Progetto di Tecnologie Informatiche per il Web

Negri Riccardo

22 maggio 2022

POLITECNICO
MILANO 1863

Docente: Fraternali Piero
Studente: Negri Riccardo 10729927 936820

Indice

1 Specifiche

1.1 Versione pure HTML

Un'applicazione web consente la gestione di trasferimenti di denaro online da un conto a un altro. L'applicazione supporta registrazione e login mediante una pagina pubblica con opportune form. La registrazione controlla la validità sintattica dell'indirizzo di email e l'uguaglianza tra i campi "password" e "ripeti password". La registrazione controlla l'unicità dello username. Un utente ha un nome, un cognome, uno username e uno o più conti correnti. Un conto ha un codice, un saldo, e i trasferimenti fatti (in uscita) e ricevuti (in ingresso) dal conto. Un trasferimento ha una data, un importo, un conto di origine e un conto di destinazione. Quando l'utente accede all'applicazione appare una pagina LOGIN per la verifica delle credenziali. In seguito all'autenticazione dell'utente appare l'HOME page che mostra l'elenco dei suoi conti. Quando l'utente seleziona un conto, appare una pagina STATO DEL CONTO che mostra i dettagli del conto e la lista dei movimenti in entrata e in uscita, ordinati per data discendente. La pagina contiene anche una form per ordinare un trasferimento. La form contiene i campi: codice utente destinatario, codice conto destinatario, causale e importo. All'invio della form con il bottone INVIA, l'applicazione controlla che il conto di destinazione appartenga all'utente specificato e che il conto origine abbia un saldo superiore o uguale all'importo del trasferimento. In caso di mancanza di anche solo una condizione, l'applicazione mostra una pagina con un avviso di fallimento che spiega il motivo del mancato trasferimento. Nel caso in cui entrambe le condizioni siano soddisfatte, l'applicazione deduce l'importo dal conto di origine, aggiunge l'importo al conto di destinazione e mostra una pagina CONFERMA TRASFERIMENTO che presenta i dati dell'importo trasferito e i dati del conto di origine e di destinazione con i rispettivi saldi precedenti al trasferimento e aggiornati dopo il trasferimento. L'applicazione deve garantire l'atomicità del trasferimento: ogni volta che il conto di destinazione viene addebitato, il conto di origine deve essere accreditato. Ogni pagina contiene un collegamento per tornare alla pagina precedente. L'applicazione consente il logout dell'utente.

1.2 Versione RIA

Si realizzi un'applicazione client server web che modifica le specifiche precedenti come segue:

- La registrazione controlla la validità sintattica dell'indirizzo di email e l'uguaglianza tra i campi "password" e "ripeti password", anche a lato client.
- Dopo il login, l'intera applicazione è realizzata con un'unica pagina.
- Ogni interazione dell'utente è gestita senza ricaricare completamente la pagina, ma produce l'invocazione asincrona del server e l'eventuale modifica del contenuto da aggiornare a seguito dell'evento.
- I controlli di validità dei dati di input (ad esempio importo non nullo e maggiore di zero) devono essere realizzati anche a lato client.
- L'avviso di fallimento è realizzato mediante un messaggio nella pagina che ospita l'applicazione.
- L'applicazione chiede all'utente se vuole inserire nella propria rubrica i dati del destinatario di un trasferimento andato a buon fine non ancora presente. Se l'utente conferma, i dati sono memorizzati nella base di dati e usati per semplificare l'inserimento. Quando l'utente crea un trasferimento, l'applicazione propone mediante una funzione di auto-completamento i destinatari in rubrica il cui codice corrisponde alle lettere inserite nel campo codice utente destinatario.

2 Database design

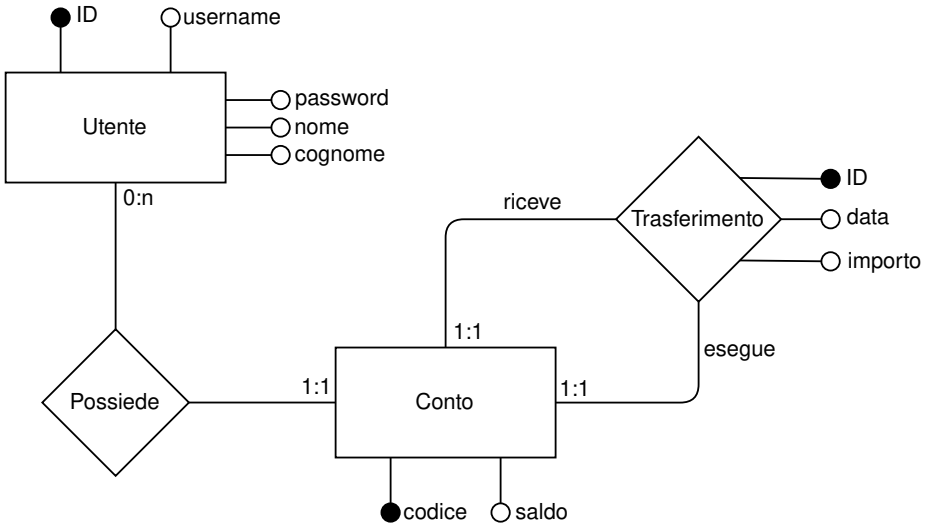
2.1 Analisi testo delle specifiche

Legenda: entità, attributi, relazioni.

Un'applicazione web consente la gestione di trasferimenti di denaro online da un conto a un altro. L'applicazione supporta registrazione e login mediante una pagina pubblica con opportune form. La registrazione controlla la validità sintattica dell'indirizzo di email e l'uguaglianza tra i campi "password" e "ripeti password". La registrazione controlla l'unicità dello username. Un utente ha un nome, un cognome, uno username e uno o più conti correnti. Un conto ha un codice, un saldo, e i trasferimenti fatti (in uscita) e ricevuti (in ingresso) dal conto. Un trasferimento ha una data, un importo, un conto di origine e un conto di destinazione. Quando l'utente accede all'applicazione appare una pagina LOGIN per la verifica delle credenziali. In seguito all'autenticazione dell'utente appare l'HOME page che mostra l'elenco dei suoi conti. Quando l'utente seleziona un conto, appare una pagina STATO DEL CONTO che mostra i dettagli del conto e la lista dei movimenti in entrata e in uscita, ordinati per data discendente. La pagina contiene anche una form per ordinare un trasferimento. La form contiene i campi: codice utente destinatario, codice conto destinatario, causale e importo. All'invio della form con il bottone INVIA, l'applicazione controlla che il conto di destinazione appartenga all'utente specificato e che il conto origine abbia un saldo superiore o uguale all'importo del trasferimento. In caso di mancanza di anche solo una condizione, l'applicazione mostra una pagina con un avviso di fallimento che spiega il motivo del mancato trasferimento. Nel caso in cui entrambe le condizioni siano soddisfatte, l'applicazione deduce l'importo dal conto di origine, aggiunge l'importo al conto di destinazione e mostra una pagina CONFERMA TRASFERIMENTO che presenta i dati dell'importo trasferito e i dati del conto di origine e di destinazione con i rispettivi saldi precedenti al trasferimento e aggiornati dopo il trasferimento. L'applicazione deve garantire l'atomicità del trasferimento: ogni volta che il conto di destinazione viene addebitato, il conto di origine deve essere accreditato. Ogni pagina contiene un collegamento per tornare alla pagina precedente. L'applicazione consente il logout dell'utente.

2.2 Progetto concettuale - Diagramma E-R (Entità-Relazione)

Da aggiornare



2.3 Progetto logico

UTENTE(ID, username, password, nome, cognome)
CONTO(ID, codice, saldo, utente)
TRASFERIMENTO(ID, data, importo, causale, origine, destinazione)

2.4 Database schema

Statement per creare il database:

```
CREATE DATABASE tiw_db;
```

Statements per creare le tabelle:

```
CREATE TABLE `tiw_db`.`utente` (
        `id` INT NOT NULL AUTO_INCREMENT,
        `username` VARCHAR(45) NOT NULL,
        `password` VARCHAR(45) NOT NULL,
        `nome` VARCHAR(45) NOT NULL,
        `cognome` VARCHAR(45) NOT NULL,
        PRIMARY KEY (`id`)
);
```

```
CREATE TABLE `tiw_db`.`conto` (
        `id` INT NOT NULL AUTO_INCREMENT,
        `codice` INT NOT NULL,
        `saldo` INT NOT NULL DEFAULT 0,
        `utente` INT NOT NULL,
        PRIMARY KEY (`codice`),
        CONSTRAINT `utente_conto` FOREIGN KEY (`utente`)
                REFERENCES `tiw_db`.`utente` (`id`)
                        ON DELETE NO ACTION ON UPDATE CASCADE
);

CREATE TABLE `tiw_db`.`trasferimento` (
        `id` INT NOT NULL AUTO_INCREMENT,
        `data` DATE NOT NULL,
        `importo` INT NOT NULL,
        `causale` VARCHAR(255) NOT NULL,
        `origine` INT NOT NULL,
        `destinazione` INT NOT NULL,
        PRIMARY KEY (`id`),
        CONSTRAINT `trasferimento_origine` FOREIGN KEY (`origine`)
                REFERENCES `tiw_db`.`conto` (`id`)
                        ON DELETE NO ACTION ON UPDATE CASCADE,
        CONSTRAINT `trasferimento_destinazione` FOREIGN KEY (`destinazione`)
                REFERENCES `tiw_db`.`conto` (`id`)
                        ON DELETE NO ACTION ON UPDATE CASCADE
);
```

2.5 Script Python per popolare il database

```
import mysql.connector
import random

USERS_NUMBER = 10
MAX_ACCOUNTS_PER_USER = 10
MAX_MONEY = 1000000
MAX_TRANSACTIONS_PER_ACCOUNT = 10
ACCOUNT_NUMBER_LENGTH = 12
NAMES = ["Francesco", "Sofia", "Alessandro", "Giulia", "Andrea",
"Aurora", "Lorenzo", "Emma", "Mattia", "Giorgia"]
SURNAMES = ["Rossi", "Russo", "Ferrari", "Esposito", "Bianchi", "Colombo", "Ricci", "Gallo",
DEFAULT_PASSWORD = "pass"

mydb = mysql.connector.connect(
host="localhost",
user="admin",
password="password",
```

```
database="tiw_db"
)

mycursor = mydb.cursor()

sql_u = "INSERT INTO utente (username, password, nome, cognome) VALUES (%s, %s, %s, %s)"
sql_c = "INSERT INTO conto (codice, saldo, utente) VALUES (%s, %s, %s)"
for _ in range(0, USERS_NUMBER):
name = random.choice(NAMES)
surname = random.choice(SURNAMES)
username = (name + surname).lower()
password = DEFAULT_PASSWORD
try:
mycursor.execute(sql_u, (username, password, name, surname))
except mysql.connector.errors.IntegrityError:
print("GERE")
continue
mydb.commit()
for _ in range(0, random.randint(0, MAX_ACCOUNTS_PER_USER)):
account_number = '{0:05}'.format(random.randint(1, 100000))
money = random.randint(0, MAX_MONEY)
try:
mycursor.execute(sql_c, (account_number, money, username))
except mysql.connector.errors.IntegrityError:
print("GERE")
continue
mydb.commit()

#mycursor.executemany(sql_u, val_u)

#mydb.commit()

print(mycursor.rowcount, "was insserted.")
```

3 Versione pure HTML

4 Versione RIA