

RELAZIONE DI PROGETTO  
DI  
PARADIGMI DI PROGRAMMAZIONE E SVILUPPO

---

**ISIQuiz**

---

Gambaletta Daniele  
[daniele.gambaletta@studio.unibo.it](mailto:daniele.gambaletta@studio.unibo.it)

Lirussi Igor  
[igor.lirussi@studio.unibo.it](mailto:igor.lirussi@studio.unibo.it)

Omiccioli Riccardo  
[riccardo.omiccioli@studio.unibo.it](mailto:riccardo.omiccioli@studio.unibo.it)

Teodorani Cecilia  
[cecilia.teodorani@studio.unibo.it](mailto:cecilia.teodorani@studio.unibo.it)

12 dicembre 2022

## **Sommario**

In questo documento viene presentata la relazione dettagliata di ogni fase dello sviluppo del software ISIQuiz. Il progetto è volto alla creazione di un gioco a quiz che possa facilitare il ripasso in vista di un esame. L'obiettivo verrà raggiunto attraverso un processo strutturato di analisi, design e implementazione descritto a seguire.

# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
1.1	Overview . . . . .	3
1.2	Stato dell'Arte . . . . .	3
<b>2</b>	<b>Processo di Sviluppo</b>	<b>5</b>
2.1	Domain Driven Design . . . . .	5
2.1.1	Aspetti principali . . . . .	5
2.2	Metodologia di Sviluppo . . . . .	5
2.2.1	Scrum . . . . .	6
2.3	Gestione di Progetto . . . . .	6
2.4	Continuous Integration e Automatizzazione . . . . .	9
2.4.1	Report . . . . .	9
2.4.2	Progetto Software . . . . .	10
<b>3</b>	<b>Analisi dello spazio del problema</b>	<b>12</b>
3.1	Intervista con il committente . . . . .	12
3.2	Ubiquitous Language . . . . .	14
<b>4</b>	<b>Analisi dei Requisiti</b>	<b>15</b>
4.1	Requisiti di Business . . . . .	15
4.2	Requisiti Utente . . . . .	15
4.2.1	User Stories . . . . .	15
4.2.2	Mockup . . . . .	16
4.3	Requisiti Funzionali . . . . .	20
4.3.1	Obbligatori . . . . .	20
4.3.2	Opzionali . . . . .	22
4.4	Requisiti Non Funzionali . . . . .	22
4.5	Requisiti Implementativi . . . . .	23
<b>5</b>	<b>Design Architetturale</b>	<b>24</b>
5.1	Architettura Generale . . . . .	24
5.1.1	View . . . . .	25
5.1.2	Controller . . . . .	25
5.1.3	Model . . . . .	26
<b>6</b>	<b>Design di Dettaglio</b>	<b>27</b>
6.1	Model . . . . .	27
6.2	View . . . . .	30
6.3	Controller . . . . .	31
<b>7</b>	<b>Implementazione</b>	<b>33</b>
7.1	Gambaletta Daniele . . . . .	33
7.2	Lirussi Igor . . . . .	34
7.3	Omiccioli Riccardo . . . . .	35
7.4	Teodorani Cecilia . . . . .	36

<b>8 Testing e Performance</b>	<b>38</b>
8.1 Testing . . . . .	38
8.2 Performance . . . . .	38
<b>9 Retrospettiva</b>	<b>40</b>
9.1 Svolgimento . . . . .	40
9.2 Riepilogo Sprint . . . . .	40
9.2.1 Sprint 1 . . . . .	40
9.2.2 Sprint 2 . . . . .	41
9.2.3 Sprint 3 . . . . .	41
9.2.4 Sprint 4 . . . . .	41
9.2.5 Sprint 5 . . . . .	41
9.2.6 Sprint 6 . . . . .	42
9.2.7 Sprint 7 . . . . .	42
9.2.8 Sprint 8 . . . . .	42
9.2.9 Sprint 9 . . . . .	42
9.2.10 Sprint 10 . . . . .	43
9.3 Diagramma di Gantt . . . . .	43
<b>10 Conclusioni</b>	<b>45</b>
10.1 Sviluppi Futuri . . . . .	45
<b>11 Guida Utente</b>	<b>46</b>

# Capitolo 1

## Introduzione

ISIQuiz è un progetto che nasce con l'intento di sviluppare un **gioco a quiz** che permetta ad uno studente di ripassare il contenuto dei corsi di **Ingegneria e Scienze Informatiche** attraverso domande a scelta multipla che coprono una o più materie di suo interesse. Lo scopo di **ISIQuiz** è quindi quello di gamificare il ripasso pre-esame attraverso uno strumento stimolante che permetta agli studenti di tracciare i propri progressi.



Figura 1.1: Il logo del progetto

### 1.1 Overview

L'idea è nata dall'osservazione degli incontri tra studenti nelle giornate antecedenti ad una prova orale o scritta. Per prepararsi al meglio sulle nozioni teoriche, nel gruppo si crea una dinamica simile a docente-studente, dove a turno ognuno fa domande di carattere generale su alcuni argomenti alle quali gli altri presenti cercano di rispondere. In questo modo, lo studente che risponde alle domande è in grado di ripassare e, con un numero sufficiente di quesiti, coprire gli argomenti dell'intera materia. Questa modalità di ripasso in gruppo favorisce una copertura migliore del programma del corso rispetto ad uno studio individuale, nel quale alcuni argomenti potrebbero essere involontariamente trascurati. Lo scopo di questo progetto è quindi quello di cercare di ricreare le dinamiche appena indicate nel caso in cui non sia possibile un incontro di gruppo tra gli studenti.

### 1.2 Stato dell'Arte

Allo stato attuale, sono molte le varianti di giochi a quiz disponibili sul mercato. La motivazione principale risiede nel fatto che di questa tipologia di gioco ne è stato fatto largo uso fin dall'antichità [qui] e, successivamente, con l'avvento della televisione ha avuto una seconda ondata di interesse [gam]. Le varianti più di rilievo mantengono sempre il concetto di rispondere ad una domanda o scegliendo tra più risposte quella corretta [who], o con degli aiuti di diversa natura, o con un tempo limite [jeo]. Anche dal punto di vista didattico, sono state sviluppate piattaforme, ad esempio **Quizlet**, ma le soluzioni trovate sono orientate alla creazione dei quiz da parte degli insegnanti per gli studenti.

L'idea di questo progetto si differenzia, in parte, per:

- la natura accademica orientata a corsi specifici di un determinato corso di laurea e laurea magistrale;
- l'utente target, in quanto ne usufruisce direttamente lo studente e non il docente.

L'intenzione con la quale si vuole portare avanti il progetto è quella di produrre un prodotto superiore a quelli disponibili nel mercato per una specifica necessità riscontrata, non per apportare innovazione tecnologica.

# Capitolo 2

## Processo di Sviluppo

In questo capitolo viene analizzato l'intero processo di sviluppo definito dal team.

### 2.1 Domain Driven Design

Fin dall'inizio, è stato scelto un approccio di design fortemente basato sul dominio. Infatti nella discussione preliminare sull'idea, sono emersi molti dubbi sul significato dei vocaboli usati e su concetti generali di funzionamento. Questo ha spinto il team ad avvalersi naturalmente di schemi per chiarificare e uniformare le astrazioni di ogni team-member. Questi schemi sono stati mantenuti e aggiornati per tutta la durata del progetto, uniformando l'artefatto generato con il modello del dominio. Questo permette anche la successiva evoluzione del progetto in modo più semplice.

#### 2.1.1 Aspetti principali

I concetti chiave che hanno assunto particolare importanza nello sviluppo sono stati:

- **Domain Expert interview:** per formalizzare le richieste e non permettere al software di allontanarsi da esse nello sviluppo, l'intervista con l'esperto del dominio è stata formalizzata e trascritta insieme a tutte le domande (capitolo 3.1)
- **Linguaggio Condiviso:** per evitare continue spiegazioni tra i membri del team, per indicare a quale concetto si stesse facendo riferimento, è stata creata una tabella di "Ubiquitous Language" (capitolo 3.2)
- **User Stories:** per assicurare che l'esperienza sia in linea con quella desiderata e che il software copra ogni interazione utente richiesta, sono state definite tutte le user stories che sono emerse dall'intervista con il committente (capitolo 4.2.1)
- **Mockup:** per verificare che il team abbia compreso a pieno l'aspettativa del risultato, sono stati sviluppati dei mockup, che il cliente ha verificato e approvato (capitolo 4.2.2)

### 2.2 Metodologia di Sviluppo

Per sviluppare il progetto è stato scelto il framework **Scrum**, infatti il processo di sviluppo è stato basato, in comune accordo, sulla metodologia agile. La motivazione risiede nel fatto che il team è composto da quattro persone e il prodotto deve essere sviluppato in tempi brevi. Il framework Scrum infatti permette di suddividere i compiti in modo preciso, sviluppando il progetto in passi incrementali, assicurando di coprire man mano tutti i requisiti richiesti dal cliente.

### 2.2.1 Scrum

Secondo il framework Scrum il lavoro deve essere diviso in più sprint seguendo un approccio iterativo. Per questo motivo l'intervallo di tempo disponibile per lo sviluppo del progetto è stato suddiviso in sotto-intervalli di una settimana. Il team mantiene due tipi di backlog, i quali riflettono l'organizzazione spiegata sopra: il product backlog e lo sprint backlog.

**Product Backlog** Il product backlog racchiude il progresso generale del progetto, in cui ogni item riflette una user-story, o una funzionalità ad alto livello, di cui il cliente può apprezzare lo sviluppo. Queste vengono ordinate per urgenza e dimensione e, scelto un sottoinsieme, assegnate allo sprint successivo. In ogni sprint è presente infatti lo sprint backlog, in cui si può apprezzare l'implementazione più dettagliata e a basso livello delle varie feature.

**Sprint Backlog** Lo sprint backlog, relativo ad ogni iterazione, va a raffinare nella riunione iniziale (sprint planning) gli item scelti (sprint goal) dal product backlog. Questi sono suddivisi in più task e assegnati a diversi membri del team. Nel meeting finale (sprint review) lo sprint backlog viene ispezionato, riordinato e utilizzato per ulteriori considerazioni sull'iterazione successiva. Per maggiori dettagli sul processo di sviluppo di ogni iterazione si veda il capitolo 9.

**Definition of done** Un item del product backlog o sprint backlog si ritiene concluso quando tutti i task che lo compongono sono stati completati.

**Scrum poker** Per stimare il livello di effort necessario per il completamento di ogni task è stata utilizzata la tecnica che viene chiamata **scrum poker**. Questa tecnica consiste in una fase preliminare di lettura e discussione di un task e degli aspetti che lo riguardano. Successivamente, ogni membro del team sceglie tra 1, 2, 3, 5, 8, 13, 20 il numero che ritiene più adeguato a rappresentare la complessità del task, ad esempio tenendo conto del tempo ritenuto necessario per lo sviluppo o la difficoltà stimata del task stesso. A seguito della scelta di un numero da parte di ogni membro, vengono rivelati i numeri scelti e si cerca di raggiungere consenso sulla scelta del numero finale, eventualmente argomentando la propria decisione. La scelta del set di numeri da assegnare è tale da avere volutamente ampi intervalli tra i numeri per ridurre conflitti e raggiungere con maggiore semplicità una situazione di consenso.

**Ruoli** Dal momento che il team è composto da solo quattro persone, il "domain-expert", lo "scrum master" e il "product owner" hanno preso parte allo sviluppo del progetto anche come programmatore.

## 2.3 Gestione di Progetto

In questa sezione verrà spiegato come il progetto è stato gestito con diversi tool seguendo la metodologia Scrum.

**Diagramma di Gantt** Per l'organizzazione generale del progetto è stato utilizzato un diagramma di Gantt realizzato grazie ai tool forniti da [Jira](#), come si può notare da Figura 2.1 e Figura 2.2. Questo ci ha permesso di avere una visione globale dei momenti del progetto e dei task da svolgere in ogni fase. Inoltre, lo strumento è stato utile per monitorare il lavoro svolto nelle varie deadline.

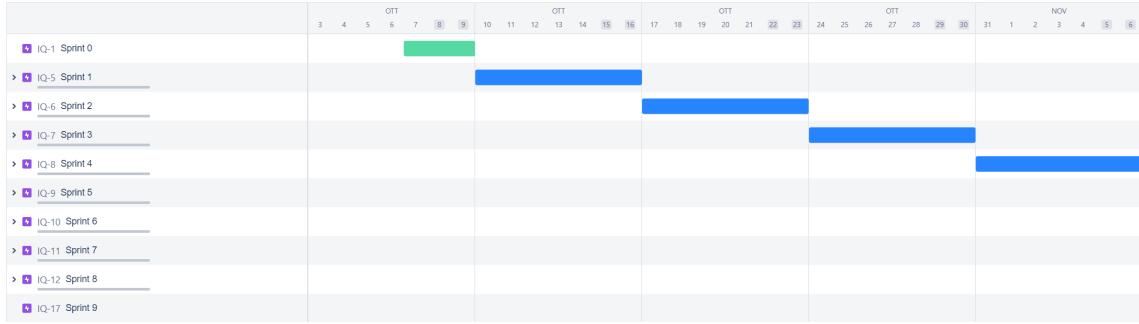


Figura 2.1: Diagramma di Gantt all'inizio del processo di sviluppo

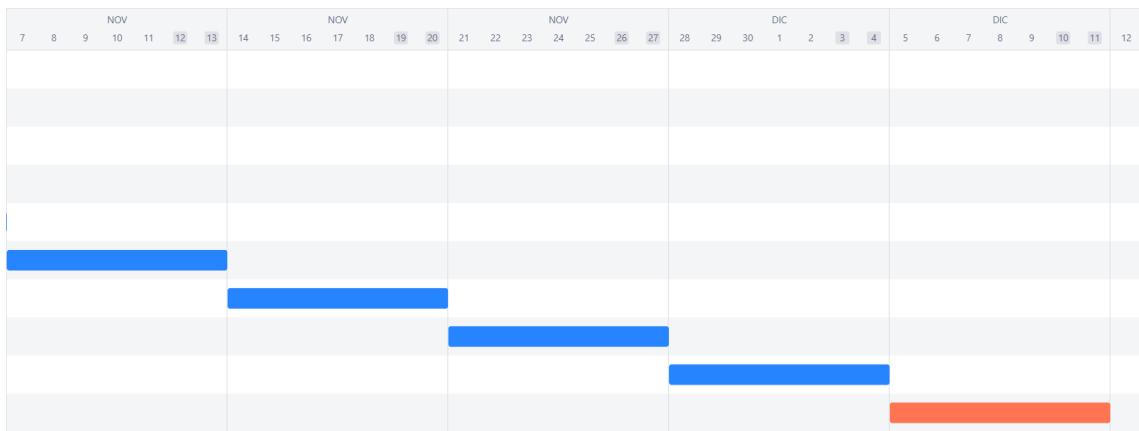


Figura 2.2: Diagramma di Gantt all'inizio del processo di sviluppo

**Collaborative Design Board** Per lavorare simultaneamente su grafici e schemi che sono stati molto utili nelle fasi iniziali di analisi e design, è stato utilizzato [Miro](#).

**Product-Backlog** Per realizzare le board secondo il framework Scrum è stato utilizzato [GitHub-Projects](#), un tool che si è rivelato molto potente, grazie anche all'integrazione diretta con [GitHub](#). Ad esempio, la chiusura delle issue avviene automaticamente con la merge del relativo branch relativo e gli item corrispondenti vengono spostati correttamente nella board. Inoltre, la possibilità di avere diverse viste sugli stessi dati ha permesso di sviluppare product e sprint backlog senza ripetizioni o duplicazioni. La creazione di diverse view ha permesso anche ai membri del team di osservare facilmente il progresso, la priorità e la grandezza dei task. I vari raffinamenti sono stati referenziati alle issue principali tramite la loro descrizione, in modo da poter osservare il completamento passo passo delle user-stories. Unica mancanza da parte del tool, è quella di non poter assegnare delle issue a più sprint backlog, qualora non venissero completate solamente in uno di essi.

**Licensing** Per la licenza è stata scelta [la licenza MIT base](#), in quanto il progetto è di natura accademica e non commerciale.

**Versioning** Riguardo al versioning, la piattaforma scelta è stata [Git](#). Dato che il progetto è stato svolto in tempi ristretti, è stato scelto il versioning semantico unito a quello temporale, per cui in ogni versione è presente un'indicazione data da MAJOR.MINOR.PATCH in unione alla data di rilascio.

**Git Flow** Per la gestione del workflow del repository del progetto, è stato utilizzato il modello di branching [Git Flow](#) (come schematizzato in Figura 2.3). Il branch *main* è stato utilizzato

solamente per pubblicare le release del software finito o suoi eventuali aggiornamenti. Il branch *dev* viene invece aggiornato con le versioni stabili di tutte le feature man mano che vengono implementate. Per ogni issue indicata negli sprint backlog, viene creato un branch apposito a partire da *dev*, nel quale verrà implementata la nuova funzionalità per poi essere infine riportata sul branch *dev* una volta terminata.

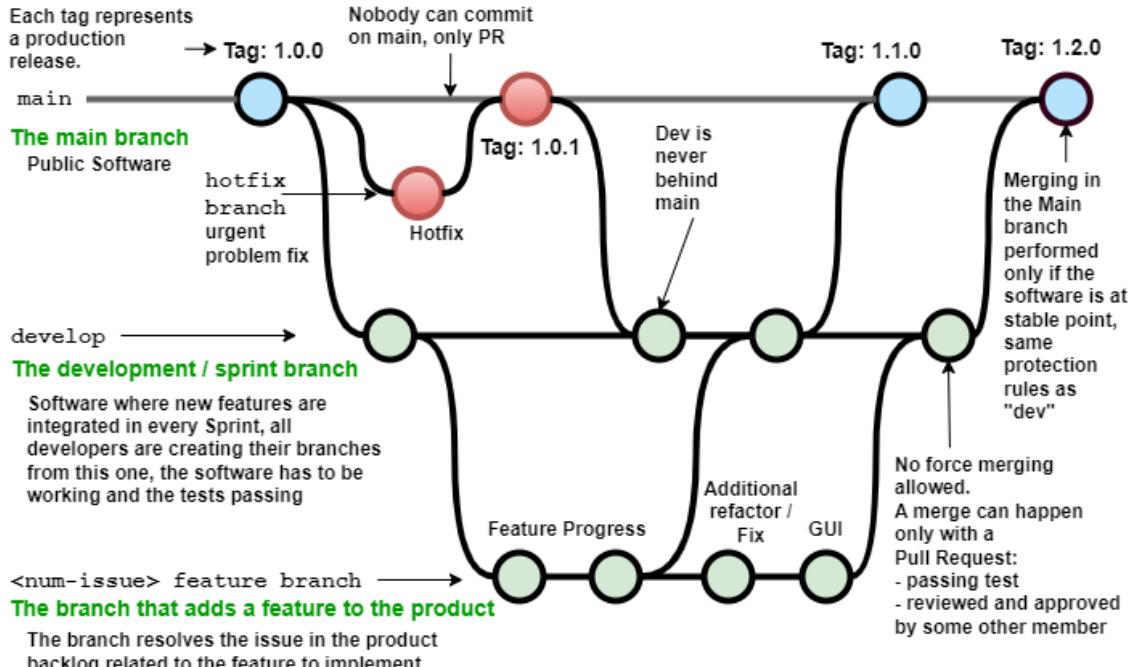


Figura 2.3: Workflow di Git utilizzato

**Quality Assurance** Per garantire il giusto processo di sviluppo appena indicato, sono state attivate delle **Branch Protection Rules**. Esse obbligano uno sviluppatore che contribuisce al progetto ad eseguire una pull request prima di poter fare la merge sul branch *dev* del codice da lui implementato nel branch di una feature. Ogni pull request deve ricevere l'approvazione dal almeno uno degli altri membri del progetto e deve anche eseguire correttamente tutti i test automatici presenti nel progetto. Le stesse regole sono state applicate anche quando viene effettuata una release sul branch *main* del codice presente sul branch *dev*.

**Discord** Discord è stato utilizzato come strumento di comunicazione vocale per i meeting, per la sua capacità di condividere lo schermo e di supportare chiamate di gruppo. Inoltre, particolarmente utile è stata la divisione delle chat in canali in cui organizzare logicamente il materiale, i link e le discussioni.

**Telegram** La scelta di **Telegram** come strumento di comunicazione informale è motivata dal fatto che dispone di funzionalità avanzate orientate ai programmati e che tutti i membri del gruppo la utilizzano personalmente come applicazione di messaggistica.

**Telegram Bot** Per mantenere tutti i membri aggiornati sul progresso del progetto, è stato creato ed inserito nel gruppo Telegram un Bot. Tale Bot è stato utilizzato per notificare ogni richiesta di review di una pull request e ciascun caricamento di una nuova versione del report. Dato l'elevato tempo necessario alla CI per terminare il processo di generazione del PDF del report o del sito web del progetto, le notifiche del Bot sono state utili per verificare l'esito di tali operazioni una volta portate a termine.

## 2.4 Continuous Integration e Automatizzazione

Nel progetto la Continuous Integration (CI) e l'automatizzazione delle mansioni più ripetitive hanno avuto notevole importanza. Una cospicua quantità di tempo è stata dedicata all'inizio per la messa in opera di pipeline che permettano di risparmiare tempo successivamente agli sviluppatori e di garantire la massima qualità al cliente finale.

### 2.4.1 Report

I tool utilizzati per la CI del report sono:

- [L<sup>A</sup>T<sub>E</sub>X](#)
- [Overleaf](#)
- [GitHub Actions](#)
- [Telegram Bots](#)
- [Pandoc](#)
- [GitHub Pages](#)

**L<sup>A</sup>T<sub>E</sub>X** La relazione di progetto è stata svolta in L<sup>A</sup>T<sub>E</sub>X. Questa decisione è motivata dal fatto che esso permette facilmente di organizzare lunghi testi in capitoli e sottocapitoli, eventualmente riordinando intere parti senza preoccuparsi dell'indice o della formattazione. Inoltre, permette la gestione in maniera più accurata, rispetto al Markdown, di elementi grafici, tabelle, stili, link e referenze, producendo un PDF finale di qualità elevata.

**Overleaf** Come strumento complementare è stato scelto Overleaf, una piattaforma che permette la collaborazione ininterrottamente e simultanea sullo stesso blocco di testo. Questo è risultato utile nella produzione dei documenti relativi agli "Sprint-planning" e, soprattutto, durante gli "Sprint-review".

**GitHub-Actions** Per la Continuous Integration, sono state sfruttate a pieno le GitHub-Actions messe a disposizione da GitHub. Queste permettono di generare il PDF finale in automatico e di renderlo pubblico con Release istantanee ad ogni cambiamento. Ciò ha non solo alleviato la preoccupazione di farlo manualmente, ma ha anche permesso di rendere disponibile al cliente sempre l'ultima versione aggiornata.

**Bot Telegram** Per garantire la massima qualità, un Bot Telegram è stato utilizzato come strumento di notifica nel gruppo Telegram dedicato. Il Bot notifica, come in Figura 2.4, qualora ci sia qualunque problema in ogni step della CI o, in caso di successo, avverte che l'azione è stata portata a termine correttamente ed inoltra il PDF sul gruppo.

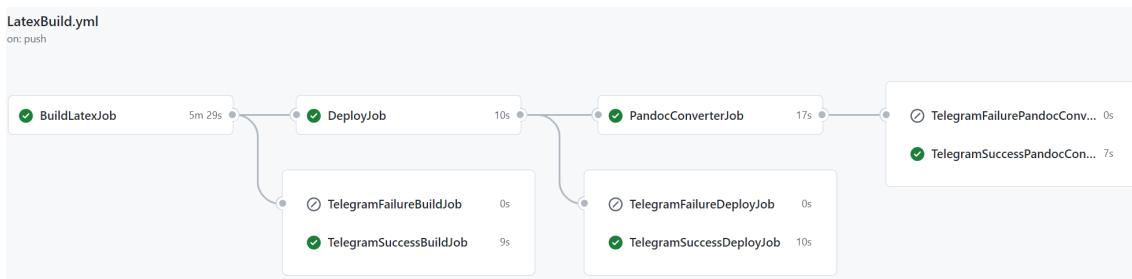


Figura 2.4: Pipeline Continuous Integration per il report

**Pandoc** Qualora una persona volesse consultare la relazione online, senza scaricare il PDF, si è fatto uso di Pandoc per generare il relativo sito web consultabile direttamente sul browser. Per la paginazione, è stato utilizzato un template HTML/CSS per rendere la vista più accattivante.

**GitHub-Pages** Per il deploy del sito è stato utilizzato GitHub-Pages che permette di semplificarne la messa in opera avendo un branch dedicato sul quale tutti i file relativi possono essere mantenuti e aggiornati.

### 2.4.2 Progetto Software

I tool utilizzati per la CI del software sono:

- Sbt 1.7.2
- ScalaTest 3.2.14
- Sbt-assembly 2.0.0
- GitHub Actions
- [GitHub Boards](#)
- GitHub Pages
- Dependabot
- ScalaDoc
- ScalaCoverage
- Telegram Bots
- Sonarcloud/Codacy/Codefactor
- Jira
- Miro
- [PlanItPoker](#)

Per quanto riguarda la Continuous Integration del software, come prima cosa è stato fatto il setup dello **Scala Build Tool**, specificando la versione di scala da utilizzare nel progetto, la versione di sbt stesso, inserendo le dipendenze di scalatest e il plugin sbt-assembly negli appositi file di configurazione. In modo analogo alla relazione, sono state poi utilizzate le **GitHub-Actions** per automatizzare la verifica dei test e per la release dei jar eseguibili. In particolare, ad ogni commit, i test vengono eseguiti nella **GitHub-Actions** bloccando, in caso di fallimento, la merge sugli altri branch. Nel caso in cui sia possibile eseguire il merge di una pull request, viene inviato un messaggio sul gruppo Telegram, grazie a **GitHub-Actions** e **Telegram-bot**, che notifica l'avvenuta risoluzione del task e la necessità di una review.

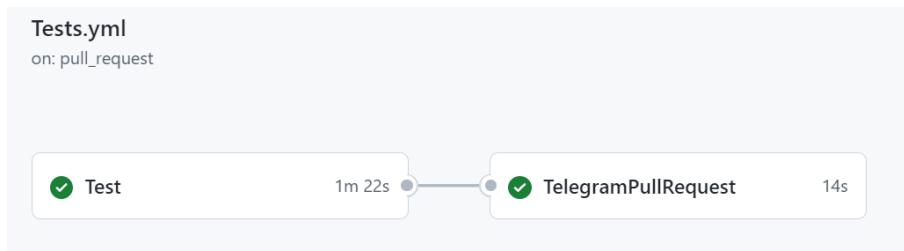


Figura 2.5: Pipeline per l'esecuzione dei test e la notifica di una pull request

Nel caso in cui una pull request venga correttamente chiusa portando il codice sviluppato sul branch *dev*, la issue relativa alla feature implementata viene automaticamente chiusa su **GitHub-Boards**. La Continuous Integration è stata utilizzata anche per generare la pagina principale del sito web del software, dove sono presenti i link al report, alla coverage e alla documentazione del software realizzata tramite **ScalaDoc**. Per automatizzare il processo di deploy del sito web è stato utilizzato **GitHub-Pages**, mentre per generare la parte relativa al riepilogo dei test è stato usato **ScalaCoverage**.

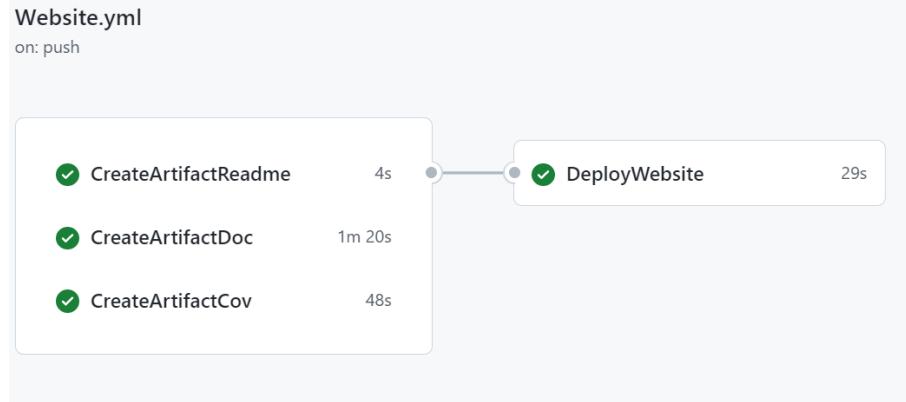


Figura 2.6: Pipeline per il deploy del sito web

Per le release del software è stata realizzata una pipeline eseguita ogni volta che delle funzionalità vengono spostate dal branch *dev* al branch *main*. Tale pipeline è strutturata in modo tale da eseguire tutti i test e generare lo *uber jar* (ovvero un *jar* contenente tutte le dipendenze e le risorse necessarie per l'esecuzione indipendente) in caso di completo successo dei test. Per la creazione del jar eseguibile è stato utilizzato **Sbt-assembly** e viene automaticamente caricato tra nelle release del progetto.

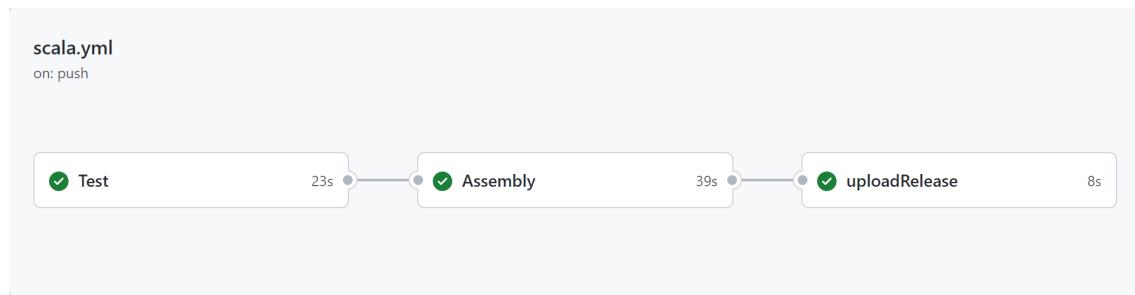


Figura 2.7: Pipeline per il deploy delle release

## Capitolo 3

# Analisi dello spazio del problema

Inizialmente l'esperto del dominio è stato contattato per un'intervista preliminare (sotto riportata). La sessione si è svolta in via telematica ed ha avuto lo scopo di prendere familiarità con l'ambito di lavoro e individuare i requisiti del software ad esso collegato. Nel colloquio il problema è emerso chiaramente e la soluzione software è stata individuata in maniera univoca. In seguito sono stati definiti incontri periodici con il committente per la valutazione del progresso e i feedback sui deliverables di ogni sprint.

### 3.1 Intervista con il committente

« Sono uno studente di Ingegneria e Scienze Informatiche dell'Università di Bologna presso la sede di Cesena. In questi anni ho dovuto studiare e ripassare la teoria per molti esami e ho trovato utile ripassare con i miei compagni rispondendo ad alcune domande alle quali dover rispondere, così da verificare la conoscenza della materia. In particolare, dopo aver acquisito una discreta quantità di nozioni, ho riscontrato la necessità di rivedere, in maniera generale, se riuscissi a ricordare i concetti principali per ogni topic dell'esame in questione. Per questo motivo, sarebbe utile poter avere a disposizione uno strumento che permetta di rendere più stimolante il ripasso pre esame. Ad esempio, vorrei non dover più scrivere su un foglio le domande e le relative risposte corrette e sbagliate. In più, vorrei poterle integrare con quelle dei miei compagni e viceversa, in modo da coprire il più possibile tutti gli argomenti di un esame. Trovo utile poter ripassare più materie contemporaneamente nei periodi di sessione, in cui ho più esami nella stessa settimana. Inoltre vorrei poter rispondere a più domande possibili nel minor tempo possibile, così da essere sicuro di riuscire a rispondere velocemente durante gli orali.

- a. **Abbiamo pensato che il modo più divertente per poter studiare e ripassare potrebbe essere quello di fare un gioco a quiz, dove viene posta una domanda e quattro possibili risposte. Potrebbe essere utile?**

Sarebbe molto interessante, così non mi annoierei. Metterei un'unica risposta corretta e le altre tre sbagliate. Ogni volta che ripasso una materia in particolare, le domande sono sempre le stesse? Perché a me piacerebbe rispondere correttamente attraverso il ragionamento, non solo ricordandomi a memoria la risposta data in precedenza.

- b. **Per quanto riguarda il numero di domande per ogni argomento, abbiamo pensato di inserirne una quantità tale da avere sempre un ricambio adeguato ed evitare il ricircolo delle stesse. Inoltre anche le risposte saranno un numero superiore alle quattro visi-**

**bili per ogni domanda. Quali modalità di gioco vorresti avere a disposizione?**

Mi piacerebbe poter scegliere se rispondere a domande di uno specifico corso o di più corsi durante lo stesso quiz.

- c. Riguardo al tempo disponibile per rispondere, preferisci avere un tempo limite per ogni singolo quesito o per l'intera sessione di gioco?**

Sicuramente vorrei avere un tempo limitato ad ogni domanda, per non incorrere nella tentazione di barare. Qualora fosse possibile avere anche una modalità di gioco con un tempo limite globale sarebbe molto divertente.

- d. Sarebbe interessante mettere un sistema di punteggi e se si come vorresti venisse implementato?**

Secondo me aggiungere un punteggio renderebbe ancora più stimolante il gioco e permetterebbe di confrontarmi con i miei colleghi di corso. Ogni domanda potrebbe avere un punteggio in base alla sua complessità, inoltre si potrebbe calcolare in base a quanto tempo si impiega per rispondere.

- e. Invece quali schermate di gioco vorresti visualizzare?**

Vorrei poter scegliere le varie modalità di gioco, poter aggiungere ed esportare le domande e le risposte dei corsi e visualizzare le statistiche sulle partite effettuate in precedenza. A fine partita vorrei visualizzare un riepilogo di tutte le domande presenti nel quiz appena concluso ed eventualmente, in caso di risposta errata, visualizzare la risposta corretta. »

## 3.2 Ubiquitous Language

Per chiarire il gergo usato tra il committente e gli sviluppatori è risultato essenziale adottare dei significati condivisi per i termini maggiormente usati. Ciò ha aiutato notevolmente anche la comunicazione tra gli sviluppatori stessi per non incorrere in malintesi.

Ubiquitous Language Table

Termino	Definizione
Corso	Insieme di TUTTI i quiz inerenti uno specifico argomento accademico corredato da nome, corso di laurea, università, descrizione
Partita	Insieme di quiz proposti al giocatore in una sessione, possono essere presi da diversi corsi o da uno singolo. La partita può comprendere solo alcuni dei quiz di un corso, avendo numero finito o proseguendo fino ad esaurimento.
Quiz	Insieme di domanda, risposte esatte e sbagliate, con eventuali immagini optionali
Domanda	Soltanto la stringa di testo interrogativa propria di ogni quiz
Risposta	Possibile scelta del giocatore associata ad una domanda, può essere giusta o sbagliata
Immagine	eventuale immagine allegata alla domanda del quiz, identificata da un image path
Score	Punteggio corrispondente alla difficoltà della domanda e ai punti guadagnati dall'utente per il completamento del corso o per le statistiche.
Tempo	Tempo limite per rispondere a una domanda di un quiz, dopo il quale viene considerata sbagliata.
Menu	Interfaccia che permetta al giocatore di interagire con l'applicazione, prendendo decisioni, può essere a riga di comando o grafica.
Riepilogo	Schermata a fine partita che permetta al giocatore di analizzare le risposte giuste e sbagliata
Statistiche	Insieme di tutti i dati del giocatore, permette ad esso di avere una visione generale delle sue partite e dei suoi progressi
Impostazioni di gioco	Decisioni che l'utente può prendere per variare la sua esperienza in gioco o personalizzarla con differenti corsi e domande
Blitz	Modalità di gioco sfida contro il tempo, in cui si rispondono più domande possibili in un determinato periodo

miro

Figura 3.1: Tabella ubiquitous language

# Capitolo 4

## Analisi dei Requisiti

A seguito dell'intervista con l'esperto del dominio, sono emersi i seguenti requisiti per l'applicativo richiesto. Questi sono stati poi discussi e approvati dal committente.

### 4.1 Requisiti di Business

L'importanza strategica del software risiede nel fornire agli studenti un valido strumento di ripasso e test delle competenze acquisite. Le aree semantiche di maggiore importanza sono l'interrogazione delle conoscenze tramite quesiti, il feedback sugli errori, la personalizzazione dei quiz, le statistiche globali e il riepilogo a fine partita. I principali punti sono sotto riportati:

- selezione di N domande tra i C corsi scelti dell'utente
- gestione delle domande e delle relative risposte permettendo l'editing, il salvataggio e il caricamento delle stesse
- elaborare un riepilogo al termine di ogni partita
- elaborare e salvare alcune statistiche di interesse relativamente ad un utente

### 4.2 Requisiti Utente

Uno utente dell'applicazione dovrà quindi essere in grado di:

- ripassare: visualizzare domande e scegliere tra le possibili risposte
- scegliere tra varie modalità di gioco
- visualizzare un riepilogo a fine partita
- visualizzare le proprie statistiche su tutte le partite effettuate
- importare ed esportare nuovi quiz, con le relative domande e risposte

#### 4.2.1 User Stories

Si riportano di seguito le user stories emerse:

1. come utente voglio poter accedere all'applicativo e iniziare una nuova partita
2. come utente voglio poter selezionare la risposta che ritengo corretta, ricevendo un feedback a riguardo
3. come utente voglio poter riguardare i quiz della partita appena terminata
4. come utente voglio poter visualizzare le mie statistiche personali

5. come utente voglio poter selezionare i corsi prima dell'inizio di ogni partita
6. come utente voglio poter selezionare la modalità di gioco
7. come utente voglio poter cambiare le impostazioni di gioco
8. come utente voglio poter aggiungere dei quiz, qualora ne volessi integrare dei nuovi
9. come utente voglio poter modificare i quiz aggiunti, qualora non siano corretti
10. come utente voglio poter aggiungere dei corsi, qualora ne volessi integrare dei nuovi
11. come utente voglio poter modificare i corsi già presenti, qualora non siano corretti
12. come utente voglio poter importare i quiz in blocco relativi a uno o più corsi
13. come utente voglio poter esportare i quiz in blocco relativi a uno o più corsi

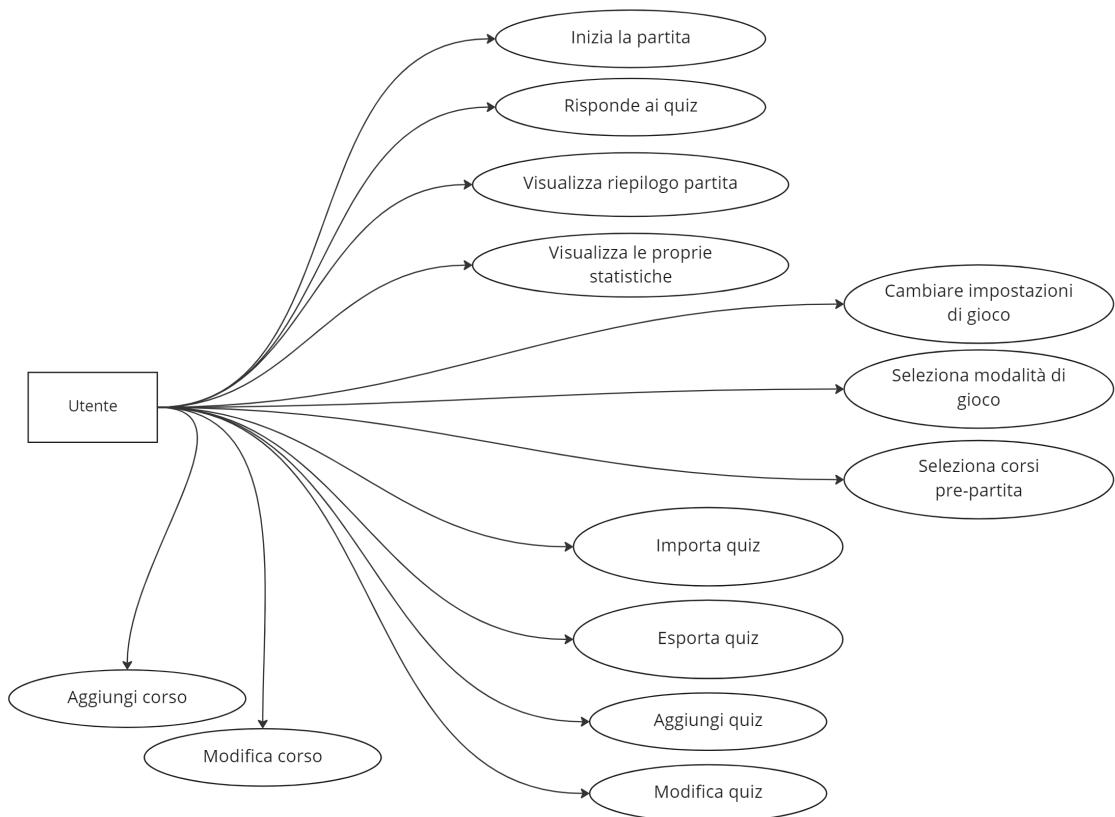


Figura 4.1: Schema Casi d'Uso

#### 4.2.2 Mockup

A seguito dell'intervista fatta con l'esperto del dominio e dei requisiti raccolti con esso, il team di sviluppo ha prodotto due versioni di mockup (4.2.2 e 4.2.2). In seguito, questi sono stati sottoposti al committente, il quale ha riscontrato aspetti positivi e negativi in entrambe le proposte. Di conseguenza, i mockup definitivi 4.2.2 sono stati sviluppati con la sua collaborazione e supervisione. Per garantire una maggior comprensione della navigazione all'interno dell'applicativo, è stato prodotto lo schema 4.18.

## Mockup prima versione



Figura 4.2: Pagina Iniziale



Figura 4.3: Settaggio di una partita prima di iniziare



Figura 4.4: Quiz

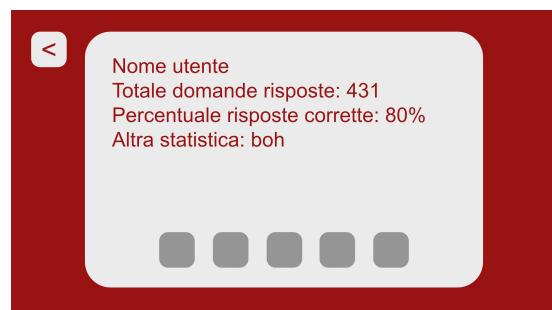


Figura 4.5: Statistiche di gioco e obiettivi raggiunti



Figura 4.6: Impostazioni Generali

## Mockup seconda versione



Figura 4.7: Pagina Iniziale

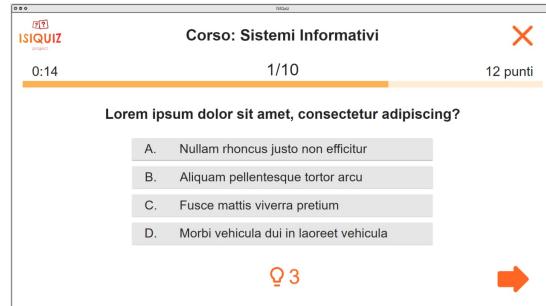


Figura 4.8: Quiz



Figura 4.9: Inserimento di nuove domande e relative risposte per un determinato corso

## Mockup definitivi



Figura 4.10: Pagina iniziale



Figura 4.11: Settaggio di una partita prima di iniziirla



Figura 4.12: Settaggio parametri per una partita personalizzata



Figura 4.13: Quiz



Figura 4.14: Riepilogo al termine di un quiz



Figura 4.15: Impostazioni generali

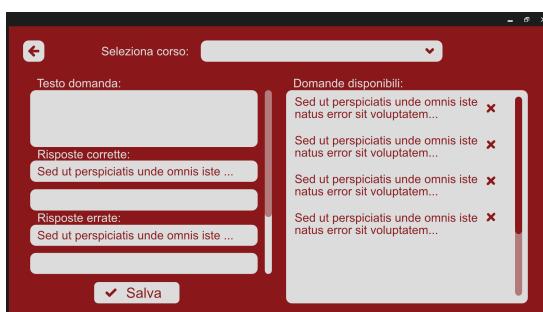


Figura 4.16: Inserimento di nuove domande e relative risposte per un determinato corso

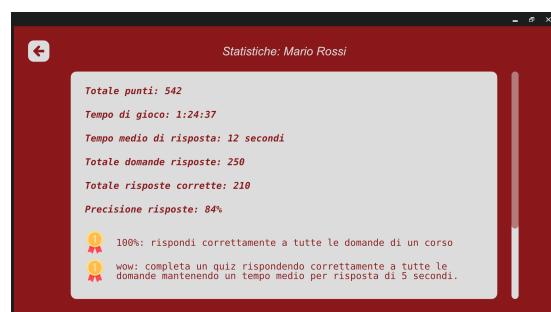


Figura 4.17: Statistiche di gioco e obiettivi raggiunti



Figura 4.18: Tutti i mockup dell'applicazione con navigazione tra le pagine

## 4.3 Requisiti Funzionali

### 4.3.1 Obbligatori

All'interno di ogni partita, vengono presentate all'utente diverse domande di natura accademica, ognuna delle quali viene accompagnata da un numero predeterminato di risposte possibili, alcune corrette ed altre sbagliate. L'utente dovrà individuare le risposte corrette in un tempo massimo per non fallire la domanda.

- 1. Modalità di gioco standard:** domande a scelta multipla su tutti i corsi disponibili: al giocatore verranno posti dieci quiz scelti casualmente tra tutti i corsi disponibili nel gioco. Ogni domanda dovrà essere risposta in massimo quindici secondi altrimenti verrà considerata errata.

2. **Modalità di gioco materie a scelta:** domande a scelta multipla su alcuni dei corsi scelti dal giocatore tra tutti i corsi disponibili: al giocatore verranno posti dieci quiz scelti casualmente tra tutti i corsi selezionati. Ogni domanda dovrà essere risposta in massimo quindici secondi altrimenti verrà considerata errata.
3. **Modalità di gioco materia specifica:** domande a scelta multipla su un corso scelto dal giocatore tra tutti i corsi disponibili: al giocatore verranno posti dieci quiz scelti casualmente tra quelli disponibili per il corso specificato. Ogni domanda dovrà essere risposta in massimo quindici secondi altrimenti verrà considerata errata.
4. **Modalità di gioco personalizzata:** domande a scelta multipla su alcuni dei corsi scelti dal giocatore tra tutti i corsi disponibili: prima dell'inizio della partita il giocatore può decidere a quanti quiz rispondere (N) e il tempo massimo (T) in cui rispondere a ciascun quiz. Al giocatore verranno quindi posti N quiz scelti casualmente tra tutti i corsi selezionati. Ogni domanda dovrà essere risposta in T tempo massimo altrimenti verrà considerata errata.
5. **Interfaccia grafica** (CLI e successivamente ScalaFX): visualizzazione menu principale, visualizzazione della domanda e di quattro possibili risposte da scegliere durante il gioco, visualizzazione dei risultati post partita. La prima iterazione del programma dovrà fornire una semplice interfaccia da terminale che permetta di interagire con le funzionalità principali del gioco scrivendo le operazioni da eseguire tramite linea di comando. Successivamente verrà sostituita da una GUI.
  - 5.1 Menu principale: la prima pagina che verrà mostrata all'avvio
    - 5.1.1 Pulsante "inizia a giocare" > pagina selezione corsi
    - 5.1.2 Pulsante "statistiche" > pagina statistiche
    - 5.1.3 Pulsante "impostazioni" > pagina impostazioni
    - 5.1.4 Pulsante "esci" > termina l'applicazione
  - 5.2 Pagina statistiche: contiene informazioni sulle statistiche relative alle partite concluse
    - 5.2.1 Totale quiz risposti
    - 5.2.2 Totale quiz risposti correttamente
    - 5.2.3 Percentuale risposte corrette
  - 5.3 Pagina impostazioni: permette di editare i quiz, cioè le domande e le risposte, oltre a salvarli ed importarli da file
    - 5.3.1 Modulo per editare i quiz
    - 5.3.2 Pulsante per salvare (esportare) nuovi quiz > salva su file tutti i quiz dei corsi
    - 5.3.3 Pulsante per importare nuovi quiz > permette di selezionare un file contenente i corsi con i relativi quiz
  - 5.4 Pagina selezione corsi: permette di modificare alcune impostazioni di gioco e di selezionare i corsi dai quali devono essere presi i quiz prima di iniziare la partita
    - 5.4.1 Selezione corsi disponibili
    - 5.4.2 Personalizza tempo per risposta
    - 5.4.3 Personalizza numero domande della partita
  - 5.5 Pagina riepilogo post partita: permette di visualizzare il punteggio ottenuto nella partita appena conclusa e la lista di tutti i quiz con un'indicazione sulla correttezza della risposta data e, in caso di risposta errata, la risposta corretta corrispondente
    - 5.5.1 Testo con punteggio della partita
    - 5.5.2 Lista quiz della partita con risposta scelta ed eventualmente risposta corretta
6. **Punteggio finale** del quiz appena effettuato: al termine della partita verrà visualizzato il punteggio ottenuto dal giocatore in base al numero di risposte corrette o errate date nella partita conclusa. Il punteggio totale della partita verrà calcolato sommando i punti relativi ad ogni quiz risposto correttamente. Nella modalità di gioco standard, quella in cui ogni quiz ha un tempo massimo di risposta, il punteggio associato ad ognuno viene diminuito man mano che passa il tempo a disposizione.

7. **Visualizzazione e salvataggio delle statistiche personali:** in specifico le statistiche personali includono:
  - 7.1 Totale dei quiz risposti
  - 7.2 Totale risposte corrette
  - 7.3 Percentuale di correttezza delle risposte
  - 7.4 Punteggio totale
8. **Più risposte giuste** e sbagliate per ogni domanda in modo da avere una rotazione tra le possibili scelte: ogni domanda dovrà necessariamente avere almeno tre risposte errate ed una corretta, ma potrebbe averne a disposizione un numero superiore, da scegliere poi casualmente durante il gioco. In ogni caso, dovrà essere rispettato il vincolo di avere tre risposte errate ed una corretta per ogni quiz durante la partita.
9. Aggiunta di **nuovi quiz** con le relative domanda e risposte da parte dell'utente:
  - 9.1 Aggiunta di un nuovo corso
  - 9.2 Aggiunta di un nuovo quiz
  - 9.3 Aggiunta di una domanda relativa al nuovo quiz
  - 9.4 Aggiunta di tre o più risposte errate relative ad un quiz
  - 9.5 Aggiunta di una o più risposta corretta relativa ad un quiz
  - 9.6 Aggiunta di un punteggio per un quiz

#### 4.3.2 Opzionali

1. **Sfida a tempo:** rispondere a più domande possibili in un intervallo di tempo limitato
2. **Aiuti su richiesta:** all'utente vengono messi a disposizione un numero specifico di aiuti ad ogni partita con i quali semplificare la scelta della risposta. Utilizzando uno degli aiuti verrà eliminata una tra le risposte errate, semplificando così la scelta. Potranno essere utilizzati al massimo due aiuti per ogni domanda
3. **Revisione solo delle risposte sbagliate:** l'utente può scegliere una modalità di gioco in cui ripassare esclusivamente le domande precedentemente sbagliate
4. **Medaglie-achievement** quando si completano delle sfide (es. più punti di un certo limite, più giornate continuativamente, aver risposto bene a tutte le domande di una materia)
5. Salvataggio delle statistiche relative al tempo medio di risposta e tempo totale di gioco

### 4.4 Requisiti Non Funzionali

- Interfaccia grafica intuitiva e che fornisca feedback coerenti all'utente per comunicargli lo stato delle sue azioni (ad esempio, indicando con colore verde una risposta qualora essa sia corretta)
- Interfaccia grafica accessibile agli utenti daltonici
- Interfaccia grafica reattiva: alle azioni di un utente devono corrispondere degli aggiornamenti dell'interfaccia stessa in tempi ragionevoli (al massimo un secondo) per non rovinare la user experience
- Realizzare un sistema modulare che permetta estensioni future (ad esempio, uno sviluppo distribuito che permetta di realizzare funzionalità di gioco multiplayer)
- Il sistema dovrà essere in grado di memorizzare dati di interesse in maniera consistente

- Il sistema deve essere funzionante in diversi sistemi operativi nei quali è installata una appropriata Java Virtual Machine (il requisito minimo considera il funzionamento su sistemi Windows, Linux e MacOS)
- Il sistema deve essere fault-tolerant, cioè deve continuare a funzionare in modo affidabile e deve essere privo di fallo che permettano al giocatore di barare

## 4.5 Requisiti Implementativi

- Linguaggio di programmazione Scala versione 3.2.0
- Libreria [JavaFX](#) e il suo wrapper scritto in Scala chiamato [ScalaFX](#) per l'interfaccia grafica
- Utilizzo di JDK  $\geq 11$
- Libreria [Play JSON](#) 2.10.0-RC7 per il salvataggio dei quiz e delle statistiche attraverso la notazione JSON.
- Per lo sviluppo dovrà essere utilizzato l'IDE IntelliJ in modo da avere consistenza tra i membri del team di sviluppo.

# Capitolo 5

# Design Architetturale

In questo capitolo verrà analizzata l'architettura del sistema ad alto livello cercando di spiegare le scelte effettuate in relazione ai requisiti che sono stati elaborati precedentemente. Lo scopo è quello di esaminare i componenti principali che formano l'applicazione e di come essi comunichino tra loro.

## 5.1 Architettura Generale

L'architettura generale del sistema si basa sul pattern **Model View Controller** (MVC) per rendere il più possibile modulari e separati i compiti svolti da ciascun componente. In particolare, è stato scelto di separare ulteriormente i ruoli di *view* e *controller* creando delle coppie di essi per ciascuna pagina dell'applicazione. In questo modo il comportamento di ciascuna pagina risulta completamente separato dalle altre, così permettendo possibili modifiche o funzionalità non previste durante la fase di progettazione. Questo è stato gestito attraverso un "AppController", che contiene al suo interno la coppia di *controller* e *view* della pagina attualmente caricata e mette a disposizione il necessario per navigare da una pagina all'altra.

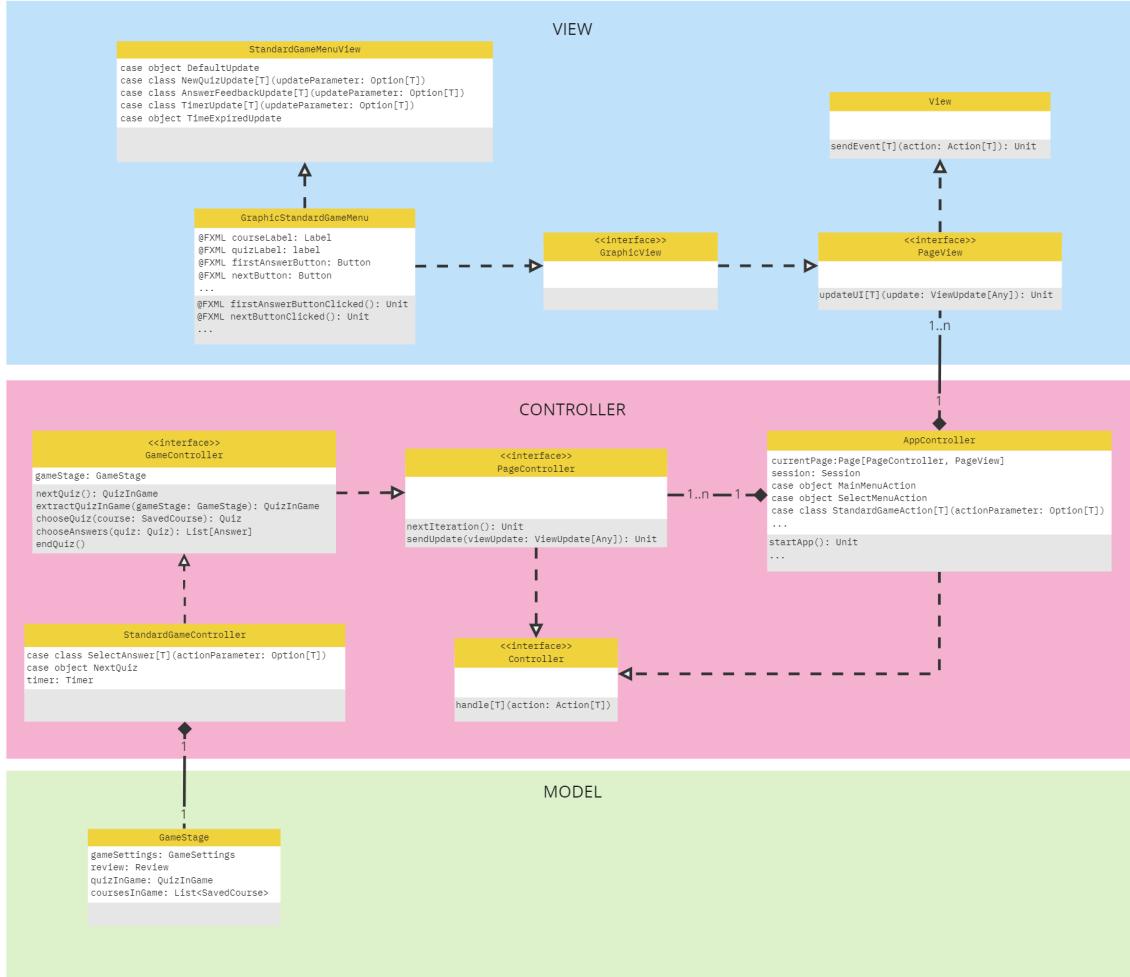


Figura 5.1: Architettura generale dell'applicazione

### 5.1.1 View

Nell'architettura realizzata, il componente "View" si occupa della comunicazione con il *controller* inviando ad esso le azioni che un utente effettua sull'interfaccia grafica. Ciascuna pagina dell'applicazione deve estendere l'interfaccia generale "PageView", la quale definisce un metodo per l'aggiornamento della pagina stessa a seguito della ricezione di aggiornamenti da parte del *controller* della pagina corrispondente. "GraphicView" estende una "PageView" aggiungendo tutti i comportamenti comuni relativi ad una GUI. Essa viene a sua volta estesa da ciascuna implementazione delle interfacce grafiche di ciascuna pagina (nella Figura 5.1 è riportato come esempio solamente l'implementazione della *view* relativa ad una partita standard). Avendo scelto di utilizzare JavaFX per realizzare l'interfaccia grafica dell'applicazione, tali classi di implementazione assumono il ruolo di FXControllers delle GUI. Ogni implementazione di una interfaccia grafica di una pagina (ad esempio "GraphicStandardGameMenu") deve anche estendere una *view* specifica della pagina ("StandardGameMenuView"), la quale contiene tutti i possibili aggiornamenti su una generica interfaccia grafica relativa a tale pagina. In questo modo, una specifica implementazione deve sia soddisfare i vincoli imposti da una generica tipologia di interfaccia grafica sia gestire gli aggiornamenti specifici da effettuare su quella pagina.

### 5.1.2 Controller

Come precedentemente accennato, l'unione e la comunicazione tra la *view* e il *controller* di una specifica pagina viene gestita tramite "AppController", il quale mantiene il riferimento ad essi.

"AppController" permette la navigazione tra le pagine andando a modificare la pagina attualmente in uso. Ogni *controller* di una pagina deve estendere l'interfaccia "PageController", la quale impone l'esecuzione di una iterazione successiva e l'invio di update all'interfaccia grafica. L'interfaccia "Controller" specifica una generica funzione di gestione delle azioni eseguite da un utente. Una "PageView" compone, insieme ad un "PageController", una pagina completamente funzionante dell'applicazione e la comunicazione tra *view* e *controller* avviene unicamente attraverso l'invio di azioni e di update. Nel caso della pagina relativa ad una partita, come mostrato in Figura 5.1, i comportamenti generali di qualsiasi modalità di partita sono stati astratti in una interfaccia "GameController". Ciascuna modalità di gioco estende "GameController", implementando poi la propria logica specifica e gestendo le possibili azioni di gioco definite nel *controller* di tale modalità.

### 5.1.3 Model

Il *model* dell'applicazione è utilizzato quasi completamente per modellare gli oggetti del dominio necessari allo svolgimento di una partita. Come in Figura 5.1, esso viene sintetizzato nel "GameStage", il quale modella tutti gli elementi generali presenti in una partita, ad esempio, le impostazioni di gioco, il riepilogo della partita e i corsi utilizzati durante la partita per estrarre i quiz da porre. "GameStage" viene utilizzato e modificato dal *controller* della modalità di gioco secondo la logica implementata e poi viene comunicato alla *view* per effettuare gli aggiornamenti necessari sull'interfaccia grafica. Come già detto, nell'architettura generale è stata presentata una versione molto riassuntiva del *model*, il quale verrà successivamente dettagliato nella parte di Design di Dettaglio 6.

# Capitolo 6

## Design di Dettaglio

In questo capitolo, partendo dal design architetturale sopra descritto, verrà analizzata in dettaglio la struttura del sistema, descrivendo le caratteristiche dei componenti e la relazione che c'è tra loro.

### 6.1 Model

Il design del *model* realizzato cerca di rispecchiare il dominio di questa applicazione in modo indipendente e generale rispetto all'implementazione che verrà eseguita successivamente. Il design dettagliato sarà di seguito esposto dividendolo in tre parti concettualmente separate. Essi riguardano rispettivamente gli elementi utilizzati durante una partita, quelli per il riepilogo e infine quelli relativi alla rappresentazione delle statistiche.

L'elemento principale del *model* utilizzato durante una partita è "GameStage", come già indicato in precedenza. Esso è contenuto nel *controller*, ma a sua volta contiene tutti i componenti necessari per lo svolgimento di una generica partita:

- "GameSettings": rappresenta le impostazioni della partita e a sua volta si specializza per modellare le specifiche impostazioni relative ad una determinata modalità di gioco
- "Review": mantiene le informazioni riguardanti lo svolgimento di una partita (verrà successivamente analizzato)
- "QuizInGame": modella il "Quiz", relativo ad uno specifico "Course", che viene proposto ad un utente in un particolare momento di gioco, comprensivo di una domanda e di alcune possibili "Answer"
- "CoursesInGame": rappresenta i "Course" selezionati tra i possibili "SavedCourse" da utilizzare per estrarre le domande da porre durante la partita

Come mostrato in Figura 6.1, gli elementi base modellati nell'applicazione sono:

- "Course": descrive un corso universitario e presenta un identificativo, "CourseIdentifier", composto dal nome del corso, il nome dell'università in cui si svolge e la facoltà relativa. "Course" si specializza in "SavedCourse" fungendo da contenitore per un insieme di quiz salvati relativi al corso.
- "Quiz": sono composti dal testo della domanda e da un insieme di possibili risposte associate. Sono contenuti in uno dei "SavedCourse" e contengono anche informazioni aggiuntive come, ad esempio, il punteggio del quiz.
- "Answer": modella una delle possibili risposte di un quiz ed è formata dal testo della risposta e da un'indicazione della correttezza di tale risposta.

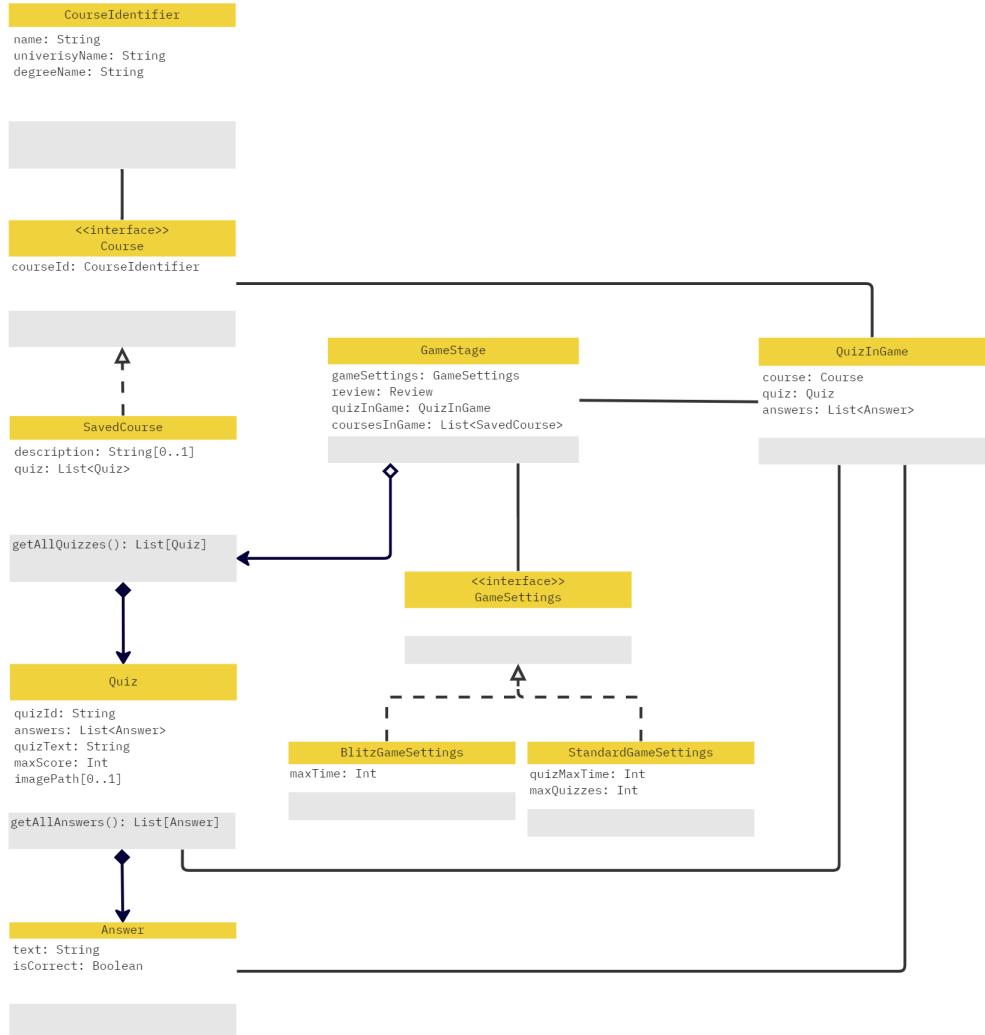


Figura 6.1: Architettura del *model* durante una partita

Come mostrato in Figura 6.2, il riepilogo di una partita è contenuta nel "GameStage" e viene aggiornato durante la partita aggiungendo una serie di "QuizAnswered", ovvero i quiz che vengono posti all'utente durante lo svolgimento del gioco. "QuizAnswered" estende il "QuizInGame" con l'aggiunta della risposta scelta, il punteggio ottenuto e il tempo di risposta.

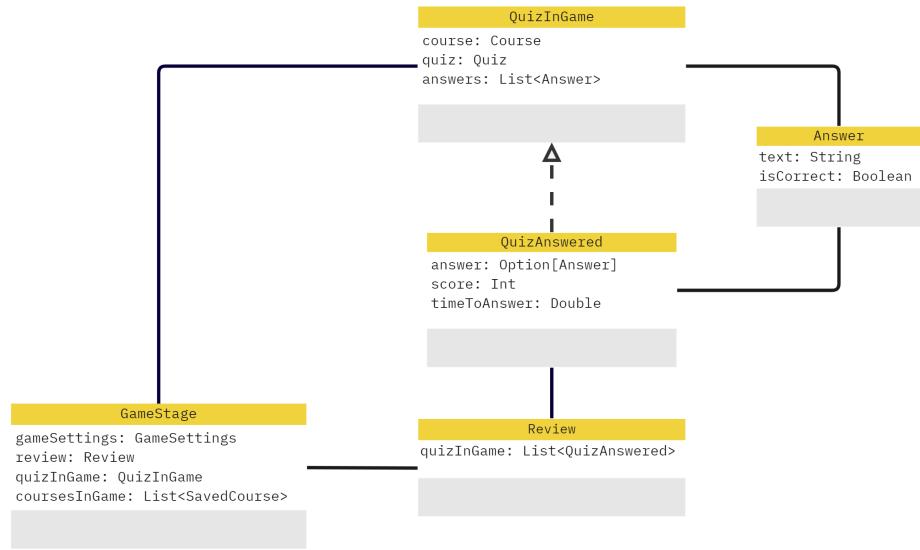


Figura 6.2: Architettura del *model* per il riepilogo

Per salvare i dati relativi alle statistiche di diverse partite, è stata realizzata una "Session", la quale viene aggiornata al termine di ogni partita con le nuove statistiche di gioco. Come si può vedere in Figura 6.3, le statistiche mantengono una struttura dati su file simile a quella utilizzata per rappresentare i quiz e sono infatti modellate attraverso un "CourseInStat" con all'interno una serie di "QuizInStat". In questo modo viene mantenuta la struttura gerarchica tra quiz e corso, permettendo di ottenere eventualmente informazioni utili relative alle statistiche dei singoli quiz di un corso.

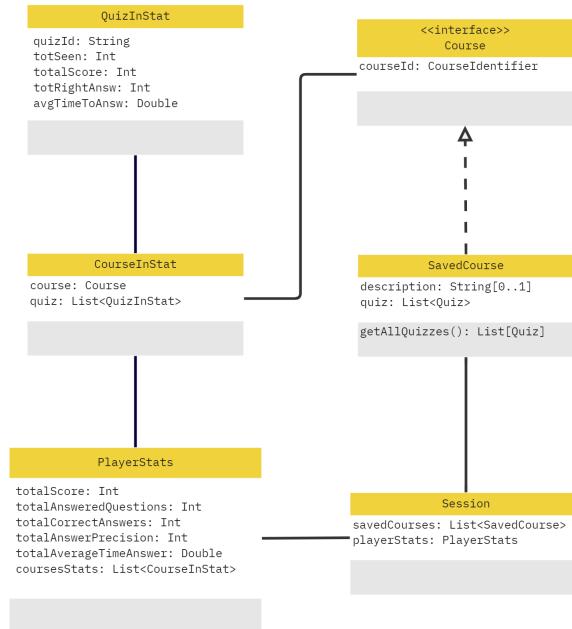


Figura 6.3: Architettura del *model* per le statistiche

## 6.2 View

Come già spiegato nell'Architettura Generale 5.1 del sistema, ogni pagina dispone di una *view* specifica. In Figura 6.4 sono mostrate come esempio le *view* relative al menu di selezione dei corsi prima di una partita, "SelectMenuView", e quella di una partita standard, "StandardGameMenuView". Avendo utilizzato un processo di sviluppo agile, è stata inizialmente proposta all'esperto del dominio una versione dell'applicazione con interfaccia grafica tramite linea di comando. Tuttavia, il design della *view* realizzato, è stato ideato in modo da rendere la *view* quanto più possibile indipendente dalla specifica implementazione. Infatti nelle fasi successive del progetto, quando è stata introdotta la GUI tramite JavaFX, è stato sufficiente realizzare la nuova implementazione rispettando le interfacce predisposte. Durante l'esecuzione dell'applicazione, la corretta implementazione della *view* relativa ad una pagina viene instanziata attraverso una "ViewFactory" che restituisce la "PageView" corretta.

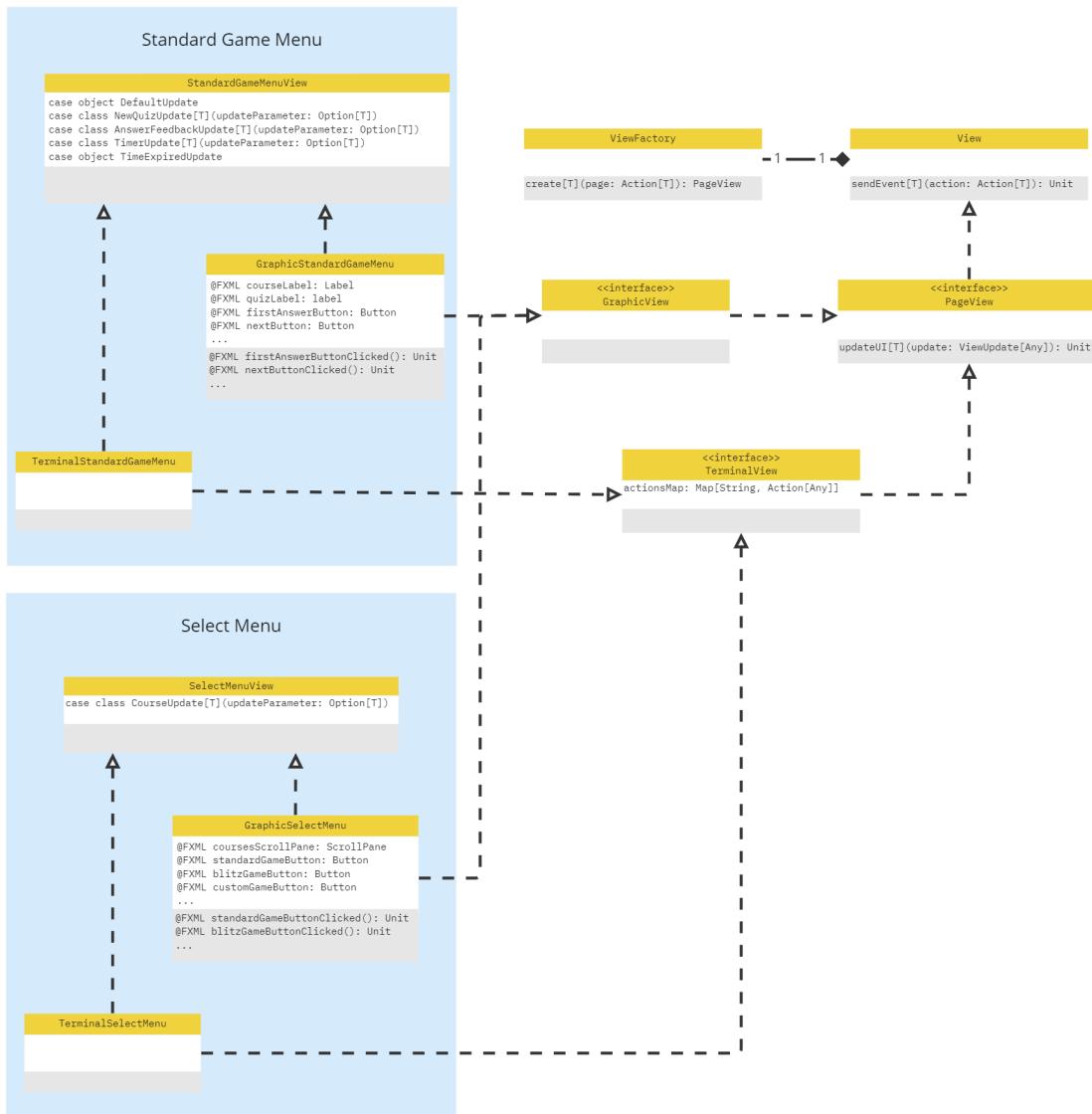


Figura 6.4: Architettura della *view*

Ciascuna *view* di una pagina mette a disposizione diversi "ViewUpdate" che è possibile richiamare per aggiornare alcuni degli elementi grafici. Gli updates della *view* possono avere dei parametri generici e opzionali il cui valore costituisce le informazioni necessarie per effettuare

l'aggiornamento (Figura 6.5). Nel caso uno degli update non necessiti di dati, è possibile utilizzare un "ParameterlessViewUpdate" il quale estende "ViewUpdate", definendo il parametro dell'aggiornamento come vuoto. Dato che tutte le interfacce grafiche delle pagine necessitano di un "DefaultUpdate", ovvero di un aggiornamento di base dell'interfaccia, esso è stato estratto in modo da evitare di definirlo molteplici volte.

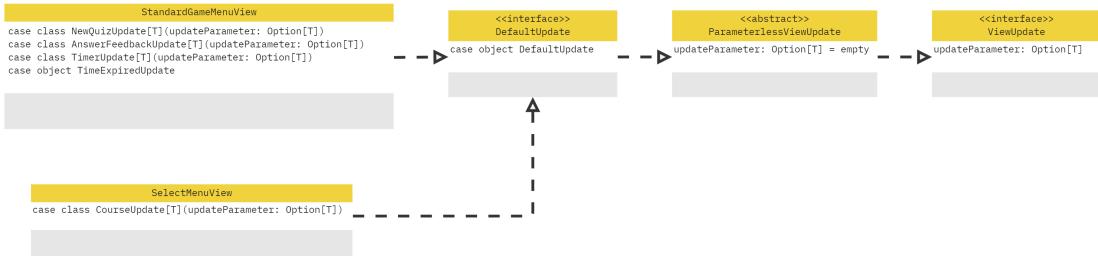


Figura 6.5: Updates della view

### 6.3 Controller

Per quanto riguarda il *controller*, sono stati definiti molti controller, uno per ciascuna pagina analogamente alla *view*. Come infatti si può notare da Figura 6.6, sono presenti un "SelectMenuController" e "StandardGameController" relativi rispettivamente alle pagine di selezione corsi e di una partita in modalità standard. Per come è stato concepito, il *controller* è completamente indipendente dall'implementazione specifica dell'interfaccia grafica, come già indicato nel capitolo Architettura Generale 5.1.2.

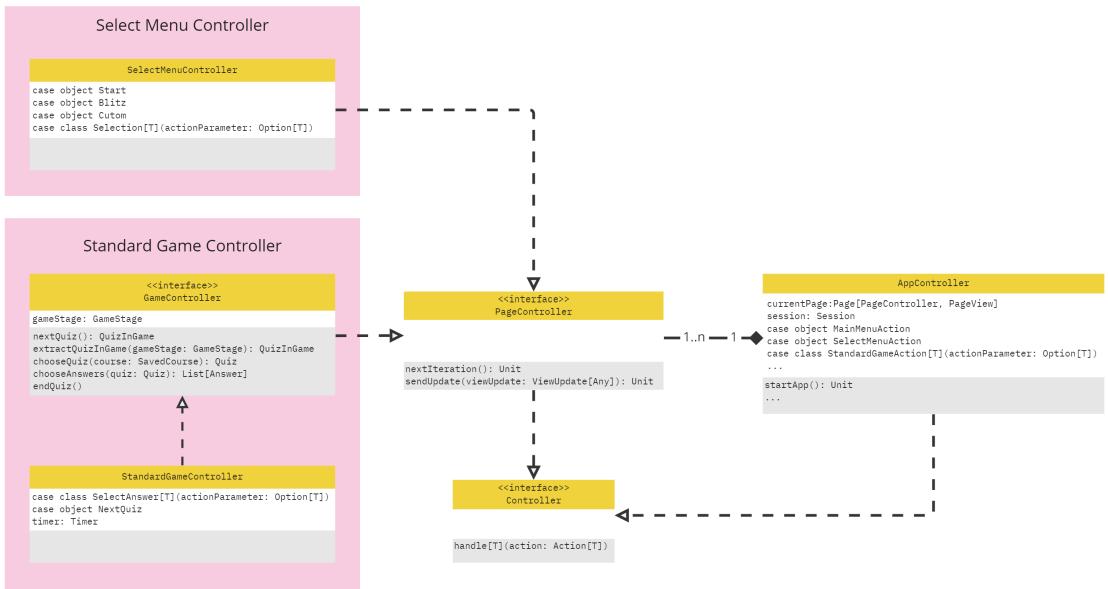


Figura 6.6: Architettura del controller

Come per la *view*, il *controller* mette a disposizione una serie di azioni che l'utente può eseguire e che vengono gestite dal *controller* della pagina. Una "Action" generale può portare con sé dei parametri generici ed opzionali o averli vuoti nel caso si tratti di una "ParameterLessAction", ovvero di una azione senza parametri aggiuntivi. Anche in questo caso le parti in comune tra i vari *controller* sono state estratte così come nell'esempio di Figura 6.7, dove è mostrata la "BackAction" che rappresenta un'azione generica per tornare alla pagina precedente.

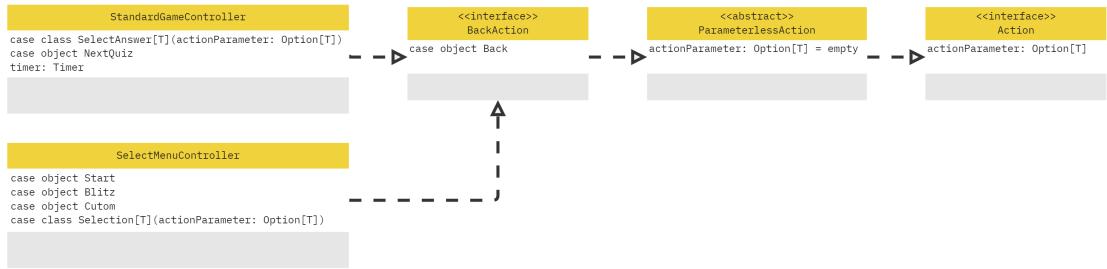


Figura 6.7: Actions del controller

Come già specificato, la comunicazione tra il *view* e il *controller* di una pagina avviene solamente attraverso le "Action" e i "ViewUpdate", nello specifico: la *view* invia al controller le azioni eseguite da un utente attraverso le "Action" specifiche definite e gestite da quel *controller*, mentre il *controller* manda alla *view* gli update che l'interfaccia grafica è in grado di visualizzare.

# Capitolo 7

## Implementazione

Nell'implementazione non ci sono aspetti particolari da segnalare in quanto lo sviluppo del codice ha seguito piuttosto fedelmente quanto deciso nel design del sistema. Si riportano quindi, per ogni membro del team, i principali componenti implementati ed i ruoli assunti durante il progetto.

### 7.1 Gambaletta Daniele

All'interno del progetto ho ricoperto il ruolo di **domain expert**, quindi sono stato intervistato all'inizio per capire i requisiti dell'applicativo, ho controllato i *mockup* e li ho fatti correggere in base ai miei gusti personali, controllando che venissero rispettate le mie indicazioni durante lo sviluppo. Per quanto riguarda l'implementazione, nel *model* mi sono occupato principalmente delle statistiche e di parte dei corsi. Inoltre ho sviluppato le *utils* necessarie alla persistenza dei quiz e delle statistiche, gestendo sia la parte di *parsing* dei dati in formato JSON che quella di salvataggio/caricamento dei *files*.

Model:

- PlayerStats
- CourseInStats
- QuizInStats - in collaborazione con Teodorani
- SavedCourse - in collaborazione con Teodorani
- Course
- CourseIdentifier - in collaborazione con Teodorani
- Quiz - in collaborazione con Teodorani
- GameStage - in collaborazione con Omiccioli, Lirussi e Teodorani
- Session - in collaborazione con Teodorani

View:

- Settings Menu View - in collaborazione con Teodorani, Lirussi
- Statistics Menu View - in collaborazione con Teodorani

FXML:

- Statistics

Controller:

- AppController - in collaborazione con Omiccioli

- StandardGameController - in collaborazione con Omiccioli, Lirussi e Teodorani
- BlitzGameController - in collaborazione con Omiccioli, Lirussi
- StatisticsMenuController - in collaborazione con Omiccioli

Utils:

- JsonParser con CourseJsonParser e StatsJsonParser
- FileHandler
- DataStorageHandler
- ImportHandler
- ExportHandler

## 7.2 Lirussi Igor

Per la durata del progetto, ricoprendo il ruolo aggiuntivo di **scrum-master**, ho svolto il compito di gestire il processo di sviluppo (capitolo 2) dal punto di vista della metodologia Agile, rispettando le sue varie fasi e utilizzando il più possibile il framework Scrum. Ricoprendo questo ruolo, la responsabilità è stata quella di portare il focus dello sviluppo sul processo, più che sul risultato. Mi sono occupato di predisporre l'organizzazione con le relative repository e i tool da utilizzare, collegando ad essi la Continuous Integration (capitolo 2.4) della reportistica e del software, affinché i cambiamenti vengano versionati e ci sia *accountability* su di essi. Inoltre, è stato aggiunto il setup della Quality Assurance (capitolo 2.3) con test, documentazione e coverage, garantendo così la massima attenzione sul metodo di sviluppo. Infine, ad essa sono stati affiancati i bot, i siti generati ed i relativi artefatti. Durante l'implementazione del software, ho organizzato le fasi di sviluppo affinché utilizzassimo correttamente i principi scelti, mantenendo il product backlog (capitolo 2.3), gli sprint e i task con i relativi branch (capitolo 2.3).

Riguardo alla parte di implementazione, il mio focus è stato sull'aggiunta di quiz (Requisiti 4.3.1.8), di corsi (Requisiti 4.3.1.10) e la modifica di quiz (Requisiti 4.3.1.9) e di corsi (Requisiti 4.3.1.11). Questo mi ha portato ad occuparmi nel model dell'implementazione di quiz con domande, e in *controller* e *view* della relativa logica e visualizzazione. Un'altra area di particolar focus è stata il riepilogo finale della partita, come da (Requisiti 4.3.1.3). Nel *model* mi sono occupato del Game Stage e della Review in esso contenuta, per utilizzarli nei *controller* relativi, i quali comunicano con le interfacce scelte (grafiche e da linea di comando).

Controller:

- Add course controller - in collaborazione con Omiccioli
- Add quiz controller - in collaborazione con Omiccioli
- Edit course controller - in collaborazione con Omiccioli
- Edit quiz controller - in collaborazione con Omiccioli
- Review controller - in collaborazione con Omiccioli

Model:

- Quiz - in collaborazione con Teodorani e Gambaletta
- Answer - in collaborazione con Teodorani
- Game Stage - in collaborazione con Omiccioli, Teodorani e Gambaletta
- Review

Utils:

- Timer - in collaborazione con Omiccioli

View:

- Add course view - in collaborazione con Teodorani
- Add quiz view - in collaborazione con Teodorani
- Edit course view - in collaborazione con Teodorani
- Edit quiz view - in collaborazione con Teodorani
- Review view
- Settings Menu View - in collaborazione con Gambaletta e Teodorani

FXML:

- Add course menu - in collaborazione con Omiccioli
- Add quiz menu - in collaborazione con Omiccioli
- Edit course menu - in collaborazione con Omiccioli
- Edit quiz menu - in collaborazione con Omiccioli
- Review menu - in collaborazione con Omiccioli e Teodorani

### 7.3 Omiccioli Riccardo

In qualità di **product owner** del progetto ho: presieduto i vari meeting svolti dal team di sviluppo, identificato i requisiti principali (raffinati poi con il team) dell'applicativo richiesto e organizzato il lavoro del team. Il mio lavoro si è principalmente concentrato nel design e realizzazione dell'architettura generale dell'applicazione implementando, in particolare, la parte relativa ai *controller*, alle *view* e alla comunicazione tra essi.

FXML:

- Add course menu - in collaborazione con Lirussi
- Add quiz menu - in collaborazione con Lirussi
- Blitz game
- Custom menu
- Default menu
- Edit course menu - in collaborazione con Lirussi
- Edit quiz menu - in collaborazione con Lirussi
- Main menu
- Review menu - in collaborazione con Lirussi
- Select menu - in collaborazione con Teodorani
- Standard game - in collaborazione con Teodorani

Controller:

- Package actions
- App controller
- Blitz game controller

- Controller
- Custom menu controller
- Game controller - in collaborazione con Teodorani
- Main menu controller
- Standard game controller - in collaborazione con Teodorani
- Select menu controller - in collaborazione con Teodorani

Model:

- Game settings
- Game stage - in collaborazione con Gambaletta, Lirussi e Teodorani
- Timer - in collaborazione con Lirussi

View:

- Blitz game menu
- Blitz view
- Custom menu
- Default menu
- Main menu - in collaborazione con Teodorani
- Main menu view - in collaborazione con Teodorani
- Select menu - in collaborazione con Teodorani
- Select menu view - in collaborazione con Teodorani
- Standard game menu
- Package updates
- View - in collaborazione con Teodorani

Altro:

- Main

## 7.4 Teodorani Cecilia

Insieme ad Omiccioli, ho seguito lo sviluppo generale dell'architettura, realizzando alcuni *controller*, *view* e componenti del *model*. Di quest'ultimo mi sono anche occupata di standardizzare tutto il codice al termine dell'implementazione del progetto. In più, mi sono assicurata che il *model* fosse adeguatamente testato, sviluppando test molto dettagliati. Infine, per quanto riguarda la GUI, ho messo a punto tecniche per la user experience e l'accessibilità. Per la prima ho adottato tecniche standard, quali ad esempio il font e la disposizione dei vari componenti nella pagina, mentre per la seconda ho utilizzato dei tool, per garantire che soddisfassero le *WCAG* (*Web Content Accessibility Guidelines*). In particolare, per il colore primario del sito e il colore del testo sovrastante, si è utilizzato [Material Design - Color Tool](#) e [WebAIM](#).

FXML:

- Review Menu - in collaborazione con Lirussi
- Select Menu - in collaborazione con Omiccioli

- Standard Game - in collaborazione con Omiccioli

Controller:

- Select Menu Controller - in collaborazione con Omiccioli
- Game Controller - in collaborazione con Omiccioli
- Standard Game Controller - in collaborazione con Gambaletta, Omiccioli e Lirussi

Model:

- Answer - in collaborazione con Lirussi
- Game Stage - in collaborazione con Gambaletta, Lirussi e Omiccioli
- Course Identifier - in collaborazione con Gambaletta
- Quiz - in collaborazione con Gambaletta, Lirussi
- SavedCourse - in collaborazione con Gambaletta
- Session - in collaborazione con Gambaletta
- QuizInStats - in collaborazione con Gambaletta

View (realizzata la struttura di tutti gli update in *nomeFileMenuView* sotto riportati, mentre dei controller grafici *nomeFileMenu* la struttura generale e la navigazione tra le varie pagine):

- Add Course Menu - in collaborazione con Lirussi
- Add Course Menu View - in collaborazione con Lirussi
- Add Quiz Menu - in collaborazione con Lirussi
- Add Quiz Menu - in collaborazione con Lirussi
- Custom Menu
- Custom Menu View - in collaborazione con Omiccioli
- Main menu - in collaborazione con Omiccioli
- Main Menu View - in collaborazione con Omiccioli
- Select Menu - in collaborazione con Omiccioli
- Select Menu View - in collaborazione con Omiccioli
- Settings Menu View - in collaborazione con Gambaletta, Lirussi
- Standard Game Menu - in collaborazione con Omiccioli
- Statistics Menu View - in collaborazione con Gambaletta
- View - in collaborazione con Omiccioli

# Capitolo 8

## Testing e Performance

### 8.1 Testing

Per i test dell'applicazione sono stati utilizzati:

- [Scala Test](#) versione 3.2.14 per il *model* dell'applicazione
- [JUnit 5](#) e [TestFX](#) versione 4.0.16-alpha per l'interfaccia grafica

Come già detto nel capitolo 7 relativo all'implementazione, si è prestata particolare attenzione nell'eseguire test dettagliati sui componenti del *model*, assicurando quindi che gli elementi del dominio rappresentati all'interno del programma fossero sufficientemente robusti. Nella figura 8.1 è possibile osservare la coverage del *model*, realizzata attraverso il tool [Scoverage](#), consultabile anche nella [documentazione online](#) del progetto. Per quanto riguarda la *view*, sono stati realizzati test utilizzando TestFX per assicurarsi che le interfacce grafiche rispettassero i vincoli imposti dal domain expert. Per la *view* non è disponibile la coverage relativa ai test a causa della difficile integrazione tra Scoverage e la suite di testing TestFX, tuttavia, l'applicazione è stata testata da almeno cinque persone esterne al team di sviluppo. Esse hanno utilizzato diverse versioni dei più diffusi e moderni sistemi operativi includendo Windows, Linux e MacOS e hanno sottolineato una buona qualità dell'interfaccia grafica soprattutto per quanto riguarda l'usabilità di essa.

Class	Source file	Lines	Methods	Statements	Invoked	Coverage	Branches	Invoked	Coverage
Answer\$	Answer.scala	45	5	6	6	<div style="width: 100.00%;">100.00%</div>	0	0	<div style="width: 100.00%;">100.00%</div>
Course\$	Course.scala	24	1	1	1	<div style="width: 100.00%;">100.00%</div>	0	0	<div style="width: 100.00%;">100.00%</div>
CourseIdentifier\$	CourseIdentifier.scala	38	3	3	3	<div style="width: 100.00%;">100.00%</div>	0	0	<div style="width: 100.00%;">100.00%</div>
GameStage	GameStage.scala	55	7	23	17	<div style="width: 73.91%;">73.91%</div>	2	1	<div style="width: 50.00%;">50.00%</div>
GameStage\$	GameStage.scala	28	5	10	10	<div style="width: 100.00%;">100.00%</div>	0	0	<div style="width: 100.00%;">100.00%</div>
Quiz	Quiz.scala	22	1	2	0	<div style="width: 0.00%;">0.00%</div>	0	0	<div style="width: 0.00%;">0.00%</div>
Quiz\$	Quiz.scala	71	8	19	9	<div style="width: 47.37%;">47.37%</div>	0	0	<div style="width: 100.00%;">100.00%</div>
QuizAnswered\$	Review.scala	6	2	2	2	<div style="width: 100.00%;">100.00%</div>	0	0	<div style="width: 100.00%;">100.00%</div>
Review	Review.scala	19	5	14	4	<div style="width: 28.57%;">28.57%</div>	2	0	<div style="width: 0.00%;">0.00%</div>
Review\$	Review.scala	9	1	2	2	<div style="width: 100.00%;">100.00%</div>	0	0	<div style="width: 100.00%;">100.00%</div>
SavedCourse\$	SavedCourse.scala	41	3	3	3	<div style="width: 100.00%;">100.00%</div>	0	0	<div style="width: 100.00%;">100.00%</div>
Session\$	Session.scala	35	4	5	5	<div style="width: 100.00%;">100.00%</div>	0	0	<div style="width: 100.00%;">100.00%</div>

Figura 8.1: Coverage del *model*

### 8.2 Performance

Per quanto riguarda le performance, il programma è stato eseguito con successo su diversi dispositivi e sistemi operativi (nello specifico Windows, Linux e MacOS). Si riportano in Figura 8.2 i risultati dell'esecuzione di una partita standard osservata attraverso il tool [VisualVM](#).

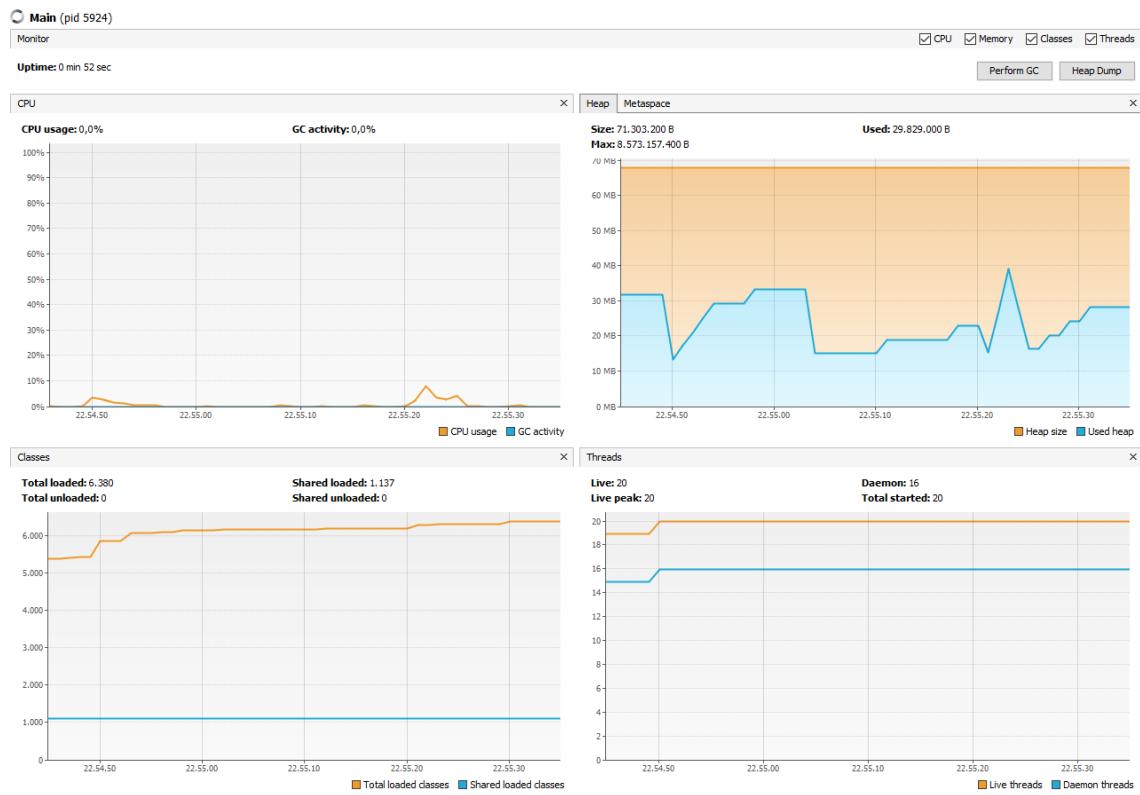


Figura 8.2: Monitoraggio esecuzione di una partita

# Capitolo 9

## Retrospettiva

Questo capitolo contiene una breve retrospettiva, con i soli deliverables relativi ad ogni sprint. Per una discussione più dettagliata è possibile consultare la [Relazione di Processo di Sviluppo](#).

### 9.1 Svolgimento

Gli sprint sono stati portati avanti nel seguente modo:

**Sprint Planning** Pianificazione a inizio sprint degli obiettivi, tempistiche e responsabilità nel periodo dello sprint corrente. Diviso in due parti:

- **parte 1** Viene raffinato e rivisto il product backlog, viene effettuata la scelta dello sprint goal (what).
- **parte 2** Si decidono gli item e viene raffinato come implementarli (how). Effettuato con solo il team senza la figura del product owner

**[Occasionale] Pair Programming** Utilizzato per risolvere problemi che causano il blocco di un componente del team per parecchio tempo su una issue.

**Meeting finale** Riflessioni e considerazioni finali sullo sprint passato. Suggerimenti per migliorare il prossimo. Diviso in tre parti:

- **Product backlog refinement** aggiunta di dettagli e riordino del product backlog
- **Sprint review** è stato ispezionato l'incremento, il Minimum Viable Product o di risultati sul processo. Discernere cosa è stato fatto e cosa no
- **Retrospettiva** Considerazioni sul team stesso e sui miglioramenti per il prossimo sprint.

### 9.2 Riepilogo Sprint

#### 9.2.1 Sprint 1

Durante questo primo sprint abbiamo completato l'organizzazione di massima.

**Deliverables** I deliverables per questo sprint sono stati i seguenti:

- Intervista con il cliente corredata da domande
- Ubiquitous Language
- Setup Organizzazione GitHub
- Setup Report, con relativa repository e CI

## 9.2.2 Sprint 2

Durante questo sprint abbiamo realizzato l'analisi dei requisiti del sistema e ideata l'architettura generale del sistema, dalla quale sono sorti alcuni dubbi che sono stati poi discussi e chiariti con l'esperto del dominio. Sono state realizzate due versioni dei mockup dell'applicazione, le quali hanno ricevuto entrambe giudizi positivi, ma andranno successivamente unite e raffinate per soddisfare al meglio i requisiti del committente.

**Deliverables** I deliverables per questo sprint sono stati i seguenti:

- Mockup
- Diagramma dei casi d'uso
- Diagramma delle classi

## 9.2.3 Sprint 3

Durante questo sprint sono stati realizzati i mockup definitivi su cui si baseranno le schermate dell'applicativo. Abbiamo realizzato il design di dettaglio tramite diagramma delle classi per descrivere l'architettura del sistema ed infine abbiamo impostato il progetto base Scala su GitHub (Main, sbt e ScalaTest).

**Deliverables** I deliverables per questo sprint sono stati i seguenti:

- Mockup definitivi dell'applicativo
- Schema di navigazione delle pagine dei mockup
- Diagramma delle classi in dettaglio
- Setup base progetto Scala su GitHub

## 9.2.4 Sprint 4

Durante questo sprint è stata implementata completamente la visualizzazione della pagina iniziale, è stata implementata la risposta di un quiz ed è stato effettuato il setup della CI.

**Deliverables** I deliverables per questo sprint sono stati i seguenti:

- Visualizzazione della pagina iniziale
- Implementata risposta di un quiz
- Setup della CI

## 9.2.5 Sprint 5

Durante questo sprint abbiamo sviluppato la possibilità di avviare una partita utilizzando il menu principale, inoltre è stata implementata la visualizzazione delle domande e delle rispettive risposte.

**Deliverables** I deliverables per questo sprint sono stati i seguenti:

- Avvio di una partita;
- Navigazione del menu principale;
- Visualizzazione domande e risposte.

## 9.2.6 Sprint 6

Durante questo sprint sono stati implementati i meccanismi di selezione risposta e feedback su di essa. Inoltre, sono ora visibili e selezionabili i corsi disponibili prima di iniziare una partita e le domande relative ad esse sono estratte dal pool generale di quiz. Per ogni domanda inoltre, per variare l'esperienza, sono anche estratte a sorte varie possibili risposte. La sessione di ogni utente è ora salvata. Infine, è stato corretto il setup della CI.

**Deliverables** I deliverables per questo sprint sono stati i seguenti:

- Selezionare le risposte e ottenere dei feedback
- Selezione dei corsi pre-partita
- Visualizzazione delle opzioni di aggiunta domande
- Test automatici nella CI GitHub

## 9.2.7 Sprint 7

In questo sprint sono stati completati tutti i goal, quindi abbiamo implementato la selezione dei corsi pre-partita, l'aggiunta di corsi e domande tramite navigazione dei rispettivi menu. Abbiamo aggiunto la possibilità di personalizzare le impostazioni ed avviare la partita con esse. Infine è stata implementata la possibilità di leggere le domande e le risposte dal file personale.

**Deliverables** I deliverables per questo sprint sono stati i seguenti:

- Completato la selezione dei corsi pre-partita
- Aggiunta di corsi e domande
- Selezione delle impostazioni di gioco
- Lettura di domande e risposte da file

## 9.2.8 Sprint 8

In questo sprint non sono stati completati tutti i goal a causa dei problemi che sono sorti principalmente durante l'aggiunta dell'interfaccia grafica basata su ScalaFX. Questo ha portato a diverse problematiche di compatibilità con la parte di interfaccia grafica da linea di comando precedentemente implementata.

**Deliverables** I deliverables per questo sprint sono stati i seguenti:

- Create schermate dell'applicazione in FXML
- Aggiunta navigazione tra le pagine tramite interfaccia grafica

## 9.2.9 Sprint 9

In questo Sprint l'obiettivo è stato quello di finire più task possibili per dedicare la prossima iterazione ai soli refinement. La mole di lavoro svolto è stata ampia, anche grazie alla capitalizzazione del buon design, ma alcuni task sono stati rimandati successivamente. Parte del tempo è stato dedicato alla risoluzione dei bug emersi e agli eventuali enhancements, sia grafici che della logica applicativa. Inoltre, buona parte dell'interfaccia grafica è stata implementata.

**Deliverables** I deliverables per questo sprint sono stati i seguenti:

- Aggiunta esportazione quiz
- Modifica di un corso
- Modifica di un quiz
- Visualizzazione delle statistiche
- Corretta visualizzazione grafica della pagina di gioco

### 9.2.10 Sprint 10

In quest'ultimo sprint ci si è focalizzati sul completamento dell'intera applicazione e sul concludere i task lasciati in precedenza. Inoltre sono stati corretti alcuni bug emersi dall'uso dell'applicazione. Ulteriori funzionalità opzionali e enhancements sono state aggiunti per completare l'applicativo e il progetto in generale.

**Deliverables** I deliverables finali sono stati i seguenti:

- Completata la visualizzazione delle statistiche personali
- Selezione della modalità di gioco
- Importare domande
- Revisione dei quiz a partita giocata
- Completata l'interfaccia grafica
- Guida utente

## 9.3 Diagramma di Gantt

Durante i vari sprint è stato aggiornato anche il diagramma di Gantt creato inizialmente su Jira, come visibile nel paragrafo 2.3. Di seguito, nella Figura 9.1 si può vedere la versione finale del diagramma con le user story.



Figura 9.1: Diagramma di Gantt aggiornato alla fine del progetto

# Capitolo 10

# Conclusioni

Al termine dello sviluppo dell'applicazione, il prodotto rispecchia le richieste fatte inizialmente dall'esperto del dominio. In particolare, sono stati rispettati tutti i requisiti funzionali e sono anche stati aggiunti due requisiti funzionali opzionali, quali la modalità di gioco con sfida a tempo e le statistiche relative al tempo medio di risposta. Per poter rendere l'applicazione utilizzabile su larga scala bisognerebbe adottare delle accortezze particolari di seguito indicate.

## 10.1 Sviluppi Futuri

Tutti i requisiti funzionali opzionali precedentemente citati (capitolo 4.3.2), tranne quelli già sopra indicati come implementati, sono sicuramente i primi aspetti da inserire in uno sviluppo futuro, dal momento che sono stati segnalati dall'esperto del dominio durante l'analisi iniziale. In più, dato che il progetto sviluppato è un'applicazione desktop, non è difficile immaginarsi ulteriori funzionalità aggiuntive quali:

- salvataggio di tutti i dati del giocatore e delle partite in una architettura cloud o server based: si può creare una classifica generale con tutti gli studenti utilizzatori dell'applicazione, basata su quante partite hanno effettuato e quante risposte corrette hanno dato sul totale dei quiz presenti;
- sfida tra più studenti: in questa modalità di gioco più utenti possono sfidarsi utilizzando ciascuno l'applicativo sul proprio dispositivo;
- condivisione dei quiz inseriti da un utente: chiunque può avere a disposizione ulteriori corsi e/o quiz sui quali esercitarsi, cercando di coprire il più possibile gli argomenti presenti negli esami;
- segnalazione da parte dei giocatori di quiz da correggere, perché incompleti o inesatti.

# Capitolo 11

## Guida Utente

Una volta aperta l'applicazione, viene visualizzato il menu iniziale:

**Gioca** Seleziona almeno un corso per poter iniziare una partita con quiz riguardanti i corsi indicati. Successivamente, scegli la modalità di gioco:

- partita standard: 10 quiz, ciascuno da rispondere in massimo 15 secondi; ogni quiz ha 4 possibili risposte, 1 sola risposta è corretta; scoprirai la correttezza o meno della risposta data dopo ogni quesito
- partita blitz: rispondi a più quiz possibili in 2 minuti (120 secondi); ogni quiz ha 4 possibili risposte, 1 sola risposta è corretta; scoprirai la correttezza o meno delle risposte date solo al termine della partita
- partita personalizzata: scegli il numero di quiz a cui rispondere e il tempo massimo (in secondi) in cui puoi rispondere a ciascuno; ogni quiz ha 4 possibili risposte, 1 sola risposta è corretta

Al termine di una partita c'è il riepilogo con:

- il numero totale di domande risposte correttamente
- il numero totale di punti guadagnati
- per ogni quiz nel gioco:
  - corso in cui è inserito
  - risposta data
  - risposta corretta, se quella data è errata

**Statistiche** Visualizza le statistiche riguardanti ai tuoi ripassi:

- relative ad un corso selezionato
- relative ad un quiz selezionato tra quelli del corso indicato
- globali

Per ciascuna tipologia di statistica ci sono:

- punti acquisiti
- quiz risposti
- risposte corrette
- precisione (%)
- tempo medio di risposta (sec)

**Impostazioni** Per poter apportare modifiche nelle impostazioni generali:

- importa quiz: importa un file JSON dal file system contenente nuovi quiz
- esporta quiz: esporta il file JSON contenente i corsi con i relativi quiz
- modifica corso
- modifica quiz
- aggiungi corso: inserisci un nuovo corso
- aggiungi quiz: inserisci nuovi quiz ad un determinato corso, così da poterti esercitare su più argomenti

**Esci** Per uscire dall'applicazione

# Bibliografia

- [gam] gameshow. URL: [https://en.wikipedia.org/wiki/Game\\_show](https://en.wikipedia.org/wiki/Game_show).
- [jeo] jeopardy. URL: <https://en.wikipedia.org/wiki/Jeopardy!>.
- [qui] quizgame. URL: <https://en.wikipedia.org/wiki/Quiz>.
- [who] whowantstobeamillionaire. URL: [https://en.wikipedia.org/wiki/Who\\_Wants\\_to\\_Be\\_a\\_Millionaire](https://en.wikipedia.org/wiki/Who_Wants_to_Be_a_Millionaire).