**­KEY-KNOWLEDGE**



# Nome Documento

Problem Statement Version 1.4.2

**SOMMARIO**

[**KEY-KNOWLEDGE** 1](#_Toc62400045)

[Nome Documento 1](#_Toc62400047)

# 

# ****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 din attesa brevi ai vari Utenti, quindi, le domande saranno caricate sul client all’inizio di ogni partia, 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\_12 | 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

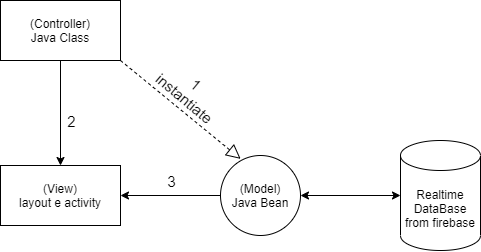
costo di rendere non disponibile quest’ultima per un lasso di tempo.

### 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.

# 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 tre 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

Poiché il Sistema proposto sarà sviluppato come estensione di “English Validation” e poiché si è deciso di non effettuare un re-engineering di tale sistema per non eccedere sia in costi che effort di tutto il team, al fine di preservare le funzionalità già esistenti, si è preferito conservare il mapping hardware/software del suddetto Sistema (documento: “EV\_SDD”, paragrafi 3.3.2 e 3.3.3).

Il Sistema, che si desidera sviluppare, utilizza un’architettura Client/Server, in cui un Server fornisce servizi a più client. Su una macchina client è eseguito un web browser che consente all’Utente di interagire con il livello View (lato Server) per inoltrare richieste e visualizzare le risposte ricevute. Il System gestisce la logica applicativa, mentre la macchina "DataBase Server" gestisce i dati persistenti. La comunicazione tra Client e Server avviene tramite protocollo HTTP.

HTTP è un protocollo di trasferimento di ipertesti che consente a due macchine, client e Server, di interagire attraverso un meccanismo di richiesta/risposta. Il Client inoltra una richiesta al Server, che verrà soddisfatta con la risposta di quest’ultimo.

Le specifiche hardware e software necessarie per il Client sono rispettivamente una macchina dotata di connessione a Internet e un sistema operativo con un web browser installato.

Le specifiche hardware necessarie per il Server consistono in una macchina connessa a Internet, la quale sia capace di immagazzinare grandi quantità di dati.

Le specifiche software necessarie comprendono un Database Management System (*MySQL*) per la gestione dei dati persistenti, un System per la gestione della logica applicativa e della comunicazione con più client.