**Documentazione del Progetto: Sistema di Gestione Hackathon**

**Corso:** Informatica - L31  
**Modulo:** Progettazione e Modellazione Software  
**Anno accademico:** 2024/2025  
**Data consegna:** 24 aprile 2025

**Componenti del gruppo:**

* Alessandro Loffredo (N86005409) – 2° Anno
* Domenico Guerrera (N86005579) – 1° Anno
* Michele D’Orso (N86005448) – 1° Anno

**1. Introduzione**

Il presente documento descrive la fase iniziale di analisi e modellazione per lo sviluppo di un sistema dedicato alla gestione e organizzazione di eventi Hackathon. L'obiettivo principale è creare una piattaforma efficiente e intuitiva, in grado di gestire eventi competitivi di programmazione

**2. Class Diagram**

**Immagine che contiene testo, diagramma, schermata, Piano

Il contenuto generato dall'IA potrebbe non essere corretto.**

**3. Descrizione e Analisi delle Classi**

**Classe: PlatformAdmin**

**Scelta progettuale:** Necessità di avere un’entità che amministri gli eventi e scelga gli organizzatori tra gli utenti.

* **Resposabilità:** Crea gli aventi, sceglie gli organizzatori;
* **Attributi:** username, password;
* **Metodi principali:** createHackathon().

**Classe: Person**

**Scelta progettuale:** Necessità di avere una generalizzazione dell’utente. In tal modo creiamo una distinzione tra gli utenti registrati o meno alla piattaforma. La persona può comunque navigare nella piattaforma.

* **Resposabilità:** Possibilità di iscrizione alla piattaforma;
* **Attributi:**
* **Metodi principali:** regPlatform().

**Classe: User**

**Scelta progettuale:** Specializzazione di persona. È necessaria per avere una generalizzazione di coloro che interagiscono con la piattaforma. Questa classe permette all’utente di svolgere in vari hackathon diversi ruoli.

* **Responsabilità:** Possibilità di registrarsi all’hackathon, possibilità di rispondere all’invito come giudice;
* **Attributi principali:** fName, lName, birthDate, username, password, isBusy;
* **Metodi principali:** regHackaton(), answerInvitation().

**Classe: Organizer**

**Scelta progettuale:** Necessità di avere una classe che si occupa di gestire gli hackathon, di determinare vari aspetti generali.

* **Responsabilità:** Invita i giudici all’hackathon, apre le registrazioni dell’hackathon;
* **Attributi:** Ereditati da user;
* **Metodi principali:** inviteJudge(), openRegDate().

**Classe: Judge**

**Scelta progettuale:** Specializzazione dell’user necessaria per avere all’occorrenza un giudice per l’hackathon e permettergli comunque di partecipare ad altri hackathon

* **Responsabilità:** Descrivere il problema dell’hackathon, esaminare i documenti, assegnare i voti;
* **Attributi principali:** ereditati da Utente;
* **Metodi principali:** assignMark(), describeProblem(), examineDoc().

**Classe: Participant**

**Scelta progettuale:** Necessità di avere una specializzazione di utente che identificasse gli utenti iscritti ad un determinato hackathon.

* **Responsabilità:** Chiedere di partecipare ad un team, rispondere alla richiesta;
* **Attributi principali:** ereditati da Utente;
* **Metodi principali:** sendRequest(), answerInvRecived(), manageDoc(), createTeam().

**Classe: Hackathon**

**Scelta progettuale:** Necessità di avere una classe che descriva proprietà e caratteristiche di ogni singolo eventi hackathon.

* **Responsabilità:** Pubblicare le classifiche
* **Attributi principali:** title, venue, startDate, endDate, maxRegistration, maxTeamParticipant, problemDescription, startRegDate, endRegDate, regCounter;
* **Metodi principali:** publishRanking(), addRegistration(), endHackathon().

**Classe: Team**

**Scelta progettuale:** Necessità di avere una classe che permetta di identificare i singoli team partecipanti all’hackathon.

* **Responsabilità:** Pubblicare periodicamente un documento di aggiornamento;
* **Attributi principali:** finalMark, nickname;
* **Metodi principali:** addDoc(), avgMark().

**Classe: Documento**

**Scelta progettuale:** Necessità di avere una classe che determini i vari documenti per essere poi giudicati dai giudici.

* **Responsabilità:**
* **Attributi principali:** description, date, comment;
* **Metodi principali:**

**4. Associazioni e gerarchie**

**PlatformAdmin**

* **“chooses”:** associazione N a N con **User**. Più admin possono scegliere più utenti come amministratori dei vari hackathon.

Abbiamo ritenuto opportuno inserire un’associazione tra queste due classi perché è necessario avere qualcuno che scelga gli organizzatori dei vari Hackathon.

**Person**

Non ha associazioni, ma soltanto una sottoclasse **User**.   
Presenza della sottoclasse User perché anche un utente non registrato alla piattaforma può navigarci all’interno.

**User**

* **“invites”:** associazione 1 a N con **Organizer**. Più utenti possono essere invitati come giudici da un organizzatore.

Abbiamo ritenuto opportuno fare questa associazione tra User e Organizer perché quest’ultimo invita gli utenti ad essere giudici che però diventano tali soltanto nel momento in cui accettano l’invito.

Presenta 3 sottoclassi: organizer, judge, participant.

La presenza delle sottoclassi è importante perché le tre classi specializzate sono comunque sempre degli utenti che nei vari Hackathon possono interscambiarsi i ruoli.

**Organizer**

* **“organizes”:** associazione 1 a 1 con **Hackathon**. Un organizzatore può organizzare un solo Hackathon alla volta.  
  Abbiamo ritenuto opportuno che un organizzatore impegnato nell’organizzazione di un Hackathon non potesse organizzarne altri contemporaneamente.
* **“invites”:** associazione 1 a N con **User.**

**Judge**

* **“judges”:** associazione 1 a N con **Hackathon**. Un Hackathon può avere più giudici.  
  Abbiamo ritenuto opportuno che un gruppo di giudici coinvolti in un Hackathon potessero giudicarne solo uno alla volta.
* **“examines”:** associazione N a N con **Team**. Il corpo giudici esamina i documenti dei team.  
  Abbiamo ritenuto opportuno che i giudici si interfacciassero con i team per esaminare i documenti in modo tale da poter riconoscere l’appartenenza di un documento ad uno specifico team sfruttando la loro associazione.
* **“asseignsMark”:** associazione N a N con **Team**. Il corpo giudici assegna voti ai team.  
  Abbiamo ritenuto opportuno che ogni giudice potesse votare ogni team e il voto finale del team sia dato dalla media dei voti dei singoli giudici.

**Participant**

* **“requests”:** associazione **ricorsiva** 1 a N. Un partecipante può chiedere ad un altro partecipante di partecipare al team. I team possono ricevere più richieste;  
  Abbiamo ritenuto opportuno che ogni partecipante potesse richiedere o meno di partecipare ad un team e che i team possano ricevere più richieste di partecipazione.
* **“aggregation”:** associazione (per definizione 1 a N) con **Team**. Un team è un insieme di partecipanti che esistono anche senza il team stesso.  
  Abbiamo ritenuto opportuno che un partecipante non ancora inserito all’interno di un Team potesse comunque esistere all’interno dell’Hackathon. Nel caso in cui le iscrizioni vengano chiuse e un partecipante sia rimasto solo, verrà creato automaticamente un team composto dal singolo partecipante.

**Hackathon**

* **“organizes”:** associazione 1 a 1 con **Organizer**;
* **“judges”:** associazione 1 a N con **Judge;**
* **“partecipates”:** associazione 1 a N con **Team**. Più team possono partecipare ad un solo hackathon per volta.  
  Abbiamo ritenuto opportuno che i team possano iscriversi ad un solo Hackathon alla volta.

**Team**

* **“examineDoc”:** associazione N a N con **Judge**;
* **“asseignsMark”:** associazione N a N con **Judge**;
* **“partecipates”:** associazione 1 a N con **Hackathon**;
* **“produces”:** associazione 1 a N con **Document**. Un team produce più documenti di aggiornamento;  
  Abbiamo ritenuto opportuno che i team, per mostrare i loro progressi, potessero produrre più documenti dello svolgimento del problema.
* **“aggregation”:** associazione (per definizione 1 a N) con **Participant.**

**Document**

* **“produces”:** associazione 1 a N con **Team.**

**5. Conclusioni**

Il risultato ottenuto da questo primo step del progetto è il modello concettuale in UML, elemento fondamentale per comprendere come le varie classi della struttura andranno a comunicare tra loro, e una prima implementazione di queste ultime in java. Questa è quindi la base su cui si andrà a costruire tutta la struttura del software per la gestione degli eventi hackathon.

**6. Link Repository GitHub**

https://github.com/AlessandroLoffredo/PROGETTOPOO.git