CARSWIP

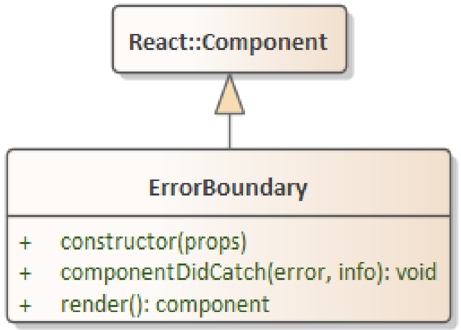
Carswip Technical Interview

Levente Nagy

nagyleventezsolt@gmail.com

Task 1

With the introduction of the “ErrorBoundary” React Component, utilizing the “componentDidCatch” operation, this component can catch all errors thrown by its children, solving the problem.



Task 2

This happens because of the Same Origin Policy, and anchor tags enforce this policy for a reason, as it protects the clients from malicious activity. The absence of this policy would make arbitrary code execution from foreign sources possible in the scope of the original webpage. This could be abused to access cookies, session storages etc., stealing the clients’ data.

The proper solution is to move the resource to the same host.

But if for some reason, bypassing the SOP must be achieved, it can be solved by the utilization of JavaScript. Saving the resource as a blob object in a script is possible with a simple XMLHttpRequest. From there the said resource can be saved to the client as the browser cache does not count as foreign origin. Besides the fact that this is an inefficient way of saving a resource (since the cache is located in and limited by the RAM), this would be an irresponsible thing to do, as it would open up some vulnerabilities.

For demonstration purposes, said script is included in /source/Task_2/SOP.html. It is conveniently coded, that giving the “bypassSOP” class to any anchor in the document, bypasses the SOP on the mentioned anchor.

Task 3

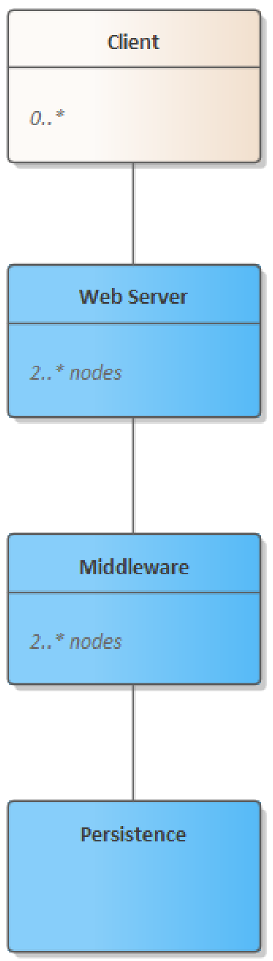
Depending on the architecture of the system, this can be solved multiple ways.
The key traits here are:

- Is it a distributed system?
- Are there any extra available resources?

First, let me tackle the case of a distributed system. Deploying the new version without any downtime is possible in 3 steps.

Step 0

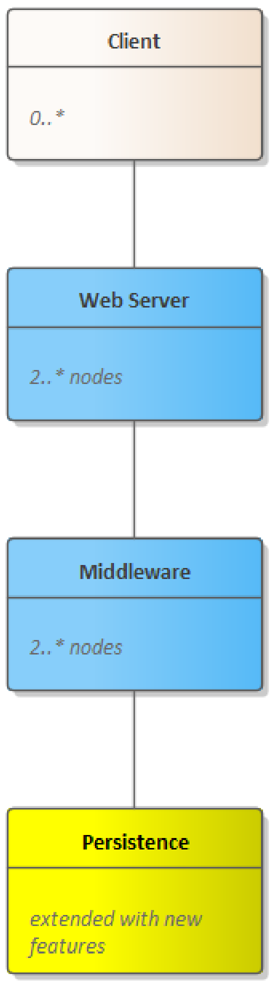
This is the skeleton of a 3-tierd distributed system, without the visualization of load balancers and other miscellaneous components like possible standby nodes of the persistence, etc.



Step 1

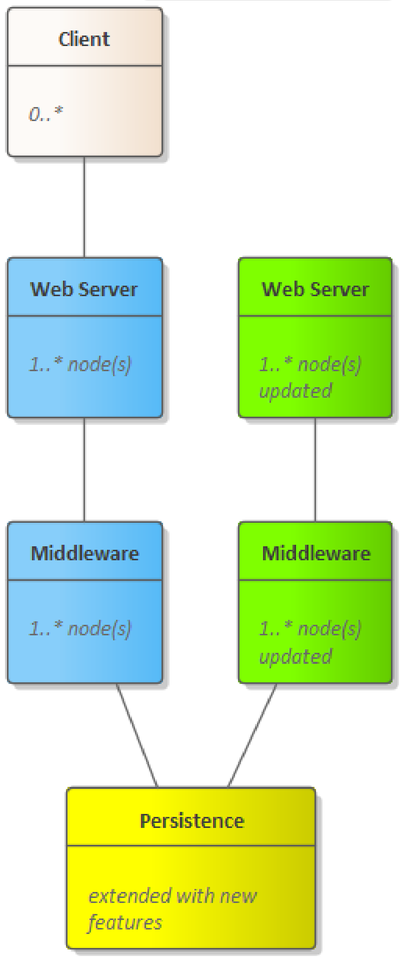
Extend the persistence with new features and migrate the data.

(Assuming this can be done without the ddl statements effecting the old version. The case of this not being possible will be detailed later on in this document.)



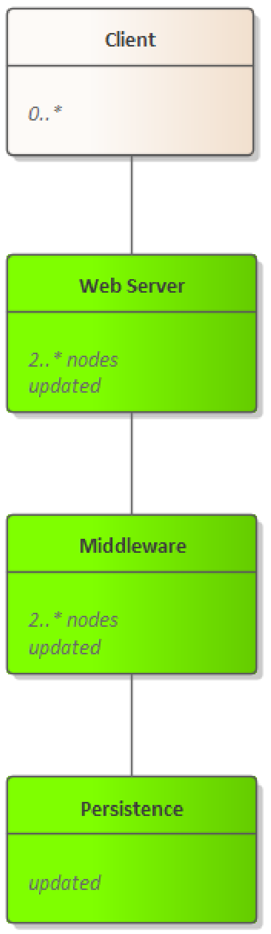
Step 2

Update the nodes separately. When enough nodes have been updated, the clients can be routed to use the new version instead, virtually having no downtime of the system.



Step 3

Update the rest of the nodes and persistence.



In the case of it not being a distributed system, and if there are available extra resource, the same practice can be done by introducing a second set of nodes temporarily.

And if extra resources are not available, not having downtime is virtually impossible.

Returning to the question of updating the persistence, if a major update completely breaks the persistence, a temporary persistence node must be introduced to avoid downtime. And the same goes here as well, if such resource is not available, not having downtime is virtually impossible.

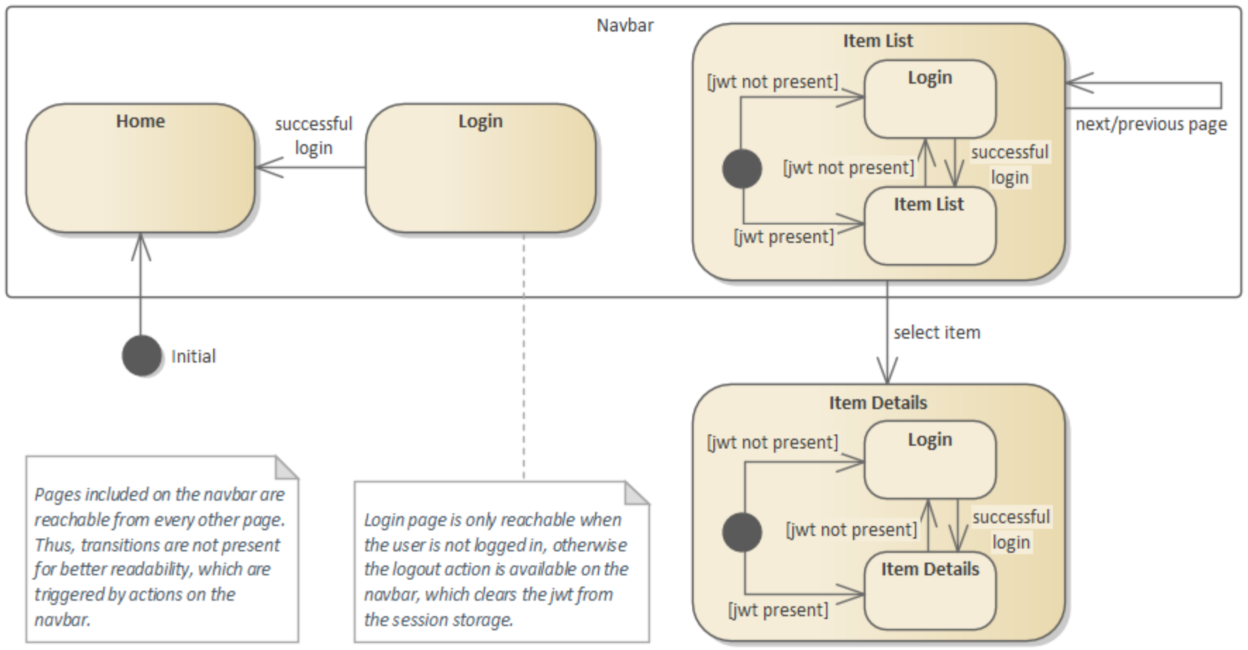
This deployment pattern is better known as Blue-Green Deployment.

Task 4

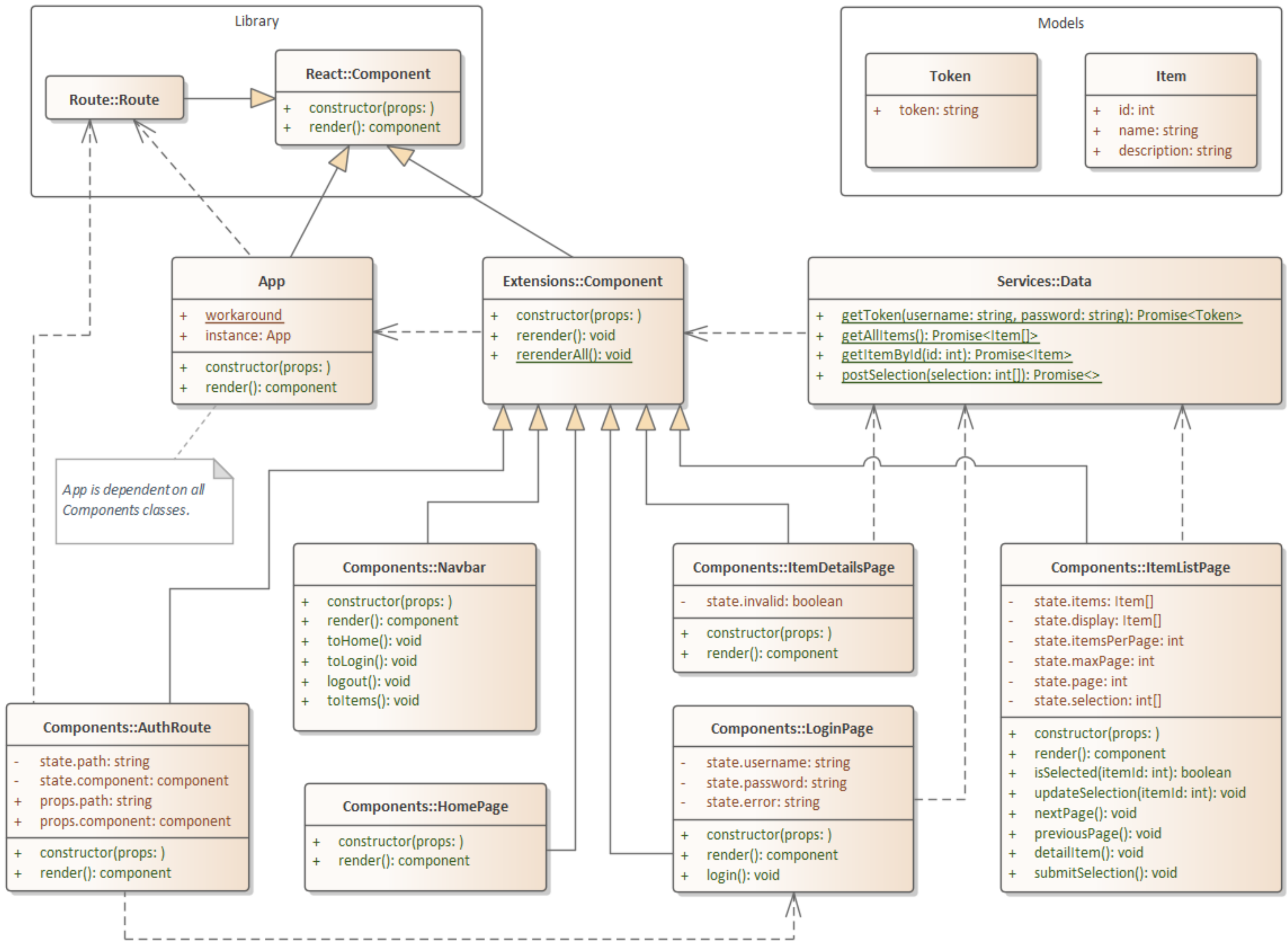
For authorization and authentication, it has been decided to use JSON Web Tokens.

The RESTful endpoints of the backend are detailed here:
Carswip | 1.0.0 | C4T4PHR4CT | SwaggerHub
(an offline version is also included under /docs/Task_4-Swagger(offline).html)
It must be said that the documentation of such API is only released because of this not being a live project. Unless such documentation would have been done on a closed swagger environment, avoiding leaking the authorization and authentication process.

The following state diagram represents the user's possible navigation and states:


Navbar
Home
successful login
Login
Item List
[jwt not present]
Login
[jwt not present]
successful login
Item List
[jwt present]
next/previous page
Initial
select item
Item Details
[jwt not present]
Login
[jwt not present]
successful login
Item Details
[jwt present]
Pages included on the navbar are reachable from every other page. Thus, transitions are not present for better readability, which are triggered by actions on the navbar.
Login page is only reachable when the user is not logged in, otherwise the logout action is available on the navbar, which clears the jwt from the session storage.

Class diagram:



(raw figures of this document can be found under /docs/Diagrams.eapx)