UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

DOCUMENTAZIONE PER PROGETTO OBJECT-ORIENTED

CdL Triennale in Informatica
Corso Object-Oriented

Alessandro Minopoli - N86004964
Daniele Megna - N86004964
Simone Iodice - N86004964
Anno Accademico: 2024/2025

Indice

1 Introduzione

1.1 Traccia del progetto e analisi dei requisiti

Obiettivo Sviluppare un sistema informativo per la gestione degli *hackathon*, composto da:

- una base di dati relazionale;
- un applicativo Java con interfaccia grafica (*Swing*).

Il sistema supporta l'organizzazione e il monitoraggio di competizioni di programmazione in modo efficiente e intuitivo.

Autenticazione e ruoli Accesso tramite *login* (utente e password). Sono previsti tre ruoli:

1. **Partecipante** (utente generico): partecipa agli hackathon, forma o si unisce ai team.
2. **Giudice**: valuta i progetti, lascia commenti e valida i documenti.
3. **Organizzatore**: crea/aggiorna hackathon, gestisce iscrizioni e richieste.

1.1.1 Gestione degli hackathon

Ogni hackathon è descritto da:

`codice` identificativo univoco dell'evento;

`nome` denominazione ufficiale;

`sede` luogo di svolgimento;

`descrizioneProblema` traccia/tema da risolvere;

`dataInizio`, `dataFine` finestra temporale dell'evento;

`orarioPrevisto` orari operativi;

`modificheProgrammazione` eventuali aggiornamenti/scheduling;

`stato` uno tra **programmato**, **in_corso**, **completato**, **cancellato**.

Gli organizzatori possono inserire nuovi eventi e aggiornare quelli esistenti.

1.1.2 Team e partecipazione

`nomeTeam` univoco;

`sizeMax` dimensione massima;

`idHackathon` riferimento all'hackathon di partecipazione;

`capoTeam` utente designato come responsabile.

Gli utenti possono:

- visualizzare i team disponibili;
- richiedere l'adesione a un team;
- essere approvati/rifiutati dagli organizzatori.

1.1.3 Documenti di progetto

Ogni documento è associato a un team e contiene:

`nomeFile` nome del file;

`descrizione` breve sintesi del contenuto;

`tipo` *codice sorgente* / *presentazione* / *documentazione*;

`dimensione` peso del file;

`dataUpload` data di caricamento;

`statoValidazione` esito della verifica da parte dei giudici.

I giudici possono visionare e validare i documenti caricati.

1.1.4 Avanzamento lavori (progressi)

Registrazioni periodiche dei team:

`titolo` descrizione sintetica dell'update;

`descrizioneDettagliata` attività svolte;

`dataRegistrazione` istante di inserimento;

`commentiGiudici` eventuali note/osservazioni dei giudici.

1.1.5 Valutazioni

Al termine dell'hackathon i giudici valutano i progetti:

`idTeam` team valutato;

`voto` numero intero da 1 a 10;

`commento` motivazione testuale;

`dataValutazione` data della valutazione.

Sono previste statistiche aggregate e l'identificazione dei vincitori.

1.1.6 Area personale e homepage

Gli utenti dispongono di:

- **Area personale**: stato avanzamento dei propri team/progetti, commenti dei giudici, valutazioni ricevute.
- **Homepage**: tabella degli hackathon *attivi* e *programmati* per una panoramica immediata.

1.1.7 Ricerca avanzata

Funzionalità di ricerca per *hackathon*, *team*, *utenti* e *documenti* in base a criteri multipli, con evidenza di:

- hackathon **in corso**;
- team che **cercano membri**.

2 Package Model

2.1 Classi principali e attributi

Il package model progettato è stato realizzato con scelte strutturali e logiche finalizzate a soddisfare i requisiti funzionali e informativi del sistema di gestione hackathon. Il modello supporta le operazioni di gestione eventi, formazione team, caricamento documenti, monitoraggio progressi e valutazione progetti.

2.1.1 Classi: Utente, Partecipante, Giudice, Organizzatore

Motivazione Progettuale:
Questa scelta nasce dall'esigenza di modellare in modo chiaro i diversi protagonisti del sistema. La superclasse Utente consente di gestire le funzionalità comuni a tutti i profili, come l'autenticazione. La specializzazione in Partecipante, Giudice e Organizzatore riflette la necessità di distinguere i permessi e le operazioni disponibili per ciascun ruolo, semplificando la gestione delle autorizzazioni.

Attributi comuni (Utente):

- `login`
- `password`
- `nome`
- `cognome`
- `email`
- `ruolo`

2.1.2 Classi: Hackathon, StatoHackathon

Motivazione Progettuale:
La classe Hackathon è un'entità centrale nel sistema, rappresentando l'evento competitivo principale. L'enumerazione StatoHackathon è stata introdotta per poter tenere costantemente traccia del ciclo di vita dell'evento, dalle fasi iniziali di programmazione fino al completamento.

Attributi Hackathon:

- `id`: identificativo univoco
- `nome`: nome dell'evento
- `sede`: luogo di svolgimento
- `descrizioneProblema`: descrizione della challenge
- `dataInizio`: data e ora di inizio
- `dataFine`: data e ora di termine
- `orarioPrevisto`: durata pianificata
- `stato`: stato corrente dell'evento

Valori StatoHackathon:

- `PROGRAMMATO`: evento pianificato ma non ancora iniziato
- `IN_CORSO`: evento attualmente in svolgimento
- `COMPLETATO`: evento terminato con successo
- `CANCELLATO`: evento annullato

2.1.3 Classi: Team, StatoTeam

Motivazione Progettuale:
La classe Team modella i gruppi di lavoro che partecipano agli hackathon. Ogni team mantiene informazioni sulla composizione, capacità e associazione con l'evento. L'enumerazione StatoTeam consente di tracciare la situazione del team durante l'evento.

Attributi Team:

- `id`: identificativo univoco
- `nome`: nome del team
- `dimensioneMassima`: numero massimo di membri
- `hackathonId`: riferimento all'hackathon di partecipazione
- `capoTeamId`: identificativo del leader del team
- `stato`: stato attuale del team

Valori StatoTeam:

- `FORMANDO`: team in fase di costituzione
- `COMPLETO`: team al completo e pronto a partecipare
- `PARTECIPANTE`: team iscritto e attivo nell'hackathon

2.1.4 Classi: Documento, TipoDocumento

Motivazione Progettuale:
La classe Documento gestisce tutti i file prodotti dai team durante l'hackathon, inclusi codice sorgente, presentazioni e documentazione. L'enumerazione TipoDocumento classifica i diversi tipi di documenti prodotti.

Attributi Documento:

- `id`: identificativo univoco
- `nome`: nome del file
- `descrizione`: descrizione del contenuto
- `tipo`: categoria del documento
- `dimensione`: dimensione in byte
- `dataCaricamento`: data di upload
- `teamId`: riferimento al team autore
- `validato`: flag di validazione da parte dei giudici

Valori TipoDocumento:

- `CODICE_SORGENTE`: programmi e algoritmi sviluppati
- `PRESENTAZIONE`: slide e materiali di presentazione
- `DOCUMENTAZIONE`: manuali tecnici e guide
- `ALTRO`: altri tipi di documenti

2.1.5 Classi: Progress, Valutazione

Motivazione Progettuale:
La classe Progress consente ai team di documentare l'avanzamento del lavoro durante l'hackathon, mentre la classe Valutazione permette ai giudici di esprimere giudizi tecnici sui progetti presentati.

Attributi Progress:

- `id`: identificativo univoco
- `titolo`: titolo descrittivo dell'aggiornamento
- `descrizione`: descrizione dettagliata del progresso
- `dataCaricamento`: data di pubblicazione

- `teamId`: riferimento al team autore
- `hackathonId`: riferimento all'evento
- `commentoGiudice`: feedback opzionale da parte dei giudici

Attributi Valutazione:

- `id`: identificativo univoco
- `voto`: punteggio numerico (1-10)
- `commento`: commento testuale del giudice
- `dataValutazione`: data della valutazione
- `teamId`: riferimento al team valutato
- `hackathonId`: riferimento all'evento
- `giudiceId`: riferimento al giudice valutatore

2.2 Associazioni

2.2.1 Utente → Team

Associazione 1:N

- Un utente può far parte di 0 o 1 team alla volta
- Un team può contenere da 1 a N utenti

2.2.2 Team → Hackathon

Associazione N:1

- Un team partecipa a 1 solo hackathon
- Un hackathon può avere da 0 a N team partecipanti

2.2.3 Documento → Team

Associazione N:1

- Un documento è prodotto da 1 solo team
- Un team può produrre da 0 a N documenti

2.2.4 Progress → Team

Associazione N:1

- Un progresso è pubblicato da 1 solo team
- Un team può pubblicare da 0 a N progressi

2.2.5 Valutazione → Team

Associazione N:1

- Una valutazione è assegnata a 1 solo team
- Un team può ricevere da 0 a N valutazioni

2.2.6 Giudice → Valutazione

Associazione 1:N

- Un giudice può assegnare da 0 a N valutazioni
- Una valutazione è assegnata da 1 solo giudice

3 Package GUI

Il progetto si propone di fornire un sistema integrato per la gestione degli accessi e delle registrazioni degli utenti, degli hackathon, dei team, dei documenti e delle valutazioni. I protagonisti del progetto si suddividono in: **partecipanti**, **giudici** e **organizzatori**. L'interazione con il sistema avviene tramite una serie di interfacce grafiche (GUI), sviluppate nel package gui, che si interfacciano con la logica di controllo situata nel package controller. Il sistema è stato progettato seguendo **due pattern architetturali complementari**:

1. **Pattern BCE + DAO (Business-Control-Entity + Data Access Object)** per la gestione della logica di business e l'accesso ai dati 2. **Pattern MVC (Model-View-Controller)** per la separazione tra logica di presentazione e gestione dei dati

Questa architettura a doppio pattern garantisce una **chiara separazione delle responsabilità** tra: - **Business Logic** (Controller e servizi specializzati) - **Data Access** (DAO con implementazioni PostgreSQL) - **Entity Management** (Modelli di dominio) - **Presentation Layer** (Interfacce grafiche)

Il package gui contiene tutte le classi dedicate alla gestione dell'interfaccia grafica tramite **Swing**, implementando un design system moderno con componenti responsive e gestione avanzata degli eventi utente.

3.1 Classi Principali e attributi

3.1.1 Classe LoginPanel

La classe LoginPanel implementa la schermata di accesso all'applicazione. Permette all'utente di inserire username e password per effettuare il login, oppure di accedere alla schermata di registrazione per nuovi utenti. La classe si occupa di gestire l'interfaccia grafica (GUI) e di inoltrare le richieste al Controller secondo il pattern MVC.

Attributi principali:

- `JTextField loginField`: campo di testo per l'inserimento dello username
- `JPasswordField passwordField`: campo per l'inserimento della password
- `JButton loginButton`: pulsante per confermare il login

- `JButton registerButton`: pulsante per accedere alla registrazione
- `JPanel mainPanel`: pannello principale contenente l'interfaccia
- `Controller controller`: riferimento al controller

3.1.2 Classe DashboardPartecipante

Questa classe implementa l'interfaccia grafica dedicata al partecipante. Permette l'interazione con le funzionalità principali: visualizzazione hackathon, gestione team, caricamento documenti, monitoraggio progressi. L'interazione con la GUI viene inoltrata al controller tramite eventi.

Attributi principali:

- `JPanel homePage`: pannello principale della dashboard
- `JPanel welcomePanel`: pannello messaggio di benvenuto
- `JButton logoutButton`: pulsante per il logout
- `JComboBox<String> actionCombo`: menù azioni disponibili
- `JLabel welcomeLabel`: etichetta personalizzata con nome utente
- `String username`: username per personalizzazione
- `Controller controller`: riferimento al controller

3.1.3 Classe DashboardGiudice

Classe GUI che rappresenta la dashboard del giudice. Permette l'interazione con le funzionalità di valutazione progetti, visualizzazione documenti, inserimento commenti sui progressi e accesso alle statistiche. Le azioni vengono inoltrate al controller.

Attributi principali:

- `JPanel dashboardPanel`: pannello principale dashboard giudice
- `JLabel giudiceLabel`: etichetta identificativa del giudice
- `JPanel navigationPanel`: pannello navigazione principale
- `JButton logoutButton`: pulsante logout
- `JComboBox<String> actionCombo`: menù azioni giudice
- `JLabel statusLabel`: etichetta per messaggi di stato
- `String username`: username del giudice
- `Controller controller`: riferimento al controller

3.1.4 Classe DashboardOrganizzatore

Classe GUI per la dashboard dell'organizzatore. Fornisce accesso completo a tutte le funzionalità amministrative: gestione hackathon, approvazione team, gestione utenti, monitoraggio generale del sistema.

Attributi principali:

- `JPanel adminPanel`: pannello principale amministrativo
- `JLabel adminLabel`: etichetta identificativa organizzatore
- `JPanel controlPanel`: pannello controlli amministrativi
- `JButton logoutButton`: pulsante logout
- `JComboBox<String> adminActionCombo`: menù azioni amministrative
- `JLabel systemStatusLabel`: etichetta stato sistema
- `String username`: username organizzatore
- `Controller controller`: riferimento al controller

3.1.5 Classe SearchAndFilterPanel

Pannello avanzato per ricerca globale e filtri multipli. Permette di cercare attraverso hackathon, team, utenti, documenti e progressi con filtri avanzati e ricerca in tempo reale.

Attributi principali:

- `JTextField globalSearchField`: campo ricerca globale
- `JButton searchButton`: pulsante avvio ricerca
- `JButton clearButton`: pulsante pulizia filtri
- `JComboBox<String> categoryFilter`: filtro per categoria
- `JComboBox<String> typeFilter`: filtro per tipo
- `JComboBox<String> statusFilter`: filtro per stato
- `JTable resultsTable`: tabella risultati ricerca
- `DefaultTableModel resultsTableModel`: modello dati tabella
- `JButton viewDetailsButton`: pulsante visualizzazione dettagli
- `JButton exportResultsButton`: pulsante esportazione risultati
- `JPanel advancedFiltersPanel`: pannello filtri avanzati
- `List<SearchResult> allResults`: lista completa risultati
- `List<SearchResult> filteredResults`: lista risultati filtrati

4 Package DAO e Implementazione

Il package DAO (Data Access Object) si occupa della comunicazione tra l'applicazione Java e il database relazionale (PostgreSQL). L'obiettivo di questo package è garantire un'interfaccia stabile e coerente per accedere ai dati.

4.1 Interfacce DAO

Le interfacce DAO definiscono in modo astratto i metodi da implementare per il corretto funzionamento dell'applicazione e della gestione dei dati.

4.1.1 Interfaccia UtenteDAO

L'interfaccia UtenteDAO contiene i metodi comuni a tutti i tipi di utente del sistema indipendentemente dal ruolo specifico:

- `findAll()`: recupera tutti gli utenti del sistema
- `findById(int id)`: recupera utente per ID
- `findByLogin(String login)`: recupera utente per login
- `insert(Utente utente)`: inserisce nuovo utente
- `update(Utente utente)`: aggiorna utente esistente
- `delete(int id)`: elimina utente
- `autentica(String login, String password)`: verifica credenziali

4.1.2 Interfaccia HackathonDAO

L'interfaccia HackathonDAO contiene i metodi per la gestione degli hackathon:

- `findAll()`: recupera tutti gli hackathon
- `findById(int id)`: recupera hackathon per ID
- `insert(Hackathon hackathon)`: crea nuovo hackathon
- `update(Hackathon hackathon)`: aggiorna hackathon esistente
- `delete(int id)`: elimina hackathon
- `findByStato(String stato)`: recupera hackathon per stato

4.1.3 Interfaccia TeamDAO

Questa interfaccia contiene i metodi per la gestione dei team:

- `findAll()`: recupera tutti i team
- `findById(int id)`: recupera team per ID
- `findByHackathon(int hackathonId)`: recupera team per hackathon
- `insert(Team team)`: crea nuovo team
- `update(Team team)`: aggiorna team esistente
- `delete(int id)`: elimina team

4.1.4 Interfaccia DocumentoDAO

Questa interfaccia contiene i metodi per la gestione dei documenti:

- `findAll()`: recupera tutti i documenti
- `findById(int id)`: recupera documento per ID
- `findByTeam(int teamId)`: recupera documenti per team
- `insert(Documento documento)`: carica nuovo documento
- `update(Documento documento)`: aggiorna documento
- `delete(int id)`: elimina documento

4.1.5 Interfaccia ProgressDAO

Questa interfaccia contiene i metodi per la gestione dei progressi:

- `findAll()`: recupera tutti i progressi
- `findById(int id)`: recupera progresso per ID
- `findByTeam(int teamId)`: recupera progressi per team
- `insert(Progress progress)`: pubblica nuovo progresso
- `update(Progress progress)`: aggiorna progresso
- `delete(int id)`: elimina progresso

4.1.6 Interfaccia ValutazioneDAO

Questa interfaccia contiene i metodi per la gestione delle valutazioni:

- `findAll()`: recupera tutte le valutazioni
- `findById(int id)`: recupera valutazione per ID
- `findByTeam(int teamId)`: recupera valutazioni per team
- `insert(Valutazione valutazione)`: assegna nuova valutazione
- `update(Valutazione valutazione)`: aggiorna valutazione
- `delete(int id)`: elimina valutazione

4.2 ImplementazionePostgresDAO

Il package dao.postgres contiene le implementazioni concrete delle interfacce DAO, specifiche per il database PostgreSQL. Ogni classe implementa i metodi definiti nelle relative interfacce.

4.2.1 Classe UtentePostgresDAO

Questa classe implementa l'interfaccia UtenteDAO e consente di interagire con il sistema di gestione utenti tramite PostgreSQL.

Metodi Principali:

- `UtentePostgresDAO(ConnectionManager cm)`: costruttore che inizializza la connessione
- `findAll()`: recupera l'elenco completo degli utenti
- `findById(int id)`: recupera utente per ID
- `findByLogin(String login)`: recupera utente per login
- `insert(Utente utente)`: inserisce nuovo utente
- `update(Utente utente)`: aggiorna utente esistente
- `delete(int id)`: elimina utente
- `autentica(String login, String password)`: verifica credenziali

4.2.2 Classe HackathonPostgresDAO

Questa classe implementa l'interfaccia HackathonDAO fornendo tutte le funzionalità di gestione degli hackathon.

Metodi Principali:

- `HackathonPostgresDAO(ConnectionManager cm)`: costruttore con inizializzazione connessione
- `findAll()`: recupera tutti gli hackathon

- `findById(int id)`: recupera hackathon per ID
- `insert(Hackathon hackathon)`: crea nuovo hackathon
- `update(Hackathon hackathon)`: aggiorna hackathon esistente
- `delete(int id)`: elimina hackathon
- `findByStato(String stato)`: recupera hackathon per stato

4.2.3 Classe TeamPostgresDAO

Questa classe implementa l'interfaccia TeamDAO fornendo tutte le funzionalità di gestione dei team.

Metodi Principali:

- `TeamPostgresDAO(ConnectionManager cm)`: costruttore con inizializzazione DB
- `findAll()`: recupera tutti i team
- `findById(int id)`: recupera team per ID
- `findByHackathon(int hackathonId)`: recupera team per hackathon
- `insert(Team team)`: crea nuovo team
- `update(Team team)`: aggiorna team esistente
- `delete(int id)`: elimina team

4.2.4 Classe DocumentoPostgresDAO

Questa classe implementa l'interfaccia DocumentoDAO fornendo la logica concreta per la gestione dei documenti nel sistema hackathon.

Metodi Principali:

- `DocumentoPostgresDAO(ConnectionManager cm)`: costruttore con connessione database
- `findAll()`: recupera tutti i documenti
- `findById(int id)`: recupera documento per ID
- `findByTeam(int teamId)`: recupera documenti per team
- `insert(Documento documento)`: carica nuovo documento
- `update(Documento documento)`: aggiorna documento
- `delete(int id)`: elimina documento

4.3 Connessione al Database

Per la connessione al database si fa uso della classe `ConnectionManager`. Questa classe implementa il pattern Dependency Injection per la gestione centralizzata della connessione al database PostgreSQL. Fornisce un punto di accesso univoco per tutte le operazioni di database, garantendo efficienza delle risorse e controllo degli accessi.

Metodi Principali:

- `ConnectionManager(DataSource dataSource)`: costruttore che inizializza la connessione PostgreSQL
- `getConnection()`: metodo per ottenere la connessione al database con controllo di validità
- `commit(Connection conn)`: metodo per eseguire commit delle transazioni
- `rollback(Connection conn)`: metodo per eseguire rollback delle transazioni

5 Package Controller

Il Package controller tramite la classe Controller gestisce la logica applicativa dell'interfaccia grafica. Funziona come coordinatore tra la GUI e le DAO che interagiscono con il database PostgreSQL. Supporta sia le operazioni lato utente che lato amministratore.

Attributi principali:

- `Utente currentUser`: riferimento all'utente corrente
- `HackathonDAO hackathonDAO`: DAO per gestione hackathon
- `UtenteDAO utenteDAO`: DAO per gestione utenti
- `TeamDAO teamDAO`: DAO per gestione team
- `DocumentoDAO documentoDAO`: DAO per gestione documenti
- `ProgressDAO progressDAO`: DAO per gestione progressi
- `ValutazioneDAO valutazioneDAO`: DAO per gestione valutazioni

5.1 Metodi principali per gestione Partecipanti

5.1.1 Gestione Hackathon

- `getTuttiHackathon()`: recupera e mostra tutti gli hackathon attivi
- `getHackathonById(int id)`: recupera dettagli specifico hackathon
- `cercaHackathon(String criterio)`: ricerca hackathon per nome o descrizione

5.1.2 Gestione Team

- `getTeamDisponibili(int hackathonId)`: mostra team che accettano membri
- `getTeamById(int id)`: recupera informazioni complete team
- `getMembriTeam(int teamId)`: recupera lista membri del team
- `richiediPartecipazioneTeam(int teamId)`: invia richiesta di partecipazione

5.1.3 Gestione Documenti

- `caricaDocumento(String nome, String descrizione, String tipo, byte[] contenuto, int teamId)`: carica nuovo documento
- `getDocumentiTeam(int teamId)`: recupera tutti i documenti di un team
- `scaricaDocumento(int documentoId)`: avvia download documento

5.1.4 Gestione Progressi

- `pubblicaProgresso(String titolo, String descrizione, int teamId)`: pubblica nuovo aggiornamento
- `getProgressiTeam(int teamId)`: recupera cronologia progressi team
- `aggiungiCommentoProgresso(int progressoId, String commento)`: aggiunge commento giudice

5.2 Metodi principali per gestione Giudici

5.2.1 Valutazione Team

- `valutaTeam(int teamId, int voto, String commento)`: assegna valutazione a team
- `getValutazioniAssegnate(int giudiceId)`: recupera valutazioni assegnate dal giudice
- `getStatisticheGenerali()`: mostra statistiche aggregate del giudice

5.2.2 Validazione Documenti

- `validaDocumento(int documentoId)`: approva documento caricato da team
- `rifiutaDocumento(int documentoId, String motivo)`: rifiuta documento con motivazione
- `getDocumentiDaValidare(int hackathonId)`: recupera documenti in attesa di validazione

5.3 Metodi principali per gestione Organizzatori

5.3.1 Gestione Eventi

- `creaEventoDaRequest(EventRequest request)`: crea nuovo evento hackathon
- `aggiornaEvento(int hackathonId, EventRequest request)`: aggiorna evento esistente
- `eliminaEventiConclusi()`: elimina eventi conclusi
- `concludeEvento(int hackathonId)`: termina evento e abilita fase valutazioni

5.3.2 Gestione Utenti e Team

- `approvaRichiestaPartecipazione(int richiestaId)`: approva richiesta partecipazione team
- `rifiutaRichiestaPartecipazione(int richiestaId, String motivo)`: rifiuta richiesta con motivazione
- `aggiornaRuolo(int utenteId, String nuovoRuolo)`: modifica ruolo utente
- `eliminaUtente(int utenteId)`: elimina account utente

5.3.3 Monitoraggio Sistema

- `getStatisticheGenerali()`: recupera statistiche complessive piattaforma
- `generaReportAttivita()`: genera report attività giornaliera/mensile
- `monitoraStatoSistema()`: verifica stato componenti sistema

6 Conclusioni

La realizzazione del progetto è stata un'importante occasione per mettere in pratica le competenze acquisite durante il corso di OOP. L'implementazione di un sistema complesso come il gestore di hackathon ha richiesto l'integrazione di più tecnologie e pattern architetturali consolidati, evidenziando la nostra capacità di progettare soluzioni scalabili e manutenibili.

Competenze specifiche messe in pratica:

- **Architettura e Design Patterns**: Implementazione rigorosa dei pattern BCE+DAO e MVC, dimostrando padronanza dei principi SOLID e della separazione delle responsabilità
- **Gestione Database** : Progettazione e implementazione di un sistema di persistenza robusto con PostgreSQL, includendo gestione transazionale e ottimizzazione delle query
- **Interfaccia Utente Avanzata**: Sviluppo di un design system moderno con Swing, implementando componenti responsive e gestione avanzata degli eventi

- **Logica di Business Complessa**: Implementazione di regole di business sofisticate per la gestione di ruoli, autorizzazioni e workflow multi-step

Si ringraziano i docenti per la guida fornita durante lo sviluppo del progetto.