Documentazione Database “GLITCH”

Progetto presentato da:

Annunziata Elefante

Ferdinando Napolitano

Santolo Mutone

Docente:

Prof. Andrea De Lucia

INDICE

1. Introduzione

1.1. Scopo del documento

Questo documento si concentra sull'analisi del Database utilizzato all'interno del sistema Glitch in tutti i suoi aspetti. Tale documento, infatti, ha l'obiettivo di mostrare tutte le scelte logiche e fisiche alla base dell'utilizzo di un DBMS.

Ogni scelta verte sui bisogni principali del sistema, quali:

- tenere in memoria i dati personali degli utenti;
- mantenere informazioni sui metodi di pagamento degli utenti;
- mantenere informazioni sull'assortimento di prodotti;
- fornire dettagli sugli ordini;
- fornire ad ogni accesso i dati del carrello personale di ogni utente;
- mantenere le informazioni sulle offerte applicate.

1.2. Definizioni, acronimi e abbreviazioni

- **Database relazionale:** Un database relazionale è una raccolta di elementi dati con relazioni predefinite tra di essi. Questi elementi sono organizzati sotto forma di set di tabelle con righe e colonne. Le tabelle vengono usate per contenere le informazioni sugli oggetti da rappresentare nel database. Ogni colonna in una tabella contiene un determinato tipo di dati e un campo archivia il valore effettivo di un attributo. Le righe rappresentano una raccolta di valori correlati di un oggetto o entità. Ogni riga può essere contrassegnata con un identificatore univoco denominato chiave principale; le righe su diverse tabelle possono essere correlate utilizzando chiavi esterne. È possibile accedere a questi dati in molti modi diversi, senza riorganizzare le tabelle di database.
- **SDD:** System Design Document
- **RAD:** Object Design Document
- **CRUD:** create, read, update e delete

1.3. Panoramica

Il seguente documento DB è diviso in sezioni ed ha la seguente composizione:

- Sezione di *INTRODUZIONE*: vi è presente una descrizione dell'esigenza da cui è scaturita l'idea di utilizzare un database relazionale per il nostro sistema. Ne segue la presenza di un elenco di definizioni, acronimi e abbreviazioni usato per facilitare la comprensione dei concetti citati al lettore

- Sezione sul *DATABASE CORRENTE*: mostra com'è la realtà attuale del database da utilizzare/rimpiazzare o, nel nostro caso, sviluppare.

- Sezione sul *DATABASE PROPOSTO*: in primo luogo si ha una panoramica sull'idea di base di come il database dovrebbe essere, accompagnata dalla rappresentazione grafica del suo "schema logico". Punto importante sono l'analisi dei singoli oggetti, mediante la loro rappresentazione in tabella, e l'elenco delle query da poter richiedere al sistema per rispondere al meglio ai requisiti dell'utente.

2. Database corrente

Trattandosi di un'implementazione di tipo "Greenfield Engineering", non abbiamo a disposizione un Database preesistente. Proprio per questo la scelta del DB verte esclusivamente sulle necessità del sistema proposto.

3. Database proposto

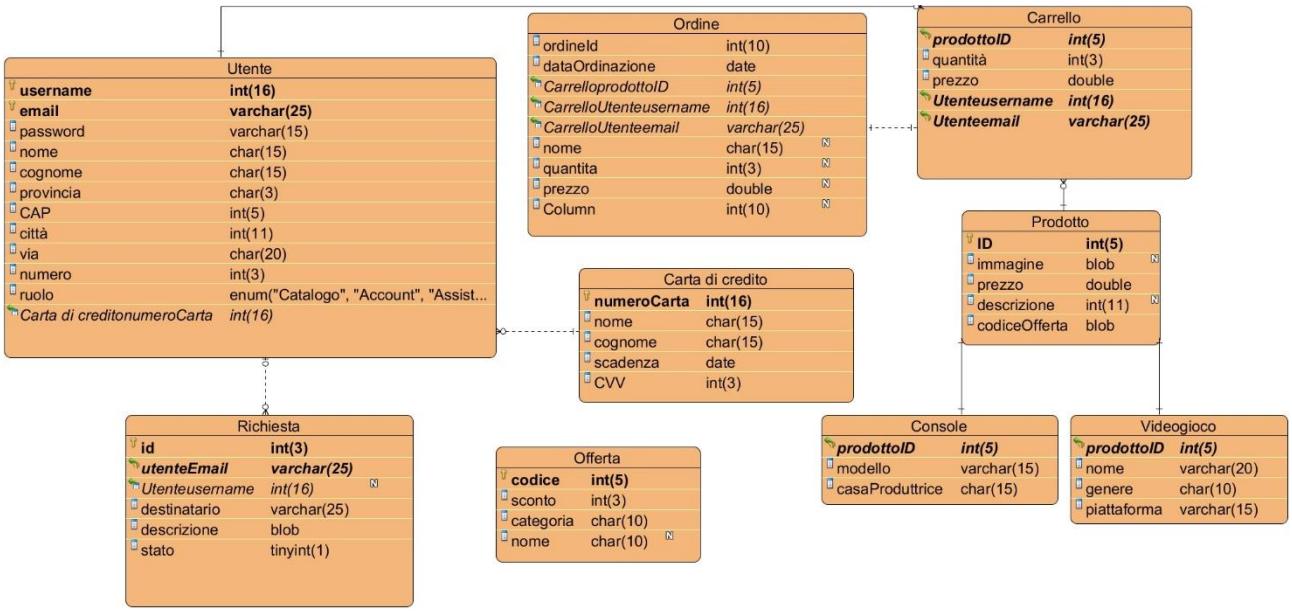
3.1. Panoramica

Le scelte sul DB proposto vengono dalla lettura dell'SDD (vedi SDD_Glitch), in particolare della sezione dedicata alla "Gestione dei dati persistenti", dove vengono mostrati tutti i dati che devono essere mantenuti in memoria dal sistema. Si prosegue, poi, con l'analisi dell'ODD (vedi ODD_Glitch), che, concentrandosi sulla parte di implementazione, ci permette di capire come formulare le query sul DB proposto secondo il linguaggio di programmazione **MySQL**.

Tale documento rappresenta un punto di partenza importante sulla gestione dei dati persistenti, in quanto determina le scelte logiche e fisiche da considerare per documentare il DBMS.

3.2. Schema logico

Offre una visione logica delle relazioni che intercorrono tra le varie informazioni che saranno rese persistenti all'interno del sistema.



3.2.1. Utente

Nome	Tipo	Null	Key
username	varchar(16)	not null	primary key

e-mail	varchar(25)	not null	primary key
password	varchar(15)	not null	
nome	char(15)	not null	
cognome	char(15)	not null	
ruolo	enum(“Catalogo”, “Account”, “Assistenza”)	nullable	
provincia	char(3)	not null	
CAP	int(5)	not null	
città	char(20)	not null	
via	char(20)	not null	
numero	int(3)	not null	
CartaDiCreditoNumeroCarta	int(16)		

3.2.2. Richiesta

Nome	Tipo	Null	Key
id	int(3)	not null	primary key
utenteUsername	varchar(10)	not null	external key
utenteEmail	varchar(25)	not null	external key
destinatario	varchar(25)	not null	
descrizione	blob	not null	
stato	tinyint(1)	not null	

3.2.3. Ordine

Nome	Tipo	Null	Key
ordineId	int(10)	not null	
carrelloUtenteUsername	varchar(10)	not null	external key
carrelloUtenteEmail	varchar(25)	not null	external key
carrelloProdottoID	int(5)	not null	external key
dataOrdinazione	date	not null	
nome	char(15)	nullable	
quantita	int(3)	nullable	
prezzo	double	nullable	

3.2.4. Offerta

Nome	Tipo	Null	Key
codice	int(5)	not null	primary key
sconto	int(3)	not null	
categoria	char(10)	not null	
nome	char(10)	nullable	

3.2.5. Carrello

Nome	Tipo	Null	Key
prodottoID	int(5)	not null	external key
utenteUsername	varchar(10)	not null	external key
utenteEmail	varchar(25)	not null	external key
quantità	int(3)	not null	
prezzoQuantità	double	not null	

3.2.6. Prodotto

Nome	Tipo	Null	Key
ID	int(5)	not null	primary key
immagine	blob	nullable	
prezzo	double	not null	
descrizione	blob	nullable	
codiceOfferta	int(5)	not null	

3.2.7. Console

Nome	Tipo	Null	Key
prodottoID	int(5)	not null	external key
modello	varchar(15)	not null	
casaProduttrice	int(5)	not null	

3.2.8. Videogioco

Nome	Tipo	Null	Key

prodottoID	int(5)	not null	external key
nome	varchar(20)	not null	
genere	char(10)	not null	
piattaforma	varchar(15)	not null	

3.2.9. Carta di credito

Nome	Tipo	Null	Key
numeroCarta	int(16)	not null	primary key
utenteUsername	varchar(10)	not null	external key
utenteEmail	varchar(25)	not null	external key
nome	char(15)	not null	
cognome	char(15)	not null	
scadenza	date	not null	
CVV	int(3)	not null	

3.3. Query sul database

Glitch, grazie al supporto del database relazionale, permette di offrire una serie di funzioni all'utente, supportato da apposite "query".

Per ogni oggetto dello schema (presentato al punto 3.2.) sono previste le basilari query di **CRUD**, ma anche query più complesse come quelle per ricercare i prodotti pur inserendo solo parte del nome, o ricercare i vari prodotti per id,…

La struttura utilizzata per la definizione delle query segue le regole standard del linguaggio SQL, partendo dalla semplice query di selezione:

```
SELECT * FROM glitch.utente;
```