Create Api folder

```python
from django.db import models

# Create your models here.

class Note(models.Model):
    body = models.TextField(null=True, blank=True)
    updated = models.DateTimeField(auto_now=True)
    created = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.body[0:50]
```

create a model

```python
from rest_framework.serializers import ModelSerializer
from .models import Note


class NoteSerializer(ModelSerializer):
    class Meta:
        model = Note
        fields = '__all__'

```

data to JSON format

Seriolise Model

from rest_framework.response import Response

is a good way to show JSON data which is serialised

```python
def getNotesList(request):
    notes = Note.objects.all().order_by('-updated')
    serializer = NoteSerializer(notes, many=True)
    return Response(serializer.data)
```

query from the database

create in Seriolizer.py used to JSONify all the data

return a JSON response (Rest framework)

# Work On Frontend Part

return a promise

```
let getNote = async () => {
    if (noteId === 'new') return
    let response = await fetch(`/api/notes/${noteId}/`)
    let data = await response.json()
    setNote(data)
}
```

When creating a new Note, It should not wait till 00. return initially

Backtick.

would call works because of proxy Url in package

would wait untill process is complete

CORS error — Django blocking React app.

```
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).

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

```
MIDDLEWARE = [
    ...,
    "corsheaders.middleware.CorsMiddleware",
    "django.middleware.common.CommonMiddleware",
    ...,
]
```

CORS_ALLOWED_ORIGION = [__]

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

```
    return (
        <div className="notes">
            <div className="notes-header">
                <h2 className="notes-title">&#9782; Notes</h2>
                <p className="notes-count">{notes.length}</p>
            </div>

            <div className="notes-list">
                {notes.map((note, index) => (
                    <ListItem key={index} note = {note} />
                ))}
            </div>
            <AddButton />
        </div>
    )
}

export default NotesListPage
```

Create a map

```jsx
function App() {
  return (
    <Router>
      <div className="container dark">
        <div className="app">
          <Header />
          <Route path="/" exact component={NotesListPage} />
          <Route path="/note/:id" component={NotePage} />
        </div>
      </div>
    </Router>
  );
}
```

Router

Dynamic.

import {Link} from 'react-router-dom'

```jsx
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
```

## Update Note Functionality

```jsx
    let updateNote = async () => {
        fetch(`/api/notes/${noteId}/`, {
            method: "PUT",
            headers: {
                'Content-Type': 'application/jso
            },
            body: JSON.stringify(note)
        })
    }
```

```jsx
    return (
        <div className="note" >
            <div className="note-header">
                <h3>
                    <ArrowLeft onClick={handleSubmit} />
                    Hello All
                </h3>
                {noteId !== 'new' ? (
                    <button onClick={deleteNote}>Delete</button>
                ) : (
                    <button onClick={handleSubmit}>Done</button>
                )}

            </div>
            <textarea onChange={(e) => { handleChange(e.target.value) }}
            value={note?.body}></textarea>
        </div>
    )
}

export default NotePage
```

```jsx
def updateNote(request, pk):
    data = request.data
    note = Note.objects.get(id=pk)
    serializer = NoteSerializer(instance=note, data=data)

    if serializer.is_valid():
        serializer.save()

    return serializer.data
```

Serializers.py

↓ pass the new Dat

```jsx
    let handleChange = (value) => {
        setNote(note => ({ ...note, 'body': value }))
        console.log('Handle Change:', note)
    }
```

# Delete Node

```
let deleteNote = async () => {
    fetch(`/api/notes/${noteId}/`, {
        method: 'DELETE',
        'headers': {
            'Content-Type': 'application/json'
        }
    })
    history.push('/')
}
```

```
let handleSubmit = () => {
    console.log('NOTE:', note)
    if (noteId !== 'new' && note.body == '') {
        deleteNote()
    } else if (noteId !== 'new') {
        updateNote()
    } else if (noteId === 'new' && note.body !== null) {
        createNote()
    }
    history.push('/')
}
```

```
let createNote = async () => {
    fetch(`/api/notes/`, {
        method: "POST",
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(note)
    })
}
```