**La ruota della fortuna**

STUDENTI

Ciapponi Alessandro (numero di matricola 733433)

Cremonesi Luca (numero di matricola 733348)

Peluso Andrea (numero di matricola 732530)

Valenti Dezsö (numero di matricola xxxxxx)



INDICE

INTRODUZIONE 4

LE REGOLE DEL GIOCO 5

I REQUISITI 8

* Requisiti Funzionali 8
* Requisiti Non Funzionali 8

ANALISI FUNZIONALE 9

* Avvio 9
* Registrazione Giocatore / Osservatore 10
* Autenticazione 11
* Monitoraggio di una Partita da parte di un Osservatore (o del Giocatore) 14
* Abbandono di una partita 15
* Modifica dei Dati di Profilo 16
* Reset di Password 16
* Gestione delle frasi misteriose 16
* Monitoraggio e Analisi 17

CLIENT/SERVER 17

* Client 17
* Server 27
* Iterazione 29
* Design Pattern Observer 30

PIANO DI PROGETTO 32

REALIZZAZIONE DEL DATABASE 33

* Progettazione Logica Iniziale 35
* Progettazione Logica Finale 36

LA CODIFICA 42

CODICE DEL DATABASE 42

* Script Necessari alla Creazione del Database 42
* Query SQL a Supporto dei Servizi Erogati da Rdf 46
  + Classe DBimplementation 46
  + Classe GameQuery 46
  + Classe LoginQuery 54
  + Classe PhraseQuery 55
  + Classe ProfileUpdateQuery 57
  + Classe RegistrationQuery 58
  + Classe StatisticsQuery 59

DOCUMENTAZIONE DEL CODICE 66

APACHE ANT 66

TEAM DI SVILUPPO 68

FONTI 68

## **Introduzione**

Il progetto RdF (Ruota della Fortuna) ha avuto come obiettivo la realizzazione di una piattaforma di gioco multiplayer ispirata all’omonimo celebre quiz televisivo a premi italiano andato in onda agli inizi degli anni ’90 (basato a sua volta sul format “Wheel of Fortune” dell'americana NBC).

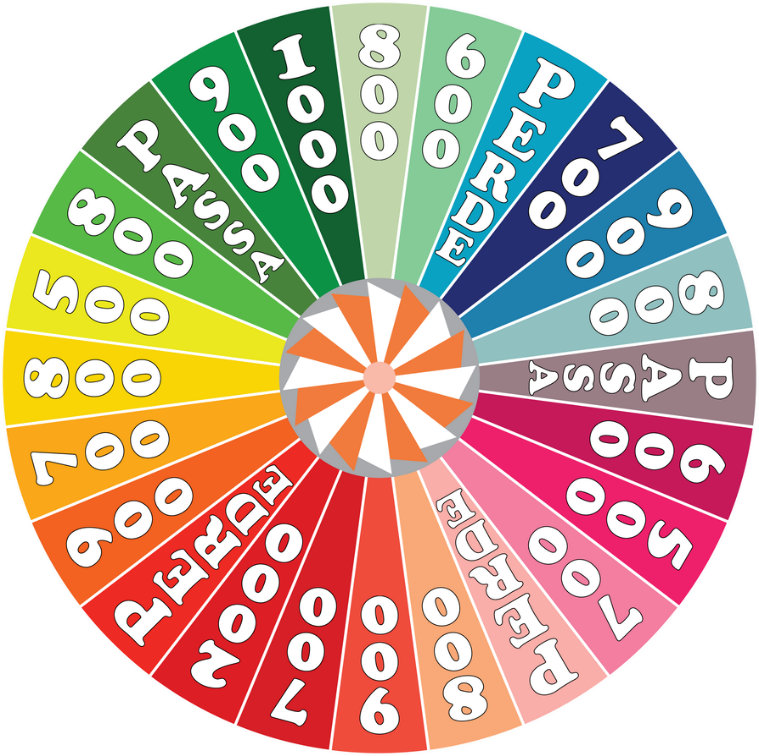
Tale obiettivo è stato raggiunto rispettando, oltreché le prescritte **specifiche funzionali** (implementazione delle corrette regole di gioco) **e tecniche** (scelta del linguaggio di programmazione, [Java] del sistema di gestione dei database [PostgreSQL] gestione del driver di connessione al database [JDBC], possibilità di accesso contemporaneo di più utenze alle stesse risorse condivise), *anche* la **conformità alle metodologie** di progettazione e sviluppo utilizzate in Università (documentazione progettuale con diagrammi UML e modelli Entity-Relationship; utilizzo di design patterns, etc.).

La piattaforma è stata sviluppata in modalità server (per la gestione sistemistica dei database e di alcune funzioni amministrative) e in modalità client (distribuita ai giocatori) e si presenta attraverso un’opportuna interfaccia grafica. Le utenze sono profilate su tre diversi livelli autorizzativi

1. Giocatore
2. Supervisor (Admin)
3. Amministratore di Sistema

## **LE regole del gioco**

Il gioco è strutturato in 5 manche, a cui partecipano 3 concorrenti il cui obiettivo è indovinare una frase misteriosa (su un argomento prefissato) presente in un tabellone in cui tutte le lettere sono state inizialmente oscurate.

All'inizio della manche il primo concorrente (nella prima manche scelto da un generatore casuale) “gira la ruota”, cioè fa partire una roulette (nel gioco televisivo era un dispositivo meccanico a forma di ruota con il disco suddiviso in tanti settori colorati) che si ferma su una casella recante un messaggio casuale.

Il messaggio può rappresentare un jolly (utile per evitare di perdere il turno di gioco); un comando (“Passa” o “Perde”) o un premio in punti. Il comando “Passa” costringe il giocatore a cedere il turno di gioco al concorrente alla sua destra; il comando “Perde” invece, azzera il montepremi guadagnato dal giocatore *in quella manche.*

Se il messaggio invece rappresenta un premio in punti il concorrente deve “chiamare una consonante”, cioè cercare di indovinare (entro 5 secondi) una consonante che secondo lui è presente nella frase misteriosa. Se la consonante chiamata è effettivamente presente nella frase misteriosa, le caselle che nel tabellone celano la consonante richiesta si rivelano e il concorrente accumula in un suo portafoglio virtuale il premio indicato dalla ruota (da 300 a 1000 punti) moltiplicato per il numero di volte che quella consonante è presente nella frase.

Dopo aver individuato la consonante corretta il giocatore può ancora “girare la ruota” (ripetendo l’esperienza precedente) o tentare di indovinare la frase nascosta. Altra possibilità è di “comprare una vocale”, cioè togliere dal proprio portafoglio virtuale (se si hanno fondi a sufficienza) 1000 punti per far scoprire, nella frase misteriosa, le celle che nascondono la vocale prescelta.

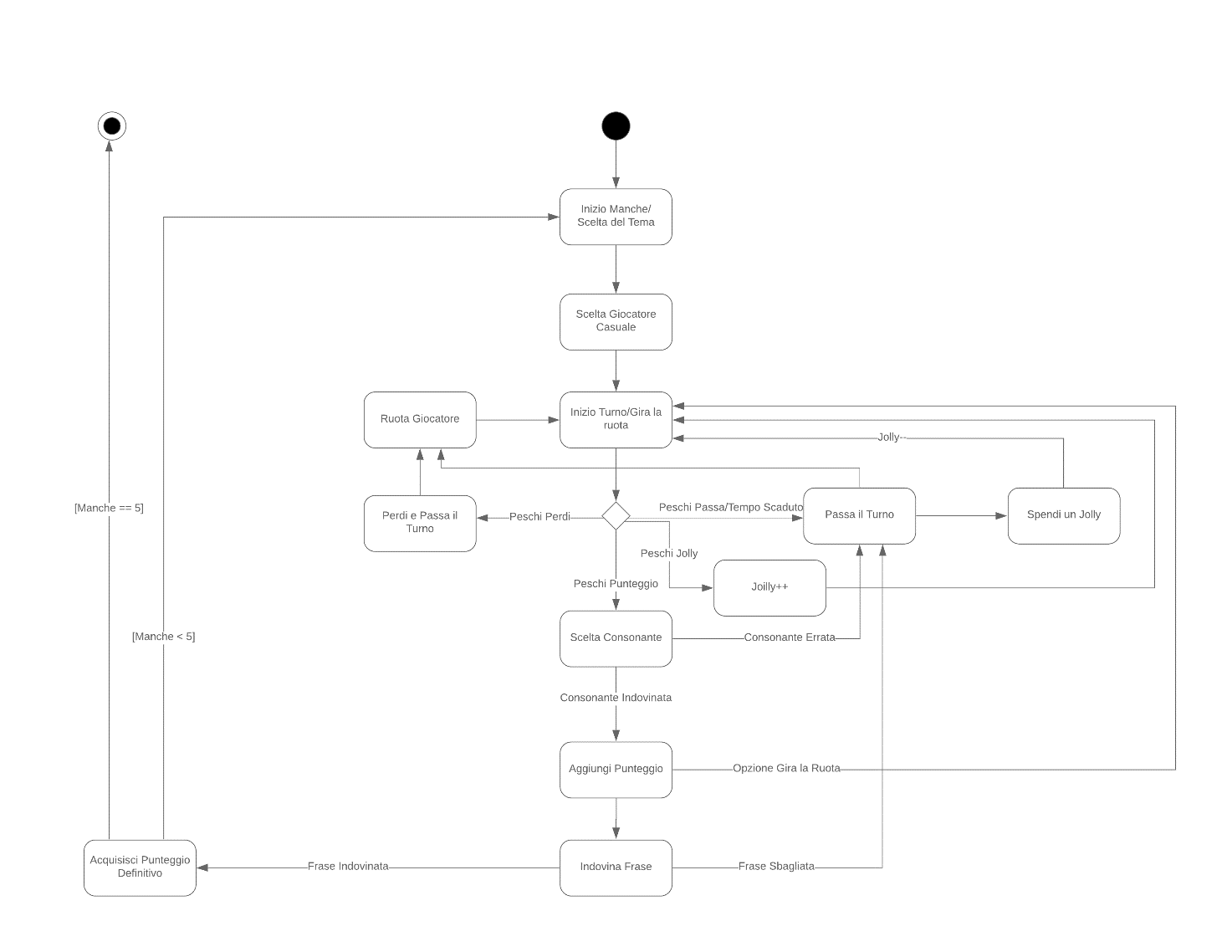
Se la lettera chiamata (vocale o consonante) NON è presente nella frase si perde il turno e il gioco passa a un altro giocatore. Si può evitare il passaggio del turno sacrificando un “jolly” eventualmente accumulato in precedenza (anche in una manche precedente).

Quando tutte le consonanti della frase sono state scoperte tutti i giocatori vengono avvisati e il concorrente di turno può soltanto tentare di indovinare la frase (o comprare una vocale e far “girare” nuovamente la ruota). Se il concorrente non indovina la frase, il turno passa al giocatore successivo. La manche finisce quando uno dei tre giocatori indovina la frase. I punti di manche accumulati dal vincitore sono messi in un deposito; agli altri due giocatori invece i punti di manche sono azzerati. Il vincitore del gioco è chi, al termine delle 5 manche, ha accumulato più punti in deposito.

Nell’applicazione informatizzata, ogni giocatore può dunque associato alle seguenti azioni

* “girare la ruota”
* “prendere un jolly”
* “passare la mano”: cedere al giocatore alla sua destra il turno di gioco
* “chiamare una consonante”: indovinare quale consonante è contenuta nella frase e guadagnare l’importo relativo [punti del premio moltiplicato per il numero di volte in cui la consonante è presente nella frase misteriosa]
* “comprare una vocale”: indovinare quale vocale è contenuta nella frase e pagare 1000 punti, prelevandoli dal portafoglio virtuale di manche.
* “giocare un jolly”: sacrificare il Jolly eventualmente accumulato, per evitare di dover passare il turno a un altro giocatore.
* “perdere tutto”: azzerare quanto si è guadagnato sino a quel punto nella manche
* “dare la soluzione”, cioè provare a indovinare (entro 10 secondi) la frase di gioco, congelando quanto sino a quel momento guadagnato nel proprio borsellino elettronico.

Le azioni sopra descritte sono formalizzate dal seguente diagramma di stato



## **I requisiti**

### I Requisiti Funzionali

Le regole del gioco appena presentate sono state una buona base di partenza per stilare una prima lista di requisiti funzionali per la realizzazione di una piattaforma applicativa che simuli il gioco: "La ruota della fortuna" (oggetto di questo progetto). Altri requisiti, pur non espressamente dichiarati, sono impliciti nella costruzione del gioco che altrimenti non potrebbe funzionare o funzionerebbe in maniera non professionale.

Dovrà essere realizzata in modo tale da:

* supportare l'interazione in parallelo con più utenti connessi alla piattaforma da postazioni differenti
* dotarsi di una accattivante interfaccia grafica
* consentire la possibilità di resettare il gioco
* ritirarsi
* salvare i risultati parziali per riprendere la volta successiva
* registrare il massimo punteggio raggiunto (personale e assoluto)
* …

### I Requisiti Non Funzionali

Ai requisiti funzionali (espliciti e impliciti) si affiancano requisiti tecnici necessari all'implementazione delle funzionalità richieste e al rispetto di alcune proprietà del sistema che devono essere soddisfatte (perché richieste dal committente o perché imposte dalle risorse a disposizione. Nel nostro progetto:

* le applicazioni devono essere scritte in Java,
* il sistema di gestione dei database deve essere PostgreSQL,
* il driver di connessione al database deve essere JDBC
* il programma dovrebbe essere abbastanza veloce nell’elaborazione risultato/risposta
* il programma deve essere in grado di interagire con il Database per potere salvare i dati degli utenti
* …

Da tali requisiti discendono tutte le altre fasi del progetto: analisi funzionale, analisi tecnica, sviluppo e collaudo del software, test di sistema e di integrazione, documentazione utente.

La nostra piattaforma di gioco (“Ruota della Fortuna”, d’ora in poi “RdF”) dovrà e potrà essere utilizzata da utenti con 3 differenti profili:

* Profilo “giocatore”: utente che partecipa attivamente al gioco
* Profilo “osservatore”: utente che non partecipa attivamente al gioco ma segue l’andamento delle partite altrui come semplice spettatore
* Profilo “amministratore”: utente privilegiato che gestisce i parametri di gioco (ad esempio si incarica di scegliere la frase misteriosa)

La gestione tecnica di questi profili è affidata a diversi moduli, fra cui

* modulo “**playerRdF**” (client) che fornisce servizi designati per i giocatori e per gli osservatori
* modulo “**adminRdF**” (client) che fornisce servizi di gestione della piattaforma RdF per utenti amministratori.

Un altro modulo “**serverRdF**” (server) si occuperà di gestire i servizi di gestione e manutenzione dell’applicazione (servizi di back-end) e del database ad essa collegata (nel nostro caso utilizzando “PostgreSQL”)

La realizzazione dei servizi offerti richiede una opportuna suddivisione di funzionalità tra le componenti, e la definizione di protocolli di interazione tra queste parti.

## 

## **Analisi funzionale**

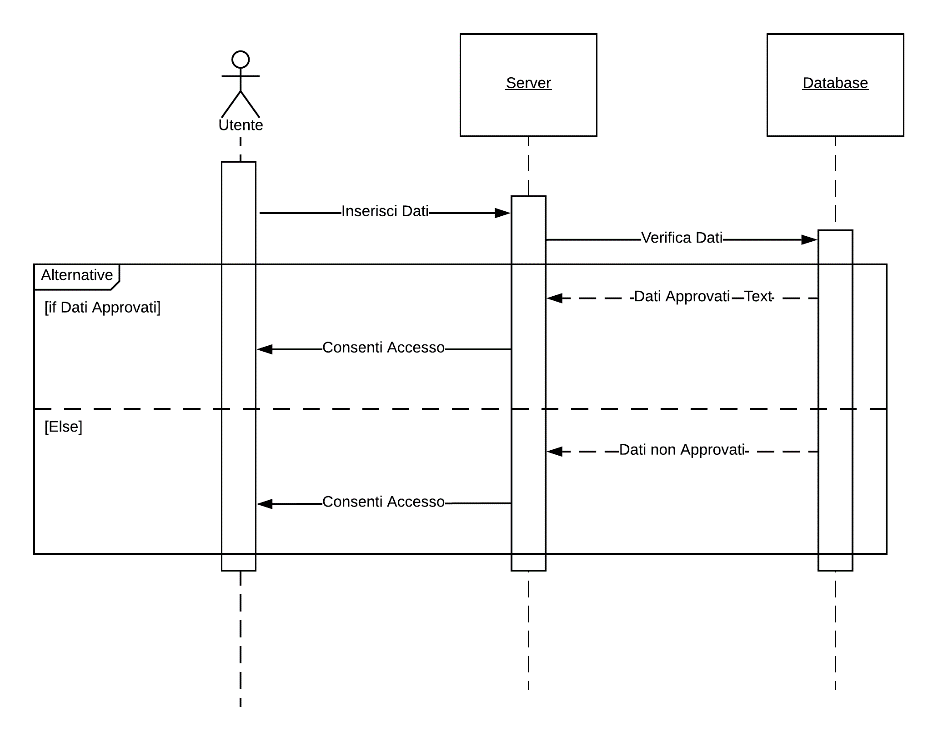
### Avvio

All’invocazione di “serverRdF”, il modulo dovrà richiedere di specificare le credenziali di accesso a dbRdF (database di supporto all’esecuzione dei servizi della piattaforma RdF) e l’identificativo dell’host del database.

Se nel database non è presente un profilo di utente amministratore, il modulo “serverRdF” dovrà avviare un processo di registrazione delle credenziali amministrative, chiamando il modulo “adminRdf” predisposto per questa funzione.

“adminRdf” richiederà all’utente che vuole essere censito come amministratore: nome, cognome, un nickname, una password e una casella di posta. Password ed email costitueranno, da quel momento, le credenziali di accesso necessarie per autenticare l’utente con diritti amministrativi.

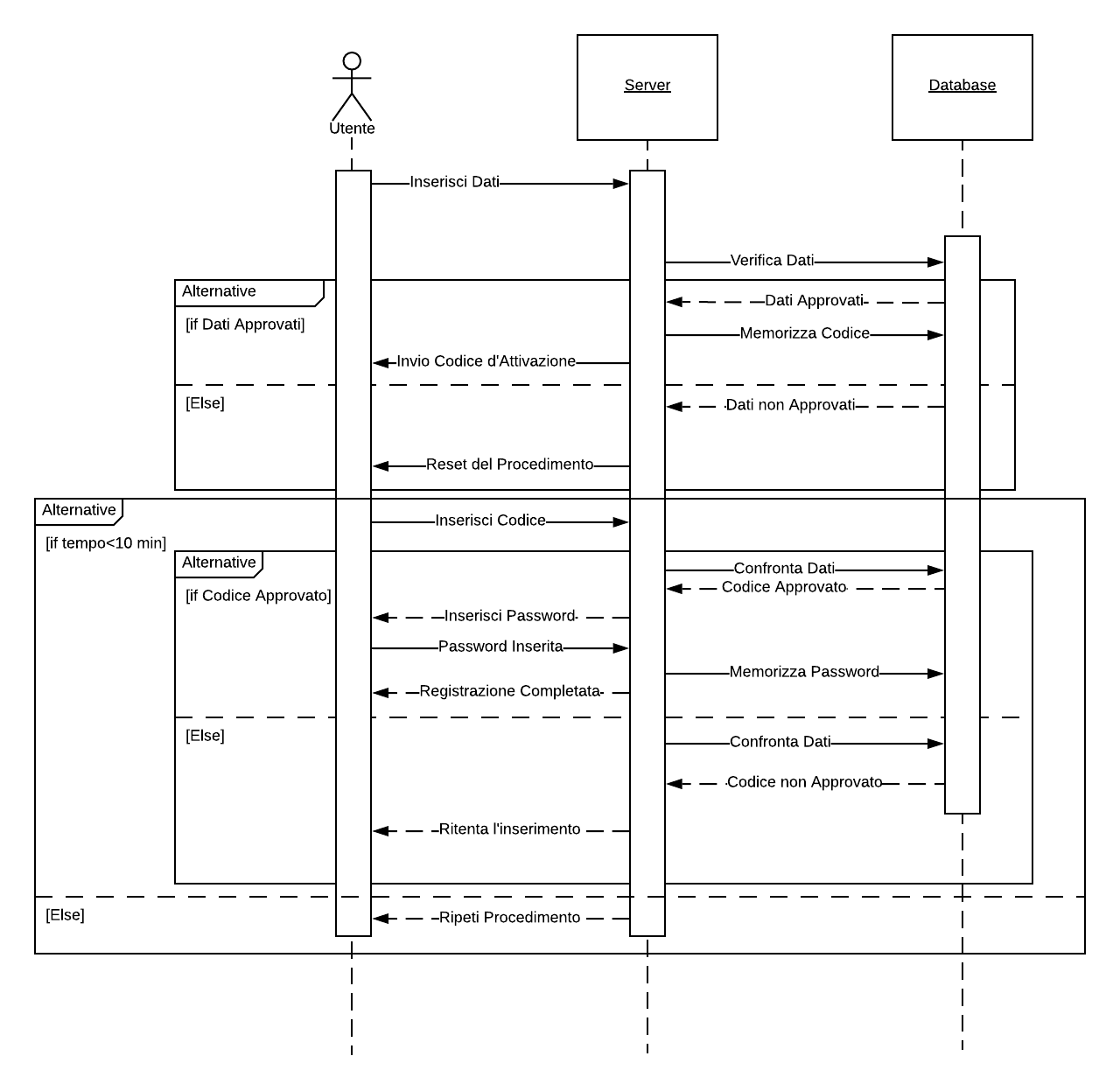
Ad accesso avvenuto correttamente il server “serverRdF” è pronto per ricevere richieste di connessione da parte dei client “playerRdF” e “adminRdF”. In caso contrario l’applicazione segnalerà l’errore e riproporrà la richiesta delle chiavi di autenticazione.



### Registrazione Giocatore / Osservatore

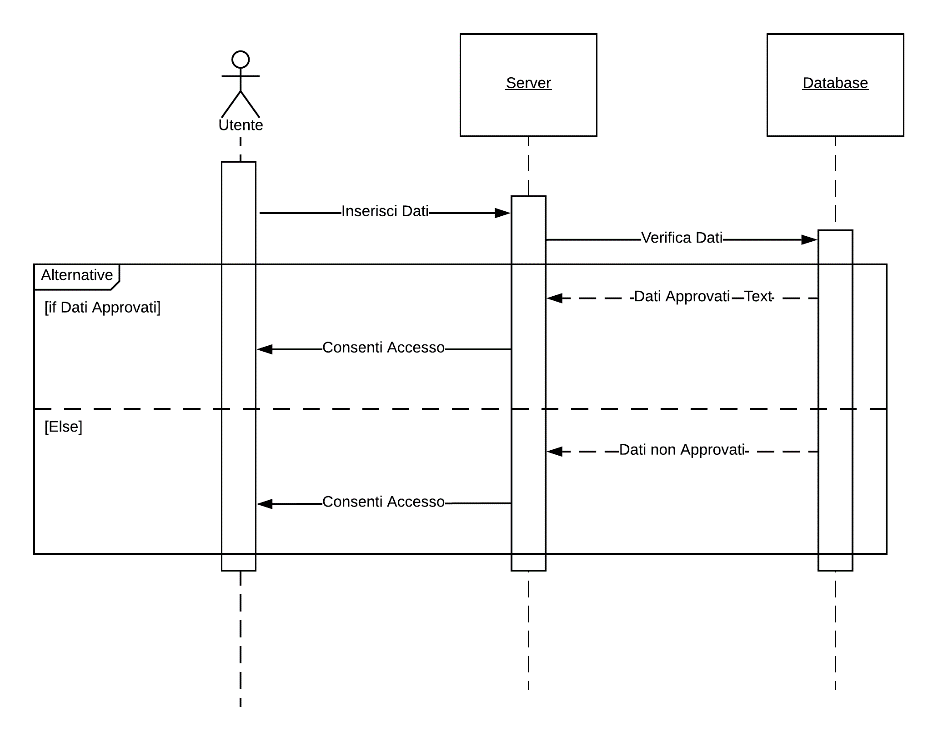
Un utente che intende accedere a RdF, come giocatore o come osservatore deve necessariamente registrarsi. In questo caso la registrazione è gestita dal modulo “playerRdf” in modo del tutto simile a quanto stabilito per la registrazione dei profili amministrativi.

Anche in questo caso, all’atto della registrazione verrà richiesto all’utente di fornire nome, cognome, un nickname, e un indirizzo di posta elettronica. Sia l’indirizzo di posta elettronica che il nickname non devono essere già stati utilizzati in precedenza da altri utenti (non devono cioè essere già stati registrati nel database). A questo punto la piattaforma invierà all’utente (alla mail comunicata) un codice di attivazione (per evitare iscrizioni fittizie o non desiderate di altri utenti che volessero mantenere l’anonimato) che gli permetterà di finalizzare la registrazione. Se l’utente non completa la registrazione entro 10 minuti, il profilo del nuovo utente e tutti i dati relativi saranno cancellati.



### Autenticazione

L’accesso alla piattaforma di gioco richiede l’autenticazione dell’utente, sia che questo sia un giocatore o un osservatore sia che questo sia un amministratore. Le credenziali di accesso sono costituite da indirizzo di posta e password associate al proprio profilo (inserite nel momento della registrazione).



Gli utenti censiti come giocatore/osservatore, dopo essersi autenticati hanno la possibilità di scegliere fra le seguenti opzioni

1. Organizzare una nuova partita, iscrivendosi come primo concorrente e aprendo le iscrizioni ad altri giocatori che volessero partecipare
2. Controllare la lista complessiva delle partite per le quale non si sono ancora chiuse le iscrizioni dei partecipanti e di quelle invece in corso di svolgimento
3. Osservare, in veste di spettatore, le partite in corso di svolgimento.
4. Iscriversi come concorrente a una partita per la quale non sono ancora state chiuse le iscrizioni
5. Abbandonare in qualsiasi momento, come giocatore o come osservatore, il gioco
6. Modificare i dati del proprio profilo (mail, password, nickname)
7. Cancellare l’iscrizione del profilo alla piattaforma e di tutti i dati associati a quel profilo
8. Analizzare le statistiche di utilizzo della piattaforma di gioco

L’amministratore della piattaforma, ovviamente non può giocare, tuttavia può ancora:

* Controllare la lista complessiva delle partite per le quale non si sono ancora chiuse le iscrizioni dei partecipanti e di quelle in corso di svolgimento
* Osservare, in veste di spettatore, le partite in corso di svolgimento
* Abbandonare il gioco in qualsiasi momento (in qualità di osservatore)
* Analizzare le statistiche di utilizzo della piattaforma di gioco
* Gestire il catalogo di frasi misteriose utilizzabili nelle manche di gioco

Nel dettaglio:

1. Organizzare una nuova partita

Un utente autenticato può organizzare una nuova partita, associando ad essa un nome. Il sistema aggiungerà al nome la data e l’ora di creazione e l’elenco dei partecipanti (giocatori e osservatori). L’organizzatore è iscritto alla lista dei partecipanti, per default.

1. Controllare la lista complessiva delle partite per le quale non si sono ancora chiuse le iscrizioni dei partecipanti e di quelle invece in corso di svolgimento

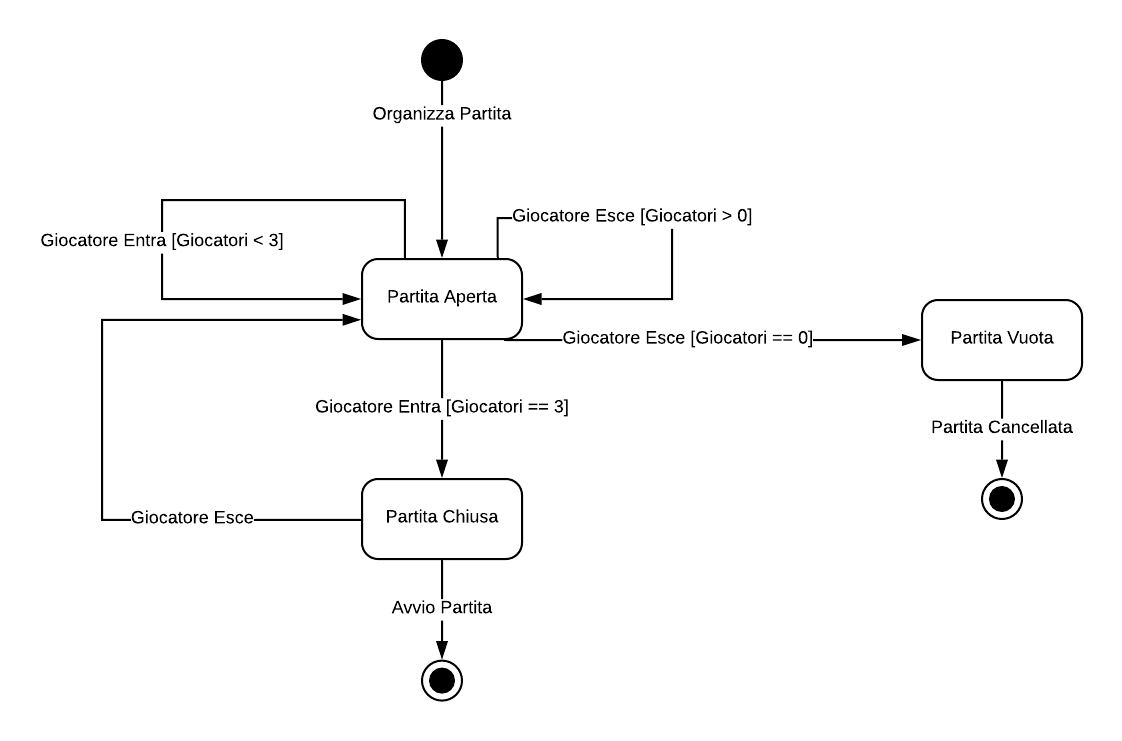
Il sistema permette a qualsiasi utente autenticato di visualizzare l’elenco completo delle partite in essere e di quelle organizzate ma con le iscrizioni dei concorrenti ancora aperte. Nel primo caso si evidenzia il numero della manche in gioco, nel secondo caso evidenzia il numero dei posti ancora disponibili (le iscrizioni si chiudono automaticamente quando i partecipanti sono 3)

1. Richiesta di partecipazione ad una partita in qualità di osservatore

Un utente autenticato può richiedere di partecipare, come osservatore, ad una partita organizzata da altri. Il sistema non pone limiti numero di osservatori che possono partecipare come osservatori ad una partita, ma la sua richiesta viene registrata[[1]](#footnote-1).

1. Richiesta di partecipazione ad una partita in qualità di concorrente

Un utente autenticato può chiedere di partecipare come concorrente ad una partita con iscrizioni ancora aperte (cioè con un numero di partecipanti inferiore a 3).

L’applicazione chiude le iscrizioni quando ad una partita risultano iscritti 3 partecipanti. Estrae dal catalogo di frasi misteriose 5 frasi (una per ogni manche) che non siano mai state usate in giochi in cui gli utenti abbiano partecipato come concorrenti od osservatori e le mette a disposizione per quella partita. Nel caso questo non fosse possibile, la piattaforma dovrà sospendere/annullare la partita, notificando ai giocatori la mancanza di nuove frasi adatte e avvisando (via mail) gli amministratori della necessita di caricare nuove frasi.

### Monitoraggio di una Partita da parte di un Osservatore (o del Giocatore)

La piattaforma RdF tiene traccia della partecipazione di concorrenti ad ogni manche, comprese quelle iniziate ma non interamente giocate e dei vincitori di ogni manche interamente giocata.

Un utente autenticato, registrato come osservatore ad una partita, deve poter monitorare l’andamento della partita. Le stesse informazioni visualizzate da un “osservatore” sono naturalmente a disposizione di ogni giocatore (i giocatori, cioè, sono anche osservatori della propria partita).

A partita iniziata i moduli “playerRdF / adminRdF” dovranno visualizzare: l’identificativo della partita e della manche in corso, il tabellone che nasconde la frase misteriosa, il punteggio nel deposito della manche e della partita di ogni concorrente, l’eventuale possesso di uno o più Jolly. Deve essere inoltre segnalato il nome del concorrente che ha il turno di gioco e tutte le mosse da lui compiute:

1. Se il concorrente “gira la ruota” dovrà essere mostrato l’esito della rotazione e gli eventuali effetti sul deposito di punti della manche.
2. Se chiama una consonante o una vocale, dovrà essere mostrata la lettera chiamata e gli effetti della mossa sul tabellone e sul suo deposito di punti della manche.
3. Se prova a dare la soluzione, dovrà essere mostrata la soluzione proposta e la sua eventuale correttezza
   * se la soluzione è corretta, il tabellone rivelerà la frase nascosta, dovrà essere evidenziato che il concorrente ha vinto la manche e che i punti acquisiti nella manche saranno trasferiti nel suo deposito punti di gara, mentre saranno azzerati i punti acquisiti in quella manche dagli altri concorrenti;
   * se la soluzione è sbagliata dovrà essere evidenziato che il concorrente ha commesso un errore
4. Se il concorrente sacrifica il proprio Jolly per mantenere il turno di gioco, dovrà essere mostrata la scelta operata, rimuovendo il jolly dall’inventario delle cose che appartengono al concorrente.

Infine, poichè le mosse hanno vincoli temporali (ad esempio la scelta della vocale/consonante o la decisione di sacrificare un "jolly" dovrà essere effettuata entro 5 secondi; il tentativo per indovinare la frase misteriosa dovrà essere completato entro 10 secondi), i moduli “playerRdF” o “adminRdF" dovranno mostrare il COUNTdown dei secondi rimanenti al concorrente per effettuare ogni mossa.

Partecipazione ad una partita da concorrente

Il modulo “playerRdF” deve permettere a un utente autenticato di giocare la partita a cui è iscritto, permettendogli, durante i suoi turni di gioco, ed entro i vincoli di tempo specificati dal regolamento, di effettuare le azioni di gioco consentite (“girare la ruota”, “chiamare una consonante”, “comprare una vocale”, “giocare un jolly”, e provare a “dare la soluzione” della frase misteriosa.

### Abbandono di una Partita

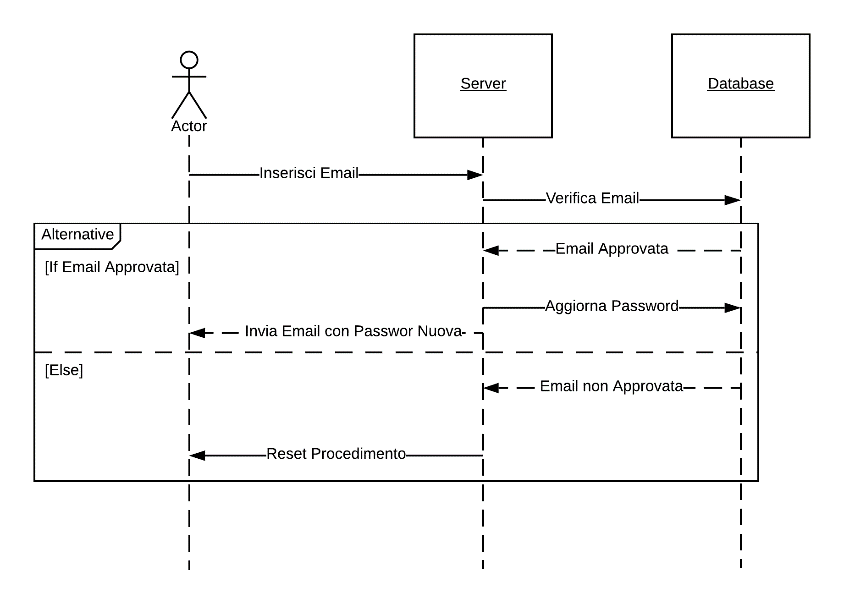
È sempre possibile, per un concorrente o per un osservatore, abbandonare la piattaforma di gioco. Nel caso si ritiri un giocatore durante lo svolgimento di una manche, l'intera partita viene annullata e la cosa verrà notificata agli altri partecipanti (giocatori e osservatori). Se l'abbandono avviene prima dell'inizio della partita il giocatore viene semplicemente rimosso dalla lista dei concorrenti iscritti. Gli osservatori possono invece abbandonare in qualsiasi momento senza altre conseguenze che quella di essere eliminati dalla lista degli osservatori. In effetti le partite possono essere giocate anche se la lista degli osservatori è vuota.

### Modifica dei Dati di Profilo

I moduli "playerRdF" e " adminRdF" permettono ad ogni utente la modifica dei dati relativi al proprio profilo (nome e cognome, nickname e password). Ogni modifica deve essere validata dall'inserimento della password corrente. L'operazione di modifica della password è notificata all'utente attraverso una mail inviata all'indirizzo di posta registrato.

**Reset di Password**

"playerRdF" e "adminRdF" devono fornire un servizio per reimpostare una password dimenticata, (ovviamente accessibile da utenti non autenticati). Il servizio richiederà all'utente di specificare la casella postale associata al profilo, rimpiazzerà la vecchia password con una nuova generata automaticamente e la invierà all'indirizzo mail di registrazione.



### Gestione delle Frasi Misteriose

L'amministratore (autenticato), mediante il modulo “adminRdF” può aggiungere / rimuovere o modificare le frasi misteriose da indovinare, e per ognuna di esse il tema associato. Dal punto di visto tecnico sia la frase misteriosa che il tema associato (contenente un indizio utile a indovinare la frase misteriosa) sono stringhe alfanumeriche con lunghezza massima di 60 caratteri (compresi gli spazi) che possono essere importate da file di testo in formato csv (linee di testo in cui le parole sono separate da un carattere divisore).

### Monitoraggio e Analisi

I moduli “playerRdF” e “adminRdF” offrono servizi accessori (non strettamente necessari allo svolgimento del gioco) come ad esempio la possibilità di visualizzare variabili utili ad analisi di tipo statistico. Fra queste variabili ci sarà quella che identifica il numero medio mosse che sono state necessarie per indovinare una frase misteriosa oppure la consonante la cui chiamata che si è rivelata più redditizia (e quella che ha fatto totalizzare più punti in una singola manche)

Altre variabili saranno riferite ai giocatori: ad esempio sarà visualizzabile il nome del giocatore che detiene il record assoluto di punteggio (in una singola manche e in una intera partita) quello del giocatore che invece ha totalizzato la media punti più alta; quello di chi ha giocato più manche; quello che ne ha vinte di più; quello di chi ha dovuto cedere la mano più volte in seguito a suoi errori echi invece l’ha dovuto cedere più volte in seguito a sfortuna (grazie un giro di ruota che l’ha costretto a perdere tutto)

Inoltre saranno mostrate le seguenti variabili statistiche riferite ad ogni singolo utente,

* numero di manche e di partite giocate,
* numero di manche e di partite osservate
* numero di manche e di partite vinte
* punteggio medio vinto per partita
* numero di volte, per manche e partita, che ha dovuto cedere il turno di gioco
* numero di volte, per manche e partita, che ha perso tutto.

**CLIENT/SERVER**

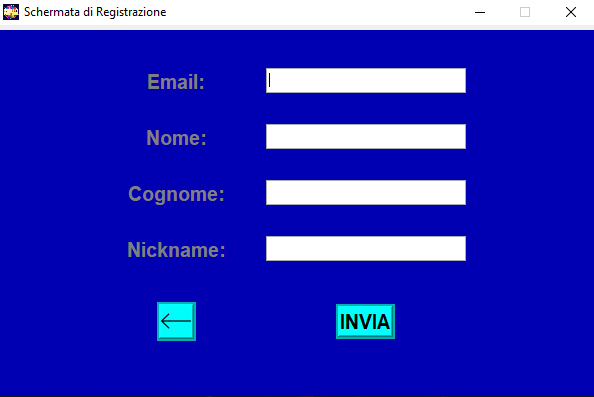
**Client**

Nel progetto RdF l’utente può accedere all’applicazione come amministratore (Admin) oppure come giocatore (Player).

Mediante un’apposita Interfaccia Grafica Utente (GUI) è possibile effettuare il Login alla piattaforma, in caso di account già registrato in precedenza, oppure accedere alla schermata Registrazione per crearne un nuovo inserendo le credenziali richieste.

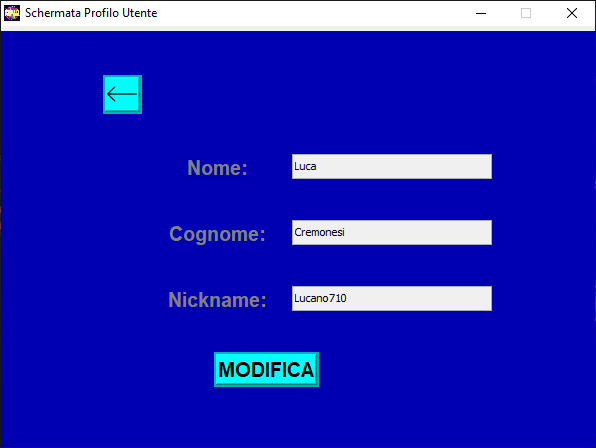


Schermata di Login

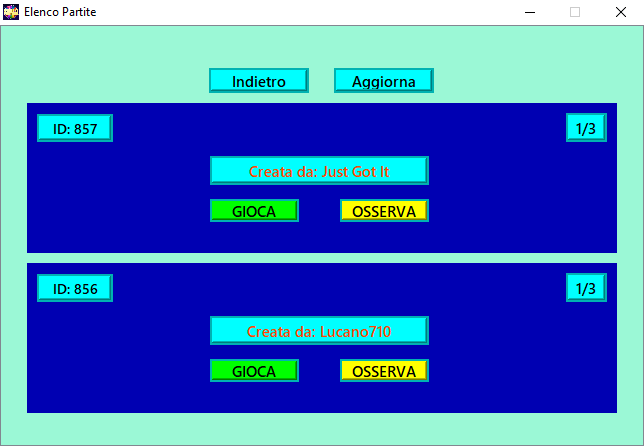


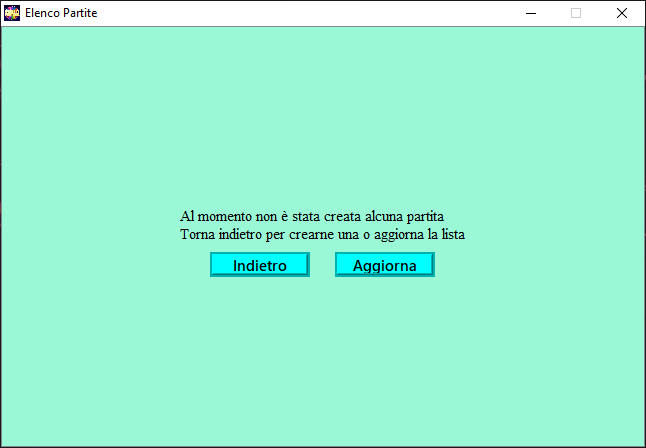
Schermata di Registrazione

Una volta effettuato l’accesso l’utente si ritroverà nella Schermata Iniziale dove entrambe le tipologie di utenti potranno gestire i propri account e visionare l’elenco delle partite



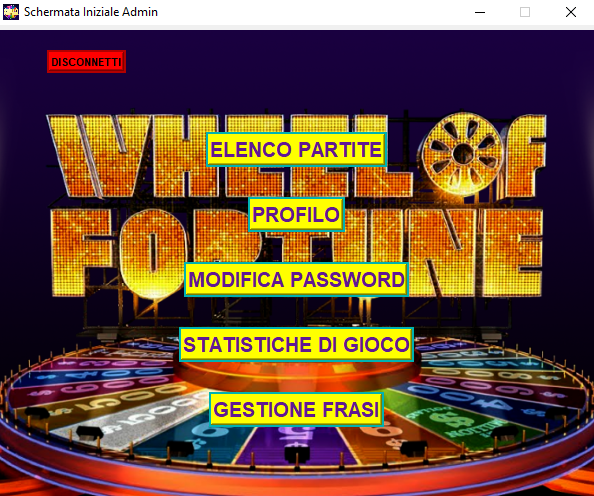
Schermata Profilo Account



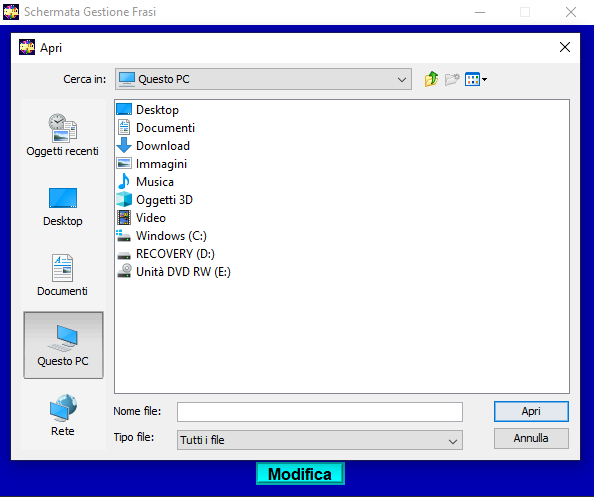


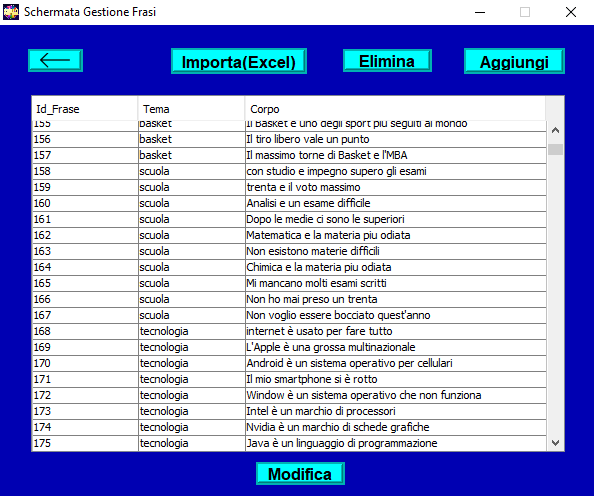
Schermata Elenco Partite

L’utente Admin potrà accedere alla funzione Gestione Frasi in cui sarà possibile gestire la creazione delle Frasi Misteriose

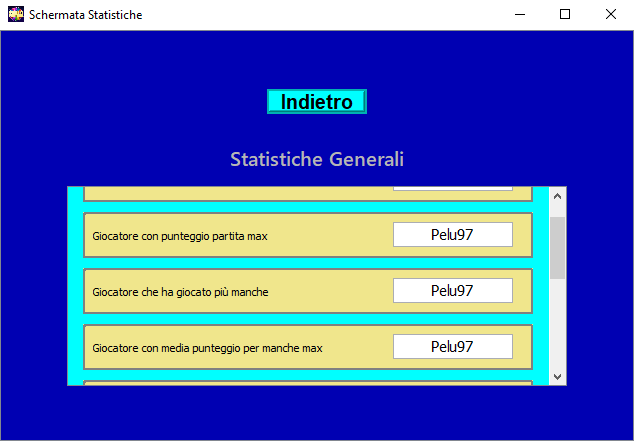


Schermata Iniziale Admin

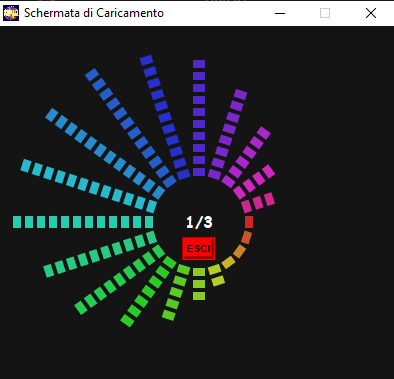




Schermate di caricamento e visualizzazione file contente le Frasi Misteriose

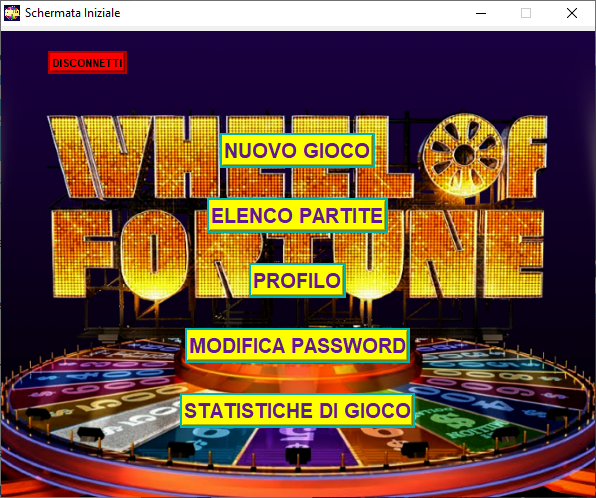


Schermata di visualizzazione delle Statistiche di Gioco

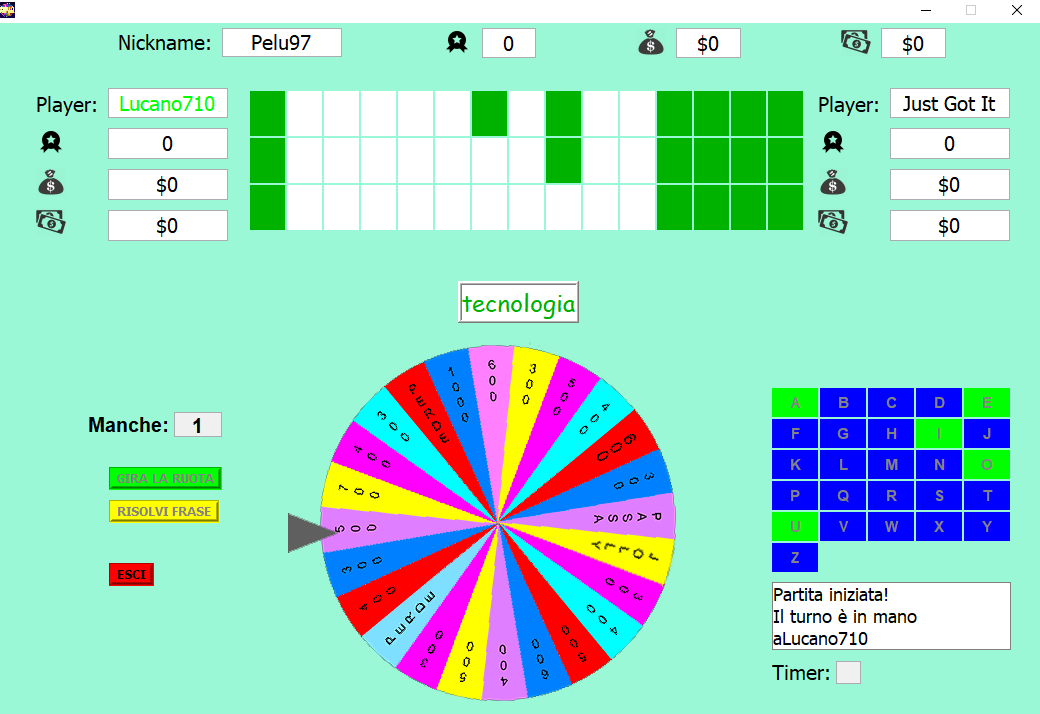


Schermata di Caricamento

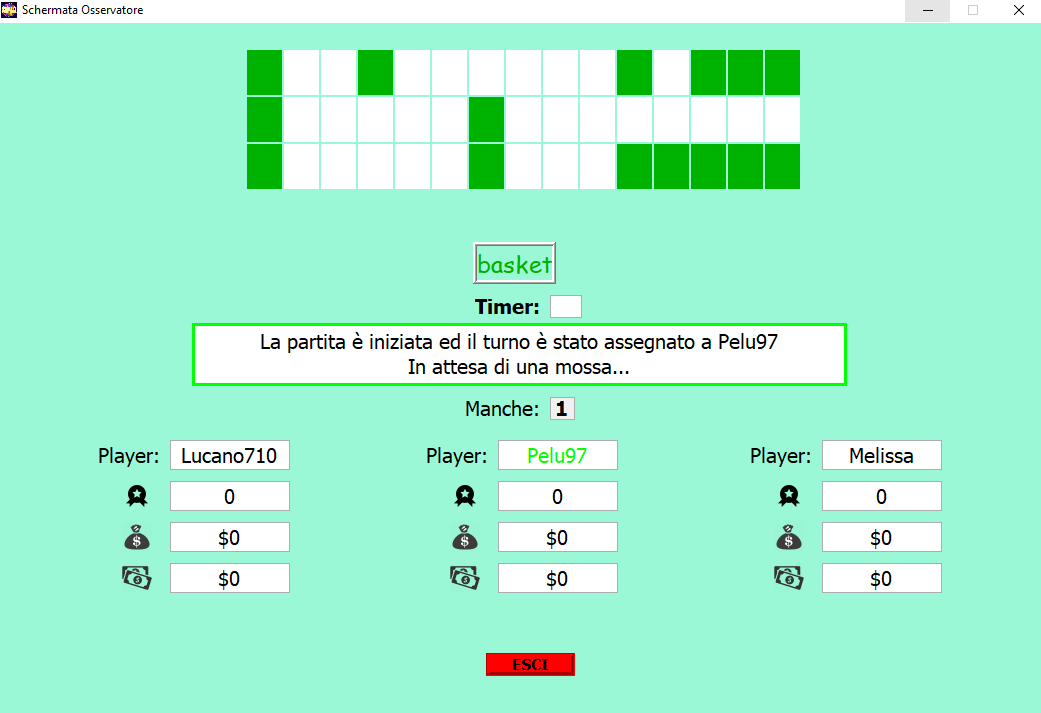
L’utente Player potrà iniziare nuove partite accedendo alla funzione Nuovo Gioco



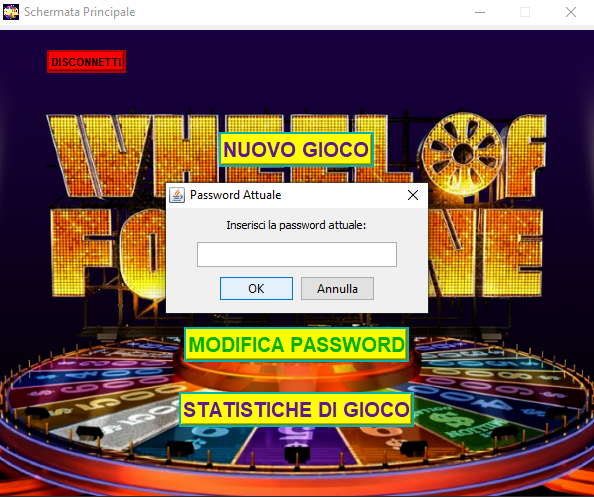
Schermata Iniziale Player

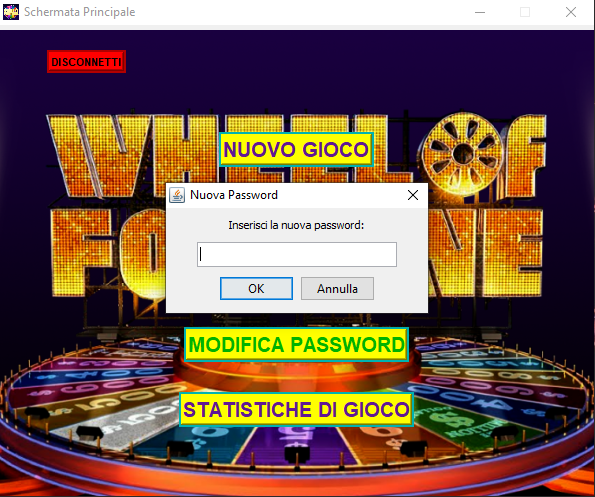


Schermata Partita



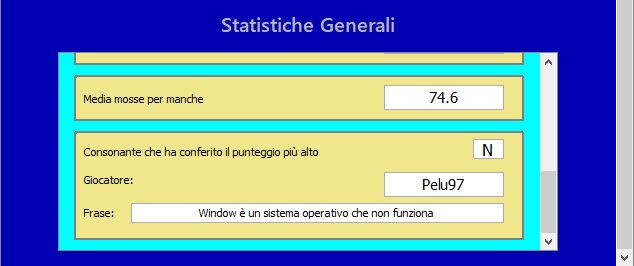
Schermata Osservatore





Schermata di Modifica Password

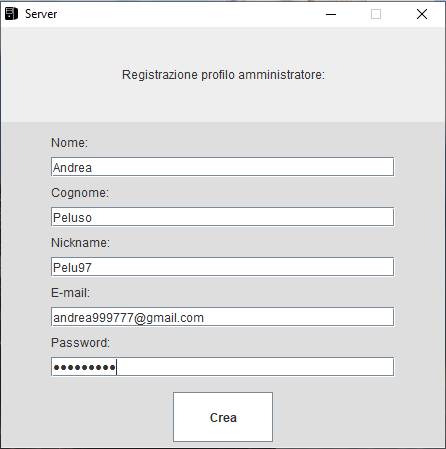


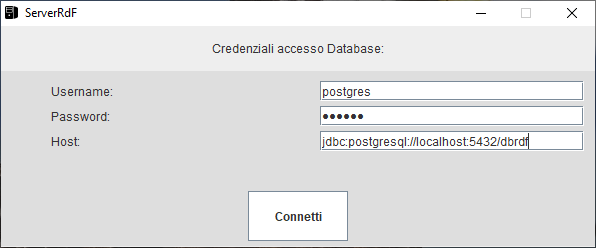


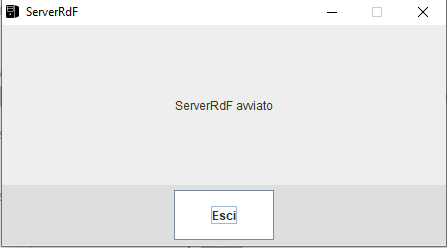
Schermata Statistiche Client

**Server**

Il server ha il compito di instaurare la comunicazione con il Client mediante RMI permettendo il trasferimento di dati relativi al profilo utente e ai dati di gioco nel Database dove saranno archiviati in apposite tabelle.







**Iterazione**

Per la comunicazione tra Server e Client è stato utilizzato l’implementazione RMI (Remote Method Invocation, ovvero Invocazione Remota di Metodi).

Questa tecnologia include una API (Application Programming Interface) il cui scopo esplicito è quello di rendere trasparenti al programmatore quasi tutti i dettagli della comunicazione su rete.

Essa consente infatti di invocare un metodo di un oggetto remoto (ovvero appartenente a un diverso processo su una diversa macchina), quasi come se tale oggetto fosse "locale" (ovvero appartenente allo stesso processo in cui viene eseguita l'invocazione).

Il vantaggio di RMI consiste nella possibilità di utilizzare la medesima sintassi Java e la relativa progettazione orientata agli oggetti, persino durante l’invocazione di metodi appartenenti a oggetti remoti.

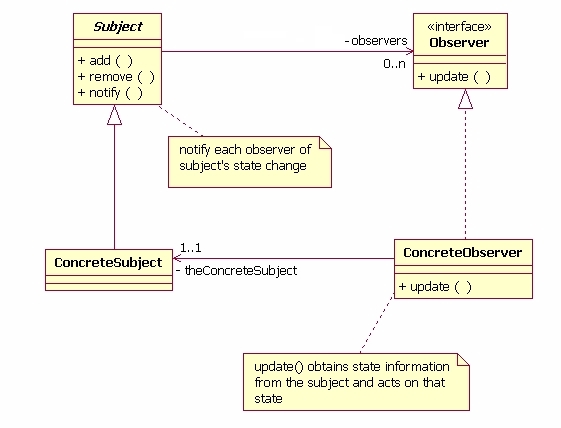


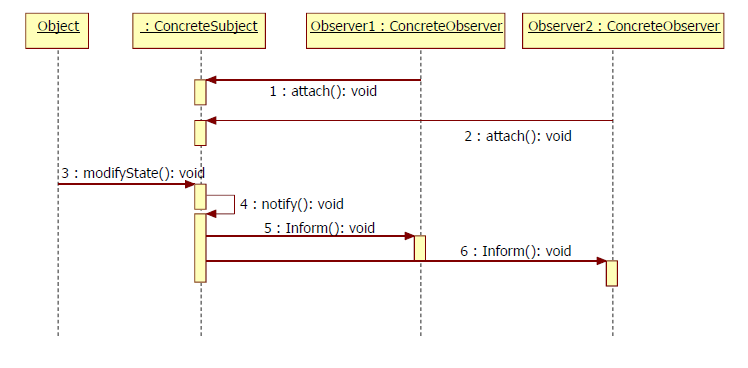
**Il Design Pattern Observer**

Il pattern Observer trova applicazione nei casi in cui diversi oggetti (Observer) devono conoscere lo stato di un oggetto (Subject).

Abbiamo pertanto un oggetto che viene “osservato” (il subject) e tanti oggetti che “osservano” i cambiamenti di quest’ultimo (gli observers).

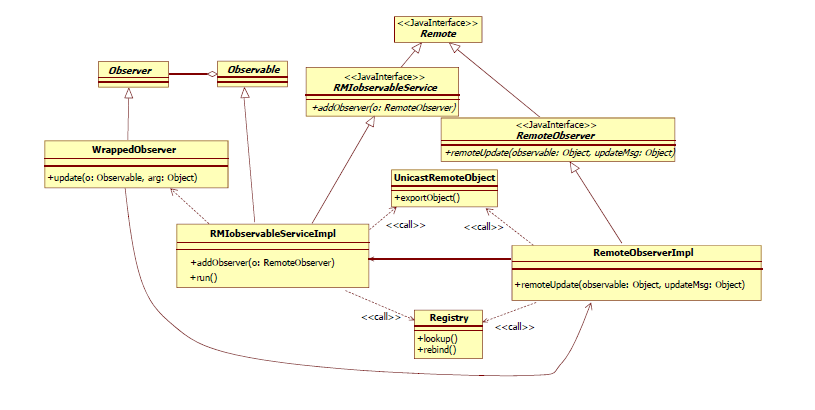
Gli elementi di cui è composto sono:





Observer Sequence Diagram

* Subject: classe Observable.
  + Conosce i propri Observer, i quali possono essere in  
    numero illimitato
  + Fornisce operazioni per l’aggiunta e cancellazione di  
    Observer
  + Fornisce operazioni per la notifica agli Observer.
* Observer: interfaccia Observer
  + Specifica un’interfaccia per la notifica di eventi agli oggetti  
    interessati in un Subject.
* ConcreteSubject: classe ObservedSubject.
  + Mantiene lo stato del soggetto osservato ed effettua una notifica agli observer in caso di un cambio di stato
  + Invoca le operazioni di notifica ereditate dal Subject, quando  
    devono essere informati i ConcreteObserver.
* ConcreteObserver:
  + Implementa l’interfaccia Observer definendo il comportamento in caso di un cambio di stato del soggetto osservato



## 

## **Piano di progetto**

La progettazione della piattaforma di gioco “RdF”, oltre a presupporre un processo strutturato di sviluppo di un sistema software (e della sua base di dati) che soddisfi i requisiti funzionali sopra descritti, ha richiesto la produzione di artefatti *(elementi documentativi che delineano funzione e architettura del software prodotto, quali diagrammi di stato, modelli UML, gli schemi Entity-Relationship per il Database, ma anche lo stesso codice sorgente e la documentazione tecnica)* che descrivano, in conformità con le metodologie seguite a lezione, quanto realizzato nelle varie fasi dello sviluppo del software.

Ci si è avvalsi, per quanto possibile, di modelli logici precostituiti (design patterns) cercando di realizzare l’applicazione con una interfaccia grafica adeguata. Inoltre è stato progettato un database relazionale utilizzando **PostgreSQL** per la sua implementazione[[2]](#footnote-2) e **JDBC** per l'accesso alla base di dati da parte del programma Java[[3]](#footnote-3).

Poiché la piattaforma di gioco può essere utilizzata da più utenti allo stesso momento (in parallelo) e che dunque è possibile che si verifichino accessi concorrenti a risorse condivise, l'applicazione è stata definita in modo tale da poter gestire la cosa.

## **Realizzazione del Database**

Il Database è stato realizzato utilizzando il software **PostgreSQL** dove sono archiviati tutti i dati relativi al gioco.

Per realizzarlo è stata effettuata la **Raccolta ed Analisi dei Requisiti:**

* Un utente è caratterizzato da: **nome, cognome, nickname e indirizzo di posta elettronica**.
* L’indirizzo di posta elettronica e il nickname non devono essere già stati usati da altri utenti.
* Una partita ha associato **un nome, una data, e un’ora di creazione, un elenco dei concorrenti e di osservatori**; una partita è composta da cinque manche e può avere le iscrizioni aperte o chiuse.
* Un utente può registrarsi come osservatore ad una partita.
* Non sono previsti limiti al numero di osservatori che partecipano ad una partita.
* Un giocatore ha associato **un punteggio relativo alla partita, un punteggio relativo alla singola manche in corso, da tutte le mosse fatte nella partita, dai punti acquisiti e persi con ogni mossa.**
* Una manche è composta da **un identificativo, dai concorrenti che la hanno giocata (anche in parte) e dai vincitori** **della stessa.**
* Una frase misteriosa è caratterizzata da **un identificativo e da un tema associato.**

**Query tipiche utente:**

* Organizzare una nuova partita.
* Visualizzare le partite organizzate per la quale non si sono ancora chiuse le iscrizioni dei concorrenti.
* Visualizzare le partite in corso di svolgimento.
* Richiedere la partecipazione, come concorrente, ad una partita organizzata da altri utenti per la quale non sono ancora state chiuse le iscrizioni (dei concorrenti).
* Richiedere la partecipazione, come osservatore, ad una partita per la quale non sono ancora chiuse le iscrizioni (dei concorrenti), o che è in corso di svolgimento.
* Abbandonare, da concorrente, o da osservatore, una partita per la quale non sono ancora state chiuse le iscrizioni, oppure che è in corso di svolgimento.
* Modificare i dati del proprio profilo.
* Analizzare le statistiche di utilizzo della piattaforma di gioco.
* Visualizzare le partite organizzate per la quale non si sono ancora completate le iscrizioni dei concorrenti.
* Visualizzare le partite in corso di svolgimento.
* Richiedere di partecipare, come osservatore, ad una partita in corso di svolgimento, o per la quale non sono ancora completate le iscrizioni dei concorrenti oppure che è in corso di svolgimento.
* Modificare i dati del proprio profilo.
* Gestire il catalogo di frasi misteriose utilizzabili nelle manche di gioco.
* Analizzare statistiche di utilizzo della piattaforma di gioco.
* Analizzare statistiche di utilizzo della piattaforma di gioco.
* Mostrare il concorrente con la media più alta di punti acquisiti per manche.
* Mostrare il concorrente che ha dovuto cedere più volte il turno di gioco in seguito ad errori.
* Mostrare il concorrente che ha perso tutto in seguito ad un giro di ruota per il maggior numero di volte.
* Mostrare la chiamata di consonante, riferita ad una manche già giocata, che ha portato all’acquisizione della maggior quantità punti, la frase misteriosa associata, e l’utente che ha fatto la mossa.
* Mostrare il numero medio mosse per manche con le quali viene indovinata una frase.
* Mostrare il numero di manche e di partite giocate.
* Mostrare il numero di manche e di partite osservate.
* Mostrare il numero di manche e partite vinte.
* Mostrare il punteggio medio vinto per partita.
* Mostrare il numero medio di volte, per manche e partita, che ha dovuto cedere il turno di gioco.
* Mostrare il numero medio di volte, per manche e partita, che ha perso tutto.
* Abbandonare, da osservatore, una partita per la quale sono ancora state chiuse le iscrizioni.

**Progettazione logica Iniziale**

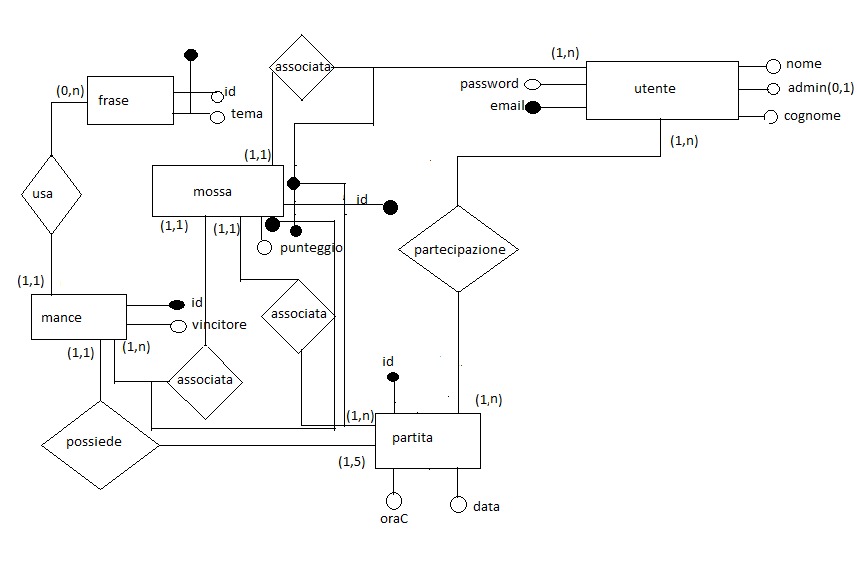


Immagine rappresentate lo Schema ER iniziale

UTENTE (email, nome, cognome, password, admin, giocatore)

PARTITA (idpartita, oracreazione, data)

PARTECIPAZIONE (emailutente, idpartitapartita)

MOSSA (idmossa,idmanchemanche, idutenteutente, idpartitapartita, punti)

MANCHE (idmanche, vincitore, idpartitapartita, idfrase)

FRASE (id, tema, corpo)

vincoli: se admin=1, allora giocatore=0

se giocatore=1, allora admin=0

**Progettazione logica Finale**

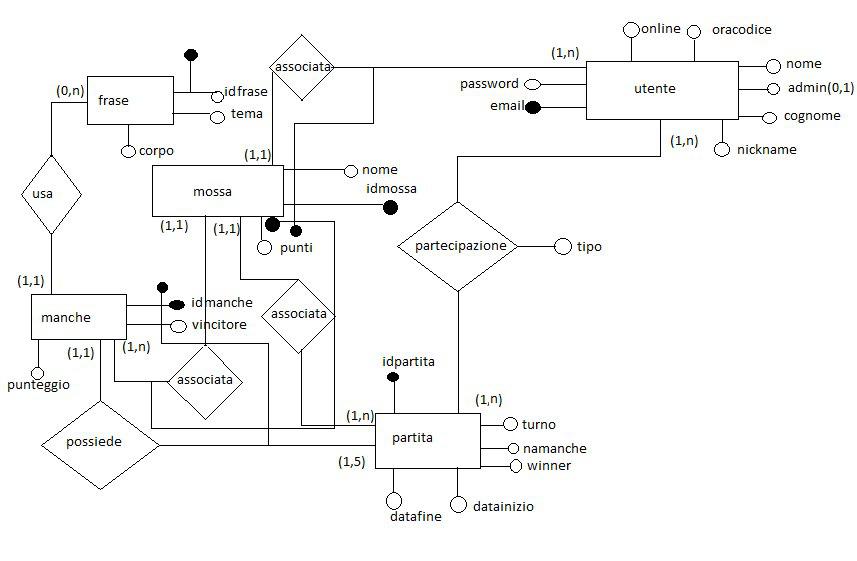


Immagine rappresentate lo Schema ER finale

UTENTE (email, nome, cognome, nickname, password, admin, oracodice, online)

PARTITA (idpartita, datainizio, datafine, emailutente, nmanche, winner, turno)

PARTECIPAZIONE (emailutente, idpartitapartita, tipo)

MOSSA (idmossa, idmanchemanche, idutenteutente, idpartitamanche, punti, nome)

MANCHE (idmanche, idpartitapartita, idfrasefrase, punteggio, vincitore)

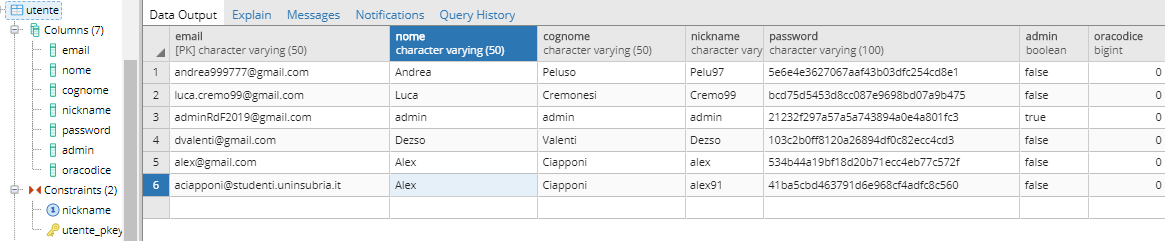
FRASE (idfrase, tema, corpo)

Successivamente la Progettazione Logica è stata strutturata mediante 6 tabelle:

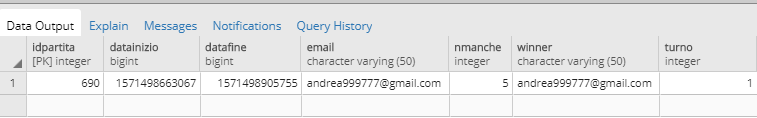
* Utente: tabella con Primary Key “email” in cui ogni tupla tiene traccia dei dati dell’utente:
* email
* nome
* cognome
* nickname
* password
* admin: un campo di tipo booleano che indica se l’utente sia un admin (true) oppure un player

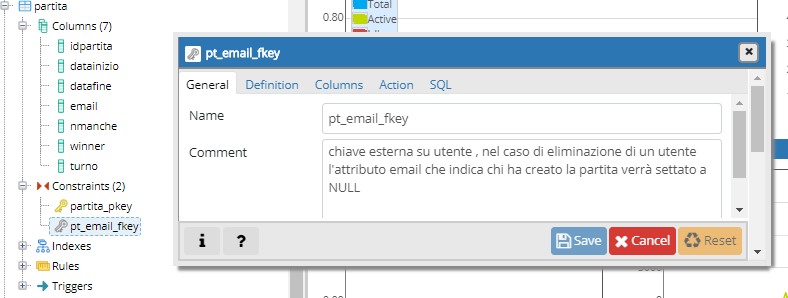
(false);

* codice: rappresentante un codice univoco assegnato a ciascun utente;

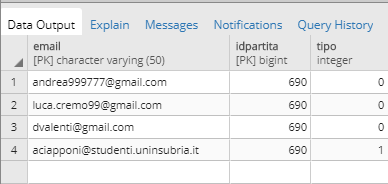


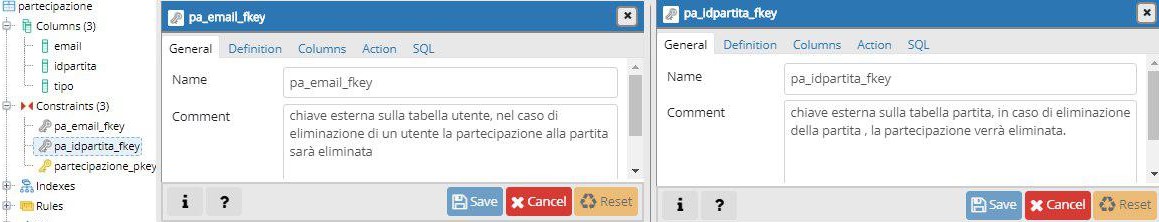
* Partita: sono registrati tutti i dati relativi allo svolgimento di una partita. “idpartita” rappresenta la Primary Key e “email” la Foreign Key sulla tabella Utente. Quando un giocatore inizia una nuova partita viene aggiunta una nuova tupla contente:
* l’id della partita;
* data inizio;
* data fine;
* l’email del giocatore creatore della partita;
* il numero della manche corrente;
* il vincitore della partita;
* il giocatore di turno in quel momento





* Partecipazione: tabella che contiene il numero di partecipanti a una partita. La Primary Key è composta da “email” e “idpartita”, Foreign Key delle tabelle Utente e Partita.

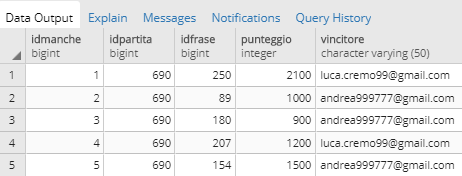


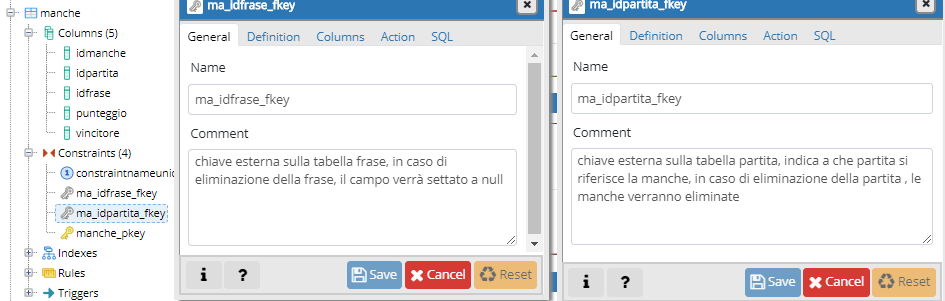


* Manche: tabella in cui ogni tupla registra i dati relativi ad una determinata manche di una determinata partita. La Primary Key è formata dalle colonne “idmanche” e “idpartita”. Quest’ultima come la colonna “frase” svolgono la funzione di Foreign Key rispettivamente su Partita e su Frase.

Compaiono le seguenti informazioni:

* il numero identificativo della manche;
* il numero identificativo della partita;
* il numero identificativo della frase;
* il punteggio ottenuto in quella manche;
* il vincitore della manche.

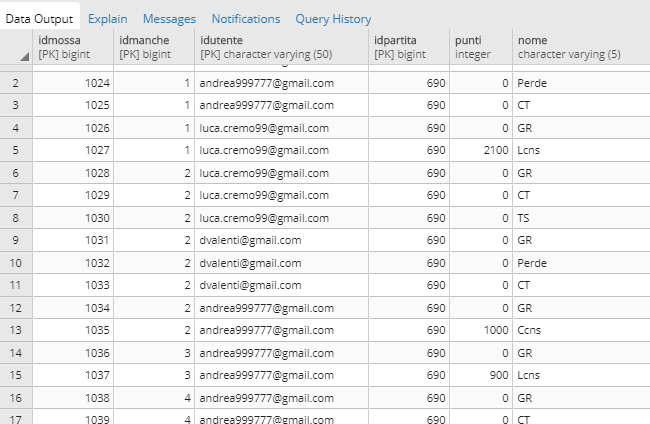


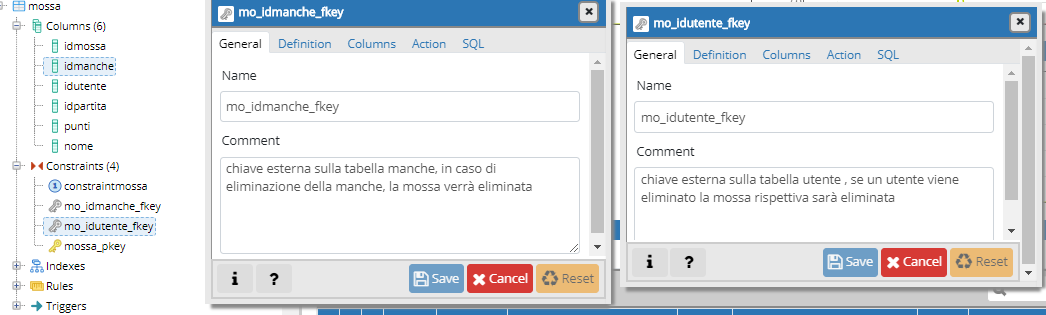


* Mossa: tabella che registra tutte le mosse effettuate dai giocatori. La Primary Key è formata da “idmossa”, “idmanche”, “idutente”, “idpartita” dove le ultime 3 colonne sono Foreign Key rispettivamente Manche, Utente e Partita.

Oltre a contenere tutti i dati numerici identificativi registrati nella Chiave Primaria, sono inclusi:

* i punti ottenuti con la mossa effettuata;
* il nome della mossa eseguita.

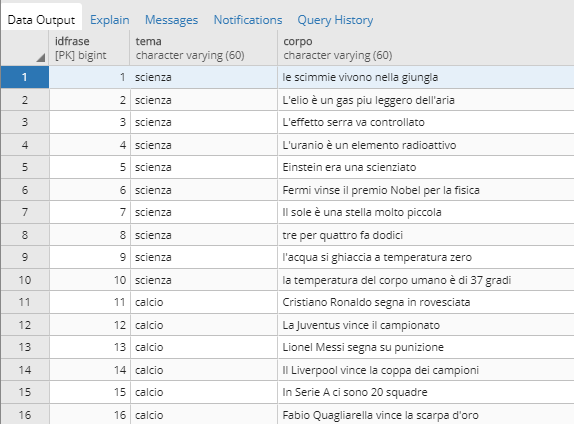




* Frase: tabella che tiene traccia di tutte le Frasi Misteriose (una volta indovinata viene vista da tutti gli Observer) già utilizzate nelle partite disputate per garantire univocità. La Primary Key è formata dalla sola colonna “idfrase” e non contiene Foreign Key.

Ogni tupla registra:

* l’identificativo della frase;
* l’argomento trattato;
* la frase effettiva.



Per ottenere i dati utente da memorizzare nel Database è stata progettata un’interfaccia grafica utente in cui ogni partecipante può effettuare il login oppure la registrazione.

Per gestire il proprio account gli utenti hanno a disposizione diverse funzionalità:

* Login: è il bottone che consente di effettuare il login per accedere alla Home del gioco.

Nickname e email differenziano gli utenti pertanto questi due campi dovranno essere univoci, quando vengono inseriti avviene un controllo dal server al database che verifica se non esistano utenti con stessi email e nickname.

* Registrati: consente di effettuare la registrazione per inserire i nuovi dati utente nel DB.

Una volta fatto l’utente riceverà una mail con il codice di autenticazione, a cui successivamente (in seguito la digitazione del codice) verrà chiesto di mettere una password.

* Codice: colonna nel DB “oracodice” che indica quando il codice di autenticazione è stato inviato serve per fare il controllo dei 10 minuti entro i quali se non completi la registrazione l’account viene eliminato

la colonna serve perché “oracodice” viene settata a 0 solo se viene confermato il codice.

* Reset Password: se un utente dimentica la password deve poter resettare la password ricevendone una nuova per mail (è stato utilizzato algoritmo di hashing md5 per la crittografia delle password).
* Modifica Password: consente di cambiare password dall'utente.
* Modifica Profilo: consente di modificarlo in qualsiasi momento.

## **La codifica**

Il codice sorgente dell'applicazione è stato definito seguendo linee guida, naming convention e best practices internazionalmente riconosciute: si è utilizzato l’ambiente di sviluppo integrato Eclipse (Photon).

È stata utilizzata la lingua inglese per etichettare classi, metodi, attributi e variabili; i commenti sono stati espressi in italiano per favorirne l’immediatezza della comprensione. Si è cercato, per quanto sia stato possibile, di ottimizzare la leggibilità e la manutenibilità del software, revisionando il codice per compattarlo eliminando parti duplicate o ridondanti e modificando quelle righe necessarie per un’eventuale messa a punto.

**Codice Database**

**Script Necessari alla Creazione del Database**

CREATE TABLE public.frase (

idfrase bigint NOT NULL,

tema character varying (60) NOT NULL,

corpo character varying (60) NOT NULL

);

CREATE TABLE public.manche (

idmanche bigint NOT NULL,

idpartita bigint NOT NULL,

idfrase bigint,

punteggio integer,

vincitore character varying (50)

);

CREATE TABLE public.mossa (

idmossa bigint NOT NULL,

idmanche bigint NOT NULL,

idutente character varying (50) NOT NULL,

idpartita bigint NOT NULL,

punti integer,

nome character varying (5)

);

CREATE TABLE public.partecipazione (

email character varying (50) NOT NULL,

idpartita bigint NOT NULL,

tipo integer

);

CREATE TABLE public.partita (

idpartita integer NOT NULL,

datainizio bigint NOT NULL,

datafine bigint NOT NULL,

email character varying (50) NOT NULL,

nmanche integer,

winner character varying (50),

turno integer

);

CREATE TABLE public.utente (

email character varying (50) NOT NULL,

nome character varying (50) NOT NULL,

cognome character varying (50) NOT NULL,

nickname character varying (10) NOT NULL,

password character varying (100) NOT NULL,

admin boolean NOT NULL,

oracodice bigint,

online boolean

);

ALTER TABLE ONLY public.mossa

ADD CONSTRAINT constraintmossa UNIQUE (idmossa, idmanche, idutente, idpartita);

ALTER TABLE ONLY public.manche

ADD CONSTRAINT constraintnameunique UNIQUE (idmanche, idpartita);

ALTER TABLE ONLY public.frase

ADD CONSTRAINT frase\_pkey PRIMARY KEY (idfrase);

ALTER TABLE ONLY public.manche

ADD CONSTRAINT manche\_pkey PRIMARY KEY (idmanche, idpartita);

ALTER TABLE ONLY public.mossa

ADD CONSTRAINT mossa\_pkey PRIMARY KEY (idmossa, idmanche, idutente, idpartita);

ALTER TABLE ONLY public.utente

ADD CONSTRAINT nickname UNIQUE (nickname);

ALTER TABLE ONLY public.partecipazione

ADD CONSTRAINT partecipazione\_pkey PRIMARY KEY (email, idpartita);

ALTER TABLE ONLY public.partita

ADD CONSTRAINT partita\_pkey PRIMARY KEY (idpartita);

ALTER TABLE ONLY public.utente

ADD CONSTRAINT utente\_pkey PRIMARY KEY (email);

ALTER TABLE ONLY public.manche

ADD CONSTRAINT ma\_idfrase\_fkey FOREIGN KEY (idfrase) REFERENCES public.frase(idfrase) ON UPDATE SET NULL ON DELETE SET NULL;

ALTER TABLE ONLY public.manche

ADD CONSTRAINT ma\_idpartita\_fkey FOREIGN KEY (idpartita) REFERENCES public.partita(idpartita) ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE ONLY public.mossa

ADD CONSTRAINT mo\_idmanche\_fkey FOREIGN KEY (idmanche, idpartita) REFERENCES public.manche(idmanche, idpartita) ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE ONLY public.mossa

ADD CONSTRAINT mo\_idutente\_fkey FOREIGN KEY (idutente) REFERENCES public.utente(email) ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE ONLY public.partecipazione

ADD CONSTRAINT pa\_email\_fkey FOREIGN KEY (email) REFERENCES public.utente (email) ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE ONLY public.partecipazione

ADD CONSTRAINT pa\_idpartita\_fkey FOREIGN KEY (idpartita) REFERENCES public.partita(idpartita) ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE ONLY public.partita

ADD CONSTRAINT pt\_email\_fkey FOREIGN KEY (email) REFERENCES public.utente(email) ON UPDATE SET NULL ON DELETE SET NULL;

**Query SQL a Supporto dei Servizi Erogati da Rdf**

**Classe DBimplementation**

Contiene le query generali che effettua il server al database:

SELECT \*

FROM utente

WHERE admin=true

Cerca se esiste un amministratore nel database

COPY frase (tema, corpo)

FROM STDIN (FORMAT csv, HEADER, ENCODING 'UTF-8')

Prende le frasi da un file “csv” e le copia nella tabella Frase del Database

**Classe GameQuery**

Contiene i metodi usati per le query effettuate durante il gioco:

SELECT nickname

FROM partecipazione

NATURAL JOIN utente

WHERE idpartita=? AND tipo=0

Consente di ottenere la lista dei giocatori alla partita tranne chi ha chiamato il metodo

SELECT nickname

FROM partecipazione

NATURAL JOIN utente

WHERE idpartita=? AND tipo=0

Consente di ottenere la lista di tutti i giocatori alla partita

SELECT nome, cognome, nickname

FROM utente

WHERE email=?

Seleziona i dati relativi all'utente

INSERT INTO partita (datainizio, datafine, email, nmanche, winner, turno)

VALUES (?, ?, ?, 0, null, ?)

Inserimento di una nuova partita

INSERT INTO partecipazione (email, idpartita, tipo)

VALUES (?, ?, 0)

Inserisce partecipazione alla partita

SELECT idpartita, nickname

FROM partita NATURAL JOIN utente

WHERE datafine=0

ORDER BY idpartita DESC LIMIT 5")

Seleziona la lista delle partite (le ultime 5)

SELECT nickname, SUM (punteggio) AS score

FROM partecipazione p

left join manche m ON p.email=m.vincitore AND p.idpartita=m.idpartita

INNER JOIN utente u on p.email=u.email

WHERE p.tipo=0 and p.idpartita=?

GROUP BY (nickname)

ORDER BY score DESC nulls last, nickname ASC

Consente di ottenere la classifica

SELECT COUNT (\*)

FROM partecipazione

WHERE idpartita=? and tipo=0

Seleziona il numero di giocatori alla partita (escludendo gli osservatori)

SELECT email

FROM partecipazione

WHERE idpartita=? AND tipo <> 2

Seleziona la lista dei giocatori e degli osservatori alla partita

UPDATE partita SET nmanche=nmanche+1

WHERE idpartita=?

Aggiorna la partita settando la manche a 1

SELECT email

FROM partecipazione

WHERE idpartita=? AND tipo=0

Seleziona la lista dei giocatori alla partita

SELECT idfrase

FROM frase

WHERE idfrase NOT IN

(SELECT DISTINCT idfrase

FROM manche

NATURAL JOIN partita

INNER JOIN partecipazione ON partita.idpartita=partecipazione.idpartita

WHERE (partecipazione.email=? OR partecipazione.email=? OR partecipazione.email=?)

AND idfrase is not null)

ORDER BY RANDOM ()

LIMIT 1

Sceglie una frase che non è mai stata usata dai tre giocatori

INSERT INTO manche (idmanche, idpartita, idfrase, punteggio, vincitore)

VALUES (1, ?, ?, 0, null)

inserisco una nuova manche nella tabella manche associando la frase

UPDATE manche SET vincitore=?, punteggio=?

WHERE idpartita=? AND idmanche=

(SELECT nmanche

FROM partita

WHERE idpartita=?)

Aggiorna l’ultima manche inserendo il vincitore e il suo punteggio

UPDATE partita SET nmanche=nmanche+1

WHERE idpartita=?

Aggiorna la tabella partita cambiando il numero della manche

SELECT nmanche

FROM partita

WHERE idpartita=?

Seleziona il numero della manche

SELECT email

FROM partecipazione

WHERE idpartita=? AND tipo=0

Seleziona i giocatori della partita

INSERT INTO manche (idmanche, idpartita, idfrase, punteggio, vincitore)

VALUES (?, ?, ?, 0, null)

Inserisce una nuova manche associandole la frase

SELECT nmanche

FROM partita

WHERE idpartita=?

Seleziona il numero della manche corrispondete alla partita

SELECT tema

FROM frase NATURAL JOIN manche NATURAL JOIN partita

WHERE idmanche=? AND partita.idpartita=? AND partita.datafine = 0

Seleziona il tema della frase

INSERT INTO mossa (idmanche, idutente, idpartita, punti, nome)

VALUES (?, ?, ?, ?, ?)

Inserisce una nuova mossa nella tabella mossa

UPDATE partita SET datafine=?

WHERE idpartita=?

Aggiorna la partita inserendo la data di fine della stessa

INSERT INTO partecipazione (email, idpartita, tipo)

VALUES (?, ?, 0)

Inserisce la partecipazione nella tabella partecipazione inserendo come tipo 0 (giocatore)

SELECT email

FROM partita

WHERE idpartita=?

Seleziona il creatore della partita

SELECT SUM (punteggio), vincitore

FROM manche

WHERE idpartita=? AND vincitore IN

(SELECT email

FROM partecipazione

WHERE idpartita=? AND tipo=0)

GROUP BY vincitore

Seleziona i punteggi ottenuti dai 3 giocatori nella partita

UPDATE partita SET winner=?

WHERE idpartita=?

Aggiorna la partita impostando il vincitore

SELECT email FROM partecipazione WHERE idpartita=? and tipo=0

LIMIT 1 OFFSET

(SELECT turno

FROM partita

WHERE idpartita=?)

Seleziona l’e-mail del giocatore di turno della partita

SELECT turno

FROM partita

WHERE idpartita=?

Seleziona il turno dalla tabella partita

UPDATE partita SET turno=?

WHERE idpartita=?

Aggiorna il turno

INSERT INTO mossa (idmanche, idutente, idpartita, punti, nome)

VALUES (?, ?, ?, ?, ?);

Aggiorna mossa quando un giocatore fa una mossa (dipende dal tipo di mossa)

DELETE FROM partecipazione

WHERE idpartita=? AND email=?

Elimina la partecipazione dalla partita quando un osservatore o giocatore esce dalla lobby

UPDATE partecipazione SET tipo=2

WHERE idpartita=? AND email=?

Elimina la partecipazione dalla partita quando un osservatore o giocatore esce dalla lobby

DELETE FROM partita

WHERE idpartita=?

Elimina la partita

SELECT \*

FROM partecipazione

WHERE idpartita=? AND email=?

Controlla se esiste già un osservatore con quella email in quella partita

UPDATE partecipazione SET tipo=1

WHERE idpartita=? AND email=?

Se esiste aggiorna lo stato a 1

INSERT INTO partecipazione (email, idpartita, tipo)

VALUES (?, ?, 1)

Inserisce la partecipazione di un osservatore nel database

SELECT SUM (punti)

FROM mossa

WHERE idutente=

(SELECT email

FROM utente

WHERE nickname=?) AND idpartita=? AND idmanche=

(SELECT nmanche

FROM partita

WHERE idpartita=?)

Consente di ottenere il punteggio corrispondente alla manche della partita che un giocatore sta disputando

SELECT SUM (punteggio)

FROM manche

WHERE vincitore=

(SELECT email

FROM utente

WHERE nickname=?) AND idpartita=?

Consente di ottenere il totale guadagnato da un giocatore

SELECT nmanche

FROM partita

WHERE idpartita=?

Indica il numero di manche di una partita

SELECT COUNT (\*) –

(SELECT COUNT (\*)

FROM mossa

WHERE mossa.nome='UJ' AND mossa.idpartita=? AND idutente=

(SELECT email

FROM utente

WHERE nickname=?)

AS njolly

FROM mossa

WHERE mossa.nome='TJ' AND mossa.idpartita=? AND idutente=

(SELECT email

FROM utente

WHERE nickname=?)

Indica il numero di Jolly che un giocatore ha a disposizione

SELECT DISTINCT nome

FROM mossa

WHERE mossa.idpartita=? AND

(mossa.nome LIKE '%cns' OR

mossa.nome LIKE '%voc')

AND idmanche=

(SELECT nmanche

FROM partita

WHERE idpartita=?)

Indica le lettere della frase nel tabellone svelate nella partita

**Classe LoginQuery**

Contiene i metodi usati per effettuate le query durante la fase di login:

SELECT \*

FROM utente

WHERE email=? AND password=? AND admin=false AND oracodice=0 AND online=false

Verifica se esiste utente (non amministratore) con quella email e password nel database

SELECT \*

FROM utente

WHERE email=? AND password=? AND admin=true AND oracodice=0 AND online=false

Verifica se esiste un amministratore (non utente) con quella email e password nel database

UPDATE utente SET online=true

WHERE email=?

Aggiorna a true il campo online della tabella utente

UPDATE utente SET online=?

WHERE email=?

Aggiorna il campo online della tabella utente con il parametro passato (true o false)

**Classe PhraseQuery**

Contiene i metodi usati per effettuate le query per la gestione delle frasi:

INSERT INTO frase (tema, corpo)

VALUES (?, ?)

## Inserisce le nuove frasi nel database

DELETE FROM frase

WHERE idfrase=?

Elimina la frase in base all'id

UPDATE frase SET tema=?, corpo=?

WHERE idfrase=?

Aggiorna il tema e il corpo della frase in base all'id

INSERT INTO frase (idfrase, tema, corpo)

VALUES (?, ?, ?)

Inserisce una nuova frase

SELECT email

FROM partecipazione

WHERE idpartita=? AND tipo=0

Seleziona i partecipanti(giocatori) della partita

SELECT COUNT (\*) AS total

FROM frase

WHERE idfrase NOT IN

(SELECT DISTINCT idfrase

FROM manche NATURAL JOIN partita

INNER JOIN partecipazione ON partita.idpartita=partecipazione.idpartita

WHERE

(partecipazione.email=? OR partecipazione.email=? OR partecipazione.email=?) AND idfrase is not null)

Controlla che ci siano almeno 5 frasi che i giocatori non abbiano mai visto

SELECT email

FROM utente

WHERE admin=true

Seleziona tutti gli amministratori

DELETE FROM partecipazione

WHERE idpartita=?

Elimina la partecipazione e la partita

DELETE FROM partita

WHERE idpartita=?

Elimina la partita

SELECT nmanche

FROM partita

WHERE idpartita=?

Seleziona il numero della manche corrispondete alla partita

SELECT corpo

FROM frase NATURAL JOIN manche NATURAL JOIN partita

WHERE idmanche=? and partita.idpartita=? and partita.datafine=0

Seleziona il corpo della frase

SELECT idfrase, tema, corpo

FROM frase

ORDER BY idfrase

Indica la lista di tutte le frasi

**Classe ProfileUpdateQuery**

Contiene i metodi usati per effettuare le query di aggiornamento del profilo

UPDATE utente SET nome=?, cognome=?, nickname=?

WHERE email=?

Aggiorna i dati del profilo dell'utente

UPDATE utente SET password=?

WHERE email=?

Aggiorna password inviando una nuova password generata casualmente

UPDATE utente SET password=?, oracodice=0

WHERE email=? AND password=?

Se il codice corrisponde e non è scaduto allora aggiorno la password

SELECT \*

FROM utente

WHERE email=? AND password=?

Verifica se il codice corrisponde a quello contenuto nel Database

**Classe RegistrationQuery**

Contiene i metodi usati per effettuare le query di registrazione del profilo:

SELECT \*

FROM utente

WHERE email=? or nickname=?

Controlla se esiste un utente già registrato con questa email o nickname

INSERT INTO utente (email, nome, cognome, nickname, password, admin, oracodice, online)

VALUES (?, ? , ?, ?, ?, ?, ?,?)

Inserisce i dati dell'utente nel database

SELECT oracodice

FROM utente

WHERE email=?

Controlla che l'utente abbia effettivamente completato la registrazione altrimenti lo faccio registrare di nuovo

UPDATE utente SET nome=?, cognome=?, nickname=?, password=?, admin=?, oracodice=?

WHERE email=?

Se non ha confermato il codice di attivazione permetto di registrare di nuovo con la stessa email

INSERT INTO public.utente (email, nome, cognome, nickname, password, admin, oracodice, online)

VALUES (?, ?, ?, ?, ?, ?, ?, ?)

Inserisce i dati dell'amministratore nella tabella utente del database con il campo admin settato a true

SELECT oracodice

FROM utente

WHERE email=? AND password=?

Verifica se il codice inserito dall'utente corrisponde a quello nel database

DELETE FROM utente

WHERE email=?

Se sì, elimina utente

**Classe StatisticsQuery**

Contiene i metodi usati per effettuare le query delle statistiche

SELECT manche. punteggio, manche. vincitore, utente. nickname

FROM manche INNER JOIN utente ON manche.vincitore=utente.email

WHERE manche.punteggio >= ALL

(SELECT manche.punteggio

FROM manche)

ORDER BY idmanche ASC

LIMIT 1

Consente di ottenere il giocatore che detiene il primato di punteggio raggiunto per manche

SELECT SUM (manche.punteggio) AS score, manche.vincitore, utente.nickname

FROM manche INNER JOIN utente ON manche.vincitore=utente.email

GROUP BY (manche.idpartita, manche.vincitore, utente.nickname)

HAVING sum(manche.punteggio) >= ALL

(SELECT SUM (manche.punteggio)

FROM manche

GROUP BY (manche.idpartita,

manche.vincitore, utente.nickname))

ORDER BY manche.idpartita ASC

LIMIT 1

Consente di ottenere il giocatore che detiene il primato di punteggio raggiunto per partita

SELECT COUNT (\*) AS ntime, utente.email, utente.nickname

FROM manche INNER JOIN partita ON manche.idpartita=partita.idpartita

INNER JOIN partecipazione ON partita.idpartita=partecipazione.idpartita

INNER JOIN utente ON partecipazione.email=utente.email

GROUP BY (utente.email, utente.nickname)

HAVING COUNT (\*) >= ALL

(SELECT COUNT (\*) AS ntime FROM manche

INNER JOIN partita ON manche.idpartita=partita.idpartita

INNER JOIN partecipazione ON partita.idpartita=

partecipazione.idpartita

INNER JOIN utente ON partecipazione.email=utente.email

GROUP BY (utente.email, utente.nickname))

LIMIT 1

Consente di ottenere il giocatore che ha giocato più manche in assoluto

SELECT ROUND (AVG (punteggio),2) AS score, manche.vincitore, utente.nickname

FROM manche INNER JOIN utente ON manche.vincitore=utente.email

GROUP BY (manche.vincitore, utente.nickname)

HAVING ROUND (AVG (punteggio),2)>= ALL

(SELECT ROUND (AVG (punteggio),2)

FROM manche INNER JOIN utente ON manche.vincitore=utente.email

GROUP BY (manche.vincitore, utente.nickname))

LIMIT 1

Consente di ottenere il giocatore con la media più alta di punti acquisiti per manche

SELECT COUNT (\*) AS ntime, mossa.idutente, utente.nickname

FROM mossa INNER JOIN utente ON mossa.idutente=utente.email

WHERE mossa.nome='CT'

GROUP BY (mossa.idutente, utente.nickname)

HAVING COUNT (\*) >= ALL

(SELECT COUNT (\*) AS ntime FROM mossa

INNER JOIN utente ON mossa.idutente=utente.email

WHERE mossa.nome='CT'

GROUP BY (mossa.idutente, utente.nickname))

LIMIT 1

Consente di ottenere il giocatore che ha dovuto cedere più volte il turno di gioco in seguito ad errori

SELECT COUNT (\*) AS ntime, mossa.idutente, utente.nickname

FROM mossa INNER JOIN utente ON mossa.idutente=utente.email

WHERE mossa.nome='Perde'

GROUP BY (mossa.idutente, utente.nickname)

HAVING COUNT (\*) >= ALL

(SELECT COUNT (\*) AS ntime

FROM mossa INNER JOIN utente ON mossa.idutente=

utente.email

WHERE mossa.nome='Perde'

GROUP BY (mossa.idutente, utente.nickname))

LIMIT 1

Consente di ottenere il giocatore che ha perso tutto in seguito ad un giro di ruota per il maggior numero di volte

SELECT ROUND (AVG (moveCOUNT),2)

FROM

(SELECT COUNT (\*) AS moveCOUNT, manche.idmanche

FROM mossa NATURAL JOIN manche

WHERE mossa.nome <> 'CT'

GROUP BY (manche.idmanche)) AS count\_move

Consente di ottenere il numero medio mosse per manche con le quali viene indovinata una frase

SELECT mossa.nome, mossa.idutente, frase.corpo

FROM mossa INNER JOIN utente ON mossa.idutente=utente.email

INNER JOIN manche ON mossa.idmanche=manche.idmanche

AND mossa.idpartita=manche.idpartita

INNER JOIN frase ON manche.idfrase=frase.idfrase

WHERE mossa.nome LIKE '%cns' and mossa.punti >= ALL

(SELECT mossa.punti FROM mossa INNER JOIN manche ON mossa.idmanche=manche.idmanche AND mossa.idpartita=

manche.idpartita

WHERE mossa.nome

LIKE '%cns'

AND manche.idfrase IS NOT NULL)

LIMIT 1

Consente di mostrare la chiamata di consonante, riferita ad una manche già giocata, che ha portato all'acquisizione della maggior quantità punti, la frase misteriosa associata, e l'utente che ha fatto la mossa

SELECT COUNT (\*)

FROM manche NATURAL JOIN partecipazione

WHERE partecipazione.email=? AND tipo=0

Consente di mostrare il numero di manche giocate

SELECT COUNT (\*)

FROM partita INNER JOIN partecipazione ON partita.idpartita=partecipazione.idpartita

WHERE partecipazione.email=? AND tipo=0

Consente di mostrare il numero di partite giocate

SELECT COUNT (\*)

FROM manche NATURAL JOIN partecipazione

WHERE partecipazione.email=? and (tipo=1 or tipo=2)

Consente di mostrare il numero di manche osservate

SELECT COUNT (\*)

FROM partita INNER JOIN partecipazione ON partita.idpartita=partecipazione.idpartita WHERE partecipazione.email=?

AND (tipo=1 or tipo=2)

Consente di mostrare il numero di partite osservate

SELECT COUNT (\*)

FROM manche

WHERE manche.vincitore=?

Consente di mostrare il numero di manche vinte

SELECT COUNT (\*)

FROM partita

WHERE partita.winner=?

Consente di mostrare il numero di partite vinte

SELECT ROUND (AVG (score),2)

FROM

(SELECT sum(punteggio) AS score, idpartita, vincitore, nome

FROM manche INNER JOIN utente ON manche.vincitore=utente.email

WHERE vincitore=?

GROUP BY (idpartita, vincitore, nome)) AS sum\_score

Consente di mostrare il numero di manche giocate

SELECT ROUND (AVG (ntime),2)

FROM

(SELECT COUNT (\*) AS ntime, idpartita, idmanche, idutente

FROM mossa

WHERE nome='CT' AND idutente=?

GROUP BY (idpartita, idmanche, idutente)) AS count\_ct

Consente di mostrare il numero medio di passaggio turno per manche

SELECT ROUND (AVG (ntime),2)

FROM

(SELECT COUNT (\*) AS ntime, idpartita, idutente

FROM mossa WHERE nome='CT' AND idutente=?

GROUP BY (idpartita,idutente)) AS count\_ct

Consente di mostrare il numero medio di volte, per manche e partita, che ha dovuto cedere il turno di gioco

SELECT ROUND (AVG (ntime),2)

FROM

(SELECT COUNT (\*) AS ntime, idpartita, idmanche, idutente

FROM mossa

WHERE nome='Perde' AND idutente=?

GROUP BY (idpartita, idmanche, idutente)) AS count\_lose

Consente di mostrare il numero medio di volte, per manche ha che perso tutto

SELECT ROUND (AVG (ntime),2)

FROM

(SELECT COUNT (\*) AS ntime, idpartita, idutente

FROM mossa

WHERE nome='Perde' AND idutente=?

GROUP BY (idpartita, idutente)) AS count\_lose

Consente di mostrare il numero medio di volte, per partita che ha perso tutto

## **Documentazione del codice**

La documentazione del codice sorgente è stata preparata sfruttando l'applicazione Javadoc inclusa nel pacchetto JDK che ha permesso la generazione automatica di documentazione relativa a ogni metodo, ogni attributo (con indicatore di visibilità "public" o "protected") e a ogni classe (con indicatore di visibilità "public").

Per facilitare la comprensione del codice i nomi delle variabili utilizzate (siano essi variabili locali, parametri, attributi, metodi, classi, package, etc.) sono stati scelti in modo da essere auto esplicativi , richiamandone lo scopo e la funzione; i metodi molto complessi sono stati scomposti in metodi più semplici in modo che se ne possa comprendere la funzione con una sola occhiata; con lo stesso scopo, le classi con metodi o attributi troppo numerosi sono state semplificate disaggregandole in classi più piccole e con meno responsabilità. La documentazione generata automaticamente è stata controllata e integrata nelle parti non sufficientemente chiare con commenti ad hoc scritti in linguaggio naturale.

## **APache ant**

Apache Ant è un [software](https://it.wikipedia.org/wiki/Software) per l'[automazione del processo di build](https://it.wikipedia.org/wiki/Build_automation).

Simile a [Make](https://it.wikipedia.org/wiki/Make), ma principalmente orientato allo sviluppo in Java, Ant è un progetto [Apache](https://it.wikipedia.org/wiki/Apache_Software_Foundation) [open source](https://it.wikipedia.org/wiki/Open_source) ed è distribuito sotto [licenza di](https://it.wikipedia.org/wiki/Licenza_Apache) quest’ultimo.

La principale caratteristica di Ant è la [portabilità](https://it.wikipedia.org/wiki/Portabilit%C3%A0);

Una discrepanza comune tra le varie piattaforme si verifica nel modo in cui il *path* delle directory viene specificato. Ad esempio, [UNIX](https://it.wikipedia.org/wiki/UNIX) usa un *forward slash* (/) per delimitare i componenti di un path mentre [Microsoft Windows](https://it.wikipedia.org/wiki/Microsoft_Windows) un *backslash* (\).

La sintassi Ant permette all'autore del codice di usare una convenzione qualsiasi: back/forward slash per le directory, punto e virgola o due punti (; o :) per i separatori dei path (classpath).

Ant converte tutto nella forma più appropriata alla piattaforma corrente.

In seguito all'installazione di Java dal sito ufficiale (https://www.java.com/it/download) sarà necessario scaricare i file relativi ad Apache Ant (https://ant.apache.org/bindownload.cgi).

Una volta completato il download sarà possibile procedere alla creazione delle seguenti variabili d'ambiente relative al sistema:

1. ANT\_HOME: ha come valore la directory della cartella relativa ad Apache Ant (precedentemente scaricata)

2. JAVA\_HOME: ha come valore la directory della cartella relativa a Java (jdk)

Sarà poi necessario modificare la variabile di sistema 'Path' aggiungendo i seguenti percorsi:

1. %ANT\_HOME%\bin

2.%JAVA\_HOME%\bin

A questo punto sarà possibile, posizionandosi nella cartella 'Project B' relativa all'applicazione, avviare il terminale di sistema [alt + right-click -> apri finestra PowerShell qui (su Windows)] e digitare il comando 'ant':

Apache Ant, seguendo le istruzioni del file build.xml seguirà i seguenti passi:

* Creazione del database (si possono specificare i parametri ‘host’, ‘porta’, ‘nome database’, ‘username’ e password nel file build.properties)
* Creazione delle tabelle tramite il file db.sql presente nella directory:

project B\Server\_src

* Crezione della cartella ServerRdF che conterrà le cartelle bin, jar e doc
* Crezione della cartella ClientRdF che conterrà le cartelle bin, jar e doc
* Crezione della cartella AdminRdF che conterrà le cartelle bin, jar e doc
* La cartella bin conterrà i file generati utlizzado il comando ‘javac’ di Apache Ant
* La cartella jar conterrà i file generati utlizzando il comando ‘jar’ di Apache Ant
* La cartella doc conterrà i file generati utlizzando il comano ‘javadoc’ di Apache Ant

Il comando javac consente di compilare i file java.

Il comando jar consente di creare un file .jar eseguibile.

Il comando javadoc permette di generare la documentazione usando lo strumento javadoc.

<https://ant.apache.org/manual/> (manuale di Apache Ant)

<https://ant.apache.org/manual/Tasks/> (elenco dei tasks di Apache Ant)

**TEAM DI SVILUPPO**

All’analisi iniziale delle specifiche e dei requisiti si sono visti coinvolti attivamente tutti i componenti del gruppo, in questo processo sono state raccolte infatti tutte le idee e le possibili soluzioni implementative. La progettazione concettuale del database, la progettazione logica e l’implementazione del database, come anche lo sviluppo delle funzionalità lato server (seatInServer) sono state svolte dallo studente Peluso Andrea. L’implementazione lato Client e l’interazione con l’interfaccia grafica sono state effettuate dallo studente Cremonesi Luca, con la collaborazione di Peluso Andrea nella realizzazione della comunicazione Client/Server.

Nella parte UML gli UCD (Use Case Diagram), i Class Diagrams, i Sequence Diagrams sono state realizzati da tutti gli studenti del gruppo.

Infine la stesura della relazione con l’integrazione di diagrammi UML rappresentati con l’ausilio del software Lucidchart è stata realizzata dagli studenti Ciapponi Alessandro e Valenti Dezsö in quanto ritenuta una delle componenti principali del progetto.

Durante lo svolgimento dell’intero progetto i componenti del gruppo hanno comunicato e si sono confrontati tra di loro, anche sul lavoro svolto individualmente, utilizzando una piattaforma di messaggistica istantanea chiamata Discord. Oltre ad utilizzare questa piattaforma abbiamo avuto modo di interagire e collaborare trovandoci fisicamente, numerose volte, in ateneo, biblioteche comunali o altri luoghi, cercando sempre però di venire incontro alle esigenze di ognuno poiché provenienti da zone diametralmente opposte gli uni dagli altri.

**Fonti**

Sono state utilizzate come fonti:

* Materiali Didattici forniti durante l’Anno Accademico
* Sito Oracle, per una definizione completa di auditing:

<https://docs.oracle.com/cd/E11882_01/server.112/e10575/tdpsg_auditing.htm#TDPSG50000>

* Community di PostgreSQL, per la selezione del pgAdmin migliore da utilizzare:

https://www.postgresql.org/community/

1. La registrazione è necessaria per impedire che a un osservatore, diventando in seguito giocatore, vengano mostrate frasi “misteriose” di cui conosce già la soluzione: le frasi sono infatti scelte casualmente da un campione limitato di frasi preimpostate [↑](#footnote-ref-1)
2. <http://www.postgresql.org> [↑](#footnote-ref-2)
3. <http://jdbc.postgresql.org/download.html> [↑](#footnote-ref-3)