

PART PICKER

ELABORATO PER IL CORSO DI BASI DI
DATI

A.A 2023/2024

JIEKAI SUN

jiekai.sun@studio.unibo.it

0001077598

Indice

Analisi dei requisiti	3
Intervista	3
Estrazione dei concetti principali	4
Progettazione concettuale	7
Schema scheletro	7
Schema finale	11
Progettazione logica	12
Stima del volume dei dati	12
Descrizione delle operazioni principali e stima della loro frequenza	13
Schemi di navigazione e tabelle degli accessi	13
Raffinamento dello schema	17
Schema raffinato	18
Eliminazione degli identificatori esterni	19
Analisi delle ridondanze	20
Traduzione di entità e associazioni in relazioni	22
Schema relazionale finale	23
Traduzione delle operazioni in query SQL	24
Progettazione dell'applicazione	28
Descrizione dell'architettura dell'applicazione realizzata.....	28

Analisi dei requisiti

Si vuole realizzare un database a supporto di un'applicazione per la creazione e condivisione di configurazioni di PC tra utenti appassionati di hardware.

Intervista

Si richiede di poter tracciare gli utenti registrati al sito memorizzandone le credenziali, la data di registrazione e l'e-mail.

Una volta registrati, gli utenti possono creare configurazioni di PC caratterizzate da una descrizione, data di pubblicazione e tutti i componenti utilizzati.

Gli utenti possono anche recensire configurazioni di PC create da altri utenti. Tali recensioni, effettuabili solo una volta ma modificabili, comprendono un rating tra 1 e 10, un eventuale commento e la data di modifica.

La media del rating di tutte le recensioni costituisce il rating di un utente.

Esiste una categoria particolare di utente denominata moderatore, il quale può effettuare tutte le operazioni di un normale utente oltre alla possibilità di bannare altri utenti qualora ritenuto necessario.

Un ban è caratterizzato da una motivazione, una data di emissione e un eventuale data di fine. Fintanto che un utente risulta bannato, esso non potrà usufruire dell'applicazione.

Una configurazione deve comprendere tutti i componenti fondamentali di un PC: CPU, GPU, RAM, STORAGE, MOTHERBOARD, PSU, COOLER e CASE.

Ogni componente è caratterizzato dal nome del modello, anno di lancio e prezzo di listino, ed è inoltre legato al suo produttore, del quale si vogliono salvare il nome e il paese.

Nella seguente tabella si esprimono le specifiche che si vogliono memorizzare di ogni componente:

CPU	Famiglia, numero di core, frequenza, TDP, SMT
GPU	Famiglia, tipo di memoria, quantità di memoria, frequenza, TGP
RAM	Frequenza, capienza, latenza, ECC
STORAGE	Capienza, tipologia (HDD, SSD), rpm, quantità di cache
MOTHERBOARD	Fattore di forma, nome del chipset, numero di slot di RAM, numero di slot per GPU, WiFi integrato
PSU	Fattore di forma, efficienza, wattaggio, modularità
COOLER	Rpm, livello di rumorosità, tipologia (ad aria, AIO)
CASE	Fattore di forma

È inoltre possibile specificare molteplici RAM, GPU e STORAGE per una singola build.

Per evitare che vengano create configurazioni con componenti incompatibili è necessario che una volta selezionata una CPU vengano mostrate solo le schede madri che utilizzano lo stesso socket, così come una scheda madre è compatibile solo con RAM che condividono la stessa generazione. Infine, una CPU è compatibile con una o più generazioni di RAM.

L'applicazione richiede la possibilità di storicizzare il prezzo negli ultimi 15 giorni di ogni componente attraverso il rilevamento quotidiano del prezzo da ogni rivenditore. Questi dati verranno in seguito utilizzati per mostrare all'utente le statistiche dei prezzi nel tempo.

Estrazione dei concetti principali

Termine	Breve descrizione	Eventuali sinonimi
Utente	Colui che è registrato all'applicazione, crea e recensisce le configurazioni di altri utenti	user
Moderatore	Un utente dotato di poteri di ban di altri utenti	
Recensione	Una valutazione di una configurazione comprendente voto ed eventuale commento	review
Ban	Una condizione che non permette all'utente di usufruire dell'applicazione	sospensione, blocco
Componente	Un'unità di hardware caratterizzata da nome del modello e il prezzo di listino	pezzo, parte
Build	Un aggregato di componenti	configurazione
Produttore	Colui che produce il componente	Azienda produttrice, casa produttrice
Rivenditore	Colui che vende il componente	
Prezzo rilevato	Il prezzo di un componente registrato in un dato momento da un rivenditore	

A seguito della lettura e comprensione dei requisiti, si procede redigendo un testo che ne riassume tutti i concetti e in particolare ne estragga quelli principali eliminando le ambiguità sopra rilevate:

Per ogni **utente** vengono memorizzati username, password, data di registrazione ed email. Esiste una categoria particolare di utente denominata **moderatore**, il quale detiene il potere di bannare gli altri utenti, bloccandoli dall'accesso all'app.

Una volta emesso il **ban**, di questo viene memorizzata la data di inizio, la data di fine e una descrizione. Un ban senza data di fine è da considerarsi permanente.

Ogni utente ha la possibilità di pubblicare le **build**, delle quali si vuole memorizzare la descrizione, i componenti utilizzati e l'ultima data di modifica.

Ogni utente ha inoltre la possibilità di pubblicare **recensioni** di una build creata da altri utenti esprimendo un rating da 1 a 10 con eventualmente un commento.

Il rating di un utente è dato dalla media dei rating ottenuti da tutte le sue build.

Una build è obbligatoriamente composta da tutti i **componenti** di un pc, ovvero: CPU, GPU, RAM, STORAGE, MOTHERBOARD, PSU, COOLER, CASE.

È inoltre possibile specificare molteplici RAM, GPU e STORAGE per una singola build.

Di un componente si memorizzano il codice univoco, il nome, l'anno di lancio e prezzo di listino. Ogni componente è inoltre caratterizzato da aspetti specifici non generalizzabili, ovvero le cosiddette "specifiche".

Si vogliono anche introdurre dei vincoli di compatibilità tra alcuni componenti:

- Una CPU ed una MOTHERBOARD sono compatibili se e solo se condividono lo stesso **socket**.
- Una MOTHERBOARD ed una RAM sono compatibili se e solo se la prima supporta la **generazione di RAM** dell'ultima.
- Una CPU è compatibile con una o più generazioni di RAM.

Inoltre, ogni componente è creato da un **produttore** di cui si vuole memorizzare il nome e paese. Infine, si vogliono tracciare i prezzi giornalieri di ogni componente dai vari **rivenditori**, dei quali si vuole memorizzare il nome.

Segue un elenco delle principali azioni richieste:

1. Registrazione di un nuovo utente
2. Registrazione del un ban di un utente
3. Login di un utente
4. Inserimento di una build
5. Inserimento di una recensione
6. Modifica di una recensione
7. Inserimento di un componente
8. Rilevamento del prezzo di un componente
9. Visualizzazione del prezzo più recente e basso di un componente
10. Visualizzazione dei dettagli di un componente
11. Visualizzazione dello storico dei prezzi di un componente
12. Visualizzazione del rating medio di un utente

Progettazione concettuale

Schema scheletro

L'entità **Moderatore** è modellata come subset di **Utente**, in tal modo esso condivide le stesse caratteristiche dell'utente, tranne la capacità di bannare altri utenti. Reificando il concetto di ban, e quindi usando come identificatore l'assegnatario e la data di inizio, si introduce la possibilità di assegnare molteplici ban ad un utente e la possibilità di storicizzarli.

Un vincolo inespresso è il fatto che un moderatore non possa bannare un altro moderatore, e che quest'ultimo non possa bannare se stesso.

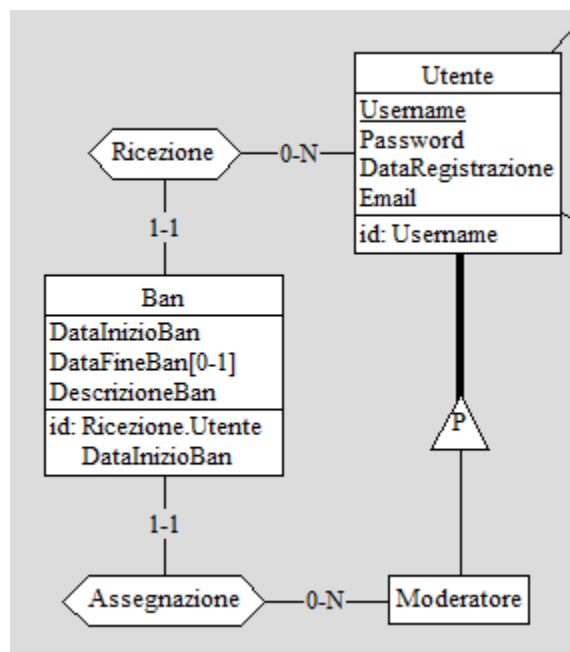


Fig 1.1 Schema E/R con le principali entità per la modellazione dei ban

Poiché un utente può recensire tutte le build create da altri utenti una sola volta con eventuali modifiche, ho ritenuto sufficiente l'associazione **Recensione**. Resta comunque il vincolo inespresso che un utente non può recensire una propria build.

Per quanto riguarda la pubblicazione di una build, dall'analisi si evince che il pubblicatore è sempre un singolo utente, e quindi ho ritenuto adatto l'utilizzo di un'associazione 1-n, **Pubblicazione**, tra le due entità.

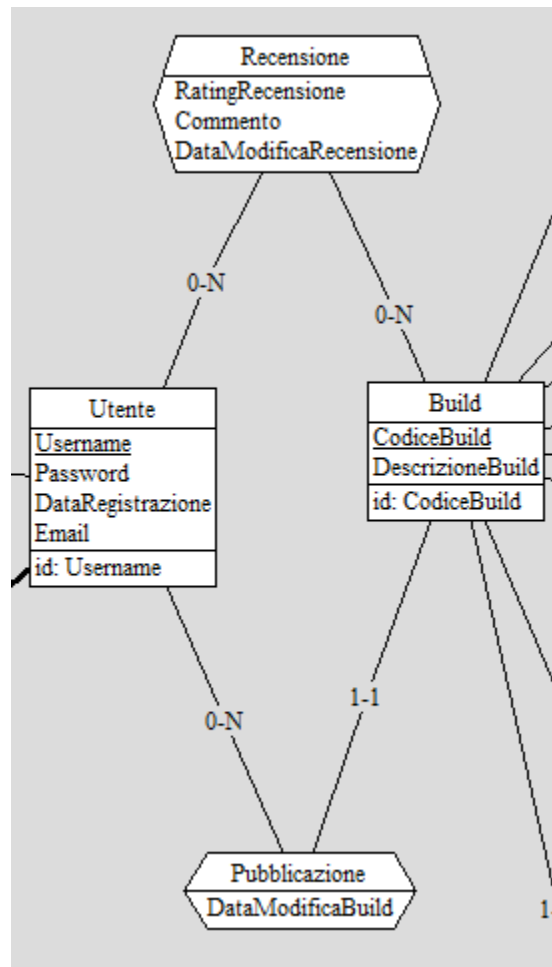


Fig 1.2 Schema E/R rappresentante il sistema di pubblicazione di build e la loro recensione

Una **Build** comprende obbligatoriamente tutti i componenti di un PC, per questo motivo ho ritenuto necessario gerarchizzare **Componente**, attribuendogli nel caso generico un codice univoco, il nome, l'anno di lancio e il suo prezzo di listino. Nel caso specifico invece, vengono espresse tutte le informazioni riguardanti ogni singola categoria di componente. Inoltre, sono reputate necessarie entità aggiuntive per modellare le compatibilità intercomponente, come ad esempio tra CPU e MOTHERBOARD via l'entità **Socket**, mentre nel caso di compatibilità tra MOTHERBOARD e RAM si utilizza l'entità **GenerazioneRam**. La compatibilità tra CPU e RAM è espressa attraverso una associazione n-n in quanto vi sono alcune CPU che supportano più generazioni di RAM.

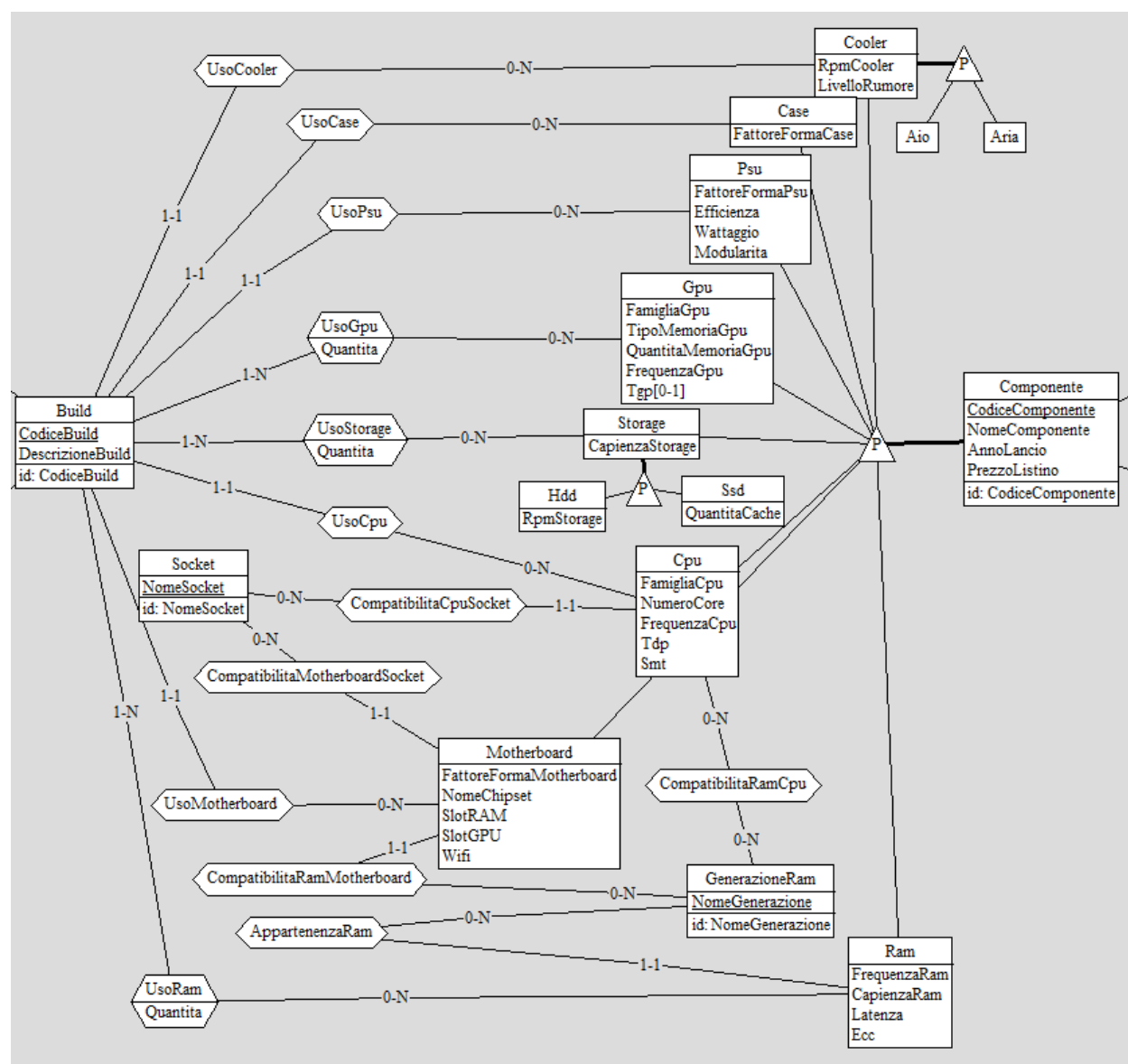


Fig. 1.3 Schema E/R con le principali entità per la modellazione dei componenti di un PC, il loro utilizzo in una build e i vincoli che intercorrono tra alcuni di essi

Per la storicizzazione giornaliera dei prezzi di un componente da specifici rivenditori è stata creata l'entità **PrezzoComponente** la quale importa la chiave di Componente e quella di Rivenditore per combinarla con la data di rilevamento.

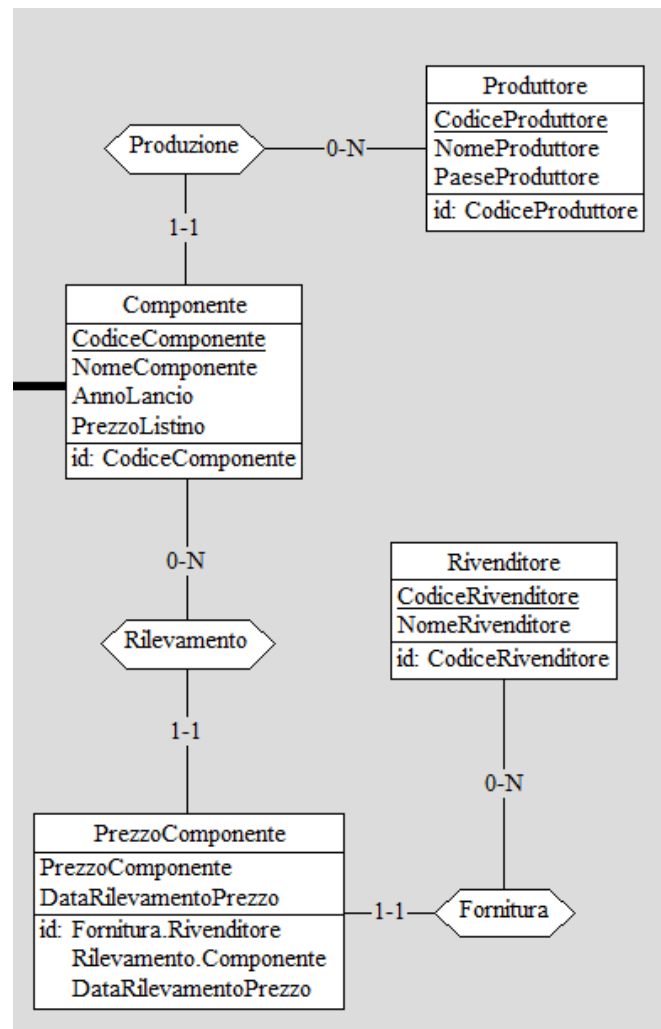
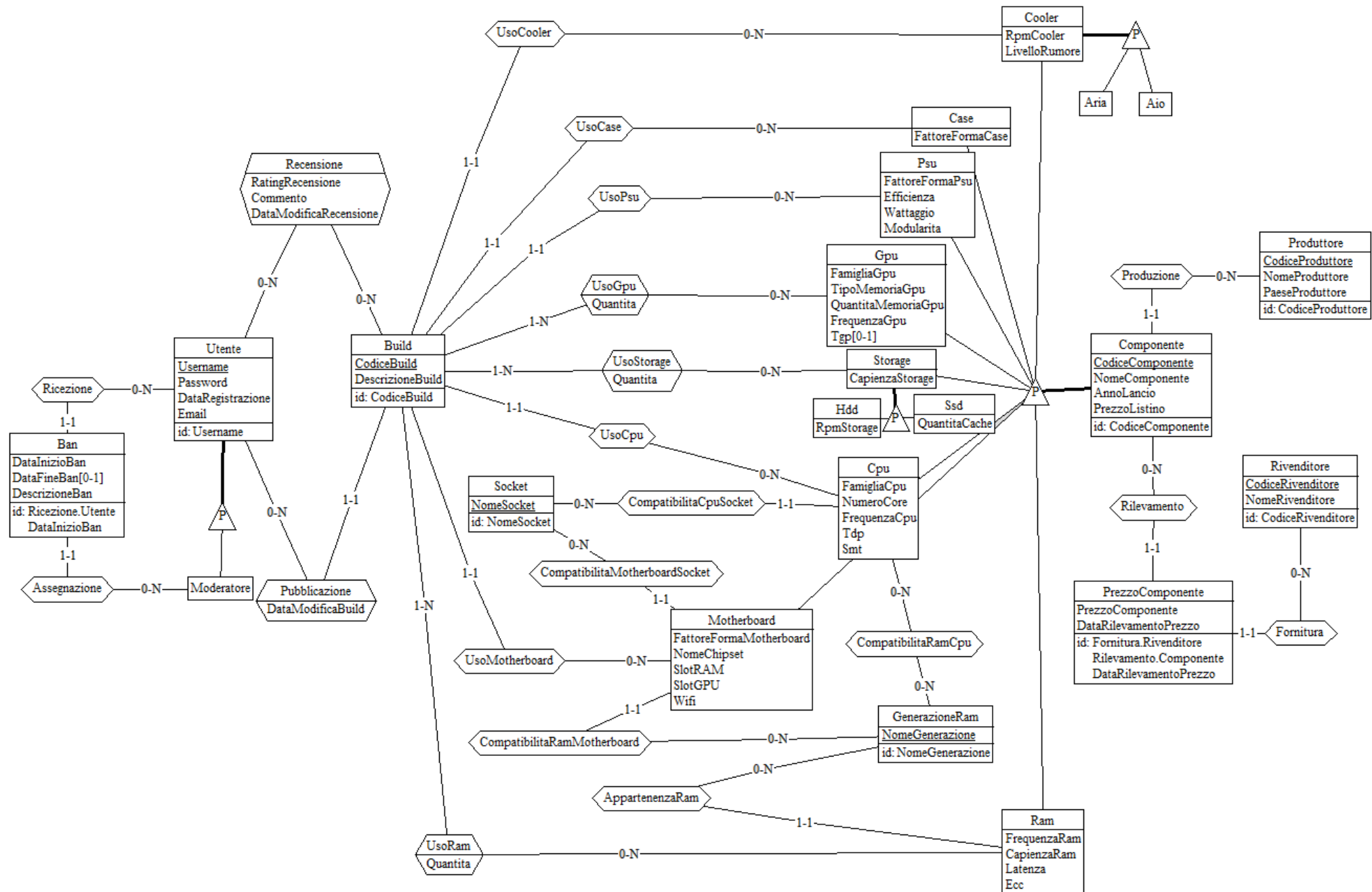


Fig 1.4 Schema E/R con le principali entità per la storicizzazione dei prezzi

Schema finale



Progettazione logica

Stima del volume dei dati

Concetto	Costrutto	Volume	Concetto	Costrutto	Volume
Utente	E	10.000	Produttore	E	100
Moderatore	E	5	Produzione	R	10.000
Ban	E	1.000	PrezzoComponente	E	300.000
Assegnazione	R	1.000	Rilevamento	R	300.000
Ricezione	R	1.000	Rivenditore	E	2
			Fornitura	R	300.000
Build	E	30.000			
Pubblicazione	R	30.000	Socket	E	5
Recensione	R	120.000	CompatibilitàCpuSocket	R	1.500
UsoCooler	R	30.000	CompatibilitàMotherboardSocket	R	1.000
UsoCase	R	30.000			
UsoPsu	R	30.000	GenerazioneRam	E	3
UsoGpu	R	40.000	CompatibilitàRamMotherboard	R	1.000
UsoStorage	R	40.000	AppartenenzaRam	R	1.000
UsoCpu	R	30.000	CompatibilitàRamCpu	R	2.000
UsoMotherboard	R	30.000			
UsoRam	R	40.000			
Componente	E	10.000			
Cooler	E	1.000			
Aria	E	500			
Aio	E	500			
Case	E	1.000			
Psu	E	1.000			
Gpu	E	2.000			
Storage	E	1.500			
Hdd	E	750			
Ssd	E	750			
Cpu	E	1.500			
Motherboard	E	1.000			
Ram	E	1.000			

Descrizione delle operazioni principali e stima della loro frequenza

Le operazioni da effettuare sono quelle già elencate nella fase di analisi. Segue una tabella riportante la loro descrizione e relativa frequenza.

Codice	Operazione	Frequenza
1	Registrazione di un nuovo utente	5 al giorno
2	Registrazione del ban di un utente	1 al giorno
3	Login di un utente	100 al giorno
4	Inserimento di una build	25 al giorno
5	Inserimento di una recensione di una build	100 al giorno
6	Modifica di una recensione	5 al giorno
7	Inserimento di un componente	5 al mese
8	Rilevamento del prezzo di un componente	10.000 al giorno
9	Visualizzazione del prezzo più recente e basso di un componente	1.000 al giorno
10	Visualizzazione dei dettagli di un componente	400 al giorno
11	Visualizzazione dello storico dei prezzi di un componente	200 al giorno
12	Visualizzazione del rating medio di un utente	20 al giorno

Schemi di navigazione e tabelle degli accessi

Sono riportate in seguito le tabelle degli accessi delle operazioni sopra riportate. Al fine del calcolo dei costi, si considerano di peso doppio gli accessi di scrittura rispetto a quelli in lettura.

OP1 – Registrazione di un nuovo utente

Concetto	Costrutto	Accessi	Tipo
Utente	E	1	S
Totale: 1S => 10 al giorno			

OP2 – Registrazione del ban di un utente

Concetto	Costrutto	Accessi	Tipo
Utente	E	2	L
Assegnazione	R	1	S
Ricezione	R	1	S
Ban	E	1	S
Totale: 2L + 3S => 8 al giorno			

OP3 – Login di un utente

Un login richiede la presenza dello specifico utente nel database, inoltre esso deve risultare non bannato o averne uno scaduto.

Si considera che un utente ha in media un ban.

Concetto	Costrutto	Accessi	Tipo
Utente	E	2	L
Ricezione	R	1	L
Ban	E	1	L
Totale: 4L => 400 al giorno			

OP4 – Inserimento di una build

Si considera che nella creazione di una build in media è utilizzato un singolo Storage, GPU e RAM.

Concetto	Costrutto	Accessi	Tipo
Utente	E	1	L
Pubblicazione	R	1	S
Build	E	1	S
UsoCooler	R	1	S
UsoCase	R	1	S
UsoPsu	R	1	S
UsoGpu	R	1	S
UsoStorage	R	1	S
UsoCpu	R	1	S
UsoMotherboard	R	1	S
UsoRam	R	1	S
Cooler	E	1	L
Case	E	1	L
Psu	E	1	L
Gpu	E	1	L
Storage	E	1	L
Cpu	E	1	L
Motherboard	E	1	L
Ram	E	1	L
Totale: 9L + 10S => 725 al giorno			

OP5 – Inserimento di una recensione

Concetto	Costrutto	Accessi	Tipo
Utente	E	1	L
Recensione	R	1	S
Build	E	1	L
Totale: 2L + 1S => 400 al giorno			

OP6 – Modifica di una recensione

Concetto	Costrutto	Accessi	Tipo
Utente	E	1	L
Recensione	R	1	L
Recensione	R	1	S
Build	E	1	L
Totale: 3L + 1S => 25 al giorno			

OP7 – Inserimento di un componente

Si presuppone che il produttore di tale prodotto non sia stato inserito precedentemente.

Concetto	Costrutto	Accessi	Tipo
Componente	E	1	S
Produzione	R	1	S
Produttore	E	1	S
Totale: 3S => 30 al mese			

OP8 – Rilevamento dei prezzo di un componente

Si presuppone che vi siano due rivenditori già presenti nel database

Concetto	Costrutto	Accessi	Tipo
Componente	E	1	L
Rilevamento	R	2	S
PrezzoComponente	E	2	S
Fornitura	R	2	S
Rivenditore	E	2	L
Totale: 3L + 6S => 150.000 al giorno			

OP9 – Visualizzazione del prezzo più recente e basso di un componente

Concetto	Costrutto	Accessi	Tipo
Componente	E	1	L
Rilevamento	R	2	L
PrezzoComponente	E	2	L
Totale: 5L => 5.000 al giorno			

OP10 – Visualizzazione dei dettagli di un componente

Concetto	Costrutto	Accessi	Tipo
Componente	E	1	L
Totale: 1L => 400 al giorno			

OP11 – Visualizzazione dello storico dei prezzi di un componente

Si presuppone che vengano mostrati gli ultimi 14 giorni rilevati e che i rivenditori siano due. Per cui si hanno 28 rilevazioni per componente.

Concetto	Costrutto	Accessi	Tipo
Componente	E	1	L
Rilevamento	R	28	L
PrezzoComponente	E	28	L
Fornitura	R	28	L
Rivenditore	E	28	L
Totale: 113L => 22.600 al giorno			

OP12 – Visualizzazione del rating medio di un utente

Un utente ha in media 3 build. Ogni build ha in media 4 recensioni.

Concetto	Costrutto	Accessi	Tipo
Utente	E	1	L
Pubblicazione	R	3	L
Build	E	3	L
Recensione	R	12	L
Totale: 19L -> 380 al giorno			

Raffinamento dello schema

Eliminazione delle gerarchie

Per l'eliminazione del subset Moderatore si è scelto di adottare l'approccio del collasso verso l'alto inserendo in Utente l'attributo booleano omonimo per determinarne il ruolo. Si adotta questa strategia in quanto il moderatore svolge pressoché le stesse operazioni di un utente più un'interazione particolare saltuariamente utilizzata.

Anche nel caso delle entità Cooler e Storage si decide di effettuare un collasso verso l'alto.

Per quanto riguarda la gerarchia dei componenti si è deciso di mantenere tutte le entità riguardanti le specifiche associandole all'entità Componente in quanto tale strategia esprime meglio la classificazione dei componenti, che hanno anche tra loro vincoli di compatibilità e sono utilizzati individualmente per la creazione di una build.

Essi sono legati all'entità Componente via associazioni 1-1 con il vincolo che quest'ultimo è classificabile in uno e uno solo di questi.

Nasce però il problema che una volta inserito un componente deve essere immediatamente seguito dall'inserimento delle sue specifiche.

L'inserimento di un nuovo Componente è quindi un'operazione delicata che deve essere effettuata via transazione. A tale scopo viene inserito in Componente l'attributo TipoComponente con dominio {Case, Cpu, Cooler, Gpu, Motherboard, Psu, Ram, Storage}, per facilitare i controlli di inserimento.

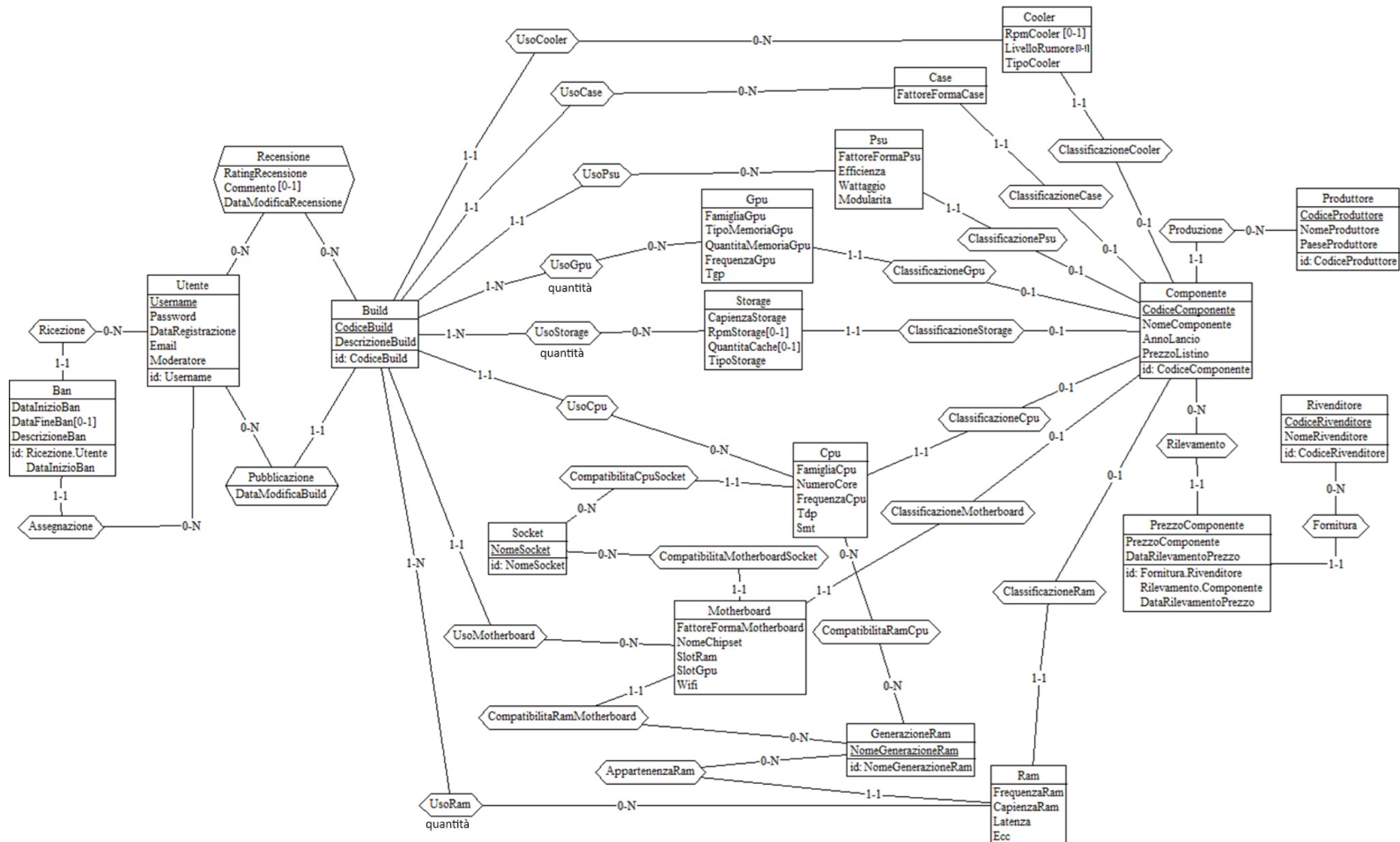
Scelta delle chiavi

Nello schema sono già evidenziate senza ambiguità tutte le chiavi primarie per la maggior parte delle entità.

Nel caso di Componente non viene importata da Produttore il codiceProduttore in quanto un singolo codice soddisfa i requisiti dell'applicazione ed evita la sua propagazione come foreign key nelle tabelle che la utilizzeranno.

Nel caso dei componenti derivati si decide di usare come chiave il codice importato da Componente. Tale rappresentazione è sottintesa nel seguente schema in quanto non rappresentabile nel tool DB-Main.

Schema raffinato



Eliminazione degli identificatori esterni

Nello schema E/R sono eliminate le seguenti relazioni:

- Ricezione, importando username in Ban
- Assegnazione, importando username in Ban
- Recensione, reificata importando username da Utente e codiceBuild da Build
- Pubblicazione, reificata importando codiceBuild da Build
- UsoCooler, importando codiceCooler in Build
- UsoCase, importando codiceCase in Build
- UsoPsu, importando codicePsu in Build
- UsoCpu, importando codiceCpu in Build
- UsoMotherboard, importando codiceMotherboard in Build
- UsoGpu, reificata importando codiceBuild e codiceGpu
- UsoStorage, reificata importando codiceBuild e codiceStorage
- UsoRam, reificata importando codiceBuild e codiceRam
- ClassificazioneCooler, importando codiceComponente in Cooler
- ClassificazioneCase, importando codiceComponente in Case
- ClassificazionePsu, importando codiceComponente in Psu
- ClassificazioneCpu, importando codiceComponente in Cpu
- ClassificazioneMotherboard, importando codiceComponente in Motherboard
- ClassificazioneGpu, importando codiceComponente in Gpu
- ClassificazioneStorage, importando codiceComponente in Storage
- ClassificazioneRam, importando codiceComponente in Ram
- CompatibilitaCpuSocket, importando nomeSocket in Cpu
- CompatibilitaMotherboardSocket, importando nomeSocket in Motherboard
- CompatibilitaRamMotherboard, importando nomeGenerazione in Motherboard
- AppartenenzaRam, importando nomeGenerazione in Ram
- CompatibilitaRamCpu, reificata importando codiceCpu e nomeGenerazione
- Produzione, importando codiceProduttore in Componente
- Rilevamento, importando codiceComponente in PrezzoComponente
- Fornitura, accorpare Rivenditore a PrezzoComponente.

Analisi delle ridondanze

È stata inserita una ridondanza accorpendo l'entità Rivenditore a PrezzoComponente. In tal modo, a costo di una maggiore occupazione di memoria, si può risalire direttamente al rivenditore di una rilevazione specifica.

OP11 – Visualizzazione dello storico dei prezzi di un componente

Con ridondanza

Concetto	Costrutto	Accessi	Tipo
Componente	E	1	L
Rilevamento	R	28	L
PrezzoComponente	E	28	L
Totale: 57L => 11.400 al giorno			

Senza ridondanza

Concetto	Costrutto	Accessi	Tipo
Componente	E	1	L
Rilevamento	R	28	L
PrezzoComponente	E	28	L
Fornitura	R	28	L
Rivenditore	E	28	L
Totale: 113L => 22.600 al giorno			

OP8 – Rilevamento dei prezzo di un componente

Con ridondanza

Concetto	Costrutto	Accessi	Tipo
Componente	E	1	L
Rilevamento	R	2	S
PrezzoComponente	E	2	S
Totale: 1L + 4S => 90.000 al giorno			

Senza ridondanza

Concetto	Costrutto	Accessi	Tipo
Componente	E	1	L
Rilevamento	R	2	S
PrezzoComponente	E	2	S
Fornitura	R	2	S
Rivenditore	E	2	L
Totale: 3L + 6S => 150.000 al giorno			

Totale accessi con ridondanza = 101.400

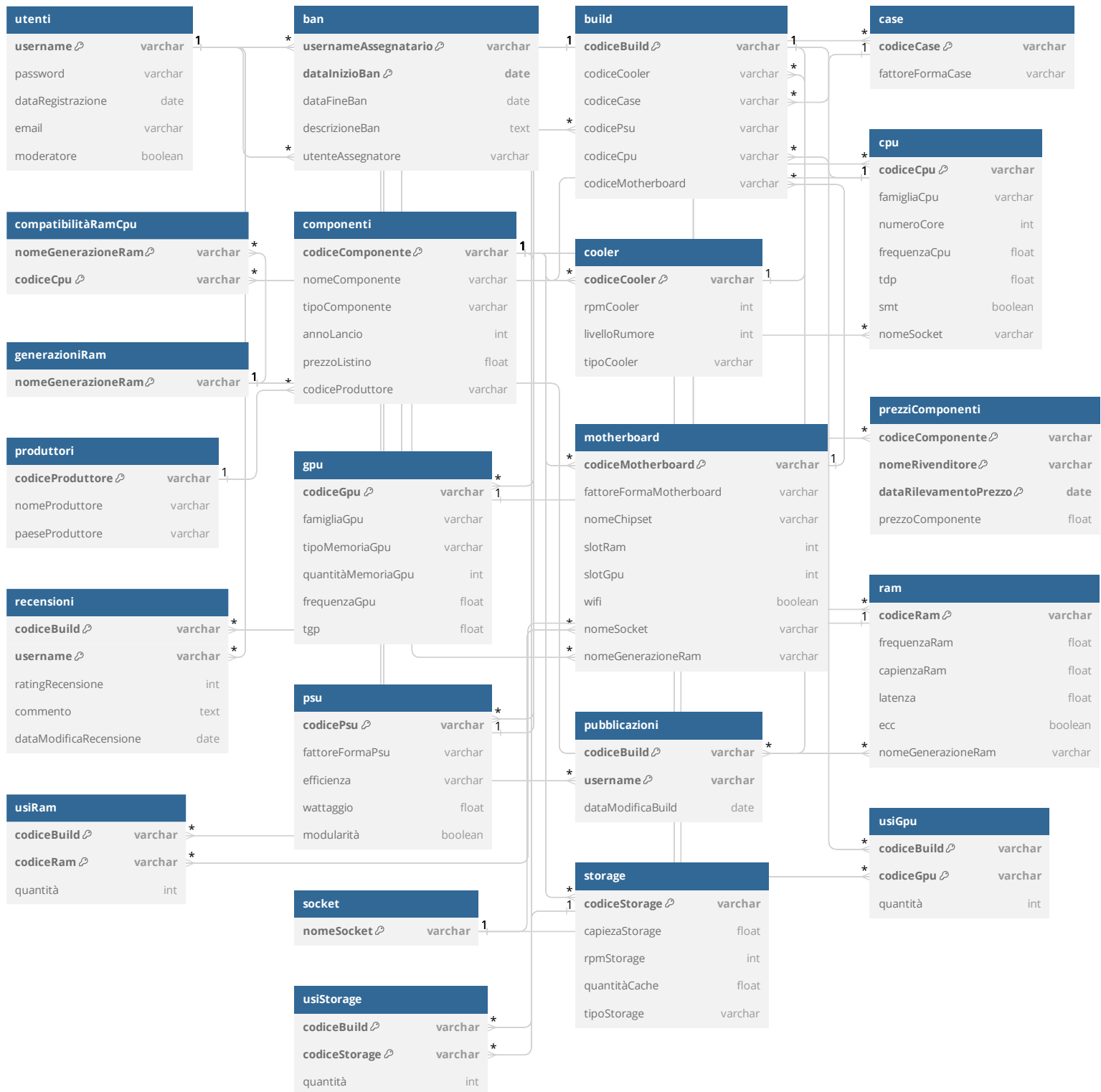
Totale accessi senza ridondanza = 172.600

E' chiaro che a fronte di una maggiore occupazione di memoria, la denormalizzazione porta ad una riduzione degli accessi.

Traduzione di entità e associazioni in relazioni

ban(usernameAssegnatario: utenti.username, dataInizioBan, dataFineBan*,
descrizioneBan, utenteAssegnatore: utenti.username)
build(codiceBuild, codiceCooler: cooler, codiceCase: case, codicePsu: psu, codiceCpu:
cpu, codiceMotherboard: motherboard)
case(codiceCase: componenti.codiceComponente, fattoreFormaCase)
compatibilitàRamCpu(nomeGenerazioneRam: generazioniRam, codiceCpu: cpu)
componenti(codiceComponente, nomeComponente, tipoComponente, annoLancio,
prezzoListino, codiceProduttore: produttori)
cooler(codiceCooler: componenti.codiceComponente, rpmCooler*, livelloRumore*,
tipoCooler)
cpu(codiceCpu: componenti.codiceComponente, famigliaCpu, numeroCore,
frequenzaCpu, tdp, smt, nomeSocket: socket)
generazioniRam(nomeGenerazioneRam)
gpu(codiceGpu: componenti.codiceComponente, famigliaGpu, tipoMemoriaGpu,
quantitàMemoriaGpu, frequenzaGpu, tgp)
motherboard(codiceMotherboard: componenti.codiceComponente,
fattoreFormaMotherboard, nomeChipset, slotRam, slotGpu, wifi, nomeSocket:
socket, nomeGenerazioneRam: generazioniRam)
prezziComponenti(codiceComponente: componenti.codiceComponente,
nomeRivenditore, dataRilevamentoPrezzo, prezzoComponente)
produttori(codiceProduttore, nomeProduttore, paeseProduttore)
psu(codicePsu: componenti.codiceComponente, fattoreFormaPsu, efficienza,
wattaggio, modularità)
pubblicazioni(codiceBuild: build, username: utenti, dataModificaBuild)
ram(codiceRam: componenti.codiceComponente, frequenzaRam, capienzaRam,
latenza, ecc, nomeGenerazioneRam: generazioniRam)
recensioni(codiceBuild: build, username: utenti, ratingRecensione, commento*,
dataModificaRecensione)
socket(nomeSocket)
storage(codiceStorage: componenti.codiceComponente, capienzaStorage, rpmStorage*,
quantitàCache*, tipoStorage)
usiGpu(codiceBuild: build, codiceGpu: gpu, quantità)
usiRam(codiceBuild: build, codiceRam: ram, quantità)
usiStorage(codiceBuild: build, codiceStorage: storage, quantità)
utenti(username, password, dataRegistrazione, email, moderatore)

Schema relazionale finale



Traduzione delle operazioni in query SQL

OP 1 – Registrazione di un utente

```
INSERT INTO utenti (username, password, dataRegistrazione, email, moderatore)
VALUES (?, ?, ?, ?, ?)
```

OP 2 – Inserimento del ban di un utente

```
INSERT INTO ban (usernameAssegnatario, dataInizioBan, dataFineBan, descrizioneBan,
utenteAssegnatore)
VALUES (?, ?, ?, ?, ?)
```

OP 3 – Login di un utente

Per effettuare con successo un login è necessario che l'utente esista già nel database, inoltre non deve avere alcun ban in corso. Tale processo è stato scomposto in due query:

```
SELECT COUNT(*)
```

```
FROM utenti
```

```
WHERE Username = ?
```

```
SELECT COUNT(*)
```

```
FROM ban
```

```
WHERE usernameAssegnatario = ? AND (dataFineBan IS NULL OR dataFineBan > NOW());
```

Il login avverrà correttamente se il numero di record della prima query risulta pari ad 1 e pari a 0 nella seconda.

OP 4 – Inserimento di una build

Data la possibilità di usare diverse RAM, GPU e STORAGE in quantità diversa è necessario creare delle query apposite per tali casi.

L'operazione di inserimento di una build è scomposta nelle seguenti query:

```
INSERT INTO build (codiceBuild, codiceCooler, codiceCase, codicePsu, codiceCpu,  
codiceMotherboard)  
VALUES (?, ?, ?, ?, ?, ?);
```

```
INSERT INTO pubblicazioni (codiceBuild, username, dataModificaBuild)  
VALUES (?, ?, ?)
```

```
INSERT INTO usiGpu (codiceBuild, codiceGpu, quantita)  
VALUES (?, ?, ?)
```

```
INSERT INTO usiRam (codiceBuild, codiceRam, quantita)  
VALUES (?, ?, ?)
```

```
INSERT INTO usiStorage (codiceBuild, codiceStorage, quantita)  
VALUES (?, ?, ?)
```

Per assicurare l'integrità del database, tali operazioni devono essere eseguite come una transazione.

OP 5 – Inserimento di una recensione

```
INSERT INTO recensioni (codiceBuild, username, ratingRecensione, commento,  
dataModificaRecensione)  
VALUES (?, ?, ?, ?, ?)
```

OP 6 – Modifica di una recensione

```
UPDATE recensioni  
SET ratingRecensione = ?, commento = ?, dataModificaRecensione = ?  
WHERE codiceBuild = ? AND username = ?
```

OP 7 – Inserimento di un componente

Tale procedura differisce nella scelta della tabella di classificazione del componente.

Si illustrano i passaggi dell'inserimento di una CPU in quanto richiede anche l'inserimento della sua compatibilità con le generazioni di RAM.

```
INSERT INTO Componenti (codiceComponente, nomeComponente, tipoComponente,  
annoLancio, prezzoListino, codiceProduttore)  
VALUES (?, ?, ?, ?, ?, ?)
```

```
INSERT INTO Cpu (codiceCpu, famigliaCpu, numeroCore, frequenzaCpu, tdp, smt,  
nomeSocket)  
VALUES (?, ?, ?, ?, ?, ?, ?)
```

```
INSERT INTO CompatibilitaRamCpu (nomeGenerazioneRam, codiceCpu)  
VALUES (?, ?)
```

OP 8 – Rilevamento del prezzo di tutti i componenti

```
INSERT prezziComponenti (codiceComponente, nomeRivenditore,  
dataRilevamentoPrezzo, prezzoComponente)  
(?, ?, ?, ?)
```

OP 9 – Visualizzazione del prezzo più recente e basso di un componente

```
SELECT *FROM prezziComponenti p  
WHERE p.codiceComponente = ? AND p.dataRilevamentoPrezzo = (  
    SELECT MAX(dataRilevamentoPrezzo)  
    FROM prezziComponenti  
    WHERE codiceComponente = p.codiceComponente)  
AND p.prezzoComponente = (  
    SELECT MIN(prezzoComponente)  
    FROM prezziComponenti  
    WHERE codiceComponente = p.codiceComponente  
    AND dataRilevamentoPrezzo = (  
        SELECT MAX(dataRilevamentoPrezzo)  
        FROM prezziComponenti  
        WHERE codiceComponente = p.codiceComponente));
```

OP 10 – Visualizzazione dei dettagli di un componente

Tale procedura restituisce i dati generali di un CPU, assieme alle informazioni più specifiche e il nome del produttore.

```
SELECT co.codiceComponente, co.nomeComponente, co.annoLancio, co.prezzoListino,  
p.nomeProduttore, c.*  
FROM componenti co, cpu c, produttori p  
WHERE codiceCpu = ? and c.codiceCpu = co.codiceComponente  
AND p.codiceProduttore = co.codiceProduttore
```

L'operazione è analoga per le altre classificazioni di un componente.

OP 11 – Visualizzazione dello storico dei prezzi di un componente

```
SELECT *  
FROM prezziComponenti  
WHERE codiceComponente = ?  
AND nomeRivenditore = ?  
ORDER BY dataRilevamentoPrezzo DESC  
LIMIT 14;
```

OP 12 – Visualizzazione del rating medio di un utente

```
SELECT AVG(r.ratingRecensione) AS AverageRating  
FROM utenti u, pubblicazioni p, recensioni r  
WHERE p.username = ?  
AND u.username = p.username  
AND p.codiceBuild = r.codiceBuild  
GROUP BY u.username
```

Progettazione dell'applicazione

Descrizione dell'architettura dell'applicazione realizzata

L'applicazione per interfacciarsi al database è stata realizzata in Java, sfruttando il DAO pattern; il database risiede in locale e il DBMS usato è MySQL.

L'applicazione è una Java Swing app che fa uso di schermate per richiedere e inserire informazioni al back-end.

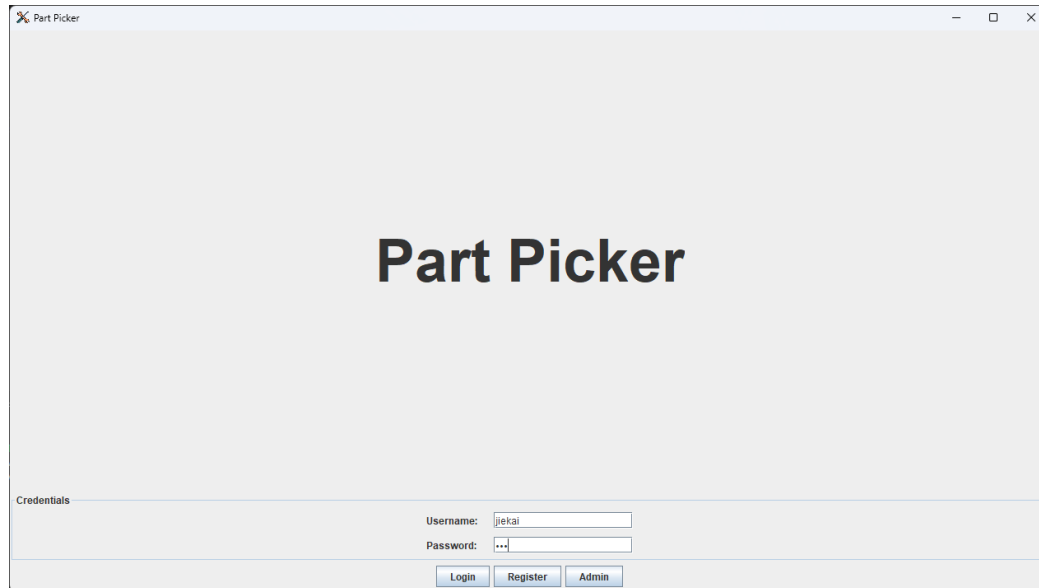


Fig. 1.5 Schermata di login dell'applicazione



Fig 1.6 Dashboard che permette la visione di build, componenti e la creazione di build

Part Picker (jekai2 (Moderator))

Home **Back**

CPU
Core i3-10100 Price: €112.64 [View Details](#)

Cooler
CoolerMaster Hyper 212 Price: €27.59 [View Details](#)

Case
Cooler Master MasterBox Q300L Price: €45.99 [View Details](#)

PSU
EVGA SuperNOVA 650 G3 Price: €82.79 [View Details](#)

Motherboard
ASUS ROG Strix Z490-E Gaming Price: €275.99 [View Details](#)

GPU
GeForce RTX 3060 Price: €303.10 [View Details](#)
[Add](#) [Remove](#)

RAM
Kingston HyperX Fury DDR4 8GB Price: €36.79 [View Details](#)
[Add](#) [Remove](#)

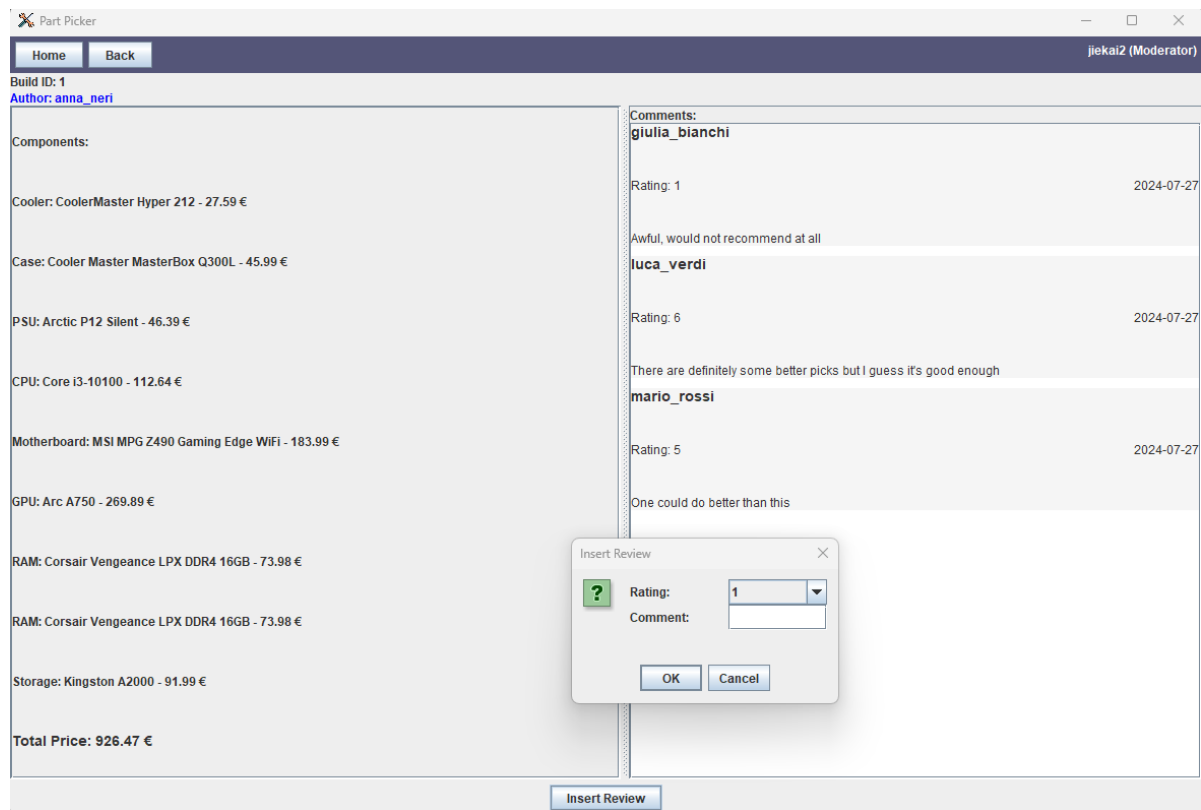
Storage
Crucial MX500 Price: €73.99 [View Details](#)
[Add](#) [Remove](#)

Total Price: €958.88 [Upload Build](#)

Fig. 1.7 Schermata per la creazione di build



Fig 1.8 Visione di una categoria di componenti e lo storico prezzi (doppio click)



Fig

Fig 1.9 Visione di una build e le relative recensioni

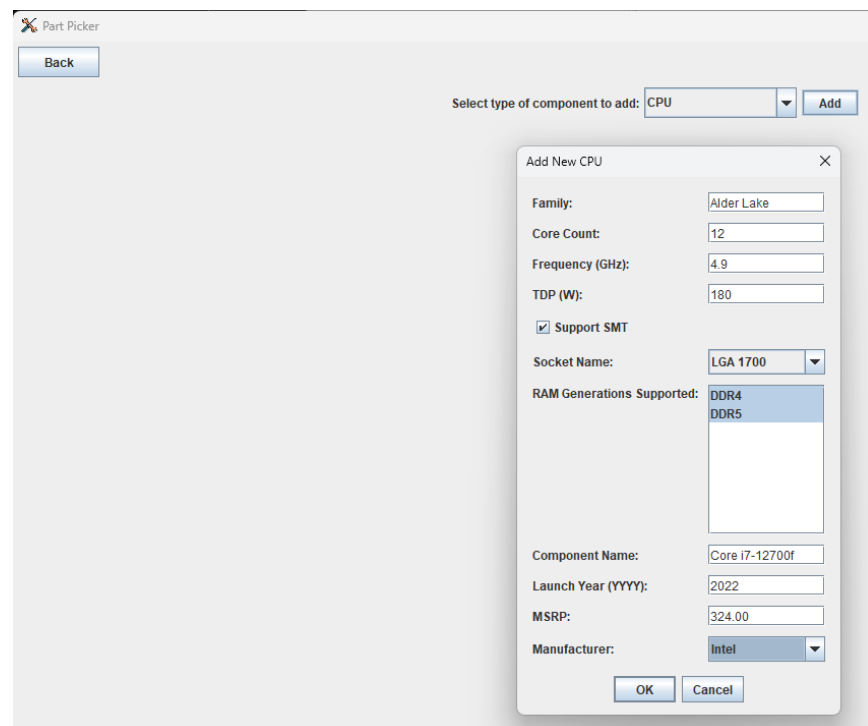


Fig 2.0 Schermata admin per l'inserimento di una CPU.

Per la maggior parte delle query, la correttezza dei dati inseriti e vincoli vari viene verificata a livello applicativo.

L'applicazione fornisce nel complesso le funzionalità richieste evidenziate nella fase di progettazione tranne l'inserimento del rilevamento dei prezzi attraverso l'interfaccia in quanto l'operazione di price scraping esula dallo scopo del corso.

In ogni caso il database è inizializzato con prezzi fittizi per componenti già esistenti.