Faculdade de Engenharia da Universidade Do Porto

Mestrado Integrado em Engenharia Informática e Computação

Engenharia de Software

T3 – Software Development in Practice - Jest

*Alunos:*

João Gama Amaral

João Nuno Ferreira

Pedro Galvão

 de Novembro, 2018

**Issue Documentation**

**Issue #6194**: Running tests in a specific order.

Allow define which files should run before others.

Sometimes, it is needed to define a specific test to be ran before others. As people want to maintain different **test cases** in different files, it would be essential to define the order for some tests.

**Requirements**

To solve this issue, you will need at least some basic JavaScript knowledge.

The case described on the issue documentation implies on environment tests when using Jest with other frameworks, like Selenium, for example. Selenium is a suite of tools to automate web browsers across many platforms.

**Example**

Need to test an email client (browser-oriented). As Jest is a robust framework, it will be used with Selenium for **environment**, **unit** and **integration** coverage. The following tests are focused only in environment coverage with Selenium. The tests are:

* Add an account and make sure that it was added successfully;
* Check messages in the account;
* Perform other actions (Open a message, download file, perform filtering, etc.);
* Remove the account.

The idea is to define in the **Jest settings** an array containing the file names and the order of files that should be ran firstly. For example:



So, the order of running tests would be **A.test.js**, **B.test.js**, **C.test.js**, **D.test.js**.

Once the tests are done, then jest could run other tests sequentially or not.

Jest already has a feature that run tests sequentially which is: **--runInBand**. The problem is that it doesn’t exist a way to define which files should run firstly.

**Possible solutions that were presented but not good enough:**

**Source Code Files**

**System architecture**

**Design of the fix**

**Issue Documentation**

**Issue #5730**: Support dynamically detecting changes when *.gitignore* is updated

A certain repository has a *.gitignore* file in it. This type of file specifies intentionally untracked files that Git should ignore. Each line in a *.gitignore* identifies a pattern that must be disregarded.

If a *\*.js* is selected, then all changes in JavaScript (*.js*) files will not be taken into account. Besides this, Jest tests are coded in TypeScript files (*.ts*). The problem consists in the fact that when editing *.ts*, although Jest notices the file changes, it does not run the tests. But, when removing *\*.js* from the *.gitignore* the Jest runs those same tests.

**Requirements**

To solve this issue, you will need some basic knowledge in how the *.gitignore* files are supposed to work.

The case described on the issue documentation, implies that the users trying to fix the problem install and use either the suggested *Yarn* and *NPM* or any other dependency managers (for example, *IED*, *PNPM* or *NPMD*) that Jest can work with. These programs that are responsible for controlling the packages will be used to the user be able test.

**Example**

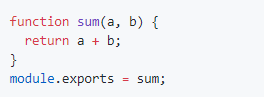
As said before, Jest is a robust framework, so it will be used along with *Yarn*/*NPM* for **package** and **dependency** management.

To test you must:

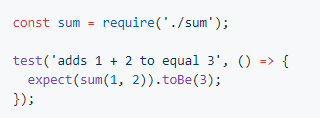
1. Install the *NPM*:

C:\Users\User\Desktop\download.png

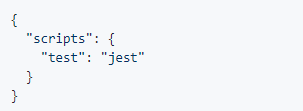
1. Create a *sum.js* file, with the following code:



1. Create a *sum.test.js* file, with the following code:



1. Modify the *package.json* file, changing the code inside the *“scripts”* curly brackets to *“"test": "jest"”*:



1. Create a file named *.gitignore*, that will be automatically hidden, with:



After the whole preparation, the user must keep running the *“tsc –watch”* and run *“jest –watch”*. Edit the *sum.test.js* and observe if the Jest notices the file change and if it runs any tests. Finally, the user should try removing the *“\*.js”*, and repeat the previous process.

**Possible solutions that were presented but not good enough:**

**Source Code Files**

**System architecture**

**Design of the fix**