**Documentazione Progetto Minopoli, Megna, Iodice**

# Sommario

• Introduzione  
• Struttura del progetto  
• Architettura MVC  
• Entity (Model)  
• Controller  
• Boundary (GUI)  
• Esecuzione  
• Flusso di utilizzo  
• Diagramma UML completo  
• Proposte di evoluzione

# 1. Introduzione

Il progetto è un'applicazione Java basata su Swing per la gestione di hackathon. Supporta tre ruoli di utenti:

* - **Partecipante**: può registrarsi, accettare inviti, creare team e caricare documenti.
* - **Organizzatore**: può creare hackathon, inviare inviti e visualizzare i propri eventi.
* - **Giudice**: può valutare i team partecipanti tramite voti.

L'applicazione segue l'architettura BCE (Boundary-Controller-Entity) e persiste lo stato su file.

# 2. Struttura del progetto

gruppo70-master/  
└── src/main/java/src/java/  
 ── model/ # Classi di dominio  
 ── controller/ # Logica applicativa e persistenza  
 └── gui/ # Interfacce grafiche (Swing)

# 3. Architettura BCE

* - **Model**: contiene le entità (Utente, Hackathon, Team, Invito, Documento, Voto) e la logica di base.
* - **Gui**: l'interfaccia grafica realizzata con Swing (classi GUI).
* - **Controller**: classe controller. Controller che coordina autenticazione, CRUD e persistenza dello stato su file.

# 4. Model

1. **Utente:** classe base con attributi nome, cognome, dataNascita, email, password.
2. **Hackathon:** rappresenta un evento con titolo, sede, date di inizio/fine, limiti di partecipanti e dimensione team.
3. **Team:** gruppo di partecipanti con nome e lista di Partecipante; registra un voto medio.
4. **Invito:** invito per un partecipante con stato (INVIATO, ACCETTATO, RIFIUTATO).
5. **Documento:** gestione di contenuti caricati da partecipanti.
6. **Voto:** associa un punteggio a un Team da parte di un Giudice.

# 5. Controller

* La classe controller.Controller gestisce:
* Registrazione e login utenti.
* CRUD su documenti.
* Creazione di hackathon e team.
* Invio e risposta a inviti.
* Valutazione dei team.
* Persistenza dello stato in data/state.dat.

# 6. GUI

* **Registrazione (Registrazione.java):** form di registrazione utente.
* **SignIn (SignIn.java):** form di login.
* **MainMenuGUI:** menu principale, instrada alle viste in base al ruolo.
* **CreaHackathonGUI, CreaTeamGUI, ValutaTeamGUI:** form specifici per azioni di organizzatore e giudice.
* **InvitiPartecipanteGUI, DocumentoGUI, ProfiloUtenteGUI, Dashboard:** viste per interazione partecipante e visualizzazione dati.

# 7. Esecuzione

Carica lo stato (o crea uno nuovo) dal file.

Avvia l'interfaccia Swing (MainMenuGUI).

Alla chiusura, salva lo stato aggiornato nel file.

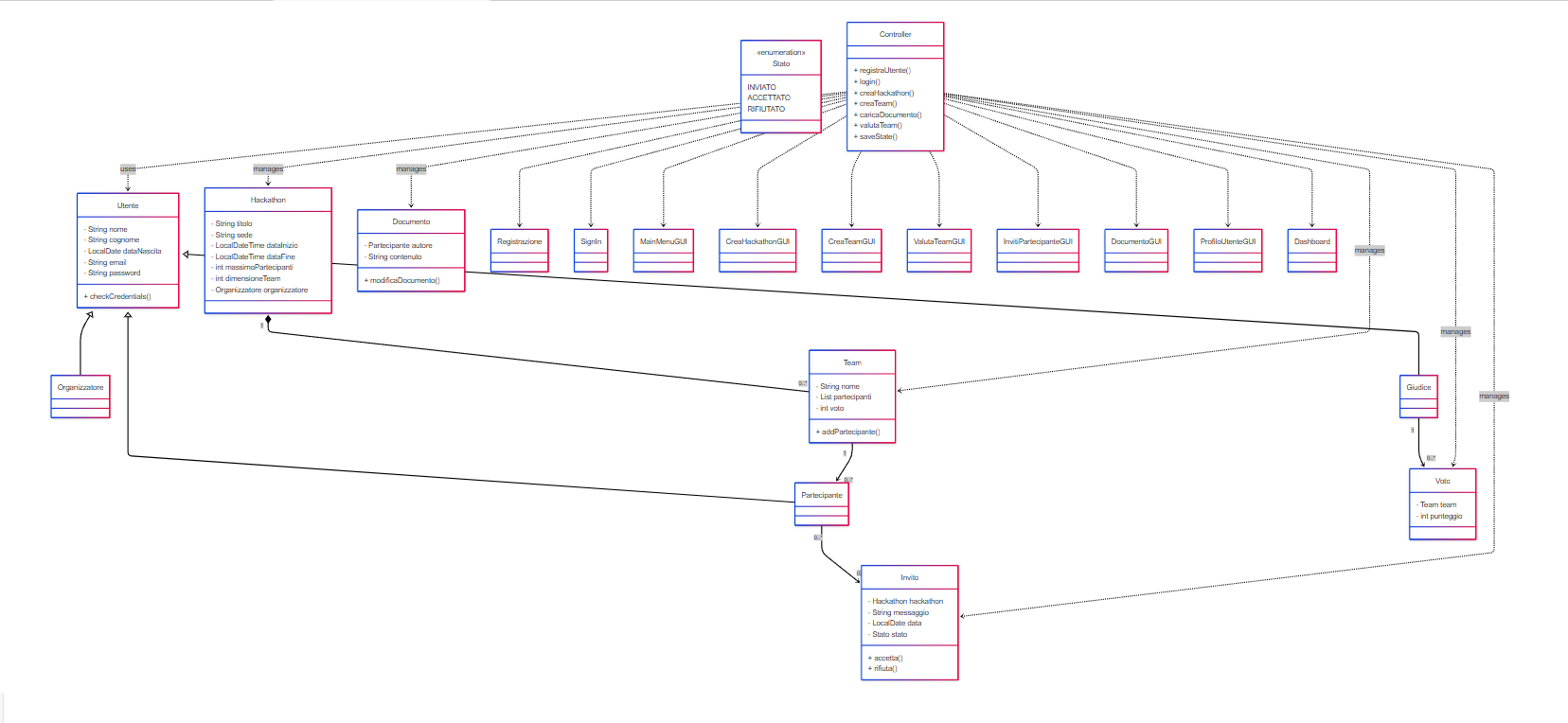
# 8. Flusso di utilizzo

* Avvio Applicazione: l'utente lancia l'applicazione eseguendo model.Main.
* Login: nella finestra SignIn, inserisce email e password.
* Menu Principale: dopo il login, si apre MainMenuGUI che mostra le azioni disponibili in base al ruolo.
* Azioni Partecipante: visualizza inviti, crea team, carica documenti, visualizza profilo.
* Azioni Organizzatore: crea hackathon, visualizza dashboard, invia inviti.
* Azioni Giudice: accede ai team e assegna voti.
* Persistenza: lo stato è memorizzato e salvato alla chiusura del programma.

# 9. Diagramma UML completo e link della repository di github

Link github repositury: <https://github.com/daniele107/gruppo70.git>

Non potendo allegare la foto del file visual paradigm, che troverete nella consegna è presente una bozza fatta su paint.



# 10. Proposte di Evoluzione

**Pubblicazione del Problema:**

* Aggiungere in Hackathon un attributo problemaPubblicato: boolean.
* In CreaHackathonGUI, un pulsante “Pubblica Problema” per organizzatori.
* Al click, impostare problemaPubblicato = true e rendere visibile il testo del problema.

**Commenti sui Progressi:**

* Definire la classe Commento con autore, testo, data.
* Estendere DocumentoGUI per inserire e visualizzare commenti.
* Persistere i commenti insieme ai documenti.

**Validazione della Dimensione del Team:**

* Verificare nel Controller che team.size <= dimensioneTeam.
* In CreaTeamGUI, disabilitare aggiunta membri al raggiungimento del limite.

**Calcolo e Visualizzazione della Classifica:**

* Aggiungere getRanking(h: Hackathon) in Controller.
* Nella Dashboard, creare tab “Statistiche” con classifica dei team.