ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

Scuola di Scienze

Campus di Bologna

CORSO DI STUDIO IN

INFORMATICA PER IL MANAGEMENT

Documentazione del Progetto – BOSTARTER

Corso di Base di Dati

Docente: Prof. Marco Di Felice

Tutor: Dr. Leonardo Ciabattini

A cura di:

Barone Dalia Valeria: 0001097620

Neamtu Sofia: 0001091725

Veroni Mattia: 0001078771

Anno Accademico 2024-2025

SOMMARIO

1. RACCOLTA/ANALISI DEI REQUISITI

TESTO COMPLETO DELLE SPECIFICA SUI DATI

La piattaforma BOSTARTER gestisce i dati degli utenti registrati. Ogni utente dispone di indirizzo e-mail (univoco), nickname, password, nome, cognome, anno di nascita, luogo di nascita. Inoltre, ogni utente può indicare le proprie skill di curriculum. Le skill di curriculum consistono in una sequenza di: , dove la competenza è una stringa ed il livello è un numero tra 0 e 5 (es.). La lista delle competenze è comune a tutti gli utenti della piattaforma. Alcuni utenti -ma non tutti possono appartenere a due sotto-categorie: utenti amministratori, o utenti creatori. Gli utenti amministratori dispongono anche di un codice di sicurezza. Solo gli utenti amministratori possono popolare la lista delle competenze. Un utente creatore dispone anche dei campi: #nr_progetti (ridondanza concettuale, vedi sotto) ed affidabilità. Un utente creatore – e solo lui- può inserire uno o più progetti. Ogni progetto dispone di un nome (univoco), un campo descrizione, una data di inserimento, una o più foto, un budget da raggiungere per avviare il progetto, una data limite entro cui raggiungere il budget, uno stato. Lo stato è un campo di tipo enum (aperto/chiuso). Ogni progetto è associato ad un solo utente creatore. Inoltre, ogni progetto prevede una lista di reward: una reward dispone di un codice univoco, una breve descrizione, una foto. I progetti appartengono esclusivamente a due categorie: progetti hardware o progetti software. Nel caso dei progetti hardware, è presente anche la lista delle componenti necessarie: ogni componente ha un nome univoco, una descrizione, un prezzo, una quantità (>0). Nel caso dei progetti software, viene elencata la lista dei profili necessari per lo sviluppo. Ogni profilo dispone di un nome (es. “Esperto AI”) e di skill richieste: come nel caso delle skill di curriculum, esse consistono in una sequenza , dove la competenza è una stringa -tra quelle presenti in piattaforma- ed il livello è un numero tra 0 e 5. Ogni utente della piattaforma può finanziare un progetto: ogni finanziamento dispone di un importo ed una data. Un utente potrebbe inserire più finanziamenti per lo stesso progetto, ma in date diverse. Nel momento in cui la somma totale degli importi dei finanziamenti supera il budget del progetto, oppure il progetto resta in stato aperto oltre la data limite, lo stato di tale progetto diventa pari a chiuso: un progetto chiuso non accetta ulteriori finanziamenti. Ad ogni finanziamento è associata una sola reward, tra quelle previste per il progetto finanziato. Un utente può inserire commenti relativi ad un progetto. Ogni commento dispone di un id (univoco), una data ed un campo testo. L’ utente creatore può eventualmente inserire una risposta per ogni singolo commento (un commento ha al massimo 1 risposta). Infine, è prevista la possibilità per gli utenti di candidarsi come partecipanti allo sviluppo di un progetto software. Un utente può candidarsi ad un numero qualsiasi di profili. Un progetto software può ricevere un numero qualsiasi di candidature per un certo profilo. La piattaforma consente ad un utente di inserire una candidatura su un profilo SOLO se, per ogni skill richiesta da un profilo, l’utente dispone di un livello superiore o uguale al valore richiesto. L’utente creatore può accettare o meno la candidatura.

LISTA DELLE OPERAZIONI

Operazioni che riguardano tutti gli utenti:

- Autenticazione/registrazione sulla piattaforma
- Inserimento delle proprie skill di curriculum
- Visualizzazione dei progetti disponibili
- Finanziamento di un progetto (aperto). Un utente può finanziare anche il progetto di cui è creatore.
- Scelta della reward a valle del finanziamento di un progetto
- Inserimento di un commento relativo ad un progetto
- Inserimento di una candidatura per un profilo richiesto per la realizzazione di un progettosoftware

Operazioni che riguardano SOLO gli amministratori:

- Inserimento di una nuova stringa nella lista delle competenze
- In fase di autenticazione, oltre a username e password, viene richiesto anche il codice di sicurezza

Operazioni che riguardano SOLO gli utenti creatori:

- Inserimento di un nuovo progetto
- Inserimento delle reward per un progetto
- Inserimento di una risposta ad un commento
- Inserimento di un profilo -solo per la realizzazione di un progetto software
- Accettazione o meno di una candidatura

Statistiche (visibili da tutti gli utenti):

- Visualizzare la classifica degli utenti creatori, in base al loro valore di affidabilità. Mostrare solo il nickname dei primi 3 utenti.
- Visualizzare i progetti APERTI che sono più vicini al proprio completamento (= minore differenza tra budget richiesto e somma totale dei finanziamenti ricevuti). Mostrare solo i primi 3 progetti.
- Visualizzare la classifica degli utenti, ordinati in base al TOTALE di finanziamenti erogati. Mostrare solo i nickname dei primi 3 utenti.

TAVOLA MEDIA DEI VOLUMI

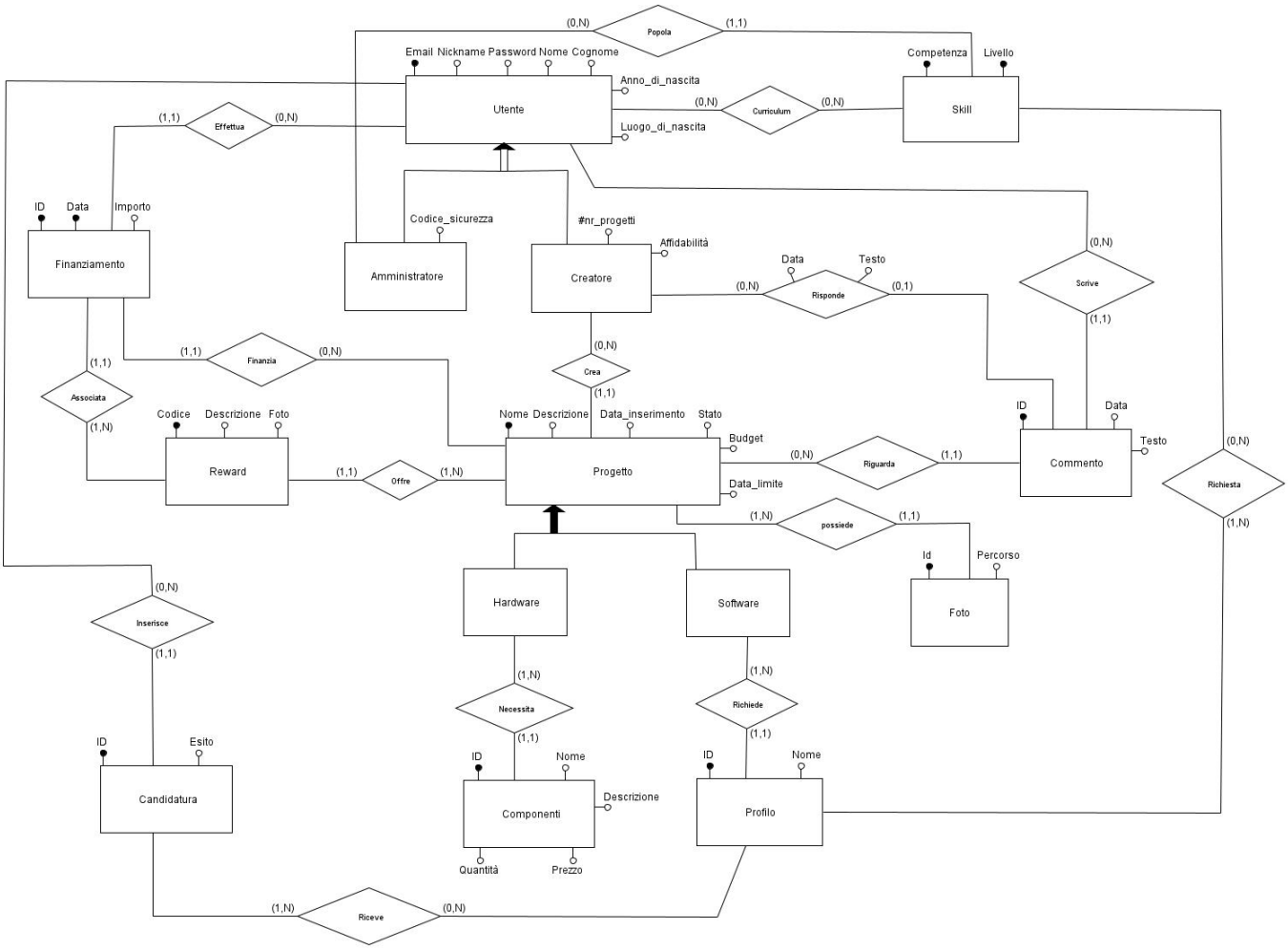
10 progetti, 3 finanziamenti per progetto, 5 utenti, 2 progetti per utente.

GLOSSARIO DEI DATI

Termine	**Descrizione**	**Collegamenti**
Utente	Un utente registrato sulla piattaforma BOSTARTER.	Skill, Progetto, Commento, Finanziamento, Reward, Candidatura
Amministratore	Utente con permessi di amministrazione.	Skill
Creatore	Utente che può creare progetti sulla piattaforma.	Progetto, Commento
Skill	Skill possedute dagli utenti oppure skill richieste per i profili dei progetti software. Formate da competenza e livello.	Utente, Amministratore, Profilo
Progetto	Progetto creato da un utente creatore.	Creatore, Commento, Foto, Finanziamento, Reward
Hardware	Progetto che prevede l'uso di componenti hardware.	Componente
Software	Progetto che richiede competenze software.	Profilo
Reward	Una ricompensa per chi finanzia un progetto.	Finanziamento, Progetto
Componente	Componenti necessarie per lo sviluppo del progetto hardware.	Hardware
Profilo	Profili richiesti per lo sviluppo di un progetto software.	Software, Candidatura, Skill
Finanziamento	Un utente può finanziare un progetto.	Utente, Progetto, Reward
Commento	Un utente può commentare un progetto e il creatore può rispondere.	Utente, Creatore, Progetto
Candidatura	Un utente può candidarsi per contribuire a un progetto software.	Profilo, Utente
Foto	Un progetto può aver una o più foto	Progetto

2. PROGETTAZIONE CONCETTUALE

DIAGRAMMA E-R



DIZIONARIO DEI DATI

DIZIONARIO DELLE ENTITÀ

Entità	**Descrizione**	**Attributi**	**Identificatore**
Utente	Un utente registrato sulla piattaforma BOSTARTER.	Email, Nickname, Password, Nome, Cognome, Anno_di_nascita, Luogo_di_nascita	Email
Amministratore	Utente con permessi di amministrazione.	Codice_sicurezza	

Creatore	Utente che può creare progetti sulla piattaforma.	#nr_progetti, Affidabilità	
Skill	Skill possedute dagli utenti oppure skill richieste per i profili dei progetti software. Formate da competenza e livello.	Competenza, Livello	Competenza, Livello
Progetto	Progetto creato da un utente creatore.	Nome, Descrizione, Data_inserimento, Stato, Budget, Data_limite	Nome
Hardware	Progetto che prevede l'uso di componenti hardware.		
Software	Progetto che richiede competenze software.		
Reward	Una ricompensa per chi finanzia un progetto.	Codice, Descrizione, Foto	Codice
Componenti	Componenti necessarie per lo sviluppo del progetto hardware.	ID, Nome, Descrizione, Prezzo, Quantità	ID
Profilo	Profili richiesti per lo sviluppo di un progetto software.	ID, Nome	ID
Finanziamento	Un utente può finanziare un progetto.	ID, Data, Importo	ID, Data
Commento	Un utente può commentare un progetto e il creatore può rispondere.	ID, Data, Testo	ID
Candidatura	Un utente può candidarsi per contribuire a un progetto software.	ID, Esito	ID
Foto	Un progetto può aver una o più foto	ID, Percorso	ID

DIZIONARIO DELLE RELAZIONI:

Relazione	**Descrizione**	**Componenti**	**Attributi**
Curriculum	Un utente possiede delle skill di curriculum.	Utente - Skill	
Crea	Un utente creatore crea un progetto.	Creatore - Progetto	

Offre	Un progetto offre delle ricompense ai finanziatori.	Progetto - Reward	
Necessita	Un progetto hardware necessita di componenti.	Hardware - Componenti	
Richiede	Un progetto software richiede profili con skill specifiche.	Software - Profilo	
Richiesta	Skill richieste per un profilo.	Profilo - Skill	
Effettua	Un utente effettua un finanziamento.	Utente - Finanziamento	
Finanzia	Un finanziamento è associato a un progetto.	Finanziamento - Progetto	
Associata	Un finanziamento è associato a UNA sola Reward, ma una Reward può essere scelta da più finanziamenti.	Finanziamento - Reward	
Scrive	Un utente scrive un commento.	Utente - Commento	
Riguarda	Un commento riguarda un progetto.	Commento - Progetto	
Risponde	Il creatore può rispondere a un commento.	Commento - Creatore	Data, Testo
Inserisce	Un utente inserisce una candidatura per un progetto software.	Utente - Candidatura	
Riceve	La candidatura dell'utente è associata a un profilo richiesto dal progetto.	Candidatura - Profilo	
Popola	L'amministratore può aggiungere nuove competenze alla piattaforma.	Amministratore - Skill	
Possiede	Un progetto può avere più foto	Progetto- Foto	

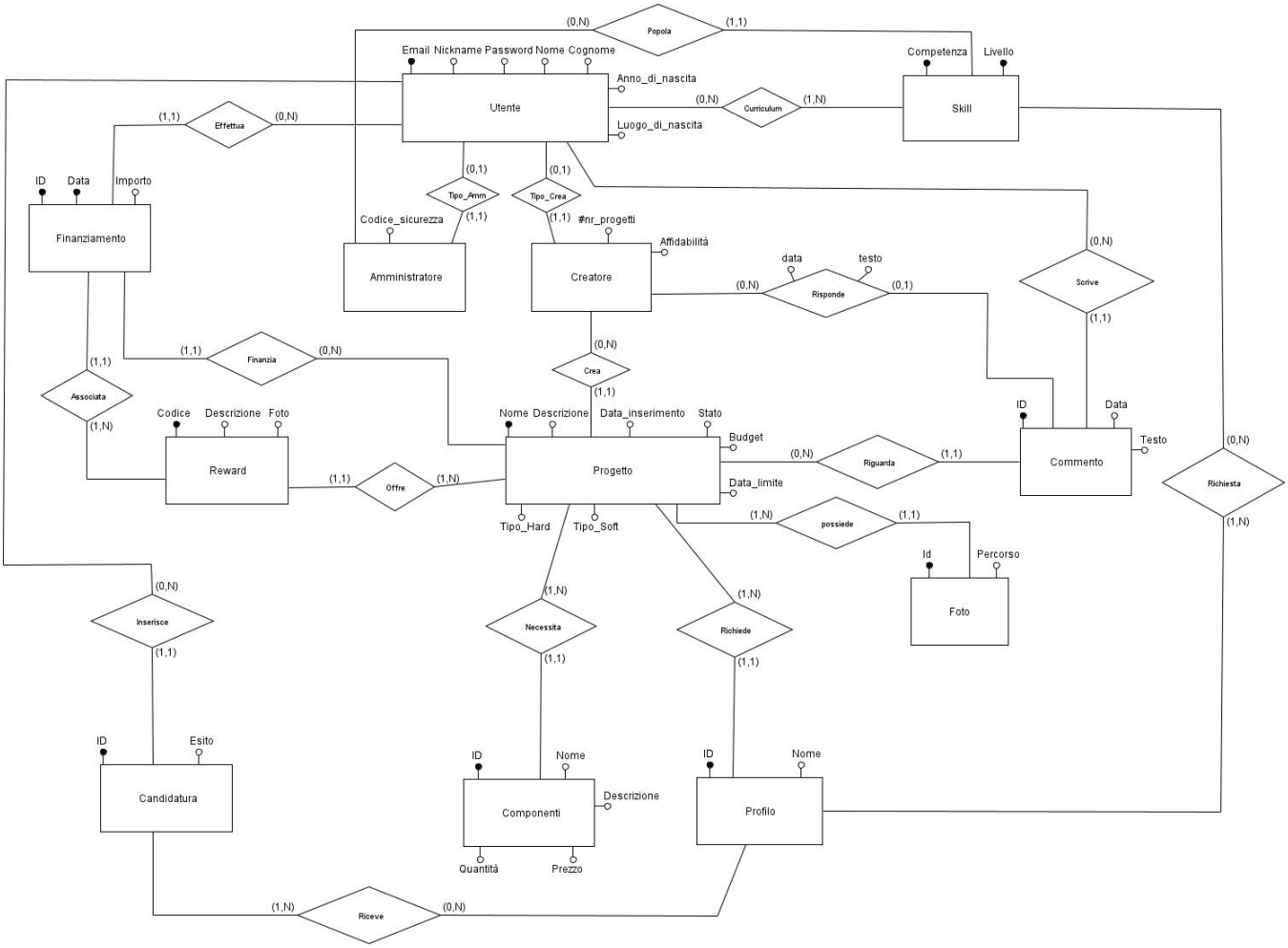
TAVOLA DELLE BUSINESS RULES

Regole di vincolo
1. Le skill di curriculum sono formate da coppie competenza-livello, dove competenza è una stringa e livello è un numero compreso tra 0 e 5
2. L'attributo Stato dell'entità Progetto è un enum che può assumere i valori "aperto" o "chiuso"
3. L'attributo quantità dell'entità Componenti deve essere strettamente maggiore di 0.
4. Un progetto chiuso non accetta più finanziamenti
5. Ad ogni finanziamento è associata una sola reward, tra quelle disponibili per il progetto finanziato.
6. Un utente può candidarsi per un progetto software solo se possiede tutte le skill richieste dal profilo con livello adeguato

7. L ’attributo Esito dell’entità Candidatura è un booleano settato di default a false, che viene cambiato a true quando la candidatura viene accettata.
Regole di derivazione
1. Quando la somma totale dei finanziamenti supera il budget del progetto, il sistema cambia automaticamente lo stato del progetto in "chiuso".
2. Quando un progetto rimane aperto oltre la data limite, il sistema lo chiude automaticamente.

3. PROGETTAZIONE LOGICA

RISTRUTTURAZIONE DELLO SCHEMA CONCETTUALE



LISTA DELLE TABELLE CON VINCOLI DI CHIAVI

- Utente(Email, Nickname, Password, Nome, Cognome, Anno_di_nascita, Luogo_di_nascita)
- Skill(Competenza, Livello, Email_ Amministratore)
- Skill_Curriculum(Email_Utente, Competenza, Livello)
- Amministratore(Email, Codice_sicurezza)
- Creatore(Email, #nr_progetti, Affidabilità)
- Progetto(Nome, Descrizione, Data_inserimento, Stato, Budget, Data_Limite, Tipo, Email_Creatore)
- Componente(ID, Nome, Descrizione, Prezzo, Quantità, Nome_Progetto)
- Profilo(ID, Nome, Nome_Progetto)
- Skill_Richiesta(ID_Profilo, Competenza, Livello)
- Commento(ID, Data, Testo, Nome_Progetto, Email_Utente)
- Risposta(ID_Commento, Email_Creatore, Testo, Data)
- Reward(Codice, Descrizione, Foto, Nome_Progetto)

- Finanziamento(ID, Data, Importo, Email_Utente, Codice_Reward, Nome_Progetto)
- Candidatura(ID, Esito, Email_Utente, ID_Profilo)
- Foto(ID, Percorso, Nome_Progetto)

LISTA DEI VINCOLO INTER-RELAZIONALI

- Skill.Email_ Amministratore → Amministratore.Email
- Skill_Curriculum.Email_Utente → Utente.Email
- Skill_Curriculum.Competenza → Skill.Competenza
- Skill_Curriculum.Livello → Skill. Livello
- Amministratore.Email → Utente.Email
- Creatore.Email → Utente.Email
- Progetto.Email_Creatore → Creatore.Email
- Componente.Nome_Progetto → Progetto.Nome
- Profilo.Nome_Progetto → Progetto.Nome
- Skill_Richiesta.ID_Profilo → Profilo.ID
- Skill_Richiesta.Competenza → Skill.Competenza
- Skill_Richiesta.Livello → Skill.Livello
- Commento.Nome_Progetto → Progetto.Nome
- Commento.Email_Utente → Utente.Email
- Risposta.ID_Commento → Commento.ID
- Risposta.Email_Creatore → Creatore.Email
- Reward.Nome_Progetto → Progetto.Nome
- Finanziamento.Email_Utente → Utente.Email
- Finanziamento.Codice_Reward → Reward.Codice
- Finanziamento.Nome_Progetto → Progetto.Nome
- Candidatura.Email_Utente → Utente.Email
- Candidatura.ID_Profilo → Profilo.ID
- Foto.Nome_Progetto → Progetto.Nome

ANALISI DELLE RIDONDANZE

Valutare se la seguente ridondanza: campo #nr_progetti relativo ad un utente creatore debba essere tenuta o eliminata, sulla base delle seguenti operazioni:

1) Aggiungere un nuovo progetto ad un utente creatore esistente (1 volte/mese, interattiva)
2) Visualizzare tutti i progetti e tutti i finanziamenti (1 volta/mese, batch)
3) Contare il numero di progetti associati ad uno specifico utente (3 volte/mese, batch)

Coefficienti per l'analisi: $w_I = 1$, $w_B = 0.5$, $\alpha = 2$

Tabella dei volumi: 10 progetti, 3 finanziamenti per progetto, 5 utenti, 2 progetti per utente

Operazioni CON ridondanza:

Op1) 1×1×(2×(1+1)+0) = 4

Motivazione: Sono necessarie due operazioni di scrittura: inserimento del nuovo progetto nella tabella Progetto e aggiornamento del contatore #nr_progetti nella tabella Creatore.

Op2) 1×0.5×((2×0)+7×10) = 35

Motivazione: La visualizzazione richiede 7 accessi per ogni progetto: lettura del progetto, navigazione della relazione con i finanziamenti (3 accessi), lettura dei finanziamenti associati (3 accessi). Totale: 7×10 progetti = 70 accessi.

Op3) 3×0.5×((2×0)+1) = 1.5

Motivazione: Il conteggio si riduce a una semplice lettura del campo #nr_progetti dalla tabella Creatore.

Costo totale CON ridondanza: 4 + 35 + 1.5 = 40.5

Operazioni SENZA ridondanza:

Op1) 1×1×((2×1)+0) = 2

Motivazione: È sufficiente un'unica operazione di scrittura per l'inserimento del progetto.

Op2) 1×0.5×((2×0)+7×10) = 35

Motivazione: La visualizzazione non è influenzata dalla presenza/assenza della ridondanza, quindi richiede gli stessi 7 accessi per progetto (70 accessi totali).

Op3) 3×0.5×(2×0+(2×2)) = 6

Motivazione: Il conteggio richiede una query COUNT sui progetti associati al creatore. Con 2 progetti per utente e applicando il fattore α, otteniamo 2×2 = 4 accessi per operazione.

Costo totale SENZA ridondanza: 2 + 35 + 6 = 43

Metriche di Performance

Speedup = $c(S)/ c(S_{rid})$

Speedup = 43 / 40.5 ≈ 1.062

Overhead = $|m(s) – m(S_{rid})|$

Overhead = 40.5 - 43 = -2.5

Valutazione Finale

La ridondanza risulta vantaggiosa perché il risparmio nelle operazioni di conteggio (frequenti) compensa ampiamente il costo aggiuntivo negli aggiornamenti (meno frequenti). Il beneficio netto di 2.5 punti giustifica il mantenimento del campo #nr_progetti.

4. NORMALIZZAZIONE

In questa sezione analizziamo le tabelle del modello logico per verificarne la conformità alle prime quattro forme normali (1NF, 2NF, 3NF e BCNF). Per ciascuna tabella valutiamo la struttura delle chiavi e le dipendenze funzionali, al fine di individuare eventuali anomalie, dipendenze indesiderate o ridondanze. Nel caso in cui emergano violazioni delle forme normali, proponiamo una ristrutturazione dello schema.

1. **Utente**(Email, Nickname, Password, Nome, Cognome, Anno_di_nascita, Luogo_di_nascita)

1NF: Tutti gli attributi sono atomici.
2NF: La chiave primaria è Email e tutti gli attributi dipendono interamente da essa.
3NF: Nessuna dipendenza transitiva.
BCNF: Email è l'unico determinante.

Non necessita di normalizzazione.

2. **Skill**(Competenza, Livello, Email_Amministratore)

1NF: Tutti gli attributi sono atomici.
2NF: La chiave primaria è composta (Competenza, Livello) e nessun attributo dipende solo da una parte della chiave.
3NF: Nessuna dipendenza transitiva.
BCNF: Nessuna anomalia.

Non necessita di normalizzazione.

3. **Skill_Curriculum**(Email_Utente, Competenza, Livello)

1NF: Tutti gli attributi sono atomici.
2NF: La chiave primaria è composta (Email_Utente, Competenza, Livello) e nessun attributo dipende solo da una parte della chiave.
3NF: Nessuna dipendenza transitiva.
BCNF: Nessuna anomalia.

Non necessita di normalizzazione.

4. **Amministratore**(Email, Codice_sicurezza)

1NF: Tutti gli attributi sono atomici.
2NF: La chiave primaria è Email, e tutti gli attributi dipendono interamente da essa.

3NF: Nessuna dipendenza transitiva.
BCNF: Nessuna anomalia.

Non necessita di normalizzazione.

5. **Creatore**(Email, #nr_progetti, Affidabilità)

1NF: Tutti gli attributi sono atomici.
2NF: La chiave primaria è Email, e tutti gli attributi dipendono interamente da essa.
3NF: Nessuna dipendenza transitiva.
BCNF: Nessuna anomalia.

Nota: #nr_progetti è ridondante ma mantenuto per motivi di efficienza.

Non necessita di normalizzazione.

6. **Progetto**(Nome, Descrizione, Data_inserimento, Stato, Budget, Data_Limite, Tipo, Email_Creatore)

1NF: Tutti gli attributi sono atomici.
2NF: La chiave primaria è Nome, e tutti gli attributi dipendono interamente da essa.
3NF: Nessuna dipendenza transitiva.
BCNF: Nessuna anomalia.

Non necessita di normalizzazione.

7. **Componente**(ID, Nome, Descrizione, Prezzo, Quantità, Nome_Progetto)

1NF: Tutti gli attributi sono atomici.
2NF: Nessuna dipendenza parziale.
3NF: Nessuna dipendenza transitiva.
BCNF: Nessuna anomalia.

Non necessita di normalizzazione.

8. **Profilo**(ID, Nome, Nome_Progetto)

1NF: Tutti gli attributi sono atomici.
2NF: Nessuna dipendenza parziale.
3NF: Nessuna dipendenza transitiva.
BCNF: Nessuna anomalia.

Non necessita di normalizzazione.

9. **Skill_Richiesta**(ID_Profilo, Competenza, Livello)

- **1NF**: Gli attributi sono atomici.
- **2NF**: La chiave primaria è (ID_Profilo, Competenza, Livello) e tutti gli attributi dipendono da essa.
- **3NF**: Nessuna dipendenza transitiva.
- **BCNF**: Ogni determinante è chiave candidata.

Non necessita di normalizzazione.

10. **Commento**(ID, Data, Testo, Nome_Progetto, Email_Utente)

- **1NF**: Tutti gli attributi contengono valori atomici.
- **2NF**: ID è chiave primaria e gli altri attributi dipendono interamente da esso.
- **3NF**: Nessuna dipendenza transitiva.
- **BCNF**: ID è chiave candidata e determina tutti gli altri attributi.

Non necessita di normalizzazione.

11. **Risposta**(ID_Commento, Email_Creatore, Testo, Data)

- **1NF**: Gli attributi sono atomici.
- **2NF**: ID_Commento è la chiave primaria e determina tutti gli altri attributi.
- **3NF**: Nessuna dipendenza transitiva.
- **BCNF**: ID_Commento è chiave candidata e determina tutti gli altri attributi.

Non necessita di normalizzazione.

12. **Reward**(Codice, Descrizione, Foto, Nome_Progetto)

- **1NF**: Tutti gli attributi sono atomici.
- **2NF**: Codice è la chiave primaria e tutti gli attributi dipendono da esso.
- **3NF**: Nessuna dipendenza transitiva.
- **BCNF**: Codice è determinante unico.

Non necessita di normalizzazione.

13. **Finanziamento**(ID, Data, Importo, Email_Utente, Codice_Reward, Nome_Progetto)

- **1NF**: Tutti gli attributi sono atomici.
- **2NF**: ID è chiave primaria e tutti gli attributi dipendono interamente da ID.
- **3NF**: Non ci sono dipendenze transitive.
- **BCNF**: ID è chiave candidata e determina tutti gli altri attributi.

Non necessita di normalizzazione.

14. **Candidatura**(ID, Esito, Email_Utente, ID_Profilo)

- **1NF**: Tutti gli attributi sono atomici.
- **2NF**: ID è chiave primaria e tutti gli altri attributi dipendono da essa.
- **3NF**: Nessuna dipendenza transitiva.
- **BCNF**: ID è chiave candidata e determina tutti gli altri attributi.

Non necessita di normalizzazione.

15. **Foto**(ID, Percorso, Nome_Progetto)

• **1NF**: Tutti gli attributi contengono valori atomici.
• **2NF**: La chiave primaria è ID, e tutti gli attributi dipendono interamente da essa.
• **3NF**: Nessuna dipendenza transitiva.
• **BCNF**: ID è chiave candidata e determina tutti gli altri attributi.

Non necessita di normalizzazione.

Tutte le tabelle sono state analizzate rispetto alle prime quattro forme normali. Dai controlli effettuati è emerso che il modello logico rispetta pienamente la 1NF, 2NF, 3NF e la BCNF. Dunque non ce stato bisogno di effettuare la normalizzazione.

5. DESCRIZIONE AD ALTO LIVELLO DELLE FUNZIONALITÀ DELL'APPLICAZIONE WEB

BOSTARTER è una piattaforma di crowdfunding progettata per supportare la realizzazione di progetti innovativi nel campo hardware e software. L'applicazione facilita l'incontro tra creatori con idee imprenditoriali e una comunità di sostenitori interessati a finanziare e partecipare allo sviluppo di nuove tecnologie.

La piattaforma fornisce interfacce specifiche ottimizzate per ogni tipologia di utente. Tutti gli utenti visualizzano i progetti finanziati con relative reward, le candidature inviate con stato aggiornato, e le proprie competenze professionali. In particolare, i creatori che hanno acquisito privilegi aggiuntivi, accedono a sezioni che mostrano progetti creati con percentuali di finanziamento, candidature ricevute da gestire, commenti con possibilità di risposta, e metriche sulla propria affidabilità. Gli amministratori dispongono di strumenti specializzati per gestire competenze globali e visualizzare statistiche di sistema

Accesso e Navigazione Iniziale

All'apertura del sito, gli utenti vengono accolti da una homepage intuitiva che presenta tutti i progetti disponibili e non, organizzati in card. Ogni card contiene informazioni essenziali: immagine rappresentativa del progetto, titolo, breve descrizione, stato attuale (aperto/chiuso), progresso di finanziamento con percentuale di completamento e giorni rimanenti per la raccolta fondi. Il sistema di navigazione offre strumenti di filtraggio per categoria (hardware/software), stato del progetto e ricerca testuale che permette di trovare rapidamente progetti di interesse specifico.

Processo di Registrazione e Autenticazione

Il sistema di registrazione è progettato per raccogliere informazioni complete necessarie per creare profili utente verificabili e completi. Durante la fase di registrazione, gli utenti forniscono dati personali come email univoca, nickname, password sicura, nome, cognome, data di nascita e luogo di nascita. Una caratteristica distintiva è la possibilità di scegliere immediatamente di diventare creatori durante la registrazione selezionando un'apposita opzione, oppure di acquisire questi privilegi successivamente attraverso una funzionalità di promozione disponibile nella home.

Il sistema di autenticazione implementa livelli di sicurezza differenziati: utenti standard e creatori accedono con credenziali base (email e password), mentre gli amministratori utilizzano un sistema di autenticazione rafforzata che richiede un codice di sicurezza aggiuntivo oltre alle credenziali standard, garantendo maggiore protezione per le funzioni amministrative critiche.

Esplorazione Dettagliata e Processo di Finanziamento

Ogni progetto dispone di una pagina dettagliata completa che fornisce una visione approfondita dell'iniziativa. Queste pagine includono informazioni complete sul creatore con indicatori di affidabilità, foto, descrizione estesa del progetto, specifiche tecniche dettagliate e cronologia dei finanziamenti ricevuti.

Per i progetti hardware, vengono presentate liste dettagliate dei componenti fisici necessari, ciascuno con nome, descrizione tecnica, prezzo unitario e quantità richiesta, permettendo ai potenziali sostenitori di comprendere esattamente come verranno utilizzati i fondi raccolti. Per i progetti software, sono mostrati i profili professionali eventualmente richiesti dal creatore, con specifiche delle competenze necessarie e livelli minimi richiesti per ogni skill.

Il processo di finanziamento rappresenta un elemento distintivo della piattaforma: ogni contributo economico deve essere obbligatoriamente accompagnato dalla selezione di una reward specifica tra quelle offerte dal creatore. Questo meccanismo trasforma il semplice atto di finanziamento in un'esperienza di partecipazione più coinvolgente e gratificante, dove i sostenitori ricevono riconoscimenti tangibili per il loro supporto.

Funzionalità Specifiche per Tipologia di Utente

La piattaforma BOSTARTER offre funzionalità differenti in base al ruolo dell’utente, ciascuna supportata da interfacce grafiche specifiche che rendono l’interazione con il sistema semplice, immediata e coerente con le esigenze dell’utente. Ogni utente, una volta autenticato, accede a una dashboard personalizzata che raccoglie tutte le funzionalità pertinenti al proprio ruolo, presentate in modo ordinato e intuitivo.

Gli utenti standard possono partecipare attivamente alla vita della piattaforma. Una delle funzionalità principali è la possibilità di costruire un profilo professionale, selezionando competenze dalla lista globale e assegnando a ciascuna un livello di expertise da 0 a 5. Questa sezione, accessibile e modificabile dalla propria area personale, consente di creare un curriculum digitale. Tramite la dashboard, gli utenti possono visualizzare i progetti finanziati, le reward ricevute, e gestire le candidature inviate per progetti software, con stato aggiornato in tempo reale (in attesa, accettata, rifiutata). Il sistema di finanziamento permette di contribuire economicamente a uno o più progetti, scegliendo una reward tra quelle proposte. Ogni operazione è registrata e consultabile all’interno della propria area utente.

Per i progetti software, gli utenti possono candidarsi come collaboratori. La piattaforma implementa un motore di matching automatico che verifica se tutte le competenze richieste dal profilo sono soddisfatte: solo in caso di corrispondenza completa, la candidatura viene accettata e inoltrata al creatore. Questo sistema garantisce che le collaborazioni siano sempre coerenti con i bisogni del progetto e le competenze reali dei candidati. Gli utenti possono inoltre interagire con i creatori tramite commenti pubblici lasciati sotto ogni progetto, favorendo il dialogo e aumentando la trasparenza.

Gli utenti creatori hanno accesso a una dashboard più estesa, progettata per la gestione professionale dei progetti. Da qui possono creare nuovi progetti specificando nome, descrizione, budget, data limite, immagine e categoria (hardware o software). Per i progetti hardware, è possibile elencare in dettaglio i componenti necessari, con prezzi e quantità. Nei progetti software, invece, i creatori possono definire i profili professionali richiesti indicando le competenze necessarie e i livelli minimi. Ogni progetto può includere una o più reward personalizzate, con immagini e descrizioni, da associare ai finanziamenti.

La dashboard dei creatori include strumenti avanzati per monitorare l'andamento delle campagne: sono visibili in tempo reale le percentuali di completamento, i finanziamenti ricevuti, il numero di sostenitori, i giorni rimanenti e le proiezioni di raccolta fondi. Per i progetti software, è disponibile una sezione dedicata alla gestione delle candidature ricevute, con visualizzazione delle competenze dei candidati, stato della candidatura e strumenti per accettare o rifiutare. I creatori hanno inoltre accesso a una sezione per la gestione dei commenti, possibilità di rispondere direttamente dalla propria interfaccia. Infine, è disponibile un modulo di monitoraggio della propria affidabilità, aggiornato automaticamente dal sistema sulla base del numero di progetti finanziati con successo.

Gli amministratori dispongono di una dashboard specializzata, dedicata alla gestione e al controllo dell'integrità del sistema. Il loro compito principale è gestire la lista globale delle competenze disponibili sulla piattaforma: ogni nuova competenza inserita viene automaticamente registrata con tutti i livelli da 0 a 5, garantendo coerenza nella classificazione e prevenendo ambiguità.

Questa organizzazione modulare delle dashboard, differenziata per ruoli, consente a ogni utente di operare in modo efficiente all'interno della piattaforma, concentrandosi solo sulle funzionalità rilevanti e contribuendo a un'esperienza d'uso fluida, professionale e coerente.

Gestione del Ciclo di Vita dei Progetti

Il sistema implementa una sofisticata logica di gestione automatica che monitora costantemente lo stato di tutti i progetti attivi sulla piattaforma. Due meccanismi principali determinano la chiusura automatica dei progetti: il raggiungimento del budget target attraverso la somma cumulativa dei finanziamenti ricevuti, e il superamento della data limite prestabilita dal creatore durante la fase di creazione del progetto.

Quando un progetto raggiunge il budget target, il sistema lo marca automaticamente come "completato con successo" e impedisce ulteriori finanziamenti, garantendo che non vengano raccolti fondi in eccesso rispetto agli obiettivi dichiarati. Allo stesso modo, i progetti che superano la data limite senza aver raggiunto il budget vengono automaticamente chiusi e marcati come "scaduti", fornendo chiarezza sui risultati della campagna.

Il sistema calcola e aggiorna dinamicamente l'affidabilità dei creatori utilizzando una formula trasparente: la percentuale dei progetti del creatore che hanno ricevuto almeno un finanziamento. Questo indicatore viene ricalcolato automaticamente ogni volta che un creatore pubblica un nuovo progetto o uno dei suoi progetti riceve un finanziamento, fornendo agli utenti un metro di valutazione oggettivo e sempre aggiornato.

Sistema Completo di Statistiche

La piattaforma implementa un sistema di statistiche pubbliche che serve sia come strumento di gamification per incentivare la partecipazione attiva, sia come meccanismo di trasparenza per aiutare gli utenti nelle decisioni di investimento e collaborazione. Tre classifiche principali sono costantemente aggiornate e visibili a tutti gli utenti.

La classifica dei creatori più affidabili presenta i primi tre creatori ordinati per percentuale di successo dei loro progetti, calcolata come rapporto tra progetti finanziati e progetti totali creati. Questa metrica incentiva i creatori a proporre progetti di qualità e ben pianificati.

La classifica dei progetti aperti più vicini al completamento mostra le tre iniziative attive che hanno la minore differenza tra budget richiesto e finanziamenti già ricevuti, aiutando i sostenitori a identificare progetti che necessitano di poco supporto aggiuntivo per raggiungere i loro obiettivi.

La classifica dei sostenitori più attivi presenta i tre utenti che hanno erogato il maggior importo totale in finanziamenti, riconoscendo pubblicamente i membri più generosi della community e incentivando comportamenti di sostegno attivo.

Sistema di Tracciabilità e Logging

Per garantire completa trasparenza, supportare attività di manutenzione e debugging, e fornire una traccia completa delle attività per eventuali analisi future, la piattaforma implementa un sistema comprensivo di logging automatico. Questo sistema registra automaticamente tutti gli eventi significativi che avvengono sulla piattaforma: registrazioni di nuovi utenti, creazioni di progetti, finanziamenti ricevuti, candidature inviate, accettazioni e rifiuti di candidature, inserimenti di commenti e risposte, e tutte le altre interazioni importanti.

Il sistema di logging utilizza tecnologie NoSQL per garantire scalabilità nella gestione di grandi volumi di dati di log e flessibilità nella struttura dei dati registrati. I dati di logging sono mantenuti completamente separati dal database principale per non impattare sulle prestazioni delle operazioni di business critiche e permettere analisi indipendenti sui pattern di utilizzo della piattaforma.

Conclusione

BOSTARTER si configura come una piattaforma web solida e ben strutturata, in grado di accompagnare ogni fase del ciclo di vita di un progetto tecnologico: dalla proposta iniziale, alla raccolta fondi, fino alla sua realizzazione. L’intera esperienza è guidata da una logica di partecipazione attiva e responsabilità condivisa tra gli utenti.

L’architettura del sistema, basata su ruoli differenziati (utenti standard, creatori, amministratori), consente di garantire accesso selettivo alle funzionalità, mantenendo al tempo stesso un’interfaccia coerente e un’esperienza d’uso fluida. Gli utenti possono contribuire con le proprie competenze e partecipare

attivamente alla community; i creatori hanno a disposizione strumenti avanzati per gestire in modo professionale i propri progetti; gli amministratori assicurano la qualità e la coerenza del sistema.

Grazie all'integrazione di sistemi di reputazione, automazioni, dashboard personalizzate e tracciabilità degli eventi, la piattaforma promuove un ambiente trasparente e meritocratico, dove la qualità dei progetti viene riconosciuta e premiata.

BOSTARTER non è soltanto uno strumento tecnico, ma un ecosistema collaborativo pensato per valorizzare idee, competenze e iniziativa individuale. La sua struttura modulare e scalabile lo rende adatto a sostenere nel tempo progetti innovativi, contribuendo a creare valore all'interno della comunità universitaria e oltre.

6. CODICE SQL COMPLETO DELLO SCHEMA DELLA BASE DI DATI.

```
DROP DATABASE IF EXISTS BOSTARTER;
CREATE DATABASE IF NOT EXISTS BOSTARTER;
USE BOSTARTER;

-- TABELLE PRINCIPALI

CREATE TABLE UTENTE(
    Email               VARCHAR(100) PRIMARY KEY,
    Nickname            VARCHAR(50) UNIQUE NOT NULL,
    Password            VARCHAR(255) NOT NULL,
    Nome                VARCHAR(50) NOT NULL,
    Cognome             VARCHAR(50) NOT NULL,
    Anno_Di_Nascita     DATE NOT NULL,
    Luogo_Di_Nascita    VARCHAR(100) NOT NULL
);

CREATE TABLE AMMINISTRATORE(
    Email               VARCHAR(100) PRIMARY KEY,
    Codice_Sicurezza    VARCHAR(50) NOT NULL,
    FOREIGN KEY (Email) REFERENCES UTENTE(Email) ON DELETE CASCADE
);

CREATE TABLE SKILL(
    Competenza  VARCHAR(100),
    Livello     INT CHECK (LIVELLO BETWEEN 0 AND 5),
    Email_Amministratore VARCHAR(100),
    PRIMARY KEY (Competenza, Livello),
    FOREIGN KEY (Email_Amministratore) REFERENCES AMMINISTRATORE(Email) ON DELETE
CASCADE
);

CREATE TABLE SKILL_CURRICULUM(
    Email_Utente    VARCHAR(100),
    Competenza      VARCHAR(100),
    Livello         INT,
    PRIMARY KEY (Email_Utente, Competenza, Livello),
    FOREIGN KEY (Email_Utente) REFERENCES UTENTE(Email) ON DELETE CASCADE,
    FOREIGN KEY (Competenza, Livello) REFERENCES SKILL(Competenza, LIVELLO) ON
DELETE CASCADE
);

CREATE TABLE CREATORE (
    Email           VARCHAR(100) PRIMARY KEY,
    Nr_Progetti     INT DEFAULT 0,
    Affidabilita    FLOAT DEFAULT 0,
    FOREIGN KEY (Email) REFERENCES UTENTE(Email) ON DELETE CASCADE
);

CREATE TABLE PROGETTO(
    Nome                VARCHAR(100) PRIMARY KEY,
    Descrizione         TEXT NOT NULL,
    Data_Inserimento    DATE NOT NULL,
    Stato               ENUM('aperto', 'chiuso')NOT NULL DEFAULT 'aperto',
    Budget              DECIMAL(10,2) NOT NULL,
    Data_Limite         DATE NOT NULL,
    Tipo                ENUM('Hardware', 'Software') NOT NULL,
    Email_Creatore      VARCHAR(100) NOT NULL,
    FOREIGN KEY (Email_Creatore) REFERENCES CREATORE(Email) ON DELETE CASCADE
);
```

```sql
CREATE TABLE FOTO (
    ID              INT AUTO_INCREMENT PRIMARY KEY,
    Percorso        TEXT NOT NULL,
    Nome_Progetto   VARCHAR(100) NOT NULL,
    FOREIGN KEY (Nome_Progetto) REFERENCES PROGETTO(Nome) ON DELETE CASCADE
);

CREATE TABLE COMPONENTE(
    ID              INT AUTO_INCREMENT PRIMARY KEY,
    Nome            VARCHAR(100),
    Descrizione     TEXT NOT NULL,
    Prezzo          DECIMAL(10,2) NOT NULL,
    Quantita        INT NOT NULL,
    Nome_Progetto   VARCHAR(100) NOT NULL,
    FOREIGN KEY (Nome_Progetto) REFERENCES PROGETTO(Nome) ON DELETE CASCADE
);

CREATE TABLE PROFILO(
    ID              INT AUTO_INCREMENT PRIMARY KEY,
    Nome            VARCHAR(100) NOT NULL,
    Nome_Progetto   VARCHAR(100) NOT NULL,
    FOREIGN KEY (Nome_Progetto) REFERENCES PROGETTO(Nome) ON DELETE CASCADE
);

CREATE TABLE SKILL_RICHIESTA(
    ID_Profilo  INT,
    Competenza  VARCHAR(100),
    Livello     INT,
    PRIMARY KEY (ID_Profilo, Competenza, Livello),
    FOREIGN KEY (ID_Profilo) REFERENCES PROFILO(ID) ON DELETE CASCADE,
    FOREIGN KEY (Competenza, Livello) REFERENCES SKILL(Competenza, LIVELLO) ON
DELETE CASCADE
);

CREATE TABLE COMMENTO(
    ID              INT AUTO_INCREMENT PRIMARY KEY,
    Data            TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Testo           TEXT NOT NULL,
    Nome_Progetto   VARCHAR(100) NOT NULL,
    Email_Utente    VARCHAR(100) NOT NULL,
    FOREIGN KEY (Nome_Progetto) REFERENCES PROGETTO(Nome) ON DELETE CASCADE,
    FOREIGN KEY (Email_Utente) REFERENCES UTENTE(Email) ON DELETE CASCADE
);

CREATE TABLE RISPOSTA(
    ID_Commento     INT PRIMARY KEY,
    Email_Creatore  VARCHAR(100) NOT NULL,
    Testo           TEXT NOT NULL,
    Data            TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (ID_Commento) REFERENCES COMMENTO(ID) ON DELETE CASCADE,
    FOREIGN KEY (Email_Creatore) REFERENCES CREATORE(Email) ON DELETE CASCADE
);

CREATE TABLE REWARD(
    Codice          VARCHAR(100) PRIMARY KEY,
    Descrizione     TEXT NOT NULL,
    Foto            TEXT NOT NULL,
    Nome_Progetto   VARCHAR(100) NOT NULL,
    FOREIGN KEY (Nome_Progetto) REFERENCES PROGETTO(Nome) ON DELETE CASCADE
);

CREATE TABLE FINANZIAMENTO(
    ID              INT AUTO_INCREMENT PRIMARY KEY,
    Data            TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Importo         DECIMAL(10,2) NOT NULL,
```

```
    Email_Utente    VARCHAR(100) NOT NULL,
    Codice_Reward   VARCHAR(100),
    Nome_Progetto   VARCHAR(100) NOT NULL,
    FOREIGN KEY (Email_Utente) REFERENCES UTENTE(Email) ON DELETE CASCADE,
    FOREIGN KEY (Codice_Reward) REFERENCES REWARD(Codice) ON DELETE SET NULL,
    FOREIGN KEY (Nome_Progetto) REFERENCES PROGETTO(Nome) ON DELETE CASCADE
);

CREATE TABLE CANDIDATURA(
    ID                  INT AUTO_INCREMENT PRIMARY KEY,
    Esito               BOOLEAN DEFAULT NULL,
    Data_Candidatura    TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Email_Utente        VARCHAR(100) NOT NULL,
    ID_Profilo          INT NOT NULL,
    FOREIGN KEY (Email_Utente) REFERENCES UTENTE(Email) ON DELETE CASCADE,
    FOREIGN KEY (ID_Profilo) REFERENCES PROFILO(ID) ON DELETE CASCADE
);

-- STORED PROCEDURES
DELIMITER $$

CREATE PROCEDURE AutenticaUtente(
    IN p_Email VARCHAR(100),
    IN p_Password VARCHAR(255)
)
BEGIN
    DECLARE v_Count INT;
SELECT COUNT(*) INTO v_Count
FROM UTENTE
WHERE Email = p_Email AND Password = p_Password
    LIMIT 1;

IF v_Count = 1 THEN
SELECT 'Autenticazione riuscita' AS Messaggio, p_Email AS Email;
ELSE
SELECT 'Autenticazione fallita' AS Messaggio, NULL AS Email;
END IF;
END $$

CREATE PROCEDURE RegistraUtente(
    IN p_Email VARCHAR(100),
    IN p_Nickname VARCHAR(50),
    IN p_Password VARCHAR(255),
    IN p_Nome VARCHAR(50),
    IN p_Cognome VARCHAR(50),
    IN p_Anno_Di_Nascita DATE,
    IN p_Luogo_Di_Nascita VARCHAR(100),
    IN p_IsCreatore BOOLEAN
)
BEGIN
    INSERT INTO UTENTE (
        Email, Nickname, Password, Nome, Cognome,
        Anno_Di_Nascita, Luogo_Di_Nascita
    ) VALUES (
        p_Email, p_Nickname, p_Password, p_Nome, p_Cognome,
        p_Anno_Di_Nascita, p_Luogo_Di_Nascita
    );

    IF p_IsCreatore THEN
        INSERT INTO CREATORE (Email, Nr_Progetti, Affidabilita)
        VALUES (p_Email, 0, 0);
    END IF;
END$$

CREATE PROCEDURE LoginUtente (
```

```
    IN in_email VARCHAR(255),
    IN in_password VARCHAR(255)
)
BEGIN
    SELECT Email
    FROM UTENTE
    WHERE Email = in_email
      AND Password = in_password;
END $$

CREATE PROCEDURE LoginAmministratore(
    IN p_Email VARCHAR(100),
    IN p_Password VARCHAR(255),
    IN p_CodiceSicurezza VARCHAR(50)
)
BEGIN
    DECLARE v_Count INT;
SELECT COUNT(*) INTO v_Count
FROM UTENTE U
         JOIN AMMINISTRATORE A ON U.Email = A.Email
WHERE U.Email = p_Email AND U.Password = p_Password AND A.Codice_Sicurezza =
p_CodiceSicurezza;

IF v_Count = 1 THEN
SELECT 'Login amministratore riuscito' AS Messaggio, p_Email AS Email;
ELSE
SELECT 'Credenziali amministratore non valide' AS Messaggio, NULL AS Email;
END IF;
END $$

CREATE PROCEDURE PromuoviACreatore(IN p_Email VARCHAR(100))
BEGIN
    -- Controlla se l'utente esiste
    IF EXISTS (SELECT 1 FROM UTENTE WHERE Email = p_Email) THEN
        -- Controlla se è già creatore
        IF NOT EXISTS (SELECT 1 FROM CREATORE WHERE Email = p_Email) THEN
            INSERT INTO CREATORE (Email, Nr_Progetti, Affidabilita)
            VALUES (p_Email, 0, 0);
END IF;
END IF;
END$$

CREATE PROCEDURE InserisciProgetto(
    IN p_Nome VARCHAR(100),
    IN p_Descrizione TEXT,
    IN p_DataInserimento DATE,
    IN p_Budget DECIMAL(10,2),
    IN p_DataLimite DATE,
    IN p_Stato ENUM('aperto', 'chiuso'),
    IN p_Tipo ENUM('Hardware', 'Software'),
    IN p_EmailCreatore VARCHAR(100)
        )
BEGIN
INSERT INTO PROGETTO (Nome, Descrizione, Data_Inserimento, Budget, Data_Limite,
Stato, Tipo, Email_Creatore)
VALUES (p_Nome, p_Descrizione, p_DataInserimento, p_Budget, p_DataLimite,
p_Stato, p_Tipo, p_EmailCreatore);
END $$

CREATE PROCEDURE InserisciFoto(
    IN p_Percorso TEXT,
    IN p_NomeProgetto VARCHAR(100)
)
BEGIN
    INSERT INTO FOTO (Percorso, Nome_Progetto)
```

```
    VALUES (p_Percorso, p_NomeProgetto);
END $$

CREATE PROCEDURE InserisciReward(
    IN p_Codice VARCHAR(100),
    IN p_Descrizione TEXT,
    IN p_Foto TEXT,
    IN p_NomeProgetto VARCHAR(100)
)
BEGIN
INSERT INTO REWARD (Codice, Descrizione, Foto, Nome_Progetto)
VALUES (p_Codice, p_Descrizione, p_Foto, p_NomeProgetto);

END $$

CREATE PROCEDURE InserisciSkill(
    IN p_Competenza VARCHAR(100),
    IN p_Livello INT
)
BEGIN
    -- Usiamo admin di default
    DECLARE v_admin_email VARCHAR(100) DEFAULT 'admin@bostarter.com';

    IF NOT EXISTS (
        SELECT 1 FROM SKILL
        WHERE Competenza = p_Competenza AND Livello = p_Livello
    ) THEN
        INSERT INTO SKILL (Competenza, Livello, Email_Amministratore)
        VALUES (p_Competenza, p_Livello, v_admin_email);
END IF;
END $$

-- Per inserimento skill con admin specifico
CREATE PROCEDURE InserisciSkillAdmin(
    IN p_Competenza VARCHAR(100),
    IN p_Livello INT,
    IN p_Email_Amministratore VARCHAR(100)
)
BEGIN
    IF NOT EXISTS (
        SELECT 1 FROM SKILL
        WHERE Competenza = p_Competenza AND Livello = p_Livello
    ) THEN
        INSERT INTO SKILL (Competenza, Livello, Email_Amministratore)
        VALUES (p_Competenza, p_Livello, p_Email_Amministratore);
END IF;
END $$

CREATE PROCEDURE InserisciComponente(
    IN p_Nome VARCHAR(100),
    IN p_Descrizione TEXT,
    IN p_Prezzo DECIMAL(10,2),
    IN p_Quantita INT,
    IN p_NomeProgetto VARCHAR(100)
)
BEGIN
INSERT INTO COMPONENTE (Nome, Descrizione, Prezzo, Quantita, Nome_Progetto)
VALUES (p_Nome, p_Descrizione, p_Prezzo, p_Quantita, p_NomeProgetto);
END $$

CREATE PROCEDURE FinanziaProgetto(
    IN p_Email VARCHAR(100),
    IN p_NomeProgetto VARCHAR(100),
    IN p_Importo DECIMAL(10,2),
    IN p_CodiceReward VARCHAR(100)
```

```
)
BEGIN
INSERT INTO FINANZIAMENTO (Data, Importo, Email_Utente, Codice_Reward,
Nome_Progetto)
VALUES (NOW(), p_Importo, p_Email, p_CodiceReward, p_NomeProgetto);
END $$

CREATE PROCEDURE InserisciCandidatura(
    IN p_Email VARCHAR(100),
    IN p_IDProfilo INT
)
BEGIN
    DECLARE v_Count INT;

SELECT COUNT(*) INTO v_Count
FROM (
         SELECT sr.Competenza, sr.Livello
         FROM SKILL_RICHIESTA sr
                  LEFT JOIN SKILL_CURRICULUM sc
                            ON sc.Email_Utente = p_Email
                                AND sc.Competenza = sr.Competenza
                                AND sc.Livello >= sr.Livello
         WHERE sr.ID_Profilo = p_IDProfilo
           AND sc.Email_Utente IS NULL
     ) AS Mancanti;

IF v_Count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'L utente non possiede tutte le skill richieste per
questo profilo';
ELSE
        INSERT INTO CANDIDATURA (Email_Utente, ID_Profilo, Esito,
Data_Candidatura)
        VALUES (p_Email, p_IDProfilo, NULL, NOW());
END IF;
END $$

CREATE PROCEDURE AccettaCandidatura(
    IN p_IDCandidatura INT,
    IN p_Esito BOOLEAN
)
BEGIN
UPDATE CANDIDATURA
SET Esito = p_Esito
WHERE ID = p_IDCandidatura;
END $$

CREATE PROCEDURE InserisciProfiloRichiesto(
    IN p_NomeProfilo VARCHAR(100),
    IN p_NomeProgetto VARCHAR(100)
)
BEGIN
    INSERT INTO PROFILO (Nome, Nome_Progetto)
    VALUES (p_NomeProfilo, p_NomeProgetto);
    SELECT LAST_INSERT_ID() AS ID_Profilo; -- Restituisce l'ID del profilo appena
inserito
END $$

CREATE PROCEDURE InserisciRisposta(
    IN p_IDCommento INT,
    IN p_EmailCreatore VARCHAR(100),
    IN p_Testo TEXT
)
BEGIN
INSERT INTO RISPOSTA (ID_Commento, Email_Creatore, Testo)
```

```
VALUES (p_IDCommento, p_EmailCreatore, p_Testo);
END $$

CREATE PROCEDURE InserisciSkillCurriculum(
    IN p_Email VARCHAR(100),
    IN p_Competenza VARCHAR(100),
    IN p_Livello INT
)
BEGIN
    DECLARE v_LivelloAttuale INT;

    -- Controlla se esiste già una skill per quell'utente e competenza
    SELECT MAX(Livello) INTO v_LivelloAttuale
    FROM SKILL_CURRICULUM
    WHERE Email_Utente = p_Email AND Competenza = p_Competenza;

    IF v_LivelloAttuale IS NULL THEN
        -- se non esiste: inserisci nuova skill
        INSERT INTO SKILL_CURRICULUM (Email_Utente, Competenza, Livello)
        VALUES (p_Email, p_Competenza, p_Livello);
    ELSEIF p_Livello > v_LivelloAttuale THEN
        -- esiste già ma il nuovo livello è superiore: elimina il vecchio e
inserisci il nuovo
        DELETE FROM SKILL_CURRICULUM
        WHERE Email_Utente = p_Email AND Competenza = p_Competenza AND Livello =
v_LivelloAttuale;

        INSERT INTO SKILL_CURRICULUM (Email_Utente, Competenza, Livello)
        VALUES (p_Email, p_Competenza, p_Livello);
    END IF;
END $$

CREATE PROCEDURE InserisciSkillRichiesta(
    IN p_IDProfilo INT,
    IN p_Competenza VARCHAR(100),
    IN p_Livello INT
)
BEGIN
    INSERT INTO SKILL_RICHIESTA (ID_Profilo, Competenza, Livello)
    VALUES (p_IDProfilo, p_Competenza, p_Livello);
END $$

CREATE PROCEDURE InserisciCommento(
    IN p_Email VARCHAR(100),
    IN p_NomeProgetto VARCHAR(100),
    IN p_Testo TEXT
)
BEGIN
INSERT INTO COMMENTO (Data, Testo, Nome_Progetto, Email_Utente)
VALUES (NOW(), p_Testo, p_NomeProgetto, p_Email);
END $$

DELIMITER ;

-- TRIGGERS
DELIMITER $$

CREATE TRIGGER IncrementaNrProgetti
    AFTER INSERT ON PROGETTO
    FOR EACH ROW
BEGIN
    UPDATE CREATORE
    SET Nr_Progetti = Nr_Progetti + 1
    WHERE Email = NEW.Email_Creatore;
END $$
```

```
CREATE TRIGGER AggiornaAffidabilita
    AFTER INSERT ON FINANZIAMENTO
    FOR EACH ROW
BEGIN
    DECLARE v_CreatoreEmail VARCHAR(100);
    DECLARE v_Finanziati INT DEFAULT 0;
    DECLARE v_Totale INT DEFAULT 0;
    DECLARE v_NuovaAffidabilita DECIMAL(5,2);

    SELECT Email_Creatore INTO v_CreatoreEmail
    FROM PROGETTO WHERE Nome = NEW.Nome_Progetto;

    SELECT COUNT(DISTINCT p.Nome) INTO v_Finanziati
    FROM PROGETTO p
             JOIN FINANZIAMENTO f ON f.Nome_Progetto = p.Nome
    WHERE p.Email_Creatore = v_CreatoreEmail;

    SELECT COUNT(*) INTO v_Totale
    FROM PROGETTO WHERE Email_Creatore = v_CreatoreEmail;

    IF v_Totale > 0 THEN
        SET v_NuovaAffidabilita = CAST(v_Finanziati AS DECIMAL(5,2)) /
CAST(v_Totale AS DECIMAL(5,2));
    UPDATE CREATORE
    SET Affidabilita = v_NuovaAffidabilita
    WHERE Email = v_CreatoreEmail;
END IF;
END $$

CREATE TRIGGER RicalcolaAffidabilitaDopoNuovoProgetto
    AFTER INSERT ON PROGETTO
    FOR EACH ROW
BEGIN
    DECLARE v_Finanziati INT DEFAULT 0;
    DECLARE v_Totale INT DEFAULT 0;
    DECLARE v_NuovaAffidabilita DECIMAL(5,2);

    SELECT COUNT(DISTINCT p.Nome) INTO v_Finanziati
    FROM PROGETTO p
             JOIN FINANZIAMENTO f ON f.Nome_Progetto = p.Nome
    WHERE p.Email_Creatore = NEW.Email_Creatore;

    SELECT COUNT(*) INTO v_Totale
    FROM PROGETTO WHERE Email_Creatore = NEW.Email_Creatore;

    IF v_Totale > 0 THEN
        SET v_NuovaAffidabilita = CAST(v_Finanziati AS DECIMAL(5,2)) /
CAST(v_Totale AS DECIMAL(5,2));
    UPDATE CREATORE
    SET Affidabilita = v_NuovaAffidabilita
    WHERE Email = NEW.Email_Creatore;
END IF;
END $$

CREATE TRIGGER ChiudiProgettoBudget
    AFTER INSERT ON FINANZIAMENTO
    FOR EACH ROW
BEGIN
    DECLARE v_TotaleFinanziamenti DECIMAL(10,2);
    DECLARE v_Budget DECIMAL(10,2);

    SELECT SUM(Importo) INTO v_TotaleFinanziamenti
    FROM FINANZIAMENTO WHERE Nome_Progetto = NEW.Nome_Progetto;
```

```
    SELECT Budget INTO v_Budget
    FROM PROGETTO WHERE Nome = NEW.Nome_Progetto;

    IF v_TotaleFinanziamenti >= v_Budget THEN
    UPDATE PROGETTO
    SET Stato = 'chiuso'
    WHERE Nome = NEW.Nome_Progetto;
END IF;
END $$

DELIMITER ;
-- EVENTO PER CHIUDERE PROGETTI SCADUTI
DELIMITER $$

CREATE EVENT ChiudiProgettiScaduti
ON SCHEDULE EVERY 1 DAY
DO
BEGIN
    UPDATE PROGETTO
    SET Stato = 'chiuso'
    WHERE Stato = 'aperto' AND Data_Limite <= CURDATE();
END $$

DELIMITER ;


-- VIEWS PER STATISTICHE

CREATE VIEW classifica_affidabilita AS
SELECT u.Nickname, c.affidabilita
FROM CREATORE c
         JOIN UTENTE u ON  c.Email = u.Email
ORDER BY c.affidabilita DESC
    LIMIT 3;

CREATE VIEW ProgettiQuasiCompletati AS
SELECT p.Nome,
       p.Budget - COALESCE(SUM(f.Importo), 0) AS DifferenzaResidua
FROM PROGETTO p
         LEFT JOIN FINANZIAMENTO f ON p.Nome = f.Nome_Progetto
WHERE p.Stato = 'aperto'
GROUP BY p.Nome, p.Budget
ORDER BY DifferenzaResidua ASC
    LIMIT 3;

CREATE VIEW ClassificaFinanziatori AS
SELECT u.Nickname, SUM(f.Importo) AS Totale
FROM FINANZIAMENTO f
         JOIN UTENTE u ON f.Email_Utente = u.Email
GROUP BY u.Nickname
ORDER BY Totale DESC
    LIMIT 3;

CREATE VIEW DebugProgetti AS
SELECT
    p.Nome AS Progetto,
    p.Stato,
    p.Budget,
    COALESCE(SUM(f.Importo), 0) AS TotaleFinanziato,
    DATEDIFF(p.Data_Limite, CURDATE()) AS GiorniResidui,
    c.Nr_Progetti,
    c.Affidabilita,
    u.Nickname AS Creatore
FROM PROGETTO p
         LEFT JOIN FINANZIAMENTO f ON p.Nome = f.Nome_Progetto
```

```
         JOIN CREATORE c ON p.Email_Creatore = c.Email
         JOIN UTENTE u ON u.Email = c.Email
GROUP BY p.Nome, p.Stato, p.Budget, p.Data_Limite, c.Nr_Progetti, c.Affidabilita,
u.Nickname;

-- SISTEMA DI LOGGING EVENTI BOSTARTER

CREATE TABLE IF NOT EXISTS LOG_EVENTI (
    id INT AUTO_INCREMENT PRIMARY KEY,
    evento VARCHAR(255) NOT NULL,
    email_utente VARCHAR(255) NOT NULL,
    data TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    descrizione TEXT NOT NULL,
    sincronizzato BOOLEAN DEFAULT FALSE
) ENGINE=INNODB;

DELIMITER //
CREATE PROCEDURE InserisciLogEvento(
    IN p_evento VARCHAR(255),
    IN p_email_utente VARCHAR(255),
    IN p_descrizione TEXT
)
BEGIN
    INSERT INTO LOG_EVENTI (evento, email_utente, descrizione)
    VALUES (p_evento, p_email_utente, p_descrizione);
END //
DELIMITER ;

-- TRIGGER PER LOGGING EVENTI
DELIMITER $$
CREATE TRIGGER LogNuovoUtente
    AFTER INSERT ON UTENTE
    FOR EACH ROW
BEGIN
    CALL InserisciLogEvento(
        'NUOVO_UTENTE',
        NEW.Email,
        CONCAT('Nuovo utente registrato: ', NEW.Nickname, ' (', NEW.Nome, ' ',
NEW.Cognome, ')')
    );
END $$

CREATE TRIGGER LogNuovoCreatore
    AFTER INSERT ON CREATORE
    FOR EACH ROW
BEGIN
    CALL InserisciLogEvento(
        'NUOVO_CREATORE',
        NEW.Email,
        CONCAT('Utente promosso a creatore: ', NEW.Email)
    );
END $$

CREATE TRIGGER LogNuovoProgetto
    AFTER INSERT ON PROGETTO
    FOR EACH ROW
BEGIN
    CALL InserisciLogEvento(
        'NUOVO_PROGETTO',
        NEW.Email_Creatore,
        CONCAT('Nuovo progetto creato: ', NEW.Nome, ' (', NEW.Tipo, ') - Budget:
€', NEW.Budget)
    );
END $$
```

```
CREATE TRIGGER LogNuovoFinanziamento
    AFTER INSERT ON FINANZIAMENTO
    FOR EACH ROW
BEGIN
    CALL InserisciLogEvento(
        'NUOVO_FINANZIAMENTO',
        NEW.Email_Utente,
        CONCAT('Nuovo finanziamento di €', NEW.Importo, ' per il progetto: ',
NEW.Nome_Progetto)
    );
END $$

CREATE TRIGGER LogChiusuraProgetto
    AFTER UPDATE ON PROGETTO
    FOR EACH ROW
BEGIN
    IF OLD.Stato = 'aperto' AND NEW.Stato = 'chiuso' THEN
        CALL InserisciLogEvento(
            'PROGETTO_CHIUSO',
            NEW.Email_Creatore,
            CONCAT('Progetto chiuso: ', NEW.Nome, ' - Motivo: ',
                   CASE
                       WHEN NEW.Data_Limite <= CURDATE() THEN 'Scadenza termine'
                       ELSE 'Budget raggiunto'
                   END)
        );
    END IF;
END $$

CREATE TRIGGER LogNuovaCandidatura
    AFTER INSERT ON CANDIDATURA
    FOR EACH ROW
BEGIN
    DECLARE v_nome_profilo VARCHAR(100);
    DECLARE v_nome_progetto VARCHAR(100);

    SELECT p.Nome, pr.Nome INTO v_nome_profilo, v_nome_progetto
    FROM PROFILO p
    JOIN PROGETTO pr ON p.Nome_Progetto = pr.Nome
    WHERE p.ID = NEW.ID_Profilo;

    CALL InserisciLogEvento(
        'NUOVA_CANDIDATURA',
        NEW.Email_Utente,
        CONCAT('Nuova candidatura per il profilo: ', v_nome_profilo, ' nel
progetto: ', v_nome_progetto)
    );
END $$

CREATE TRIGGER LogAccettazioneCandidatura
AFTER UPDATE ON CANDIDATURA
FOR EACH ROW
BEGIN
    DECLARE v_nome_profilo VARCHAR(100);
    DECLARE v_nome_progetto VARCHAR(100);
    DECLARE v_email_creatore VARCHAR(100);

    IF OLD.Esito IS NULL AND NEW.Esito IS NOT NULL THEN
        IF EXISTS (
            SELECT 1
            FROM PROFILO p
            JOIN PROGETTO pr ON p.Nome_Progetto = pr.Nome
            WHERE p.ID = NEW.ID_Profilo
        ) THEN
            SELECT p.Nome, pr.Nome, pr.Email_Creatore
```

```
            INTO v_nome_profilo, v_nome_progetto, v_email_creatore
            FROM PROFILO p
            JOIN PROGETTO pr ON p.Nome_Progetto = pr.Nome
            WHERE p.ID = NEW.ID_Profilo;

            CALL InserisciLogEvento(
                'CANDIDATURA_VALUTATA',
                v_email_creatore,
                CONCAT(
                    'Candidatura ',
                    CASE WHEN NEW.Esito = 1 THEN 'accettata' ELSE 'rifiutata'
END,
                    ' per il profilo: ', v_nome_profilo,
                    ' nel progetto: ', v_nome_progetto
                )
            );
        END IF;
    END IF;
END $$

CREATE TRIGGER LogNuovoCommento
    AFTER INSERT ON COMMENTO
    FOR EACH ROW
BEGIN
    CALL InserisciLogEvento(
        'NUOVO_COMMENTO',
        NEW.Email_Utente,
        CONCAT('Nuovo commento sul progetto: ', NEW.Nome_Progetto)
    );
END $$

CREATE TRIGGER LogNuovaRisposta
    AFTER INSERT ON RISPOSTA
    FOR EACH ROW
BEGIN
    CALL InserisciLogEvento(
        'NUOVA_RISPOSTA',
        NEW.Email_Creatore,
        CONCAT('Nuova risposta al commento ID: ', NEW.ID_Commento)
    );
END $$

CREATE TRIGGER LogNuovaSkill
    AFTER INSERT ON SKILL
    FOR EACH ROW
BEGIN
    CALL InserisciLogEvento(
        'NUOVA_SKILL',
        'admin@bostarter.com',
        CONCAT('Nuova skill aggiunta alla piattaforma: ', NEW.Competenza, '
livello ', NEW.Livello)
    );
END $$

CREATE TRIGGER LogAggiornamentoSkillCurriculum
    AFTER INSERT ON SKILL_CURRICULUM
    FOR EACH ROW
BEGIN
    CALL InserisciLogEvento(
        'AGGIORNAMENTO_SKILL_CURRICULUM',
        NEW.Email_Utente,
        CONCAT('Skill curriculum aggiornata: ', NEW.Competenza, ' livello ',
NEW.Livello)
    );
END $$
```

```
CREATE TRIGGER LogNuovoComponente
    AFTER INSERT ON COMPONENTE
    FOR EACH ROW
BEGIN
    DECLARE v_email_creatore VARCHAR(100);

    SELECT Email_Creatore INTO v_email_creatore
    FROM PROGETTO
    WHERE Nome = NEW.Nome_Progetto;

    CALL InserisciLogEvento(
        'NUOVO_COMPONENTE',
        v_email_creatore,
        CONCAT('Nuovo componente aggiunto al progetto ', NEW.Nome_Progetto, ': ',
NEW.Nome, ' (€', NEW.Prezzo, ')')
    );
END $$

CREATE TRIGGER LogNuovoProfilo
    AFTER INSERT ON PROFILO
    FOR EACH ROW
BEGIN
    DECLARE v_email_creatore VARCHAR(100);

    SELECT Email_Creatore INTO v_email_creatore
    FROM PROGETTO
    WHERE Nome = NEW.Nome_Progetto;

    CALL InserisciLogEvento(
        'NUOVO_PROFILO_RICHIESTO',
        v_email_creatore,
        CONCAT('Nuovo profilo richiesto per il progetto ', NEW.Nome_Progetto, ':
', NEW.Nome)
    );
END $$

CREATE TRIGGER LogNuovaReward
    AFTER INSERT ON REWARD
    FOR EACH ROW
BEGIN
    DECLARE v_email_creatore VARCHAR(100);

    SELECT Email_Creatore INTO v_email_creatore
    FROM PROGETTO
    WHERE Nome = NEW.Nome_Progetto;

    CALL InserisciLogEvento(
        'NUOVA_REWARD',
        v_email_creatore,
        CONCAT('Nuova reward aggiunta al progetto ', NEW.Nome_Progetto, ': ',
NEW.Codice)
    );
END $$

DELIMITER ;

-- DATI DI ESEMPIO

INSERT INTO UTENTE (Email, Nickname, Password, Nome, Cognome, Anno_Di_Nascita,
Luogo_Di_Nascita) VALUES
    ('dalia.barone@email.com','dalia28','password123','Dalia','Barone','2004-02-
20','Termoli'),
    ('sofia.neamtu@email.com','sofia_n','securepass','Sofia','Neamtu','2003-12-
10','Padova'),
```

```
    ('admin@bostarter.com','admin','password123','Admin','System','1990-01-
01','Bologna'),

('dalia.barone@bostarter.com','DaliaAdmin','password123','Dalia','Barone','2004-
02-20','Termoli'),

('sofia.neamtu@bostarter.com','SofiaAdmin','securepass','Sofia','Neamtu','2003-
12-10','Padova');

INSERT INTO AMMINISTRATORE (Email, Codice_Sicurezza) VALUES
    ('admin@bostarter.com','ADMIN2025'),
    ('dalia.barone@bostarter.com','DaliaAdmin2025'),
    ('sofia.neamtu@bostarter.com','SofiaAdmin2025');

INSERT INTO CREATORE (Email, Affidabilita) VALUES
    ('dalia.barone@email.com',0),
    ('sofia.neamtu@email.com',0);

INSERT INTO SKILL (COMPETENZA, LIVELLO) VALUES
    ('AI', 1), ('AI', 2), ('AI', 3), ('AI', 4), ('AI', 5),
    ('Machine Learning', 1), ('Machine Learning', 2), ('Machine Learning', 3),
('Machine Learning', 4), ('Machine Learning', 5),
    ('Web Development', 1), ('Web Development', 2), ('Web Development', 3), ('Web
Development', 4), ('Web Development', 5),
    ('Database Management', 1), ('Database Management', 2), ('Database
Management', 3), ('Database Management', 4), ('Database Management', 5),
    ('Cybersecurity', 1), ('Cybersecurity', 2), ('Cybersecurity', 3),
('Cybersecurity', 4), ('Cybersecurity', 5),
    ('Data Analysis', 1), ('Data Analysis', 2), ('Data Analysis', 3), ('Data
Analysis', 4), ('Data Analysis', 5),
    ('Cloud Computing', 1), ('Cloud Computing', 2), ('Cloud Computing', 3),
('Cloud Computing', 4), ('Cloud Computing', 5),
    ('Networking', 1), ('Networking', 2), ('Networking', 3), ('Networking', 4),
('Networking', 5),
    ('Software Engineering', 1), ('Software Engineering', 2), ('Software
Engineering', 3), ('Software Engineering', 4), ('Software Engineering', 5),
    ('Embedded Systems', 1), ('Embedded Systems', 2), ('Embedded Systems', 3),
('Embedded Systems', 4), ('Embedded Systems', 5);

INSERT INTO SKILL_CURRICULUM (Email_Utente, Competenza, Livello) VALUES
    ('dalia.barone@email.com','Web Development',4),
    ('dalia.barone@email.com','Database Management',3),
    ('dalia.barone@email.com','Networking',4),
    ('dalia.barone@email.com','AI',2),
    ('sofia.neamtu@email.com','Data Analysis',3),
    ('sofia.neamtu@email.com','AI',4),
    ('sofia.neamtu@email.com','Web Development',4),
    ('sofia.neamtu@email.com','Software Engineering',3),
    ('sofia.neamtu@email.com','Machine Learning',5);

INSERT INTO PROGETTO (Nome, Descrizione, Data_Inserimento, Stato, Budget,
Data_Limite, Tipo, Email_Creatore) VALUES
('GreenPower Box','Power bank solare per ricarica dispositivi in condizioni off-
grid','2025-06-20','aperto',8000,'2025-07-
31','Hardware','dalia.barone@email.com'),
('SmartGarden','Sistema intelligente di irrigazione basato su sensori e
controllabile via app','2025-06-15','aperto',6500,'2025-07-
01','Hardware','sofia.neamtu@email.com'),
('SafeDrive AI','Assistente vocale per la guida sicura con comandi AI hands-
free','2025-06-02','aperto',9500,'2025-07-
10','Software','dalia.barone@email.com'),
('EcoCharge Station','Stazione pubblica per ricarica bici elettriche alimentata a
energia solare','2025-06-01','aperto',11000,'2025-07-
15','Hardware','sofia.neamtu@email.com'),
('CodeLink','Piattaforma collaborativa per team di sviluppo distribuiti con
```

```
gestione task e revisioni di codice','2025-06-25','aperto',11000,'2025-07-
01','Software','dalia.barone@email.com'),
('PackTrack','Piattaforma open-source per la gestione e tracciamento delle
spedizioni per piccoli e-commerce','2025-06-30','aperto',8900,'2025-07-
05','Software','sofia.neamtu@email.com');

INSERT INTO COMPONENTE (Nome, Descrizione, Prezzo, Quantita, Nome_Progetto)
VALUES
('Modulo Solare 12V', 'Pannello monocristallino ad alta efficienza', 45.00, 6,
'GreenPower Box'),
('Power Bank USB-C', 'Batteria portatile da 20000mAh', 32.00, 5, 'GreenPower
Box'),
('ESP32', 'Microcontrollore WiFi+Bluetooth', 8.50, 4, 'SmartGarden'),
('Sensore Umidità', 'Sensore per terreno capacitivo', 3.50, 10, 'SmartGarden'),
('Valvola Irrigazione', 'Valvola elettrica 12V', 11.00, 6, 'SmartGarden'),
('Stazione Ricarica', 'Struttura con connettori bici e pannelli', 120.00, 2,
'EcoCharge Station'),
('Batteria Solare', 'Batteria al litio per accumulo energia', 90.00, 2,
'EcoCharge Station');

INSERT INTO PROFILO (ID, Nome, Nome_Progetto) VALUES
(1, 'AI Engineer', 'SafeDrive AI'),
(2, 'Mobile Dev', 'SafeDrive AI'),
(3, 'Frontend Dev', 'CodeLink'),
(4, 'Backend Dev', 'CodeLink'),
(5, 'Full Stack Dev', 'PackTrack'),
(6, 'Logistics Analyst', 'PackTrack');

INSERT INTO SKILL_RICHIESTA (ID_Profilo, Competenza, Livello) VALUES
(1, 'AI', 4),
(1, 'Machine Learning', 3),
(2, 'Web Development', 4),
(2, 'Software Engineering', 3),
(3, 'Web Development', 4),
(3, 'Database Management', 3),
(4, 'Software Engineering', 4),
(4, 'Database Management', 4),
(5, 'Web Development', 4),
(5, 'Networking', 3),
(6, 'Data Analysis', 3),
(6, 'Cloud Computing', 3);

INSERT INTO FOTO (Percorso, Nome_Progetto) VALUES
('GreenPower.jpg', 'GreenPower Box'),
('SmartGarden.jpg', 'SmartGarden'),
('EcoStation.jpg', 'EcoCharge Station'),
('SafeDrive.jpg', 'SafeDrive AI'),
('CodeLink.jpg', 'CodeLink'),
('PackTrack.jpg', 'PackTrack');

INSERT INTO REWARD (Codice, Descrizione, Foto, Nome_Progetto) VALUES
('GP_REW1', 'Sticker e ringraziamento social', '/gp_sticker.jpg', 'GreenPower
Box'),
('SG_REW1', 'Guida digitale al giardinaggio smart', '/sg_guide.jpg',
'SmartGarden'),
('EC_REW1', 'Menzione sulla colonnina pubblica', '/ecostation_reward.jpg',
'EcoCharge Station'),
('SD_REW1', 'Accesso beta all''app', '/safedrive_beta.jpg', 'SafeDrive AI'),
('CL_REW1', 'Abbonamento pro gratuito', '/codelink_reward.jpg', 'CodeLink'),
('PT_REW1', 'Inserimento nella documentazione open-source',
'/packtrack_reward.jpg', 'PackTrack');

INSERT INTO FINANZIAMENTO (Data, Importo, Email_Utente, Codice_Reward,
Nome_Progetto) VALUES
('2025-07-01', 120.00, 'dalia.barone@email.com', 'GP_REW1', 'GreenPower Box'),
```

```
('2025-07-02', 150.00, 'sofia.neamtu@email.com', 'SG_REW1', 'SmartGarden'),
('2025-07-03', 180.00, 'dalia.barone@email.com', 'EC_REW1', 'EcoCharge Station'),
('2025-07-05', 200.00, 'sofia.neamtu@email.com', 'SD_REW1', 'SafeDrive AI'),
('2025-07-06', 250.00, 'dalia.barone@email.com', 'CL_REW1', 'CodeLink'),
('2025-07-07', 220.00, 'sofia.neamtu@email.com', 'PT_REW1', 'PackTrack');

INSERT INTO CANDIDATURA (Esito, Email_Utente, ID_Profilo) VALUES
(NULL, 'sofia.neamtu@email.com', 1),
(NULL, 'dalia.barone@email.com', 5);
```