UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

DOCUMENTAZIONE
BASI DI DATI

Un progetto svolto da:

MARIO PALLADINO

MATRICOLA: N86005143

ENRICO ROMANO

MATRICOLA: N86005059

CLAUDIO ZANNETTI

MATRICOLA: N86005146

Anno Accademico 2024/2025

INDICE

1. INTRODUZIONE

Il seguente elaborato ha lo scopo di documentare la progettazione e lo sviluppo di una base di dati relazionale del DBMS PostgreSQL. Ad opera degli studenti Mario Palladino, Enrico Romano e Claudio Zannetti del CdL in Informatica presso l'Università degli Studi di Napoli "Federico II. Il Database nasce come progetto a scopi valutativi per il corso di Basi di Dati, ed implementa un sistema di gestione delle attività personali da svolgere.

1.1 DESCRIZIONE DEL PROBLEMA

Il presente elaborato ha lo scopo di documentare la progettazione e lo sviluppo di una base di dati relazionale per la gestione avanzata di attività personali (ToDo). Il sistema permette agli utenti di registrarsi e organizzare le proprie attività in diverse bacheche. Rispetto ai sistemi tradizionali, questo progetto introduce il concetto di **Checklist**: ogni attività può essere suddivisa in sotto-attività puntuali. Il sistema gestisce automaticamente lo stato di completamento del ToDo principale in base all'avanzamento della relativa checklist.

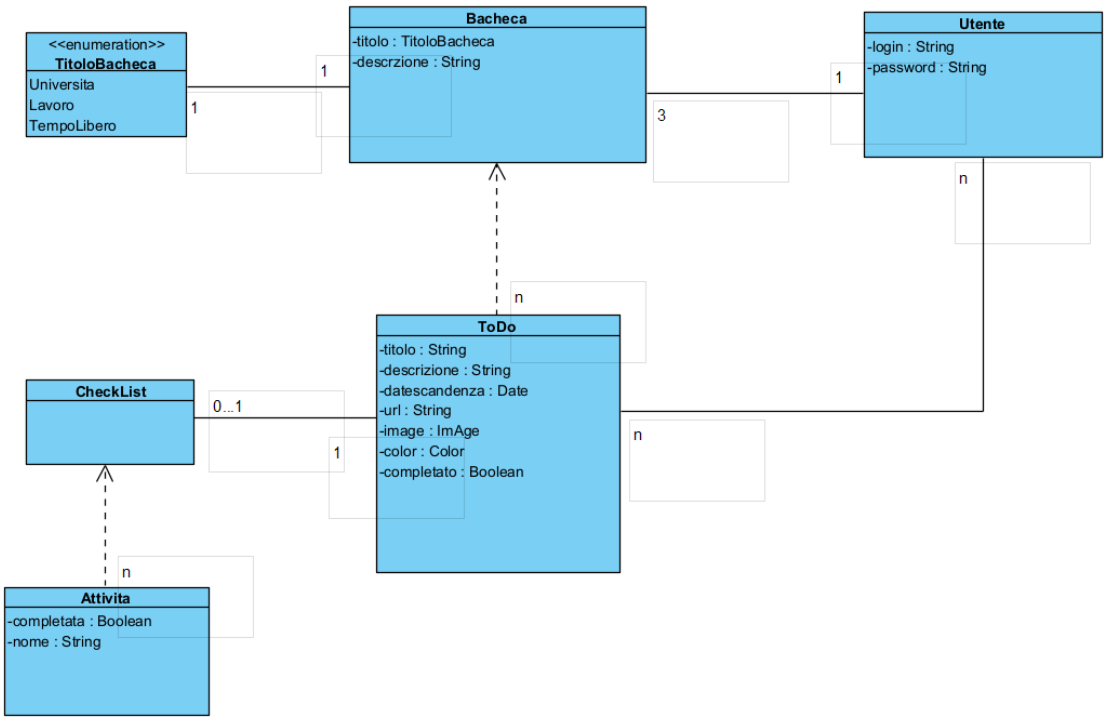
Le funzionalità principali includono:

1. Gestione utenti e autenticazione.
2. Organizzazione dei ToDo in Bacheche numerate.
3. Suddivisione dei ToDo in voci di Checklist.
4. Condivisione dei ToDo tra utenti diversi.
5. Automatismi per l'aggiornamento dello stato (completato/non completato).

2. PROGETTAZIONE CONCETTUALE

In questo capitolo viene analizzata la struttura del database a livello di astrazione alto, definendo le entità e le relazioni tramite diagrammi UML.

2.1 CLASS DIAGRAM



2.2 RISTRUTTURAZIONE DEL CLASS DIAGRAM

Durante la fase di analisi, sono state effettuate le seguenti scelte progettuali per ottimizzare lo schema e renderlo aderente ai requisiti implementativi:

2.2.1 ANALISI DELLE CHIAVI

- **Utente (login_utenti):** Si è scelto di utilizzare il login (username) come chiave primaria, in quanto identifica univocamente l'utente nel sistema.
- **Bacheca, ToDo, Checklist:** Per queste entità si è optato per chiavi surrogate autoincrementali (SERIAL), identificate rispettivamente come idbacheca, idtodo, idchecklist, per garantire un riferimento stabile.
- **Condivisi:** La tabella di relazione utilizza una chiave composta dalla coppia (idtodo, login_condiviso).

2.2.2 Analisi degli attributi derivati e ridondanze

L'attributo completato nell'entità ToDo è tecnicamente derivabile dallo stato delle voci nella Checklist. Tuttavia, si è deciso di mantenere questo attributo e aggiornarlo tramite un Trigger (verifica_completamento_todo) per ottimizzare le query di lettura, evitando join costosi ogni volta che si visualizza l'elenco dei ToDo.

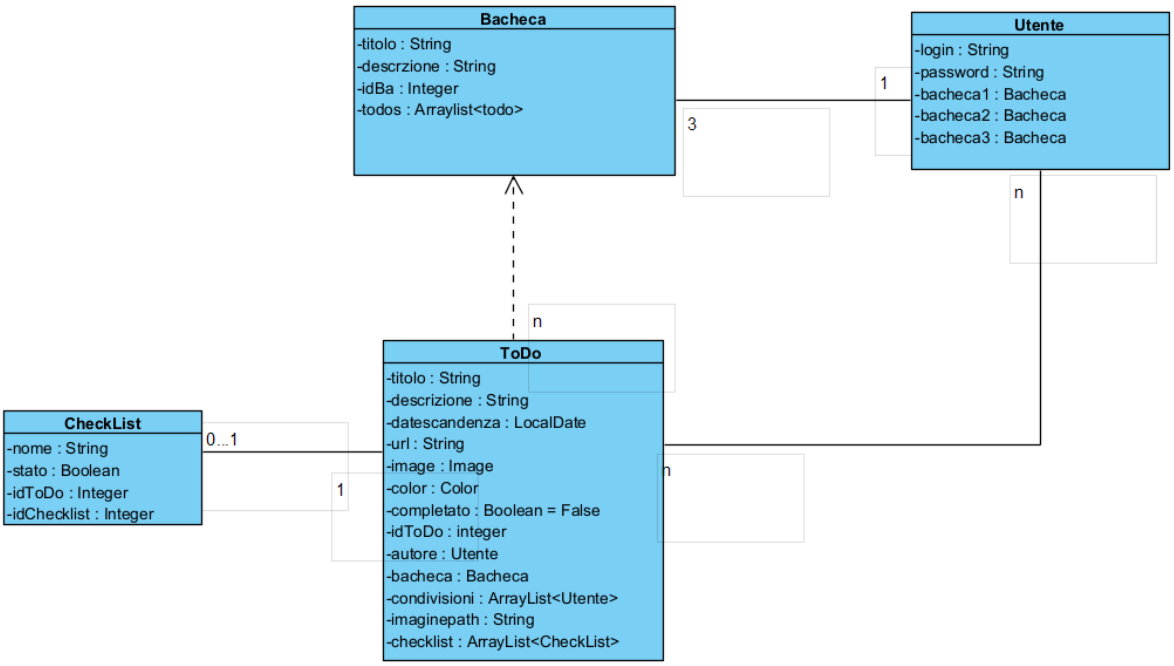
2.2.3 Rimozione delle Enumerazioni

Nel diagramma iniziale era prevista un'enumerazione TitoloBacheca. In fase di ristrutturazione, questa è stata sostituita da un semplice attributo stringa (VARCHAR) nell'entità **Bacheca**. Questa modifica rende il sistema più flessibile, permettendo agli utenti di creare bacheche con nomi personalizzati (es. "Progetto Basi di Dati") invece di essere vincolati a categorie predefinite.

2.2.4 Accorpamento delle entità

La generalizzazione tra Checklist e Attività presente nel diagramma iniziale è stata rimossa per semplificazione. Le due entità sono state fuse nell'unica entità **Checklist**, che ora contiene direttamente gli attributi nome e stato, eliminando una complessità strutturale non necessaria.

2.3 CLASS DIAGRAM RISTRUTTURATO



2.4 DIZIONARIO DELLE CLASSI

In questa tabella vengono descritti gli attributi presenti nel **Class Diagram Ristrutturato**, mappandoli con la relativa implementazione nel database relazionale.

Classe	Attributo UML/Java	Corrispondente SQL	Descrizione e Note
Utente	login	login (PK)	Username univoco.
	password	password	Hash della password.
	nome	nome	Nome visualizzato dell'utente.
	bacheca1..3	*(Relazione FK)*	Nel codice Java sono attributi espliciti. Nel DB sono record nella tabella Bacheca collegati tramite FK proprietario.
Bacheca	idBa	idbacheca (PK)	Identificativo numerico univoco.
	titolo	titolo	Nome della bacheca.
	descrizione	descrizione	Dettagli sulla bacheca.
	numero	numero	(Attributo logico) Definisce l'ordinamento (1, 2, 3) per l'utente.
	todos	*(Relazione FK)*	Lista (ArrayList) in Java. Nel DB gestita tramite chiave esterna in todo.
ToDo	idToDo	idtodo (PK)	ID attività.
	titolo, descrizione	titolo, descrizione	Dati testuali.
	datescadenza	data_scadenza	Oggetto LocalDate.
	url, image	url, immagine	Link e percorso file (Stringa).
	completato	completato	Stato del task (Boolean).
	color	colore	Codice HEX colore.
	autore, bacheca	autore, idba	Chiavi esterne verso Utente e Bacheca.
	checklist	*(Relazione FK)*	Lista di sotto-attività.
CheckList	idChecklist	idchecklist (PK)	ID voce univoco.
	nome, stato	nome, stato	Testo e stato (boolean).
	idToDo	idtodo (FK)	Riferimento al padre.

2.5 DIZIONARIO DELLE ASSOCIAZIONI

Relazione	Entità Coinvolte	Descrizione
Creazione	Utente (1) ↔ ToDo (N)	Un utente (autore) crea molteplici ToDo. Il database traccia l'autore in ogni ToDo.
Appartenenza	Bacheca (1) ↔ ToDo (N)	Un ToDo è contenuto in una specifica bacheca.
Composizione	ToDo (1) ↔ Checklist (N)	Un ToDo è composto da varie voci. Eliminando il ToDo, si eliminano a cascata le voci associate.
Condivisione	ToDo (N) ↔ Utente (N)	Relazione Molti-a-Molti. Un ToDo può essere condiviso con più utenti (diversi dall'autore). Questa associazione è gestita dalla tabella di giunzione *Condivisi*.

2.6 Dizionario dei Vincoli

Tipologia	Vincolo / Regola	Descrizione
Chiavi Esterne (FK)	• todo.autore → login_utenti.login • todo.idba → bacheca.idbacheca • checklist.idtodo → todo.idtodo • condivisi.idtodo → todo.idtodo • condivisi.login_condiviso → login_utenti.login	Garantiscono l'integrità referenziale: non è possibile creare un ToDo senza autore o bacheca, né voci di checklist o condivisioni orfane (senza ToDo padre).
Chiave Primaria Composta	• condivisi (idtodo, login_condiviso)	Garantisce che lo stesso ToDo non possa essere condiviso più volte con lo stesso utente (evita duplicati nella tabella di relazione).
Integrità Logica	bacheca.numero	Identifica il numero d'ordine della bacheca (1, 2 o 3) all'interno del profilo dell'utente proprietario.
Vincoli Procedurali (Trigger)	• Coerenza Stato ToDo	Un automatismo (Trigger verifica_completamento_todo) impedisce incongruenze tra il ToDo e la sua Checklist: se tutte le voci sono spuntate, il ToDo diventa automaticamente "Completato"; se se ne deseleziona una, torna "Non Completato".
Vincoli Procedurali (Trigger)	• Integrità Spostamenti	Un automatismo (Trigger tgr_propaga_modifiche_todo) intercetta il cambio di bacheca (idba) per gestire eventuali operazioni di pulizia o notifiche necessarie al trasferimento dell'attività.

3. PROGETTAZIONE LOGICA

3.1 SCHEMA LOGICO

SCHEMA RELAZIONALE

1. **LOGIN_UTENTI** (login, password, nome)
2. **BACHECA** (idbacheca,titolo, descrizione, numero, *proprietario*)
3. **TODO** (idtodo, titolo, descrizione, data_scadenza, url, immagine, colore, completato, posizione, *autore*, *idba*)
4. **CHECKLIST** (idchecklist, nome, stato, *idtodo*)
5. **CONDIVISI** (*idtodo*, *login_condiviso*)

1. UTENTE

Gestisce le credenziali di accesso e l'identità degli utenti nel sistema.

Attributo	Tipo SQL	Descrizione	Vincoli
login	VARCHAR(20)	Username univoco per l'accesso	PK
password	VARCHAR(255)	Password (o hash) dell'utente	Not Null
nome	VARCHAR(20)	Nome visualizzato nell'interfaccia	Opzionale

2. BACHECA

Rappresenta i contenitori logici (es. Università, Lavoro) in cui l'utente organizza i propri task.

Attributo	Tipo SQL	Descrizione	Vincoli
idbacheca	SERIAL	Identificativo univoco della bacheca	PK
titolo	VARCHAR(20)	Nome della categoria (es. "Lavoro").	Not Null
descrizione	VARCHAR(100)	Dettagli aggiuntivi	Opzionale
numero	INTEGER	Numero d'ordine (1, 2, 3) per l'interfaccia	Not Null
proprietario	VARCHAR(20)	Utente che possiede la bacheca	FK su Utente

3. ToDo

L'entità centrale del sistema, rappresenta la singola attività da svolgere.

Attributo	Tipo SQL	Descrizione	Vincoli
idtodo	SERIAL	Identificativo univoco del task	PK
titolo	VARCHAR(20)	Oggetto dell'attività	Not Null
completato	BOOLEAN	Stato del task (Vero/Falso)	Default FALSE
data_scadenza	DATE	Data limite per il completamento	Opzionale
posizione	INTEGER	Ordine di visualizzazione nella lista	Opzionale
autore	VARCHAR(20)	Utente che ha creato il task	FK su Utente
idba	INTEGER	Bacheca in cui è contenuto il task	FK su Bacheca
Altri	*Varchar/Color*	Url, Immagine, Descrizione, Colore	-

ENTITÀ CORRELATE E RELAZIONI

4. CHECKLIST

Rappresenta le sotto-attività atomiche di un ToDo complesso.

Attributo	Tipo Sql	Descrizione	Vincoli
idchecklist	identity	identificativo della voce	PK
nome	varchar(255)	testo dell'attività da spuntare	Not Null
stato	boolean	stato (fatto/da fare)	Default False
idtodo	integer	riferimento al todo padre	FK

5. CONDIVISI

Tabella di associazione che gestisce la condivisione dei task tra utenti diversi.

Attributo	Tipo SQL	Descrizione	Vincoli
idtodo	INTEGER	Il ToDo condiviso.	PK, FK
login_condiviso	VARCHAR(20)	L'utente con cui si condivide.	PK, FK

RIEPILOGO DELLE RELAZIONI

Le chiavi esterne (FK) definiscono la struttura relazionale del database, garantendo l'integrità referenziale e definendo il comportamento in caso di cancellazione (Cascade).

1. Relazione bacheca a login_utenti (Proprietà):

- FK: bacheca.proprietario → login_utenti.login
- Tipo: Molti-a-Uno (Molte bacheche appartengono a un singolo utente).

Vincolo ON DELETE CASCADE attivo.

2. Relazione todo a login_utenti (Autore):

- FK: todo.autore → login_utenti.login
- Tipo: Molti-a-Uno (Molti todo sono creati da un singolo login_utenti).

3. Relazione todo a bacheca:

- FK: todo.idba → bacheca.idbacheca
- Tipo: Molti-a-Uno (Molti todo appartengono a una singola bacheca).

4. Relazione checklist a todo:

- FK: checklist.idtodo → todo.idtodo
- Tipo: Molti-a-Uno (Molti elementi di checklist appartengono a un singolo todo).

Vincolo ON DELETE CASCADE attivo.

5. Relazione condivisi a todo:

- FK: condivisi.idtodo → todo.idtodo
- Tipo: Molti-a-Uno (Un todo può essere presente in molte righe di condivisi).

Vincolo ON DELETE CASCADE attivo.

6. Relazione condivisi a login_utenti (Condiviso):

- FK: condivisi.login_condiviso → login_utenti.login
- Tipo: Molti-a-Uno (Un login_utenti può apparire in molte righe di condivisi).

Vincolo ON DELETE CASCADE attivo.

4. PROGETTAZIONE FISICA

In questa sezione viene riportata l'implementazione del database attraverso il linguaggio SQL (PostgreSQL), includendo la definizione delle tabelle, i vincoli di integrità, il popolamento iniziale dei dati e gli automatismi tramite trigger.

4.1 DEFINIZIONE TABELLE

4.1.1 DEFINIZIONE DELLA TABELLA LOGIN_UTENTI

Questa tabella gestisce le credenziali di accesso e l'identità digitale degli utenti. L'attributo login funge da chiave primaria per garantire l'univocità dello username, mentre il campo nome memorizza il nome visualizzato nell'applicazione.

```
CREATE TABLE login_utenti (
        login VARCHAR(20) PRIMARY KEY,
        password VARCHAR(255) NOT NULL,
        nome VARCHAR(20)
);
```

4.1.2 DEFINIZIONE DELLA TABELLA BACHECA

Definisce i contenitori logici utilizzati per organizzare le attività. Ogni bacheca è collegata a un utente proprietario tramite chiave esterna. L'attributo numero è fondamentale per gestire l'ordinamento delle tre bacheche personali (Università, Lavoro, Tempo Libero) nell'interfaccia Java. Il vincolo ON DELETE CASCADE assicura che, eliminando un utente, vengano rimosse anche le sue bacheche.

```
CREATE TABLE bacheca (
  idbacheca SERIAL PRIMARY KEY,
  titolo VARCHAR(20) NOT NULL,
  descrizione VARCHAR(100),
  proprietario VARCHAR(20),
  numero INTEGER NOT NULL,
  FOREIGN KEY (proprietario) REFERENCES login_utenti (login) ON DELETE CASCADE
);
```

4.1.3 Definizione della tabella TODO

Costituisce l'entità centrale dello schema. Contiene i dettagli operativi dell'attività (titolo, scadenze, colore, stato) e gestisce le relazioni con l'autore e la bacheca di appartenenza. L'attributo posizione è stato aggiunto per permettere il futuro ordinamento personalizzato dei task all'interno della lista.

```
CREATE TABLE todo (
  idtodo SERIAL PRIMARY KEY,
  titolo VARCHAR(20) NOT NULL,
  completato BOOLEAN DEFAULT FALSE,
  url VARCHAR(500),
  immagine VARCHAR(500),
  descrizione VARCHAR(100),
  data_scadenza DATE,
  colore VARCHAR(7),
  autore VARCHAR(20),
  idba INTEGER,
  posizione INTEGER,
  FOREIGN KEY (autore) REFERENCES login_utenti (login),
  FOREIGN KEY (idba) REFERENCES bacheca (idbacheca)
);
```

4.1.4 Definizione della tabella CHECKLIST

Implementa la scomposizione di un ToDo in sotto-attività atomiche. Utilizza una chiave primaria autoincrementale di tipo IDENTITY. La chiave esterna verso la tabella todo include il vincolo ON DELETE CASCADE, garantendo che l'eliminazione di un'attività principale comporti la cancellazione automatica di tutte le sue voci di checklist.

```
CREATE TABLE checklist (
  idchecklist INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  nome VARCHAR(255),
  stato BOOLEAN DEFAULT FALSE,
  idtodo INTEGER REFERENCES todo(idtodo) ON DELETE CASCADE
);
```

4.1.5 Definizione della tabella CONDIVISI

Risolve la relazione Molti-a-Molti necessaria per la collaborazione. Collega un ToDo a un utente diverso dall'autore (il destinatario della condivisione). Entrambe le chiavi esterne sono dotate di ON DELETE CASCADE per mantenere la consistenza del database in caso di rimozione di utenti o attività.

```
CREATE TABLE condivisi (
  idtodo INTEGER,
  login_condiviso VARCHAR(20),
  PRIMARY KEY (idtodo, login_condiviso),
  FOREIGN KEY (idtodo) REFERENCES todo (idtodo) ON DELETE CASCADE,
  FOREIGN KEY (login_condiviso) REFERENCES login_utenti (login) ON DELETE CASCADE
);
```

4.2 IMPLEMENTAZIONE DEI VINCOLI E TRIGGER

Funzione: verifica_completamento_todo()

```
CREATE OR REPLACE FUNCTION verifica_completamento_todo()
RETURNS TRIGGER AS $$
DECLARE
  v_id_todo INTEGER;
  v_non_completate INTEGER;
BEGIN
  IF TG_OP = 'DELETE' THEN
    v_id_todo := OLD.idtodo;
  ELSE
    v_id_todo := NEW.idtodo;
  END IF;

  SELECT COUNT(*) INTO v_non_completate
  FROM checklist
  WHERE idtodo = v_id_todo AND stato = FALSE;

  IF v_non_completate = 0 THEN
    UPDATE todo SET completato = TRUE WHERE idtodo = v_id_todo;
  ELSE
    UPDATE todo SET completato = FALSE WHERE idtodo = v_id_todo;
  END IF;

  RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

Trigger: tgr_aggiorna_stato_todo

```
CREATE TRIGGER tgr_aggiorna_stato_todo
AFTER INSERT OR UPDATE OR DELETE ON checklist
FOR EACH ROW
EXECUTE FUNCTION verifica_completamento_todo();
```

Funzione: propaga_spostamento_todo()

```
CREATE OR REPLACE FUNCTION propaga_spostamento_todo()
RETURNS TRIGGER AS $$
BEGIN
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Trigger: tgr_propaga_modifiche_todo

```
CREATE TRIGGER tgr_propaga_modifiche_todo
AFTER UPDATE ON todo
FOR EACH ROW
WHEN (OLD.idba IS DISTINCT FROM NEW.idba)
EXECUTE FUNCTION propaga_spostamento_todo();
```

4.3 POPOLAMENTO INIZIALE (DATI DI PROVA)

INSERIMENTO PRIMO UTENTE

```
INSERT INTO login_utenti (login, password, nome) VALUES ('mario', '1234', 'Mario Rossi');
INSERT INTO bacheca (idbacheca, titolo, descrizione, proprietario, numero)
VALUES
(101, 'Università', 'Bacheca per lo studio', 'mario', 1),
(102, 'Lavoro', 'Bacheca per il lavoro', 'mario', 2),
(103, 'Tempo Libero', 'Bacheca per il tempo libero', 'mario', 3);
```

INSERIMENTO SECONDO UTENTE

```
INSERT INTO login_utenti (login, password, nome)
VALUES ('claudio', '1234', 'Claudio Bianchi');
INSERT INTO bacheca (idbacheca, titolo, descrizione, proprietario, numero)
VALUES
(201, 'Università', 'Bacheca per lo studio', 'claudio', 1),
(202, 'Lavoro', 'Bacheca per il lavoro', 'claudio', 2),
(203, 'Tempo Libero', 'Bacheca per il tempo libero', 'claudio', 3);
```