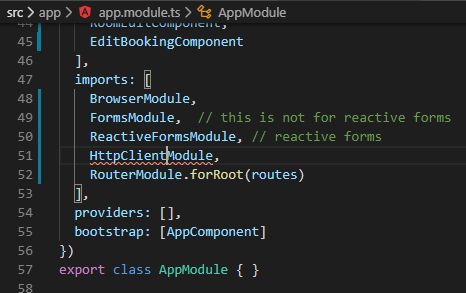
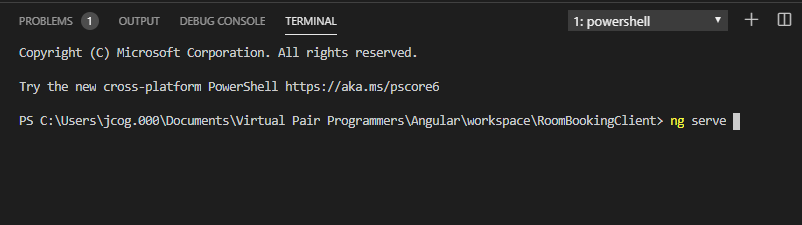
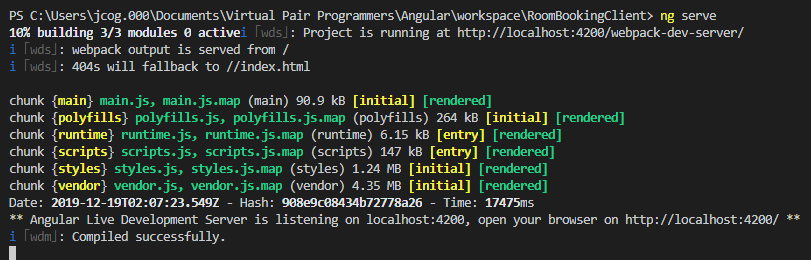
**CORS Configuration:**

Import the HTTP app module:



Make sure you running in the Development Environment:

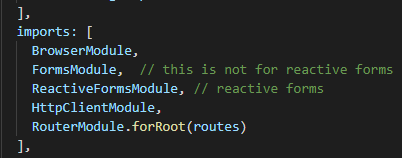




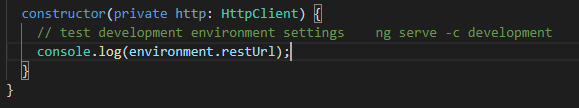
Missing angular/common



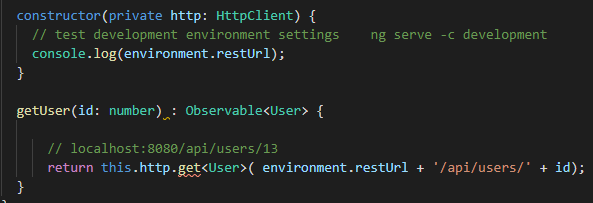




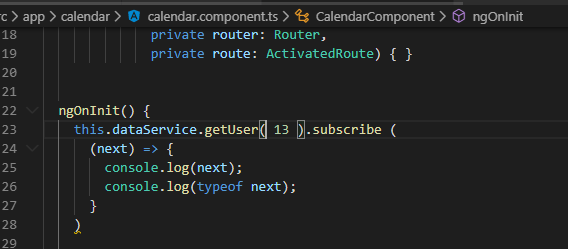
Go to the dataService.ts file and dependency inject the http service



Testing this: Create a getUser()

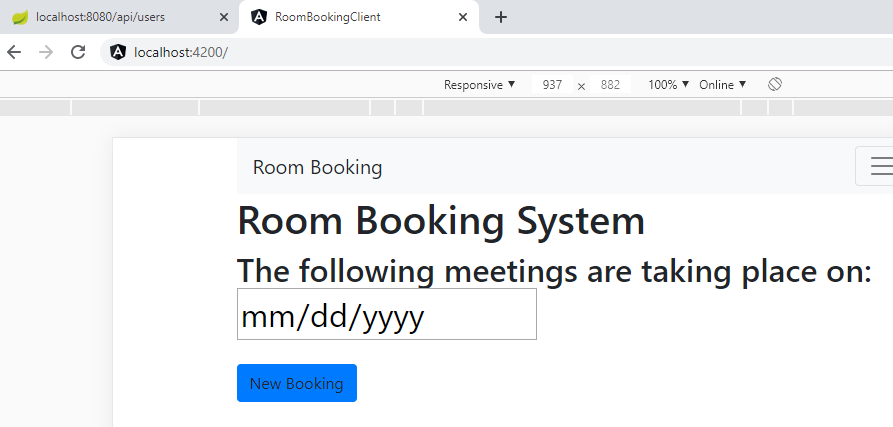


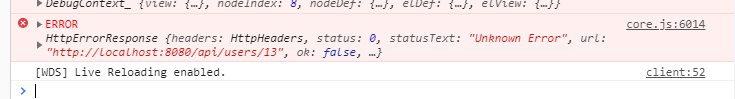
In the CalendarComponent we will subscribe in order to fire the method in the dataService



Look into the console with both servers running:

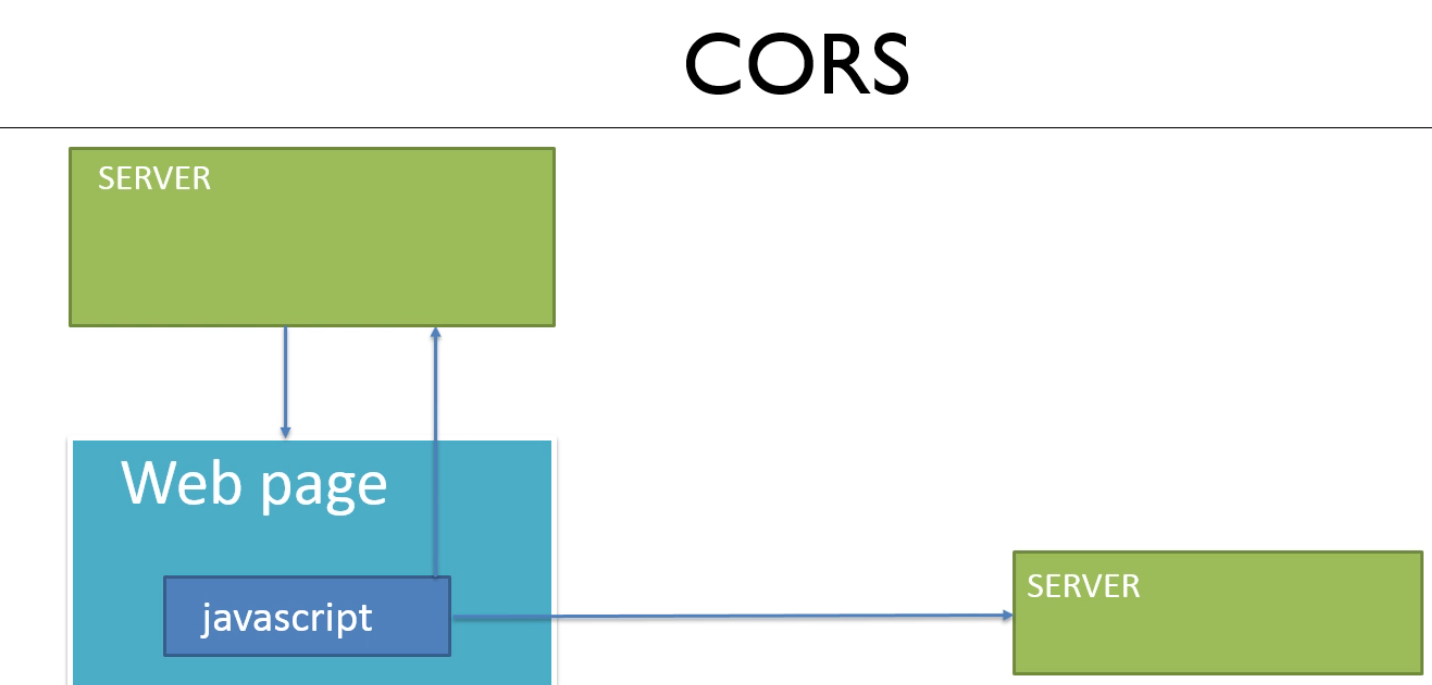
See that the url: <http://localhost:8080/api/users/13> is being sent





Cross-Site scripting error is blocked

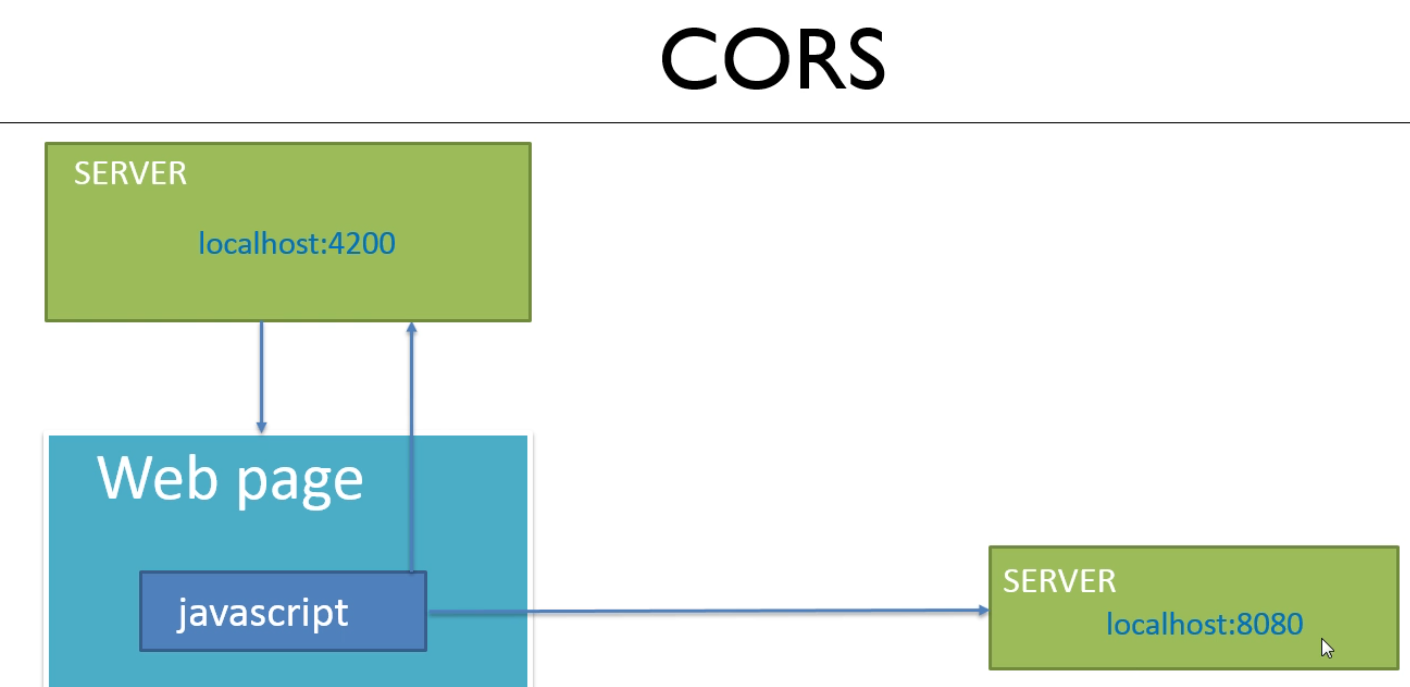


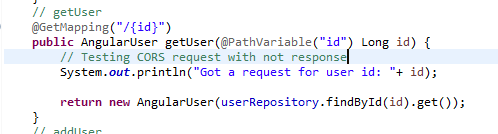


The second server is the problem the CORS is describing

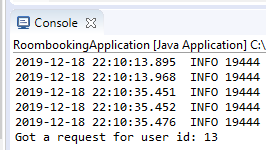
This needs to be enabled to occur: ports are different Cross Origin Servers

The response from the 8080 server is blocked



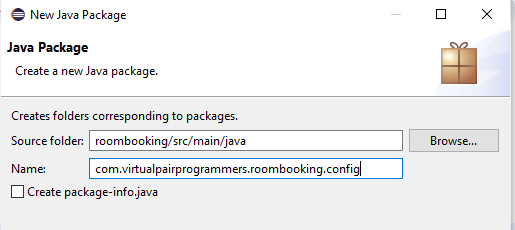


Restart both servers



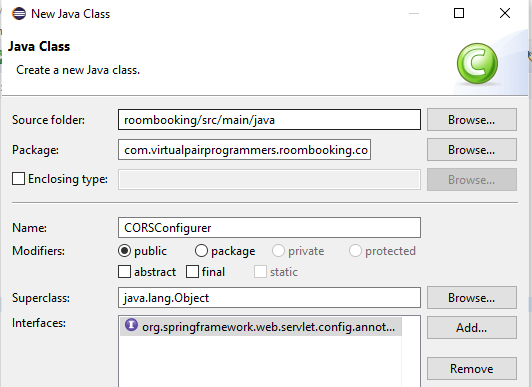
Next, set up Spring configuration to allow CORS to work!

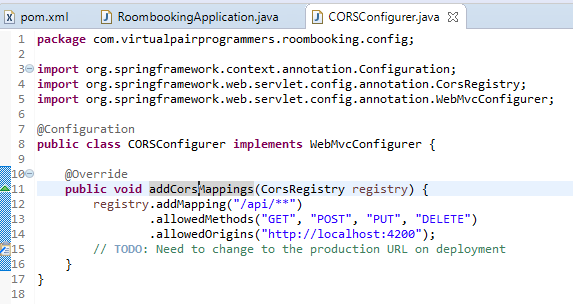
Create a new package:

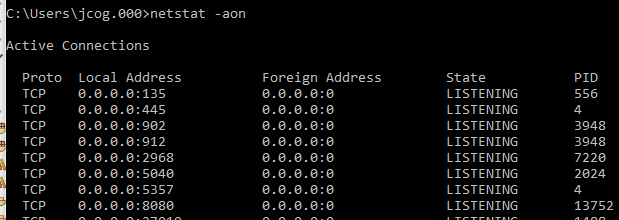


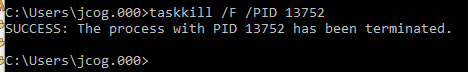
Create a new class:

With the interface WebMvcConfigurer from Spring Annotations:

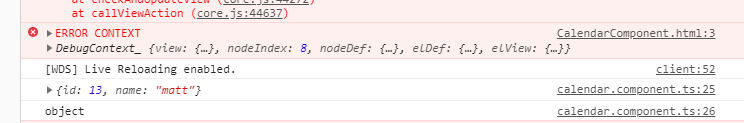








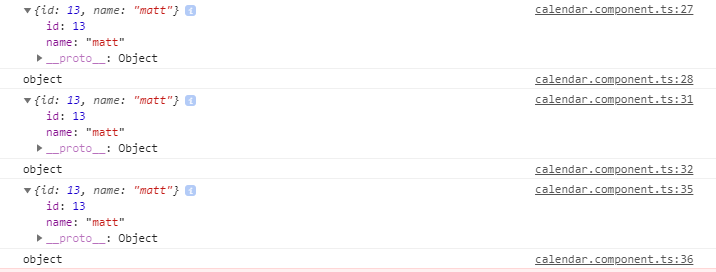
Restarted application and the red light stayed on!!



Got the return with the CORS blocking error!!!

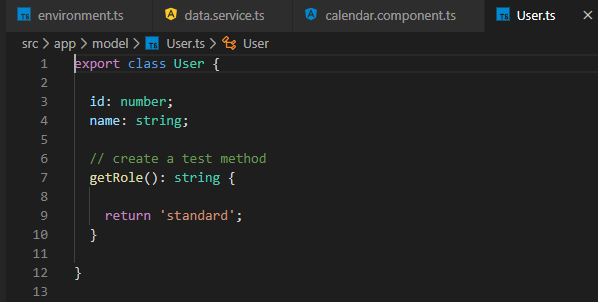
**Next, Manipulating the REST return data Type:**





They output the same results: This is not an instance of User

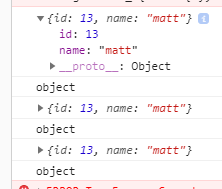
Demo that the object does not know about methods, like an instance would



Can we getRole()



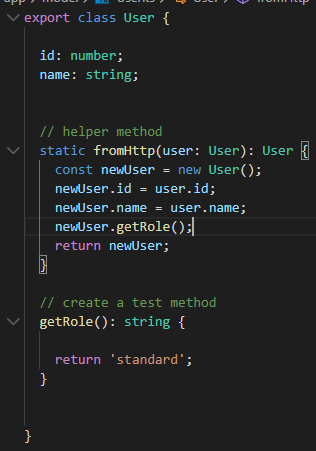
Notice in the console there is no getRole() return “standard”

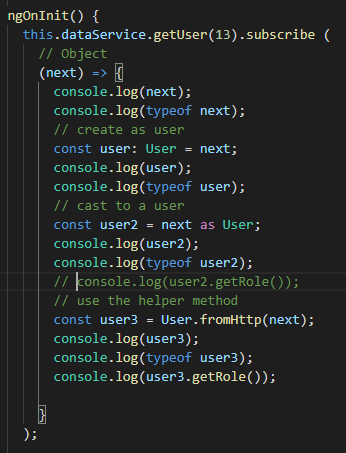




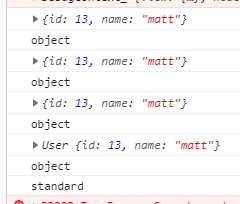
Create a helper method to solve the problem

Implement the helper method:





Test this:



Now, we get a User type return that has access to methods!!

**Next, preprocessing the REST return data:**