Conventions, QAT & UAT, nontent.

1. Coding convention and standards

a. Github

Branches :

New branch creation for every new feature development. Branch's name must be in English, and describe clearly the developed feature.

Commits :

Commits must be in English.
We'll use the following convention for our commits : <[type]: [sujet]>
Example : [ADD]: User authentication.
Developers should commit frequently, which will allow us to follow code progression easily.

Pull-requests :

When a functionality is finished, a pull request needs to be created, allowing us to merge the new functionality on our main branch.
The pull request needs a clear description, summarizing the modifications developed on the feature.
A pull request must be processed and validated by two people before it can be merged into the main branch.

b. Naming conventions

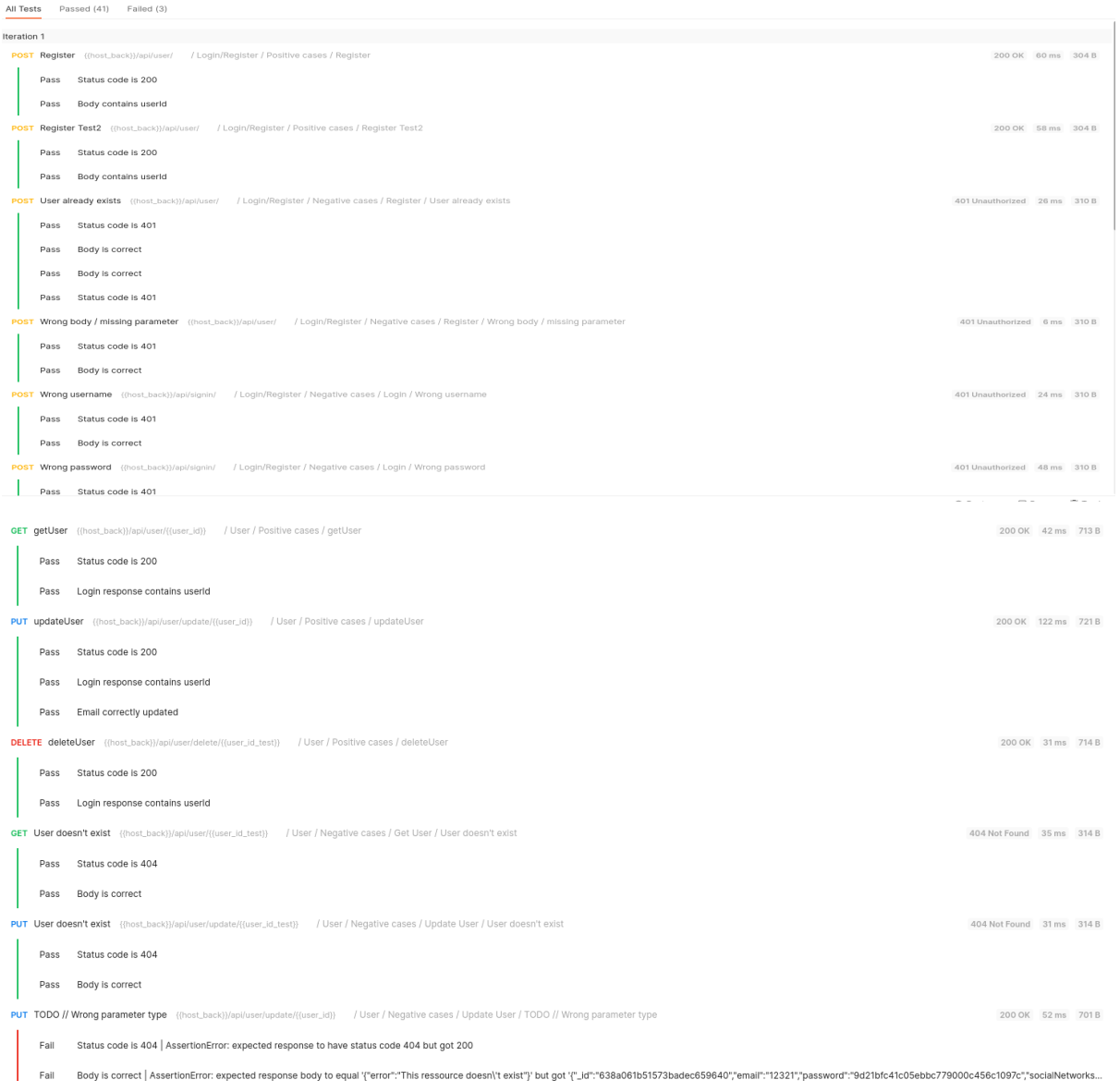
For the naming of our files, variables, class and instance, we use CamelCase (more precisely camelCase, so with the first character in lower case). Our collections are named entirely in lower case, and the fields of our documents in camelCase.

2. Testing

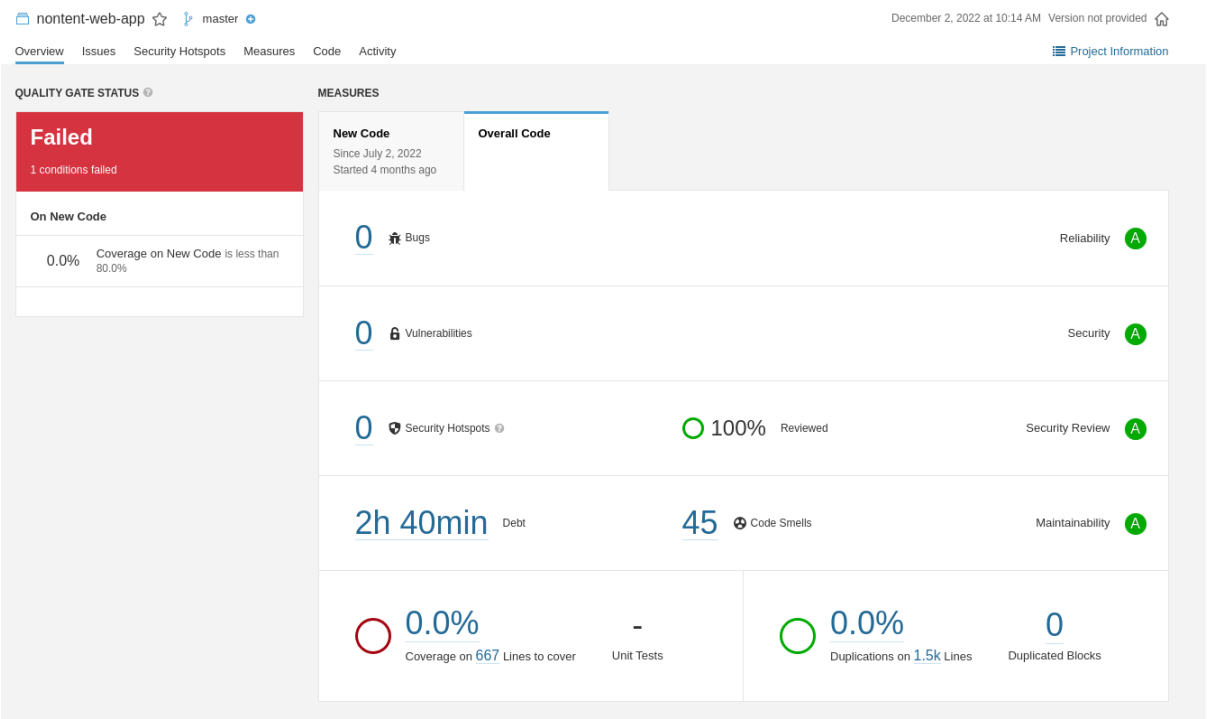
a. Quality Assurance Testing

We use QAT tools to check the quality of our code and application.
With quality assurance testing, we verify that our application respect our specifications and requirements.
All the QAT process is managed within nontent. development team, to ensure the prevention of problems before delivering our product to the UAT team.
Here is the tools we used in our QAT :

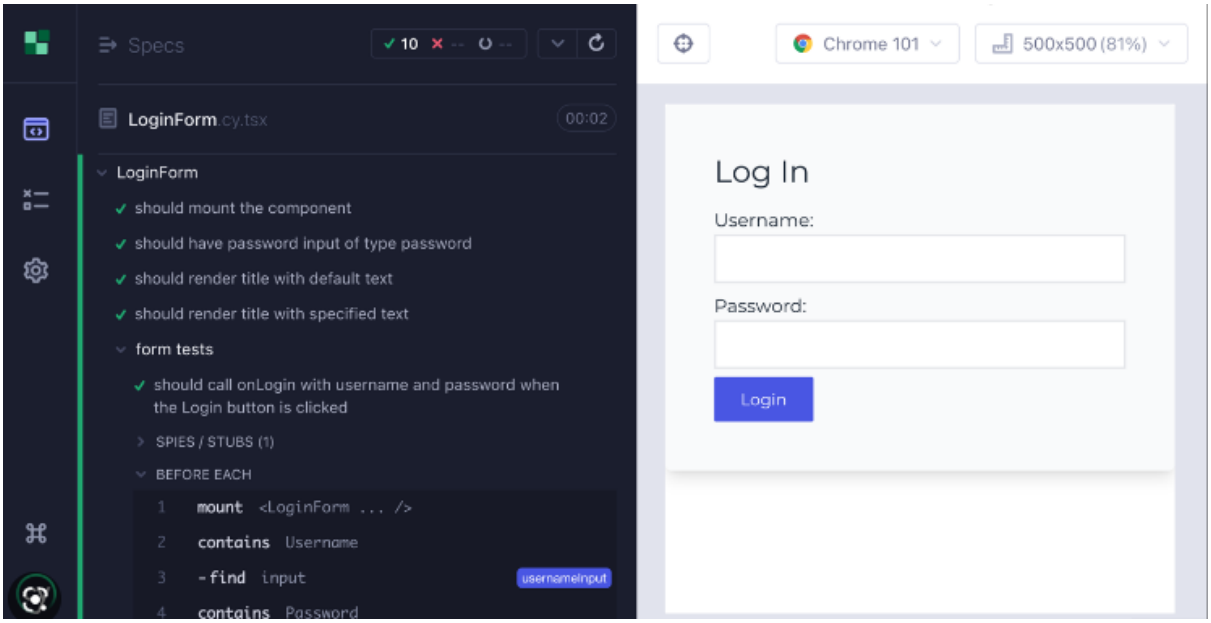
Postman: To test our multiple api endpoints, we used postman testing. We have multiples test on our own api routes, and on api routes we are calling from the multiples socialnetworks we implemented.



Sonarqube: We implemented Sonarqube too, to keep a track on our code safety and accessibility. Sonarqube warn us on potential security issues.



Cypress: Cypress is our front-end testing framework, used on javascript frameworks, to operate directly in the browser



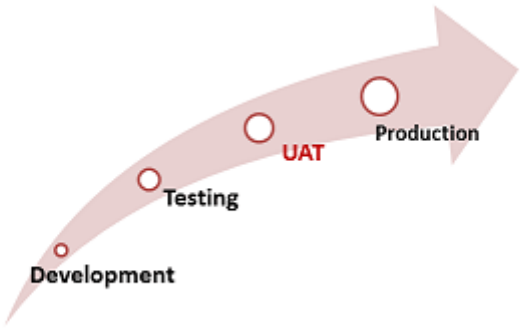
b. User Acceptance Testing

This step comes after the QAT, and is a client sided test phase.
Our product may match all our specs and requirements, but here we need real users testing, to actually know if the product is well-designed, easy to use, and to detect some bugs and problems we didn't expect / think about.

We'll handle our UAT in multiple steps. First of all, when we actually think our product is finished, we are going to drive an alpha test internally.
We are going to test the whole application and functionalities, while comparing them to our specifications.
With this, we can actually find some technical problems and correct them instantly.

While we are doing this, we are going to identify and write test scenarios and test cases.
Then, we'll use some data we already collected and used for development, modify some fields if needed to keep privacy of our current users, and prepare this data for the test cases

Next, we are going to ask some people we know to join our 2nd phase, the beta test.
This beta test will give us some insights about the application, and other perspective tests. We may find other problems or cases we actually didn't expect. We may find unexpected issues concerning our functionalities, our design, etc.
When all our stakeholders and our own team members agree that the product is ready to be delivered, and ready to go on production, we'll then leave this phase and present our product to everyone.


Production
UAT
Testing
Development