

ALMA MATER STUDIORUM · UNIVERSITÀ DI
BOLOGNA

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

**SonarQube: uno strumento
per l'analisi statica del software**

Relatore:
Chiar.mo Prof.
Paolo Ciancarini

Presentata da:
Davide Giampaolletti

Sessione
Anno Accademico
2020/2021

Indice

1	Introduzione	1
1.1	Analisi Statica del software	3
1.1.1	Software di analisi statica	5
1.2	Perché scegliere SonarQube	6
2	La qualità	9
2.1	Debito tecnico	9
2.1.1	Riconoscere il debito tecnico	12
2.2	Cattiva qualità	13
2.3	Come ottenere un codice di buona qualità	15
2.4	Complessità cognitiva	15
2.4.1	Criteri di base	16
2.5	Come SonarQube influenza la qualità	17
3	Prevenzione dei difetti	19
3.1	Vantaggi nello scoprire i difetti il prima possibile	20
3.2	Principi di prevenzione dei difetti	21
3.2.1	Analisi dei requisiti software	22
3.2.2	Revisioni	24
3.2.3	Cause principali di difetti e determinazione di misure preventive misure preventive	25
3.2.4	Inclusione nel processo sviluppo software	27

4	Evoluzione di SonarQube	29
4.1	Versioni 1.x	29
4.2	Versioni 2.x	29
4.3	Versioni 3.x	29
4.4	Versioni 4.x	30
4.5	Versioni 5.x	30
4.6	Versioni 6.x	31
4.7	Versioni 7.x	31
4.8	Versioni 8.x	31
4.9	Versione 8.9	32
4.10	Come funziona SonarQube	32
4.10.1	Analisi con SonarQube	33
4.10.2	Dimensioni	35
4.10.3	Eventi	36
4.10.4	Descrizione	37
4.11	I sette assi della qualità	38
4.11.1	Possibili bug e regole del codifica	39
4.11.2	Test	40
4.11.3	Commenti e duplicati	41
4.11.4	Architettura, design e complessità	41
5	Esempi di uso di SonarQube	43
5.1	Stockfish	43
5.2	Counter	45
5.3	Crocodile	45
5.4	Analisi della prima versione di SonarQube attraverso SonarQube	46
6	Conclusioni e sviluppi futuri	49
A	Come installare SonarQube e le sue componenti su Windows	53
A.1	Dove scaricare SonarQube	53
A.2	Come installare il server di SonarQube	57

A.3	Come scaricare JDK	59
A.4	Configurare Sonarqube	69
A.5	Come accedere all'interfaccia di SonarQube	72
A.6	Effettuare una scansione	75
A.6.1	Installare Sonar Scanner	79
A.6.2	Eseguire la scansione	83
A.7	Installare i Plug-in	85
A.7.1	Installazione da SonarQube	86
A.7.2	Installazione manuale	87
	Bibliografia	89

Capitolo 1

Introduzione

La qualità del software viene spesso messa in secondo piano o a volte del tutto ignorata ed è per questo che bisogna evidenziare quanto sia importante. Ha un ruolo così centrale da dover essere considerata durante tutta la routine dello sviluppo software; analizziamo il perché in cinque punti:

- **Rende il software Robusto:**

L'utente finale non si sentirebbe a suo agio se ricevesse strani messaggi causati dall'utilizzo incorretto del software ed è qui che entra in gioco la robustezza, ossia la capacità di far fronte agli errori durante l'esecuzione in situazioni insolite.

Un software di alta qualità ha messaggi d'errore chiari e comprensibili per l'utente finale.

- **Aumenta l'affidabilità e la malleabilità:**

Un codice di buona qualità risulta di semplice comprensione. L'uso dei commenti, una buona indentazione, una notazione chiara e la semplicità nel flusso sono dei fattori.

Il codice di alta qualità è più facile da leggere e modificare, tutto ciò favorisce di molto il lavoro in team.

- **Rende lo sviluppo sostenibile:**

Si dice che il software sia sostenibile quando può sopravvivere nel tem-

po con modifiche minime. Si stima che una vita media del software sia di circa sei anni, ma un software di scarsa qualità non dura nemmeno la metà di questo tempo. È perché, con il costante sviluppo di programmi operativi e hardware, i cambiamenti nel software aumentano. È impegnativo e talvolta impossibile eseguire trasformazioni sulla scarsa qualità del codice.

- **Promuove la trasferibilità:**

Le pratiche di qualità del codice rendono semplice la traduzione del software tra le piattaforme. Con numerose piattaforme su cui lavorare sul software, è essenziale che la trasferibilità richieda modifiche minime.

- **Riduce il debito tecnico:**

Lo sviluppo del software è di per sé un lavoro ad alto budget e per questo ci si aspetta che il software funzioni il più a lungo possibile, con il minimo errore. Ma il software di scarsa qualità è destinato a fallire presto a meno che un numero significativo di modifiche non venga introdotto repentinamente e ripetutamente nel programma e più tardi vengono effettuate, tanto più il debito tecnico sarà aumentato. Il lavoro di sviluppo aggiuntivo richiede tempo e capitale, costi che un codice di alta qualità permette di ridurre.

Quindi è essenziale avere degli indici e degli strumenti per misurare la qualità del codice. In questo elaborato viene presentato lo strumento SonarQube, disponibile di base per ventisette linguaggi di programmazione, ma aumentabili attraverso i plug-in. SonarQube esegue un'analisi statica del programma quindi non ha bisogno di eseguire il codice per analizzarlo, è uno strumento condivisibile in grado di migliorare la qualità, la produttività riducendo difetti e code smell consigliando anche delle correzioni da effettuare. Tutto ciò favorisce la creazione di una coding etiquette tra programmatori. SonarQube risulta essere molto comodo anche per l'applicazione della metodologia agile, in quanto essendo la sua analisi molto rapida si adatta

perfettamente alle modifiche continue ad esempio attraverso la continua integrazione (pratica agile).

1.1 Analisi Statica del software

L'analisi statica del software è il processo di rilevamento di errori e difetti nel codice sorgente di un software. L'analisi statica può essere vista come un processo automatizzato di revisione del codice sorgente.

”A software quality framework tool provides different views over multiple aspects and characteristics of the source code that yield an overall quality essence. Basically, a quality framework supports analysis of the source code files. A powerful framework should also be able to analyse the results from external tools, providing a suitable enhancement mechanism for this. The result is that by integration of the own analysis algorithms and those of external tool comprehensive views over the project is provided to engineers.”

[3]

La revisione del codice è uno dei metodi più vecchi e sicuri di rilevamento dei difetti. Si tratta di una lettura attenta congiunta del codice sorgente, e di dare consigli su come migliorarlo. Questo processo rivela errori o difetti del codice che possono diventare errori in futuro. Si ritiene inoltre che l'autore del codice non debba fornire spiegazioni su come funzionano determinate parti del programma. L'algoritmo di esecuzione del programma dovrebbe essere chiaro dal testo del programma e dai commenti. In caso contrario, il codice deve essere migliorato. La revisione del codice di solito funziona bene, perché i programmatori possono notare gli errori nel codice di qualcun altro molto più facilmente che nel proprio. L'unico grande svantaggio del metodo di revisione del codice effettuata da più programmatori è un prezzo estremamente elevato: è necessario riunire diversi programmatori a intervalli regolari per rivedere un codice nuovo o dopo che sono state applicate le modifiche consigliate. I programmatori hanno anche bisogno di riposa-

re regolarmente, poichè la loro attenzione potrebbe indebolirsi rapidamente se esaminano grandi frammenti di codice alla volta, quindi non servirà a nulla nella revisione del codice. Da un lato è importante rivedere il codice regolarmente. D'altra parte, è troppo costoso. Gli strumenti di analisi del codice statico, come SonarQube, sono un buon compromesso. Possono gestire instancabilmente i testi sorgente dei programmi e dare consigli al programmatore su quali frammenti di codice dovrebbe prendere in considerazione. Naturalmente, un programma non può mai sostituire una revisione completa del codice, eseguita da un team di programmatori, ma il rapporto utilizzo/prezzo rende l'utilizzo dell'analisi statica una pratica piuttosto buona che può essere sfruttata da molte aziende.

I problemi risolti dal software di analisi del codice statico possono essere suddivisi in tre categorie:

- Rilevamento di errori nei programmi.
- Raccomandazioni sulla formattazione del codice. Alcuni analizzatori statici consentono di verificare se il codice sorgente corrisponde allo standard di formattazione del codice accettato dalla propria azienda, SonarQube utilizza delle regole comunemente riconosciute. Ciò che intendiamo con questo è il controllo del numero di rientri nei vari costrutti, l'uso di spazi/tabulazioni e così via.
- Calcolo delle metriche. Le metriche del software, sono una misura che consente di ottenere un valore numerico di alcune proprietà del software o delle sue specifiche. Ci sono molte diverse metriche che possono essere calcolate con l'aiuto di determinati strumenti come ad esempio in SonarQube il *code coverage*.

Come qualsiasi altra metodologia di rilevamento degli errori, l'analisi statica ha i suoi punti di forza e punti deboli. Non esistono metodi di test del software ideali. Metodi diversi produrranno risultati diversi per diverse classi di software. Solo la combinazione di vari metodi consentirà di ottenere la massima qualità nel software. Il vantaggio principale dell'analisi statica è

questo: consente di ridurre notevolmente i costi di eliminazione dei difetti nel software. Prima viene rilevato un errore, minore è il costo per risolverlo. Gli strumenti di analisi statica consentono di rilevare rapidamente molti errori in fase di codifica, il che riduce significativamente il costo di sviluppo per l'intero progetto. Ecco altri vantaggi dell'analizzare il codice attraverso dei software di analisi statica:

- Copertura completa del codice. Gli analizzatori statici controllano anche i frammenti di codice che ottengono il controllo molto raramente. Questi frammenti di codice in genere non possono essere testati con altri metodi. Consente di trovare difetti nei gestori delle eccezioni o nel sistema dei log.
- L'analisi statica non dipende dal compilatore in uso e dall'ambiente in cui verrà eseguito il programma compilato. Permette di trovare errori nascosti che possono rivelarsi solo pochi anni dopo essere stati creati, ad esempio, errori di comportamento non definiti. Tali errori possono verificarsi quando si passa a un'altra versione del compilatore o quando si utilizzano altre opzioni di ottimizzazione del codice. Quindi il controllo non è influenzato dal compilatore utilizzato.
- È possibile rilevare facilmente e rapidamente errori di stampa e le conseguenze dell'utilizzo di Copia-Incolla. Rilevare questi errori con altri metodi è solitamente estremamente inefficiente e una perdita di tempo e fatica. Le persone di solito non ricordano questi problemi quando discutono di errori tipici. Ma la pratica dimostra che ci vuole molto tempo per rilevarli.

1.1.1 Software di analisi statica

Esistono molti software per l'analisi statica, non tutti opensource e non tutti gratuiti.

Tra i software gratuiti posso nominare Infer Static Analyzer che supporta

Java, C, C++ e Objective-C, è stato sviluppato da ingegneri del software per Facebook, per analizzare le app android.

Un altro software gratuito ed anche opensource è semgrep che offre supporto per 17 linguaggi tra cui Go, Java, JavaScript, Json e Ruby è stato sviluppato da "Return to Corporation".

L'ultimo software gratuito ed opensource che presento è PMD (Programming Mistake Detector). PMD include set di regole integrate e supporta la possibilità di scrivere regole personalizzate. I problemi segnalati da PMD sono codici piuttosto inefficienti o cattive abitudini di programmazione, che possono ridurre le prestazioni e la manutenibilità del programma se si accumulano. Può analizzare file scritti in Java, JavaScript, Apex e Visualforce, PLSQL, Apache Velocity, XML e XSL.

1.2 Perché scegliere SonarQube

"SonarQube is one of the most frequently used open-source static analysis tools, having been adopted by more than 120K users, including more than 200K development teams and adopted by more than 100K open source projects. SonarQube analyze the code compliance against a set of rules classified as: Code Smells, i.e., issues that increase change-proneness and the related maintenance effort; Bugs, i.e., issues that will result in a fault; and Security Vulnerabilities. SonarQube also assigns one of the following five severity levels to each rule: Blocker, Critical, Major, Minor, and Info. Rules with a higher severity level have a higher impact on the system." [4]

SonarQube è una piattaforma open source sviluppata da SonarSource per il controllo della qualità del codice. Sonar esegue l'analisi statica del codice, che fornisce un rapporto dettagliato di bug, code smells, vulnerabilità, duplicazioni di codice. Supporta 27 principali linguaggi di programmazione tramite dei set di regole integrati e può anche essere esteso con vari plug-in.

Quindi ho scelto SonarQube perchè supporta un maggior numero di linguaggi rispetto ad altri scanner ed il numero di linguaggi analizzati si può

aumentare tramite vari plug-in, inoltre è molto utile anche a livello formativo in quanto allega dei suggerimenti per la risoluzione di errori e difetti. Gli sviluppatori lavorano con scadenze rigide per fornire le funzionalità richieste al cliente. Sono così importante per gli sviluppatori che molte volte compromettono la qualità del codice, potenziali bug, duplicazioni di codice e cattiva distribuzione della complessità. Inoltre, tendono a lasciare variabili, metodi e così via inutilizzati. In questo scenario, il codice funzionerebbe nel modo desiderato, ma non è il modo corretto di scrivere codice. Per evitare questi problemi, gli sviluppatori dovrebbero sempre seguire una buona pratica di codifica, ma a volte non è possibile seguire le regole e mantenere una qualità elevata per molteplici ragioni. Per ottenere l'integrazione e la distribuzione continue del codice, gli sviluppatori hanno bisogno di uno strumento che funzioni non solo una volta per verificare e comunicare loro i problemi nel codice, ma anche per monitorare e controllare il codice per verificarne la qualità di continuo. Per soddisfare tutte queste esigenze, SonarQube potrebbe essere il programma più adatto.

Nel prossimo capitolo parleremo della qualità del codice e di come SonarQube la va ad influenzare, nel terzo capitolo tratteremo la prevenzione dei difetti, nel quarto capitolo andremo ad approfondire l'evoluzione di SonarQube nelle varie versioni ed un'analisi della beta di SonarQube attraverso la versione attuale di SonarQube, nel quinto capitolo osserveremo l'evoluzione dei difetti su tre chess engine durante le ultime tre versioni di questi programmi ed infine in appendice è presente una guida all'installazione di SonarQube per Windows.

Capitolo 2

La qualità

Dati i soliti vincoli già presenti nello sviluppo del software, perché dovrebbe essere così importante lo sforzo di produrre codice di qualità? La scrittura di codice di qualità non dovrebbe essere considerata come una perdita di tempo, ma piuttosto uno degli obiettivi principali durante lo sviluppo del software; va considerato come un investimento essenziale al quale seguirà quasi immediatamente il ritorno: in sostanza un codice di alta qualità dovrebbe essere uno dei metodi più efficaci per ridurre il debito tecnico.

2.1 Debito tecnico

Il debito tecnico è un concetto dello sviluppo software che riflette il carico di lavoro aggiuntivo dovuto alla modifica di un codice implementato attraverso le soluzioni più facili nel breve periodo piuttosto che le migliori. Il debito tecnico è comunemente associato all'extreme programming una metodologia di sviluppo software che fa parte della famiglia agile, specialmente durante la pratica del refactoring: riscrivere il codice senza alterarne le capacità esterne cambiando l'architettura in modo da renderlo più semplice e generico. Detto ciò il refactoring non è solo il risultato di un codice scritto male ma va fatto basandosi sulla comprensione del problema e sul modo migliore di risolvere quel problema.

Punti fondamentali del debito tecnico [2]:

- "Technical debt reifies an abstract concept."
- "If you do not incur any form of interest, then you probably do not have actual technical debt."
- "All systems have technical debt."
- "Technical debt must trace to the system."
- "Technical debt is not synonymous with bad quality."
- "Architecture technical debt has the highest cost of ownership."
- "All code matters!"
- "Technical debt has no absolute measure—neither for principal nor interest."
- "Technical debt depends on the future evolution of the system."

Quindi il debito tecnico è una metafora che consente agli sviluppatori di spiegare la necessità di ristrutturazioni e di comunicare compromessi tecnici agli uomini d'affari. Quando noi ci assumiamo un debito tecnico scegliamo di rilasciare il nostro software più velocemente ma aumentando le spese future, poiché il debito tecnico influisce sulla nostra capacità di sviluppare un sistema. Proprio come la sua controparte finanziaria, il debito tecnico genera interessi da pagare in futuro. Ci concentreremo sugli aspetti per lo più invisibili dell'evoluzione e della manutenzione. Il debito tecnico assume forme diverse in diversi tipi di artefatti di sviluppo, come ad esempio il codice, l'architettura e l'infrastruttura di produzione. Le diverse forme di debito tecnico incidono sul sistema in modi diversi. Il codice sorgente incarna molte decisioni di progettazione e programmazione. Il codice può essere sottoposto a revisione, ispezione e analisi con controllori statici per trovare problemi di maggiore granularità: violazioni degli standard di codifica, cattiva denominazione, codice clonato, complessità del codice non necessaria e commenti

fuorvianti o errati. Molti di questi sintomi di debito tecnico sono indicati come code smells.

”Li and Shatnawi (2007) investigated the relationship between six code smells and the probability of error in classes in an industrial system, and found the presence of Shotgun Surgery to be connected with a statistically significant higher probability of defects; Yamashita (2014) investigated the relationship between code smells and the incidence of problems during software maintenance, and she recommends, based on her results, that future studies should focus on the interaction effect between code smells, between code smells and other design properties, and between code smells and the program size.” [6]

Quando un sistema accumula debito tecnico nel codice sorgente, questo tende ad ostacolare la manutenibilità in modo da rendere difficoltoso apportare correzioni al sistema quando necessario.

Non tutto il debito tecnico è associato a una cattiva qualità interna del sistema. Il debito tecnico generato dal passare del tempo e dall’evoluzione dell’ambiente circostante non è il risultato di una cattiva qualità. Un sistema può avere il miglior design possibile (o codice) al momento della creazione ma pochi anni dopo è possibile che abbia generato debito tecnico a causa di cambiamenti nell’ambiente e non perché il sistema sia peggiorato. La caratteristica costante del debito tecnico è la sua invisibilità. Il debito tecnico non è visibile al di fuori dell’organizzazione di sviluppo del sistema, è per lo più invisibile a clienti, acquirenti e utenti finali. Tutti i sistemi contenenti una grande quantità di software, indipendentemente dal loro dominio o dimensione, soffrono di alcune forme di debito tecnico che vanno ad incidere negativamente sulla loro evoluzione, se non viene gestita tempestivamente. Questa forma di debito tecnico non è atomica o monolitica, ma può essere scomposta in decine o centinaia di voci, che chiamiamo voci di debito tecnico, le quali si accumulano nel tempo. È importante evitare di confondere il debito tecnico con la sua causa: la necessità di rispettare delle tempistiche molto serrate non va considerata come debito tecnico, ma come una sua causa.

”Given the significance of technical debt, it is important to be able to measure it. A prerequisite for measuring something is to be able to quantify it. In this context, technical debt is very difficult to quantify accurately. There are two main reasons for this. First, there is no clear consensus on the factors that contribute to technical debt. Second, there is no standard way to measure these factors. It is pertinent to note that there are some recent attempts that try to quantify technical debt; however, these approaches consider only a few factors which limits the usefulness of these approaches.” [9]

2.1.1 Riconoscere il debito tecnico

Per riconoscere il debito tecnico che perlopiù è invisibile dobbiamo prima capire la sua catena di cause ed effetti:

Cause → Debito Tecnico → Conseguenze → Sintomi

E quindi per capire dove si trova il debito tecnico e in cosa consiste dobbiamo andare a ritroso, come un dottore con il suo paziente:

Sintomi → Conseguenze → **Debito Tecnico** → Cause

Si dovrebbe provare a rilevare più conseguenze del debito tecnico, forse meno visibili, guardando all'interno del sistema, e queste conseguenze alla fine indicheranno gli artefatti di sviluppo che contengono il debito. Seguendo questo percorso di analisi, La domanda sorge spontanea: ”Perché abbiamo questo debito?” Le risposte a questa domanda serviranno ad individuare le cause del debito tecnico, spesso causa principale. Grazie a SonarQube molti sintomi e conseguenze del debito tecnico ci vengono segnalati, rendendo più facile identificarlo. Spesso il debito tecnico viene generato dagli schemi mentali dei programmatori.

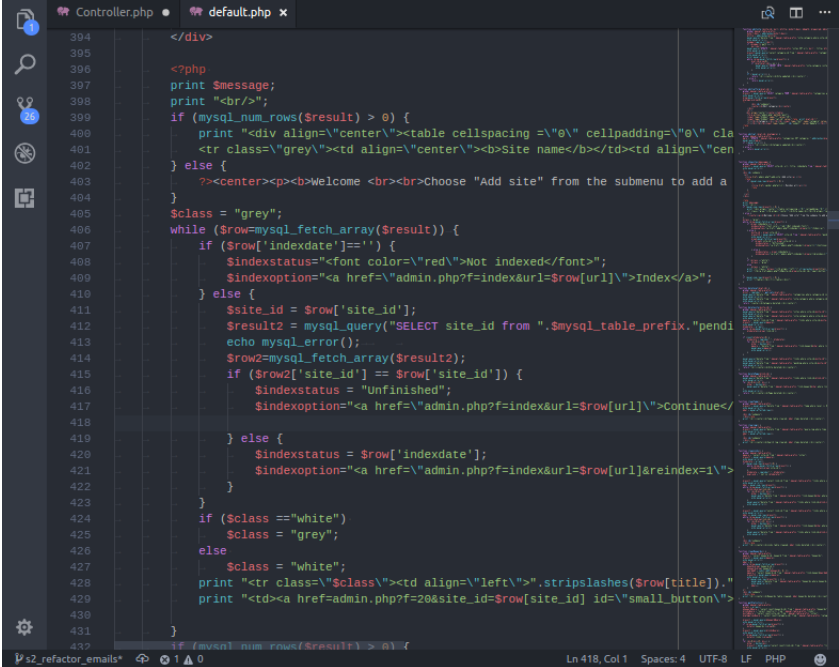
”We use the same mental processes to understand code as those we use in everyday life beyond our keyboards (evolution wasn't kind enough to equip our brains with a coding center). As we learn a topic we build mental representations of that domain. Psychologists refer to such mental models as

schemas. A schema is a theoretical construct used to describe the way we organize knowledge in our memory and how we use that knowledge for a particular event. You can think of a schema as a mental script implemented in neurons rather than code.” [11]

2.2 Cattiva qualità

Una cattiva qualità può essere causata da:

- mancanza di standard del codice;
- mancanza di documentazione;
- pessima architettura;
- metodi molto complessi.



```
394 </div>
395
396 <?php
397 print $message;
398 print "<br/>";
399 if (mysql_num_rows($result) > 0) {
400     print "<div align='center'><table cellspacing='0' cellpadding='0' cla
401     <tr class='grey'><td align='center'><b>site name</b></td><td align='cen
402 } else {
403     ?><center><p><b>welcome <br><br>Choose "Add site" from the submenu to add a
404 }
405 $class = "grey";
406 while ($row=mysql_fetch_array($result)) {
407     if ($row['indexdate']=='') {
408         $indexstatus="<font color='red'>Not indexed</font>";
409         $indexoption="<a href='admin.php?f=index&url=$row[url]'>Index</a>";
410     } else {
411         $site_id = $row['site_id'];
412         $result2 = mysql_query("SELECT site_id from ".$mysql_table_prefix."pendi
413         echo mysql_error();
414         $row2=mysql_fetch_array($result2);
415         if ($row2['site_id'] == $row['site_id']) {
416             $indexstatus = "Unfinished";
417             $indexoption="<a href='admin.php?f=index&url=$row[url]'>Continue/
418         } else {
419             $indexstatus = $row['indexdate'];
420             $indexoption="<a href='admin.php?f=index&url=$row[url]&reindex=1'>
421         }
422     }
423 }
424 if ($class == "white")
425     $class = "grey";
426 else
427     $class = "white";
428 print "<tr class='".$class.'"><td align='left'>".stripslashes($row[title])."
429 print "<td><a href=admin.php?f=20&site_id=$row[site_id] id='small_button'>
430 }
431 if (mysql_num_rows($result) > 0) {
```

Figura 2.1: Codice di cattiva qualità [17]

In questo esempio lo scopo dei metodi non può essere identificato chiaramente senza un'attenta analisi.

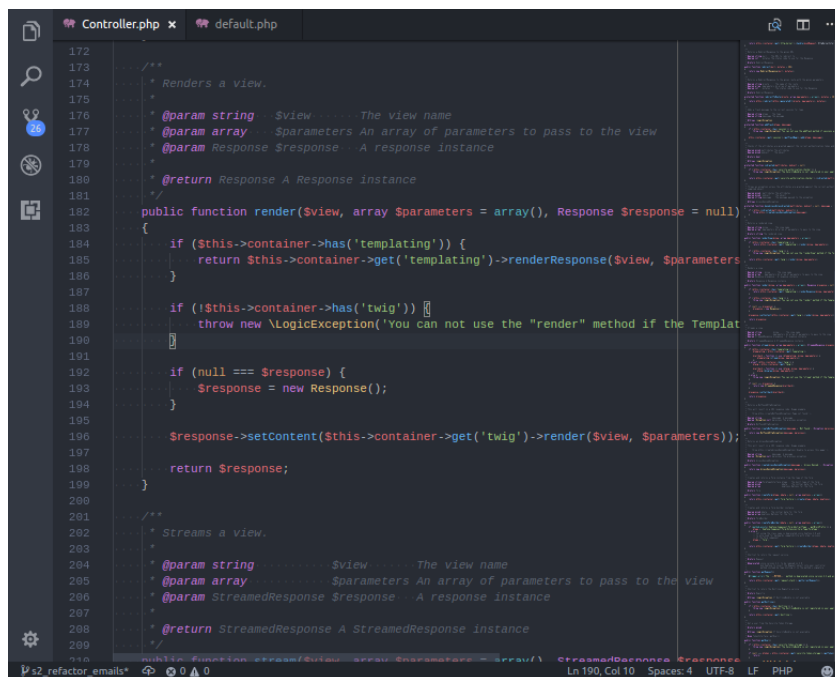


Figura 2.2: Codice di buona qualità [17]

In questa immagine invece notiamo chiaramente che:

- Il codice è di semplice comprensione;
- Logiche differenti sono separate da linee vuote;
- Ci sono pochi livelli di annidamento/indentazione con dei return immediati;
- Viene tenuto conto del design con separazione tra diverse classi/oggetti.

In un codice chiaro e di alta qualità le classi/metodi dovrebbero essere semplici da testare e da mantenere con il minimo sforzo; la probabilità che si verifichino bug è bassa.

2.3 Come ottenere un codice di buona qualità

Andiamo ora a vedere delle norme da seguire per scrivere del codice di qualità elevata:

- Usare un coding style standard appropriato per il linguaggio o il framework.
- Usare dei nomi facilmente riconoscibili in maniera consistente per metodi, classi e variabili in modo da aumentarne la leggibilità.
- Creare una documentazione efficace dove serve con commenti semplici, concisi ed efficaci.
- Effettuare il refactoring sui metodi/classi e scegliere il design corretto per promuovere la riusabilità e la estendibilità, riducendone la complessità.
- Utilizzare degli strumenti per analizzare il codice, come ad esempio SonarQube.

Quindi SonarQube interagisce direttamente solo con i punti del modello che hanno a che fare con la qualità del codice e non con i punti che ne riguardano l'utilizzo o la correttezza del codice in relazione alle funzionalità richieste.

2.4 Complessità cognitiva

La complessità cognitiva o "cognitive complexity" è una metrica che serve a misurare la difficoltà di comprensione del flusso di codice in un metodo e quindi da mantenere nel tempo attraverso aggiornamenti e modifiche. Tenere sotto controllo la complessità cognitiva attraverso SonarQube non solo è possibile, ma anche fortemente consigliato in quanto attraverso una comprensione della complessità cognitiva è possibile ridurla e migliorare come programmatori.

2.4.1 Criteri di base

- L'incremento avviene quando c'è un'interruzione nel flusso lineare (dall'alto verso il basso, da sinistra a destra) del codice.
- L'incremento avviene quando ci sono strutture annidate che interrompono il flusso.
- L'incremento non avviene quando ci sono delle strutture brevi o "shorthand" che condensano più righe in una.

Andiamo ora a vedere le differenze con la complessità ciclomatica usata spesso per misurare la complessità del codice (lo fa misurando il numero di cammini linearmente indipendenti attraverso il grafo di controllo di flusso) in modo da mettere in evidenza la differenza tra le due metriche nel campo della valutazione della mantenibilità del codice.

```
String getWords(int number) { // Cyclomatic Complexity   Cognitive Complexity
  switch (number) { // // //
    case 1: // // //
      return "one";
    case 2: // // //
      return "a couple";
    default: // // //
      return "lots";
  } // // //
}
```

Figura 2.3: Differenza tra complessità ciclomatica e cognitiva [16]

```

// Cyclomatic Complexity   Cognitive Complexity
int sumOfPrimes(int max) { // +1
    int total = 0;
    OUT: for (int i = 1; i ≤ max; ++i) { // +1
        for (int j = 2; j < i; ++j) { // +1
            if (i % j == 0) { // +1
                continue OUT; //
            }
        }
        total += i;
    }
    return total;
} // =4                               =7

```

Figura 2.4: Differenza tra complessità ciclomantica e cognitiva [16]

Come possiamo notare nella seconda immagine il codice è più difficile da comprendere rispetto alla prima, avendo delle strutture annidate, quindi la complessità cognitiva aumenta molto più rapidamente.

2.5 Come SonarQube influenza la qualità



Figura 2.5:
Modello di qualità.

Fonte: [19]

Usando come modello di qualità ISO/IEC 25010 SonarQube va a toccare i seguenti punti:

- Performance Efficacy: un codice migliore utilizza meno risorse, si esegue in minor tempo ed è quindi più efficiente
- Security: SonarQube controlla anche i problemi relativi alla sicurezza del codice, riconoscendone le vulnerabilità e segnalandole all'utente, non interagisce però con l'autenticità del software né la non ripudiabilità del software e nemmeno è in grado di controllare l'autenticità di chi entra con il proprio account nel software creato. Si occupa solo di correggerne i difetti.
- Maintainability: siccome SonarQube segnala la duplicazione del codice promuove la modularità, promuovendo la modularità promuove anche la riusabilità, il codice corretto da SonarQube più semplice da analizzare in quanto scritto in maniera migliore, più facile da modificare in quanto più facile da analizzare e quindi comprendere ed essendo più comprensibile sarà anche più facile da testare.

Gli altri punti non sono influenzati direttamente da SonarQube in quanto non riguardano la qualità del codice ma solo del processo software software per la maggior parte a livello organizzativo.

Capitolo 3

Prevenzione dei difetti

”Prevenire è meglio che curare” questo proverbio si applica alle malattie della scienza medica come ai difetti nel ciclo di vita dello sviluppo software. I difetti, come definiti dagli sviluppatori di software, sono differenze da un attributo desiderato. Questi attributi includono requisiti e specifiche completi e corretti tratti dai desideri dei potenziali clienti. Pertanto, i difetti fanno sì che il software non soddisfi i requisiti e rendano i clienti insoddisfatti. E quando un difetto passa durante il processo di sviluppo, prima viene diagnosticato, più facile ed economica è la rettifica del difetto. Il risultato finale nella prevenzione o nella diagnosi precoce è un prodotto con difetti nulli o minimi.

”Defect prevention involves a structured problem solving methodology to identify, analyze and prevent the occurrence of defects. Defect prevention is a framework and ongoing process of collecting the defect data, doing root cause analysis, determining and implementing the corrective actions and sharing the lessons learned to avoid future defects” [\[12\]](#)

Quanto sono gravi i difetti nello sviluppo del software? Negli Stati Uniti, fino al 60 % degli sviluppatori di software è coinvolto nella correzione degli errori, secondo quanto riportato da Computer Finance Magazine. Questo fatto da solo, senza considerare la qualità necessaria per soddisfare i clienti,

mostra il valore della prevenzione dei difetti del software.

3.1 Vantaggi nello scoprire i difetti il prima possibile

Il bisogno di risolvere i difetti software il prima possibile è supportato da diverse ricerche. Il National Institute of Standard Tecnology (NIST) ha pubblicato uno studio secondo il quale il tempo richiesto per risolvere un difetto in fase di produzione è di quindici ore mentre per risolvere lo stesso difetto in fase di scrittura del codice il tempo richiesto è di sole cinque ore. Il System Sciences Institute all'IBM ha segnalato che il il costo richiesto per risolvere un errore dopo che il prodotto è stato rilasciato è cinque volte più grande rispetto allo scoprire il difetto durante il design mentre è fino a cento volte più costoso se scoperto durante la fase di manutenzione.

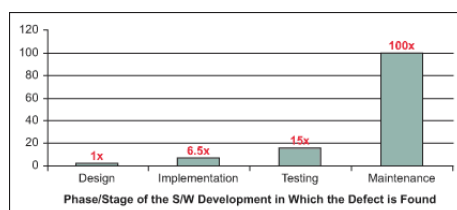


Figura 3.1: Costo di correzione dei difetti nel tempo [12]

Si possono prevenire tutti i difetti software semplicemente facendo analizzare il codice da programmi come SonarQube in modo che questi ultimi provvedano a documentare e risolvere i difetti? La risposta ovvia è no , ma questo è il primo passo da effettuare per prevenire i difetti. La prevenzione dei difetti ruota attorno a metodologie di risoluzione problemi strutturate da identificare, analizzare un modo da prevenire che si verifichino tali difetti. La prevenzione dei difetti è un framework ed un processo costante di raccolta dati analizzando la causa alla radice, nel quale si determinano e si implementano le azioni correttive e si condividono le lezioni apprese per evitare difetti futuri.

3.2 Principi di prevenzione dei difetti

Come funziona un meccanismo di prevenzione dei difetti? La risposta è in un ciclo di prevenzione dei difetti. La parte integrante del processo di prevenzione dei difetti inizia con l'analisi dei requisiti, traducendo i requisiti del cliente in specifiche del prodotto senza introdurre errori aggiuntivi. Viene progettata l'architettura del software, vengono eseguiti controlli e test del codice per scoprire i difetti, seguiti dalla loro registrazione e documentazione.

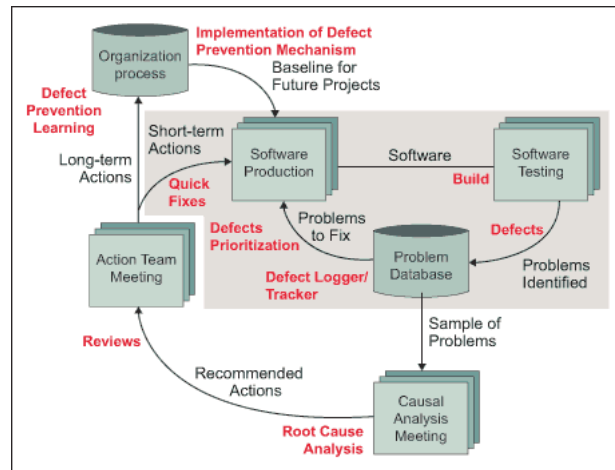


Figura 3.2: Processo di prevenzione dei difetti [12]

Le azioni e i processi nei blocchi di colore grigio rappresentano la gestione dei difetti all'interno della filosofia esistente della maggior parte dell'industria del software: rilevamento dei difetti, tracciamento/documentazione e analisi dei difetti per arrivare a soluzioni rapide a breve termine. Sullo sfondo bianco si trovano i processi che costituiscono parte integrante della metodologia di prevenzione dei difetti. Il processo vitale della metodologia di prevenzione dei difetti è analizzare i difetti per ottenere le loro cause alla radice, per determinare una soluzione rapida e un'azione preventiva. Queste misure preventive, dopo il consenso e gli impegni dei membri del team, sono incorporate nell'organizzazione come base per i progetti futuri. La metodologia mira a fornire all'organizzazione una soluzione a lungo termine e la maturità per

imparare dagli errori. La maggior parte delle attività della metodologia di prevenzione dei difetti richiedono un facilitatore. Il facilitatore può essere il leader del progetto di sviluppo del software o qualsiasi membro del team. Il coordinatore designato per la prevenzione dei difetti è attivamente coinvolto nella guida degli sforzi di prevenzione dei difetti, facilitando le riunioni e la comunicazione tra il team e la direzione e consolidando le misure / linee guida per la prevenzione dei difetti.

Vi sono cinque attività principali atte a prevenire i difetti, andiamole a vedere.

3.2.1 Analisi dei requisiti software

Gli errori nei requisiti del software e nei documenti di progettazione del software sono più frequenti degli errori nel codice sorgente stesso, secondo Computer Finance Magazine. I difetti introdotti durante i requisiti e la fase di progettazione non solo sono più probabili, ma sono anche più gravi e più difficili da rimuovere. Gli errori front-end nei requisiti e nella progettazione non possono essere trovati e rimossi tramite i test, ma necessitano invece di revisioni e ispezioni pre-test.

Table 1: Division of Defects Introduced into Software by Phase	
Software Development Phases	Percent of Defects Introduced
Requirements	20 Percent
Design	25 Percent
Coding	35 Percent
User Manuals	12 Percent
Bad Fixes	8 Percent

Figura 3.3: Percentuali di difetti introdotti durante le varie fasi del processo [12]

La tabella mostra i difetti introdotti durante le diverse fasi del ciclo di vita dello sviluppo del software.

Dagli studi effettuati da varie comunità di sviluppo software, è evidente che la maggior parte dei guasti nei prodotti software sono dovuti a errori nei requisiti e nelle fasi di progettazione fino al 64% dei costi totali dei difetti, secondo Crosstalk, il Journal di Defense Software Engineering.

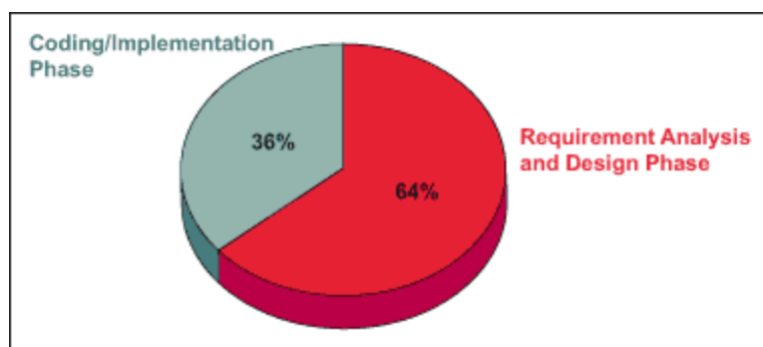


Figura 3.4: Percentuali di implementazione dei difetti [12]

Pertanto, è importante disporre di un adeguato processo di analisi dei requisiti per garantire che le esigenze del cliente siano tradotte correttamente nelle specifiche del prodotto. Più sessioni di analisi con il cliente possono essere di grande aiuto per verificare la comprensione dello sviluppatore sui requisiti effettivi.

3.2.2 Revisioni

L'auto-revisione è una delle attività più efficaci per scoprire i difetti che possono essere successivamente scoperti da un team di test o direttamente da un cliente. La maggior parte delle organizzazioni di software sta ora rendendo questo una parte delle "coding best practice" e ciò contribuisce ad aumentare la qualità dei propri prodotti. Spesso, un'auto-revisione del codice aiuta a ridurre i difetti relativi alle implementazioni degli algoritmi, alla logica errata o ad alcune condizioni mancanti. Una volta che lo sviluppatore si sente pronto con il proprio, un controllo su cosa fa rispetto a ciò che dovrebbe fare, completerebbe l'auto-revisione. La revisione tra pari è simile all'auto-revisione in termini di obiettivo: l'unica differenza è che è un pari (qualcuno che comprende molto bene la funzionalità del codice, solitamente un collega) che esamina il codice. Il vantaggio è quello di un "opinione nuova e fresca".

3.2.3 Cause principali di difetti e determinazione di misure preventive

Dopo che i difetti sono stati registrati e documentati, il passaggio successivo è analizzarli. Generalmente il coordinatore designato per la prevenzione dei difetti o il leader del progetto di sviluppo facilita un incontro per esplorare le cause profonde. L'analisi della causa principale di un difetto è guidata da tre principi chiave:

- Ridurre i difetti per migliorare la qualità: l'analisi dovrebbe portare all'implementazione di modifiche nei processi che aiutano a prevenire i difetti e garantiscono una loro veloce scoperta.
- Applicazione dell'esperienza locale: le persone che capiscono veramente cosa è andato storto sono le persone presenti quando sono stati inseriti i difetti: i membri del team. Possono dare i migliori suggerimenti su come evitare tali difetti in futuro.
- Individuazione degli errori sistematici: potrebbero esserci molti errori o difetti da gestire in un forum di analisi di questo tipo; tuttavia, alcuni errori tendono a ripetersi. Questi errori sistematici rappresentano gran parte dei difetti riscontrati nel tipico progetto software. Identificare e prevenire errori sistematici può avere un grande impatto sulla qualità (in termini di difetti) per un investimento relativamente piccolo.

Con queste linee guida, i difetti vengono analizzati per determinarne l'origine. Una raccolta di tali cause aiuterà ad effettuare l'analisi della causa principale. I difetti sono classificati in base alla loro tipologia. Un grafico di Pareto è preparato per mostrare la categoria di difetti con la più alta frequenza di occorrenza.

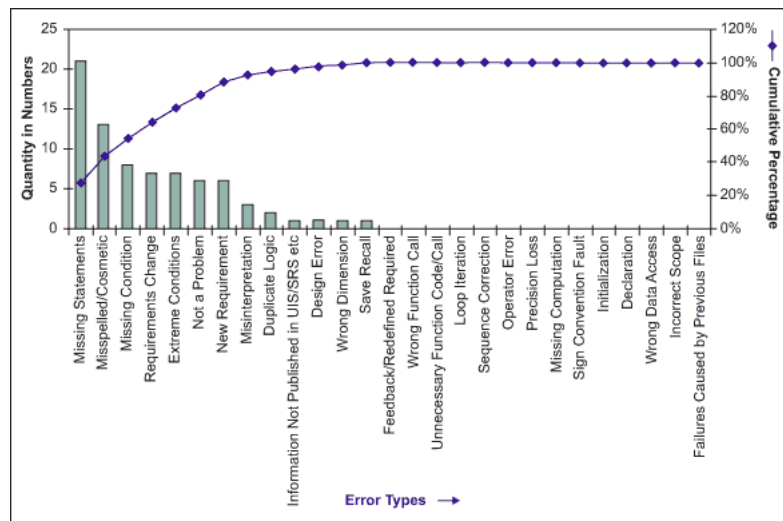


Figura 3.5: Diagramma di Pareto [12]

L'analisi della causa principale è il processo di individuazione ed eliminazione della causa, che impedirebbe il ripetersi del problema. Trovare le cause ed eliminarle sono operazioni importanti allo stesso modo. Ciascuna categoria di difetto e le cause che determinano tali difetti possono essere rappresentate utilizzando un diagramma causa-effetto.

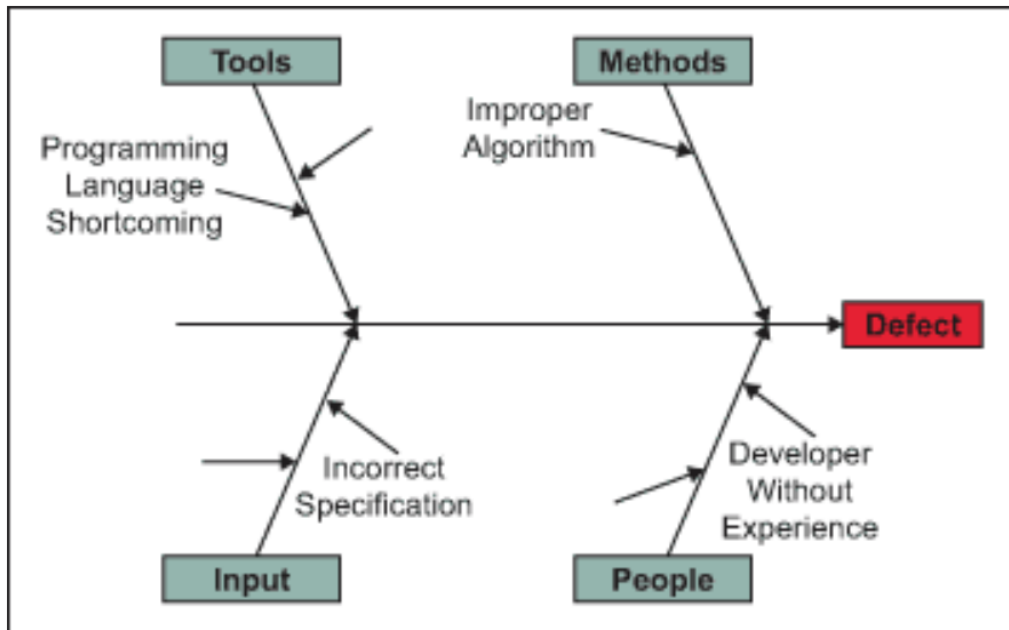


Figura 3.6: Diagramma causa-effetto [12]

Il diagramma causa-effetto, noto anche come diagramma a lisca di pesce, è una tecnica grafica per l'ordinamento e la correlazione dei fattori che contribuiscono a una determinata situazione. Un team di solito sviluppa il diagramma causa-effetto in una sessione di brainstorming facilitata. Una volta documentate le cause profonde, trovare modi per eliminarle richiede un altro round di brainstorming. L'obiettivo è determinare quali modifiche devono essere incorporate nei processi in modo da ridurre al minimo la ricorrenza dei difetti.

3.2.4 Inclusione nel processo sviluppo software

L'implementazione è la più difficile di tutte le attività di prevenzione dei difetti. Richiede un impegno totale da parte del team di sviluppo e della direzione. Viene elaborato un piano d'azione per l'implementazione della modifica dei processi esistenti o l'introduzione di nuovi con il consenso della direzione e del team. Alcune delle attività sono:

- Effettuare un aggiornamento mensile sull'analisi dei difetti da parte del team.
- Implementare le misure di prevenzione dei difetti.
- Imparare dai progetti precedenti per implementare dei controlli più efficaci.
- Monitorare i progressi nella prevenzione dei difetti.

Infine, la prevenzione dei difetti non è un esercizio individuale ma uno sforzo di squadra. Il team di sviluppo software dovrebbe sforzarsi di migliorare il proprio processo identificando tempestivamente i difetti, riducendo al minimo i tempi di risoluzione e quindi riducendo i costi del progetto.

Capitolo 4

Evoluzione di SonarQube

In questo capitolo espongo l'evoluzione di SonarQube negli anni esponendo i dettagli fra le Versioni maggiori.

4.1 Versioni 1.x

La prima versione di SonarQube è stata rilasciata il 14/12/2007 come sonar-1.0.2 le due versioni precedenti: sonar-1.0 e sonar-1.0.1 sono parte della beta.

4.2 Versioni 2.x

In sonar-2.0 rilasciato il 10/3/2010 viene migliorato il funzionamento del controllo della programmazione ad oggetti, introducendo le metriche di misurazione di Chidamber e Kemerer.

4.3 Versioni 3.x

In sonar-3.0 rilasciato il 14/5/2012 rispetto a sonar-2.0 sono stati introdotti numerosi cambiamenti di seguito sono elencati i principali:

- 20/05/2010 Aggiunte nuove regole per trovare e segnalare metodi non utilizzati in Java
- 22/10/2010 Aggiunta la possibilità di personalizzare regole riguardanti il codice.
- 30/11/2011 Migliorate regole per trovare e segnalare codice duplicato.
- Un generale miglioramento della sicurezza sotto vari aspetti e l'inserimento estensioni.

4.4 Versioni 4.x

In sonar-4.0 rilasciato il 7/11/2013 rispetto a sonar-3.0 sono stati introdotti vari cambiamenti, i principali sono i seguenti:

- 03/10/2012 Introduzione delle regole per il controllo del debito tecnico.
- 21/11/2012 Implementazione della possibilità di scansionare progetti con linguaggi misti.
- 08/01/2013 Aggiunta la possibilità di confrontare progetti.
- Un generale miglioramento del motore di ricerca grazie all'implementazione di filtri e moduli.

4.5 Versioni 5.x

In sonar-5.0.1 rilasciato il 24/2/2015 rispetto a sonar-4.0 sono stati introdotti vari cambiamenti, i principali sono il rinnovo dell'interfaccia grafica ed è stato introdotto il concetto di quality gate per indicare la qualità del codice tollerata invece di utilizzare gli alert. Tutte le versioni sono disponibili nel seguente [Link](#)

4.6 Versioni 6.x

In sonar-6.0 rilasciato il 04/08/2016 rispetto a sonar-5.0.1 sono stati introdotti i seguenti cambiamenti:

- 02/11/2015 Risolti dei bug riguardanti la gestione dei database e implementazione delle notifiche.
- 03/01/2016 Aggiornata l'interfaccia grafica e inserito il meccanismo di token d'accesso.
- 03/06/2016 Miglioramento dell'api implementata, migliorata la scalabilità e la sicurezza e sono stati implementati nuovi modelli per il controllo della qualità.

4.7 Versioni 7.x

In sonar-7.0 rilasciato il 02/02/2018 rispetto a sonar-6.0 sono stati introdotti i seguenti cambiamenti:

- 12/04/2017 Aggiunti OHP, Python e Flex ai linguaggi supportati.
- 02/06/2017 Migliorata la distinzione tra progetto pubblico e privato ed il meccanismo di filtraggio riguardo ciò.
- Molti bug risolti e un generale aumento delle regole per il controllo della qualità.

4.8 Versioni 8.x

In sonar-8.0 rilasciato il 16/10/2019 rispetto a sonar-7.0 sono stati introdotti i seguenti cambiamenti:

- 19/06/2018 Introdotta la possibilità di scannerizzare progetti in Go e aggiunto il rilevamento delle SQL injection.

- 13/08/2018 I linguaggi Kotlin e Css vengono introdotti.
- 29/10/2018 Ruby è ora un linguaggio supportato.
- 20/12/2018 Aggiunti Scala e Apex ai linguaggi supportati, migliorate le regole analisi.
- 16/10/2019 GitLab viene integrato in SonarQube.

4.9 Versione 8.9

In sonar-8.9 rilasciato il 04/05/2021 rispetto a sonar-8.0 sono stati introdotti i seguenti cambiamenti ai quali per l'aggiornamento alla versione 8.9 ho assistito al webinar organizzato da SonarQube :

- 30/04/2020 Aggiunte nuove regole per Python e migliorati i controlli sulla sicurezza delle app scannerizzate.
- 25/02/2021 Introdotta il supporto per i servizi Bitbucket Cloud e Azure DevOps.
- 01/04/2021 Migliorata la scansione di vulnerabilità lato server. Infine in sonar-8.9 abbiamo un upgrade delle funzionalità di revisione del progetto scannerizzato: uno scan più rapido, i difetti vengono evidenziati e si possono analizzare singolarmente con dei consigli passati per la correzione di difetto in difetto e migliorate le interazioni con GitHub.

4.10 Come funziona SonarQube

SonarQube valuta il codice rispetto a una serie di regole chiamate *profili di qualità*. I profili possono essere impostati sui valori predefiniti globali o possono essere configurati in modo univoco per un linguaggio o un progetto specifici. I livelli di gravità mostrano quanto sia importante la regola infranta: SonarQube fornisce suggerimenti per correggere i problemi riscontrati.

SonarQube classifica inoltre il codice in base a una serie di criteri chiamati "Quality Gates". Queste metriche possono essere configurate in base al profilo di qualità, per progetto o impostate sui valori predefiniti globali. Le impostazioni predefinite globali includono manutenibilità, affidabilità, sicurezza, copertura del codice e righe duplicate. Inoltre, SonarQube traduce queste metriche e statistiche non descrittive sul codice in valori aziendali reali, come il rischio e il debito tecnico, facendo risaltare SonarQube tra strumenti di ispezione del codice simili.

4.10.1 Analisi con SonarQube

Durante l'analisi, i dati vengono richiesti al server, i file forniti all'analisi vengono analizzati e i dati risultanti vengono restituiti al server alla fine sotto forma di report, che viene quindi analizzato in modo asincrono lato server. I report di analisi vengono messi in coda ed elaborati in sequenza, quindi è del tutto possibile che per un breve periodo dopo che il registro di analisi mostra il completamento, i valori aggiornati non siano visibili nel progetto SonarQube. Tuttavia, verrà aggiunta un'icona nella home page del progetto a destra del nome del progetto. Passandoci sopra con il mouse si hanno maggiori dettagli.

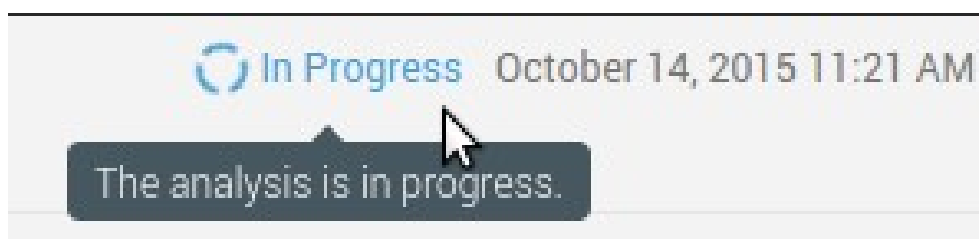


Figura 4.1:

Fonte: [16]

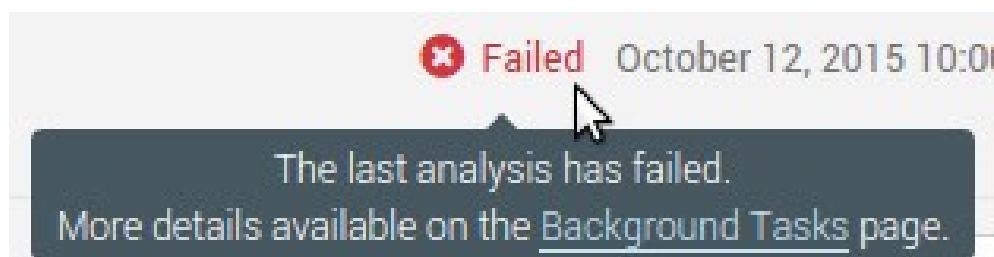


Figura 4.2:

Fonte: [16]

C'è molto da tenere d'occhio nella dashboard predefinita (altre dashboard sono disponibili), e non esiste un modo migliore per rappresentare i dati ma ci sono solo delle preferenze. Ogni casella qui è chiamata *widget*. In generale, i widget sono rappresentazioni mirate di un singolo aspetto della qualità del codice.

I widget nella dashboard predefinita mostrano tre tipi di metriche. Il primo più è basso meglio è. Il secondo più è alto e meglio è. E il terzo varia a seconda della tua prospettiva, ma che è solo un rapporto di valore neutrale dello stato attuale.

”The first kind is like a golf score; lower is better. The second is like bowling; high score wins. And the third is like age, which could go either way depending on your perspective, but which is just a value-neutral report of the current state.”

[1]

SonarQube non ha come punto principale il benchmarking, ma migliorare la qualità del codice giorno per giorno, per questo è anche un ottimo programma per la didattica. Ora diamo un'occhiata ad alcune metriche

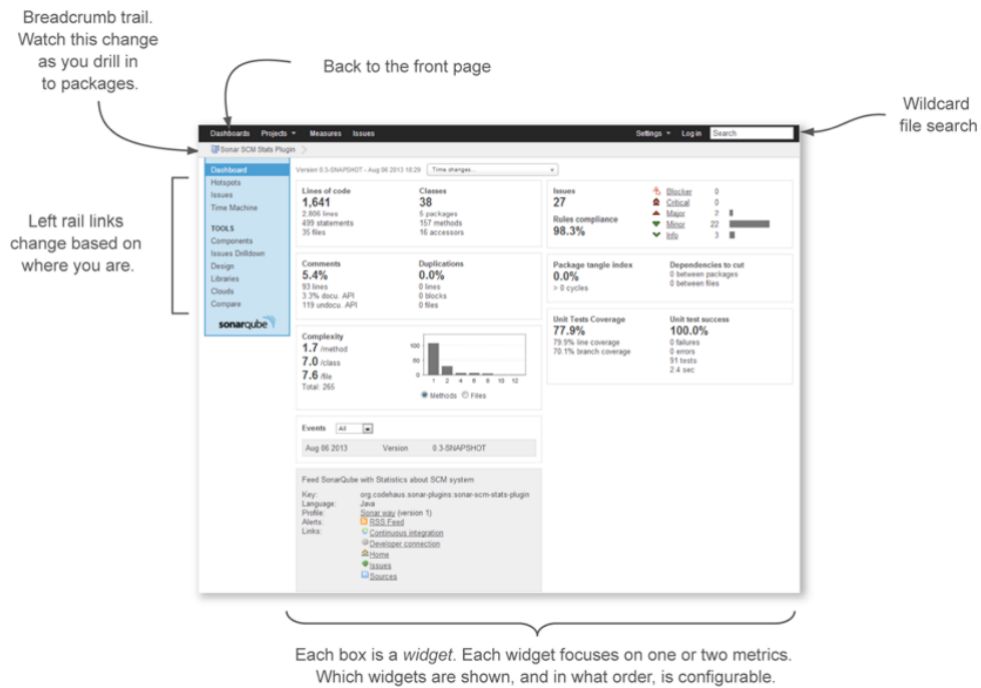


Figura 4.3: Dashboard dell' analisi[1]

Il widget delle metriche in alto a sinistra rientra nella categoria neutra. Ti dice quante righe di codice, metodi, classi e pacchetti sono stati trovati durante l'analisi.

4.10.2 Dimensioni

Il widget che rappresenta le dimensioni in alto a sinistra rientra nella categoria neutra. Rappresenta le righe di codice, metodi, classi e pacchetti trovati durante l'analisi:

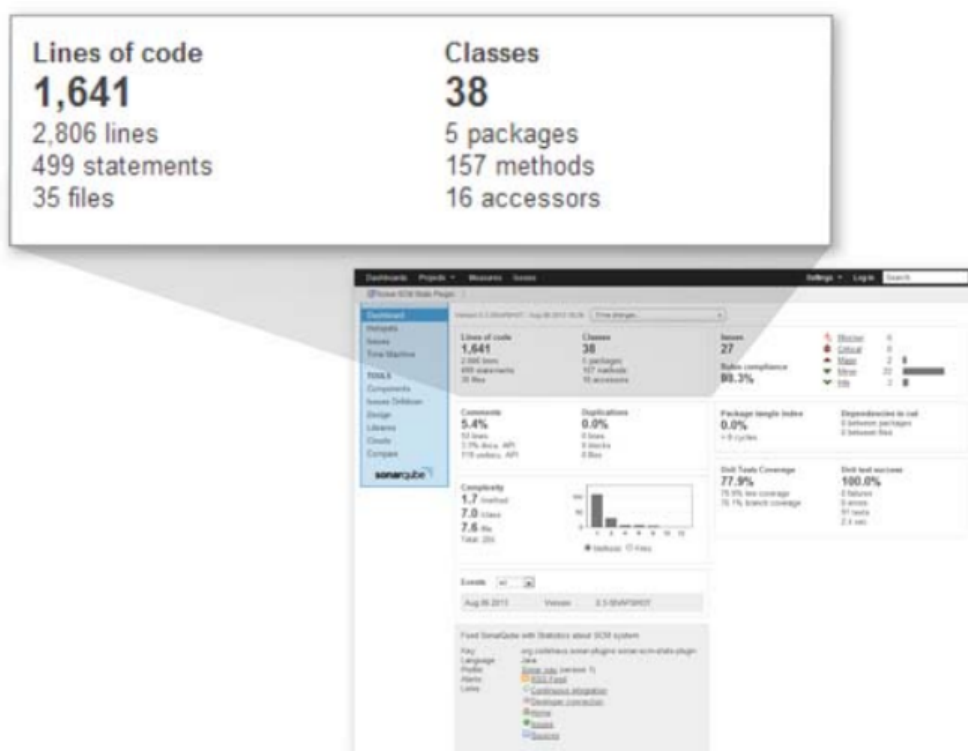


Figura 4.4: Il widget delle metriche mostra le righe di codice, i metodi, le classi ed i pacchetti che sono stati trovati durante l'analisi [1]

Dovremmo fare una distinzione a questo punto tra le righe di codice (spesso indicate come LOC) e linee fisiche, che riporta anche SonarQube. Il numero di linee fisiche nel progetto è un conteggio del numero di volte in cui qualcuno preme il tasto Invio, a prescindere che ci sia del codice sulla linea. Le righe di codice, d'altra parte, sono un conteggio del numero di linee "funzionanti" nel progetto. La definizione LOC è specifica del linguaggio, ma SonarQube le calcola per sottraendo commenti e righe vuote dalle righe fisiche.

4.10.3 Eventi

Anche i due widget nella parte inferiore della colonna di sinistra sono neutri. Secondo dal in basso, il widget degli eventi ti offre un rapido elenco

degli eventi registrati nel progetto. Esistono diversi tipi di eventi, uno dei quali è una modifica alla stringa di versione del progetto che viene passata all'analisi. Gli eventi sono significativi, almeno in parte perché contrassegnano un'istantanea di analisi per la qualità a lungo termine. Ogni volta che viene eseguita un'analisi, viene visualizzata un'istantanea dello stato del progetto. Ciò potrebbe rapidamente aggiungere molte istantanee, gonfiando il database, ma le rigorose routine di pulizia di SonarQube impediscono che ciò accada. Quelle routine sono preconfigurate per valori predefiniti standard ma regolabili successivamente.

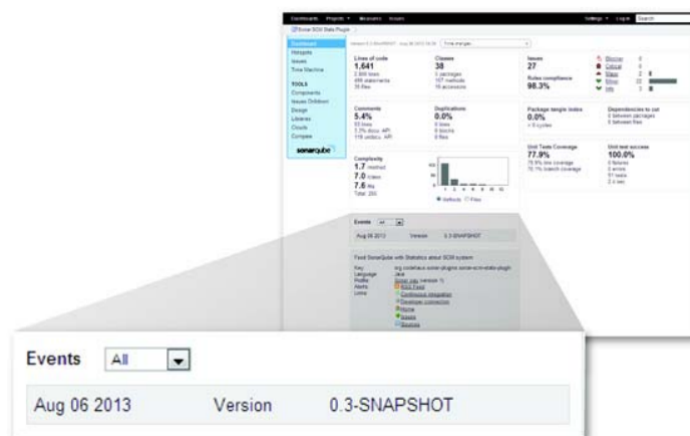


Figura 4.5: Widget eventi [1]

4.10.4 Descrizione

Il widget di descrizione in basso a sinistra è un breve curriculum vitae del progetto, che mostra il linguaggio e l'ID impostati durante l'analisi. Mostra anche il nome del set di regole, oppure il profilo, applicato all'ultima analisi. Infine, il widget di descrizione termina con un collegamento a un feed RSS di eventi di avviso(alert) sul progetto. Abbiamo detto prima che ci sono diversi tipi di eventi. Un tipo di evento si riferisce alle soglie di avviso che è possibile impostare su un profilo delle regole. Quando quelle soglie sono attraversate in entrambe le direzioni, viene generato un avviso.

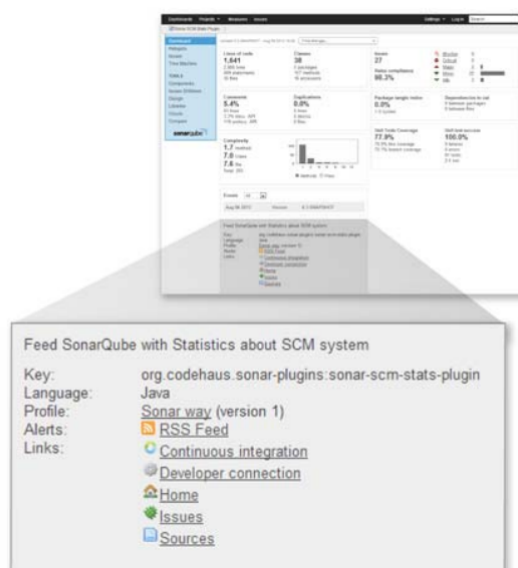


Figura 4.6: Widget descrizione [1]

4.11 I sette assi della qualità

Il resto dei widget sulla dashboard predefinita si riferisce a ciò che i creatori di SonarQube chiamano i sette assi della qualità (e talvolta i sette peccati capitali degli sviluppatori). Innanzitutto, cerchiamo di essere chiari: assi qui è il plurale di asse. Un asse è una linea rispetto alla quale misuri la distanza o l'altezza. Gli assi su cui SonarQube misura un progetto sono i seguenti:

- Possibili bug.
- Regole del codifica.
- Test.
- Duplicati.
- Commenti.
- Architettura e design.

- Complessità

Nelle sezioni successive andiamo ad approfondire i punti sopraccitati.

4.11.1 Possibili bug e regole del codifica

Il widget dei problemi segna sia i possibili bug che le regole del codifica. I creatori della lista di SonarQube hanno pensato ai potenziali bug ed regole di codifica come assi separati, ma per la segnalazione li raggruppa insieme sotto il widget dei problemi. In generale, si può considerare il conteggio dei problemi come un indicatore di qualità in ritardo; mostrano ciò che è già andato storto. Nel loro insieme, i potenziali bug e le infrazioni alle regole di codifica abbracciano un continuum. Alcuni problemi sono peggiori di altri, ecco perché SonarQube li classifica con diversi livelli di gravità: Blocker, Critical, Major, Minor e Informazioni. Viene fornita la percentuale di conformità alle regole visibile in basso a sinistra nel widget dei problemi.

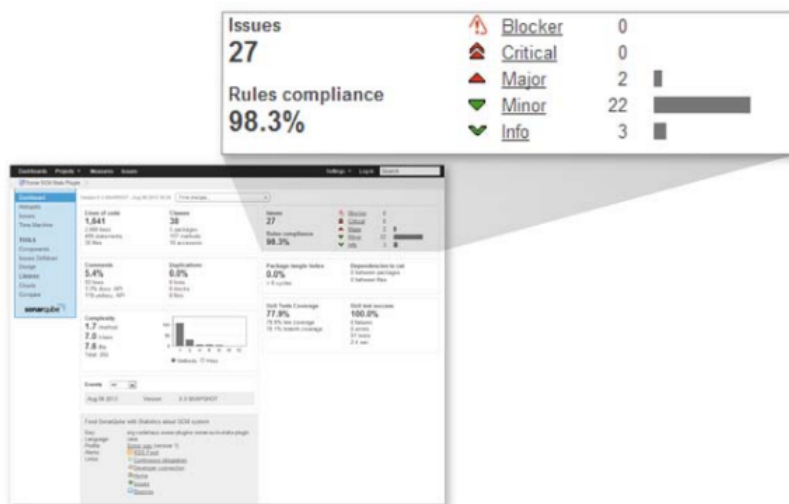


Figura 4.7: Widget dei problemi [1]

Si basa sul numero e sulla gravità dei problemi rispetto alle righe di codice nel progetto. Considerando che il conteggio dei problemi più è basso e meglio è, l'indice al contrario più ha un punteggio alto e meglio è.

4.11.2 Test

Il prossimo nell'elenco è la copertura del test dell'unità, ogni unità di lavoro nel programma dovrebbe idealmente essere bilanciata da un test che verifica che funzioni correttamente. Il widget di copertura del test, mostra l'equilibrio dell'equazione di copertura e se i test sono stati superati, falliti o se sono presenti errori. Le percentuali in questo widget sono metriche che più sono elevate e meglio è, i conteggi di errori e fallimenti sono invece il contrario. Il conteggio e la durata del test sono neutri. Tutte le metriche qui sono indicatori principali.

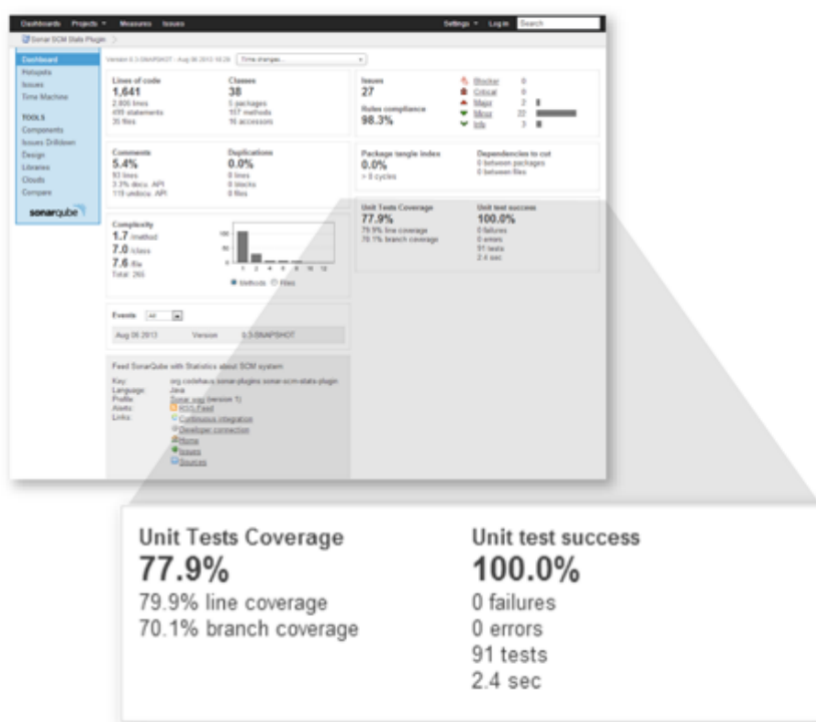


Figura 4.8: Widget dei test [1]

4.11.3 Commenti e duplicati

Il widget dei commenti e duplicati è un altro due per uno. Più commenti ci sono e meglio e meno codice duplicato c'è meglio è. Entrambe sono metriche di qualità importanti.

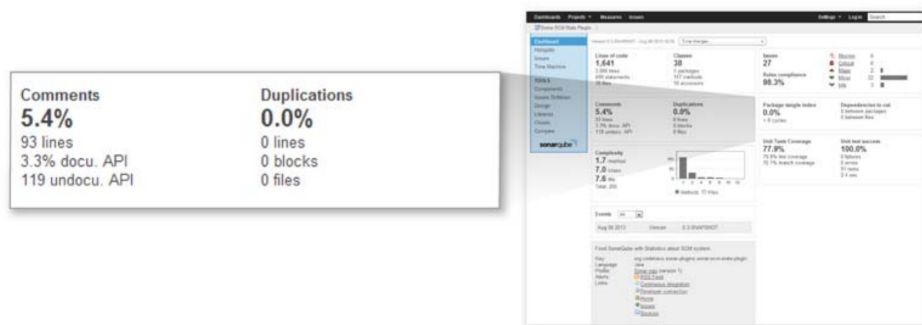


Figura 4.9: Widget dei commenti e dei duplicati [1]

4.11.4 Architettura, design e complessità

Winston Churchill disse: "Per quanto bella sia la strategia, ogni tanto dovresti guardare ai risultati". Non stava parlando di qualità del software, ma avrebbe potuto. Ciò che SonarQube misura è la comprensibilità attraverso annidamenti del codice ed attraverso la complessità cognitiva che abbiamo trattato nel capitolo 2.4. La complessità cognitiva è migliore se bassa.

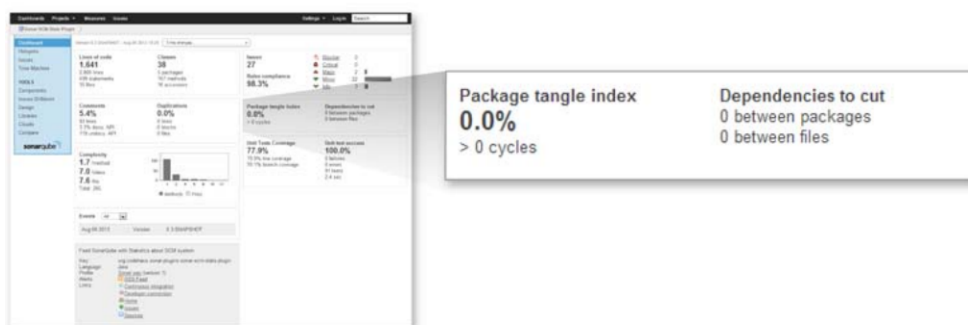


Figura 4.10: Widget dell'architettura e del design [1]

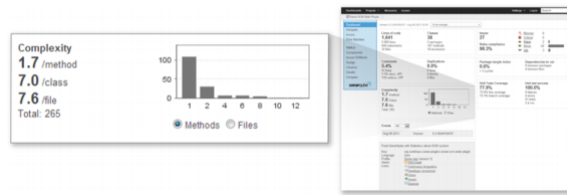


Figura 4.11: Widget della complessità [1]

Capitolo 5

Esempi di uso di SonarQube

Ho effettuato l'analisi di tre progetti *chess engine* open source in tre versioni differenti dello stesso progetto per monitorare l'evoluzione dei difetti attraverso SonarQube.

5.1 Stockfish

Il primo progetto analizzato è stato Stockfish.



Figura 5.1: Fonte: [20]

Stockfish è un chess engine gratuito e open source disponibile in più piattaforme sia desktop che mobile. Stockfish è classificato come il chess engine più forte al mondo avendo vinto ai Thoresen Chess Engines Competition, i campionati mondiali non ufficiali di chess engine, nel 2014, 2016, 2018, 2020, 2021. Essendo scritto in C++ per analizzarlo con la versione open source di SonarQube è stato necessario installare il seguente plug-in: [sonar-cxx](#). Ho analizzato Stockfish 11, Stockfish 12 e Stockfish 13 chiamandoli rispettivamente SF11, SF12 e SF13

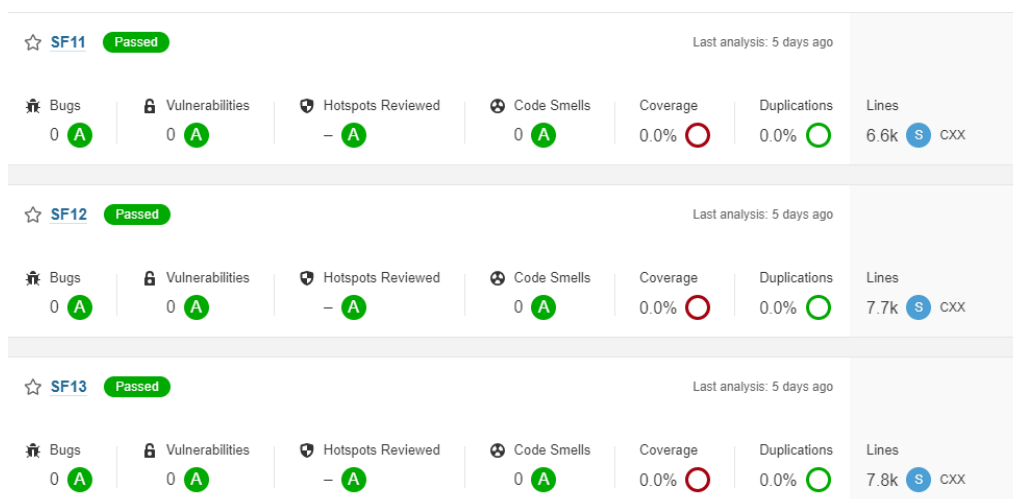


Figura 5.2:

Risultati delle ultime tre versioni di Stockfish analizzate in SonarQube.

Fonte: "(illustrazione creata dall'autore)"

Come possiamo notare le versioni di Stockfish, nonostante le dimensioni: 6,6 kloc, 7,7 kloc e 7,8 kloc non presentano difetti né Code Smells Ottenendo così S come valutazione in tutte e tre le versioni, dunque possiamo affermare che la qualità del codice viene tenuta in considerazione e controllata.

5.2 Counter

Counter è un chess engine gratuito ed open source scritto in Go da Vadim Chizhov. È valutato come settantacinquesimo chess engine nella classifica CCRL 40/15 nella quale Stockfish è classificato primo. Ho analizzato le ultime tre versioni, rispettivamente la CounterGo-1.36.0, CounterGo-1.37.0 e CounterGo-1.38.0 chiamandole in SonarQube CGO36, CGO37 e CGO38.

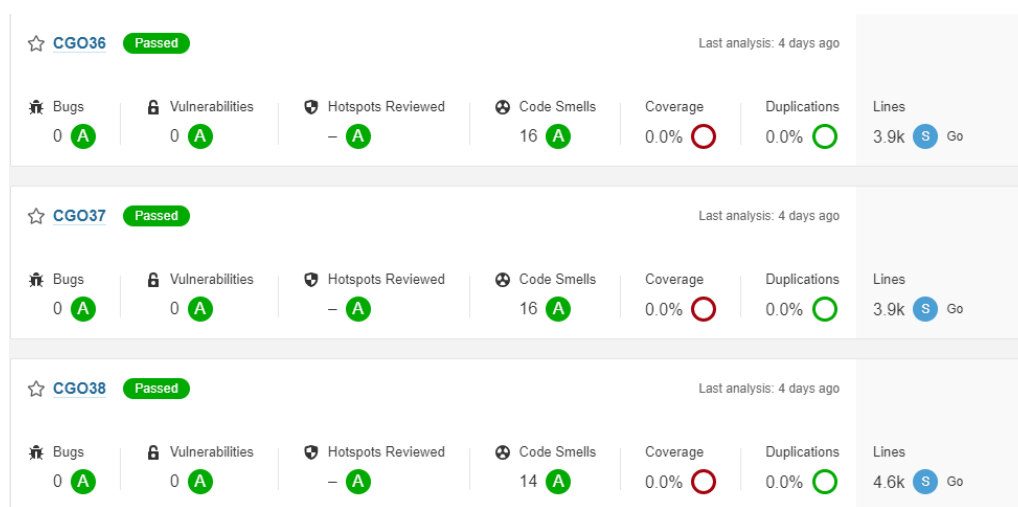


Figura 5.3:

Risultati delle ultime tre versioni di Counter analizzate in SonarQube.

Fonte: ”(illustrazione creata dall’autore)”

Come possiamo notare dall’immagine sopra riportata possiamo affermare che l’autore presta molta attenzione alla qualità del codice in quanto aumentando le dimensioni nelle versioni successive, si passa da 3,9 kloc a 4,6 kloc, i difetti decrescono passando da 16 Code Smells a 14 Code Smells.

5.3 Crocodile

Crocodile è un chess engine amatoriale creato dall’utente di GitHub Virinas-code in Python, la prima versione è stata rilasciata il 31/03/2021,

fino ad oggi sono uscite cinque versioni, io ho analizzato le ultime tre: la versione Crocodile-1.0.2, Crocodile-1.1.0 e Crocodile-2.0.0, che su Stockfish ho rinominato come V102, V110 e V200.

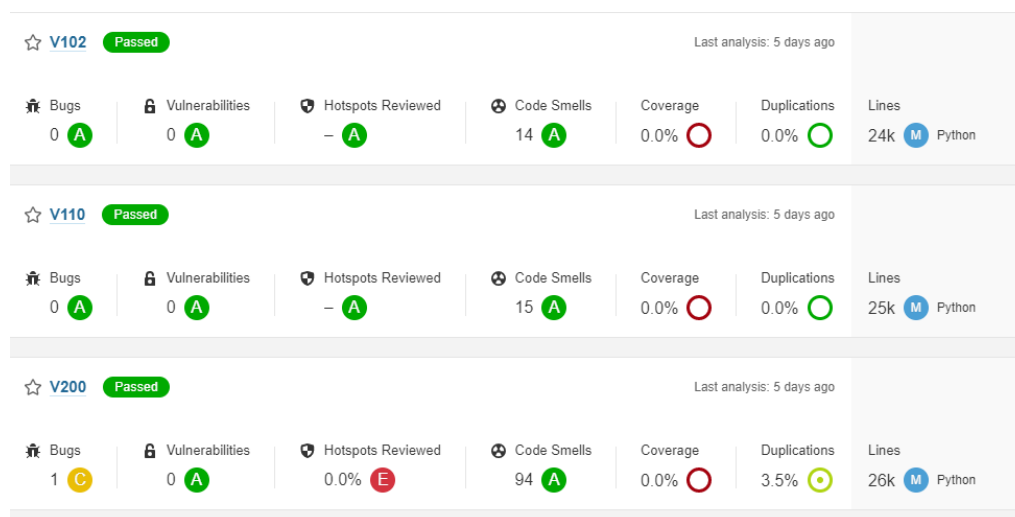


Figura 5.4:

Risultati delle ultime tre versioni di Crocodile analizzate in SonarQube.

Fonte: "(illustrazione creata dall'autore)"

In Crocodile la dimensione del programma in kloc è nettamente maggiore dei precedenti passando da 24 kloc nella versione 1.0.2 a 26 kloc nella versione 2.0.0. La qualità del codice non viene tenuta in considerazione e ciò si può notare dall'enorme incremento di difetti dalla versione 1.1.0 alla versione 2.0.0.

5.4 Analisi della prima versione di SonarQube attraverso SonarQube

Inoltre ho anche effettuato mediante la versione 8.9 di SonarQube l'analisi della versione beta 1.0.

5.4 Analisi della prima versione di SonarQube attraverso SonarQube

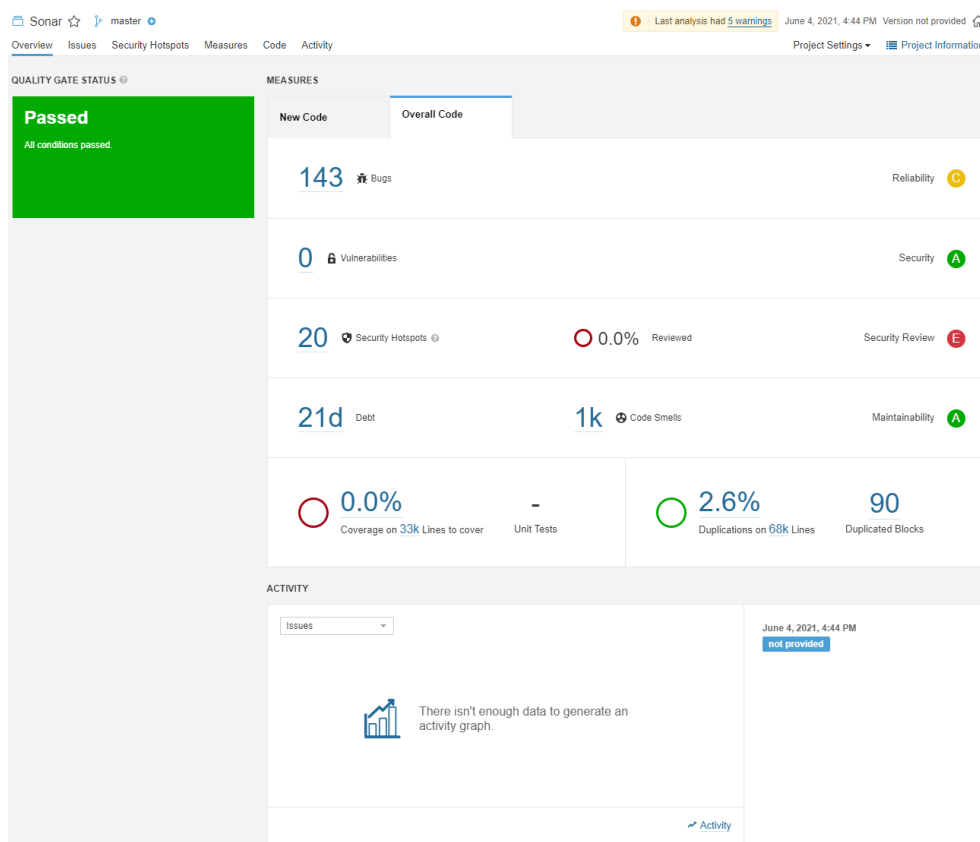


Figura 5.5:
Risultati della scansione di SonarQube 1.0 Beta
Fonte: ”(illustrazione creata dall'autore)”

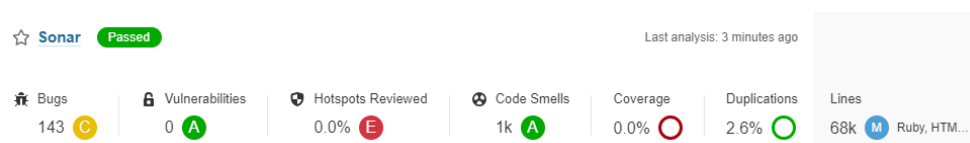


Figura 5.6:
Risultati della scansione di SonarQube 1.0 Beta
Fonte: ”(illustrazione creata dall'autore)”

La versione è chiaramente una beta siccome presenta 143 bug, 20 security hotspots, 1000 code smells e 21 giorni, stimati come giorni lavorativi da otto

ore, per recuperare il debito tecnico. Bisogna anche tenere in considerazione che stiamo parlando di 68 kloc(migliaia di linee di codice) quindi dato il codice molto lungo stiamo parlando di difetti in percentuale contenuti.

Capitolo 6

Conclusioni e sviluppi futuri

In conclusione il software si presenta sotto molteplici forme e vista l'importanza e la quantità di informazioni sensibili presenti è fondamentale che sia il più sicuro ed affidabile. Per questo la qualità di quest'ultimo non va solo presa in considerazione ma costantemente monitorata e migliorata avendo mostrato quanto sono elevati i costi di correzione soprattutto nelle fasi più avanzate di sviluppo.

Nel primo capitolo sono stati introdotti gli argomenti che serviranno nei capitoli successivi e perchè scegliere SonarQube. Nel secondo capitolo si è parlato di qualità del codice nei suoi vari aspetti dei modi per misurarla, del debito tecnico e di come migliorare la qualità del proprio codice. Nel terzo capitolo viene trattata la prevenzione dei difetti i suoi principi ed i vantaggi che ne derivano. Nel quarto capitolo si è parlato SonarQube, sia attraverso una descrizione dei più importanti cambiamenti da versione a versione, sia un'analisi di SonarQube utilizzando SonarQube stesso. Nel quinto capitolo abbiamo analizzato con SonarQube tre versioni successive di tre programmi di scacchi in modo da monitorarne i cambiamenti nei difetti e quindi studiarne l'evoluzione da versione a versione. Infine nell'appendice è presente una guida molto dettagliata sull'installazione di SonarQube su Windows.

Come sviluppi futuri possibili per SonarQube si parla di incrementare il numero di regole per i linguaggi aggiunti recentemente, come Go e Py-

thon. Inoltre sarebbe interessante vedere insegnato agli studenti un metodo di controllo della qualità del codice ed i vari metodi di controllo possibili, magari attraverso l'utilizzo di SonarQube in modo da integrarlo alle materie riguardanti la programmazione.

Ringraziamenti

A conclusione di questo elaborato, desidero menzionare tutte le persone senza le quali questa tesi non esisterebbe nemmeno.

Ringrazio il mio relatore Paolo Ciancarini, che in questi mesi è sempre stato molto disponibile ed ha saputo guidarmi con suggerimenti pratici nelle ricerche e nella stesura dell'elaborato.

Ringrazio i miei genitori, che non hanno fatto mai mancare il loro sostegno sia economico che morale.

Ringrazio tutti i miei amici e compagni di corso per tutti i bei momenti passati insieme.

Ringrazio mio cugino e compare Marco, che mi ha sempre aiutato e sostenuto in questi anni.

Infine, un ringraziamento di cuore va ad Annachiara, senza la quale non sarei qui.

Grazie per tutto il tempo che mi hai dedicato e per avermi trasmesso la tua forza di volontà per andare sempre avanti.

Appendice A

Come installare SonarQube e le sue componenti su Windows

Abbiamo discusso molto sull'importanza di SonarQube, ora andiamo a vedere passo passo come analizzare il nostro primo programma partendo da zero.

A.1 Dove scaricare SonarQube

Il primo passaggio da effettuare è aprire il nostro browser e digitare "sonarqube"

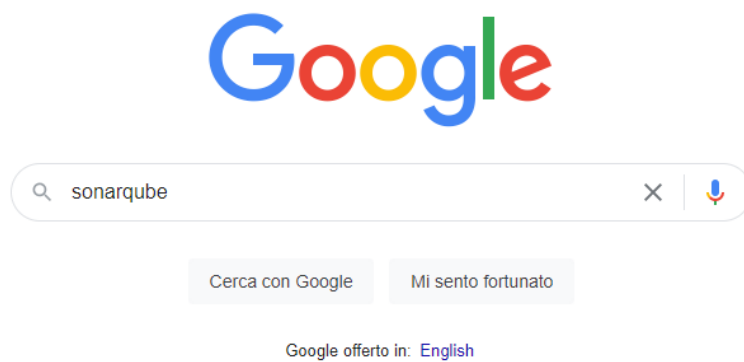



Figura A.1:
”(illustrazione creata dall'autore)”

Dopodichè si accede ad: <https://www.sonarqube.org>

<https://www.sonarqube.org> ▾ [Traduci questa pagina](#)

SonarQube: Code Quality and Code Security



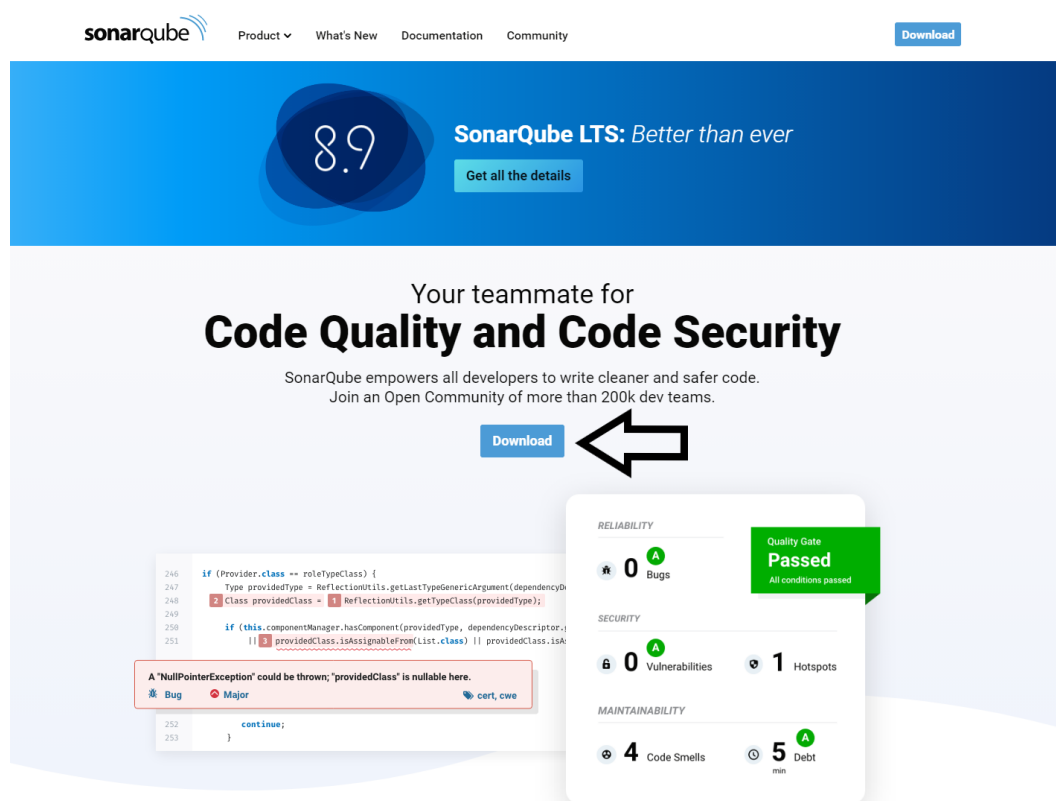
SonarQube empowers all developers to write cleaner and safer code. Join an Open Community of more than 200k dev teams. Download. **SonarQube** detects ...

Hai visitato questa pagina 3 volte. Ultima visita: 13/05/21

<h3>Download</h3> <p>Get the latest LTS and version of SonarQube the leading product ...</p>	<h3>About SonarQube</h3> <p>SonarQube is the leading tool for continuously inspecting the ...</p>
<h3>Documentation</h3> <p>Install the Server - Try Out SonarQube - Overview - ...</p>	<h3>OWASP Top 10</h3> <p>OWASP Top 10 - We've got you covered! - Developer-led ...</p>
<h3>Overview</h3> <p>Docs 8.9. Try Out SonarQube. Requirements. Setup and ...</p> <p>Altri risultati in sonarqube.org »</p>	<h3>Developer Edition</h3> <p>Pull request decoration and taint analysis for developer-led Code ...</p>

Figura A.2:
”(illustrazione creata dall'autore)”

Vi ritroverete nella home page del sito di SonarQube, qui dovrete andare su "Download".



Enhance Your Workflow with Continuous Code Quality & Code Security

Thousands of automated **Static Code Analysis** rules, protecting your app

Figura A.3:
”(illustrazione creata dall'autore)”

Io ho utilizzato la Community edition in quanto gratuita ed open source.

The screenshot shows the SonarQube download page. At the top, there is a navigation bar with the SonarQube logo and links for Product, What's New, Documentation, and Community. A 'Download' button is in the top right. The main heading is 'Download SonarQube' with the subtitle 'The leading product for Code Quality and Security' and 'HELPING DEVS SINCE 2008'. Below this, there are links for Version: 8.9 LTS, Release: May 2021, Getting Started, Release Notes, Upgrade Notes, and Available From DockerHub.

The page is divided into four columns representing different editions:

- Community Edition:** Described as 'The starting point for adopting code quality in your CI/CD'. It is 'FREE & OPEN SOURCE'. A 'Download for free' button is present. It lists features like static code analysis for 15 languages, bug detection, security hotspots, and CI/CD integration.
- Developer Edition:** Focuses on 'Maximum Application Security'. It includes a 'Download' button and a 'Request a Free Trial' button. It lists features like support for various languages, injection flaw detection, and integration with GitHub, Bitbucket, Azure DevOps, and GitLab.
- Enterprise Edition:** For 'Manage your Application Portfolio'. It includes a 'Download' button and a 'Request a Free Trial' button. It lists features like portfolio management, security reports, and support for various databases.
- Data Center Edition:** For 'High Availability, for global deployments'. It includes a 'Download' button and a 'Learn More' button. It lists features like component redundancy, data resiliency, and horizontal scalability.

A double-headed arrow is drawn between the Community and Developer edition columns.

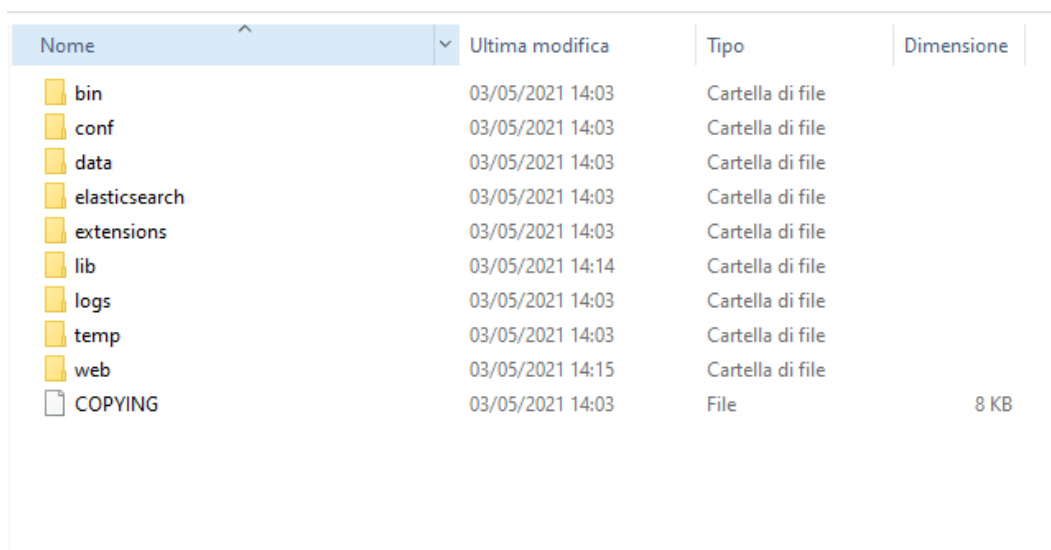
Figura A.4:
”(illustrazione creata dall'autore)”

Una volta premuto su ”Download for free” avrà inizio il download di SonarQube

A.2 Come installare il server di SonarQube

Ora che abbiamo scaricato SonarQube dobbiamo estrarlo dal file sonarqube-”versione”.zip. effettuato questo passaggio entriamo nella cartella estratta.

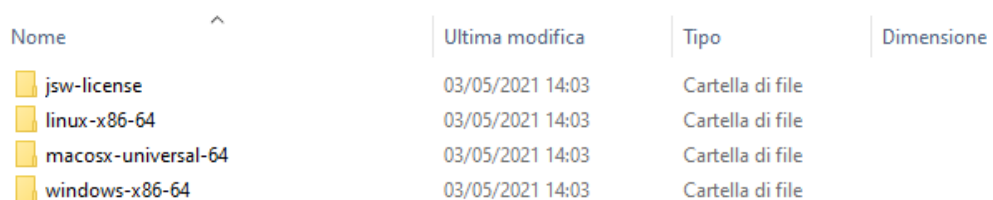
58 A. Come installare SonarQube e le sue componenti su Windows



Nome	Ultima modifica	Tipo	Dimensione
bin	03/05/2021 14:03	Cartella di file	
conf	03/05/2021 14:03	Cartella di file	
data	03/05/2021 14:03	Cartella di file	
elasticsearch	03/05/2021 14:03	Cartella di file	
extensions	03/05/2021 14:03	Cartella di file	
lib	03/05/2021 14:14	Cartella di file	
logs	03/05/2021 14:03	Cartella di file	
temp	03/05/2021 14:03	Cartella di file	
web	03/05/2021 14:15	Cartella di file	
COPYING	03/05/2021 14:03	File	8 KB

Figura A.5:
”(illustrazione creata dall'autore)”

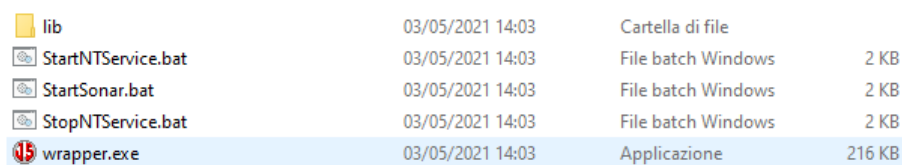
Entriamo in "bin"



Nome	Ultima modifica	Tipo	Dimensione
jsr-license	03/05/2021 14:03	Cartella di file	
linux-x86-64	03/05/2021 14:03	Cartella di file	
macosx-universal-64	03/05/2021 14:03	Cartella di file	
windows-x86-64	03/05/2021 14:03	Cartella di file	

Figura A.6:
”(illustrazione creata dall'autore)”

Poi entriamo in "windows-x86-64"



lib	03/05/2021 14:03	Cartella di file	
StartNTService.bat	03/05/2021 14:03	File batch Windows	2 KB
StartSonar.bat	03/05/2021 14:03	File batch Windows	2 KB
StopNTService.bat	03/05/2021 14:03	File batch Windows	2 KB
wrapper.exe	03/05/2021 14:03	Applicazione	216 KB

Figura A.7:
”(illustrazione creata dall'autore)”

Da qui per avviare il server di SonarQube si clicca su StartSonar.bat.

A.3 Come scaricare JDK

Attualmente per funzionare SonarQube ha bisogno di JDK(Java Development Kit) 11. Andiamo a vedere passo passo come installarlo senza interferire con gli altri programmi java installati.

Per prima cosa cerchiamo sul nostro browser ”jdk 11 download”

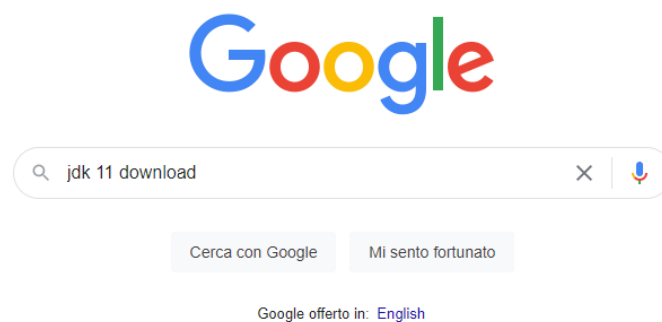


Figura A.8:
”(illustrazione creata dall’autore)”

Dovremo poi andare su ”Java SE development kit” o al seguente url:
<https://www.Oracle.com/it/java/technologies/javase-jdk11-downloads.html>

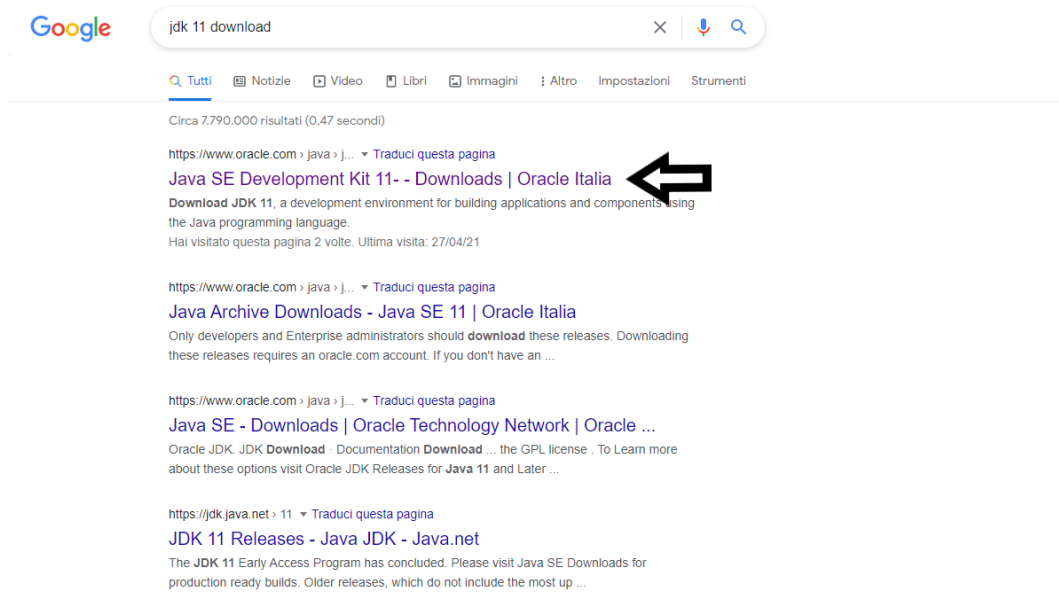













Figura A.9:

”(illustrazione creata dall’autore)”

Da qui andremo sul download per windows di ”Windows x64 Compressed Archive”

62 A. Come installare SonarQube e le sue componenti su Windows

Java SE Development Kit 11.0.11
This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE

Product / File Description	File Size	Download
Linux ARM 64 Debian Package	145.61 MB	 jdk-11.0.11_linux-aarch64_bin.deb
Linux ARM 64 RPM Package	152.16 MB	 jdk-11.0.11_linux-aarch64_bin.rpm
Linux ARM 64 Compressed Archive	169.53 MB	 jdk-11.0.11_linux-aarch64_bin.tar.gz
Linux x64 Debian Package	149.39 MB	 jdk-11.0.11_linux-x64_bin.deb
Linux x64 RPM Package	156.07 MB	 jdk-11.0.11_linux-x64_bin.rpm
Linux x64 Compressed Archive	173.46 MB	 jdk-11.0.11_linux-x64_bin.tar.gz
macOS Installer	167.15 MB	 jdk-11.0.11_osx-x64_bin.dmg
macOS Compressed Archive	167.67 MB	 jdk-11.0.11_osx-x64_bin.tar.gz
Solaris SPARC Compressed Archive	184.51 MB	 jdk-11.0.11_solaris-sparcv9_bin.tar.gz
Windows x64 Installer	152.05 MB	 jdk-11.0.11_windows-x64_bin.exe
Windows x64 Compressed Archive	171.53 MB	 jdk-11.0.11_windows-x64_bin.zip




Figura A.10:
”(illustrazione creata dall'autore)”

Accettiamo termini e condizioni d'uso e andiamo su "Download jdk-11.0.11_windows-x64_bin.zip"

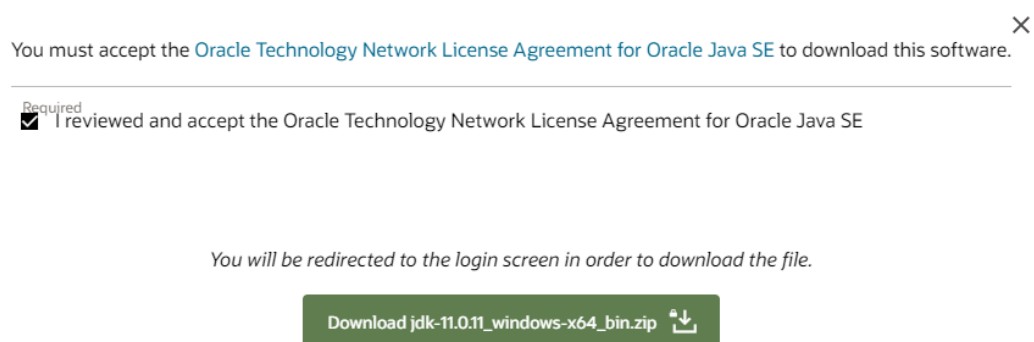


Figura A.11:
”(illustrazione creata dall’autore)”

Ora dobbiamo semplicemente effettuare il login con l’account Oracle e il download inizierà, se non avete un account Oracle createlo attraverso la procedura guidata.

Una volta scaricato jdk bisogna estrarlo, come programma per estrarlo consiglio 7zip, in quanto open source, ma ce ne sono molti altri. dopodichè apriamo la cartella estratta, per la versione attuale ”jdk-11.0.11”

64 A. Come installare SonarQube e le sue componenti su Windows









 bin	18/05/2021 11:03	Cartella di file	
 conf	18/05/2021 11:03	Cartella di file	
 include	18/05/2021 11:03	Cartella di file	
 jmods	18/05/2021 11:03	Cartella di file	
 legal	18/05/2021 11:03	Cartella di file	
 lib	18/05/2021 11:03	Cartella di file	
 README.html	18/03/2021 18:55	Chrome HTML Do...	1 KB
 release	18/03/2021 18:55	File	2 KB

Figura A.12:
”(illustrazione creata dall'autore)”

Ora entriamo nella cartella "bin" e copiamo il percorso file

Nome	Ultima modifica	Tipo	Dimensione
server	18/05/2021 11:03	Cartella di file	
api-ms-win-core-console-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	12 KB
api-ms-win-core-console-l1-2-0.dll	18/03/2021 18:55	Estensione dell'ap...	12 KB
api-ms-win-core-datetime-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	12 KB
api-ms-win-core-debug-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	12 KB
api-ms-win-core-errorhandling-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	12 KB
api-ms-win-core-file-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	15 KB
api-ms-win-core-file-l1-2-0.dll	18/03/2021 18:55	Estensione dell'ap...	12 KB
api-ms-win-core-file-l2-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	12 KB
api-ms-win-core-handle-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	12 KB
api-ms-win-core-heap-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	12 KB
api-ms-win-core-interlocked-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	12 KB
api-ms-win-core-libraryloader-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	13 KB
api-ms-win-core-localization-l1-2-0.dll	18/03/2021 18:55	Estensione dell'ap...	14 KB
api-ms-win-core-memory-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	12 KB
api-ms-win-core-namedpipe-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	12 KB
api-ms-win-core-processenvironment-l1...	18/03/2021 18:55	Estensione dell'ap...	13 KB
api-ms-win-core-processthreads-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	14 KB
api-ms-win-core-processthreads-l1-1-1.dll	18/03/2021 18:55	Estensione dell'ap...	12 KB
api-ms-win-core-profile-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	11 KB
api-ms-win-core-rtlsupport-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	12 KB
api-ms-win-core-string-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	12 KB
api-ms-win-core-synch-l1-1-0.dll	18/03/2021 18:55	Estensione dell'ap...	14 KB

Figura A.13:

” (illustrazione creata dall'autore)”

Andiamo su visualizza impostazioni di sistema avanzate

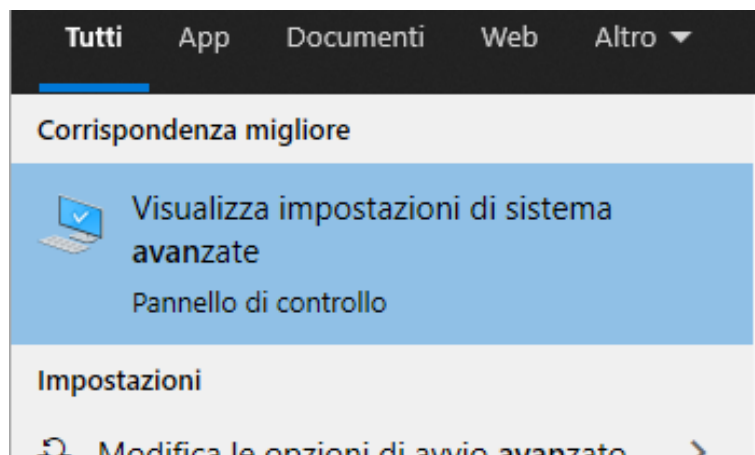


Figura A.14:
”(illustrazione creata dall’autore)”

Poi su ”variabili d’ambiente...”

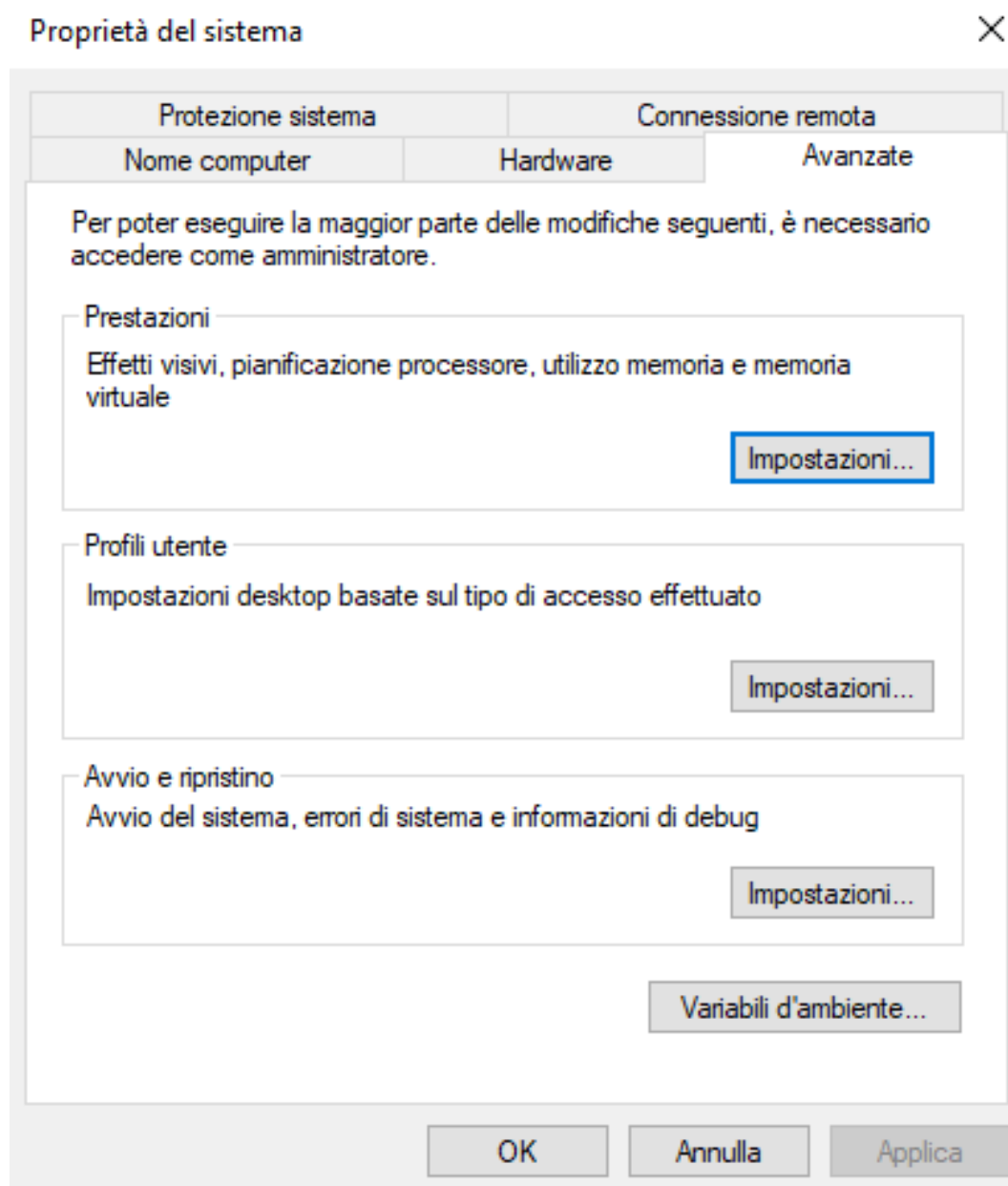


Figura A.15:
”(illustrazione creata dall'autore)”

Andiamo su path con doppio click

68 A. Come installare SonarQube e le sue componenti su Windows

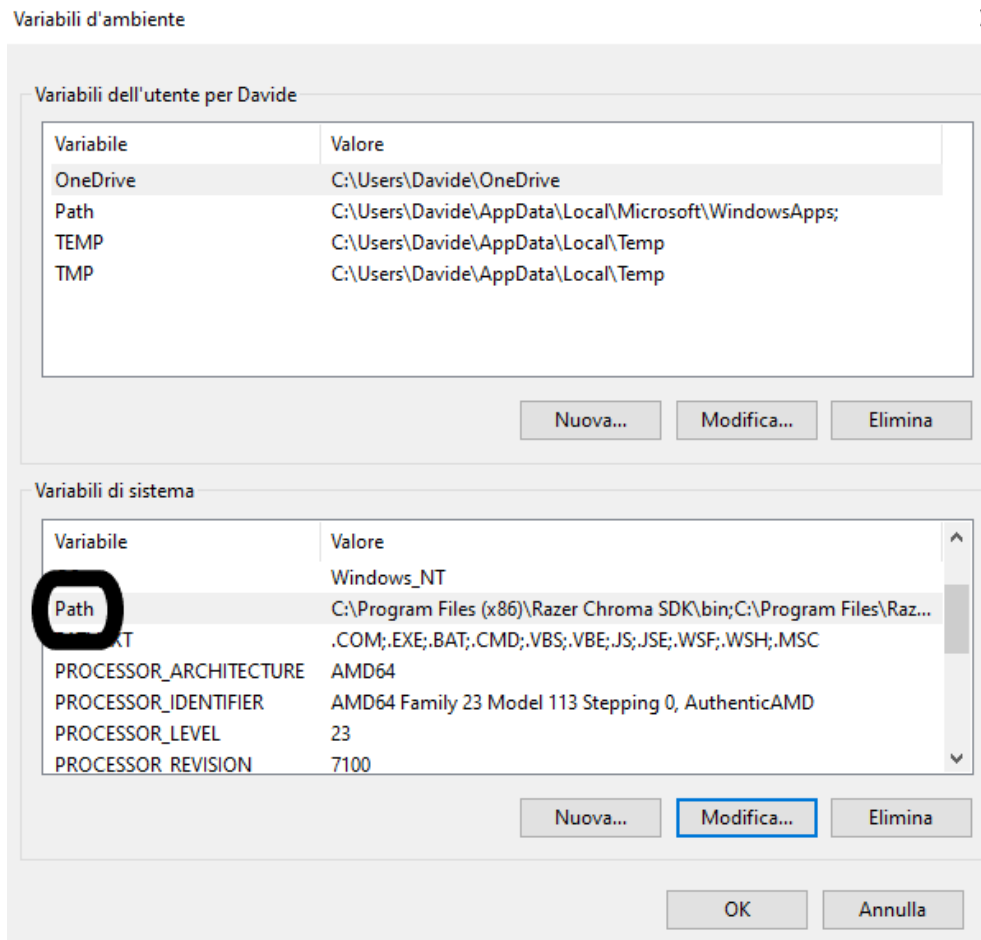


Figura A.16:
”(illustrazione creata dall'autore)”

Dobbiamo solo andare su nuovo ed incollare il link del percorso di jdk copiato in precedenza.

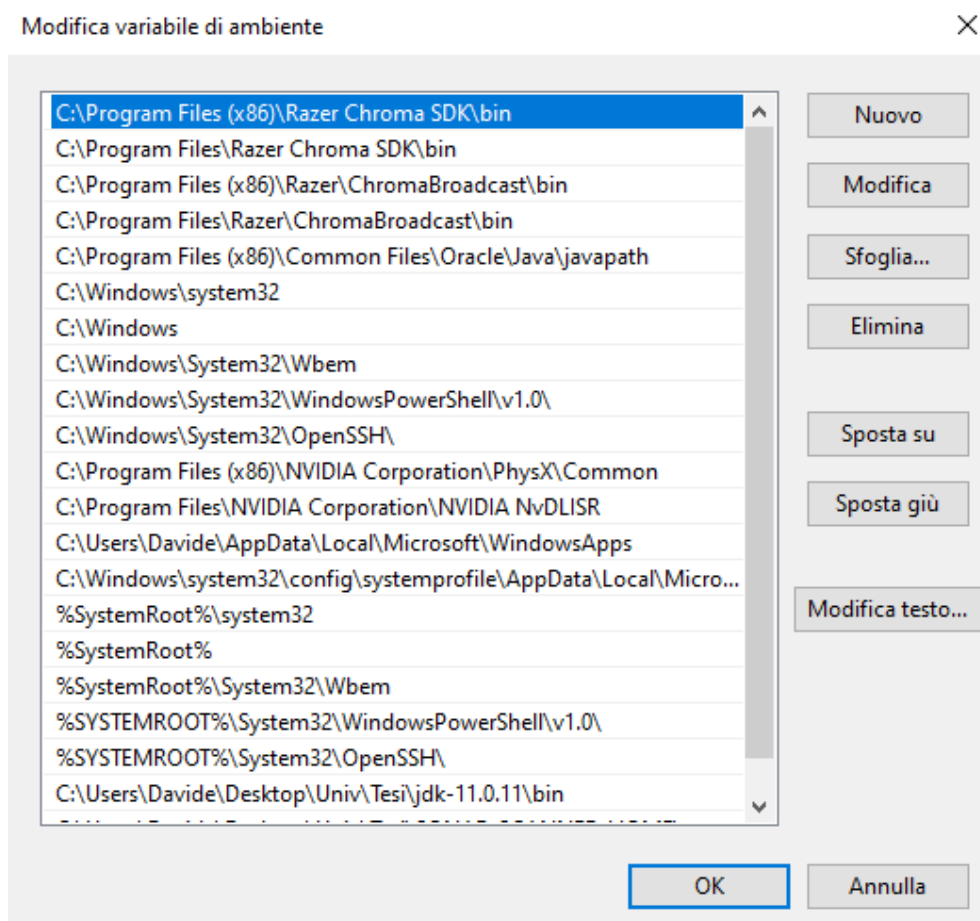
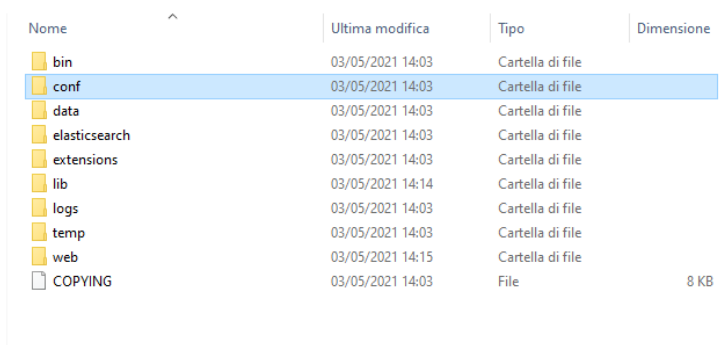


Figura A.17:
”(illustrazione creata dall'autore)”

A.4 Configurare Sonarqube

Dobbiamo ora configurare SonarQube per l'utilizzo di jdk 11. Per prima cosa entriamo nella cartella "conf" di SonarQube.

70 A. Come installare SonarQube e le sue componenti su Windows



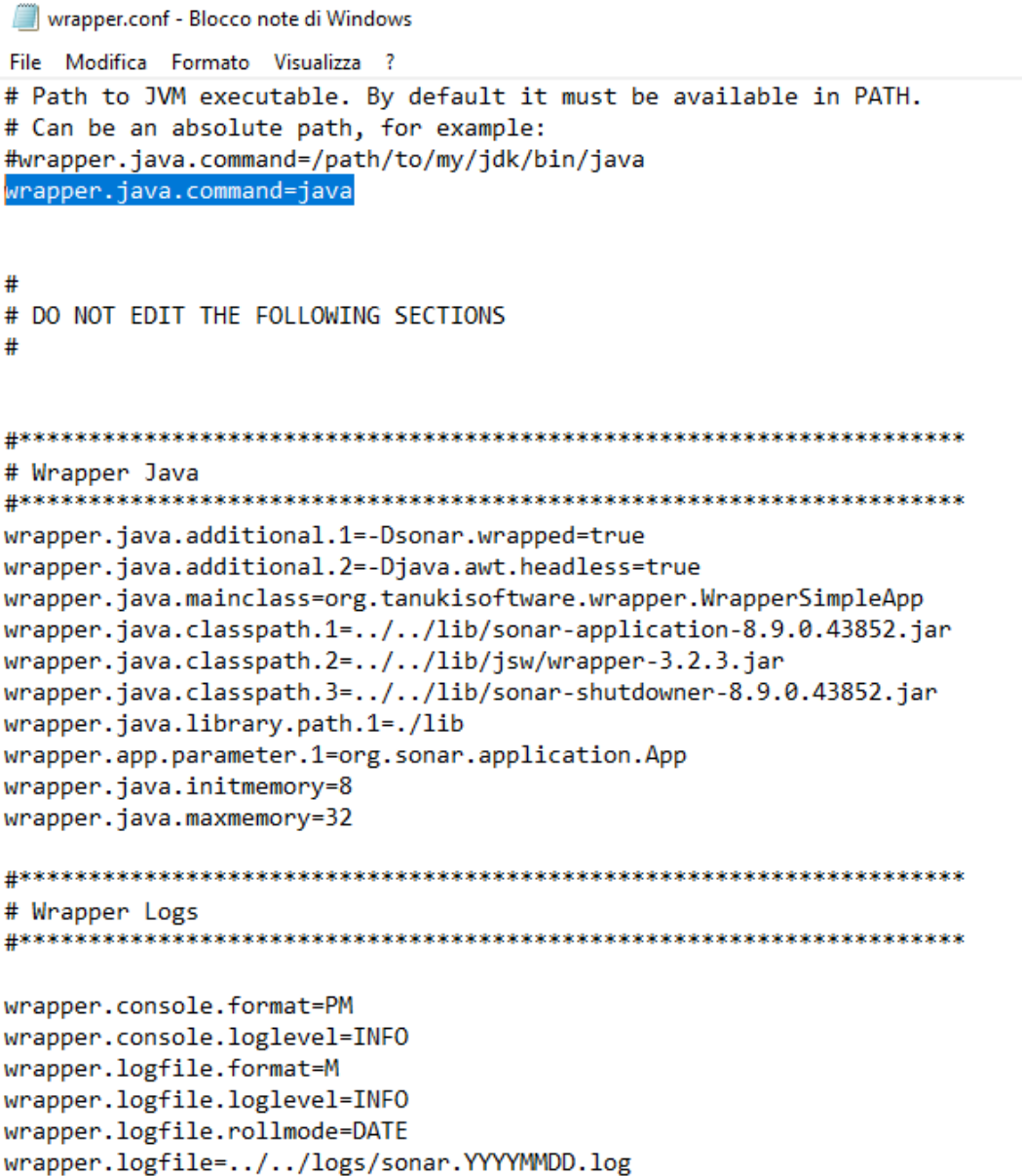
Nome	Ultima modifica	Tipo	Dimensione
bin	03/05/2021 14:03	Cartella di file	
conf	03/05/2021 14:03	Cartella di file	
data	03/05/2021 14:03	Cartella di file	
elasticsearch	03/05/2021 14:03	Cartella di file	
extensions	03/05/2021 14:03	Cartella di file	
lib	03/05/2021 14:14	Cartella di file	
logs	03/05/2021 14:03	Cartella di file	
temp	03/05/2021 14:03	Cartella di file	
web	03/05/2021 14:15	Cartella di file	
COPYING	03/05/2021 14:03	File	8 KB

[!h]

Figura A.18:
”(illustrazione creata dall’autore)”

Qui troveremo ”wrapper.conf” che andremo ad editare, io per questa operazione utilizzo il blocco note di windows.

[!h]



```

wrapper.conf - Blocco note di Windows
File Modifica Formato Visualizza ?
# Path to JVM executable. By default it must be available in PATH.
# Can be an absolute path, for example:
#wrapper.java.command=/path/to/my/jdk/bin/java
wrapper.java.command=java

#
# DO NOT EDIT THE FOLLOWING SECTIONS
#

#*****
# Wrapper Java
#*****
wrapper.java.additional.1=-Dsonar.wrapped=true
wrapper.java.additional.2=-Djava.awt.headless=true
wrapper.java.mainclass=org.tanukisoftwares.wrapper.WrapperSimpleApp
wrapper.java.classpath.1=../../lib/sonar-application-8.9.0.43852.jar
wrapper.java.classpath.2=../../lib/jsw/wrapper-3.2.3.jar
wrapper.java.classpath.3=../../lib/sonar-shutdowner-8.9.0.43852.jar
wrapper.java.library.path.1=./lib
wrapper.app.parameter.1=org.sonar.application.App
wrapper.java.initmemory=8
wrapper.java.maxmemory=32

#*****
# Wrapper Logs
#*****

wrapper.console.format=PM
wrapper.console.loglevel=INFO
wrapper.logfile.format=M
wrapper.logfile.loglevel=INFO
wrapper.logfile.rollmode=DATE
wrapper.logfile=../../logs/sonar.YYYYMMDD.log

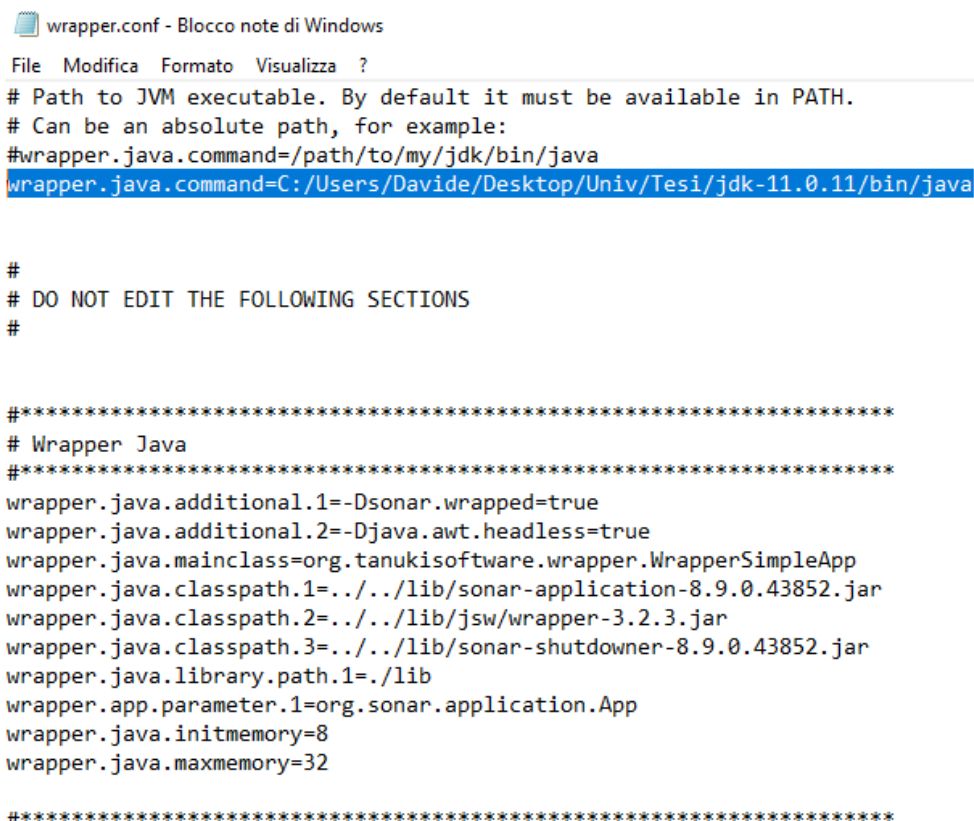
```

Figura A.19:

” (illustrazione creata dall’autore)”

Nella riga evidenziata sostituiremo il java dopo l’uguale con il percorso file della cartella bin di jdk 11 sostituendo ai backslash ”\” gli slash ”/” ed

aggiungento java in fondo.



```

wrapper.conf - Blocco note di Windows
File Modifica Formato Visualizza ?
# Path to JVM executable. By default it must be available in PATH.
# Can be an absolute path, for example:
#wrapper.java.command=/path/to/my/jdk/bin/java
wrapper.java.command=C:/Users/Davide/Desktop/Univ/Tesi/jdk-11.0.11/bin/java

#
# DO NOT EDIT THE FOLLOWING SECTIONS
#

*****
# Wrapper Java
*****
wrapper.java.additional.1=-Dsonar.wrapped=true
wrapper.java.additional.2=-Djava.awt.headless=true
wrapper.java.mainclass=org.tanukisoftware.wrapper.WrapperSimpleApp
wrapper.java.classpath.1=../../lib/sonar-application-8.9.0.43852.jar
wrapper.java.classpath.2=../../lib/jsw/wrapper-3.2.3.jar
wrapper.java.classpath.3=../../lib/sonar-shutdowner-8.9.0.43852.jar
wrapper.java.library.path.1=./lib
wrapper.app.parameter.1=org.sonar.application.App
wrapper.java.initmemory=8
wrapper.java.maxmemory=32

*****

```

Figura A.20:
 ”(illustrazione creata dall’autore)”

SonarQube e jdk11 ora sono configurati.

A.5 Come accedere all’interfaccia di SonarQube

Andiamo a vedere come avviare Sonarqube ed accedere all’interfaccia. Per avviare il server di SonarQube dobbiamo entrare nella cartella di Sonarqube, andare in ”bin” poi in ”windows-x86-64” ed infine avviare ”StartSonar.bat”

Nome	Ultima modifica	Tipo	Dimensione
lib	03/05/2021 14:03	Cartella di file	
StartNTService.bat	03/05/2021 14:03	File batch Windows	2 KB
StartSonar.bat	03/05/2021 14:03	File batch Windows	2 KB
StopNTService.bat	03/05/2021 14:03	File batch Windows	2 KB
wrapper.exe	03/05/2021 14:03	Applicazione	216 KB

Figura A.21:

”(illustrazione creata dall'autore)”

Ora che il server è stato avviato non ci resta che accedere all'interfaccia di SonarQube, per farlo dobbiamo andare sul browser ed inserire come url:”http://localhost:9000/”

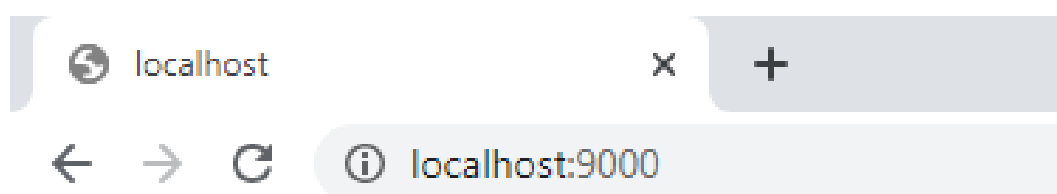


Figura A.22:

”(illustrazione creata dall'autore)”

Vi verranno chieste le credenziali che al primo accesso saranno ”admin” sia nel campo login che in password

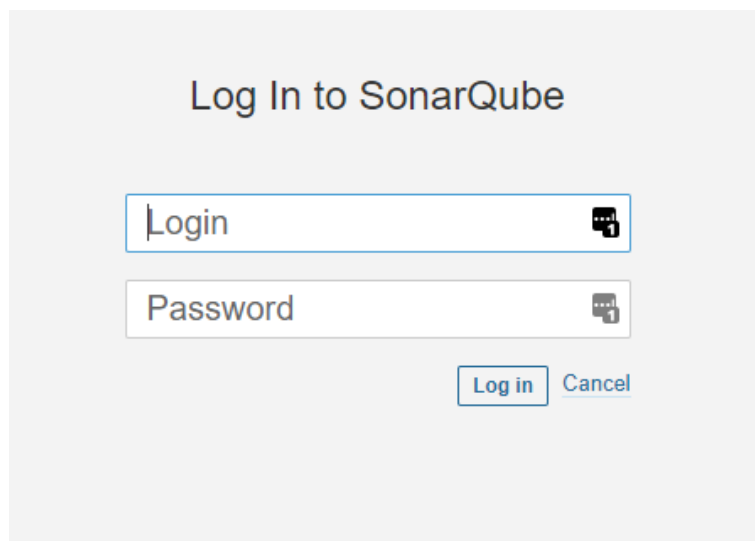


Figura A.23:
”(illustrazione creata dall’autore)”

Dopo il primo login verrà chiesto di cambiare la password ed una volta cambiata riusciremo ad accedere all’interfaccia di SonarQube.

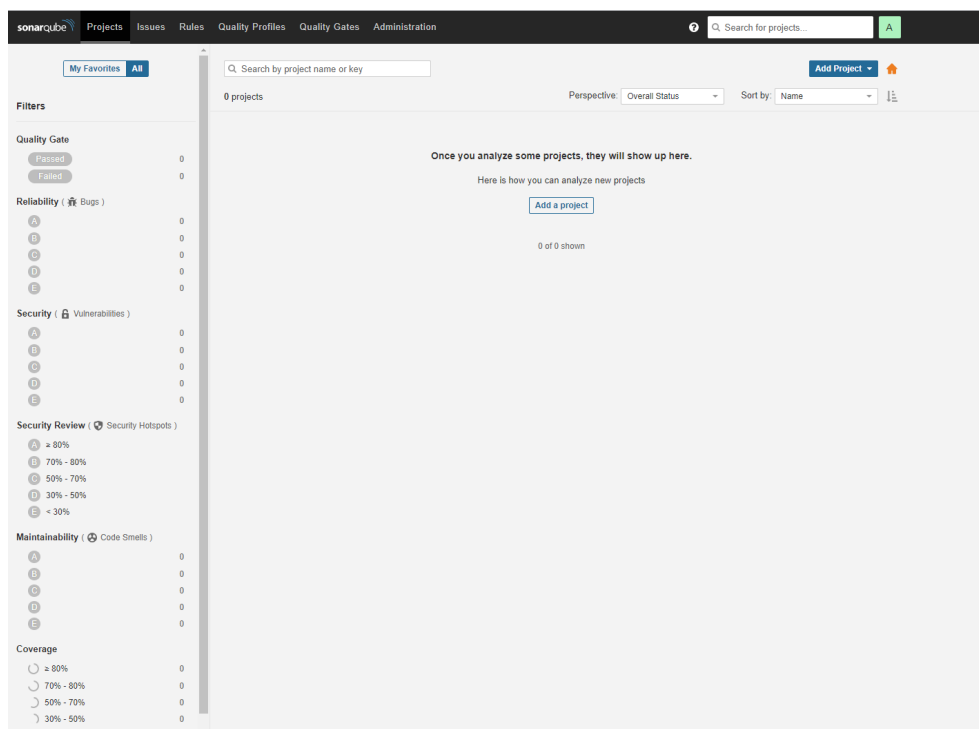
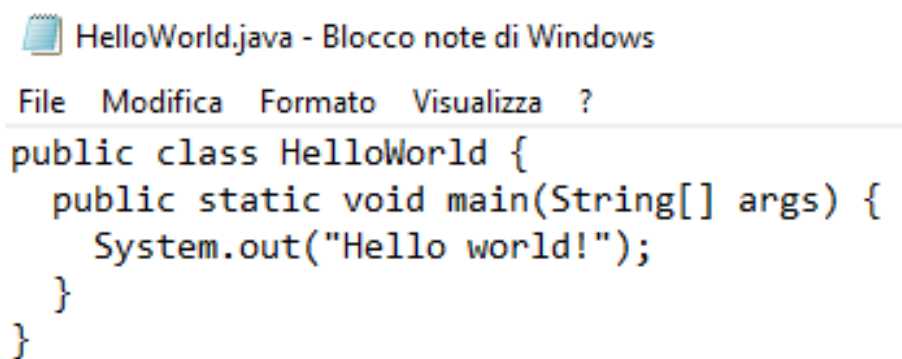


Figura A.24:
”(illustrazione creata dall'autore)”

A.6 Effettuare una scansione

Ora che abbiamo installato SonarQube andiamo a vedere come effettuare una scansione. Come progetto di prova da scansionare ho creato un "HelloWorld" in java

76 A. Come installare SonarQube e le sue componenti su Windows



HelloWorld.java - Blocco note di Windows

File Modifica Formato Visualizza ?

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out("Hello world!");  
    }  
}
```

Figura A.25:
”(illustrazione creata dall’autore)”

Dall’interfaccia di SonarQube andiamo su ”Add a project”

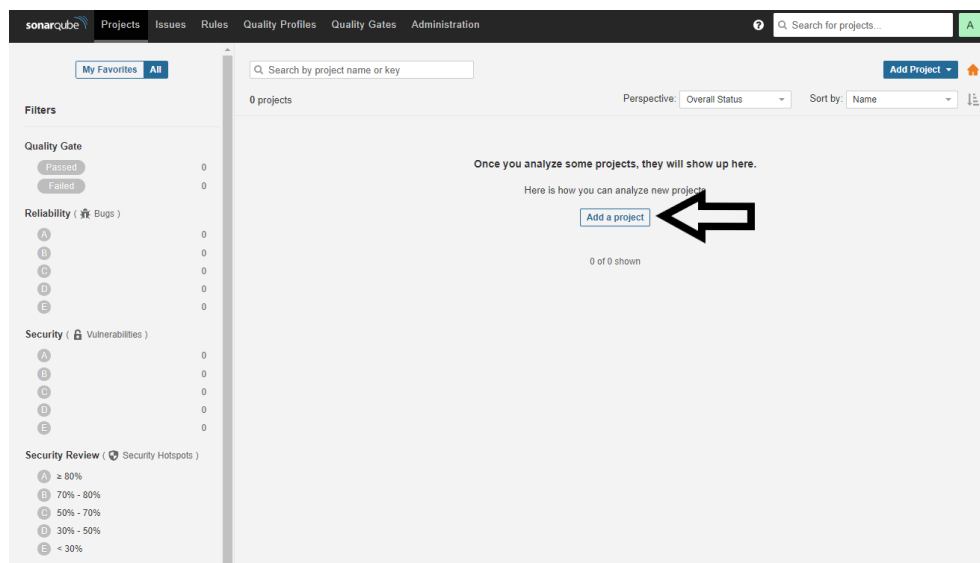


Figura A.26:
”(illustrazione creata dall’autore)”

Andiamo su Manually essendo l’unico metodo disponibile per la versione open source.

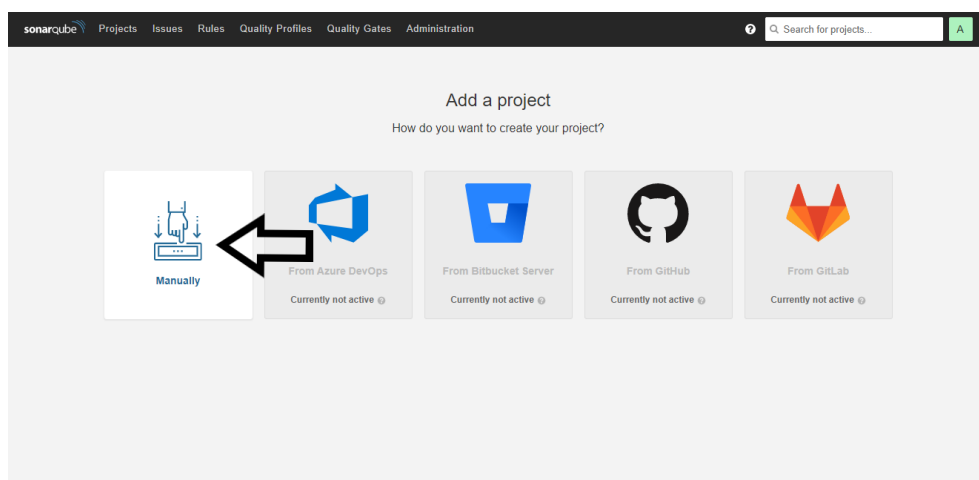


Figura A.27:

”(illustrazione creata dall'autore)”

Ora dobbiamo scegliere una ”project key” che sarà la chiave unica identificativa del progetto ed il ”project name” che è il nome con cui vedremo visualizzato il progetto

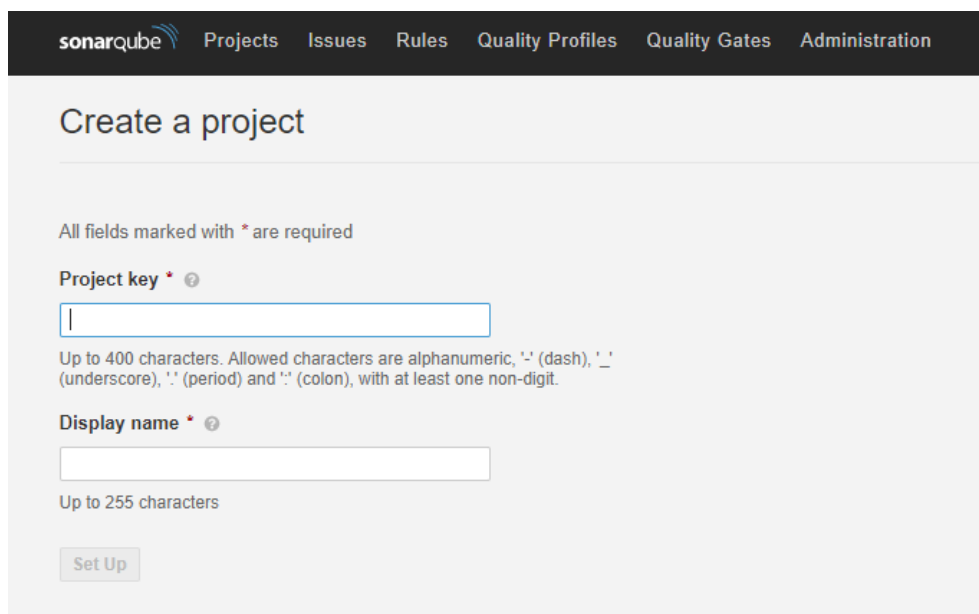


Figura A.28:

”(illustrazione creata dall'autore)”

78 A. Come installare SonarQube e le sue componenti su Windows

Procediamo poi a generare il token della scansione

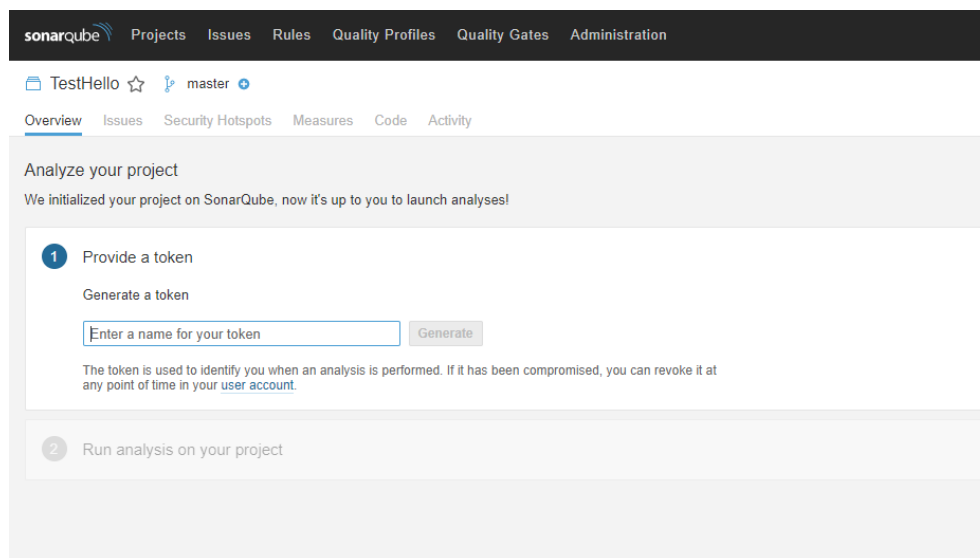


Figura A.29:

”(illustrazione creata dall'autore)”

Poi selezioniamo il programma utilizzato per effettuare la build del progetto.

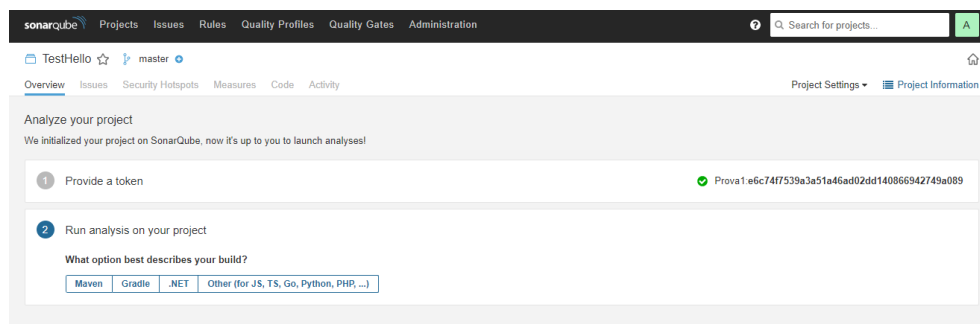


Figura A.30:

”(illustrazione creata dall'autore)”

Successivamente andiamo a selezionare il sistema operativo.

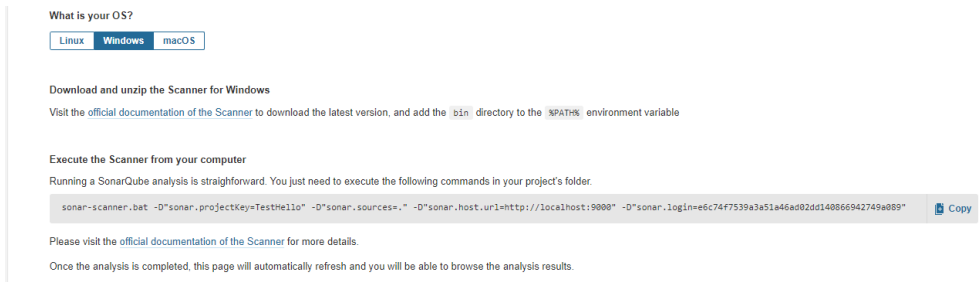
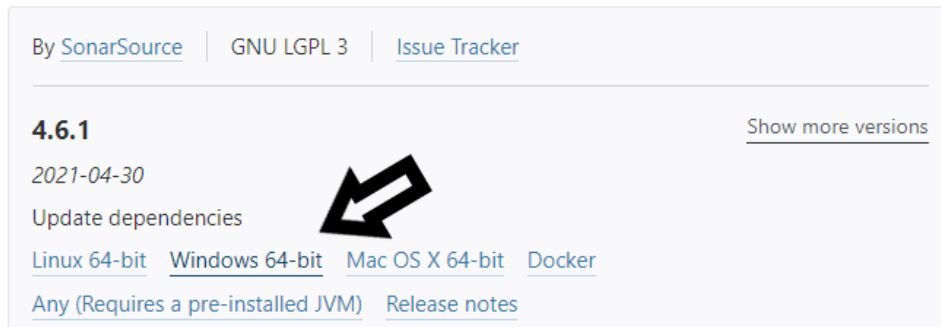


Figura A.31:
”(illustrazione creata dall’autore)”

A.6.1 Installare Sonar Scanner

SonarScanner è il programma che si occupa di esaminare i progetti per poi inviare il resoconto al server di SonarQube. Per prima cosa dobbiamo scaricare SonarScanner dal seguente link: [SonarScanner](#)

SonarScanner



The SonarScanner is the scanner to use when there is no specific scanner for your build system.

Figura A.32:
”(illustrazione creata dall’autore)”

Quando il download è terminato andiamo ad estrarre SonarScanner

80 A. Come installare SonarQube e le sue componenti su Windows

Nome	Ultima modifica	Tipo	Dimensione
HelloWorldTest	21/05/2021 10:29	Cartella di file	
jdk-11.0.11	18/05/2021 11:03	Cartella di file	
sonarqube-8.9.0.43852	03/05/2021 14:14	Cartella di file	
sonar-scanner-cli-4.6.1.2450-windows	21/05/2021 12:30	Cartella di file	
sonarqube-8.9.0.43852.zip	15/05/2021 10:21	Cartella compressa	270.480 KB
sonar-scanner-cli-4.6.1.2450-windows.zip	21/05/2021 12:12	Cartella compressa	38.696 KB

Figura A.33:
”(illustrazione creata dall'autore)”

Ora andiamo ad aggiungere a path il percorso della cartella bin di Sonar-scanner, come fatto in precedenza per jdk.

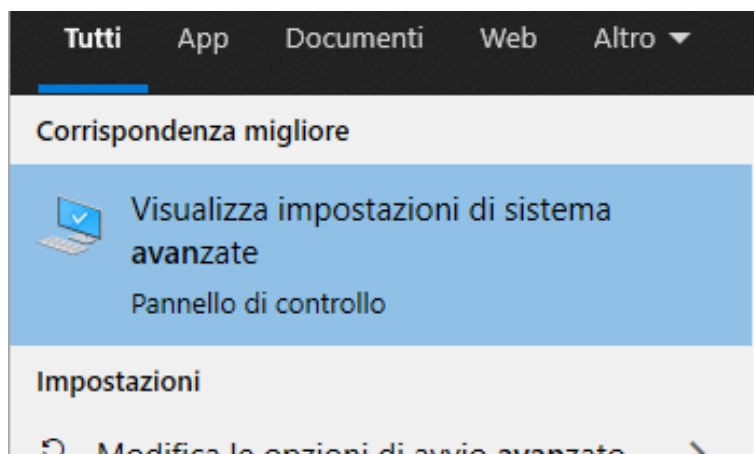


Figura A.34:
”(illustrazione creata dall'autore)”

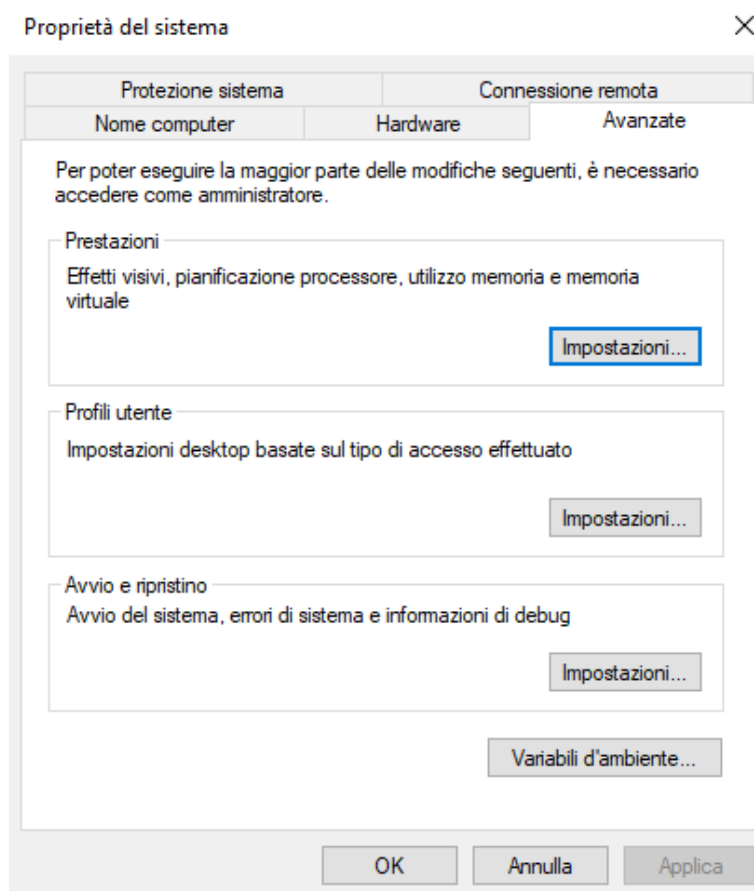


Figura A.35:
”(illustrazione creata dall'autore)”

82 A. Come installare SonarQube e le sue componenti su Windows

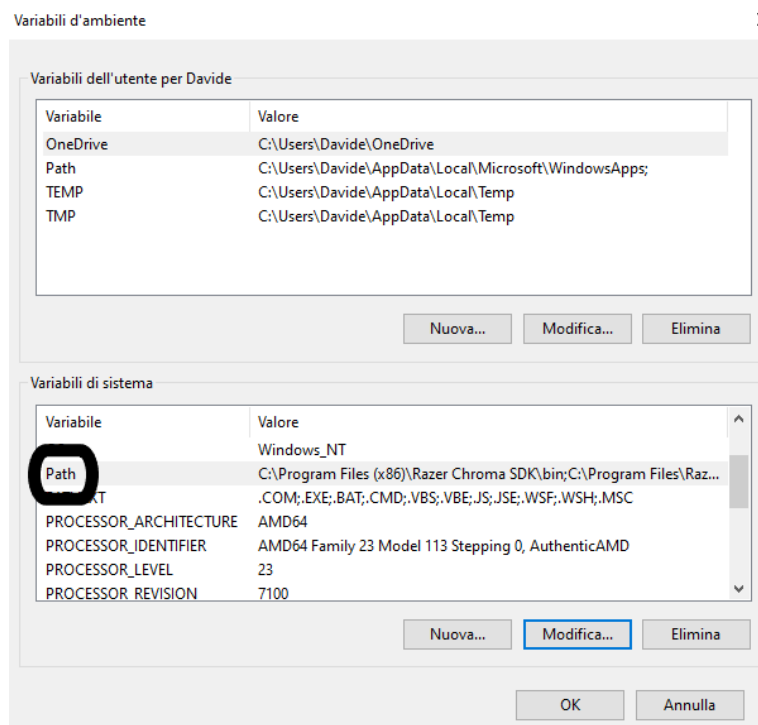


Figura A.36:
”(illustrazione creata dall'autore)”

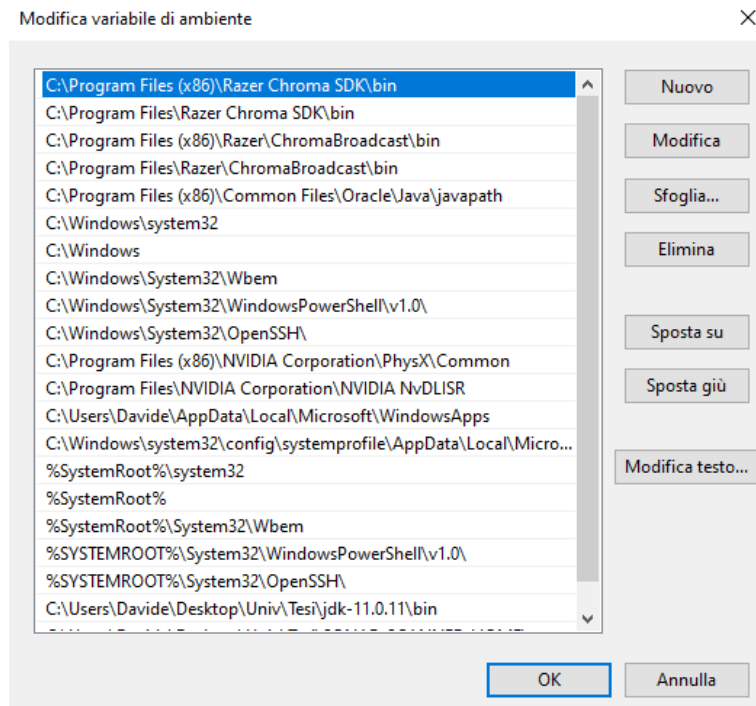
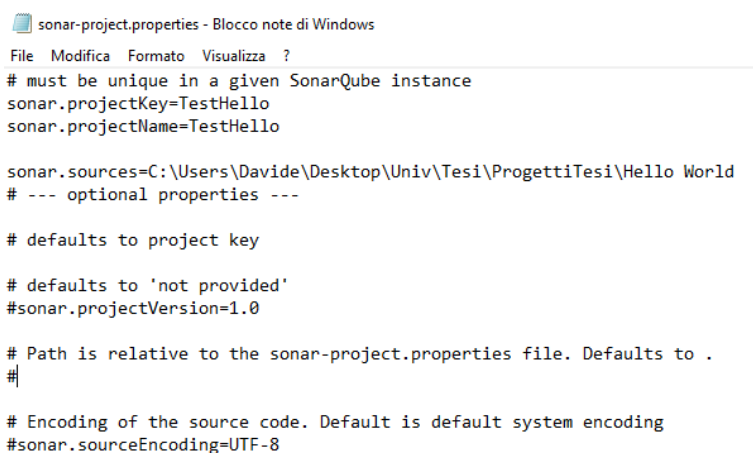


Figura A.37:
” (illustrazione creata dall'autore)”

A.6.2 Eseguire la scansione

Per prima cosa per eseguire la scansione bisogna creare un file chiamato ”sonar-project.properties” all’interno della cartella del progetto da esaminare.



```
sonar-project.properties - Blocco note di Windows
File Modifica Formato Visualizza ?
# must be unique in a given SonarQube instance
sonar.projectKey=TestHello
sonar.projectName=TestHello

sonar.sources=C:\Users\Davide\Desktop\Univ\Tesi\ProgettiTesi\Hello World
# --- optional properties ---

# defaults to project key

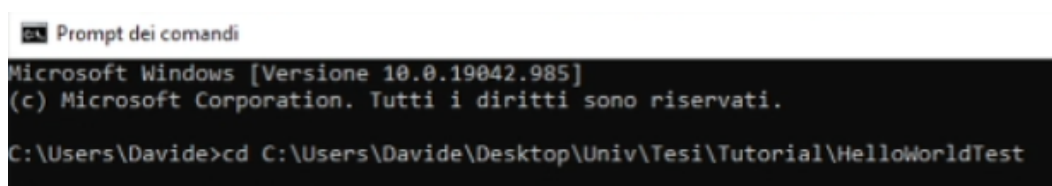
# defaults to 'not provided'
#sonar.projectVersion=1.0

# Path is relative to the sonar-project.properties file. Defaults to .
#

# Encoding of the source code. Default is default system encoding
#sonar.sourceEncoding=UTF-8
```

Figura A.38:
”(illustrazione creata dall’autore)”

Il parametro obbligatorio da inserire è ”sonar.projectKey” immettendo come nome la projectKey inserita quando abbiamo aggiunto il progetto a SonarQube. Gli altri parametri facoltativi sono disponibili al seguente link: [Parametri sonar-project.properties](#) Impostato sonar-project.properties è giunto il momento di eseguire la scansione, per fare ciò dobbiamo aprire il prompt dei comandi dentro la cartella del progetto oppure aprire il prompt dei comandi ed eseguire il comando: cd ”percorso file”



```
Prompt dei comandi
Microsoft Windows [Versione 10.0.19042.985]
(c) Microsoft Corporation. Tutti i diritti sono riservati.
C:\Users\Davide>cd C:\Users\Davide\Desktop\Univ\Tesi\Tutorial\HelloWorldTest
```

Figura A.39:
”(illustrazione creata dall’autore)”

Dobbiamo poi eseguire il comando passato da SonarQube

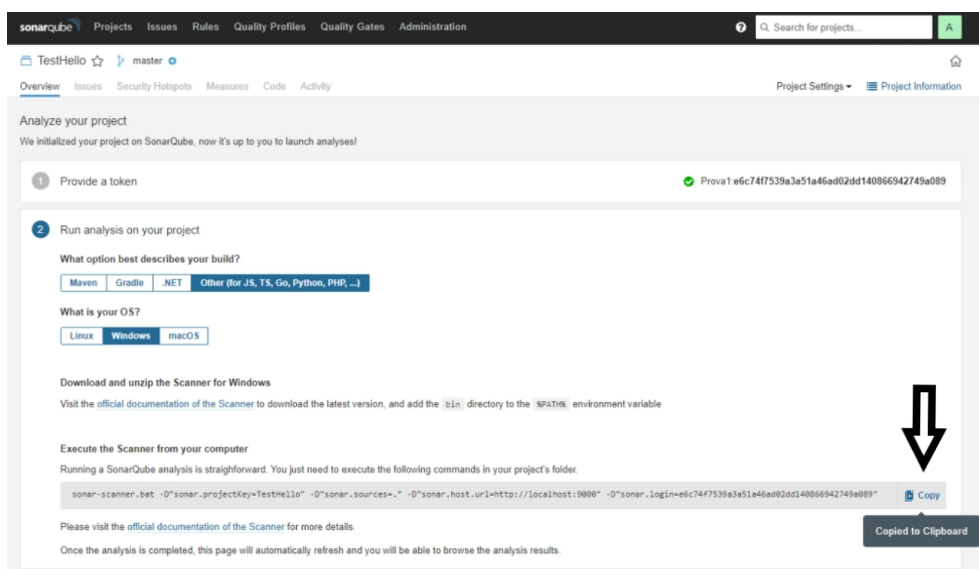


Figura A.40:

”(illustrazione creata dall'autore)”

Quando la scansione sarà completata si aggiornerà la pagina di SonarQube e ne mostrerà i risultati.

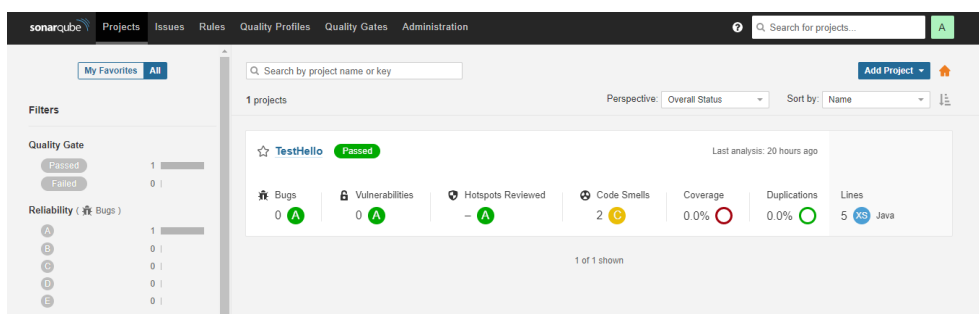


Figura A.41:

”(illustrazione creata dall'autore)”

A.7 Installare i Plug-in

I plug-in sono una parte importante di SonarQube in quanto ne incrementano le funzionalità, possiamo vederli come delle espansioni del programma di

base. Alcuni plug-in sono installabili direttamente da SonarQube in marketplace, mentre altri vanno installati manualmente. Vediamo come installarli in entrambi i modi.

A.7.1 Installazione da SonarQube

Per installare i plug-in da SonarQube sarà sufficiente andare in "Administration" dall' interfaccia principale.

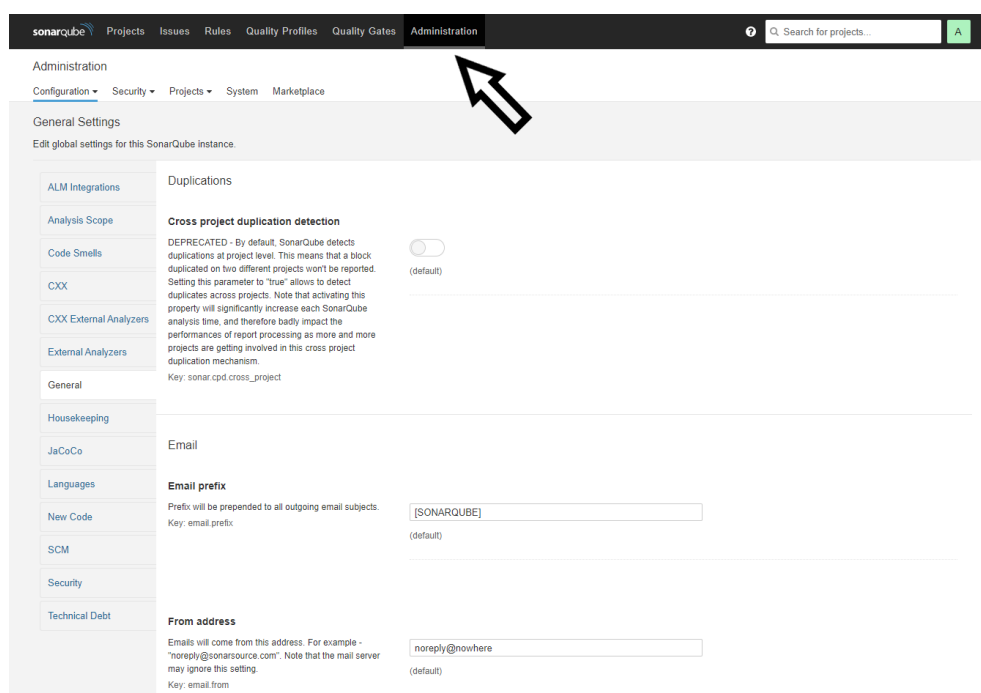


Figura A.42:

”(illustrazione creata dall'autore)”

Poi su "Marketplace"

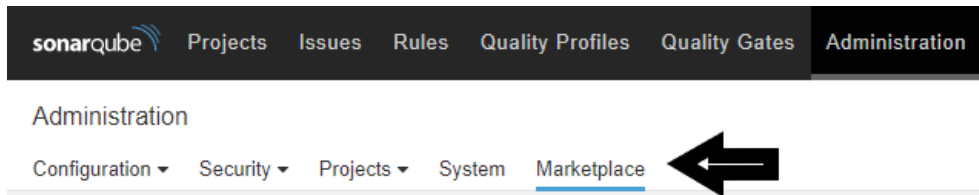


Figura A.43:
”(illustrazione creata dall'autore)”

Ed infine basterà premere ”Install” sul plug-in da installare e riavviare SonarQube, fatto ciò il plug-in sarà installato.

AEM Rules for SonarQube EXTERNAL ANALYZERS Adds rules for AEM Java development	1.3 Brought back Java 8 compatibility to allow the rules to be used outside SonarQube Server. ... Installing this plugin will also install: Java Code Quality and Security	Homepage Issue Tracker Licensed under The Apache Software LI... Developed by Cognifide Limited	Install
Ansible Lint EXTERNAL ANALYZERS Analyze Ansible playbooks.	2.4.0 Fixes some security issues and brings some new rules ...	Homepage Issue Tracker Licensed under Apache License, Versio...	Install
Apigee EXTERNAL ANALYZERS Adds XML rules test Apigee apiproxies.	3.0.1 Support for SQ 8.8+ ...	Homepage Issue Tracker Licensed under Apache License, Versio... Developed by Crédit Mutuel Arkéa	Install
Azure Active Directory (AAD) Authentication Plug-in for SonarQube INTEGRATION Allows the use of Azure Active Directory as an authentication source for SonarQube.	1.2.0 Group sync enhancements including transitive members. Code cleanup. ...	Homepage Issue Tracker Licensed under The MIT License (MIT) Developed by ALM DevOps Rangers	Install
Bitbucket Authentication for SonarQube INTEGRATION Enables OAuth delegation to BitBucket	1.1 (BUILD 381) Team restriction and account renaming ...	Homepage Issue Tracker Licensed under GNU LGPL 3 Developed by SonarSource	Install

Figura A.44:
”(illustrazione creata dall'autore)”

A.7.2 Installazione manuale

L'installazione manuale è necessaria quando il plug-in cercato non è disponibile. Per installare un plug-in manualmente bisogna inserirlo in: SONARQUBE_HOME/extensions/plugin una volta inserito si controlla da SonarQube su ”Marketplace se è installato e se tutto è stato eseguito correttamente si riavvia SonarQube ed il plug-in è installato.

88 A. Come installare SonarQube e le sue componenti su Windows

Nome	Ultima modifica	Tipo	Dimensione
bin	03/05/2021 14:03	Cartella di file	
conf	03/05/2021 14:03	Cartella di file	
data	20/05/2021 09:59	Cartella di file	
elasticsearch	03/05/2021 14:03	Cartella di file	
extensions	20/05/2021 09:59	Cartella di file	
lib	03/05/2021 14:14	Cartella di file	
logs	31/05/2021 11:29	Cartella di file	
temp	31/05/2021 11:29	Cartella di file	
web	03/05/2021 14:15	Cartella di file	
COPYING	03/05/2021 14:03	File	8 KB

Figura A.45:
”(illustrazione creata dall'autore)”

Nome	Ultima modifica	Tipo	Dimensione
downloads	28/05/2021 09:43	Cartella di file	
jdbc-driver	03/05/2021 14:03	Cartella di file	
plugins	28/05/2021 09:43	Cartella di file	

Figura A.46:
”(illustrazione creata dall'autore)”

Bibliografia

- [1] G. Ann Campbell, Patroklos P. Papapetrou (2014) *SonarQube in Action*. Manning Publications Co.
- [2] McConnell, S. (2008). *Managing technical debt*. Construx Software Builders, Inc, 1-14.
- [3] García-Munoz J., García-Valls M., Escribano-Barreno J. (2016) Improved Metrics Handling in SonarQube for Software Quality Monitoring. In: Omatu S. et al. (eds) *Distributed Computing and Artificial Intelligence, 13th International Conference. Advances in Intelligent Systems and Computing*, vol 474. Springer, Cham.
- [4] Lomio, F., Moreschini, S., & Lenarduzzi, V. (2021). Fault Prediction based on Software Metrics and SonarQube Rules. *Machine or Deep Learning?*. arXiv preprint arXiv:2103.11321.
- [5] Hitz, Martin & Montazeri, Behzad. (1996). Chidamber and Kemerer's metrics suite: A measurement theory perspective. *Software Engineering, IEEE Transactions on*. 22. 267 - 271. 10.1109/32.491650.
- [6] Arcelli Fontana, F., Mäntylä, M.V., Zanoni, M. et al. Comparing and experimenting machine learning techniques for code smell detection. *Empir Software Eng* 21, 1143–1191 (2016).
- [7] Fontana, F. A., Braione, P., & Zanoni, M. (2012). Automatic detection of bad smells in code: An experimental assessment. *J. Object Technol.*, 11(2), 5-1.

- [8] Philippe Kruchten, Robert Nord, and Ipek Ozkaya. 2019. Managing Technical Debt: Reducing Friction in Software Development (1st. ed.). Addison-Wesley Professional.
- [9] Girish Suryanarayana, Ganesh Samarthayam, and Tushar Sharma. 2014. Refactoring for Software Design Smells: Managing Technical Debt (1st. ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- [10] G. Ann Campbell. 2017. Cognitive Complexity A new way of measuring understandability. SonarSource SA. Switzerland
- [11] Adam Thornhill. 2018. Software Design X-Rays Fix Technical Debt with Behavioral Code Analysis (1st. ed.). Pragmatic Bookshelf.
- [12] Soni, M. (2006). Defect prevention: reducing costs and enhancing quality. IBM: iSixSigma. com, 19.

Sitografia:

- [13] ISO/IEC 25010. Iso. Pagina visitata il giorno: 08/06/2021.
Link: ["https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?start=0"](https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?start=0).
- [14] Emil Wallner, 42, QCon San Paolo,11-09-2019,6 Automating Software Development with Deep Learning
Link:"<https://www.infoq.com/presentations/automated-software-dl/>".
- [15] Why is code quality important? CodeGrip. Pagina visitata il giorno: 17/03/2021.
Link:"<https://www.codegrip.tech/productivity/why-is-code-quality-important/>".
- [16] SonarQube Data prima consultazione: 10/03/2021. Link:
"<https://www.sonarqube.org/>".

-
- [17] Why is code quality such a big deal for developers? Cleverti. Pagina visitata il giorno: 04/05/2021.
Link: "<https://www.cleverti.com/blog/why-is-code-quality-such-a-big-deal-for-developers/>".
- [18] Technical Debt. Techopedia. Pagina visitata il giorno: 15/04/2021.
Link: "<https://www.techopedia.com/definition/27913/technical-debt>".
- [19] ISO/IEC 25000. ISO. Pagina visitata il giorno: 07/06/2021.
Link: "<http://www.iso25000.it/styled/>".
- [20] Stockfish. Pagina visitata il giorno: 02/05/2021.
Link: "<https://stockfishchess.org/>".
- [21] Counter. Pagina visitata il giorno: 02/05/2021.
Link: "<https://github.com/ChizhovVadim/CounterGo>".
- [22] Crocodile. Pagina visitata il giorno: 02/05/2021.
Link: "<https://github.com/Virinas-code/Crocodile>".