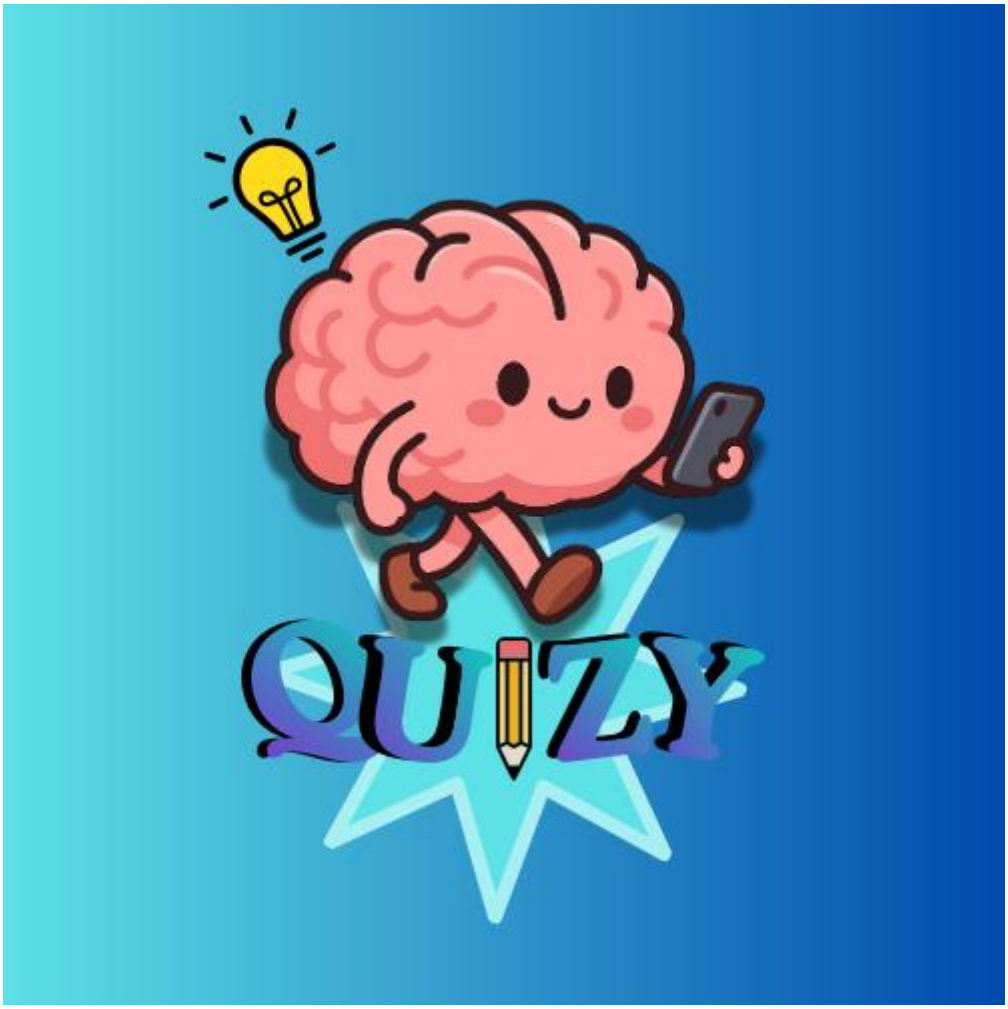
Università degli Studi di Salerno

Corso di Ingegneria del Software

Quiz

System Design Document

Version 0.8



Data: 25/11/2025

Progetto: Quizy	Versione: 0.8
Documento: System Design Document	Data: 25/11/2025

Coordinatore del progetto

Nome	Matricola
Davide Nino Longobardi	0512119971

Partecipanti

Nome	Matricola
Roberto Caiazza	0512116335

Revision History

17/11/2025	0.0	Stesura dell'indice e introduzione	Roberto Caiazza
18/11/2025	0.1	Stesura del System Decomposition	Davide Nino Longobardi
19/11/2025	0.2	Stesura del Persistent Data Management e overview	Fabiana Paciello
24/11/2025	0.3	Rifinitura dell'introduzione e Proposed software architecture	Davide Nino Longobardi, Fabiana Paciello, Roberto Caiazza
25/11/2025	0.4	Current Software Architecture e Glossario	Davide Nino Longobardi, Fabiana Paciello, Roberto Caiazza
08/12/2025	0.5	Aggiornamento della matrice degli accessi e stesura del Subsystem services	Davide Nino Longobardi, Roberto Caiazza
24/12/2025	0.6	Aggiornamento della matrice degli accessi e stesura del Subsistem services	Davide Nino Longobardi, Roberto Caiazza
06/02/2026	0.7	Revisione del documento	Roberto Caiazza
08/02/2026	0.8	Revisione finale del documento	Davide Nino Longobardi, Roberto Caizza

Indice

1.INTRODUZIONE

1.1. Purpose of the system

Lo scopo del sistema Quizy e fornire una piattaforma avanzata per la creazione, gestione e somministrazione di quiz a scelta multipla, rivolta principalmente a docenti e studenti in ambito accademico. Il sistema garantisce un controllo completo sulle sessioni di quiz, sugli utenti e sui dati generati, offrendo strumenti per la personalizzazione dei test, la gestione sicura degli accessi e la protezione della privacy degli utenti. Rispetto alle soluzioni esistenti, Quizy si distingue per la sua flessibilità, la scalabilità, facilitando sia la creazione che la distribuzione dei quiz in modo efficiente e sicuro.

1.2. Design Goals

1. **Privacy e sicurezza dei dati**: Il sistema deve garantire la privacy e la protezione dei dati degli utenti, adottando la crittografia delle password tramite bcrypt per prevenire attacchi brute force e utilizzando il protocollo HTTPS per tutte le comunicazioni tra client e server.
2. **Prestazioni**: L'applicazione deve supportare almeno 200 utenti simultanei. Le prime richieste, in particolare durante il login, possono richiedere 2-3 secondi, mentre le successive devono mantenere tempi di risposta di circa 200-300 ms.
3. **Tempi di risposta**: Il sistema deve rispondere alle richieste degli utenti, dopo la fase di warm-up, entro 300 ms nel 95% dei casi, mentre le prime richieste possono richiedere fino a 3 secondi.
4. **Usabilità**: Il flusso di utilizzo deve minimizzare gli errori utente, garantendo che almeno il 90% delle sessioni di quiz venga completato senza errori di navigazione o blocchi, come verificato tramite test interni e simulazioni.
5. **Accuratezza e coerenza dei dati**: Grazie all'uso di un database relazionale e all'implementazione di un'API RESTful, il sistema deve garantire la coerenza e l'integrità dei dati, evitando errori di corrispondenza o perdita di informazioni.
6. **Efficienza delle risorse**: L'app mobile deve essere ottimizzata per consumare poche risorse di sistema e batteria, garantendo prestazioni elevate anche su dispositivi di fascia media.
7. **Gestione dei ruoli**: Il sistema deve implementare un controllo degli accessi basato su ruoli, separando chiaramente le funzionalità di creazione quiz da quelle di partecipazione, con workflow di richiesta e approvazione gestiti da un amministratore.
8. **Scalabilità**: L'architettura deve permettere una facile estensione per supportare un numero crescente di utenti e quiz senza degrado delle prestazioni.
9. **Accessibilità**: Non sono previste funzionalità specifiche per l'accessibilità nella versione attuale del sistema.

1.3. Definitions, acronyms and abbreviations

Campo	Formato corretto	Eccezione

Username	Univoco	Username già presente nel sistema.
Password	Alfanumerico. Minimo 8 caratteri e obbligatoriamente 1 lettera maiuscola, 1 minuscola, una cifra e 1 carattere speciale	
Punti risposta corretta	Valore reale. Deve essere positivo.	Punteggio negativo
Punti risposta sbagliata	Valore reale. Deve essere negativo o valore 0.	

1.4.Reference

- Università degli Studi di Salerno – Corso di Ingegneria del Software – "Proposta Progetto Quizy"
- Università degli Studi di Salerno – Corso di Ingegneria del Software – "Problem Statement Quizy"
- Università degli Studi di Salerno – Corso di Ingegneria del Software – "Requirements Analysis Document Quizy"

2. CURRENT SOFTWARE ARCHITECTURE

L'attuale sistema di riferimento sul mercato è rappresentato dalla piattaforma Kahoot, la quale adotta un modello SaaS (SaaS - Software as a Service). In questo modello, i client (dispositivi degli utenti) si collegano a un server centrale che gestisce la logica applicativa, la memorizzazione dei dati e la sincronizzazione in tempo reale delle sessioni di quiz.
Il flusso tipico prevede che un amministratore o docente crei un quiz tramite l'interfaccia web, che viene poi distribuito agli utenti partecipanti. Le risposte vengono raccolte in tempo reale e i risultati sono immediatamente disponibili grazie alla gestione centralizzata del server.
Sebbene questa architettura sia efficace per la gamification e la gestione di sessioni con molti utenti, presenta alcune limitazioni rispetto agli obiettivi di Quizy:

- **Personalizzazione limitata**: Le possibilità di adattare la piattaforma alle esigenze specifiche di un'organizzazione sono ridotte.

- **Controllo sui dati**: I dati degli utenti e dei quiz sono gestiti da terze parti, con limitato controllo sulla privacy e sulla sicurezza.
- **Scalabilità e costi**: L'utilizzo di servizi cloud può comportare costi crescenti e dipendenza da infrastrutture esterne.
- **Gestione dei ruoli**: Le funzionalità di gestione avanzata dei ruoli e dei permessi sono spesso limitate o non personalizzabili.

Per questi motivi, il sistema proposta da Quizy mira a superare tali limiti adottando un'architettura più flessibile, sicura e personalizzabile, in grado di garantire maggiore controllo sui dati e sulle funzionalità offerte agli utenti.

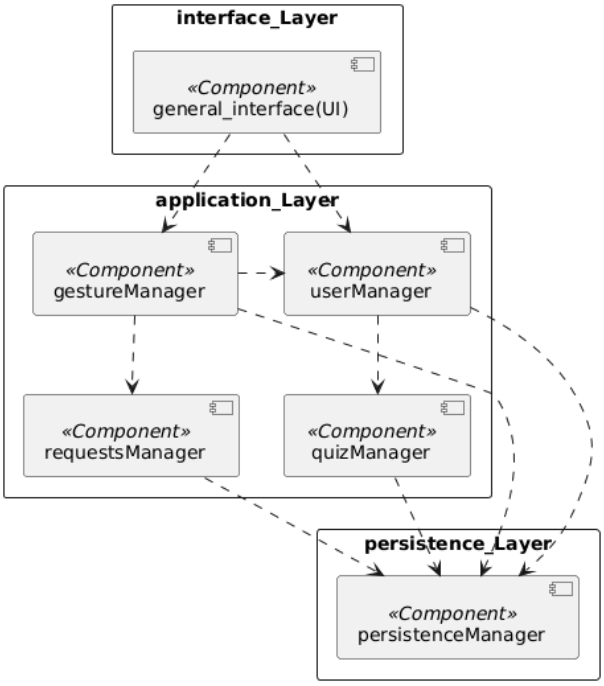
3. PROPOSED SOFTWARE ARCHITECTURE

3.1. Overview

Il sistema Quizy è strutturato per far dialogare l'applicazione Android (che l'utente ha in mano) con un archivio centrale sicuro (server) dove risiedono tutte le informazioni. Questa divisione permette agli utenti di accedere ai quiz da qualsiasi luogo tramite il proprio smartphone, mentre il sistema si occupa di tenere i dati sincronizzati e protetti in un unico posto.Per rendere il software facile da aggiornare e sicuro, non verrà costruito come un blocco unico, ma è organizzato in moduli indipendenti.L'organizzazione interna del software segue uno schema logico a tre livelli, pensato per separare chiaramente le responsabilità (revisione della Boundary-Control-Entity viste nell'analisi):

- **L'Interfaccia (Boundary):** È la "faccia" dell'applicazione, ovvero tutto ciò che l'utente vede e tocca sullo schermo (form di registrazione, pulsanti, lista dei quiz). Si occupa solo di mostrare le informazioni e raccogliere i comandi dell'utente.
- **La Logica di Controllo (Control):** È il "cervello" invisibile che lavora dietro le quinte. Quando l'utente preme un pulsante, è questo livello che decide cosa deve succedere, verifica se i dati inseriti sono giusti e coordina il passaggio da una schermata all'altra.
- **I Dati (Entity):** È la "memoria" del sistema. Qui vengono conservate e protette le informazioni essenziali che non devono andare perse, come i profili degli utenti, le domande dei quiz e i risultati ottenuti.

3.2. Subsystem Decomposition

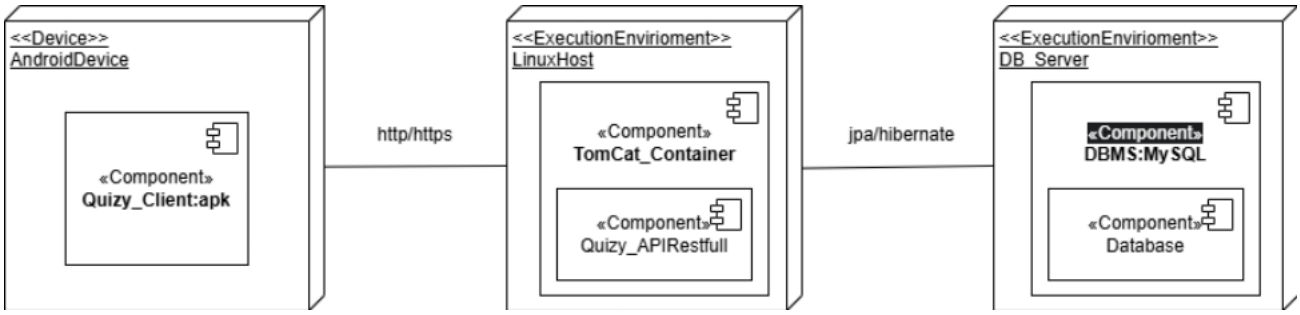


Il sistema è strutturato secondo un'architettura a **tre livelli (Layer) distinti**, progettata per garantire una separazione netta delle responsabilità e una gestione modulare dei sottosistemi:

1. **Interface Layer**: Livello dedicato all'Interfaccia Utente (UI). Rappresenta il front-end del sistema, ottimizzato per consentire all'utente la comunicazione con l'applicazione attraverso dispositivi mobile.
2. **Application Layer**: Livello preposto alla gestione della **logica di business**. Funge da intermediario elaborando le richieste degli utenti e coordinando la logica interna attraverso i seguenti componenti specializzati:
    a. **GestureManager**: questo strumento aiuta il Gestore Account a gestire gli account in modo efficiente. Le funzionalità principali includono la possibilità di accedere al sistema, uscire dal profilo e recuperare la password quando necessario, tutte relative al profilo del Gestore
    b. **UserManager**: si occupa di gestire gli accessi e gli account degli utenti. Le sue funzionalità, come ad esempio il login, il logout e il recupero della password, sono accessibili solo al Gestore Account.
    c. **RichiesteManager**: Componente dedicato alla gestione delle richieste per cambi di ruolo o supporto tecnico. Gestisce l'intero ciclo di vita delle istanze, occupandosi sia della loro creazione che del monitoraggio dello stato.
    d. **QuizManager**: Componente dedicato alla gestione operativa dei quiz. Permette la creazione, la visualizzazione, il completamento e l'eliminazione dei test a sistema.
3. Persistence Layer: Livello dedicato alla **persistenza dei dati** e alla loro integrità nel tempo.
    a. **PersistenceManage:** Rappresenta l'unico modulo responsabile dell'interazione con il **DBMS**, gestendo tutte le operazioni di lettura, scrittura e mantenimento sulla base dati.

3.3. Hardware/software mapping

Quiz verrà implementato attraverso un'architettura a tre livelli (Tree-Tier Architecture), che prevede più nodi per la distribuzione che dividono l'implementazione e permettono la scalabilità del sistema e una divisione pratica della distribuzione.



I tre nodi principali:

- **Interface Layer**: AndroidDevice

  Il nodo client è rappresentato da un AndroidDevice ospitante l'artefatto Quizy_Client:apk.

  Caratteristiche: L'applicazione è sviluppata per ambiente Android, ma l'architettura è progettata per essere agnostica rispetto al dispositivo.

  La portabilità è un aspetto importante. Grazie al fatto che il front-end è separato dalla logica applicativa, possiamo facilmente adattarlo o estenderlo ad altre piattaforme. Questo significa che non dovremo cambiare il nucleo del sistema, il che è un grande vantaggio. Il front-end può essere facilmente modificato per adattarsi a nuove esigenze, senza compromettere la stabilità del sistema. Ciò consente di essere più flessibili e di poter utilizzare il sistema su diverse piattaforme, senza dover ricominciare da zero. La separazione della logica applicativa dal front-end rende tutto più semplice e veloce.

- **Application Layer**: LinuxHost

  La logica di business risiede su un nodo LinuxHost, configurato per l'esecuzione di un ambiente containerizzato o server.

  Tecnologia: Ospita un TomCat_Container che esegue Quizy_APIRestfull, un applicativo Java enterprise.

Servizi: Questo componente espone le funzionalità del sistema tramite servizi RESTful, fungendo da unico punto di accesso per l'elaborazione dei dati e il coordinamento delle operazioni.

- **Persistence Layer**: DB_server

  Il nodo dedicato alla gestione dei dati è il DB_server, configurato per garantire la persistenza e la sicurezza delle informazioni.

  Tecnologia: Integra il DBMS MySQL, che funge da database relazionale del sistema, gestendo l'archiviazione strutturata di tutti i dati applicativi

Vincoli Architetturali

Il sistema è regolato da una norma fondamentale di sicurezza e design: il client Android ha il vincolo esclusivo di comunicare esclusivamente con le interfacce esposte da Quizy_APIRestfull.

Questo approccio garantisce che:

- Il database non sia mai esposto direttamente all'utente finale.
- Tutte le richieste devono essere controllate e validate attraverso la logica di business prima di essere elaborate.
- La logica di business deve intervenire in ogni fase di richiesta per assicurare che tutto sia corretto e conforme alle regole stabilite. In questo modo, tutte le richieste saranno validate e intermediate dalla logica di business, garantendo che i processi siano eseguiti in modo coerente e affidabile.

3.4. Persistent data management

Il Data Tier cruciale per la **manutenibilità**, **sicurezza** e **affidabilità** dell'applicazione, garantendo che le informazioni sui quiz, le risposte e le sessioni degli utenti siano conservate in modo strutturato e coerente.

La nostra applicazione di basa sull'utilizzo di un **database relazionale** per la persistenza dei dati di sistema. Il DBMS selezionato è **MySQL,** scelto per le seguenti motivazioni:

- **Robustezza e Open-Source:** È un DBMS robusto e la sua natura open-source ne facilità l'adozione e la manutenibilità a lungo termine.
- **Integrità Transazionale (ACID): Supporta** pienamente le **t**ransazioni ACID (Atomicità, Consistenza, Isolamento, Durabilità), requisito fondamentale per le operazioni critiche come l'invio finale di un quiz.

- **Prestazioni e Sicurezza**: Offre ottime prestazioni per carichi di lavoro misti (CRUD operations) e garantisce l'integrità referenziale richiesta dal modello.
- **Compatibilità**: È ampiamente compatibile con l'ecosistema tecnologico moderno utilizzato per lo sviluppo del backend.

Il sistema utilizza JPA, che è una specifica standard per la gestione della persistenza dei dati in ambiente Java, per collegare il backend applicativo con il livello di persistenza dei dati. In particolare, il sistema fa uso di Hibernate come provider di riferimento per JPA, che è una delle implementazioni più note e mature di questa specifica. Ciò consente una gestione efficiente e standardizzata dei dati all'interno del sistema. JPA e Hibernate lavorano insieme per semplificare la gestione della persistenza dei dati, offrendo una soluzione robusta e affidabile per il sistema. Il fatto che JPA sia uno standard significa che il sistema può essere facilmente integrato con altri sistemi e tecnologie che supportano questo standard, mentre Hibernate fornisce una soluzione concreta e testata per la gestione della persistenza dei dati.

L'adozione di JPA/Hibernate porta a un approccio di mappatura degli oggetti sui dati, ovvero il cosiddetto Object-Relational Mapping. Questo metodo permette di collegare gli elementi principali della nostra applicazione ai dati presenti nelle tabelle di un database MySQL. Inoltre, quando dobbiamo apportare delle modifiche al nostro modello di dati, con questo sistema è molto più facile farlo.

Dal punto di vista operativo, l'ORM astrae la scrittura manuale delle query SQL, migliorando la manutenibilità del codice e riducendo la probabilità di errori. Hibernate fornisce inoltre meccanismi di caching che permettono di ottimizzare le prestazioni, limitando l'accesso diretto al database nelle operazioni ripetitive di lettura.

La sicurezza è un aspetto molto importante. Quando si utilizza JPA, esso si avvale di JPQL e query parametrizzate. Queste query vengono poi tradotte in Prepared Statement JDBC. Ciò aiuta a ridurre il rischio di SQL Injection, perché i parametri delle query non vengono uniti in modo dinamico. Invece, vengono gestiti in modo sicuro dal layer JDBC sottostante.

In sintesi, l'utilizzo di JPA con Hibernate consente di ottenere un livello di persistenza robusto, sicuro e facilmente estendibile, migliorando sia l'efficienza operativa del sistema sia la qualità complessiva del codice.

3.5. Access control and security

3.5.1. Access Matrix

Quizy implementa meccanismi di autenticazione e protezione:

- **Autenticazione**: L'accesso iniziale si basa sull'accoppiata Username/Password, che garantisce l'accesso alle operazioni permesse. Sono definiti i seguenti ruolo principali, che determinano l'accesso ai sottosistemi.
- **Player**: Il Player e colui che può partecipare alle sessioni di quiz.
- **Creatore quiz**: Il Creatore quiz detiene il controllo sui quiz; Può creare, modificare ed eliminare i quiz.
- **Gestore Account**: Gestore account è l'entità che effettua controlli e fornisce lo status a un utente di "Creatore quiz"

Ruolo	**Quiz**	**Domande/Risposte**	**Utente (profilo)**	**Ticket (supporto)**	**Ruolo Utente**
Player	Visualizza, esegue	Visualizza	Visualizza, modifica	Crea	Nessun accesso
Creatore quiz	Crea, visualizza, modifica, elimina	Crea, visualizza, modifica, elimina	Visualizza, modifica	Crea	Nessun accesso
Gestore account	Nessun accesso	Nessun accesso	Nessun accesso	Visualizza, modifica	Modifica

Il sistema di Quizy implementa un modello di accesso alle risorse basato su ruoli **(Role-Based Access Control),** ciò garantisce che le operazioni sugli oggetti siano limitate in base al ruolo assegnato.

3.5.2. Security Strategy

In accordo con i requisiti non funzionali quizy implementa una serie di strategie atte a mitigare i principali problemi noti:

1. Data Security:
    a. Query **injection protection:** per prevenire possibili manipolazioni illecite utilizzeremo white list di controllo (per query native) è preparateStatment di JDBC (base di JPA) JPQL che impone query sicure.

    b. Password Hashing: per sicurezza di accesso e il sistema conserva nel Database tutte le password con una funzione di hashing BCrypt rendendo difficile la decriptazione di quest' ultima e proteggendolo dalla brut force.
2. Access Security:
    a. Token block: Tramite appositi token cifrati blocchiamo accessi anomali ai ruoli al di fuori delle possibilità dell'account usati in tutta la comunicazione col sistema (al di fuori dell'autenticazione).
    b. Access Filter: attraverso ad una gestione dei ruoli e ad un mantenimento dei ruoli designati nella persistenza, in modo da garantire l'accesso al ruolo corretto.

3.6. Global software control

Il sistema utilizza un modello di controllo event-driven per gestire le comunicazioni e le interazioni tra i vari sottosistemi. Le richieste sono inviate dall'interfaccia utente e da eventi interni al sistema, garantendo un flusso coerente e sincronizzato tra i diversi componenti. Questo tipo di approccio assicura che le operazioni siano gestite in modo efficiente e che i dati siano coerenti, anche in presenza di operazioni concorrenti.

3.7. Boundary condition

> Inizializzazione

Nome caso d'uso: Start System

Attori: Tecnico

Flusso di eventi:

- Il tecnico installa il sistema sulla macchina di lavoro
- Viene configurato il container
- All'avvio il sistema di Quizy si collega al database da remoto

> Terminazione

Nome caso d'uso: Shut down

Attori: Sistema

Flusso di eventi:

- Il Sistemista isola un modulo (ad esempio un'istanza di servizio)
- Il modulo isolato blocca l'accettazione di nuove richieste
- Il modulo rilascia le risorse e si spegne

Terminazione: Prima di terminare, il sottosistema deve assicurarsi di committare (rendere permanenti) tutte le modifiche

> Fallimento

Nome caso d'uso: Failure

Attori: Sistema di riavvio automatico

Flusso di eventi:

- Un modulo fallisce
- Il sistema effettua la chiusura di tutti i servizi gestiti in start up
- Le sessioni utente attive vengono automaticamente reindirizzate alla nuova istanza operativa.

Terminazione: Il modulo specificato viene disattivato senza interrompere il servizio globale per gli utenti rimanenti.

4. Subsystem services

4.1

Servizio:

- Autenticazione e Gestione utente

Descrizione:

- Fornisce le funzionalità per identificare gli utenti e gestire il loro ciclo di vita all'interno dell'applicazione. Si occupa dell'accesso sicuro e del recupero delle credenziali

Operazioni coinvolte:

- Login
- Logout
- Recupero password

4.2

Servizio:

- Gestione quiz

Descrizione:

- Rappresenta il cuore del sistema. Permette ai creatori di costruire i quiz e ai player di svolgerli. Gestisce tutto il ciclo di vita del quiz, dalla creazione alla visualizzazione dei risultati.

Operazioni coinvolte:

- Creazione nuovo quiz
- Eliminazione quiz
- Visualizzazione quiz
- Svolgimento/Completamento quiz

4.3

Servizio:

- Amministrazione account

Descrizione:

- Fornisce le funzionalità dedicate al ruolo "Gestore Account". Permette la gestione dei privilegi amministrativi.

Operazioni coinvolte:

- Autenticazione gestore
- Cambio ruolo utente

4.4

Servizio:

- Gestione richieste e token

Descrizione:

- Gestisce il sistema di ticket/token utilizzato per le richieste di supporto tecnico o per le richieste di elevazione di ruolo (cambio ruolo).

Operazioni coinvolte:

- Creazione token richiesta
- Gestione/Chiusura richiesta

5. Glossario

- **Quiz:** Serie di domande a risposta multipla, finalizzate alla valutazione delle conoscenze dell'utente.
- **Utente**: Qualsiasi persona registra nel sistema, che può accedere all'app per partecipare ai quiz o visualizzare i risultati.
- **API:** interfaccia software che gestisce la comunicazione tra l'app client e il server dell'applicazione.
- **Modulo**: Con questa parola che appare nel punto 3.7 ci riferiamo a una istanza del servizio, ovvero, un oggetto in esecuzione su un server ma traslato al livello di un sottosistema.
- **JDBC (Java Data Base Connector): è** un' API standard che permette all'applicazione scritta in Java di connettersi e interagire con il database.
- **JPA (Java Persistence API)**: connettore per il DB basato su JDBC.