React + Node.js + Express + MySQL example: Build a CRUD App

Last modified: September 23, 2022 (https://www.bezkoder.com/react-node-express-mysql/) bezkoder (https://www.bezkoder.com/author/bezkoder/) Full Stack (https://www.bezkoder.com/category/full-stack/), Node.js (https://www.bezkoder.com/category/node-js/), React (https://www.bezkoder.com/category/react/)

In this tutorial, I will show you how to build full-stack React + Node.js + Express + MySQL example with a CRUD Application. The back-end server uses Node.js + Express for REST APIs, front-end side is a React.js client with React Router, Axios & Bootstrap.

Related Posts:
– React Redux + Node.js + Express + MySQL example: Build a CRUD App (https://bezkoder.com/react-redux-mysql-crud/)
– React + Node.js Express: Login example with JWT (https://bezkoder.com/react-express-authentication-jwt/)
– React File Upload with Axios and Progress Bar to Rest API (https://bezkoder.com/react-file-upload-axios/)

Run both projects in one place:
How to integrate React with Node.js Express on same Server/Port (https://bezkoder.com/integrate-react-express-same-server-port/)

Dockerize: Docker Compose: React, Node.js, MySQL example (https://www.bezkoder.com/docker-compose-react-nodejs-mysql/)

Contents [hide]

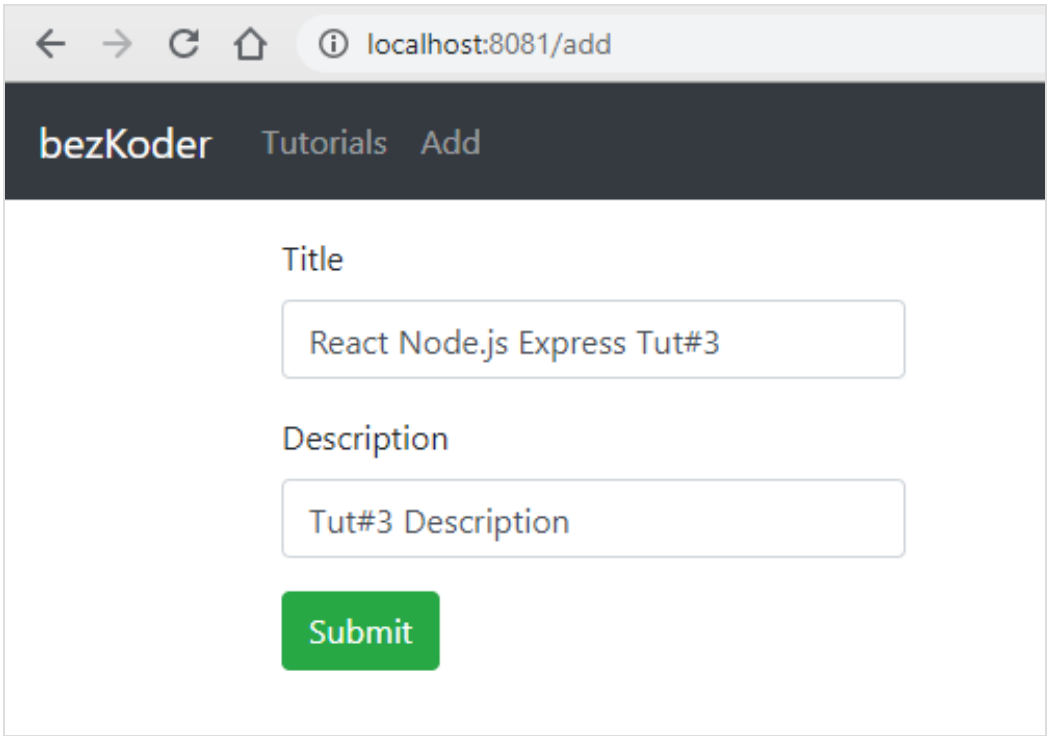
React + Node.js + Express + MySQL example Overview

We will build a full-stack Tutorial Application in that:

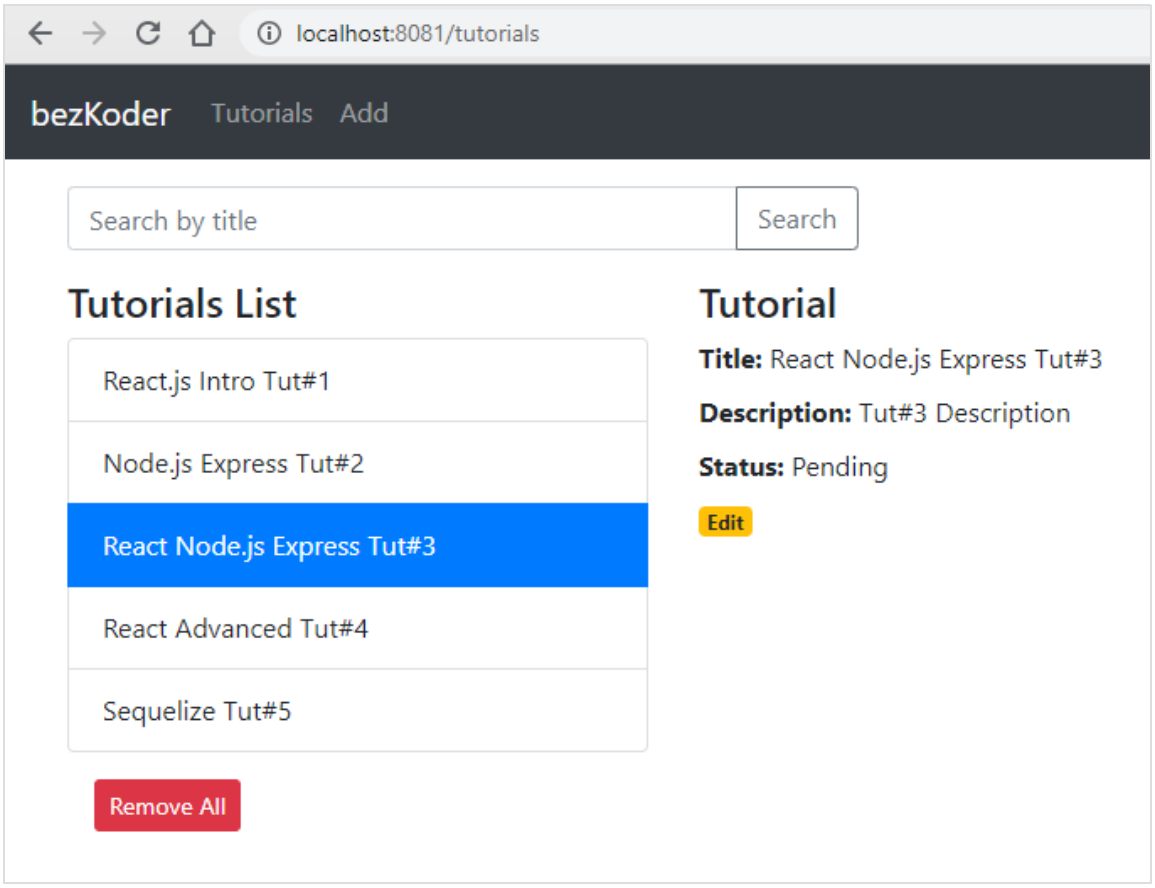
- Tutorial has id, title, description, published status.
- User can create, retrieve, update, delete Tutorials.
- There is a search box for finding Tutorials by title.

Here are screenshots of the example.

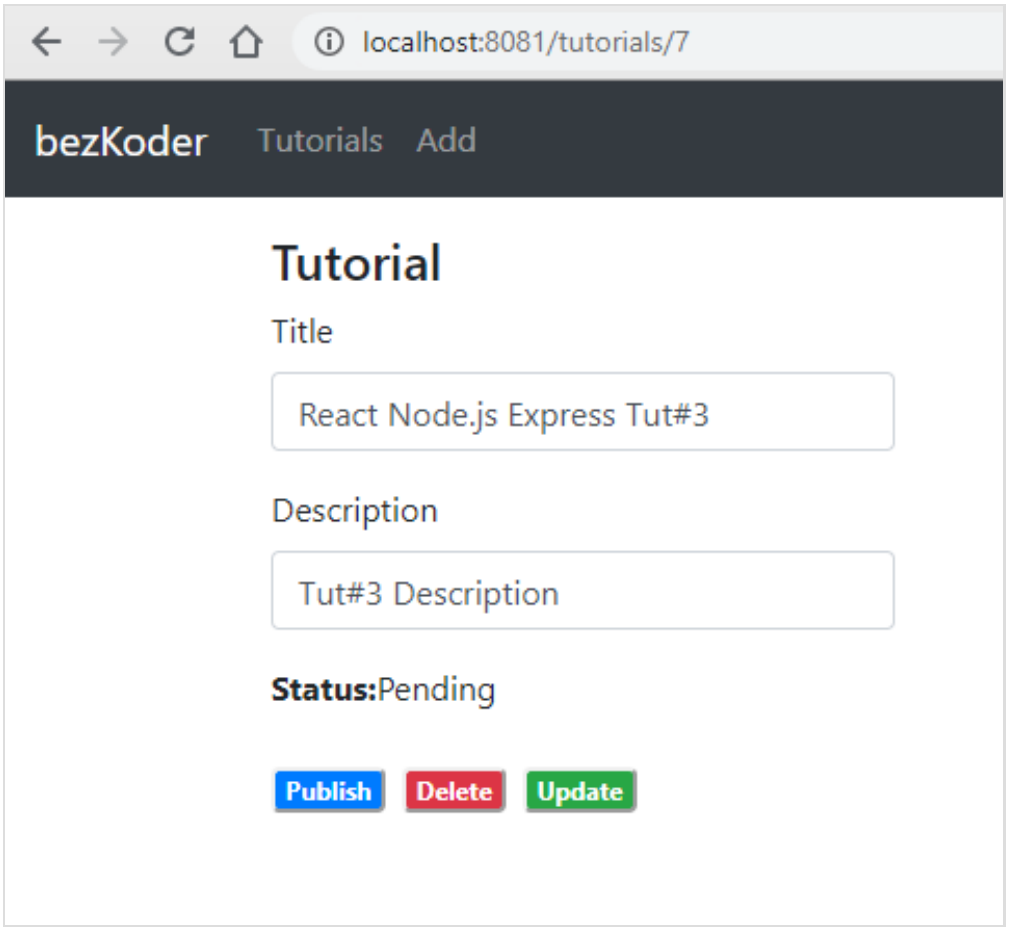
– Add an item:



– Show all items:

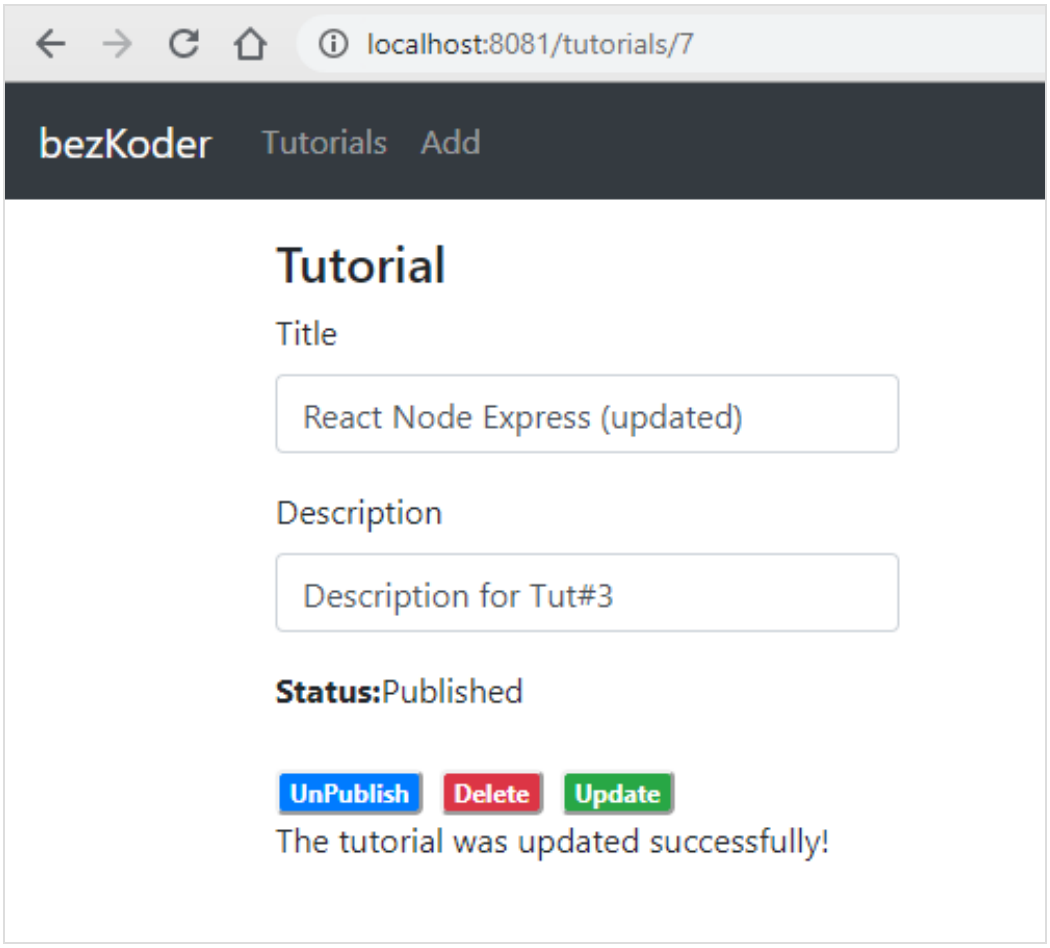


– Click on **Edit** button to view details of an item:

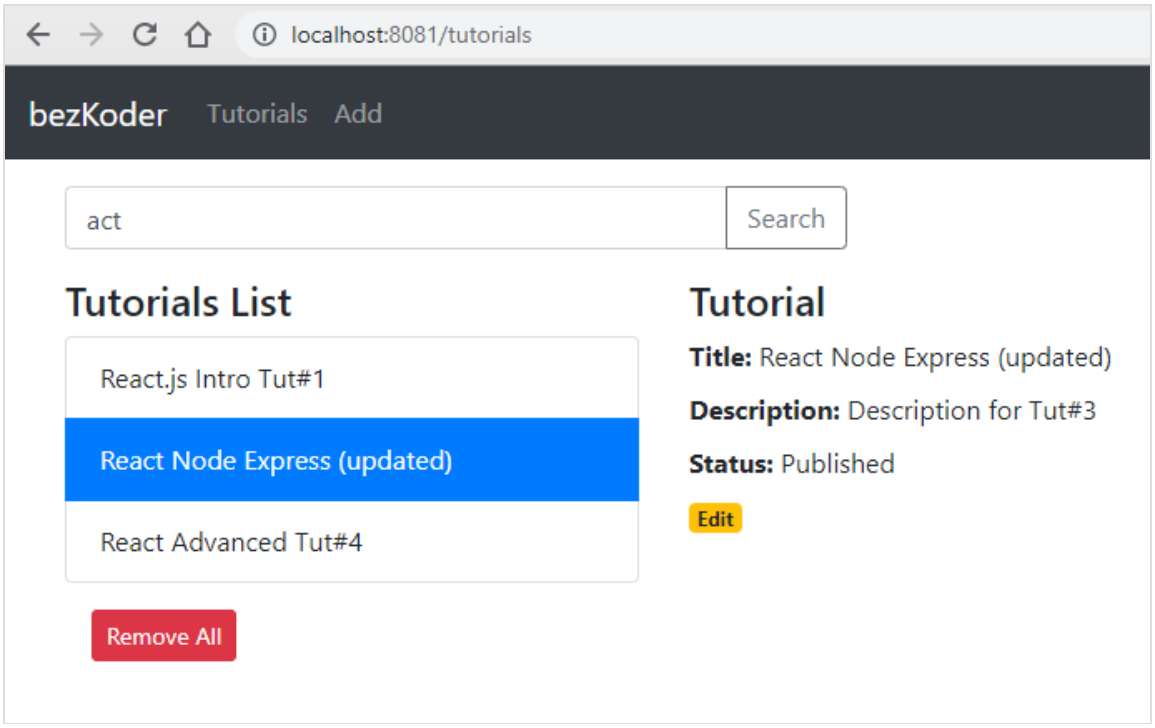


On this Page, you can:

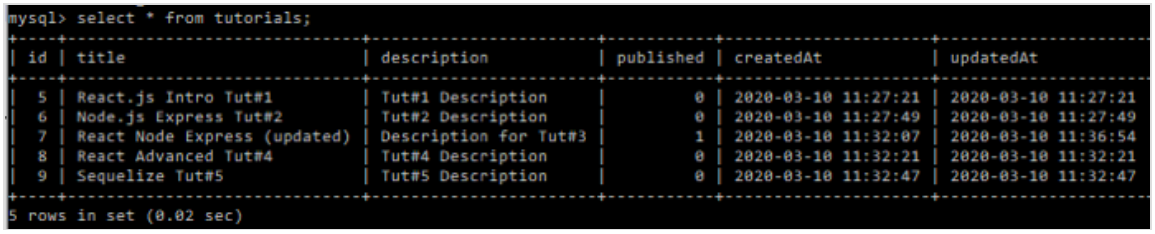
- change status to **Published**/**Pending** using **Publish**/**UnPublished** button
- remove the object from MySQL Database using **Delete** button
- update this object's details on Database with **Update** button



– Search objects by field 'title':



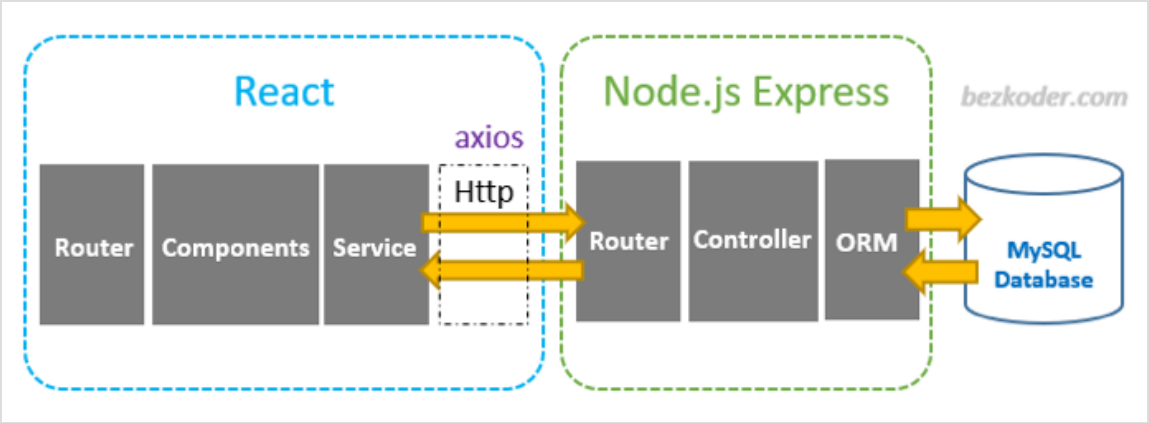
– Check MySQL database:



```
mysql> select * from tutorials;
+----+-----------------------------+----------------------+-----------+---------------------+---------------------+
| id | title                       | description          | published | createdAt           | updatedAt           |
+----+-----------------------------+----------------------+-----------+---------------------+---------------------+
|  5 | React.js Intro Tut#1        | Tut#1 Description    |         0 | 2020-03-10 11:27:21 | 2020-03-10 11:27:21 |
|  6 | Node.js Express Tut#2       | Tut#2 Description    |         0 | 2020-03-10 11:27:49 | 2020-03-10 11:27:49 |
|  7 | React Node Express (updated)| Description for Tut#3|         1 | 2020-03-10 11:32:07 | 2020-03-10 11:36:54 |
|  8 | React Advanced Tut#4        | Tut#4 Description    |         0 | 2020-03-10 11:32:21 | 2020-03-10 11:32:21 |
|  9 | Sequelize Tut#5             | Tut#5 Description    |         0 | 2020-03-10 11:32:47 | 2020-03-10 11:32:47 |
+----+-----------------------------+----------------------+-----------+---------------------+---------------------+
5 rows in set (0.02 sec)
```

React, Node.js Express, MySQL Architecture

We're gonna build the application with following architecture:



– Node.js Express exports REST APIs & interacts with MySQL Database using Sequelize ORM.
– React Client sends HTTP Requests and retrieves HTTP Responses using *Axios*, consume data on the components. React Router is used for navigating to pages.

Video

This is our React Node.js Express Sequelize application demo (with brief instruction) running with MySQL database.

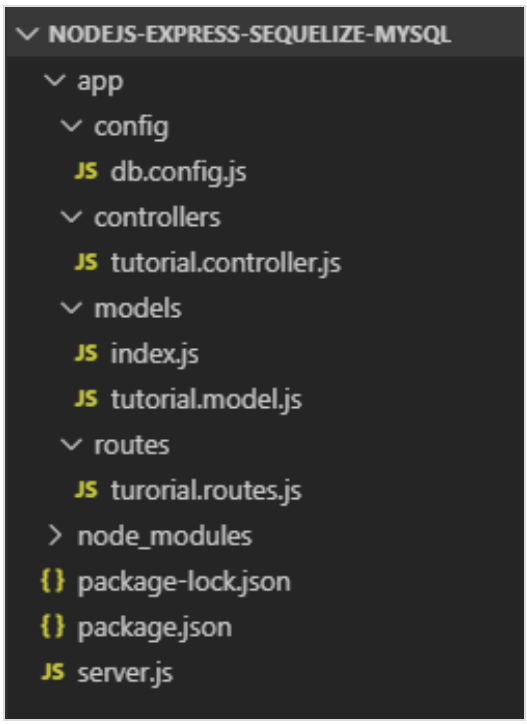
Node.js Express Back-end

Overview

These are APIs that Node.js Express App will export:

Methods	Urls	Actions
GET	api/tutorials	get all Tutorials
GET	api/tutorials/:id	get Tutorial by `id`
POST	api/tutorials	add new Tutorial
PUT	api/tutorials/:id	update Tutorial by `id`
DELETE	api/tutorials/:id	remove Tutorial by `id`
DELETE	api/tutorials	remove all Tutorials
GET	api/tutorials?title=[kw]	find all Tutorials which title contains `'kw'`

Project Structure



– *db.config.js* exports configuring parameters for MySQL connection & Sequelize.
– **Express** web server in *server.js* where we configure CORS, initialize & run Express REST APIs.
– Next, we add configuration for MySQL database in **models**/*index.js*, create **Sequelize** data model in **models**/*tutorial.model.js*.
– Tutorial controller in **controllers**.
– Routes for handling all CRUD operations (including custom finder) in *tutorial.routes.js*.

If you want to use raw SQL (without Sequelize), kindly visit:
Build Node.js Rest APIs with Express & MySQL (https://www.bezkoder.com/node-js-rest-api-express-mysql/)

This backend works well with frontend in this tutorial.

Implementation

Create Node.js App

First, we create a folder:

```
$ mkdir nodejs-express-sequelize-mysql
$ cd nodejs-express-sequelize-mysql
```

Next, we initialize the Node.js App with a *package.json* file:

```
npm init

name: (nodejs-express-sequelize-mysql)
version: (1.0.0)
description: Node.js Rest Apis with Express, Sequelize & MySQL.
entry point: (index.js) server.js
test command:
git repository:
keywords: nodejs, express, sequelize, mysql, rest, api
author: bezkoder
license: (ISC)

Is this ok? (yes) yes
```

We need to install necessary modules: `express`, `sequelize`, `mysql2` and `cors`.
Run the command:

```
npm install express sequelize mysql2 cors --save
```

Setup Express web server

In the root folder, let's create a new *server.js* file:

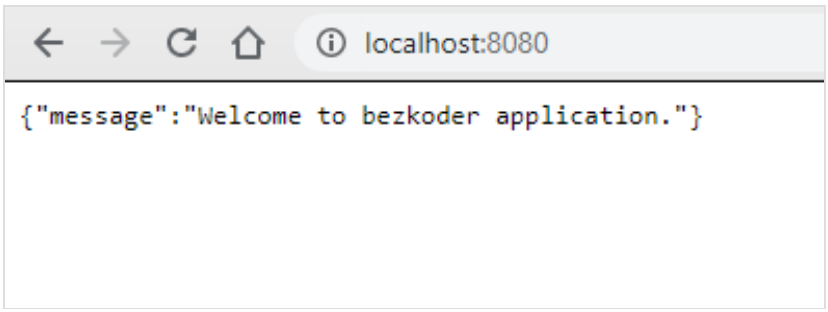
```
const express = require("express");
const cors = require("cors");

const app = express();

var corsOptions = {
  origin: "http://localhost:8081"
};

app.use(cors(corsOptions));

// parse requests of content-type - application/json
app.use(express.json());

// parse requests of content-type - application/x-www-form-urler
app.use(express.urlencoded({ extended: true }));

// simple route
app.get("/", (req, res) => {
  res.json({ message: "Welcome to bezkoder application." });
});

// set port, listen for requests
const PORT = process.env.PORT || 8080;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}.`);
});
```

What we do are:
– import `express`, and `cors` modules:

- Express is for building the Rest apis
- cors (https://www.npmjs.com/package/cors) provides Express middleware to enable CORS with various options.

– create an Express app, then add body-parser (`json` and `urlencoded`) and `cors` middlewares using `app.use()` method. Notice that we set origin: `http://localhost:8081`.
– define a GET route which is simple for test.
– listen on port 8080 for incoming requests.

Now let's run the app with command: `node server.js`.
Open your browser with url http://localhost:8080/ (http://localhost:8080/), you will see:



Yeah, the first step is done. We're gonna work with Sequelize in the next section.

Configure MySQL database & Sequelize

In the *app* folder, we create a separate *config* folder for configuration with *db.config.js* file like this:

```
module.exports = {
  HOST: "localhost",
  USER: "root",
  PASSWORD: "123456",
  DB: "testdb",
  dialect: "mysql",
  pool: {
    max: 5,
    min: 0,
    acquire: 30000,
    idle: 10000
  }
};
```

First five parameters are for MySQL connection.
`pool` is optional, it will be used for Sequelize connection pool configuration:

- `max`: maximum number of connection in pool
- `min`: minimum number of connection in pool
- `idle`: maximum time, in milliseconds, that a connection can be idle before being released
- `acquire`: maximum time, in milliseconds, that pool will try to get connection before throwing error

For more details, please visit API Reference for the Sequelize constructor (https://sequelize.org/master/class/lib/sequelize.js~Sequelize.html#instance-constructor-constructor).

Initialize Sequelize

We're gonna initialize Sequelize in **app**/**models** folder that will contain model in the next step.

Now create **app**/**models**/*index.js* with the following code:

```
const dbConfig = require("../config/db.config.js");

const Sequelize = require("sequelize");
const sequelize = new Sequelize(dbConfig.DB, dbConfig.USER, dbCo
  host: dbConfig.HOST,
  dialect: dbConfig.dialect,
  operatorsAliases: false,

  pool: {
    max: dbConfig.pool.max,
    min: dbConfig.pool.min,
    acquire: dbConfig.pool.acquire,
    idle: dbConfig.pool.idle
  }
});

const db = {};

db.Sequelize = Sequelize;
db.sequelize = sequelize;

db.tutorials = require("./tutorial.model.js")(sequelize, Sequeli

module.exports = db;
```

Don't forget to call `sync()` method in *server.js*:

```
...
const app = express();
app.use(...);

const db = require("./app/models");
db.sequelize.sync();

...
```

In development, you may need to drop existing tables and re-sync database. Just use `force: true` as following code:

```
db.sequelize.sync({ force: true }).then(() => {
  console.log("Drop and re-sync db.");
});
```

Define the Sequelize Model

In *models* folder, create *tutorial.model.js* file like this:

```
module.exports = (sequelize, Sequelize) => {
  const Tutorial = sequelize.define("tutorial", {
    title: {
      type: Sequelize.STRING
    },
    description: {
      type: Sequelize.STRING
    },
    published: {
      type: Sequelize.BOOLEAN
    }
  });

  return Tutorial;
};
```

This Sequelize Model represents **tutorials** table in MySQL database. These columns will be generated automatically: *id*, *title*, *description*, *published*, *createdAt*, *updatedAt*.

After initializing Sequelize, we don't need to write CRUD functions, Sequelize supports all of them:

- create a new Tutorial: `create (https://sequelize.org/master/class/lib/model.js~Model.html#static-method-create)(object)`
- find a Tutorial by id: `findByPk (https://sequelize.org/master/class/lib/model.js~Model.html#static-method-findByPk)(id)`
- get all Tutorials: `findAll (https://sequelize.org/master/class/lib/model.js~Model.html#static-method-findAll)()`
- update a Tutorial by id: `update (https://sequelize.org/master/class/lib/model.js~Model.html#static-method-update)(data, where: { id: id })`
- remove a Tutorial: `destroy (https://sequelize.org/master/class/lib/model.js~Model.html#static-method-destroy)(where: { id: id })`
- remove all Tutorials: `destroy(where: {})`
- find all Tutorials by title: `findAll({ where: { title: ... } })`

These functions will be used in our Controller.

We can improve the example by adding Comments for each Tutorial. It is the One-to-Many Relationship and I write a tutorial for this at:
Sequelize Associations: One-to-Many example – Node.js, MySQL (https://bezkoder.com/sequelize-associate-one-to-many/)

Or you can add Tags for each Tutorial and add Tutorials to Tag (Many-to-Many Relationship):
Sequelize Many-to-Many Association example with Node.js & MySQL (https://bezkoder.com/sequelize-associate-many-to-many/)

Create the Controller

Inside **app**/**controllers** folder, let's create *tutorial.controller.js* with these CRUD functions:

- create
- findAll
- findOne
- update
- delete
- deleteAll
- findAllPublished

```
const db = require("../models");
const Tutorial = db.tutorials;
const Op = db.Sequelize.Op;

// Create and Save a new Tutorial
exports.create = (req, res) => {

};

// Retrieve all Tutorials from the database.
exports.findAll = (req, res) => {

};

// Find a single Tutorial with an id
exports.findOne = (req, res) => {

};

// Update a Tutorial by the id in the request
exports.update = (req, res) => {

};

// Delete a Tutorial with the specified id in the request
exports.delete = (req, res) => {

};

// Delete all Tutorials from the database.
exports.deleteAll = (req, res) => {

};

// Find all published Tutorials
exports.findAllPublished = (req, res) => {

};
```

You can continue with step by step to implement this Node.js Express App in the post:
Node.js Rest APIs example with Express, Sequelize & MySQL

(https://bezkoder.com/node-js-express-sequelize-mysql/)

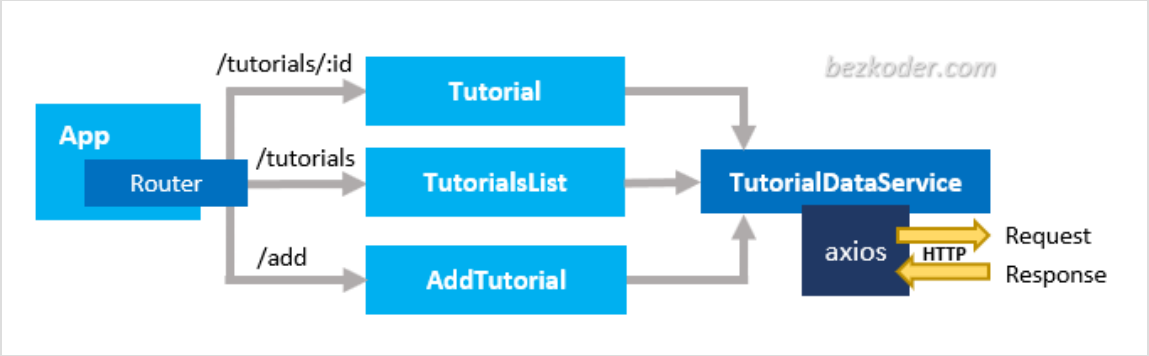
Run the Node.js Express Server

Run our Node.js application with command: `node server.js`.
The console shows:

```
Server is running on port 8080.
Executing (default): DROP TABLE IF EXISTS `tutorials`;
Executing (default): CREATE TABLE IF NOT EXISTS `tutorials` (`id` I
NTEGER NOT NULL auto_increment , `title` VARCHAR(255), `description
` VARCHAR(255), `published` TINYINT(1), `createdAt` DATETIME NOT NU
LL, `updatedAt` DATETIME NOT NULL, PRIMARY KEY (`id`)) ENGINE=InnoD
B;
Executing (default): SHOW INDEX FROM `tutorials`
Drop and re-sync db.
```

React.js Front-end

Overview

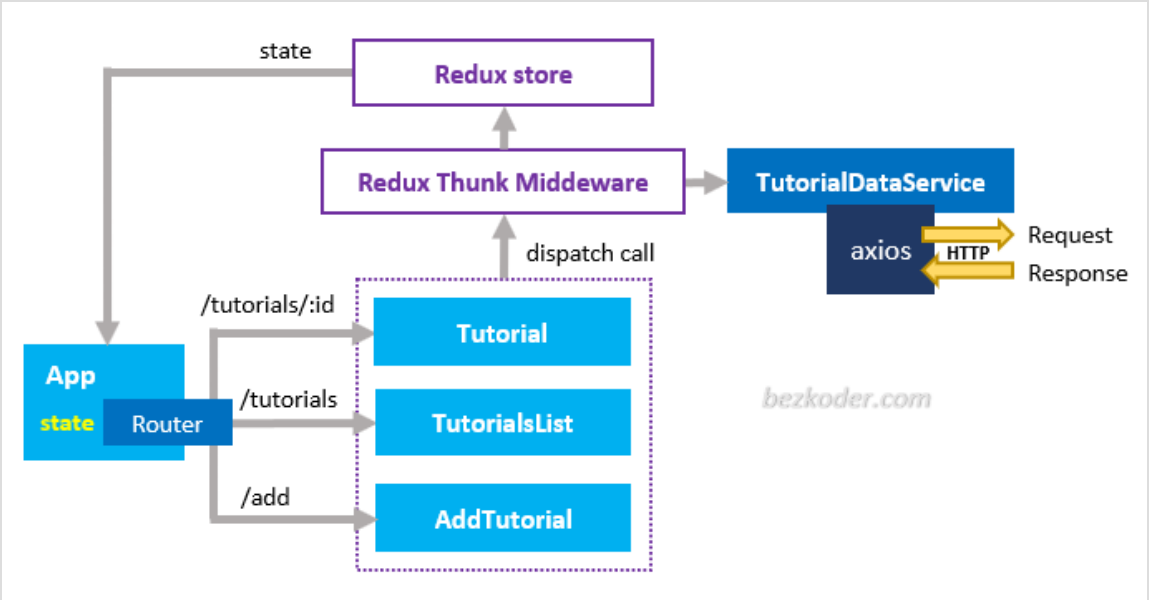


– The `App` component is a container with React `Router`. It has `navbar` that links to routes paths.

– `TutorialsList` component gets and displays Tutorials.
– `Tutorial` component has form for editing Tutorial's details based on `:id`.
– `AddTutorial` component has form for submission new Tutorial.

– These Components call `TutorialDataService` methods which use `axios` to make HTTP requests and receive responses.

Or you can use React with Redux:

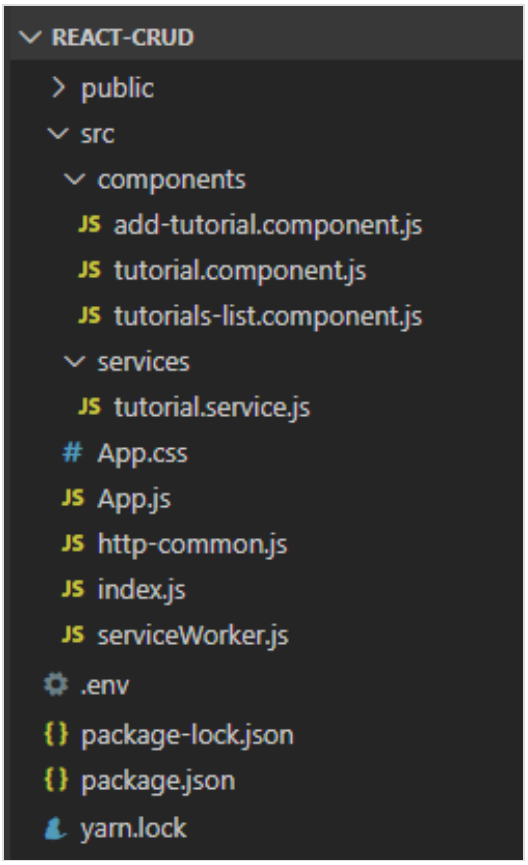


More details at: React Redux CRUD App example with Rest API (https://bezkoder.com/react-redux-crud-example/)

Technology

- React 18/17
- react-router-dom 6
- axios 0.27.2
- bootstrap 4

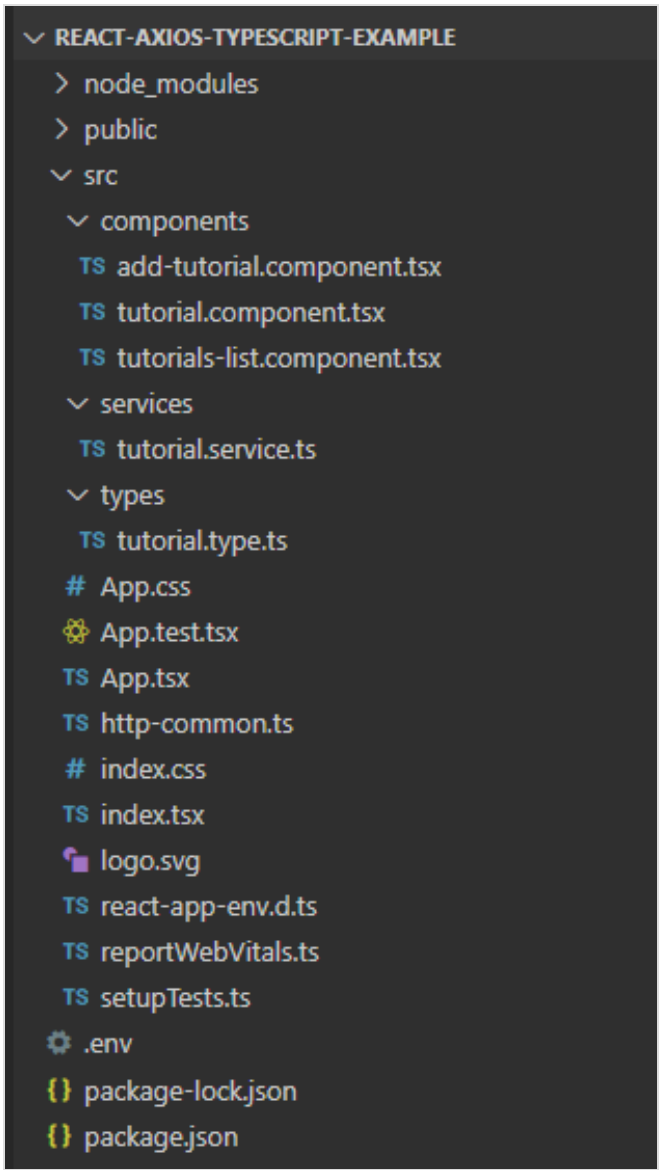
Project Structure



– **package.json** contains 4 main modules: `react`, `react-router-dom`, `axios` & `bootstrap`.
– `App` is the container that has `Router` & navbar.
– There are 3 components: `TutorialsList`, `Tutorial`, `AddTutorial`.
– **http-common.js** initializes axios with HTTP base Url and headers.
– `TutorialDataService` has methods for sending HTTP requests to the Apis.
– **.env** configures *port* for this React CRUD App.

For React Hooks version, kindly visit this tutorial (https://www.bezkoder.com/react-hooks-crud-axios-api/).

For Typescript version:



Please visit:
React Typescript CRUD example with Web API (https://bezkoder.com/react-typescript-axios/)

Implementation

Setup React.js Project

Open cmd at the folder you want to save Project folder, run command:
`npx create-react-app react-crud`

After the process is done. We create additional folders and files like the following tree:

---

public

- src
  - components
    - add-tutorial.component.js
    - tutorial.component.js
    - tutorials-list.component.js
  - services
    - tutorial.service.js
  - App.css
  - App.js
  - index.js
- package.json

Import Bootstrap to React CRUD App

Run command: `npm install bootstrap`.

Open **src**/*App.js* and modify the code inside it as following-

```
import React, { Component } from "react";
import "bootstrap/dist/css/bootstrap.min.css";

class App extends Component {
  render() {
    // ...
  }
}

export default App;
```

Add React Router to React CRUD App

– Run the command: `npm install react-router-dom`.
– Open **src**/*index.js* and wrap `App` component by `BrowserRouter` object.

```
import React from "react";
import { createRoot } from "react-dom/client";
import { BrowserRouter } from "react-router-dom";

import App from "./App";

const container = document.getElementById("root");
const root = createRoot(container);

root.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>
);
```

Add Navbar to React CRUD App

Open **src**/*App.js*, this `App` component is the root container for our application, it will contain a `navbar`, and also, a `Routes` object with several `Route`. Each `Route` points to a React Component.

```
import React, { Component } from "react";
...

class App extends Component {
  render() {
    return (
      <div>
        <nav className="navbar navbar-expand navbar-dark bg-dark
          <a href="/tutorials" className="navbar-brand">
            bezKoder
          </a>
          <div className="navbar-nav mr-auto">
            <li className="nav-item">
              <Link to={"/tutorials"} className="nav-link">
                Tutorials
              </Link>
            </li>
            <li className="nav-item">
              <Link to={"/add"} className="nav-link">
                Add
              </Link>
            </li>
          </div>
        </nav>

        <div className="container mt-3">
          <Routes>
            <Route path="/" element={<TutorialsList/>} />
            <Route path="/tutorials" element={<TutorialsList/>}
            <Route path="/add" element={<AddTutorial/>} />
            <Route path="/tutorials/:id" element={<Tutorial/>} /
          </Routes>
        </div>
      </div>
    );
  }
}

export default App;
```

Initialize Axios for React CRUD HTTP Client

Let's install *axios* with command: `npm install axios`.
Under **src** folder, we create *http-common.js* file with following code:

```
import axios from "axios";

export default axios.create({
  baseURL: "http://localhost:8080/api",
  headers: {
    "Content-type": "application/json"
  }
});
```

You can change the `baseURL` that depends on REST APIs url that your Server configures.

Create Data Service

In this step, we're gonna create a service that uses axios object above to send HTTP requests.

services/*tutorial.service.js*

```
import http from "../http-common";

class TutorialDataService {
  getAll() {
    return http.get("/tutorials");
  }

  get(id) {
    return http.get(`/tutorials/${id}`);
  }

  create(data) {
    return http.post("/tutorials", data);
  }

  update(id, data) {
    return http.put(`/tutorials/${id}`, data);
  }

  delete(id) {
    return http.delete(`/tutorials/${id}`);
  }

  deleteAll() {
    return http.delete(`/tutorials`);
  }

  findByTitle(title) {
    return http.get(`/tutorials?title=${title}`);
  }
}

export default new TutorialDataService();
```

We call axios `get`, `post`, `put`, `delete` method corresponding to HTTP Requests: GET, POST, PUT, DELETE to make CRUD Operations.

Create React Components/Pages

Now we're gonna build 3 components corresponding to 3 Routes defined before.

- Add new Item
- List of items
- Item details

You can continue with step by step to implement this React App in the post:
– React.js CRUD example to consume Web API (https://bezkoder.com/react-crud-web-api/)
– or React Hooks CRUD example to consume Web API (https://bezkoder.com/react-hooks-crud-axios-api/)

Using React with Redux:
– React Redux CRUD example with Rest API (https://bezkoder.com/react-redux-crud-example/)
– React Hooks + Redux: CRUD example with Rest API (https://bezkoder.com/react-hooks-redux-crud/)

For Typescript version:
React Typescript CRUD example to consume Web API (https://bezkoder.com/react-typescript-axios/)

Run React CRUD App

You can run our App with command: `npm start`.
If the process is successful, open Browser with Url: `http://localhost:8081/` and check it.

Source Code

You can find Github source code for this tutorial at: React + Node App Github (https://www.githubcode.com/react-node-app-github/)

Conclusion

Today we have an overview of React.js + Node.js Express + MySQL example when building a full-stack CRUD App.

We also take a look at client-server architecture for REST API using Express & Sequelize ORM, as well as React.js project structure for building a front-end app to make HTTP requests and consume responses.

Next tutorials show you more details about how to implement the system (including source code):
– Back-end:

- With Sequelize ORM (https://bezkoder.com/node-js-express-sequelize-mysql/)
- Without Sequelize (https://www.bezkoder.com/node-js-rest-api-express-mysql/)

– Front-end:

- Using React Components (https://bezkoder.com/react-crud-web-api/)
- Using React Typescript Components (https://bezkoder.com/react-typescript-axios/)
- Using React Redux (https://bezkoder.com/react-redux-crud-example/)
- Using React Hooks (https://bezkoder.com/react-hooks-crud-axios-api/)
- Using React Hooks + Redux (https://bezkoder.com/react-hooks-redux-crud/)
- Using React with Material UI (https://bezkoder.com/react-material-ui-examples-crud/)

You will want to know how to run both projects in one place:
How to integrate React with Node.js Express on same Server/Port (https://bezkoder.com/integrate-react-express-same-server-port/)

With Pagination:
React Pagination with API using Material-UI (https://bezkoder.com/react-pagination-material-ui/)

Or Serverless with Firebase:
– React Firebase CRUD with Realtime Database (https://bezkoder.com/react-firebase-crud/)
– React Firestore CRUD App example | Firebase Cloud Firestore (https://bezkoder.com/react-firestore-crud/)

Happy learning, see you again!

Further Reading

- https://www.npmjs.com/package/express (https://www.npmjs.com/package/express)
- https://www.npmjs.com/package/mysql2 (https://www.npmjs.com/package/mysql2)
- React Component (https://reactjs.org/docs/react-component.html)
- React Custom Hook (https://www.bezkoder.com/react-custom-hook/)

Dockerize: Docker Compose: React, Node.js, MySQL example (https://www.bezkoder.com/docker-compose-react-nodejs-mysql/)

(https://www.bezkoder.com/tag/crud/)

(https://www.bezkoder.com/tag/express/)

(https://www.bezkoder.com/tag/mysql/)

(https://www.bezkoder.com/tag/node-js/)

(https://www.bezkoder.com/tag/react/)

(https://www.bezkoder.com/tag/react-router/)

(https://www.bezkoder.com/tag/rest-api/)

(https://www.bezkoder.com/tag/sequelize/)

54 thoughts to “React + Node.js + Express + MySQL example: Build a CRUD App”

John (https://github.com/johnvenkiah)

June 6, 2022 at 10:46 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-23319)

Hi, great tutorials, thanks for that! Rather new to Node and SQL.

I have got the tutorial from https://www.bezkoder.com/node-js-rest-api-express-mysql/ (https://www.bezkoder.com/node-js-rest-api-express-mysql/) working via Postman, but don’t really know how to

sync it t work with the React project. Should I start both node and npm start on both projects at the same time? Because the React live server is returning a 404 for all requests.

I have a feeling it is not connecting with my database.. Should I configure MySQL to port 8080? Or is there something I have missed to add to the React project from the Node JS rest API project? I take it they should be separate repositories?

Working with MySQL via MySQL Workbench.

Thanks a bunch!

bezkoder

June 7, 2022 at 3:33 am (https://www.bezkoder.com/react-node-express-mysql/#comment-23334)

Hi, you need to run Node Express backend first. It will export APIs at `localhost:8080`.
Then you can run React for frontend at `localhost:8081` with `npm start`.
Finally, open url with address: `http://localhost:8081/`.

JOAO PAULO SOARES FRAGOSO

May 13, 2022 at 11:48 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-22152)

how was the serviceWorker.js file created?

I think I made a mistake in something that ended up not creating him.

bezkoder

May 14, 2022 at 2:37 am (https://www.bezkoder.com/react-node-express-mysql/#comment-22161)

Hi, it is generated by old version of React through the `create-react-app` command. If you use new React version, don't care 🙂

Steve
February 7, 2022 at 11:42 am (https://www.bezkoder.com/react-node-express-mysql/#comment-19217)

Such a terrific series of tutorials, thank you so much.

Do you have a tutorial which covers the construction of React web forms for relational Sequelize data?

I have two related models (belongsTo/hasOne) and I want to build a form with a select list which respects the relationship. E.g. a Employer hasMany Staff and a Staff belongsTo an Employer. A select list on the Employer form should show Companies by name and stores the companyId on the Employee record. (This is a made-up example but you get the idea.)

Is there any built-in tooling to create the a select list from the model? I can imagine how to do it manually, but I wonder if there are helpers that I am unaware of.

jhanvi mehta
February 4, 2022 at 5:52 am (https://www.bezkoder.com/react-node-express-mysql/#comment-19102)

Thank you very much. It is very helpful for any beginners who want to start a CRUD. I especially like in this is that you added diagrams and code because people easily understand. Very knowledgeable and creative information. My appreciation to you.

ThantZinTun
January 6, 2022 at 2:37 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-17884)

Please pass me source codes for this tutorial.

bezkoder
January 7, 2022 at 11:15 am (https://www.bezkoder.com/react-node-express-mysql/#comment-17923)

Hi, you can find Github in the tutorials I mention at Conclusion section 🙂

KA
January 5, 2022 at 7:06 am (https://www.bezkoder.com/react-node-express-mysql/#comment-17828)

Hi, do you have any tutorial to show how to connect cloud database with a website? Since the local database can not be accessed by other users.

bezkoder
January 5, 2022 at 9:12 am (https://www.bezkoder.com/react-node-express-mysql/#comment-17831)

Hi, you can read following tutorial:
Deploying/Hosting Node.js app on Heroku with MySQL database (https://www.bezkoder.com/deploy-node-js-app-heroku-cleardb-mysql/)

GUEYE
December 28, 2021 at 2:00 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-17378)

good job

Swatski
December 16, 2021 at 5:41 am (https://www.bezkoder.com/react-node-express-

mysql/#comment-16677)

Thanks for the well-written tutorial. I think I learned more from this.

Gark

November 24, 2021 at 6:08 am (https://www.bezkoder.com/react-node-express-mysql/#comment-15621)

Whenever I learn something new, or want to check if my implementations are right, I always go to your website. Thank you so much for everything! <3 Regards from Philippines!

mickael

November 5, 2021 at 11:12 am (https://www.bezkoder.com/react-node-express-mysql/#comment-14911)

Hi, bezkoder,

Your tutorial was useful for me.
But I have a question, regarding the list, I tried to add the `setInterval()` method.
But it does not work, would you have a clue, thank you in advance

Neng Xiong

October 21, 2021 at 10:21 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-14236)

Hi, bezkoder,

Nice tutorial! I am new to node.js and I noticed that you didn't define "findByTitle" in routes.js of the server end, but you used it in the server.js in the front end. How does this work? Thanks!

bezkoder

October 22, 2021 at 12:48 am (https://www.bezkoder.com/react-node-express-mysql/#comment-14242)

Hi, you can look at **controllers**/*tutorial.controller.js*. We have `findAll` method with `title` as query param.
So the client (frontend) just sends http request with suffix `/api/tutorials?title=[title]`.

In *tutorial.routes.js*, we have:

```
router.get("/", tutorials.findAll);
...
app.use('/api/tutorials', router);
```

Genie

October 17, 2021 at 9:15 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-13829)

I am getting experience all the time by reading your tutorials. Thanks!

Rudy

October 13, 2021 at 4:54 am (https://www.bezkoder.com/react-node-express-mysql/#comment-13587)

Will likely be back to get more tutorials from you. Thanks!

Kisha

September 23, 2021 at 9:05 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-12569)

I needed to thank you for this fantastic tutorial!! I certainly loved every bit of it.

Elwood
September 14, 2021 at 5:32 am (https://www.bezkoder.com/react-node-express-mysql/#comment-12183)

Thanks for the tutorial. Regards

charbel
September 12, 2021 at 1:15 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-12083)

one question here you are using mysql database if i want to use sql express can i put
dialect:”mssql”
instead of dialect:”mysql” ???

bezkoder
September 14, 2021 at 1:59 am (https://www.bezkoder.com/react-node-express-mysql/#comment-12173)

Hi, you can use the backend with MSSQL database with the tutorial: Node.js CRUD example with SQL Server (MSSQL) (https://www.bezkoder.com/node-js-sql-server-crud/)

SleepyCatzzz (http://none%20yet)
June 29, 2021 at 1:29 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-9368)

Thank you so much, am learning so much! How can I expand your example so I can interact with multiple different tables? If I have table “user”, with its own fields, do I make a user.routes.js, user.controller.js, etc for each table? Is there a sleek way to make this more general? Thank you!

Taiwo
June 4, 2021 at 1:09 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-8874)

I also encountered this error, what I did was that, I left the password space to an empty string

Dai
April 14, 2021 at 4:47 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-8089)

very interesting, good job and thanks for sharing this fullstack tutorial.

Altin
March 27, 2021 at 11:45 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-7779)

A very detailed and informative tutorial
What would be the best hosting option for a custom app that uses React + Node.js + Express + MySQL , as explained in your tutorial?
Thank you!

Gatis
March 13, 2021 at 5:17 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-7586)

VSC says 'bodyParser' is deprecated.ts(6385)
Possibly this is the solution https://stackoverflow.com/a/62396576 (https://stackoverflow.com/a/62396576)

Marco de Boer
March 5, 2021 at 9:41 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-7424)

Hi,

It can’t find the three objects, Tutorials, AddTutorial, Tutorial, what do I miss?

Kind regards,

Marco de Boer

bezkoder
March 6, 2021 at 12:56 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-7435)

Hi, please visit the tutorial I mentioned right there for next steps.

peter
February 10, 2021 at 7:47 am (https://www.bezkoder.com/react-node-express-mysql/#comment-6944)

SequelizeAccessDeniedError: Access denied for user ‘root’@’172.18.0.1’ (using password: YES)

what is the correct initial password for mysql2 ?

peter
February 10, 2021 at 4:43 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-6961)

My fault, had no sqlsever just teh client installed 😉 thx

Wambetsa
March 2, 2021 at 3:21 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-7340)

You need to ensure that this password is the same as the one you use in localhost.

Christopher Sykes
May 14, 2021 at 12:01 am (https://www.bezkoder.com/react-node-express-mysql/#comment-8527)

Sorry, what do you mean the same password as in localhost? What localhost password? Also, do we need sqlserver installed? Do we have to create the database (I don't recall any instruction to do that)? It's a great tutorial but I feel like I am missing some very key instructions. My problem is EACCESS denied.

tiana
October 18, 2021 at 8:48 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-13979)

no

porios
October 20, 2020 at 11:18 am (https://www.bezkoder.com/react-node-express-mysql/#comment-5129)

Hi, i want some help, so i follow the tuto and i have this error i can't find why "(node:13696) UnhandledPromiseRejectionWarning: SequelizeDatabaseError: Table 'testdb.tutorials' doesn't exist", i searched to resolve this but I tried on my own and look for where the problem could come from, even if I suck I try to manage but here I block, I would like to know if you have any leads

Atif
December 10, 2020 at 12:02 pm (https://www.bezkoder.com/react-node-express-

mysql/#comment-5714)

That means you don't have Table in your local MySQL. First create "testdb" named database and create "tutorials" named table inside it.

Arthur

September 30, 2020 at 3:20 am (https://www.bezkoder.com/react-node-express-mysql/#comment-4889)

Hi, I would like to use your app on my own host (with Planet Hoster).
All it's done, and well set, React and NodeJS launched and the files compiled, but I have a 503 error on the server. The support told me it's because of the use of the port 8081 and the fact I'm trying to launch an other server on their server.
How I can disallow the use of the localhost server and use only the server of my host ?
Thanks for your help, and thanks for the tutorial ! 🙂
Arthur

trikot

August 20, 2020 at 11:42 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-4411)

This excellent website really has all the information and facts I needed concerning this subject and
didn't know who to ask.

Many thanks for the tutorial.

karthika

August 19, 2020 at 11:47 am (https://www.bezkoder.com/react-node-express-mysql/#comment-4377)

Hi..I've a problem in running the application.
Can I know how to run and which file to run with commands please?

bezkoder
August 19, 2020 at 2:09 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-4380)

Hi,

For React Client: npm run start
For Node.js Server: node server.js

Nitin Betharia
July 31, 2020 at 2:04 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-4103)

Can’t we create a similar front-end without react (i mean in node.js – javascript)

Veikel
July 24, 2020 at 6:11 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-3930)

thanks; i cannot connect with mysql directly? i need always build a API?

Jeffersom Guilherme Machado Barreto
September 11, 2020 at 1:06 am (https://www.bezkoder.com/react-node-express-mysql/#comment-4679)

yes, you need to create a code in node js that queries your database and then throws that data in an api, then you access it in your application from a fetch if it is in react js

Pinthip
June 19, 2020 at 8:56 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-3366)

I'm very new in node.js and your demo is very useful. I follow it step by step but I got an error as below.
Please tell me wrong I did? It was occurred in model file while define data type.
I got an error below
type: Sequelize.INTEGER
^
TypeError: Cannot read property 'INTEGER' of undefined

bezkoder
June 21, 2020 at 4:42 am (https://www.bezkoder.com/react-node-express-mysql/#comment-3383)

Hi, you should initialize Sequelize first, please read this section (https://bezkoder.com/node-js-express-sequelize-mysql/#Initialize_Sequelize). Once you have the Sequelize object, pass it to the function which define Sequelize model:
`db.tutorials = require("./tutorial.model.js")(sequelize, Sequelize);`

Oscar
April 22, 2020 at 10:09 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-1281)

This React tutorial is really good, I like your web site so much, waiting for new React tutorial!

Kirk
April 21, 2020 at 7:16 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-1065)

Thank you so much for your effort to make this fullstack React Node tutorial!

Ravi

April 16, 2020 at 1:26 am (https://www.bezkoder.com/react-node-express-mysql/#comment-876)

Hi,
Nice tutorial. Please provide a git url , etc. which I can download and start. I want something to start with for my project.

bezkoder

April 16, 2020 at 7:15 am (https://www.bezkoder.com/react-node-express-mysql/#comment-882)

Hi, you can find github source code in the next posts (Conclusion section). 🙂

Mohammad Atlaf

April 11, 2020 at 4:04 pm (https://www.bezkoder.com/react-node-express-mysql/#comment-833)

I follow the code in your site of https://bezkoder.com/react-node-express-mysql/ (https://bezkoder.com/react-node-express-mysql/)

i run the node server on port 3000. it run successful

In your site you have example to connect react by using Sequelize but not without this. as in node you connect to MySQL directly without using the Sequelize .

Please give the link which connect the front end to back-end without using Sequelize.

Thanks

Thanks

bezkoder

April 11, 2020 at 10:35 pm (https://www.bezkoder.com/react-node-express-

mysql/#comment-837)

Hi, the front-end doesn't need to care about whether the back-end uses Sequelize or not. They connect toghether via Rest APIs.

So you can implement Node Express backend without Sequelize,refer to this post:
Build Node.js Rest APIs with Express & MySQL (https://bezkoder.com/node-js-rest-api-express-mysql/)

Nat
April 8, 2020 at 3:17 am (https://www.bezkoder.com/react-node-express-mysql/#comment-776)

How do you connect the front and backend?

bezkoder
April 8, 2020 at 4:02 am (https://www.bezkoder.com/react-node-express-mysql/#comment-777)

Hi, we have CORS configuration for the front-end port 8080.
Then you just run the React app and open browser with url: `http://localhost:8080/`.

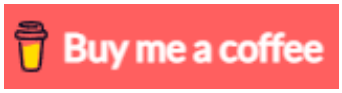
Jeffersom Guilherme Machado Barreto
September 11, 2020 at 1:08 am (https://www.bezkoder.com/react-node-express-mysql/#comment-4680)

It is not possible to connect the front and the back directly for security reasons, you have to throw the data from the back in an api and then consume the api in the front.

Comments are closed to reduce spam. If you have any question, please send me an email.

‹ Spring Boot, React & MongoDB example: Build a CRUD Application (https://www.bezkoder.com/react-spring-boot-mongodb/)

React Hooks CRUD example with Axios and Web API › (https://www.bezkoder.com/react-hooks-crud-axios-api/)



(https://www.buymeacoffee.com/bezkoder)

FOLLOW US

(htt
ps://
ww
w.yo
utub
e.co
m/c
han
nel/
UCp
0mx
9RH
0Jxa
Fsm
MvK
XA8
6Q)

(htt
ps://
face
boo
k.co
m/b
ezko
der)

(htt
ps://
gith
ub.c
om/
bezk
oder
)

TOOLS

Json Formatter (https://www.bezkoder.com/json-formatter/)

. Privacy Policy (https://www.bezkoder.com/privacy-policy/)

Contact (https://www.bezkoder.com/contact-us/) About (https://www.bezkoder.com/about/)