**Documentazione del codice**

*package model*

**Hackathon:** La classe Hackathon rappresenta un evento di programmazione competitiva e gestisce: Informazioni base, partecipanti, team e giudici, una classifica per valutare i team. Struttura e Campi Principali. Inizialmente vengono configurati gli attributi, in seguito vengono inizializzati attraverso i costruttori. Infine vengono impostati i metodi per acceder e modificare i campi (getter e setter);

**Classifica:** gestisce una graduatoria ordinata di team partecipanti a un hackathon, fornendo un meccanismo per aggiungere team e una visualizzazione testuale della classifica (toString()). La lista vuota viene inizializzata con il costruttore, il metodo “aggiungiTeam” aggiunge un team all’interno della lista e il metodo toString() genera una visualizzazione della classifica dei team: controlla se la lista è vuota ed eventualmente restituisce un messaggio di avviso, altrimenti costruisce una stringa per ogni team che trova nella lista mostrando informazioni come posizione, nome e punteggio di ogni team.

**Documento:** rappresenta un documento associato a un team, presenta informazioni come id, titolo e idTeam. il costruttore viene lasciato vuoto utile per la creazione di oggetti prima di settare i valori, successivamente vengono impostati i vari setter e getter per accedere e modificare i campi e il metodo toString che restituisce solo il titolo.

**Utente:** rappresenta un utente generico del sistema che presenta come attributi un id univoco, un nome, email, password e un ruolo. La classe utilizza 5 costruttori di diverse esigenze: registrazione nuovo utente, per la lettura dal database, per un caso specifico senza l’utilizzo della password, un caso specifico senza id e un caso specifico senza informazioni di base. Infine imposta i vari getter e setter

**Giudice**: classe figlia della classe utente e ne eredita i campi. Chiama il costruttore della classe padre attraverso la keyword super e impone come ruolo fisso “giudice” . Il secondo costruttore viene utilizzato per caricare un giudice nel database per cui non è necessaria la password:

**Partecipante:** anche questa classe estende la classe utente e ne eredita i campi, ma in più presenta anche l’attributo teamId per identificare il team di appartenenza dei partecipanti. Vengono definiti due costruttori: il primo è usato quando l'utente è già registrato nel database quindi chiama il costruttore di Utente dei parametri, impostando automaticamente il ruolo a "partecipante". Il secondo è usato quando si vuole caricare partecipante nel database, quindi la password non è necessaria. Infine vengono impostati il getter e il setter del nuovo parametro teamId.

**Team:** la classe presenta come attributi le informazioni che ogni team deve avere: id, nome, descrizione, progresso, riferimento all’hackathon a cui partecipa e una lista di voti. Questi parametri vengono inizializzati con un costruttore e poi vengono impostati i vari getter e setter per ogni attributo della classe. CI sono metodi aggiuntivi come getPuntitotali che se il voto non c’è restituisce 0, altrimenti per ogni voto che scorre all’interno del ciclo viene chiamato il metodo getPunteggio con il quale viene incrementata la somma. un altro metodo è getPunteggiomedio per calcolare la media dei voti controllando sempre se ci sono voti, così in caso restituisce 0 e fa lo stesso incremento della somma del metodo getPuntitotali, però restituisce la somma fratto il metodo size della classe voto che viene chiamato per conoscere il numero dei voti assegnati.

**Voto:** Questa classe rappresenta un voto assegnato da un giudice a un team durante un Hackathon, presenta come informazioni l’Id del voto nel database, l’Id del team che verrà valutato, l’Id del giudice che assegna il voto e il punteggio assegnato. Gli attributi vengono inizializzati tramite il costruttore e vengono impostati i vari getter e setter. Vengono utilizzati altri metodi come il metodo metodo toString, che restituisce una presentazione leggibile dell’oggetto voto, il metodo equals, che confronta due oggetti Voto in base a idTeam, idGiudice e punteggio, e il metodo hashCode che genera un valore numerico (hash) che rappresenta l'oggetto in modo compatto.

*Package controller*

**Controller:** La classe Controller gestisce tutte le operazioni principali del sistema di gestione degli hackathon, inclusa l'autenticazione, la gestione degli hackathon, dei team, dei partecipanti, dei giudici e delle votazioni. Inizializza tutti i DAO necessari e carica l'hackathon esistente dal database o ne crea uno nuovo se non esiste. I metodi della gestione dell’hackathon sono: creaHackathon(), che crea un nuovo hackathon con i parametri base o completi, e creaHackathonDefault(), che crea un hackathon con valori predefiniti (usato internamente). Vengono gestiti i processi di autenticazione attraverso i metodi: registraUtente che registra un nuovo utente nel sistema, il metodo login che esegue appunto il login di un utente, e getUtenteLoggato(), che restituisce l'utente attualmente loggato. Viene definita la gestione dei team come la loro creazione attraverso il metodo creaTeam, che crea un team e associa dei partecipanti gestendo anche il caso in cui i partecipanti per team superino il numero presvisto. il metodo getlistaTeam() che restituisce la lista dei team che partecipano. Per quanto riguarda la gestione di giudici e partecipanti la classe presenta metodi per aggiungere partecipanti e aggiungere giudici: aggiungiGiudice e aggiungiPartecipante. Per la gestione dei documenti sono presenti dei metodi che si occupano di inviare documenti o restituirli cioè: inviaDocumento e getDocumentiDiTeam. Si occupa anche delle funzionalità di assegnazione di un voto da parte di un giudice e presenta anche dei metodi di supporto come getTeamIdByPartecipante che tova l'ID del team a cui appartiene un partecipante, caricaGiudiciDaDB() che ricarica la lista dei giudici dal database e getHackathon() che restituisce l'hackathon corrente.

*Package gui*

**Gui:** rappresenta l’interfaccia grafica dell’applicazione. Le componenti principali sono panelLogin(JPanel) per il form di login/registrazione, il panelOperativo (JPanel) con operazioni base (visibile dopo login), txtNome, txtEmail, txtPassword(campi per l'input dell'utente), comboRuolo(JComboBox), un menù a tendina per selezionare il ruolo tra organizzatore, giudice e partecipante e il controller: Istanza della classe Controller. Il costruttore Inizializza la finestra principale con: titolo "Hackathon Manager", dimensioni 450x350 pixel, posizione centrata, layout a BorderLayout e pannello di login iniziale. I metodi implementati sono destinati alla creazione del pannello principale dell’operazione: creaPanelLogin() che rea il pannello di login/registrazione con nome, email, password, ruolo e pulsante di login, la logica del pulsante è implementata con una verifica che tutti i campi siano stati compilati in modo da provare il primo login, se il login fallisce allora prova la registrazione, se l’autenticazione ha successo apre la finestra specifica per il ruolo. Successivamente il metodo implementato è creaPanelOperativo() che crea un pannello con pulsanti per operazioni generiche: Crea Team permette a un partecipante di creare un nuovo team, lista Giudici Mostra l'elenco dei giudici lista Partecipanti mostra l'elenco dei partecipanti, valuta Team permette a un giudice di assegnare un voto a un team e infine il logout che fa in modo che l’utente orni alla schermata di login. Infine implementa apriFinestraRuolo() che apre una finestra specifica in base al ruolo dell'utente: FinestraOrganizzatore per gli organizzatori, FinestraGiudice per i giudici e FinestraPartecipante per i partecipanti.

**FinestraOrganizzatore:** rappresenta l'interfaccia grafica dedicata agli organizzatori nell'applicazione Hackathon Manager. Si tratta di una finestra Swing che estende JFrame e fornisce tutte le funzionalità necessarie per gestire un hackathon in modo completo. Il costruttore accetta tre parametri: un'istanza del Controller, l'utente organizzatore loggato e la finestra di login principale per permettere il ritorno alla schermata iniziale. L'interfaccia è organizzata con un layout a griglia (7 righe, 1 colonna) che contiene sette pulsanti principali. Il primo pulsante "Crea Hackathon" permette di definire un nuovo evento richiedendo titolo, sede, numero massimo di partecipanti e numero massimo di team, con appositi controlli per gli input numerici. Il secondo pulsante "Invita Giudice" aggiunge direttamente un nuovo giudice al sistema chiedendo nome e email, con password generata automaticamente Analogamente, il terzo pulsante "Invita Partecipante" svolge la stessa funzione per i partecipanti. Il quarto pulsante "Visualizza Team" recupera e mostra la lista completa dei team con il relativo progresso in percentuale, gestendo anche il caso in cui non siano presenti team. Il quinto pulsante "Classifica" visualizza la graduatoria basata sulla media dei voti, mostrando per ogni team posizione, nome, punteggio medio formattato a due decimali e numero di voti ricevuti, in ordine decrescente di punteggio. Il sesto pulsante "Informazioni Hackathon" genera un report completo con dettagli base (titolo, sede, limiti) e statistiche aggiornate (numero team, giudici e partecipanti), presentato in una JScrollPane per una comoda consultazione anche di contenuti lunghi. Infine, il settimo pulsante "Logout" chiude la finestra corrente e ripristina la schermata di login.

**FinestraGiudice:** rappresenta l'interfaccia grafica dedicata ai giudici nell'applicazione Hackathon Manager. Si tratta di una finestra Swing che estende JFrame e fornisce le funzionalità essenziali per la valutazione dei team e il monitoraggio della competizione. Il costruttore accetta tre parametri: un'istanza del Controller, l'oggetto Utente rappresentante il giudice loggato, e il riferimento alla finestra di login principale per permettere il logout. L'interfaccia presenta un layout organizzato in una griglia 4x1 che contiene i quattro pulsanti principali. Il primo pulsante "Visualizza Team" recupera dal database e mostra l'elenco completo dei team partecipanti, indicando per ciascuno il nome e la percentuale di progresso, con un apposito messaggio nel caso non siano presenti team. Il secondo pulsante "Assegna Voto a un Team" implementa il cuore della funzionalità di valutazione: dopo aver caricato la lista dei team disponibili, presenta all'utente una dialog box con un JComboBox per selezionare il team (mostrando sia nome che ID) e uno JSpinner per inserire il voto (limitato al range 1-10 con valore predefinito 7). Una volta confermato, il sistema estrae l'ID del team dalla stringa selezionata, verifica la validità del voto e lo assegna tramite il controller, mostrando appositi messaggi di successo o errore. Il terzo pulsante "Visualizza Classifica" recupera e mostra la graduatoria aggiornata basata sulla media dei voti, presentando per ogni team la posizione, il nome, la media voti formattata a due decimali e il numero di valutazioni ricevute, tutto in un ordinamento decrescente. Il quarto pulsante "Logout" semplicemente ripristina la finestra di login e chiude l'interfaccia corrente.

**FinestraPartecipante:** rappresenta l'interfaccia grafica dedicata ai partecipanti nell'applicazione Hackathon Manager. Estendendo JFrame, questa finestra offre tutte le funzionalità necessarie per gestire la partecipazione a un hackathon. Il costruttore accetta tre parametri fondamentali: un'istanza del Controller, l'oggetto Utente rappresentante il partecipante loggato, e un riferimento alla finestra di login principale per permettere il corretto logout. L'interfaccia è organizzata con un layout a griglia 6x1 che contiene sei pulsanti principali. Il primo pulsante "Crea Team" permette al partecipante di fondare un nuovo team specificandone il nome, con controlli sull'input e feedback immediato sull'esito dell'operazione. Il secondo pulsante "Invia Documento" offre una ulteriore funzionalità: dopo aver verificato l'appartenenza a un team, presenta una dialog box completa con campo per il nome del documento e un JComboBox per selezionarne il tipo (Presentazione, Codice sorgente, Documentazione o Altro), garantendo così una corretta categorizzazione dei file inviati. Il terzo pulsante "Visualizza Documenti" mostra tutti i documenti associati al team del partecipante, con un appropriato messaggio nel caso non ve ne siano. Il quarto pulsante "Visualizza Membri dei Team" implementa una funzionalità dettagliata: recupera dal database l'elenco completo di tutti i team con i relativi membri, mostrando per ciascuno il nome, l'email dei partecipanti e il conteggio aggiornato (es. "2/3 membri"), il tutto presentato in una JScrollPane per agevolare la consultazione anche con molti team.

Il quinto pulsante "Unisciti a un Team" gestisce in modo completo la transizione tra team: verifica prima se l'utente è già in un team (chiedendo conferma per il cambio), mostra solo i team con meno di 3 membri (rispettando così il limite implementato), e permette una selezione facile attraverso un JComboBox che mostra anche il numero corrente di membri. Il sesto e ultimo pulsante "Logout" semplicemente ripristina la finestra di login e chiude l'interfaccia corrente.

*Package database*

**ConnessioneDatabase:** gestisce le connessioni a un database PostgreSQL utilizzando il pattern Singleton per garantire un'unico punto di accesso. Implementa le seguenti caratteristiche: configura la connessione tramite URL " con credenziali utente e password; utilizza un costruttore privato e il metodo statico getInstance() per il controllo dell'istanza; gestisce la connessione verificando lo stato con isClosed() e ripristinandola se necessario; gestisce gli errori stampando su System.err nel costruttore e propagando le SQLException tramite getConnection(); mantiene le credenziali hardcoded come costanti statiche final. Il metodo getConnection() assicura che la connessione sia sempre attiva e disponibile, rigenerandola automaticamente se risulta chiusa o non inizializzata. La classe si occupa solo della gestione base della connessione, delegando la gestione delle transazioni e degli statement SQL alle classi chiamanti.