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

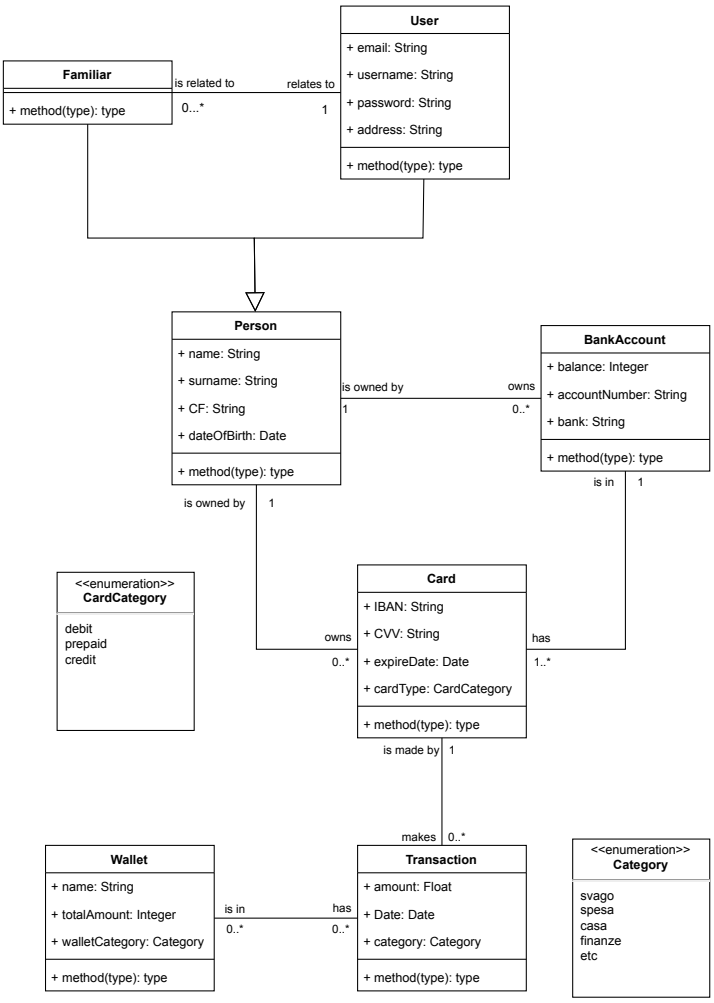

Figura 2.1: Diagramma UML

2.2 Diagramma ER (Entità Relazione)

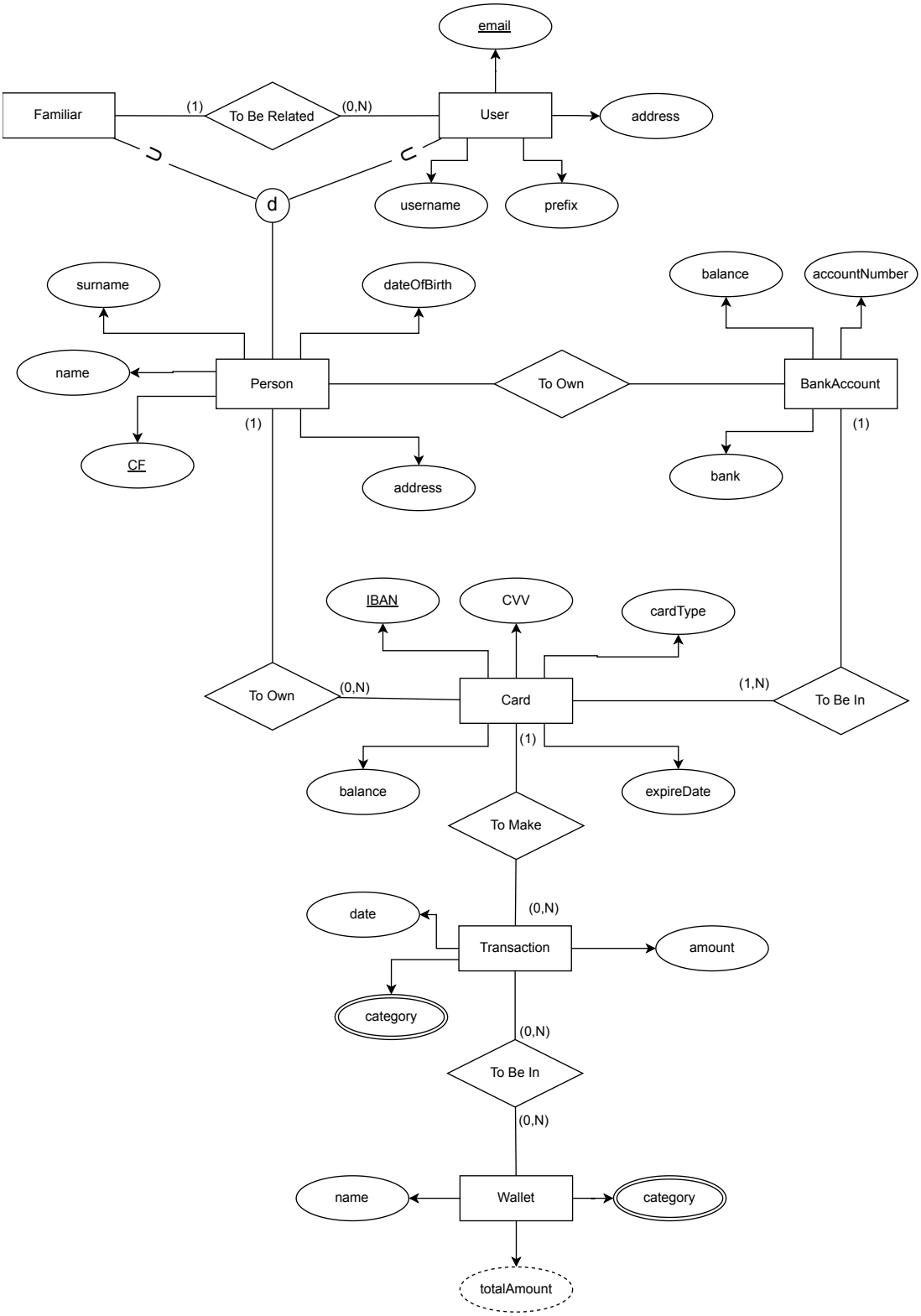

Figura 2.2: Diagramma ER

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.

Invece, per l'attributo *cardType* della tabella **Card**, è stato deciso di non applicare lo stesso metodo, poiché le tipologie di carte sono ben definite e non possono essere modificate.

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

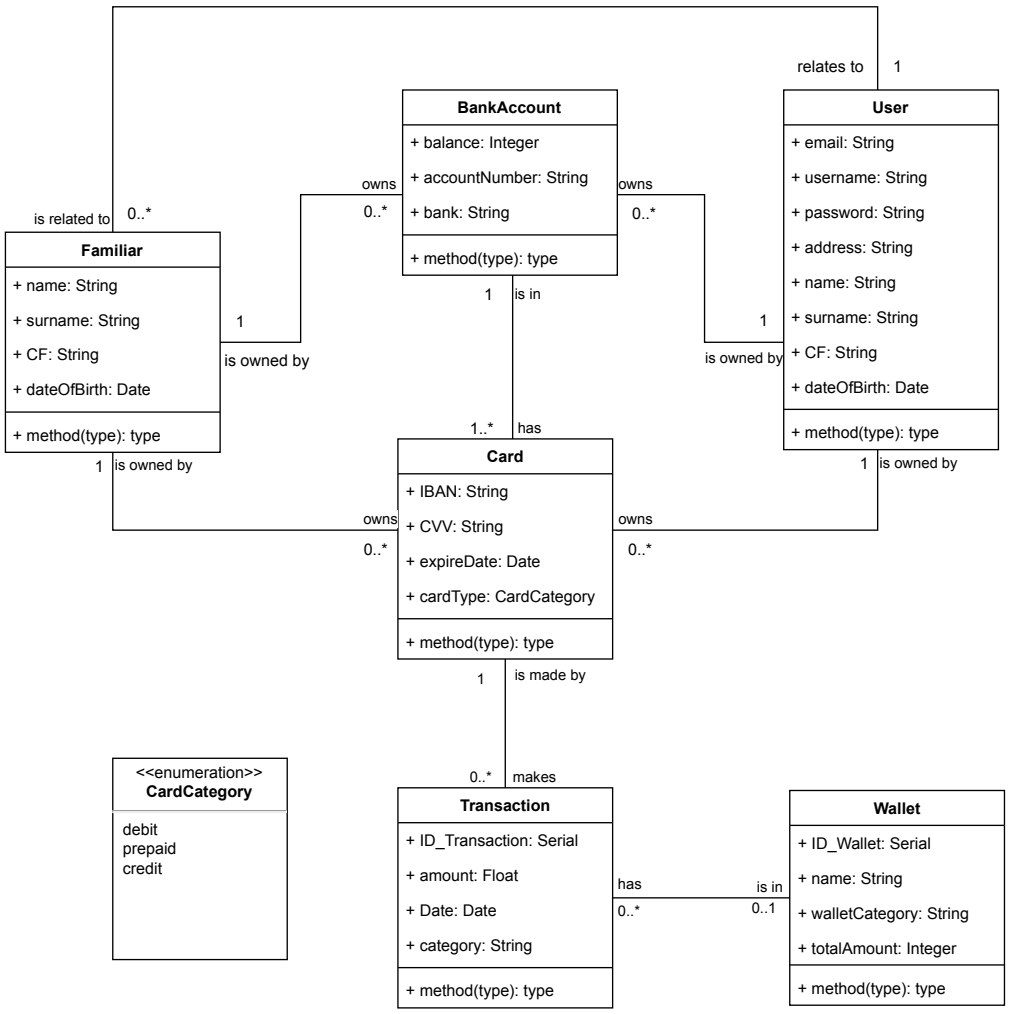

Figura 2.3: Diagramma UML Ristrutturato

2.4 Dizionari

2.4.1 Dizionario delle classi

Classe	Descrizione	Attributi
User	Classe utilizzata per identificare gli utenti registrati alla piattaforma.	**email** (*String*): email con la quale l'utente si è registrato. **username** (*String*): chiave primaria, identificativa dell'utente. È anche il 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** (*CardCategory*): 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] (is related to)**: indica, per ogni familiare, con quale utente è imparentato. **User [0..*] (relates to)**: indica quali sono i familiari che sono imparentati con esso.
To Be Related	Esprime la parentela tra gli utenti e i familiari	**Familiar [1] (is related to)**: indica, per ogni familiare, con quale utente è imparentato. **User [0..*] (relates to)**: indica quali sono i familiari che sono imparentati con esso.
To Be Related	Esprime la parentela tra gli utenti e i familiari	**Familiar [1] (is related to)**: indica, per ogni familiare, con quale utente è imparentato. **User [0..*] (relates to)**: indica quali sono i familiari che sono imparentati con esso.
To Be Related	Esprime la parentela tra gli utenti e i familiari	**Familiar [1] (is related to)**: indica, per ogni familiare, con quale utente è imparentato. **User [0..*] (relates to)**: indica quali sono i familiari che sono imparentati con esso.
To Be Related	Esprime la parentela tra gli utenti e i familiari	**Familiar [1] (is related to)**: indica, per ogni familiare, con quale utente è imparentato. **User [0..*] (relates to)**: indica quali sono i familiari che sono imparentati con esso.
To Be Related	Esprime la parentela tra gli utenti e i familiari	**Familiar [1] (is related to)**: indica, per ogni familiare, con quale utente è imparentato. **User [0..*] (relates to)**: indica quali sono i familiari che sono imparentati con esso.

Associazione	Descrizione	Classi coinvolte
To Be Related	Esprime la parentela tra gli utenti e i familiari	**Familiar** [**1**] (**is related to**): indica, per ogni familiare, con quale utente è imparentato. **User** [**0..***] (**relates to**): indica quali sono i familiari che sono imparentati con esso.
To Be Related	Esprime la parentela tra gli utenti e i familiari	**Familiar** [**1**] (**is related to**): indica, per ogni familiare, con quale utente è imparentato. **User** [**0..***] (**relates to**): indica quali sono i familiari che sono imparentati con esso.

2.4.3 Dizionario dei vincoli

Vincolo	Tipo	Descrizione
name	type	description

Capitolo 3

Progettazione Logica

3.1 Schema Logico

3.1.1 Traduzione delle classi

3.1.2 Traduzione delle associazioni

3.1.3 Schema logico definitivo

Capitolo 4

Schema Fisico

4.1 Definioni SQL delle tabelle