UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

Dipartimento di Ingegneria
Corso di Laurea Triennale in Informatica

Progettazione e sviluppo della base di dati SavingMoneyUnina

Docente:
Prof. Mara Sangiovanni

Autori:
Francesco Donnarumma
N86004658
Arturo Donnarumma
N86004837

Anno Accademico 2023/2024

Indice

Capitolo 1

Introduzione

Benvenuti nella documentazione dettagliata relativa alla struttura del database di SavingMoneyUnina. Questo documento fornisce una panoramica completa degli elementi chiave che costituiscono la base di dati, offrendo informazioni essenziali sulla progettazione e organizzazione necessarie per una gestione efficiente delle transazioni finanziarie.

Il database di SavingMoneyUnina è stato progettato per facilitare la registrazione, il recupero e l'analisi efficiente delle informazioni finanziarie personali e familiari. Attraverso una struttura intuitiva, consentiamo agli utenti di tracciare e gestire le transazioni provenienti da diverse fonti finanziarie.

La documentazione dettaglierà le tabelle principali, le relazioni chiave e gli schemi di collegamento tra i dati, fornendo una visione chiara sulla gestione automatica e manuale delle transazioni.

Questa guida è essenziale per coloro che necessitano di una visione approfondita sulla progettazione del database, utile sia nello sviluppo che nella manutenzione del sistema nell'ecosistema finanziario.

Capitolo 2

Progettazione Concettuale

2.1 Diagramma Delle Classi UML

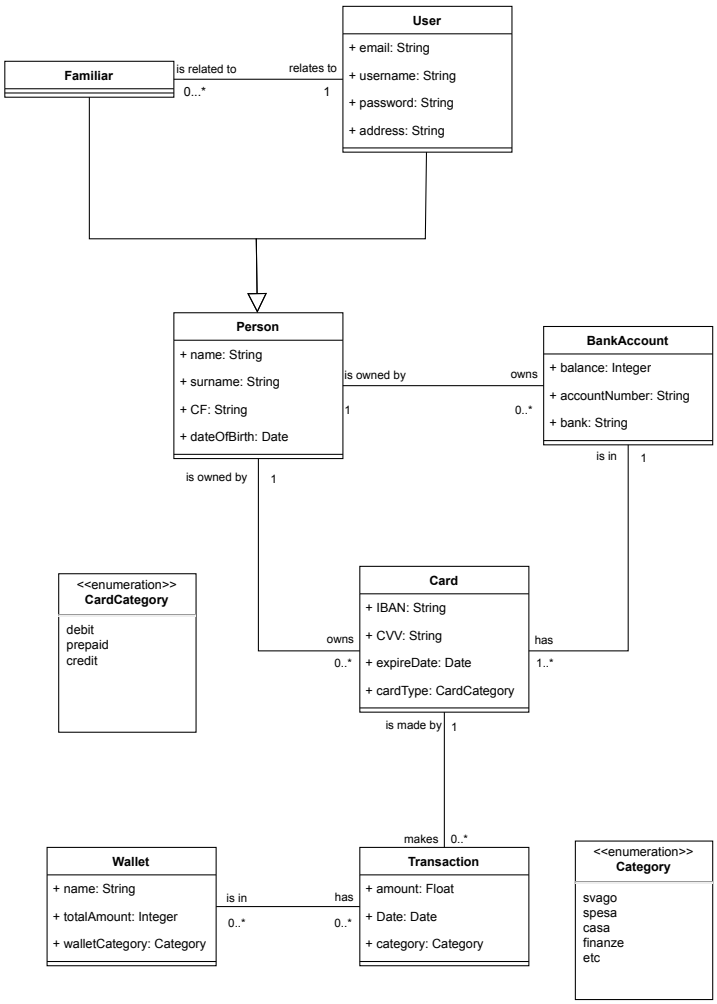

Figura 2.1: Diagramma UML

2.2 Diagramma ER (Entità Relazione)

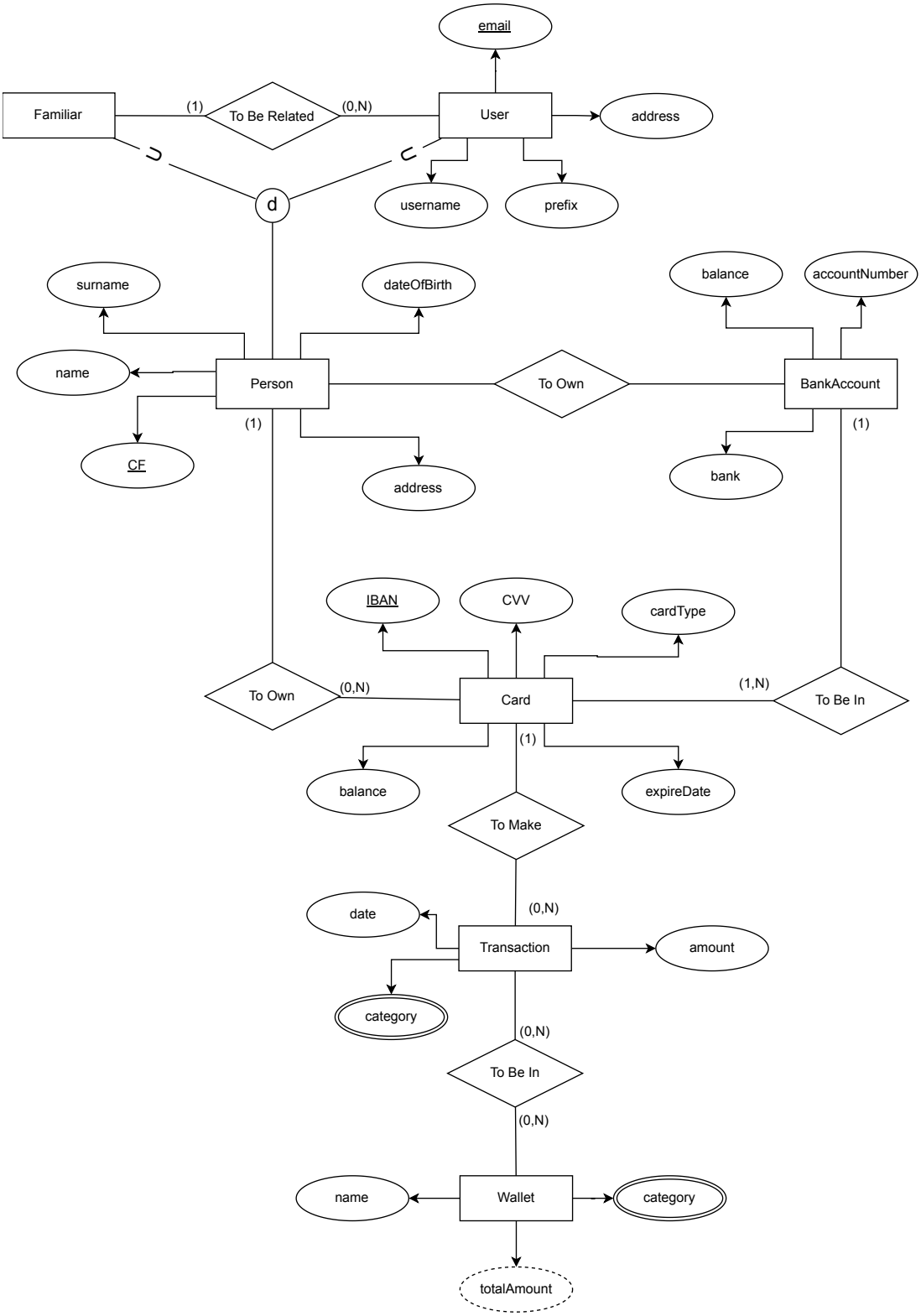

Figura 2.2: Diagramma ER

2.3 Ristrutturazione

2.3.1 Attributi multipli

Per quanto riguarda la gestione di attributi multipli, abbiamo deciso di gestire l'attributo *category* della tabella **Transaction**, originariamente definito come enumerazione, trasformandolo in una stringa, poiché non abbiamo bisogno di valori specifici, trattandosi di una categoria personalizzabile.

Anche per l'attributo *cardType* è stata applicata la stessa procedura. Il controllo dell'attributo verrà gestito tramite i vincoli approfonditi nel dizionario dei vincoli.

2.3.2 Generalizzazioni

Per la generalizzazione, essendo di tipologia totale e disgiunta, abbiamo optato per il metodo di eliminare la classe generale. Abbiamo trasferito tutti gli attributi di essa nelle classi specializzate, conservando le relative relazioni.

2.3.3 Analisi degli identificativi

Per la maggior parte delle classi, saranno utilizzati come identificativi attributi già presenti di natura nelle classi stesse, poiché risultano sufficienti e non richiedono l'uso di una chiave surrogata. Tuttavia, in alcune classi, sono presenti chiavi surrogate, identificate con il prefisso **ID_**.

2.3.4 Diagramma UML ristrutturato

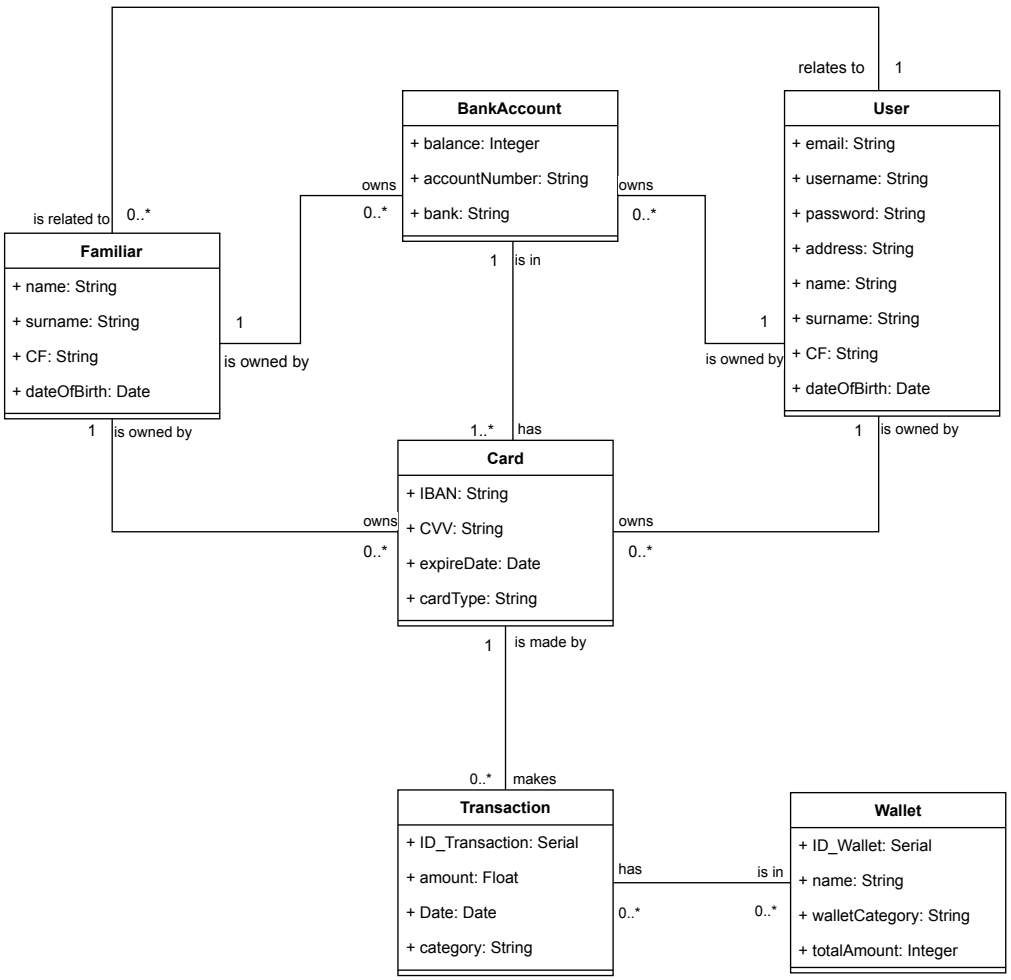

Figura 2.3: Diagramma UML Ristrutturato

2.4 Dizionari

2.4.1 Dizionario delle classi

Classe	Descrizione	Attributi
User	Classe utilizzata per identificare gli utenti registrati alla piattaforma.	**email** (*String*): chiave primaria, email con la quale l'utente si è registrato. **username** (*String*): nome che viene mostrato per riconoscere lo stesso. **password** (*String*): stringa atta alla convalidazione durante l'accesso all'account. **address** (*String*): indirizzo del domicilio. **name** (*String*): nome. **surname** (*String*): cognome. **CF** (*String*): codice fiscale. **dateOfBirth** (*Date*): data di nascita.
Familiar	Classe utilizzata per identificare i familiari degli utenti presenti nel database.	**name** (*String*): nome. **surname** (*String*): cognome. **CF** (*String*): codice fiscale, chiave primaria nel caso del familiare. **dateOfBirth** (*Date*): data di nascita.
BankAccount	Classe utilizzata per identificare i conti correnti appartenenti a utenti o familiari.	**balance** (*Integer*): indica il saldo disponibile sul conto corrente. **accountNumber** (*String*): chiave primaria, identificativa del conto corrente. **bank** (*String*): nome della banca alla quale è associato il conto corrente.
Card	Classe utilizzata per identificare le carte appartenenti a utenti o familiari.	**IBAN** (*String*): codice identificativo della carta. **CVV** (*String*): codice di sicurezza per le transazioni delle carte. **expireDate** (*Date*): data che indica la scadenza della carta. **cardType** (*String*): campo che indica la tipologia della carta.
Transaction	Classe utilizzata per tenere traccia di tutte le transazioni effettuate.	**ID_Transaction** (*Serial*): chiave surrogata, identificativo della singola transazione. **amount** (*Float*): indica l'ammontare della transazione. **date** (*Date*): data in cui è avvenuta la transazione. **category** (*String*): tipologia di transazione. Serve per l'associazione automatica ai portafogli.

Classe	Descrizione	Attributi
Wallet	Classe utilizzata per raggruppare transazioni.	**ID_Wallet** (*Serial*): chiave surrogata, identificativo del singolo protafoglio. **name** (*String*): nome del portafoglio. **walletCategory** (*String*): categoria del portafoglio. **totalAmount** (*Float*): indica la somma di tutte le transazioni relative al portafoglio.

2.4.2 Dizionario delle associazioni

Associazione	Descrizione	Classi coinvolte
To Be Related	Esprime la parentela tra gli utenti e i familiari	**Familiar** [1] (**relates to**): indica, per ogni familiare, con quale utente è imparentato. **User** [**0..***] (**is related to**): indica quali sono i familiari che sono imparentati con esso.
To Own	Esprime il possesso degli utenti sui conti correnti	**User** [**0..***] (**owns**): indica, per ogni utente, quali sono i conti correnti che possiede. **BankAccount** [1] (**is owned by**): indica l'utente che possiede il conto corrente in questione.
To Own	Esprime il possesso dei familiari sui conti correnti	**Familiar** [**0..***] (**owns**): indica, per ogni familiare, quali sono i conti correnti che possiede. **BankAccount** [1] (**is owned by**): indica il familiare che possiede il conto corrente in questione.
To Be In	Esprime la correlazione tra le carte e i conti correnti	**Card** [1] (**is in**): indica, per ogni carta, qual è il conto corrente a cui sono associate. **BankAccount** [**1..***] (**has**): indica quali sono le carte che sono associate al conto corrente in questione.
To Own	Esprime il possesso degli utenti sulle carte	**User** [**0..***] (**owns**): indica, per ogni utente, quali sono le carte che possiede. **Card** [1] (**is owned by**): indica l'utente che possiede la carta in questione.
To Own	Esprime il possesso dei familiari sulle carte	**Familiar** [**0..***] (**owns**): indica, per ogni utente, quali sono le carte che possiede. **Card** [1] (**is owned by**): indica l'utente che possiede la carta in questione.

Associazione	Descrizione	Classi coinvolte
To Make	Esprime il collegamento una la transazione e la carta con la quale è stata effettuata	**Card** [**0..***] (**makes**): indica, per ogni carta, tutte le transazioni effettuate. **Transaction** [**1**] (**is made by**): indica con quale carta è stata effettuata la transazione in questione.
To Be In	Esprime la correlazione tra le transazioni e i portafogli	**Wallet** [**0..***] (**has**): indica, per ogni portafoglio, quali sono le transazioni che lo compongono. **Transaction** [**0..***] (**is in**): indica qual è il portafoglio a cui fa riferimento la transazione in questione.

2.4.3 Dizionario dei vincoli

Vincolo	Tipo	Descrizione
unique_email	Intrarelazionale	Nella tabella User non ci possono essere t-uple diverse con la stessa email.
unique_CF	Intrarelazionale	Nella tabella User non ci possono essere t-uple diverse con lo stesso CF.
check_BirthDate_User	Dominio	Nella tabella User la data deve essere necessariamente antecedente alla data odierna.
check_BirthDate_Familiar	Dominio	Nella tabella Familiar la data deve essere necessariamente antecedente alla data odierna.
ownership_check_BA	N-upla	Per ogni t-upla della tabella BankAccount, essa deve essere associata necessariamente o ad un Utente o ad un Familiare, ma non ad entrambi.
ownership_check_Card	N-upla	Per ogni t-upla della tabella Card, essa deve essere associata necessariamente o ad un Utente o ad un Familiare, ma non ad entrambi.
cardType_Check	Dominio	Nella tabella Card, per ogni t-upla, il campo cardCategory deve essere necessariamente "prepaid", "debit" o "credit".
check_Transaction_Date	Dominio	Nella tabella Transaction, per ogni t-upla, la data deve essere necessariamente antecedente o coincidente alla data odierna.

Capitolo 3

Progettazione Logica

3.1 Schema Logico

3.1.1 Traduzione delle classi e delle associazioni

User (email, username, password, address, name, surname, CF, dateOfBirth)

Familiar (name, surname, CF, dateOfBirth, familiarEmail)
Chiavi esterne: familiarEmail → User.email

BankAccount (balance, accountNumber, bank, ownerCF, ownerEmail)
Chiavi esterne: ownerCF → Familiar.CF
ownerEmail → User.email

Card (IBAN, CVV, expireDate, cardCategory, BA_Number, ownerCF, ownerEmail)
Chiavi esterne: BA_Number → BankAccount.accountNumber
ownerCF → Familiar.CF
ownerEmail → User.email

Wallet (ID_Wallet, name, WalletCategory, totalAmount)

Transaction (ID_Transaction, amount, date, category, CardIBAN)
Chiavi esterne: cardIBAN → Card.IBAN

TransactionInWallet (ID_Transaction, ID_Wallet)
Chiavi esterne: ID_Transaction → Transaction.ID_Transaction
ID_Wallet → Wallet.ID_Wallet

3.1.2 Modalità di traduzione delle associazioni

Associazione	Implementazione
To Be Related	Relazione 0..* a 1, è stata migrata la chiave primaria di *User* (username) in *Familiar* (familiarUsername)
To Own	Tutte le relazioni di questo genere, tra *Familiar*, *User*, *BankAccount* e *Card*, sono di tipologia 0..* a 1, di conseguenza sono state gestite tutte allo stesso modo. Ovvero migrando la chiave primaria dell'entità debole, nell'entità forte. Per controllare nel dettaglio le chiavi esterne in ognuna delle tabelle indicate vedere la tabella Traduzione delle classi.
To Be In (Card/BankAccount)	Relazione 1..* a 1, è stata migrata la chiave primaria di *BankAccount* (accountNumber) in *Card* (BA_Number)
To Make	Relazione 0..* a 1, è stata migrata la chiave primaria di *Card* (IBAN) in *Transaction* (CardIBAN)
To Be In (Transaction/Wallet)	Relazione 0..* a 0..*, è stata creata la tabella ponte *TransactionInWallet* che contiene le chiavi primarie di *Transaction* (ID_Transaction) e di *Wallet* (ID_Wallet)

Capitolo 4

Schema Fisico

4.1 Definizoni SQL delle tabelle