Full-Stack Restaurant App Part 2

Video Transcript

Video 1 – Context for Cart

We're still talking about our flagship project and we're getting near the end of it now. We've managed to get the data from the Strapi database. So, we've got restaurant data and we've got the dishes. And you've seen how to set up the GraphQL queries to do that. Now, we want to look at putting dishes into the cart so that we can then go to checkout and pay. So, to do that, we need to make sure that we can use the AppContext Provider, useContext hook in React to share data about the cart. And so, let's take a look at that and see now how we put items into the cart and take them out. It's a very slick interface, I think. And this is a real app that you could deliver to, you know, your local community.

Video 2 – Add Dishes To Checkout Cart

So, let's take a look now at how we add dishes to the cart. Now, one thing to make sure is that when we put these images into Strapi, we specify that we're only going to have a single image per restaurant. If you don't click on that in Strapi, Strapi makes an array for the images. And when we're picking them up in our data here, it's going to be, cause a problem. So, you need to make sure you specify that there's only one image per restaurant in Strapi. Okay, now let's see how we add to the cart. So, we need to find out where this Add To Cart button is. And it'll be under dishes. And when we click on it, we'll see that it gets added to the Cart. We have one item. So that one times Swordfish grill. If we add another. Now, we have 2 times here. Notice that we don't repeat the Swordfish grill twice. We just put times 2, and that requires quite a lot of logic.

And we'll also see that there's two buttons here, plus and a minus. If I click minus, we're going to take one out of the cart. We're down to 27. If I keep adding plus, you will see now that I have 3, 4. So, let's take a look at how this cart is manipulated. So, let's take a look at dishes. Here we are in dishes.js, and I scroll down. So, here in dishes.js, the first thing we're doing is making a GraphQL query 'gql'query' to Strapi. And notice that it's actually specifying that '{id:' is the variable 'restId}'. And that's the number of the restaurant, the 'ID!' of the restaurant. Here, for example, we'll do a 'searchQuery' on the restaurant that matches the filter. So, what I mean by that is if I type a k here, and we're going to come up with Karma, and we get the dishes for Karma. So, that all works.

Now, let's take a look at how the information about the dishes that we've chosen. So, we need to find Add To Cart. And we said that that was in dishes.js. So, let's look at dishes.js now. So, here we are in dishes.js. And we found the '< Button >' '+ Add To Cart'. And that calls 'addItem'. And 'addItem' is way back up in '_app.js'. Here it is, 'addItem'. So, we're now going to 'addItem' to the

© MIT xPRO 2020. All rights reserved.

'cart'. And the 'cart' is part of the 'useContext'. But also we're going to store it locally here, in the 'state'. We're setting this 'state' to this object with an property 'cart:', and this will be the 'cart' and all its items. So, setting the 'state', as you know, will cause a re-render.

And we want that to happen because we're going to add something into the cart and we need to actually display what has been added to the cart. The logic in here is fairly complicated. So, if we've already got an item in the cart, we don't want to add the full item in again, we just want to increase the quantity that is in the cart. So, you will need to go through here, and you'll see here that we're creating another property in that item called 'quantity'. So, for example, if it's a swordfish, at the moment, we don't have any 'quantity' in that, and we're adding it in here as a property. So, we're getting the 'item', and we're adding 'quantity' to it. There's some problems. We can't just do 'item.quantity' because 'item' is coming through here, and it's a property of React that won't let us change some of those properties that are coming through.

So, take a look at how this 'setState' and the 'cart' works. And now, the question is, how do we render the cart? Well, the cart is going to get rendered in cart.js. So, here's 'Cart()'. Here's the 'function' so that we need to understand how does that card information get into here. Well, it comes through 'useContext' that you can see. We've got 'cart', and in fact, we've got functions that 'addItem, removeItem'. So, that's the way the information is passed through. Take a look at printing out. For example, here we print out the cart. So, we want to make sure that the information is being passed from _app.js to cart.js. And that's a reasonably tricky part, but once you understand exactly how that's done, then you will come a long way. Okay, so here we're rendering the cart, and you'll see that we've got a remove 'item' here that changes the 'quantity'.

And we've got another 'addItem'. So, these are the plus and the minus signs. Okay, there's the '+' sign, yup so it's part of that '< Button >'. And here's the ' - ' sign, that's part of this '< Button >'. So, I'm referring to we add something to the Cart. Here, I'm referring to that plus and that minus. So, we click the plus. Now, we've got two of them in there. So, the logic for that is all in cart.js. Okay, so understand how we move things into the cart and can move them out. The logic, obviously, if we've only got one left in the cart, then we have to remove the whole item from the cart. Whereas if we've got three, say in the cart and we remove one, we don't remove the item, we just change the number, the quantity that is in the cart.

Video 3 – Stripe Credit Card Processing Set Up

If we want to checkout and actually pay using a credit card, we're going to use Stripe. So, stripe.com allows us as a test user to set up an account and use a fake credit card so that we can actually pay for the dishes that we order. So, let's go to Stripe and sign in, get an account, and we also need to get a security token. There's a couple of security tokens that you can develop, and we'll now take a look and show you how to do that. And then we'll be finally able to use our app and actually pay for the dishes that we want to order.

MIT xPRO

Video 4 – How Stripe Works

So, we've seen how to order dishes and add them to the cart. And here we see we can, we've got 4 Dragon rolls. So, if we want to make it 3, now, it's 3 Dragon rolls. So, let's, for example, go WoodsHill and add some Swordfish. Here we go, and it costs us $101. Let's go out and order now, and let's go through this process of actually paying. So, now we're at checkout and we've got our order here. Now, we can checkout. And now, we're going to go to Stripe and pay with our credit card. We've got a dummy credit card here, but we'll take a look at filling this in. So, let's put New York, State is New York, and the card number. Now, the fake card number is just 4242 repeated. And we'll make it. Okay. So, now we filled this in and we want to confirm our order. Now, confirming our order, I made a mistake there.

Confirming our order does a number of things. One, it's going to write into the Strapi database. So, let's take a look at the scrappy orders. So, I need to go to localhost:1337. So, let's look at the orders. So, the last order was number 6, and the City was Main. Let's now place our order. So, I want to confirm the order. Now, the other thing it's going to do is it's going to go to Stripe and place our order there. Let's go to Stripe. And I set up in Stripe and I showed you how you can also set up a dummy account. But we need to log in to here. So, I go to the dashboard and I'll look at Payments. And at the moment, the last one, October 29th was $65. So, let's now go and pay here. So, we're going to pay $101. We confirm the order.

Okay. There was no error shown so it means that it's gone through. Let's go and take a look at Strapi. So, let's refresh Strapi, and we'll see we've got number 7, Main St, New York. And if we look at the order, will see that it's got the number of dishes, ' "quantity": 2' for that, ' "quantity": 3' for that, and the amount was 101.00. It actually multiplies by 100, that's the way they keep these things. And it gives us a Charge_id as well, that came from Stripe. So, here's Stripe, our payments. And let's refresh that page. And we'll see here is the $101 charge which was made today, which is October 30th. Okay. So, we've succeeded all the way through and we can take a look at the code that placed that order. Let's take a look here.

This would be in the cart form, and so if we look here first, it's going to go to 'stripe', and we get a ' "token" ' back from Stripe, or we get our 'Cookies', ' ("token") ', we make the 'userToken'. And now, we go to 'orders'. So, this is 'stripe' here. But all of this now a Strapi. We're making a 'fetch' to Strapi to '/orders' ', and we're posting and we get our '{useerToken}'. So, we're putting our JSON web token in here. And in the body, we're putting all the dishes and the '.address', '.city', and '.state', et cetera. Okay. So, we've succeeded in that and we've paid and it's gone into Stripe, which is quite slick. And we've got our record in the Strapi database if anyone questions whether what they'd bought. We have all the details in Strapi. Okay, so that's the full cycle, and I'll set you an exercise to update this.

© MIT xPRO 2020. All rights reserved.

Video 5 – Stripe Back End Set Up

So, let's continue to build the backend of our food ordering app. So, we're using Strapi, and we're also now going to use Stripe to actually checkout. So, last time we got to this point where we could make an order, but we couldn't yet confirm that order. And what we mean by confirm is we need to checkout. We need to put it into the Strapi database. So, for example, here, we need to be able to create orders. So, let's see how we do that. So, the first thing is we need to create a content type called Order. So, here it is. And it consists of dishes, which is a JSON object. We need to have the amount. So, let me show you, the amount is a big integer. And we need the address, which is Text, and the city. And we're going to write into this. So, we need to have the ability to create Orders. We need to be able to write orders in here.

So, to do that, we need to go to Settings, go to Roles, and we've got Public and Authenticated. We need to make sure that for both of those with order, we need to make sure we have create ticked. I've selected them all just to be safe, but we definitely need that create. I suggest first, you select them all, and then later when everything is working, we go back and deselect some of these, but we need the create one. And we Save that. And we can also go to the Public one just to be sure, and Select all there. But we should be an authenticated user. So, we will be using the authenticated role. Okay, so that's the Roles, and we've set up Orders. Now, we need to add a capability to take this order and interact with Stripe. So, here's Stripe. So, if you go to the Stripe dashboard, you need to be able to create an account.

And like you can see here, they allow you to create an account just for testing. So, go and create an account under Stripe. And you need to get a secret key. You need to go to Developers, and you need to get an API key, a Standard key. And here, the test key token. So, we need to bring that up. So, this is a Publishable one that you can give to anyone else, but we need this one. So, I need to log in to get that. But I can't do that. Make sure you get that and copy this Secret key. So, get both of those keys, and then we need to go back to Strapi. And here's my Strapi running. Now, we need to, let's kill that. So, I'm in my backend database. We run that. Once we've installed Strapi, then we need to, actually, come back to a backend. So, here I am in the backend, and we need to navigate to the api. We need to go to order and then controllers.

And under controllers, we need to go to this file order.js. Now, if you look under here, it tells us how to create this file order.js. So, we just did 'npm i stripe'. Now, we need to copy this code, 'Order.js'. So, we copy this, and now, we go back and we paste it into here, and then we get our secret key. It starts with ' "sk_test_', then we need to paste it in here, okay? And save that. Now, we need to recompile. We'll go to here, and we'll do a 'yarn build', and we'll do a 'yarn develop'. You can do a 'yarn develop' or a 'yarn start'. I prefer to do develop. Right, now, we can go back to our order here. Let's, I'm going to go back and create it again. Notice, I've logged in already. I go, and now, I'm going to add to the cart a broccoli and a cod. Now, we can go to order. Let me go back, I didn't get the broccoli.

So, I'm going to add this to the cart. There's the broccoli and the cod. Let me add another broccoli. So, $52. Now, we can come, and we'll see this CONFIRM ORDER is highlighted. For the address, we can put in any address. It doesn't actually matter. I'm going to put where I live, but, and now for the card number, you need to put in 42 4242, and this is a test card, and keep going with 42s. And it should look like that. Now, we can confirm the order. Now, this is going to do a number of things. It's going to write it into the Strapi database.

Notice, we don't have anything for, I think it's $58. It's going to round it up by multiplying by a 100. So, it's going to, the last two digits, a cents, and the first two are dollars. We notice we've got 4 and 5 in here. Now, let's confirm this order. And now, if we go back to the content, and we'll see we've got a number 6 in here for $52. So, it's written in here. Now, if we go to Stripe and we look at our Payments. We'll notice we just made a payment for $52. So, we've got everything going through, and it's recorded. And basically, if I were a merchant, if I were the restaurant, I would now have $52 credited to my account in Stripe. So, we've got complete integration of the forward and backend. So, our backend now is stable, and it's working. We've finished, and now, we can concentrate on the front end.

Video 6 – Strapi Register User

So, let's take a look now at the frontend of our app. It's running on localhost:3000. Now, what I want to focus on now is how Sign up and Sign in works. So, let's go to Sign up. So, what Sign up is going to do is it's going to allow us to add Users to Strapi so that it knows about them, okay? Let's do that. So, let's Sign up. And we'll be, we're signed up. And it automatically logged us in. Let's Logout and see if we can Sign in. So, here, yup, we can Sign in. So, we've registered. Now, let's look in Strapi, and let me refresh this. We should see, yes, we've got a new user. John is registered. Okay. So, let's go and look at the code to see where this is happening. So, I'm in the frontend. And we've got a number of pages here that Next has specified. And I want to focus on these two register.js and login.js.

So, register.js, we go to that form. And this is, you can see it's a React component and it's called, the web component is called 'Register'. And here's our 'return', and it's got this '< Container >'. Notice there's an image here which I'm pulling up from 'MIT_logo', you can change that. And we'll have an exercise where you change that image. And let's see now what it's doing. It's asking us to input the data. We input a ' "username" ', an 'onChange'. It's using React's 'setData'. So, this will allow us, if we go back up and see here, we've got '[data, setData]', which is 'useState'. And we're going to set eventually the 'email:', 'username:', and 'password:'. And we're going to do them one at a time. So, the first one is '.username}', then it's going to 'setData', 'email:'. Notice it's using the spread operator and then adding to whatever is already in data.

This value and the 'password:', it wraps it all up. And then it calls 'registerUser'. And that's going to push it into the Strapi database. Okay. So, that's it, that's the register. And let's find out where that is. So, that needs to be in one of these libraries. Here it is, '{ registerUser }', and it tells us in '../lib/auth";'. So, we get a lib, an auth.js, and we find 'registerUser'. Okay, so this is going to use 'axios' notice to post it to Strapi's and the root in Strapi is '/auth/local/register' ' and it passes the 'username', 'email', and ' password' in. And then, it gets the response '(res)'. And in the '(res)',

© MIT xPRO 2020. All rights reserved.

there's a JSON Web Token, and we use the 'Cookie' capability to '.set' ' " token" ' in 'Cookie' as that JSON Web Token.

And then it takes us back to the login page, the default route on 3000, okay? So, that's the authorization or the registration rather are now also it's going to be used for login. Same thing again, except that this time we're going to post an '{ identifier, password} '. And again, we need to set JSON Web Token that comes back from Strapi. So, we're setting this into our cookies on the client-side. So, some of this is running on the web server side, and some of it's on the client, and to make sure that this is on the client. If there's no 'window' object, so the server won't have a 'window' object if it's undefined, we return, we don't do anything. But if we're on the browser side, then there'll be a 'windows' object.

And we'll do all this and we'll '.set' the 'Cookie' on the browser. So, that's how it's setting the cookies. And as we saw as well, now we can look at login.js, login.js is very much the same. We get our little image or logo from MIT, and now you enter your username and password. Here we go. And it's going to, so there's those. And we're going to call 'login', and we're pretty sure 'login' is going to be here and here it is. We exported 'login', and so that's the 'link'. So, that's taken care of register.js, login.js and auth.js Okay, so that's one important part of how we maintain security. We're doing it by posting a JSON web token into the cookie. Okay. So, that allows client and server to recognize each other and verify that it's somebody that we know that's talking to us.

Video 7 – Rebuild Database

So, in this exercise, I want you to rebuild the restaurant exercise that I went through with you so that you can order dishes and checkout, and pay on Stripe. Okay, so clone the repo that we're giving you. And what you should see is that you should see this directory. So, there's the backend and this is the front-end. So, what I want you to do first, is to go to the backend. So, we're in the right place. And I want you to do 'npm install'. So, there's a package.json that you can look at and follow. I'm running node '12.16'. So, check out the version of node that you're running. So, anything above 12 should be fine. And then we can, once we've done that, install 'npm run start'. So, this is 'strapi'. So, it should start up Strapi on 'localhost:1337'. And you can always hit that. Just to check that it's up but it is running.

Now, we can come to the front-end, do the same thing, 'npm install'. And then 'npm run' and do run 'dev'. And that's now running on '3000', 'localhost:3000'. So, let's go to 3000, and we should see this. Okay, so once you've done that, I want you to go to Strapi. Now, since you don't have administrative privileges on that Strapi database because it's mine. And remember to fill in, and remember your password for superuser on Strapi. But you can follow the instructions that we gave before on how to build that dataset. And I want you to have three restaurants in your own location and at least nine dishes. So, three per restaurant. So, step through exactly as I showed you before and build up the data so that you can get everything running with your own restaurants.

© MIT xPRO 2020. All rights reserved.

Video 8 – Deploy to Docker Container Introduction

So now, we're at the end of our flagship project. We've managed to order dishes from restaurants and to checkout using Stripe and to pay using a fake credit card. Now, of course, in reality, we need to use real credit cards if we're deploying this. But deploying is the next step. So, let's take a look now at how we can put both the frontend and the backend into a Docker container. So, we're going to create images for the frontend that run in one container. And then we're going to show you how to set up Strapi in another container. And finally, MongoDB, the database that we're going to use in yet another container.

Video 9 – Deploy Restaurant App to Docker Container

So now, we'd like to deploy our application. So, we've got our frontend part, which is Next React, and we've got our backend, which is Strapi, and the database. So first, let's take a look at how we might put our database and Strapi into a Docker container. So, I've got a docker-compose.yml file here. And the first part I'm showing here builds Strapi. Now, there's a 'strapi' image that's already in Docker Hub. So, we can pull down the 'strapi' image and we'll then load it into the container and run it on port 1337. Now, we're going to need passwords for the database. And so, we're going to use a '.env' file to store those, and you'll see that. And that will set these 'environment:' variables, '{DATABASE_CLIENT}', '{DATABASE_NAMET}' et cetera. Okay. So, all those are going to be pulled in from a file called '.env', which I'll show you.

So, that'll take care of Strapi. And in the same file, the docker-compose.yml, docker-compose.yml. We've got an example here, we're spinning up MongoDB. And there's a 'mongo' image already in Docker Hub. So, that's no problem, but we do need to provide a username and password for that. And it's going to spin up on port 27017. So, let's take a look at the code. Now, we don't need any code in the sense of JavaScript because the images are already in existence in Docker Hub. So, all I need to do is I need to do Docker Compose Up. And it's going to run those, and it's going to go and get those images. Now, I already had downloaded them previously, and so it's quite fast. It'll take you a little longer. But then if we go to our Docker extension, then we'll see that it's already spun up. A container with 'strapi' in it, and a container with 'mongo'. So, we've got these two containers running.

So, we'll be able to hit 'strapi' on 1337. Now, let's go to the second part where we need to do the frontend. So now, in the frontend, we need a Dockerfile. And here, I'm showing you the Dockerfile and it's quite straightforward. Now, in this case, we need to load up our app. So, we've got a lot of code here, but here's the Dockerfile that we need to execute. So, I'm going to right-click on this and scroll down, and I've got a build image here, which is just off the screen but I'm going to click on it and it's going to build, we'll see here, build a Docker image. My machine is, the fan is spinning up because it's doing a lot of work. It's copying a whole load of file. So, you'll see that it successfully tagged our Docker image. And if we go to the Docker image we'll find it under restaurantapp, a few seconds ago. So now, let's, I will need to spin this up and get a container for this.

© MIT xPRO 2020. All rights reserved.

So, I'm going to Run this. So now, it's spinning up. You'll see a container here with the restaurantapp. So, I've got three Docker containers running now. Let's give that a few seconds and we can go visit it. So, let's go to 1337/admin. And it's asking us to set up Strapi. So, we're running in the container, we can set up Strapi. Fine. Let's now just take a look at the frontend. So, that's localhost:3000. So here, we are, localhost:3000. It's giving us an ERROR because we've got no data in the Strapi database at the moment. But as soon as we put data that will also go away, we can check some of the other, the other Sign Up form. Again, we don't have that image in the Strapi database yet. But everything else, it's running in the containers. So, that was putting the frontend into the container, putting Strapi into another container, and finally, putting MongoDB to backup Strapi. Okay, so that's deploying our application that we can now deploy it into any Cloud quite easily.

© MIT xPRO 2020. All rights reserved.