

# Relazione di progetto

Progetto di Ingegneria del Software 2021/2022 - Team 10

Giuseppe Carrino ( PO )  
Umberto Carlucci ( SM )  
Agata Cavigioli  
Andrea Loretti  
Andrea Schinoppi

## 1 Prima partita a Scrumble

### 1.1 La preparazione

In data 10 Ottobre si è svolta la prima partita del team a Scrumble.

Le User Stories utilizzate nel corso del gioco sono state quelle riguardanti lo shop BuyMe inserite a pagina 37 della tesi di laurea di Roberto Barbone (AA 2019/2020), che ci è stata fornita come esempio.

US	Value	Loretti	Cavigioli	Carlucci	Schinoppi	Carrino	Finale
1	5	3	3	3	3	Alta	3, A
2	1	1	1	2	1	Bassa	1, B
3	1	1	2	2	2	Bassa	2, B
4	4	3	3	3	3	Media	3, M
5	2	3	3	2	2	Bassa	3, B
6	5	3	4	5	4	Alta	4, A
7	4	5	5	4	4	Media	5, M
8	2	1	1	1	1	Bassa	1, B
9	4	2	3	4	4	Bassa	4, B
10	3	1	1	3	2	Bassa	2, B
11	2	4	4	4	3	Media	4, M
12	4	5	5	5	5	Media	5, M
13	2	3	4	4	3	Alta	3, A
14	5	1	4	2	2	Alta	4, A
15	1	1	1	1	1	Bassa	1, B

Figura 1: Tabella delle opinioni sulla difficoltà delle User Stories proposte

Il Product Owner (indicato in verde) ha deciso la priorità delle User Stories, mentre per le difficoltà abbiamo riportato l'opinione di ognuno dei membri prima della discussione, che si è conclusa nelle decisioni riportate nella colonna finale.

Lo stato definitivo della tabella di Scrumble era dunque il seguente.

		PRIORITA'		
		Alta	Normale	Bassa
D I F F I C O L T A	1 - XS			2, 8, 15
	2 - S			3, 10
	3 - M	13, 1	4	5
	4 - L	6, 14	11	9
	5 - XL		7, 12	

## 1.2 La partita

La partita si è svolta in 4 sprint da 9 giorni l'uno. Segue una breve descrizione degli Sprint Backlog.

### 1.2.1 Primo sprint

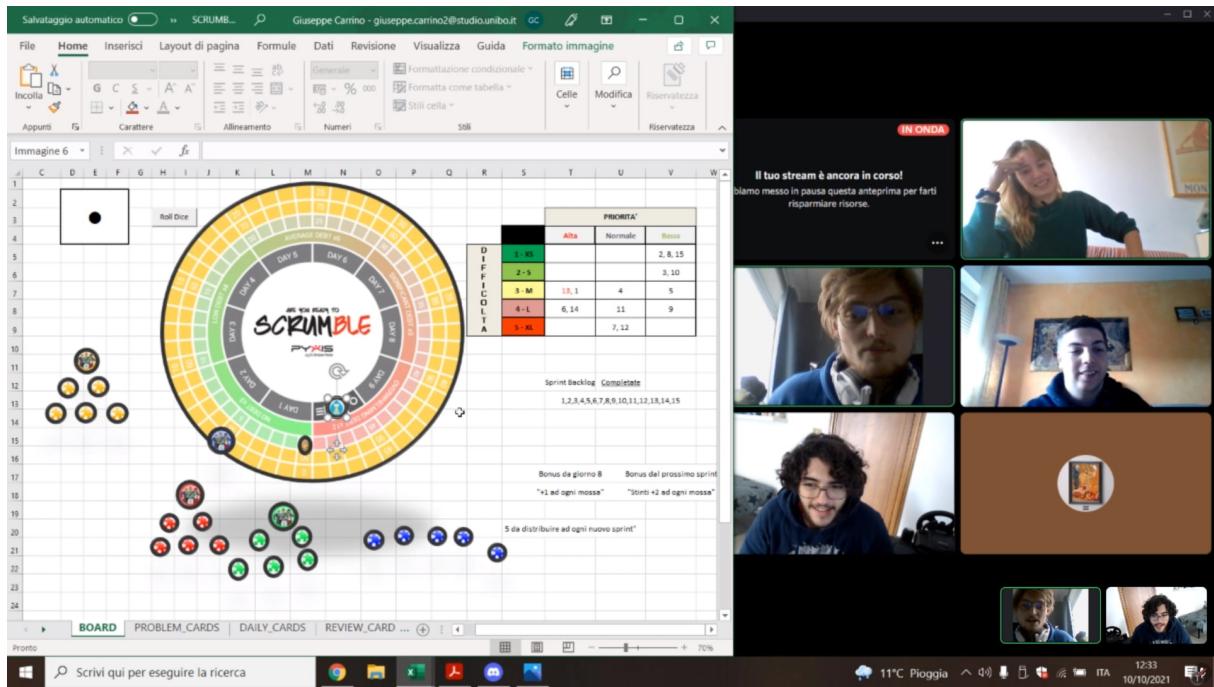
Durante il primo sprint abbiamo completato le User Stories n° 8, 13, 11. Abbiamo azzerato il fattore di debito alla fine dei giorni di sviluppo, tuttavia a causa delle review cards è riaumentato fino a 5. Durante la review il PO ha fatto notare al team come sono state scelte le user stories da attuare, ovvero tornando a guardare la definizione iniziale delle stories e il valore per l'utente e tralasciando il lavoro fatto sulla valutazione della priorità a livello di sviluppo: un errore da tenere a mente durante i prossimi sprint. Durante la retrospettiva è stato posto come obiettivo minimo raggiungere 70/75 tasks. Lo SM, infatti, ha fatto notare che fossero state completate tutte le task già al giorno 8, utilizzando quindi un intero giorno solo per abbassare il debito, denotando un certo pessimismo.

### 1.2.2 Secondo sprint

Durante il secondo sprint abbiamo completato le User Stories n° 12, 1, 2. Abbiamo azzerato il fattore di debito alla fine dei giorni di sviluppo. Durante la review il debito è rimasto 0 e visto il buon risultato il PO ha chiesto se fosse possibile porsi un obiettivo più alto. Le daily cards pescate in questo turno hanno permesso di avere dei bonus (considerevoli) nei successivi lanci del dado, a discapito della perdita di diversi giorni di sviluppo da parte di due membri del team. Inoltre, un'altra daily card permanente ci ha permesso di ignorare molte problem card successive. Durante la retrospettiva lo SM si è mostrato soddisfatto e ha quindi alzato l'obiettivo per il prossimo sprint a 85/90 tasks completate.

### 1.2.3 Terzo sprint

Sono state completate le User Stories n° 7, 6, 10, 14, 15 per un totale di ben 5, anche grazie ai bonus ottenuti dalle daily cards dello sprint precedente. In particolare, le prime 3 sono state completate già al giorno 7, aggiungendo quindi le altre 2 nei giorni successivi. Alla fine della review il debito è aumentato fino ad 8. Il PO si è limitato a complimentarsi. Durante la retrospettiva lo SM ha deciso di alzare l'obiettivo di tasks completate a 100, basandosi sull'ottimo lavoro del team durante l'ultimo sprint.



#### 1.2.4 Quarto sprint

Sono state completate le User Stories n° 4, 5, 3, 9, terminando il lavoro e azzerando il debito. La review non ha aggiunto debito, permettendo la conclusione della partita. Il PO ha apprezzato il rischio di voler raggiungere le 100 tasks in uno sprint. Lo SM si è complimentato a sua volta per aver terminato le user stories ed essere riusciti anche ad azzerare il debito accumulato.

### 1.3 Autovalutazione

GOAL	QUESTIONS	EVALUATION	Carlucci	Carrino	Cavigioli	Loretti	Schinoppi
Learn	Q1	<b>1 = no idea of the Scrum roles</b> <b>5 = perfect knowledge of the roles and their jobs</b>	4	4	4	3	3
	Q2	<b>1 = couldn't repeat the game</b> <b>5 = could play the game as a Scrum Master by himself</b>	5	5	4	4	4
	Q3	<b>1 = totally lost</b> <b>5 = leads the game driving the other players</b>	4	4	3	3	3
Practice	Q4	<b>1 = feels the game is unrepeatable</b> <b>5 = feels the game could be played in any situation</b>	5	5	5	5	4
	Q5	<b>1 = 0 to 3 stories</b> <b>2 = 4 to 6</b> <b>3 = 7 to 9</b> <b>4 = 10 to 12</b> <b>5 = 13 to 15</b>	5	5	5	5	5
	Q6 <b>ONLY DEV TEAM</b>	<b>1 = abnormal difference from the other players</b> <b>5 = coherent and uniform with the group most of the time</b>	Dev	Dev	4	5	5
Cooperation	Q7	<b>1 = never speaks with the other players</b> <b>5 = talks friendly to anyone in every situation</b>	4	5	4	4	4
	Q8	<b>1 = never puts effort in doing something</b> <b>5 = every time is willing to understand what is going on</b>	4	4	4	4	3
	Q9	<b>1 = never asks for an opinion</b> <b>5 = wants to discuss about every topic</b>	5	5	4	5	4
Motivation	Q10	<b>1 = not involved by the game</b> <b>5 = always makes sure everyone is on point</b>	4	5	3	5	4
	Q11 <b>ONLY FOR PO</b>	<b>1 = poor/absent advices</b> <b>5 = wise and helpful suggestions when is required</b>	POo	4	POo	POo	POo
	Q12	<b>1 = doesn't express opinions during retrospective</b> <b>5 = feels the retrospective fundamental to express opinions</b>	4	3	4	4	3
Problem Solving	Q13	<i>On the game board, if the debt pawn is on the lowest stage, the evaluation is 5, for every higher stage it decreases by 1</i>	5	5	5	5	5
	Q14 <b>ONLY DEV TEAM</b>	<i>Calculate the average of tasks left for each sprint: 1 = 21+ 2 = 16-20 3 = 11-15 4 = 6-10 5 = 0-5</i>	Dev	Dev	5	5	5
	Q15 <b>ONLY FOR PO</b>	<i>Same evaluation as Q14 for the PO</i>	POo	5	POo	POo	POo

L'esperienza di gioco ci è sembrata molto educativa, oltre che piacevole, per entrare a poco a poco nella mentalità Agile e iniziare a capire i ruoli. Ci è risultata chiara fin da subito la differenza nello stile di lavoro rispetto a come abbiamo affrontato altri progetti in passato: la natura ciclica del gioco (e del metodo) ci pone davanti a problemi diversi dal solito e a soluzioni originali ed interessanti.

Ci siamo resi conto alla fine del progetto di quanto l'andamento del lavoro effettivo abbia ricalcato i passi del gioco di Scrumble: gli Story Point portati a termine sono cresciuti ad ogni Sprint grazie alla presa di coscienza crescente verso il metodo ed è stato molto importante per noi arrivare ad uno stato di soddisfazione (senza debito tecnico) ad ogni iterazione.

## 1.4 Decisione dei ruoli

Poiché abbiamo trovato particolarmente calzanti i ruoli assegnati durante la partita di prova abbiamo scelto di mantenerli. La decisione finale è stata dunque la seguente:

- Giuseppe Carrino come Product Owner operativo
- Umberto Carlucci come Scrum Master
- Il team degli sviluppatori composto da:
  - Agata Cavigioli
  - Andrea Loretta
  - Andrea Schinoppi

## 2 Product Backlog

Giuseppe Carrino, nel ruolo di Product Owner, si è occupato della stesura delle user stories del progetto: la progettazione ha restituito le stories riportate di seguito insieme ai relativi criteri di accettazione (CA):

- Come padre apprensivo
  - ”voglio poter visualizzare i tweet geolocalizzati relativi ad un utente, un hashtag o un testo” per sapere da dove twitta mio figlio.  
**CA:** i tweet devono essere identificati da un punto sulla mappa
- Come sindaco di un piccolo comune
  - ”voglio poter cercare i tweet pubblicati in una data località o da utenti residenti in una data località” per sapere quali tweet vengono pubblicati dal mio paese.  
**CA:** deve essere possibile inserire nella ricerca un'area geografica  
**Test:** Tutti i tweet ottenuti dalla ricerca sono geolocalizzati nella zona inserita.
- Come utente di Twitter
  - ”voglio avere accesso a una dashboard più o meno dettagliata dove poter visualizzare i tweet raccolti e filtrarli” per utilizzare l'app in maniera più profonda e interessante.  
**CA:** la dashboard deve contenere una barra di ricerca
- Come social media manager
  - ”nella dashboard voglio poter visualizzare i tweet con un dato hashtag” per sapere cosa viene tweettato secondo i trend attuali.  
**CA:** L'hashtag deve essere del tipo ”#xyz”  
**Test:** Tutti i tweet ottenuti dalla ricerca contengono l'hashtag inserito.

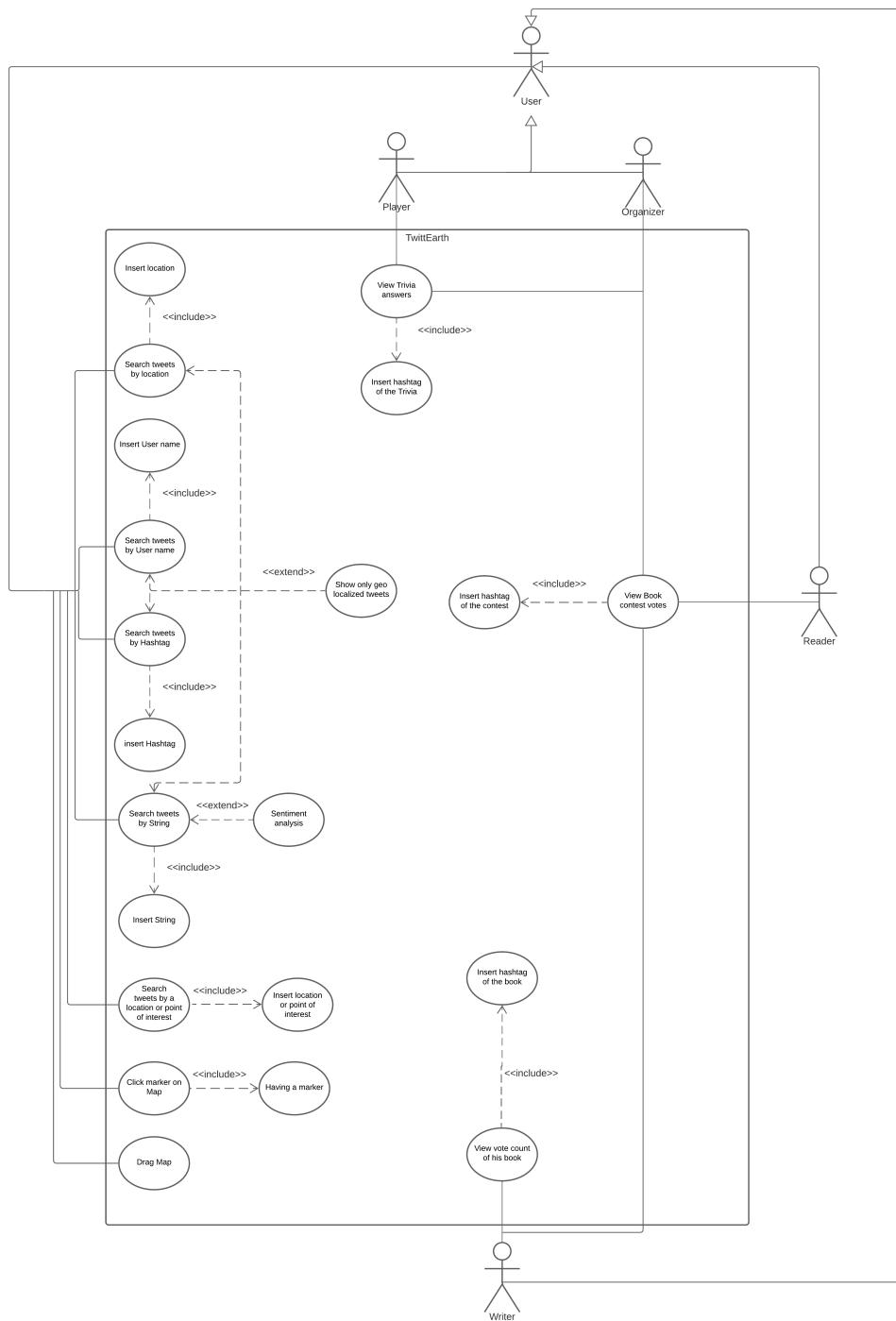
- Come fan di un artista  
 "nella dashboard voglio poter visualizzare solo i tweet di un determinato utente"  
 per vedere tutti i tweet del mio idolo.  
**CA:** devo poter specificare la forma "utente" della mia ricerca  
**Test:** Verifica coerenza della user timeline dato un utente conosciuto.
- Come giornalista  
 "nella dashboard voglio poter visualizzare i tweet contenenti una data stringa"  
 per cercare frasi di impatto politico nei tweets.  
**CA:** devo poter specificare la forma "testo" della mia ricerca  
**Test:** Tutti i tweet ottenuti dalla ricerca contengono il testo inserito.
- Come utente poco istruito sulla geografia  
 "voglio poter visualizzare una mappa dalla quale reperire informazioni interessanti sui tweet"  
 per capire dove esattamente sono geolocalizzati i tweet che cerco.  
**CA:** Mappa creata grazie alle API di OpenStreetMap
- Come social media manager  
 "voglio poter visualizzare dei grafici di vario tipo che mostrino informazioni sui tweet"  
 per conoscere meglio le statistiche e i target dei miei e altrui contenuti.  
**CA:** Grafici a barre, a torta...  
 Informazioni rilevanti sono la presenza di geolocalizzazione, il numero di tweet di un utente in relazione al tempo ecc.
- Come responsabile di un museo  
 "voglio poter avere informazioni sul gradimento di un dato luogo/avvenimento sulla base dei tweet che lo riguardano."  
 per sapere se le visite guidate che organizzo sono gradite.  
**CA:** Gli eventi di cui analizzare il sentimento vengono cercati attraverso una ricerca testuale.  
 Focalizzarsi sugli algoritmi di sentiment analysis usando come input il testo dei tweet  
**Test:** Inserendo una parola solitamente positiva/negativa, si controlla che il sentimento sia positivo/negativo almeno per quella parola.
- Come scrittore  
 "voglio poter visualizzare una "term cloud" dei tweet"  
 per conoscere meglio il dizionario delle nuove generazioni che usano i social media e poterlo utilizzare nei miei romanzi.  
**CA:** La term cloud deve essere interessante, con visualizzazione interattiva (es. poter cliccare su una singola parola ed avere informazioni aggiuntive)
- Come forza dell'ordine  
 "voglio poter essere avvisato per e-mail in caso di situazione d'emergenza"  
 per intervenire tempestivamente.  
**CA:** Le parole da monitorare devono essere legate a situazioni di emergenza

In seguito, all'inizio del terzo sprint sono state aggiunte due Epiche ulteriori da parte del PO:

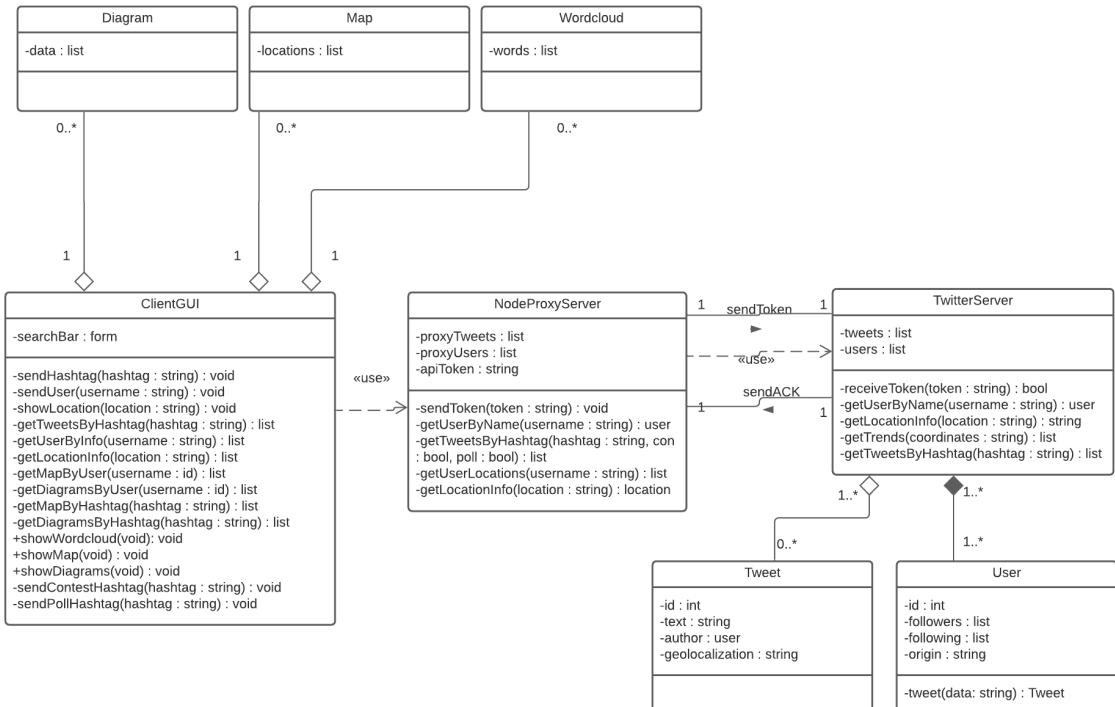
#### GESTIONE DI UN CONCORSO LETTERARIO:

- Come organizzatore
    - ”voglio bandire un concorso letterario della durata di due mesi gestito via Twitter”
    - CA:** Per bandire il concorso si deve usare l'hashtag #bandiscoconcorso[nomeconcorso]
    - Test:** Dato l'hashtag di un concorso, si controlla che i campi relativi al concorso identificato da quell'hashtag siano definiti.
  - Come organizzatore
    - ”voglio poter visualizzare i risultati della votazione in tempo reale.”
    - CA:** Per visualizzare i risultati bisogna inserire [nomeconcorso] nella barra di ricerca
  - Come scrittore speranzoso
    - ”voglio poter far votare il mio libro”
    - CA:** Per partecipare al concorso è necessario usare l'hashtag #partecipoconcorso[nomeconcorso]
  - Come lettore
    - ”voglio poter disporre di 10 voti per votare i miei libri preferiti”
    - CA:** Per votare per il concorso è necessario usare l'hashtag #votoconcorso[nomeconcorso]
- GIOCARE CON TWITTER:**
- Come organizzatore
    - ”voglio fare una domanda (trivia) e contare le risposte per vedere quanti giocatori ci sono”
    - CA:** I trivia devono essere identificati da un hashtag e contenere un poll
    - Test:** Dato l'hashtag di un trivia, si controlla che il tweet relativo al trivia contenga un poll.
  - Come giocatore
    - ”voglio rispondere ad una domanda”
    - CA:** La domanda deve essere contenuta in un poll di twitter

## 2.1 Diagramma dei casi d'uso



## 2.2 Diagramma delle classi



## 3 Primo Sprint

### 3.1 Preparazione

Il team è stato composto utilizzando Trello, riflettendo ognuno sui propri punti di forza e debolezze nell'ambito dello sviluppo software.

Prima di iniziare il progetto vero e proprio, tutti i membri del team hanno attivato un account per i servizi CAS di Dipartimento.

Per cominciare è stato utilizzato Taiga per creare un product backlog, aggiungendo e valutando le User Stories (riportate di seguito) e una breve wiki del nostro progetto.

In seguito abbiamo creato un Team privato su Mattermost dove ci siamo inviati comunicazioni e abbiamo discusso di alcuni dettagli inerenti il progetto.

Il terzo passaggio è stata la creazione da parte dello Scrum Master di una repository sull'istanza Gitlab di Ateneo, destinata ad ospitare i files del nostro progetto.

Infine, sono stati attivati gli account su Jenkins.

#### 3.1.1 Altri strumenti

Oltre agli strumenti CAS, ogni membro del team ha installato sul proprio IDE (nel nostro caso, Atom) il plugin atom-logger per monitorare la propria attività durante le sessioni di programmazione.

Gitlab, il logger e Jenkins si trovano riuniti anche in una comoda dashboard che permette di visualizzare alcuni dati inerenti la repository e la propria attività di modifica del codice, anche con l'ausilio di grafici. Oltre a quanto descritto in precedenza, abbiamo attivato un account developer su Twitter a testa e letto la documentazione di alcune API, sia appartenenti alla versione 1.1 che alla 2.0.

Come ultimo "strumento" per portare avanti il progetto, abbiamo richiesto un dominio web ai Tecnici e ci è stato assegnato il dominio <http://site202136.tw.cs.unibo.it>, che abbiamo provveduto ad attivare tramite Gocker (Docker sulle macchine di laboratorio).

Dal 16 novembre ci è stato fornito anche l'accesso a Sonarqube.

## 3.2 Sprint Backlog

Il progetto finale risulterà nella realizzazione di una web app, sviluppata con HTML, CSS e Javascript, con relativi framework. In questa prima fase abbiamo pensato di implementare un "proxy server" che permetesse l'utilizzo delle twitter API tramite Javascript, dato che non è possibile sfruttare la cross origin.

Abbiamo anche lavorato all'implementazione di alcuni endpoint per il backend della nostra applicazione. Dato che la maggior parte di questo sprint è stata dedicata a familiarizzare con gli strumenti CAS, le prime User Stories che abbiamo deciso di implementare non hanno avuto un coefficiente di difficoltà particolarmente elevato.

In particolare, avvalendoci della libreria npm-twitt e delle API v.1.1 di Twitter abbiamo implementato le seguenti funzionalità:

- User Timeline
  - Dato uno username, restituisce i tweet più recenti di quell'utente
- Text search
  - Data una stringa, restituisce i tweet più recenti che la contengono
- Hashtag search
  - Dato un hashtag, restituisce i tweet più recenti che la contengono

Filtraggio Utente	Filtraggio Hashtag	Filtraggio Testo
<p>Come fan di un artista, nella dashboard voglio poter visualizzare solo i tweet di un determinato utente... per vedere tutti i tweet del mio idolo.</p> <p>C.A. : Lo username deve essere del tipo "<u>MarioRossi</u>". - Particolare focalizzazione sulle informazioni di geolocalizzazione del dato utente.</p>	<p>Come social media manager, nella dashboard voglio poter visualizzare solo i tweet con un dato hashtag... per sapere cosa viene tweettato secondo i trend attuali.</p> <p>C.A. : L'hashtag deve essere del tipo "#xyz"</p>	<p>Come giornalista, nella dashboard voglio poter visualizzare I tweet contenenti una data stringa... per cercare frasi di impatto politico nei tweets.</p> <p>C.A. : la stringa deve essere di lunghezza massima di 500 caratteri.</p>
Priorità: 8/10 Difficoltà: 6/10	Priorità: 7/10 Difficoltà: 6/10	Priorità: 6/10 Difficoltà: 6/10

I valori di difficoltà indicati sono serviti unicamente al team per regolarsi nello svolgimento delle task, e sono stati stabiliti durante la prima riunione dello Sprint con una tecnica simile allo Scrum Poker: ogni membro ha proposto una stima di difficoltà al team, spiegando la sua posizione e, discutendone, abbiamo cercato di raggiungere una decisione unanime. In caso di indecisione la decisione ultima è stata affidata al Product Owner.

La priorità è stata stabilita dal Product Owner operativo.

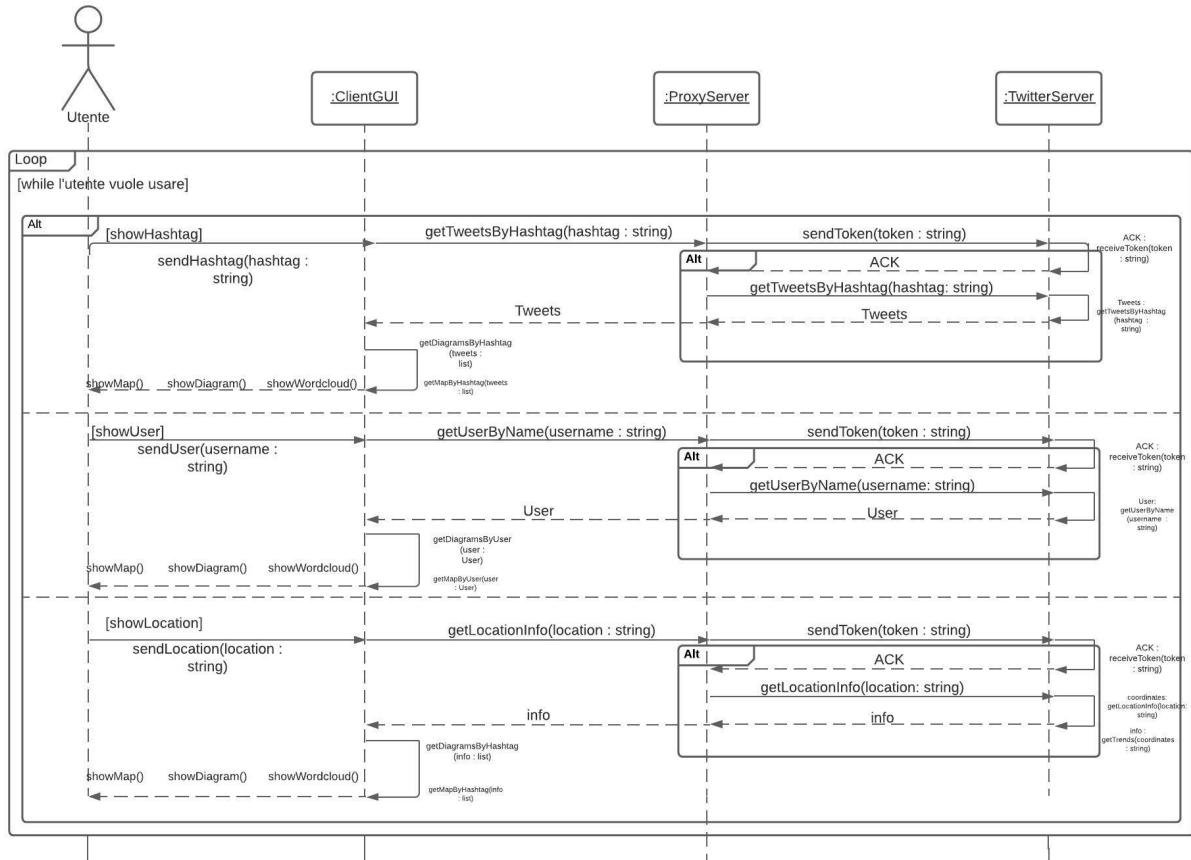
Il totale degli Story Points forniti da queste stories è 189 su 1127.

**Definition of done:** Permettere semplici interrogazioni all'utente senza artifici estetici

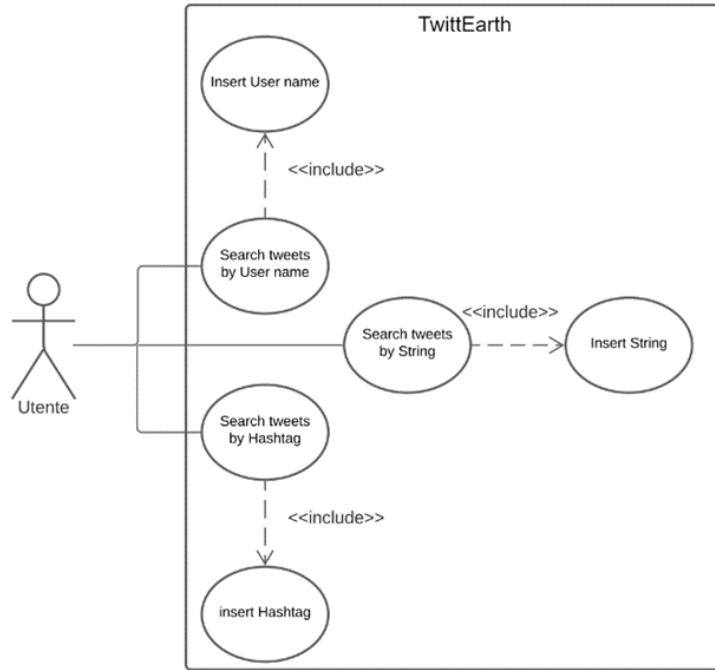
### 3.3 Progettazione

Quelli che seguono sono il diagramma di sequenza e gli use cases per le User Stories scelte durante questo sprint.

#### 3.3.1 Diagramma di sequenza



### 3.3.2 Use cases



## 3.4 Interfaccia Utente

Durante questo primo sprint abbiamo deciso di dare priorità, come funzionalità che avessero un valore per l'utente, quelle riguardanti alcune ricerche basilari; ci è sembrato infatti che potesse essere più interessante per un cliente visualizzare subito, durante la review, le prime funzioni che quest'app offre piuttosto che ricevere un servizio graficamente accattivante ma povero di contenuti.

La grafica non è stata quindi oggetto di particolari attenzioni durante il primo sprint, infatti il sito alla fine delle tre settimane aveva questo aspetto:

# TwittEarth

- Search User Timeline
- Search Tweets with given Text
- Search Tweets with given Hashtag

Inserisci il nome utente

## 3.5 Burndown

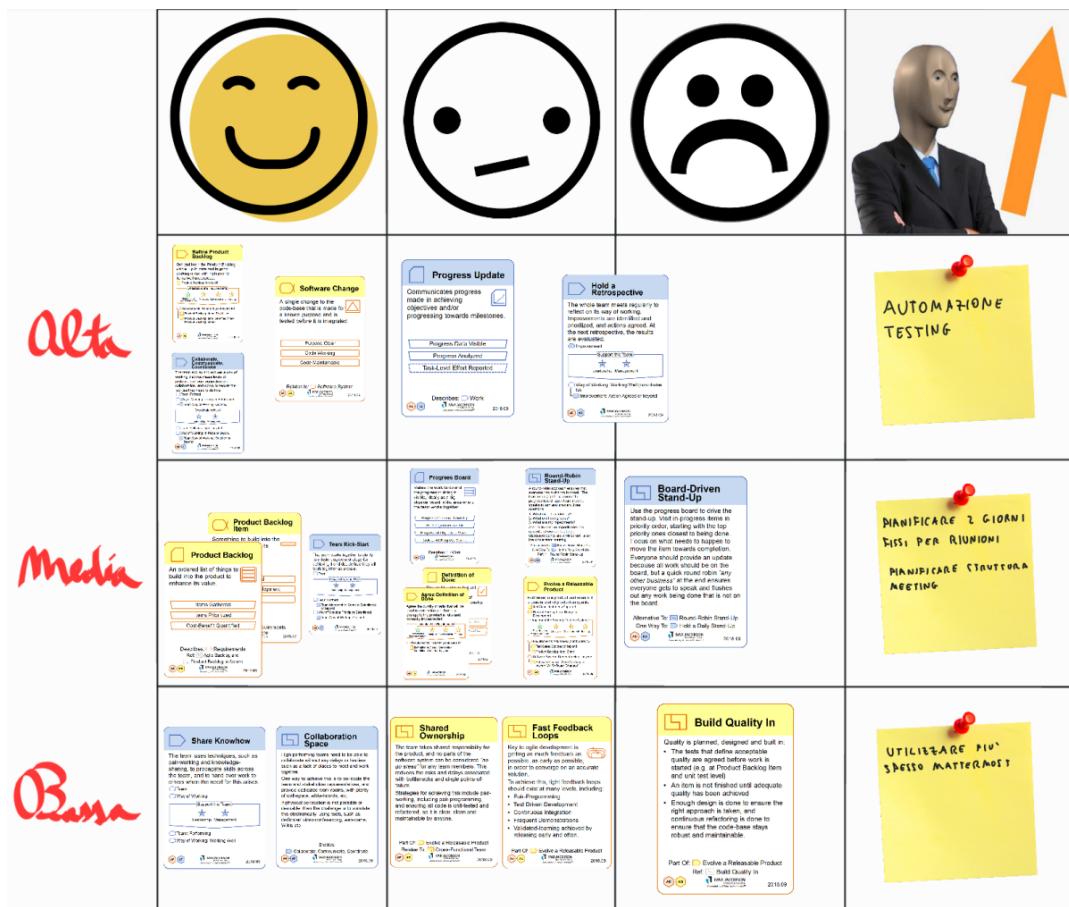
Sprint 1 11 Oct 2021 to 31 Oct 2021



## 3.6 Retrospectiva

Alla fine del primo sprint è stata svolta una riunione per confrontarci sui risultati ottenuti durante le tre settimane. Molto utili allo scopo sono state le carte Essence.

Di seguito il risultato (consultabile anche su Taiga):



Durante il primo sprint non siamo riusciti ad utilizzare Jenkins per CI/CD, quindi come obiettivo principale per lo Sprint 2 ci siamo posti l'inizializzazione di una toolchain per il deploy e il testing automatico.

La scrittura del codice è avvenuta per la maggior parte sfruttando la tecnica del pair programming, utilizzando il plugin Teletype su Atom e il relativo logger.

Al termine di ogni sessione di programmazione, veniva regolarmente eseguito un push su Gitlab da parte dell'host della sessione e il relativo pull da parte degli altri membri.

## 4 Sprint 2

### 4.1 Sprint Backlog

Diversamente dal primo sprint, in cui principalmente si è sviluppato in pair programming, è stato deciso di dividere le task delle US tra i membri del team in modo da sfruttarne al massimo le capacità individuali.

- Giuseppe Carrino (PO): FrontEnd, Supervisione Continuous Integration e BackEnd (API)
- Umberto Carlucci (SM): Testing
- Agata Cavigioli (DEV): Continuous Integration
- Andrea Loretti (DEV): FrontEnd (Mappa)
- Andrea Schinoppi (DEV): BackEnd (API)

Nel caso in cui un membro del team presentasse difficoltà nel completamento di una task tutto il team ha contribuito all'adempimento della suddetta.

Le user stories sviluppate nel secondo sprint sono le seguenti:

- Sentiment Analysis
  - data una ricerca per testo, restituisce il sentimento degli utenti in relazione alla stringa data
- Visualizzazione Tweet Geolocalizzati
  - data una ricerca, filtra unicamente i tweet geolocalizzati
- Visualizzazione Mappa
  - aggiunta di una mappa interattiva nell'interfaccia utente
- Filtraggio località
  - Data una località, restituisce i tweet più recenti geolocalizzati presso la stessa.

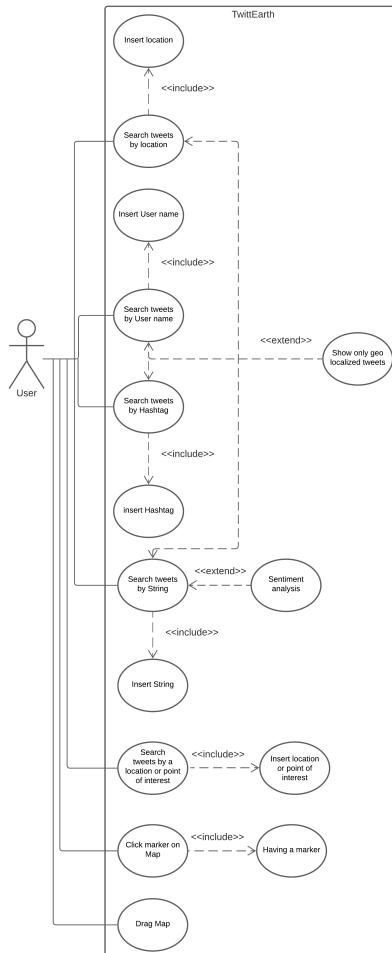
Sentiment Analysis	Visualizzazione Tweet Geolocalizzati	Visualizzazione Mappa	Filtraggio Località
Come responsabile di un museo, voglio poter avere informazioni sul gradimento di un dato luogo/avvenimento sulla base dei tweet che lo riguardano, per sapere se le visite guidate che organizzo sono gradite.  C.A. : <ul style="list-style-type: none"><li>- Gli eventi di cui analizzare il sentimento vengono chiariti in base agli hashtags usati o al testo cercato.</li><li>- Focalizzarsi sugli algoritmi di sentiment analysis usando come input il testo dei tweet.</li></ul>	Come padre apprensivo, voglio poter visualizzare i tweet geolocalizzati relativi ad un utente, un hashtag o un testo, per sapere da dove twitta mio figlio.  C.A. : <ul style="list-style-type: none"><li>- I tweet devono essere identificati sulla mappa</li><li>- Richiede US:Visulizzazione Mappa</li></ul>	Come utente poco istruito sulla geografia, voglio poter visualizzare una mappa dalla quale reperire informazioni interessanti sui tweet, per capire dove esattamente sono geolocalizzati i tweet che cerco.  C.A. : <ul style="list-style-type: none"><li>- Mappa creata grazie alle API di OpenStreetMap</li><li>- Possibilità di inserire dei punti d'interesse</li></ul>	Come sindaco di un piccolo comune, voglio poter cercare i tweet pubblicati in una data località o da utenti residenti in una data località, per sapere quali tweet vengono pubblicati dal mio paese.  C.A. : <ul style="list-style-type: none"><li>- Deve essere possibile inserire nella ricerca un'area geografica</li></ul>
Priorità: 9/10	Priorità: 8/10	Priorità: 8/10	Priorità: 7/10

Il totale degli Story Points forniti da queste stories e' 254 su 1127: il valore fornito all'utente è decisamente più alto del primo sprint, infatti avendo preso confidenza con gli strumenti di progettazione e implementazione ci siamo prefissati di lavorare più velocemente ed efficacemente. Inoltre avendo fatto molte riunioni e programmazione a coppie o in gruppo durante lo scorso ciclo di lavoro, il gruppo lavora già molto meglio e pensiamo di poterci permettere di osare di più a livello di carico di lavoro.

**Definition of done:** Fornire tutte le interrogazioni (anche complesse) all'utente e mostrare i risultati su una mappa.

## 4.2 Progettazione

Anche per questo sprint abbiamo creato il diagramma degli use cases per tutte le US implementate.



## 4.3 Passaggio alle API v2

Sia per una proposta del PO, sia per comodità, abbiamo deciso durante questo sprint di passare dalle API di Twitter v1.1 alle v2.

A tale scopo abbiamo utilizzato la libreria Javascript twitter-api-v2.

La differenza fondamentale che abbiamo subito notato tra le due versioni è che le API 1.1 restituiscono subito tutte le informazioni relative ai tweet, con risposte molto ampie e ricche di dettagli, anche superflui per i nostri scopi, mentre le API v2 restituiscono molti meno dati ma è possibile aggiungere delle 'expansions', ovvero scegliere a proprio piacimento i campi dei tweet che si desiderano ricevere.

Ciò si è rivelato molto utile perchè, oltre ad accorciare i tempi di attesa grazie alla nuova dimensione delle risposte, ci ha permesso di aver ben presente sin da subito la struttura del JSON che stavamo richiedendo a Twitter.

Il passaggio è stato quasi completo, abbiamo mantenuto le API v1.1 solo per le funzioni legate alla geolocalizzazione, dato che le v2 non fornivano gli stessi servizi.

## 4.4 API di OpenStreetMap

Durante lo svolgimento del progetto ci siamo avvalsi delle API di OpenStreetMap sia per la creazione della mappa, sia per poter ottenere le coordinate di luoghi che non fanno parte del database di Twitter. Infatti non tutti i paesi sono conosciuti da Twitter e ciò ci ha causato non pochi problemi in fase di sviluppo.

Un esempio è stata la ricerca dei Tweet geolocalizzati presso il comune San Giustino, in provincia di Perugia che, siccome sconosciuto da Twitter, veniva sostituito (per somiglianza) da San Paolo, in Brasile. Per ovviare a questo problema abbiamo utilizzato OpenStreetMap, che invece ha un database di luoghi e coordinate molto più ampio.

## 4.5 Interfaccia utente

L'interfaccia grafica è stata ampliata implementando tutte le funzionalità precedentemente descritte, tuttavia il design è provvisorio quindi verrà utilizzato solo per una demo e in fase di testing; l'obiettivo durante questa fase di lavoro è stato quello di rendere sufficientemente interattiva la pagina web, in modo che potesse avere già un valore interessante per l'utente finale.

### TwittEarth

- Search User Timeline
  - Search Tweets with given Text
  - Search Tweets with given Hashtag
  - Search Tweets of people living in a given Location
- 
- Show only geolocalized Tweets
- Show sentiment for word searching tweets

### Inserisci il nome utente

## 4.6 Burndown



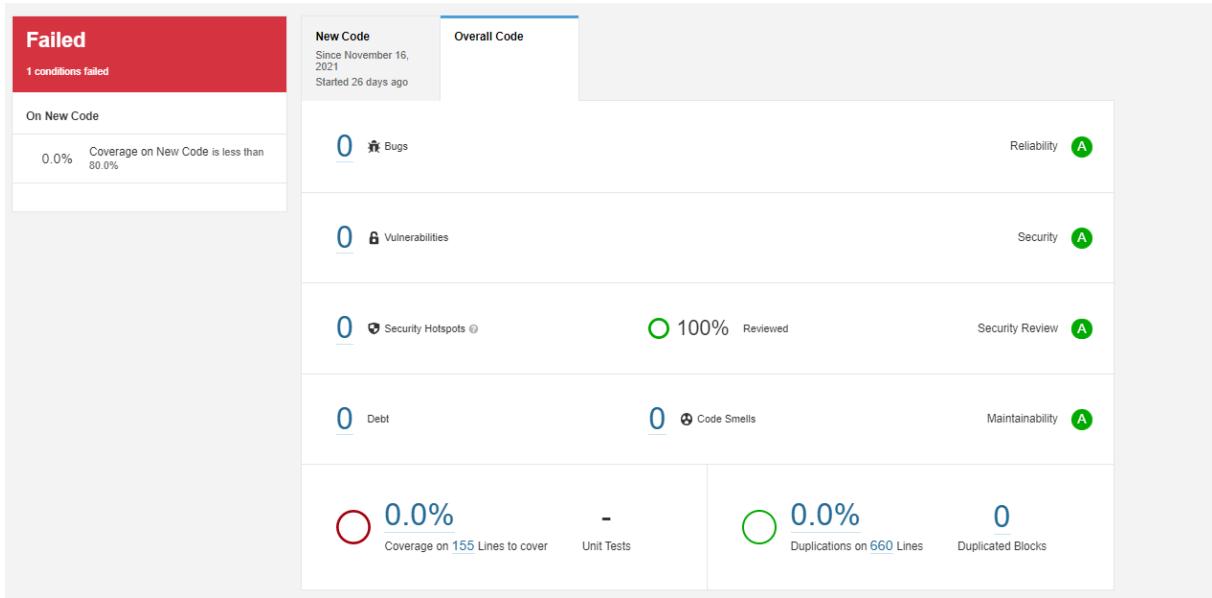
Come è possibile notare dal grafico, in questo sprint abbiamo cominciato a lavorare sin da subito, cercando di completare più US possibili nel corso della prima settimana.

Questa si è rivelata una scelta vincente dato che per le successive due settimane abbiamo potuto diminuire il carico di lavoro, concentrandoci sul refactoring e altre migliorie.

## 4.7 Sonarqube

Dal 16 Novembre ci è stato reso disponibile Sonarqube su CAS, quindi abbiamo subito iniziato ad analizzare il codice in modo automatico, lanciando il comando manualmente.

I risultati per questo sprint sono stati i seguenti:



Purtroppo, nonostante la presenza di due test, ci siamo resi conto che questi non vengono rilevati da Sonarqube se si utilizza Javascript, quindi il quality control fallisce sempre.

Tuttavia, siamo riusciti a non avere duplicazioni, bugs, problemi di sicurezza o code smells alla fine di questo sprint.

## 4.8 Jenkins

Durante questo sprint abbiamo realizzato una pipeline su Jenkins, installato in locale, in modo che si attivasse ad intervalli regolari qualora fosse stato effettuato un push su Gitlab. La pipeline si occupa del deploy sulle macchine di laboratorio, tramite ssh. L'utilizzo di questo strumento è stato piuttosto complicato all'inizio, soprattutto perché ci siamo resi conto che il fatto di attivare le pipeline su Aminsep (<https://aminsep.disi.unibo.it/jenkins/>) rendeva tutte le nostre informazioni visibili e accessibili; essendo necessarie operazioni piuttosto delicate, come il deployment tramite ssh, abbiamo deciso di installare Jenkins in locale. Lo script che gestisce la pipeline di Jenkins è il Jenkinsfile, consultabile nella repository di Gitlab.

Durante il terzo sprint ci siamo posti come obiettivo di aggiungere uno stage di controllo qualità tramite SonarQube alla pipeline.

## 4.9 Retrospettiva



Durante questa retrospettiva sono emerse alcune criticità presenti nel nostro processo di sviluppo. In particolare, alla fine dello sprint non avevamo ancora finito tutti i test delle User stories scelte, che sono diventati debito tecnico per lo sprint successivo. Inoltre ci siamo posti come obiettivo di utilizzare Mattermost, anche per ricevere le notifiche riguardanti i push su Gitlab e la pipeline di Jenkins, e di fornire una documentazione più approfondita del nostro codice.

Abbiamo notato che, durante il secondo sprint, abbiamo avuto la tendenza a lavorare e pensare in modo convenzionale, dedicandoci alla scrittura di codice in modo meccanico: mentre nel primo sprint la parte di progettazione è stata creativa e stimolante con molte idee differenti da parte di tutti i membri del team, nel secondo ci siamo occupati perlopiù di completare il più in fretta possibile le User Stories.

Ci siamo riproposti dunque di fare più attenzione al fatto di rendere personali le User Stories e proporre continuamente le nostre idee durante il terzo sprint.

Siamo stati molto soddisfatti, invece, per quanto riguarda il lavoro di gruppo e la comprensione delle pratiche di lavoro agili, come l'utilizzo del Product Backlog e le due riunioni a settimana, che sono diventate sempre più efficaci e meno dispendiose a livello di tempo.

## 5 Sprint 3

### 5.1 Divisione delle task

In linea con il processo di sviluppo adottato nel secondo sprint, è stato deciso di affidare le task delle US tra i membri del team in modo da sfruttarne al massimo le capacità individuali.

- Giuseppe Carrino (PO): FrontEnd, Supervisione Continuous Integration e BackEnd (API)
- Umberto Carlucci (SM): Testing
- Agata Cavigioli (DEV): Continuous Integration e FrontEnd
- Andrea Loretti (DEV): FrontEnd (Grafici e termcloud)
- Andrea Schinoppi (DEV): BackEnd (API)

Nel caso in cui un membro del team presentasse difficoltà nel completamento di una task tutto il team ha contribuito all'adempimento della suddetta task.

### 5.2 User Stories scelte

Le user stories sviluppate nel terzo sprint sono le seguenti:

- Creazione Dashboard
  - avere accesso a una dashboard più o meno dettagliata dove poter visualizzare i tweet
- Visualizzazione Grafici
  - visualizzare dei grafici di vario tipo che comparano informazioni sui tweet
- Term Cloud
  - visualizzare una "term cloud" dei tweet
- Aggiunta Opzioni di Ricerca
  - Come utente sensibile ricerche più personalizzate, ad esempio rimuovendo determinate parole, escludendo tweets con immagini o video, o ammettendo solo tweet di account verificati
- Stream di Tweet
  - voglio poter essere avvisato per e-mail in caso di situazione d'emergenza

Creazione dashboard	Visualizzazione Grafici	Term Cloud	Organizzazione concorso	Aggiunta Opzioni di Ricerca
Come utente di Twitter, voglio avere accesso a una dashboard più o meno dettagliata dove poter visualizzare i tweet raccolti e filtrarli, per utilizzare l'app in maniera più profonda e interessante.  C.A.: - Grafici a barre, a torta... - Informazioni rilevanti sono la presenza di geolocalizzazione, il numero di tweet per ogni hashtag ecc.	Come social media manager, voglio poter visualizzare dei grafici di vario tipo che comparano informazioni sui tweet, per conoscere meglio le statistiche e i target dei miei e altri contenuti.	Come scrittore, voglio poter visualizzare una "term cloud" dei tweet, per conoscere meglio il dizionario delle nuove generazioni che usano i social media e poterlo utilizzare nei miei romanzi.  C.A.: - La term cloud deve essere interessante, con visualizzazione interattiva (es. poter clickare su una singola parola ed avere informazioni aggiuntive)	Come organizzatore, voglio bandire un concorso letterario di racconti brevi della durata di due mesi gestito via twitter.	Come utente sensibile, voglio poter effettuare delle ricerche più personalizzate, ad esempio rimuovendo determinate parole, escludendo tweets con immagini o video, o ammettendo solo tweet di account verificati, per evitare di visualizzare contenuti che non gradisco.

Prima dell'inizio del terzo sprint, il professore ci ha fornito due Epic, ovvero due funzionalità aggiuntive da implementare composte da diverse User Stories, che seguono:

- Contest
  - bandire un concorso letterario di racconti brevi della durata di due mesi gestito via twitter
  - far votare il mio libro
  - disporre di 10 voti per votare i miei libri preferiti
  - visualizzare i risultati della votazione in tempo reale
- TriviaPoll
  - fare una domanda (trivia) e contare le risposte
  - rispondere ad una domanda e ottenere punti

Candidatura libro	Votazione libri	Visualizzazione voti	Risposta Trivia	Domanda Trivia	Stream di Tweet
Come scrittore speranzoso, voglio poter far votare il mio libro.	Come lettore, voglio poter disporre di 10 voti per votare i miei libri preferiti.	Come organizzatore, voglio poter visualizzare i risultati della votazione in tempo reale.	Come giocatore, voglio rispondere ad una domanda e ottenere punti.	Come organizzatore, voglio fare una domanda (trivia) e contare le risposte, per vedere quanti giocatori ci sono.	Come forza dell'ordine, voglio poter essere avvisato per e-mail in caso di situazione d'emergenza, per intervenire tempestivamente
Priorità: 9/10	Priorità: 8/10	Priorità: 8/10	Priorità: 9/10	Priorità: 9/10	Priorità: 9/10

Il totale degli Story Points forniti da queste stories e' 684 su 1127.

In quest'ultimo sprint la mole di lavoro è stata di molto maggiore rispetto agli altri due sprint: l'aggiunta delle due epiche infatti ha sbilanciato il numero di Story Points, che altrimenti sarebbe stato piuttosto simile ai due cicli precedenti. Tuttavia il fatto che durante il secondo sprint avessimo finito tutte le task molto prima della fine dell'iterazione ci ha spinti a darci un obiettivo più alto e a decidere di finire tutte le User Stories.

**Definition of done:** Creare un'interfaccia grafica accattivante, permettere la visualizzazione di grafici interessanti e creare una stream per le emergenze.

### 5.3 Gestione Trivia e concorso letterario

Per la gestione del concorso letterario ci siamo basati su un sistema ad hashtag: **su Twitter**

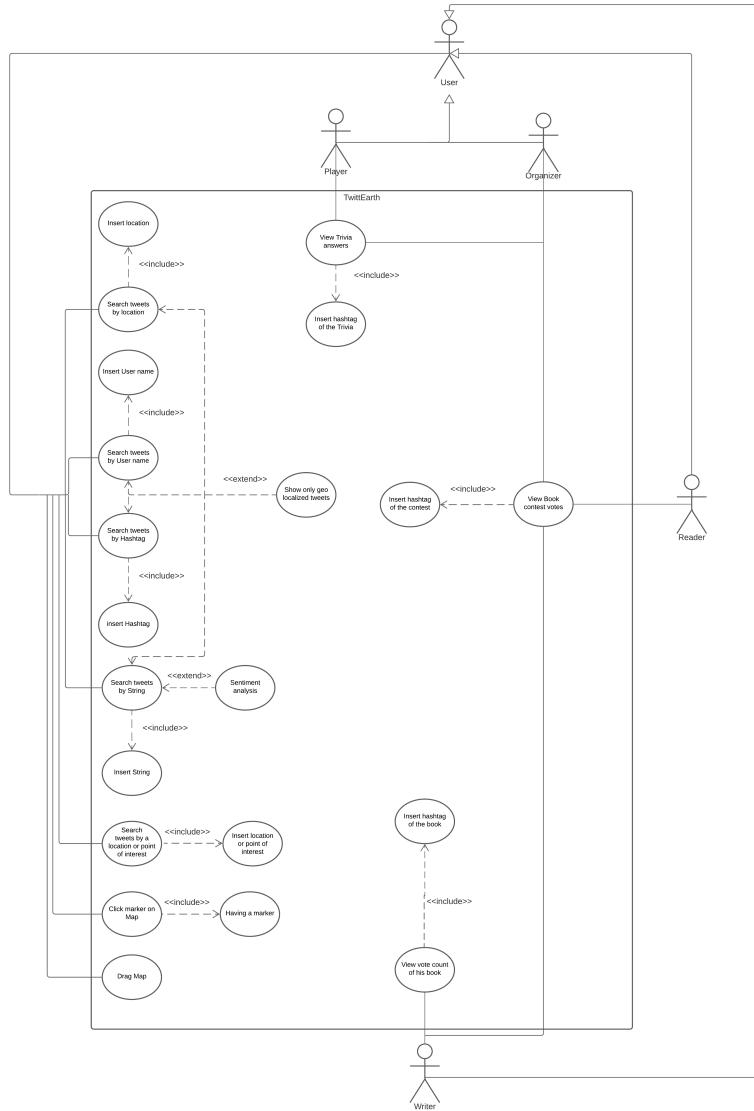
- il banditore del concorso può bandirlo usando **#bandiscoconcorso[nomeconcorso]**,
- si può partecipare usando **#partecipoconcorso[nomeconcorso] [nomeracconto]** ,
- votare con **#votoconcorso[nomeconcorso] [nomeracconto]**.
- Per vedere un contest letterario creato da noi è possibile digitare nell'apposito spazio *\_contest\_letterario*

Per quanto riguarda i Trivia, abbiamo chiesto al PO di poter sacrificare il meccanismo a punti in favore dei poll di Twitter che non permettono di vedere chi ha votato quale opzione ma solo di ottenere dati aggregati. Infatti ci è sembrato interessante lavorare su un servizio offerto direttamente da Twitter invece che implementare un meccanismo basato su hashtag che sarebbe risultato molto simile, nello sviluppo e nella visualizzazione, alla gestione del concorso letterario. Vista la risposta positiva, i Trivia sono dunque implementati usando i sondaggi di Twitter e la correttezza delle risposte è determinata da un tweet dell'organizzatore in cui rivela quelle corrette.

- ogni trivia è associato ad un hastag **#[nometrivia]**,
- i partecipanti rispondono tramite il sondaggio,
- le risposte vengono inviate alla fine del trivia tramite un tweet nella forma **#risposta[nometrivia]** [**risposta1, risposta2,...**]
- Per vedere un esempio di trivia creato da noi, è possibile digitare nell'apposito spazio *triviaigsw10*

### 5.3.1 Use Cases

È stato aggiornato il diagramma degli Use Case con tutte le nuove possibili interazioni.



### 5.4 Mappa dei bagni pubblici

Utilizzando la ricerca tramite hastag e inserendo l'hashtag **#WCigsw10** è possibile visualizzare sulla mappa alcuni bagni pubblici di Bologna e provincia con relativo commento e valutazione

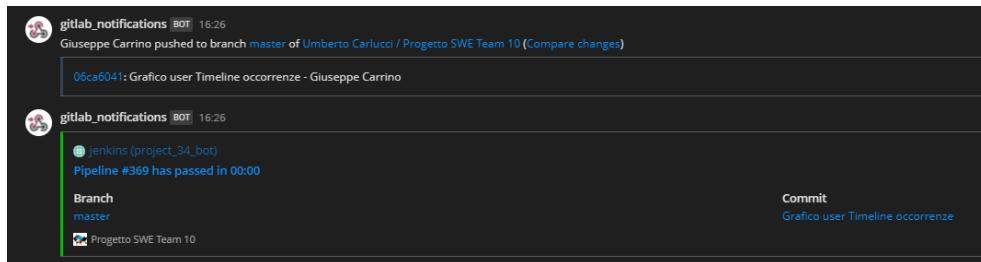
## 5.5 Grafica

L'interfaccia grafica è stata completata senza l'utilizzo di framework di modo da ottenere una semplicità e un minimalismo coerenti con la complessità del progetto.

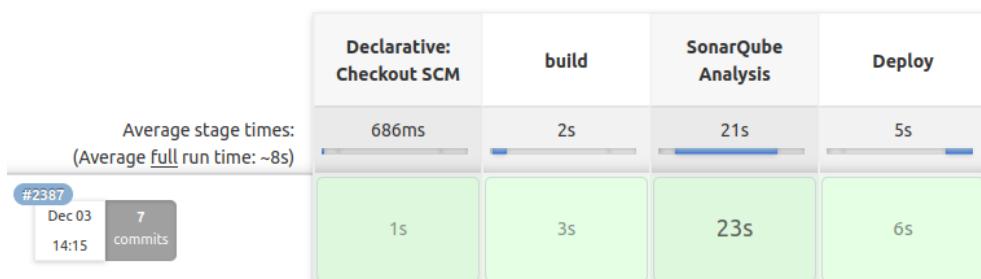


## 5.6 Toolchain

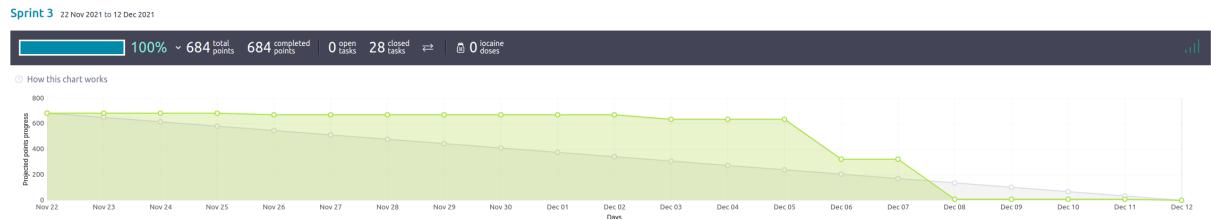
Durante questo sprint abbiamo ultimato tutte le toolchain tra i vari strumenti a disposizione. Abbiamo aggiunto Mattermost per ricevere una notifica quando qualcuno eseguiva un push e quando Jenkins terminava un deploy.



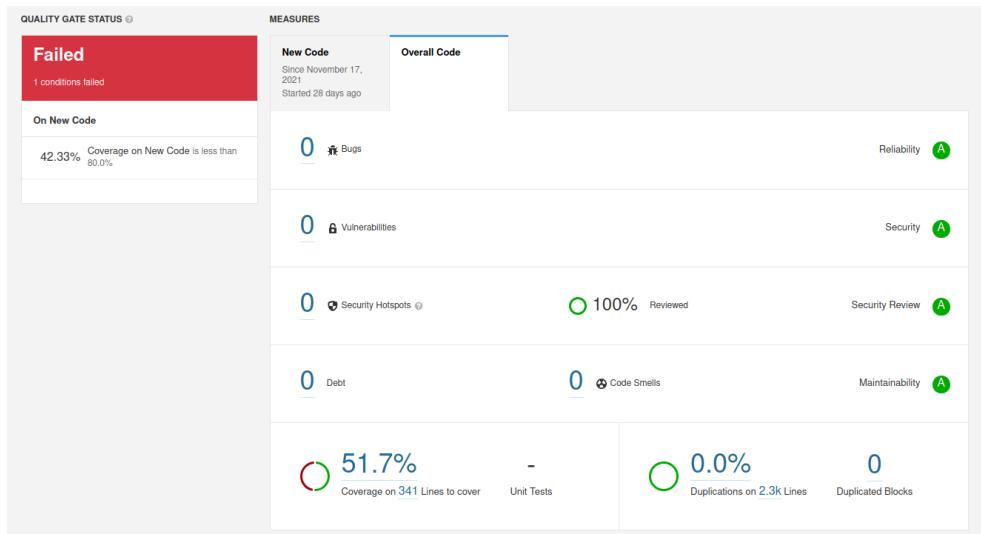
Infine abbiamo aggiunto alla pipeline di Jenikins l'analisi automatica di Sonarqube.



## 5.7 Burndown



## 5.8 Sonarqube



I test coprono solo il 51.7% del codice poiché un gran numero di linee di codice utilizza funzioni di librerie esterne (prime fra tutte, le API di Twitter) che non abbiamo ritenuto opportuno testare.  
Il testing da noi effettuato copre per intero le funzioni che abbiamo sviluppato per il nostro backend, in larga parte API.

## 5.9 Test eseguiti

```
PASS ./functions.test.js (22.001 s)
Testing GETs
  with given default hashtag
    ✓ Status code is 200 (54 ms)
    ✓ Every returned tweet contains #nasa (1 ms)
  with given default text
    ✓ Status code is 200
    ✓ Every returned tweet contains the word "nasa" (2 ms)
  with given default Username
    ✓ Status code is 200
    ✓ The only returned tweet has the expected text
  with given default location
    ✓ Status code is 200
    ✓ The location is the one identified by latitude: 44.49382035 and longitude: 11.3426327 (6 ms)
Testing other functions
  Testing sentiment analysis
    ✓ Status code is 200 for given positive word (1 ms)
    ✓ Status code is 200 for given negative word (1 ms)
    ✓ There is at least one positive word "good" (1 ms)
    ✓ There is at least one negative word "bad"
  Testing number of tweets
    ✓ Number of returned tweets is equal to the number of requested tweets (12)
  Testing tweets without media
    ✓ Status code is 200 (1 ms)
  Testing tweet excluding some words
    ✓ Status code is 200 (1 ms)
    ✓ Every tweet does not contain the word "space" (3 ms)
```

## 5.10 Retrospettiva

<i>Alta</i>				
<i>Média</i>				
<i>Bassa</i>				

Siamo molto soddisfatti di come è andato questo ultimo sprint: siamo riusciti a completare una grande quantità di Story Points e a risolvere le problematiche riscontrate nelle scorse retrospettive, in particolare sono stati migliorati i test e l'utilizzo degli strumenti CAS.

Abbiamo inoltre diviso in maniera efficace il carico di lavoro.

Alla fine di questa ultima retrospettiva abbiamo anche valutato il nostro intero percorso di sviluppo del software. Ci riteniamo molto soddisfatti sia del prodotto finale che, principalmente, della nostra crescita come Ingegneri del Software: le esercitazioni svolte durante le lezioni sono servite a fissare dei punti cardine dello sviluppo software che prima di questo corso tendevamo a sottovalutare e che, invece, si sono rivelati fondamentali per una scrittura del codice consapevole e professionale.

Lo sviluppo di test e l'utilizzo di strumenti di CI ci hanno sicuramente resi dei migliori sviluppatori capaci di comprendere le dinamiche che ci troveremo ad affrontare in un futuro ambiente lavorativo.

## 6 Conclusioni

### 6.1 Valutazione complessiva del progetto

Calcolo dello sforzo: 2.2 mesi e 5 persone = 11 mesi/persona  
Calcolo della durata complessiva: 9 settimane

Misurazione in LoC: 2,275

Valutando le controindicazioni della misurazione in linee di codice (la dipendenza dal linguaggio scelto, la penalizzazione di programmi concisi e refattorizzati e il fatto che non discriminò tra istruzioni con diversa complessità o potenza) abbiamo deciso di valutare il nostro codice secondo l'analisi Function Point. Ci sembra che questa misurazione sia più compatibile con la linea che abbiamo mantenuto durante questo corso e questo progetto ovvero l'attenzione al punto di vista dell'utente durante le varie fasi di sviluppo.

Abbiamo iniziato con l'individuazione dei seguenti elementi nella descrizione dei requisiti utente:

12 External Input: 6 scelte nella barra di ricerca, testo da inserire, 4 filtri, numero di tweet

- 5 External Output: tweet, mappa, wordcloud, grafico, stream
- 7 External Inquiry: 6 scelte nella barra di ricerca, stream
- 3 External Interface File: interfaccia con twitter, mappa, interfaccia grafica

Consultando la tabella fornita a lezione abbiamo dato degli indici di difficoltà ad ogni elemento funzionale:

- 12 Input "medi" x 4 = 48
- 3 Output "medi" x 5 = 15
- 2 Output "complessi" x 7 = 14
- 1 Inquiries "complesse" x 6 = 6
- 6 Inquiries "semplici" x 2 = 12
- 3 Interfacce "complesse" x 15 = 45

**Unadjusted FC = 140**

Definire il fattore di complessità tecnica dell'applicazione (TFC):

- F1 (Reliable backup and recovery) 3/5
- F2 (Data Communication) 5/5
- F6 (Online data entry) 5/5
- F7 (Operational ease) 4/5
- F11 (Reusability) 3/5

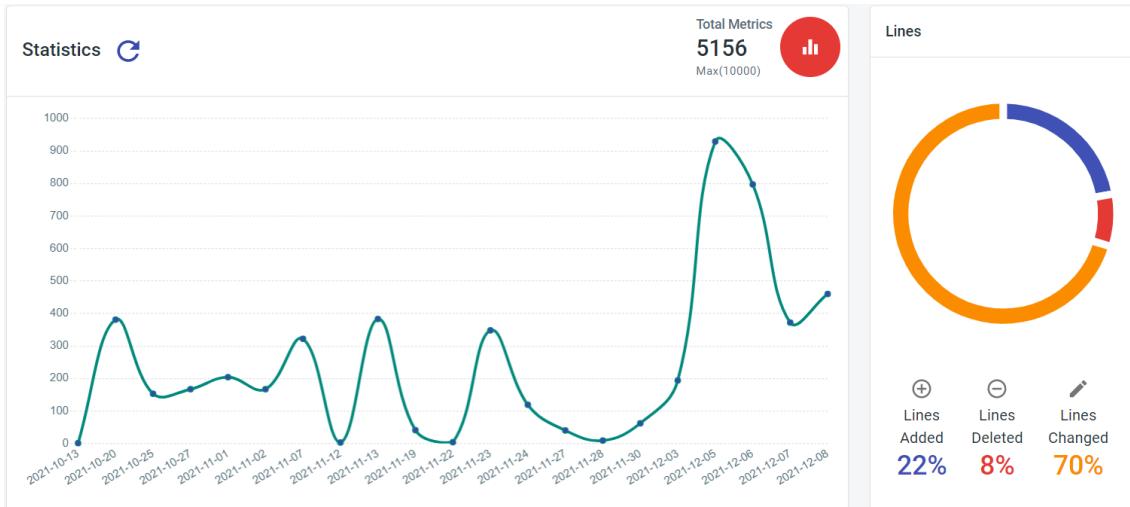
$$TFC = 0.65 + 0.01 * \sum_{i=1}^4 Fi = 0.65 + 0.01 * 20 = 0.85$$

$$FP = UFC * TFC = 119$$

## 6.2 Logger

Seguono i grafici del logger di ogni componente del gruppo:

### 6.2.1 Carrino



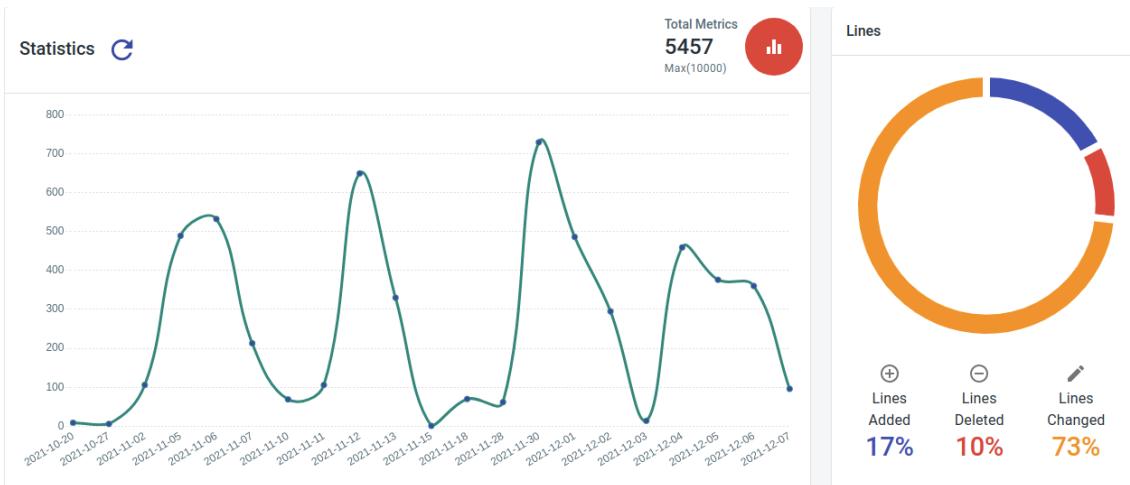
### 6.2.2 Carlucci



### 6.2.3 Cavigioli

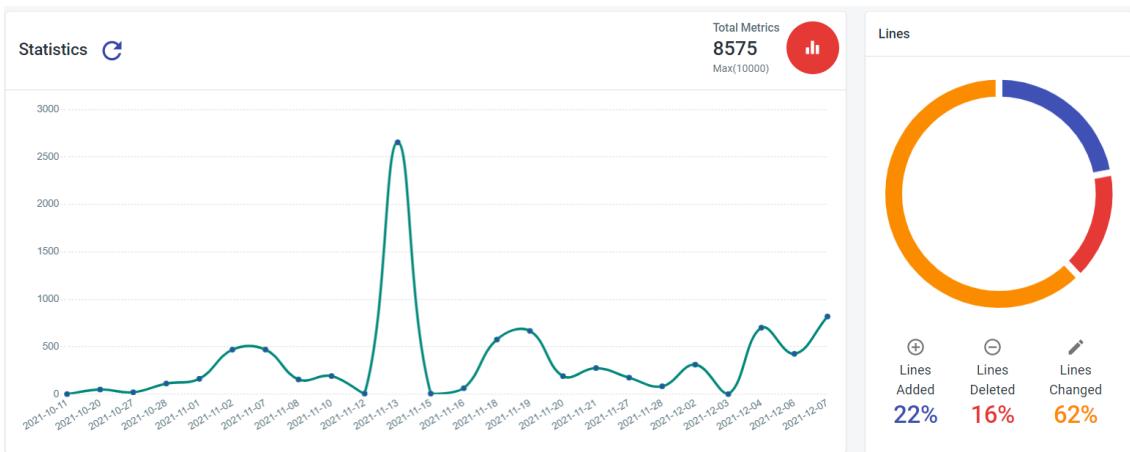


### 6.2.4 Loretta

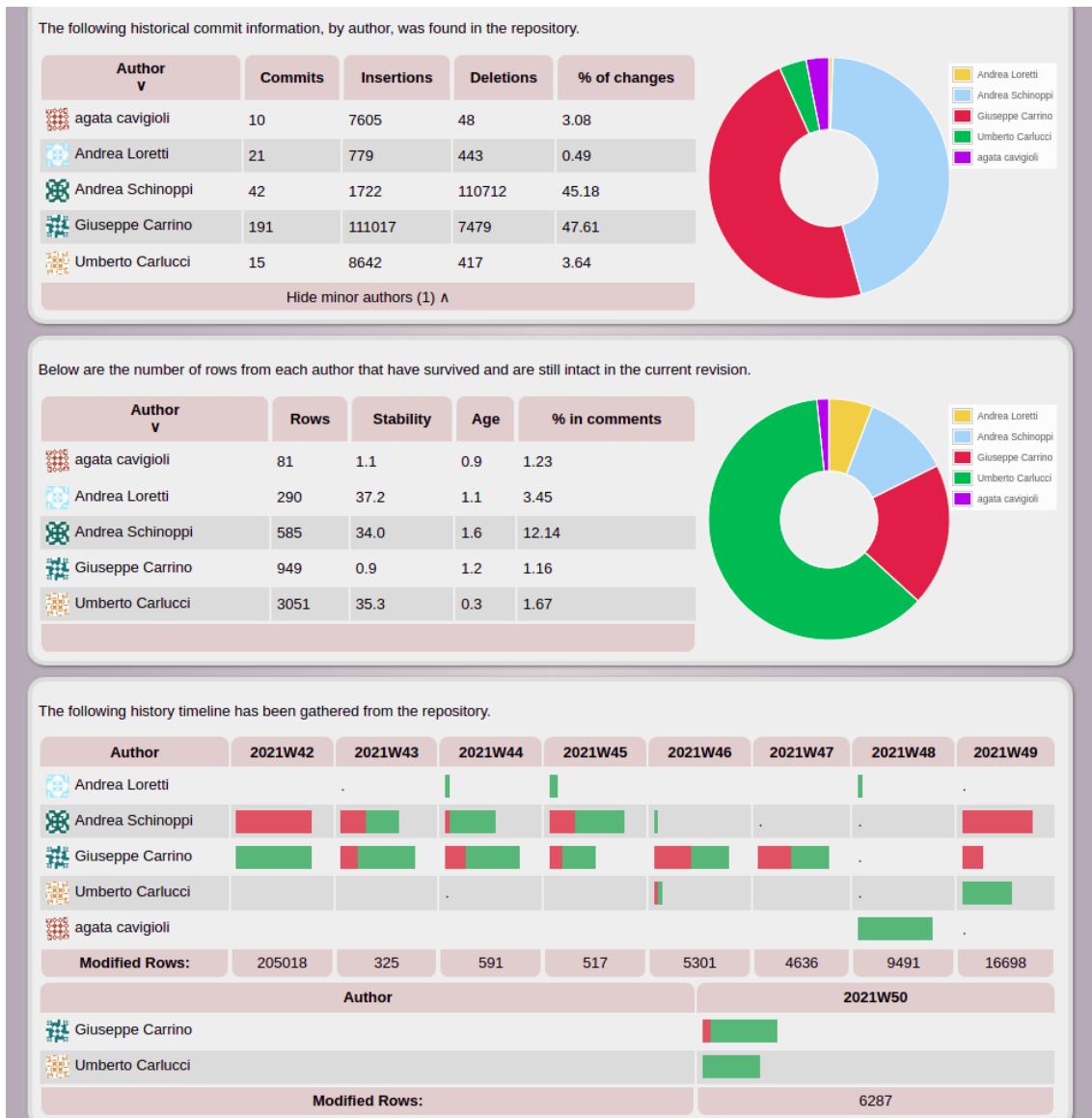


### 6.2.5 Schinoppi

Nota: l'alto numero di metrics maturate in data 13 novembre è dovuto ad un bug del logger, un numero più realistico di metrics per tale data si aggira intorno a 1000.



## 6.3 Gitinspector



## 6.4 Artefatti e link utili

Sito: <http://site202136.tw.cs.unibo.it>

Gitlab: <https://aminsep.disi.unibo.it/gitlab/UmbertoCarlucci/progetto-swe-team-10>

Taiga: <https://aminsep.disi.unibo.it/project/admin-progetto-2021-team-10>

Sonarqube: [https://aminsep.disi.unibo.it/sonarqube/dashboard?id=10\\_TwittEarth](https://aminsep.disi.unibo.it/sonarqube/dashboard?id=10_TwittEarth)

Video di presentazione: <https://drive.google.com/drive/folders/10Xech449bb2shTI7q7M1pruKJKDW5HFw?usp=sharing>

I professori P. Ciancarini e M. Missiroli sono stati invitati al progetto su Gitlab: una volta accettato l'invito potranno accedere ai file sulla repository.

I file, oltre ad essere su Gitlab, sono stati deployati nella directory /web/site202136/html/ delle macchine di laboratorio ed è possibile gestire il server grazie al Gocker offerto dal dipartimento.