

DOCUMENTAZIONE HACKATHON

CAPITOLI DA AGGIUNGERE ALLA FINE:

ANALISI DELLA TRACCIA:

Nella traccia possiamo identificare: le **classi**, le **associazioni** e gli **attributi**, le **generalizzazioni** e le **responsabilità**

Un **hackathon**, ovvero una "maratona di hacking", è un evento durante il quale **team di partecipanti** si sfidano per progettare e implementare nuove **soluzioni** basate su una certa tecnologia o mirate a un certo ambito applicativo.

Ogni hackathon ha un **titolo identificativo**, si svolge in una certa **sede** e in un certo **intervallo di tempo** (solitamente 2 giorni) e ha un **organizzatore** specifico (**registrato alla piattaforma**).

L'**organizzatore** seleziona un gruppo di **giudici** (**selezionati tra gli utenti della piattaforma, invitandoli**). Infine, **l'organizzatore** apre le registrazioni, che si chiuderanno 2 giorni prima dell'evento. Ogni evento avrà un **numero massimo di iscritti** e una **dimensione massima del team**.

Durante il periodo di registrazione, gli **utenti** possono **registrarsi** per **l'Hackathon** di loro scelta (eventualmente registrandosi sulla piattaforma se non lo hanno già fatto). Una volta iscritti, gli utenti possono **formare team**. I **team** diventano definitivi quando si chiudono le iscrizioni. All'inizio dell'hackathon, i **giudici** **pubblicano una descrizione del problema** da affrontare.

Durante **l'hackathon**, i **team** lavorano separatamente per risolvere il problema e **devono caricare periodicamente gli aggiornamenti sui "progressi"** sulla piattaforma come **documento**, che può essere esaminato e commentato dai giudici. Alla fine **dell'hackathon**, ogni **giudice** **assegna un voto** (da 0 a 10) a ciascun team e la piattaforma, dopo aver acquisito tutti i voti, pubblica le **classifiche** dei team.

SCELTE IMPLEMENTATIVE DEL DIAGRAMMA: CLASSI E RESPONSABILITÀ:

Abbiamo scelto di creare una classe hackathon con varie informazioni, la quale non sarà munita di funzioni set perché un Hackathon una volta creato non dovrebbe poter cambiare le proprie informazioni.

Creiamo una classe **Team** con un nome che è una composizione di **Partecipanti**, i quali possono invitare e accettare/rifiutare inviti ad un team, inoltre Partecipante è una delle tre specializzazione del generico utente.

Utente è una classe munita di email, password e utente univoco. Tutti questi attributi sono protected e hanno funzioni setter e getter, a parte password che non ha una getPassword in quanto a livello teorico le password dovrebbero sempre essere offuscate.

Il **Giudice** è un'altra classe che estende Utente, le sue responsabilità sono quelle di pubblicare il problema, commentare i progressi dei team e dare voti. Questa classe viene invitata dalla classe **Organizzatore**, che può anche pubblicare la classifica e aprire le registrazioni.

ASSOCIAZIONI:

Ci sono innanzitutto 3 classi associative: il **Voto**, il **Commento** e la **Classifica**.

La classifica è una classe associativa perché è determinata dai team(e i loro voti) e l'hackathon a cui sono iscritti.

Il commento è una classe associativa presente tra il documento e il giudice. Nel nostro scenario inoltre, i giudici possono scrivere un commento a ogni documento e un singolo documento può essere commentato da più giudici.

Il voto è dato dal giudice ed è associato ad un team, ovviamente un giudice da un voto a più team e il team riceve un voto da più giudici.

Il resto delle associazioni sono abbastanza intuitive (team e hackathon, un team può partecipare a più hackathon e un hackathon ha più team di partecipanti; un organizzatore invita più giudici e i giudici possono essere invitati da più organizzatori ad hackathon diversi...)

IMPLEMENTAZIONE IN JAVA:

Per l'implementazione in java delle associazioni abbiamo seguito questa logica:

Associazioni 1-1:

Nelle due classi sono presenti un attributo riferito all'altra classe.

In ogni classe che avesse il numero di **partecipazione** pari ad uno abbiamo inserito un costruttore che desse un valore agli prima menzionati.

Associazioni N-N:

Nelle due classi presenti, hanno un ArrayList di riferimenti all'altra classe.

Nelle classi che necessitano una **partecipazione** minima abbiamo inserito un costruttore che aggiungesse almeno un elemento alla ArrayList.

Composizione:

La classe composta presenta una ArrayList di riferimenti all'altra, mentre l'altra presenta un attributo e un costruttore che si riferiscono alla composta.

GUI:

Per la GUI è stato utilizzato Swing UI Designer, un plugin di IntelliJ Idea.

Ad alto livello la GUI permette ad un utente di eseguire l'accesso e di registrarsi.

Dopo l'accesso ogni utente può accedere alla sezione in cui può eseguire le sue rispettive funzioni (abbiamo immaginato un utente potesse essere un partecipante di un Hackathon A, un organizzatore di un Hackathon B, e un Giudice di un Hackathon C.

A livello implementativo sono state usate JLabel, JComboBox, Jbuttons e Jtables per poter permettere all'utente di interagire con il package Model via una classe chiamata "HackathonController", la quale presenta vari metodi per soddisfare i requisiti degli utenti.