noteId]);

  return (

    <div className="note">

      <div className="note-header">

        <h3>

          <ArrowLeft onClick={handleSubmit}></ArrowLeft>

        </h3>

      </div>

      <textarea

        onChange={(e) => setNote({ ...note, body: e.target.value })}

        defaultValue={note?.body}

      ></textarea>

    </div>

  );

};

export default NotePage;

**Delete Note Functionality**

**Views.py**

from tkinter import NO

from urllib import response

from django.shortcuts import render

# from django.http import JsonResponse

from rest\_framework.response import Response

from rest\_framework.decorators import api\_view

from .models import Note

from .serializers import NoteSerializer

# Create your views here.

@api\_view(['GET'])

def getRoutes(request):

    routes = [

        {

            'Endpoint': '/notes/',

            'method': 'GET',

            'body': None,

            'description': 'Returns an array of notes'

        },

        {

            'Endpoint': '/notes/id',

            'method': 'GET',

            'body': None,

            'description': 'Returns a single note object'

        },

        {

            'Endpoint': '/notes/create/',

            'method': 'POST',

            'body': {'body': ""},

            'description': 'Creates new note with data sent in post request'

        },

        {

            'Endpoint': '/notes/id/update/',

            'method': 'PUT',

            'body': {'body': ""},

            'description': 'Creates an existing note with data sent in post request'

        },

        {

            'Endpoint': '/notes/id/delete/',

            'method': 'DELETE',

            'body': None,

            'description': 'Deletes and exiting note'

        },

    ]

    return Response(routes)

# this will fetch notes written in models

@api\_view(['GET'])

def getNotes(request):

    notes = Note.objects.all().order\_by('-updated')  # reordering latest query set

    serializer = NoteSerializer(notes, many=True)

    return Response(serializer.data)

# this will fetch single note written in models

@api\_view(['GET'])

def getNote(request, pk):

    notes = Note.objects.get(id=pk)  # query set

    serializer = NoteSerializer(notes, many=False)

    return Response(serializer.data)

@api\_view(['PUT'])

def updateNote(request, pk):

    data = request.data  # request.body or request.POST() if not using REST F/W

    note = Note.objects.get(id=pk)

    serializer = NoteSerializer(instance=note, data=data)

    if serializer.is\_valid():

        serializer.save()

    return Response(serializer.data)

@api\_view(['DELETE'])

def deleteNote(request, pk):

    note = Note.objects.get(id=pk)

    note.delete()

    return Response("Note was Deleted")

**urls.py**

from django.urls import path

from . import views

urlpatterns = [

    path('', views.getRoutes, name="routes"),

    path('notes/', views.getNotes, name="notes"),

    path('notes/<str:pk>/update/', views.updateNote, name="update-note"),

    path('notes/<str:pk>/delete/', views.deleteNote, name="delete-note"),

    path('notes/<str:pk>/', views.getNote, name="note")

]

**NotePage.js**

import React, { useEffect, useState } from "react";

import { useNavigate, useParams } from "react-router-dom";

import { ReactComponent as ArrowLeft } from "../assets/arrow-left.svg";

const NotePage = ({ match, history }) => {

  const params = useParams();

  const noteId = params.id;

  const navigate = useNavigate();

  const [note, setNote] = useState(null);

  const getNote = async () => {

    const response = await fetch(`/api/notes/${noteId}`);

    const data = await response.json();

    setNote(data);

  };

  const updateNote = async () => {

    fetch(`/api/notes/${noteId}/update/`, {

      method: "PUT",

      headers: {

        "Content-Type": "application/json",

      },

      body: JSON.stringify(note),

    });

  };

  const deleteNote = async () => {

    fetch(`/api/notes/${noteId}/delete/`, {

      method: "DELETE",

      headers: {

        "Content-Type": "application/json",

      },

    });

    navigate("/");

  };

  const handleSubmit = () => {

    updateNote();

    navigate("/");

  };

  useEffect(() => {

    getNote();

  }, [noteId]);

  return (

    <div className="note">

      <div className="note-header">

        <h3>

          <ArrowLeft onClick={handleSubmit}></ArrowLeft>

        </h3>

        <button onClick={deleteNote}>DELETE</button>

      </div>

      <textarea

        onChange={(e) => setNote({ ...note, body: e.target.value })}

        defaultValue={note?.body}

      ></textarea>

    </div>

  );

};

export default NotePage;

Create Notes Functionality

<https://mumbleui.com/icons>

views.py

from tkinter import NO

from urllib import response

from django.shortcuts import render

# from django.http import JsonResponse

from rest\_framework.response import Response

from rest\_framework.decorators import api\_view

from .models import Note

from .serializers import NoteSerializer

from api import serializers

# Create your views here.

@api\_view(['GET'])

def getRoutes(request):

    routes = [

        {

            'Endpoint': '/notes/',

            'method': 'GET',

            'body': None,

            'description': 'Returns an array of notes'

        },

        {

            'Endpoint': '/notes/id',

            'method': 'GET',

            'body': None,

            'description': 'Returns a single note object'

        },

        {

            'Endpoint': '/notes/create/',

            'method': 'POST',

            'body': {'body': ""},

            'description': 'Creates new note with data sent in post request'

        },

        {

            'Endpoint': '/notes/id/update/',

            'method': 'PUT',

            'body': {'body': ""},

            'description': 'Creates an existing note with data sent in post request'

        },

        {

            'Endpoint': '/notes/id/delete/',

            'method': 'DELETE',

            'body': None,

            'description': 'Deletes and exiting note'

        },

    ]

    return Response(routes)

# this will fetch notes written in models

@api\_view(['GET'])

def getNotes(request):

    notes = Note.objects.all().order\_by('-updated')  # reordering latest query set

    serializer = NoteSerializer(notes, many=True)

    return Response(serializer.data)

# this will fetch single note written in models

@api\_view(['GET'])

def getNote(request, pk):

    notes = Note.objects.get(id=pk)  # query set

    serializer = NoteSerializer(notes, many=False)

    return Response(serializer.data)

@api\_view(['POST'])

def createNote(request):

    data = request.data

    note = Note.objects.create(

        body=data['body']

    )

    serializer = NoteSerializer(note, many=False)

    return Response(serializer.data)

@api\_view(['PUT'])

def updateNote(request, pk):

    data = request.data  # request.body or request.POST() if not using REST F/W

    note = Note.objects.get(id=pk)

    serializer = NoteSerializer(instance=note, data=data)

    if serializer.is\_valid():

        serializer.save()

    return Response(serializer.data)

@api\_view(['DELETE'])

def deleteNote(request, pk):

    note = Note.objects.get(id=pk)

    note.delete()

    return Response("Note was Deleted")

urls.py

from django.urls import path

from . import views

urlpatterns = [

    path('', views.getRoutes, name="routes"),

    path('notes/', views.getNotes, name="notes"),

    path('notes/create', views.createNote, name="create-note"),

    path('notes/<str:pk>/update/', views.updateNote, name="update-note"),

    path('notes/<str:pk>/delete/', views.deleteNote, name="delete-note"),

    path('notes/<str:pk>/', views.getNote, name="note")

]

React Frontend

NotePage.js

import React, { useEffect, useState } from "react";

import { useNavigate, useParams } from "react-router-dom";

import { ReactComponent as ArrowLeft } from "../assets/arrow-left.svg";

const NotePage = ({ match, history }) => {

  const params = useParams();

  const noteId = params.id;

  const navigate = useNavigate();

  const [note, setNote] = useState(null);

  const getNote = async () => {

    if (noteId === "new") return;

    const response = await fetch(`/api/notes/${noteId}`);

    const data = await response.json();

    setNote(data);

  };

const createNote = async () => {

    fetch(`/api/notes/create/`, {

      method: "POST",

      headers: {

        "Content-Type": "application/json",

      },

      body: JSON.stringify(note),

    });

  };

  const updateNote = async () => {

    fetch(`/api/notes/${noteId}/update/`, {

      method: "PUT",

      headers: {

        "Content-Type": "application/json",

      },

      body: JSON.stringify(note),

    });

  };

  const deleteNote = async () => {

    fetch(`/api/notes/${noteId}/delete/`, {

      method: "DELETE",

      headers: {

        "Content-Type": "application/json",

      },

    });

    navigate("/");

  };

 const handleSubmit = () => {

    console.log("NOTE:", note);

    if (noteId !== "new" && note.body !== "") {

      deleteNote();

    } else if (noteId !== "new") {

      updateNote();

    } else if (noteId === "new" && note !== null) {

      createNote();

    }

    navigate("/");

  };

  useEffect(() => {

    getNote();

  }, [noteId]);

  return (

    <div className="note">

      <div className="note-header">

        <h3>

          <ArrowLeft onClick={handleSubmit}></ArrowLeft>

        </h3>

        {noteId !== "new" ? (

          <button onClick={deleteNote}>DELETE</button>

        ) : (

          <button onClick={handleSubmit}>Done</button>

        )}

      </div>

      <textarea

        onChange={(e) => setNote({ ...note, body: e.target.value })}

        defaultValue={note?.body}

      ></textarea>

    </div>

  );

};

export default NotePage;

List Item Styling Title , Date , Content

import React from "react";

import { Link } from "react-router-dom";

const getTime = (note) => {

  return new Date(note.updated).toLocaleDateString();

};

const getTitle = (note) => {

  let title = note.body.split("\n")[0];

  if (title.length > 45) {

    return title.slice(0, 45);

  }

  return title;

};

let getContent = (note) => {

  let title = getTitle(note);

  let content = note.body.replaceAll("\n", " ");

  content = content.replaceAll(title, "");

  if (content.length > 45) {

    return content.slice(0, 45) + "...";

  } else {

    return content;

  }

};

const ListItem = ({ note }) => {

  return (

    <Link to={`/note/${note.id}`}>

      <div className="notes-list-item">

        <h3>{getTitle(note)}</h3>

        <p>

          <span>{getTime(note)}</span>

          {getContent(note)}

        </p>

      </div>

    </Link>

  );

};

export default ListItem;

Adding React App to Django Project

Move Frontend to Backend Folder

\notepad> cd frontend

\frontend> npm run build

That will create build folder inside frontend

Go to settings.py

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [

            BASE\_DIR / 'frontend/build'

        ],

        'APP\_DIRS': True,

STATIC\_URL = 'static/'

STATICFILES\_DIRS=[

    BASE\_DIR / 'frontend/build/static'

]

Urls.py

from re import template

from django.contrib import admin

from django.urls import path, include

from django.views.generic import TemplateView

urlpatterns = [

    path('admin/', admin.site.urls),

    path('api/', include('api.urls')),

    path('', TemplateView.as\_view(template\_name='index.html')),

]

Now React App is hosted on

Port 8000

Now to go frontend folder

npm start

if you make any changes here then it will reflect only in port 3000

and wont reflect changes in port 8000

in order to transcend changes you need to npm run build

hence you need to do npm run build for every changes you wish to see

App.js

import "./App.css";

import Header from "./components/Header";

import NotesListPage from "./pages/NotesListPage";

import { HashRouter as Router, Routes, Route } from "react-router-dom";

// import { BrowserRouter as Router, Routes, Route } from "react-router-dom";

import NotePage from "./pages/NotePage";

function App() {

  return (

    <Router>

      <div className="container ">

        <div className="app">

          <Header></Header>

          <Routes>

            <Route path="/" element={<NotesListPage></NotesListPage>}></Route>

            <Route path="/note/:id" element={<NotePage></NotePage>}></Route>

          </Routes>

        </div>

      </div>

    </Router>

  );

}

export default App;

NotePage.js

import React, { useEffect, useState } from "react";

import { useNavigate, useParams } from "react-router-dom";

import { ReactComponent as ArrowLeft } from "../assets/arrow-left.svg";

const NotePage = ({ match, history }) => {

  const params = useParams();

  const noteId = params.id;

  const navigate = useNavigate();

  const [note, setNote] = useState(null);

  let CSRF = document.cookie.slice(10);

  const getNote = async () => {

    if (noteId === "new") return;

    const response = await fetch(`/api/notes/${noteId}`);

    const data = await response.json();

    setNote(data);

  };

  const createNote = async () => {

    fetch(`/api/notes/create/`, {

      method: "POST",

      headers: {

        "Content-Type": "application/json",

      },

      body: JSON.stringify(note),

    });

  };

  const updateNote = async () => {

    fetch(`/api/notes/${noteId}/update/`, {

      method: "PUT",

      headers: {

        "Content-Type": "application/json",

        "X-CSRFToken": CSRF,

      },

      body: JSON.stringify(note),

    });

  };

  const deleteNote = async () => {

    fetch(`/api/notes/${noteId}/delete/`, {

      method: "DELETE",

      headers: {

        "Content-Type": "application/json",

      },

    });

    navigate("/");

  };

  const handleSubmit = () => {

    console.log("NOTE:", note);

    if (noteId !== "new" && note.body === "") {

      deleteNote();

    } else if (noteId !== "new") {

      updateNote();

    } else if (noteId === "new" && note.body !== null) {

      createNote();

    }

    navigate("/");

  };

  useEffect(() => {

    getNote();

  }, [noteId]);

  return (

    <div className="note">

      <div className="note-header">

        <h3>

          <ArrowLeft onClick={handleSubmit}></ArrowLeft>

        </h3>

        {noteId !== "new" ? (

          <button onClick={deleteNote}>DELETE</button>

        ) : (

          <button onClick={handleSubmit}>Done</button>

        )}

      </div>

      <textarea

        onChange={(e) => setNote({ ...note, body: e.target.value })}

        defaultValue={note?.body}

      ></textarea>

    </div>

  );

};

export default NotePage;

Logout from admin panel if facing forbidden POST DELETE PUT