Technical University of Moldova
Software Engineering and Automatics department
Study program in Software Engineering

WEB Programming
Laboratory Work Nr.2

Author:
Botnaru Victor FAF-202

Supervisior:
Alexei Șerșun

2023

Contents

1 Story

Imagine, you're a frontend developer from Melbourne, Australia. You and your company Sleep2nigth Pty Ltd are working on a brand new product - smart To-do lists!

You've pitched the idea in the local startup accelerator and you've won first round of investitions. The only thing what's left - to build the app itself.

2 Task

Your task for this lab is:

- Copy index.html, style.css and [app.js](app.js) to your repo;
- Modify them to build an application for to-do list;
- The app has to cover basic needs: - to add to the list; - to remove from the list; - to mark as done; - see "done" and "to-do" lists separately.
- The app has to look attractive.

3 Grading

Points:

- basic app - '6' points
- attractive app - '+1' point
- the list is preserved after page refresh - '+1' point;

  You can get '+1' point if the app implements a notification mechanism. You can get '+1' point for each useful feature the app implements, besides the ones listed above (up to '3' points, approved by the teacher).

  You can get '+2' bonus points for an adapted design for mobile devices.

4 Hints

– For UI, you are free to use any CSS library. You can try Tailwind CSS, Bootstrap or Bulma;

– Use the latest standards of ECMAScript (2015 or later) to write JS code;

– To debug JS, put ‘debugger‘ somewhere in the code you want to check what’s going wrong there, open Developer Tools and then refresh the page;

– Use JS classes to separate business logic from data representation.

5 Implementation

I’ve implemented this tasks using only html and css, and this is the start page

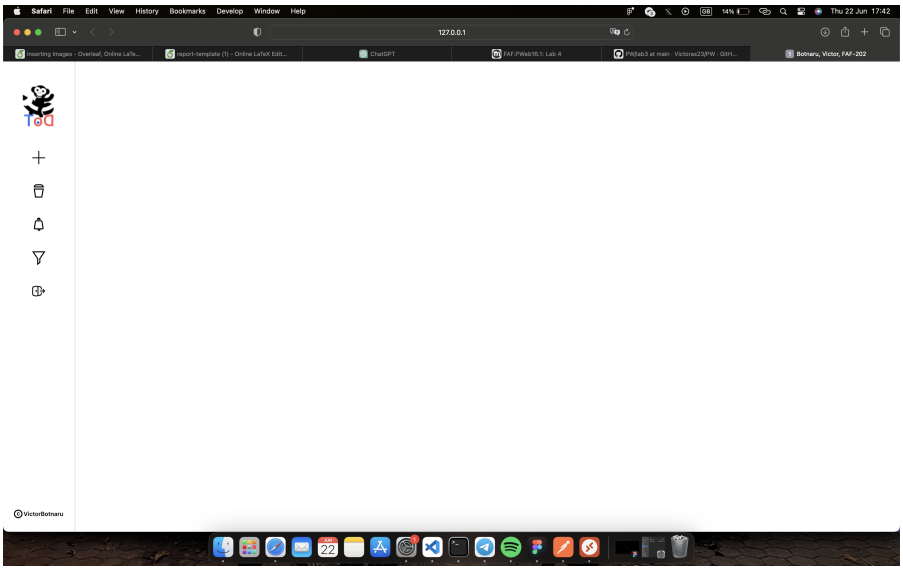


Figure 1: —

Now you can add the tasks in the task list, all of them will be saved in the local storage and on the loading and reloading of the page they will be untouched.

This is how the tasks are saved:

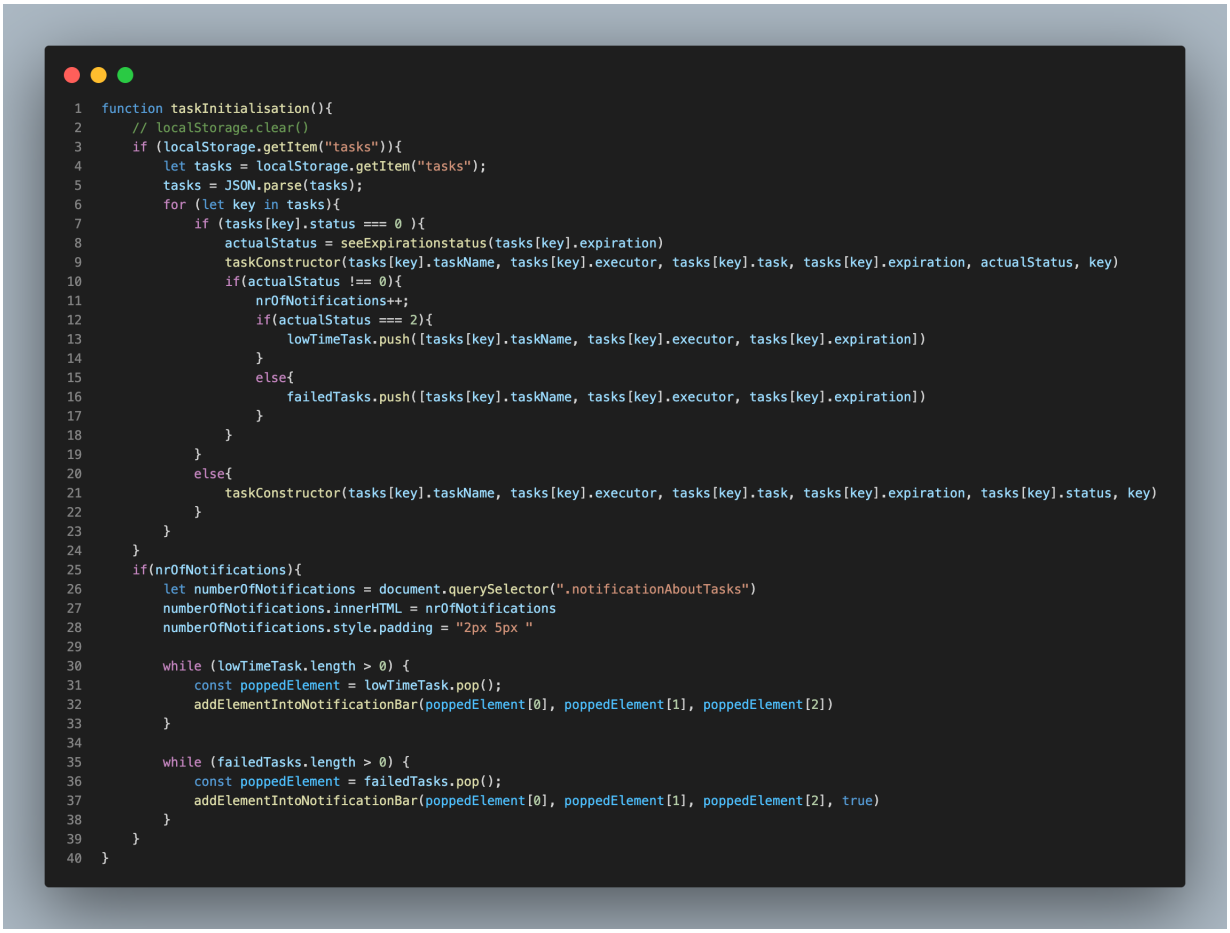
```
function addTaskToList(){
    hidePopUp()
    let newTask = {
        taskName: taskName.value,
        executor: executor.value,
        task: task.value,
        expiration: expiration.value,
        status: 0
    }

    if (localStorage.getItem("lastTasks")){
        let lastTask = JSON.parse(localStorage.getItem("lastTasks")) ;
        let tasks = JSON.parse(localStorage.getItem("tasks")) ;
        tasks[lastTask+1] = newTask;
        localStorage.setItem("tasks", JSON.stringify(tasks));
        localStorage.setItem("lastTasks", JSON.stringify(lastTask+1));
        taskConstructor(taskName.value, executor.value, task.value, expiration.value, 0, lastTask+1)
    }
    else{
        let lastTask = 0;
        let tasks = {};
        tasks[lastTask+1] = newTask;
        localStorage.setItem("tasks", JSON.stringify(tasks));
        localStorage.setItem("lastTasks", JSON.stringify(lastTask+1));
        taskConstructor(taskName.value, executor.value, task.value, expiration.value, 0, lastTask+1)
    }


}
```

Figure 2: —

And this is how the tasks are shown after that the page is reloaded



```
function taskInitialisation(){
    // localStorage.clear()
    if (localStorage.getItem("tasks")){
        let tasks = localStorage.getItem("tasks");
        tasks = JSON.parse(tasks);
        for (let key in tasks){
            if (tasks[key].status === 0 ){
                actualStatus = seeExpirationstatus(tasks[key].expiration)
                taskConstructor(tasks[key].taskName, tasks[key].executor, tasks[key].task, tasks[key].expiration, actualStatus, key)
                if(actualStatus !== 0){
                    nrOfNotifications++;
                    if(actualStatus === 2){
                        lowTimeTask.push([tasks[key].taskName, tasks[key].executor, tasks[key].expiration])
                    }
                    else{
                        failedTasks.push([tasks[key].taskName, tasks[key].executor, tasks[key].expiration])
                    }
                }
            }
            else{
                taskConstructor(tasks[key].taskName, tasks[key].executor, tasks[key].task, tasks[key].expiration, tasks[key].status, key)
            }
        }
    }
    if(nrOfNotifications){
        let numberOfNotifications = document.querySelector(".notificationAboutTasks")
        numberOfNotifications.innerHTML = nrOfNotifications
        numberOfNotifications.style.padding = "2px 5px "

        while (lowTimeTask.length > 0) {
            const poppedElement = lowTimeTask.pop();
            addElementIntoNotificationBar(poppedElement[0], poppedElement[1], poppedElement[2])
        }

        while (failedTasks.length > 0) {
            const poppedElement = failedTasks.pop();
            addElementIntoNotificationBar(poppedElement[0], poppedElement[1], poppedElement[2], true)
        }
    }
}
```

Figure 3: —

This is how you can add tasks:

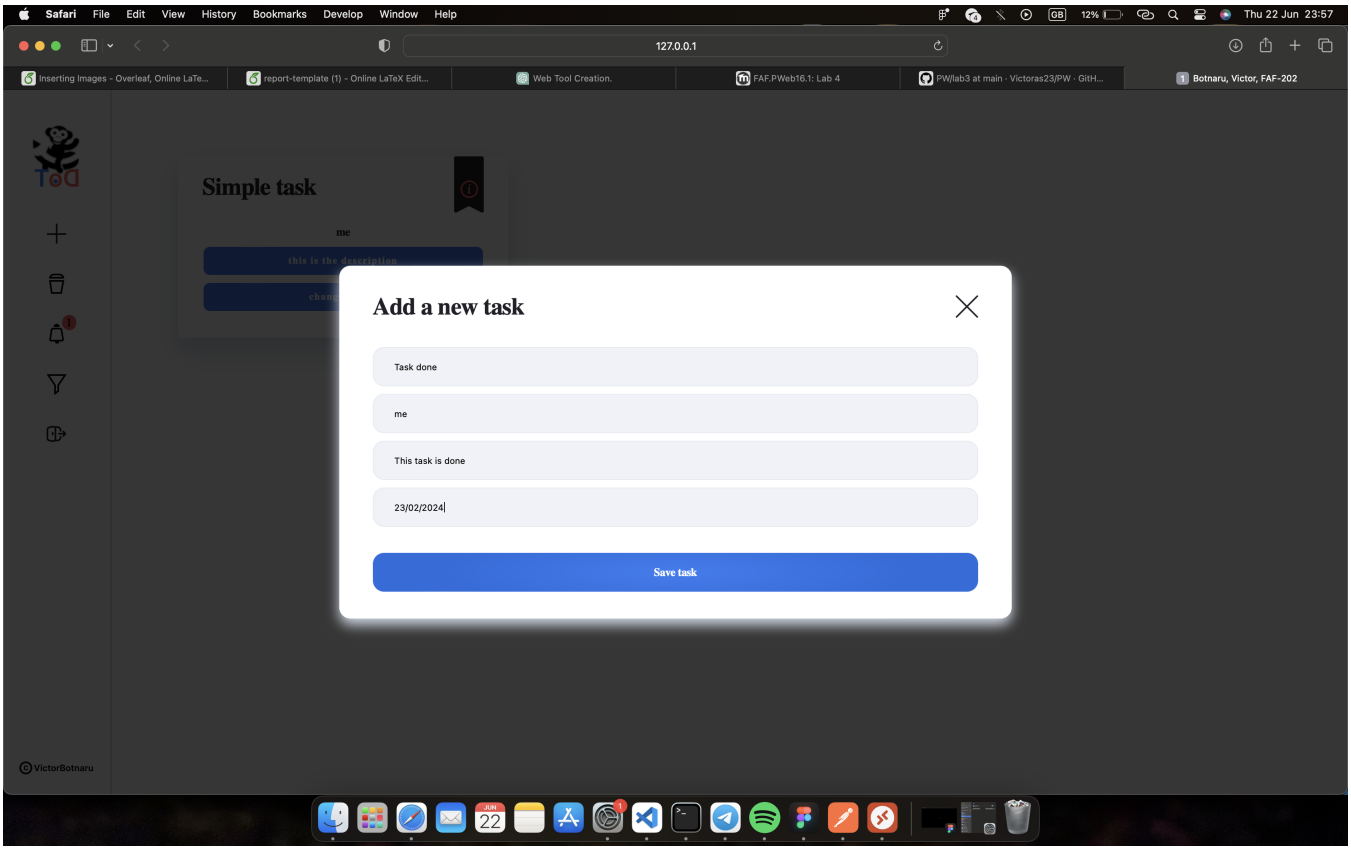


Figure 4: —

They are several types of :

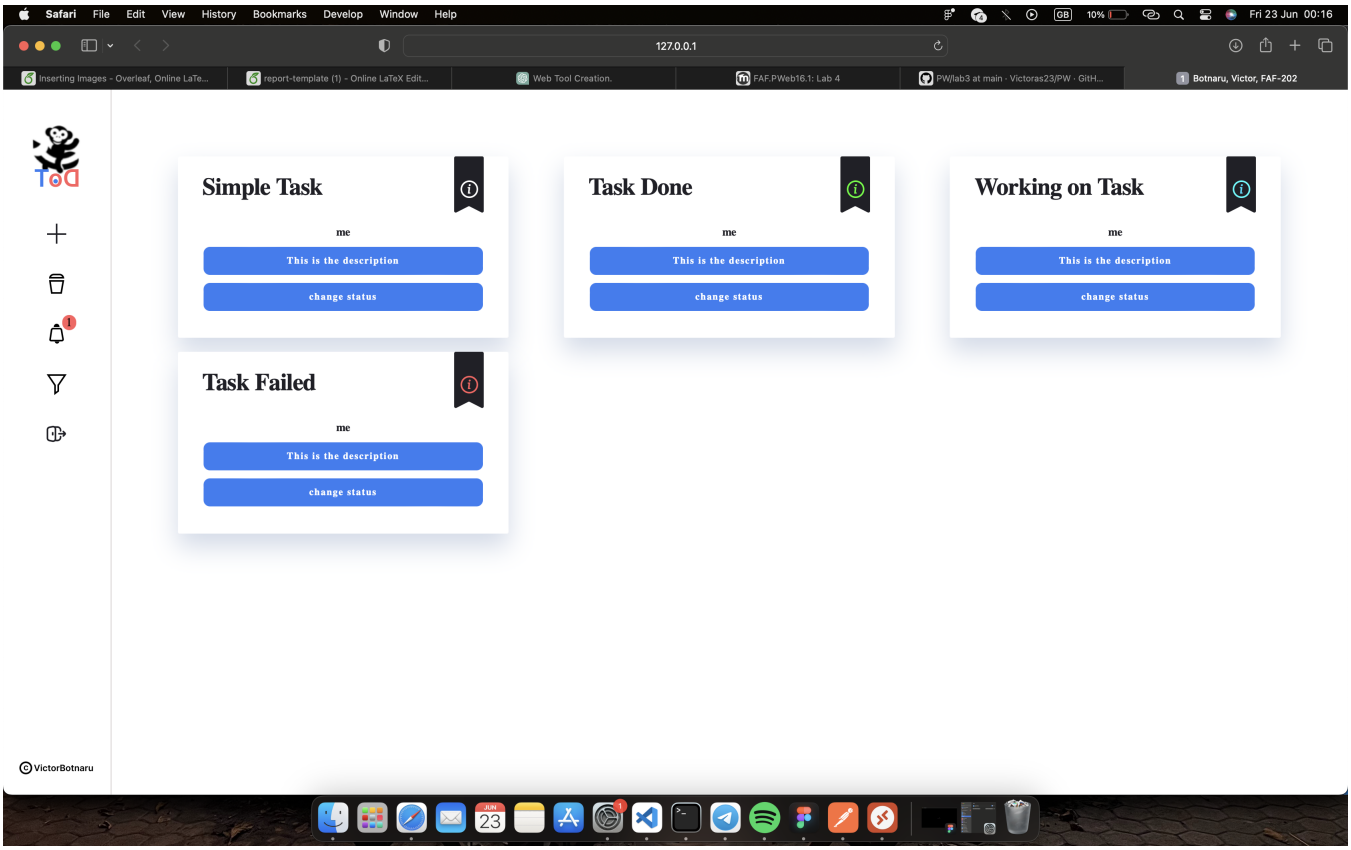


Figure 5: —

You can remove them :

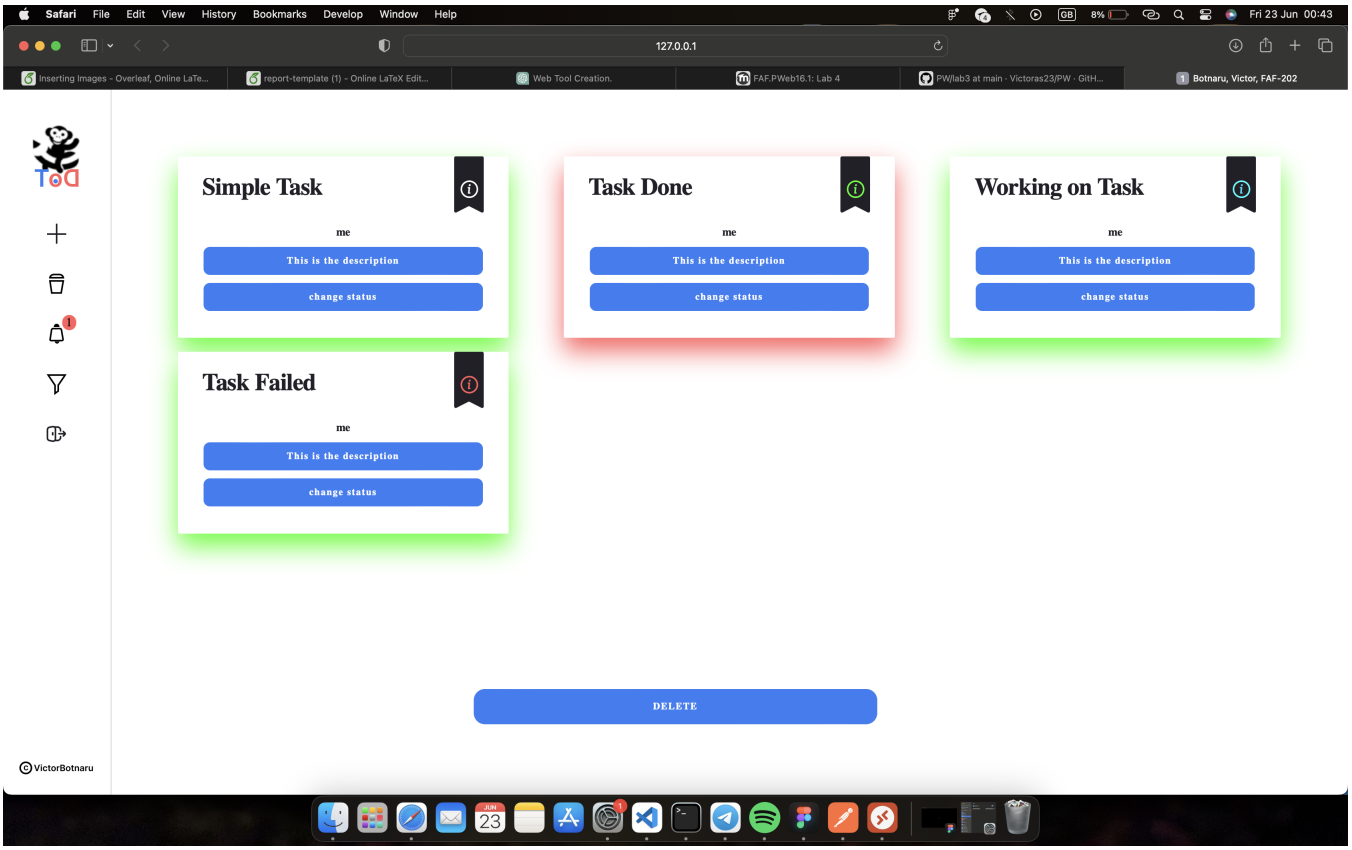


Figure 6: —

Filter them :

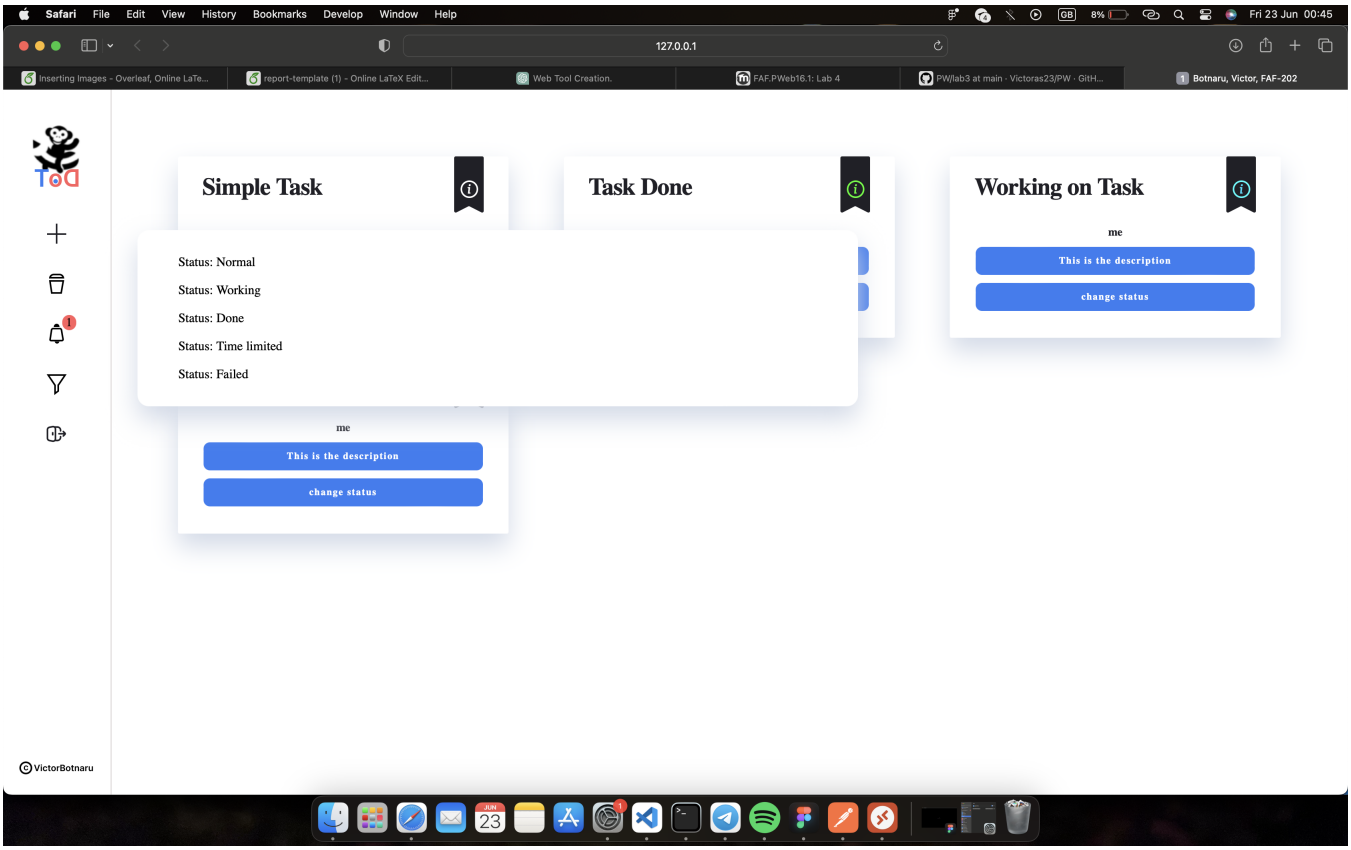


Figure 7: —

Change status :

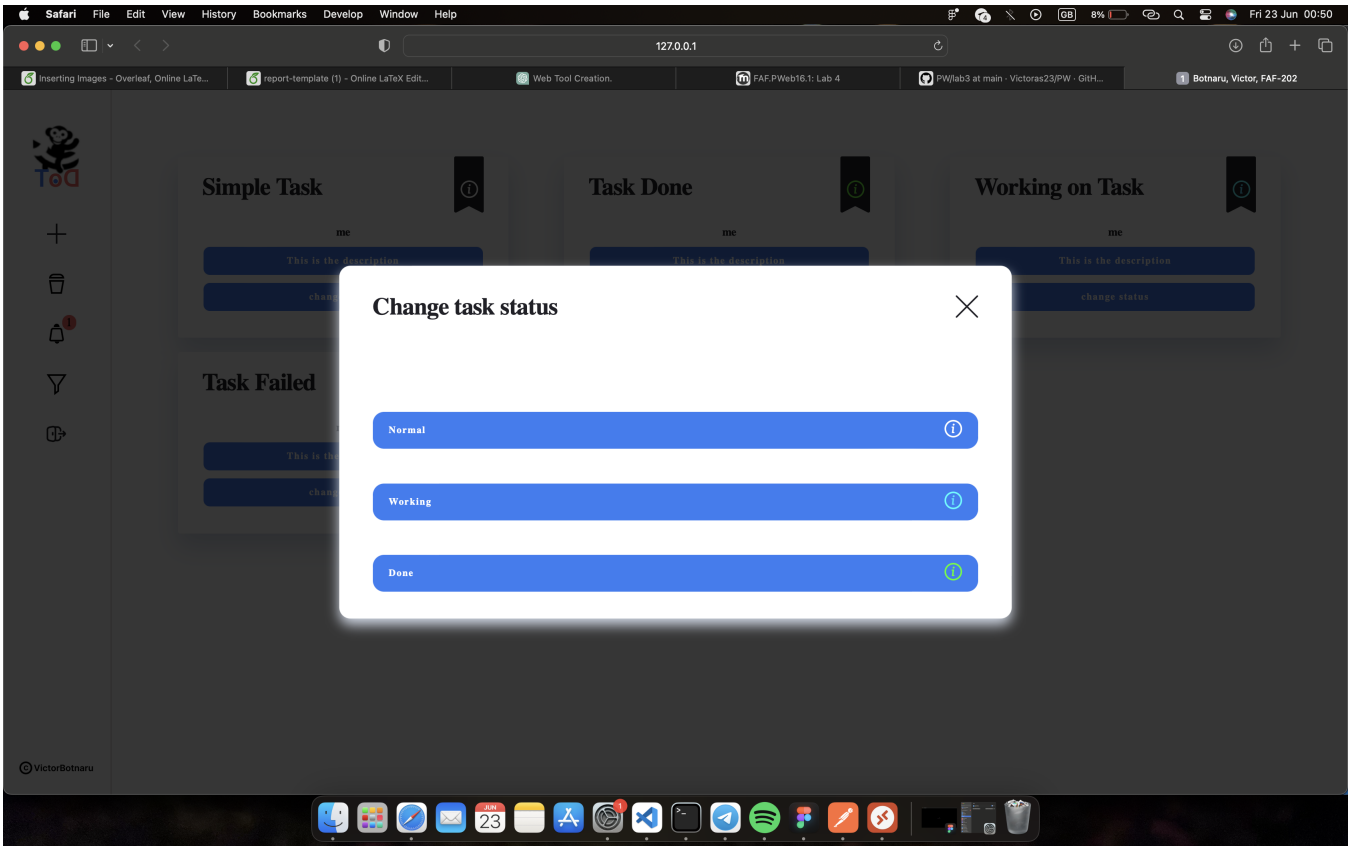


Figure 8: —

6 Conclusion

References

- GitHub Repository `https://github.com/Victoras23/PW`