Ardeeno WebApp

T41-SE22

D5-Report Finale

v1.0

alessandro.manfucci@studenti.unitn.it enrico.cescato@studenti.unitn.it
m.sottocornola-1@studenti.unitn.it

29-12-2022

Indice

Abstract

Questo documento rappresenta la parte finale di ogni progetto, ovvero l'introspezione. Riassumere come è stato svolto il lavoro – e quale flusso è stato implementato – è fondamentale per migliorare il processo di lavoro.

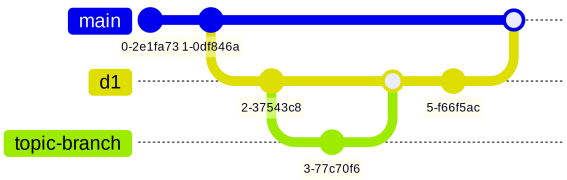
1 Organizzazione del lavoro

1.1 Flusso di lavoro

Il flusso di lavoro è stato organizzato in maniera Agile con la suddivisione dei deliverables in tasks (in genere composte da un singolo capitolo del deliverable), che sono poi state scelte dai membri del team durante i meeting settimanali.

Per gestire il flusso di lavoro è stato utilizzato GitHub, con il repository `t41-se22/ardeeno-workondeliverables`. Il flusso di lavoro è stato sviluppato in tre branches:

- `main` branch: contenente la versione dei documenti pubblicata in `t41-se22/Deliverables`
- `d<i>` branch: utilizzato per lavorare sul deliverable-`<i>`
  - `topic-branch`: utilizzato per lavorare su una task del deliverable-`<i>` (solo nel caso di lavoro in contemporanea di più membri del team)



Il branch `d<i>` non comprende il lavoro effettuato esclusivamente sul deliverable-`<i>`, piuttosto comprende il lavoro effettuato durante la scrittura della **prima** versione del deliverable-`<i>` su tutti i deliverable (da 1 a `<i>`) . Raggiunta la prima versione del deliverable-`<i>` si effettua il merge sul `main` e si crea il nuovo branch `d<i+1>`. Dunque, quasi sempre il tempo di vita del branch `d<i>` coincide con le scadenze stabilite dal corso.

Per scrivere i Deliverables è stato usato il linguaggio Quarto MarkDown, il quale – essendo testo semplice – ha facilitato il merging dei branch. Per facilitare l'utilizzo di questi strumenti e di questo flusso di lavoro sono stati caricati più `README.md`. Inoltre, si è usato GoogleDrive per condividere documenti non prettamente associati ai deliverable, soprattutto per ideare il sistema da realizzare e condividere il work-time.

Per quanto riguarda lo sviluppo del codice, si è utilizzato il metodo KanBan, con una KanBan Board integrata con GitHub.

Per quanta riguarda le comunicazioni tra i membri del team è stato usato Telegram per avvisare problemi/cambiamenti/assenze. Inoltre, sono stati organizzati brevi meeting settimanali, così suddivisi:

- KickOff Meeting all'inizio del progetto per comprendere la realtà di riferimento del sistema e installare l'ambiente di lavoro comune (VSCode, git, …)
- Meetings settimanali in Saletta BUP per fare il punto della situazione, gestire problemi/cambiamenti e scegliere le tasks da svolgere – organizzati nel periodo di tempo delle scadenze `D1`, `D2`
- StandUp Meetings settimanali prima/dopo le lezioni di ISW per fare un breve punto della situazione e scegliere le tasks da svolgere – organizzati nel periodo di tempo delle scadenze `D3`, `D4`
- Meeting Finale in Saletta BUP per rivedere insieme il Report Finale

1.2 Ruoli

Non vi è stata una specializzazione dei ruoli.

	Ruolo
Alessandro Manfucci	Analyst, Programmer, Team Leader
Enrico Cescato	Analyst, Programmer
Matteo Sottocornola	Analyst, Programmer

1.3 Tasks List

Deliverable	Task	Chosen By (A,E,M)	Due To
D1-Documento di Progetto	Definizione Obiettivi	A,E,M	27/09
	Definizione Requisiti	A,E	10/10
	Mock-Up FrontEnd	A,M	10/10
	Schema BackEnd	A	10/10
D2-Documento di Specifica	Spec ReqF	A,E,M	24/10
	Spec ReqNF	A	24/10
	Diagramma di Contesto	A	30/10
	Diagramma dei Componenti	A,M	9/11
D3-Documento di Architettura	Class Diagrams	A,E	23/11
	OCL	A	02/12
D4-Documento di Sviluppo	User Flows	A,M	22/12
	Resources Extraction	A	22/12
	Resources Models	A	22/12
	Sviluppo BackEnd	A,E	02/01/23
	Sviluppo FrontEnd	A	02/01/23
	Deployment	A	02/01/23
	Testing	A,E	02/01/23

1.4 KanBan Board

Si riporta la sola struttura delle KanBan Board BackEnd e FrontEnd

1.4.1 BackEnd

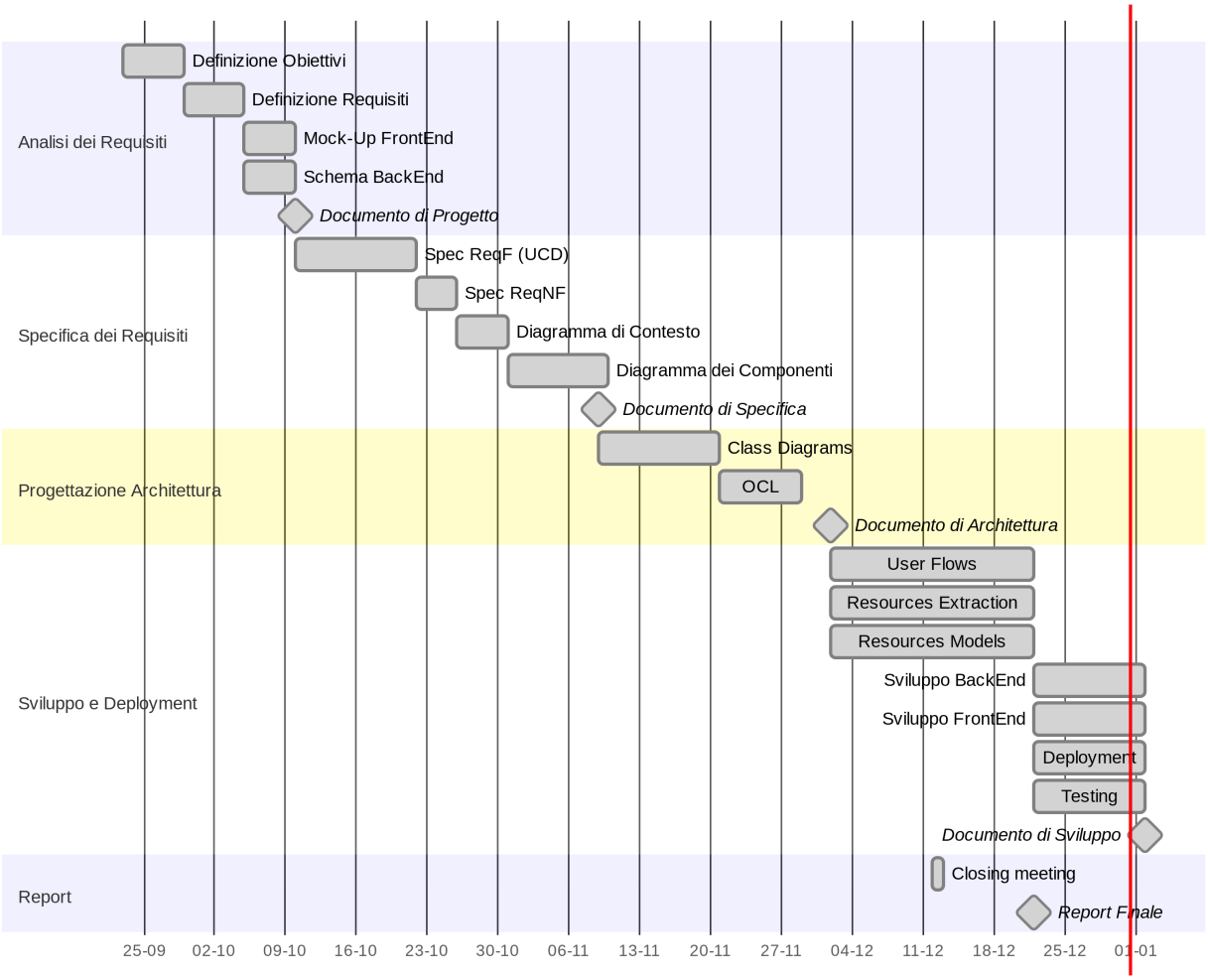
To do	Development	Swagger Docs	Jest Testing	Done

1.4.2 FrontEnd

To do	Development	Done

1.5 Gantt Diagram

Anche se non prettamente Agile, è stato utilizzato un Gantt per avere un'idea approssimativa delle dipendenze tra le singole task. Chiaramente, dato il processo iterativo utilizzato, il Gantt rappresenta solo il processo a cascata della **prima** versione di ogni deliverable.



2 Carico e distribuzione del lavoro

Per la rendicontazione del lavoro svolto si sono utilizzati i messaggi di commit di git, stabilendo un formato standard che estende quello di GitHub:

```
git commit -m "My Commit message

hh:mm


Co-authored-by: name <name@example.com>
Co-authored-by: another-name <another-name@example.com>"
```

Dove `hh:mm` rappresenta il work-time impiegato tra il commit appena effettuato e il precedente commit con il campo `hh:mm` (dunque `hh:mm` può essere omesso). Il co-authoring permette di associare un certo commit (e dunque un certo `work-time`) a più membri del team.

Dare un work-time ad ogni commit pone due vantaggi:

- Il work-time è sempre associato ad un lavoro concretamente realizzato. Questo aiuta a distinguere il tempo di studio dal tempo impiegato concretamente al progetto.
- Il work-time è sempre associato ad un branch e ad una repository. Quindi, visto il workflow utilizzato, ad un deliverable.

Tuttavia, come accennato prima, dato il processo di lavoro Agile sarebbe complicato separare il lavoro impiegato e rendicontato per ogni singolo deliverable. Dunque, per associare un certo `work-time` ad un certo deliverable (uno e uno solo) si considera il branch `d<i>` su cui è stato pubblicato. Quindi, il `work-time` calcolato per il deliverable-`<i>` comprende il `work-time` impiegato su tutti i deliverables (da 1 a `<i>`), durante il periodo di scrittura della **prima** versione del deliverable-`<i>`. Fa eccezione il branch `d5` – che comprende modifiche esclusivamente per il D5.

2.1 WorkTime[1]

	D1	**D2**	**D3**	**D4+BE+FE**	**D5**	**TOT**
Alessandro Manfucci	46.5	60	43.5	122.5	15.5	288
Enrico Cescato	31	17	5	1	1.5	55.5
Matteo Sottocornola	16.5	17	0	11.5	1.5	44.5
TOT	94	94	48.5	135	18.5	388

I commit `co-authored` (che assegnano un certo `work-time` a più membri contemporaneamente) possono falsare i tempi totali per Deliverable. Questi sono stati utilizzati per lo più nel D1. Si propone una seconda tabella:

	D1	**D2**	**D3**	**D4+BE+FE**	**D5**	**TOT**
Alessandro Manfucci	28	57	43.5	122.5	14	265
Enrico Cescato	12.5	14	5	1	0	32.5
Matteo Sottocornola	10.5	13	0	11.5	0	35
A+E	12.5	1	0	0	0	13.5
A+E+M	6	2	0	0	1.5	9.5
TOT	69.5	87	48.5	135	15.5	355.5

2.1.1 Statistiche

Si riportano alcune statistiche[2] sui tempi di rendicontazione in ore decimali:

[1] Database .json disponibile sul repository github.com/T41-SE22/ardeeno-workondeliverables
[2] Le statistiche sono solo su quei commit con $T > 0$

- $N = 140$
- $N_{T>0} = 68$
- $T_{avg} = 5.22$
- $T_{max} = 21.5$
- $T_{min} = 0.5$
- $\sigma^2 = 12.89$

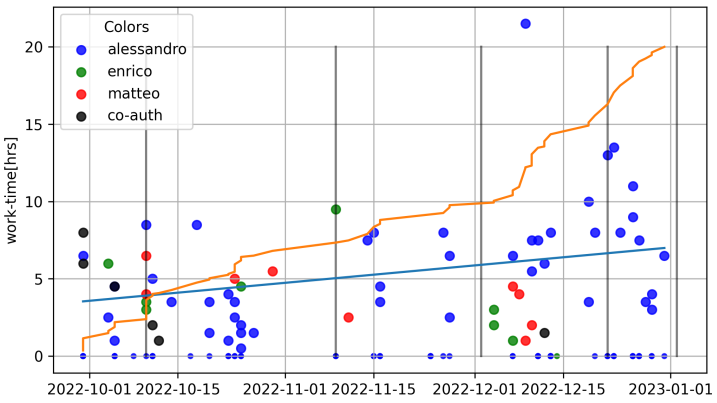


Figura 1: Commits

Quindi, il metodo di rendicontazione è stato usato in maniera costante.[a]

[a] Ulteriori statistiche su docs.google.com

3 Autovalutazione

	Voto
Alessandro Manfucci	30
Enrico Cescato	23
Matteo Sottocornola	21