# CREATE FIREBASE PROJECT:

In Realtime Database, Rules must be:

{

“rules: {

“.read”: “true”,

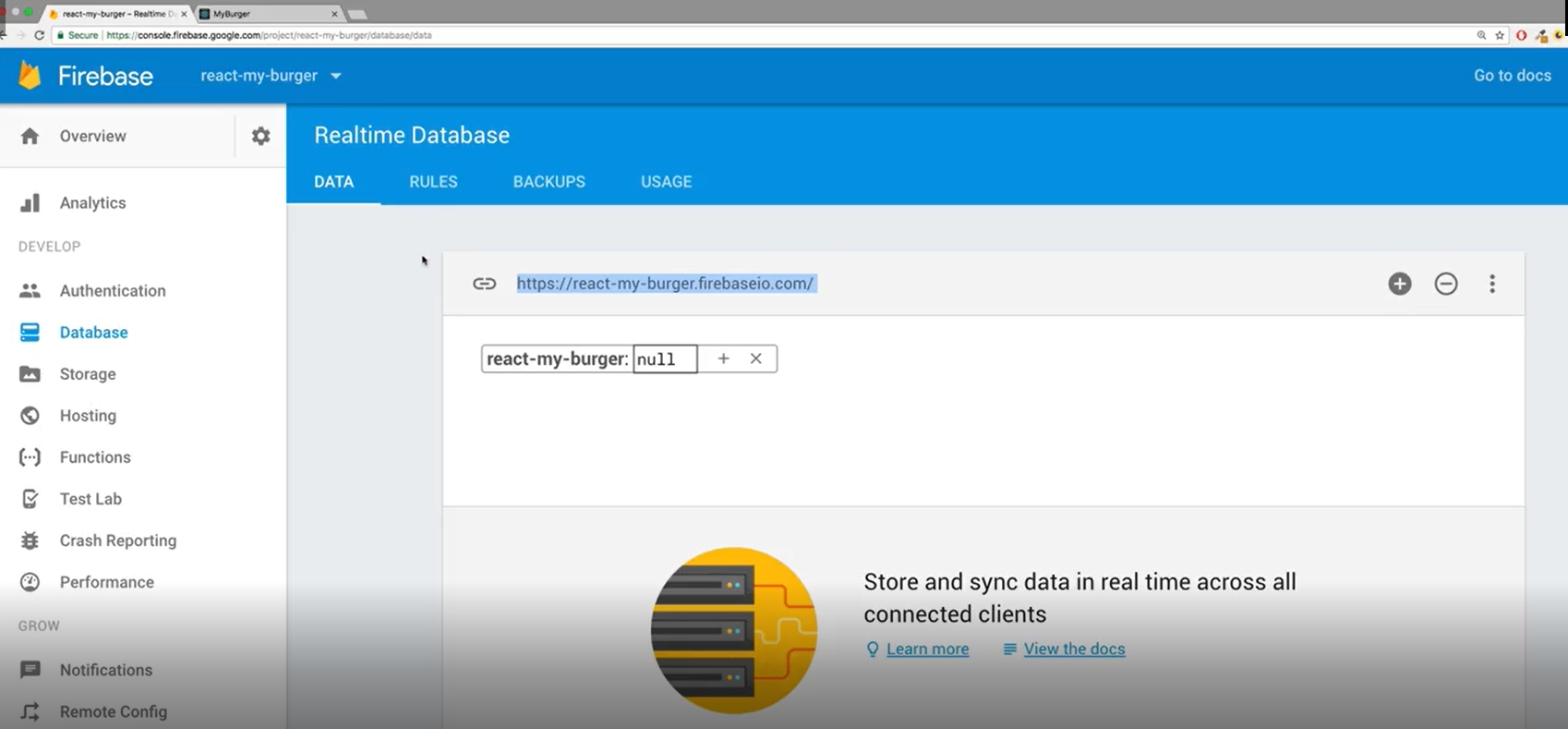
“.write”: “true”

}

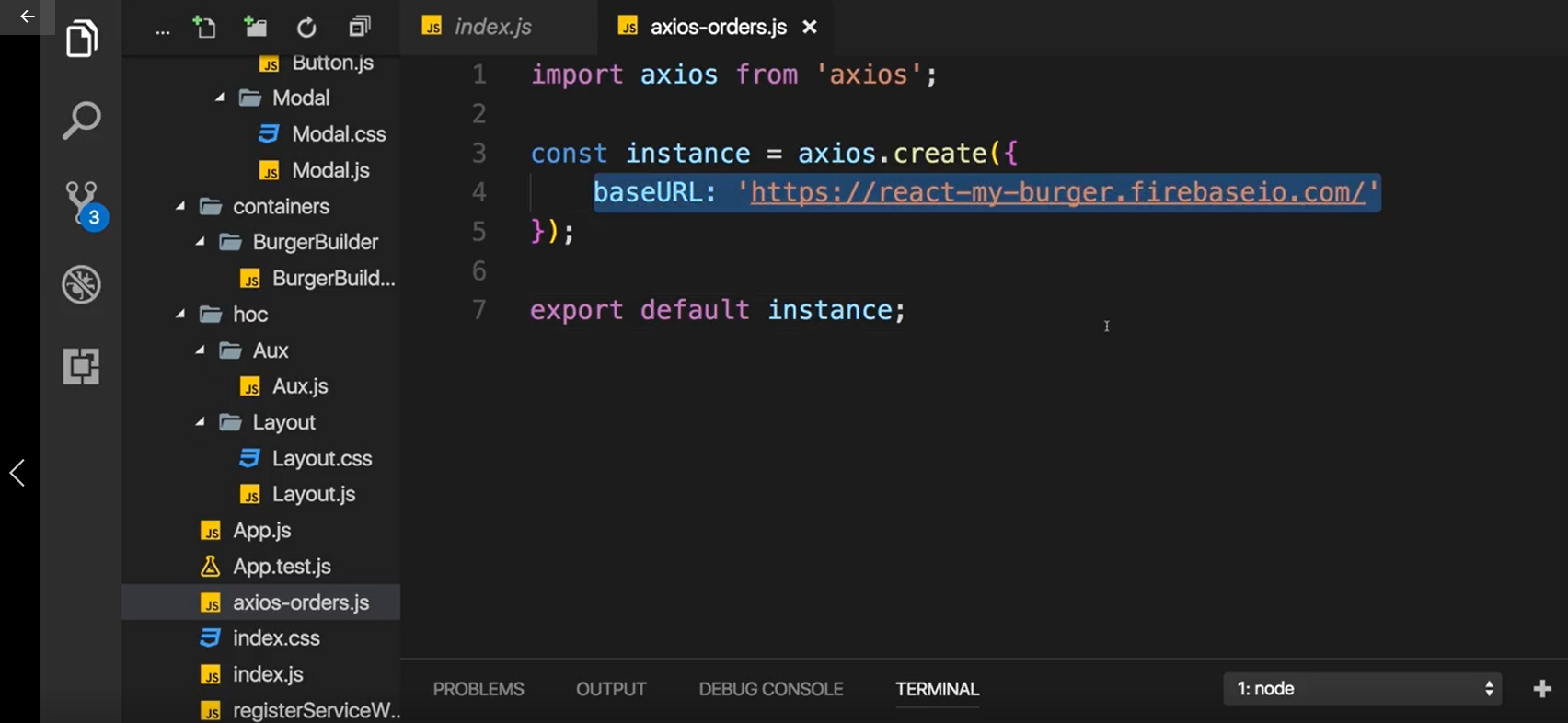
}

# CREATING THE AXIOS INSTANCES:

NOTE: npm install –save axios



IN CODE:



# SENDING POST REQUEST:

NOTE: Importing baseUrl of firebase from axios-orders.js



NOTE:

Blue color highlighted: JSX Object to passed into firebase database

Purple color highlighted: Posting user’s object to Real time database of firebase.

const purcharseContinueHandler = () => {

        // alert('continue');

        const order ={

            ingredients: state.ingredients,

            price: state.totalPrice,

            customer: {

                name: 'Shakti',

                address: {

                    street: 'Test',

                    zipCode: '4255',

                    country: 'Nepal'

                },

                email:'tes@gmail.com'

            },

            deliveryMethod: 'Fast'

        }

        axios.post('/orders.json', order)

        .then(res => {

            console.log(res);

        })

        .catch(error => console.log(error));

    }

# Displaying Spinner while Sending a Request:

NOTE: SEARCH CSS SPINNERS IN GOOGLE

## Spinner.js

import './Spinner.css'

import React from 'react';

const Spinner  = () => {

    <div className='Loader' >Loading...</div>

}

export default Spinner;

## Spinner.css

.Loader,

.Loader:before,

.Loader:after {

  border-radius: 50%;

}

.Loader {

  color: #521751;

  font-size: 11px;

  text-indent: -99999em;

  margin: 55px auto;

  position: relative;

  width: 10em;

  height: 10em;

  box-shadow: inset 0 0 0 1em;

  -webkit-transform: translateZ(0);

  -ms-transform: translateZ(0);

  transform: translateZ(0);

}

.Loader:before,

.Loader:after {

  position: absolute;

  content: '';

}

.Loader:before {

  width: 5.2em;

  height: 10.2em;

  background: #fff;

  border-radius: 10.2em 0 0 10.2em;

  top: -0.1em;

  left: -0.1em;

  -webkit-transform-origin: 5.2em 5.1em;

  transform-origin: 5.2em 5.1em;

  -webkit-animation: load2 2s infinite ease 1.5s;

  animation: load2 2s infinite ease 1.5s;

}

.Loader:after {

  width: 5.2em;

  height: 10.2em;

  background: #fff;

  border-radius: 0 10.2em 10.2em 0;

  top: -0.1em;

  left: 5.1em;

  -webkit-transform-origin: 0px 5.1em;

  transform-origin: 0px 5.1em;

  -webkit-animation: load2 2s infinite ease;

  animation: load2 2s infinite ease;

}

@-webkit-keyframes load2 {

  0% {

    -webkit-transform: rotate(0deg);

    transform: rotate(0deg);

  }

  100% {

    -webkit-transform: rotate(360deg);

    transform: rotate(360deg);

  }

}

@keyframes load2 {

  0% {

    -webkit-transform: rotate(0deg);

    transform: rotate(0deg);

  }

  100% {

    -webkit-transform: rotate(360deg);

    transform: rotate(360deg);

  }

}

## PizzaBuilder.js

import React,{useState} from 'react';

import Aux from '../../hoc/Auxilliary/Auxilliary';

import Pizza from '../../components/Pizza/Pizza';

import BuildControls from '../../components/Pizza/BuildControls/BuildControls';

import Modal from '../../components/UI/Modal/Modal';

import OrderSummary from '../../components/Pizza/OrderSummary/OrderSummary';;

import axios from '../../axios-orders';

import Spinner from '../../components/UI/Spinner/Spinner';

const INGREDIENTS\_PRICES = {

    salad:0.5,

    cheese: 0.4,

    meat: 1,

    bacon: 2,

}

const PizzaBuilder = () => {

    const [state, setstate] = useState({

        ingredients: {

            salad: 0,

            bacon: 0,

            cheese: 0,

            meat: 0

        },

        totalPrice: 4,

    });

    const [purchases, setpurchases] = useState({

        purchasable: false,

        purchasing: false,

        loading: false

    })

    function updatePurchaseState(ingredients) {

        const sum = Object.keys( ingredients )

        .map( igKey => {

            return ingredients[igKey];

        } )

        .reduce( ( sum, el ) => {

            return sum + el;

        }, 0 );

        setpurchases( { purchasable: sum > 0 } );

    }

    const addIngredientsHandler = (type) => {

        const oldCount = state.ingredients[type];

        const updatedCount = oldCount +1;

        const updatedIngredients ={

            ...state.ingredients

        };

        updatedIngredients[type] =updatedCount;

        const priceAddition = INGREDIENTS\_PRICES[type];

        const oldPrice= state.totalPrice;

        const newPrice = oldPrice + priceAddition;

        setstate({ingredients: updatedIngredients, totalPrice: newPrice});

        updatePurchaseState(updatedIngredients);

    }

    const removeIngredientsHandler = (type) => {

        const oldCount = state.ingredients[type];

        if( oldCount <= 0) {

            return;

        }

        const updatedCount = oldCount -1;

        const updatedIngredients ={

            ...state.ingredients

        };

        updatedIngredients[type] =updatedCount;

        const priceDeduction = INGREDIENTS\_PRICES[type];

        const oldPrice= state.totalPrice;

        const newPrice = oldPrice - priceDeduction;

        setstate({ingredients: updatedIngredients, totalPrice: newPrice});

        // PurchasesUpdate();

    }

    const purchaseHandler = () => {

        setpurchases({purchasing: true});

    }

    const purchasedCancelHandler = () => {

        setpurchases({purchasing:false})

    }

    const purcharseContinueHandler = () => {

        setpurchases({loading: true});

        // alert('continue');

        const order ={

            ingredients: state.ingredients,

            price: state.totalPrice,

            customer: {

                name: 'Shakti',

                address: {

                    street: 'Test',

                    zipCode: '4255',

                    country: 'Nepal'

                },

                email:'tes@gmail.com'

            },

            deliveryMethod: 'Fast'

        }

        axios.post('/orders.json', order)

        .then(res => {

            // console.log(res);

            //stop loading

            setpurchases({loading: false,purchasing: false});

        })

        .catch(error =>{setpurchases({loading: false,purchasing: false});

    });

    }

    const disableInfo = {

        ...state.ingredients

    }

    for( let key in disableInfo){

        disableInfo[key] = disableInfo[key] <=0

    }//{salad: true, meat:false, ...}

     let OrderSummary  = <OrderSummary

     price={state.totalPrice}

     ingredients={state.ingredients}

     purchaseCancel={purchasedCancelHandler}

     purchaseContinue ={purcharseContinueHandler}

     />;

    if(state.loading){

        OrderSummary = <Spinner />

    }

    return(

        <Aux>

            <Modal show={purchases.purchasing}

modalClosed={purchasedCancelHandler}>

                {OrderSummary}

            </Modal>

             <Pizza ingredients={state.ingredients}/>

             <BuildControls

             removeIngredient= {removeIngredientsHandler}

             moreIngredient={addIngredientsHandler}

             disabled={disableInfo}

             ordered = {purchaseHandler}

             purchasabled = {purchases.purchasable}

             price={state.totalPrice}

             />

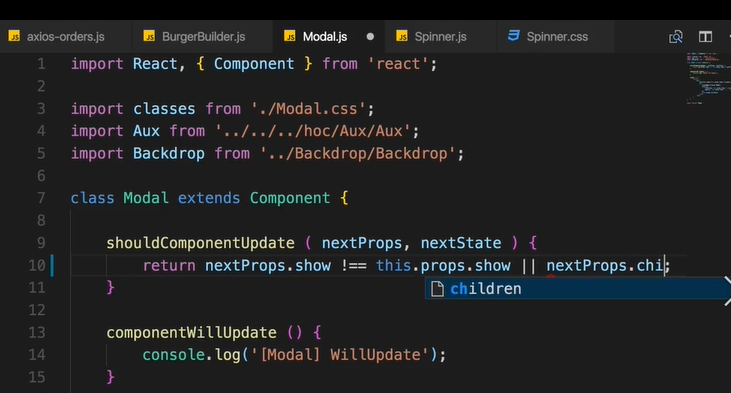
        </Aux>

    );

}

export default PizzaBuilder;

## Modal.js

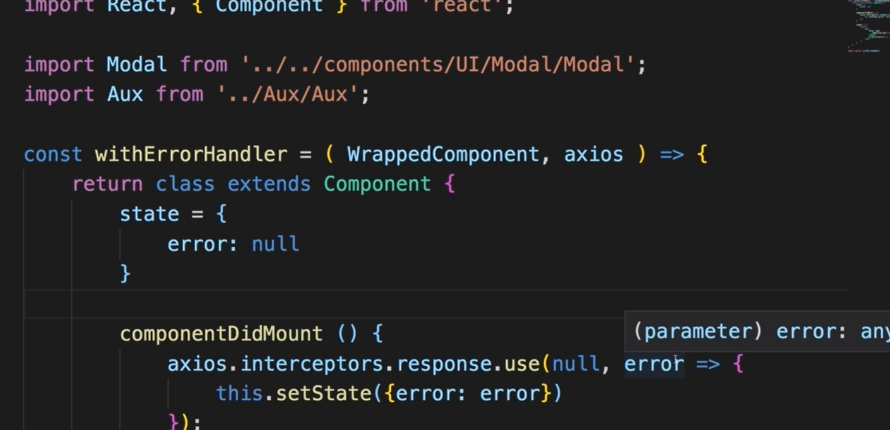


TO

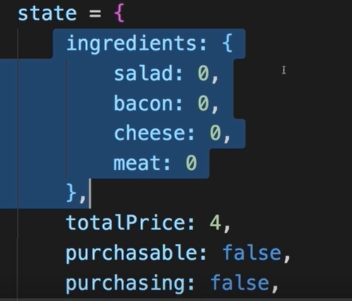


# HANDLING ERRORS:

<https://www.youtube.com/watch?v=Z78qvpyejQc>



# RETRIEVING DATA FROM BACKEND



INTO:

