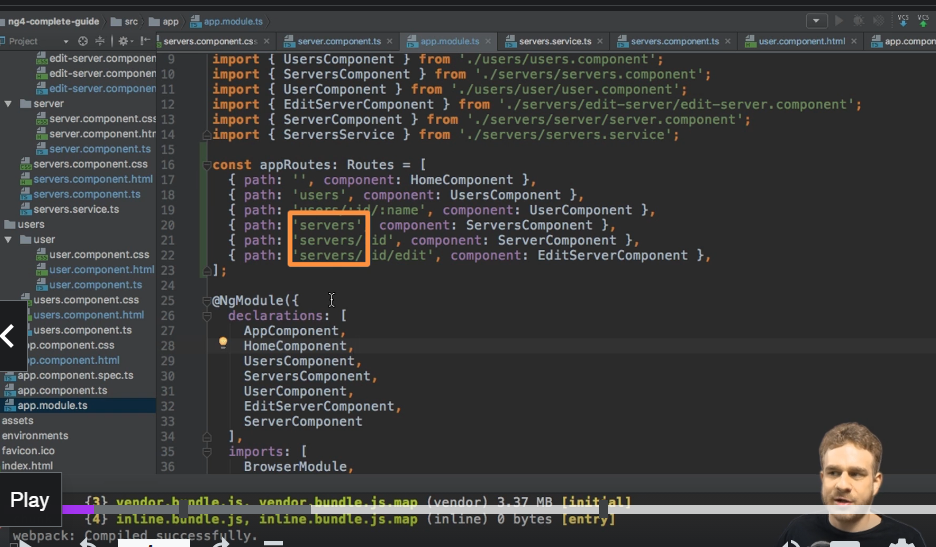
**139. Setting up Child (Nested) Routes**

* -: We improved out app quite a bit.
* But the issue we have is that if you click on a server or a user, you load a brand new page.
* It would be nicer if it would load it next to this menu.
* So we need some nested routing.
* We can also see that in the app module, we kind of have some duplication here.



* All these routes here start with servers, and all these routes with users.
* So it would be nice to nest them and to have some child routes which all start with servers down here in this case, for example.
* Let's ***add such child routes.***
* To do this, I'll go to my servers component route, to the slash service route, and I'll add another property.
* Children.
* Children takes another array of routes, so now I can simply take my two other service routes here, and add them inside of this array as children.
* Now, servers should now be removed, because that is now always prepended at the beginning anyways.
* So we have to make sure to get rid of it here.
* And now we have just ID and ID edit.
* So this is now already better because now we group that together.
* It gives us more than this visual grouping here though.

Text

Description automatically generated

* You notice that on the servers here, we have the servers component as the, well, component to be loaded still.
* **The question now is where will these server components be loaded**, then? Because that is yet another child route of our service component.
* And you can see that this is an issue.
* If we reload our app here, everything's working.
* If I click on test server, now we get an error.

Graphical user interface, text, application

Description automatically generated

* Because now the *error tells us that it cannot find an outlet to load our server component in.*
* And indeed it can't because the only route or outlet, the only hook in our code where it should load components, is in our app component here.

Text

Description automatically generated

* ***Now that is reserved for all our routes on the top level.***
* So, slash nothing, slash users, users ID name, and servers.
* But the child routes of servers need a separate outlet, because they can't override the service component.
* Instead, they should be loaded nested into this service component.
* And that actually is the behavior we want.
* So we can quickly get there by going to the service component HTML file.
* And here where we load the edit page or the app server page.

Well here I will simply comment out all that code and instead add a **139. Setting up Child (Nested) Routes**

* -: We improved out app quite a bit.
* But the issue we have is that if you click on a server or a user, you load a brand new page.
* It would be nicer if it would load it next to this menu.
* So we need some nested routing.
* We can also see that in the app module, we kind of have some duplication here.

Text

Description automatically generated

* All these routes here start with servers, and all these routes with users.
* So it would be nice to nest them and to have some child routes which all start with servers down here in this case, for example.
* Let's ***add such child routes.***
* To do this, I'll go to my servers component route, to the slash service route, and I'll add another property.
* Children.
* Children takes another array of routes, so now I can simply take my two other service routes here, and add them inside of this array as children.
* Now, servers should now be removed, because that is now always prepended at the beginning anyways.
* So we have to make sure to get rid of it here.
* And now we have just ID and ID edit.
* So this is now already better because now we group that together.
* It gives us more than this visual grouping here though.

Text

Description automatically generated

* You notice that on the servers here, we have the servers component as the, well, component to be loaded still.
* **The question now is where will these server components be loaded**, then? Because that is yet another child route of our service component.
* And you can see that this is an issue.
* If we reload our app here, everything's working.
* If I click on test server, now we get an error.

Graphical user interface, text, application

Description automatically generated

* Because now the *error tells us that it cannot find an outlet to load our server component in.*
* And indeed it can't because the only route or outlet, the only hook in our code where it should load components, is in our app component here.

Text

Description automatically generated

* ***Now that is reserved for all our routes on the top level.***
* So, slash nothing, slash users, users ID name, and servers.
* But the child routes of servers need a separate outlet, because they can't override the service component.
* Instead, they should be loaded nested into this service component.
* And that actually is the behavior we want.
* So we can quickly get there by going to the service component HTML file.
* And here where we load the edit page or the app server page.
* Well here I will simply comment out all that code and instead add a router outlet here.
* This now adds a new hook which will be used on all child routes off the route being loaded on the service component, which of course is our slash service route here.
* So all these child routes will be loaded in this router out of place now.
* So if we save this, and now we let this reload.
* You don't see anything here.
* But if I click a server, it's now loaded next to the menu because this is where we added our second router outlet.

Graphical user interface, text, application, email

Description automatically generated

* And this is how you can easily add child routes.
* Let's do the same for the user routes now.
* In the app module I'll add children here.
* So this property which takes an array of routes.
* And I only have one nested route here, but that's fine.
* So we'll add it here.
* Get rid of the users at the beginning.
* Just have slash, well, oh, and the ID and name, the two dynamic parameters, and load the user component.

Text

Description automatically generated

* And now in my users component HTML file, I'll replace my app user here with another router outlet where all the user-related child routes, or the one user-related child route we have, will be loaded.
* So if we save this, you now see we load the user next to it, and that is why it is super important to dynamically update the ID and so on.

Graphical user interface, application

Description automatically generated

* Because the component wasn't exchanged, we were able to switch the loaded user while this old component, the user component, the single user component, was already loaded.
* And this is how you can implement child routing, nested routes, with this children property which then holds all the nested routes.
* router outlet here.
* This now adds a new hook which will be used on all child routes off the route being loaded on the service component, which of course is our slash service route here.
* So all these child routes will be loaded in this router out of place now.
* So if we save this, and now we let this reload.
* You don't see anything here.
* But if I click a server, it's now loaded next to the menu because this is where we added our second router outlet.
* And this is how you can easily add child routes.
* Let's do the same for the user routes now.
* In the app module I'll add children here.
* So this property which takes an array of routes.
* And I only have one nested route here, but that's fine.
* So we'll add it here.
* Get rid of the users at the beginning.
* Just have slash, well, oh, and the ID and name, the two dynamic parameters, and load the user component.
* And now in my users component HTML file, I'll replace my app user here with another router outlet where all the user-related child routes, or the one user-related child route we have, will be loaded.
* So if we save this, you now see we load the user next to it, and that is why it is super important to dynamically update the ID and so on.
* Because the component wasn't exchanged, we were able to switch the loaded user while this old component, the user component, the single user component, was already loaded.
* And this is how you can implement child routing, nested routes, with this children property which then holds all the nested routes.