SPRINT RETROSPECTIVE

SPRINT 4

Software Engineering II - Group E

Process measures

Macro statistics

	COMMITTED	DONE
Total number of stories	**28**	28
Total number of stories in sprint	**4**	4
Total points in sprint	**25**	25

Detailed statistics

STORY ID	NUMBER OF TASKS	POINTS	TOT HOURS ESTIMATION	TOT HOURS SPENT	HOURS PER TASK
0	2		3	3	1,5
23	5	8	10	11	2,2
24	12	13	18	21	1,75
26	2	3	6	5	2,5
28	3	1	2	2	0,667
REDUCE TECHNICAL DEBT			4	6	
UNIT TESTING			8	8	
E2E TESTING			3	3	
REFACTORING			10	10	
CODE REVIEW			6	6	
TOTALS			**70**	**75**	
HOURS PER TASK		MEAN	1,779	STD DEV	0,707
TOTAL TASK ESTIMATION ERROR RATIO			0,933		

Story#0:
Configure SonarCloud and produce PHPunit coverage.

Quality measures

- **UNIT TESTING**

Total hours estimated: 8h
Total hours spent: 8h
# of automated unit tests: 58

Coverage: 66.9% (weighted average between coverage of *db.php* and *functions.php*)
NOTE: we decide to test only meaningful functions (in the code they are all commented).

- **E2E TESTING**

Total hours estimated: 3h
Total hours spent: 3h

- **CODE REVIEW**

Total hours estimated: 6h
Total hours spent: 6h

- **Technical Debt management:**

Total hours estimated: 4h
Total hours spent: 6h

- **Overall Technical Debt at the demo:**

number of days: 7
debt ratio: 3.2%
rating: A

Assessment

1. What caused your errors in estimation (if any)?

   We underestimated very few stories, since we had recognized their difficulty during the estimation. Nonetheless the error in the estimation was produced anyway because some stories were even more complicated than expected, and moreover we did not assume there would have been complicated interactions between old and new stories.

2. What lessons did you learn (both positive and negative) in this sprint?

   We tried to perform better estimation than in the previous sprint, committing for an higher number of expected hours. We learned that we can be more precise, but still the margin we left was too low and we ended up doing more hours than our budget. We understood also that is much more difficult to estimate the number of hours needed to complete large stories, but things become easier if we split the story in smaller tasks at estimation time. Finally, we understood that it is more important the quality than the quantity, since also this time we had to perform some refactoring of the previous stories.

3. Which improvement goals set in the previous retrospective were you able to achieve?

   In the previous sprint we proposed for ourselves to take care, before any commit, of reviewing the portion of tests dedicated to the story implemented and to see if their execution succeed or not. We decided to solve the problem of "failing tests" by introducing a protocol for which all the functions inserted should have to be well commented (in order to make the work of the tester easier), by highlighting the inserted or modified functions with special comments like "NEEDS TO BE TESTED" and signaling to the tester via private message if the semantics of the function had been changed.

4. Which ones you were not able to achieve? Why?

   What we proposed for the previous sprint was successfully implemented.

5. Propose 1 or 2 improvement goals for the next sprint and specify how to achieve them (technical tasks, team coordination, etc.)

   We propose ourselves to remove any code smell present in the system.
   We aim to improve the GUI in order to make the user experience more rewarding.
   We also would like to implement the principles of pair programming in particular the work in pairs in order to augment the coordination among the team and to make us more compact and unified. Moreover, considering that the story for the insertions in the timetables of a specific teacher have been inserted in the current sprint, the selection of the day/hour for new lectures/assignments is for the moment not coherent with the teacher's timetable. This is a little inconvenience, since this is possible only because we trust teachers; anyway we aim to solve this little problem in order to make the teacher experience a little bit more secure and smooth.

6. One thing you are proud of!!

   We learned to use SonarCloud, which allowed us to improve the quality of the code (for example to reduce the duplicated code blocks and to make functions available for other team members) and moreover to increment the security of our software.