**PERN STACK**

P - Postgress

E - Express

R - React.js

N - Node.js (backing language)

Kada kliknemo npr submit u front endu (react app), to salje HTTP request (post) serveru tj resful API

Restful API koristi transfer protokol (HTTP request) kako bi pokrenuo CRUD operacije; daje instrukcije DB koji tip CRUD operacije treba da se uradi (npr create);

CRUD operacije se desavaju samo unutar baze podataka!!

CRUD - create, read, update, delete

submit => POST => Create

edit => PUT => Update

delete => DELETE => Delete

GET => Read

0. **Create Restful API**

mkdir server, cd server

npm init => prati pakete

npm i express pg cors

Express - nam dozvoljava da brzo napravimo sever u node.js u

cors - allows different domain apps to interact w each other (npr server radi na host 5000, a react na host 3000)

pg - to. conect our DB with our server in order to run to run **postgre** queries =>

const express = require('express');

const app = express(); //varijabla koja pokrece express

const cors = require('cors');

// middleware - da dobijemo od klijenta sta je kliknuo

app.**use**(cors());

app.use(express.json());

// za server

app.listen(5000, () => {

console.log(“sever started on port 5000”);

});

0. **Create postgres DB and table**

new file database.sql

CREATE DATABASE perntodo; (create a database)

CREATE TABLE todo(..); (create table called to do)

Komande sa postgres terminal:

u CMD: psql -U postgres

\l list all database in postgresql

\c move inside a database

\dt show table in database

kopi/pejst create DB, pa create table

0. **Connect postgress DB and server**

db.js

const Pool = require("pg").Pool;

const pool = new Pool({

user: "vika", ili postgres

host: "localhost",

port: 5432,

database: "perntodo"

});

module.exports = pool;

index.js

const pool = require("./db");

**! by using the pool we can run queries with postgres**

0. **Build Routes w postgres queries**

app.use(express.json()); //req.body

//ROUTES//

**//create a todo**

**async** - alati putem kojih se olaksava req i res; stvaki put kada create ili get data traje neko vreme, a async provides await koje ceka da se funkcija zavrsi pre nego sto nastavi

app.post("/todos", async (req,res) => {

try {

console.log(req.body);

} catch (err) {

console.error(err.message);

}

});

Postman: + POST http://localhost:5000/todos

Body

raw, JSON

{

“description” : “lalala”

}

app.post("/todos", async (req,res) => {

try {

const { description } = req.body;

const newTodo = await pool.query(

"INSERT INTO todo (description) VALUES($1) RETURNING \*",[description]

);

res.json(newTodo.rows[0]);

} catch (err) {

console.error(err.message);

}

})

VALUES($1) [description] - description je vrednost od $1; to ne mora da se doda, to je pg biblioteka koja nam daje da imamo dynamic datu tj. umesto toga moze direkt u CMD da se doda (insert into todo(description) values(‘hello’);)

SERIAL PRIMARY KEY automatski povecava brojeve kako bi bili jedinstevni tj. 1 treba da ucim 2 hello itd

RETURNING \* - koristi se kad god apdejtujemo, insertujemo ili dilitujemo datu, treba da se vrati tj return data

newTodo.rows[0] - zato sto nas zanimaju redovi (u postmanu)

**//get all todos**

app.get("/todos", async (req,res) => {

try {

const allTodos = await pool.query(

"SELECT \* FROM todo"

);

res.json(allTodos.rows);

} catch (err) {

console.error(err.message);

}

});

**//get a todo**

app.get("/todos/:id", async (req, res) => {

try {

const { id } = req.params;

const todo = await pool.query(

"SELECT \* FROM todo WHERE todo\_id = $1", [id]

);

res.json(todo.rows[0]);

} catch (err) {

console.error(err.message);

}

});

:id - dozovljava URL da bude dinamican, onda moze (req.params); npr http://localhost:5000/todos/random onda id:random; ne mora da se zove id, moze da se zove i subject

**//update a todo**

app.put("/todos/:id", async (req,res) => {

try {

const { id } = req.params;

const { description } = req.body;

const updateTodo = await pool.query(

"UPDATE todo SET description = $1 WHERE todo\_id = $2", [description, id]

);

res.json("to do updated");

} catch (err) {

console.log(err.message);

}

})

**//delete a todo**

app.delete("/todos/:id", async (req,res) => {

try {

const {id} = req.params;

const deleteTodo = await pool.query(

"DELETE FROM todo WHERE todo\_id = $1", [id]

);

res.json("todo was deleted");

} catch (err) {

console.log(err.message);

}

})

0. **Set up client side**

CMD: npx create-react-app client

Client i Sever moraju da budu odvojeni!!!

u App.js

import React, { Fragment } from 'react';

import './App.css';

function App() {

return <Fragment></Fragment>;

}

export default App;

naprave se u src folder za 3 komponente

u index.html se doda sa bootstrap css i JS, Popper.js, and jQuery (pre kraja body-a)

0. **Building the Input Todo Component**

cd client, npm start

\*\*posto me zezalo isla na client, copy file path, pa cd pejst path

u InputTodo.js

import React from "react";

const InputTodo = () => {

return <h1>Input Todo</h1>;

};

export default InputTodo;

u App.js

function App() {

return (

<Fragment>

<div className="container">

<InputTodo/>

</div>

</Fragment>);

}

u Input

import React, { Fragment } from "react";

const InputTodo = () => {

return (

<Fragment>

<h1 className="text-center mt-5">Pern Todo List</h1>

<form className="d-flex mt-5">

<input type="text" className="form-control" />

<button className="btn btn-success">Add</button>

</form>

</Fragment>

);

};

mt - margin top from 5

React hooks - useState;

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

const InputTodo = () => {

const [description, setDescription] = useState("")

description je state, setDescription je nacin da se state menja; useState pokazuje default value

<input type="text"

className="form-control"

value={description}

onChange={e => setDescription(e.target.value)}

/>

onda ispod const […] useState

const onSubmitForm = async e => {

e.preventDefault();

try {

const body = {description};

const response = await fetch("http://localhost:5000/todos", {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(body)

});

window.location = "/";

} catch (err) {

console.error(err.message);

}

};

i u <form className="d-flex mt-5" onSubmit={onSubmitForm}>

da bi se sve to testiralo, treba da budu ukljucen i server, tj i port 3000 za react(npm start) i port 5000 za server(nodemon index.js)

a da bi se proverilo dal radi, ide se u postgres \c perntodo, select \* from todo; i onda treba tu da se pokaze sta je kucano u input

i na devtools , Response treba da bude status: 200

0. **Build the List Todo Component**

u listtodos.js

import React from "react";

const ListTodos = () => {

return <h1>List Todos</h1>;

};

export default ListTodos;

u app.js

import InputTodo from "./components/InputTodo";

import ListTodos from "./components/ListTodos";

function App() {

return (

<Fragment>

<div className="container">

<InputTodo />

<ListTodos />

</div>

u listtodos.js

import React, { Fragment } from "react";

const ListTodos = () => {

return (

<Fragment>

izmedju fragment kopi pejst kod za basic tabelu sa w3 schools

<Fragment>

<table class="table mt-5 text-center">

<thead>

<tr>

<th>Description</th>

<th>Edit</th>

<th>Delete</th>

</tr>

</thead>

<tbody>

{/\*

<tr>

<td>John</td>

<td>Doe</td>

<td>john@example.com</td>

</tr> \*/}

</tbody>

</table>

</Fragment>

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

useffect - going to make a fetch req to restful API svaki put kada je ta komponenta (listtodos) renderovana

1 const ListTodos = () => {

useEffect(() => {

getTodos();

})

2 const ListTodos = () => {

const getTodos = async () => {

try {

const response = await fetch("http://localhost:5000/todos");

} catch (err) {

console.error(err.message);

}

}

3 const response = await fetch("http://localhost:5000/todos");

const jsonData = await response.json();

console.log(jsonData);

4 const [todos, setTodos] = useState([]);

5 const response = await fetch("http://localhost:5000/todos");

const jsonData = await response.json();

setTodos(jsonData);

6 zato sto usefeect stalno pravi req, da bi bio jedan rikvest

useEffect(() => {

getTodos();

},[]);

?????

7 mapiranje

brise se console.log(todos);

{todos.map(todo => (

<tr>

<td>{todo.description}</td>

<td>Edit</td>

<td>Delete</td>

</tr>

))};

0. **Build the Delete Button**

<td>

<button className="btn btn-danger">Delete</button>

</td>

2 <tr key={todo.todo\_id}>

3 <td>

<button className="btn btn-danger"

onClick={() => deleteTodo(todo.todo\_id)}

>Delete</button>

</td>

4 // delete todo function

const deleteTodo = async (id) => {

try {

const deleteTodo = await fetch(`http://localhost:5000/todos/${id}`, {

method: "DELETE"

});

} catch (err) {

console.error(err.message);

}

};

5 da ne bi moralo stalno da se rifresuje kada se brise, koristi se filter, koji namece uslov koji mora da se ispuni

try {

const deleteTodo = await fetch(`http://localhost:5000/todos/${id}`, {

method: "DELETE"

});

setTodos(todos.filter(todo => todo.todo\_id !== id));

tj pokazace sve todos osim onih koje pokusavamo da obrisemo

0. **Build the Edit Todo Component**

u edittodo.js

import React from "react";

const EditTodo = () => {

return <h1>Edit Todo</h1>

};

export default EditTodo;

listtodos.js

import EditTodo from "./EditTodo";

{todos.map(todo => (

<tr key={todo.todo\_id}>

<td>{todo.description}</td>

<td><EditTodo /></td>

<td>

<button className="btn btn-danger"

onClick={() => deleteTodo(todo.todo\_id)}

>Delete</button>

</td>

</tr>

w3 schools, bootstrap 4 modals

const EditTodo = () => {

return <Fragment>

<button type="button"

class="btn btn-warning"

data-toggle="modal"

data-target="#myModal">

Edit

</button>

<div class="modal" id="myModal">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h4 class="modal-title">

Edit Todo

</h4>

<button type="button"

class="close"

data-dismiss="modal">&times;

</button>

</div>

<div class="modal-body">

<input

type='text'

className='form-control'/>

</div>

<div class="modal-footer">

<button type="button"

class="btn btn-warning"

data-dismiss="modal">

Edit

</button>

<button type="button"

class="btn btn-danger"

data-dismiss="modal">

Close

</button>

</div>

</div>

</div>

</div>

</Fragment>;

};

listtodos.js

<td>

<EditTodo todo={todo} />

</td>

edittodo.js

const EditTodo = ({todo}) =>

todo je prop

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

const EditTodo = ({todo}) => {

const [description, setDescription] = useState(todo.description);

**Modal** koristi samo svoj id, mora sve gde pise modalid da se izmeni:

<div class="modal-body">

<input

type='text'

className='form-control'

value={description}

/>

</div>

<div class="modal"

id={`id${todo.todo\_id}`}

>

<div class="modal-dialog">

<div class="modal-content">

<button type="button"

class="btn btn-warning"

data-toggle="modal"

data-target={`#id${todo.todo\_id}`}>

Edit

</button>

**Da se transformise tj edituje opis:**

<div class="modal-body">

<input

type='text'

className='form-control'

value={description}

onChange={e => setDescription(e.target.value)}

/>

</div>

**Kada se klikne Edit treba da se posalje nova data, tj da se salje edit opisa kako bi se apdejtovao:**

<div class="modal-footer">

<button

type="button"

class="btn btn-warning"

data-dismiss="modal"

onClick={e => updateDescription(e)}

>

Edit

</button>

const EditTodo = ({todo}) => {

const [description, setDescription] = useState(todo.description);

// edit description function

const updateDescription = async (e) => {

e.preventDefault();

try {

const body = { description };

const response = await fetch(`http://localhost:5000/todos/${todo.todo\_id}`,

{

method: "PUT",

headers: {"Content-Type": "application/json"},

body: JSON.stringify(body)

}

);

window.location = "/";

} catch (err) {

console.log(err.message)

}

};

**poslednja stvar, kad se klikne edit, napise nesto drugo,kad se klikne van, upamti to napisano, a ne vrati kako je bilo ranije**

<div class="modal-header">

<h4 class="modal-title">

Edit Todo

</h4>

<button type="button"

class="close"

data-dismiss="modal"

onClick={() => setDescription(todo.description)}

>&times;

</button>

</div>

<button

type="button"

class="btn btn-danger"

data-dismiss="modal"

onClick={() => setDescription(todo.description)}

>

Close

</button>

<div class="modal"

id={`id${todo.todo\_id}`}

onClick={() => setDescription(todo.description)}

>

<div class="modal-dialog">

<div class="modal-content">