Relazione finale team 1

Progetto Heimdall: analisi traffico web

FABIO ZANICHELLI, ALEX CARAFFI, FRANCESCO CASTORINI, LUCA DALL'OLIO, FRANCESCO MALFERRARI, ANTONIO BENEVENTO VITALE NIGRO

Sommario

Scope e backlog del prodotto complessivo

Il progetto consiste nella realizzazione di una web app che permetta l'analisi del traffico di un web server Apache anche per persone non tecniche mostrando anche in un formato semplificato i log di quest'ultimo.

La web app è raggiungibile anche da remoto tramite il seguente indirizzo: http://64.225.69.78:3000/

Di seguito il backlog del prodotto:

ID	Storia	Punteggio	Stato
1	Io cliente voglio registrarmi al sito per poter usare la piattaforma	5	Realizzata in sprint 1
2	Io utente voglio accedere alla piattaforma attraverso un sistema di autenticazione per avere un'analisi del traffico del sito	13	Realizzata in sprint 1
3	Io utente voglio vedere con dei grafici sui paesi l'andamento del traffico del sito per poter ottenere informazioni velocemente sulle richieste al server	5	Realizzata in sprint 2
4	Io utente voglio vedere, con dei grafici sul risultato delle comunicazioni, l'andamento del traffico per poter ottenere informazioni velocemente sulla situazione attuale	5	Realizzata in sprint 2
5	Io utente voglio poter vedere da dove arriva il traffico attraverso una mappa per localizzare la provenienza dei pacchetti di rete	3	Realizzata in sprint 2
6	Io utente voglio poter vedere il grafico con i byte ricevuti per giorno per poter comprendere più velocemente la quantità di dati scambiati	5	Realizzata in sprint 3
7	Io tecnico/admin voglio poter conoscere i dettagli tecnici di ogni singola comunicazione per ottenere più informazioni	3	Realizzata in sprint 2
8	Io tecnico/admin voglio filtrare le comunicazioni per data per sapere quali e quanti pacchetti sono arrivati in un determinato intervallo di tempo	7	Realizzata in sprint 2
9	Io tecnico/admin voglio filtrare le comunicazioni per posizione per concentrarmi meglio sui pacchetti provenienti da una determinata nazione	7	Realizzata in sprint 2
10	Io tecnico/admin voglio filtrare le comunicazioni avvenute per concentrarmi meglio su eventuali azioni di marketing da intraprendere	7	Realizzata in sprint 2
11	Io tecnico/admin voglio filtrare le comunicazioni fallite per poter indagare eventuali azioni anomale	7	Realizzata in sprint 2
12	Io tecnico/admin voglio capire agevolmente se il traffico è malevolo per poter individuare eventuali problemi	13	Realizzata nello sprint 3
13	Io utente voglio che il software esegua autonomamente tutte le operazioni di controllo per non doverlo costantemente presidiare	20	Realizzata in sprint 2
14	Io utente voglio poter effettuare il logout per questioni di sicurezza	1/2	Realizzata in sprint 1
15	Io utente voglio poter contattare il sito tramite browser per poter accedere sempre alle sue funzionalità	13	Realizzata in sprint 2
16	Io admin voglio eliminare un utente dal database per evitare che faccia azioni non consentite	5	Realizzata in sprint 2
17	Io admin voglio inserire nuovi tecnici per consentire loro di entrare con degli accessi specifici	5	Realizzata in sprint 2
18	Io tecnico/admin voglio ricevere delle mail di notifica per sapere in tempo reale se sto subendo azioni potenzialmente pericolose	20	Realizzata in sprint 3
19	Io tecnico/admin voglio che il software agisca tempestivamente a determinate azioni malevoli per motivi di sicurezza	40	Programmata per sprint 3
20	Io utente admin voglio che il software generi predizioni in base allo stato attuale del traffico per capire come agire nel miglior modo possibile nel futuro	80	Realizzata in sprint 3

Diagramma dei casi d'uso

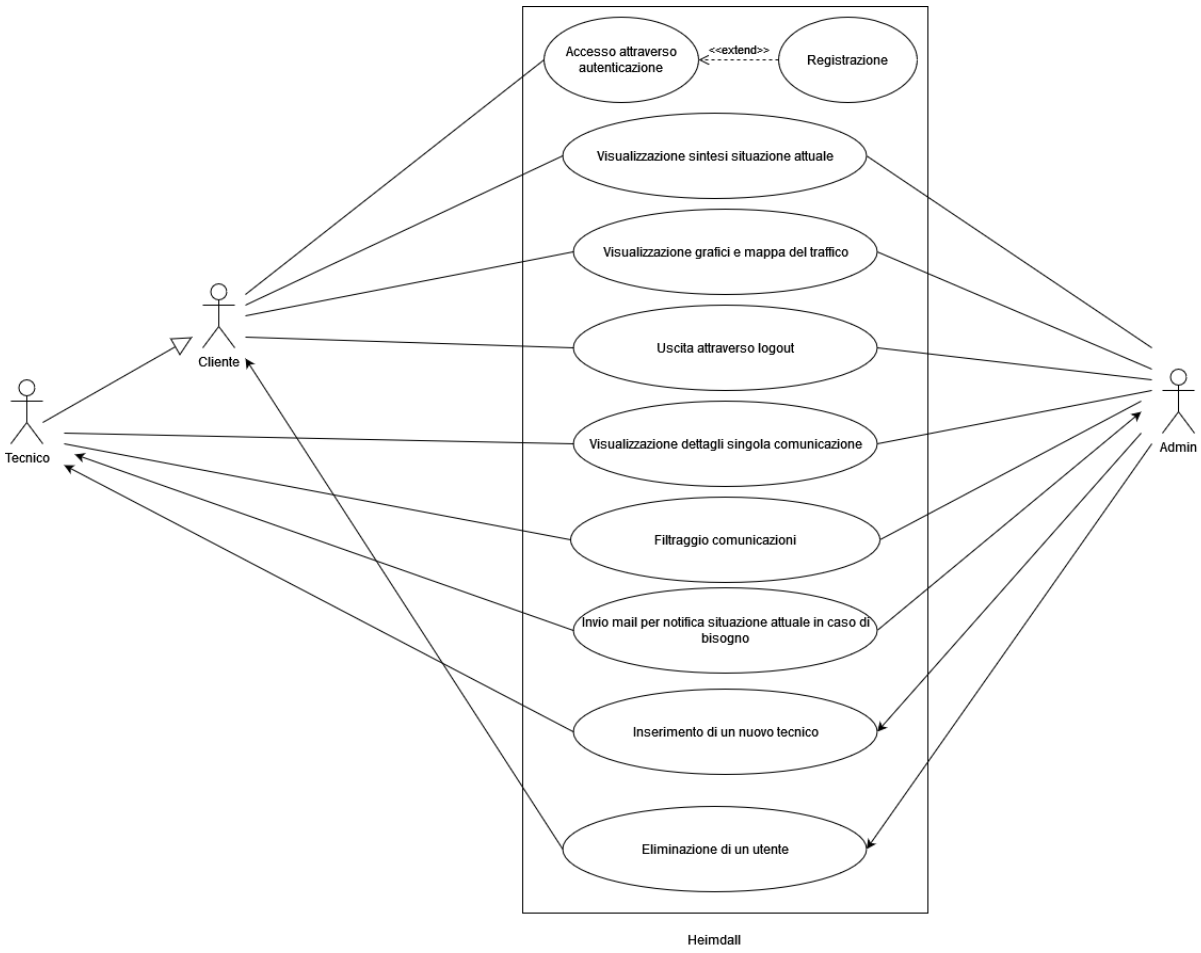

Figura 1: Diagramma dei casi d'uso

Il diagramma mostra le azioni possibili da parte dai vari tipi di utenti:

- Cliente: può vedere le statistiche giornaliere all'interno della web app
- Tecnico: è un cliente, registrato dall'admin, che può oltre a vedere le statistiche, anche filtrarle in base a vari campi e vedere le singole comunicazioni
- Admin: può creare nuovi tecnici, eliminarli, filtrare e visualizzare le statistiche e le singole comunicazioni

Scenari legati al diagramma dei casi d'uso:

1. Antonio vuole entrare come cliente, si registra al sito premendo il tasto ***Signup*** compilando i campi e successivamente può visualizzare la mappa e i grafici riguardo il traffico del giorno corrente.
2. Luca vuole entrare come Tecnico, quindi dovrà chiedere all'admin di registrarlo, dopodiché potrà accedere attraverso la schermata principale premendo ***Login*** e da lì può visualizzare la mappa, i grafici riguardo il traffico e premendo ***Filtra*** potrà vedere il traffico filtrato secondo

il criterio da lui voluto, in questo caso nazione = Italia. Cliccando su uno dei log potrà vedere un resoconto dei dettagli di quest'ultimo. In caso di traffico malevolo riceverà una mail per avvertirlo.

3. Fabio vuole entrare come Admin, quindi dalla schermata principale premendo ***Login*** potrà autenticarsi nella web app. Da lì potrà visualizzare la mappa, i grafici riguardo il traffico e premendo ***Filtra*** potrà vedere il traffico fallito dalla Germania nell'ultima settimana. Cliccando su uno dei log potrà vedere un resoconto dei dettagli di quest'ultimo. In caso di traffico malevolo riceverà una mail per avvertirlo. Inoltre, Fabio potrà registrare il nuovo tecnico Luca premendo il tasto ***Registra*** ed eliminare il tecnico Alex tramite il tasto ***Elimina***.

Diagramma architetturale

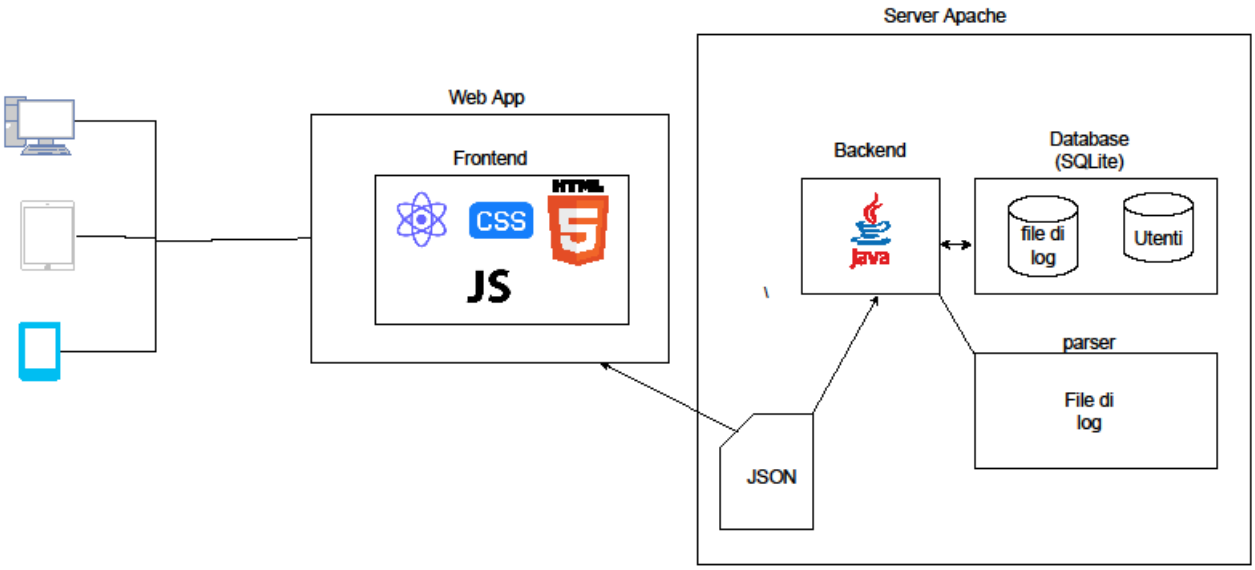

Figura 2: Schema architetturale

La figura mostra lo schema architetturale utilizzato per la realizzazione del progetto.

Per il front-end è stato utilizzato ReactJS, CSS, JavaScript e HTML, mentre per il back-end Java.

I file di log parsati (così come gli utenti) sono stati memorizzati in un database per un'operazione di filtraggio che al gruppo è sembrata più semplice da realizzare.

Diagramma delle classi

Segue il diagramma delle classi:

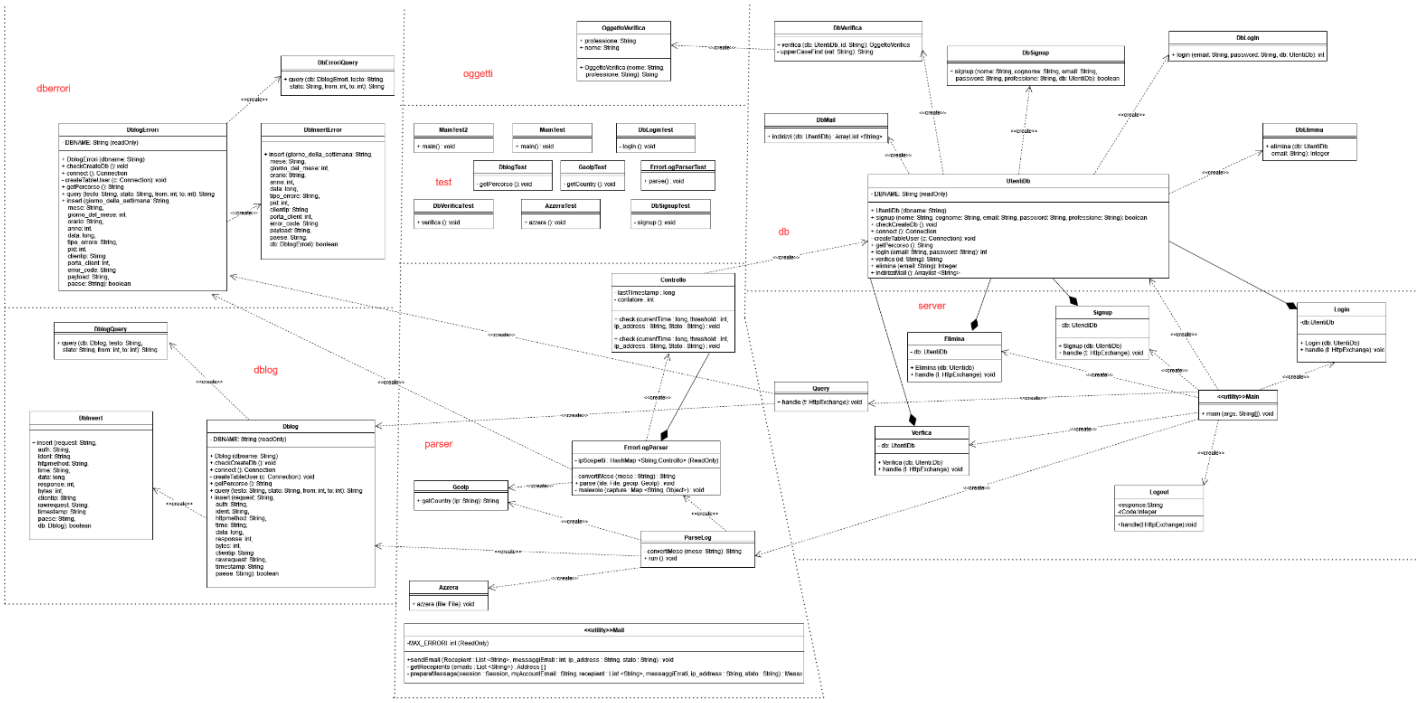

Figura 3: UML delle classi

Questo diagramma mostra le classi Java all'interno del progetto e le varie relazioni tra loro.

Diagramma di sequenza

Segue il diagramma di sequenza:

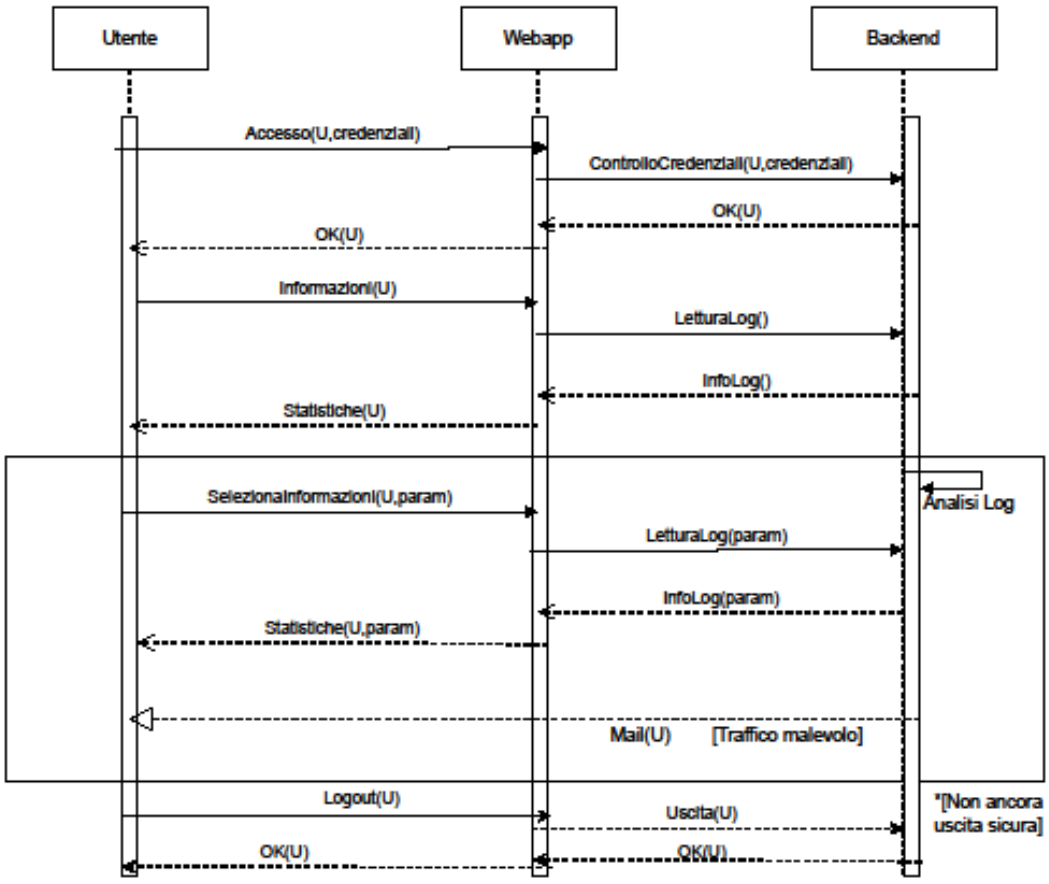

Figura 4: Diagramma di sequenza

Questo diagramma mostra la sequenza di eventi che si succedono all'interno della web app quando un utente si autentica. Dal login alle operazioni di filtraggio, fino al logout.

Documento dei rischi

Segue il documento dei rischi:

Nome criticità	Impatto	Difficoltà	Val.
Piattaforma usabile sul server	5 - Senza questa funzione, il progetto non funziona	5 - Difficile	25
Leggere i file di Log	5 - Senza questa funzione, il progetto non funziona	4 - Medio Difficile	20
Inserire marcatore di posizione file di log	4 - Senza questa funzionalità si perde una funzionalità del progetto	5 - Difficile	20
Predizione situazioni future tramite l'applicazione di algoritmi	4- Importante per lo stakeholder	5- Molto difficile	20
Mandare una mail di notifica in caso di azioni strane	4 - Importante, serve per sapere quando c'è qualcosa che non va nell'immediato	4 - Medio Difficile	16
Creazione della mappa lato client con posizione indirizzi ip	4 - Importante, serve agli utenti non esperti per capire da dove arrivano i pacchetti	4 - Medio Difficile	16
Ricerca testuale in entrambe le direzioni nel file di log	4 - Senza questa funzionalità si perde una funzionalità del progetto	4 - Medio Difficile	16
Applicazioni filtri sul file di log	4 - Importante	4 - Medio Difficile	16
Individuazione traffico malevolo	4 - Importante	4 - Medio difficile	16
Rendere il sito disponibile h24	5- Senza questa richiesta, il sito non è contattabile	3- Medio	15
Autenticare gli utenti usando il JWT	3 - Implementazione consigliata per la sicurezza ma non indispensabile	5 - Difficile	15
Parsare il file di log in un contesto dove le persone non tecniche capiscano	4 - Importante	3 - Medio	12

Figura 5:documento dei rischi

Questo diagramma rappresenta i rischi che presentava la realizzazione del progetto, con la valutazione dell'impatto e la probabilità di averlo, ognuno con una valutazione da 1 a 5 punti (1 per il minimo, 5 per il massimo).

Moltiplicando i due valori si ottiene un numero che da idea delle cose più rischiose da fare e quindi a quali funzionalità dare priorità; questo per evitare che problemi molto impattanti o comunque prevedibili non possano essere risolti per carenza di tempo.

Tool/IDE utilizzati

Per la realizzazione del progetto sono stati utilizzati i seguenti strumenti:

- **Intellij IDEA (community edition)** come IDE principale
- **Visual Studio Code** per il front-end e per vedere i contenuti dei database tramite l'estensione SQLviewer
- **Gitlab** per il versioning
- **Taiga (link)** per il backlog
- **Sonarqube (installato in locale)** per avere una valutazione della qualità del codice
- **Google Drive** per la condivisione dei documenti a tempo di scrittura e per i video da consegnare
- **Mattermost (link)** per avere una chat in cui discutere sul progetto
- **Slack** per essere in comunicazione con gli stakeholders
- **Jenkins** per le build
- **Essence** per le retrospettive a partire dallo sprint 2
- **DigitalOcean** come servizio di hosting per la web app
- **Draw.io** per la creazione dei grafici
- **Trello (https://trello.com/it)** per la creazione del team
- **Scrum Poker Cards (Agile)** per stimare le user stories

Burndown chart complessivo

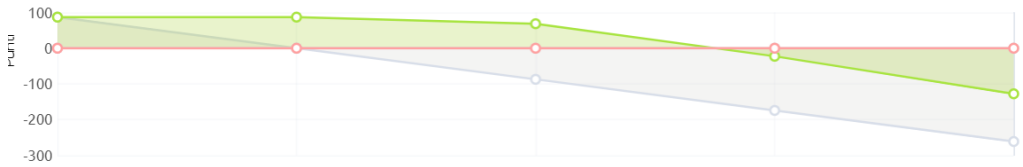

Figura 6: Burndown chart complessivo

Team

Auto descrizione del team

Il team è composto dai seguenti membri:

- Alex Caraffi (Project Owner)
- Fabio Zanichelli (Scrum Master)
- Antonio Benevento Vitale Nigro
- Francesco Castorini
- Francesco Malferrari
- Luca dall'Olio

Molti dei componenti del gruppo si conoscevano già di persona prima dell'inizio del progetto; i membri mancanti sono stati individuati guardando su Trello le caratteristiche e le descrizioni pubblicate, con l'obiettivo di creare una squadra più coesa, compente e completa possibile.

Descrizione del processo seguito

Il progetto si è articolato in quattro sprint ognuno di due settimane; il primo (denominato sprint 0) riguardava solo la parte di progettazione e di stesura dei documenti

La filosofia seguita per tutta la durata del progetto è stata quella del pair programming; tutti i compiti sono stati svolti a gruppi di 2-3 persone, con l'intenzione di ridurre il tempo di debugging (tendenzialmente per la persona che non sta scrivendo è molto più semplice individuare i bug) ed avere un maggior numero di idee.

Ogni coppia/terzetto era libera di organizzarsi a proprio piacimento, a patto che per la deadline successiva avesse svolto il proprio compito. Se questo fosse risultato troppo difficile, si sarebbe proceduto a spostare più persone nel gruppo più in difficoltà in ottica di un aiuto reciproco.

L'organizzazione del lavoro consisteva in brevi riunioni con tutti i membri ogni giorno. In queste riunioni ogni sottogruppo esponeva il proprio avanzamento e ci si accordava sui successivi compiti da realizzare.

Scrumble

Di seguito le GQM della partita di Scrumble:

GOAL	QUESTION	METRIC	
Learn	Do team members understand the Scrum roles?	Knowledge of Scrum roles by questions	Q1
	Do team members feel they learned the process?	Opinions from the participants	Q2
	Does everyone keep up with the other players?	Check during every sprint retrospective if every one is on point	Q3
Practice	Are the game mechanics linear and repeatable?	Opinions from the participants	Q4
	Do team success in completing the game?	Number of User Stories completed	Q5
	Do team members efficiently estimate during sprint planning?	Uniformity in evaluating the size and the priority of user stories	Q6
Cooperation	Do team members know each other better?	Level of players' serenity throughout the game	Q7
	Does the game let all players cooperate?	Contribution of every player during the game	Q8
	Do team member consult each other about a topic?	Sharing of ideas	Q9
Motivation	Do team members encourage collegues in need?	Players explain something other players don't understand	Q10
	Does PO help the team?	Quality of PO's advices to get better in the next sprints	Q11
	Does the team come up with good ideas?	Effectiveness of sprint retrospective	Q12
Problem Solving	Do team members behave well when facing a problem?	Level of the technical debt at the end of the game	Q13
	Does team organize their tasks properly?	Average of tasks left at the end of each sprint	Q14
	Does PO plan efficiently the Sprint Backlog?	Average of tasks left at the end of each sprint	Q15

Figura 7: Domande GQM

QUESTIONS	EVALUATION	Zanichelli	Caraffi	Castorini	dall'Olio	Benevento	Malferrari
Q1	1 = *no idea of the Scrum roles* 5 = *perfect knowledge of the roles and their jobs*	5	5	4	4	5	4
Q2	1 = *couldn't repeat the game* 5 = *could play the game as a Scrum Master by himself*	3	2	1	3	4	3
Q3	1 = *totally lost* 5 = *leads the game driving the other players*	5	4	3	5	5	5
Q4	1 = *feels the game is unrepeatable* 5 = *feels the game could be played in any situation*	3	1	1	4	4	3
Q5	1 = *0 to 3 stories* 2 = *4 to 6* 3 = *7 to 9* 4 = *10 to 12* 5 = *13 to 15*	4	4	4	4	4	4
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
Q7	1 = *never speaks with the other players* 5 = *talks friendly to anyone in every situation*	4	5	4	5	3	3
Q8	1 = *never puts effort in doing something* 5 = *every time is willing to understand what is going on*	5	4	3	4	5	5
Q9	1 = *never asks for an opinion* 5 = *wants to discuss about every topic*	4	1	4	4	4	4
Q10	1 = *not involved by the game* 5 = *always makes sure everyone is on point*	3	3	4	3	5	4
Q11 ONLY FOR PO	1 = *poor/absent advices* 5 = *wise and helpful suggestions when is required*	-	5	-	-	-	-
Q12	1 = *doesn't express opinions during retrospective* 5 = *feels the retrospective fundamental to express opinions*	2	5	5	5	5	5
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*	3	4	3	5	5	4
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	5
Q15 ONLY FOR PO	*Same evaluation as* **Q14** *for the PO*	-	5	-	-	-	-

Figura 8: Domande GQM

Analisi git e ore lavorate

Viene mostrato di seguito il numero di commit effettuati da ogni persona. Per avere questo valore è stato eseguito **`git shortlog -s -n`**

I commit che risultano effettuati con username diversi ma che corrispondono alla medesima persona sono stati sommati.

- Alex Caraffi: 101 commit
- Francesco Castorini: 66 commit
- Francesco Malferrari: 48 commit
- Fabio Zanichelli: 40 commit
- Antonio Benevento Vitale Nigro: 23 commit
- Luca dall'Olio: 18 commit

Questi valori non tengono conto dei commit effettuati con l'opzione **`co-authored-by`**. Il commit viene assegnato solo a colui che ha concretamente fatto il commit. Si precisa inoltre che si è utilizzata questa opzione solo dal termine del secondo sprint.

Segue il log delle ore lavorate personali e delle riunioni di gruppo:

Componente	Ore di lavoro (s0)	Ore di lavoro (s1)	Ore di lavoro (s2)	Ore di lavoro (s3)	Totale (persona)
Alex Caraffi (PO)	10	25	30	55	120
Fabio Zanichelli (SM)	10	20	30	51	111
Francesco Malferrari	7	20	25	53	105
Antonio Benevento	7	20	22	44	93
Luca Dall'Olio	7	20	22	43	92
Francesco Castorini	7	25	35	51	118
Totale (Sprint)	48	130	164	297	**639**

Data	Ora	Durata (ore)	Attività
11/04/2022	10.00.00	1.50	Analisi, valutazione e compilazione user story, documento dei rischi e documento delle features
14/04/2022	9.30.00	3	Conclusione, revisione e consegna sprint 0
14/04/2022	15.00	4	Partita a scrumble e relazione finale su scrumble
21/04/2022	16.00	2	Rivisitazione documenti dopo incontro con il tecnico
28/04/2022	15.00	1	Revisione generale sprint1 e controllo del codice
30/04/2022	9.00.00	4	Creazione nuovi documenti di consegna e ulteriore revisione
30/04/2022	14.00	3	Consegna del materiale per sprint 1

01/05/2022	21.00	0.25	Riunione giornaliera
02/05/2022	21.00	0.25	Riunione giornaliera
03/05/2022	21.00	0.25	Riunione giornaliera
04/05/2022	21.00	0.25	Riunione giornaliera
05/05/2022	11.00	0.25	Riunione con il cliente
05/05/2022	21.00	0.25	Riunione giornaliera
06/05/2022	21.00	0.25	Riunione giornaliera
07/05/2022	21.00	0.25	Riunione giornaliera
08/05/2022	21.00	0.25	Riunione giornaliera
09/05/2022	21.00	0.25	Riunione giornaliera
10/05/2022	21.00	0.25	Riunione giornaliera
11/05/2022	21.00	0.25	Riunione giornaliera
12/05/2022	21.00	0.25	Riunione giornaliera
13/05/2022	21.00	0.25	Riunione giornaliera
14/05/2022	21.00	0.25	Riunione giornaliera
14/05/2022	10.00	6	Consegna sprint 2
16/05/2022	21.00	2	Riunione con il cliente
17/05/2022	21.00	0.20	Riunione giornaliera
18/05/2022	21.00	0.20	Riunione giornaliera
19/05/2022	9.00	2	Preparata retrospective e fixati alcuni bug dati da SonarQube
20/05/2022	21.00	0.20	Riunione giornaliera
21/05/2022	21.00	0.20	Riunione giornaliera
22/05/2022	21.00	0.20	Riunione giornaliera
23/05/2022	21.00	0.20	Riunione giornaliera
24/05/2022	21.00	0.20	Riunione giornaliera
25/05/2022	21.00	0.20	Riunione giornaliera
26/05/2022	21.00	0.20	Riunione giornaliera
27/05/2022	14.00	2	Riunione giornaliera + diagramma UML
28/05/2022	10.00	6	Consegna sprint 3

Retrospettiva finale con Essence

Si rimanda alla retrospettiva Sprint 3 retrospective.

Sprint 1

Sprint 1 Goal

Realizzazione del sistema di autenticazione.

Sprint 1 Backlog

Si vogliono realizzare le User Stories 1, 2 e 13.

NB: la numerazione delle storie è cambiata più volte, per questa ragione il numero delle storie è diverso dal documento pubblicato all'inizio dello sprint 1.

Definizione di Ready

Una User Stories è considerata Ready quando:

1. Tutti hanno compreso la storia ed è stimabile
2. È testabile dai tester
3. Tutte le sue dipendenze sono già state implementate

Definizione di Done

Una User Stories è considerata Done quando:

1. È stata completamente sviluppata
2. Ha superato i test
3. È stata preparata una breve documentazione
4. Il branch "main" di Git contiene questa funzionalità

Test delle User Stories

- **US 1:** il test consiste nel provare a registrarsi utilizzando una mail già collegata ad un account esistente; la procedura deve quindi fallire.
  File: `DbSignupTest`
- **US 2:** il test prevede di simulare la procedura di autenticazione con una mail e una password corrette; in caso di login andato a buon fine ci si aspetta che l'id nel database corrisponda a quello previsto (la mail e la password riguardano un utente fittizio noto agli sviluppatori).
  File: `DbLoginTest`

Burndown chart

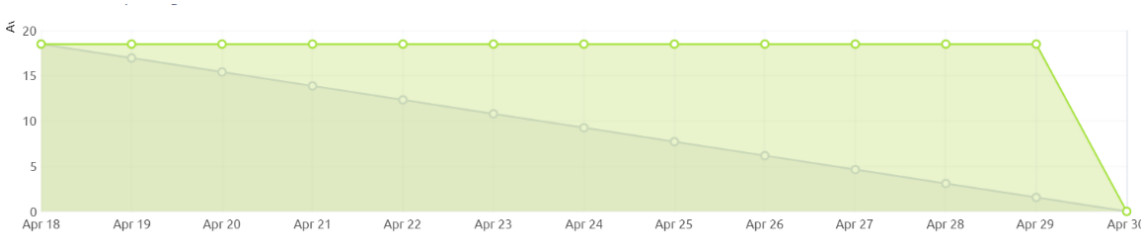

Figura 9: sprint 1 burn down

Il grafico ha questa “cascata” alla fine dello sprint perché ci sono stati dei problemi con l’utilizzo di Taiga.

Risultati SonarQube

SonarQube è stato installato verso la fine dello sprint 2, pertanto per questo sprint non sono disponibili i suoi risultati.

Retrospective

Segue la retrospettiva dello sprint 1; la metodologia Essence è stata utilizzata a partire dallo sprint 2.

SPRINT RETROSPECTIVE

Per la prima retrospettiva è usato il metodo base:

Cosa ha funzionato:

- Il team risulta coeso e flessibile per affrontare le difficoltà riscontrate
- Il backlog è stato modificato in base ai feedback
- Sono state realizzate le user stories prefissate al meglio possibile ed è stato creato un eseguibile lanciabile da riga di comando per il backend e uno script di avvio per il frontend
- La modalità di programmazione pair programming è stata effettuata quasi interamente
- Ogni membro del gruppo comunicava tempestivamente gli upgrade effettuati
- Riunioni molto vicine per scambiarsi opinioni e feedback

Cosa poteva essere migliorato:

- La sicurezza del login (non è stato usato JWT)
- Test migliori e più specifici
- L’integrazione con tool come Sonarqube e Jenkins
- Utilizzo migliore del GIT

Cosa si può fare per migliorare nel prossimo sprint:

- Realizzazione test più efficaci e precisi entro la fine del prossimo sprint
- Seria analisi dei tools sopracitati per essere integrati dallo sprint 2 in poi
- L’utilizzo di GIT in una maniera più professionale a partire da adesso
- Documentazione migliore

Sprint 2

Sprint 2 Goal

Realizzazione della lettura del file di log, la sua presentazione all'utente.

Sprint 2 Backlog

Si vogliono realizzare le User Stories 3, 4, 5, 6, 7, 8, 9 ,10, 12, 14, 15 e 16.

NB: la numerazione delle storie è cambiata più volte, per questa ragione il numero delle storie è diverso dal documento pubblicato all'inizio dello sprint 2.

Definizione di Ready

Una User Stories è considerata Ready quando:
1. Tutti hanno compreso la storia ed è stimabile
2. È testabile dai tester
3. Tutte le sue dipendenze sono già state implementate

Definizione di Done

Una User Stories è considerata Done quando:
1. È stata completamente sviluppata
2. Ha superato i test
3. È stata preparata una breve documentazione
4. Il branch "main" di Git contiene questa funzionalità

Ogni codice che fa parte del progetto deve avere il suo codice di test, indicativamente con un coverage minimo del 40%.

Test delle User Stories

- **US 5:** il test consiste nel verificare che un indirizzo IP italiano venga effettivamente riconosciuto come tale
  File: `GeoIpTest`
- **US 6:** il test si assicura che il codice sappia il giusto percorso dei file di log.
  File: `DblogTest`
- **US 7:** il test si assicura che vi è una connessione aperta sulla porta 3000, essenziale per poter accedere al sito.
  File: `MainTest2`

Burndown chart

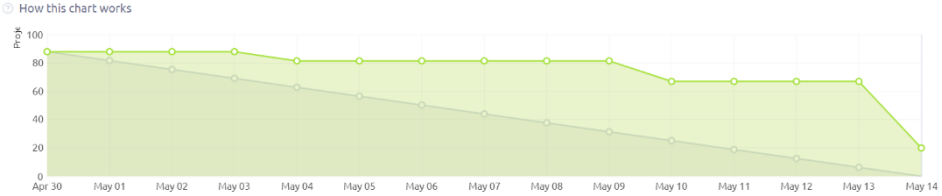

Figura 10: Burndown chart sprint 2

Risultati SonarQube

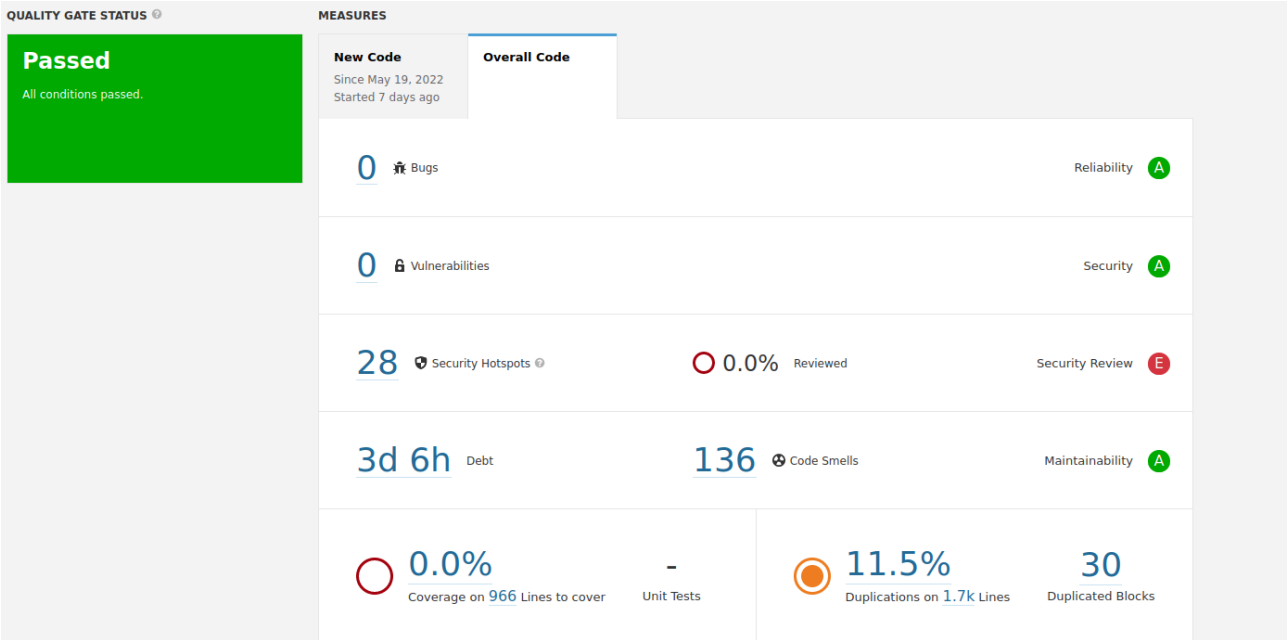

Figura 11: Risultati SonarQube sprint 2

Retrospective

Segue l'immagine della retrospettiva realizzata con Essence:

Figura 12: Essence sprint 2

Cosa ha funzionato:

- Il team risulta coeso e flessibile per affrontare le difficoltà riscontrate grazie alla disponibilità di tutti i membri e l'utilizzo predominante del pair programming

- Tutta la documentazione è stata modificata in base ai feedback ricevuti, secondo il metodo agile, in particolare focalizzandosi sul Backlog, spezzando le US in altre più dettagliate (presente nel sprint_1 retrospective)
- La modalità di programmazione in pair programming
- Ogni membro del gruppo comunica tempestivamente gli upgrade effettuati
- Riunioni quotidiane per scambiarsi opinioni e feedback, utilizzando appositi tools, soprattutto dopo incontri con stakeholders
- Sono stati studiati e implementati programmi per il controllo dello sviluppo del progetto come: Mattermost, Jenkins e SonarQube (presente nel sprint_1 retrospective)
- Utilizzo più professionale e distribuito di Git (presente nel sprint_1 retrospective)

Cosa poteva essere migliorato:

- Meno codice hard codato e scrittura di codice più flessibile ai cambiamenti, in linea con lo sviluppo agile
- Ulteriore partizionamento delle User Stories in task, per rendere più chiaro e immediato l'avanzamento del progetto.
- Sono stati realizzati test dove c'era un ritorno importante per verificare la corretta azione delle funzioni osservate, purtroppo si sono rilevati Integration test, quando erano stati richiesti Unit test (presente nel sprint_1 retrospective)
- Maggior comprensione dei requisiti richiesti per evitare di compiere lavoro inutile

Cosa si può fare per migliorare nel prossimo sprint:

- Migliorare la comunicazione tra i colleghi sfruttando Mattermost a partire da adesso
- Utilizzo delle informazioni e statistiche date da Jenkins e SonarQube per migliorare la qualità del codice una volta che si è arrivati a soddisfare gran parte dei requisiti
- Revisione periodica della documentazione a partire dai prossimi feedback
- Richiesta di ulteriori riunioni e chiarimenti se necessario a partire da questo sprint

Sprint 3

Sprint 3 Goal

Finire il progetto con la realizzazione delle User Stories rimanenti, riguardanti prevalentemente le funzioni di analisi/previsione e di notifica.

Sprint 3 Backlog

Si vuole completare la storia 9 e realizzare le User Stories 17, 18, 19, 20.

NB: la numerazione delle storie è cambiata più volte, per questa ragione il numero delle storie è diverso dal documento pubblicato all'inizio dello sprint 3.

Definizione di Ready

Una User Stories è considerata Ready quando:

1. Tutti hanno compreso la storia ed è stimabile
2. È testabile dai tester
3. Tutte le sue dipendenze sono già state implementate

Definizione di Done

Una User Stories è considerata Done quando:

1. È stata completamente sviluppata
2. Ha superato i test
3. È stata preparata una breve documentazione
4. Il branch "main" di Git contiene questa funzionalità

Ogni codice che fa parte del progetto deve avere il suo codice di test, indicativamente con un coverage minimo del 40%.

Burndown chart

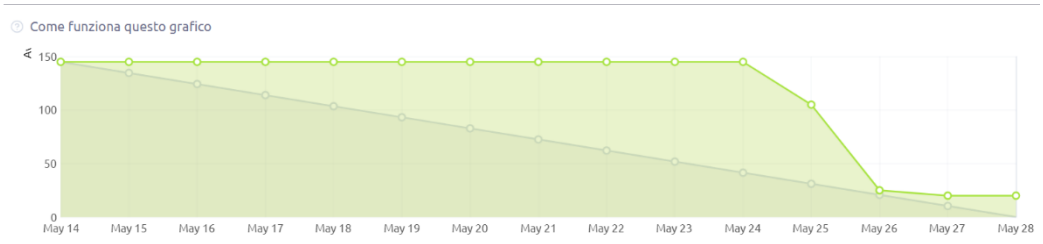

Figura 13: Burndown sprint 3

Risultati SonarQube

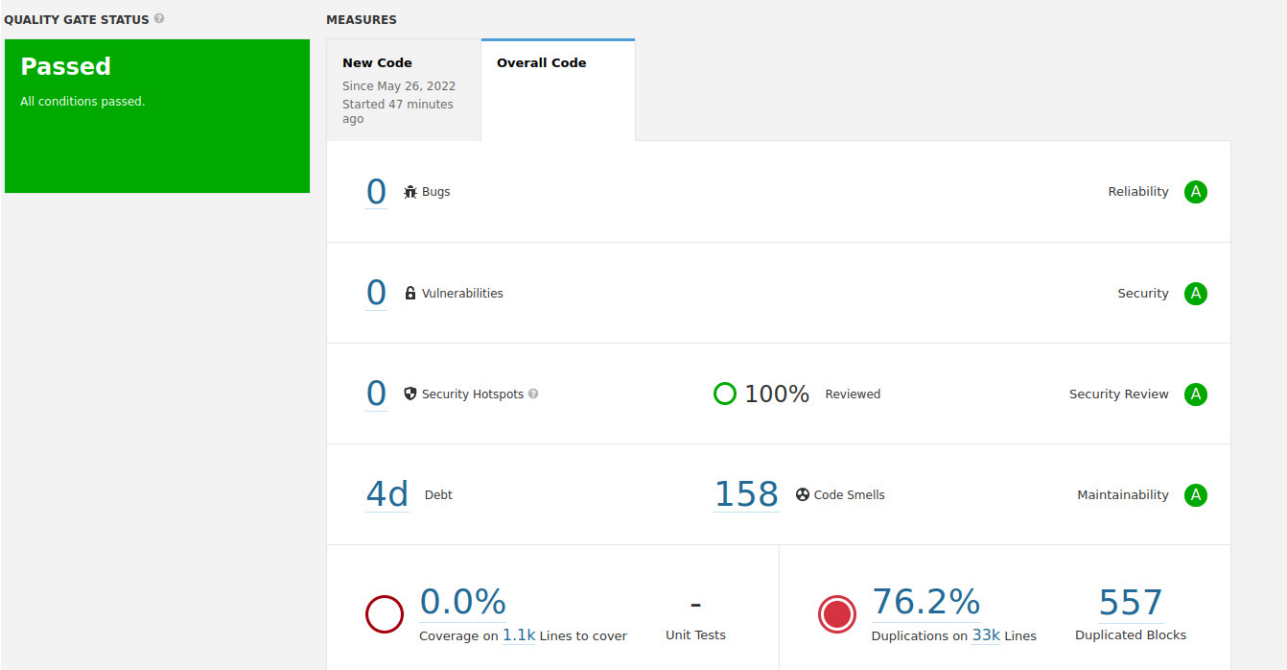

Figura 14: Risultati SonarQube sprint 3

Retrospective

Segue l'immagine della retrospettiva con Essence:

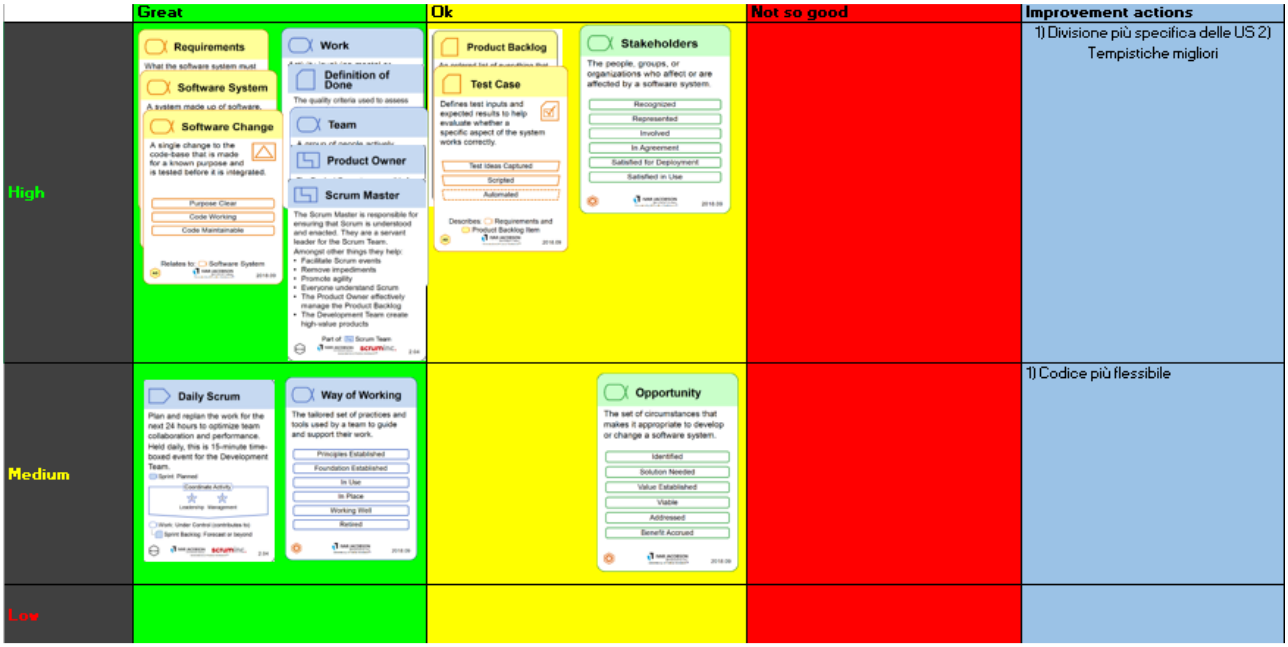

Figura 15: Essence sprint 3

Cosa ha funzionato:

- Il team risulta coeso e flessibile per affrontare le difficoltà riscontrate grazie alla disponibilità di tutti i membri e l'utilizzo predominante del pair programming
- Tutta la documentazione è stata modificata in base ai feedback ricevuti, secondo il metodo agile, in particolare focalizzandosi sul Backlog, spezzando le US in altre più dettagliate (presente nel sprint_2 retrospective)
- Ogni membro del gruppo comunica tempestivamente gli upgrade effettuati
- Riunioni quotidiane per scambiarsi opinioni e feedback e utilizzando appositi tools come Mattermost (presente nel sprint_2 retrospective) e Discord
- Sono stati studiati e implementati programmi per il controllo dello sviluppo del codice come: Jenkins e SonarQube (presente nel sprint_2 retrospective)
- Utilizzo più professionale e distribuito di Git (presente nel sprint_2 retrospective)
- Sono stati aggiunti ulteriori Unit test dove è stato possibile (presente nel sprint_2 retrospective)

Cosa poteva essere migliorato:

- Meno codice hard codato e scrittura di codice più flessibile ai cambiamenti, in linea con lo sviluppo agile
- Migliore gestione del tempo in vista dei task più complessi

Capacità apprese al termine del progetto:

- Capacità di lavorare in team scambiandosi continui feedback, opinioni e problemi
- La realizzazione di documentazione, molto presente nel mondo del lavoro per quanto concerne lo sviluppo software
- L'utilizzo professionale di GIT da linea di comando e la comprensione della sua struttura
- L'utilizzo di alcuni tool per il lavoro in team e il controllo dei progetti software (installazione, integrazione e uso)

  Approccio agli stakeholder in simulazione al mondo lavorativo vero e proprio

Lista e descrizione degli artefatti (di prodotto e di processo)

Per quanto riguarda il back-end è stato prodotto un file JAR eseguibile nella cartella "2022t1" attraverso il comando:

```
java -jar ./code/code.jar
```

Mentre per quanto riguarda il front-end è stata realizzata un'applicazione ReactJS che è eseguibile nella cartella "2022t1/code/frontend" attraverso i comandi:

`npm install` (Per installare le dipendenze, da eseguire una sola volta)

```
npm start
```

L'intera applicazione è disponibile sul server concesso da DigitalOcean accessibile attraverso il link HeimdallServer

Sprint review finale

Il video dello sprint rewiew finale si trova al seguente link:

https://drive.google.com/file/d/14iNESvm_ncVe5wiBbrAfNwpFLGj8OOsx/view?usp=sharing

Conclusioni e suggerimenti relativi al corso

Come gruppo abbiamo sicuramente imparato a programmare in team, esperienza che fino ad oggi nessuno aveva fatto precedentemente; inizialmente la differenza si è notata parecchio, ma con il passare dei giorni abbiamo iniziato a migliorare questo aspetto rendendo molto più efficiente la codifica.

L'utilizzo massivo della metodologia del pair programming ha sicuramente contribuito a migliorare le idee di realizzazione del progetto oltre alla coesione del gruppo, collaborando di fatto in maniera diretta tra i suoi vari componenti.

Sicuramente sono state apprese numerose nozioni riguardanti la filosofia Agile, che ci ha consentito di strutturare il progetto da una prospettiva che non riguardasse solo la mera scrittura del codice, ma anche da un punto di vista più organizzativo e sociale.

Tra i problemi riscontrati vi è stato quello della sovrapposizione del progetto con altri lavori svolti dai membri del gruppo; il problema è stato risolto cercando di lavorare contemporaneamente su funzionalità separate tra di loro, in modo che ogni coppia/terzetto potesse organizzarsi come meglio credeva prima della deadline.

Come gruppo riteniamo che il progetto sia stato eccessivamente complesso da realizzare, individuando come principale causa la carenza di conoscenze acquisite nei precedenti tre anni di studi universitari; questa motivazione ci ha portato a scegliere un goal per lo sprint 1 non ottimale secondo la metodologia Agile, in modo da iniziare a famigliarizzare con le tecnologie necessarie.

Crediamo che un progetto più semplice possa aiutare a concentrarsi meno sulla sua implementazione (ovvero sulla codifica) a favore della progettazione, che secondo noi è il fulcro di questo insegnamento.