JobExecRequestWebUI manual

V. 1.0

Breogan COSTA

https://www.linkedin.com/in/breocosta/
https://github.com/covelus

Contents

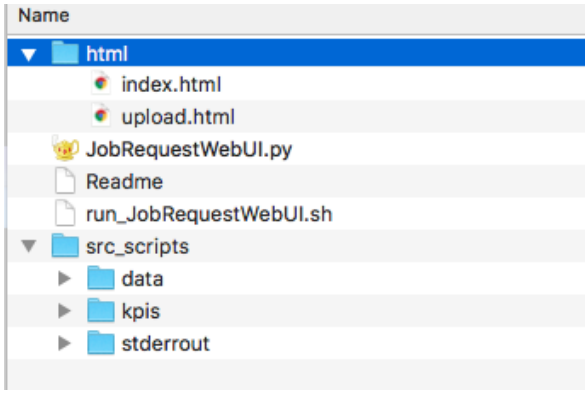
1. Idea, what this small application does:

Imagine you have some scripts deployed in a datacenter machine, inside a trusted environment. You already have the mechanism to upload them (i.e: a software repository), but you want to give users the chance to manually run the scripts and recover the execution status output (STDOUT, STDERR).

Also, this application would take some statistics and KPI’s (Key Performance Indicators).

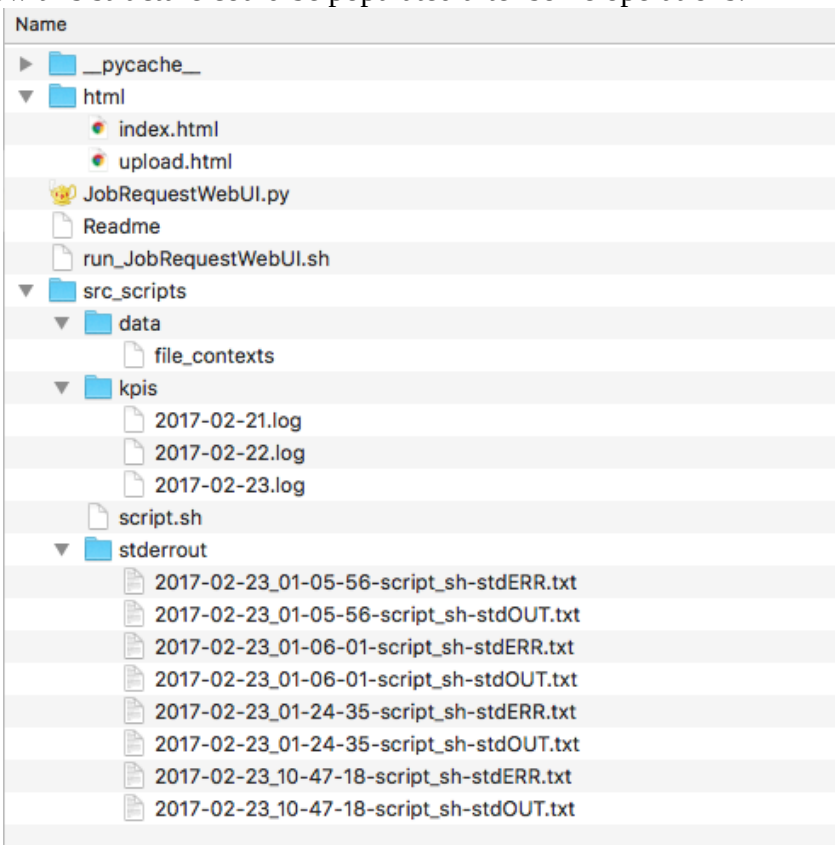
2. Folder structure:

Inside the application's directory, they could be found some s src_scripts subdirectory and files:



- html/
  - *auxiliary HTML pages that will be used by the WebApp's*
- JobExecRequestWebUI.py
  - *WebApp Python 3 script*
- Readme
  - *Something you must read*
- run_JobExecRequestWebUI.sh
  - *shell script to launch the Web server and WebApp*
- src_scripts/
  - *data used by the script and captured stdout/stderr and KPIs*
  - *Scripts are also deployed here. It is not responsibility of this application.*
  - src_scripts/data/
    - *uploaded data files by the user, that will be used by the deployed scripts (developer responsibility)*
  - src_scripts/kpis/
    - *data gathered to generate the KPIs*
  - src_scripts/stderrout/
    - *captured stdout and stderr*

An example of how this structure could be populated after some operations:



3. Run:

This application requires a embedded server (and micro-framework), flask.
It was automated through an script located in the application's directory.
In the shell in that folder, we just run the script "*run_JobExecRequestWebUI .sh*":

```
$ ./run_JobExecRequestWebUI.sh
```

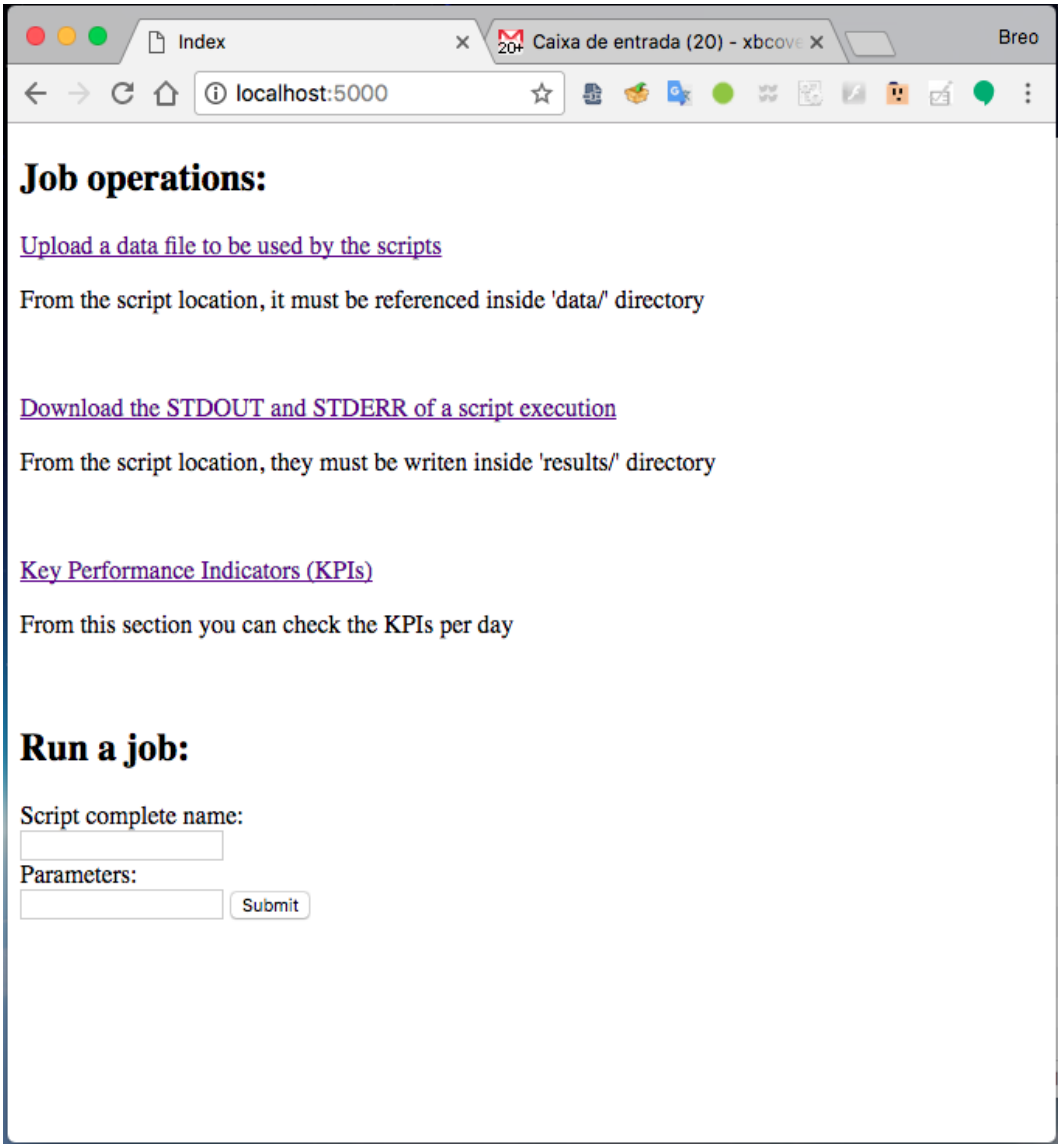
3.1 Opening the WebApp:

It can be found in http://[server-url]:5000/, for example,
http://localhost:5000/

4. Graphical explanation of the WebApp:

This WebApp is very intuitive, but some screenshots could be helpful.

4.1 Initial page



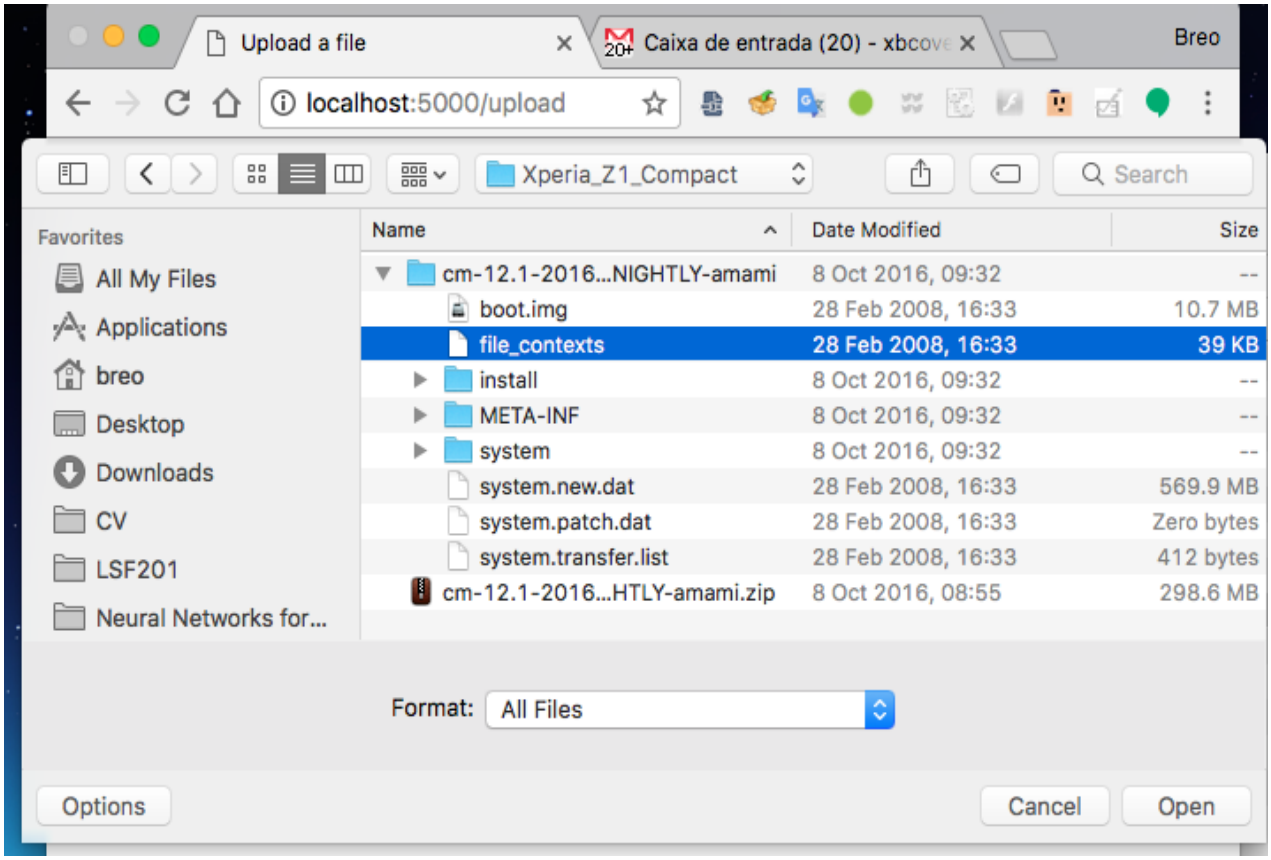
4.2 Running a script

You can do this in the bottom of the page, typing the name of the script and the parameters. For example:

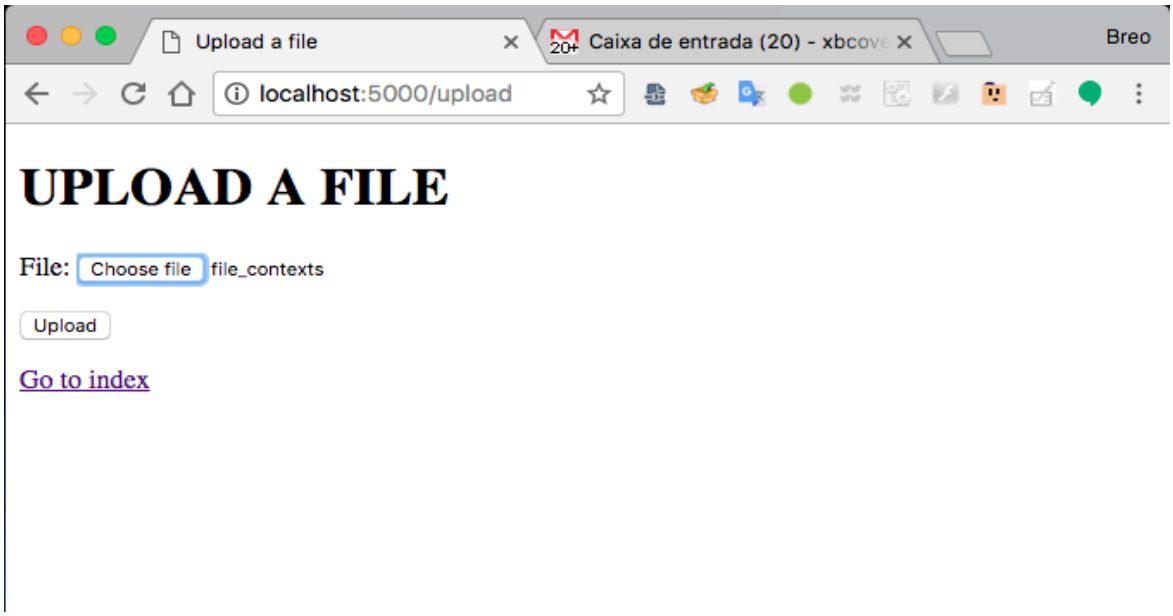
4.3 Uploading a file

This option is presented in the top of the page.

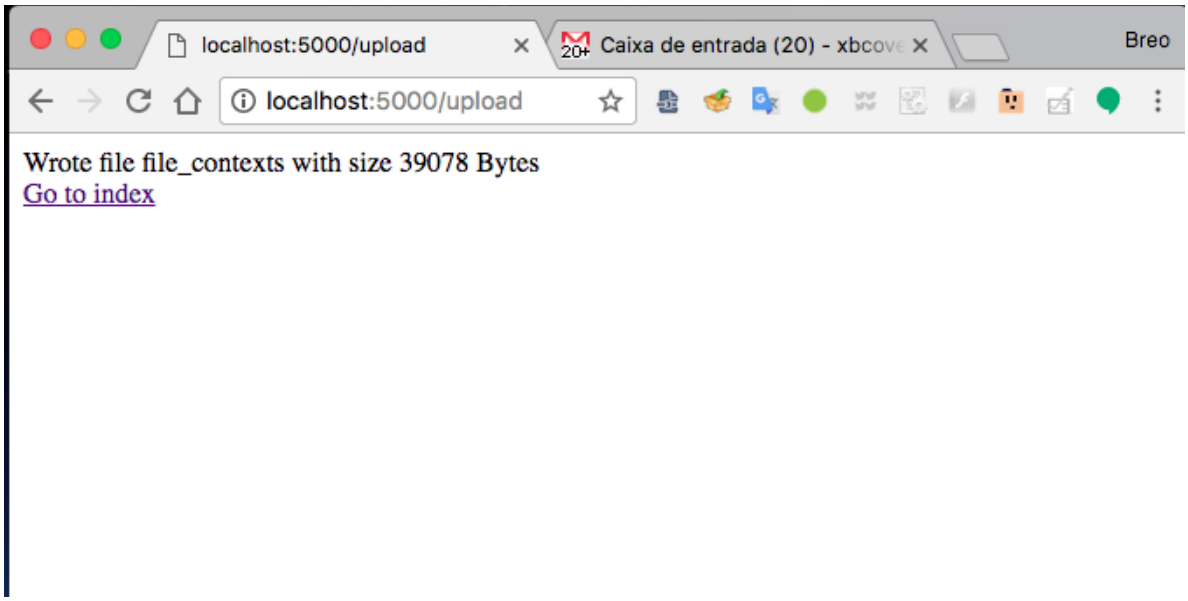
It is simple, but an example of the whole process is shown below:



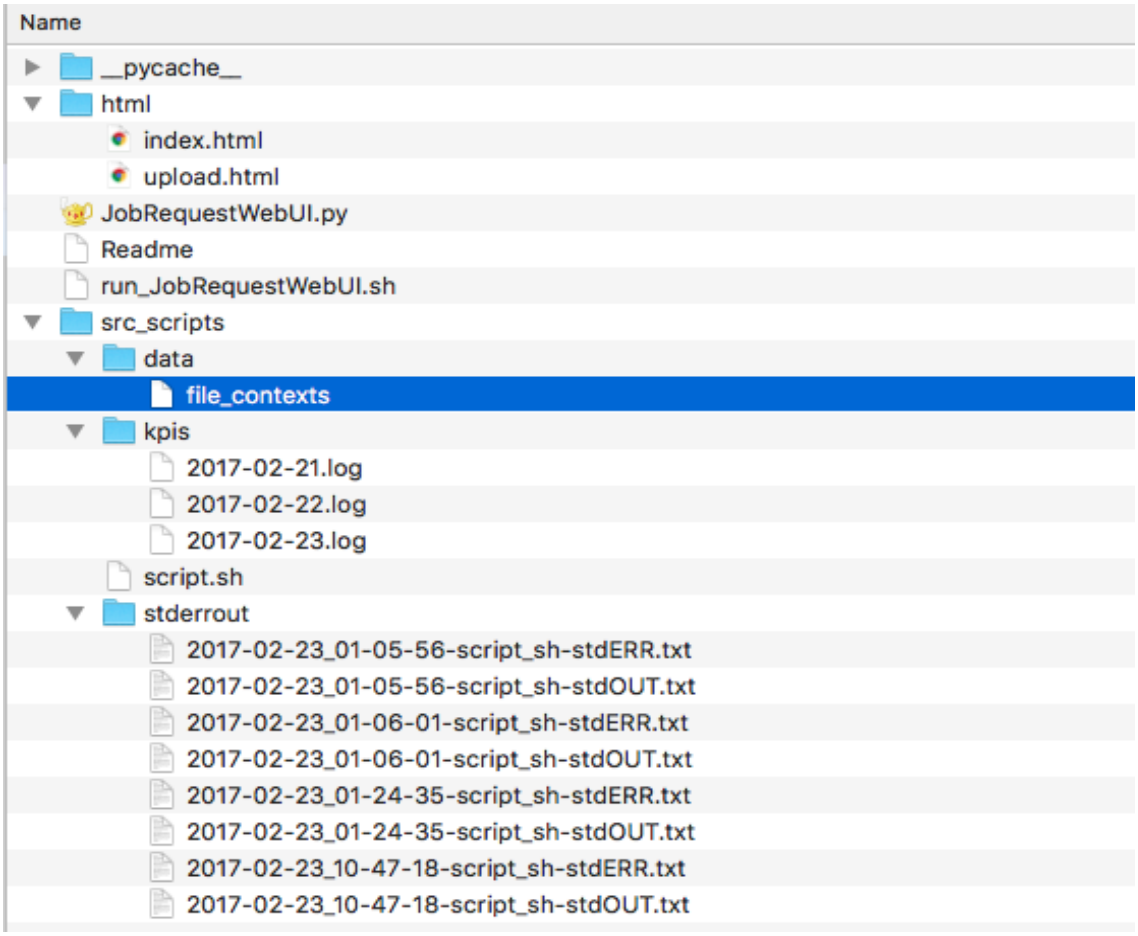
After, you must press upload:



And the result of the upload will be presented:



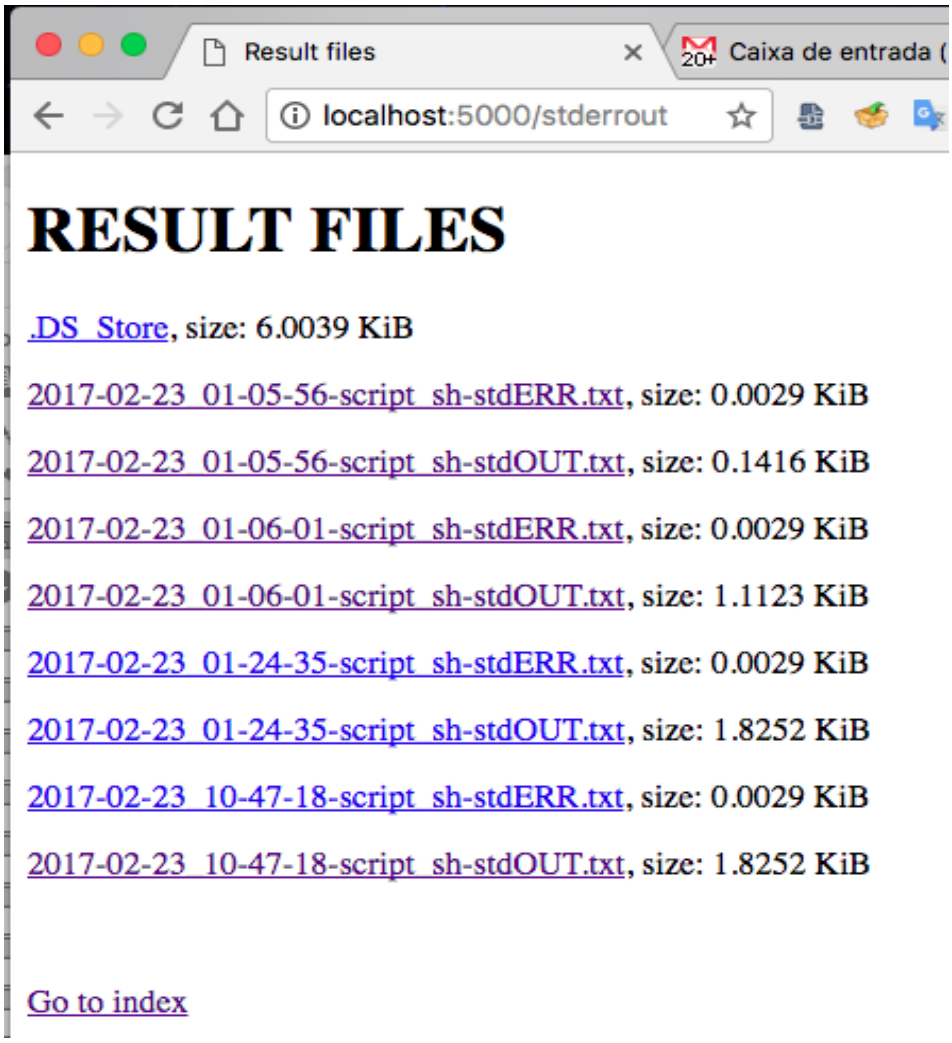
Also, having read rights in the server, it can be seen the file in the *src_scripts/data/* directory:



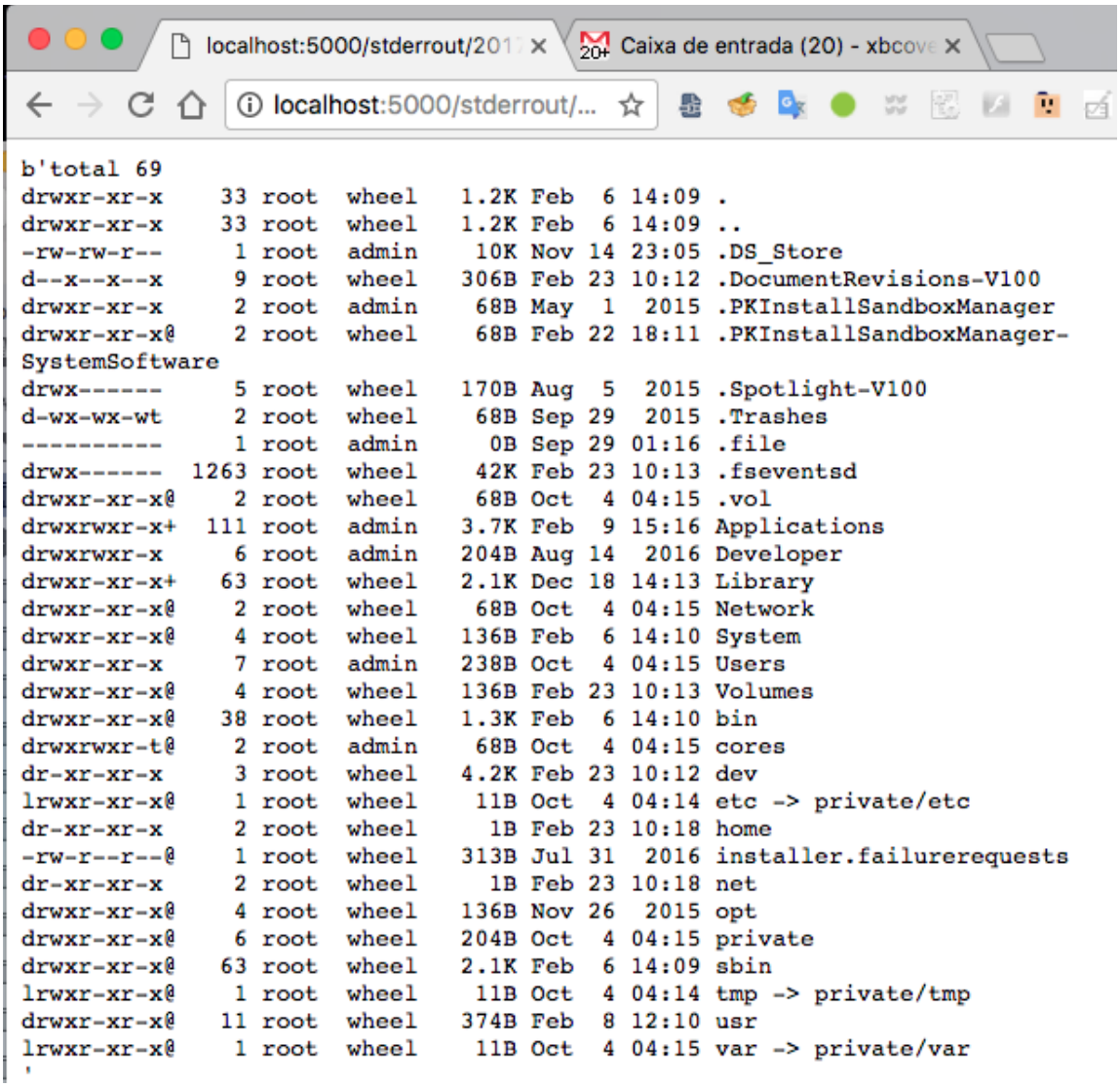
4.4 Checking STDERR and STDOUT

This is presented in the second option of the WebApp index page.

Once there, all captured stdout and stderr files are listed and can be opened.



Opening the last file (result of the script execution example shown in a previous section) would show:




```
b'total 69
drwxr-xr-x    33 root  wheel   1.2K Feb  6 14:09 .
drwxr-xr-x    33 root  wheel   1.2K Feb  6 14:09 ..
-rw-rw-r--     1 root  admin    10K Nov 14 23:05 .DS_Store
d--x--x--x     9 root  wheel   306B Feb 23 10:12 .DocumentRevisions-V100
drwxr-xr-x     2 root  admin    68B May  1  2015 .PKInstallSandboxManager
drwxr-xr-x@    2 root  wheel    68B Feb 22 18:11 .PKInstallSandboxManager-
SystemSoftware
drwx------     5 root  wheel   170B Aug  5  2015 .Spotlight-V100
d-wx-wx-wt     2 root  wheel    68B Sep 29  2015 .Trashes
----------     1 root  admin     0B Sep 29 01:16 .file
drwx------  1263 root  wheel    42K Feb 23 10:13 .fseventsd
drwxr-xr-x@    2 root  wheel    68B Oct  4 04:15 .vol
drwxrwxr-x+  111 root  admin   3.7K Feb  9 15:16 Applications
drwxrwxr-x     6 root  admin   204B Aug 14  2016 Developer
drwxr-xr-x+   63 root  wheel   2.1K Dec 18 14:13 Library
drwxr-xr-x@    2 root  wheel    68B Oct  4 04:15 Network
drwxr-xr-x@    4 root  wheel   136B Feb  6 14:10 System
drwxr-xr-x     7 root  admin   238B Oct  4 04:15 Users
drwxr-xr-x@    4 root  wheel   136B Feb 23 10:13 Volumes
drwxr-xr-x@   38 root  wheel   1.3K Feb  6 14:10 bin
drwxrwxr-t@    2 root  admin    68B Oct  4 04:15 cores
dr-xr-xr-x     3 root  wheel   4.2K Feb 23 10:12 dev
lrwxr-xr-x@    1 root  wheel    11B Oct  4 04:14 etc -> private/etc
dr-xr-xr-x     2 root  wheel     1B Feb 23 10:18 home
-rw-r--r--@    1 root  wheel   313B Jul 31  2016 installer.failurerequests
dr-xr-xr-x     2 root  wheel     1B Feb 23 10:18 net
drwxr-xr-x@    4 root  wheel   136B Nov 26  2015 opt
drwxr-xr-x@    6 root  wheel   204B Oct  4 04:15 private
drwxr-xr-x@   63 root  wheel   2.1K Feb  6 14:09 sbin
lrwxr-xr-x@    1 root  wheel    11B Oct  4 04:14 tmp -> private/tmp
drwxr-xr-x@   11 root  wheel   374B Feb  8 12:10 usr
lrwxr-xr-x@    1 root  wheel    11B Oct  4 04:15 var -> private/var
'
```

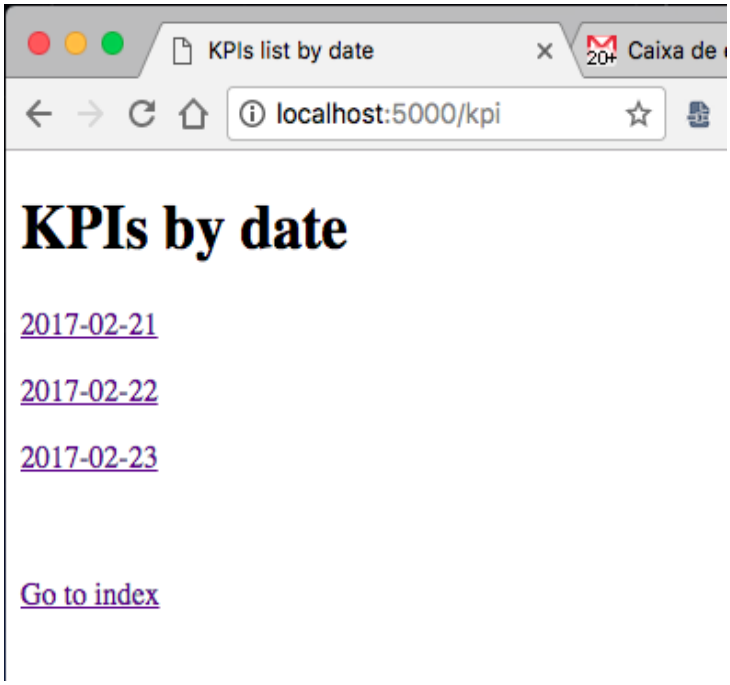
4.5 Checking the generated Key Performance Indicators (KPIs)

Some Key Performance Indicators were manually generated using Python ***pandas*** package and HTML code.

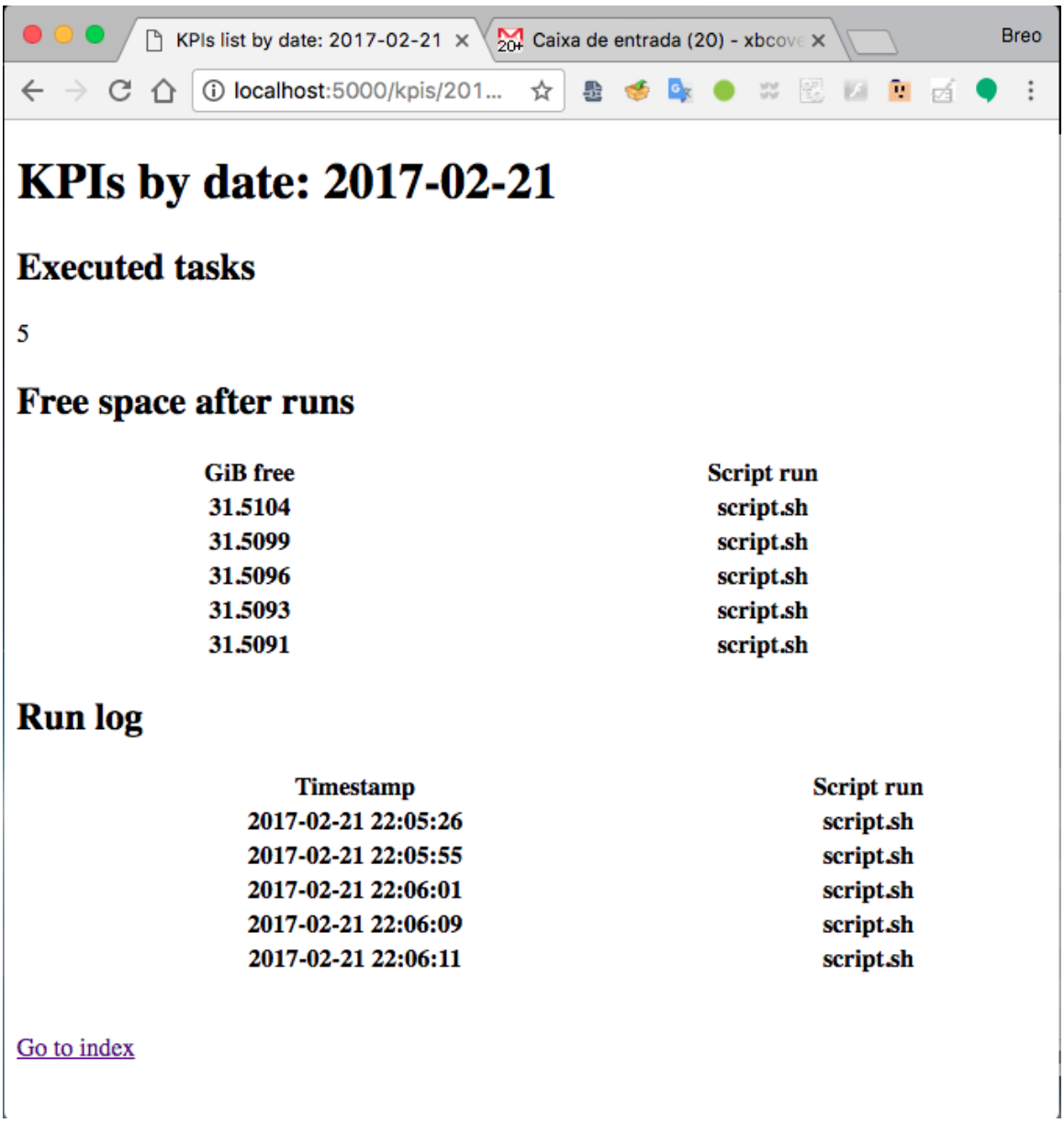
There are some KPI package projects for Python (***Ax_Metrics***), but they were not adapted to Python 3 yet, so, it is possible that they are not maintained.

Some results could be seen with ***matplotlib*** and ***mpld3***, but they were not included in this solution, due to they require a further research regarding to the Web integration with ***flask***.

In the WebApp, the option is presented in 3rd position, and it will bring a list of links to KPIs by date, that can be pressed.



Once opened, they will show the number of executed tasks in that day, the free space after each script run, and the runs log ordered by timestamp:



KPIs by date: 2017-02-21

Executed tasks

5

Free space after runs

GiB free	Script run
31.5104	script.sh
31.5099	script.sh
31.5096	script.sh
31.5093	script.sh
31.5091	script.sh

Run log

Timestamp	Script run
2017-02-21 22:05:26	script.sh
2017-02-21 22:05:55	script.sh
2017-02-21 22:06:01	script.sh
2017-02-21 22:06:09	script.sh
2017-02-21 22:06:11	script.sh

Go to index