

POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Informatica



Tesi di Laurea Magistrale

Gamification per il Web Testing: Studio ed Applicazione ad un Tool di Augmented Testing

Relatori

Prof. Luca ARDITO

Dott. Riccardo COPPOLA

Dott. Tommaso FULCINI

Candidato

Lorenzo APPENDINI

Anno Accademico 2021/22

Sommario

Il mondo del Software Testing è ricco di approcci differenti tra loro, ideati nel corso del tempo per far fronte ai vari tipi di applicazioni sul mercato: il progetto di questa tesi si occupa di GUI Testing, ossia si basa sul testare la diretta interazione tra l'utente ed il front-end del sistema, cioè tutto ciò che è offerto tramite l'interfaccia grafica dell'applicazione, molto adatto e diffuso nel contesto delle applicazioni web; nello specifico, il prototipo di applicazione di web testing su cui si basa il progetto, Scout, applica il concetto di Augmented Testing, andando ad aggiungere un strato di informazioni utili al tester direttamente sul SUT (System Under Test), evidenziando per esempio gli elementi da analizzare e suggerendo delle azioni da eseguirci, prendendo le informazioni necessarie direttamente dai metadati della pagina.

L'obiettivo di questa tesi è arricchire l'appena citato software di Testing applicandovi degli elementi di Gamification, ovvero aggiungendo elementi ludici, come ad esempio punteggi, classifiche o competizioni tra gli utenti, in modo da renderne l'esperienza d'uso più piacevole e coinvolgente, in contrapposizione con la tendenza di disinteresse che affligge questo tipo di attività. Questa tesi, basata su un progetto di ricerca arrivato al suo secondo anno di attività, ha come intento primario quello di implementare altri tre elementi di Gamification inediti per studiarne l'impatto (positivo e non) sull'esperienza di Testing, oltre che migliorarne alcuni già implementati e creare un ambiente di sviluppo possibilmente migliore per le future possibili iterazioni.

Dopo aver sottoposto il tool ad un gruppo ristretto di persone, composto principalmente di studenti laureandi o neolaureati con vari profili di esperienza nell'ambito del Software Testing o comunque dell'Ingegneria Informatica, i risultati sono in linea con le aspettative iniziali: le feature di Gamification sono state percepite positivamente da tutti i soggetti, dimostrando che la sua applicazione sia effettivamente una scelta valida per quanto riguarda questo settore; allo stesso tempo però le performance e l'usabilità di Scout dimostrano alcune incertezze che spesso tendono a peggiorare l'esperienza dell'utente, essendo il tool relativamente pesante computazionalmente, richiedendo quindi delle migliorie in caso in cui questa tecnologia volesse poi approdare sul mercato.

Ringraziamenti

Ringrazio i miei Relatori Prof. Ardito Luca e Dott. Coppola Riccardo e Fulcini Tommaso per avermi permesso di intraprendere questa tesi ed avermi supportato durante il suo svolgimento. La fiducia che hanno riposto in me è stata molto importante per spronarmi ad un costante impegno in questo progetto. Spero di aver lasciato, anche solo in minima parte, un contributo per i futuri lavori che verranno proposti in questo ambito, che a parere mio non possono che fiorire ulteriormente nei prossimi anni.

Ringrazio amici, colleghi e tutti i miei cari che mi hanno accompagnato durante questo lungo percorso, dal principio fino al tanto agognato traguardo. Senza il loro supporto morale e tutti i bei momenti passati insieme probabilmente non avrei superato allo stesso modo le difficoltà incontrate in questi anni di laurea magistrale, iniziati proprio al principio della pandemia.

Indice

Elenco delle tabelle	IX
Elenco delle figure	X
Acronimi	XIII
1 Background	1
1.1 Introduzione	1
1.2 Gamification	2
1.2.1 Framework ed applicabilità	3
1.2.2 Applicazioni nell'Istruzione	13
1.2.3 Applicazioni nell'Ingegneria del Software	15
1.2.4 Applicazioni nel Software Testing	17
1.3 GUI Testing	21
2 Tool e Gamification Engine	24
2.1 Scout	24
2.1.1 Caratteristiche	24
2.2 Gamification Engine	27
2.2.1 Statistiche e Metriche	27
2.2.2 Moduli principali	30
2.3 Elementi di Gamification precedenti	32
2.3.1 Barra di progresso	32
2.3.2 Punteggio	33
2.3.3 Classifica	35
2.3.4 Easter Egg	35
2.3.5 Stella delle pagine scoperte	35
2.3.6 Login e Registrazione	35
2.3.7 Profilo	36
2.3.8 Avatar e negozio	38
2.3.9 Obiettivi	39

2.4	Nuovi elementi di Gamification	41
2.4.1	Punti esperienza e livello	41
2.4.2	Missioni	42
2.4.3	Sfide	44
2.5	Octalysis del Gamification Engine	47
3	Implementazione del Back End	50
3.1	Spring	50
3.1.1	Spring Boot	52
3.2	Progetto	56
3.2.1	Componenti	56
3.2.2	Documentazione API	57
3.2.3	Entità e tabelle	60
3.2.4	Interfaccia con il microservizio Avataars	62
4	Sperimentazione del Tool	65
4.1	Preparazione	65
4.1.1	Partecipanti	65
4.1.2	Procedimento	66
4.1.3	Questionario	69
4.1.4	Ambiente di Esecuzione	70
4.2	Valutazione delle Performance	71
4.3	Valutazione delle Plugin di Gamification	75
4.3.1	Barra di Progresso	75
4.3.2	Stella	76
4.3.3	Easter Egg	76
4.3.4	Classifica	76
4.3.5	Profilo	76
4.3.6	Obiettivi	77
4.3.7	Avatar e Negozio	77
4.3.8	Livello	77
4.3.9	Missioni	77
4.3.10	Sfide	78
4.3.11	Valutazione complessiva	78
5	Conclusioni	82
5.1	Limiti del Tool	82
5.2	Lavori Futuri	83

Appendice	83
A GOAL	84
A.1 Requisiti di business	84
A.2 Analisi del giocatore	84
A.3 Game Design	84
B GAMEX	85
Bibliografia	88

Elenco delle tabelle

2.1	Elementi di Gamification e rispettivi core di Octalysis	49
4.1	Domande del Questionario	69
1	Template per la documentazione dei requisiti di business	85
2	Tabella di analisi del giocatore	85
3	Riepilogo dell'analisi dei giocatori	86
4	Definizione delle regole di gioco	86
5	Dimensioni di GAMEX e relativi elementi	87

Elenco delle figure

1.1	Costo del bug-fixing in relazione allo stadio di produzione del software (raygun.com)	2
1.2	Definizione di Gamification tra "game" e "play" (Deterding et al. 2011)	3
1.3	Componenti di GOAL	4
1.4	Gamification Methodology di GOAL	6
1.5	Caption	9
1.6	Rappresentazione di Octalysis ed i suoi 8 punti chiave	11
1.7	Punto di vista dell'attaccante sfidato ad un duello di equivalenza in Code Defenders	19
2.1	Struttura di Scout	25
2.2	State Model	26
2.3	Schema del Gamification Engine e interfacce grafiche derivate	28
2.4	classi coinvolte nella gestione delle statistiche	29
2.5	Classi principali del plugin di Gamification	32
2.6	Classi principali del plugin di Gamification - pt. 2	33
2.7	Schermata di login	36
2.8	Schermata di registrazione	37
2.9	Profilo utente	38
2.10	Schermata delle quest	42
2.11	Feedback grafici per il completamento delle quest, rispettivamente di una quest singola e di una questline	43
2.12	Schermata iniziale del plugin di Gamification	45
2.13	Schermata della sfida selezionata	45
2.14	Classifica della sfida	46
2.15	Confronto tra il modello di Octalysis del plugin allo stato precedente (vecchio) e quello allo stato attuale (nuovo)	48
3.1	UML component diagram	51
3.2	Architettura di Spring Boot	52
3.3	Flusso dei dati in Spring Boot	53

3.4	Esempio di entity JPA e relativa tabella nel database	55
3.5	Esempio di dipendenza del MainController espressa nei confronti della classe AchievementService	55
3.6	Modello E-R di Tester ed entità correlate	62
3.7	Modelli E-R delle altre entity, pt.1	63
3.8	Modelli E-R delle altre entity, pt.2	64
4.1	Anni di esperienza con la programmazione di applicazioni web . . .	66
4.2	Linguaggi più utilizzati nello sviluppo di applicazioni web	66
4.3	Esperienza con il testing di applicazioni web (1: scarsa, 5: ottima) .	67
4.4	Procedura dell'esperimento	67
4.5	Coverage media raggiunta	72
4.6	Punteggio ottenuto	72
4.7	Nuovi widget analizzati	73
4.8	Pagine scoperte	73
4.9	Easter egg trovati	74
4.10	Insieme dei likert riguardanti gli elementi di Gamification	75
4.11	Likert sul gradimento complessivo del tool nella sua versione ludicizzata	78
4.12	Elementi ritenuti più utili durante la sessione	79
4.13	Likert sull'integrabilità del tool in contesti reali	80
4.14	Likert sulla consapevolezza indotta dal plugin	81

Acronimi

SUT

System Under Test

GUI

Graphical User Interface

AGUI

Augmented GUI

API

Application Programming Interface

IDE

Integrated Development Environment

Capitolo 1

Background

1.1 Introduzione

Il Software Testing è un processo che permette di valutare e verificare il corretto funzionamento di un'applicazione o di un prodotto software rispetto a quelli che sono i suoi requisiti. Parte integrante del processo di sviluppo software, si stima che circa la metà del suo costo totale venga investito in questa fase. Proprio per questo motivo, spesso la tendenza è quella di dedicarvi meno risorse, sia economiche che di tempo, di quante effettivamente ne necessiti, in modo da incrementare i propri guadagni o eventualmente rispettare a tutti i costi una scadenza imposta, anche al costo di rilasciare la propria applicazione con degli errori irrisolti. Come ulteriore aggravante, spesso l'attività di Testing può risultare ripetitiva e noiosa nei casi in cui sia richiesto l'intervento umano. Quest'ultima eventualità è inevitabile, in quanto ciò che caratterizza un buon test è spesso l'intuito ed una buona comprensione del contesto del programma (Fraser 2017), e tutto ciò ancora non è totalmente automatizzabile.

Nonostante da queste considerazioni risulti lampante la sua importanza, la negligenza nei suoi confronti è sempre stata un problema: si stima infatti che nel 2016, limitatamente agli Stati Uniti, circa 1.1 bilioni di dollari siano stati sprecati per dei malfunzionamenti software, coinvolgendo circa 4.4 miliardi di clienti. La maggior parte di questi si sarebbe potuta evitare, se solo fossero stati eseguiti i dovuti test. E' inoltre opportuno sottolineare quanto sia importante che le pratiche di Testing seguano il più possibile lo sviluppo del software in ogni suo stadio, in quanto si stima che il costo del bug-fixing aumenti vertiginosamente dalla scrittura del codice fino alla sua release (Figura 1.1), enfatizzando ancora di più il ruolo fondamentale della prevenzione.

Entrando nello specifico, questa tesi si occupa del *GUI Testing*, ovvero una tipologia di Software Testing che si occupa di analizzare il corretto funzionamento

di un'applicazione web o mobile attraverso l'utilizzo della sua interfaccia grafica così come farebbe l'utente finale, consistendo quindi in controlli dei vari elementi visivi che la compongono (quali bottoni, icone o aree di testo). In particolare, lo studio svolto consiste nell'implementazione di un plugin che vada ad introdurre degli elementi di *Gamification* ad un software di GUI Testing, *Scout*, al fine di capire quali di questi possano stimolare in modo efficace il lavoro dei tester che lo utilizzano. Nella sezione seguente verrà espresso il concetto di Gamification e verranno portati alcuni casi della sua applicazione in diversi ambiti affini, illustrando anche gli studi che hanno in qualche modo influenzato le scelte di implementazione prese.

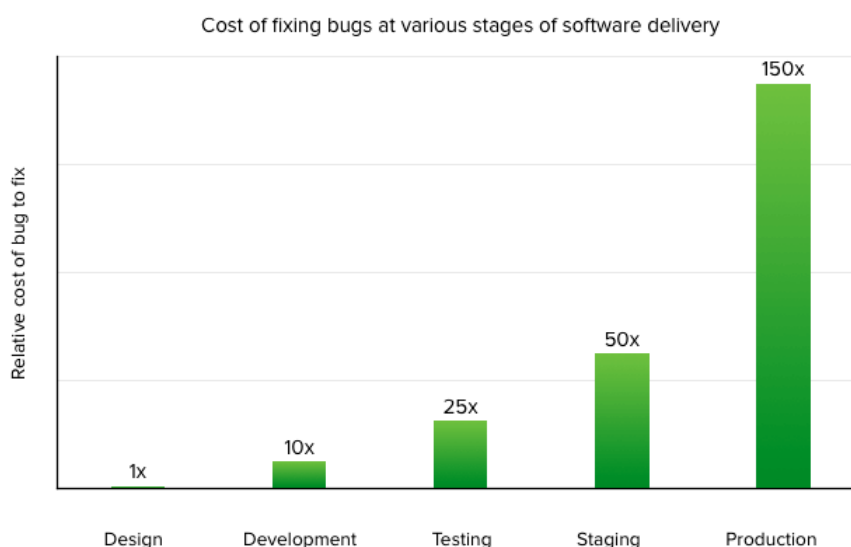


Figura 1.1: Costo del bug-fixing in relazione allo stadio di produzione del software (raygun.com)

1.2 Gamification

Il termine *Gamification* iniziò a prendere larga diffusione solo dopo la seconda metà del 2010, soprattutto all'interno dell'industria dei media digitali; nonostante questo, già dal 2008 venivano utilizzati molti altri termini nello stesso contesto, come ad esempio "giochi produttivi" o "funware". Una vera e propria definizione venne proposta a seguito degli studi descritti in (Deterding et al. 2011) come *l'uso di elementi tipici del game design in contesti non ludici*. Il concetto di base è quindi quello di integrare gli elementi che rendono un gioco interessante ed intrattenente

in contesti che necessitano questo tipo di attenzioni per essere affrontati in modo più produttivo, che siano lavorativi, scolastici o personali.

Questa idea non è la prima nel suo genere, bensì va inserita in uno spettro di fenomeni più ampio (Figura 1.2) ed è quindi utile farne una breve panoramica, evidenziando le differenze tra gli elementi che ne fanno parte.

A differenza dei cosiddetti *serious game*, ossia giochi veri e propri con finalità educative, un sistema gamificato rimane comunque un sistema non ludico nonostante le integrazioni ludiche, non essendo quindi un gioco con dei contorni "seri" bensì esattamente il contrario.

Allo stesso modo è importante differenziare il concetto di *gameful design* (alla base della Gamification) con quello di *playful design*: mentre con il termine "playful" si va ad indirizzare una sfera emotiva più giocosa, libera e fine a sé stessa, come quella dei giocattoli veri e propri, con il termine "gameful" ci si riferisce invece ad un contesto con delle regole e degli obiettivi per cui competere, come ad esempio in un gioco da tavolo. E' evidente come quest'ultima definizione sia più associabile a contesti reali in cui l'obiettivo è il completamento di una serie di attività il cui fulcro non è l'intrattenimento dell'utente, ma il cui svolgimento può essere reso più piacevole e divertente andando ad agire proprio sull'ambiente di lavoro.

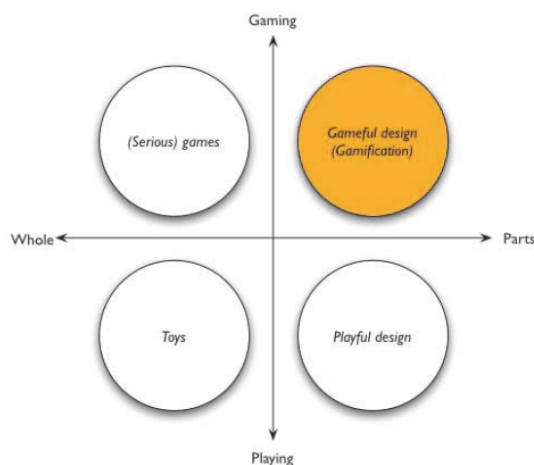


Figura 1.2: Definizione di Gamification tra "game" e "play" (Deterding et al. 2011)

1.2.1 Framework ed applicabilità

Con la crescente popolarità di questa tecnica risulta necessario, o quantomeno utile, definire delle metriche con cui poter guidare il design del proprio sistema ludicizzato e provare a valutarne l'efficacia, in modo da poterne identificare i punti di forza

e soprattutto le possibili falle che potrebbero limitarne i benefici, o addirittura renderne l'implementazione controproducente. A tal fine sono stati creati diversi framework nel corso degli anni, e di seguito ne verranno elencati alcuni; tra questi è incluso il framework principale utilizzato per determinare le scelte implementative di questo progetto, ovvero *Octalysis*.

Un primo esempio di framework è quello proposto in (Garcia et al. 2017) nel contesto dell'Ingegneria del Software, denominato *GOAL* (Gamification focused On Application Lifecycle Management): secondo gli autori infatti gli studi in questo ambito erano ancora piuttosto preliminari e vedevano l'implementazione di concetti basilari, senza ancora offrire una metodologia sistematica per integrare la gamificazione nell'Ingegneria del Software e migliorarne così fruibilità e performance. Il framework, la cui struttura è visibile in Figura 1.3, è composto dei seguenti componenti:

1. *Ontology*: studio dei concetti principali che caratterizzano il dominio di gamification in questione e delle relazioni che intercorrono tra di essi, tratti dagli articoli e libri più significativi nei campi di *systematic mapping* e *gamification*.
2. *Gamification Methodology*: processo il cui scopo è quello di guidare la ludicizzazione dell'ambiente target secondo alcuni punti ben definiti, come verrà descritto a breve.
3. *Technological Framework*: tool proposto per l'integrazione effettiva degli elementi di Gamification scelti.

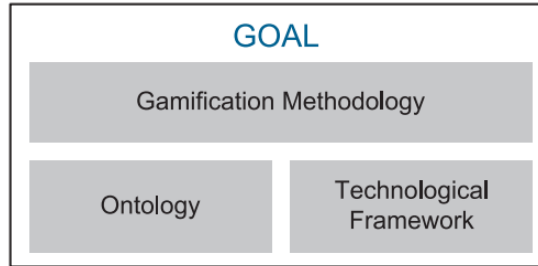


Figura 1.3: Componenti di GOAL

Per quanto riguarda il processo di Gamification vero e proprio, il framework lo suddivide nelle seguenti fasi (Figura 1.4):

1. *Identify gamification objectives*: fase in cui si stabiliscono una serie di obiettivi che si vogliono raggiungere tramite questo processo di ludicizzazione; questi dovrebbero essere scelti in base ai propri requisiti di business, alla condizione

attuale della situazione e a come la si vorrebbe migliorare. Questi obiettivi vengono poi tradotti in obiettivi *SMART* (Specific, Measurable, Actionable, Realistic, Time-bound), ovvero obiettivi raggiungibili con metriche ben stabilite. Nell'Appendice A.1 è proposto un template per la documentazione dei requisiti di business aziendali.

2. *Player analysis*: analisi degli utilizzatori a cui è indirizzato il sistema. Secondo gli autori è importante seguire un approccio di design *player centered*, ovvero costruire il proprio sistema di Gamification focalizzandosi sull'utente e tutti quelli che possono essere i suoi fattori motivanti, obiettivi e meccaniche. A tal scopo, viene analizzato sia lo *stato dell'organizzazione* (inteso come ambiente di lavoro, livello di preparazione ed età dei suoi dipendenti, etc.) che la *tipologia* dei suoi dipendenti, sfruttando le proposte dei vari articoli della letteratura di Gamification come la tassonomia in (Bartle 1996). Esempi di template proposti sono riportati nell'Appendice A.2.
3. *Scope and preliminary solution design*: viene formulata una prima proposta di soluzione di Gamification basandosi sugli obiettivi stabiliti nel primo punto, scegliendone il design sulla base degli studi condotti nel secondo punto. In questa fase viene anche fatta un'analisi sulla fattibilità di questa soluzione preliminare basandosi su fattori economici, tecnici, legali ed operativi.
4. *Gamified platform analysis and design*: fase in cui viene effettivamente costruito il design della soluzione di Gamification proposta, andando a stabilire quali elementi includere sulla base di quanto possano essere adatti al tipo di utenza che andrà ad utilizzare l'applicazione. Un concetto fondamentale è quello di stabilire delle *ricompense*, classificabili in quattro categorie quali *autostima* (competizione, miglioramento delle proprie skill, obiettivi), *divertimento* (scoperta, eccitazione, sorpresa), *socialità* (dinamiche di gruppo e feedback) ed *oggetti* (punti, moneta, risorse). Nel pratico questo si traduce spesso in una combinazione di alcune meccaniche molto diffuse quali punti, livelli e riconoscimenti (achievement).
5. *Development*: implementazione del design appena concepito all'interno del proprio software aziendale. Insieme alle fasi adiacenti è in realtà concepita come fase iterativa, in quanto il processo di sviluppo può essere lungo e complesso ed è quindi consigliabile affrontarlo in modo incrementale, potendo così migliorare costantemente l'applicazione grazie a feedback periodici (in pieno stile Scrum).
6. *Managing, monitoring, measuring*: fase molto importante in quanto consente di evidenziare eventuali problemi di implementazione attraverso un'analisi costante degli effetti che queste hanno sull'esperienza d'uso dell'applicazione.

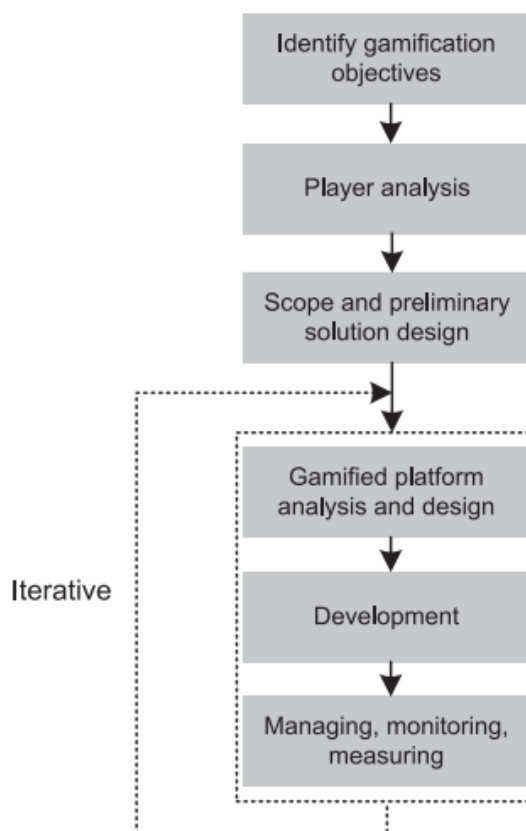


Figura 1.4: Gamification Methodology di GOAL

Un esempio successivo di framework è proprio il sopracitato *Octalysis*: quest'ultimo è descritto da Yu-Kai Chou, esperto di design comportamentale, nel suo libro (Chou 2019), dove teorizza otto *principi* che secondo lui descrivono in modo completo i modi in cui la motivazione umana può essere influenzata e stimolata, andando quindi a formare un ottagono da cui prende appunto il nome. Grazie a questo framework è possibile progettare le funzionalità di Gamification del proprio sistema in modo piuttosto specifico, basandosi su quelle che sono le proprie intenzioni ed esigenze nei confronti dell'esperienza che si vuole fornire ai propri utenti. Nello specifico, le otto meccaniche individuate sono:

1. *Epic Meaning e Calling*: motivazione suscitata dal sentirsi parte di qualcosa più grande di sé, o comunque sentirsi "prescelto" nel dover svolgere un ruolo. E' una sensazione tipica del videogioco, in cui spesso il protagonista è l'unico ad essere in grado di salvare il mondo dal male. Alcuni esempi di utilizzo di questo tipo di motivazione sono il concetto base

di *Wikipedia*, che spinge i singoli utenti a contribuire alla creazione dell'enciclopedia definitiva, oppure il progetto *SpaceX* di Elon Musk, che punta molto sul pubblicizzarsi come il progetto che porterà l'umanità ad essere una razza interplanetaria.

2. *Development e Accomplishment*: motivazione che nasce dal desiderio di migliorarsi continuamente e di ottenere degli obiettivi concreti, svolgendo attività la cui difficoltà sia adeguata alla crescita individuale; è un motore molto importante per l'essere umano, in quanto senso di esso non ci sarebbe progresso o comunque, ridimensionando il discorso, non ci sarebbe stimolo a realizzarsi nella vita.

Alcuni esempi di questo tipo possono essere la *palestra*, dove le schede di esercizi assegnate ai clienti sono studiate in modo da adattarsi al proprio livello personale e prevedono una crescita graduale dell'intensità e difficoltà, oppure il *SERP*, ovvero l'insieme delle pagine che l'algoritmo di Google posiziona per prime in seguito ad una query, simboleggiando il fatto che al crescere all'interno di questa classifica corrisponda un miglioramento della propria pagina.

3. *Empowerment of Creativity e Feedback*: motivazione che si prova nel momento in cui si riesce a risolvere un problema utilizzando la propria creatività, sperimentando soluzioni nuove e meno convenzionali. E' molto importante lasciare spazio di manovra sulle proprie azioni, ove possibile, in quanto l'essere umano eccelle nell'immaginare oltre ciò che lo circonda, nel pensare fuori dalle righe, e questo può portare potenzialmente a soluzioni migliori di quelle standard. E' inoltre importante che il feedback positivo per i risultati così ottenuti sia percepito dall'utente.

Un esempio che fa parte del substrato culturale moderno ed è quindi doveroso citare è quello di *Lego*, che offre attraverso una vasta gamma di mattoncini colorati la possibilità di costruire un'infinità di composizioni diverse, per cui l'unico limite è quello della propria creatività.

4. *Ownership e Possession*: motivazione legata al desiderio di aumentare, migliorare e proteggere la propria collezione personale, talvolta incentivando anche a cercare qualcosa di migliore. Ovviamente con collezione non si intende solo oggetti reali, bensì, soprattutto nell'ambito informatico, qualsiasi valuta o bene interno al gioco/applicazione di turno.

Gli esempi di questo tipo sono veramente ovunque, in quanto il desiderio di collezionismo, più o meno accentuato, è comunque presente nella vita quotidiana di tutti: basti pensare al *lavoro*, che benché sia desiderabile che sia affine alle proprie passioni, ha come fine primario quello di ottenere denaro. Per quanto riguarda il mondo virtuale, spesso gli utenti sono associati ad un

avatar e le varie attività che devono svolgere possono avere come incentivo quello di sbloccare loro delle personalizzazioni per il proprio alter-ego.

5. *Social Influence e Relatedness*: motivazione intrinseca nella natura umana, essendo un "animale sociale" l'uomo è spesso e volentieri condizionato dalla mentalità collettiva, e questo può alterare le proprie percezioni, opinioni e atteggiamenti. Questo concetto, in base alla propria personalità, può scaturire inoltre anche la volontà di mettersi in mostra e primeggiare sulla massa, sviluppando un senso competitivo e sbilanciando le opinioni nei confronti di chi ha successo nel proprio percorso. E' da notare però che mantenere un clima competitivo all'interno di un'attività può non essere sempre proficuo, dato che può sviluppare degli attriti e dell'ansia sociale a lungo andare; va quindi valutato come dosare elementi di Gamification di questo tipo. Anche in questo caso gli esempi sono innumerevoli: quando utilizziamo per esempio i siti di e-commerce, come *Amazon* o *Ebay*, siamo più inclini ad effettuare un determinato acquisto se il venditore ha molte recensioni positive; un altro esempio tipico è il *conformismo*, cioè il fenomeno per cui la fruizione di un'opera (che sia televisiva o letteraria) venga condizionata dalle mode.
6. *Scarcity e Impatience*: motivazione che deriva dal desiderio di ottenere dei beni esclusivi, difficili da ottenere e/o limitati nella loro quantità. E' un principio ampiamente sfruttato nel marketing, che introduce spesso *edizioni limitate* di un certo bene, disponibili solo per un certo tempo, che spingono quindi i clienti ad affrettarsi nell'acquisto o arrendersi all'idea di doverne fare a meno per un tempo imprecisato. Un altro esempio tipico è presente in molti giochi *freemium*, ovvero quasi sempre utilizzabili gratuitamente ma che nascondono delle micro-transazioni dietro ad alcune meccaniche limitanti, come ad esempio il numero di vite che dopo essere concluse richiedono l'attesa di un certo lasso di tempo o il pagamento tramite una valuta acquistabile con soldi reali.
7. *Unpredictability e Curiosity*: motivazione che sfrutta il senso di scoperta e curiosità che si prova assistendo a fenomeni imprevedibili e inaspettati. L'esempio tipico di questo principio, sfruttato più negativamente, è quello del *gioco d'azzardo*, dove l'esito di ogni gioco è denominato dall'aleatorietà. Un altro esempio tipico del mondo videoludico è quello degli *easter egg*, ovvero degli elementi ben nascosti nel modo di gioco che premiano i giocatori più attenti ai dettagli.
8. *Loss e Avoidance*: motivazione scaturita dalla paura di perdere qualcosa, è alla base di concetti semplici come la paura di fallire o le già menzionate *edizioni limitate*, è un principio di Gamification abbastanza delicato da utilizzare, in quanto può generare frustrazione e demotivare nel momento in cui potrebbero diventare vani tempo e risorse impiegati per l'ottenimento di qualcosa; una

volta evitati questi aspetti, può al contrario essere molto importante per accrescere il senso di soddisfazione di aver ottenuto un certo obiettivo o accrescere ulteriormente l'orgoglio verso ciò che si possiede.

Visualizzando l'ottagono in Figura 1.6, i suoi elementi appena descritti vengono ulteriormente classificati in due modi: dividendo la figura verticalmente vengono suddivisi in *Left Brain* (sinistra) e *Right Brain* (destra), mentre dividendola orizzontalmente vengono suddivisi in *White Hat Gamification* (sopra) e *Black Hat Gamification* (sotto).

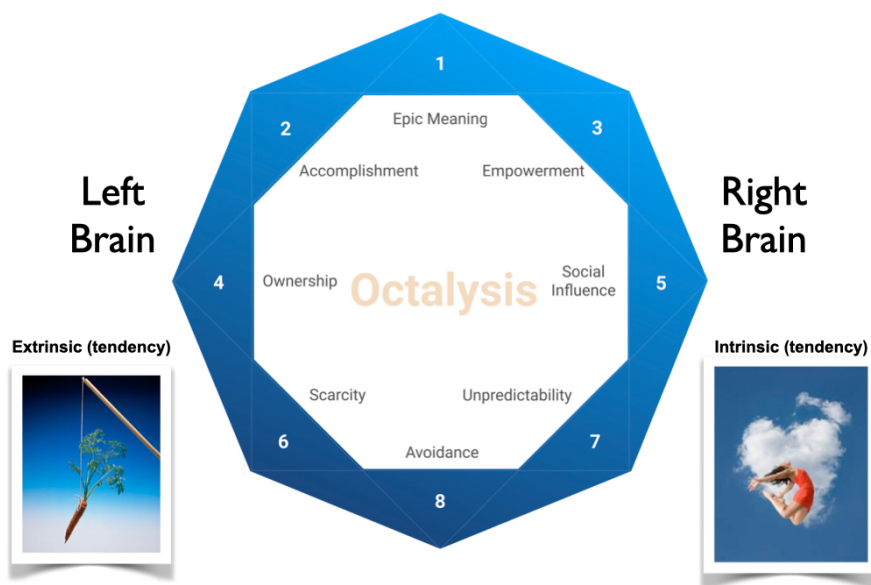


Figura 1.5: Caption

Nella prima classificazione, i principi appartenenti al Left Brain sono associati a *logica* e *possesso* e sono considerati *Motivatori Estrinseci*, cioè si basano sulla soddisfazione di ottenere qualcosa di concreto, che sia un obiettivo o un bene; i principi appartenenti al Right Brain sono invece associati a *creatività*, *espressione personale* e *fattori sociali*, e, in quanto *Motivatori Intrinseci*, sono basati sul coinvolgimento dell'attività in sé piuttosto che sul fatto che preveda una ricompensa. Nel libro viene fatto notare come la maggior parte delle compagnie tenda ad utilizzare principi del Left Brain, principalmente offrendo dei premi all'utente: secondo alcuni studi però, in caso questi motivatori vengano tolti, la produttività media tende a diminuire fino a raggiungere numeri inferiori a prima che venissero implementati. Viene quindi consigliato di progettare le attività sulla base dei principi Right Brain, rendendone la fruizione più divertente ed appagante in sé, rendendo gli utenti costantemente coinvolti nell'attività.

Nella seconda classificazione, i principi associati al White Hat Gamification sono considerati motivatori *positivi*, caratterizzati dal rendere un'attività coinvolgente permettendo di esprimere la propria creatività, dando la sensazione di star imparando sempre di più una tecnica o aggiungendo significato alle proprie azioni, facendo di fatto sentire bene l'utente; d'altro canto, se durante lo svolgimento della propria mansione si sente una costante ansia di ciò che potrebbe succedere a breve, la paura di perdere qualcosa o la consapevolezza di qualcosa che non si può ottenere, si tratta di motivatori *negativi*, associati al Black Hat Gamification. Nel testo viene sottolineato che quest'ultimi motivatori non sono sempre necessariamente negativi, semmai possono essere utilizzati sia in modo positivo, mirando a dei risultati produttivi ed eticamente sani, che in modo malizioso, mirando a manipolare l'utente al fine di massimizzare il guadagno che si può trarre da esso. Sta ad un buon designer il compito di sfruttare tutti gli elementi di Gamification che più ritenga opportuni, senza escludere a priori quelli conformi al Black Hat, in modo che l'utenza sia soddisfatta della propria esperienza.

Viene infine proposto un *Octalysis Score*, ulteriore metrica ottenuta dopo un'attenta valutazione della propria attività analizzandone l'utilizzo di ciascun principio, assegnando a ciascun di essi un punteggio da 0 a 10 ed infine sommandoli. Riguardo a questa metrica, viene sottolineato quanto non sia importante il punteggio in sé o includere necessariamente ogni possibile principio di Octalysis, bensì concentrare le proprie risorse nell'implementare bene i principi scelti e concentrarsi più sul punteggio del singolo principio, in modo da analizzare le proprie mancanze ed eventualmente correggerle.

Mentre i framework appena descritti hanno lo scopo di aiutare nella fase di design e sviluppo della propria applicazione ludicizzata, il seguente esempio riguarda invece la valutazione delle sue performance: secondo gli autori di (Eppmann et al. 2018) infatti, nonostante la crescente popolarità del fenomeno Gamification, ancora non era stato formulato un metodo per valutare l'esperienza ludica di un ambiente gamificato; fu così che nel 2018 venne sviluppata una nuova cosiddetta *gameful experience scale* (*GAMEX*), sulla base della revisione di diversi articoli della letteratura del settore e di numerosi studi successivi.

Come appena menzionato, nella fase di sviluppo di questa scala sono stati presi in considerazione un gran numero di *elementi* tipici sia del mondo videoludico vero e proprio sia dei precedenti studi di applicabilità di Gamification, come ad esempio alcune misurazioni simili ma applicate ad ambiti più specifici. Tra queste ultime le più rilevanti sono l'*immersion questionnaire* (IQ), il *game engagement questionnaire* (GEQ) ed infine il *game experience questionnaire* (GExpQ). Il problema di quest'ultimi è che sono applicabili a contesti troppo ristretti (nel caso di IQ ed GEQ sono pensati per il contesto di giochi violenti come gli sparatutto) e comunque indirizzati ad applicazioni strettamente ludiche, non essendo quindi

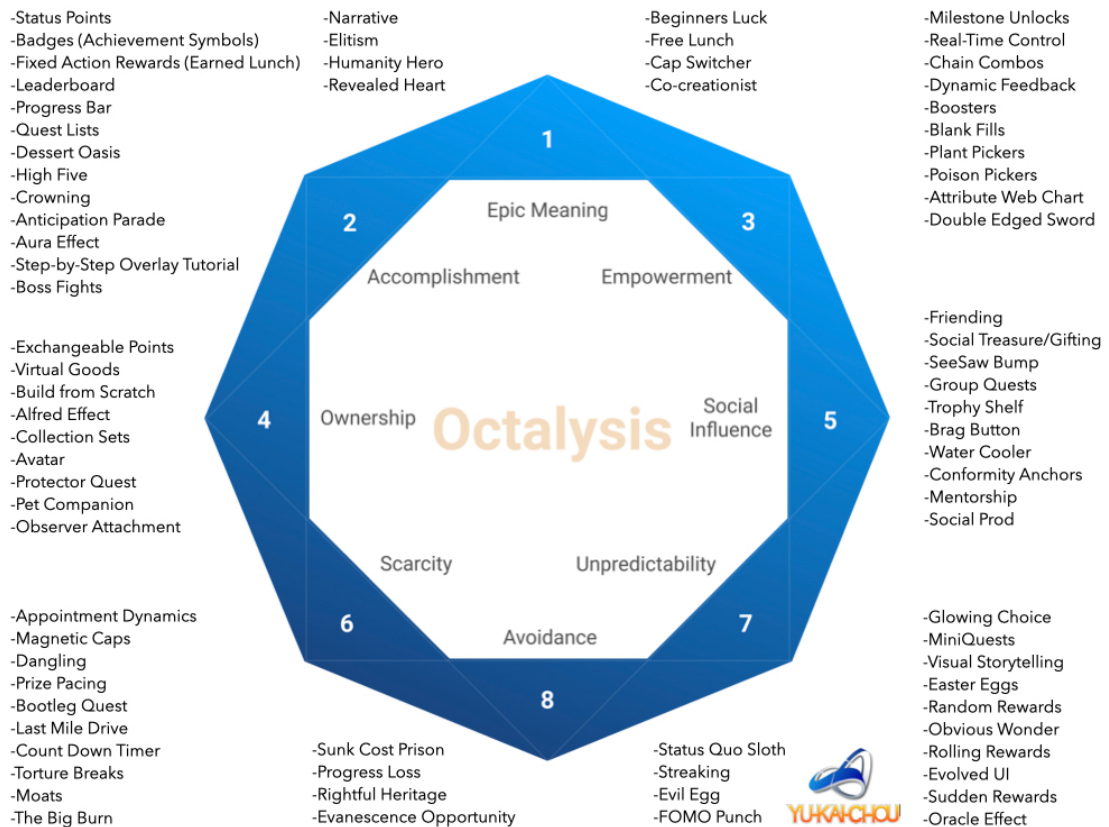


Figura 1.6: Rappresentazione di Octalysis ed i suoi 8 punti chiave

totalmente adatti ad analizzare applicazioni ludicizzate.

In definitiva, questi elementi sono stati filtrati tramite una fase di *analisi fattoriale esplorativa*, andando a formare una scala di 27 elementi, elencati nel dettaglio nella Tabella 5, visualizzabile nell'Appendice B, e suddivisi nelle seguenti sei *dimensioni*:

- *Enjoyment*: dimensione che raggruppa gli elementi positivi come *gradimento* e tendenza ad utilizzare il sistema analizzato.
- *Absorption*: dimensione i cui elementi sono caratterizzati dall'aspetto tipico dei videogiochi della *perdita del senso della realtà*, per cui l'utilizzo del sistema coglie maggiormente l'attenzione dell'utente rispetto al mondo circostante.
- *Creative thinking*: dimensione riguardante elementi come la possibilità di esprimere *libertà* e *creatività* nelle proprie azioni.
- *Activation*: dimensione i cui elementi analizzano quanto l'esperienza di utilizzo del sistema sia stata *stimolante*, sia in senso positivo che non.

- *Absence of negativity affect*: dimensione che indaga la presenza di fattori *negativi* (come la frustrazione), misurandone però l'assenza.
- *Dominance*: dimensione che analizza i fattori *sociali*, derivanti tipicamente da dinamiche competitive.

Dopo la sua formulazione, diversi studi sono stati effettuati per provarne la validità, spesso incentrati sull'utilizzo di applicazioni commercializzate da parte di un certo numero di utenti: questi hanno riportato dei risultati soddisfacenti, anche confrontandoli con gli altri questionari citati precedentemente. Viene inoltre fatto notare che spesso si tende a fare convergere molte delle feature della propria applicazione su elementi della stessa dimensione (per esempio implementando molte feature basate sulla cooperazione e sulla competitività, entrambe incluse nella metrica di *dominance*), aprendo la possibilità di studiare la correlazione tipologica con quegli elementi di Gamification. Con l'evoluzione degli approcci di ludicizzazione e con l'analisi di applicazioni diverse, per esempio quelle utilizzate in ambito scolastico, potrebbe sorgere la necessità di includere nuovi elementi nella scala o di modificarne altri, lasciando quindi quest'ultima aperta ad ulteriori studi.

1.2.2 Applicazioni nell'Istruzione

Le tecniche di Gamification si rivelano particolarmente adatte al campo dell'Istruzione in quanto spesso gli studenti hanno bisogno di approcci più coinvolgenti, soprattutto quando vengono affrontati argomenti un po' più macchinosi e con metodi arretrati. Di seguito verranno quindi presentati alcuni casi di applicazione di meccaniche ludiche in contesti scolastici.

Un primo esempio di applicazione è presentato da (Fu e Clarke 2016) come tentativo di proporre un corso di Software Testing *virtuale* e analizzarne i benefici su alcune classi dell'*Alabama A&M University*. Nello specifico, l'ambiente di apprendimento virtuale proposto si chiama *WReSTT-CyLE* (Web-Based Repository of Software Testing Tutorials - a Cyberlearning Environment), un tool di apprendimento online integrato con alcune meccaniche di Gamification che utilizza i suoi cui elementi chiave quali *interazione sociale*, *apprendimento collettivo* e *Learning Objects* (LO) al fine di supportare gli studenti nel processo di apprendimento dei concetti di Software Testing in modo efficiente ed efficace.

Il design delle feature di Gamification introdotte è pensato in modo da favorire il *coinvolgimento* e la *motivazione* degli studenti tramite le attività ludicizzate e la possibilità di iscriversi ed interagire all'interno di un apposito forum. Gli elementi chiave di questo sistema sono:

- un sistema di *ricompense a punti* che premia l'utente sia per il completamento di attività riguardanti il corso ed i Learning Objects, sia per aspetti psicosociali come l'interazione sul forum.
- un sistema di *badge* visibile nei report di fine classe.
- una *classifica* dove vengono visualizzati gli studenti con più punti esperienza, ossia tutti quelli che abbiano raggiunto il livello massimo sulla base dei punti accumulati e dei punteggi di fine classe.

I risultati raccolti in varie sperimentazioni svolte nel corso del 2014 dimostrano una relazione positiva tra coinvolgimento e motivazione degli studenti e l'ambiente di apprendimento online, specialmente nell'area del Software Testing. Vengono tuttavia evidenziati una serie di limiti che sono sorti in questi casi di studio: in primo luogo, i Learning Objects sono stati presentati agli studenti per un lasso di tempo troppo breve, di circa tre settimane. In secondo luogo, i dati raccolti sono insufficienti per valutare appieno come gli studenti utilizzano il tool WReSTT-CyLE, ed andrebbero raccolti per periodi superiori al singolo semestre e possibilmente per insegnanti differenti. Viene inoltre fatto notare che sarebbe opportuno integrare ulteriori elementi di Gamification, dato che ad esempio limitatamente ai punti non c'è alcuna valutazione collaborativa dei dati riguardante punti e voti dei test. Infine

viene specificato che non sono ancora stati inclusi fattori psicologici in questi studi.

Un ulteriore caso di applicazione viene descritto in (Barata et al. 2013) e consiste in uno studio su un corso di *Multimedia Content Production* presso l'*Istituto Superior Técnico* di Lisbona della durata di due anni accademici, il primo dei quali prevedeva che il corso fosse svolto normalmente mentre il secondo prevedeva invece una versione ludicizzata di quest'ultimo.

Il corso è stato svolto attraverso una piattaforma Moodle, in modo da sincronizzare i dati raccolti nel corso dei due anni. Il punto di partenza posto dai risultati raccolti nel primo anno (non ludicizzato) consisteva in una scarsa partecipazione da parte degli studenti, una bassa percentuale di presenze ed uno scarso utilizzo del materiale fornito (misurato in termini di download dei file). A partire da questo background, l'approccio di Gamification scelto è stato quello di introdurre elementi ludici come i *punti esperienza* (XP), *livelli*, *classifiche*, *sfide* e *obiettivi*.

Ogni studente inizia quindi il suo percorso partendo dal livello minimo, e ad ogni attività proposta gli viene attribuito un certo quantitativo di punti XP, in modo da dare una gratificazione istantanea allo studente. Raggiungendo determinate soglie di punti lo studente può quindi salire di livello, ottenendo un titolo adeguato al livello raggiunto. Tutto ciò viene integrato in un unico punto focale di Gamification, la *classifica*, in cui è possibile visualizzare un elenco ordinato degli studenti in base a livello e punti esperienza acquisiti. Per quanto riguarda gli *obiettivi*, essi riguardano azioni piuttosto comuni nel contesto del corso, come ad esempio seguire le lezioni o completare sfide, e sono anch'essi visibili ad ogni utente tramite la classifica. Ultime ma non meno importanti, le *sfide* costituiscono un altro dei motori principali del corso, consistendo in una serie di azioni che lo studente deve eseguire per ottenere punti XP e obiettivi. Queste possono riguardare argomenti più teorici, venendo proposte ad esempio in conclusione di una lezione, o attività di laboratorio.

I risultati ottenuti in conclusione di questo esperimento sono piuttosto incoraggianti, includendo un numero di download del materiale didattico e di post inviati sul forum del corso di gran lunga superiore durante l'anno scolastico ludicizzato rispetto a quello normale, con conseguente aumento delle presenze durante la prima metà dell'anno. In generale, il feedback ricevuto dagli studenti è stato molto positivo, dato che nel questionario a cui sono stati sottoposti la maggior parte ha dichiarato di aver seguito il corso con più continuità rispetto agli altri, trovandolo più intrattenente e motivante.

1.2.3 Applicazioni nell'Ingegneria del Software

Un altro campo in cui l'applicazione della Gamification risulta sinergica è quello dell'Ingegneria del Software, in quanto lo sviluppo di software è un processo potenzialmente lungo che richiede una costante interazione dell'utente con il sistema; può essere quindi potenzialmente utile rendere queste operazioni più stimolanti e coinvolgenti nei confronti degli sviluppatori.

Uno dei primi approcci proposti in questo ambito viene teorizzato e proposto in (Sheth et al. 2011) con il nome di *HALO* (Higly Addictive, socialLly Optimized). L'idea iniziale degli autori di questo articolo era studiare una nuova tendenza videoludica che stava nascendo in quegli anni, cioè il fenomeno MMOG (Massive Multiplayer Online Game), i quali prevedevano la presenza di un gran numero di giocatori e di missioni che richiedevano la collaborazione di quest'ultimi, creando un senso di soddisfazione e progressione collettivo contagioso ed intrattenente; uno tra tutti, per esempio, World of Warcraft, MMORPG (MMO role playing game) i cui server sono ancora attivi e che riceve tuttora aggiornamenti, a prova del successo che questo genere ha mantenuto nel corso degli anni.

Le due dinamiche principali che sono state prese da questo genere videoludico sono il sistema di *livelli* e le *quest*: queste ultime in particolare sono sfruttate per aggiungere uno strato ludico ai task che normalmente un utente dovrebbe svolgere nelle sue mansioni di sviluppatore, ricevendo poi delle ricompense come punti esperienza, valuta interna, titoli o effettive ricompense reali. Queste possono essere inoltre molto diverse tra loro sia in termini di lunghezza che di difficoltà: alcune sono più brevi e semplici, altre sono organizzate invece in sequenze, con alcuni passaggi eventualmente opzionali, ponendo all'utente la scelta di saltare il passaggio facoltativo e completare la sequenza più velocemente oppure portarlo a compimento per delle ricompense extra.

All'atto pratico, HALO dovrebbe essere implementato come un plugin al servizio degli IDE; in contesti simili, le quest possono riguardare azioni semplici come risolvere un bug o più complesse come ad esempio adattare del codice ad una piattaforma differente. Nel caso di quest più difficili viene data la possibilità, similmente a quanto accade negli MMO, di affrontarle formando un gruppo di più persone, alimentando quindi il senso di collaborazione che rende così appetibile il genere.

Come citato precedentemente, a fronte del completamento di certe condizioni, come ad esempio la conclusione di un certo numero di quest o il rilevamento di molti bug, l'utente viene ricompensato con dei *titoli*, dei prefissi o suffissi da aggiungere al proprio nickname per mostrare agli altri utenti i propri ottenimenti più importanti e prestigiosi; si rivela quindi essere un ulteriore fattore sociale che invoglia l'utente ad ottenere performance migliori per ottenere un riconoscimento collettivo.

Questo esempio di applicazione è particolarmente interessante in quanto introduce il concetto di quest, il quale nonostante sia stato introdotto così "presto" nella letteratura della Gamification non ha visto molte applicazioni in altre implementazioni di questo tipo, per poi venire rivisitato nel progetto di questa tesi, come verrà spiegato successivamente.

Un esempio successivo di applicazione viene descritto in (Berkling e Thomas 2013), in questo caso come tentativo di preparazione di un corso di Ingegneria del Software ludicizzato, sperimentato poi su alcune classi di studenti universitari al secondo anno dell' *Università Statale Cooperativa di Baden Württemberg* (Germania). Questo corso consisteva nel dividere gli argomenti trattati in tre *pilastr*i principali, ciascuno dei quali è ulteriormente diviso in tre argomenti sequenziali: *sviluppo del Software* (Design Patterns, Metriche, Testing), *comunicazione* (Definizione dei requisiti, Stima del lavoro necessario, Reverse Engineering) e *gestione di progetto* (Processi, Gestione della configurazione, Gestione del Lifecycle), terminando infine con un progetto vero e proprio.

L'applicazione è stata implementata tramite il framework *Vaadin*, il quale permette di creare un'applicazione web in Java con una conoscenza minima di HTML, CSS o JavaScript: l'applicazione risultava molto pratica per l'uso da parte degli studenti nei brevi tempi che avevano a disposizione, tuttavia risultava scarna dal punto di vista estetico in quanto molto simile ad una applicazione desktop. Il concetto chiave dell'applicazione è quello dei *sentieri*: dopo aver effettuato il *login*, lo studente può scegliere in piena autonomia quale strada percorrere, andando sostanzialmente a percorrere i tre macro-argomenti descritti poc'anzi. L'*obiettivo* ultimo è quello di completare almeno due su tre macro-argomenti, in modo da sbloccare l'accesso alla possibilità di creare un gruppo ed iniziare a sviluppare il proprio progetto.

La piattaforma di Gamification è strutturata in modo da permettere all'utente di visualizzare la propria situazione attuale, i propri progressi e la situazione degli altri studenti, in modo da dare la possibilità di interagire tra loro (sia in modo collaborativo che competitivo). Tutto questo è attuato condensando questi elementi su di una mappa che va a costituire il menu principale dell'applicazione. Ogni area di apprendimento mostra quindi una serie di *task* che l'utente deve completare per passare al *livello* successivo, potendo poi visualizzare la propria situazione complessiva tramite una *barra di progresso*. Ad ogni livello completato lo studente ottiene dei *punti*; quest'ultimo ha inoltre la possibilità di chiedere aiuto ad altri studenti, garantendo loro dei *punti esperienza* visibili nel proprio profilo, il cui fine dovrebbe essere quello di incentivare l'altruismo in quanto porta ad un riconoscimento collettivo. Dopo aver completato due dei tre pilastri del corso gli studenti hanno accesso ad un *project marketplace*, all'interno del quale possono proporre un proprio progetto e cercare di reclutare dei collaboratori o

unirsi a progetti già esistenti, formando quindi dei gruppi con cui collaborare a lungo termine.

Al termine del corso sono state raccolte le opinioni degli studenti tramite un questionario interno all'applicazione: su 90 studenti, 59 sono quelli che hanno contribuito al sondaggio, portando risultati inaspettati che dimostravano una concezione apparentemente distorta della Gamification. In primo luogo, nonostante molti studi statistici confermino la larga diffusione del media videoludico, solo il 18% degli studenti delle classi coinvolte dichiarava di giocare regolarmente tutti i giorni, mentre più del 50% giocava meno di una volta a settimana. Sono inoltre sorte delle criticità su quelle che erano le *aspettative* nei confronti dell'applicazione, le quali si sono dimostrate più sul puro *divertimento* che sui reali scopi che il sistema si poneva, quali il raggiungimento di *obiettivi* e di una buona *autonomia* di studio. Riguardo a quest'ultima, molti studenti hanno espresso una preferenza per la conoscenza degli argomenti rilevanti per esame, piuttosto che sulla libertà di fruizione.

In generale, il feedback complessivo sulla piattaforma è stato piuttosto divisivo, raccogliendo sia feedback positivi che negativi in egual misura; nessuno di questi riguardava gli elementi di Gamification, ma piuttosto la facilità di reperire il materiale necessario ai fini dell'esame. Questo caso di applicazione non si è dimostrato quindi un successo come sperato, tuttavia è stato un tentativo di implementare molti dei sistemi di ludicizzazione più diffusi nella letteratura, e questo ha messo in evidenza molti fattori utili ai fini delle future implementazioni di questo tipo.

1.2.4 Applicazioni nel Software Testing

La fase di Testing è, come già menzionato nell'introduzione, una fase piuttosto delicata in quanto è dispendiosa sia in termini di tempo che di sforzi da parte degli sviluppatori, ma allo stesso tempo è importante per la correzione e prevenzioni di bug che potrebbero compromettere ore e ore di lavoro. Per rendere più fruibile e meno tedioso questo processo, che consiste in una serie di operazioni spesso ripetitive e poco stimolanti, sono molti gli approcci che sono stati utilizzati; tra questi, ovviamente, vi è anche la Gamification, e lo studio della sua applicabilità a quest'ambito è il fulcro di questa tesi. Di seguito verranno esposti alcuni esempi.

Un primo caso di applicazione può essere trovato nell'ambito del *mutation testing*, una tecnica caratterizzata dal cercare di fornire una stima pratica della capacità di rilevare errori di una test suite, permettendo quindi al tester di perfezionarla e renderla più flessibile. La popolarità di questa tecnica è ostacolata dal fatto che qualsiasi software, che non sia estremamente basilare, tende a generare un gran numero di mutanti, molti dei quali risultano troppo *banali* o *equivalenti*: nel primo

caso, questo porta alla perdita di interesse da parte del tester e a domandarsi l'efficacia della tecnica stessa; nel secondo caso invece l'utente si trova dinanzi ad un problema la cui risoluzione può rivelarsi molto difficile e frustrante. Ed è proprio in questo contesto che è stato proposto Code Defenders (Rojas e Fraser 2016), un gioco web-based che si pone l'obiettivo di rendere l'attività del tester più interessante aggiungendo uno strato ludico competitivo. Il target originale del tool è il linguaggio Java, in particolare i test case JUnit.

Il concetto del gioco, relativamente semplice, consiste nel dividere l'utenza, che si suppone di almeno due tester, in due gruppi:

- gli *attaccanti*, il cui scopo è quello di creare varianti del programma sotto test (dette appunto *mutanti*), con l'obiettivo di mettere alla prova le test suite associate a quelle classi, guadagnando punti in caso queste riescano effettivamente a "sopravvivere" ai vari controlli.
- i *difensori*, il cui scopo è invece quello di creare unit test in grado di individuare (nel contesto del gioco *eliminare*) i mutanti creati dagli attaccanti, cercando quindi di rendere le test suite le più robuste e flessibili possibile.

Il gioco si svolge a turni, per cui attaccanti e difensori si alternano, rispettivamente, nel proporre mutanti efficaci e test suite in grado di individuare questi ultimi. Un'altra dinamica chiave del tool, rivolta al problema sopracitato dei mutanti equivalenti, è quella dei *duelli* (Figura 1.7): nel momento in cui un difensore sospetta che un mutante ricevuto possa essere equivalente, questo può decidere di segnalarlo all'attaccante, sfidandolo a duello. L'attaccante può quindi accettare la dichiarazione di equivalenza, perdendo punti, o accettare il duello e proporre a sua volta una test suite in grado di rilevare il proprio mutante, guadagnando punti extra e facendone perdere al difensore. In questo modo viene incentivata un'attenta analisi da parte dei difensori, i quali devono essere sufficientemente sicuri delle proprie dichiarazioni.

A seguito della teorizzazione ed implementazione del tool, in (Fraser et al. 2019) ne viene descritto un tentativo di integrazione all'interno di un corso scolastico di software testing: oltre allo svolgimento normale delle lezioni sono stati aggiunte delle sessioni settimanali di Code Defenders, all'interno delle quali gli studenti venivano messi in competizione nel modo descritto precedentemente. In generale, i feedback ricevuti da questa esperienza sono molto incoraggianti: oltre a registrare un'elevata produttività e partecipazione da parte degli studenti, questo impegno si è poi concretizzato per la maggior parte di questi in miglioramenti nel corso del periodo di prova del tool, registrando anche una correlazione importante tra il bacino di utenza più attivo e quello di chi ha ottenuto i voti migliori.

Un altro caso di applicazione può essere invece individuato nel contesto della *tracciabilità* del software, ovvero l'abilità di poter collegare un artefatto software

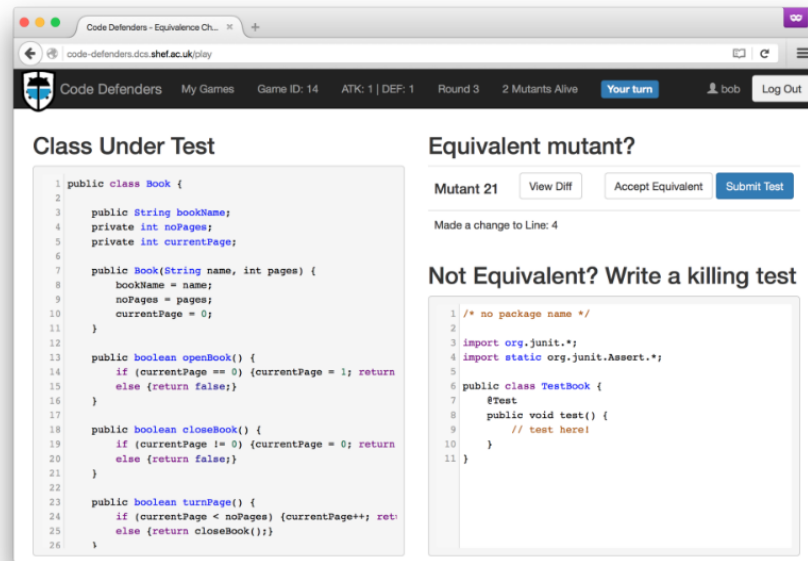


Figura 1.7: Punto di vista dell'attaccante sfidato ad un duello di equivalenza in Code Defenders

ad un altro tramite una dipendenza. Nel caso del testing, la tracciabilità *test-to-code*, ovvero la relazione tra una test suite ed il codice target, può essere molto importante per individuare e risolvere i problemi che possono sorgere durante il ciclo di vita dell'applicazione, soprattutto in caso quest'ultima cresca in complessità o sia poi implementata in sistema basati principalmente su software, richiedendo che questo sia altamente affidabile e certificato. Nello studio in (Reza Meimandi Parizi et al. 2015) vengono messi in evidenza alcuni problemi dei sistemi di tracciabilità più diffusi al momento: questi infatti spesso non sono completamente esaustivi, limitandosi a recuperare collegamenti tra unit test e classi, tralasciando informazioni più dettagliate ed utili come ad esempio i metodi coinvolti. Il problema forse peggiore è però che il processo è comunque altamente *human-intensive*, e in quanto tale richiede molto tempo, risorse ed un maggior supporto di visualizzazione nei confronti del tester. Gli autori della ricerca hanno quindi teorizzato un'*agenda* di ricerca al fine di individuare le feature principali a cui puntare con l'introduzione di un framework di tracciabilità ludicizzato, individuandole in

- *self-adaption*: il sistema dovrebbe essere in grado di adattarsi autonomamente in base alle situazioni incontrate dall'utente, anche considerando le possibili dinamiche di Gamification introdotte.
- *predictive feature*: il sistema dovrebbe monitorare le attività dell'utente e

rielaborare i dati raccolti al fine di aiutare quest'ultimo con delle operazioni guidate o semi-automatiche, in modo da ridurre le risorse umane necessarie.

- *visualization feature*: l'utente dovrebbe poter tenere traccia dei collegamenti individuati per ciascun test case, visualizzando quest'ultimo come il *root* di un grafo ad albero e tutti gli artefatti software come nodi ad esso collegati.

Le ricerche appena citate hanno poi visto implementazione, come spiegato in (R. Parizi 2016), nel prototipo di framework *GamiTracify*: sviluppato in Microsoft Visual Studio 2013, si basa principalmente sul fatto che durante la scrittura dei test gli sviluppatori siano ben consci dei metodi e delle classi coinvolte nel test case, ed è quindi proprio in corso d'opera che questi dovrebbero mantenerne aggiornate le informazioni di tracciabilità. Invogliando l'utente ad aggiornare volta per volta queste informazioni, attraverso il framework di Gamification, risulta quindi molto più semplice mantenere una buona tracciabilità dei test, senza bisogno di verifiche o di operazioni di recupero successive. Per quanto riguarda le feature ludiche introdotte, queste includono

- un sistema di *punti*, i quali ricompensano l'utente nell'immediato.
- un sistema di *feedback* che consente all'utente di rendere partecipe sé stesso e gli altri dei propri traguardi.
- un classifica basata sulla *reputazione*, la quale aggiunge una componente competitiva e sociale al framework.
- la possibilità di avere un *avatar*.
- un indicatore del *tempo trascorso*, utile per tracciare i propri progressi.
- delle *sfide/quest*, dei task pensati rispettivamente per essere conclusi in breve tempo o essere particolarmente ardui.

Questo framework è stato poi messo a confronto con un altro approccio di tracciabilità non ludicizzato, SCOTCH, tramite una sessione sperimentale in cui sono stati coinvolti dei membri di un gruppo di sviluppatori con ruoli ed esperienze differenti ma comunque affini sia allo sviluppo software che al testing: nel complesso, le performance ottenute con il tool ludicizzato sono di gran lunga superiori rispetto a quello di base, consolidando l'idea che un possibile approccio a questo settore sia appunto migliorarne il coinvolgimento nei confronti degli utenti.

1.3 GUI Testing

Il GUI Testing è, come già menzionato precedentemente, una disciplina del Software Testing che consiste nell'analisi dell'applicazione tramite la sua interfaccia grafica e nel controllo delle funzionalità che i vari elementi di quest'ultima forniscono all'utente finale. Gli approcci con cui questo viene eseguito sono principalmente due:

- quello *manuale*, che richiede l'intervento diretto del tester nell'effettuare tutti i controlli del caso. E' il metodo più classico e permette una pianificazione accurata e una semplice organizzazione, anche in caso di modifiche in corso d'opera (dato che comunque prevede l'interazione umana), tuttavia può risultare molto costoso (sia in termini di tempo che di denaro) e lungo, richiedendo potenzialmente molti tester in caso di applicazioni su larga scala.
- quello *automatizzato*, nato per sopperire alle debolezze del metodo manuale appena citate, il quale consiste nella creazione di script da parte del tester e nella loro esecuzione automatica. Quest'ultimi possono essere scritti con degli opportuni linguaggi di programmazione (spesso supportati da librerie dello stesso linguaggio dell'applicazione target, come ad esempio la libreria Jest che offre utili funzioni di *mock* per eseguire Unit Testing delle applicazioni web sviluppate con React.js), oppure creati in modalità *Record & Replay*, in cui il tester esegue manualmente le operazioni e le registra, in modo che possano essere poi ripetute automaticamente ogni qualvolta sia necessario. Quest'ultimo approccio è sicuramente più efficiente, permettendo di risparmiare molto in termini di risorse umane, ma presenta anch'esso alcuni problemi: oltre a poter essere potenzialmente costoso nella sua integrazione, è sicuramente meno flessibile e richiede comunque l'intervento umano in caso di *Regression Testing*, ovvero di testing su più versioni dello stesso prodotto, le quali possono aver subito alcuni cambiamenti più o meno importanti, richiedendo quindi la modifica dei test automatizzati che altrimenti potrebbero risultare invalidi sulla nuova versione.

Nell'analisi svolta in (Alégroth et al. 2015) gli approcci di GUI Testing esistenti vengono classificati cronologicamente in *tre generazioni*, differenziate principalmente dal metodo di *localizzazione* degli elementi del sistema sotto analisi:

- la *prima* è caratterizzata dall'utilizzo delle *coordinate* esatte dell'elemento, estratte durante l'interazione con l'applicazione e successivamente riutilizzate tali e quali nelle fasi di regression testing. Questo metodo risulta ovviamente molto costoso nel suo mantenimento, dato che dipende da fattori estremamente variabili, come eventuali cambiamenti effettuati sulla GUI, o dipendenti

dall'ambiente di utilizzo, quali ad esempio la risoluzione. Per questo motivo è ormai in disuso.

- la *seconda* è invece caratterizzata dall'interazione con il GUI model tramite i cosiddetti *widgets*, i quali raccolgono proprietà e valori dei componenti, rendendo l'approccio di GUI testing più robusto a cambiamenti, più performante e adatto a processi di *recording* ed *automatizzazione*. Per fare tutto ciò, questa tipologia di tool ha necessità di accedere alle librerie della GUI sottostante del sistema sotto test, andando a limitare l'applicabilità a linguaggi di programmazione specifici ed utilizzando determinate API fornite dal GUI.
- la *terza* generazione, anche definita come *Visual GUI Testing* (VGT), è infine caratterizzata dall'utilizzo di algoritmi di *image recognition*, i quali permettono di interagire ed eseguire controlli sulla correttezza dell'applicazione in esame tramite la sua GUI così com'è renderizzata sullo schermo del tester. Nonostante sia concettualmente molto promettente non è anch'essa esente da problemi di robustezza ed applicabilità.

Nello studio svolto in (Linares-Vásquez et al. 2017), rivolto particolarmente al testing di applicazioni mobile, i tool di GUI Testing vengono invece classificati in base al metodo di *generazione dei test* stessi, individuando diverse categorie; tra di queste, quelle più consolidate e diffuse sono le seguenti:

- *Automation Framework and APIs*: questi tool offrono all'utente delle interfacce che consentono di ottenere la struttura grafica dell'applicazione (gerarchia di componenti/widget che la compongono) e di simulare l'interazione dell'utente con essa. Tipicamente la definizione dei GUI test viene quindi permessa tramite la scrittura o la registrazione di script, nei quali viene definita una sequenza di azioni che verranno eseguite sull'ambiente di test e da una serie di asserzioni sul risultato di queste ultime. Nonostante permettano la scrittura di test in molti casi compatibili su più piattaforme diverse, il fatto che ci siano casi limite o situazioni in cui alcune azioni più complesse non vengono gestite, oltre che ad un elevato costo di mantenimento all'evolversi dell'applicazione, ha fatto sì che molti sviluppatori optassero per metodi differenti.
- *Record & Replay Tools*: come già menzionato precedentemente, questo tipo di tool mira a ridurre le risorse umane e di tempo necessarie a definire dei test case, permettendo la creazione di script di test significativi anche a persone con meno esperienza con la scrittura di test convenzionale. Alcuni approcci R&R offrono inoltre delle opzioni di cattura molto precise (per esempio in termini di latenza di interazione con il sistema), permettendo di simulare situazioni molto precise con buoni risultati. Al contempo però è necessario sottolineare che più questi tool utilizzano strutture specifiche di alcuni dispositivi o sistemi

operativi per permettere interazioni precise, più viene messa in secondo piano l'effettiva interazione con i componenti del GUI; al contrario, usando metriche più ad alto livello può aumentare la flessibilità del tool a discapito della precisione.

- *Automated Input Generation Tools*: altra tipologia di tool indirizzata a ridurre il lavoro umano dietro al testing, questa è tuttavia basata sul concetto di testare alcune tipologie di applicazioni fornendole una serie di input differenti in modo automatico. A sua volta, in base alla metodologia di definizione di questi ultimi, la categoria può essere ulteriormente divisa nelle sotto categorie di *random-based*, *systematic* e *model-based* input generation. Nonostante i progressi ottenuti nel corso degli anni, numerose ricerche hanno sottolineato le problematiche di questo metodo, tra cui il costo del riavvio dell'applicazione, il bisogno di inserire manualmente alcuni input specifici in casi più complessi o la gestione di effetti imprevisti che potrebbero alterare le differenti run dell'applicazione, richiedendo quindi ulteriori studi.

Date queste premesse, una plausibile direzione può essere quella di investire studi e risorse al fine di sopperire alle mancanze del testing *automatizzato*, in quanto risulta sempre più necessario nel momento in cui le applicazioni odierne sono sempre più grandi e complesse. Nel corso degli anni sono stati apportati diversi miglioramenti al GUI Testing grazie alla creazione di nuovi approcci; in particolare, nella prossima sezione verrà analizzata la tecnica su cui si basa il tool scelto per questo studio, ovvero l'applicazione dell'*Augmented Testing* in *Scout*.

Capitolo 2

Tool e Gamification Engine

2.1 Scout

Prima di descrivere le feature di Gamification teorizzate ed implementate in questo progetto è opportuno illustrare il software di Testing su cui verranno utilizzate, chiamato *Scout*: software prototipale parzialmente open-source sviluppato in Java da ricercatori del Blenkinge Institute of Technology (Nass e Alégroth n.d.; Nass, Alégroth e Feldt 2020), esso è basato sulla tecnica dell'*Augmented Testing*, approccio relativamente nuovo il cui obiettivo è migliorare la comunicazione tra il tester ed il sistema in modo da aumentare l'efficienza e l'efficacia del GUI Testing.

Il concetto di base dell'*Augmented Testing* è testare il cosiddetto *System Under Test* (applicazione target) per mezzo dell'*Augmented GUI*, il quale contiene delle informazioni in sovrimpressione al *SUT GUI* (l'interfaccia grafica dell'applicazione). Questa ulteriore interfaccia grafica può essere utilizzata in vari modi per semplificare il lavoro del tester, evidenziando ad esempio delle aree e dando così dei consigli all'utente, monitorandone al tempo stesso le interazioni ed utilizzando questi dati per imparare come effettuare quelle operazioni e verificare le condizioni.

2.1.1 Caratteristiche

L'architettura dell'applicazione, visibile in Figura 2.1, può essere suddivisa in quattro componenti principali:

- l'*Augmented GUI*, unica interfaccia grafica proposta all'utente, a discapito dell'effettiva interfaccia grafica dell'applicazione sotto test; tutte le interazioni effettuate sull'AGUI vengono propagate sull'effettiva SUT GUI tramite un *driver*, che nel nostro caso di applicazioni web è *Selenium*.
- un modulo utilizzato per registrare gli scenari di test; anziché esprimerlo tramite script come la maggior parte degli altri strumenti di test, Scout

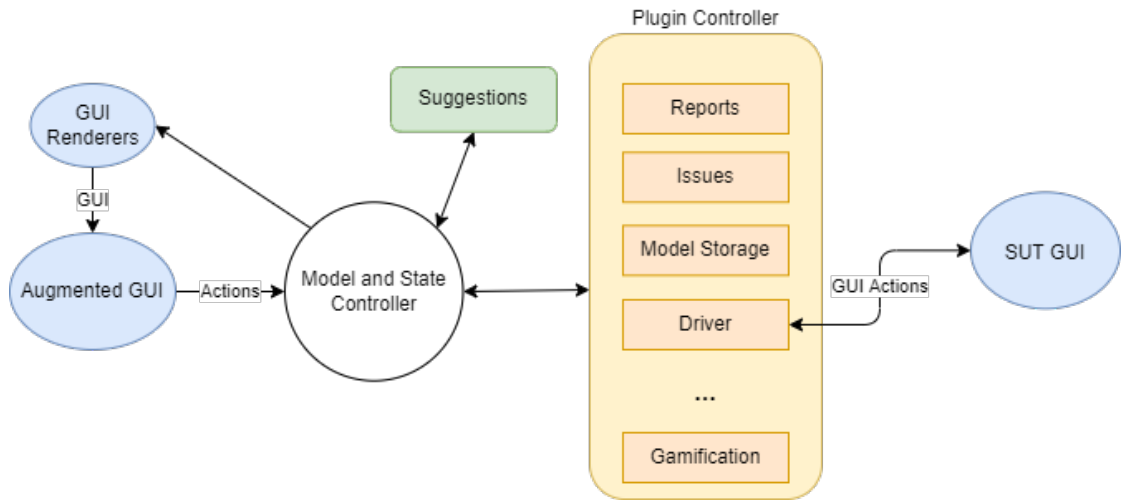


Figura 2.1: Struttura di Scout

utilizza uno *State Model* (Figura 2.2), ovvero un grafo pesato in cui ogni *nodo* rappresenta lo stato dell'applicazione (incluso ogni parametro rilevante quali memoria interna, dati utilizzati ed interazioni eventuali con applicazioni esterne), mentre i rami che li collegano rappresentano *azioni*, cioè le varie interazioni dell'utente quali pressione del mouse o input da tastiera. Questo State Model è gestito dallo *State Controller*, il quale svolge il ruolo cruciale di sincronizzazione tra AGUI e SUT GUI.

- un layer di algoritmi di machine learning che sfruttano le informazioni raccolte al fine di generare dati e suggerimenti per l'utente.
- un *Plugin Controller* il cui ruolo è quello di gestire tutti i plugin open-source che definiscono di fatto il cuore delle funzionalità dell'applicazione; nel nostro caso, tra di questi vi è appunto il plugin di Gamification.

Quest'ultimo è uno dei motivi per cui questo tool risulta così interessante anche accademicamente: nonostante le librerie interne di Scout siano private, gli stessi sviluppatori hanno reso possibile l'implementazione di plugin personalizzati tramite interfaccia ben definite, al fine di incentivare un lavoro collettivo per migliorare efficacia ed efficienza dello strumento nel corso del tempo, e nel caso di questo progetto risulta particolarmente comodo data la possibilità di personalizzare l'ambiente di test a proprio piacimento (nei limiti imposti dal linguaggio).

Volendo descrivere in modo più tecnico le feature fondamentali del tool, queste consistono in:

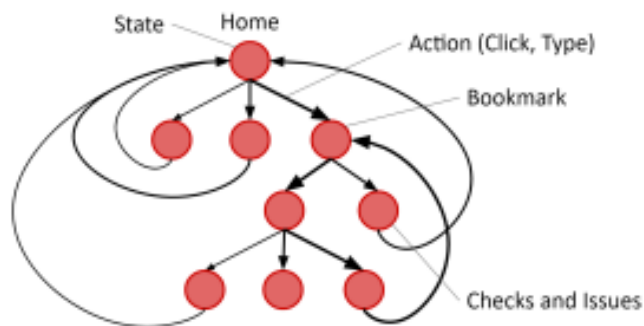


Figura 2.2: State Model

- *Record and Replay scriptless (R&R)*: la registrazione permette all'utente di sviluppare rapidamente scenari di test tramite l'interazione con il SUT, approccio ampiamente abbracciato nel corso degli anni; tuttavia Scout implementa il R&R in modo differente dagli altri tool, differenziandosi fondamentalmente per due aspetti. In primo luogo, i test vengono registrati direttamente nella struttura dati del tool, invece di comporre degli script come la maggior parte degli altri strumenti simili. Inoltre, per come Scout gestisce l'intera operazione, è possibile ricavare informazioni più dettagliate a partire dai widget. Nello specifico, il tool definisce dei cosiddetti *multi-locator* dei widget: mentre nella maggior parte dei sistemi che sfruttano Selenium vengono utilizzati dei *single-locator*, definiti da un XPath, un widget-ID e/o altre informazioni univoche, Scout utilizza più di un locator per ogni widget.
- *Suggerimenti (machine learning)*: Scout gestisce ogni input come uno scenario a sé stante, implicando che input testuali o numerici vengano registrati parallelamente nella struttura dati dello State Model. Questa scelta di design permette delle analisi incrociate al fine di identificare i pattern degli input dell'utente ed estrapolarne dei suggerimenti su input futuri compatibili con quest'ultimi. Questa particolare analisi è permessa dal plugin *Mimic*, il quale cerca di imparare il comportamento del SUT nel corso del tempo facendo *reverse engineering* dalla risposta del SUT nei confronti del comportamento dell'utente. Sono presenti anche altri plugin che svolgono analisi simili, come per esempio quello che analizza i *casi limite*, il quale mira a identificare debolezze o falle nel SUT fornendo vari tipi di input.
- *Plugin Open source*: come già scritto precedentemente, uno dei fattori chiave di Scout è proprio la possibilità di aggiungere plugin, in grado di amplificare e modificare le funzionalità di base dello strumento, sfruttando delle API fornite dagli sviluppatori.

- *Automazione del Testing Web-based*: siccome il tool è basato su *Selenium*, è nativamente indirizzato verso la *web-based automation*; questa caratteristica è fornita da un plugin, implicando la possibilità di estendere questo comportamento ad altre piattaforme. Il Web Testing è supportato in due modi: il primo è quello *scriptless* descritto precedentemente, mentre il secondo consiste nello sfruttare il tool per registrare dei test script di Selenium, i quali possono essere esportati dall'applicazione.
- *Report e Continuous Integration*: il tool fornisce diverse funzionalità richieste anche in ambiti più estesi, come nei software utilizzati a livello industriale; tra queste vi è la possibilità di generare dei *report* dei casi di test superati o falliti, l'integrazione con ambienti di *Continuous Integration* utilizzando dei comandi remoti REST ed altre ancora.

2.2 Gamification Engine

La progettazione di questo plugin è nata con la tesi di Tommaso Fulcini "Gamification Applicata al GUI Testing di Applicazioni Mobile" e successivamente portata avanti nella tesi di Davide Gallotti "Ideazione e prototipazione di un sistema di gamification per il testing di applicazioni web". Prima di illustrare gli elementi di Gamification implementati e le scelte che ne hanno influenzato la realizzazione, è opportuno spiegare come il package contenente tutte le funzionalità aggiunte, il *Gamification Engine*, sia integrato con il resto dell'applicazione (Figura 2.3), e le principali classi in esso definite.

2.2.1 Statistiche e Metriche

Un aspetto importante per l'applicazione è quello di tener traccia delle *statistiche* di sessione ed utilizzarle per fornire all'utente delle *metriche* con cui possa stabilire le performance della propria sessione, e gestirne inoltre la conservazione in modo che l'utente possa valutarne l'andamento nel corso di sessioni successive. Nello specifico le metriche prodotte sono:

- *coverage calcolata sulla singola pagina*, il cui calcolo verrà illustrato nei capitoli a seguire.
- *coverage media* della sessione, calcolata appunto come media delle coverage delle singole pagine. Si noti che le pagine con coverage nulla non contribuiscono alla media, in quanto l'utente potrebbe aver acceduto a quelle pagine erroneamente o comunque essersi accorto successivamente di non avere interesse nell'analizzarle.

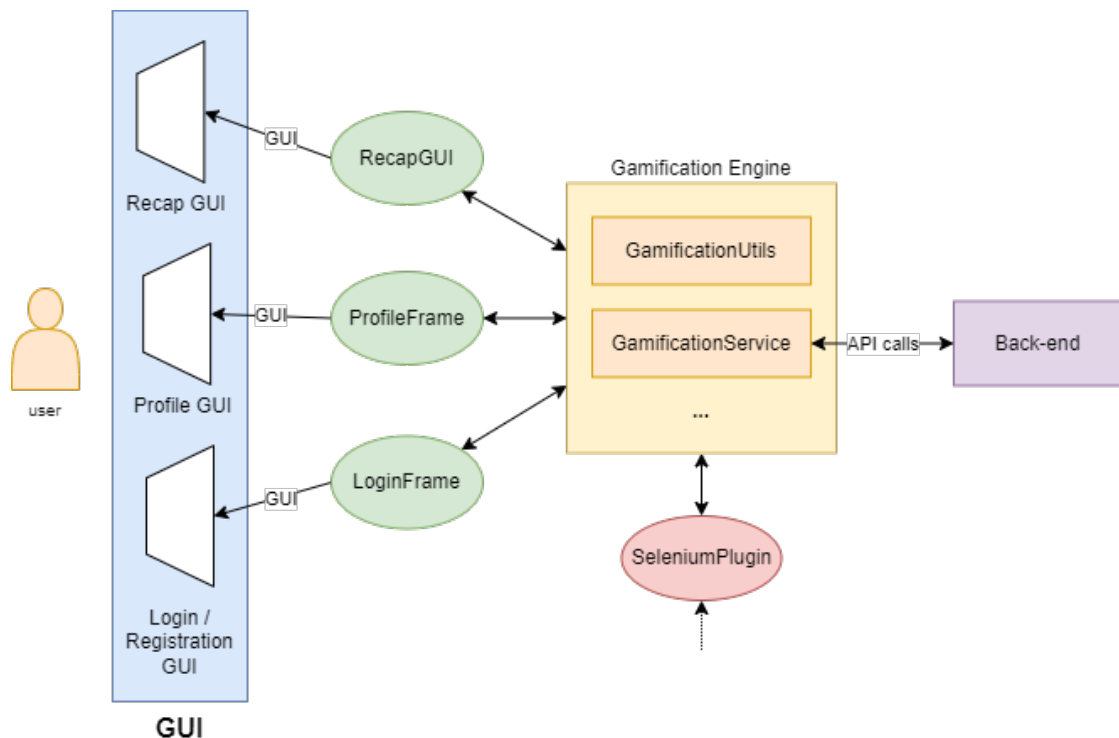


Figura 2.3: Schema del Gamification Engine e interfacce grafiche derivate

- *tempo medio per iterazione*, calcolato come rapporto tra la durata della sessione ed il totale delle interazioni eseguite.
- *numero medio di interazioni per pagina*, calcolato come rapporto tra il totale delle interazioni avvenute ed il numero di pagine visitate.
- *numero di pagine scoperte*, ovvero le pagine visitate per la prima volta nel corso di una certa sessione. Si noti che risulta quindi necessario salvare l'elenco delle pagine scoperte ed aggiornarlo di conseguenza ad ogni nuova pagina scoperta, e questo viene fatto su un apposito file *pages.txt* (come si può vedere in fig. 2.4).
- *numero di widget scoperti*, definiti in modo analogo alla metrica precedente; in questo caso è possibile tracciarli dato che per ogni pagina viene tenuto un elenco dei widget con cui l'utente ha interagito, eventualmente aggiornato ad ogni nuova interazione.
- *numero di issue*, ovvero problemi riscontrati nel sistema segnalati dall'utente.
- *numero di easter egg trovati*, la cui natura verrà spiegata successivamente.

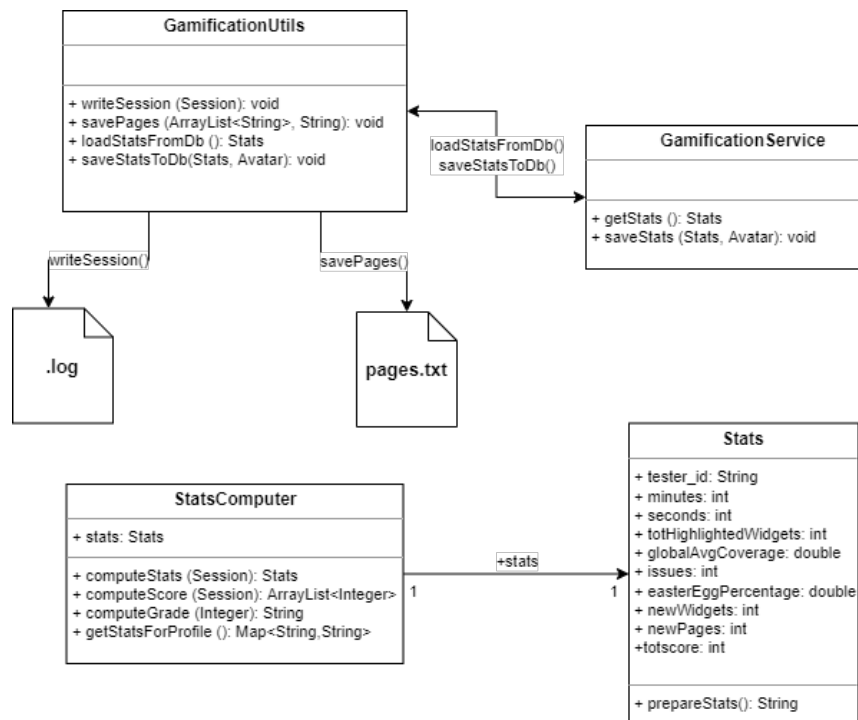


Figura 2.4: classi coinvolte nella gestione delle statistiche

Per quanto riguarda le statistiche, esse vengono gestite all'interno dell'applicativo tramite la classe *Stats*, come illustrato in Figura 2.4. All'inizio della sessione di Testing, non appena l'utente accede al sistema di Gamification, viene istanziata una classe singleton *StatsComputer*, il cui compito è essenzialmente quello di gestire un oggetto della sopracitata classe *Stats* che sia coerente con il contenuto del database; a differenza delle precedenti iterazioni del progetto quest'ultimo è collocato nel back-end, necessitando quindi di richiedere i dati tramite le sue API esposte: questa operazione è gestita dalla classe *GamificationService*, la quale contiene appunto una serie di funzioni che vanno ad eseguire le chiamate al back-end, tra cui appunto i metodi *saveStats()* (che si occupa di inviare al back-end le statistiche aggiornate alla sessione corrente) e *getStats()* (il cui scopo è invece ricevere dal database le statistiche salvate). Questi ultimi due metodi vengono di fatto chiamati rispettivamente dai metodi *saveStatsToDb()* e *loadStatsToDb()* della classe *GamificationUtils*, i quali fungono da wrapper.

Per quanto riguarda i metodi di *GamificationUtils* elencati in figura e non ancora descritti, essi consistono in:

- *writeSession()*: questo metodo permette di salvare tutte le interazioni che l'utente ha effettuato durante la sessione appena conclusa, andando a creare

un file di log personale (il cui nome è composto tramite il *TesterName* scelto ad inizio sessione e dal timestamp di fine sessione).

- *savePages()*: questo metodo si occupa di aggiornare il sopracitato file *pages.txt* al fine del tracciamento delle pagine scoperte.

Per quanto riguarda invece *StatsComputer*, i metodi forniti consentono di computare le statistiche di fine sessione ed aggiornare il file *Stats* di conseguenza (metodo *computeStats()*), di calcolare il punteggio e voto a partire dalle statistiche appena raccolte (metodi *computeScore()* e *computeGrade()*), con dei criteri che verranno descritti dettagliatamente in seguito, ed infine di restituire una Map delle statistiche che vogliono essere mostrate nel *profilo* (metodo *getStatsFroProfile()*).

2.2.2 Moduli principali

Il punto cardine della precedente iterazione del progetto è stata proprio l'introduzione di un *profilo utente*, e con esso l'introduzione di una classe *Avatar* che contenesse tutte le informazioni essenziali mostrate al suo interno. Tutto ciò che consegue dalla sua introduzione, oltre che tutto ciò che è stato aggiunto ad esso in questa iterazione, verranno spiegati nelle apposite sezioni a seguire; è tuttavia utile eseguire una panoramica sulle classi principali che vengono inizializzate con la sessione di Testing (Figura 2.5, 2.6).

1. Per prima cosa, prima dell'inizio effettivo della sessione, viene inizializzata la classe singleton *LoginFrame*, la quale ha il ruolo di generare un GUI in cui l'utente possa inserire le proprie credenziali ed effettuare il login al plugin di Gamification.
2. In caso sia la prima volta che si utilizza il framework, o in caso si voglia creare un nuovo profilo, il frame permette di navigare in un altro frame apposito, creato dalla classe *RegisterFrame*, tramite la pressione di un apposito bottone. Entrambi i frame appena citati andranno poi ad utilizzare rispettivamente i metodi *login()* e *register()* del modulo *GamificationService* per effettuare la connessione con il back-end.
3. Una volta che il login viene effettuato con successo, vengono subito reperiti una serie di dati relativi all'utente, utili a popolarne il profilo, tra cui *Achievement*, progressi di *livello* ed *avatar* (immagine utente), sempre utilizzando i relativi metodi di *GamificationService*; oltre a queste, viene anche inizializzato lo *StatsComputer*, come descritto nella sezione precedente.
4. Dopo aver inizializzato la classe *ProfileFrame*, il GUI del profilo viene mostrato all'utente tramite il metodo *getProfile()*. A partire da quest'ultimo, è possibile

accedere a delle sotto-sezioni del profilo tramite i metodi *getShop()*, *getQuests()* e *getChallenges()*.

5. Nel primo caso, il GUI relativo al negozio riceve i dati necessari tramite le funzioni *getShop()* e *getShopped()*, sotto forma di oggetti della classe *ShopItem*.
6. Nel secondo caso, il GUI relativo alle quest riceve i dati necessari grazie ad una classe singleton *QuestController*, la quale viene inizializzata anch'essa successivamente al login, e il cui ruolo è quello di reperire le quest dell'utente ed aggiornarne il progresso nel momento in cui vengano effettuate le giuste interazioni con il SUT, tramite rispettivamente i metodi *getQuests()* e *progressQuest()* di *GamificationService*, le quali sono rappresentate con oggetti delle classi *Quest* e *LongQuest*.
7. Nell'ultimo caso, il GUI relativo alle sfide riceve i suoi dati con il metodo *getPerformances()* di *GamificationService* sotto forma di oggetti della classe *ChallengePos*, rappresentanti le sfide a cui ha partecipato l'utente e la posizione in classifica ottenuta.

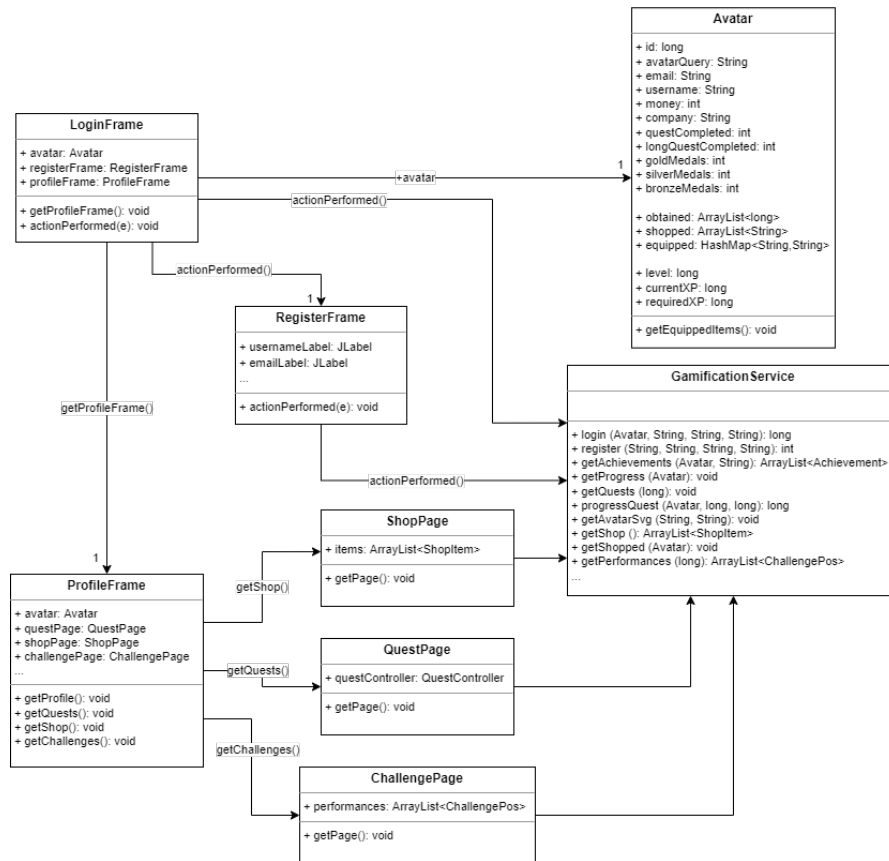


Figura 2.5: Classi principali del plugin di Gamification

2.3 Elementi di Gamification precedenti

2.3.1 Barra di progresso

Visibile sul bordo superiore della finestra, questa barra rende visibile la *coverage* della pagina corrente, ovvero il rapporto tra il numero di widget analizzati ed il totale dei widget analizzabili su quella determinata pagina: nello specifico, viene mostrata in rosso la barra complessiva mentre una barra verde ricoprirà gradualmente quest'ultima procedendo con il lavoro di analisi; vi è inoltre presente una barra blu che mostra la massima coverage raggiunta dagli altri utenti su quella stessa pagina. Così facendo, l'utente ha una preziosa informazione qualitativa sul proprio progresso, oltre che un senso di soddisfazione all'idea di riempire completamente la barra verde o il senso di sfida di superare la barra blu degli altri utenti, in accordo con i principi di *Social Influence* e *Development* di Octalysis.

Essendo un espediente piuttosto comune ed efficace gli esempi di implementazione

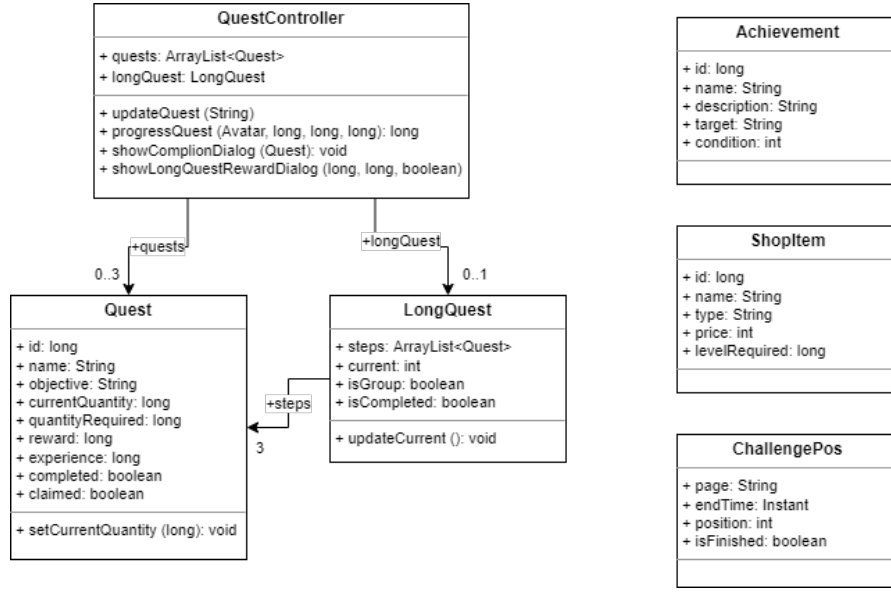


Figura 2.6: Classi principali del plugin di Gamification - pt. 2

di questo elemento in altre applicazioni in commercio sono molteplici: per porre un paio di esempi, il noto social dedicato ai contatti professionali *LinkedIn* fa uso di una barra di progressione per incentivare la creazione di un profilo il più completo possibile, mentre l'agenzia di viaggi online *Booking* sfrutta questa feature per concretizzare il progresso dei propri utenti nel suo programma fedeltà *Genius*.

2.3.2 Punteggio

Il punteggio è una misura numerica che ha lo scopo di creare una metrica oggettiva che valuti la qualità di una prestazione; in questo caso, viene mostrato al termine della sessione di Testing in una finestra di riepilogo, dove vengono anche mostrate tutte le metriche monitorate da Scout, le quali vengono utilizzate nel calcolo del punteggio nel seguente modo:

$$P = a \cdot \mathbf{C} + b \cdot \mathbf{EX} + c \cdot \mathbf{EF} + [d \cdot \mathbf{T}] + [e \cdot \mathbf{PR}] \quad (2.1)$$

I primi tre componenti determinano il punteggio *base* all'interno di questa somma pesata, ciascuno moltiplicato per il proprio coefficiente. In ordine:

- **C**: rappresenta la già precedentemente citata *coverage*, calcolata come media delle singole coverage raggiunte su tutte le pagine visitate durante la sessione

$$C = \frac{\sum_{i \in P} pc_i}{|P|} \quad (2.2)$$

con $|P|$ cardinalità dell'insieme delle pagine e pc_i coverage della singola pagina i , calcolata come

$$pc_i = \frac{\omega_{hl,i}}{\omega_{tot,i}} \quad (2.3)$$

con $\omega_{hl,i}$ numero di widget analizzati nella pagina i e $\omega_{tot,i}$ numero totale di widget della stessa pagina.

- **EX**: rappresenta la componente *esplorativa* della sessione, basandosi sul numero di pagine *scoperte*, cioè visitate dall'utente per primo rispetto agli altri. La componente prende inoltre in considerazione i widget *scoperti*, ed è calcolata come

$$EX = \frac{k}{b} \cdot \frac{p_{new}}{p_{tot}} + \frac{h}{b} \cdot \frac{h\omega_{new}}{h\omega_{tot}}, \quad k + h = b \quad (2.4)$$

dove il primo componente è il rapporto tra pagine scoperte e il totale delle pagine visitate, mentre il secondo è il rapporto tra i widget scoperti ed il totale dei widget analizzati.

- **EF**: rappresenta la componente *efficienza* e serve ad evitare che un tester cerchi di aumentare il proprio punteggio cliccando ripetutamente su widget già analizzati, ed è calcolata come

$$EF = \frac{\omega_{hl}}{\omega_{int}} \quad (2.5)$$

dove ω_{hl} è il numero totale di widget analizzati della sessione, mentre ω_{int} è il numero di interazioni effettuate sui widget (che comprende ogni click effettuato su widget, compresi quelli già evidenziati). Così facendo, se l'utente effettua la sessione in modo coscienzioso, questo valore sarà prossimo ad 1.

La somma pesata di questi componenti permette quindi di stabilire un punteggio da 0 a 100, a cui viene associato anche un *voto*, che parte da D (che rappresenta la sufficienza, partendo da 49 punti) fino ad S (100).

Le ultime due componenti invece, T e PR , rappresentano rispettivamente la componente *temporale* e la componente legata alle *problematiche* rilevate dall'utente, e la loro somma pesata compone il *punteggio bonus*, che può ammontare fino al 50% di quello base.

I coefficienti a , b , c , d ed e , che determinano il peso con cui vengono considerati i vari componenti, possono essere liberamente calibrati modificando un opportuno file di configurazione, tuttavia sono anche presenti dei valori di default, scelti in caso quest'ultimo file non rispetti alcuni vincoli.

In questa iterazione del progetto il calcolo del punteggio è rimasto invariato, tuttavia viene utilizzato anche per assegnare *monete* e *punti esperienza* all'utente, la cui utilità verrà spiegata nelle sezioni successive.

2.3.3 Classifica

Utilizzando il punteggio appena descritto è possibile ottenere una classifica dei migliori tester per prestazione ottenuta, comodamente raggiungibile dalla schermata di riepilogo del punteggio tramite un apposito tasto: questo elemento è molto importante in quanto introduce una componente competitiva, stimolando potenzialmente gli utenti ad ottenere risultati e performance sempre migliori per poter primeggiare sugli altri.

Già dalla precedente iterazione del progetto vengono inoltre mostrati gli *avatar* dei vari classificati, ma da quest'ultima è stato aggiunto anche il *livello* raggiunto, aggiungendo ulteriore componente sociale a questa feature. Un'idea per ulteriori miglioramenti futuri potrebbe essere quella di permettere di visitare il *profilo* degli altri utenti, in una versione parziale non interattiva, sempre dall'elenco della classifica.

2.3.4 Easter Egg

Gli easter egg sono una componente inserita per simulare la condizione, che può capitare frequentemente, di collegamenti ipertestuali che puntino a risorse non più disponibili o corrotti: ogni volta che l'utente entra in una pagina, viene estratto casualmente dal plugin un elemento cliccabile tra quelli presenti al suo interno, che una volta cliccato dall'utente genererà una figura ovoidale dorata nella nuova pagina raggiunta (non interattiva ma comunque di dimensioni non eccessivamente invasive). Questo elemento dovrebbe stimolare il tester a svolgere delle sessioni con coverage media superiore, proprio perché la ricerca degli easter egg richiede di interagire il più possibile su più widget diversi.

2.3.5 Stella delle pagine scoperte

Ulteriore elemento visivo, è stato introdotto per concretizzare l'abilità del tester all'interno della sessione: qualora quest'ultimo entri in una pagina che non era ancora stata visualizzata da nessun altro, comparirà infatti un simbolo a forma di stella nell'angolo superiore sinistro della finestra. Concetto piuttosto semplice, risulta essere una via di mezzo tra un premio ed un semplice feedback visivo, in quanto dovrebbe generare sì soddisfazione nell'utente ma rimane visibile solamente nel contesto della sessione in corso.

2.3.6 Login e Registrazione

Introdotta nella precedente iterazione, un sistema di accesso era necessario in quanto il punto cardine di quella iterazione è stato il sistema di *profilo* utente, il quale ha permesso lo spostamento di gran parte della logica di conservazione

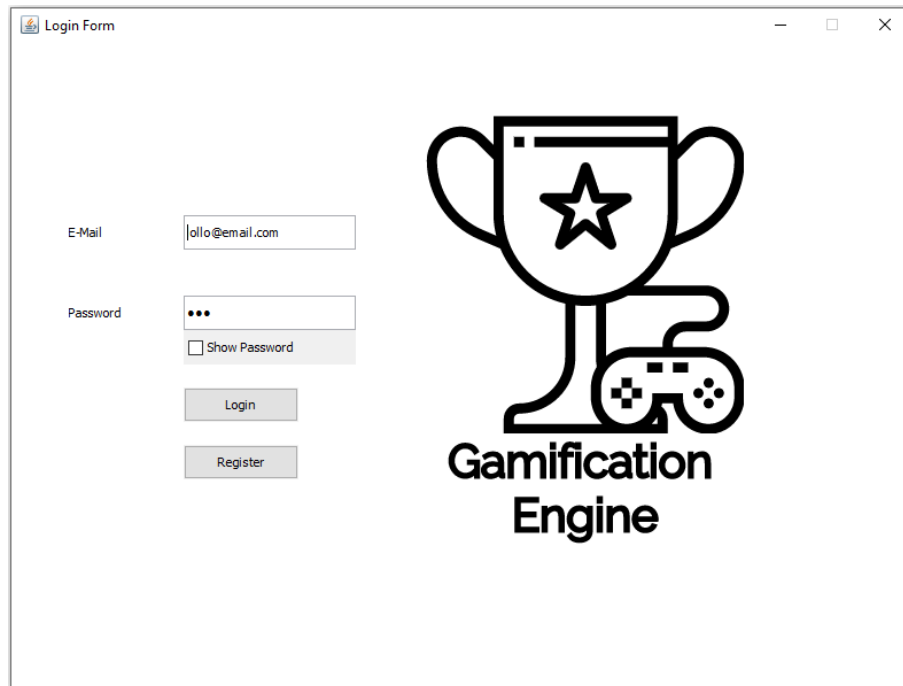


Figura 2.7: Schermata di login

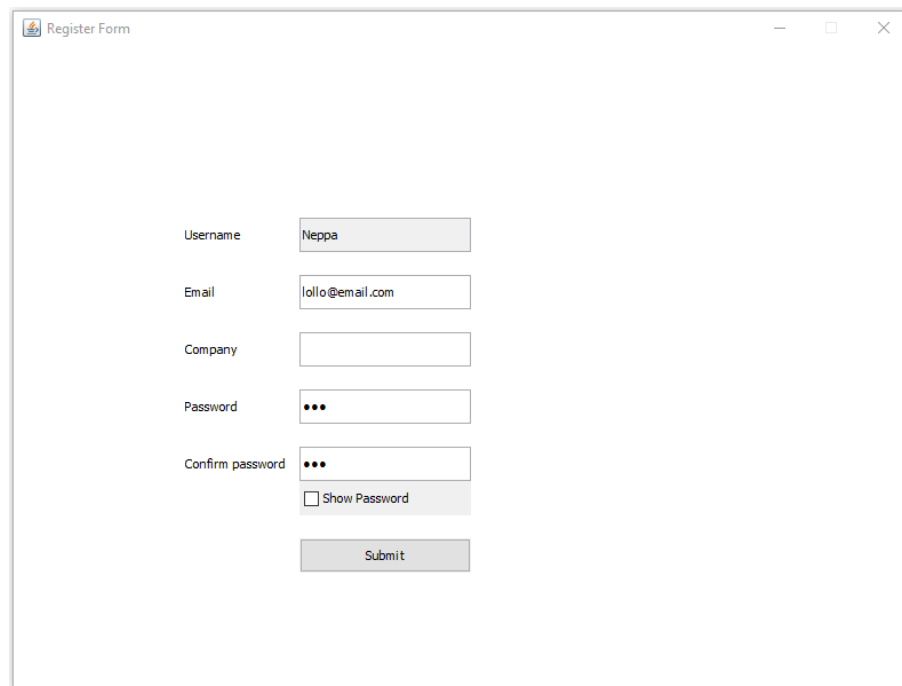
dei dati statistici raccolti nelle sessioni da un'implementazione base di database testuale ad un vero e proprio database interfacciato con un server back-end, scelta sicuramente necessaria per aumentare la scalabilità dell'applicazione.

Già dall'implementazione iniziale, il sistema di login permette un semplice modo di autenticarsi per l'utente, il quale come effetto collaterale potrà provare un senso di appartenenza e possesso, in accordo con i principi di *Ownership* e *Social Influence* di Octalysis.

A partire da quest'ultima iterazione è stato introdotto anche un sistema di *registrazione* per i nuovi utenti (Figura 2.8), ed è inoltre stato necessario vincolare il nome utente scelto all'interno di Scout (il campo *TesterName*) con le credenziali del sistema di Gamification, sempre per il sopracitato spostamento delle statistiche raccolte da Scout; quest'ultimo non è stato un problema in quanto di fatto il server back-end è stato rifatto da zero.

2.3.7 Profilo

Il profilo utente è senza dubbio uno degli elementi precedentemente introdotti che ha subito più modifiche in quest'iterazione e rappresenta il nucleo del sistema di gamification in quanto è il centro presso cui l'utente può accedere al resto delle



The screenshot shows a web browser window titled "Register Form". Inside the window, there is a registration form with the following fields and elements:

- Username:** A text input field containing the text "Neppa".
- Email:** A text input field containing the text "lollo@email.com".
- Company:** An empty text input field.
- Password:** A text input field with three dots (•••) indicating a password.
- Confirm password:** A text input field with three dots (•••) indicating a password.
- Show Password:** A checkbox with the label "Show Password".
- Submit:** A button labeled "Submit".

Figura 2.8: Schermata di registrazione

proprie informazioni personali o ad altri servizi offerti dal plugin. Le informazioni in esso contenuto, come si può vedere in Figura 2.9, sono:

1. e-mail
2. azienda di appartenenza
3. moneta interna all'app, spendibile presso il *negozio*
4. *medaglie* guadagnate partecipando alle *sfide*
5. *obiettivi* (achievements)
6. *livello* e indicatore di progresso dei *punti esperienza*
7. sommario di alcune statistiche relative alle precedenti sessioni, come numero di widget analizzati o percentuale media di coverage raggiunta

Sono inoltre presenti dei collegamenti ad altre sezioni del profilo, quali *Shop*, *Quests* e *Challenges*, il cui contenuto verrà spiegato nelle prossime sezioni.



Figura 2.9: Profilo utente

2.3.8 Avatar e negozio

L'avatar è la rappresentazione visiva dell'utente all'interno del contesto virtuale del programma, con il ruolo di coinvolgere quest'ultimo nell'ambiente di Testing: se l'utente si sente soddisfatto del proprio avatar sarà più motivato nelle proprie attività e, dato che è visibile anche dagli altri utenti all'interno della classifica, può sentirsi socialmente coinvolto.

Nell'iterazione precedente del progetto era prevista un'implementazione base del sistema di avatar, che consisteva in un set di sei avatar fissi, ottenuti da un progetto GitHub open source chiamato *Avataaars Generator* dell'utente *fangpenlin*, lasciando un'implementazione più completa alle future iterazioni.

Con la versione attuale del progetto, è stato implementato un microservizio back-end con una release di *Avataaars* dell'utente *gkoberger*, un progetto basato su *Express.js*, un micro-framework usato per sviluppare delle applicazioni web *Node.js* in modo veloce e pratico; è stata scelta questa particolare versione perché, nonostante fosse ideata e sviluppata principalmente per generare gli elementi di *Avataars* e restituirli in un formato HTML atto ad essere direttamente inseribili nel possibile front-end di una applicazione web, erano disponibili delle funzioni per ottenere direttamente il testo delle corrispondenti immagini vettoriali SVG con la giusta query.

Sono sorte inizialmente delle difficoltà con questo approccio: mentre ottenere direttamente delle immagini in un formato più classico e compatibile con le librerie grafiche di Java, su cui si basano le interfacce grafiche dei plugin di Gamification, sarebbe certamente risultato più pratico, l'autore stesso del progetto ha sottolineato più volte nel codice il fatto che convertire un documento vettoriale nel formato PNG fosse potenzialmente pericoloso sia per le prestazioni che per la memoria, soprattutto per quanto riguarda richieste di immagini di grandi dimensioni. Risulta quindi un'idea migliore mantenere il formato testuale, molto più leggero sia per il back-end che per il front-end, con l'unico contro di richiedere l'aggiunta di librerie esterne per il rendering sull'applicazione.

Detto questo, il negozio è passato dall'offrire i sei avatar di default ad essere diviso in diverse sezioni, ciascuna delle quali offre diversi tipi di accessori applicabili al proprio avatar base a proprio piacimento, permettendo un maggiore grado di personalizzazione. Questa soluzione è ancora limitata, ma sicuramente più scalabile e migliorabile ulteriormente in futuro.

2.3.9 Obiettivi

Gli *obiettivi*, chiamati *achievement* nell'applicazione, sono un espediente molto efficace e diffuso che attinge pienamente al principio di *Empowerment*, dando una linea guida concreta all'utente che mira alla soddisfazione del loro ottenimento.

Essendo ben visibili nel proprio profilo (Figura 2.9), hanno la duplice funzione di ricordare all'utente sia la gratificazione provata nell'ottenerli, sia le proprie mancanze durante le sessioni (in caso di quelli non ottenuti), fungendo quindi da stimolo ad ottenere migliori performance o fare più attenzione a determinati aspetti più trascurati.

Allo stato attuale gli achievement implementati sono dieci, scelti in modo da coprire tematicamente un po' tutte le metriche raccolte da Scout e gli altri elementi di Gamification disponibili all'utente:

- *Prime scoperte*: si sblocca raggiungendo una *coverage media* del 20%.
- *Esploratore esperto*: si sblocca raggiungendo una *coverage media* del 90%. Questo obiettivo ed il precedente hanno lo scopo di guidare gradualmente l'utente ad ottenere un grado di coverage sempre più alto, siccome di fatto è una metrica molto importante dato che porta ad analizzare una pagina più interamente, fornendo maggiori possibilità di trovare e segnalare eventuali malfunzionamenti.
- *Nessun segreto*: si sblocca trovando tutti gli *easter egg* in una sessione. Anche questo obiettivo è di fatto legato alla coverage, in quanto sprona ad analizzare ogni pagina visitata nel dettaglio.

- *Giusto in tempo*: si sblocca svolgendo una sessione di Testing di almeno 30 minuti. Questo obiettivo mira ad educare il tester ad un'analisi svolta rispettando i propri tempi, scoraggiando un lavoro fatto di fretta che potrebbe eventualmente tralasciare qualche elemento o comunque essere qualitativamente inferiore.
- *Repellente*: si sblocca trovando 10 o più *issue* in un'unica sessione. Questo obiettivo serve a quantificare e premiare quello che è l'obiettivo principale di ogni sessione di Testing, ovvero quello di localizzare i problemi.
- *Zio Paperone*: si sblocca ottenendo 500000 *monete*. Le monete sono una valuta interna all'applicazione che viene maturata svolgendo varie attività, ed è quindi giusto incentivarne l'acquisizione, tuttavia questo obiettivo è volutamente esagerato per rappresentare l'archetipo dell'achievement a lungo termine, difficile da ottenere e per questo molto desiderato.
- *All'ultimo grido*: si sblocca comprando 10 oggetti al *negozio*. Questo obiettivo è stato inserito per sfruttare la rinnovata disponibilità di oggetti acquistabili nello shop, oltre che trasversalmente ad incentivare il guadagno di monete così come l'obiettivo precedente.
- *Avventuriero provetto*: si sblocca completando 10 *quest*. Anche questo obiettivo è stato inserito per incentivare l'utilizzo della nuova feature delle missioni, le quali permettono all'utente il guadagno di monete e progressi per il proprio livello.
- *Campione*: si sblocca guadagnando 10 *medaglie*. Così come il precedente, anche questo obiettivo è stato inserito principalmente per invogliare gli utenti alla partecipazione della nuova feature delle *Sfide*, la quale permette di coinvolgere l'utenza in modo più competitivo.
- *Tester di Platino*: si sblocca ottenendo tutti gli altri achievement. Questo obiettivo è una diretta citazione al sistema di trofei adottato da *PlayStation*, per cui ad ogni gioco vengono assegnati una serie di trofei ed infine l'ambito trofeo di platino, ottenibile solo dopo la fatica ed il tempo necessari all'ottenimento di tutti gli altri, generando un'alta soddisfazione nell'utente.

2.4 Nuovi elementi di Gamification

2.4.1 Punti esperienza e livello

I *punti esperienza* sono un espediente molto comune nei GDR (giochi di ruolo) con cui gli sviluppatori guidano la progressione del giocatore: al completamento di ogni tipo di task questo otterrà dei punti esperienza, di quantità proporzionale alla difficoltà nello svolgere quel compito o alla sua complessità; più interagisce con il sistema, più ottiene esperienza.

Lo scopo dei punti esperienza non è quello di essere spesi come una possibile valuta interna, bensì vengono accumulati dal giocatore, facendogli guadagnare un grado superiore dopo il raggiungimento di una determinata soglia. Questo grado può essere rappresentato in vari modi, ad esempio con un indicatore numerico, altre volte tramite un sistema basato sulla rarità, per esempio con un rango che parte da bronzo fino ad arrivare ad oro. Un altro vantaggio principale di questo sistema è che permette agli sviluppatori di meglio bilanciare il gioco, tarando la difficoltà e le ricompense in relazione al livello del giocatore, bloccando per esempio determinate sfide troppo difficili dietro ad una barriera superabile solo acquisendo un livello sufficientemente alto.

Un altro aspetto importante di questo grado, riferito in questo contesto come livello, è quello sociale: di fatto, essendo esso spesso visibile agli altri utenti, può diventare motivo di pregio in caso di raggiungimento di un livello alto o, a parti invertite, motivo di invidia (positiva) che possa generare motivazione per il raggiungimento dello stesso traguardo, dello stesso apparente prestigio, esattamente come gli obiettivi o la classifica.

Esempi di questo elemento di Gamification in altre applicazioni sono innumerevoli, dato che è un concetto molto semplice ed efficace; un esempio tra tanti è *Todoist*, una delle app di task-management più utilizzate, che implementa il cosiddetto *Karma*-system: ogni utente può impostare i propri task giornalieri o settimanali e se riesce nel loro completamento ottiene dei punti Karma, perdendoli invece in caso fallisse nelle proprie scadenze; al momento della registrazione viene assegnato il livello *Beginner*, ed è possibile scalare i ranghi progressivamente attraverso otto livelli Karma, fino ad *Illuminated*.

In questo progetto è quindi stato integrato un sistema di punti esperienza affiancato da un sistema di livelli numerico, partendo dal livello 1 fino al livello 10. Il livello dell'utente è visibile nella sezione principale del proprio profilo, come si può vedere in figura 2.9, ma anche nella classifica, di fianco all'avatar. Essendo visibile da tutti gli altri utenti, il livello va ad aggiungersi agli altri fattori sociali implementati, rappresentando tutti gli sforzi ed il tempo passati con l'applicazione, andando in accordo coi principi di *Accomplishment*, *Social Influence* e *Empowerment*

di Octalysis.

Le soglie necessarie per salire di livello sono calcolate in modo che sia relativamente semplice per i primi livelli ma che diventi gradualmente più difficile; il quantitativo effettivo di esperienza è proporzionale al quantitativo dato dalle varie attività, come completare una sessione di Testing oppure altri elementi introdotti in questa iterazione, le *missioni*.

Oltre al fattore sociale, il livello è utilizzato anche per bloccare certi accessori del negozio, che ora risultano disponibili solo agli utenti che riescono a raggiungere un certo livello: questa decisione dovrebbe incentivare gli utenti ad interagire con il sistema nel modo più completo possibile, ottenendo un punteggio alto e completando tutte le quest disponibili. Allo stesso tempo, l'utente potrebbe probabilmente provare frustrazione non essendo in grado di ottenere un certo oggetto al negozio, o sentire l'urgenza di completare sessioni o quest solo al fine di ottenere esperienza, andando a perdere di senso l'applicazione stessa, in accordo con i principi *Black Hat* di Octalysis come la *Scarcity*; questo aspetto dovrebbe essere attenuato dal fatto che gli oggetti bloccati non sono limitati nel tempo, quindi prima o poi tutti gli utenti possono essere in grado di ottenerli.

2.4.2 Missioni

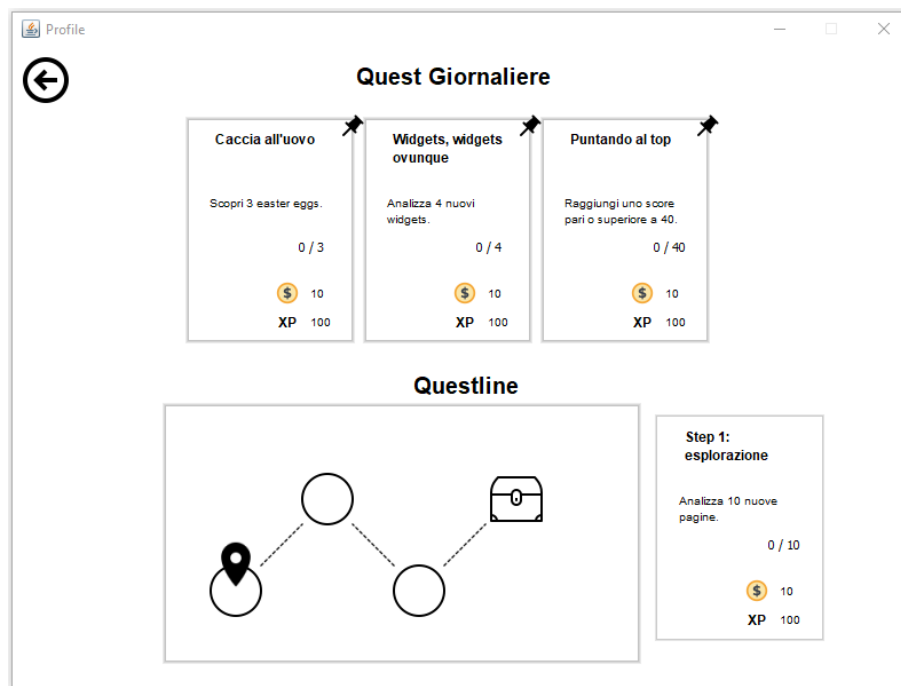


Figura 2.10: Schermata delle quest

Le *missioni*, o come sono chiamate nell'applicazione *quest*, sono un elemento tipico dell'immaginario fantasy, dove vengono usate per indicare un'avventura che l'eroe deve intraprendere alla ricerca o per l'ottenimento di qualcosa, che sia trovare un tesoro o salvare il mondo dal male; in altre parole, escludendo la parte di fantasia, consistono in una o più attività da svolgere, alla conclusione delle quali spetta una ricompensa. Come può un concetto del genere essere applicato al Software Testing?

Il concetto di attività è di fatto molto semplice da applicare a questo contesto, dove il processo generale può essere diviso in attività man mano più basilari, in pieno stile *divide et impera*, ma la mia idea iniziale si è ispirata ad un'altra applicazione di task management, *Habitica*, la quale porta il concetto di "fantasy" su un livello superiore, basando di fatto l'intero design dell'applicazione su di esso: non appena completa la registrazione, l'utente viene introdotto ad un menu di personalizzazione del proprio avatar che risulterà familiare a chi è fruitore dei giochi di ruolo, presentando elementi come barra della vita, esperienza e classe; sempre da quel menu, l'utente può imporsi dei task relativi alla propria vita quotidiana, con una scadenza a propria scelta, e al completamento dei quali otterrà sia una valuta interna all'applicazione che punti esperienza, con i quali potrà migliorare il proprio avatar per poter svolgere varie attività con gli altri utenti, come sconfiggere mostri o partecipare ad avventure.

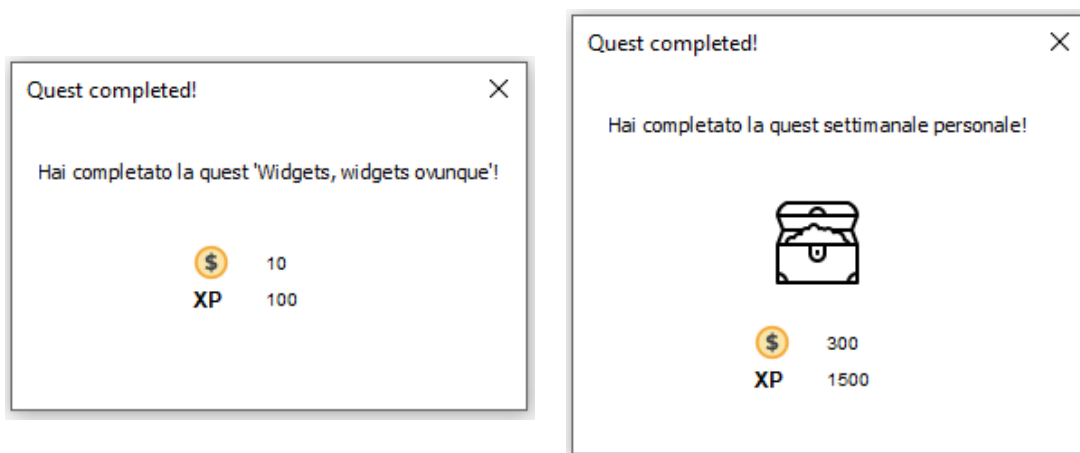


Figura 2.11: Feedback grafici per il completamento delle quest, rispettivamente di una quest singola e di una questline

Nel caso di Scout, l'intenzione non è stata quella di sconvolgere completamente l'intera applicazione, ma comunque di introdurre qualcosa che potesse incoraggiare l'utente ad operare in modo migliore, facendolo concentrare anche su aspetti più ignorati ma al contempo senza chiedergli di svolgere azioni troppo complicate o differenti da quelle necessarie per una normale sessione di Testing, rendendo

quindi le sessioni più premianti ma non più complesse. In particolare, le azioni richieste spaziano dal dover analizzare un certo quantitativo di widget o pagine al dover raggiungere un determinato punteggio a fine sessione, comprendendo quindi metriche che vengono comunque rilevate dal tool e quindi già integrate. Il quantitativo richiesto e le ricompense partono da una base decisa a priori e vengono poi bilanciati in base al livello del tester, cercando quindi di proporre missioni più impegnative e premianti a chi ha più esperienza con il tool.

All'interno del proprio profilo è stata quindi introdotta una sezione apposita, *Quest*, dove l'utente può controllare le proprie missioni ed i propri progressi: queste sono illustrate come se fossero delle taglie affisse ad una bacheca (Figura 2.10), sempre per rimanere in tema giochi di ruolo, su di cui è possibile vedere sia il proprio progresso che le ricompense guadagnabili a fine sessione in caso vengano completate. Ogni utente può completare fino a tre *missioni semplici* al giorno e due *missioni lunghe*, denominata in app *questline*, con scadenza settimanale; quest'ultima consiste in una serie di quest il cui completamento fa proseguire l'utente su di una sorta di *mappa del tesoro*, permettendo di aprirlo alla fine di questa serie per una ricompensa ben superiore alle normali missioni. Va fatto notare inoltre che per questioni di sincronizzazione ed ottimizzazione le ricompense vengono assegnate solo a fine sessione, tramite feedback grafici come in Figura 2.11.

Essendo un ulteriore metodo per guadagnare risorse utili a personalizzare il proprio avatar, l'introduzione delle missioni risulta in linea con i principi di *Accomplishment*, *Social Influence* e *Ownership*; allo stesso tempo però sono stati introdotti dei limiti temporali per impedire che questo metodo potesse in qualche modo sbilanciare il sistema di monete e di progresso, facendo guadagnare entrambe le risorse in quantità eccessiva, andando in parte in linea con i principi negativi di *Scarcity* ed *Avoidance*: per rimediare a quest'ultima considerazione, è probabilmente sufficiente bilanciare la difficoltà di completamento e/o le ricompense.

Infine, dopo aver completato la prima questline settimanale, l'utente riceverà una questline *collettiva*, in cui contribuirà ai progressi di ciascuna missione intermedia con tutti gli altri utenti della propria azienda: in questo modo, l'utente può sentirsi coinvolto in qualcosa di più grande di sé e risultare quindi ancora più motivato nell'ottenere performance migliori, andando in accordo con i principi di *Social Influence* ed *Epic Meaning*.

2.4.3 Sfide

Per quanto riguarda gli elementi di Gamification spiegati finora vengono di fatto coperti gran parte dei principi positivi di Octalysis e solo parzialmente alcuni di quelli negativi: questa scelta è considerata generalmente valida in quanto i principi negativi fanno leva su aspetti come l'ansia di perdere qualcosa o limitazioni temporali che possono impedire l'ottenimento di un qualche oggetto o la partecipazione

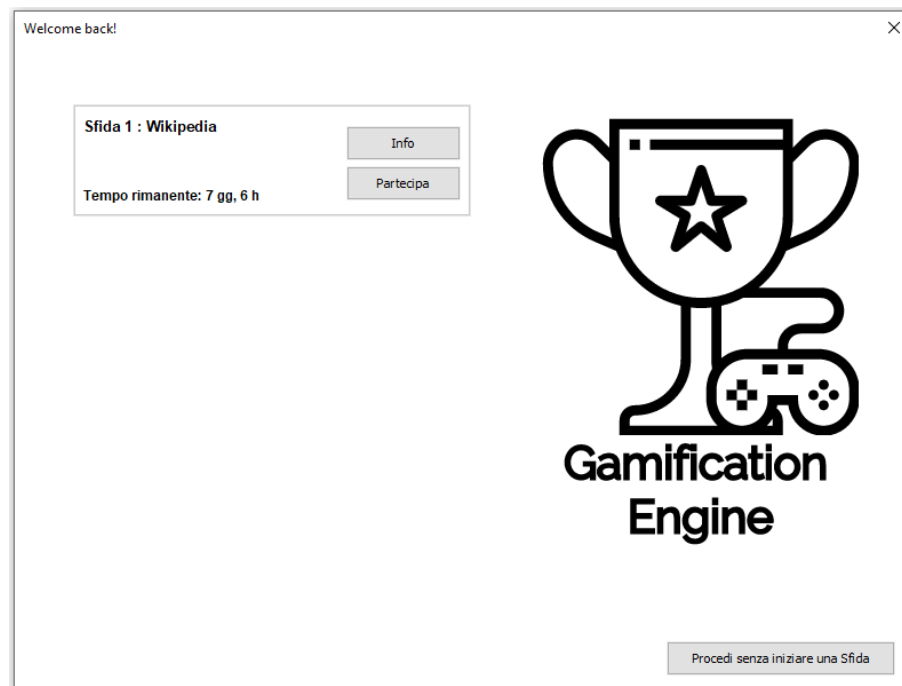


Figura 2.12: Schermata iniziale del plugin di Gamification

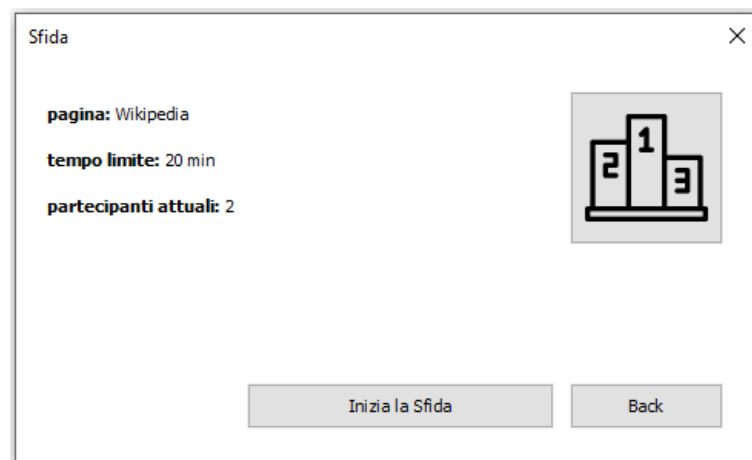


Figura 2.13: Schermata della sfida selezionata

ad un'attività. In realtà, se sfruttati per accrescere la produttività dell'utente e non con scopi maliziosi e manipolativi anche questi principi possono essere utilizzati per ottenere una buona esperienza nella propria applicazione.

Alla luce di questa considerazione, l'intenzione è stata quella di provare a sfruttare principi come *Avoidance* e *Scarcity*, combinati ad una classica componente

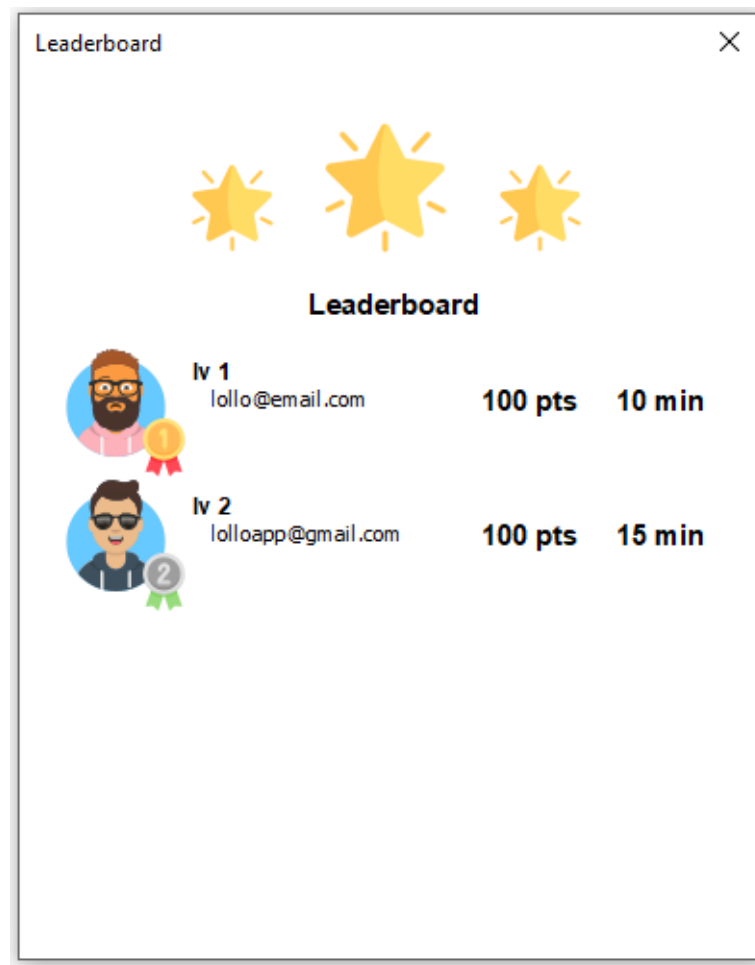


Figura 2.14: Classifica della sfida

competitiva che risulta essere stimolante per l'utente medio: sono state quindi implementate le *sfide*, ovvero delle competizioni proposte solo per un certo periodo di tempo (visibile in Figura 2.12) e caratterizzate dal trattare una certa pagina target e da un certo tempo limite per la sessione (Figura 2.13). Una volta aderito ad una sfida, l'utente si ritrova con un'interfaccia visivamente analoga ad una normale sessione, eccetto per il timer, che anziché fungere da contatore crescente ha la funzione di conto alla rovescia, indicando il tempo rimasto all'utente: l'obiettivo della sfida è quindi quello di ottenere il *punteggio* migliore possibile nel minor *tempo*; in situazioni di parità di punteggio viene quindi favorito l'utente la cui sessione è durata di meno, come si può vedere nella classifica di Figura 2.14.

Un ulteriore fattore che dovrebbe incentivare la partecipazione a questa feature è la possibilità di ottenere delle *medaglie* per i primi tre classificati, rispettivamente

d'oro, argento e bronzo, la cui quantità guadagnata è ben visibile nel proprio profilo, svolgendo un ruolo simile a quello degli *obiettivi* andando in accordo con i principi di *Accomplishment* ed *Ownership*.

Questo elemento è forse quello che risulta meno sinergico con tutti gli altri, introducendo il concetto di *evento limitato* e di *prova a tempo*, che non è detto che si sposino bene con l'ambito, andando contro a concetti precedentemente presi in considerazione come quello di favorire sessioni lunghe e proficue; rimane tuttavia interessante testarne l'effettivo impatto sull'esperienza d'uso del plugin, al fine di comprendere se sia possibile in futuro migliorare l'idea o scartarla.

2.5 Octalysis del Gamification Engine

Avendo utilizzato il framework Octalysis per il design di buona parte delle funzionalità introdotte in questa e nella precedente iterazione del progetto risulta interessante eseguire un'analisi ed un confronto di come il plugin di Gamification sia cambiato: mentre nella Tabella 2.1 sono visualizzabili gli elementi, sia già implementati precedentemente che quelli appena introdotti, con le relative dimensioni di Octalysis, in Figura 2.15 è possibile visualizzare un diretto confronto tra le due situazioni.

Si può notare come i nuovi elementi introdotti vadano principalmente a rafforzare core in cui il tool era già sviluppato, andandoli ad accentuare ulteriormente, in particolare per quanto riguarda *Empowerment*, *Social Influence* ed *Ownership*. Al contempo però sono stati introdotti, anche se in modo lieve, alcuni elementi con aspetti appartenenti ai core di *Black Hat Gamification*, soprattutto *Scarcity* e *Avoidance*; questa scelta è probabilmente positiva in quanto spesso i sistemi con elementi principalmente di *White Hat Gamification* finiscono per suscitare esperienze sì piacevoli ma senza imporre alcuna urgenza nello svolgere i propri compiti. Risulta quindi adeguato cercare di trovare un buon equilibrio tra le due scuole di pensiero, senza trascurare a prescindere nessun core, e allo stesso tempo senza inserirne in modo superfluo.

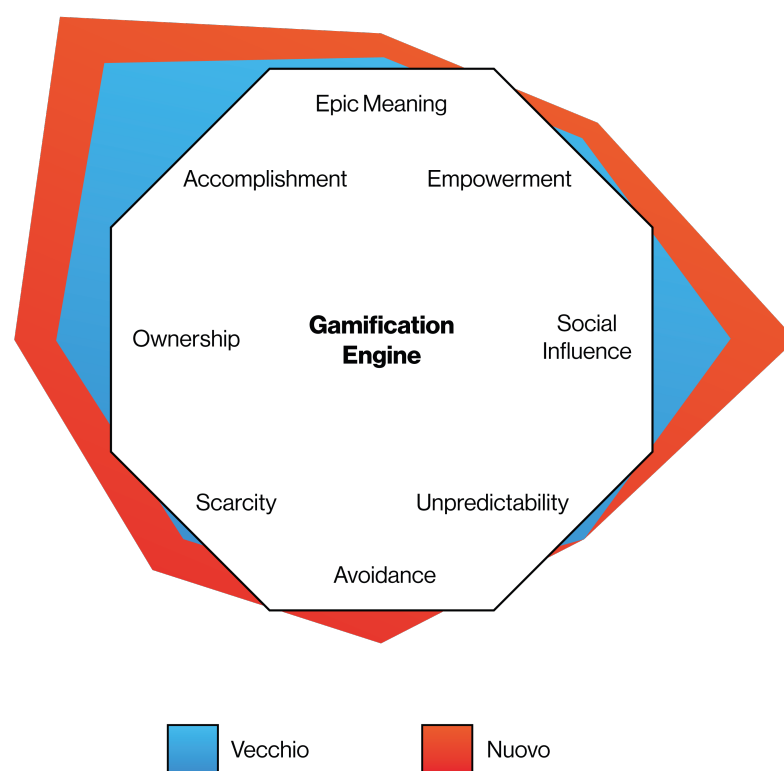


Figura 2.15: Confronto tra il modello di Octalysis del plugin allo stato precedente (vecchio) e quello allo stato attuale (nuovo)

Elemento	Presente/Nuovo	Octalysys core
Barra di progresso	presente	Accomplishment, Social Influence
Punteggio	presente	Accomplishment
Classifica	presente	Accomplishment, Social Influence
Easter Egg	presente	Unpredictability
Stella	presente	Accomplishment
Profilo utente	presente	Accomplishment, Social Influence, Ownership, Epic Meaning
Avatar e Negozio	presente	Social Influence, Ownership, Empowerment, Scarcity
Obiettivi	presente	Accomplishment, Ownership
Punti esperienza e Livello	nuovo	Accomplishment, Social Influence, Empowerment
Missioni	nuovo	Accomplishment, Social Influence, Ownership, Epic Meaning, Avoidance, Scarcity
Sfide	nuovo	Accomplishment, Social Influence, Ownership, Avoidance, Scarcity

Tabella 2.1: Elementi di Gamification e rispettivi core di Octalysis

Capitolo 3

Implementazione del Back End

Il sistema di Gamification finora descritto è composto principalmente da due parti, come si può vedere in Figura 3.1:

- la parte di *Scout*, che è la vera e propria applicazione *front-end* utilizzata dall'utente. Questa consiste a sua volta di quattro parti, tra cui la *GUI*, l'interfaccia grafica con cui l'utente interagisce, e *Selenium*, il plugin principale che permette il Testing e l'interazione con l'applicazione web. La componente che comunica con il *back-end* per ricevere tutti i dati utili alla sessione è la classe *GamificationService*, mentre *GamificationUtils* consiste in una serie di funzioni atte a rielaborare questi dati ricevuti.
- la parte di *Docker*, che consiste in un'applicazione *multi-container* contenente il server *back-end* (Spring Boot Application), il micro-servizio di *Avataars* per la creazione degli avatar utente ed infine il *database*. Questo ecosistema di micro-servizi non è completamente esposto, bensì soltanto il server Spring risulta raggiungibile tramite un'interfaccia *API RESTful*, attingendo esso stesso dagli altri due componenti.

Mentre della parte di Scout sono già state discusse le funzionalità e le feature di Gamification introdotte, di seguito viene analizzata tutta la parte di *back-end*, la quale implementazione ha richiesto tempo e risorse non inferiori alla precedente.

3.1 Spring

Framework open-source per lo sviluppo di applicazioni su piattaforma Java, è stato scelto *Spring* in quanto valida alternativa per lo sviluppo di applicazioni

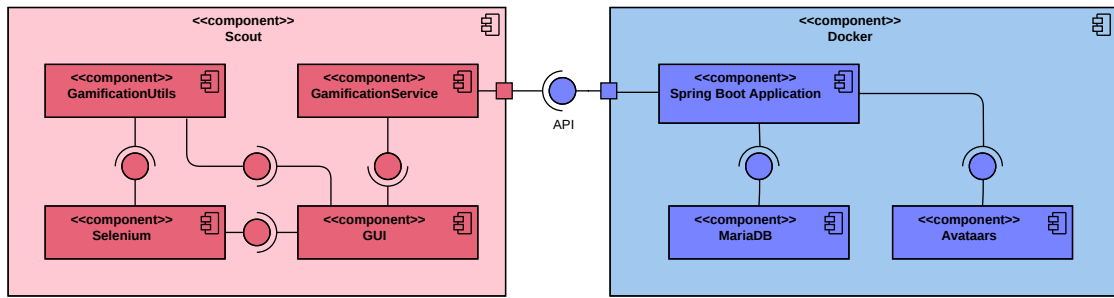


Figura 3.1: UML component diagram

web, capace di adattarsi ad ogni propria esigenza grazie alle sue funzionalità e molteplici estensioni. Nato nel 2002 per mano di Rod Johnson, fu tra i primi framework noti per sfruttare i concetti di *Inversion of Control* (IoC) e *Aspect Oriented Programming* (AOP).

- Il primo è sostanzialmente il core del framework, e consiste in un *IoC container* che si occupa di gestire il ciclo di vita di ogni oggetto all'interno del contesto applicativo, della configurazione e di reperimento delle dipendenze e creazione delle singole istanze, tramite il concetto di *Dependency Injection* (DI); in questo modo, lo sviluppatore ha molti meno vincoli e si trova ad utilizzare un ambiente altamente *flessibile e modulare*.
- Il secondo è una tecnica di programmazione che permette di scrivere della logica una volta sola applicandola però in modo *orizzontale* all'intera applicazione, andandola ad applicare ovunque sia richiesto in *load time*, modificandone il *byte code* aggiungendo delle istruzioni oppure estendendole dinamicamente in modo da sovrascriverne alcune. Anche questo concetto aggiunge molte possibilità allo sviluppatore, il quale ha un potente strumento aggiuntivo per sviluppare la logica della propria applicazione.

Oltre a questi due importanti aspetti, va considerato che nel corso degli anni sono stati sviluppati un gran numero di progetti satellite a Spring, che vanno ad estenderne le funzionalità ed i campi d'uso, come *Spring Security* che va a gestire le problematiche di sicurezza oppure *Spring Data* che si propone invece come interfaccia per gestione ed accesso di database relazionali e non. Ultimo e non meno importante aspetto da considerare è il fatto che la community che segue il progetto sia relativamente numerosa, rendendone quindi frequente l'aggiunta di funzionalità ed il *bug-fixing*. Di contro, risulta inizialmente complicata la configurazione di un progetto Spring in quanto è necessaria la sinergia ed il setup di diversi moduli anche per l'inizializzazione di un progetto semplice.

3.1.1 Spring Boot

Dopo aver esposto i pro e contro dell'utilizzo del framework viene spontaneo pensare che fosse necessaria una semplificazione per renderlo più accessibile: nel 2013 avviene quindi la prima release del progetto *Spring Boot*, il quale non è un nuovo framework che prende le basi da Spring, bensì una "sovrastuttura" che permette di sfruttare al meglio gli strumenti forniti da quest'ultimo tramite una soluzione *convention over configuration*, cioè fornendo una *configurazione di default* che include tutto ciò che viene utilizzato tipicamente in un'applicazione Spring. In particolare, un'applicazione Spring Boot consiste in una vera e propria applicazione *standalone*, eseguibile tramite il classico metodo *main*, contenente una versione *embedded* del *web container* (tipicamente *Tomcat* o *Jetty*), rimuovendo la necessità di dover installare e configurare un web server da sé.

L'architettura di un'applicazione Spring Boot consiste di quattro livelli logici (Figura 3.2), ciascuno dei quali interagisce con i suoi adiacenti scambiandosi determinate classi di informazioni.

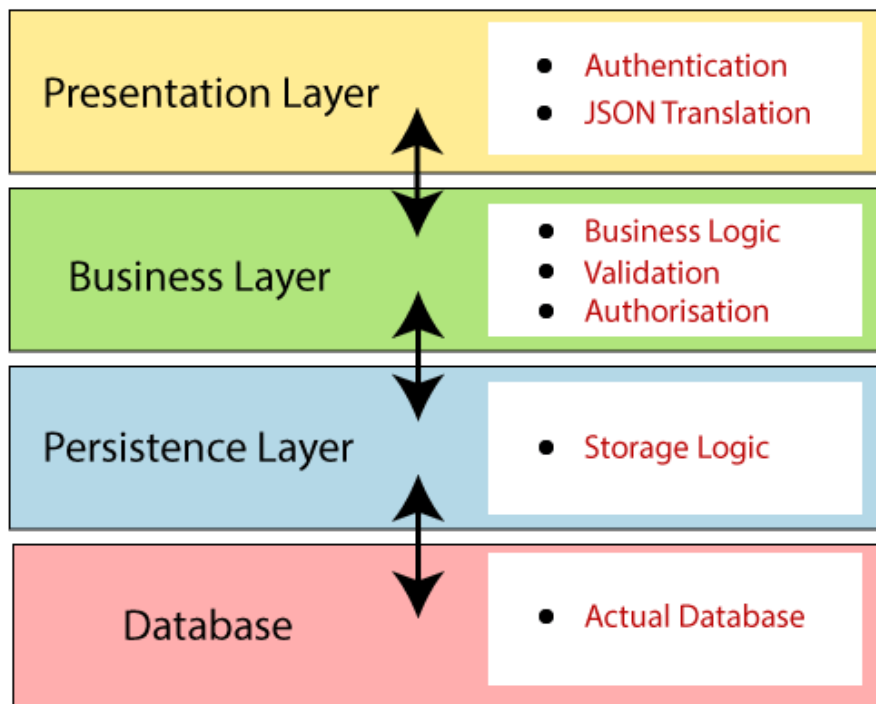


Figura 3.2: Architettura di Spring Boot

- *Presentation Layer*: strato dell'applicazione che gestisce le richieste HTML, converte i parametri JSON in oggetti ed autentica le richieste in modo da propagarle al livello sottostante. E' inoltre possibile implementare un *front-end* su questo livello restituendo delle *View*, cioè delle pagine statiche HTML popolate dagli oggetti restituiti dai risultati delle richieste. Nel caso di questo progetto le richieste vengono mandate dalla classe *GamificationService.java* del plugin di Gamification e quindi è sufficiente restituire i dati richiesti.
- *Business Layer*: strato in cui risiede tutta la *logica* dell'applicazione, cioè tutte le classi e funzioni che definiscono come vengono rielaborati e restituiti i dati ricevuti dal livello sottostante; è inclusa anche la logica di *autorizzazione* e *validazione*, tuttavia al momento non è stata implementata.
- *Persistence Layer*: strato in cui risiede la logica di *storage*, dove vengono quindi definiti gli oggetti trattati dall'applicazione e come vengono convertiti nella tabelle del database.
- *Database*: strato dove avvengono le operazioni *CRUD* (Create, Retrieve, Update, Delete) con il database vero e proprio.

Dal punto di vista pratico però il flusso di informazioni che avviene dopo una qualsiasi richiesta HTTP è il seguente (Figura 3.3):

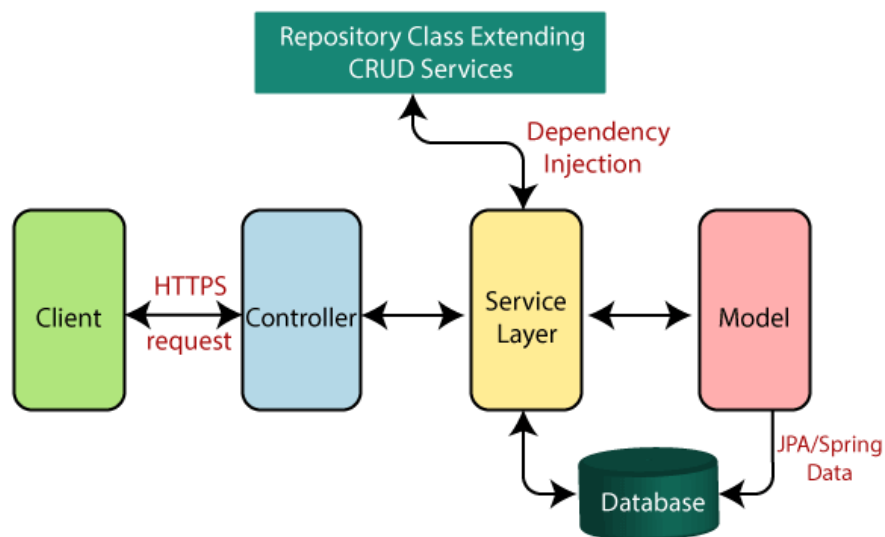


Figura 3.3: Flusso dei dati in Spring Boot

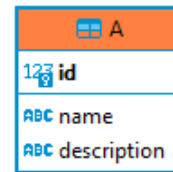
1. il *client* (nel nostro caso il plugin di Gamification) effettua una *richiesta HTTPS*.
2. questa richiesta viene presa in considerazione da un *Controller* ed in base all'*url* viene identificato il corretto *API Endpoint*.
3. quest'ultimo chiama una funzione offerta da un *Servizio*, il quale si occupa di interrogare il database attraverso un'interfaccia detta *Repository*, ed eventualmente rielaborare il dato secondo la logica imposta.

Per definire questi moduli ed elementi costituenti l'applicazione viene fatto uso di una delle feature caratteristiche di Spring, ovvero l'*Aspect Oriented Programming*, attraverso l'utilizzo delle *annotazioni*. Le annotazioni principali utilizzate sono le seguenti:

- *@Component*: le classi annotate con questa annotazione generica vengono prese in considerazione dall'IOC container (*ApplicationContext*) e vengono quindi istanziate, in stile *singleton*, e collegate automaticamente agli altri componenti che ne esprimono una dipendenza. Per quanto fondamentale, non viene spesso utilizzata così com'è ma ben più spesso ne vengono utilizzate delle annotazioni derivate, come quelle seguenti.
- *@Controller*: annotazione che designa una classe come *controller*, al suo interno vengono definite le funzioni che vanno a gestire i vari *API Endpoint* che verranno decorate a seconda del tipo di richiesta HTTP che devono gestire con annotazioni come *@GetMapping(url)*, *@PostMapping(url)* o *@PutMapping(url)*.
- *@Service*: annotazione che va a caratterizzare i *servizi*, ovvero le classi che si occupano di recuperare i dati dal *Persistence Layer*, rielaborarli secondo le logiche opportune e consegnarli al *Presentation Layer*. E' spesso accompagnata dall'annotazione *@Transactional* in modo da gestire le operazioni *atomicamente*, ovvero gestendo correttamente i casi di operazioni concorrenti con il database e la recovery in caso di errori sulla singola operazione.
- *@Repository*: annotazione che va a decorare le interfacce che si occupano di effettuare le operazioni *CRUD* su una determinata tabella del database.
- *@Entity*: grazie a questa annotazione non è necessario creare le tabelle del database ad hoc in quanto il modulo *Spring Data* (nel caso del progetto *Spring Data JPA*) si occupa di inizializzarle a partire dalle classi così annotate, come in Figura 3.4, all'avvio dell'applicazione.

- *@Autowired*: questa annotazione è forse la più importante essendo quella che permette effettivamente la *Dependency Injection* in sinergia con le annotazioni derivate da *@Component*, permettendo di esprimere le dipendenze tra moduli come mostrato in Figura 3.5.

```
1 @Entity
2 class A(
3   @Id
4   @GeneratedValue
5   var id: Long? = null,
6   @NotNull
7   var name: String,
8   @NotNull
9   var description: String
10 )
11
```



A	
123	id
ABC	name
ABC	description

Figura 3.4: Esempio di entity JPA e relativa tabella nel database

```
1 @RestController
2 @RequestMapping(value = ["/scout"], produces = [MediaType.
3   APPLICATION_JSON_VALUE])
4 class MainController {
5   @Autowired private lateinit var achievementService:
6     AchievementService
7   ...
8 }
```

Figura 3.5: Esempio di dipendenza del MainController espressa nei confronti della classe AchievementService

3.2 Progetto

3.2.1 Componenti

Dopo aver svolto un'analisi generale della struttura di un'applicazione Spring Boot, seguono quelle che sono le componenti effettivamente implementate nel progetto, divise per livello logico.

Per quanto riguarda il livello di *Presentazione*, le classi implementate sono

- *AuthController*: modulo che contiene gli endpoint di autenticazione, che al momento sono ridotti semplicemente alla restituzione dei dati utente e non contengono ancora una vera e propria implementazione della sicurezza.
- *MainController*: modulo che contiene tutti gli altri endpoint di gestione/modifica dati.

Passando invece al livello di *Business*, la classi implementate sono

- *TesterService*: modulo core dell'applicazione, si occupa di fornire tutte le informazioni dell'*utente* e delle classi ad esso direttamente connesse, comprendendo quindi gran parte della logica implementata.
- *AchievementService*: modulo che permette l'aggiunta di nuovi *achievement* o di restituirne l'elenco.
- *CompanyService*: modulo che si occupa della gestione delle *company* (aziende a cui l'utente viene associato in fase di registrazione), permettendone anche l'aggiunta e l'assegnazione di *quest settimanali* (come descritto nel paragrafo di Gamification).
- *LevelService*: modulo che contiene le metriche con cui viene calcolato il progresso di *livello* dell'utente in risposta all'ottenimento di esperienza con il completamento della sessione o di una *quest*.
- *PageService*: modulo che si occupa di gestire tutto ciò che concerne le *pagine* (intese come le applicazioni web analizzate con Scout), gestendo anche tutto ciò ad esse collegate, come le *sfide* o la scoperta di *sottopagine*.
- *QuestService*: modulo che si occupa della generazione ed assegnazione delle *quest* all'utente in fase di login.
- *ShopItemService*: modulo che permette l'aggiunta di nuovi *item*, che saranno poi acquistabili dall'utente nel *negozio*, o di restituirne l'elenco.

Per quanto riguarda infine il livello di *Persistence*, per ogni classe *@Entity* implementata è stata inserita una relativa interfaccia *@Repository*: per esempio, per l'entità *Tester* è stata implementata un'interfaccia *TesterRepo*, e così via per ogni altra entità da gestire, come elencato nelle prossime sezioni.

3.2.2 Documentazione API

Partendo dall'*AuthController*, esso consiste nei seguenti endpoint:

- **LOGIN : POST**
Endpoint: /auth/sign-in
Descrizione: verifica che le credenziali siano corrette ed in caso affermativo ritorna le informazioni necessarie a popolare il profilo utente.
Input: credenziali d'accesso (nome utente, password, email)
Output: dati utente (id, email, monete, medaglie, etc.)
- **REGISTER : POST**
Endpoint: /auth/sign-up
Descrizione: permette all'utente di registrarsi alla piattaforma.
Input: credenziali di registrazione (nome utente, password, email, azienda)
Output: messaggio di fallimento/successo

Per quanto riguarda invece il *MainController*, questo consiste in numerosi endpoint che possono essere utilizzati sia dal plugin di Gamification che dagli amministratori del back-end come interfaccia al database; quelli elencati di seguito sono gli endpoint che vengono utilizzati dal "front-end".

- **STATS : GET**
Endpoint: /scout/users/stats
Descrizione: restituisce le statistiche di sessione dell'utente dato il suo username
Input: username (scelto in fase di registrazione)
Output: statistiche aggiornate all'ultima sessione (coverage media, widget analizzati, etc.)
- **STATS : PUT**
Endpoint: /scout/users/stats
Descrizione: salva le statistiche di sessione dell'utente e restituisce il progresso dell'utente.
Input: statistiche di fine sessione appena computate da Scout
Output: livello, esperienza e monete dell'utente aggiornati
- **ACHIEVEMENTS : GET**
Endpoint: /scout/users/achievements?email=email

Descrizione: restituisce l'elenco degli achievement, sia quelli ottenuti dall'utente che non, a partire dalla sua email.

Parametri: email (scelta in fase di registrazione)

Output: lista di achievement (id, nome, condizione di ottenimento, etc.), ottenuti dall'utente o meno

- **ACHIEVEMENTS : POST**

Endpoint: /scout/users/**userId**/achievements

Descrizione: permette l'ottenimento di un achievement per l'utente selezionato.

Parametri: id utente

Input: id achievement

Output: achievement appena sbloccato

- **PROGRESS : GET**

Endpoint: /scout/users/**userId**/level

Descrizione: restituisce il progresso dell'utente.

Parametri: id utente

Output: livello, esperienza e monete attuali dell'utente

- **AVATAR : GET**

Endpoint: /scout/avatar/**query**

Descrizione: restituisce la stringa SVG corrispondente alla query inviata.

Parametri: query corrispondente all'avatar richiesto

Output: stringa SVG restituita dal microservizio Avataars.js

- **AVATAR : PUT**

Endpoint: /scout/users/**userId**/avatar

Descrizione: permette di modificare la query associata all'avatar dell'utente selezionato e restituisce la nuova query.

Parametri: id utente

Input: lista delle parti di query che si vogliono modificare e relativi valori desiderati

Output: query aggiornata

- **SHOP : GET**

Endpoint: /scout/items

Descrizione: restituisce l'elenco degli oggetti in vendita al negozio.

Output: lista di oggetti (id, nome, prezzo, etc.)

- **SHOPPED : GET**

Endpoint: /scout/users/**userId**/items

Descrizione: restituisce l'elenco degli oggetti acquistati dall'utente.

Parametri: id utente

Output: lista di oggetti acquistati

- **SHOPPED : POST**

Endpoint: /scout/users/**userId**/items

Descrizione: permette l'acquisto di un oggetto per l'utente selezionato.

Parametri: id utente

Input: id dell'oggetto da acquistare

Output: oggetto appena acquistato

- **QUESTS : GET**

Endpoint: /scout/users/**userId**/quests

Descrizione: restituisce l'elenco delle quest assegnate all'utente selezionato.

Parametri: id utente

Output: lista delle quest assegnate

- **QUESTS : PUT**

Endpoint: /scout/users/**userId**/quests/**questId**?newQty=newQty

Descrizione: permette di aggiornare il progresso di una quest per l'utente selezionato.

Parametri: id utente, id quest da aggiornare e relativo progresso

Output: progresso della quest aggiornato (utile in caso di quest collettive in cui l'aggiornamento può essere *asincrono* per progressi svolti da altri utenti)

- **REWARDS : GET**

Endpoint: /scout/users/**userId**/quests/**questId**/reward

Descrizione: permette di controllare se una quest sia conclusa per l'utente selezionato ed in caso affermativo restituisce il progresso utente aggiornato.

Parametri: id utente, id quest

Output: livello, esperienza e monete dell'utente aggiornati

- **REWARDS : GET**

Endpoint: /scout/users/**userId**/long-quest/reward

Descrizione: permette di controllare se la questline attualmente assegnata all'utente sia conclusa ed in caso affermativo restituisce il progresso utente aggiornato.

Parametri: id utente

Output: livello, esperienza e monete dell'utente aggiornati

- **RANK : GET**

Endpoint: /scout/users/rankings

Descrizione: restituisce la classifica utenti in base al punteggio di fine sessione.

Output: lista ordinata di utenti (email, avatar, livello) e relativi punteggi

- **COMPANY : GET**
Endpoint: /scout/companies/**name**
Descrizione: restituisce l'azienda corrispondente al nome selezionato.
Parametri: nome azienda
Output: azienda e relativo elenco dei suoi impiegati (utenti)
- **CHALLENGES : GET**
Endpoint: /scout/pages/challenges?**pages**=name&..
Descrizione: restituisce l'elenco delle sfide disponibili per le pagine selezionate.
Parametri: elenco delle pagine (siti web registrati e proposti nel tool)
Output: lista di sfide (pagina target, tempo, partecipanti, scadenza)
- **PERFORMANCES : GET**
Endpoint: /scout/users/**userId**/challenges/performances
Descrizione: restituisce l'elenco delle performance effettuate nelle sfide a cui l'utente selezionato ha partecipato.
Parametri: id utente
Output: lista di performance dell'utente (pagina target, posizione)
- **PERFORMANCES : PUT**
Endpoint: /scout/users/**userId**/challenges/**challengeId**/performance
Descrizione: aggiorna le performance dell'utente selezionato riguardanti la sfida selezionata, restituendone inoltre la classifica.
Parametri: id utente, id sfida
Input: punteggio e tempo impiegati
Output: lista ordinata delle performance attuali della sfida selezionata (email utente, avatar, livello, punteggio, tempo)
- **CHALLENGE RANK : GET**
Endpoint: /scout/challenges/**challengeId**/ranking
Descrizione: restituisce la classifica della sfida selezionata.
Parametri: id sfida
Output: lista ordinata delle performance attuali della sfida selezionata

3.2.3 Entità e tabelle

Di seguito sono mostrati i *modelli E-R* del database mappati dalle classi *@Entity* implementate, divise da Figura 3.6 a 3.8 in quanto il diagramma completo risulterebbe complicato ed ingombrante; dai diagrammi si possono quindi estrapolare le seguenti tabelle:

1. Contiene tutti i dati degli utenti rappresentati poi nel profilo.
Tester (id, email, username, lastLogin, password, questCompleted, long-questCompleted, money, goldMedals, silverMedals, bronzeMedals, avatarId, companyId, testerLevelId, rankingId, statsId)
2. Lista di tutti gli obiettivi sbloccabili.
Achievement (id, amount, description, name, objective)
3. Lista degli oggetti acquistabili.
ShopItem (id, name, price, type, levelRequired)
4. Lista delle quest proposte agli utenti.
Quest (id, name, objective, quantityRequired, experience, reward, isStep, longQuestId)
5. Lista delle questline.
LongQuest (id, experience, reward, creationDate, isGroup, isCompleted)
6. Lista delle aziende iscritte al servizio.
Company (id, name, longQuestId)
7. Lista delle pagine analizzate dal plugin.
Page (id, name, challengeId, subpages)
8. Lista delle sfide proposte agli utenti.
Challenge (id, duration, startTime, endTime, claimed)
9. Relazione tra utenti e corrispettivi obiettivi sbloccati.
TesterAchieved (testerId, achievementId)
10. Relazione tra utenti e relativi oggetti acquistati.
TesterShopped (testerId, shopItemId)
11. Lista dei progressi di ciascun utente.
TesterLevel (id, experience, level)
12. Relazione tra utenti e quest ad essi assegnate, contengono i progressi di ciascuno ed il fatto che le ricompense siano già state riscattate o meno.
TesterQuest (questId, testerId, progress, claimed)
13. Lista dei migliori punteggi associati a ciascun utente, utilizzata al fine della composizione delle classifica.
Ranking (id, score)
14. Lista degli avatar associati a ciascun utente, divise per query tematiche.
Avatar (id, accessoriesQuery, clotheQuery, eyebrowsQuery, eyesQuery, facialHairQuery, graphicsQuery, mouthQuery, skinQuery, styleQuery, topQuery)

15. Lista delle performance che ciascun utente ha svolto in una determinata sfida, contiene anche il fatto che le medaglie siano già state riscattate o meno.

Performance (testerId, challengeId, score, time, claimed)

16. Lista delle statistiche di sessione di ciascun utente.

Stats (id, issues, newPages, newWidgets, seconds, minutes, totHighlightedWidgets, eePercentages, coverages)

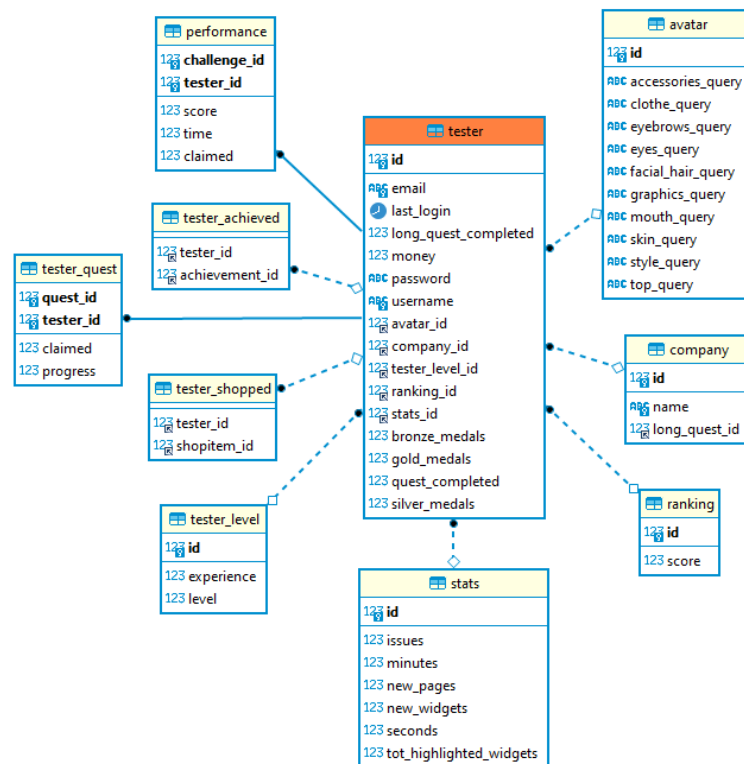


Figura 3.6: Modello E-R di Tester ed entità correlate

3.2.4 Interfaccia con il microservizio Avataars

Come detto precedentemente, l'applicazione Spring Boot fa da interfaccia agli altri due microservizi, occupandosi di fornire delle API frontali e nascondendo l'effettiva comunicazione con essi. Mentre la connessione con il *database* viene effettuata con le interfacce *repository*, la comunicazione con il microservizio di *generazione degli avatar* (il modulo Node.js del progetto Avataars descritto precedentemente) avviene invece tramite la classe *AvatarConnector*: questa semplice classe consiste

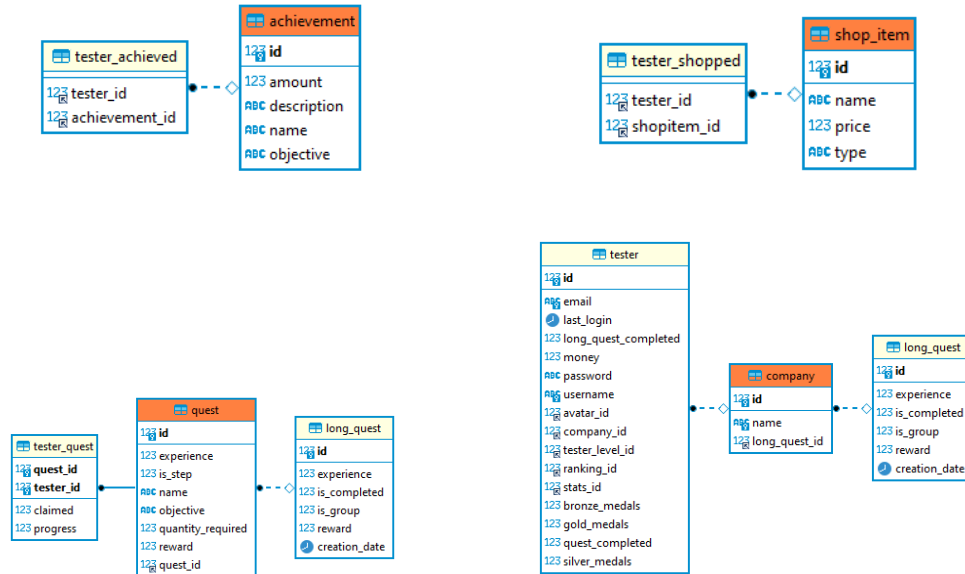


Figura 3.7: Modelli E-R delle altre entity, pt.1

della sola funzione *getSvgString()*, nella quale viene instaurata una connessione *Http* con il servizio e viene richiesta, tramite la query opportuna, la stringa *SVG* rappresentante l'avatar desiderato. L'url della richiesta che viene effettuata segue il seguente schema:

<http://avatar-generator:3000/type/?query>

dove *type* può essere utilizzato per richiedere un avatar nella sua interezza oppure un suo singolo componente (utilizzato in fase di composizione del negozio front-end). Si può notare come l'applicazione non conosca l'effettivo indirizzo del micro-servizio, ma grazie al servizio di *DNS discovery* nativo di Docker la parte evidenziata viene opportunamente sostituita nel momento in cui venga effettuata la comunicazione.

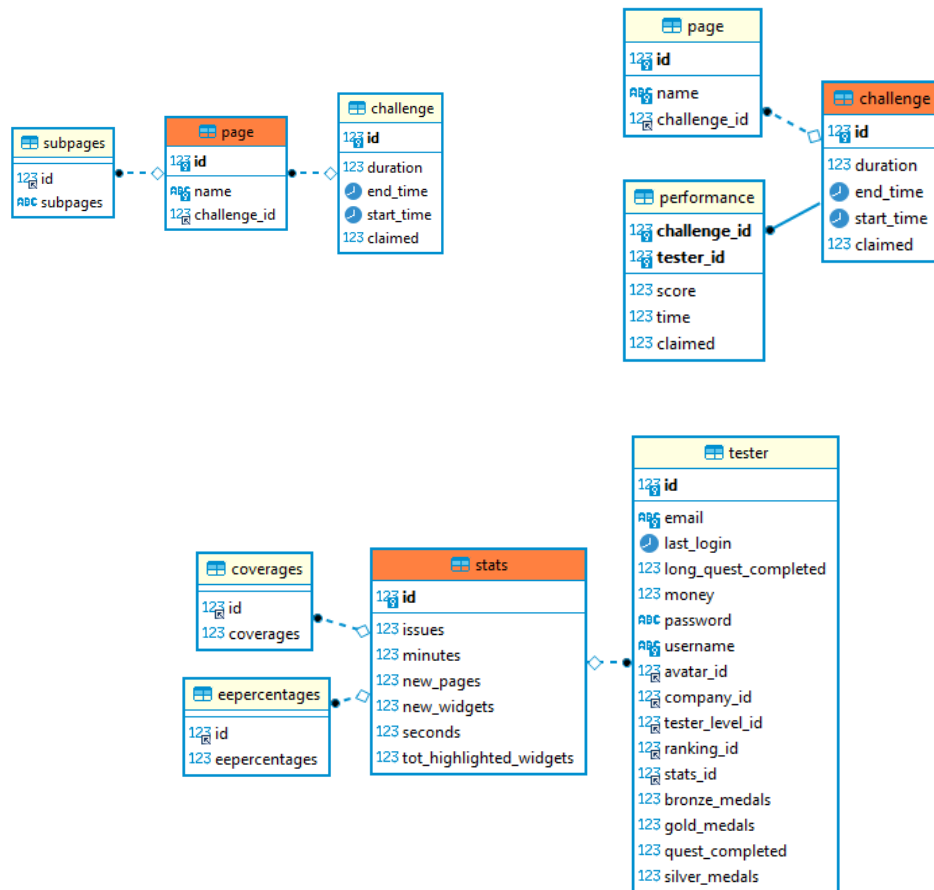


Figura 3.8: Modelli E-R delle altre entity, pt.2

Capitolo 4

Sperimentazione del Tool

Dopo aver concluso l'implementazione del plugin è iniziata quindi una fase sperimentale su un ristretto gruppo di persone con l'obiettivo di valutare l'impatto delle feature ludiche introdotte nel tool.

4.1 Preparazione

4.1.1 Partecipanti

I partecipanti dell'esperimento sono stati cercati con un minimo comune denominatore, ovvero un background nello sviluppo e nel testing di applicazioni web, che è il target a cui è rivolto il tool Scout; questi sono stati individuati perlopiù internamente al Politecnico di Torino come laureandi o laureati in Ingegneria Informatica ma anche come dottorandi o con ruoli di ricerca sempre nel medesimo settore, con infine una piccola parte che lavora nel mondo del testing.

Il gruppo di partecipanti effettivo è stato composto da otto persone tra i 23 e i 33 anni, il cui background, visibile nei grafici di Figura 4.1, 4.2 e 4.3, è piuttosto diverso nonostante la maggior parte avesse un'età a metà di questo intervallo, offrendo preziosi feedback da punti di vista differenti. Si può notare che la maggior parte dei partecipanti avesse almeno due anni di esperienza con la programmazione web (cinque persone), due persone da meno di un anno e solamente una persona tra uno e due anni. La maggior parte dei partecipanti utilizza abitualmente Java e Javascript, seguito da Python e infine seguito dal gruppo di linguaggi Typescript, Kotlin e PHP, con quest'ultimo utilizzato abitualmente solo da una persona. Si noti infine come un buon numero di partecipanti abbia dichiarato di avere un'ottima esperienza con il testing di applicazioni web (tre persone), due persone si sono collocate su una buona fascia di esperienza mentre solo tre persone si sono collocate nella fascia medio-bassa.

Quanti anni di esperienza hai con la programmazione Web?

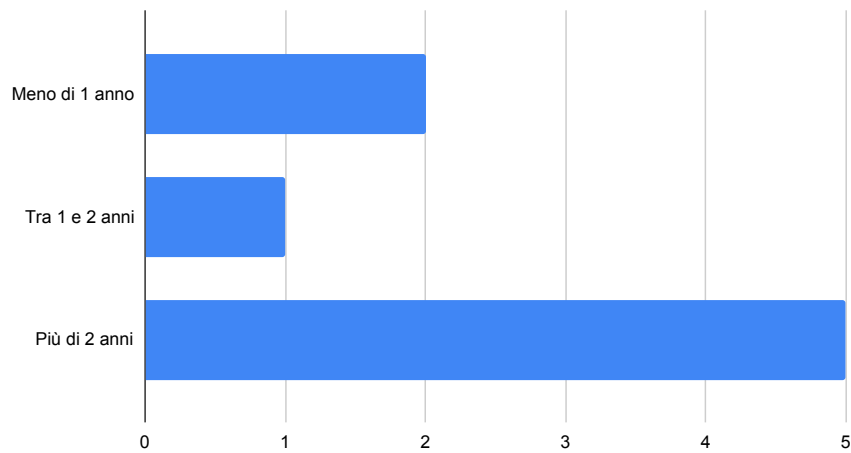


Figura 4.1: Anni di esperienza con la programmazione di applicazioni web

Quali linguaggi, tra i più conosciuti, hai mai utilizzato per la programmazione Web?

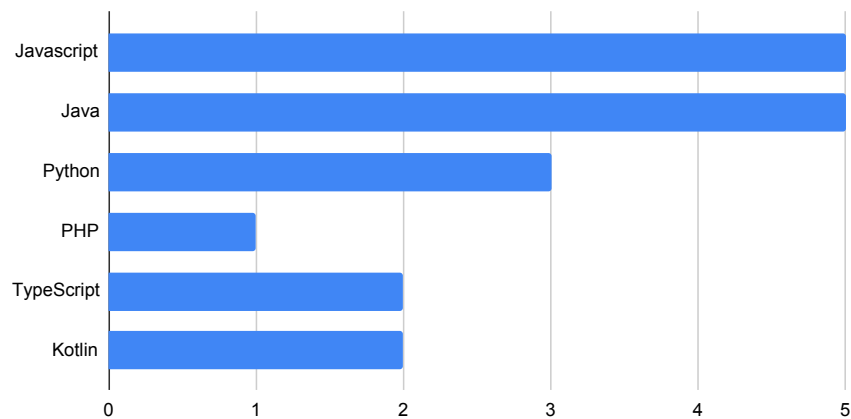


Figura 4.2: Linguaggi più utilizzati nello sviluppo di applicazioni web

4.1.2 Procedimento

L'esperimento ha seguito il seguente schema (Figura 4.4): ciascun utente ha avuto modo di provare il tool prima nella sua versione originale su un sito A, poi la sua versione ludicizzata su un sito B ed infine sempre la sua versione ludicizzata, ma provando la modalità *Sfida*, su un sito C.

Hai esperienza con il testing di applicazioni Web?

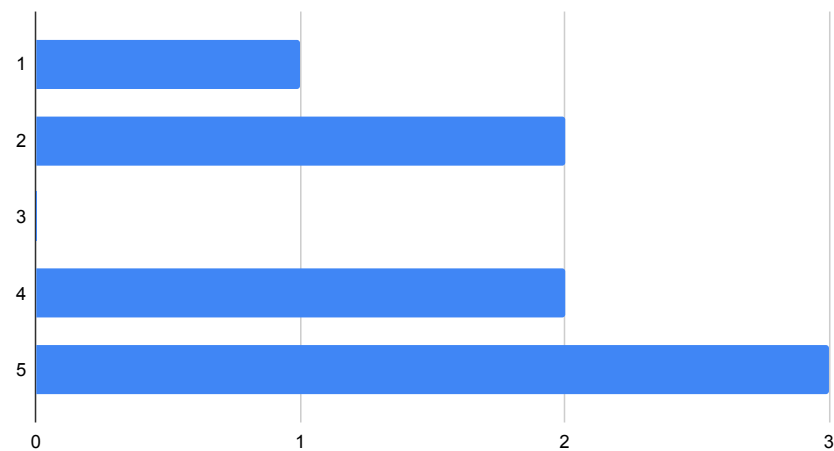


Figura 4.3: Esperienza con il testing di applicazioni web (1: scarsa, 5: ottima)

Prima di iniziare è stato spiegato brevemente il funzionamento di Scout ed i suoi principi caratterizzanti, in modo che ciascun utente avesse la consapevolezza di come il tool funzionasse e di come avrebbe dovuto operare a breve. Dopo di che è iniziata la prima sessione di testing sul sito A, della durata massima di 10 minuti.

Dopo aver concluso la sessione a ciascun partecipante sono stati spiegati tutti i componenti di Gamification introdotti ed il loro funzionamento. Dopo aver effettuato la registrazione al sistema di login è quindi iniziata la seconda sessione, questa volta ludicizzata, su un sito B differente dal precedente, sempre con 10 minuti circa a disposizione.

Questa seconda sessione è infine seguita da una terza ed ultima sessione, sempre caratterizzata dall'attivazione del plugin di Gamification ma scegliendo la modalità Sfida: questa modalità è infatti caratterizzata dal negare alcune feature in favore di una sessione più "competitiva", ed è quindi stato pensato di testarla in una sessione a sé stante, anch'essa di 10 minuti al massimo, su di un sito C.



Figura 4.4: Procedura dell'esperimento

Da ciascuna delle sessioni precedenti sono state raccolte una serie di metriche interessanti al fine di valutare le prestazioni dei vari partecipanti e trarne delle

conclusioni; queste sono state comunicate visivamente agli utenti solo nelle due sessioni ludicizzate grazie alla schermata di riepilogo, tuttavia sono state generate anche durante la sessione standard, permettendone quindi il confronto a posteriori. Le pagine web utilizzate durante l'esperimento sono, nell'ordine:

- **MagiGiovanni** (<https://magi-giovanni-funghi-e-tartufi.webnode.it/>)
- **PoliTo** (<https://www.polito.it/>)
- **Wikipedia** (<https://www.wikipedia.org/>)

La prima considerazione da fare è che sono pagine molto differenti tra loro, sia in tipologia che in scala: mentre il primo è un sito web relativamente ristretto e semplice, le altre due presentano delle pagine mediamente più dense e lunghe da testare, per cui alcuni risultati come la *coverage* o il *punteggio* non sono molto interessanti da confrontare in modo assoluto; ciò non significa che non si possano comunque fare una serie di considerazioni egualmente utili con la cognizione di queste differenze, come verrà spiegato nelle sezioni successive. Si fa inoltre notare come, non essendo applicazioni web fatte ad hoc per la fase sperimentale, non sia possibile assicurare l'eventualità di individuare degli *issue* (bug), per cui questa metrica non è stata analizzata.

4.1.3 Questionario

Per concludere l'esperienza, una volta concluse le tre sessioni di testing, ciascun partecipante ha dovuto compilare il seguente questionario (Tabella 4.1), al fine di analizzare i risultati ottenuti.

Tabella 4.1: Domande del Questionario

ID	Domanda (Tipo)
1.1	Età (Aperta)
1.2	Hai esperienza professionale con Java, come studente o lavoratore? (Likert)
1.3	Quanti anni di esperienza hai con la programmazione Web? (Scelta Multipla)
1.4	Quali linguaggi, tra i più conosciuti, hai mai utilizzato per la programmazione Web? (Aperta)
1.5	Hai esperienza con testing di applicazioni Web? (Likert)
1.6	Quali altri tool hai utilizzato per fare testing di applicazioni Web? (Aperta)
2.1	Trovi comprensibile il modo d'uso di Scout ed il suo contesto di utilizzo? (Likert)
2.2	Hai trovato utile la Barra di Progresso a mostrare i tuoi progressi nel testing di una pagina? (Likert)
2.3	Pensi che la Stella che contrassegna le pagine scoperte abbia influito sulla tua esplorazione dell'applicazione? (Likert)
2.4	Credi che la possibilità di trovare Easter Egg ti abbia condizionato ad interagire con un maggior numero di widget? (Likert)
2.5	Il fatto di ricevere un Punteggio ha posto maggiore attenzione sulle tue performance? (Likert)
2.6	Trovi stimolante la presenza della Classifica? (Likert)
2.7	Hai trovato comprensibile la Schermata di Riepilogo e le informazioni che fornisce? (Likert)
2.8	Trovi le informazioni presenti nel Profilo soddisfacenti? (Likert)
2.9	Come trovi il grado di personalizzazione dell'Avatar? (Likert)
2.10	Come pensi che gli Obiettivi abbiano influito sulla tua sessione di testing? (Likert)
2.11	Trovi che la disponibilità del Negozio ti abbia stimolato in qualche misura? (Likert)
2.12	Come pensi che abbia influito la possibilità di salire di Livello? (Likert)

Continua nella prossima pagina

Tabella 4.1 – Domande del Questionario

ID	Domanda (Tipo)
2.13	Come valuteresti le Missioni in termini di difficoltà di completamento e ricompense? (Likert)
2.14	Cosa pensi delle Sfide? (Likert)
2.15	Quali tra i seguenti elementi erano a te familiari prima dello svolgimento delle prove? (Checkbox)
2.16	Quali, tra gli elementi selezionati nella risposta precedente, sono stati utilizzati in modo a te familiare? (Checkbox)
2.17	Quali tra i seguenti elementi ritieni essenziali in una sessione di testing? (Checkbox)
2.18	Trovi che il tool sia facilmente integrabile in un ambiente di testing esistente (lavorativo o di studio)? (Likert)
2.19	Credi che la versione gamificata di Scout abbia influito positivamente sulle tue performance rispetto a quella standard? (Likert)
2.20	Nel complesso, trovi migliore la versione gamificata di Scout rispetto a quella standard? (Likert)
2.21	Hai riscontrato problemi durante le prove? Se sì, descrivi brevemente (Aperta)
2.22	Hai dei suggerimenti per migliorare il tool dal punto di vista delle feature offerte da Scout? (Aperta)
2.23	Hai dei suggerimenti per migliorare il tool dal punto di vista delle feature di gamification (che siano miglioramenti o idee inedite)? (Aperta)

4.1.4 Ambiente di Esecuzione

L'esecuzione della procedura appena descritta ha avuto luogo per tutti i partecipanti con il medesimo ambiente di esecuzione, evitando quindi che questo fosse un fattore discriminante tra le varie sessioni. Le caratteristiche rilevanti di quest'ultimo sono:

- Sistema Operativo Windows 10 Home
- CPU Intel Core i7-6500 2.50GHz
- 12GB DDR4 RAM
- Memoria SSD NAND 3D

E' importante tenere in considerazione queste informazioni in quanto, come verrà anche sottolineato nella sezione successiva dei risultati, un fattore piuttosto

importante è quello delle performance del tool. Queste ultime infatti non sono state ottimali, ma comunque migliori rispetto alla sperimentazione delle iterazioni precedenti del progetto, dove per vari motivi si è dovuto far uso di macchine virtuali, aggiungendo ulteriore input lag e riducendo le performance.

Nonostante l'uso dello stesso PC è stato scelto di proporre uno stato del tool identico ad ogni partecipante: ciascuno di essi ha infatti svolto le tre sessioni di testing come se fosse il primo utente a testare le rispettive pagine su quel dispositivo, trovando poi nella classifica solamente uno stesso altro partecipante (placeholder); sarebbe potuto essere interessante mantenere la classifica dopo ciascun gruppo di sessioni, ma così facendo probabilmente ci sarebbe stato sbilanciamento tra l'esperienza del primo partecipante, il quale non si sarebbe trovato in competizione con nessuno, e quella dell'ultimo, che avrebbe invece visto le prestazioni di tutti gli utenti precedenti.

4.2 Valutazione delle Performance

Come già accennato precedentemente, tutti i dati raccolti sono stati monitorati dal programma per le sessioni come se fossero tutte e tre ludicizzate; in questo modo, oltre alle metriche raccolte come numero di widget e pagine analizzate o la coverage media di sessione, il punteggio e il numero di easter egg trovati sono stati calcolati anche nella sessione standard, senza che l'utente non fosse consapevole. Così facendo risulta più facile fare un confronto quantitativo e soprattutto qualitativo tra i tre differenti approcci.

Detto questo, la grande differenza di scala delle pagine scelte nelle tre sessioni ha fatto sì che il dato delle coverage medie (Figura 4.5) sia poco rilevante ai fini di un confronto: al di là del fatto che nella maggior parte dei casi, seppur ci sia una differenza visibile tra le tre sessioni, si tratta comunque di valori molto bassi. In quasi tutti i casi la coverage più alta è stata registrata nella sessione in cui la pagina target era *MagiGiovanni*, il che ha favorito questo esito in quanto era un sito piuttosto piccolo con una quantità limitata di elementi da analizzare; al tempo stesso però si possono notare dei casi in cui la differenza è quasi esigua o addirittura il risultato ottenuto in una delle due sessioni ludicizzate è notevolmente superiore, e questo può essere considerato comunque un successo del plugin di Gamification, che ha in qualche modo aumentato la consapevolezza di questa metrica.

Per quanto riguarda invece il punteggio di fine sessione, visibile in Figura 4.6, si può notare come questo segua nella maggior parte dei casi le aspettative sperate: ad eccezione di un paio di casi, le due sessioni con plugin attivo hanno registrato un punteggio superiore, con una prevalenza tra le due della sessione ludicizzata di Sfida. Questa metrica si dimostra più significativa della sola coverage, in quanto comprensiva anche della componente esplorativa data dal numero di widget e pagine

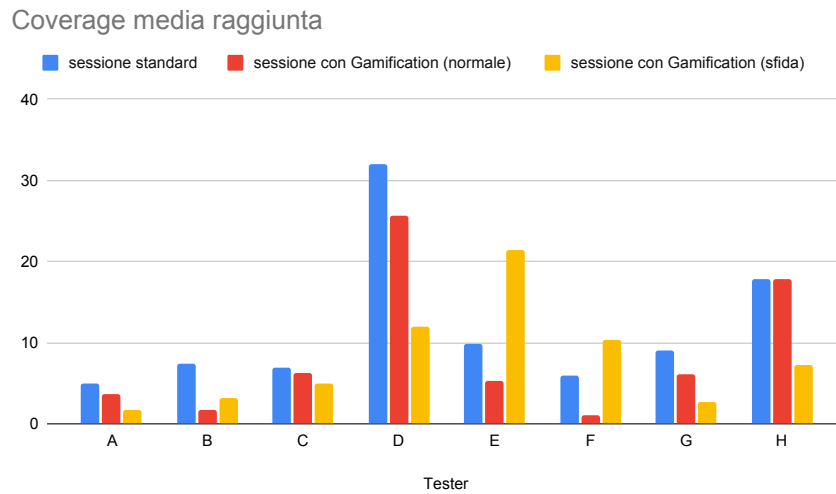


Figura 4.5: Coverage media raggiunta

scoperti e della componente di efficienza (in cui viene valutato il tempo di analisi dei widget).

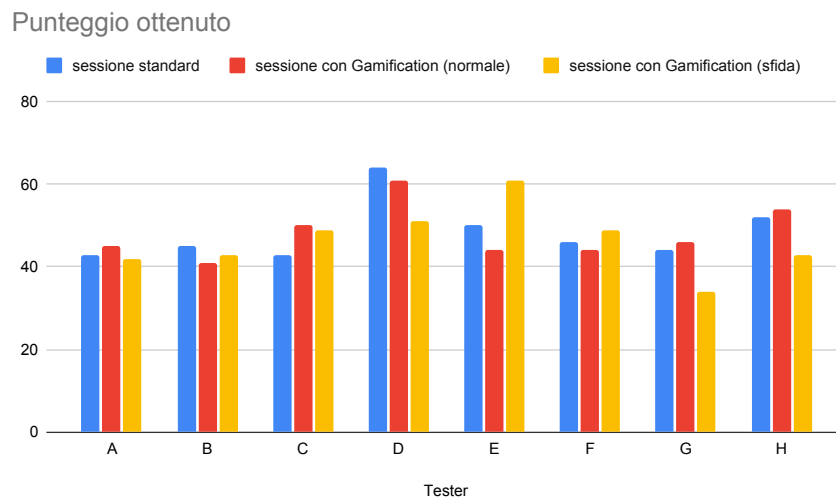


Figura 4.6: Punteggio ottenuto

Come si può notare dalla Figura 4.7 il numero di widget analizzati è generalmente aumentato, in alcuni casi addirittura più che raddoppiato, soprattutto nella sessione ludicizzata di Sfida. Per quanto riguarda il numero di pagine scoperte (Figura 4.8), anche questo elemento ha visto un aumento significativo; questo dato però è anche

correlato sia alla coverage che al numero di Easter Egg trovati. Nel caso di coverage più alta infatti (soggetto D), il partecipante ha preferito esplorare meno pagine, che infatti risultano molte meno rispetto alla media, in favore di un'esplorazione migliore sulla singola pagina.

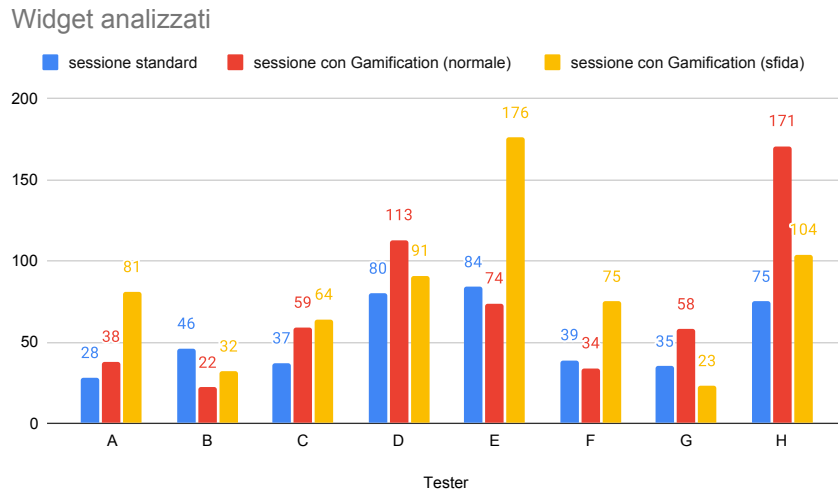


Figura 4.7: Nuovi widget analizzati

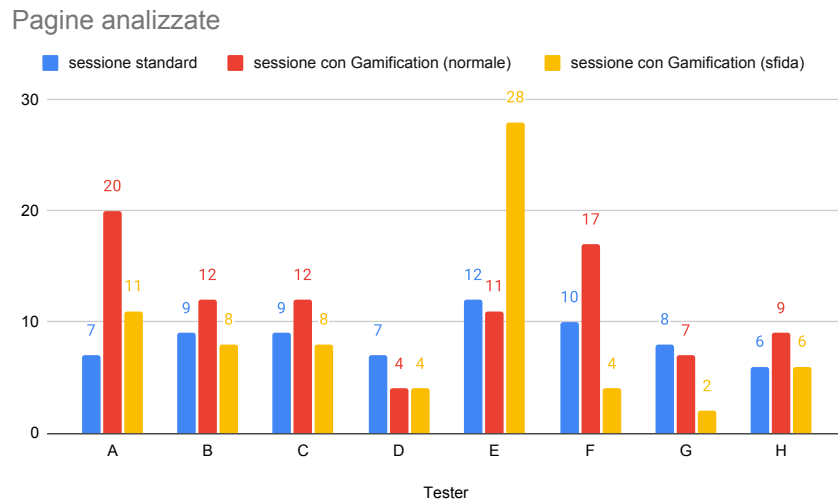


Figura 4.8: Pagine scoperte

Anche gli Easter Egg sono strettamente correlati al numero di pagine scoperte: per ogni pagina che viene visitata infatti viene sorteggiato un collegamento

ipertestuale tra tutti quelli presenti e, nel momento in cui questo venga utilizzato, l'easter egg compare a schermo. E' naturale quindi che l'utente, una volta presa la consapevolezza di questo elemento, possa sentirsi stimolato a cercarlo e per fare ciò venga portato ad esplorare più sotto-pagine possibili di ogni pagina in cui si ritrovi; così facendo però, in caso in cui abbia una stesso tempo a disposizione (come nel caso di questo esperimento), la sua coverage media diminuirà, in quanto avrà meno tempo per analizzare ogni singola pagina. In questo caso (Figura 4.9), sempre per via della scala ridotta del primo sito web, la maggior parte dei partecipanti ha trovato un Easter Egg inconsapevolmente durante la sessione standard; le altre due pagine invece, contenendo un numero piuttosto elevato di collegamenti, sono state un terreno piuttosto difficile in cui cercare easter egg, che in qualche caso sono comunque stati trovati (e in corrispondenza di queste occorrenze sono state analizzate un buon numero di pagine).

In generale si può dire senza dubbio che il plugin di Gamification abbia portato i partecipanti ad esplorare più dettagliatamente i domini scelti, analizzando molti più widget e pagine differenti; per confrontare effettivamente la coverage ottenuta, che è comunque una metrica piuttosto importante nel Software Testing, sarebbe opportuno analizzare siti più simili tra loro, possibilmente per un tempo superiore e con un campione di partecipanti più grande.

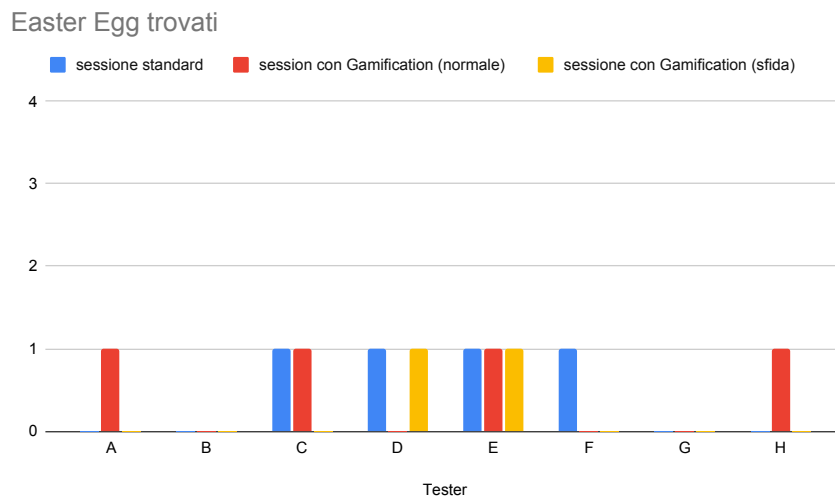


Figura 4.9: Easter egg trovati

4.3 Valutazione delle Plugin di Gamification

Uno dei principali obiettivi dell'esperimento, oltre alla raccolta delle metriche delle sessioni, è stato quello di ottenere dei feedback sull'impatto che ciascun elemento di Gamification ha avuto, in modo da comprendere cosa possa essere effettivamente utile e cosa debba essere modificato o rimosso nelle future implementazioni di questo tipo. A tal fine, sono state inserite nel questionario (Tabella 4.1) le domande dalla 2.2 alla 2.15, ciascuna dedicata ad un elemento differente, i cui risultati sono condensati nel grafico in Figura 4.10. Di seguito ne verranno discussi brevemente gli esiti ed i feedback ricevuti.

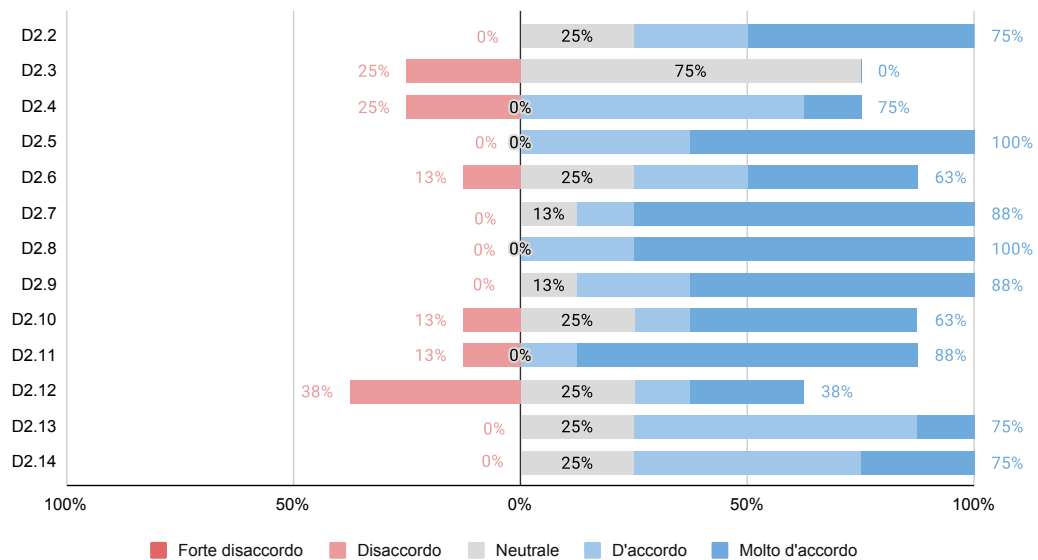


Figura 4.10: Insieme dei likert riguardanti gli elementi di Gamification

4.3.1 Barra di Progresso

Questo elemento è stato recepito bene da praticamente tutti i partecipanti, senza alcun voto negativo. In generale, quasi tutti hanno approvato questo elemento in quanto è molto comodo ed importante avere un indicatore della coverage raggiunta sulla pagina corrente in tempo reale. Un suggerimento mosso però è quello di svincolarne la posizione nel momento in cui la visuale scende lungo una pagina sviluppata in lunghezza, dato che allo stato attuale rimane fissa al bordo superiore della pagina.

4.3.2 Stella

Questa feature ha riscontrato una reazione neutrale nella maggior parte degli individui, con un paio di voti leggermente negativi: questo di per sé può essere dovuto alla natura dell'esperimento, dato che ciascun utente ha esplorato i vari domini con uno stato di progresso iniziale nullo, perciò di fatto ogni pagina era contrassegnata dalla stella.

4.3.3 Easter Egg

L'inserimento di Easter Egg è stata prevalentemente gradita dai partecipanti, con un minor numero di voti negativi: come già spiegato precedentemente, questa divisione può essere dovuta al fatto che risulti potenzialmente insidiosa la ricerca di questo elemento nel momento in cui la pagina che si sta testando è molto estesa e presenta molti potenziali link da controllare. Quest'ultimo aspetto è da combinare al fatto che le performance del tool in sé non sono del tutto ottimali e quindi il caricamento di una nuova pagina risulta lento, da un certo punto di vista disincentivando questo tipo di ricerca.

4.3.4 Classifica

Il punteggio di fine sessione e la relativa classifica sono elementi che risultano indubbiamente utili e necessari per definire visivamente le performance dell'utente e stimolarlo a fare di meglio: hanno di fatto raccolto la maggior parte dei voti nella fascia neutrale-positiva. La presenza di un voto leggermente negativo nei confronti della classifica potrebbe essere dovuta al fatto che questo tipo di elemento è fortemente *sociale*, quindi non tutti i tipi di utenti potrebbero essere egualmente stimolati dall'idea di comparire e competere con altri.

4.3.5 Profilo

Per quanto riguarda il profilo utente, esso è stato recepito bene all'unanimità, prendendo tutti voti positivi ed estremamente positivi. E' stato però suggerito, nella fase di feedback, di renderne il contenuto aggiornato in tempo reale durante la sessione, dato che allo stato attuale la schermata delle statistiche (che mostra alcuni dati utili raccolti durante le sessioni di testing, come ad esempio le nuove pagine scoperte o la percentuale di Easter Egg raccolta) viene aggiornata solamente al concludersi della sessione, per motivi estremamente pratici. Dovrebbe comunque risultare abbastanza fattibile l'implementazione di questo suggerimento.

4.3.6 Obiettivi

Questa feature ha raccolto la maggior parte dei voti nella fascia neutrale-positiva, con un solo voto in quella leggermente negativa: di per sè è abbastanza normale dato che gli Obiettivi sono quasi tutti scelti per essere ottenuti sul lungo termine, quindi era abbastanza probabile che non venissero ottenuti nell'arco delle due sessioni ludicizzate (anche se due partecipanti sono riusciti ad ottenerne uno). La persona che ha dato il voto più basso ha fatto però notare quanto fosse poco visibile il testo che spiega le condizioni di ottenimento di ciascun Achievement, suggerendo di trovare un modo alternativo per mostrarlo. Potrebbe quindi essere interessante un restyling della schermata del profilo, eventualmente dedicando una sezione appositamente per questa feature.

4.3.7 Avatar e Negozio

Le modifiche apportate al sistema di composizione dell'Avatar, e quindi anche al Negozio, sembra siano state gradite dalla maggioranza dei partecipanti, avendo entrambe le relative domande preso voti prevalentemente nella fascia medio-alta. Un suggerimento rivolto al Negozio è stato quello di rendere visibile al suo interno la quantità di monete possedute dall'utente, una mancanza di facile risoluzione.

4.3.8 Livello

Una delle feature inedite di questa iterazione del progetto, anche in questo caso la maggior parte dei voti si colloca nella fascia neutrale-positiva, ma in questo caso il voto più frequente è stato leggermente negativo: questo potrebbe significare che in generale è stato ben recepito come indicatore del progresso individuale all'interno dell'applicazione, ma allo stesso tempo non sono stati messi grandi incentivi che spingano l'utente a salire di livello, dato che allo stato attuale l'unico blocco presente è sull'acquisto di alcuni oggetti nel Negozio. Sarebbe interessante quindi approfondire questo aspetto nelle iterazioni future del plugin.

4.3.9 Missioni

Seconda feature inedita del progetto, è probabilmente uno degli elementi del plugin che ha avuto più influenza sulle sessioni dei partecipanti: quasi tutti hanno infatti provato a completare almeno due quest, e in tutti i casi ne era presente almeno una che richiedesse di scoprire un certo numero di pagine, portando quindi ad una sessione esplorativa più proficua. All'interno del questionario è stato chiesto, con la domanda 2.13, un parere sul bilanciamento di requisiti di completamento e ricompense delle quest, ricevendo dei feedback prevalentemente positivi. Non è stato tuttavia testato, per ovvie ragioni, l'incremento di difficoltà in corrispondenza

del livello dell'utente, dato che nel corso delle due sessioni ludicizzate è stata molto improbabile la possibilità di salire di livello, fattibile solo in caso di completamento di ogni singola quest.

4.3.10 Sfide

Ultima delle nuove feature del progetto, il feedback ricevuto è migliore di quello sperato: come si può vedere nella sezione delle performance, il clima più competitivo delle Sfide ha quasi in tutti i casi posto una maggiore attenzione sul proprio punteggio, portando spesso ad analizzare una quantità superiore di widget e di pagine. Anche per quanto riguarda il questionario, ha ricevuto feedback sulla fascia medio-alta. Non è stato purtroppo possibile, in quanto l'esperimento è per natura di breve durata, ricevere feedback sulla natura temporalmente limitata delle Sfide o sull'ottenimento effettivo di medaglie in corrispondenza della scadenza di queste ultime. In caso in cui questa feature venisse esplorata in qualche modo nelle iterazioni future del plugin, potrebbe essere interessante stabilire nuovi vincoli imposti o nuovi premi per i partecipanti, dato che allo stato attuale l'implementazione è ancora piuttosto embrionale, non aggiungendo particolari dinamiche se non appunto un contesto più competitivo, che è comunque la colonna portante della feature.

4.3.11 Valutazione complessiva

Dalla domanda 2.15 in poi sono stati chiesti dei pareri sull'esperienza complessiva del tool confrontandone la versione standard e quella ludicizzata, gli elementi di quest'ultimo che sono risultati più efficaci e sulla sua integrabilità in contesti pratici, ponendo infine alcune domande per cogliere eventuali consigli sia sul tool in sé che sulle feature di Gamification.

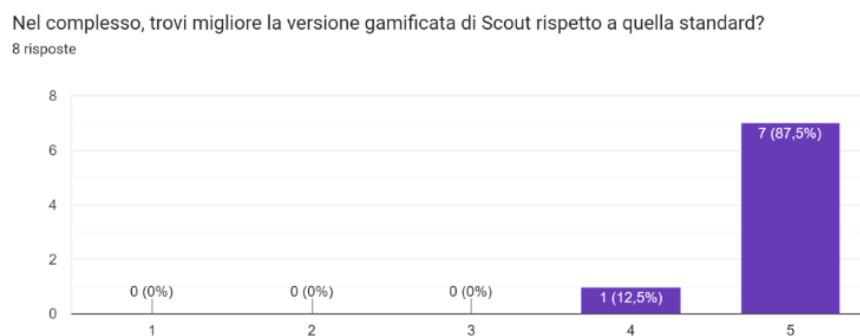


Figura 4.11: Likert sul gradimento complessivo del tool nella sua versione ludicizzata

In primo luogo, tutti i partecipanti hanno espresso di essere stati molto più consapevoli delle proprie performance grazie al plugin di Gamification, come si evince dalla Figura 4.14.

Quest'ultima considerazione è ulteriormente avvalorata dai risultati della domanda 2.17 (Figura 4.12), dove si nota come tutti i partecipanti abbiano concordato all'unanimità l'importanza della barra di progresso e del punteggio, due degli elementi che maggiormente indicano all'utente il proprio progresso, che sia in corso d'opera o quello di fine sessione. I due elementi che si schierano subito dopo come essenziali all'interno del plugin sono gli Obiettivi e le neo-inserite Missioni: mentre il primo elemento si conferma come ottimo metodo per dare un senso di progressione sul lungo termine, le missioni (quest) si sono rivelate un componente utile per arricchire la sessione dell'utente con dei task facilmente risolvibili nel breve termine che lo premiano ulteriormente per le azioni svolte. Tutti gli altri componenti che hanno ricevuto meno voti sono stati comunque graditi ma probabilmente sono visti come "di contorno" rispetto alle funzionalità principali del plugin. Per quanto riguarda infine le componenti che non hanno ricevuto voti, nel caso degli Easter Egg probabilmente può essere dovuto alla natura dell'esperimento, per cui il fatto di trovarne pochi o nessuno può aver indotto un'opinione negativa. Per quanto riguarda invece il Livello, come già detto nella sezione, probabilmente andrebbe meglio integrato con il resto del sistema.

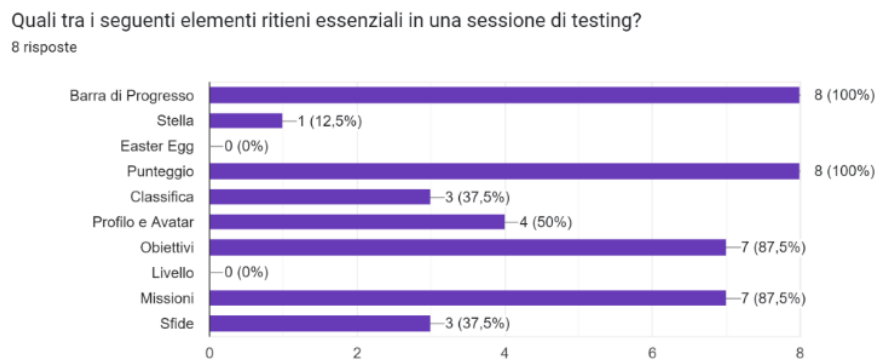


Figura 4.12: Elementi ritenuti più utili durante la sessione

Come si nota dalla Figura 4.13, la domanda 2.18 è stata piuttosto divisiva: il fatto che metà dei voti sia nella fascia negativa è abbastanza significativo. In molti, nella sessione di feedback, hanno infatti espresso delle lamentele nei confronti delle prestazioni e nei comandi del tool Scout, in particolare riguardo all'input lag dovuto all'architettura piuttosto pesante dell'applicazione e al comando di *scroll*, che consente di cambiare l'azione attuabile su widget aventi più di un'azione consigliata e questo in molti casi ha portato a cliccare dei link involontariamente.

In questi casi, inoltre, l'assenza di un pulsante *indietro* ha portato ulteriori perdite di tempo, dato che l'utente era costretto a tornare alla pagina iniziale (con il pulsante *GoHome*) e ripercorrere tutti i collegamenti utilizzati per accedere alla pagina desiderata. Al contempo, però, il fatto che l'altra metà dei voti sia nello spettro positivo fa ben sperare, a patto che gli sviluppatori del tool riescano a trovare delle soluzioni di ottimizzazione.

Nel complesso comunque il plugin ha riscontrato il successo sperato: la domanda 2.20 (Figura 4.11) vede infatti un consenso unanime sulla superiorità dell'esperienza d'uso del tool con plugin di Gamification attivo rispetto alla sua versione base.

Trovi che il tool sia facilmente integrabile in un ambiente di testing esistente (lavorativo o di studio)?
8 risposte

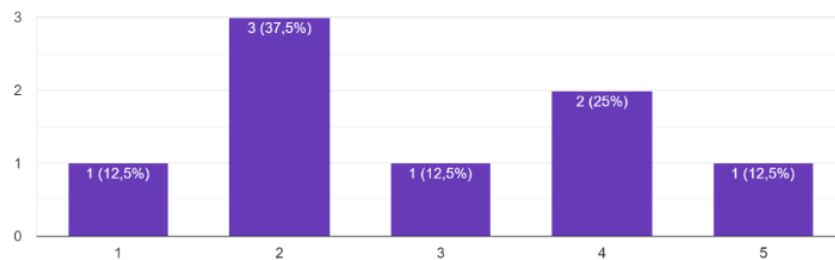


Figura 4.13: Likert sull'integrabilità del tool in contesti reali

Credi che la versione gamificata di Scout abbia influito positivamente sulle tue performance rispetto a quella standard?

8 risposte

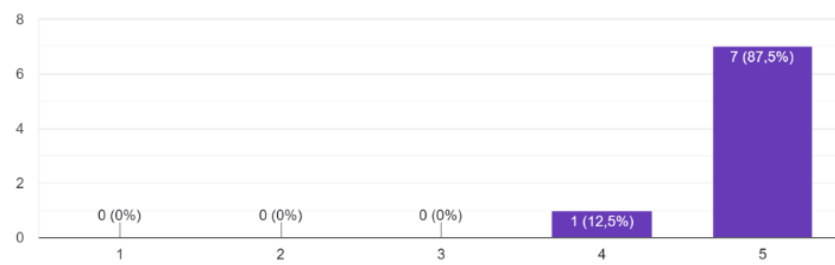


Figura 4.14: Likert sulla consapevolezza indotta dal plugin

Capitolo 5

Conclusioni

Dopo aver descritto i risultati ottenuti dalla fase sperimentale, nonostante questa sia stata molto limitata e abbia coinvolto un piccolo numero di partecipanti, è comunque possibile trarre delle conclusioni su ciò che avuto successo e su ciò che invece ha sottolineato ancor di più le limitazioni del tool, con l'intento di fare un resoconto utile alle future iterazioni di questo plugin di Gamification.

5.1 Limiti del Tool

Come già descritto precedentemente, il tool è stato utilizzato nella sua interezza (incluso quindi sia Scout che i container di Docker costituenti il back-end) su di un unico PC portatile di prestazioni medie su cui la memoria installata è di tipo SSD: nonostante questo setup, molti dei partecipanti hanno riscontrato comunque dei problemi di ottimizzazione come ad esempio input-lag a seguito di un'azione effettuata o del caricamento di una nuova pagina, in base alle dimensioni e alla densità di quest'ultima anche abbastanza importanti, fino a momenti di veri e propri malfunzionamenti, come ad esempio casi in cui l'utente si è trovato il layer di widget analizzati nella pagina precedente sovrapposto alla pagina appena raggiunta, rendendo difficile ed in alcuni casi impossibile interagire con i nuovi widget sottostanti. Va comunque evidenziato che le prestazioni, come fatto notare da alcuni soggetti che avevano avuto esperienza anche delle iterazioni passate del progetto, sono state di gran lunga superiori, considerando però che queste erano state eseguite su *virtual machine* e comprendevano una struttura ancora più pesante, composta anche di un emulatore di dispositivi mobile Android. Alla luce di questo quindi è chiaro come sulla carta Scout possa essere un tool estremamente valido ed interessante, le cui feature non possono che crescere soprattutto grazie al *plugin engine*, tuttavia allo stato attuale necessita di macchine molto potenti

per funzionare in modo ottimale, limitando quindi la sua applicabilità a contesti scolastici/lavorativi su larga scala.

5.2 Lavori Futuri

Alla luce di quanto appena detto sarebbe quindi necessario un lavoro di ottimizzazione del tool; essendo quest'ultimo codice proprietario è tuttavia poco utile pretendere o discutere di questo tipo di considerazioni, dato che il massimo che progetti di questo tipo possono fare è cercare di ottimizzare e rendere il proprio plugin il meno pesante possibile, concentrandosi quindi sulla propria parte dell'applicazione.

Trattando quindi del plugin di Gamification, a partire dal lavoro svolto durante la sua implementazione e dai feedback ricevuti dal questionario è possibile trarre il seguente elenco di tematiche su cui le future implementazioni potrebbero concentrarsi, siano esse affiancate a Scout o ad altri tool di testing:

- Esplorare maggiormente l'elemento ludico dei Livelli, cercando nuove interazioni con le altre dinamiche ludiche.
- Migliorare l'interfaccia ed i feedback grafici, comunicando ad esempio il raggiungimento di un Obiettivo o il completamento di una Missione in tempo reale.
- Implementare il back-end in modo centralizzato, con tutte le opportune modifiche necessarie a gestire casi di sincronizzazione.
- Esplorare maggiormente le Sfide, trovando eventualmente nuove metriche che le rendano esperienze maggiormente diverse da una sessione normale.
- Migliorare la Classifica, aggiungendo ad esempio delle classifiche basate su metriche diverse dal solo punteggio.

Appendice

A GOAL

Qui di seguito vengono elencati alcuni dei template proposti in (Garcia et al. 2017) a supporto del processo di design di Gamification illustrato nell'apposita sezione. Nell'articolo viene sottolineato quanto non sia necessario seguire alla lettera questi modelli, ma che possa comunque risultare utile averli come traccia nella maggior parte dei casi.

A.1 Requisiti di business

Il seguente template può essere utilizzato per definire i requisiti di business.
(Tabella 1)

A.2 Analisi del giocatore

La Tabella 2 illustra un possibile template per l'analisi individuale del giocatore. Il profilo demografico può includere caratteristiche come età, genere, livello di istruzione, posizione lavorativa, etc. Il profilo psicografico potrebbe invece includere interessi, stile di vita, opinioni, aspettative, etc. La Tabella 3 mostra invece il riepilogo delle informazioni appena citate per tutta l'organizzazione. Si fa notare come in questo caso venga utilizzata la tassonomia di Bartle (Bartle 1996) per la classificazione dei giocatori, ma anche altre classificazioni sono altrettanto valide.

A.3 Game Design

Il seguente template può essere utilizzato nel processo di design delle regole di gioco dell'ambiente ludicizzato.
(Tabella 4)

Tabella 1: Template per la documentazione dei requisiti di business

Id	Identificatore del requisito
Obiettivo	Nome.
Descrizione	Descrizione.
Scenario attuale	Descrizione dello scenario riguardante questo requisito allo stato attuale.
Scenario desiderato	Miglioramenti che l'azienda vorrebbe concretizzare.
Indicatori	Indicatori qualitativi/quantitativi che l'azienda vuole utilizzare per analizzare i miglioramenti riguardanti questo requisito.
Valori desiderati	Valori obiettivo da raggiungere con gli indicatori scelti.

Tabella 2: Tabella di analisi del giocatore

Tabella di analisi del giocatore
Cognome, Nome
Profilo demografico
...
...
Profilo psicografico
...
...
Categoria/e di Bartle

B GAMEX

Nella seguente tabella (Tabella 5) vengono illustrati gli elementi che definiscono la scala di valutazione teorizzata in (Eppmann et al. 2018).

Tabella 3: Riepilogo dell'analisi dei giocatori

Ambiente ludicizzato	
Stato dell'organizzazione	
Motivatori Intrinsechi	
Motivatori Estrinsechi	
Tipo di giocatori %	Commenti aggiuntivi
Killer	
Achiever	
Socializer	
Explorer	

Tabella 4: Definizione delle regole di gioco

Regola	Identificatore
Nome	Nome.
Descrizione	Breve descrizione.
Requisiti di business correlati	Dai requisiti identificati nelle fasi precedenti.
Giocatori target	E.g. project manager, sviluppatori, tester, etc.
Elementi di Gamification correlati	E.g. punti, obiettivi, livelli, etc.
Condizioni	Condizioni per il controllo della regola.
Comportamento valutabile	Tipo di comportamento che porta al controllo della regola.
Scenario #1	Primo scenario di controllo: e.g. if<condition ₁ > then <reward ₁ >
Scenario #2	if<condition ₂ > then <reward ₂ >

Tabella 5: Dimensioni di GAMEX e relativi elementi

GAMEX	
<i>Enjoyment</i>	
Enj1	Usare il gioco è stato divertente.
Enj2	Mi è piaciuto usare il gioco.
Enj3	Mi ha divertito davvero molto usare il gioco.
Enj4	L'esperienza di gioco è stata piacevole.
Enj5	Trovo l'utilizzo del gioco davvero intrattenente.
Enj6	Utilizzerei il gioco anche per conto mio, senza che mi venga imposto.
<i>Absorption</i>	
Ab1	Utilizzare il gioco mi ha fatto dimenticare dove fossi.
Ab2	Ho perso di vista le mie immediate vicinanze mentre utilizzavo il gioco.
Ab3	Dopo aver smesso di giocare, ho sentito come se stessi tornando nel "mondo reale" dopo un lungo viaggio
Ab4	Utilizzare il gioco mi ha "portato via da tutto" ^a .
Ab5	Mentre utilizzavo il gioco ho perso la totale percezione di tutto ciò che mi circondasse.
Ab6	Mentre utilizzavo il gioco ho perso il senso del tempo.
<i>Creative thinking</i>	
CT1	Utilizzare il gioco ha acceso la mia immaginazione.
CT2	Mentre utilizzavo il gioco mi sono sentito creativo.
CT3	Mentre utilizzavo il gioco mi sono sentito libero di esplorare.
CT4	Mentre utilizzavo il gioco mi sono sentito avventuroso.
<i>Activation</i>	
Act1	Mentre utilizzavo il gioco mi sono sentito attivo.
Act2	Mentre utilizzavo il gioco mi sono sentito agitato.
Act3	Mentre utilizzavo il gioco mi sono sentito frenetico.
Act4	Mentre utilizzavo il gioco mi sono sentito eccitato.
<i>Absence of negative affect</i>	
ANA1 ^b	Mentre utilizzavo il gioco mi sono sentito sconvolto.
ANA2 ^b	Mentre utilizzavo il gioco mi sono sentito ostile.
ANA3 ^b	Mentre utilizzavo il gioco mi sono sentito frustrato.
<i>Dominance</i>	
Dom1	Mentre utilizzavo il gioco mi sono sentito dominante/ho avuto la sensazione di essere al comando.
Dom2	Mentre utilizzavo il gioco mi sono sentito influente.
Dom3	Mentre utilizzavo il gioco mi sono sentito autonomo.
Dom4	Mentre utilizzavo il gioco mi sono sentito sicuro di me.

^a in inglese dovrebbe essere una citazione ad una canzone di Frank Sinatra, traducendo perde di significato

^b sono elementi *reverse coded*, cioè ne viene misurata l'assenza.

Bibliografia

- Fraser, Gordon (2017). «Gamification of Software Testing». In: *2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST)*, pp. 2–7. DOI: 10.1109/AST.2017.20 (cit. a p. 1).
- Deterding, Sebastian, Dan Dixon, Rilla Khaled e Lennart Nacke (2011). «From game design elements to gamefulness: defining "gamification"». In: *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*, pp. 9–15 (cit. alle pp. 2, 3).
- Garcia, Felix, Oscar Pedreira, Mario Piattini, Ana Cerdeira-Pena e Miguel Penabad (2017). «A framework for gamification in software engineering». In: *Journal of Systems and Software* 132, pp. 21–40 (cit. alle pp. 4, 84).
- Bartle, Richard (1996). «Hearts, clubs, diamonds, spades: Players who suit MUDs». In: *Journal of MUD research* 1.1, p. 19 (cit. alle pp. 5, 84).
- Chou, Y. (2019). *Actionable Gamification: Beyond Points, Badges, and Leaderboards*. Packt Publishing. ISBN: 9781839210778. URL: <https://books.google.it/books?id=9ZfBDwAAQBAJ> (cit. a p. 6).
- Eppmann, René, Magdalena Bekk e Kristina Klein (2018). «Gameful experience in gamification: Construction and validation of a gameful experience scale [GAMEX]». In: *Journal of interactive marketing* 43, pp. 98–115 (cit. alle pp. 10, 85).
- Fu, Yujian e Peter J Clarke (2016). «Gamification-based cyber-enabled learning environment of software testing». In: *2016 ASEE Annual Conference & Exposition* (cit. a p. 13).
- Barata, Gabriel, Sandra Gama, Joaquim Jorge e Daniel Gonçalves (2013). «Engaging engineering students with gamification». In: *2013 5th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES)*. IEEE, pp. 1–8 (cit. a p. 14).
- Sheth, Swapneel, Jonathan Bell e Gail Kaiser (2011). «Halo (highly addictive, socially optimized) software engineering». In: *Proceedings of the 1st International Workshop on Games and Software Engineering*, pp. 29–32 (cit. a p. 15).
- Berkling, Kay e Christoph Thomas (2013). «Gamification of a Software Engineering course and a detailed analysis of the factors that lead to it's failure». In:

- 2013 international conference on interactive collaborative learning (ICL)*. IEEE, pp. 525–530 (cit. a p. 16).
- Rojas, José Miguel e Gordon Fraser (2016). «Code defenders: a mutation testing game». In: *2016 IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, pp. 162–167 (cit. a p. 18).
- Fraser, Gordon, Alessio Gambi, Marvin Kreis e José Miguel Rojas (2019). «Gamifying a software testing course with code defenders». In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pp. 571–577 (cit. a p. 18).
- Parizi, Reza Meimandi, Asem Kasem e Azween Abdullah (2015). «Towards gamification in software traceability: Between test and code artifacts». In: *2015 10th International Joint Conference on Software Technologies (ICSOFT)*. Vol. 1. IEEE, pp. 1–8 (cit. a p. 19).
- Parizi, RM (2016). «On the gamification of human-centric traceability tasks in software testing and coding. In 2016 IEEE/ACIS 14th International Conference on Software Engineering Research, Management and Applications, SERA (pp. 193–200)». In: *IEEE*. <https://doi.org/10.1109/SERA> (cit. a p. 20).
- Alégroth, Emil, Zebao Gao, Rafael Oliveira e Atif Memon (2015). «Conceptualization and evaluation of component-based testing unified with visual gui testing: an empirical study». In: *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, pp. 1–10 (cit. a p. 21).
- Linares-Vásquez, Mario, Kevin Moran e Denys Poshyvanyk (2017). «Continuous, evolutionary and large-scale: A new perspective for automated mobile app testing». In: *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, pp. 399–410 (cit. a p. 22).
- Nass, Michel e Emil Alégroth (n.d.). «SCOUT: A revised approach to GUI-test automation». In: () (cit. a p. 24).
- Nass, Michel, Emil Alégroth e Robert Feldt (2020). «On the industrial applicability of augmented testing: An empirical study». In: *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, pp. 364–371 (cit. a p. 24).