Quizzy - Analisi e Progettazione

Università degli studi di Bergamo
Ingegneria Informatica

Arnoldi Elisa Matr. 1080572
Colpani Filippo Matr. 1078874
Foglieni Luca Matr. 1081399

Gennaio 2024

Indice

1 Introduzione

Il presente documento si propone di esplorare in profondità l'analisi e la progettazione di Quizzy, integrando gli standard UML per rappresentare in modo chiaro e conciso gli elementi che compongono il sistema e la realtà analizzata. Saranno fornite dettagliate descrizioni delle componenti chiave, delle relazioni e delle modalità di interazione tra i diversi moduli. L'obiettivo è fornire una visione esaustiva delle scelte di progettazione e delle strategie implementative che guidano lo sviluppo di Quizzy.

1.1 Termini e definizioni

- Quiz: insieme di domande di vario tipo
- Partita: istanza giocabile di un quiz
- Docente: classe di utenti del sistema che amministrano quiz e partite
- Studente: classe di utenti che partecipano ad una partita

2 Analisi dei Requisiti

2.1 Specifica dei requisiti

I requisiti del progetto sono stati estratti inizialmente dopo un paio di incontri per stabilire le linee guida del progetto e i comportamenti basilari che ci si aspettava dall'applicazione. Successivamente e altri più particolari in fase di sviluppo. In particolare:

1. Il docente che utilizza l'applicazione deve essere in grado in ogni momento di poter creare, modificare ed eliminare nuovi quiz, una volta fatto accesso al suo account personale.
   (a) L'account del docente in fase di registrazione è costituito da una email, nome e password, tutti necessari, dove l'univocità è data dall'email.
   (b) Il docente deve essere in grado di accedere in ogni momento alla sua sezione quiz e poter creare nuovi quiz.
   (c) Il docente deve essere in grado di poter far partire in ogni momento il quiz da esso desiderato dalla sua area personale, accessibile dagli studenti tramite PIN.
2. I quiz possono essere costituiti da un numero variabile di domande, in particolare:
   (a) I quiz possono avere due tipologie di domande, a risposta multipla costituita da 4 risposte, oppure a risposta vera o falso, con risposte fisse: "Vero" o "Falso".
   (b) Solo una delle possibili risposte è corretta.
   (c) Le domande possono essere create nella sezione apposita dell'applicazione e devono essere confermate ogni volta prima di passare alla creazione della successiva
   (d) Ogni domanda può essere eliminata, in caso di errore, sia prima che dopo la conferma di salvataggio della domanda.
   (e) Una volta eseguita la creazione delle domande per uscire dalla schermata di creazione e effettivamente eseguire il quiz, bisogna confermare tutti i dati inseriti nella loro interezza, tramite il pulsante "Confirm".
3. Per procedere all'avvio dei quiz bisogna accedere alla sezione "My Quiz" dell'area personale del docente, cliccando sul pulsante "Start" presente su ogni quiz nell'elenco della schermata principale.
   (a) Per accedere all'area personale è necessario solo fare accesso all'applicazione che può essere utilizzata su ogni dispositivo.
   (b) Bisogna essere connessi per l'effettivo uso ad una rete locale.
   (c) Non sono richieste particolari specifiche hardware per l'utilizzo dell'applicazione
4. All'avvio del quiz il docente dovrà presentare lo schermo ai propri studenti, in particolare:
   (a) Sullo schermo del docente verrà presentato un pin a 6 cifre che gli studenti dovranno inserire nella schermata iniziale, una volta fatto accesso a Quizzy, per fare accesso alla partita avviata dal docente.
   (b) Gli studenti possono fare accesso alla partita da qualsiasi dispositivo, l'importante è che siano collegati alla stessa rete locale del docente.
   (c) Una volta entrati nella partita gli studenti dovranno scegliere un nome e un avatar a scelta dall'elenco, in modo da identificarsi.
   (d) Scelto l'avatar e il nome, gli studenti dovranno attendere che il docente faccia l'avvio della partita, nell'attesa, sullo schermo del docente verrà visualizzato l'elenco delle persone nella partita con relativo nome e avatar.
   (e) Non c'è limite al numero di partecipanti al quiz
5. Una volta iniziata la partita verranno visualizzate le domande:

(a) Sullo schermo del docente verrà visualizzata la domanda

(b) Sugli schermi degli studenti, in base alla loro grandezza, verranno visualizzate sia le domande che la risposta. Invece se lo studente utilizza uno smartphone, a quel punto verranno visualizzate solo le possibili opzioni e al posto dell'effettiva risposta, verranno visualizzati dei simboli con sfondi colorati che combaciano a quelli delle risposte proiettate dal docente.

(c) Gli studenti hanno un tempo limite per rispondere alla domande di un minuto.

6. Dopo ogni domanda viene visualizzato il risultato della risposta.

   (a) In modo automatico allo scadere del tempo limite imposto viene visualizzato singolarmente dagli studenti l'esito della loro risposta

   (b) In base alla correttezza della domanda e alla velocità della risposta viene attribuito un punteggio.

   (c) Sullo schermo del docente si può invece visualizzare la classifica provvisoria

7. Al termine del quiz gli studenti devono essere in grado di vedere la classifica finale

   (a) Verranno visualizzati sullo schermo del professore i tre più bravi giocatori della partita.

   (b) La visualizzazione è costituita da un'animazione dove viene visualizzato il nome, il punteggio e l'avatar del personaggi sopra specificati.

8. Clonare quiz esistenti
   Clonare quiz esistenti consente ai docenti di clonare un quiz già esistente, precedentemente creato da loro o da un altro professore, per poi modificarlo nel caso lo si desideri.

9. Aggiungere immagini alle domande del quiz
   Per arricchire l'esperienza è possibile aggiungere un'immagine per domanda. Questo permette al professore di fare una domanda specifica su una specifica immagine a cui gli studenti dovranno rispondere.

10. Creazione di domande con numero di risposta a scelta.
    A discrezione del docente, si possono inserire dalle 2 alle 4 risposte per ogni domanda.

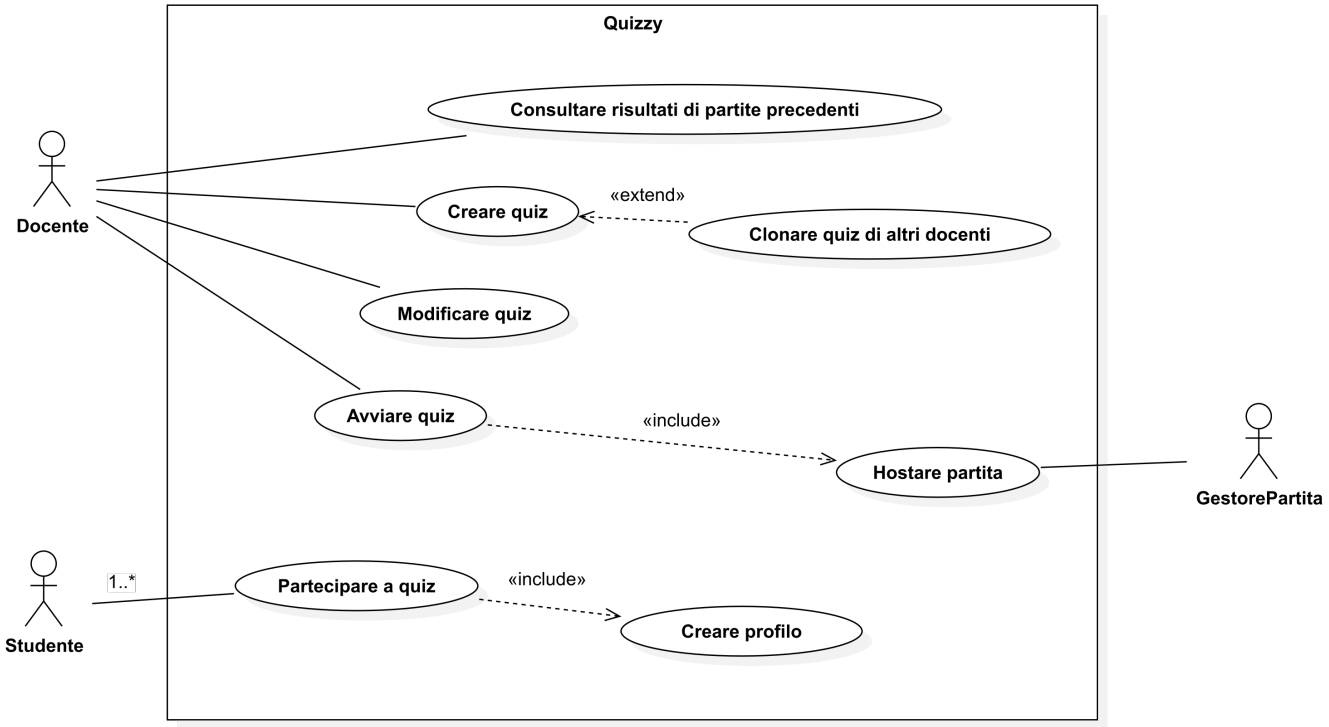
11. Modalità di presentazione offline tramite app
    I quiz sono scaricabili localmente dal professore tramite l'app Quizzy per Windows, Linux MacOS, Android ed iOS. Nel caso in cui ci si trovi in una situazione in cui la connessione non è sufficiente per un'esperienza ottimale, il professore potrà presentare i quiz scaricati localmente nell'app in modo da poter comunque utilizzare le domande per una lezione interattiva con gli studenti.

2.2 Classificazione MoSCoW

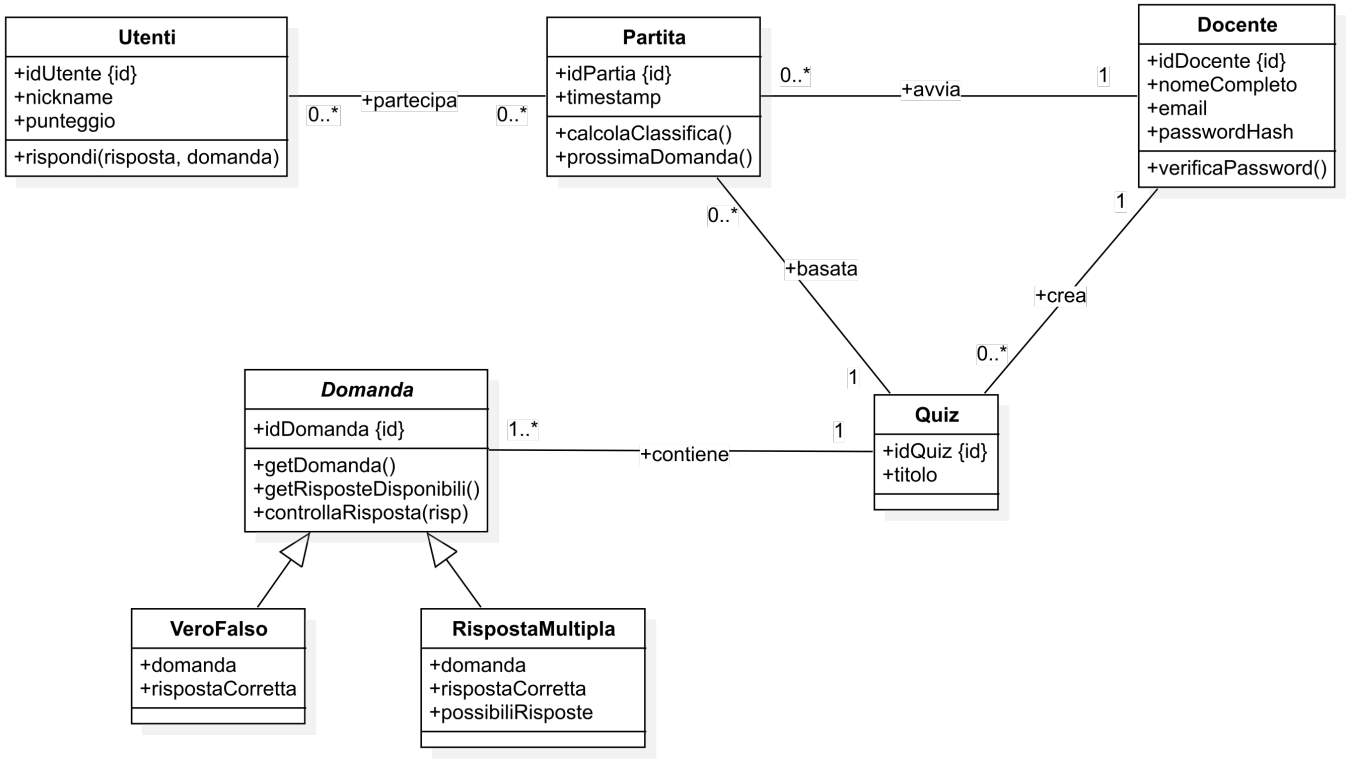
Must Have	**Should Have**	**Could Have**	**Won't Have**
1	4.a, 4.e	4.c	8
2	5.c	7.b	9
3	6		10
4.b, 4.d	7.a		11
5.a, 5.b			

3 UML

3.1 Diagramma dei casi d'uso

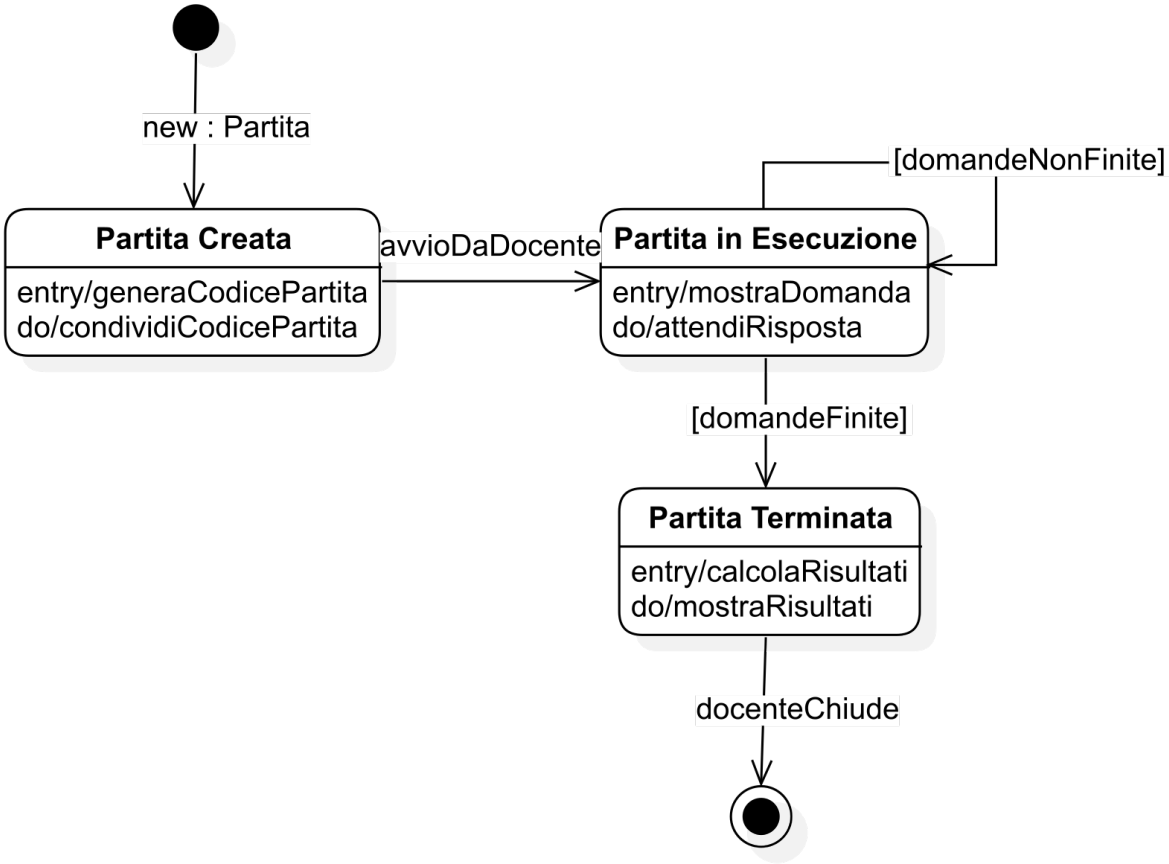


3.2 Diagramma delle classi



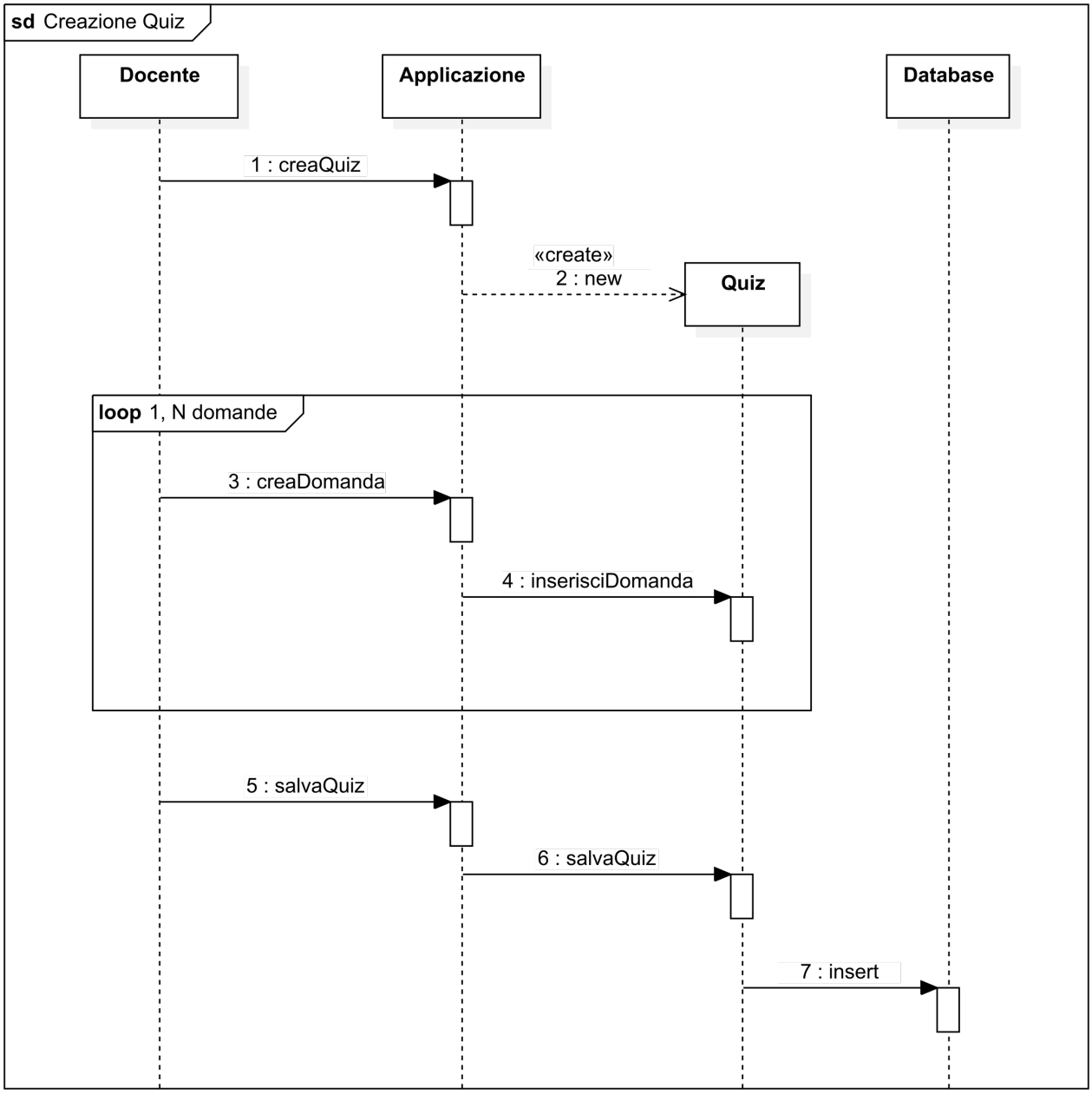
3.3 Diagramma di stato

3.3.1 Oggetto Partita

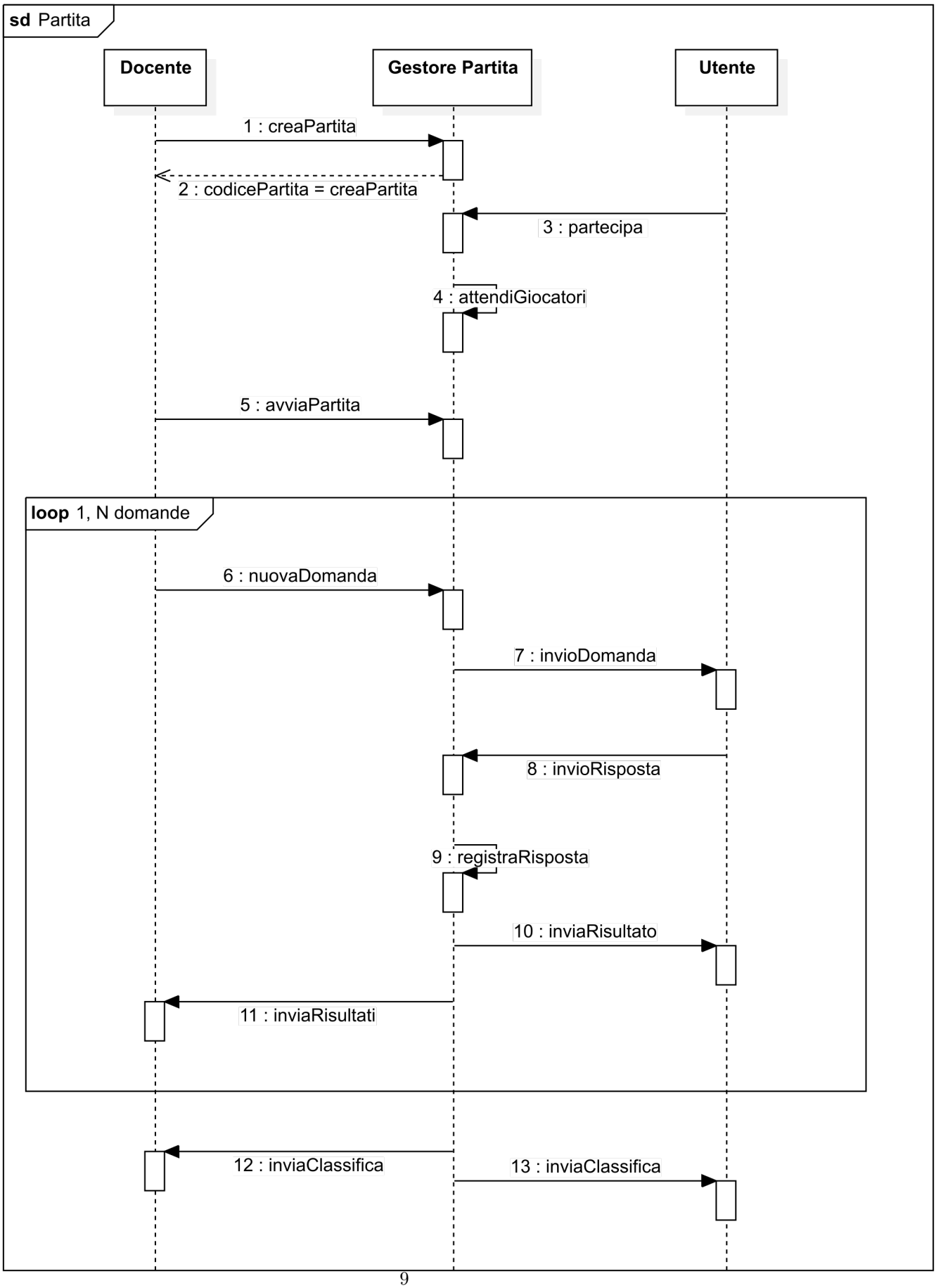


3.4 Diagramma di sequenza

3.4.1 Creazione Quiz

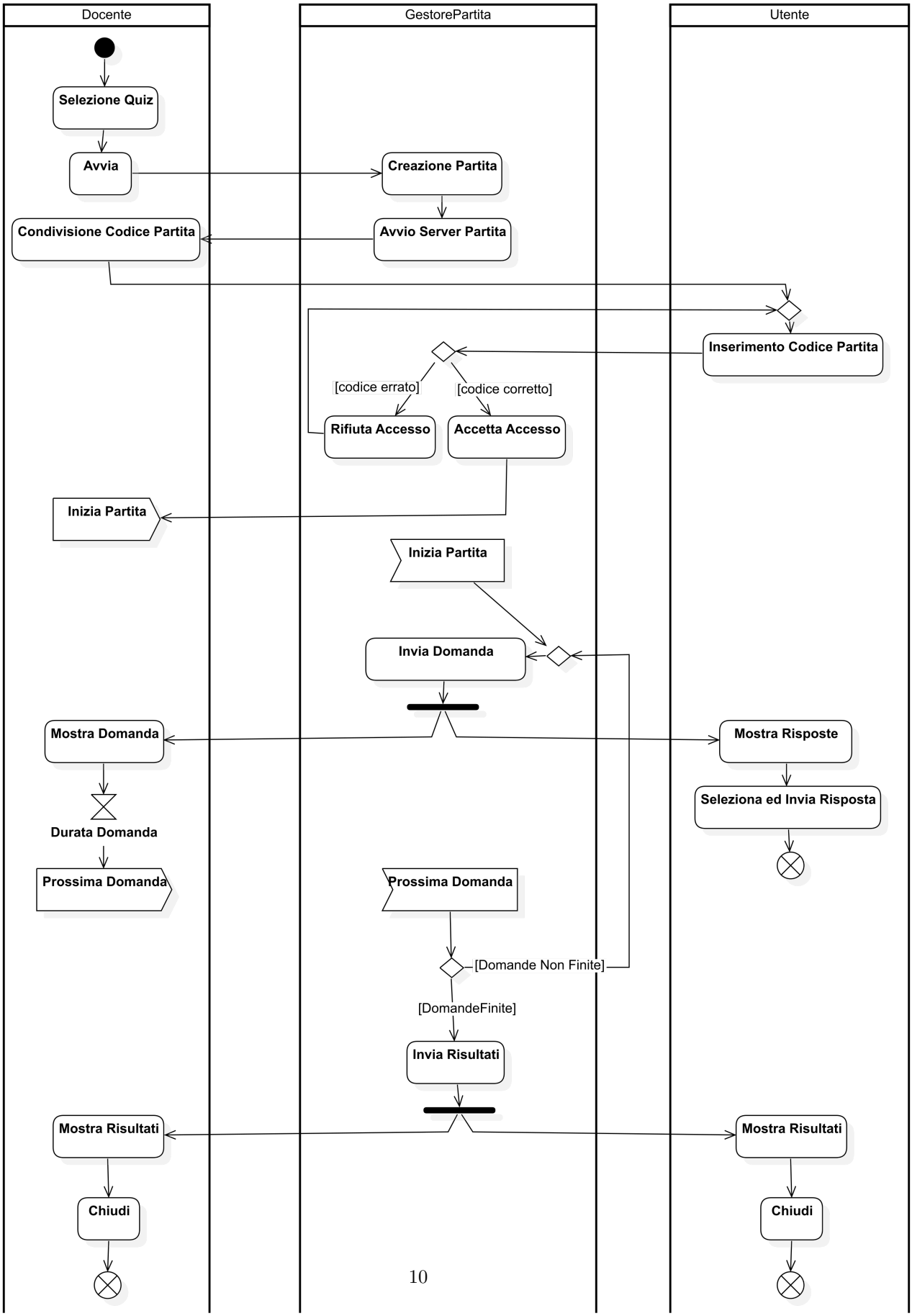


3.4.2 Creazione e svolgimento Partita



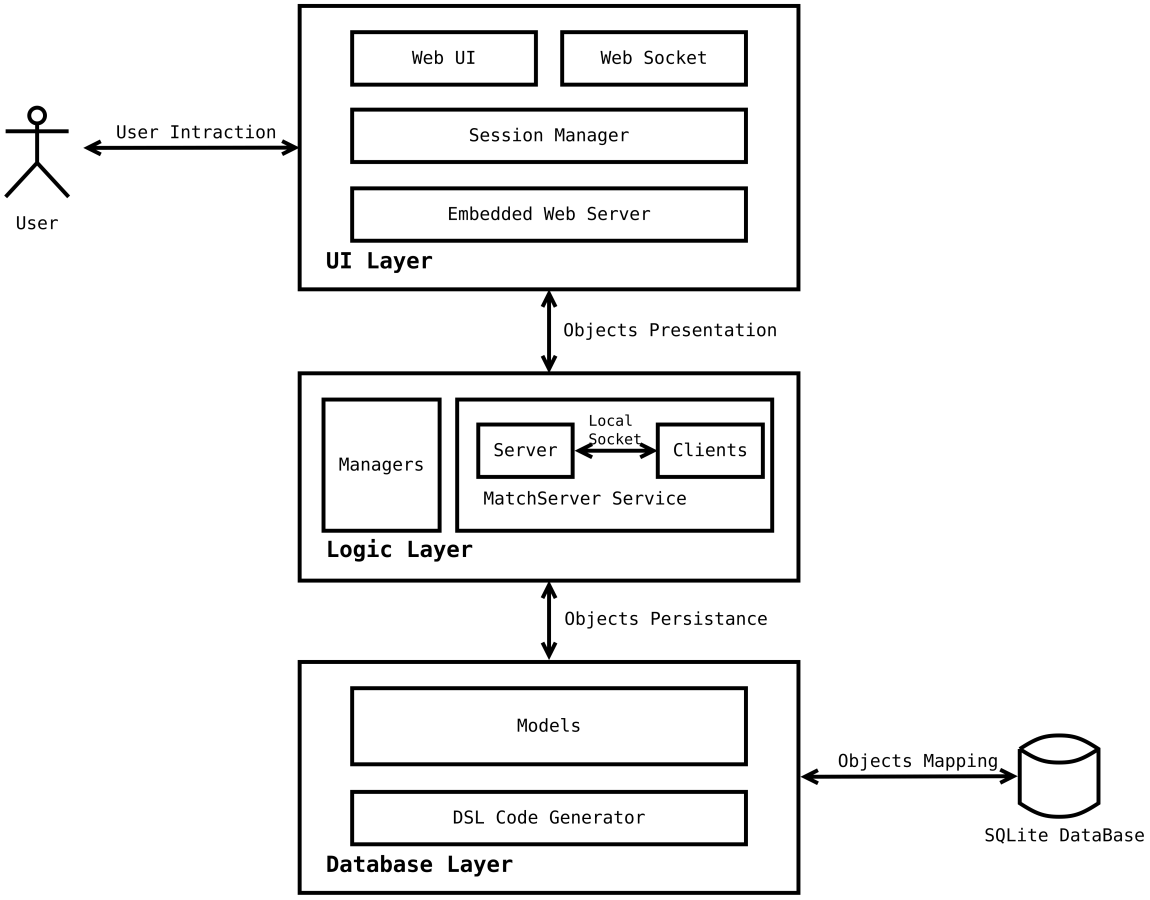
3.5 Diagramma delle attività

3.5.1 Creazione e svolgimento Partita



4 Software Architecture

L'architettura del software realizzato è suddivisa in 3 layer: Database Layer, Logic Layer, UI Layer, ogni layer espone funzionalità al layer superiore e sfrutta le funzionalità di quello inferiore.



4.1 Database Layer

Questo layer gestisce le transazioni con il database fisico. Espone al layer superiore le classi model per la gestione persistente dei dati, rendendo trasparente l'accesso alla base di dati. Viene suddiviso in due componenti principali: il *DSL Code Generator* che gestisce la connessione e lo scambio di dati con il database, classi *Models* che rappresentano gli oggetti del sistema.

4.2 Logic Layer

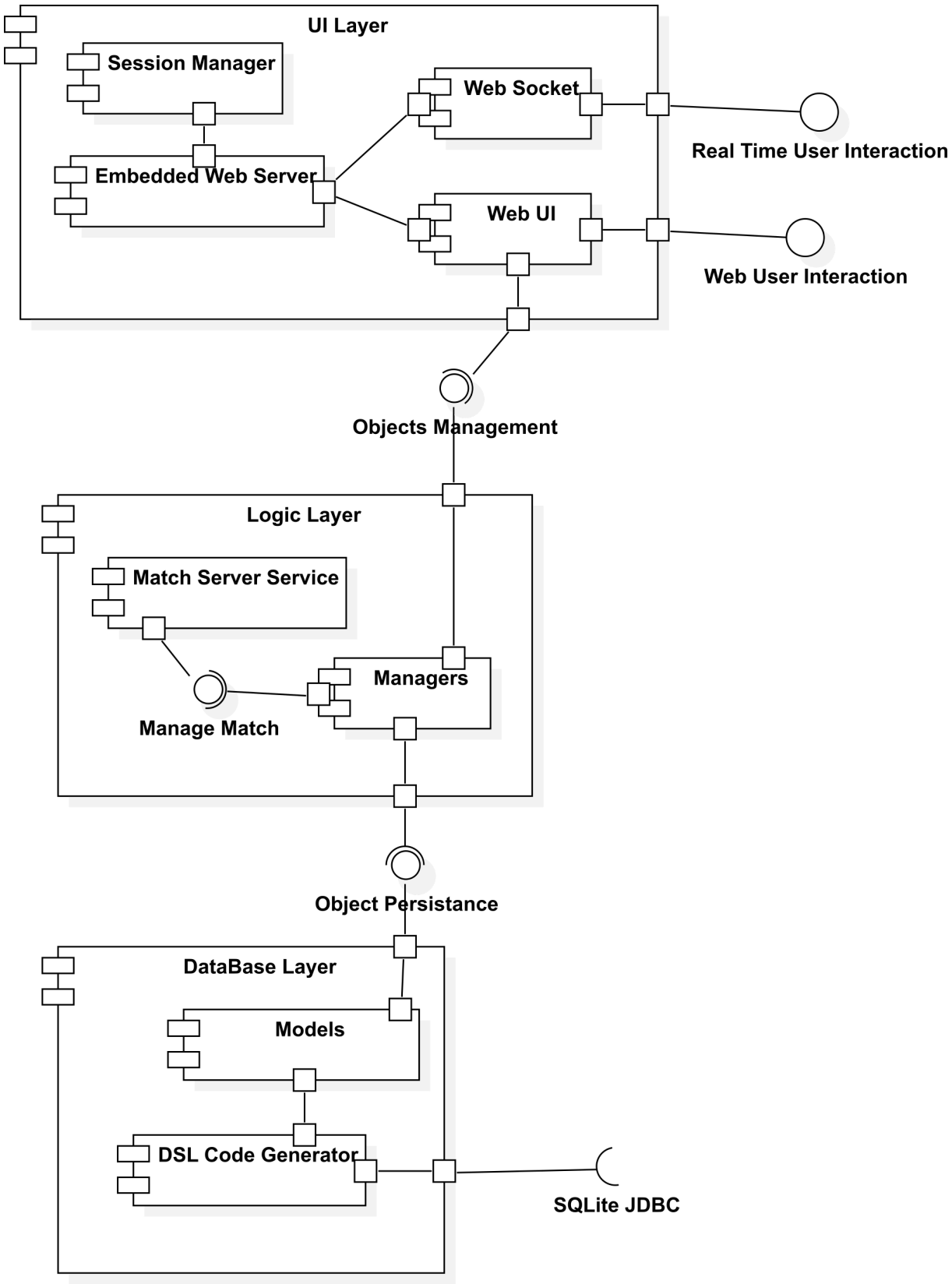
Questo layer gestisce la logica di business del sistema, è suddiviso in due componenti principali: le classi *Manager*, per la gestione delle logiche di business delle classi *Models* del *Database Layer*, ed il *Match Server Service*, che instanzia e gestisce il microservizio per la gestione in tempo reale della partita; in questo modo sarà possibile una migliore scalabilità futura del componente.

4.3 UI Layer

Questo layer gestisce le interazioni con gli utenti ed espone i servizi del *Logic Layer* mediante *Web UI*, e svolge la funzione di *Wrapper* del *Match Server Service*, nello specifico le socket locali vengono esposte dal componente *Web Socket* che gestisce le interazioni in tempo reale.

5 Software Design

5.1 Diagramma dei componenti



5.2 Design pattern utilizzati

Il design pattern del *Wrapper* è stato utilizzato per le classi *Models* dato che la libreria JOOQ, utilizzata per le funzionalità di DSL e generazione del codice, crea classi che rappresentano i record delle tabelle è stato necessario incapsulare tali oggetti in classi che ne gestissero i controlli nonché la persistenza delle modifiche, limitando la modifica degli attributi che altrimenti avrebbero minato l'integrità del sistema. Questo pattern è stato anche applicato alle classi del componete *Match Server Service*, infatti i client che comunicano a livello locale con il servizio del server sono stati incapsulati in una socket web così da rendere il servizio indipendente dalla macchina su cui viene eseguito favorendo l'interoperabilità del servizio. in questo modo l'utente integrità con il sistema mediante un ristretto numero di operazioni utilizzando il protocollo delle websocket.
Il secondo design pattern utilizzato è la *Factory*, per gestire la creazione di domande di diverso tipo, come i vero o falso o quelle a risposta multipla. Essendo la classe *it.quizzy.databaselayer.models.Domanda* la generalizzazione sotto forma di classe astratta di tutte le classi di domande, la classe DomandeFactory consiste in un metodo statico che mediante un parametro enumerativo definisce il tipo di domanda richiesto e ritorna l'oggetto specificato.

5.3 Analisi dei moduli

Per valutare la qualità della progettazione del software sono state calcolate le seguenti metriche, mediante l'ausilio del software JDepend, suddivise per pacchetto. Nella seguente tabella sono state riportate le metriche principali adottate per valutare la qualità del design del software, come si può notare l'astrattezza complessiva è minima, possibili refactor potrebbero includere il migliorare l'astrazione di alcune classi. Inoltre è possibile notare come nei layer esterni l'intabilità sia maggiore rispetto al layer logico.

Package	**Afferent Couplings**	**Efferent Couplings**	**Abstractness**	**Instability**
it.quizzy.databaselayer	0	3	0	1
it.quizzy.databaselayer.exceptions	5	0	0	0
it.quizzy.databaselayer.models	8	6	0.2	0.43
it.quizzy.databaselayer.models.domande	4	8	0	0.67
it.quizzy.databaselayer.util	4	2	0	0.33
it.quizzy.logiclayer.factory	1	3	0	0.75
it.quizzy.logiclayer.manager	4	8	0	0.67
it.quizzy.logiclayer.server	3	3	0.29	0.5
it.quizzy.uilayer.launch	0	5	0	1
it.quizzy.uilayer.websocket	0	4	0	1

Per una migliore comprensione delle dipendenze e l'accoppiamento tra moduli è stato anche utilizzato il tool https://github.com/glato/emerge, per generare un grafo delle dipendenze interattivo.

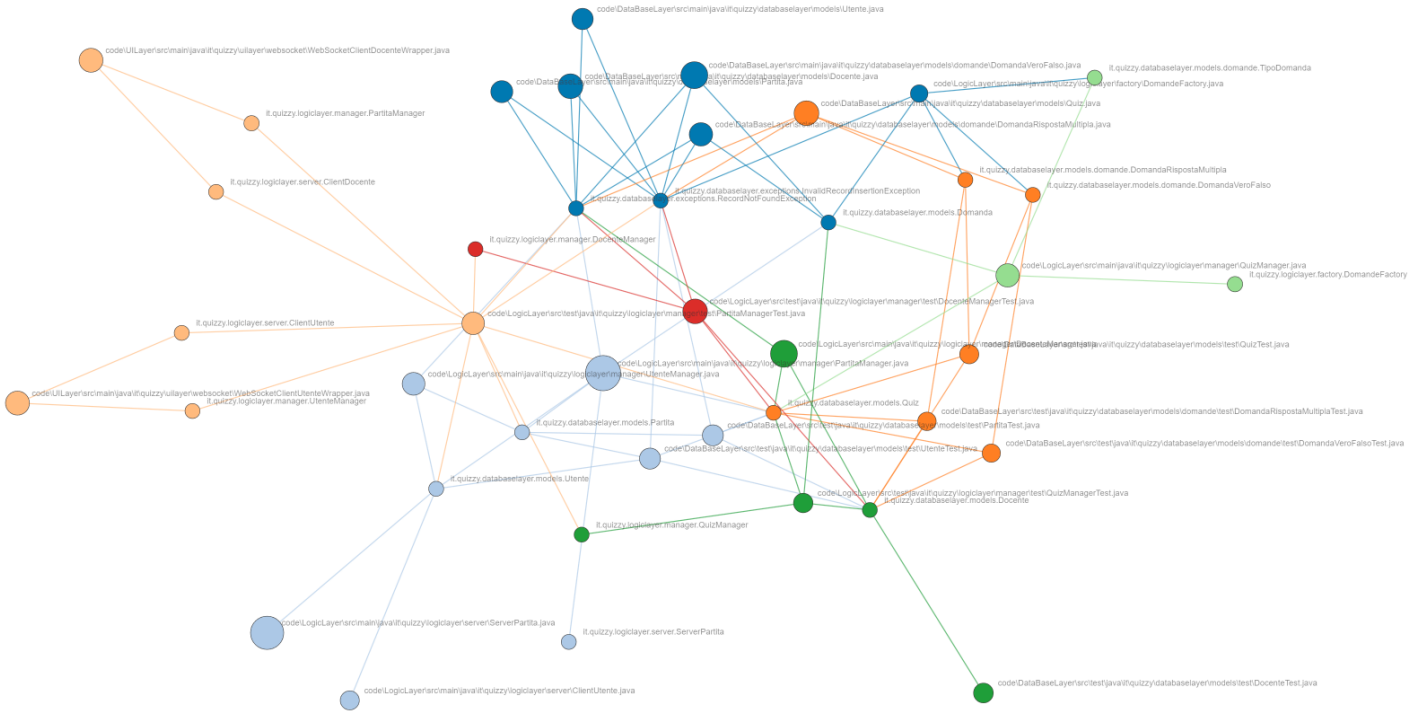


Figura 1: Emerge dependency graph

6 Software Testing

La fase di testing, dato il metodo di sviluppo adottato, è stata continua a ha guidato buona parte dell'implementazione dei layer. Nello specifico per il *Database Layer* sono stati realizzati test JUnit per verificare il corretto uso del database e la persistenza degli oggetti, nonché l'integrità dei dati. Per il *Logic Layer* sono state testate le classi manager con test Junit in grado di verificare il corretto comportamento e logica di business relativi. In fine per lo *UI Layer* sono stati adottati test di natura manuale per la validazione dei requisiti e la verifica del corretto funzionamento del sistema.
Per valutare la qualità dei test è stato utilizzato il tool di Eclipse Coverage, di seguito sono riportati i risultati della copertura dei test JUnit

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
src/main/java	80,0 %	774	193	967
it.quizzy.databaselayer.models.domande	82,3 %	292	63	355
DomandaVeroFalso.java	76,4 %	152	47	199
DomandaRispostaMultipla.java	87,9 %	116	16	132
TipoDomanda.java	100,0 %	24	0	24
it.quizzy.databaselayer.models	87,8 %	438	61	499
Quiz.java	85,6 %	125	21	146
Docente.java	86,7 %	130	20	150
Partita.java	88,1 %	89	12	101
Utente.java	91,3 %	84	8	92
Domanda.java	100,0 %	10	0	10

Figura 2: Coverage DataBaseLayer

Element	Coverage	Covered Instructi	Missed Instructions	Total Instructions
LogicLayer	86,5 %	1.534	239	1.773
src/main/java	82,4 %	1.015	217	1.232
it.quizzy.logiclayer.factory	89,3 %	25	3	28
DomandeFactory.java	89,3 %	25	3	28
it.quizzy.logiclayer.manager	89,0 %	560	69	629
DocenteManager.java	85,1 %	74	13	87
PartitaManager.java	91,2 %	331	32	363
QuizManager.java	87,6 %	99	14	113
UtenteManager.java	84,8 %	56	10	66
it.quizzy.logiclayer.server	74,8 %	430	145	575

Figura 3: Coverage LogicLayer

7 Manutenzione

Adottando un metodo di sviluppo agile le attività di manutenzione e di adattamento del software sono risultate implicite e ben integrate con il processo di sviluppo. Sono state svolte diverse attività di refactoring, tra le principali:

- Utilizzo del Factory design pattern:
  Per migliorare a manutenibilità del codice relativo alla creazione e la modifica dei quiz, si è deciso di applicare il design pattern *Factory* così da fornire un'interfaccia unificata per la creazione di domande, permettendo alle sottoclassi di decidere quali tipi di oggetti creare. Questo porta a una maggiore flessibilità nel codice, in quanto consente di aggiungere nuovi tipi di oggetti senza modificare l'intero codice. Il risultato è un approccio strutturato alla creazione degli oggetti, contribuendo a rendere il codice più flessibile, estendibile, mantenibile e coeso.

- Aggiunta di un pin per la partita:
  In seguito alla necessità di utilizzare un pin per la registrazione di un utente ad una partita è stato modificato il database, aggiungendo la relativa colonna nella tabella partite. Successivamente mediante la libreria JOOQ sono state ri mappate le classi delle tabelle del database, sono stati modificati i test per verificare le funzionalità richieste, come la generazione del pin casuale e sono state riscritte le classi *Models* coinvolte così da passare i test JUnit. In fine sono state propagate le modifiche a layer superiori.

- Aggiunta immagine avatar:
  In seguito alla necessità degli utenti di poter selezionare un immagine per l'avatar è stato modificato il database, aggiungendo la relativa colonna nella tabella utente. Successivamente mediante la libreria JOOQ sono state ri mappate le classi delle tabelle del database, sono stati modificati i test per verificare le funzionalità richieste, sono state riscritte le classi *Models* coinvolte così da passare i test JUnit. In fine sono state propagate le modifiche a layer superiori.