

COACHING PROGRAM Week 7

React.js

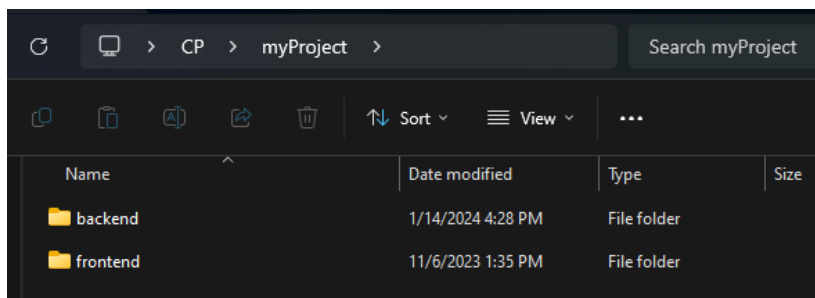
Bagian 4: Consume API Pribadi

Pengantar

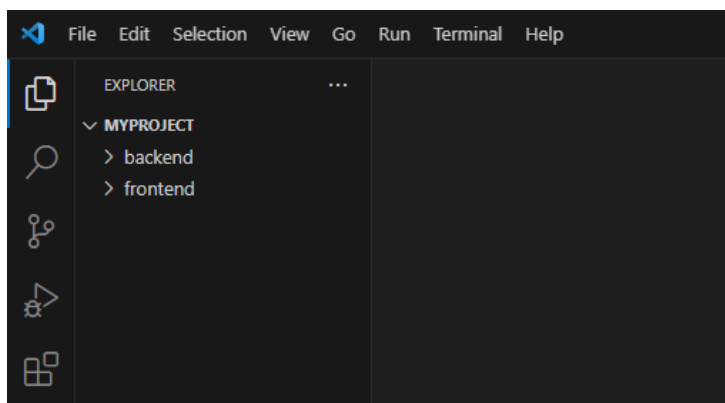
Setelah mengikuti Coaching Program pada week 3, 4, dan 5, kita akan lanjut untuk menyambungkan API yang telah dibuat pada pertemuan ke 5 seperti yang telah diajarkan pada pertemuan ke 3.

Setup Project

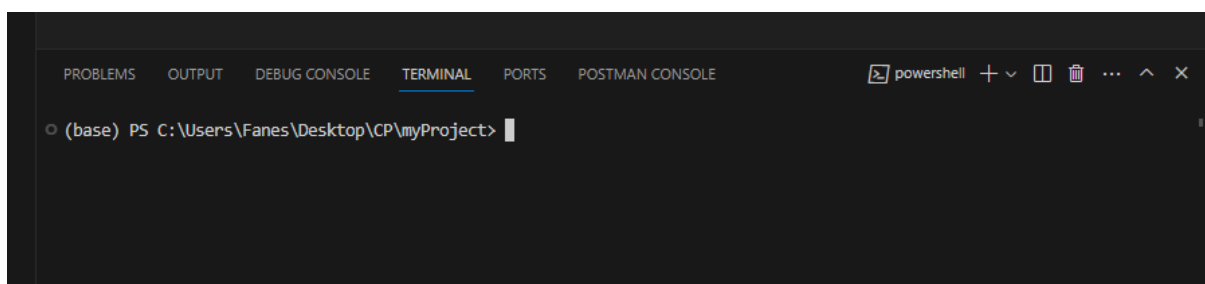
Saat ini, ada dua project terpisah yang kalian miliki, yaitu backend menggunakan strapi, dan frontend menggunakan reactjs.



Namai folder kalian seperti ini dan taruh di satu folder khusus agar memudahkan kalian dalam proses pengembangan. Setelah itu, buka vscode pada folder myProject ini, dengan cara salah satunya menarik folder myFolder ke vscode kalian, sehingga tampilannya seperti:



Buka terminal melalui menu Terminal > new terminal



COACHING PROGRAM Week 7

React.js

Lakukan cd ke directory backend, lalu jalankan `npm run develop`

```
(base) PS C:\Users\Fanes\Desktop\CP\myProject> cd backend
(base) PS C:\Users\Fanes\Desktop\CP\myProject\backend> npm run develop

> backend@0.1.0 develop
> strapi develop

Building your admin UI with development configuration...
Admin UI built successfully

Project information

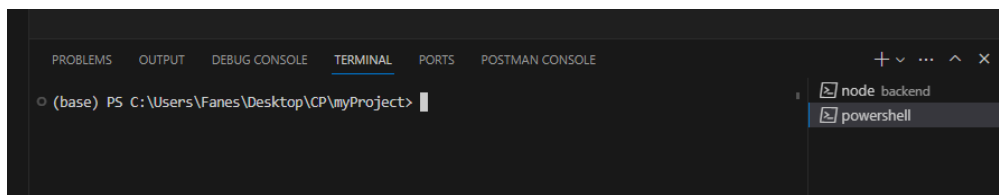
Time      Mon Jan 15 2024 13:57:28 GMT+0700 (Indochina Ti...
Launched in 1169 ms
Environment development
Process PID 23908
Version 4.15.0 (node v18.17.0)
Edition Community
Database sqlite

Actions available

Welcome back!
To manage your project 🚀, go to the administration panel at:
http://localhost:1337/admin

To access the server ⚡, go to:
http://localhost:1337
```

Setelah itu, kita akan menjalankan terminal kedua untuk bagian frontend, dengan cara ke menu Terminal > new terminal, atau bisa kalian klik logo + pada terminal saat ini.



Seperti sebelumnya, lakukan cd ke folder frontend, lalu jalankan `npm run dev`

```
(base) PS C:\Users\Fanes\Desktop\CP\myProject> cd frontend
(base) PS C:\Users\Fanes\Desktop\CP\myProject\frontend> npm run dev

> projects@0.0.0 dev
> vite

VITE v4.5.0 ready in 860 ms
→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h to show help
```

Sampai sini, kita telah melakukan setup terhadap fullstack project. Selanjutnya kita akan melakukan fetching dan mengelola data terhadap API yang telah kalian buat.

Sebelum itu, kalian bisa melakukan cek terhadap API secara manual, dengan mengakses ke URL localhost yang muncul pada terminal kalian sebelumnya, di kasus ini, <http://localhost:1337>, lalu akses path /api/namaCollection, jika mengikuti API yang telah kita buat sebelumnya, yaitu todos, maka <http://localhost:1337/api/todos>.

COACHING PROGRAM Week 7

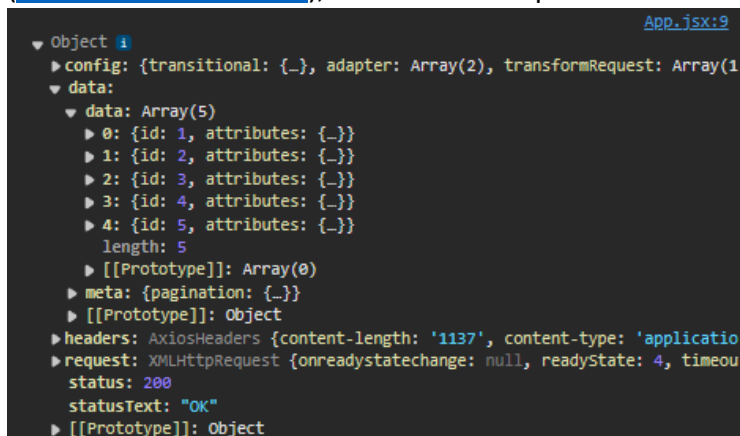
React.js

Fetching API

Kalian bebas ingin melakukan fetching API terhadap file yang terpisah atau langsung di App.jsx, karena sampai sini, seharusnya kalian sudah paham terkait konsep import-export react component dan routing.

```
function App() {  
  
  const fetchData = async () => {  
    const res = await axios.get('http://localhost:1337/api/todos')  
    console.log(res);  
  }  
  
  useEffect(() => {  
    fetchData();  
  }, []);  
  
  return (  
    <div>  
  
    </div>  
  )  
}
```

Dari kode diatas, bisa kita cek melalui console browser saat kalian mengakses URL frontend (<http://localhost:5173/>), akan muncul seperti ini



```
Object {  
  config: {transitional: {_, adapter: Array(2), transformRequest: Array(1)}, data: {  
    data: Array(5) {  
      0: {id: 1, attributes: {_}}  
      1: {id: 2, attributes: {_}}  
      2: {id: 3, attributes: {_}}  
      3: {id: 4, attributes: {_}}  
      4: {id: 5, attributes: {_}}  
      length: 5  
    },  
    meta: {pagination: {_}}  
    [[Prototype]]: Array(0)  
  },  
  headers: AxiosHeaders {content-length: '1137', content-type: 'applicatio  
  request: XMLHttpRequest {onreadystatechange: null, readyState: 4, timeou  
    status: 200  
    statusText: "OK"  
    [[Prototype]]: Object  
  }  
}
```

Dapat kita lihat pada result.data.data memiliki data array sebanyak 5, karena pada dashboard strapi, saya memiliki 5 todo entries. Berikut contoh data detailnya:

COACHING PROGRAM Week 7

React.js

```
▼ data:
  ▼ data: Array(5)
    ▼ 0:
      ▼ attributes:
        createdAt: "2023-10-26T09:29:04.943Z"
        description: "Buy groceries and prepare dinner"
        isDone: false
        publishedAt: "2023-10-26T09:29:05.729Z"
        updatedAt: "2023-10-26T09:29:05.731Z"
        ► [[Prototype]]: Object
      id: 1
      ► [[Prototype]]: Object
```

Sampai sini, kalian sudah berhasil melakukan fetching terhadap API yang telah kalian buat, dan telah terhubung terhadap frontend kalian. Karena Strapi itu memiliki konsep GraphQL API, kalian dapat melakukan filtering data melalui URL param, tapi di coaching program kali ini, kita tidak akan mempelajari hal tersebut. Kita akan lebih berfokus terhadap basic fetching dan filtering oleh frontend.

Displaying Data

Selanjutnya, kita akan membuat tampilan sederhana yang menampilkan data todo dari API yang telah kita fetching seperti ini:

My Todo List

<input type="checkbox"/>	Buy groceries and prepare dinner
<input checked="" type="checkbox"/>	Finish reading the last chapter of the book
<input checked="" type="checkbox"/>	Pay the monthly bills online
<input type="checkbox"/>	Water the plants and tend to the garden
<input checked="" type="checkbox"/>	Send a birthday message to a family member

Pertama, kita siapkan komponen parent yang nantinya akan memanggil komponen TodoCard sebagai child.

COACHING PROGRAM Week 7

React.js

```
function App() {  
  const [todos, setTodos] = useState([]);  
  
  const fetchData = async () => {  
    const res = await axios.get('http://localhost:1337/api/todos')  
    setTodos(res.data.data);  
  }  
  
  useEffect(() => {  
    fetchData();  
  }, []);  
  
  return (  
    <div className='w-50 p-5'>  
      <h3>My Todo List</h3>  
  
    </div>  
  )  
}
```

Disini, data yang kita fetching tadi akan disimpan pada react state menggunakan useState seperti yang telah dipelajari pada pertemuan kedua.

```
<div className='w-50 p-5'>  
  
  <h3>My Todo List</h3>  
  
  {todos.map((item, index) => (  
    <div key={index}>  
      {item.attributes.description}  
    </div>  
  ))}  
  
</div>
```

Selanjutnya, variable state tadi kita lakukan mapping data untuk menampilkan list deskripsi

My Todo List

Buy groceries and prepare dinner
Finish reading the last chapter of the book
Pay the monthly bills online
Water the plants and tend to the garden
Send a birthday message to a family member

Karena kita ingin membuat komponen terpisah untuk todo cardnya, maka kita lakukan setup terhadap TodoCard komponen seperti ini:

COACHING PROGRAM Week 7

React.js

```
const TodoCard = ({ data }) => {  
  return (  
    <div className="d-flex gap-2 border p-2">  
      <div className="my-auto">  
        <input type="checkbox" checked={data.attributes.isDone} />  
      </div>  
      <div className="p-1">  
        {data.attributes.description}  
      </div>  
    </div>  
  )  
}  
  
export default TodoCard
```

Kode diatas dapat dilihat [disini](#).

Dari kode diatas, data di passing melalui property TodoCard melalui kode `{{ data }}`, sehingga saat ingin kita akses melalui parent, kita akan melakukan passing data seperti ini:

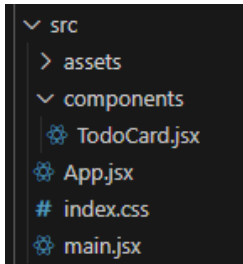
```
<div className='w-50 p-5'>  
  <h3>My Todo List</h3>  
  {todos.map((item, index) => (  
    <div key={index}>  
      <TodoCard data={item} />  
    </div>  
  ))}  
</div>
```

Note: jangan lupa melakukan import terhadap komponen TodoCard.

Pastikan format file project kalian seperti ini, agar memudahkan proses development.

COACHING PROGRAM Week 7

React.js



Sampai disini, kalian telah berhasil melakukan fetching API dan menampilkannya menggunakan konsep reusable component dari react, karena komponen TodoCard hanya perlu ditulis 1x namun bisa menampilkan banyak data melalui proses mapping tersebut.

My Todo List

<input type="checkbox"/>	Buy groceries and prepare dinner
<input checked="" type="checkbox"/>	Finish reading the last chapter of the book
<input checked="" type="checkbox"/>	Pay the monthly bills online
<input type="checkbox"/>	Water the plants and tend to the garden
<input checked="" type="checkbox"/>	Send a birthday message to a family member

Bagian 4: **CRUD**

Apa itu CRUD?

CRUD atau Create Read Update Delete adalah istilah yang biasa digunakan dalam pengembangan website. Sesuai dengan namanya, suatu web yang memiliki fitur CRUD adalah website yang dapat melakukan:

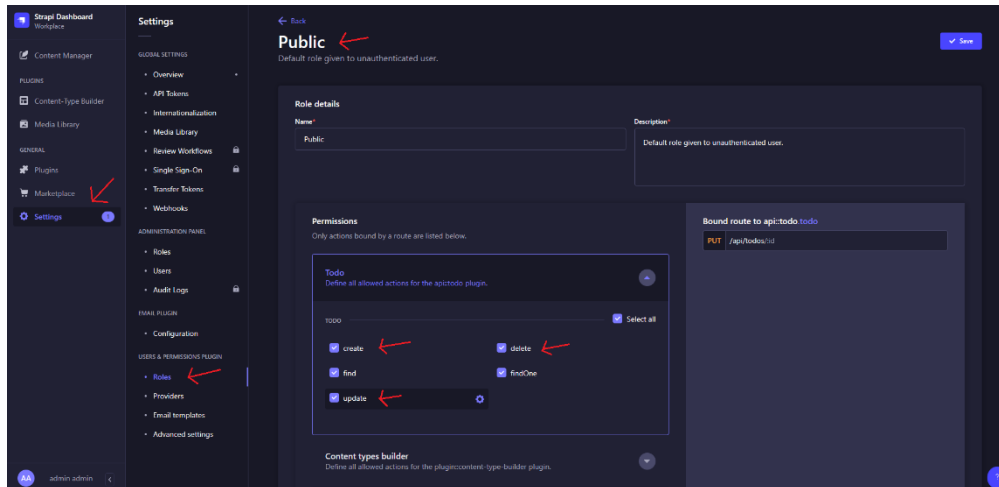
- Create atau penambahan terhadap data baru ke database
- Read atau membaca data dari database dan disampaikan kepada user
- Update atau memperbaharui data yang sudah ada
- Delete atau menghapus data yang sudah ada

Strapi sendiri sudah memiliki fitur built-in CRUD sehingga memudahkan proses pengembangan tanpa harus melalui codingan yang rumit disisi backend.

COACHING PROGRAM Week 7

React.js

Setup Backend



Seperti yang sudah dicontohkan pada pertemuan ke-5, disini kita hanya perlu mengubah permission yang hanya find dan findOne saja, menjadi find, findOne, create, delete, dan update. Lalu klik Save dan API siap digunakan untuk fungsi CRUD.

Jika kita lihat pada panel disisi kanan foto diatas, disitu disebutkan metode fetching yang diperlukan untuk menggunakan fitur tersebut (klik logo pengaturan di samping permissionnya).

- Untuk **create** menggunakan metode **POST** terhadap route `/api/todos`
- Untuk **read** menggunakan metode **GET** terhadap route `/api/todos` dan `/api/todos/:id`
- Untuk **update** menggunakan metode **PUT** terhadap route `/api/todos/:id`
- Untuk **delete** menggunakan metode **DELETE** terhadap route `/api/todos/:id`

Metode inilah yang perlu dicatat oleh sisi frontend untuk melakukan aksi CRUD ini.

Melakukan CRUD

Untuk melakukan CRUD disisi frontend sangatlah mudah, terlebih karena kita sudah menerapkan module Axios. Jika kita lihat pada fungsi fetching yang telah kita lakukan:

```
const fetchData = async () => {
  const res = await axios.get('http://localhost:1337/api/todos')
  setTodos(res.data.data);
}
```

Disitu terhadap sintaks `axios.get()` yang artinya fetching tersebut menggunakan metode GET. Jika kita ingin menggunakan metode sesuai yang telah kita lihat pada sisi backend tadi, kita tinggal mengubahnya menjadi `axios.post()`, `axios.delete()`, dan `axios.put()`. Perlu diingat bahwa metode seperti POST dan PUT bersifat memberikan data dari frontend, sehingga selain menaruh URL sebagai parameter: `axios.post("URL-backend")`, kita perlu

COACHING PROGRAM Week 7

React.js

menambahkan parameter kedua yang berbentuk Object data: `axios.post("URL-backend", { data: mydata })`. Berikut penjelasan secara detail:

Create

Untuk melakukan create, pertama kita akan membuat sebuah komponen baru yang bernama `AddTodo.jsx` (untuk nama komponen itu bebas). Selanjutnya kita membutuhkan state, handler, dan komponen input

```
const [description, setDescription] = useState('')

const handleSetDescription = (e) => {
  setDescription(e.target.value)
}
```

```
<input
  value={description}
  onChange={handleSetDescription}
/>
```

Setelah itu, kita akan membuat fungsi handler untuk melakukan POST terhadap API

```
const handleSubmit = async () => {
  const res = await axios.post('http://localhost:1337/api/todos', {
    data: {
      description: description,
      isDone: false
    }
  })
}
```

Seperti yang sudah dijelaskan diatas, disini method POST membutuhkan parameter kedua yang berupa object data baru yang ingin ditambahkan. Format data ini berbeda-beda tergantung backend developer. Namun di Strapi, dijelaskan pada [website dokumen ini](#), bahwa formatnya semua field (description dan isDone) harus dibungkus oleh property "data". Sehingga format parameter keduanya seperti pada gambar diatas.

Lalu tinggal kita terapkan pada button submit

```
<button onClick={handleSubmit}>
  Submit
</button>
```

Dan untuk memperbagus, kita tambahkan info terhadap hasil dari submitan

COACHING PROGRAM Week 7

React.js

```
const handleSubmit = async () => {
  const res = await axios.post('http://localhost:1337/api/todos', {
    data: {
      description: description,
      isDone: false
    }
  })
  if (res.status === 200) {
    alert('Todo baru berhasil ditambahkan!')
  } else {
    alert('Todo baru gagal ditambahkan!')
  }
}
```

Property “status” dari result fetching memiliki kode-kode khusus seperti 200 yang artinya berhasil, 400 artinya bad request, 401 artinya tidak memiliki hak akses, 500 artinya internal server error, dll yang dapat dilihat pada [web ini](#).

Untuk code AddTodo.jsx secara keseluruhan, dapat dilihat [disini](#).

Setelah komponen AddTodo selesai, dapat kita panggil di file utama kita

```
<div className='mt-5'>
  <h3>Add Todo</h3>
  <AddTodo />
</div>
```

Kalian bisa tes dengan menginput deskripsi dan menekan tombol submit, jika terjadi error, pastikan data yang disubmit itu memiliki format yang sama dengan nama field yang kalian input.

Delete

Selanjutnya, kita akan melakukan penghapusan data tertentu dengan menambahkan button hapus di setiap komponen todo card. Jadi pada komponen TodoCard.jsx, kita tambahkan:

```
<button
  className="ms-auto"
  onClick={handleDelete}
>
  Delete
</button>
```

Lalu kita buat fungsi baru yang bernama handleDelete

COACHING PROGRAM Week 7

React.js

```
const handleDelete = async () => {  
  const res = await axios.delete(`http://localhost:1337/api/todos/${data.id}`)  
  if (res.status === 200) {  
    alert('Berhasil menghapus data!')  
  } else {  
    alert('Gagal menghapus data!')  
  }  
}
```

Sampai disini, kalian telah berhasil membuat fungsi Create, Read, dan Delete untuk TodoApp ini. Untuk keseluruhan kode TodoCard.jsx dapat kalian lihat [disini](#).

Untuk Update, dapat kalian kerjakan secara mandiri sebagai tugas minggu ini.