

Corso di ingegneria del software  
**Report finale di progetto**  
**Twitter Tracker**

Stefano Colamonaco, Andrea Corradetti, Alessio Di Pasquale,  
Leonardo Naldi, Filip Radovic

gennaio 2022

## Indice

<b>1</b>	<b>Descrizione del prodotto</b>	<b>3</b>
1.1	Scope e backlog di prodotto . . . . .	3
1.1.1	Scope . . . . .	3
1.1.2	Funzionalità del prodotto finale . . . . .	3
1.1.3	Backlog . . . . .	5
1.2	Diagramma dei casi d'uso . . . . .	6
1.3	Diagramma delle classi . . . . .	7
<b>2</b>	<b>Analisi degli sprint</b>	<b>7</b>
2.1	Primo sprint . . . . .	7
2.1.1	Sprint Goal . . . . .	7
2.1.2	Sprint Backlog . . . . .	8
2.1.3	Sprint Definition of Done . . . . .	8
2.1.4	Sprint Burndown . . . . .	8
2.1.5	Sprint Retrospective . . . . .	9
2.2	Secondo sprint . . . . .	10
2.2.1	Sprint Goal . . . . .	10
2.2.2	Sprint Backlog . . . . .	10
2.2.3	Sprint Definition of Done . . . . .	11
2.2.4	Sprint Burndown . . . . .	11
2.2.5	Sprint Retrospective . . . . .	12
2.3	Terzo sprint . . . . .	13
2.3.1	Sprint Goal . . . . .	13
2.3.2	Sprint Backlog . . . . .	13
2.3.3	Sprint Definition of Done . . . . .	14
2.3.4	Sprint Burndown . . . . .	14
2.3.5	Sprint Retrospective . . . . .	15
2.4	Quarto sprint . . . . .	16
2.4.1	Sprint Goal . . . . .	16
2.4.2	Sprint Backlog . . . . .	16
2.4.3	Sprint Definition of Done . . . . .	16

2.4.4	Sprint Burndown . . . . .	16
2.4.5	Sprint Retrospective . . . . .	16
<b>3</b>	<b>Descrizione del processo di lavoro</b>	<b>16</b>
3.1	Autodescrizione del team . . . . .	17
3.2	Risultato dello Scrumble iniziale . . . . .	18
3.3	Sintesi dei dati dalle applicazioni dell'ambiente CAS . . . . .	19
3.3.1	Dati del logger . . . . .	19
3.3.2	Dati da Gitinspector . . . . .	20
3.3.3	Dati da Gitlab . . . . .	21
3.3.4	Analisi da SonarQube . . . . .	22
3.3.5	Dati relativi al testing e a Jenkins . . . . .	23
3.4	Retrospeztiva finale . . . . .	24
3.5	Diagramma di deployment del prodotto . . . . .	25
<b>4</b>	<b>Artefatti</b>	<b>26</b>

# 1 Descrizione del prodotto

## 1.1 Scope e backlog di prodotto

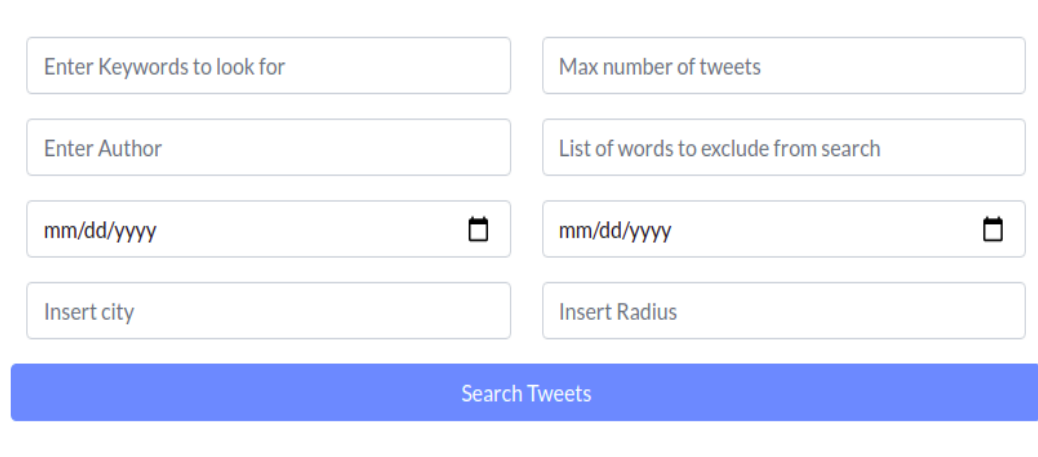
### 1.1.1 Scope

Lo scopo principale di questo progetto è la creazione di uno strumento di raccolta e analisi dei tweet di Twitter capace di manipolare i dati ottenuti, mostrandoli poi all'utente attraverso particolari integrazioni quali mappe, grafici e term clouds.

### 1.1.2 Funzionalità del prodotto finale

Il prodotto Twitter Tracker realizzato dal team 4 si compone di quattro pagine principali, ognuna con funzionalità specifiche:

- Home:  
Contiene tutte le funzionalità relative a ricerca e filtraggio dei tweet. In particolare è possibile cercare un numero fissato di tweet filtrandoli per keyword, hashtags, utente creatore, parole presenti, range di data e geolocalizzazione (con relativo raggio di ricerca).

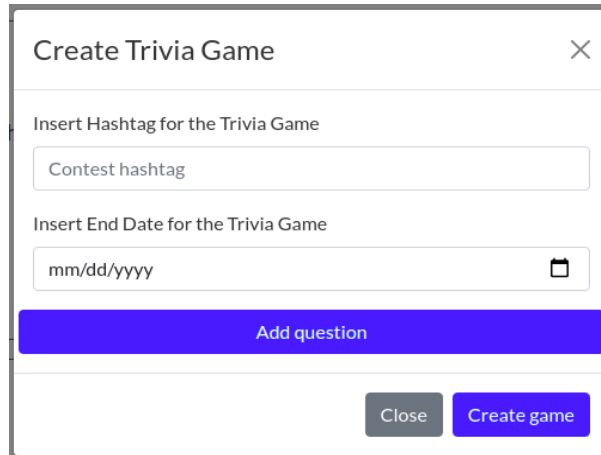


The screenshot displays a search interface for Twitter Tracker. It features a grid of input fields: 'Enter Keywords to look for', 'Max number of tweets', 'Enter Author', 'List of words to exclude from search', two date pickers labeled 'mm/dd/yyyy', 'Insert city', and 'Insert Radius'. A prominent blue button labeled 'Search Tweets' is positioned at the bottom of the form.

Su tutti i tweet risultanti dalla ricerca è poi possibile effettuare una serie di operazioni, quali: analisi dei sentimenti (singola o di gruppo), costruzione di una term cloud (singola o di gruppo), ricerca di retweet e retweeters per il singolo tweet in esame.

- Contest handler:

Contiene i modal dedicati alla creazioni dei tre diversi tipi di stream supportati da questo prodotto, vale a dire creazione di contest letterari, giochi in stile trivia e custom stream con keyword e utente impostabili.



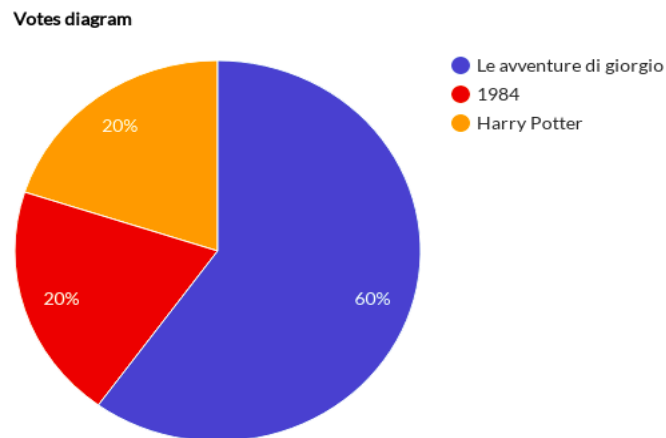
The image shows a modal window titled "Create Trivia Game" with a close button (X) in the top right corner. Inside the modal, there are two input sections. The first section is labeled "Insert Hashtag for the Trivia Game" and contains a text input field with the placeholder "Contest hashtag". The second section is labeled "Insert End Date for the Trivia Game" and contains a date input field with the placeholder "mm/dd/yyyy" and a calendar icon. Below these sections is a large blue button labeled "Add question". At the bottom right of the modal are two buttons: a grey "Close" button and a blue "Create game" button.

- Contest viewer:

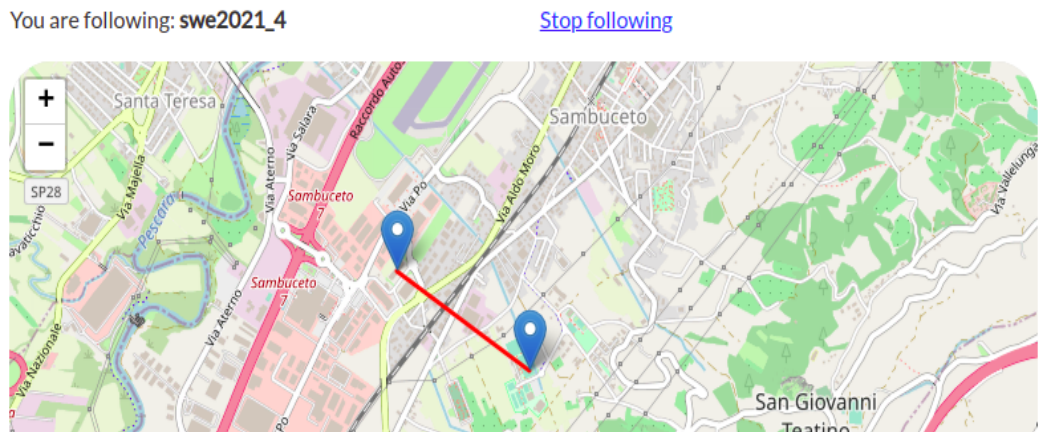
In questa pagina è possibile osservare in tempo reale i dati relativi alle tre categorie di stream costruite nella pagina precedente.

Grazie alla costruzione via socket è possibile ricevere e visualizzare nuovi dati in tempo reale senza dover effettuare il refresh della pagina.

Le stream relative a contest letterari e giochi di trivia sono corredate da grafici che aiutano nell'analisi dei dati.



- User tracking:  
In questa pagina è possibile selezionare un utente (inserendo correttamente il suo username) di cui visualizzare i tweet in tempo reale. Inoltre tutti i tweet geolocalizzati vengono inseriti in una mappa e collegati da linee rosse che ne descrivono gli spostamenti tra uno e l'altro cronologicamente.



### 1.1.3 Backlog

Di seguito sono riportati i titoli delle US sviluppate. Ogni user story viene poi descritta in dettaglio (descrizione + task associate) all'interno dello sprint in cui è stata realizzata.

#### Primo sprint

- Ricerca di tweets attraverso una parola chiave
- All'unica user story implementata in questo sprint si affiancano una serie di storyless tasks relative a costruzione del team, scelta dell'ambiente e dei dettagli di sviluppo.

#### Secondo sprint

- Ricerca geolocalizzata dei tweets
- Costruzione di term clouds interagibili
- Nuovi filtri di ricerca per i tweets
- Analisi del sentimento sui tweets trovati
- Analisi di dati relativi ai retweets

#### Terzo sprint

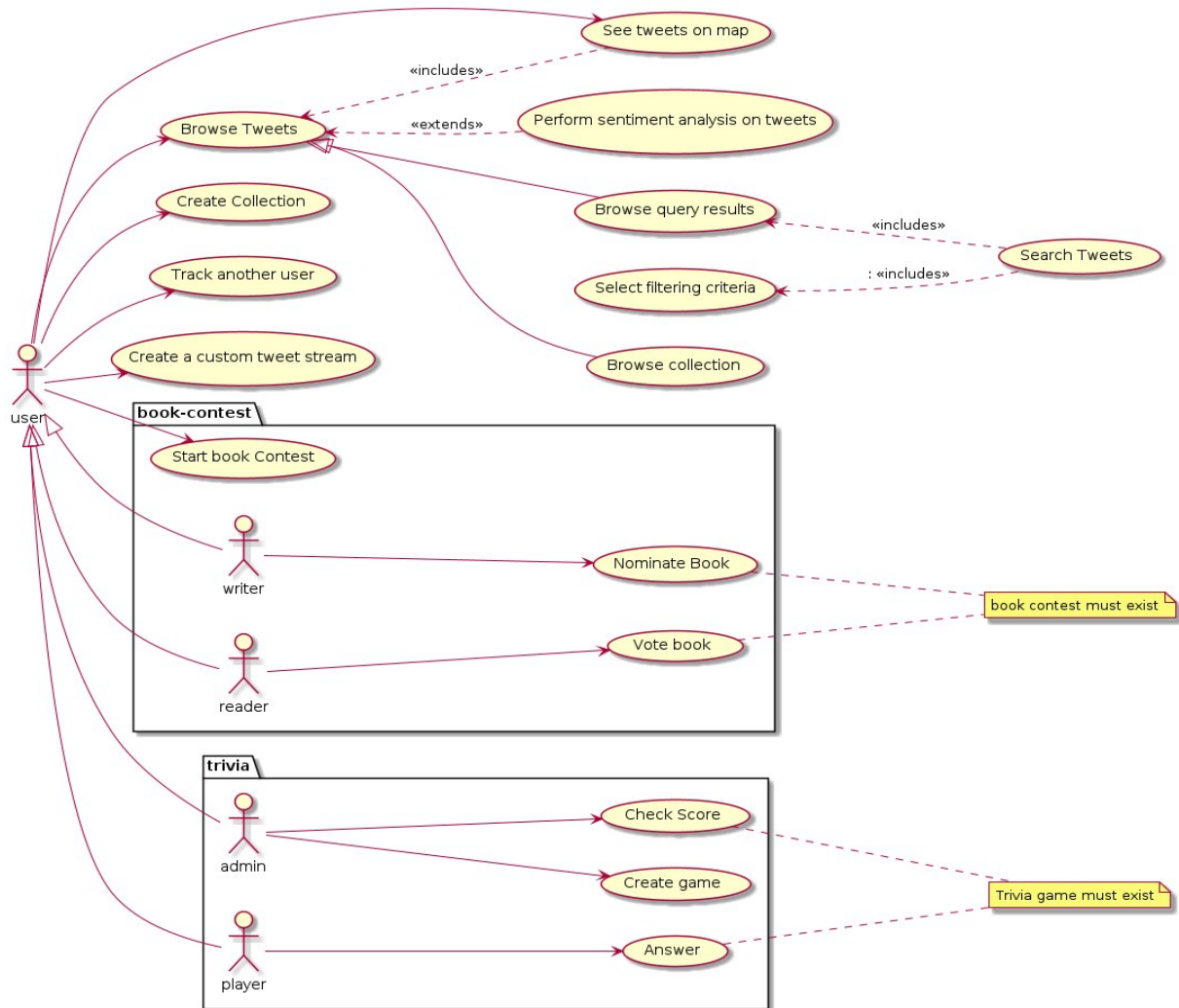
- Implementazione di stream relativi a contest letterari
- Implementazione di stream relativi a giochi di trivia
- Implementazione di stream custom
- Implementazione della funzione di user tracking

- Generazione di informazioni sulle performance di ricerca

## Quarto sprint

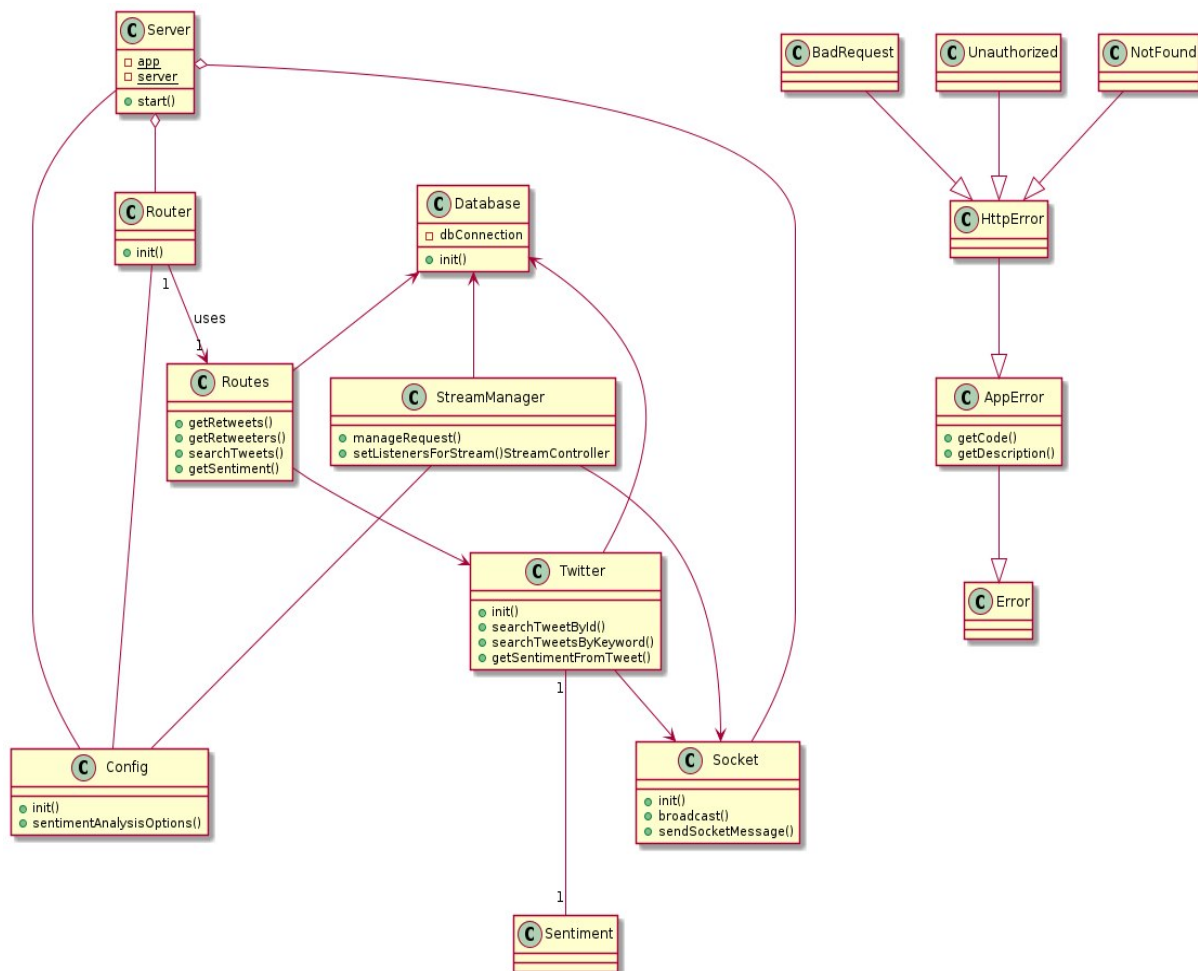
- Dedicato al miglioramento generale e al completamento della documentazione del progetto. A questo scopo, non avendo ulteriori feature da implementare, tutto è stato diviso in storyless tasks.

### 1.2 Diagramma dei casi d'uso



## 1.3 Diagramma delle classi

Il diagramma è stato autogenerato utilizzando un apposito programma: PlantUML.



## 2 Analisi degli sprint

### 2.1 Primo sprint

Periodo di riferimento: 11/10/21 → 31/10/21

#### 2.1.1 Sprint Goal

L'obiettivo principale del primo sprint è stato quello di organizzare il gruppo e gettare le basi per l'inizio dello sviluppo. In particolare le priorità erano: la divisione dei ruoli, una definizione precisa del tipo di applicazione da realizzare, la scelta dello stack, prendere confidenza con l'ambiente CAS, richiedere le credenziali per le API di Twitter, studiare la documentazione di queste ultime, trovare un IDE appropriato e creare delle user stories. Una volta completate tutte le priorità appena elencate, saremmo dovuti essere in grado di completare almeno una delle user stories.

### 2.1.2 Sprint Backlog

Come previsto nello sprint goal, è stata implementata una sola user story:

- As a user, I want to search tweets to know what people think on some subject

**Stima di difficoltà:** 5 (story points)    **Priorità:** Alta

**Tasks:** Search by keyword, search by hashtag, search by author.

- Storyless tasks: Setup dev environments, chose an appropriate stack, acquire credentials for Twitter APIs, play Scrumble.

La scelta è ricaduta su quest'ultima poichè considerata una delle più semplici e allo stesso tempo la più importante (come evidente da indici di priorità e difficoltà), in quanto necessitava di inizializzare un server che si occupasse delle richieste e fosse in grado di interfacciarsi con le API di Twitter.

### 2.1.3 Sprint Definition of Done

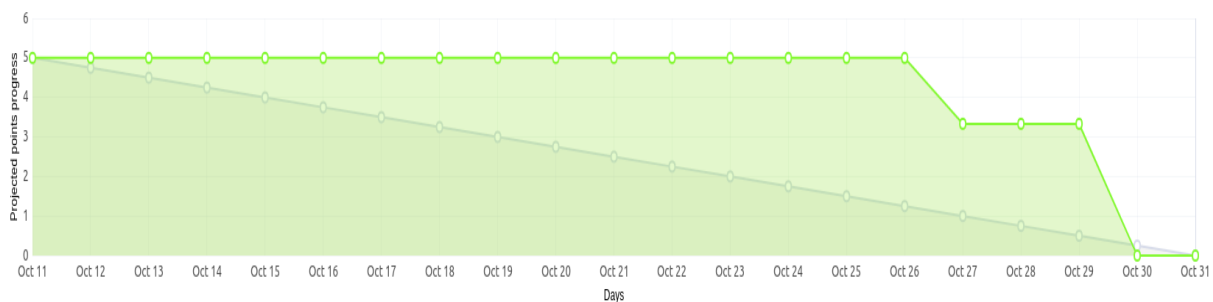
La Definition of Done nel primo sprint 'e composta dai seguenti requisiti:

- Avere un ambiente funzionante e integrato in cui lavorare tutti insieme.
- Codice funzionante, privo di errori e warnigs.
- Semplice UI che permetta l'interazione da parte dell'utente

La DOD per questo sprint risulta particolarmente lasca, questo è dovuto alle incertezze che ci sono state durante la fase di planning, in particolare nei tempi necessari per esplorare e prendere dimestichezza con le tecnologie da utilizzare.

### 2.1.4 Sprint Burndown

Guardando il diagramma di burndown è evidente la presenza di un grande periodo di planning iniziale, ma la storia in programma per lo sprint è stata completata nei tempi previsti.

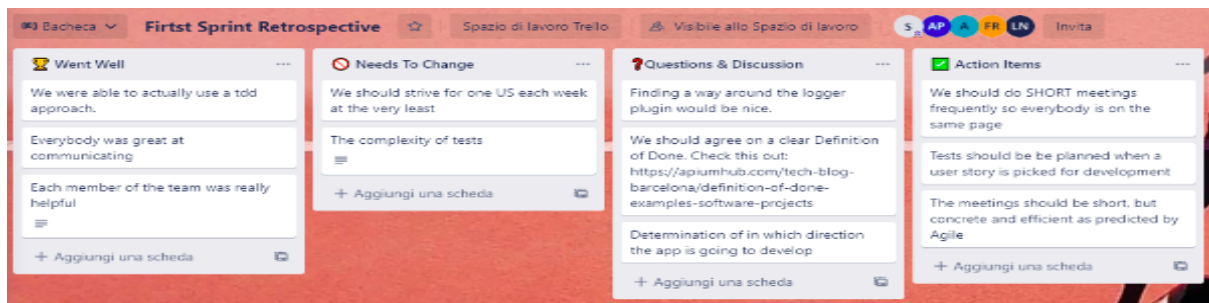




## 2.1.5 Sprint Retrospective

Alla fine del primo sprint sono state fatte due retrospettive, utilizzando due metodologie differenti e in questo ordine:

- Utilizzando un metodo proposto da Atlassian, disponibile al seguente link: [Manuale](#)



- Attraverso le carte Essence sviluppate da Ivar Jacobson (sarà il metodo mantenuto per le retrospettive successive). Per via delle carte, lo schema risultante ha dimensioni piuttosto elevate e può risultare difficile leggerne il contenuto all'interno di questa relazione. Ecco però un link alla versione originale: [Retrospettiva primo sprint](#)

High	relevance	      		
Medium	relevance	 	  	
Low	relevance			

Da entrambe le versioni si evince un risultato simile, e si può notare che la parte relativa agli improvements non è stata replicata nella versione con essence poichè sarebbe risultata ripetitiva. Il risultato comune ci dice che il team è riuscito a raggiungere molto velocemente un livello di

affiatamento e lavoro di squadra ottimali, restano però delle lacune in alcune pratiche a noi nuove, quale la definizione di una DOD, e l'integrazione automatizzata di buona parte delle applicazioni CAS.

## 2.2 Secondo sprint

Periodo di riferimento: 01/11/21 → 21/11/21

### 2.2.1 Sprint Goal

Con il secondo sprint il team 'e entrato nel vivo dello sviluppo dell'applicazione. Gli obiettivi proposti sono stati i seguenti:

- Avere un sistema di testing automatico ed efficiente, utilizzando un approccio TDD
- Avere un sistema di deployment automatico
- Completare del tutto la parte del progetto relativa a ricerca e geolocalizzazione dei tweets

### 2.2.2 Sprint Backlog

- As a user, I want to search tweets from a geographic location to see what's a happening there

**Stima di difficoltà:** 16 (story points) **Priorità:** Alta

**Tasks:** Select area on map to search tweets within, search by city or region name.

- As a user, i want to individually search for the words that make up a tweet on a search engine, so that I can better understand their meaning

**Stima di difficoltà:** 31 (story points) **Priorità:** Media

**Tasks:** Interface to select words, find search engine API.

- As a user, I want to display tweets from a specific timeframe to see what's people opinion during the elections

**Stima di difficoltà:** 4.5 (story points) **Priorità:** Media

**Tasks:** Add "to:" and "from:" to search tags, If "to:" is not specified, we should fetch tweets continuously

- As a user, I want to exclude tweets from a search so I can be very accurate

**Stima di difficoltà:** 8 (story points) **Priorità:** Media

**Tasks:** Pick out exclude-tags from query.

- As a user I want an estimate of the feelings expressed in a tweet, so that I can better understand their meaning

**Stima di difficoltà:** 37 (story points) **Priorità:** Alta

**Tasks:** Creation of a simple user interface, study of the "sentiment" library.

- As a user, I want to see a list of people that have retweeted a post so I can see who shares similar interests

**Stima di difficoltà:** 10 (story points)    **Priorità:** Bassa

**Tasks:** Graphinc interface for the list.

- As a user I want to see the information contained in a tweet in a wordcloud, so that I can use it in a presentation

**Stima di difficoltà:** 21 (story points)    **Priorità:** Alta

**Tasks:** Search for wordcloud API, simple interface.

- Storyless tasks: Design a page-based layout

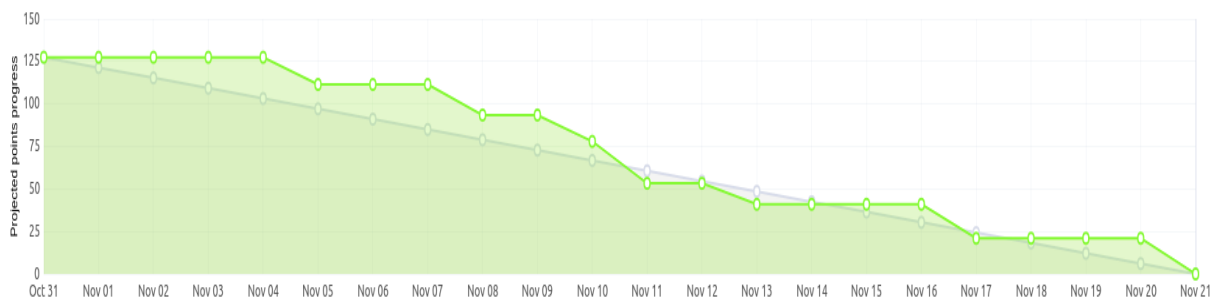
### 2.2.3 Sprint Definition of Done

La Definition of Done nel secondo sprint 'e composta dai seguenti requisiti:

- Implementazione di un test per ogni feature;
- Codice funzionante, privo di errori e warning;
- Deployment su server remoto funzionante e automatizzato.

### 2.2.4 Sprint Burndown












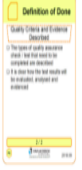



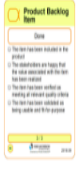

Durante il secondo sprint il carico di lavoro 'e stato distribuito in maniera più uniforme e lineare rispetto al primo, come si può notare dall'andamento notevolmente migliorato del grafico.



Anche in questo caso tutte le user stories proposte sono state completate nei tempi previsti.

## 2.2.5 Sprint Retrospective

Dalla posizione delle carte è evidente che il team e il progetto stesso hanno avuto un notevole miglioramento nel corso del secondo sprint. Per raggiungere un workflow del tutto corretto restava solo da ottenere una maggiore descrittività nelle DOD e una pianificazione settimanale migliore. Anche in questo caso è opportuno inserire un link alla retrospettiva originale: [Retrospettiva secondo sprint](#)

				
High	relevance	       		
Medium	relevance	  		
Low	relevance			

Improvements
Use Twitter api 2.0
Improve responsiveness of the UI
Improve front-end tests

## 2.3 Terzo sprint

Periodo di riferimento: 22/11/21 → 12/12/21

### 2.3.1 Sprint Goal

Nel terzo sprint abbiamo dovuto integrare nuove user stories fornite dal PO. Purtroppo proprio all'inizio di questo sprint abbiamo avuto qualche difficoltà nell'effettuare un refactoring completo del codice che consentisse anche il porting alle API di Twitter nella versione 2.0. Da questo ne consegue che gli obiettivi per questo sprint sono stati i seguenti:

- Terminare il code refactoring
- Integrare le nuove API rendendo compatibili tutte le funzioni già implementate
- Curare la parte relativa ad API che utilizzano lo stream, in particolar modo per il conseguimento delle nuove US proposte dal PO.
- Integrare SonarQube nella pipeline di Jenkins e risolvere più code smells possibile.
- Implementare almeno uno dei design pattern visti a lezione.

### 2.3.2 Sprint Backlog

- As a player I want to be able to answer a question and get points

**Stima di difficoltà:** 39 (story points) **Priorità:** Alta

**Tasks:** Filter correct tweets, check if the answer is right, button to answer from the site

- As a reader I want to have 10 votes to vote for my favorite books

**Stima di difficoltà:** 37 (story points) **Priorità:** Alta

**Tasks:** Keep track of votes, Count votes without replications

- As a hopeful writer I want to get my book voted

**Stima di difficoltà:** 36 (story points) **Priorità:** Alta

**Tasks:** Track a specific hashtag used to nominate books

- As a user I want to see a user's tweets in real-time, so that I can follow their movements on a map

**Stima di difficoltà:** 37 (story points) **Priorità:** Media

**Tasks:** Filtered stream implementation, map integration.

- As a user I want to know how many tweets are returned to me and in how long, so that I can evaluate the performance of the system

**Stima di difficoltà:** 7 (story points)    **Priorità:** Bassa

**Tasks:** Back-end data retrieving, visualization.

- As an organizer I want to be able to view the voting results in real time

**Stima di difficoltà:** 41 (story points)    **Priorità:** Alta

**Tasks:** Graphic design for votes, implementation with sockets.

- As an organizer I want to ask a question and count the answers, to see how many players there are

**Stima di difficoltà:** 39 (story points)    **Priorità:** Alta

**Tasks:** Track a specific hashtag that represent a game, graphic interface using diagrams.

- As organizer, I want to create a two-month short story literary contest run with twitter

**Stima di difficoltà:** 41 (story points)    **Priorità:** Alta

**Tasks:** Study stream API, back end data management, front-end implementation.

- Storyless tasks: Back-end refactoring, Front-end refactoring, improvements in sentiment analysis especially in performances.

### 2.3.3 Sprint Definition of Done

Le DOD del terzo sprint restano in parte simili a quelle del secondo:

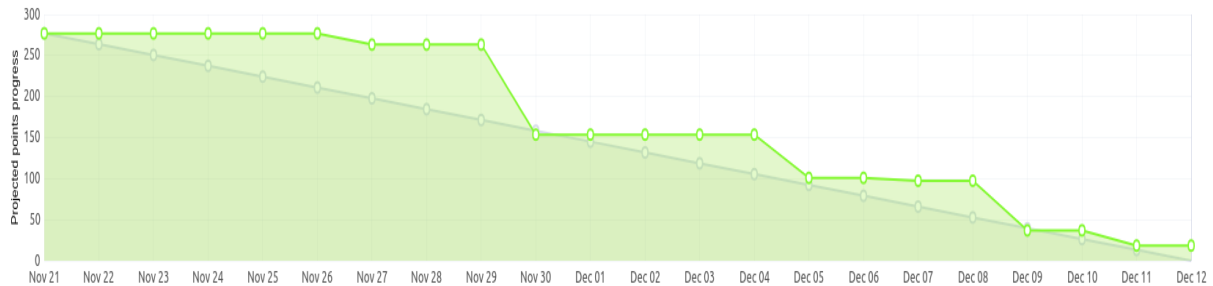
- Implementazione di un test per ogni feature;
- Codice funzionante, privo di errori e warning;
- Deployment su server remoto funzionante e automatizzato.
- Mantenere i code smells al minimo e azzerare bugs e vulnerabilità.

### 2.3.4 Sprint Burndown

Il grafico di burndown nel terzo sprint ha subito un peggioramento rispetto a quello del secondo. Anche se il team ritiene che quest'ultimo abbia comunque una forma accettabile, abbiamo stilato una lista dei motivi che ci hanno portato a peggiorare apparentemente la nostra produttività:

- Il refactoring ha portato via molto del tempo a disposizione nella prima settimana dello sprint (il refactoring è considerato una storyless task, quindi non appare nel grafico).
- Interfacciarsi con la stream e capire come utilizzarla al meglio inizialmente non è stato semplice, specialmente perchè passando alle API 2.0 abbiamo anche dovuto studiare una nuova documentazione con una nuova formattazione di query e tweets.

- Abbiamo avuto una serie di dubbi su come utilizzare le stream per poter accedere a più dati filtrati possibile contemporaneamente (problema risolto sfruttando due credenziali diverse contemporaneamente).



### 2.3.5 Sprint Retrospective

Siamo molto contenti di questa retrospettiva, che concettualmente dovrebbe essere l'ultima visto che nell'ultimo sprint l'unico obiettivo è migliorare ciò che è già stato fatto e creare la documentazione richiesta per la conclusione effettiva del progetto. [Link alla retrospettiva del terzo sprint](#)

High	relevance		
Medium	relevance		
Low	relevance		

Gli improvements che abbiamo individuato per l'ultimo sprint sono un miglioramento delle performance di ricerca e in generale maggiori accorgimenti in ambito di UI/UX.

## 2.4 Quarto sprint

Periodo di riferimento: 13/12/21 → 02/01/22

### 2.4.1 Sprint Goal

L'obiettivo da raggiungere durante questo sprint è il completamento del progetto nella sua interezza. I compiti specifici da svolgere sono presenti nello sprint backlog.

### 2.4.2 Sprint Backlog

Come già accennato, in questo sprint sono presenti solo storyless tasks:

- Complete code refactoring started in previous sprint;
- Add more graphics;
- Produce as many tests as possible;
- Complete code documentation.

### 2.4.3 Sprint Definition of Done

Le Definitions of done per questo sprint sono rimaste invariate rispetto a quello precedente, anche perchè non è richiesta l'implementazione di nuove features, ma solo un miglioramento di quelle presenti.

### 2.4.4 Sprint Burndown

Non è possibile presentare un diagramma di burndown per questo sprint per assenza di US. In compenso nonostante le vacanze natalizie di mezzo il team ha lavorato con costanza per poter presentare il progetto nei primi giorni disponibili di gennaio.

### 2.4.5 Sprint Retrospective

Generare una retrospettiva per questo sprint con il metodo utilizzato in quelli precedenti sarebbe stato certamente ripetitivo e non ottimale. Pertanto il team ha deciso di riunirsi e parlare di come è stato il progetto per noi, cosa abbiamo imparato, cosa potevamo fare meglio e quali sono state le difficoltà principali, per poi concludere che siamo soddisfatti del lavoro da noi svolto.

## 3 Descrizione del processo di lavoro

Sono stati svolti 4 sprint da 3 settimane ciascuno utilizzando un approccio Agile, implementato usando la metodologia SCRUM. In ognuno di essi sono state svolte attività di Sprint Planning e Sprint Review. Per lo sviluppo del prodotto, il team ha deciso di creare una web-app (disponibile a [questo indirizzo](#)) che sfrutta le seguenti tecnologie:

- Per il front-end:
  - React JS per UI/UX
  - Jest per il testing



- Per il back-end:
  - Node TS per la logica
  - Express per gestire le comunicazioni
  - Mocha-Chai per il testing
- Deploy: Gocker
- CI/CD: Jenkins
- Quality: SonarQube

Per planning, programmazione condivisa, versioning e comunicazione sono stati utilizzati anche altri strumenti dell'ambiente CAS, quali: Gitlab, Taiga e Mattermost.

Il code editor prediletto da tutti i membri del team per questo progetto è stato Visual Studio Code nella sua versione open source.

### 3.1 Autodescrizione del team

Per la costruzione del team ci siamo affidati a conoscenze pregresse dei membri e abbiamo utilizzato la piattaforma Telegram per metterci d'accordo. In seguito i nomi dei membri sono stati aggiunti su Trello per costruire il team e renderlo disponibile alla visione del professore. Prima di rendere la cosa ufficiale però c'è stata una prima riunione conoscitiva per mettersi d'accordo sul piano di consegna e sui ruoli dei membri. Abbiamo adottato i seguenti ruoli:

- Scrum Master: Stefano Colamonaco
- Product Owner: Andrea Corradetti
- Developers:
  - Alessio Di Pasquale
  - Leonardo Naldi
  - Filip Radovic

Oltre questo c'è stata una divisione dei ruoli più pratica dedicata all'assegnamento dei compiti in ambito di programmazione. I ruoli individuati sono stati: produttore di test, programmatori di UI/UX e programmatori back-end.

La costruzione del team e la divisione dei compiti si sono rivelate ottimali e hanno permesso l'utilizzo di alcune pratiche dell'extreme programming, in particolare:

- Continuous Integration;
- Pair Programming;
- Refactoring.

### 3.2 Risultato dello Scrumble iniziale

I risultati di Scrumble sono riportati nella figura sottostante.

Nel complesso riteniamo che questa attività svolta all'inizio del progetto sia stata utile sotto più aspetti: sicuramente ci ha dato un'infarinatura generale su quali sono i ruoli pratici di Scrum Master e PO all'interno del team, ma sicuramente la cosa più importante è che *serious games* come questo sono fondamentali a rompere il ghiaccio ed iniziare ad instaurare un rapporto tra i membri del team, in modo che sia più facile lavorare come gruppo.

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

### 3.3 Sintesi dei dati dalle applicazioni dell'ambiente CAS

Tutti i dati sono aggiornati alla data immediatamente successiva alla fine dell'ultimo sprint.

#### 3.3.1 Dati del logger

Purtroppo per problemi legati alle diverse architetture dei componenti del gruppo i dati forniti dai logger non sono accurati e mancano di una discreta quantità di dati, ma tutti i membri del gruppo hanno provveduto ad utilizzare questo strumento al meglio delle proprie possibilità.



### 3.3.2 Dati da Gitinspector

I dati e i grafici ottenuti attraverso l'analisi con lo strumento GitInspector risultano errati confrontati con quelli forniti direttamente da GitLab, dove risultano alcune commit in più.

Inoltre ovviamente questi grafici non tengono conto del lavoro svolto dai membri al di fuori del codice sorgente.

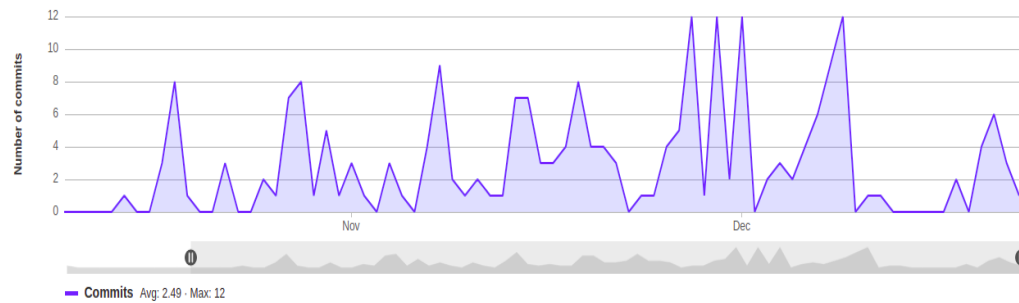


### 3.3.3 Dati da Gitlab

Seguono i dati complessivi relativi alle commits effettuate sul branch master del progetto e alcuni dati interessanti relativi alla distribuzione di queste ultime nel tempo:

#### Commits to master

Excluding merge commits. Limited to 6,000 commits.



#### Commit statistics for master Oct 07 - Jan 02

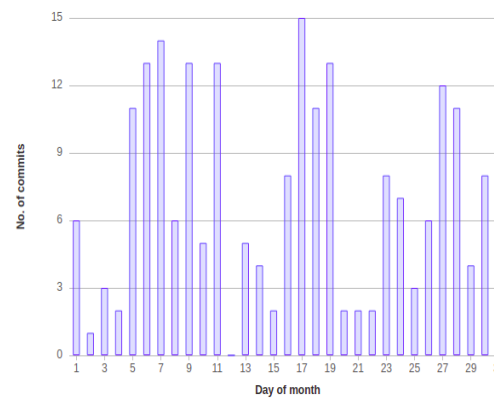
Excluding merge commits. Limited to 2,000 commits.

- Total: 219 commits
- Average per day: 2.5 commits
- Authors: 7

master

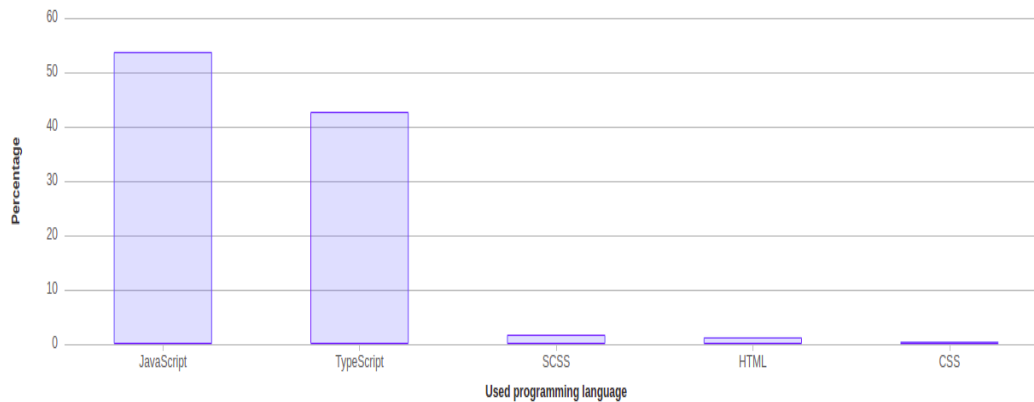
progetto-swe

Commits per day of month



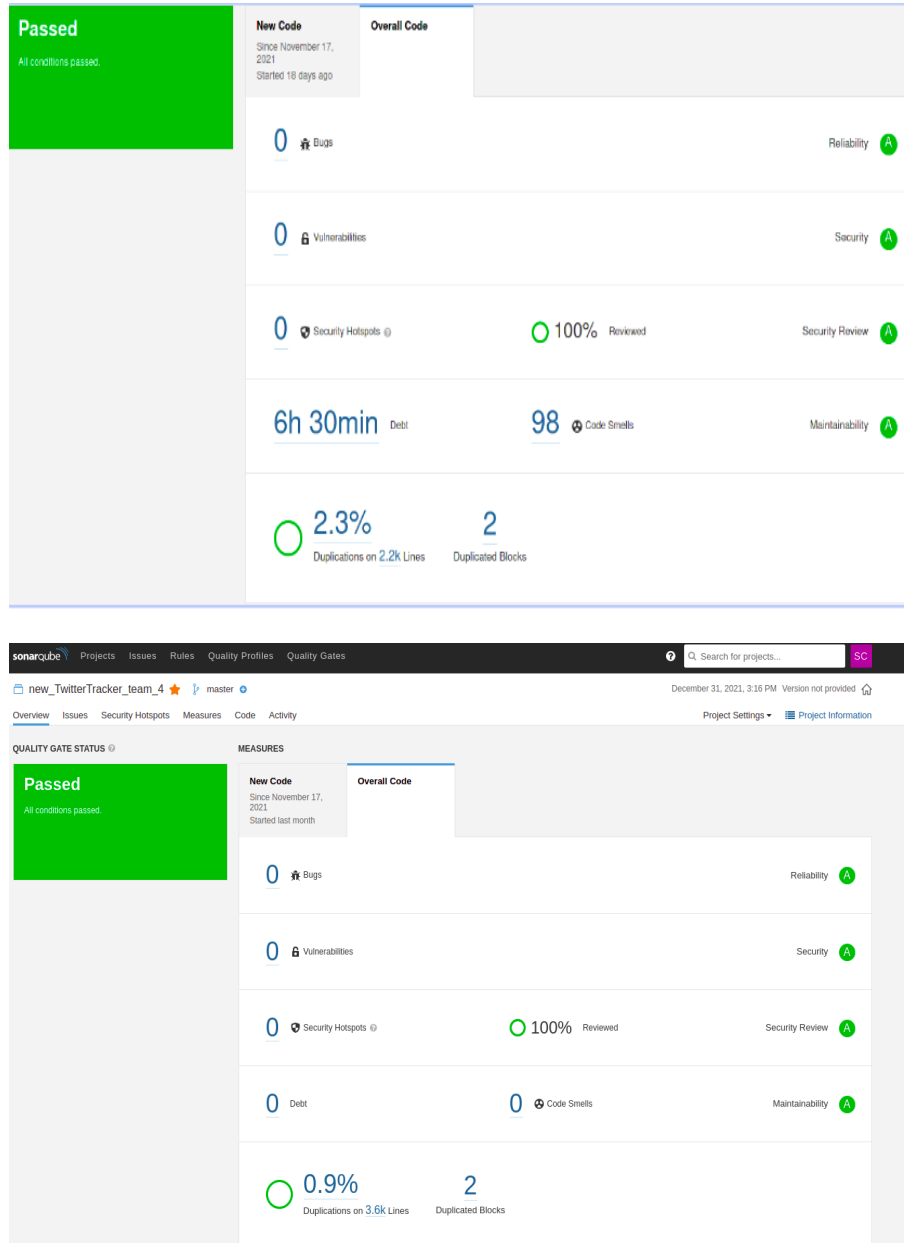
#### Programming languages used in this repository

Measured in bytes of code. Excludes generated and vendored code.

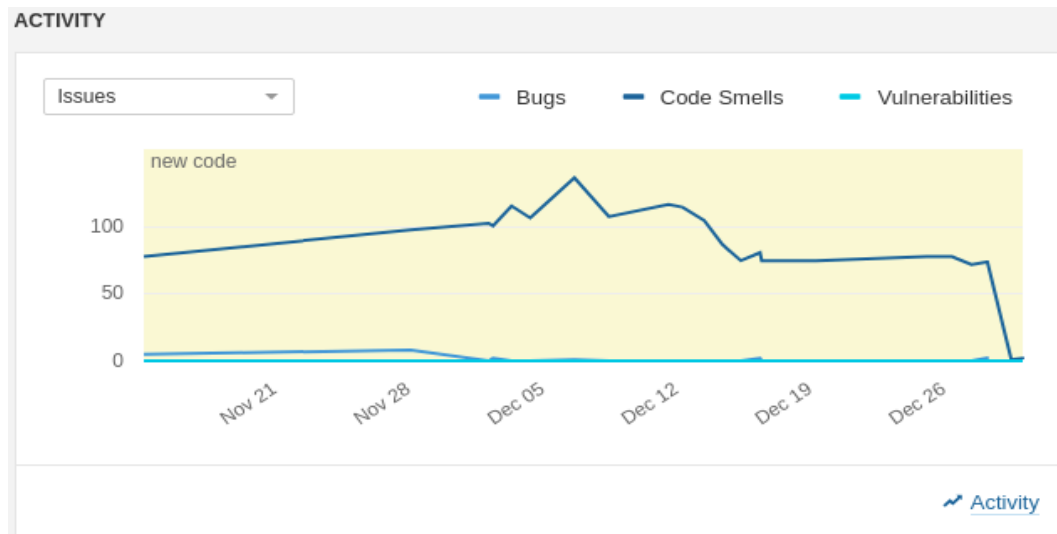


### 3.3.4 Analisi da SonarQube

Qui di seguito sono riportati i dati relativi alla situazione del codice in un momento intermedio all'interno del terzo sprint e alla fine dell'ultimo sprint.



Inoltre è presente un grafico che mostra l'andamento generale lungo tutto il periodo dello sviluppo.



### 3.3.5 Dati relativi al testing e a Jenkins

L'approccio TDD è stato alla base dello sviluppo dell'applicazione da circa la metà del secondo sprint in poi. Per permettere questo è stato necessario sviluppare un numero sufficiente di tests, in particolare 17 per il back-end e 15 per il front-end.

Per sviluppare questi tests sono state utilizzati i seguenti framework:

- **Mocha-Chai** per il back-end (per la precisione nella sua versione compatibile con TypeScript)
- **Jest** per il front-end

Il progetto è stato diviso in due pipeline differenti: una dedicata al front-end ed una per il back-end. Vediamole più nel dettaglio.

#### front-end

- Fetch del codice sorgente (con controllo automatico della commit ogni 15 minuti);
- Installazione dei pacchetti necessari attraverso npm;
- Esecuzione dei test front-end;
- Build della React Application;
- Generazione della documentazione attraverso un apposito script che richiama il servizio JSDoc;
- Deploy di codice e documentazione aggiornati sul server remoto (site202142.cs.unibo.it);

Per la parte relativa al front-end sono state eseguite oltre 160 build.

#### back-end

- Fetch del codice sorgente (con controllo automatico della commit ogni 15 minuti);
- Installazione dei pacchetti necessari attraverso npm;
- Transpilazione del codice TypeScript;

- Esecuzione dei test back-end;
- Modifica dei permessi nei file interessati, in particolare quello relativo al database, in quanto deve essere possibile modificarlo attraverso il programma;
- Generazione della documentazione attraverso un apposito script che richiama il servizio TypeDoc;
- Deploy di codice e documentazione aggiornati sul server remoto (site202142.cs.unibo.it);
- Analisi con SonarQube per tutto il codice e invio dei dati al server CAS.

Per la parte relativa al back-end sono state eseguite oltre 240 compilazioni.

### 3.4 Retrospettiva finale

Per generare una retrospettiva a conclusione del progetto, i membri del team si sono riuniti in videochiamata a discutere di vari aspetti relativi ad esso. Abbiamo preferito questo metodo per tirare le conclusioni perchè abbiamo pensato che una retrospettiva con Essence sarebbe stata ripetitiva rispetto all'ultimo sprint e probabilmente non sufficientemente espressiva in questa situazione.

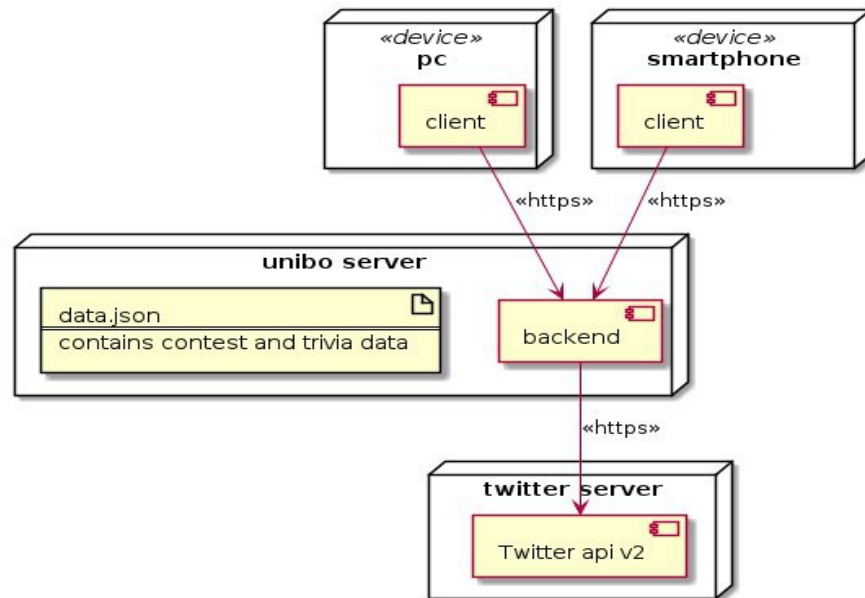
Di seguito sono elencati gli argomenti di discussione e le relative conclusioni:

- **Impressioni sull'ambiente di sviluppo:** Abbiamo trovato l'ambiente proposto da CAS completo e complessivamente facile ed intuitivo da utilizzare, non neghiamo però di aver trovato alcuni problemi che hanno reso meno funzionali alcuni strumenti, in particolare l'utilizzo del logger è stato piuttosto discontinuo a causa della sua compatibilità solo con alcuni IDE su specifici tipi di architettura. Inoltre abbiamo abbandonato dopo poco tempo l'utilizzo di Mattermost per lo scambio di informazioni preferendo Telegram poichè in grado di generare notifiche all'arrivo di nuove comunicazioni.
- **Rapporto costruito con i compagni di team:** Siamo completamente soddisfatti del rapporto che si è creato tra di noi, dentro e fuori dal progetto. Riteniamo inoltre di essere stati fortunati ad aver trovato membri capaci di mettersi in gioco e di lavorare in team con ottime performance. Ci riteniamo inoltre soddisfatti sia dell'assegnazione iniziale dei ruoli che della divisione dei compiti durante lo sviluppo.
- **Autovalutazione sul prodotto:** Siamo molto soddisfatti del prodotto che stiamo presentando. Riteniamo che sia completo, performante, stilisticamente gradevole e facile da utilizzare. Si tratta di caratteristiche fondamentali per un'applicazione di questo tipo.
- **Autovalutazione sul processo:** Riteniamo che la fase di planning sia stata fondamentale per la buona riuscita del progetto, infatti il carico di lavoro è stato distribuito equamente e gli strumenti a disposizione sono stati fondamentali per mantenere un work-flow in linea con gli obiettivi. L'unico dispiacere è stato quello di non aver consegnato il progetto a dicembre per via di alcune mancanze nella generazione degli artefatti. L'approccio TDD è stato davvero utile e ci ha permesso di scrivere codice più agevolmente e senza dover tornare indietro a correggere errori passati. Siamo tutti d'accordo che riutilizzeremo questo approccio in futuro.
- **Conclusioni sul progetto:** Dopo una discussione a riguardo siamo tutti d'accordo che questo progetto è stato davvero utile sotto diversi frangenti: ci ha avvicinati ad un ambiente più simile a quello lavorativo, ci ha insegnato ad interfacciarci con dei compagni di team per la



produzione collaborativa di un software dalle dimensioni notevoli, ci ha mostrato le difficoltà che si possono incontrare nel capire e realizzare le richieste di un cliente, ci ha insegnato il valore dei programmi che aiutano nello sviluppo software e l'importanza di rispettare una deadline. Sicuramente è stata un'esperienza fondamentale per la nostra crescita come programmatori.

### 3.5 Diagramma di deployment del prodotto



## 4 Artefatti

Di seguito vengono riportati gli artefatti prodotti:

- Codice sorgente: su [Gitlab](#)
- Documentazione del codice: [Front-end](#) e [Back-end](#)
- Grafi UML:
  - Diagramma delle classi
  - Diagramma dei casi d'uso
  - Diagramma di deployment
- Dati prodotti dal logger di CAS
- Diagrammi di burndown prodotti da Taiga
- Analisi effettuate da SonarQube
- Analisi effettuate da Gitinspector
- Tabella di autovalutazione per il gioco Scrumble fornita dal professore.
- Retrospective realizzate attraverso le carte [Essence](#) e seguendo un [manuale di Atlassian](#)