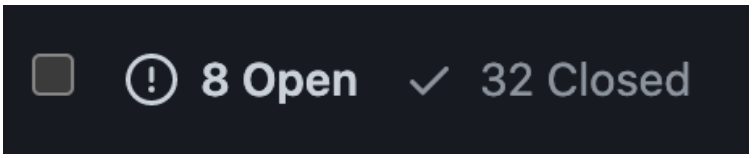
# Les Frais project report

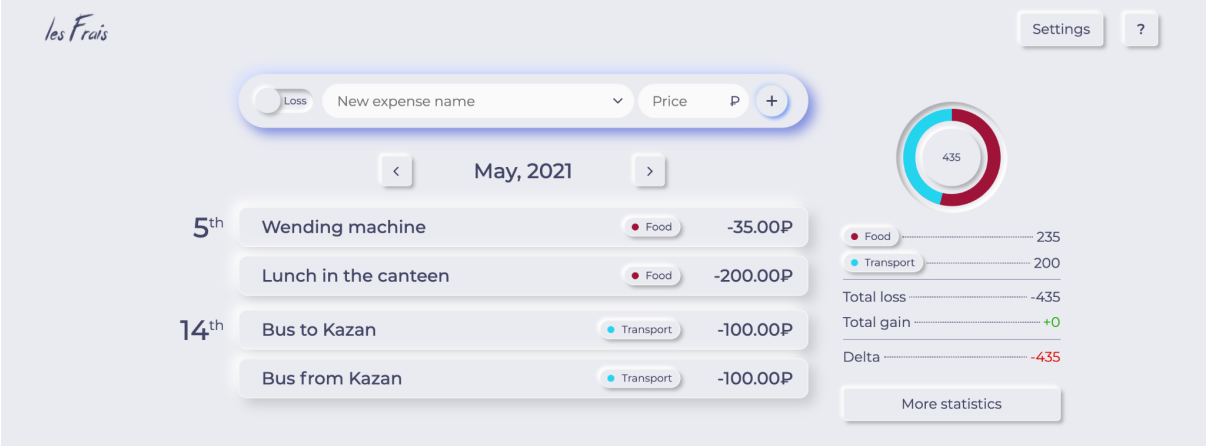
## 1. Before and after

In the beginning of the project, the only thing we had was an idea and some part of the application design made in Figma (here's the link btw).

During the Sprint 0, we defined 9 User stories which expanded into the 40 GitHub Issues. And this is what's left of it after the Sprint 5:

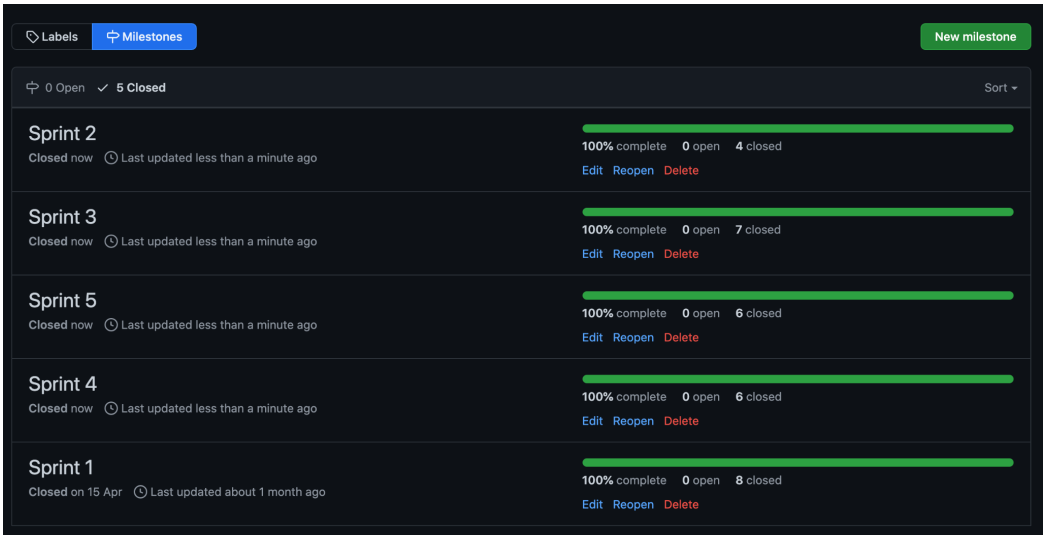



Concerning the project, we've made the expense tracking web application from scratch. It has all the features to properly start working with it. Good thing is: the app is fast, performant and nice-looking (better than designs).



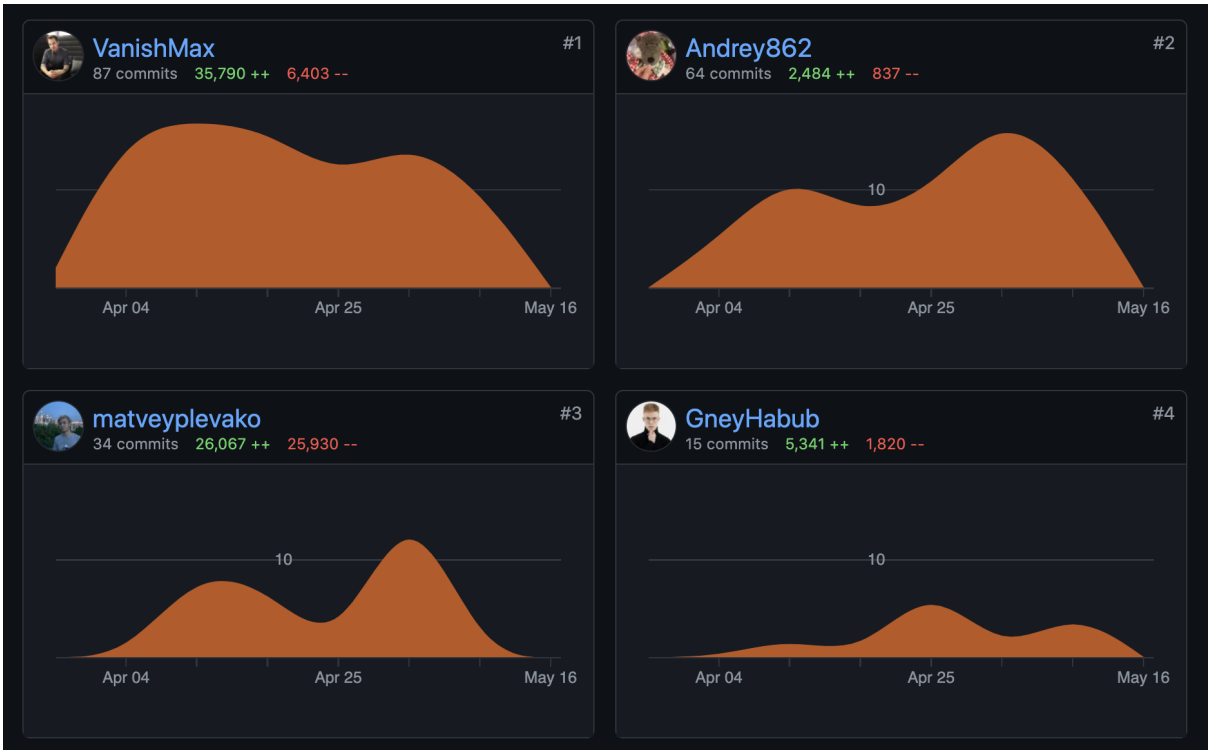

## 2. Organization and Traceability

During the course, we had 6 sprints lasting one week each. User Stories were assigned to Sprints initially, but later on changed because of their difficulties and our lack of time.



Starting from Sprint 1, we started working with GitHub milestones, and along with the project Kanban board it provided us with a clear view on the project.

All work is transparent since the repo is public and we tried our best to keep up with the Scrum methods and put necessary information to the Readme and issues. The improvements are written step by step in the Readme.




The picture above represents the amount of effort the team put to the project. We've been spending approximately 20 human hours a week on the project finishing on average 6 issues.

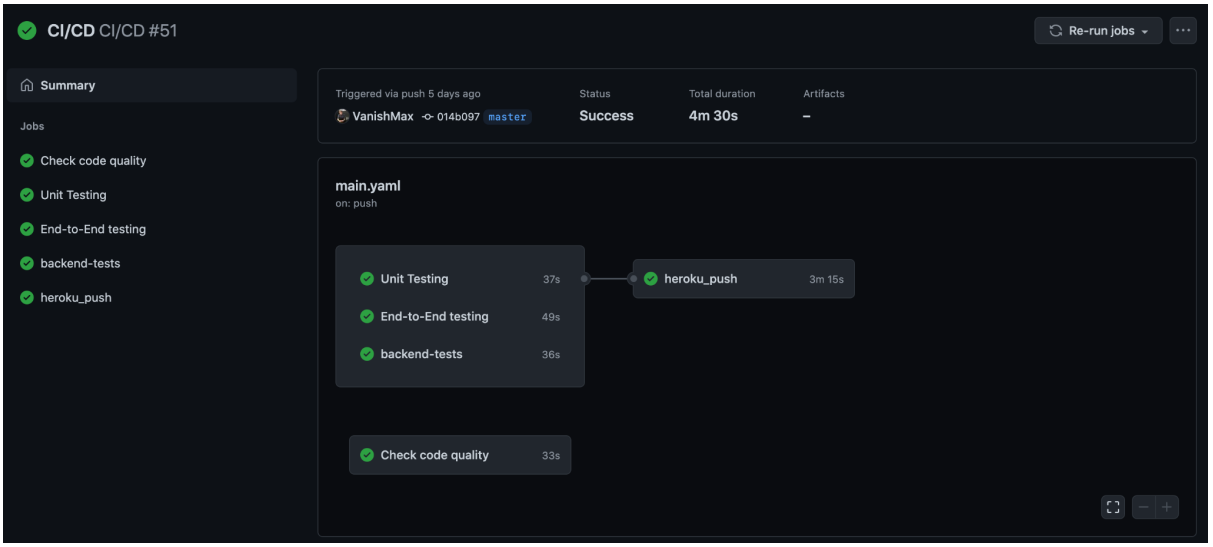
## 3. Documentation

We have several branches of documentation.

1. Organizational – extensive Readme
2. Backend endpoints documentation in the form of Swagger
3. User documentation placed right in the website.
4. Code documentation – extensive code comments.

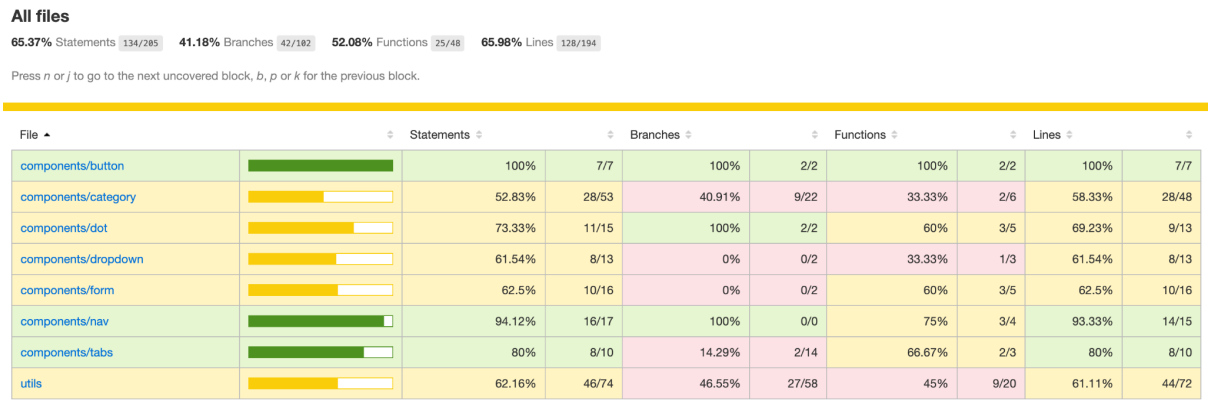
## 4. Tests, Quality Assurance and Releases

All the work was performed in branches called *"feature/issue-name"* and merged to "DEV" since DEV had CI/CD integrated, which resulted in deployments to Heroku.




As you can see, we made several workflows in GitHub Actions. Deployment to Heroku is dependent on frontend and backend tests:

### 1. Frontend Unit tests



**All files**

**65.37%** Statements 134/205 **41.18%** Branches 42/102 **52.08%** Functions 25/48 **65.98%** Lines 128/194

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

| File | Statements | | Branches | | Functions | | Lines | |
|---|---|---|---|---|---|---|---|---|
| components/button | 100% | 7/7 | 100% | 2/2 | 100% | 2/2 | 100% | 7/7 |
| components/category | 52.83% | 28/53 | 40.91% | 9/22 | 33.33% | 2/6 | 58.33% | 28/48 |
| components/dot | 73.33% | 11/15 | 100% | 2/2 | 60% | 3/5 | 69.23% | 9/13 |
| components/dropdown | 61.54% | 8/13 | 0% | 0/2 | 33.33% | 1/3 | 61.54% | 8/13 |
| components/form | 62.5% | 10/16 | 0% | 0/2 | 60% | 3/5 | 62.5% | 10/16 |
| components/nav | 94.12% | 16/17 | 100% | 0/0 | 75% | 3/4 | 93.33% | 14/15 |
| components/tabs | 80% | 8/10 | 14.29% | 2/14 | 66.67% | 2/3 | 80% | 8/10 |
| utils | 62.16% | 46/74 | 46.55% | 27/58 | 45% | 9/20 | 61.11% | 44/72 |

Screenshot above shows the tests coverage of frontend components with unit tests. It doesn't cover everything, but the main features are present ensuring proper quality and components stability.

### 2. Frontend End-To-End tests

Made with Cypress, those tests go through application features the same way a user would. These tests automatically generate videos of how a program enters the websites, puts the values to the input fields and interacts with the app.

### 3. Backend unit tests

```
System check identified 6 issues (0 silenced).
.......
----------------------------------------------------------------------
Ran 7 tests in 2.247s

OK
```

The tests are done on the most important parts of the backend ensuring the stability of the whole system and preventing bugs before they are created.

**4. Code quality checks**

We have set-up the linters on our code, so the whole team follows the same code style throughout the project. This way any possible newcomer can easily understand what's going on in the code

**5. Deployment**

Free Heroku server allowed us to easily integrate Continuous Delivery of the app to the web after all previous checks were passed. After every push to DEV or Master, the app becomes deployed to the website (might require 10-30s to warm-up).

## 5. Self-criticism

All in all, we've been working hard to produce the first-class quality product, but we were still making several mistakes on our way:

- Sometimes, working on the features, we've been forgetting about the Sprint planning, retrospectives and the backlog. So, even though the features were released, the milestones stayed the samee way they were a week before.
- It would be way better to write tests in a TDD manner, so it gives more meaning to the tests. For now, they were written for already made good components of the system.
- We didn't manage to prepare the burnout charts.
- We didn't do code reviews.
- We forgot about release history and versioning of the app.