Strapi

What is Strapi?

- Strapi is a free and open-source content management system, it is a headless CMS that allows you to create a website taking care of backend and give us control over the frontend.
- Strapi provides restful API for your website.
- we can use fetch/axio to fetch data from Strapi.

Install Strapi

- `npx` stands for node package execute, an npm package runner that can execute packages installed with npm.

```
npx create-strapi-app course-feedback
```

Project Over View

- FeedBack Website
  - User can give feedback
  - User can see their feedback
  - Feedback can be sorted by category
- Tech
  - react
  - graphql
  - strapi

Content Type and Endpoints

- Content Type: blueprint for a piece of content what fields should the content have and what data type should it be
- Collection Type: this is when we create a collection of content type, like collection of comments, feedbacks
- single Type: a homepage that has single
- components Type: a component is a reusable piece of code that we can use in our website

Feedback Content Type

- strapi automatically pluralizes the content type name
- feedback
- fields:
  - title: text
  - rating: number
  - body: richtext

Content and API

- When a content is created strapi automatically creates an endpoint for it, for example, if we create a feedback, we will have a feedback endpoint.
- the api route will be /api/feedback and the endpoint will be /feedback
- the endpoint will be stored in folder /src/api/feedback
- the schema for the endpoint will be stored in folder /src/content-types/feedback/schema.json

Creating react front end

- add react project `npx create-react-app frontend`
- add router `npm i react-router-dom`

run frontend and backend

- `npm run develop` in backend
- `npm start` in frontend
- should start strapi and react
- strapi on `http://localhost:1337/`
- react on `http://localhost:3000/`

React

- build routes for frontend

```
import Feedbacks from './pages/Feedbacks';
export default function App() {
  return (
    <Router>
      <Switch>
        <Route path="/feedbacks">
          <Feedbacks />
        </Route>
        <Route path="/">
          <Home />
        </Route>
      </Switch>
```

fetch and render data

- fetch data from strapi using end point `/feedbacks`

```
export default function Feedbacks() {
  const [feedbacks, setFeedbacks] = useState([]);
  useEffect(() => {
    axios.get('http://localhost:1337/feedbacks').then((res) => {
      setFeedbacks(res.data);
    });
  }, []);
  return (
    <div>
      <h1>Feedbacks</h1>
      {feedbacks.map((feedback) => (
        <div key={feedback.id}>
          <Link to={`/feedbacks/${feedback.id}`}>
            <h2>{feedback.title}</h2>
          </Link>
          <p>{feedback.body}</p>
        </div>
      ))}
    </div>
  );
}
```

specific feedback using useParams

- useParams is a hook that allows us to get the id from the url
  `const { id } = useParams();` // id is the id from the url

```
export default function Feedback() {
  const { id } = useParams();
  const [feedback, setFeedback] = useState({});
  useEffect(() => {
    axios.get(`http://localhost:1337/feedbacks/${id}`).then((res) => {
      setFeedback(res.data);
    });
  }, [id]);
  return (
    <div>
      <h1>{feedback.title}</h1>
      <p>{feedback.body}</p>
    </div>
  );
}
```

Graphql

- graphql is a query language for api
- its different from rest api because it allows us to get specific data from the api
- sample graphql user data query vs get all users

```
// graphql
query {
  users {
    id
    username
    email
  }
} // get all users and returns only id, username and email
// rest api
/users // get all users and returns all data
```

Installing graphql for strapi

- on admin panel go to
- marketplace -> graphql or `npm i strapi-plugin-graphql`

Setting up react to use graphql

- `npm i @apollo/client graphql` appolo client will help us to use graphql in react
- add appolo to app.js and create connection to graphql
- ```
  import { ApolloClient, InMemoryCache, ApolloProvider } from '@apollo/client';
  ```
  - `ApploClient` creates a new connection to graphql server
  - `InMemoryCache` is a cache that stores data from graphql server
  - `ApolloProvider` is a provider that allows us to use graphql in react
  ```

adding Appolo to our react app

- `uri: 'http://localhost:1337/graphql',` // graphql endpoint
- `cache: new InMemoryCache(),` // cache
- `ApolloProvider` // to provide graphql to react

```
import { ApolloClient, InMemoryCache, ApolloProvider } from '@apollo/client';

const client = new ApolloClient({
  uri: 'http://localhost:1337/graphql',
  cache: new InMemoryCache(),
});
return (
  <ApolloProvider client={client}> // client connection now provided to entire app
    <Router>
      <Switch>
        <Route path="/feedbacks/:id">
          <Feedback />
        </Route>
        ...
      </Switch>
    </Router>
  </ApolloProvider>
);
```

using appolo client in react

- `import { useQuery, gql } from '@apollo/client';` // useQuery is a hook that allows us to use graphql in react
- gql request

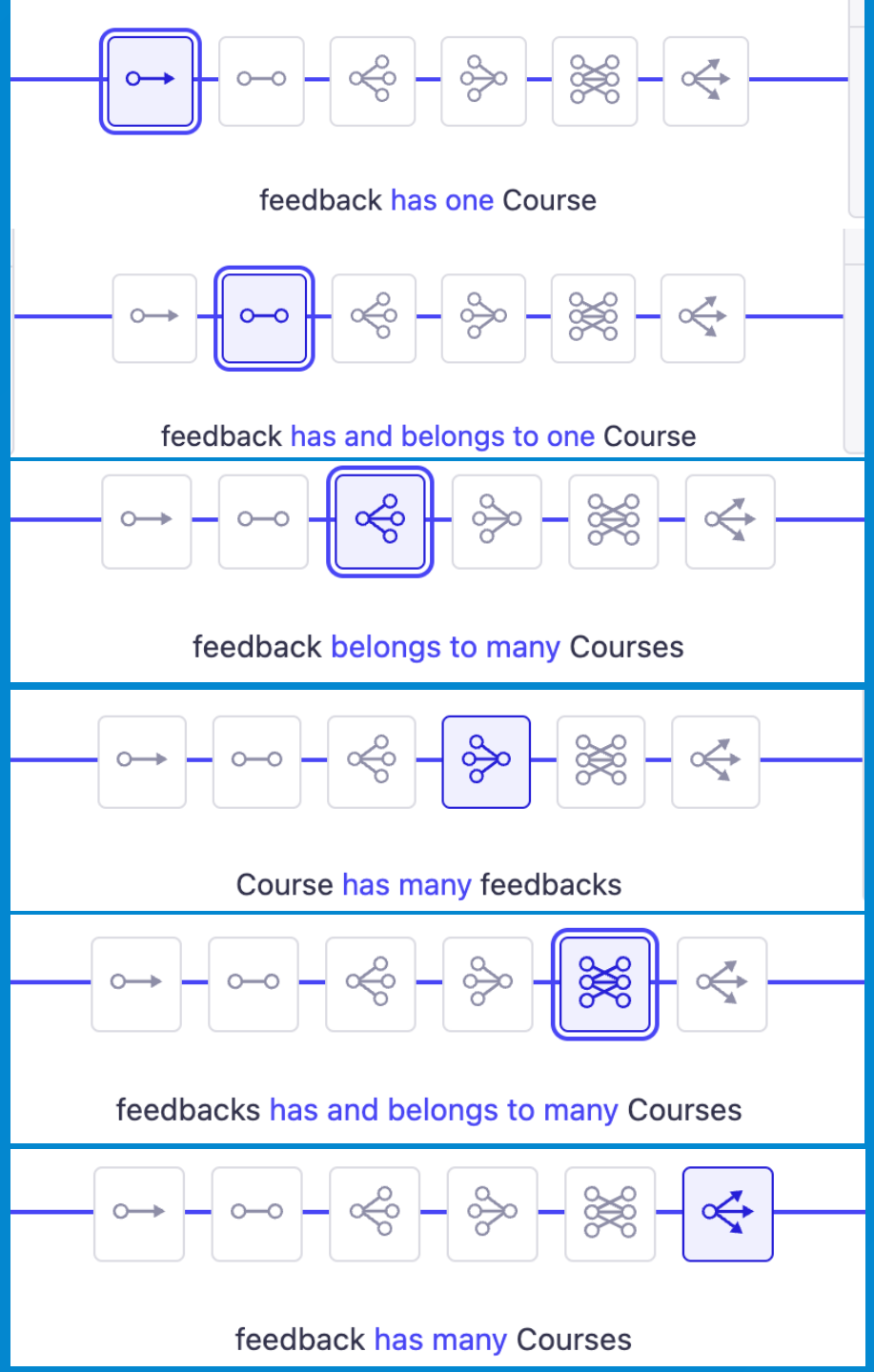
```
import { useQuery, gql } from '@apollo/client';
const FEEDBACKS_QUERY = gql`
  query {
    feedbacks {
      id
      title
      body
    }
  }
```

requesting specific feedback

```
const FEEDBACK_QUERY = gql
  query Feedback($id: ID!) {
    feedback(id: $id) {
      id
      title
      body
    }
  }
;
const { loading, error, data } = useQuery(FEEDBACK_QUERY, {
  variables: { id },
});
```

Relational data!

- relational data is data that is related to another data
- we will related feedback to courses



types of relations

- one to one
- one to one bidirectional
- one to many
- many to one
- many to many
- one has many

Getting courses data

- add courses on site header

react markdown for styles

- `npm i react-markdown` // to render markdown in react

```
import ReactMarkdown from 'react-markdown';
<ReactMarkdown>{feedback.body}</ReactMarkdown>
```