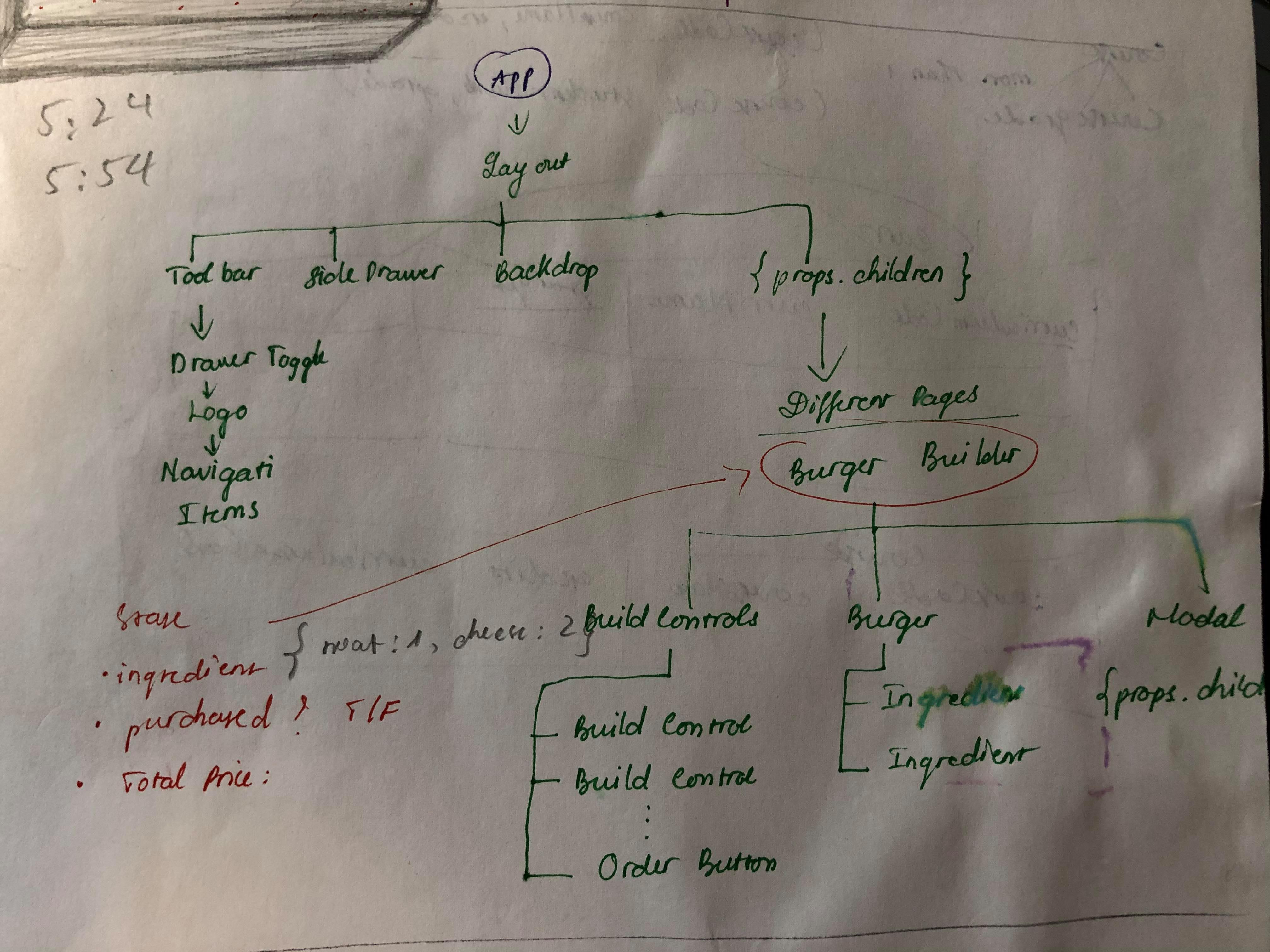
**BURGER APP Diaries**

1. **Structure**



* In which BurgerBuilder.js is a stateful component.
* Inside Layout.js and Aux.js components:

{this.props.children}

* Means that all other stuff inside <Layout></Layout> & <Aux></Aux> will be rendered under control/setup of Layout and Aux components.

1. **Steps and re-explanations**

* Setup project: Ejecting react project to be able to add CSS?
* Create layout components to style and hold other child component in a concrete style.
* BurgerBuilder.js is the container where state about price, total price, Boolean of purchasable and purchasing is set up. It returns the window showing selected ingredients, total price, cancel and continue button. All of these are wrapped inside <Modal> component, and <Aux> component.
* BurderBuilder.js has methods to add, remove ingredients, and handle purchases.
* BuildControl.js is where it returns MORE/LESS buttons for each ingredient.
* Outputting MORE/LESS buttons for each ingredient in BuildControls.js by declaring an array of object, mapping then, and returning them inside <BuildControl> wrapper.
* **NEW:** In Burger.js, there is:

//only return array of string of ingredients

    let transformedIngredients = Object.keys(props.ingredients)

* Which will transform Object to an array of string: [cheese, salad, meat, bacon]

  .map(igKey => {

        return [...Array(props.ingredients[igKey])].map((\_, i) => {

        return <BurgerIngredient key={igKey + i} type={igKey} />;

        });

    })

    //tranform the array of arrays to one single array

    .reduce((arr, el) => {

        //connecting elements

        return arr.concat(el)

        },[]);

* Then map each string into array:
* [Array(), Array(), Array(), Array()]
* 0[2] 1[] 2[] 3[]
* [{…}] [{…}]
* Ingredients is from BurderBuider, if not dynamically render ingredients, the code will just simple as:

Return (

<div>

<BurgerIngredient type=’bread-top’ />

</div>

)

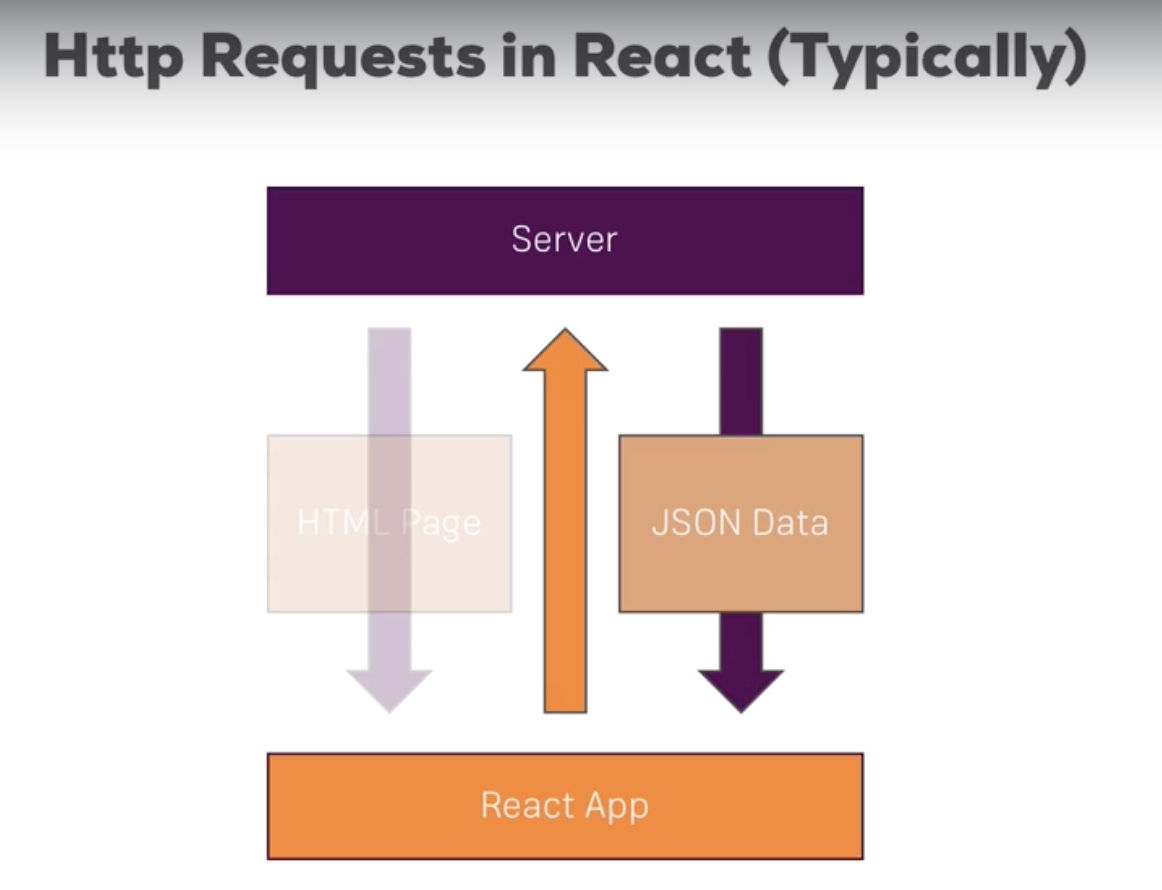
And number of each ingredient will be defined in state of ingredients in BurgerBuilder.

* Modal
* Backdrop
* Toolbar
* Navigation Items
* SideDrawer
* Toggler

1. **Improve performance**

* **We have states and props changes that trigger a re-rendering and UI-update.**
* OrderSummary will be re-rendered when we change selected ingredients, price, etc
* Adding componentWillUpdate() to check
* Also, in Modal, check if there are unnecessary updates by:
* Adding shouldComponentUpdate()

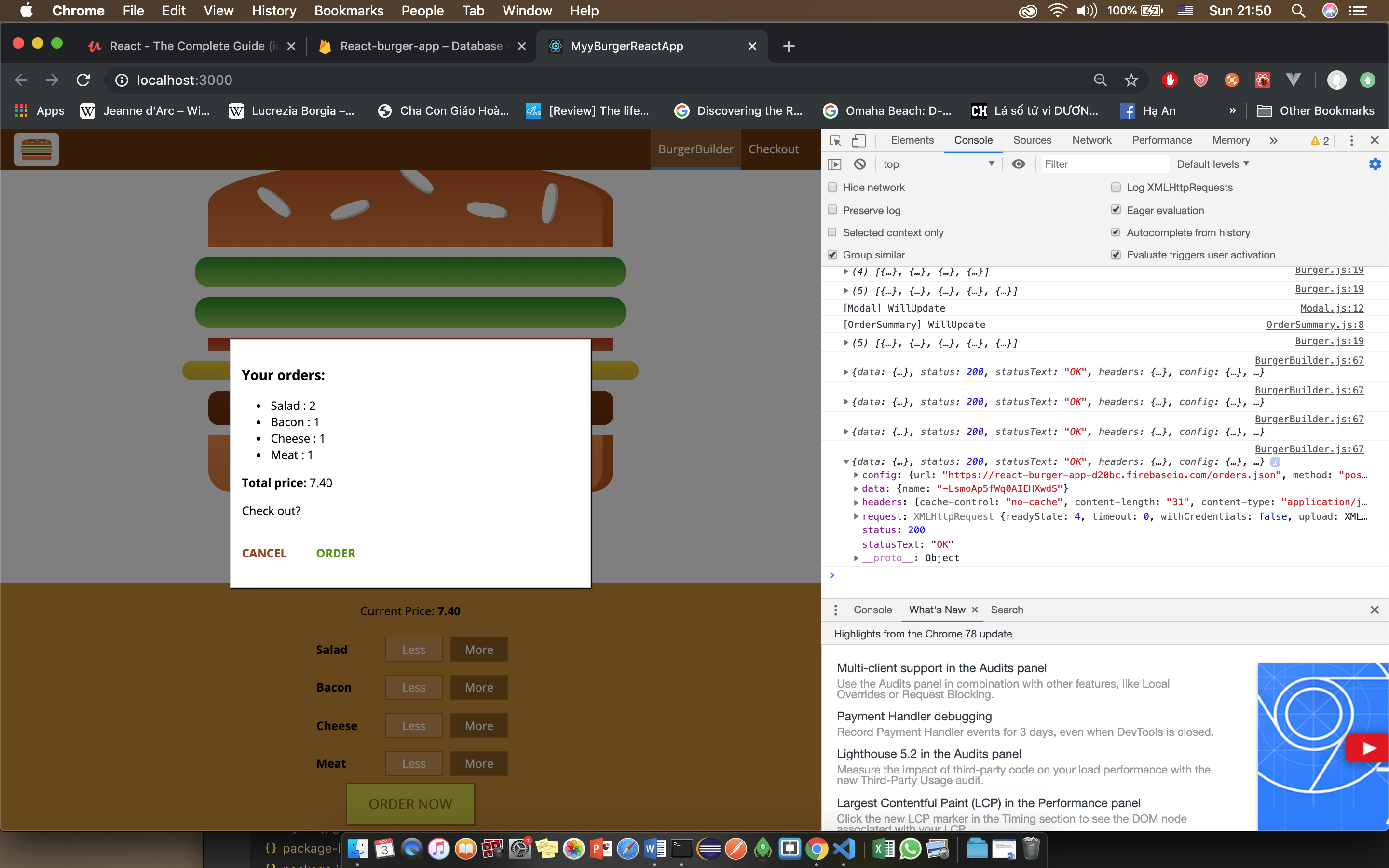
1. **Back-End, Reaching the Web**



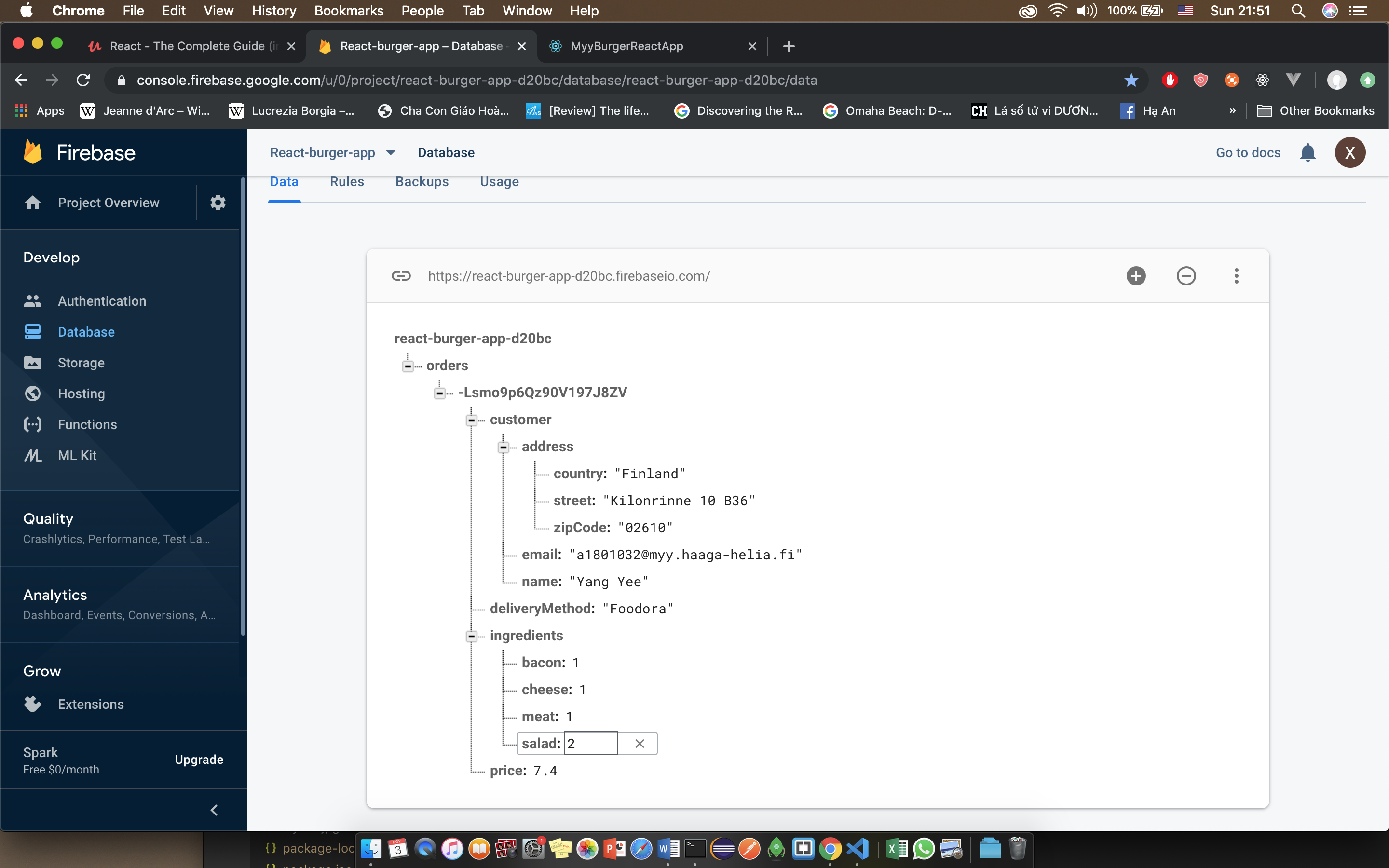
* **Use axios:**

Npm install –save axios

* Create Firebase Project: using realtime database.
* Create a js file which hold the base URL (global URL)
* Import axios-orders.js to BurgerBuilder.js
* **Post data to Firebase:**
* Instead of having alert in continuePurchaseHandler method, we going to post the order data to Firebase by POST method. It will create a list of data in my app in Firebase. But first, I have to create an object to load data from state.
* In POST method, we use an endpoint that is just our choice of name.json. The axios.post method having two parameters: the end point to post data, and the data object from customer input.



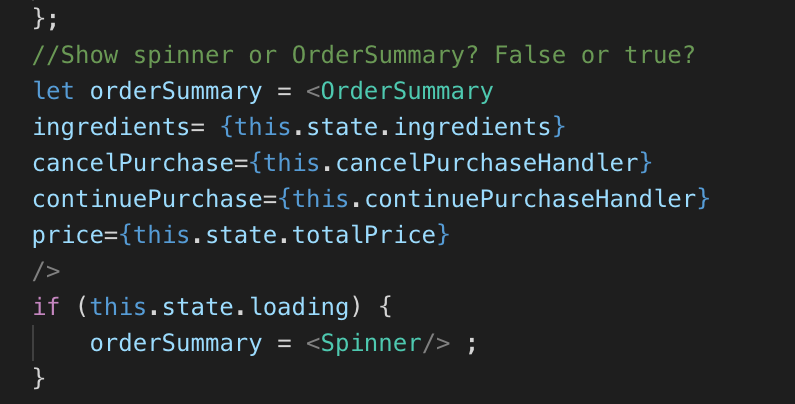
* Clicking ‘ORDER’, we have response data in console window.



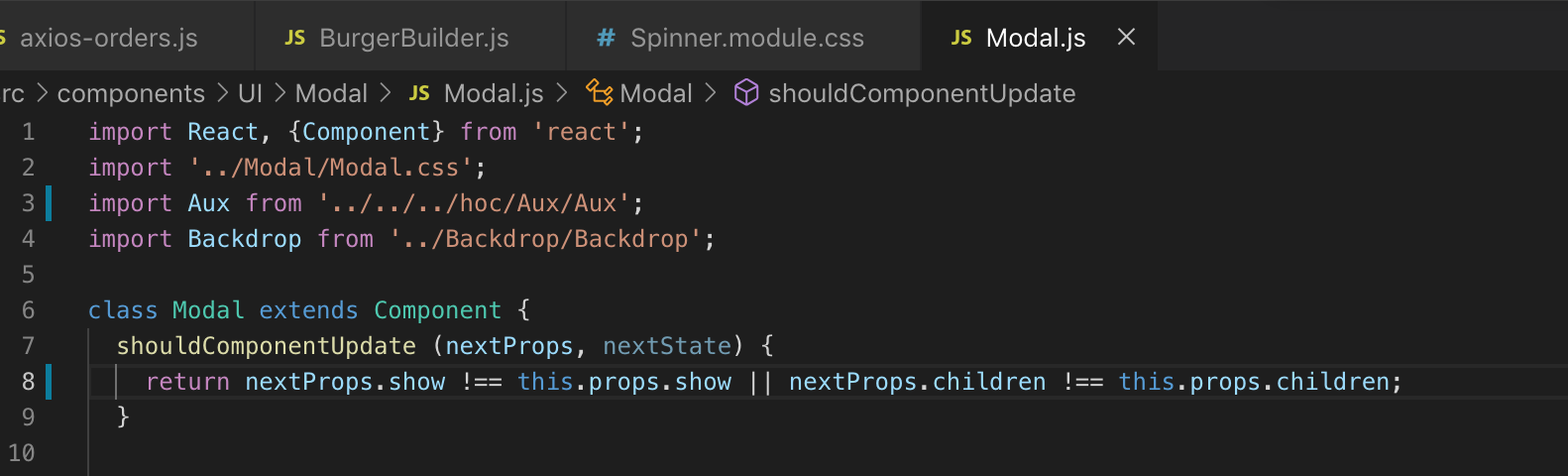
* I also got a basic data model in Firebase app.
* **Display a spinner when use Checkout:**
* The purpose is to give customer the feeling of something is happening.
* The spinner is defined inside my UI folder.
* Using available CSS source code from:

<https://projects.lukehaas.me/css-loaders/>

* The orderSummary is shown when loading is true; in case it is false, then the Spinner should be shown:



* Because orderSummary is rendered inside <Modal>, I have to make sure that the Modal will update:

****

* **Error Handler:**
* **Request data from back-end:**