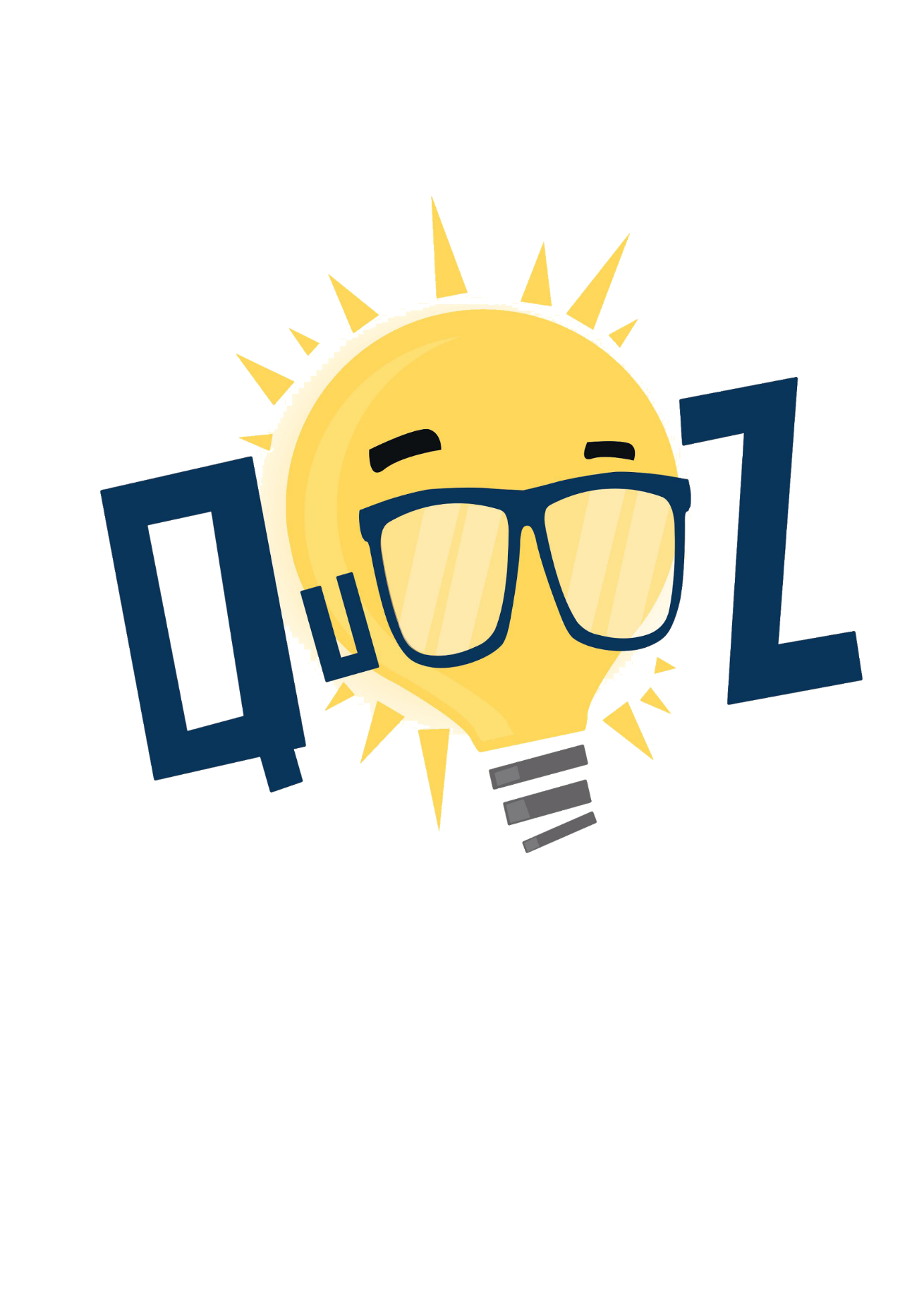
Progetto di Web Technologies

D’Amicis Salvatore

Marone Christian

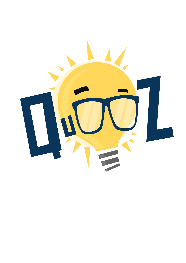
Romeo Matteo

Urzino Davide

****

**Introduzione**

Fasi di lavoro



Visto e considerato che il database è una delle parti fondamentali di Quooz, la prima fase è stata la realizzazione di un modello E-R che potesse rappresentarlo. In secondo luogo sono stati generati tutti i package e le classi con i relativi metodi anche se privi di codice al loro interno. Uno alla volta, tutti i metodi sono stati implementati e integrati tra di loro al fine di realizzare la logica applicativa del programma. Infine sono state integrate le pagine html, le servlet e le jsp che hanno dato vita al sito vero e proprio.

**IMPLEMENTAZIONE**

Modello E - R

C:\Users\Christian\Downloads\ER.png

Il modello E-R dà una rappresentazione della struttura concettuale dei dati.

**Author** (username, password, name, surname, dateOfBirth, email)

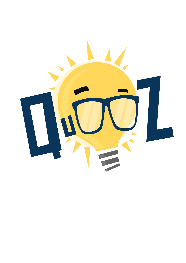
**Question** (id, text, difficulty, image, category, userAuthor(FK))

**Option** (id, questionId(FK), text, correct, image)

**Answer** (timestamp, userPlayer (FK), optionId(FK))

**Player** (username, password, name, surname, dateOfBirth, email, score, level, correctSequence)

**Log** (date, action, userPlayer (FK))



**CORE**

Database

**Database connection**

Questa classe si occupa di gestire le connessioni al database. Utilizza un pool di connessioni, istanziato dal metodo setupDataSource(), che vengono prelevate tramite getConnection() ed utilizzate per poter effettuare interrogazioni e modifiche sul database.

Question

**Question creator e option creator**

Sono le due classi che si preoccupano di creare oggetti di tipo Domanda e Opzione. Quando l’autore vorrà inserire una nuova domanda, questi metodi saranno chiamati in modo da creare la domanda e le relative opzioni.

**Question handler**

Essa gestisce tutta la logica delle domande (fa uso del pattern Singleton). Si occupa di:

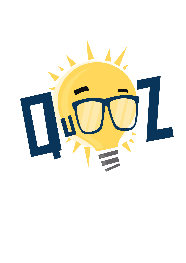
* Fornire la lista delle domande agli autori (getQuestionList())
* Fornire una domanda valida ai giocatori durante la partita (getGameQuestion())
* Prelevare una domanda dal database quando necessario (getQuestionById())
* Prelevare le opzioni dal database supportando getQuestionById() (getOptions())
* Inserire una nuova domanda nel database (createNewQuestion())
* Aggiungere nuove opzioni relative ad una domanda supportando il metodo precedente (addTextualOptionsToDb(), addImageOptionsToDb())
* Modificare le domande nel database (editQuestion())
* Rimuovere le domande dal database (deleteQuestion())

System

**Game handler**

È la classe che prende in carico la gestione del gioco per tutti i giocatori connessi:

* Fornisce le domande al giocatore (getQuestion());
* Permette di rispondere alla domanda verificandone la correttezza (answerQuestion(), createFeedback());
* Modifica le statistiche riguardanti il giocatore sulla base delle risposte che ha dato (updateStats())
* Permette di visualizzare classifica e percentuali legate alle risposte del giocatore (getLeaderbord(), getCorrectAnswerHistory(), getCategoriesStats())



System Log

**Log record handler**

Essa si occupa della gestione del file di log del giocatore attraverso l’inserimento in memoria delle  
azioni svolte dallo stesso (stabilite dall’enumeration Action):

* Sign in;
* Sign out;
* Risposta alla domanda.

Ha due funzionalità principali, la creazione del log record ad azione avvenuta e la sua aggiunta al database.

**Log record**

Contiene le informazioni riferite all’azione del giocatore, cioè: la tipologia di azione, l’istante temporale nel quale si è verificata e l’identificativo del giocatore (username).

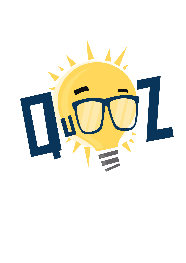
User

**Author handler e player handler**

Queste classi si occupano di gestire la logica legata ai due tipi di utenti, nel particolare:

* La creazione di un’istanza di tipo Author/Player (createAuthor(), createPlayer())
* L’accesso all’area personale (signInAuthor(), signInPlayer())
* L’uscita da questa stessa (signOutAuthor(), signOutPlayer())

Entrambe fanno uso del pattern Singleton e tengono traccia degli utenti connessi.

La classe PlayerHandler, inoltre, ha funzionalità aggiuntive:

* Restituisce un oggetto di tipo Player aggiornato all’ultima domanda a cui ha risposto (getUpdatedPlayer()) da updatePlayer()
* Fornisce la possibilità di registrare un nuovo account (signUpNewPlayer()) e di eliminarlo (unregisterPlayer())
* La registrazione è supportata da:
  + isFieldAvaiable(): verifica che username o e-mail non siano già presenti nel database
  + generateVerificationCode() e sendVerificationCode(): la prima genera e la seconda invia un codice di verifica alla e-mail di un utente che intende registrarsi al gioco allo scopo di accertarne l’identità.

**SERVLET**

**AddQuestion**

Questa servlet riceve dalla richiesta i parametri di una nuova domanda da aggiungere al database e si occupa quindi del suo inserimento

**CheckAnswer**

Si occupa della verifica della correttezza di una risposta data da un giocatore.

**DeleteAccount**

E’ la servlet che si occupa di eliminare dal database l’account di un giocatore che lo desidera.

**DeleteQuestion**

Il suo compito è quello di eliminare una domanda dal database su richiesta di un autore.

**Leaderboard**

Questa servlet genera un file di tipo XML contenente la classifica dei giocatori.

**Login**

Si occupa di permettere il login ad autori e giocatori (riconoscendo il loro status) nel caso abbiano le credenziali corrette.

**Logout**

E’ la servlet che si occupa di eliminare dalla sessione un giocatore o un autore che desidera effettuare il logout

**SignUp**

Questa servlet si occupa della fase iniziale della registrazione di un giocatore verificando che username e email da lui inseriti siano disponibili nel database, generando il codice di verifica e inviando la mail che lo contiene al potenziale giocatore

**VerificationCode**

La parte conclusiva della registrazione viene gestita da questa servlet che verifica se il codice di registrazione inserito dall’utente sia uguale a quello a lui inviato per mail, in tal caso la registrazione andrà a buon fine, altrimenti fallirà.

**ImageServlet**

È la servlet che si occupa di fornire le immagini alle pagine jsp che lo richiedono estraendole dal database.

