Progetto “BOSTARTER”

Relazione del progetto – Corso di Basi di Dati
Anno Accademico 2024-2025
Corso di Laurea in Informatica per il Management

Docente del corso – Prof. Marco Di Felice
Tutor del corso – Dr. Leonardo Ciabattini

Autori
Francesco Maria Fuligni – Matricola: 0001068987
Roberto Zanolli – Matricola: 0001070505

Sommario

1. Raccolta e analisi dei requisiti

Specifica della piattaforma

La piattaforma **BOSTARTER** gestisce i dati degli utenti registrati. Ogni utente dispone di indirizzo email (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: ‹competenza, livello›, dove la competenza è una stringa ed il livello è un numero tra 0 e 5 (es. ‹AI, 3›). 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*) 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 e 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 e 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 e 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 ‹competenza, livello›, 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 e 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 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 e 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.

Operazioni sui dati

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 a un progetto.
- Inserimento di una candidatura per un profilo richiesto per la realizzazione di un progetto software.

Operazioni che riguardano SOLO gli amministratori:

- Inserimento di una nuova stringa nella lista delle competenze.
- Autenticazione con richiesta di username, password e 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 a un commento.
- Inserimento di un profilo, solo per la realizzazione di un progetto software.
- Accettazione o rifiuto di una candidatura.

Statistiche (visibili da tutti gli utenti):

- Visualizzazione della classifica degli utenti creatori, in base al loro valore di affidabilità (mostrando solo il nickname dei primi 3 utenti).
- Visualizzazione dei progetti aperti più vicini al completamento (minore differenza tra budget richiesto e totale finanziamenti ricevuti). Mostrare solo i primi 3 progetti.
- Visualizzazione della classifica degli utenti ordinati per totale finanziamenti erogati (mostrando solo i nickname dei primi 3 utenti).

Tavola dei volumi

Volumi per la valutazione della ridondanza concettuale.

Concetto	Tipo	Volume
Creatore	E	5
Progetto	E	10 (2 per Creatore)
Finanziamento	E	30 (3 per Progetto)

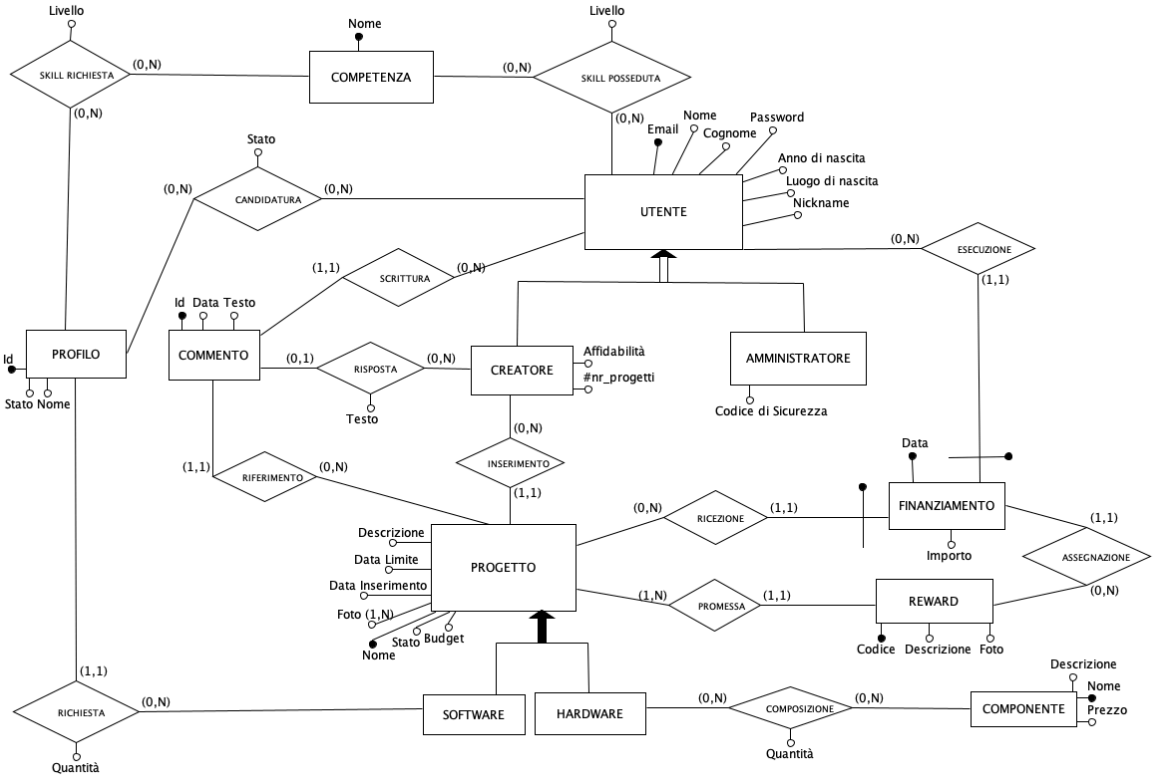
Tavola delle operazioni

Volumi per la valutazione della ridondanza concettuale.

Operazione	Tipo	Frequenza
1. Aggiungere un Progetto a un Creatore esistente	Interattiva	1 volta/mese
2. Visualizzare tutti i Progetti e tutti i Finanziamenti	Batch	1 volta/mese
3. Contare il numero di progetti associati ad un Creatore	Batch	3 volte/mese

2. Progettazione concettuale

Diagramma E-R



Dizionario delle entità

Entità	Descrizione	Attributi	Identificatore
Competenza	Competenze che compongono le skills	Nome	Nome
Utente	Generico utente del sistema	Email, Password, Nickname, Nome, Cognome, Anno nascita, Luogo nascita	Email
Creatore	Utente con possibilità di creare progetti	Affidabilità, #nr_progetti	Email

Amministratore	Utente amministratore del sistema	Codice sicurezza	Email
Progetto	Progetto inserito nel sistema	Nome, Descrizione, Budget, Data inserimento, Data limite, Stato, Foto	Nome
Hardware	Progetto di tipo hardware	-	Nome
Software	Progetto di tipo software	-	Nome
Finanziamento	Finanziamento di un utente a un progetto	Data, Importo	Data, Nome progetto, Email utente
Reward	Reward di un progetto per un finanziamento	Codice, Foto, Descrizione	Codice
Componente	Componente di un progetto hardware	Nome, Descrizione, Prezzo	Nome
Profilo	Profilo di skills ricercato per un progetto software	Id, Nome	Id
Commento	Commento di un utente su un progetto	Id, Data, Testo	Id

Dizionario delle relazioni

Relazione	Descrizione	Componenti	Attributi
Skill posseduta	Associa un utente alle competenze possedute (con un certo livello)	Utente, Competenza	Livello
Skill richiesta	Associa un profilo	Competenza,	Livello

	alle competenze richieste (con un certo livello)	Profilo	
Candidatura	Associa un utente al profilo cui si candida	Utente, Profilo	-
Esecuzione	Associa un utente al finanziamento da lui eseguito per un progetto	Utente, Finanziamento	-
Scrittura	Associa un utente al commento da lui scritto	Utente, Commento	-
Risposta	Associa un commento al creatore che risponde al commento	Commento, Creatore	Testo
Riferimento	Associa un commento al progetto cui si riferisce	Commento, Progetto	-
Inserimento	Associa un utente creatore ad un progetto da lui inserito	Creatore, Progetto	-
Ricezione	Associa un progetto ai finanziamenti ricevuti	Progetto, Finanziamento	-
Promessa	Associa un progetto alle reward da esso promesse per i finanziamenti	Progetto, Reward	-
Assegnazione	Associa ad un finanziamento la reward scelta	Reward, Finanziamento	-
Richiesta	Associa un	Software, Profilo	Quantità

	progetto software ai profili che richiede (in una certa quantità)		
Composizione	Associa un progetto hardware ai componenti (in una certa quantità)	Hardware, Componente	Quantità

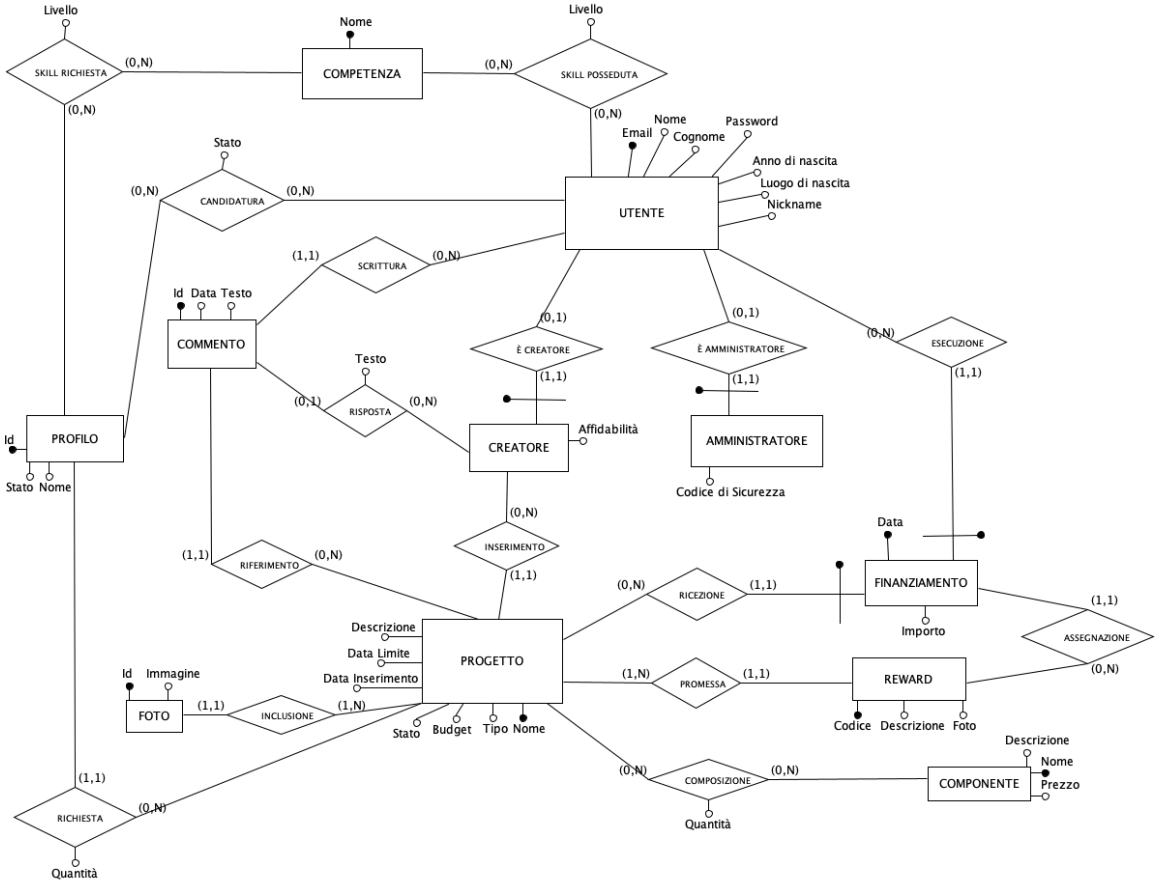
Tabella delle business rules

Regole di vincolo
1. L'attributo *Livello* è un valore intero compreso tra 0 e 5.
2. Solo gli utenti amministratori possono popolare la lista delle competenze.
3. Solo gli utenti creatori possono inserire progetti.
4. L'attributo *Stato* di un progetto è un campo enum (aperto/chiuso).
5. L'attributo *Stato* di un profilo è un campo enum (disponibile/occupato)
6. L'attributo *Quantità* è un valore intero positivo (> 0).
7. Un progetto chiuso non accetta finanziamenti.
8. Un utente può inserire una candidatura per un profilo solo se, per ogni skill richiesta dal profilo, egli dispone di un livello superiore o uguale a quello richiesto.
9. L'attributo *Stato* di una candidatura è un campo enum (accettata/rifiutata/in attesa)
10. Solo l'utente creatore del progetto può modificare lo stato delle candidature.
11. Solo l'utente creatore del progetto può inserire risposte a commenti sul progetto.
12. Non possono essere presenti più profili con i medesimi nome e skill per lo stesso progetto.
13. Un utente creatore può inserire una risposta solamente su commenti di progetti che di cui egli è il creatore
Regole di derivazione

1. L'attributo *Stato* di un progetto diventa uguale a chiuso quando la somma totale degli importi dei finanziamenti per quel progetto supera il budget o viene superata la data limite.

3. Progettazione logica

Diagramma E-R ristrutturato



- La **generalizzazione parziale su Utente** è stata eliminata introducendo una relazione uno a uno tra Utente e le sue specializzazioni Amministratore e Creatore
- La **generalizzazione totale su Progetto** è stata eliminata accorpando le specializzazioni Hardware e Software in Progetto, con l'aggiunta dell'attributo *Tipo*
- L'**attributo multivalore *Foto*** dell'entità Progetto è stato sostituito da una relazione uno a molti con l'entità Foto
- La **ridondanza concettuale #nr_progetti** è stata eliminata *(analisi di seguito)*

Vincoli ulteriori introdotti:

1. Un'istanza dell'entità Utente non può partecipare contemporaneamente alle relazioni È Creatore ed È Amministratore
2. L'attributo Tipo dell'entità Progetto è un campo enum (hardware/software)
3. Un Progetto di tipo software non ha componenti associati
4. Un Progetto di tipo hardware non ha profili associati
5. Il numero di progetti per un Creatore si deriva contando il numero di Progetti inseriti dal Creatore

Analisi della ridondanza concettuale #nr_progetti

Formula per l'analisi: $c(O_T) = f(O_T) \times w_T \times (\alpha \times NC_{write} + NC_{read})$
Coefficienti per l'analisi: $w_{Interattiva} = 1\ ,\quad w_{Batch} = 0.5\ ,\quad \alpha = 2$

Costo operazioni con Ridondanza:

- **Operazione 1** (1 volta/mese, interattiva)
  Si considerano 1 accesso in scrittura alla relazione Inserimento, 1 accesso in scrittura all'entità Progetto, 1 accesso in scrittura per l'incremento dell'attributo #nr_progetti.
  Nessun accesso in lettura.

  $$c(O_1)_R = 1 \times 1 \times (2 \times 3 + 0) = 6$$

- **Operazione 2** (1 volta/mese, batch)
  Si considerano 10 accessi in lettura all'entità Progetto e 30 accessi in lettura all'entità Finanziamento.
  Nessun accesso in scrittura.

  $$c(O_2)_R = 1 \times 0.5 \times (2 \times 0 + 40) = 20$$

- **Operazione 3** (3 volte/mese, batch)
  Si considera 1 accesso in lettura all'attributo #nr_progetti.
  Nessun accesso in scrittura.

  $$c(O_3)_R = 3 \times 0.5 \times (2 \times 0 + 1) = 1.5$$

COSTO TOTALE CON RIDONDANZA: $c(R) = 27.5$

Costo operazioni Senza Ridondanza:

- **Operazione 1** (1 volta/mese, interattiva)
  Si considerano 1 accesso in scrittura alla relazione Inserimento e 1 accesso in scrittura all'entità Progetto.

Nessun accesso in lettura.

$$c(O_1)_{SR} = 1 \times 1 \times (2 \times 2 + 0) = 4$$

- **Operazione 2** (1 volta/mese, batch)
  Si considerano 10 accessi in lettura all'entità Progetto e 30 accessi in lettura all'entità Finanziamento.
  Nessun accesso in scrittura.

  $$c(O_2)_{SR} = 1 \times 0.5 \times (2 \times 0 + 40) = 20$$

- **Operazione 3** (3 volte/mese, batch)
  Si considerano 2 accessi in lettura alla relazione Inserimento.
  Nessun accesso in scrittura.

  $$c(O_3)_{SR} = 3 \times 0.5 \times (2 \times 0 + 2) = 3$$

COSTO TOTALE SENZA RIDONDANZA: $c(SR) = 27$

Valutazione della ridondanza

- **Rapporto di speed-up**

  $$\text{speed-up} = \frac{c(SR)}{c(R)} = \frac{27}{27.5} = 0.982$$

  Sebbene il valore ottenuto sia molto vicino a 1, indica comunque un miglioramento in termini di efficienza, evidenziando che l'eliminazione della ridondanza risulta più conveniente.

- **Overhead di memoria**
  Considerando il valore #nr_progetti come un intero memorizzato utilizzando 4 Byte, l'overhead di memoria dovuto alla ridondanza è stimato come:
  $4 \times 5 = 20 \text{ Byte}$
  Anche se l'overhead introdotto è relativamente piccolo (sulla base della tabella dei volumi corrente), la sua presenza conferma che rimuovere la ridondanza è la scelta più efficiente.

La ridondanza è stata **eliminata**.

Traduzione nel modello logico relazionale

Tabelle con vincoli di chiave

COMPETENZA (Nome)
UTENTE (Email, Password, Nickname, Nome, Cognome, AnnoNascita, LuogoNascita)
CREATORE (EmailUtente, Affidabilità)
AMMINISTRATORE (EmailUtente, CodiceSicurezza)
PROGETTO (Nome, Descrizione, Budget, DataInserimento, DataLimite, Stato, Tipo, EmailCreatore)
FOTO (Id, Immagine, NomeProgetto)
REWARD (Codice, Immagine, Descrizione, NomeProgetto)
FINANZIAMENTO (Data, NomeProgetto, EmailUtente, Importo, CodiceReward)
COMPONENTE (Nome, Descrizione, Prezzo)
PROFILO (Id, Nome, NomeProgetto)
COMMENTO (Id, Data, Testo, EmailUtente, NomeProgetto)
SKILL POSSEDUTA (EmailUtente, NomeCompetenza, Livello)
SKILL RICHIESTA (IdProfilo, NomeCompetenza, Livello)
CANDIDATURA (EmailUtente, IdProfilo, Stato)
RISPOSTA (IdCommento, Testo, EmailCreatore)
COMPOSIZIONE (NomeProgetto, NomeComponente, Quantità)

Vincoli inter-relazionali

CREATORE.EmailUtente → **UTENTE**.Email
AMMINISTRATORE.EmailUtente → **UTENTE**.Email
PROGETTO.EmailCreatore → **CREATORE**.EmailUtente
FOTO.NomeProgetto → **PROGETTO**.Nome
FINANZIAMENTO.NomeProgetto → **PROGETTO**.Nome
FINANZIAMENTO.EmailUtente → **UTENTE**.Email
FINANZIAMENTO.CodiceReward → **REWARD**.Codice
REWARD.NomeProgetto → **PROGETTO**.Nome
PROFILO.NomeProgetto → **PROGETTO**.Nome
COMMENTO.EmailUtente → **UTENTE**.Email
COMMENTO.NomeProgetto → **PROGETTO**.Nome

SKILL POSSEDUTA.EmailUtente → **UTENTE**.Email
SKILL POSSEDUTA.NomeCompetenza → **COMPETENZA**.Nome
SKILL RICHIESTA.IdProfilo → **PROFILO**.Id
SKILL RICHIESTA.NomeCompetenza → **COMPETENZA**.Nome
CANDIDATURA.EmailUtente → **UTENTE**.Email
CANDIDATURA.IdProfilo → **PROFILO**.Id
RISPOSTA.IdCommento → **COMMENTO**.Id
RISPOSTA.EmailCreatore → **CREATORE**.EmailUtente
COMPOSIZIONE.NomeProgetto → **PROGETTO**.Nome
COMPOSIZIONE.NomeComponente → **COMPONENTE**.Nome

4. Normalizzazione

Tutte le tabelle presentano una chiave primaria minimale, ovvero un insieme minimo di attributi che identifica univocamente ogni tupla. Inoltre, in ciascuna tabella, le dipendenze funzionali presenti sono determinate dalla chiave primaria: al ripetersi dei valori della chiave, si ripetono anche i valori degli altri attributi.
Secondo la definizione di **Forma Normale di Boyce-Codd** (FNBC), uno schema R(X) è in FNBC se, per ogni dipendenza funzionale non banale Y → Z, l'insieme Y è una superchiave di R. Poiché in tutte le tabelle ogni dipendenza funzionale ha come determinante la chiave primaria, risulta verificata la condizione necessaria per la FNBC.
Pertanto, l'intero schema è già in FNBC e non presenta anomalie di aggiornamento, cancellazione o inserimento dovute a ridondanze fisiche. Di conseguenza, **non è necessaria alcuna ulteriore normalizzazione.**

5. Descrizione delle funzionalità dell'applicazione Web

L'applicazione web mostra nella sua schermata home una raccolta di alcune statistiche relative a progetti ed utenti della piattaforma, insieme alla possibilità (in alto a destra nella barra di navigazione) di accedere o registrarsi. Per vedere il dettaglio dei progetti mostrati nella home è necessario eseguire l'accesso. È possibile tornare alla home in qualsiasi momento, cliccando sul logo "BoStarter" (in alto a sinistra nella barra di navigazione).
In fase di registrazione è possibile decidere di registrarsi come utente Amministratore, utente Creatore o utente Normale. Dopo essersi registrato, l'utente deve effettuare l'accesso con le sue credenziali.
Per utenti creatori e standard il processo di registrazione e login è il medesimo, per quanto riguarda gli utenti amministratori invece, viene generato un codice in fase di registrazione che occorre inserire in fase di login (è consigliato annotarselo).

Una volta eseguito l'accesso, la schermata mostrerà una dashboard con le azioni possibili (in base alla tipologia di utente), un elenco di tutti i progetti della piattaforma e, nel caso di utente Creatore, un elenco di tutti i progetti creati dall'utente.
Cliccando su un progetto, l'utente può vederne i dettagli ed eseguire su di esso le operazioni di finanziamento (una volta al giorno), commento, i profili disponibili per la candidatura e, per quanto riguarda l'utente creatore del progetto, gestire le risposte ai commenti, inserire i profili richiesti per il progetto e accettare/rifiutare le candidature a un profilo.
Solo l'utente creatore del progetto o il proprietario di un determinato commento possono effettuarne la rimozione attraverso l'apposito bottone.

Attraverso la dashboard un utente creatore può creare un nuovo progetto compilando tutti i campi obbligatori, creando le reward annesse e scegliendo la tipologia di progetto tra HARDWARE E SOFTWARE. Nel caso di tipologia HARDWARE, l'utente potrà usare una sezione per aggiungere i componenti al progetto con tanto di quantità o di creare un nuovo componente HARDWARE che diverrà utilizzabile da tutti nella piattaforma.
Per quanto riguarda i progetti SOFTWARE, come spiegato in precedenza è possibile creare profili richiesti specifici per un progetto dalla apposita pagina di dettaglio.

Dalla dashboard di un utente amministratore è possibile aggiungere nuove competenze alla piattaforma inserendo come conferma il codice amministratore.

Ogni utente, dalla sua dashboard può caricare le proprie competenze selezionando un livello per ognuna di esse.

6. Appendice: Codice SQL dello schema della base di dati

DDL

```sql
DROP DATABASE IF EXISTS bostarter_db;

CREATE DATABASE IF NOT EXISTS bostarter_db;
USE bostarter_db;

CREATE TABLE IF NOT EXISTS UTENTE (
   email VARCHAR(32) PRIMARY KEY,
   password CHAR(64) NOT NULL,
   nome VARCHAR(32) NOT NULL,
   cognome VARCHAR(32) NOT NULL,
   nickname VARCHAR(32) UNIQUE NOT NULL,
   luogo_nascita VARCHAR(32) NOT NULL,
   anno_nascita INT NOT NULL
) ENGINE = 'InnoDB';


CREATE TABLE IF NOT EXISTS COMPETENZA (
   nome VARCHAR(32) PRIMARY KEY
) ENGINE = 'InnoDB';


CREATE TABLE IF NOT EXISTS UTENTE_CREATORE (
 email_utente VARCHAR(32) PRIMARY KEY REFERENCES UTENTE(email) ,
  affidabilita INT DEFAULT 0 CHECK (affidabilita >= 0 AND affidabilita
<= 100)     -- valore percentuale % (intero)
  --  nr_progetti  INT  DEFAULT  0  CHECK  (nr_progetti  >=  0)         --
RIDONDANZA CONCETTUALE RIMOSSA
) ENGINE = 'InnoDB';


CREATE TABLE IF NOT EXISTS UTENTE_AMMINISTRATORE (
```

```
 email_utente VARCHAR(32) PRIMARY KEY REFERENCES UTENTE(email) ,
 codice_sicurezza CHAR(64) UNIQUE NOT NULL
) ENGINE = 'InnoDB';


CREATE TABLE IF NOT EXISTS PROGETTO (
   nome VARCHAR(32) PRIMARY KEY,
   descrizione VARCHAR(255) NOT NULL,
   budget DECIMAL(16,2) NOT NULL,
   data_inserimento DATE NOT NULL,
   data_limite DATE NOT NULL,
   stato ENUM ('APERTO', 'CHIUSO') DEFAULT 'APERTO',
   tipo ENUM ('SOFTWARE', 'HARDWARE') NOT NULL,
         email_utente_creatore   VARCHAR(32)   NOT   NULL   REFERENCES
UTENTE_CREATORE(email_utente),
   somma_raccolta DECIMAL(16,2) DEFAULT 0
) ENGINE = 'InnoDB';


CREATE TABLE IF NOT EXISTS FOTO (
   id INT AUTO_INCREMENT PRIMARY KEY,
   immagine LONGBLOB NOT NULL,
   nome_progetto VARCHAR(32) NOT NULL REFERENCES PROGETTO(nome)
) ENGINE = 'InnoDB';


CREATE TABLE IF NOT EXISTS REWARD (
   codice INT AUTO_INCREMENT PRIMARY KEY,
   immagine LONGBLOB NOT NULL,
   descrizione VARCHAR(255) NOT NULL,
   nome_progetto VARCHAR(32) NOT NULL REFERENCES PROGETTO(nome)
) ENGINE = 'InnoDB';


CREATE TABLE IF NOT EXISTS FINANZIAMENTO (
   data DATE NOT NULL,
   nome_progetto VARCHAR(32) NOT NULL REFERENCES PROGETTO(nome),
   email_utente VARCHAR(32) NOT NULL REFERENCES UTENTE(email),
   importo DECIMAL(16,2) NOT NULL CHECK (importo > 0),
```

```
    codice_reward INT REFERENCES REWARD(codice),
    PRIMARY KEY (data, nome_progetto, email_utente)
) ENGINE = 'InnoDB';


CREATE TABLE IF NOT EXISTS COMPONENTE (
    nome VARCHAR(32) PRIMARY KEY,
    descrizione VARCHAR(255) NOT NULL,
    prezzo DECIMAL(16,2) NOT NULL
) ENGINE = 'InnoDB';


CREATE TABLE IF NOT EXISTS PROFILO (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(32) NOT NULL,
    nome_progetto VARCHAR(32) NOT NULL REFERENCES PROGETTO(nome),
    stato ENUM ('DISPONIBILE', 'OCCUPATO') DEFAULT 'DISPONIBILE'
) ENGINE = 'InnoDB';


CREATE TABLE IF NOT EXISTS COMMENTO (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome_progetto VARCHAR(32) NOT NULL REFERENCES PROGETTO(nome),
    email_utente VARCHAR(32) NOT NULL REFERENCES UTENTE(email),
    testo VARCHAR(255) NOT NULL,
    data DATE NOT NULL
) ENGINE = 'InnoDB';


CREATE TABLE IF NOT EXISTS SKILL_POSSEDUTA (
    email_utente VARCHAR(32) NOT NULL REFERENCES UTENTE(email),
    nome_competenza VARCHAR(32) NOT NULL REFERENCES COMPETENZA(nome),
    livello INT NOT NULL CHECK (livello >= 0 AND livello <= 5),
    PRIMARY KEY (email_utente, nome_competenza)
) ENGINE = 'InnoDB';


CREATE TABLE IF NOT EXISTS SKILL_RICHIESTA (
    id_profilo INT NOT NULL REFERENCES PROFILO(id),
```

```
    nome_competenza VARCHAR(32) NOT NULL REFERENCES COMPETENZA(nome),
    livello INT NOT NULL CHECK (livello >= 0 AND livello <= 5),
    PRIMARY KEY (id_profilo, nome_competenza)
) ENGINE = 'InnoDB';


CREATE TABLE IF NOT EXISTS CANDIDATURA (
    email_utente VARCHAR(32) NOT NULL REFERENCES UTENTE(email),
    id_profilo INT NOT NULL REFERENCES PROFILO(id),
    stato ENUM ('ACCETTATA', 'ATTESA', 'RIFIUTATA') DEFAULT 'ATTESA',
    PRIMARY KEY (email_utente, id_profilo)
) ENGINE = 'InnoDB';


CREATE TABLE IF NOT EXISTS RISPOSTA (
    id_commento INT PRIMARY KEY,
    email_creatore VARCHAR(32) NOT NULL,
    testo VARCHAR(255) NOT NULL,
    FOREIGN KEY (id_commento) REFERENCES COMMENTO(id) ON DELETE CASCADE,
    FOREIGN KEY (email_creatore) REFERENCES UTENTE(email)
) ENGINE = 'InnoDB';


CREATE TABLE IF NOT EXISTS COMPOSIZIONE (
    nome_progetto VARCHAR(32) NOT NULL REFERENCES PROGETTO(nome),
    nome_componente VARCHAR(32) NOT NULL REFERENCES COMPONENTE(nome),
    quantita INT NOT NULL CHECK (quantita >= 0),
    PRIMARY KEY (nome_progetto, nome_componente)
) ENGINE = 'InnoDB';
```

Eventi

```
USE bostarter_db;
```

```sql
SET GLOBAL event_scheduler = ON;

-- Evento per chiudere i progetti scaduti (eseguito ogni giorno)
DROP EVENT IF EXISTS chiudi_progetti_scaduti;

DELIMITER //
CREATE EVENT chiudi_progetti_scaduti
ON SCHEDULE EVERY 1 DAY
STARTS CURRENT_TIMESTAMP
DO
BEGIN
   UPDATE PROGETTO
   SET stato = 'CHIUSO'
   WHERE stato = 'APERTO'
   AND data_limite < CURDATE();
END //
DELIMITER ;
```

Stored procedures

```sql
USE bostarter_db;



-- PROCEDURE DI CONTROLLO ------------------------------------------


-- Procedura per verificare se un utente è creatore
DROP PROCEDURE IF EXISTS verifica_creatore;

DELIMITER //
CREATE PROCEDURE verifica_creatore(
   IN in_email VARCHAR(32),
   OUT esito BOOLEAN
```

```
)
BEGIN
    SELECT EXISTS(SELECT 1 FROM UTENTE_CREATORE WHERE email_utente =
in_email) INTO esito;
END //
DELIMITER ;


-- Procedura per verificare se un utente è amministratore
DROP PROCEDURE IF EXISTS verifica_amministratore;

DELIMITER //
CREATE PROCEDURE verifica_amministratore(
   IN in_email VARCHAR(32),
   OUT esito BOOLEAN
)
BEGIN
    SELECT EXISTS(SELECT 1 FROM UTENTE_AMMINISTRATORE WHERE email_utente
= in_email) INTO esito;
END //
DELIMITER ;


-- Procedura per verificare se un utente è amministratore e ha un
codice di sicurezza valido
DROP PROCEDURE IF EXISTS verifica_amministratore_con_codice;

DELIMITER //
CREATE PROCEDURE verifica_amministratore_con_codice(
   IN in_email VARCHAR(32),
   IN in_codice_sicurezza CHAR(64),
   OUT esito BOOLEAN
)
BEGIN
    SELECT EXISTS(SELECT 1 FROM UTENTE_AMMINISTRATORE WHERE email_utente
= in_email AND codice_sicurezza = in_codice_sicurezza) INTO esito;
END //
DELIMITER ;
```

```
-- Procedura per verificare se un progetto è aperto
DROP PROCEDURE IF EXISTS verifica_progetto_aperto;

DELIMITER //
CREATE PROCEDURE verifica_progetto_aperto(
   IN in_nome_progetto VARCHAR(32),
   OUT esito BOOLEAN
)
BEGIN
    SELECT EXISTS(SELECT 1 FROM PROGETTO WHERE nome = in_nome_progetto
AND stato = 'APERTO') INTO esito;
END //
DELIMITER ;


-- Procedura per verificare se un utente è creatore di un progetto
specifico
DROP PROCEDURE IF EXISTS verifica_creatore_progetto;

DELIMITER //
CREATE PROCEDURE verifica_creatore_progetto(
   IN in_nome_progetto VARCHAR(32),
   IN in_email_creatore VARCHAR(32),
   OUT esito BOOLEAN
)
BEGIN
    SELECT EXISTS(SELECT 1 FROM PROGETTO WHERE nome = in_nome_progetto
AND email_utente_creatore = in_email_creatore) INTO esito;
END //
DELIMITER ;


-- Procedura per verificare la tipologia di un progetto
DROP PROCEDURE IF EXISTS verifica_tipo_progetto;

DELIMITER //
CREATE PROCEDURE verifica_tipo_progetto(
   IN in_nome_progetto VARCHAR(32),
```

```
   IN in_tipo ENUM('SOFTWARE', 'HARDWARE'),
   OUT esito BOOLEAN
)
BEGIN
    SELECT EXISTS(SELECT 1 FROM PROGETTO WHERE nome = in_nome_progetto
AND tipo = in_tipo) INTO esito;
END //
DELIMITER ;




--                                                                  TUTTI
-------------------------------------------------------------------------


-- Procedura per l'autenticazione di un utente (utente normale o
creatore)
DROP PROCEDURE IF EXISTS autenticazione_utente;

DELIMITER //
CREATE PROCEDURE autenticazione_utente (
   IN in_email VARCHAR(32),
   IN in_password CHAR(64),
   OUT esito BOOLEAN
)
BEGIN
     SELECT EXISTS(SELECT 1 FROM UTENTE WHERE email = in_email AND
password = in_password) INTO esito;
END //
DELIMITER ;


-- Procedura per la registrazione di un utente
DROP PROCEDURE IF EXISTS registrazione_utente;

DELIMITER //
CREATE PROCEDURE registrazione_utente (
```

```
    IN in_email VARCHAR(32),
    IN in_password CHAR(64),
    IN in_nome VARCHAR(32),
    IN in_cognome VARCHAR(32),
    IN in_nickname VARCHAR(32),
    IN in_luogo_nascita VARCHAR(32),
    IN in_anno_nascita INT
)
BEGIN
     INSERT INTO UTENTE (email, password, nome, cognome, nickname,
luogo_nascita, anno_nascita)
     VALUES (in_email, in_password, in_nome, in_cognome, in_nickname,
in_luogo_nascita, in_anno_nascita);
END //
DELIMITER ;


-- Procedura per la registrazione di un creatore
DROP PROCEDURE IF EXISTS registrazione_creatore;

DELIMITER //
CREATE PROCEDURE registrazione_creatore (
    IN in_email VARCHAR(32)
)
BEGIN
    INSERT INTO UTENTE_CREATORE (email_utente)
    VALUES (in_email);
END //
DELIMITER ;


-- Procedura per l'inserimento di un amministratore
DROP PROCEDURE IF EXISTS registrazione_amministratore;

DELIMITER //
CREATE PROCEDURE registrazione_amministratore (
    IN in_email VARCHAR(32),
    IN in_codice_sicurezza CHAR(64)
)
```

```
BEGIN
   INSERT INTO UTENTE_AMMINISTRATORE (email_utente, codice_sicurezza)
   VALUES (in_email, in_codice_sicurezza);
END //
DELIMITER ;


-- Procedura per l'inserimento di una skill nel curriculum
DROP PROCEDURE IF EXISTS aggiungi_skill;

DELIMITER //
CREATE PROCEDURE aggiungi_skill(
   IN in_email VARCHAR(32),
   IN in_competenza VARCHAR(32),
   IN in_livello INT
)
BEGIN
   INSERT INTO SKILL_POSSEDUTA (email_utente, nome_competenza, livello)
   VALUES (in_email, in_competenza, in_livello);
END //
DELIMITER ;


-- Procedura per il finanziamento di un progetto
DROP PROCEDURE IF EXISTS finanzia_progetto;

DELIMITER //
CREATE PROCEDURE finanzia_progetto(
   IN in_email_utente VARCHAR(32),
   IN in_nome_progetto VARCHAR(32),
   IN in_importo DECIMAL(16,2),
   OUT is_progetto_aperto BOOLEAN
)
BEGIN
   CALL verifica_progetto_aperto(in_nome_progetto, is_progetto_aperto);

   IF is_progetto_aperto THEN
         INSERT INTO FINANZIAMENTO (data, nome_progetto, email_utente,
importo)
```

```
                VALUES  (CURDATE(),  in_nome_progetto,  in_email_utente,
in_importo);
   END IF;
END //
DELIMITER ;


-- Procedura per la scelta della reward
DROP PROCEDURE IF EXISTS scegli_reward;

DELIMITER //
CREATE PROCEDURE scegli_reward(
   IN in_email_utente VARCHAR(32),
   IN in_nome_progetto VARCHAR(32),
   IN in_codice_reward INT
)
BEGIN
   UPDATE FINANZIAMENTO
   SET codice_reward = in_codice_reward
   WHERE email_utente = in_email_utente
   AND nome_progetto = in_nome_progetto
   AND codice_reward IS NULL;
END //
DELIMITER ;


-- Procedura per l'inserimento di un commento
DROP PROCEDURE IF EXISTS inserisci_commento;

DELIMITER //
CREATE PROCEDURE inserisci_commento(
   IN in_nome_progetto VARCHAR(32),
   IN in_email_utente VARCHAR(32),
   IN in_testo VARCHAR(255)
)
BEGIN
   INSERT INTO COMMENTO (nome_progetto, email_utente, testo, data)
   VALUES (in_nome_progetto, in_email_utente, in_testo, CURDATE());
END //
```

```
DELIMITER ;



-- Procedura per l'inserimento di una candidatura
DROP PROCEDURE IF EXISTS inserisci_candidatura;

DELIMITER //
CREATE PROCEDURE inserisci_candidatura(
   IN in_email_utente VARCHAR(32),
   IN in_id_profilo INT,
   OUT esito BOOLEAN
)
BEGIN
   DECLARE nome_progetto VARCHAR(32);

   -- Ricava il nome del progetto associato al profilo
   SELECT nome_progetto INTO nome_progetto
   FROM PROFILO
   WHERE id = in_id_profilo;

      CALL  verifica_creatore_progetto(nome_progetto,  in_email_utente,
esito);

   IF NOT esito THEN
       SELECT NOT EXISTS (
           SELECT *
           FROM SKILL_RICHIESTA sr
           LEFT JOIN SKILL_POSSEDUTA sp
               ON sr.nome_competenza = sp.nome_competenza
               AND sp.email_utente = in_email_utente
               AND sp.livello >= sr.livello
           WHERE sr.id_profilo = in_id_profilo
           AND sp.email_utente IS NULL
       ) INTO esito;

       IF esito THEN
           INSERT INTO CANDIDATURA (email_utente, id_profilo)
           VALUES (in_email_utente, in_id_profilo);
       END IF;
```

```
   END IF;
END //
DELIMITER ;



-- Procedura per verificare se un utente ha già finanziato un progetto
nella data odierna
DROP PROCEDURE IF EXISTS ha_finanziato_oggi;

DELIMITER //
CREATE PROCEDURE ha_finanziato_oggi(
   IN in_nome_progetto VARCHAR(32),
   IN in_email_utente VARCHAR(32),
   OUT esito BOOLEAN
)
BEGIN
   SELECT EXISTS(
       SELECT 1
       FROM FINANZIAMENTO
       WHERE nome_progetto = in_nome_progetto
         AND email_utente = in_email_utente
         AND data = CURDATE()
   ) INTO esito;
END //
DELIMITER ;









--                          SOLO                          AMMINISTRATORI
-------------------------------------------------------------



-- Procedura per l'autenticazione (solo amministratori)
DROP PROCEDURE IF EXISTS autenticazione_amministratore;

DELIMITER //
CREATE PROCEDURE autenticazione_amministratore(
```

```
    IN in_email VARCHAR(32),
    IN in_password CHAR(64),
    IN in_codice_sicurezza CHAR(64),
    OUT esito BOOLEAN
)
BEGIN
     SELECT EXISTS(SELECT 1 FROM UTENTE u JOIN UTENTE_AMMINISTRATORE ua
ON u.email = ua.email_utente WHERE u.email = in_email AND u.password =
in_password AND ua.codice_sicurezza = in_codice_sicurezza) INTO esito;
END //
DELIMITER ;


-- Procedura per l'inserimento di una nuova competenza (solo
amministratori)
DROP PROCEDURE IF EXISTS aggiungi_competenza;

DELIMITER //
CREATE PROCEDURE aggiungi_competenza(
    IN in_competenza VARCHAR(32),
    IN in_email VARCHAR(32),
    IN in_codice_sicurezza CHAR(64),
    OUT is_amministratore BOOLEAN
)
BEGIN
                  CALL      verifica_amministratore_con_codice(in_email,
in_codice_sicurezza, is_amministratore);

    IF is_amministratore THEN
        INSERT INTO COMPETENZA (nome)
        VALUES (in_competenza);
    END IF;
END //
DELIMITER ;
```

```
--                              SOLO                                CREATORI
-------------------------------------------------------------------



-- Procedura per l'inserimento di un nuovo progetto (solo creatori)
DROP PROCEDURE IF EXISTS crea_progetto;

DELIMITER //
CREATE PROCEDURE crea_progetto(
   IN in_nome VARCHAR(32),
   IN in_descrizione VARCHAR(255),
   IN in_budget DECIMAL(16,2),
   IN in_data_limite DATE,
   IN in_tipo ENUM ('SOFTWARE', 'HARDWARE'),
   IN in_email_creatore VARCHAR(32),
   OUT esito BOOLEAN
)
BEGIN
   CALL verifica_creatore(in_email_creatore, esito);

   IF esito THEN
       IF CURDATE() > in_data_limite THEN
                   INSERT INTO PROGETTO (nome, descrizione, budget,
data_inserimento, data_limite, stato, tipo, email_utente_creatore)
               VALUES (in_nome, in_descrizione, in_budget, CURDATE(),
in_data_limite, 'CHIUSO', in_tipo, in_email_creatore);
       ELSE
                   INSERT INTO PROGETTO (nome, descrizione, budget,
data_inserimento, data_limite, stato, tipo, email_utente_creatore)
               VALUES (in_nome, in_descrizione, in_budget, CURDATE(),
in_data_limite, 'APERTO', in_tipo, in_email_creatore);
       END IF;
   END IF;
END //
DELIMITER ;



-- Procedura per l'inserimento di una foto per un progetto (solo
creatore del progetto)
```

```
DROP PROCEDURE IF EXISTS inserisci_foto;

DELIMITER //
CREATE PROCEDURE inserisci_foto(
   IN in_immagine LONGBLOB,
   IN in_nome_progetto VARCHAR(32),
   IN in_email_creatore VARCHAR(32),
   OUT esito BOOLEAN
)
BEGIN

   CALL verifica_creatore_progetto(in_nome_progetto, in_email_creatore,
esito);

   IF esito THEN
       INSERT INTO FOTO (immagine, nome_progetto)
       VALUES (in_immagine, in_nome_progetto);
   END IF;
END //
DELIMITER ;


-- Procedura per l'inserimento di una reward (solo creatore del
progetto)
DROP PROCEDURE IF EXISTS inserisci_reward;

DELIMITER //
CREATE PROCEDURE inserisci_reward(
   IN in_immagine LONGBLOB,
   IN in_descrizione VARCHAR(255),
   IN in_nome_progetto VARCHAR(32),
   IN in_email_creatore VARCHAR(32),
   OUT esito BOOLEAN
)
BEGIN
   CALL verifica_creatore_progetto(in_nome_progetto, in_email_creatore,
esito);

   IF esito THEN
```

```
        INSERT INTO REWARD (immagine, descrizione, nome_progetto)
        VALUES (in_immagine, in_descrizione, in_nome_progetto);
    END IF;
END //
DELIMITER ;


-- Procedura per l'inserimento di una risposta ad un commento (solo creatore del progetto)
DROP PROCEDURE IF EXISTS inserisci_risposta;

DELIMITER //
CREATE PROCEDURE inserisci_risposta(
    IN in_id_commento INT,
    IN in_testo VARCHAR(255),
    IN in_email_creatore VARCHAR(32),
    OUT is_creatore_progetto BOOLEAN
)
BEGIN
    CALL verifica_creatore_progetto(
            (SELECT c.nome_progetto FROM COMMENTO c WHERE c.id = in_id_commento),
        in_email_creatore,
        is_creatore_progetto);

    IF is_creatore_progetto THEN
        INSERT INTO RISPOSTA (id_commento, email_creatore, testo)
        VALUES (in_id_commento, in_email_creatore, in_testo);
    END IF;
END //
DELIMITER ;


-- Procedura per l'inserimento di un profilo (solo creatore del progetto SOFTWARE)
DROP PROCEDURE IF EXISTS inserisci_profilo;

DELIMITER //
CREATE PROCEDURE inserisci_profilo(
```

```
    IN in_nome VARCHAR(32),
    IN in_nome_progetto VARCHAR(32),
    IN in_email_creatore VARCHAR(32),
    OUT is_creatore_progetto BOOLEAN
)
BEGIN
    CALL verifica_creatore_progetto(in_nome_progetto, in_email_creatore,
is_creatore_progetto);

    IF is_creatore_progetto THEN
            CALL  verifica_tipo_progetto(in_nome_progetto,  'SOFTWARE',
is_creatore_progetto);
        IF is_creatore_progetto THEN
            INSERT INTO PROFILO (nome, nome_progetto)
            VALUES (in_nome, in_nome_progetto);
        END IF;
    END IF;
END //
DELIMITER ;


-- Procedura per l'inserimento di una skill richiesta (solo creatore
del progetto SOFTWARE)
DROP PROCEDURE IF EXISTS inserisci_skill_richiesta;

DELIMITER //
CREATE PROCEDURE inserisci_skill_richiesta(
    IN in_id_profilo INT,
    IN in_email_creatore VARCHAR(32),
    IN in_competenza VARCHAR(32),
    IN in_livello INT,
    OUT is_creatore_progetto BOOLEAN
)
BEGIN
    DECLARE nome_progetto VARCHAR(32);

    SELECT pr.nome_progetto INTO nome_progetto
    FROM PROFILO pr
    WHERE pr.id = in_id_profilo;
```

```
    CALL  verifica_creatore_progetto(nome_progetto,  in_email_creatore,
is_creatore_progetto);

   IF is_creatore_progetto THEN
              CALL  verifica_tipo_progetto(nome_progetto,  'SOFTWARE',
is_creatore_progetto);
       IF is_creatore_progetto THEN
             INSERT INTO SKILL_RICHIESTA (id_profilo, nome_competenza,
livello)
           VALUES (in_id_profilo, in_competenza, in_livello);
       END IF;
   END IF;
END //
DELIMITER ;


-- Procedura per gestire una candidatura (solo creatore del progetto
SOFTWARE)
DROP PROCEDURE IF EXISTS gestisci_candidatura;

DELIMITER //
CREATE PROCEDURE gestisci_candidatura(
   IN in_email_candidato VARCHAR(32),
   IN in_id_profilo INT,
   IN in_email_creatore VARCHAR(32),
   IN in_stato ENUM ('ACCETTATA', 'RIFIUTATA'),
   OUT is_creatore_progetto BOOLEAN
)
BEGIN
   DECLARE nome_progetto VARCHAR(32);

   SELECT pr.nome_progetto INTO nome_progetto
   FROM PROFILO pr
   WHERE pr.id = in_id_profilo;

    CALL  verifica_creatore_progetto(nome_progetto,  in_email_creatore,
is_creatore_progetto);
```

```
   IF is_creatore_progetto THEN
               CALL  verifica_tipo_progetto(nome_progetto,  'SOFTWARE',
is_creatore_progetto);

       IF is_creatore_progetto THEN
           UPDATE CANDIDATURA
           SET stato = in_stato
           WHERE email_utente = in_email_candidato
           AND id_profilo = in_id_profilo;

           IF in_stato = 'ACCETTATA' THEN
               UPDATE PROFILO
               SET stato = 'OCCUPATO'
               WHERE id = in_id_profilo;
           END IF;
       END IF;
   END IF;
END //
DELIMITER ;


-- Procedura per l'inserimento di una componente su un progetto
hardware (solo creatore del progetto HARDWARE)
DROP PROCEDURE IF EXISTS inserisci_composizione;

DELIMITER //
CREATE PROCEDURE inserisci_composizione(
   IN in_nome_componente VARCHAR(32),
   IN in_quantita INT,
   IN in_nome_progetto VARCHAR(32),
   IN in_email_creatore VARCHAR(32),
   OUT is_creatore_progetto BOOLEAN
)
BEGIN
   CALL verifica_creatore_progetto(in_nome_progetto, in_email_creatore,
is_creatore_progetto);
   IF is_creatore_progetto THEN
            CALL  verifica_tipo_progetto(in_nome_progetto,  'HARDWARE',
is_creatore_progetto);
```

```
        IF is_creatore_progetto THEN
            INSERT INTO COMPOSIZIONE (nome_progetto, nome_componente,
quantita)
            VALUES (in_nome_progetto, in_nome_componente, in_quantita);
        END IF;
    END IF;
END //
DELIMITER ;


-- Procedura per l'inserimento di una nuova componente (solo creatori)
DROP PROCEDURE IF EXISTS inserisci_componente;

DELIMITER //
CREATE PROCEDURE inserisci_componente(
    IN in_nome VARCHAR(32),
    IN in_descrizione VARCHAR(255),
    IN in_prezzo DECIMAL(16,2),
    IN in_email_creatore VARCHAR(32),
    OUT esito BOOLEAN
)
BEGIN
    DECLARE is_creatore BOOLEAN;
    CALL verifica_creatore(in_email_creatore, is_creatore);

    IF is_creatore THEN
        INSERT INTO COMPONENTE (nome, descrizione, prezzo)
        VALUES (in_nome, in_descrizione, in_prezzo);
        SET esito = TRUE;
    ELSE
        SET esito = FALSE;
    END IF;
END //
DELIMITER ;


-- Procedura per la rimozione di un commento (solo creatore del
progetto o proprietario del commento)
DROP PROCEDURE IF EXISTS rimuovi_commento;
```

```
DELIMITER //
CREATE PROCEDURE rimuovi_commento(
   IN in_id_commento INT,
   IN in_email_utente VARCHAR(32),
   OUT esito BOOLEAN
)
BEGIN
   DECLARE nome_progetto VARCHAR(32);
   DECLARE email_proprietario_commento VARCHAR(32);
   DECLARE is_creatore_progetto BOOLEAN DEFAULT FALSE;
   DECLARE is_proprietario_commento BOOLEAN DEFAULT FALSE;

   SELECT c.nome_progetto, c.email_utente
   INTO nome_progetto, email_proprietario_commento
   FROM COMMENTO c
   WHERE c.id = in_id_commento;

   IF nome_progetto IS NOT NULL THEN
        CALL verifica_creatore_progetto(nome_progetto, in_email_utente,
is_creatore_progetto);
   END IF;

      SET  is_proprietario_commento  =  (email_proprietario_commento  =
in_email_utente);
   SET esito = (is_creatore_progetto OR is_proprietario_commento);

   IF esito THEN
       DELETE FROM COMMENTO WHERE id = in_id_commento;
   END IF;
END //
DELIMITER ;
```

Trigger

```
USE bostarter_db;



-- Trigger per aggiornare l'affidabilità quando viene creato un nuovo
progetto
DROP TRIGGER IF EXISTS aggiorna_affidabilita_nuovo_progetto;

DELIMITER //
CREATE TRIGGER aggiorna_affidabilita_nuovo_progetto
AFTER INSERT ON PROGETTO
FOR EACH ROW
BEGIN
   DECLARE progetti_totali INT DEFAULT 0;
   DECLARE progetti_finanziati INT DEFAULT 0;

   -- Conta il numero totale di progetti dell'utente
   SELECT COUNT(*) INTO progetti_totali
   FROM PROGETTO
   WHERE email_utente_creatore = NEW.email_utente_creatore;

   -- Conta il numero di progetti finanziati almeno una volta
   SELECT COUNT(DISTINCT p.nome) INTO progetti_finanziati
   FROM PROGETTO p
   JOIN FINANZIAMENTO f ON p.nome = f.nome_progetto
   WHERE p.email_utente_creatore = NEW.email_utente_creatore;

   -- Aggiorna l'affidabilità
   IF progetti_totali > 0 THEN
       UPDATE UTENTE_CREATORE
       SET affidabilita = (progetti_finanziati * 100 / progetti_totali)
       WHERE email_utente = NEW.email_utente_creatore;
   END IF;
END //
DELIMITER ;
```

```sql
-- Trigger per aggiornare l'affidabilità quando viene aggiunto un finanziamento
DROP TRIGGER IF EXISTS aggiorna_affidabilita_nuovo_finanziamento;

DELIMITER //
CREATE TRIGGER aggiorna_affidabilita_nuovo_finanziamento
AFTER INSERT ON FINANZIAMENTO
FOR EACH ROW
BEGIN
   DECLARE progetti_totali INT DEFAULT 0;
   DECLARE progetti_finanziati INT DEFAULT 0;
   DECLARE email VARCHAR(32) DEFAULT '';

   -- Ottiene l'email dell'utente creatore
   SELECT email_utente_creatore INTO email
   FROM PROGETTO
   WHERE nome = NEW.nome_progetto;

   -- Conta il numero totale di progetti dell'utente
   SELECT COUNT(*) INTO progetti_totali
   FROM PROGETTO
   WHERE email_utente_creatore = email;

   -- Conta il numero di progetti finanziati almeno una volta
   SELECT COUNT(DISTINCT p.nome) INTO progetti_finanziati
   FROM PROGETTO p
   JOIN FINANZIAMENTO f ON p.nome = f.nome_progetto
   WHERE p.email_utente_creatore = email;

   -- Aggiorna l'affidabilità
   IF progetti_totali > 0 THEN
       UPDATE UTENTE_CREATORE
       SET affidabilita = (progetti_finanziati * 100 / progetti_totali)
       WHERE email_utente = email;
   END IF;
END //
DELIMITER ;
```

```sql
-- Trigger per cambiare lo stato di un progetto al raggiungimento del budget
DROP TRIGGER IF EXISTS aggiorna_stato_progetto;

DELIMITER //
CREATE TRIGGER aggiorna_stato_progetto
AFTER INSERT ON FINANZIAMENTO
FOR EACH ROW
BEGIN
   DECLARE budget_progetto DECIMAL(16,2);
   DECLARE totale_finanziamenti DECIMAL(16,2);

   SELECT budget INTO budget_progetto
   FROM PROGETTO
   WHERE nome = NEW.nome_progetto;

   SELECT SUM(importo) INTO totale_finanziamenti
   FROM FINANZIAMENTO
   WHERE nome_progetto = NEW.nome_progetto;

   IF totale_finanziamenti >= budget_progetto THEN
       UPDATE PROGETTO
       SET stato = 'CHIUSO'
       WHERE nome = NEW.nome_progetto;
   END IF;
END //
DELIMITER ;


-- Trigger per incrementare il numero di progetti di un utente creatore
-- RIDONDANZA CONCETTUALE RIMOSSA
/*
DROP TRIGGER IF EXISTS incrementa_nr_progetti;

DELIMITER //
CREATE TRIGGER incrementa_nr_progetti
AFTER INSERT ON PROGETTO
FOR EACH ROW
BEGIN
```

```
    UPDATE UTENTE_CREATORE
    SET nr_progetti = nr_progetti + 1
    WHERE email_utente = NEW.email_utente_creatore;
END //
DELIMITER ;
*/


-- Trigger per aggiornare somma_raccolta quando viene inserito un finanziamento --
DROP TRIGGER IF EXISTS aggiorna_somma_raccolta;

DELIMITER //
CREATE TRIGGER aggiorna_somma_raccolta
AFTER INSERT ON FINANZIAMENTO
FOR EACH ROW
BEGIN
    UPDATE PROGETTO
    SET somma_raccolta = somma_raccolta + NEW.importo
    WHERE nome = NEW.nome_progetto;
END //
DELIMITER ;


-- Trigger per rifiutare le altre candidature all'accettazione di una candidatura --
DROP TRIGGER IF EXISTS rifiuta_candidature;

DELIMITER //
CREATE TRIGGER rifiuta_candidature
AFTER UPDATE ON PROFILO
FOR EACH ROW
BEGIN
    IF NEW.stato = 'OCCUPATO' THEN
        UPDATE CANDIDATURA
        SET stato = 'RIFIUTATA'
        WHERE id_profilo = NEW.id AND stato = 'ATTESA';
    END IF;
END //
```

Viste

```
USE bostarter_db;


-- Vista per tutti i progetti con la prima foto associata
DROP VIEW IF EXISTS progetti_con_foto;

CREATE VIEW progetti_con_foto AS
WITH prima_foto AS (
   SELECT f.immagine, f.nome_progetto
   FROM FOTO f
   WHERE f.id = (
       SELECT MIN(f2.id)
       FROM FOTO f2
       WHERE f2.nome_progetto = f.nome_progetto
   )
)
SELECT
   p.*,
   pf.immagine
FROM PROGETTO p
LEFT JOIN prima_foto pf
   ON p.nome = pf.nome_progetto;


-- Vista per la classifica dei 3 migliori utenti creatori per
affidabilità
DROP VIEW IF EXISTS classifica_creatori;

CREATE VIEW classifica_creatori AS
SELECT u.nickname, uc.affidabilita
FROM UTENTE u
```

```sql
JOIN UTENTE_CREATORE uc ON u.email = uc.email_utente
WHERE u.nickname IS NOT NULL
 AND u.nickname != ''
ORDER BY uc.affidabilita DESC
LIMIT 3;


-- Vista per i 3 progetti aperti più vicini al completamento
DROP VIEW IF EXISTS progetti_in_scadenza;

CREATE VIEW progetti_in_scadenza AS
SELECT
   p.nome, p.immagine, p.budget, p.somma_raccolta,
   (p.budget - COALESCE(SUM(f.importo), 0)) as differenza_budget
FROM progetti_con_foto p
JOIN FINANZIAMENTO f ON p.nome = f.nome_progetto
WHERE p.stato = 'APERTO'
 AND p.nome IS NOT NULL
 AND p.nome != ''
GROUP BY p.nome, p.budget
ORDER BY differenza_budget ASC
LIMIT 3;


-- Vista per la classifica dei 3 migliori utenti per totale finanziamenti erogati
DROP VIEW IF EXISTS classifica_finanziatori;

CREATE VIEW classifica_finanziatori AS
SELECT
   u.nickname,
   COALESCE(SUM(f.importo), 0) AS tot_finanziamenti
FROM UTENTE u
JOIN FINANZIAMENTO f ON u.email = f.email_utente
WHERE u.nickname IS NOT NULL
 AND u.nickname != ''
GROUP BY u.nickname
ORDER BY tot_finanziamenti DESC
LIMIT 3;
```

```sql
-- Vista per tutti i progetti aperti
DROP VIEW IF EXISTS progetti_aperti;

CREATE VIEW progetti_aperti AS
SELECT p.nome, p.descrizione, p.budget, p.tipo,
p.email_utente_creatore, p.immagine
FROM progetti_con_foto p
WHERE p.stato = 'APERTO';


-- Vista per tutte le foto di ogni progetto
DROP VIEW IF EXISTS foto_progetto;

CREATE VIEW foto_progetto AS
SELECT nome_progetto, immagine
FROM FOTO;


-- Vista per tutti i commenti di ogni progetto (con nickname)
DROP VIEW IF EXISTS commenti_progetto;

CREATE VIEW commenti_progetto AS
SELECT
   c.id,
   c.nome_progetto,
   c.email_utente,
   c.testo,
   u.nickname,
   c.data,
   r.testo AS risposta
FROM COMMENTO c
JOIN UTENTE u ON c.email_utente = u.email
LEFT JOIN RISPOSTA r ON c.id = r.id_commento
ORDER BY c.data DESC;

-- Vista per tutti i componenti presenti nel sistema
DROP VIEW IF EXISTS componenti;
```

```sql
CREATE VIEW componenti AS
SELECT *
FROM COMPONENTE;


-- Vista per tutti i componenti di un progetto con annessa quantità
DROP VIEW IF EXISTS componenti_progetto;
CREATE VIEW componenti_progetto AS
SELECT
   c.*,
   cp.nome_progetto,
   cp.quantita
FROM COMPONENTE c
JOIN COMPOSIZIONE cp ON c.nome = cp.nome_componente;
```