UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA IN INGEGNERIA DEL SOFTWARE

Repository Analyzer: un Plugin per Analizzare il Contributo di Sviluppatori Software

RELATORE

Prof. **Filomena Ferrucci**

Università degli Studi di Salerno

CANDIDATO

Kevin Pacifico

Matricola: 0512108611

Anno Accademico 2021-2022

Alla mia famiglia ed ai miei amici che mi hanno sostenuto ed aiutato in questo splendido percorso.

Sommario

L'ingegneria del software è una disciplina informatica che si occupa dei processi produttivi e delle metodologie di sviluppo finalizzate alla realizzazione di sistemi software.

Negli ultimi anni si è molto diffuso lo sviluppo open source, una metodologia di sviluppo che rende i progetti e le risorse ad esso associati pubbliche ed accessibili a chiunque, tramite alcune piattaforme come Github, con l'obiettivo di espanderne il team di sviluppo e facilitare il riutilizzo di funzionalità software.

Con l'affermarsi della comunità open source è diventato di fondamentale importanza prevenire l'abbandono di un progetto software open source e ciò può essere possibile analizzando quanto ogni singolo sviluppatore abbia contribuito e stia contribuendo alla realizzazione del progetto software.

Github è la piattaforma dei progetti software open source per eccellenza e l'obiettivo è quello di creare un plugin che per una qualsiasi repository pubblica su Github sia in grado di effettuare un'analisi quantitativa ed un'analisi qualitativa ai fini di prevenire l'abbandono dei progetti software.

Indice

Elenco delle figure

CAPITOLO 1

Introduzione

1.1 Contesto

L'ingegneria del software è una disciplina informatica che si occupa dei processi produttivi e delle metodologie di sviluppo finalizzate alla realizzazione di sistemi software.

Negli ultimi anni si è molto diffuso lo sviluppo open source, una metodologia di sviluppo che rende i progetti e le risorse ad esso associati pubbliche ed accessibili a chiunque in modo da permettere ai programmatori di apportare modifiche, correggere errori o creare nuove funzionalità all'interno di esso, consentendo a qualsiasi programmatore la libera interpretazione personale di un progetto software.

Github è la piattaforma per eccellenza dello sviluppo open source e conta oltre 100 milioni di progetti software, si tratta di una piattaforma di hosting per codice sorgente di progetti software basato su un sistema di controllo decentralizzato.

1.2 Problemi e Motivazioni

I progetti open source spesso sono gestiti da un numero piccolo di sviluppatori senza alcun sostegno finanziario, molti sviluppatori open source sono volontari o lavorano part-time su un progetto, la mancanza di un'equa retribuzione può comportare all'abbandono di un progetto.

L'abbandono di uno sviluppatore potrebbe rallentare o bloccare lo sviluppo di un progetto software open source. Un progetto open source può offrire varie funzionalità che potrebbero avere componenti in comune, se si hanno delle dipendenze tra le varie componenti e lo sviluppatore che stava implementando una specifica componente decide di abbandonare il progetto potrebbe compromettere lo sviluppo di altre componenti del progetto software, bloccando lo sviluppo ad altri implementatori.

1.3 Contributo della Tesi

Tramite un'analisi del contributo effettivo apportato da uno sviluppatore su un progetto software open source è possibile intuire l'interesse che esso mostra verso il progetto ed analizzando alcuni fattori è possibile capire se uno sviluppatore sta abbandonando un progetto software.

Se si intuisce a priori l'intenzione dello sviluppatore di abbandonare il progetto è possibile stimolarlo a continuare oppure sostituirlo con un altro sviluppatore evitando di inficiare in maniera negativa sul processo di sviluppo del progetto.

L'obiettivo è la creazione di uno strumento che sia in grado di analizzare il contributo effettivo apportato per ogni singolo sviluppatore di un progetto open source, in modo da prevenirne l'abbandono.

1.4 Risultati Ottenuti

Repository Analyzer è un progetto open source scaricabile ed accessibile a chiunque, si tratta di un plugin disponibile nel Chrome Web Store che permette tramite un'interfaccia user-friendly anche ad utenti meno esperti di effettuare un'analisi di una qualsiasi repository pubblica su Github.

Una volta inserito il link della repository nell'interfaccia principale del plugin viene aperta una pagina che fornisce nome, descrizione, data di creazionee e data di ultima modifica della repository.

Lungo la pagina è possibile trovare un'analisi del contributo effettivo ottenuta tramite delle metriche che valutano la quantità e la qualità dell'operato da parte dello sviluppatore.

1.5 Struttura della Tesi

Nel capitolo "Background"[capitolo 2] si analizza lo stato dell'arte tramite alcuni articoli scientifici con l'obiettivo di investigare sulla nascita ed i benefici che la comunità open source ha apportato nello sviluppo di un progetto software, le metriche per stabilire quando un programmatore sta abbandonando un progetto open source ed alcuni metodi per stabilire il lavoro effettivo apportato da uno sviluppatore su un progetto open source.

Nel capitolo Repository Analyzer[capitolo 3] è illustrato il progetto Repository Analyzer descrivendone le caratteristiche, il funzionamento, come vengono estrapolati i dati, le metriche per stabilire il contenuto, gli obiettivi e come essi vengono raggiunti.

Nel capitolo "Validazione"[capitolo 4] vengono fatte una serie di analisi volte alla validazione del progetto Repository Analyzer paragonandone le statistiche con quelle fornite da Github.

Nel capitolo "Conclusioni"[capitolo 5] vengono presentati gli sviluppi futuri.

CAPITOLO 2

Background

In questo capitolo è illustrato lo stato dell'arte e vengono analizzati alcuni articoli scientifici con l'obiettivo di investigare sulla nascita ed i benefici che la comunità open source ha apportato nello sviluppo di un progetto software, le metriche per stabilire quando un programmatore sta abbandonando un progetto open source ed alcuni metodi per stabilire il lavoro effettivo apportato da uno sviluppatore su un progetto open source.

2.1 Comunità Open Source

2.1.1 Cos'è la Comunità Open Source

Un software closed source, detto anche software proprietario o commerciale, è un software la cui licenza permette l'utilizzo ad un numero limitato di sviluppatori, il codice sorgente non viene diffuso perché ritenuto come un segreto aziendale.

L'informatico Richard Stallman nel 1984 propose di creare la Free Software Foundation con l'idea di rendere lo sviluppo di progetti software accessibile e modificabile da tutti gratuitamente. Nel 1997 nacque la open source Definition e successivamente vennero sviluppati i primi progetti open source.

Un software open source non è vincolato da alcuna licenza ed è accessibile a chiunque in modo da permettere ai programmatori di apportare modifiche, correggere errori o creare nuove funzionalità all'interno di un progetto software, consentendo ai programmatori la libera interpretazione personale di un determinato progetto.

La piattaforma proprietaria del software può decidere di rendere il software totalmente gratuito oppure fornire una licenza per uso commerciale, consentendo così il libero accesso al codice sorgente in modo da controllare come organizzazioni o singoli individui programmano e adattano il codice per soddisfare determinate esigenze aziendali o personali.

Quando un autore rilascia un progetto open source aumenta l'aspettativa per la quota di mercato e ha la sicurezza di costruire una base di sviluppo per un progetto aumentando la sostenibilità a lungo termine. Per gli autori che non cercano di commercializzare i loro prodotti software l'open source offre molti vantaggi.

Inoltre un progetto open source offre un particolare sbocco creativo all'autore aumentando il numero di parti interessate che anche se contribuiscono nel proprio interesse infieriscono positivamente sulle nuove versioni di un software.

Lo sviluppo open source riguarda un numero elevato di sviluppatori e programmatori in quanto si tratta di un approccio tecnico ed efficace allo sviluppo del software che permette di rilasciare presto e spesso più versioni di un software delegando tutto ciò che potrebbe risultare promiscuo nello sviluppo.

Inoltre con lo sviluppo open source è possibile cooperare concorrentemente sull'implementazione di un progetto software, questo permette di avere idee differenti nelle fasi di sviluppo del software che viene implementato da un team formato da programmatori provenienti da Paesi diversi e con competenze differenti.

Le persone che lavorano in un progetto open source sono guidati da fattori motivazionali, secondo lo studio di Eric Raymond [1] il libero accesso al codice sorgente di un software incentiva la motivazione degli sviluppatori.

Un'altra osservazione argomentata da Raymond nel suo studio è l'importanza della competenza e dell'intelligenza nelle fasi di progettazione dei coordinatori o del leader, che tramite delle buone capacità di comunicazione incentivano gli sviluppatori nel proprio lavoro.

2.1.2 Github

Github [figura 2.1] è una piattaforma di hosting per codice sorgente di progetti software basato su un sistema di controllo decentralizzato.

All'atto della creazione di un progetto software da parte di un'azienda o di uno sviluppatore viene creata un'apposita repository che ospiterà i documenti ed il codice del software; si tratta di una sorta di cartella che contiene documenti e codice sorgente riguardante un progetto software.

Il creatore della repository stabilisce i collaboratori, ovvero gli sviluppatori che possono accedere al contenuto della repository ed apportarne modifiche.

Inoltre il proprietario della repository deve impostarne la visibilità, che può essere privata o pubblica. Il contenuto di una repository privata è accessibile e modificabile dai soli collaboratori, mentre il contenuto di una repository pubblica è accessibile a qualsiasi sviluppatore che tramite l'operazione "fork" di Github può creare e sviluppare una versione personale di quel progetto. Quando uno sviluppatore effettua una "fork" su una repository pubblica crea una nuova repository di cui sarà il proprietario, gestendola quindi a proprio piacimento.

Una repository Github è composta da più "branch" che consistono in delle partizioni dove si sviluppano diverse parti di uno stesso progetto.

Ogni collaboratore della repository avrà salvato in locale un clone della repository e lavorerà su di esso.

Gli sviluppatori apportano modifiche concorrentemente anche su stessi file o snippet di codice, la piattaforma Github permette ciò tramite le operazioni di "commit", "pull" e "fetch".

Quando uno sviluppatore vuole "consegnare" le proprie modifiche apportate ai documenti effettua l'operazione di "commit" contenente un titolo ed un messaggio che riassumono le modifiche apportate alle parti del progetto software.

Una volta effettuato il commit lo sviluppatore effettua l'operazione di "pull" apportando le modifiche all'intera repository, mentre gli altri sviluppatori con l'operazione di "fetch" aggiornano la propria versione della repository in locale con tutte le versioni consegnate dagli altri sviluppatori.

Considerando le statistiche [2] su Github sono presenti oltre 100 milioni di repository e l'84% delle migliori 100 aziende internazionali di sviluppo software utilizzano la piattaforma Github.

Sulla piattaforma Github lavorano oltre 73 milioni di sviluppatori di cui il 47,8% lavora per un'azienda privata, il 28% sono studenti, il 13,5% lavora in progetti open source mentre il

Figura 2.1: Logo della piattaforma Github.

5,5% lavora a progetti open source per aziende private.

La piattaforma Github tramite l'operazione di "fork" e le repository pubbliche offre grandi opportunità sia per i programmatori individuali che possono lavorare su codice sorgente di grandi aziende, sia alle stesse aziende che possono contare su sviluppatori esterni per la correzioni di errori e l'evoluzione di un progetto software.

2.2 Abbandono di un Progetto Open Source

2.2.1 Cause dell'Abbandono di un Progetto Open Source

La piattaforma Github fornisce dei grafici [figura 2.2] che mostrano la frequenza dei commit da parte degli sviluppatori su un progetto. In questi grafici spesso è possibile notare come alcuni sviluppatori riducono la frequenza del lavoro su un progetto, predendo delle pause o addirittura abbandonando il progetto.

I progetti open source spesso sono gestiti da un numero piccolo di sviluppatori senza alcun sostegno finanziario. Molti sviluppatori open source sono volontari o lavorano part-time su un progetto, la mancanza di un'equa retribuzione è una delle cause principali che porta all'abbandono di un progetto, in quanto uno sviluppatore potrebbe preferire concentrarsi sullo studio oppure su contribuire a lavori più appaganti dal punto di vista personale.

Altri fattori che possono portare all'abbandono di un progetto open source sono direttamente correlati al progetto. In un progetto può capitare che alcune mansioni non rientrano nelle competenze di un determinato sviluppatore oppure che ci siano altre parti interessate al progetto, inducendo lo sviluppatore a prendersi una pausa.

Il comportamento sociale della comunità può essere un fattore determinante, dare consigli o feedback può aiutare lo sviluppatore ad integrarsi nella comunità e contribuire in maniera più attiva al progetto. Mentre se il contributo di uno sviluppatore viene ignorato o non valorizzato dalla comunità può comportare all'abbandono dello sviluppatore.

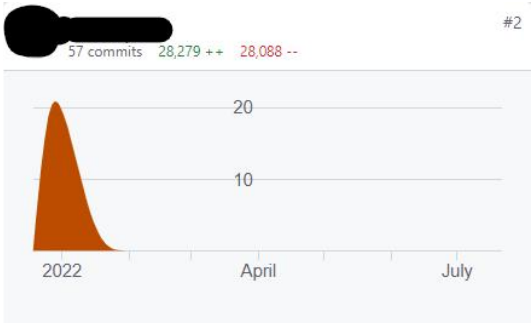

Figura 2.2: Grafico che mostra il contributo di uno sviluppatore.

2.2.2 Identificare l'Abbandono di un Progetto Open Source

L'abbandono di un progetto open source da parte di uno sviluppatore potrebbe bloccare la costruzione e l'evoluzione del progetto, specialmente se lo sviluppatore ricopriva un ruolo fondamentale nel progetto. Potrebbe essere molto utile capire a priori se uno sviluppatore sta per abbandonare un progetto, in modo da limitare i "danni" nello sviluppo del progetto.

Secondo lo studio [3] esistono alcuni segnali che permettono di capire se uno sviluppatore sta abbandonando un progetto.

Un primo segnale da analizzare riguarda l'interesse dello sviluppatore, in particolare se la frequenza di sviluppo è inconstante o subisce una drastica diminuzione, come ad esempio un calo del ritmo dei commit.

L'interesse dello sviluppatore per il progetto a cui sta lavorando è un fattore cruciale, ad esempio se uno studente lavora su un progetto per un corso accademico, è molto probabile che una volta finito il corso lo studente abbandoni il progetto.

In un progetto open source gli sviluppatori che sono risultati essenziali per l'evoluzione e lo sviluppo svolgono un ruolo fondamentale ed in molti progetti l'abbandono di uno sviluppatore esperto potrebbe rallentare o bloccare lo sviluppo progetto.

D'altra parte includere nuovi sviluppatori può comportare un'innovazione del progetto, introducendo nuove idee all'interno della comunità.

I vecchi sviluppatori, in generale, sono molto restii nell'apportare delle modifiche ad un progetto in quanto partecipano sin dall'inizio all'evoluzione del progetto, mentre i nuovi sviluppatori tendono a rivoluzionare il software per introdurre nuove funzionalità o modificare quelle esistenti.

2.2.3 Conseguenze dell'Abbandono di uno Sviluppatore

L'abbandono di uno sviluppatore potrebbe rallentare o bloccare lo sviluppo di un progetto software open source.

Un progetto open source può offrire varie funzionalità che potrebbero avere componenti in comune, se si hanno delle dipendenze tra le varie componenti e lo sviluppatore che stava implementando una specifica componente decide di abbandonare il progetto potrebbe compromettere lo sviluppo di altre componenti del progetto software, bloccando lo sviluppo ad altri implementatori.

Inoltre gli utenti che usufruiscono del software potrebbero essere soggetti a rischi in termini di privacy, in quanto una possibile conseguenza per i fruitori risulta essere quella di un'esposizione elevata ai rischi per quanto riguarda la sicurezza dei dati.

Secondo l'analisi [4] effettuata considerando 315 progetti, il 55% dei progetti open source viene abbandonato da almeno uno sviluppatore. Nel 59% dei progetti che hanno subito un abbandono, esso è avvenuto nei primi due anni, il 71% di questi progetti ha tra i quattro ed i sette anni di sviluppo. Il 41% dei progetti che hanno subito un abbandono sono "sopravvissuti" integrando nuovi sviluppatori.

Da queste statistiche possiamo capire quanto sia importante integrare nuovi sviluppatori in un progetto software.

Una soluzione all'abbandono potrebbe essere quella di integrare un nuovo sviluppatore prima che un altro abbandoni il progetto. Ma è necessario capire a priori se uno sviluppatore sta per abbandonare il progetto.

Una possibile soluzione sarebbe quella di scansionare regolarmente le repository ed i rispettivi branch, analizzando le date dei commit e le modifiche apportate recentemente.

Se si notano dei cali nei ritmi di implementazione bisogna segnalarlo allo sviluppatore stesso ed al proprietario della repository, in modo da trovare una soluzione ed eventualmente andare a sostituire lo sviluppatore se ha intenzione di abbandonare il progetto.

Questa soluzione potrebbe ridurre significamente il numero di progetti abbandonati.

2.3 Analisi del Contributo Effettivo di uno Sviluppatore

2.3.1 Non-Essential Change

In un progetto software open source vengono apportati numerosi cambiamenti da diversi sviluppatori durante il ciclo di vita del software. Questi cambiamenti possono essere identificati come "essential change" o "non-essential change".

Per essential change si intendono i cambiamenti che sono essenziali all'interno di un progetto, sono delle modifiche fondamentali per introdurre nuove funzionalità all'interno del software che senza di esse potrebbe presentare dei deficit.

I cambiamenti non essenziali sono delle modifiche a snippet di codice che non modificano il funzionamento del software ma mirano a migliorarne la parte "estetica".

Lo studio [5] ha effettuato una classificazione dei cambiamenti non essenziali, dimostrando che il 15,5% degli aggiornamenti in un progetto software riguardano non-essential change.

I cambiamenti non essenziali si dividono in:

- **aggiornamenti di tipo**: gli aggiornamenti del tipo dichiarato di una variabile non sono essenziali se il tipo dichiarato effettivo non è influenzato dall'aggiornamento.
- **estrazione di variabili locali**: gli sviluppatori possono migliorare la leggibilità del codice utilizzando variabili temporanee per memorizzare alcune informazioni.
- **refactoring di un'entità**: quando uno sviluppatore rinomina un'entità come ad esempio il nome di una classe, di una variabile o di un parametro qualsiasi istruzione di codice che fa riferimento a tale entità viene modificata testualmente come se fosse una rinomina. Se si modifica un parametro a cui viene fatto spesso riferimento verranno attribuite allo sviluppatore un numero considerevole di modifiche, anche se non ha apportato cambiamenti sul comportamento esterno del software.
- **modifiche di parole chiave banali**: gli sviluppatori possono inserire o eliminare ridondantemente istanze di alcune parole chiave, anteponendo una parola chiave ad un'entità del programma si influenza il comportamento del software solo in un numero limitato di casi.
- **rinomina di variabili locali**: gli sviluppatori possono rinominare le variabili locali solo per aumentare la leggibilità del codice. Anche se tali cambiamenti possono migliorare la qualità del codice non risultano modifiche che migliorano la natura del software.

- **spazi bianchi e documentazione**: gli spazi bianchi e le modifiche relative alla documentazione vengono già non considerati da molti strumenti di analisi delle modifiche. Anche in questo caso si fa riferimento a modifiche che possono aiutare i programmatori nella comprensione degli snippet di codice, ma queste modifiche non tangono la struttura logica del software.

Dall'analisi [5] possiamo ricavare che tra il 2,8% ed il 22,9% delle linee di codice modificate in un commit riguardano modifiche non essenziali.

Le modifiche non essenziali migliorano la leggibilità del codice e tramite anche una buona documentazione aiutano nell'interpretazione di alcune caratteristiche del software consentendo una migliore coordinazione all'interno di un team di sviluppo.

D'altra parte però risultano una parte importante ma non fondamentale per la costruzione e l'evoluzione di un progetto software, quindi potrebbe essere utile monitorare uno sviluppatore analizzando la percentuale di modifiche non essenziali apportate nel lavoro totale.

2.3.2 Refactoring

Il refactoring è tecnica che ha l'obiettivo di modificare la struttura interna di porzioni di codice senza modificare il comportamento esterno di un software.

Il refactoring è una pratica importante nei processi di sviluppo ed è applicabile grazie a strumenti appositi forniti da quasi tutti gli ide (ambienti di sviluppo).

Gli sviluppatori applicano il refactoring al codice per estendere l'usabilità ed il ciclo di vita di un progetto software, migliorandone il supporto sui vari ambienti di sviluppo.

Un obiettivo importante del refactoring è quello di rendere il codice modulare in modo da semplificare i processi di manutenibilità ed usabilità, dato che alcuni snippet di codice possono essere utilizzati per estendere le funzionalità del software o riutilizzati in altri progetti software per implementare funzionalità simili.

Il refactoring permette quindi di velocizzare i tempi dello sviluppo di un progetto software.

Esistono diverse strategie per applicare il refactoring a porzioni di codice:

- **abstract refactoring**: è una tecnica che permette di estrarre il codice per evitare linee di codice ridondanti.
- **metodo composto**: è una tecnica che semplifica la gestione del codice riducendone la complessità, consiste nell'estrarre metodi, classi, interfacce e variabili locali semplificando sia il codice che le sue successive modifiche.
- **refactoring preparatorio**: è una tecnica in cui lo sviluppatore analizza il refactoring da applicare in fase di progettazione, prima che una funzionalità venga implementata. Modificando il codice prima di aggiungerne funzionalità si semplificano le fasi di sviluppo successive del progetto software.

Lo studio [6] ha suddiviso il processo di refactoring in più fasi:

1. Identificare in quali parti del software il refactoring deve essere applicato.
2. Determinare che tipo di refactoring bisogna applicare alle parti di codice identificate.
3. Garantire che il refactoring da applicare non modifichi il comportamento esterno del software.
4. Applicare il refactoring.
5. Valutare i risultati ottenuti applicando il refactoring al progetto software.

6. Verificare la coerenza tra la documentazione del progetto ed il codice del programma.

Tramite la pratica del refactoring è possibile semplificare il lavoro degli sviluppatori che possono utilizzare snippet di codice già esistenti, velocizzando anche i processi di sviluppo di un software. Inoltre il refactoring migliora anche la qualità del software garantendo una maggiore usabilità e semplificando i processi di manutenibilità.

2.3.3 Metriche per valutare il Contributo di uno Sviluppatore

Un aspetto importante da analizzare durante le fasi di sviluppo di un progetto software open source è la valutazione del contributo degli sviluppatori che lavorano al progetto.

Le valutazioni dei contributi degli sviluppatori vengono effettuate per monitorare il tasso di sviluppo del progetto, identificare possibili congestioni nei processi di sviluppo ed identificare se uno sviluppatore ha intenzione di prendersi una pausa o abbandonare il progetto. I risultati prodotti dalle valutazioni dei contributi aiutano la pianificazione del progetto.

Il problema principale consiste nel come valutare il contributo effettivo dato che gli sviluppatori possono ricoprire ruoli diversi nelle fasi di sviluppo. Una soluzione consiste nell'utilizzare alcune metriche come standard di valutazione per stabilire il contributo effettivo apportato da uno sviluppatore.

L'analisi del contributo si divide in due parti: analisi quantitativa e qualitativa.

L'analisi quantitativa consiste nell'analizzare quanto uno sviluppatore ha lavorato sul progetto. Alcune metriche potrebbero essere il numero di commit, il numero di linee di codice inserite e rimosse, il numero di file modificati oppure il numero di issue aperte e risolte.

L'analisi qualitativa consiste nell'analizzare l'efficienza delle modifiche apportate da un singolo sviluppatore. Come metriche potremmo considerare i cambiamenti non essenziali come ad esempio, le modifiche riguardanti la documentazione, il refactoring, la percentuale di non essential-change per ogni commit all'interno del progetto software.

CAPITOLO 3

Repository Analyzer

In questo capitolo è illustrato il progetto Repository Analyzer descrivendone le caratteristiche, il funzionamento, come vengono estrapolati i dati, le metriche per stabilire il contenuto, gli obiettivi e come essi vengono raggiunti.

3.1 Obiettivi

Nel capitolo precedente sono stati analizzati alcuni articoli scientifici con l'obiettivo di investigare sulla nascita ed i benefici che la comunità open source ha apportato nello sviluppo di un progetto software, le metriche per stabilire quando un programmatore sta abbandonando un progetto open source ed alcuni metodi per stabilire il lavoro effettivo apportato da uno sviluppatore su un progetto open source.

Repository Analyzer è un progetto open source che ha come obiettivo principale quello di investigare e fornire supporto per aiutare a prevenire e risolvere le problematiche analizzate nel capitolo precedente su una qualsiasi repository pubblica di Github.

Per rendere il progetto fruibile su qualsiasi repository pubblica è necessaria una raccolta dati che non si basi su un Dataset o un archivio dati che presenterebbe dei limiti sul numero di progetti che possono essere analizzati ed inoltre dovrebbe essere aggiornata costantemente per avere dati veritieri e per analizzare progetti più recenti.

L'obiettivo principale è quello di analizzare il contributo effettivo apportato da ogni sviluppatore su un qualsiasi progetto Github open source. L'analisi del contributo si divide in analisi quantitativa ed analisi qualitativa.

L'analisi quantitativa consiste nell'analizzare la quantità di modifiche apportate da ogni singolo sviluppatore analizzando i seguenti fattori:

- **numero di commit**: tramite il numero totale di commit effettuati sull'intero progetto possiamo stimare il numero di "consegne" da parte dello sviluppatore;
- **linee di codice**: tramite il numero di linee di codice inserite sull'intero progetto è possibile ricavare il numero di cambiamenti apportati al progetto;
- **percentuale del contributo**: consiste nella percentuale di linee di codice inserite dal singolo sviluppatore rispetto al totale delle linee di codice implementate da tutto il team di sviluppo nell'intero progetto;
- **numero di "issue" aperte**: quando uno sviluppatore apre "un'issue" segnala che esiste un bug all'interno del codice, influendo positivamente sulla collaborazione del team di sviluppo.

L'analisi qualitativa consiste nell'analizzare la qualità delle modifiche apportate da ogni singolo sviluppatore e può essere ottenuta considerando i seguenti fattori:

- **media del numero di linee di codice aggiunte per il numero di commit**: tramite il numero di linee di codice inserite mediamente per ogni commit è possibile capire se uno sviluppatore ha effettuato dei commit con pochi o addirittura senza alcun cambiamento, oppure se i commit effettuati sono corposi;
- **percentuale di linee di codice**: le linee di codice sono considerate modifiche essenziali all'interno di un progetto a differenza di documentazione e spazi bianchi, stimare una percentuale delle linee di codice sui cambiamenti totali è un'ottima strategia per dare una valutazione delle modifiche apportate su un progetto;
- **percentuale della documentazione**: le modifiche riguardanti documentazione e commenti consistono in modifiche non essenziali, stimandone la percentuale sulle modifiche è possibile capire se uno sviluppatore si è dedicato a lavorare su modifiche non essenziali;
- **percentuale di spazi bianchi**: gli spazi bianchi consistono in linee di codice vuote che potrebbero "esaltare" le statistiche di uno sviluppatore ed anche esse fanno parte delle modifiche non essenziali che non giovano alcun beneficio ad un progetto.

Figura 3.1: Plugin di Repository Analyzer disponibile nel Chrome Web Store.

Un altro obiettivo del progetto Repository Analyzer è quello di analizzare se un progetto open source sta per essere o se è già stato abbandonato.

Mostrare la data di quando la repository è stata creata e la data di quando è avvenuta l'ultima modifica aiuta a capire quello che è lo stato del progetto, un'altra strategia consiste nell'analizzare i dati più recenti evitando di analizzare commit troppo datati, in modo da fornire una panoramica più attuale riguardo l'approccio degli sviluppatori rispetto al progetto e laddove fosse possibile prevenirne l'abbandono da parte di essi.

Inoltre il progetto Repository Analyzer consiste in un progetto open source che è accessibile a qualsiasi sviluppatore che può scaricare ed apportare modifiche al codice sorgente.

Google Chrome è uno dei migliori browser per la gestione delle estensioni. Composto da un design elegante che viene costantemente aggiornato ed integrato con nuove funzionalità che si traducono nell'alta velocità che il browser offre rispetto ad altri.

Chrome presenta una console per gli sviluppatori efficiente che consente di monitorare la larghezza di banda della rete, visualizzare il codice sorgente e scegliere i colori per regolare il design online.

Inoltre Chrome è il browser che presenta più estensioni e sono facilmente fruibili tramite il Chrome Web Store, un'apposita pagina che mostra ogni plugin che può essere scaricato con un click.

Il browser Chrome presenta un modello di sicurezza per le estensioni molto efficiente che secondo lo studio [7] non permette lo spam di email, offre protezione da attacchi DDoS e garantisce la sicurezza delle password.

Per facilitare l'utilizzo anche ad utenti meno esperti Repository Analyzer è stato progettato come un plugin per Google Chrome disponibile nel Chrome Web Store[figura 3.1] dove è possibile aggiungere l'estensione al browser con un click.

3.2 Design

3.2.1 Raccolta dei Dati

Una delle caratteristiche principali del progetto Repository Analyzer è quella di non avere un Dataset o un archivio dati prestabilito e fisso, infatti non è presente alcun limite sul numero di progetti che il plugin può analizzare.

Repository Analyzer è in grado di analizzare qualsiasi repository pubblica su Github grazie alla raccolta dei dati che avviene tramite Github REST API.

Le API(Application Programming Interface) sono applicazioni che espongono le funzionalità di altre applicazioni per agevolare la programmazione e l'integrazione di esse in altri progetti software, facilitando l'utilizzo di servizi resi disponibili.

I progettisti di API implementano questo codice attraverso una programmazione standardizzata che espone le funzioni necessarie per accedere alla piattaforma in questione, con l'obiettivo di semplificare l'integrazione di una o più funzionalità tra un'applicazione ed un'altra in modo da evitare ridondanze in snippet di codice.

Github REST API è un API di Github che fornisce una serie di informazioni relative ad una repository pubblica. L'API funziona tramite un meccanismo di richieste nelle quali bisogna specificare il metodo HTTP ed il percorso relativo alla repository di Github. L'API restituisce il codice di stato della risposta, intestazioni ed il corpo della risposta.

Le informazioni relative alla repository vengono restituite sotto forma di oggetto JSON, venendo quindi trattate come un vettore di stringhe[figura 3.2].

Effettuando una richiesta su una repository vengoro restituite delle informazioni e dei collegamenti accedibili tramite un'altra richiesta, in particolare:

- **name**: nome della repository;
- **description**: descrizione del progetto derivante dal file "ReadMe" della repository;
- **created_at ed updated_at**: data di creazione e di ultima modifica apportata alla repository;
- **languages_url**: collegamento tramite il quale è possibile accedere ai linguaggi di programmazione utilizzati per implementare il progetto ed ogni linea di codice inserita per ogni linguaggio di programmazione[figura 3.3];

```
{
  "id": 513481990,
  "node_id": "R_kgDOHpsdBg",
  "name": "RepositoryAnalyzer",
  "full_name": "Zengr098/RepositoryAnalyzer",
  "private": false,
  "owner": {
    "login": "Zengr098",
    "id": 81223413,
    "node_id": "MDQ6VXNlcjgxMjIzNDEz",
    "avatar_url": "https://avatars.githubusercontent.com/u/81223413?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/Zengr098",
    "html_url": "https://github.com/Zengr098",
    "followers_url": "https://api.github.com/users/Zengr098/followers",
    "following_url": "https://api.github.com/users/Zengr098/following{/other_user}",
    "gists_url": "https://api.github.com/users/Zengr098/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/Zengr098/starred{/owner}{/repo}",
    "subscriptions_url": "https://api.github.com/users/Zengr098/subscriptions",
    "organizations_url": "https://api.github.com/users/Zengr098/orgs",
    "repos_url": "https://api.github.com/users/Zengr098/repos",
    "events_url": "https://api.github.com/users/Zengr098/events{/privacy}",
    "received_events_url": "https://api.github.com/users/Zengr098/received_events",
    "type": "User",
    "site_admin": false
  },
```


Figura 3.2: Alcune informazioni restituite da REST API sulla repository del progetto Repository Analyzer.


```
{
  "CSS": 213154,
  "JavaScript": 17056,
  "HTML": 15062
}
```


Figura 3.3: Risultato della richiesta sulla sezione languages sul progetto Repository Analyzer.

- **contributors_url**: collegamento tramite il quale è possibile accedere alle informazioni di tutti gli sviluppatori che hanno contribuito sulla repository, come lo username, il numero di commit e l'immagine del profilo Github[figura 3.4];
- **commits_url**: collegamento che mostra la lista degli ultimi trenta commit, ad ogni commit è associato un codice identificativo detto "sha", tramite esso si può accedere ad un singolo commit;
- **issue_url**: collegamento che mostra la lista delle issue, per ogni issue viene mostrato il nome, la descrizione, lo stato, da quale sviluppatore sono state aperte e da quale sono state chiuse;
- **altre informazioni**: nella richiesta inoltre sono contenute altre informazioni relative alle fork ed ai branch della repository, ai commenti ed informazioni più specifiche sulla struttura del progetto.

```
[
  {
    "login": "kevinpcf",
    "id": 81223981,
    "node_id": "MDQ6VXNlcjgxMjIzOTgx",
    "avatar_url": "https://avatars.githubusercontent.com/u/81223981?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/kevinpcf",
    "html_url": "https://github.com/kevinpcf",
    "followers_url": "https://api.github.com/users/kevinpcf/followers",
    "following_url": "https://api.github.com/users/kevinpcf/following{/other_user}",
    "gists_url": "https://api.github.com/users/kevinpcf/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/kevinpcf/starred{/owner}{/repo}",
    "subscriptions_url": "https://api.github.com/users/kevinpcf/subscriptions",
    "organizations_url": "https://api.github.com/users/kevinpcf/orgs",
    "repos_url": "https://api.github.com/users/kevinpcf/repos",
    "events_url": "https://api.github.com/users/kevinpcf/events{/privacy}",
    "received_events_url": "https://api.github.com/users/kevinpcf/received_events",
    "type": "User",
    "site_admin": false,
    "contributions": 52
  },
```


Figura 3.4: Informazioni relative ad un singolo collaboratore del progetto Repository Analyzer.

Accedendo a commits_url si ottiene la lista degli ultimi trenta commit con i relativi codici identificativi(sha) e se si effettua la richiesta ad un singolo commit si ottengono ulteriori informazioni sotto forma di oggetti JSON[figura 3.5], in particolare:

- **committer**: questo oggetto JSON contiene tutte le informazioni relative allo sviluppatore che ha effettuato il commit;
- **parent**: questo oggetto JSON mostra il contenuto dei file prima che venissero apportate le modifiche da parte dello sviluppatore;
- **stats**: in questo oggetto sono presenti il numero di linee di codice totali, inserite e rimosse dallo sviluppatore effettuando il commit;
- **files**: oggetto che mostra tutti i file modificati dallo sviluppatore e per ogni file sono presenti le modifiche ed il numero di linee di codice aggiunte e rimosse.

La raccolta dei dati tramite Github REST API consente al plugin Repository Analyzer di funzionare su qualsiasi repository pubblica, tramite una serie di richieste.

L'API presenta però un limite di trentamila commit massimi in un'ora limitando l'utilizzo del plugin da parte di un utente, nell'API inotre è presente una cache interna che salva le informazioni richieste da un utente rendendo il plugin utilizzabile più volte senza alcun limite sullo stesso progetto.

```
"stats": {
  "total": 288,
  "additions": 194,
  "deletions": 94
},
"files": [
  {
    "sha": "8dc9602587b8963fb9f326a979ee42035e52998a",
    "filename": "RepositoryAnalyzer/js/informations.js",
    "status": "modified",
    "additions": 173,
    "deletions": 84,
    "changes": 257,
    "blob_url": "https://github.com/Zengr098/RepositoryAnalyzer/blob/f8a9c2
    "raw_url": "https://github.com/Zengr098/RepositoryAnalyzer/raw/f8a9c292
    "contents_url": "https://api.github.com/repos/Zengr098/RepositoryAnalyz
    "patch": "@@ -4,20 +4,20 @@ window.onload = function(){\n \n //Funzione
  },
```


Figura 3.5: Alcune informazioni relative ad un commit del progetto Repository Analyzer.

3.2.2 Calcolo delle Metriche

Fornire delle statistiche che uno sviluppatore apporta su un progetto consente di capire quanto e come lo sviluppatore si è realmente impegnato, in particolar modo se si tratta di un progetto importante.

L'analisi del contributo è di fondamentale importanza perché permette ai project manager di prendere decisioni nell'assegnare i task agli sviluppatori in modo da bilanciare equamente il lavoro ad ogni membro del team di sviluppo.

C'è bisogno di un metodo di giudizio oggettivo ed universale in grado di analizzare il lavoro compiuto da ogni singolo sviluppatore su un progetto e ciò è possibile tramite l'utilizzo di alcune metriche.

Una metrica software è uno standard per la misura di alcune proprietà del software o delle sue specifiche, sono fondamentali nel campo dell'ingegneria del software.

Le metriche permettono di migliorare il processo software, controllare l'andamento di un progetto, valutare la qualità del lavoro prodotto da uno sviluppatore, stimare lo sforzo richiesto per sviluppare un sistema software. Una metrica dovrebbe essere semplice da interpretare, oggettiva, valida e robusta.

L'analisi del contributo effettuata dal progetto Repository Analyzer si basa su alcune metriche che fanno riferimento alla quantità ed alla qualità delle modifiche apportate sul progetto.

Numero di Commit

Un commit consiste in una consegna effettuata da uno sviluppatore che ha modificato le linee di codice interne ad alcuni file del progetto.

Mostrare il numero di commit effettuati sull'intero progetto da parte di ogni singolo sviluppatore fornisce una statistica di quanto un implementatore ha lavorato sul progetto.

Linee di Codice Aggiunte

Le funzionalità di un progetto software sono implementate grazie alle linee di codice.

In un progetto abbiamo molti file ed ognuno di essi contiene caratteristiche diverse che sono tutte realizzate tramite le linee di codice inserite da un programmatore.

Questa metrica mostra il numero di linee di codice inserite da ogni sviluppatore sull'intero progetto, una statistica fondamentale per analizzare la quantità di lavoro apportata sul progetto software.

Numero di Issue Aperte

Quando su Github viene aperta un issue significa che è presente un errore in alcune parti di codice che potrebbe compromettere una o più funzionalità del progetto software e potrebbe bloccare il lavoro ad uno o più sviluppatori.

Lo studio[8] descrive quanto sia importante analizzare le issue di un progetto, in particolare lo sviluppatore che ha aperto l'issue ha dimostrato proattività andando a cercare e segnalare un errore nel progetto ed andando a migliorare la coordinazione e la comunicazione all'interno del team di sviluppo che dovrà organizzarsi per la correzione dell'errore.

Questa metrica mostra il numero totale di issue aperte da ogni sviluppatore sull'intero progetto.

Percentuale del Contributo

Questa metrica consiste nella percentuale di linee di codice inserite da un singolo sviluppatore rispetto al totale delle linee di codice inserite da tutti gli sviluppatori sull'intero progetto.

L'obiettivo è quello di quantificare l'effort sostenuto dagli sviluppatori mettendoli a paragone tramite le linee di codice inserite che rappresentano le funzionalità e caratteristiche implementate o modificate sul progetto software.

Media delle Linee di Codice Aggiunte per Ogni File

Questa metrica consiste nella media delle linee di codice inserite in rapporto con il numero di file modificati all'interno degli ultimi trenta commit.

Tramite questa statistica è possibile capire se uno sviluppatore ha effettuato commit superflui con poche o senza alcuna modifica in modo da eludere ed esaltare le statistiche che Github fornisce di base sulla repository riguardo il numero di commit.

Mentre le metriche finora analizzate facevano riferimento ad un'analisi quantitativa, l'obiettivo di questa metriche è quello di analizzare la qualità dei commit effettuati dagli sviluppatori.

Analisi Qualitativa

L'analisi qualitativa ha come obiettivo quello di valutare l'effort di uno sviluppatore non solo in base alla quantità ma andando ad analizzare se all'interno delle modifiche sono presenti cambiamenti non essenziali che alterano le statistiche riguardo il contributo apportato dagli sviluppatori.

All'interno del progetto Repository Analyzer è stato implementato un parser in grado di analizzare le linee di codice e distinguerle in cambiamenti essenziali e non essenziali. I cambiamenti non essenziali analizzati nel progetto consistono in modifiche riguardo la documentazione e gli spazi bianchi.

In particolare per ogni commit vengono analizzate tutte le linee di codice inserite all'interno di ogni file modificato tramite il parametro patch nella richiesta del commit[paragrafo 3.2.1], vengono eliminati gli spazi all'interno delle righe e possiamo avere tre casi:

- se la riga è vuota si tratta di uno spazio bianco;
- se la riga comincia con un carattere relativo ad un commento si tratta di una linea di documentazione;
- altrimenti si tratta di una linea di codice.

Per quanto riguarda le modifiche relative alla documentazione sono state considerate tutte le linee che cominciano con i caratteri "//", mentre se cominciano con i caratteri "/*" vengono considerate linee riguardo la documentazione tutte le linee fino alla chiusura del commento con i caratteri "*/". Questa strategia permette al plugin di rilevare le modifiche inerenti a documentazione per gran parte dei principali linguaggi di programmazione come Java, Javascript, C#, Springboot, C, Android Studio e tutti i framework ad essi associati.

Vengono analizzati anche i commenti che iniziano con i caratteri "<!–" e terminano con i caratteri "–>" rendendo l'analisi valida anche per i file HTML, CSS, XML ed ulteriori linguaggi per la programmazione front-end.

L'analisi qualitativa consiste in tre metriche:

- percentuale delle linee di codice che fanno riferimento a modifiche essenziali sulle linee inserite totali per ogni singolo sviluppatore;
- percentuale di linee inserite riguardanti documentazione sulle linee totali inserite per ogni singolo sviluppatore;
- percentuale di linee inserite riguardanti spazi bianchi sulle linee totali inserite per ogni singolo sviluppatore.

Le metriche considerano l'intero progetto e fanno riferimento a tutti i branch di esso con l'unico limite di analizzare soltanto gli ultimi trenta commit per la problematica già citata nel paragrafo precedente[paragrafo 3.2.1].

Lo studio[8] evidenzia come sia importante stimare le statistiche su dati più recenti in modo da avere una panoramica più aggiornata riguardo il progetto e permettendo di capire se uno o più sviluppatori hanno intenzione di abbandonarlo.

La problematica relativa al limite dei trenta commit può rivelarsi utile, che seppur rimanendo un limite del progetto, può fornire statistiche per prevenirne l'abbandono.

3.3 Deployment

Il progetto Repository Analyzer, quindi, consiste in un plugin per Google Chrome disponbile nel Chrome Web Store. L'interfaccia del plugin[figura 3.6] consiste in un form in cui incollando il link di una qualsiasi repository pubblica di Github e cliccando il bottone di submit viene fornita un'apposita pagina con le informazioni relative alla repository.

Nella pagina relativa al progetto in primis vengono mostrati il nome della repository e la descrizione[figura 3.7] in modo da permettere all'utente di capire di che tipo di progetto si tratta.

Nella stessa sezione sono presenti le date di creazione e di ultima modifica[figura 3.8] della repository in modo da fornire un'immediata panoramica sullo stato del progetto.

Scorrendo si ha la sezione "Contributors" dove per ogni sviluppatore che ha contribuito allo sviluppo del progetto è presente un riquadro con tutte le statistiche[figura 3.9], successivamente un'apposita sezione che descrive ogni metrica[figura 3.10] per facilitarne la comprensione all'utente.

Infine a fondo pagina è presente la sezione "About"[figura 3.11] che descrive il progetto e presenta un collegamento alla repository del progetto che è scaricabile e modificabile da qualsiasi utente.

Il plugin è scaricabile in un paio di click e con un interfaccia "user-friendly" che contiene spiegazioni dettagliate per ogni azione possibile rendendo la pagina accedibile anche ad utenti meno esperti.

Figura 3.6: Interfaccia del plugin.

Figura 3.7: Nome e descrizione del progetto Repository Analyzer.

Figura 3.8: Data di creazione e di ultima modifica del progetto Repository Analyzer.

Figura 3.9: Statistiche per un singolo sviluppatore del progetto Repository Analyzer.

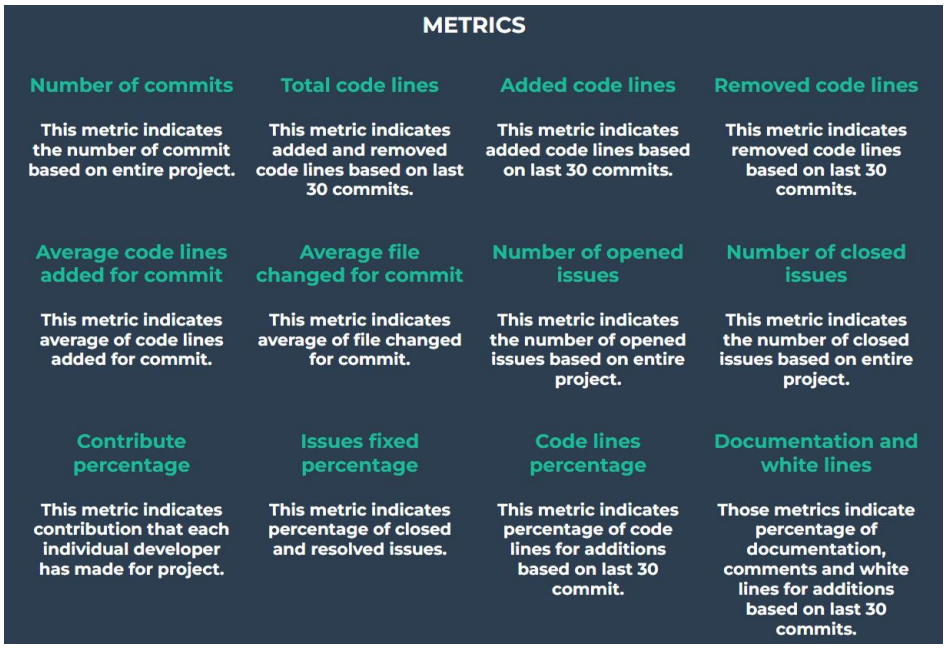

Figura 3.10: Sezione che descrive le metriche del plugin.

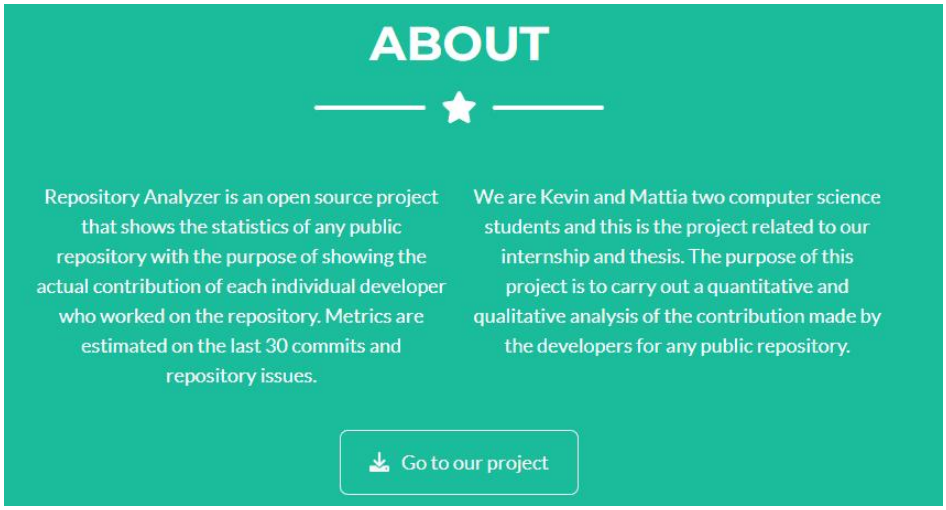

Figura 3.11: Sezione "About" del plugin.

CAPITOLO 4

Validazione

In questo capitolo vengono fatte una serie di analisi volte alla validazione del progetto Repository Analyzer paragonandone le statistiche con quelle fornite da Github.

Nei capitoli precedenti si è parlato del plugin Repository Analyzer, di come vengono raccolti i dati e di come vengono calcolate le statistiche relative all'effort apportato ad un progetto open source da ogni sviluppatore. L'obiettivo di questo capitolo è quello di valutare l'efficienza del plugin considerando le statistiche di Github come metodo di paragone. Github fornisce delle statistiche accompagnate da alcuni grafici che mostrano il contributo analizzato da ogni sviluppatore.

La sezione "insights" di Github[figura 4.1] mostra le statistiche relative ad un singolo sviluppatore basate sull'ultimo mese e le statistiche sono relative esclusivamente al branch main di ogni progetto. Inoltre le statistiche relative alle operazioni di merge ed agli account bot non vengono considerate.

Per lo stesso sviluppatore e sullo stesso progetto è possibile notare alcune differenze con le statistiche di Repository Analyzer[figura 4.2] che si occupa di analizzare le statistiche relative a tutti i branch del progetto con il numero di commit totali anche se le metriche qualitative fanno riferimento solo agli ultimi trenta commit.

Nelle statistiche Github le issue non vengono considerate un valido metodo di giudizio per l'analisi del contributto effettivo apportato da ogni sviluppatore.

La repository del progetto Repository Analyzer contiene un unico branch "main" per cui seppur presenti le differenze risultano essere lievi.

Figura 4.1: Statistiche mostrate dalla piattaforma Github per il progetto Repository Analyzer.

Figura 4.2: Statistiche mostrate dal plugin Repository Analyzer per il progetto stesso.

Ora analizziamo il progetto software Digital Donation, una piattaforma web con l'obiettivo di facilitare le operazioni di prenotazione per le sedute relative alle donazioni di sangue da parte dei volontari donatori. La repository si tratta di un progetto con un maggior numero di commit e di collaboratori che organizza il lavoro in più branch.

Se paragoniamo le figure [figura 4.3] e [figura 4.4] notiamo una differenza di 20 commit da parte dello sviluppatore ma soprattutto notiamo una differenza di quasi quattordicimila righe di codice inserite, ciò è dovuto al fatto che la piattaforma Github considera soltanto il branch "main" della repository. Mentre la maggior parte del lavoro di questo progetto risiede su altri branch, ciò dimostra la precisione e l'efficienza nel calcolare le statistiche del plugin Repository Analyzer, che però lavora soltanto sugli ultimi trenta commit.

Figura 4.3: Statistiche mostrate dalla piattaforma Github per il progetto Digital Donation.

Figura 4.4: Statistiche mostrate dal plugin Repository Analyzer per il progetto Digital Donation.

CAPITOLO 5

Conclusioni

In questo capitolo viene riassunto quanto detto nei capitoli precedenti e vengono presentati gli sviluppi futuri.

In questo lavoro di tesi si è analizzato il "fenomeno" dello sviluppo open source, analizzando la comunità open source in alcune delle sue caratteristiche essenziali e descrivendone i problemi, come l'abbandono dei progetti open source.

Sono stati raggiunti gli obiettivi di creare uno strumento in grado di identificare l'abbandono di progetti software da parte di uno sviluppatore, tramite il plugin Repository Analyzer che si occupa di analizzare il contributo effettivo di uno sviluppatore apportato ad un progetto open source.

Tutto questo è stato possibile tramite la raccolta dei dati effettuata con Github REST API ed una serie di metriche che mostrano delle statistiche chiare ed interpretabili da qualsiasi tipo di utente.

Tra i possibili sviluppi futuri potrebbe essere "abbattuto" il limite dei trenta commit ed il limite che permette ad un utente di effettuare massimo trentamila richieste in un'ora, limitandone l'utilizzo su un numero limitato di progetti.

Inoltre potrebbero essere aggiunte nuove metriche e nuove statistiche come l'analisi dei commenti ed i titoli assegnati ad un commit al momento della consegna e nell'apertura e chiusura di un issue, andando a stimare la volontà da parte di ogni singolo sviluppatore nell'aumentare la comunicazione e la coordinazione all'interno di un team di sviluppo esaltandone il morale.

Ringraziamenti

Ringrazio la professoressa Filomena Ferrucci ed il mio tutor accademico Giulia Sellitto per la disponibilità ed il grande aiuto che mi hanno fornito per la stesura di questo lavoro di tesi.

Ringrazio infinitamente i miei genitori che mi hanno sempre sostenuto nel percorso di studi, tutta la mia famiglia, gli zii ed i cugini per avermi insegnato i valori della vita, ringrazio in particolare Nonno Giovanni e Nonna Carmelina.

Ringrazio Biagio, Antonio, Alessandro, Francesco ed Anna noi ragazzi di Cerrocupo che siamo cresciuti tra una Peroni ed una partita a carte ed ovviamente ringrazio i "Cerrocupani acquisiti" Raffaele, Alessio, Fernando, Enzo, Luca e Merola per tutti i momenti indimenticabili passati insieme che mi hanno dato l'inspirazione per la scrittura di nuovi teoremi.

Ringrazio i compagni di classe: Antonio Scognamillo, Giovanni, Matteo, Elpidio, Alex, Fabio, Natale, Ilaria, Annamaria, Alessia, Leonardo, Rebecca, Mimmo e tutti gli altri, grazie a voi questi tre anni sembrano essere volati. Ringrazio in particolare Mattia collega di tutti i progetti universitari, insieme abbiamo preparato tutti gli esami e grazie alla tua compagnia ed al tuo sostegno questo percorso è stato molto più semplice e divertente.

Ringrazio i miei amici e coinquilini Andrea, Tommaso e Germano per questa convivenza all'insegna del divertimento e dell'esaurimento grazie anche a tutti gli ospiti Antonio, Christian, Teresa ed Angioletto insieme siamo stati bene ed in salute, la salute vostra però.

Ringrazio Mirko, Sica, Carrozza, Antonello e Damiano per le incredibili serate passate insieme che anche se poche si sono rivelate davvero speciali, anche se purtroppo i vari impegni personali non ci permettono di organizzarci spesso, d'altronde si sa che questa è la vita disse il giravita.

Ringrazio Carrozza, Simone, Davide, Marco, Joey, Alessio, Mari e Silvio per i bellissimi momenti passati insieme sia su Discord sia nelle serate passate insieme ma anche nelle

partite a pallone quando non vengo scammato. L'aggettivo "bellissimo" in questo caso è stato usato solo per i momenti passati insieme perchè allo stato attuale purtroppo non può essere utilizzato per il fantacalcio.

Ringrazio Domenico Antonio Gioia per il sostegno e l'aiuto che mi ha fornito come ottimo compagno di corsi e soprattutto per avermi fatto conoscere i ragazzi di Gaiano: Antonio Sica, Vincenzo, Massimo, Falcone, Pasquale, Francesco, Peppe ed il resto dell'entourage Gaianese. Insieme abbiamo passato serate indimenticabili e mi sono divertito tantissimo(tranne quando Antonio Sica aveva delle bottigliette di plastica) ed anche se è passato solo un anno mi sembra di conoscervi da moltissimo tempo, abbiamo creato un bel gruppo, tramite l'alleanza appoggiata da Alessandro, con due passioni da cui tutti noi siamo accomunati: l'invidia verso lo stile di vita di Antonio Gioia e l'hobby di fare esplodere le bombe carta.

Grazie a tutti, rimarrete per sempre impressi nel mio cuore.

Bibliografia

[1] E. Raymond, "The cathedral and the bazaar," *Knowledge, Technology & Policy*, vol. 12, no. 3, pp. 23–49, 1999. (Citato a pagina 5)

[2] Github. (2021) Github octoverse report. [Online]. Available: https://octoverse.github.com/ (Citato a pagina 6)

[3] G. Iaffaldano, I. Steinmacher, F. Calefato, M. Gerosa, and F. Lanubile, "Why do developers take breaks from contributing to oss projects? a preliminary analysis," *arXiv preprint arXiv:1903.09528*, 2019. (Citato a pagina 8)

[4] G. Avelino, E. Constantinou, M. T. Valente, and A. Serebrenik, "On the abandonment and survival of open source projects: An empirical investigation," in *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2019, pp. 1–12. (Citato a pagina 9)

[5] D. Kawrykow and M. P. Robillard, "Non-essential changes in version histories," in *2011 33rd International Conference on Software Engineering (ICSE)*. IEEE, 2011, pp. 351–360. (Citato alle pagine 10 e 11)

[6] T. Mens and T. Tourwé, "A survey of software refactoring," *IEEE Transactions on software engineering*, vol. 30, no. 2, pp. 126–139, 2004. (Citato a pagina 12)

[7] L. Liu, X. Zhang, G. Yan, S. Chen *et al.*, "Chrome extensions: Threat analysis and countermeasures." in *NDSS*, 2012. (Citato a pagina 16)

[8] T. Diamantopoulos, M. D. Papamichail, T. Karanikiotis, K. C. Chatzidimitriou, and A. L. Symeonidis, "Employing contribution and quality metrics for quantifying the software

development process," in *Proceedings of the 17th International Conference on Mining Software Repositories*, 2020, pp. 558–562. (Citato alle pagine 21 e 23)