**­KEY-KNOWLEDGE**



# Nome Documento

Problem Statement Version 1.4.2

**SOMMARIO**

[**KEY-KNOWLEDGE** 1](#_Toc65680063)

[Nome Documento 1](#_Toc65680065)

[1. TABELLA REVISIONI 3](#_Toc65680066)

[2. Design Goals 4](#_Toc65680067)

[3. Trade-off: 5](#_Toc65680068)

[4. Rappresentazione delle classi del sistema 5](#_Toc65680069)

[5. Architettura del Sistema proposto 6](#_Toc65680070)

[6. Decomposizione in sottosistemi 7](#_Toc65680071)

[7. Mapping hardware/software 9](#_Toc65680072)

[8. Rappresentazione UML](#_Toc65680073) [del DataBase noSQL proposto 10](#_Toc65680074)

[9. Controllo degli accessi e sicurezza 10](#_Toc65680076)

[10. System Control 12](#_Toc65680077)

[11. Servizi dei sottosistemi 13](#_Toc65680078)

[11.1 Gestione Account 13](#_Toc65680079)

[11.2 Gestione Domande 14](#_Toc65680105)

[11.2 Gestione Partite 14](#_Toc65680121)

[11.3 Gestione Report 15](#_Toc65680137)

# ****TABELLA REVISIONI****

|  |  |  |  |
| --- | --- | --- | --- |
| **DATA** | **VERSIONE** | **DESCRIZIONE** | **AUTORE** |
| **17/10/2020** | **1.0** | **Prima versione del problemStatemant** | **Crescenzo Manzone**  **Franco Nicola Fernando**  **Giovanni Battista Mercurio** |
| **23/10/2020** | **1.1** | **Modifica scenari** | **Crescenzo Manzone** |
| **24/10/2020** | **1.2** | **Modifica Requisiti Funzionali** | **Franco Nicola Fernando** |
| **25/10/2020** | **1.3** | **Modifica Requisiti non Funzionali** | **Giovanni Battista Mercurio** |
| **27/10/2020** | **1.4** | **Unione delle modifiche effettuate** | **Crescenzo Manzone**  **Franco Nicola Fernando**  **Giovanni Battista Mercurio** |
| **6/11/2020** | **1.4.2** | **Revisione requisiti** | **Crescenzo Mazzone** |

# Design Goals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **CATEGORIA** | **Nome** | **Descrizione** | **Priorità** |
| DG\_1 | Criteri di End User: | Usabilità | Il Sistema dovrà avere un’interfaccia funzionale e di facile comprensione. L’interfaccia dovrà adattarsi ad ogni device di ogni utente , includendo widget di facile utilizzo. | Alta |
| DG\_2 | Criteri di Affidabilità: | Attendibilità | Il Sistema garantirà l’attendibilità, intesa come correttezza e affidabilità dei dati. | Alta |
| DG\_3 | Criteri di Affidabilità: | Disponibilità | Il Sistema dovrà essere permanentemente fruibile dagli utenti. | Alta |
| DG\_4 | Criteri di Affidabilità: | Sicurezza | Ogni Utente è dotato di nickname e password con cui poter effettuare l’accesso al Sistema e utilizzare solo le funzionalità associate alla propria categoria. | Bassa |
| DG\_5 | Criteri di Affidabilità: | Tolleranza ai guasti | Ogni guasto parziale non influenzerà l’intera applicazione, se una funzionalità non è disponibile , l’utente ne verrà a conoscenza solo se la richiederà. | Bassa |
| DG\_6 | Criteri di Performance: | Tempi di risposta | Il prodotto Software deve consentire una partita con tempi di attesa brevi ai vari utenti, le domande saranno caricate sul client all’inizio di ogni partita, per ottimizzare le prestazioni. | Bassa |
| DG\_7 | Criteri di Performance: | Memoria | Il Sistema memorizzerà uno storico delle risposte nel client di ogni utente. Le domande di ogni partita vengono scaricate dal DataBase e cancellate nel momento in cui termina la partita. | Bassa |
| DG\_8 | Criteri di Manutenibilità: | Modificabilità | Il Sistema dovrà essere facilmente modificabile. | Alta |
| DG\_9 | Criteri di Manutenibilità: | leggibilità | Il sistema dovrà essere facilmente comprensibile leggendo il codice.Ci dovranno essere opportuni commenti in tutti I file di codice. | Alta |
| DG\_10 | Criteri di Manutenibilità: | Tracciabilità | Il codice sarà suddiviso in porzioni in base ai requisiti. | Alta |

# Trade-off:

* **Attendibilità vs Tempi di risposta**

Il Sistema sarà implementato in modo tale da preferire il tempo di risposta all’attendibilità, in modo tale

da garantire attese brevi agli utenti.

* **Disponibilità vs Tolleranza ai guasti**

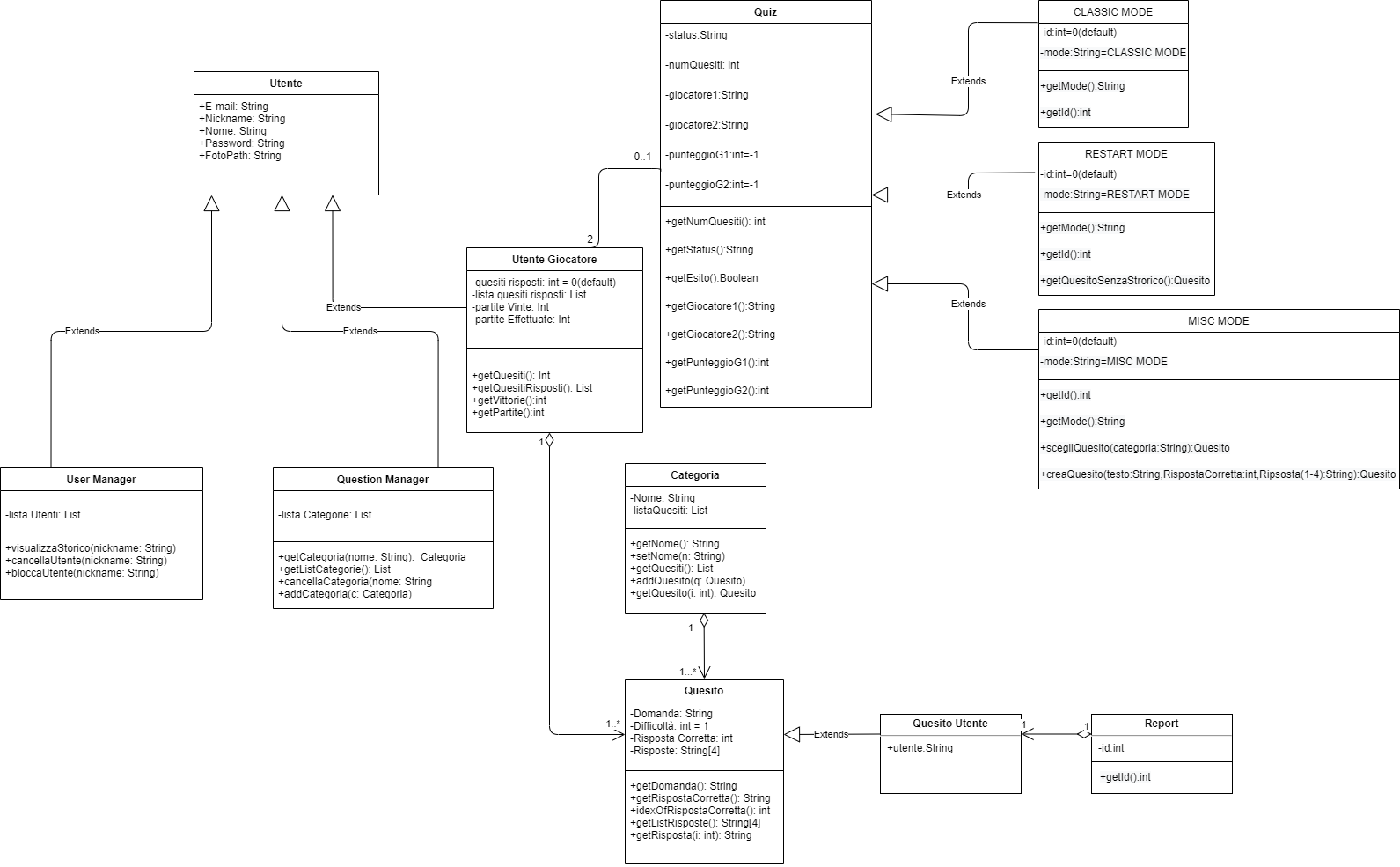
Il Sistema deve sempre essere disponibile all’Utente in caso di errore in una funzionalità, anche al

costo di rendere non disponibile quest’ultima per un lasso di tempo.

* **Criteri di Manutenibilità vs Criteri di Performance**

Il Sistema sarà implementato preferendo la manutenibilità ai tempi di risposta in modo da facilitare gli sviluppatori nel processo di aggiornamento del software a discapito delle performance del sistema.

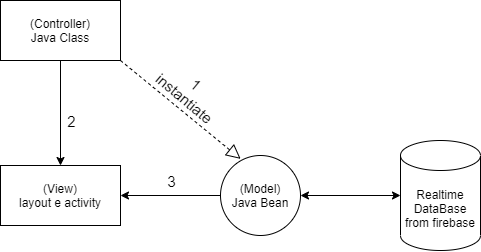
# Rappresentazione delle classi del sistema



# 

# Architettura del Sistema proposto

**Panoramica**

Il pattern architetturale scelto è MVC (Model, View, Control).

**Model:** contiene i le classi DAO con i rispettivi getters e setters.

**View:** si occupa di visualizzare i dati all’Utente e gestisce l’interazione fra quest’ultimo e l’infrastruttura

sottostante.

**Controller:** riceve i comandi dell’Utente attraverso il View e reagisce eseguendo delle operazioni che possono interessare il Model e che portano generalmente ad un cambiamento di stato del View.

# Decomposizione in sottosistemi

Il Sistema è stato diviso in quattro sottosistemi principali:

* **Gestione Account**

Il Sottosistema ***Gestione Account*** gestisce l’autenticazione dell’Utente, in particolare le funzioni di Login, Logout, Recupero Password e Registrazione. L’utente può accedere alla sua area personale e visualizzare i suoi dati, in particolare: nickname, nome, cognome, email, foto. L’utente inoltre può accedere alla sua area KNOWLEDGE che gli permette di visualizzare i dati relativi alle domande risposte nel corso delle sue partite. Inoltre l’UserManager può sospendere o eliminare utenti accedendo alla sua area gestionale.

* **Gestione Domande**

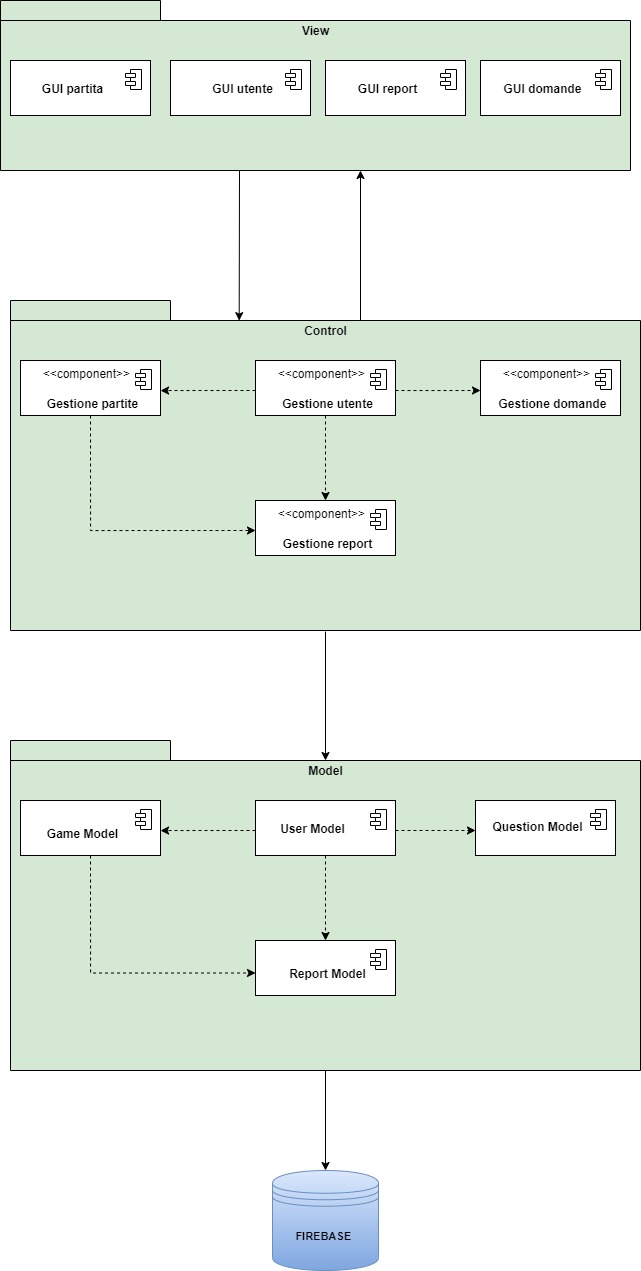
Il Sottosistema ***Gestione Domande*** gestisce l’elenco di domande, ogni domanda può essere cancellata o modificata accedendo alla relativa categoria dal Question Manager (inserendo testo domanda, difficoltà, risposte (4), risposta esatta). Possono essere aggiunte nuove categorie con un nome univoco.

* **Gestione Partite**

Il Sottosistema ***Gestione Partite*** gestisce la creazione di partite da parte degli utenti. Ogni partita si svolge tra due utenti. Ogni partita deve contenere gli identificativi dei due utenti coinvolti, chi ha vinto la partita e la sua modalità (MISC, RESTART, CLASSIC). Inoltre si gestisce la possibilità da parte degli utenti di terminare anticipatamente la partita.

* **Gestione Report**

Siccome ogni utente ha la possibilità di creare domande nella modalità MISC è stato introdotto il sottosistema Gestione Report. Ogni utente che risponde a una domanda creata può farne il report semplicemente cliccando un pulsante per segnalare l’utente avversario. Il Report viene gestito dall’UserManager che lo visualizza e può sospendere o eliminare l’utente che ha generato la domanda.



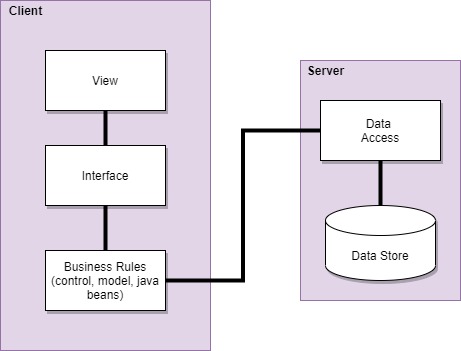
# 

# 

# Mapping hardware/software

Il sistema proposto sarà sviluppato come un’applicazione android che si connette a un server remoto per la memorizzazione e il recupero dei dati nel Database. Si utilizza un’architettura client/server dove un server fornisce servizi a più client. Su un dispositivo android è eseguita un’applicazione client che consente all’utente di interagire attraverso il livello View inviando richieste attraverso il livello Control e Model al server. La comunicazione tra applicazione client android e server avviene tramite json. Le specifiche hardware e software necessarie per il client sono avere un dispositivo con sistema operativo android che può connettersi alla rete. Per la parte server, invece, il client accede a un server firebase con un database. Quest’ultimo non è altro che un servizio offerto da Google, quindi la parte server è gestita dal servizio utilizzato. Le specifiche software per il server comprendono un Database noSql per la gestione dei dati persistenti.

La struttura hardware/software del sistema è di tipo **FAT-CLIENT**: il client dispone dei meccanismi di visualizzazione dei dati (View), le interfacce relative alla logica di business e il modello MVC. Il Server è caratterizzato esclusivamente dal Data store, nel caso specifico è un database noSql ottenuto come servizio dal cloud provider Google al quale accediamo utilizzando le API di Android.



# 

# Rappresentazione UML del DataBase noSQL proposto

# 

# Controllo degli accessi e sicurezza

Nel sistema “Key-Knownledge” è previsto un controllo dei dati forniti dall’utente in fase di registrazione e di autenticazione.

**In fase di registrazione**: l’utente inserisce le credenziali che desidera (**nickname, password**) e inoltre un **e-mail** valida, alla quale verrà inviato un messaggio con un link mediante il quale ha la possibilità di confermare il proprio account.

**In fase di autenticazione:** Se l’utente ha confermato il proprio account tramite e-mail, può autenticarsi ed entrare nell’app inserendo il proprio nickname e la password.

Nel momento in cui avviene l’autenticazione, con il corretto utilizzo di nickname e password, l’utente accede all’area HOME dell’applicazione contenente le specifiche funzioni da poter effettuare.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attore** | **Gestione Account** | **Gestione domande** | **Gestione partite** | **Gestione report** |  | |
| **Utente non**  **Registrato** | * Registrazione * Log-in | NA | NA | NA |  | |
| **Utente Giocatore** | * Log-out * Recupero password * Modifica password * Aggiunta foto | NA | * Crea partita * Crea domanda * Risposta domanda * Scegli domanda | * Creazione report | |  |
| **User Manager** | * Log-out * Cancella utente * Sospendi utente | NA | NA | NA |  | |
| **Question Manager** | * Log-out | * Aggiungi categoria * Cancella categoria * Aggiungi domanda * Cancella domanda | NA | NA |  | |

# System Control

* 1. **Controllo flusso globale del sistema**

Il controllo del flusso globale è di tipo event-driven. Le richieste vengono generate da un client, e dalle classi preposte a gestire quel determinato evento associato alla richiesta. In particolare, il click sullo schermo dello smartphone genera un evento, il quale viene catturato nella View (Activity). A questo punto, la View inoltra la richiesta all’oggetto Control che lo invia al manager di competenza. Il Server remoto (real-time database) attende le richieste dei client generate dalle opportune classi manager. Una volta ricevuta la richiesta, la processa.

* 1. **Condizioni limite**

Il Server dovrà essere sempre attivo, permettendo l’utilizzo continuo dei servizi. In caso di malfunzionamento di una funzionalità, il sistema continuerà a funzionare normalmente. Nel caso in cui l’utente richiede quella specifica funzionalità, gli viene mostrato un messaggio di avviso. Nel caso di un crash del Sistema dovuto ad un errore, si effettuerà un ripristino al backup più recente. Il sistema accede a un server remoto fornito come servizio cloud da Google e non deve essere gestito nell’avvio e nello spegnimento.

# Servizi dei sottosistemi

# Gestione Account

|  |  |
| --- | --- |
| **Sottosistema** | **Gestione account** |
| **Descrizione** | Questo sottosistema permette di effettuare le operazioni relative agli utenti. |
| **Servizi offerti** | |
| **Servizio** | Descrizione |
| Log-in | Servizio di autenticazione al sistema |
| Login-DEMO | Servizio di autenticazione al sistema semplificato |
| Log-out | Servizio di uscita dal sistema |
| Registrazione | Servizio che permette a un utente ospite di effettuare la regitrazione al sistema |
| Recupera password | Servizio che consente di recuperare la password a un utente |
| Modifica password | Servizio che consente di modificare la password a un utente |
| Conferma utente | Servizio che consente di confermare la registrazione al sistema da parte di un utente ospite |
| Cancella utente | Servizio che consente al gestore degli utenti di cancellare un utente dal sistema. |
| Sospendi utente | Servizio che consente al gestore degli utenti di sospendere un utente dal sistema. |

# Gestione Domande

|  |  |
| --- | --- |
| **Sottosistema** | **Gestione domande** |
| **Descrizione** | Questo sottosistema permette di effettuare le operazioni relative alle domande di gioco. |
| **Servizi offerti** | |
| **Servizio** | Descrizione |
| Aggiungi categoria | Servizio per aggiungere una categoria di domande. |
| Cancella categoria | Servizio per cancellare una categoria di domande. |
| Aggiungi domanda | Servizio per aggiungere una domanda di una specifica categoria. |
| Cancella domanda | Servizio per cancellare una domanda di una specifica categoria. |

# Gestione Partite

|  |  |
| --- | --- |
| **Sottosistema** | **Gestione partite** |
| **Descrizione** | Questo sottosistema permette di effettuare le operazioni utili durante una partita. |
| **Servizi offerti** | |
| **Servizio** | Descrizione |
| Crea partita | Servizio che permette a un utente giocatore di abbinarsi con un altro utente giocatore e creare una partita. |
| Crea partita-DEMO | Servizio che permette a un utente giocatore di abbinarsi con un altro utente giocatore e creare una partita nella sola modalità RESTART. |
| Crea domanda | Servizio che permette a un utente giocatore di creare una domanda durante una partita. (Nel caso in cui ha selezionato la modalità misc-mode) |
| Risposta domanda | Servizio che permette a un utente giocatore di rispondere a una delle domande proposte. |
| Risposta domanda-DEMO | Servizio che permette a un utente giocatore di rispondere a una delle domande proposte.Il sistema non controlla il tempo impiegato. |
| Scegli domanda | Servizio che permette a un utente giocatore di scegliere le domande di una specifica categoria, da sottoporre all’avversario. |

# Gestione Report

|  |  |
| --- | --- |
| **Sottosistema** | **Gestione report** |
| **Descrizione** | Questo sottosistema permette di gestire i report dovuti a segnalazioni da parte degli utenti. |
| **Servizi offerti** | |
| **Servizio** | Descrizione |
| Creazione report | Servizio che permette a un utente giocatore di segnalare un altro utente. Viene creato un report e poi il gestore degli utenti rileva la segnalazione decidendo se bloccare/eliminare l’utente dal sistema. |