UNIVERSITÀ DEGLI STUDI DI UDINE

Dipartimento di Scienze Matematiche, Informatiche e Fisiche

TESI DI LAUREA IN INFORMATICA

Titolo Provvisorio

CANDIDATO
Gabriele Castellani

RELATORE
Prof. Ivan Scagnetto

TUTOR AZIENDALE
Alberto Picco

Anno accademico 2021-2022

Contatti dell'Istituto
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine — Italia
+39 0432 558400
https://www.dmif.uniud.it/

Ringraziamenti

Indice

1 Introduzione

1.1 Ambito del tirocinio

Il tirocinio, su cui verte questa tesi, è stato svolto presso una IT Company che si occupa principalmente di Digital Transformation, fornendo soluzioni e servizi su misura per i propri clienti. L'azienda è divisa in 6 Business Unit specializzate nei vari settori dell'Information Technology, tra queste, quella presso cui è stato effettuato il tirocinio si occupa di sviluppo software.

1.2 Obiettivi del progetto

Il progetto consiste nell'automatizzazione del flusso di selezione tecnica dell'azienda, quest'ultima è attualmente svolta da del personale apposito (recruiter), che si occupa di testare e valutare degli elaborati consegnati dai candidati.
Per facilitare questo processo risparmiando tempo e risorse si è pensato di utilizzare una soluzione software per automatizzare in parte o del tutto il processo.
Più in dettaglio gli obiettivi espressi dall'azienda sono:

- Fornire ai recruiter uno strumento immediato ed intuitivo con il quale scremare le candidature legate all'invio di una prova tecnica, escludendo chi fallisca i requisiti funzionali minimi del test
- Ridurre il tempo necessario alla correzione dei test pratici evitando che per testarne le funzionalità di base i recruiter siano costretti ad aggiungere codice

Tra i vari ruoli all'interno dell'azienda che richiedono una verifica tecnica è stato preso in considerazione solo il ruolo di Java software engineer, dunque le soluzioni di automatizzazione dei processi di valutazione delle prove tecniche proposte di seguito faranno riferimento esclusivamente alla ~~tecnlogia~~ Java.
L'obiettivo del tirocinio è la realizzazione di una PoC che determini la fattibilità o meno della soluzione pensata. A questo scopo è stata fatta un'analisi dei requisiti, uno studio di fattibilità, una progettazione architetturale e una possibile implementazione esemplificativa.

1.3 Struttura della tesi

sezione -> capitolo

La tesi è suddivisa in sezioni come spiegato in questo paragrafo:

- Nella seconda sezione viene esposto il problema e analizzato lo stato dell'arte delle soluzioni attualmente esistenti e della ricerca accademica sul tema.
- Nella terza sezione viene descritta la progettazione dell'applicativo a partire dall'analisi dei requisiti fino alla realizzazione di una PoC.
- Nella quarta sezione vengono discussi eventuali sviluppi futuri e problematiche della soluzione individuata
- Nella quinta sezione vengono esposte le conclusioni finali sul progetto

2

Stato dell'arte

2.1 Letteratura Scientifica

Trovare il miglior candidato possibile per le posizioni lavorative disponibili è una delle maggiori sfide delle aziende informatiche moderne. In un'indagine dei due ricercatori Agarwal e Ferrat[1] vengono individuate 4 categorie di best practices diffuse tra i reclutatori presi in esame:

- definizione precisa dell'origine della necessità di assumere un programmatore
- definizione precisa delle competenze e abilità necessarie per il posto di lavoro offerto
- differenziazione competitiva dell'azienda rispetto alle altre
- eventuali incentivi offerti al momento dell'offerta al candidato.

Una assunzione che porti effetti benefici all'azienda, inoltre, deve considerare il fatto che non si tratti solo di trovare il candidato tecnicamente più abile tra tutti, quanto piuttosto la ricerca del candidato tecnicamente più consono al ruolo. Quest'ultimo dovrà infine condividere i valori e la cultura dell'azienda, altrimenti aumenta la probabilità che lasci il posto spontaneamente dopo essere stato assunto, in tal caso quell'assunzione rappresenterebbe un costo per l'azienda.
A tale proposito, uno studio condotto dal Brandon Hall Group nel 2015[4] ha stimato il costo medio per un'assunzione errata in relazione a diverse figure professionali:

Figure 6 Model for Calculating Cost of a Bad Hire Tool

Job Level	Cost of Hire	Median Compensation*	Training Costs**	Additional Variable	Additional Variable	Total Cost
Executives	$ 3,798.65	$145,084	$ 1,959	$	$	$
Mid-level managers	$ 1,501.79	$ 71,289	$ 1,369	$	$	$
Individual contributors	$ 1,062.72	$ 37,000	$ 907	$	$	$

Le "Additional Variable" menzionate in figura possono essere date da fattori quali: impatto sulle performance del team in cui l'impiegato viene inserito, perdita di clienti a causa dell'impiegato, costo per sostituire il nuovo assunto, danno all'immagine dell'azienda, varie ed eventuali.
Inoltre lo stesso studio ha evidenziato come un approssimativo sistema di intervista del candidato fosse la causa principale di un'assunzione infruttuosa.
Risulta quindi, ~~una~~ necessità di automatizzare il processo di valutazione del candidato, non solo per far risparmiare tempo ai reclutatori, ma anche per standardizzare un procedimento che può potenzialmente comportare un considerevole costo per l'azienda se condotto in maniera non collaudata.
Da uno studio sulle pratiche di reclutamento di programmatori in Russia[3] si evince, intervistando un campione di reclutatori presi da varie aziende del settore IT locale, come la fase più difficile e dispendiosa del reclutamento in termini di tempo sia appunto la verifica delle abilità del candidato. Si riporta di seguito la lista di use case, elicitati dai succitati reclutatori, per un possibile software di automazione di tale processo:

- poter cercare candidati con CV simili
- classificare gli sviluppatori in base alla pertinenza per un certo ruolo
- analizzare il codice dei candidati
- analizzare i precedenti ruoli lavorativi ricoperti
- analizzare la presenza del candidato sul web (Github, StackOverflow, ...)
- comparare diversi candidati

2.1.1 Il sistema SMSCQA

Un sistema per misurare la garanzia di qualità del codice dei candidati è stata proposto nel 2012 dal ricercatore indiano Odeh[5], che propone un sistema denominato SMSCQA (che sta per "System for Measuring Source Code Quality Assurance") per testare in maniera automatica il codice secondo 9 fattori di qualità:

- maintainability
- portability
- reliability
- reusability
- audit-ability
- readability
- understandability
- simplicity
- testability

In questo studio viene inoltre data una sommaria descrizione dell'architettura del software, che si basa su diverse componenti, come illustrato in figura:

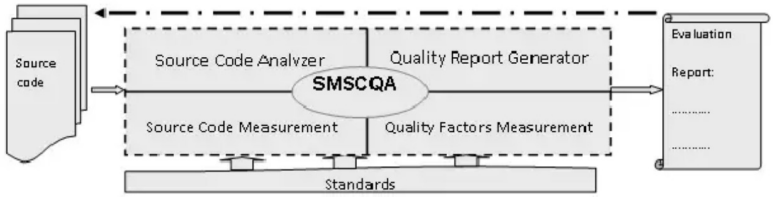


Figure-1: Architecture of SMSCQA

La valutazione del codice in questo sistema si basa sui pesi dati a 12 metriche ~~sul codice~~ (tra cui numero totale di linee, numero di spazi, numero di variabili, vocabolario utilizzato nelle procedure, ecc) e sui pesi dati a 24 criteri di qualità (tra cui complessità di Halstead, linee di codice medie per modulo, gestione degli errori, percentuale di commenti, documentazione interna, ecc).
Un algoritmo infine manipola questi pesi per assegnare un effettivo punteggio decimale in centesimi a tutti i fattori presi in considerazione e alla fine vengono prodotti 4 report. Il sistema implementato dal ricercatore si basa dunque su un set di regole deterministiche e sembra preciso nelle sue analisi, ma essendo stato testato su un campioni relativamente piccoli di codice non si possono trarre particolari conclusioni a riguardo. Inoltre in gran parte questa soluzione ripercorre gli studi di Halstead, noto ricercatore nel campo del software development, sull'analisi statica del codice per determinarne la qualità e i suoi risultati non sono universalmente accettati, infatti, ad esempio, nei suoi studi non vengono presi in considerazione fattori di efficienza misurabili solo eseguendo effettivamente il codice.
Un altro aspetto non approfondito nello studio di Odeh è la parte in cui il codice presentato viene effettivamente testato, questo procedimento viene svolto e riportato nell'articolo dai ricercatori, ma non viene posta particolare enfasi sul testing, nè viene riportato se questo sia stato automatizzato e integrato nella piattaforma oppure svolto a mano.
Dunque la parte di testing automatizzato nella valutazione del codice, nonostante venga svolta, non è in questo approccio molto considerata, in quanto si concentra più sull'analisi statica del codice per determinarne la qualità.

2.1.2 Un approccio basato sull'intelligenza artificiale

Un approccio diverso all'automazione della valutazione dei candidati potrebbe basarsi sull'intelligenza artificiale e in tale campo uno studio di due ricercatori dell'università di Malaya descrive una tecnica per individuare i requisiti tecnici ricercati nei candidati basata sul teorema di Bayes e un approccio basato su una rete neurale per classificare i programmatori in base a tali requisiti[6].
Questo approccio, come descritto nello studio, identifica con un alto grado di precisione i candidati più adatti. Lo studio si basa sull'ipotesi che avendo a disposizione dei rapporti sulle performance lavorative precedenti dei candidati si possa prevedere la loro performance nel ruolo cercato.
Si propone di utilizzare il teorema di Bayes per identificare, in base al ruolo, gli attributi più idonei, per poi estrarre, dai fascicoli dei candidati, le persone più idonee a possedere gli attributi trovati e infine classificarle tramite una rete neurale che consideri questi, dando a ciascuno il suo peso, secondo

le disposizioni aziendali.
Questo sistema, tra le soluzioni proposte, è quello che più si avvicina al modello di un software unico che possa essere usato dai reclutatori durante tutto il processo di reclutamento, sin dalle prime fasi. Tuttavia per funzionare correttamente necessita di una grande quantità di dati relativa alle performance passate dei candidati. Inoltre è necessario settare nuovamente i pesi della rete ogni volta che si sta cercando una figura lavorativa con attributi diversi da quelli su cui la rete era stata addestrata precedentemente e questa procedura è costosa.

2.1.3 La piattaforma Recruit

Un ulteriore, completamente diverso, approccio alla valutazione dei candidati seguito da due ricercatori del Royal Institute of Technology di Stoccolma è stato quello di costruire una piattaforma online, denominata "Recruit"[2].
Questa si basa su diversi strumenti forniti da Amazon Web Services (Lambda function, Amazon API Gateway, Amazon DynamoDB, Amazon Cognito, AWS Amplify) per permettere ai candidati di "giocare" a una lista di giochi tramite delle chiamate a delle API, senza alcuna interfaccia grafica. Questi giochi hanno più livelli per testare diverse abilità dei candidati.
L'articolo si concentra sull'implementazione del gioco a turni Minesweeper ed al suo utilizzo sulla piattaforma e mostra come anche questo diverso approccio al problema dell'efficacia del reclutamento abbia i suoi vantaggi rispetto alle modalità attualmente più diffuse. Questo sistema, basato su un'architettura a microservizi, permette l'aggiunta di nuovi eventi molto rapidamente ed inoltre è semplice anche introdurre nuovi giochi per testare in più modi i candidati.
Si differenzia dalle soluzioni precedentemente presentate per il suo approccio alla risoluzione del problema, che si caratterizza per una forte modularità e per la scelta di mettere al centro l'esperienza utente durante il reclutamento.

2.2 Soluzioni individuate da altre aziende

Ad oggi la maggior parte delle aziende del settore IT segue questi passi durante il reclutamento di programmatori: vengono identificate le figure lavorative atte a soddisfare gli attuali bisogni dell'azienda, si decide una strategia di reclutamento, si pubblicizza su siti di annunci di lavoro e attraverso diverse agenzie le proposte di lavoro individuate formando così una lunga lista di candidati, infine si valutano e scelgono i candidati più adatti.
Tra le strategie attualmente più diffuse per effettuare uno screening già alle prime fasi della creazione della lista dei possibili candidati c'è la consultazione di piattaforme online (Github, Stack Overflow, Bitbucket, ecc.) per determinare le competenze e l'esperienza del candidato per il ruolo cercato.
Inoltre sono pratiche comuni anche il collaborare con altre aziende e università locali per trovare dei candidati o inviare proposte di lavoro ad impiegati attualmente dipendenti per altre aziende competitor.

Per testare i candidati, invece, ci sono 4 possibilità:

- eseguire dei test/interviste in azienda ("in-house test")
- dare un test da svolgere in autonomia e consegnare entro una data di scadenza ("take-home assignments")
- usare strumenti software esterni all'azienda per eseguire dei test generici non personalizzati
- usare strumenti software interni all'azienda per eseguire dei test personalizzati

Ognuna di queste soluzioni ha i suoi pro e i suoi contro. La soluzione più adottata dalle più strutturate aziende del settore è l' "in-house test", svolto solitamente tramite interviste. Per tutte e 4 le possibilità di test l'industria ha tentato di automatizzare il processo di analisi dei risultati ottenuti al termine di questi.

2.2.1 Unity e Codility

Unity Technologies

Unity Technologies è una società di sviluppo software per videogiochi con sede a San Francisco. È stata fondata in Danimarca nel 2004 come Over the Edge Entertainment (OTEE) e ha cambiato nome nel 2007. Unity Technologies è meglio conosciuta per lo sviluppo di Unity, motore grafico utilizzato per creare videogiochi e altre applicazioni. Inizialmente l'azienda individua le posizioni lavorative per cui ha bisogno di assumere personale e le pubblicizza sul suo sito[7]. Il potenziale candidato può inviare tramite quest'ultimo la sua domanda che se viene presa in considerazione dagli addetti alle risorse umane dà inizio al processo di reclutamento. Un altro modo per entrare in Unity è venire contattati dalla stessa e, successivamente, ha inizio lo stesso percorso di reclutamento.

Il processo di reclutamento è composto di 4 macro-fasi:

- Chiamata al telefono con un reclutatore
- 30 minuti circa in cui il reclutatore conosce il candidato, presenta l'azienda e risponde a eventuali domande
- Chiamata con un responsabile delle assunzioni
- 30-60 minuti di chiamata in cui il manager approfondisce la domanda presentata dal candidato e pone domande comportamentali
- Valutazione delle competenze relative al ruolo a seconda del ruolo vengono svolti detetrminati test online seguiti da delle eventualil interviste online o in presenza
- Incontro col team assegnato

Per alcuni ruoli tecnici all'interno dell'azienda è stato scelto di utilizzare la piattaforma Codility, con lo scopo di automatizzare o quantomeno facilitare il processo di valutazione delle competenze dei candidati agli esaminatori. Si è stimato che in 90 giorni si sono presentati 750 candidati con un risparmio di circa 2200 ore tra creazione dei test e valutazione successiva di questi grazie al nuovo metodo adottato,

registrando così un concreto e sostanziale risparmio in termini di tempo.
Sul sito ufficiale di Codility viene riportato che inizialmente l'azienda Unity aveva scelto di utilizzare la piattaforma CodeCheck per valutare i loro candidati per poi passare successivamente nel 2020 alla piattaforma CodeLive.

Codility

Codility è una piattaforma che propone diverse soluzioni alle aziende per testare efficientemente i propri candidati. Il software è sviluppato per garantire la miglior esperienza utente possibile, è compatibile con molti linguaggi e garantisce una buona compatibilità anche con le librerie più diffuse per questi. Le librerie interne di test della piattaforma sono molto varie, tuttavia nessuna delle soluzioni software di Codility presenta la possibilità di creare dei test custom, dunque è necessario fare affidamento alle loro librerie di test, inoltre non è neanche chiaro il metodo di valutazione per questi test predisposti dall'azienda. Nonostante la presenza di test già precedentemente generati apposta e la possibilità di scegliere dei template di test già predisposti siano le principali caratteristiche che permettono di risparmiare tempo ai reclutatori, sono anche le stesse che rendono il procedimento limitato, senza possibilità di personalizzazione. È tuttavia degno di nota che la piattaforma produca delle valutazioni del codice controllando non solo il rispetto dei requisiti funzionali, ma anche la qualità del codice e le performance. Le soluzioni proposte sono CodeLive, CodeChallenge e CodeCheck. CodeLive è una soluzione focalizzata sul live coding, fornisce dunque una lavagna virtuale per disegnare schemi e un editor condiviso per scrivere codice anche in più persone contemporaneamente. CodeChallenge permette di creare delle sfide di programmazione aperte a tutti, per poi selezionare dalle classifiche i programmatori più abili. CodeCheck, infine, è una piattaforma che permette di creare dei task e validare gli elaborati proposti dai candidati attraverso delle batterie di test già predisposte dalla stessa. Viene messo a disposizione del candidato uno spazio virtuale in cui è possibile scrivere codice attraverso un sistema basato su Monaco, l'editor alla base di VsCode, per poi eseguire su questo una serie di test in maniera automatizzata e leggerne infine la valutazione basata su requisiti funzionali, qualità del codice e prestazioni espressa come punteggio percentuale. È inoltre implementato un sistema per controllare eventuali plagi nelle soluzioni consegnate e un sistema per scorrere una timeline delle azioni svolte dal candidato nel quesito.

2.2.2 ManoMano e Codingame

ManoMano

ManoMano è un'azienda francese operante nel commercio elettronico, specializzata in bricolage e fai-da-te, edilizia, giardinaggio, decorazione e arredo bagno. Recluta 60-80 sviluppatori all'anno. Il loro team tecnico è composto da collaboratori esterni e 232 sviluppatori a tempo pieno, che lavorano a Parigi, Bordeaux e Barcellona. Il loro processo di reclutamento è composto da 4 macro fasi:

- Chiamata al telefono o su Zoom con un talent recruiter
- Valutazione competenze tecniche (test su una piattaforma online, CodinGame, seguito eventualmente delle interviste)
- Intervista tecnica su Zoom con alcuni programmatori del team assegnato
- Intervista su Zoom con un hiring manager e incontro con il team assegnato

Per alcuni ruoli tecnici l'azienda ha valutato di poter utilizzare la piattaforma di CodinGame per testare i candidati, passano alle fasi successive del reclutamento tutti i candidati che rispondono correttamente ad almeno il 60% delle domande. L'utilizzo della piattaforma CodinGame for Work, come riportato sul sito di CodinGame, ha consentito all'azienda ManoMano di risparmiare notevoli quantità di tempo, diventando parte integrante del loro processo di reclutamento.

CodinGame

Questo applicativo consente di valutare le competenze tecniche del candidato proponendogli dei test, i quali vengono poi automaticamente valutati e viene calcolato un relativo punteggio percentuale. Questo sistema si disitngue da software simili per la possibilità di inserire diversi tipi di domande, senza dover per forza scegliere da una libreria predefinita. Gli unici test automatizzati che vengono eseguiti, però, riguardano i quesiti algoritmici e i relativi requisiti funzionali, dunque nei progetti più strutturati è comunque ancora necessario l'intervento del reclutatore per valutarli e inoltre nei test creabili viene considerato solo il rispetto dei requisiti funzionali per la valutazione. CodinGame offre due piattaforme per il reclutamento in ambito IT: CodinGame for Work e CoderPad.
CoderPad è una piattaforma per reclutatori che si concentra maggiormente sul LiveCoding e permette di:

- presentare ai candidati quesiti personalizzati in diversi livelli di difficoltà
- mettere a disposizione del candidato dei test per questi quesiti (sempre in riferimento ai soli requisiti funzionali delle domande)
- poter osservare e collaborare col candidato durante la risoluzione dei problemi presentati.

CodinGame for Work è una piattaforma che si basa si basa sull'applicativo CodinGame Assessment ed è utilizzabile dai reclutatori per inserire degli annunci di lavoro e i relativi coding test. Viene offerta al reclutatore la possibilità di personalizzare questi test o di crearne di nuovi. Nei test è infatti possibile creare e inserire dei quesiti algoritmici, per i quali si possono anche aggiungere dei basilari test oppure

dei quesiti a scelta multipla. Per i test più strutturati come i "take-home assignments", si possono creare dei quesiti a cui rispondere ma non si ha la possibilità di effettuare dei code test. Infine viene prodotto un risultato percentuale per ogni sezione del test, è possibile scorrere una timeline delle azioni svolte dal candidato nel quesito di coding ed è presente un sistema per assicurare che non ci siano plagi nelle soluzioni proposte.

2.2.3 Altre soluzioni disponibili sul mercato

Nella maggior parte delle aziende di informatica la valutazione dei candidati durante il reclutamento avviene tramite degli "in-house test", delle interviste o delle prove scritte svolte in azienda. Volendo automatizzare questo processo le soluzioni più diffuse si basano sull'affidamento a software esterni all'azienda, solitamente sono piattaforme di testing online che si focalizzano su test e sfide che durano complessivamente tra i 60 e i 90 minuti, quindi adatte a valutare le soluzioni dei candidati rispetto a problemi strutturalmente non troppo complessi. Di solito i quesiti presentati al candidato riguardano algoritmi, strutture dati e rompicapo in generale. La valutazione delle relative soluzioni proposte è pertanto quasi immediata, tuttavia esistono anche piattaforme online più specifiche per i "take-home assignments", tra le più note ricordiamo HackerRank.

HackerRank

HackerRank è una piattaforma che, nella sua versione HackerRank for work, permette di condurre delle interviste di live coding oppure di postare dei test algoritmici o dei "take-home assignments" per il candidato, con eventualmente una batteria di unit test associata, e di valutarli successivamente, controllando anche che non vi siano plagi. Il vantaggio di questa piattaforma rispetto alle altre è la sua versatilità, infatti, oltre a supportare la maggior parte dei lingauggi oggi utilizzati, attraverso il suo IDE integrato, permette di definire i propri test personalizzati e non solo prenderli da una libreria predefinita. Non vengono tuttavia considerati nella valutazione del codice fattori come la qualità di questo o l'efficienza delle soluzioni presentate, ma solamente il rispetto dei requisiti funzionali.

Confronto soluzioni individuate

Tutte queste piattaforme permettono di creare test strutturati personalizzati e forniscono un proprio metodo di valutazione della soluzione proposta, hanno tutte un'ottima integrazione con svariati linguaggi e le relative librerie/framework attualmente più diffusi e generalmente possiedono anche dei meccanismi di valutazione di qualità del codice, non si limitano ad eseguire delle suite di test. Il metodo di valutazione dei candidati che viene maggiormente utilizzato nel mercato attuale è quello di un test algoritmico/logico seguito da successive interviste tecniche. La soluzione dei "take-home assignments" è adottata da alcune aziende, ma in modo minore. Nelle soluzioni analizzate, inoltre, non viene mai menzionata la possibilità di generare dei test in automatico, questa opzione non è possibile nelle piattaforme attualmente più diffuse. Tra le aziende prese in esame, infine, nessuna ha scelto di automatizzare anche il processo del reclutamento che coinvolge le risorse umane. Nonostante esistano diversi tipi di questionari per la valutazione della personalità (es. MTBI, PAPI) e diversi software per la valutazione automatica del candidato basata su questi o simili, le aziende continuano a preferire l'interazione umana per rapportarsi col candidato, ottenendo così anche delle valutazioni più precise. Il punto debole di tutti questi software è la personalizzazione, infatti solitamente le piattaforme di questo tipo, comprese quelle analizzate nel dettaglio precedentemente, non permettono di strutturare i test a piacere, ma secondo una serie di opzioni già preconfigurate e non modificabili, ad esempio è possibile creare un proprio test personalizzato, ma solamente scegliendo le domande che lo compongono da un insieme di quesiti già predisposti dall'applicativo. Dunque per avere un software che si avvicini il più possibile ai requisiti originali e che possa scalare facilmente nel tempo, la scelta migliore è creare appositamente una soluzione personalizzata, in questo modo si abbattono anche i costi imposti dai software commerciali presi in esame.

3

Progettazione e Sviluppo

3.1 Analisi dei requisiti

Il sistema consentirà di definire batterie di test automatici a cui sottoporre i test dei candidati. Ogni suite sarà associata ad uno specifico tipo di test. I test verificheranno la corretta funzionalità tramite verifiche dell'API esposta dall'elaborato. Il sistema di testing degli elaborati verrà eseguito in un ambiente controllato (isolato per motivi di sicurezza), le applicazioni di test verranno caricate sul sistema che le valuterà, successivamente sarà possibile raccoglierne gli esiti. Il sistema terrà traccia degli elaborati caricati e permetterà di rimuoverli o modificarne le informazioni come il nome (o l'identificativo) del candidato, la data di ricezione e altre proprietà che si rendessero rilevanti. Al fine di rendere standardizzabile il test di un elaborato, andrà fornito ai candidati un'applicazione-template che aiuti ad esporre l'API richiesta e a rendere automatizzabile il suo deploy. Sono stati individuati tre ruoli che interagiscono con la piattaforma:

- Editor
- Reviewer
- User

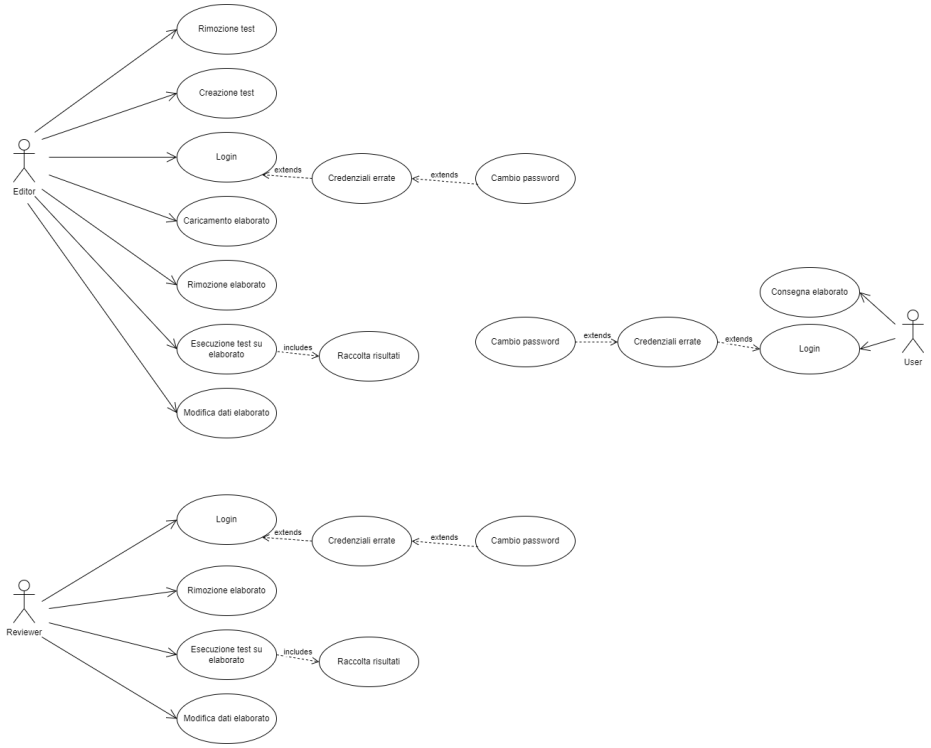
L'utente editor sarà in grado di:

- Definire e caricare una nuova suite di test per una determinata tipologia di test
- Rimuovere una suite di test
- Caricare un elaborato nel sistema
- Eseguire tutte le suite associate ad una tipologia contro un elaborato del tipo adeguato
- Rimuovere un elaborato dal sistema
- Aggiungere o modificare i dati di un elaborato

L'utente reviewer avrà un set più limitato che permetterà di:

- Caricare un elaborato nel sistema
- Eseguire tutte le suite associate ad una tipologia contro un elaborato del tipo adeguato
- Rimuovere un elaborato dal sistema.
- Aggiungere o modificare i dati di un elaborato

L'utente user, infine sarà un ruolo dedicato esclusivamente ai candidati che interagiscono con la piattaforma, e permetterà a questi solamente di inviare gli elaborati ai reviewer e editor, fornendo un numero limitato di tentativi.
È possibile rappresentare quanto detto con dei diagrammi dei casi d'uso (UML) per avere una visione complessiva del sistema dal punto di vista degli utenti:



Per separare i ruoli è necessario introdurre un sistema di login nell'applicativo. La registrazione degli utenti alla piattaforma, invece, è gestita a livello aziendale, all'esterno del software.

3.2 Modello di sviluppo

3.3 Scelte progettuali

Per valutare correttamente i candidati sulla posizione per la quale si candidano è necessario testare il codice da questi presentato in modo da considerare tutti gli attributi ritenuti rilevanti dall'azienda, considerando due aspetti: il rispetto dei requisiti funzionali, assicurato attrraverso un sistema di testing degli elaborati, e il rispetto di determinati standard di qualità del codice, assicurato attraverso un sistema di analisi del codice sorgente consegnato. Per questi processi si può pensare di utilizzare degli

strumenti artigianali, come ad esempio il sistema SMSCQA, discusso precedentemente, che è in grado si stimare la qualità di un codice anche, eventualmente, in base a regole personalizzate. Esistono, tuttavia, diversi strumenti software già predisposti per questo compito e largamente diffusi nel mercato. Buona parte di questi software, inoltre, utilizza parte o tutti gli stessi parametri del sistema menzionato, come ad esempio le metriche di Halstead, oltre che a diverse altre metriche, come ad esempio la complessità cicomatica, le metriche di McCabe o altre ancora.
In questa sezione saranno dunque analizzate e confrontate diverse soluzioni software disponibili sul mercato per entrambi gli aspetti menzionati.

3.3.1 Testing

Esistono diversi tipi di testing, che si differenziano in base agli aspetti del software che il test vuole andare a verificare e validare:

- Functional Testing:
  - Unit Testing
  - Integration testing
  - System Testing
  - Acceptance Testing
- Non-Functional Testing:
  - Security testing
  - Performance Testing
  - Usability Testing
- Altro

L'approccio al testing consisterà nel definire i test prima di disporre di qualsiasi implementazione della consegna da parte dei candidati. Esistono diverse metodologie di testing nell'ingegneria del software basate su questo approccio, ma per quanto concerne il progetto la scelta della metodologia più adatta si riduce al confronto fra due approcci: Test Driven Development (TDD) e Behavior Driven Development (BDD).

TDD: Test-Driven Development

Il Test-Driven Development(TDD) è una metodologia agile di testing implementata dal punto di vista del programmatore. Si differenzia dal testing standard perchè i test vengono scritti prima di implementare qualunque feature dell'applicazione e per questo vengono usati come riferimento durante le fasi di sviluppo. In questa metodologia un programmatore progetta e implementa manualmente i test case per ogni piccola funzionalità dell'applicazione. Questa tecnica vuole provare a rispondere a una semplice domanda: questo codice è valido?

BDD: Behavioral Driven Development

Il Behavioral-Driven Development (BDD) è un approccio di testing derivato dalla metodologia Test-Driven Development (TDD), si differenzia da questa principalmente nel flusso di lavoro che impone, cioè nel BDD prima si descrive uno scenario con una sintassi apposita e solo dopo lo si implementa con i test veri e propri. Questo approccio definisce vari modi per sviluppare una feature in base al suo comportamento. Nella maggior parte dei casi, per scrivere i casi di test si utilizza l'approccio **Given-When-Then**: si descrive inizialmente uno scenario in un meta-linguaggio basato su queste tre parole chiave, per poi trasporlo nel relativo codice di test, tenendo presente lo scenario precedentemente realizzato.
Ad esempio:
Given the user has entered valid login credentials
When a user clicks on the login button
Then display the successful validation message
Come mostrato sopra, il comportamento è illustrato attraverso uno scenario in un linguaggio inglese molto semplice, noto anche come linguaggio condiviso. A partire da queste semplici indicazioni verranno poi elaborati i veri e propri test eseguibili. Questo aiuta tutti i membri del team responsabile dello sviluppo a comprendere il comportamento della feature e a scrivere i test più velocemente.

3.3.2 Principali librerie e framework per il testing

JUnit

È un framework di testing che, come suggerisce il nome si focalizza sul fornire strumenti per il Unit Testing, è diventato nel tempo uno standard de facto per quanto riguarda questo tipo di testing, è inoltre open source e permette di implementare un approccio TDD. Fornisce: annotazioni per identificare i metodi di test, asserzioni per testare i risultati attesi e test runner per l'esecuzione dei test. I test JUnit possono essere eseguiti automaticamente e vengono eseguiti in parallelo fornendo poi un feedback immediato. I test JUnit possono essere organizzati in suite di test contenenti casi di test e persino altre suite di test. JUnit mostra l'avanzamento dei test in una barra che è verde se il test si svolge senza problemi e diventa rossa quando il test fallisce, presenta dunque un'interfaccia molto intiuitiva. Junit tuttavia non supporta il Dependency Testing e non è possibile fare dei Group Test, inoltre il report finale non viene generato automaticamente di default, ma i dati vengono semplicemente presentati in un file XML, da cui successivamente è possibile ottenere un report.

TestNG

È un framework per eseguire diversi tipi di testing, principalmente Integration Testing, ma anche Unit Testing e UI testing, permette di implementare un approccio TDD ed è open source. Storicamente sul mercato si è sempre presentato come la controparte di JUnit. É dunque sicuramente più versatile di JUnit, fornisce annotazioni per identificare i metodi di test, permette di fare dei Group Test, supporta il Dependency Testing, i test vengono eseguiti in parallelo e prevedono automaticamente la generazione di un report finale, fornendo così un feedback immediato. Tuttavia, rispetto a JUnit ha una minor integrazione con alcuni ambienti e linguaggi, ad esempio per quanto riguarda Java. Per concludere

è una delle soluzioni più complete prese in considerazione, ma è meno specifico e generalmente meno utilizzato di Junit.

Cucumber

È un framework per eseguire principalmente Unit Testing, è pensato per implementare l'approccio BDD ed è open source. Si distingue da framework simili per una sintassi molto semplice, basata sulla definizione di "scenari". Simile a Cucumber è Karate, framework basato su quest'ultimo e che permette di fare sostanzialmente le stesse cose, ma in maniera più semplice, con una sintassi ancora più intuitiva. Si focalizza su testare sistemi nella loro interezza, basandosi sulle indicazioni date negli scenari. Il grande vantaggio di questo strumento e di suoi derivati è che viene reso possibile di scrivere efficacemente le indicazioni per implementare i test praticamente in linguaggio naturale, coinvolgendo dunque nel processo di testing e valutazione del codice anche figure meno tecniche. Tuttavia è complessivamente meno versatile di altri framework, come ad esempio JUnit.

Altro

Jmock, Mockito, FitNesse, Dbunit, Jmeter, Jtest, Selenium... sono alcuni tra i possibili framework/librerie che è possibile integrare con le soluzioni precedenti per avere un risultato più completo. Ciascuna di queste ultime si concentra infatti sul testing di particolari feature del codice o particolari aspetti del testing, che non sono altrettanto approfonditi nei software sopra elencati. Tuttavia sono meno utilizzati risolvono problematiche specifiche.

3.3.3 Analisi di qualità del codice

In questo progetto per effettuare un'analisi di qualità del codice sarà eseguita un'analisi statica degli elaborati consegnati. Nell'analisi statica del codice il software preso in esame viene esaminato senza essere eseguito. Lo scopo di questo processo è di trovare problemi quali: errori di programmazione, errori della sintassi, falle di sicurezza, ecc. Il codice verrà esaminato da un apposito analizzatore, questo verificherà il rispetto delle regole di codifica definite dagli standard e delle regole predefinite personalizzate.

3.3.4 Principali librerie e framework per l'analisi di qualità del codice

SonarQube

SonarQube è una piattaforma open-source sviluppata da SonarSource per l'analisi della qualità del codice, può vantare l'integrazione con 27 linguaggi di programmazione e i più diffusi strumenti di build del codice, quali: Maven, Ant, Gradle, MSBuild. Il software è in grado di fare anche analisi dinamica del codice. Al giorno d'oggi è lo standard de facto per la misurazione della qualità del codice, è un software che produce una valutazione basata su molteplici parametri rappresentata attraverso dei report chiari, facilmente leggibili. Le analisi svolte da questo software coprono aspetti del codice come qualità e sicurezza. Permette una verifica della qualità del codice veloce e ha un'interfaccia utente molto intuitiva.

Veracode

Questo software, come SonarQube si basa su molteplici parametri per valutare il codice, ed è in grado di fare analisi sia statica che dinamica del codice, tuttavia si focalizza principalmente sugli aspetti di sicurezza e vulnerabilità. Rispetto a SonarQube i report prodotti sono meno leggibili e più verbosi, tuttavia è lo standard de facto sul mercato per la valutazione della sicurezza del codice.

3.3.5 Confronto soluzioni

Tutte le soluzioni sopra esaminate sono singolarmente ottimi strumenti per analizzare e valutare il codice, tuttavia per testare completamente un codice non basta un singolo framework o libreria, ma è necessario integrarne assieme molteplici. Dunque a seconda del tipo di test che i candidati andranno a svolgere per il reclutamento, sarà necessario costruire anche una apposita pipeline di testing, ad esempio se c'è un particolare interesse nel verificare gli aspetti di sicurezza del codice consegnato, uno strumento utile a tale scopo è Veracode, altrimenti sarà sufficiente SonarQube per valutare la qualità del codice. Per la posizione di Java Software Engineer il test proposto ai candidati riguarda solitamente problemi di tipo algoritmico, sarà dunque sufficiente utilizzare JUnit o Cucumber/Karate per la valutazione, la scelta tra questi due equivale a scegliere tra un approccio di testing TDD o BDD rispettivamente e per ragioni di efficienza e comodità sarebbe meglio l'approccio BDD di Cucumber e Karate, tuttavia l'integrazione di Junit con altri framework e librerie simili, facilmente importabili al suo interno, lo rende il candidato più valido e sopratutto il più versatile. La pipeline di testing da definire deve avere in ogni caso anche dei tool di analisi di qualità del codice appositi per controllarne al meglio la qualità e la sicurezza attraverso diverse metriche, in quanto il mero testing sui requisiti funzionali non darebbe abbastanza informazioni sull'elaborato. A tale scopo lo strumento più consono è sicuramente SonarQube. Il testing va inoltre effettuato attraverso un apposito Test Automation Server, un servizio apposito che permetta di comporre l'analisi del codice come si preferisce e che possa essere richiamato quando richiesto.

3.4 Strumenti utilizzati

Nella sezione precedente sono state discusse le possibili soluzioni per testare gli elaborati consegnati dai candidati. In questo capitolo vengono elencate e giustificate le scelte progettuali a livello di architettura del sistema.

3.4.1 Docker

Docker è una piattaforma software che permette di creare, testare e distribuire applicazioni con rapidità. Docker raccoglie il software in unità standardizzate, isolate le une dalle altre, chiamate container che offrono tutto il necessario per la loro corretta esecuzione, incluse librerie, strumenti di sistema, codice e runtime. Poiché tutti i container condividono i servizi di un singolo kernel del sistema operativo, utilizzano meno risorse delle macchine virtuali. Questa piattaforma è stata presa in considerazione principalmente per la sicurezza che aggiunge all'applicazione, attraverso l'isolamento dei container.

3.4.2 Spring boot

Java Spring Framework (Spring Framework) è un popolare framework open source per la creazione di applicazioni standalone che vengono eseguite sulla Java Virtual Machine (JVM). Java Spring Boot (Spring Boot) è uno strumento che rende più semplice e veloce lo sviluppo di applicazioni web e microservizi con Spring Framework. Spring Framework offre una funzione di dependency injection che consente agli oggetti di definire le proprie dipendenze, che il contenitore Spring inietta successivamente in essi. Ciò consente agli sviluppatori di creare applicazioni modulari composte da componenti con un basso accoppiamento, ideali per i microservizi e le applicazioni di rete distribuite.

3.4.3 Jenkins

Jenkins è un server di automazione completo e open-source che può essere usato per automatizzare tutti i tipi di attività legate alla creazione, al collaudo e alla distribuzione di software. Fornisce dei servizi di integrazione continua per lo sviluppo del software. Si distingue da competitor simili per l'integrazione che ha con gli attuali sistemi esistenti per testare e validare il codice, anche grazie alla grande varietà di plugin realizzati dalla sua community, facilmente scaricabili e installabili al suo interno. All'interno di questo progetto è stato usato per implementare l'automation server in cui che vengono richiamati gli strumenti di analisi e test del codice analizzati precedentemente.

3.4.4 Github

GitHub è un sito web e un servizio basato sul cloud che aiuta gli sviluppatori a archiviare e gestire il proprio codice, nonché a tracciare e controllare le modifiche apportate al codice stesso. Permette di gestire delle cartelle pubbliche o private. In questo progetto è stato utilizzato in supporto a Jenkins per archiviare e tenere traccia degli esempi di elaborati da testare.

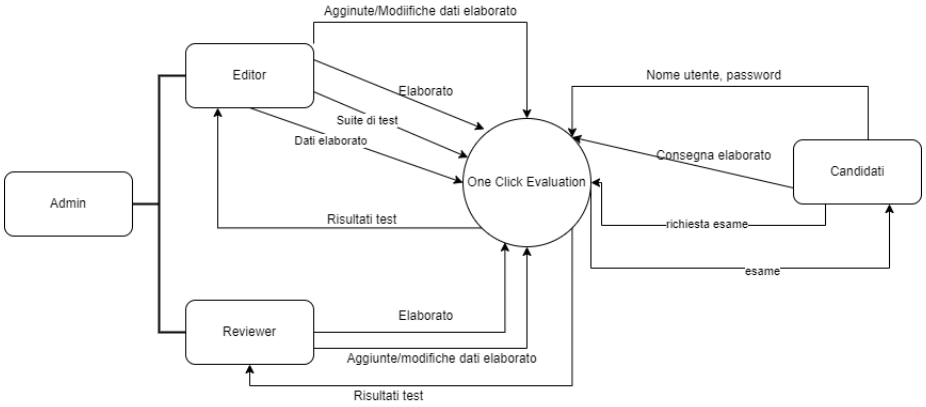
3.4.5 Postman

Postman è una piattaforma per il test delle API (Application Programming Interface) per costruire, testare, progettare, modificare e documentare le API. Si tratta di una semplice interfaccia grafica per l'invio e la visualizzazione di richieste e risposte HTTP. L'utilizzo di Postman consente di poter testare facilmente delle applicazioni web senza scrivere il codice per lanciare le richieste HTTP necessarie.

3.5 Architettura e Implementazione

Per la realizzazione della PoC sono state prese in considerazione le scelte precedenti per creare un flusso che porti a una valutazione degli elaborati consegnati. L'architettura complessiva del sistema può essere riassunta, attraverso delle viste ad alto livello, dai seguenti DFD:

Questo schema rappresenta una vista dal punto di vista di un utente, viene anche definito un "diagramma di contesto", che traccia i limiti tra il sistema e gli attori esterni ad esso. In questo diagramma vengono infatti rappresentati i flussi di dati nel modo più astratto e semplice possibile. Una vista più approfondita del sistema si può trovare nel seguente DFD di primo livello:


Agginute/Modiifiche dati elaborato
Editor
Elaborato
Suite di test
Dati elaborato
Nome utente, password
Consegna elaborato
Candidati
Admin
One Click Evaluation
Risultati test
richiesta esame
esame
Reviewer
Elaborato
Aggiunte/modifiche dati elaborato
Risultati test

4

Sviluppi futuri

5

Conclusioni

Bibliografia

[1] Ritu Agarwal e Thomas W. Ferratt. Recruiting, retaining, and developing IT professionals: an empirically derived taxonomy of human resource practices. http://portal.acm.org/citation.cfm?doid=279179.279223, 1998.

[2] Amanda Heynes e Ina Rickman. Recruit – an adaptable recruitment platform, 2021.

[3] Vladimir Ivanov, Sergey Masyagin, Marat Mingazov, e Giancarlo Succi. Recruiting software developers a survey of current russian practices In *Proceedings of 6th International Conference in Software Engineering for Defence Applications*. A cura di Paolo Ciancarini, Manuel Mazzara, Angelo Messina, Alberto Sillitti, e Giancarlo Succi. Springer International Publishing, 2020.

[4] Laurano Madeline. The True Cost of a Bad Hire. https://www.bdo.com/getattachment/fc989309-6824-4ad6-9f8d-9ef1138e3d42/the-true-cost-of-a-bad-hire.pdf.aspx?lang=en-US, agosto 2015.

[5] Ayman Odeh e Tariq Soomro. Recruiting a quality software developer. *Wulfenia Journal*, 20, 05 2013.

[6] Sorada Prathan e Siew Hock Ow. Determining the best-fit programmers using bayes' theorem and artificial neural network. *IET Software*, 14(4):433–442, 2020.

[7] Unity website. https://careers.unity.com/.