



# CM-Forniture

Tecnologie Web



Progettazione Web App per la vendita e  
l'acquisto di prodotti monouso .

Salvatore Bianco



di

# Introduzione

---

CM-Forniture nasce da un'esigenza personale. La Wep App è stata pensata per l'attività di famiglia , che possa ampliare il cerchio dei clienti finali in modo semplice e veloce. In più ai fini del progetto viene data al possibilità ad un altro fornitore di registrarsi nell'app , al momento in maniera basilare, e poter così caricare e gestire i suoi prodotti e le sue vendite.

Il fornitore avrà a disposizione anche un piccolo controllo sui suoi prodotti che vengono acquistati.

# Requisiti

L'applicazione deve permettere:

- L'utente anonimo deve poter visitare la web app , deve potersi però registrare. La registrazione prevede due modalità , clienti e fornitori.
- L'imprenditore deve poter caricare i suoi prodotti, questo in base alle categorie che vengono proposte, questo ovviamente comprendendo tutti i dettagli.
- I clienti finali devono poter inserire delle recensioni e ,se loggati , visualizzare le altre inserire da altri utenti.
- Gli utenti non identificati devono poter vedere i prodotti e solo i voti dati al fornitore e al prodotto
- I fornitori possono chiedere all'admin l'inserimento di una nuova categoria prodotti.
- Ricerca filtrata dei prodotti prevista per nome, prezzo , tipo materiale e prezzo
- Prodotti consigliati quando vengono visualizzati dei prodotti in base alle categorie
- Pagina gestione prodotti caricati per il fornitore
- Pagina di gestione recensioni per il cliente
- Forum in cui discutere di alcuni problemi con i fornitori e/o con altri clienti , a seconda dei diversi thread
- Il cliente correttamente loggato potrà inserire un prodotto nel carrello e procedere con l'ordine.

# Descrizione Progetto

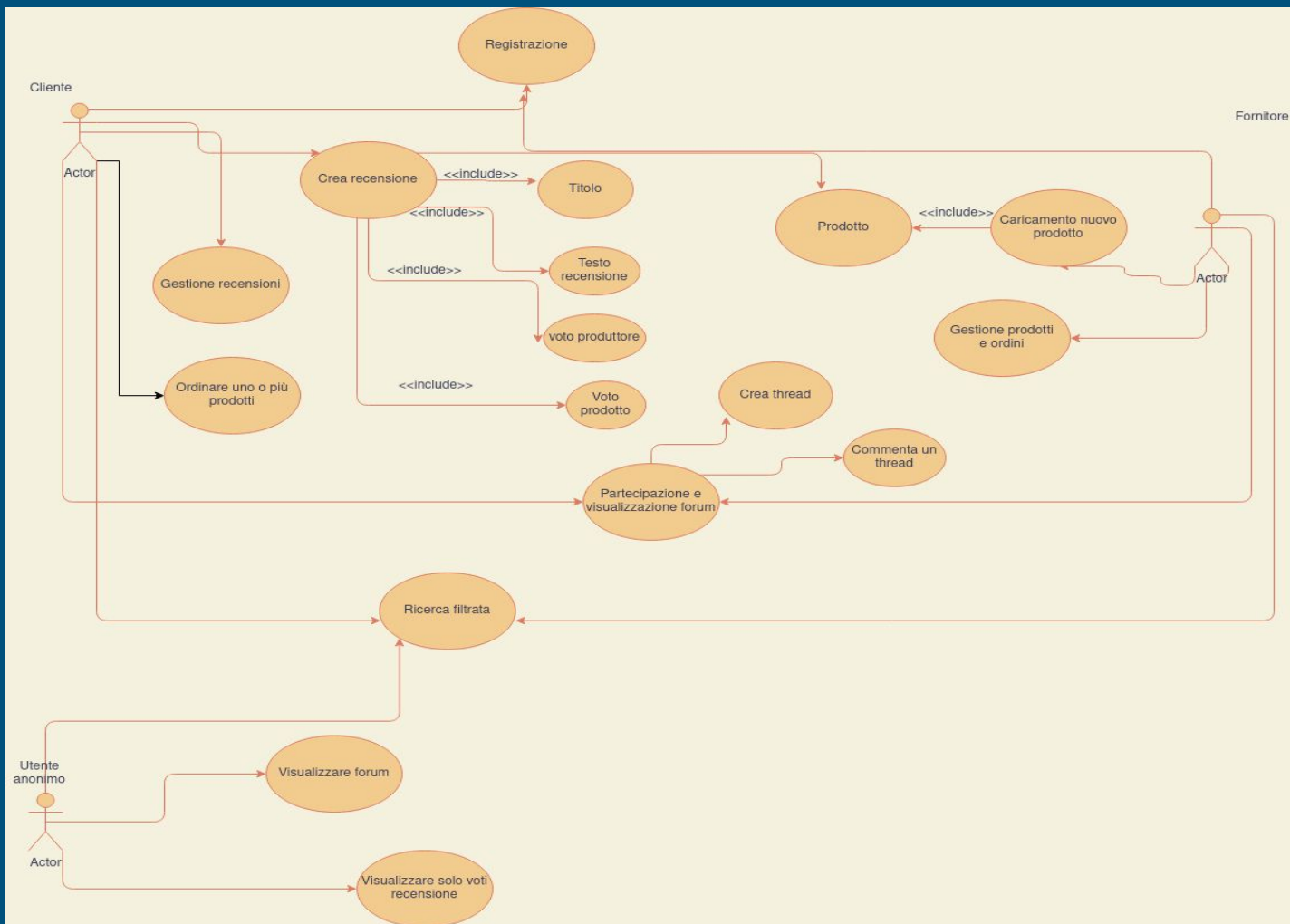
---

Termine	Sinonimo
Utente	Cliente, Owner
Client	Cliente
Owner	Fornitore
Search	Ricerca di un prodotto attraverso i filtri

# Tipologia utenti e permessi

- Utenti registrati come clienti . Possono utilizzare a pieno la web app una volta effettuata la registrazione , in cui al momento è previsto solo username e password. Potranno effettuare ricerche filtrate , visualizzare i prodotti , potranno ordinarli e recensirli. Avranno accesso al forum per discutere con i fornitori e/o altri clienti . Avranno a disposizione anche una pagina di gestione delle proprie recensioni.
- Utenti registrati come fornitori(Owner). Questi potranno utilizzare la Web App a pieno , potranno visualizzare i prodotti ed essendo fornitori avranno la possibilità di caricare i prodotti in base alle categorie messe a disposizione. Potranno ,nel caso avessero prodotti fuori categoria , di richiedere e motivare l'inserimento di una determinata categoria. Avranno a disposizione una pagina di gestione per i prodotti caricati dal fornitore ,che permette una ricerca per nome-prodotto. La gestione prevede l'update e la delete.
- Utenti anonimi. Questi possono utilizzare la Web App senza interazione, possono visualizzare solo i prodotti e i dettagli e i voti ma senza vedere le recensioni ed effettuare ricerche.. Possono visualizzare i vari thread del forum ma non possono creare thread e non possono commentare quelli esistenti.

## In sintesi vediamo l'UML



# Database

---

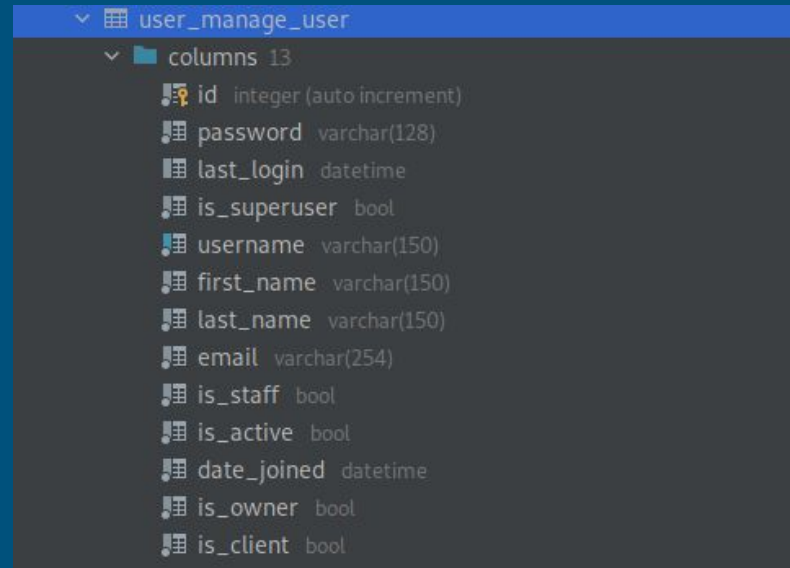
Per il database è stato scelto Sqlite per una consegna più rapida del progetto.

La costruzione del DB ha previsto l'utilizzo di diversi modelli.

Di seguito verranno spiegati i più interessanti.

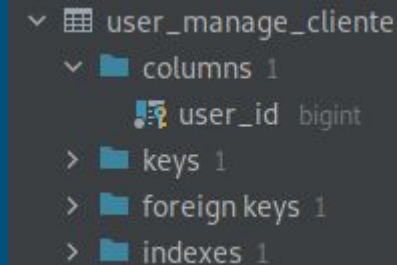
# Gestione utenti

Per il modello User è stato utilizzato quello presente all'interno dell'App django Auth. Nel nostro caso però era necessaria la differenziazione degli utenti, tra fornitore e cliente. Utilizzando dei flag booleani, `is_client` e `is_owner`, il risultato ottenuto è stato quello ricercato. Questo modello è relazionato con associazioni di tipo 1:1 con i modelli Owner e il modello Cliente. Questi due modelli, come possiamo osservare negli screen laterali, hanno un solo attributo `user_id` che è Foreign Key, facendo riferimento al modello User.



user\_manage\_user

columns 13	
id	integer (auto increment)
password	varchar(128)
last_login	datetime
is_superuser	bool
username	varchar(150)
first_name	varchar(150)
last_name	varchar(150)
email	varchar(254)
is_staff	bool
is_active	bool
date_joined	datetime
is_owner	bool
is_client	bool



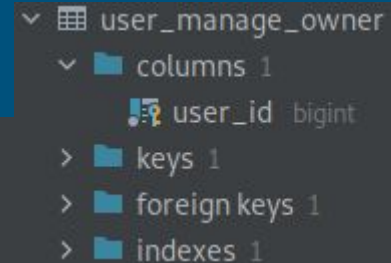
user\_manage\_cliente

columns 1	
user_id	bigint

> keys 1

> foreign keys 1

> indexes 1



user\_manage\_owner

columns 1	
user_id	bigint

> keys 1

> foreign keys 1

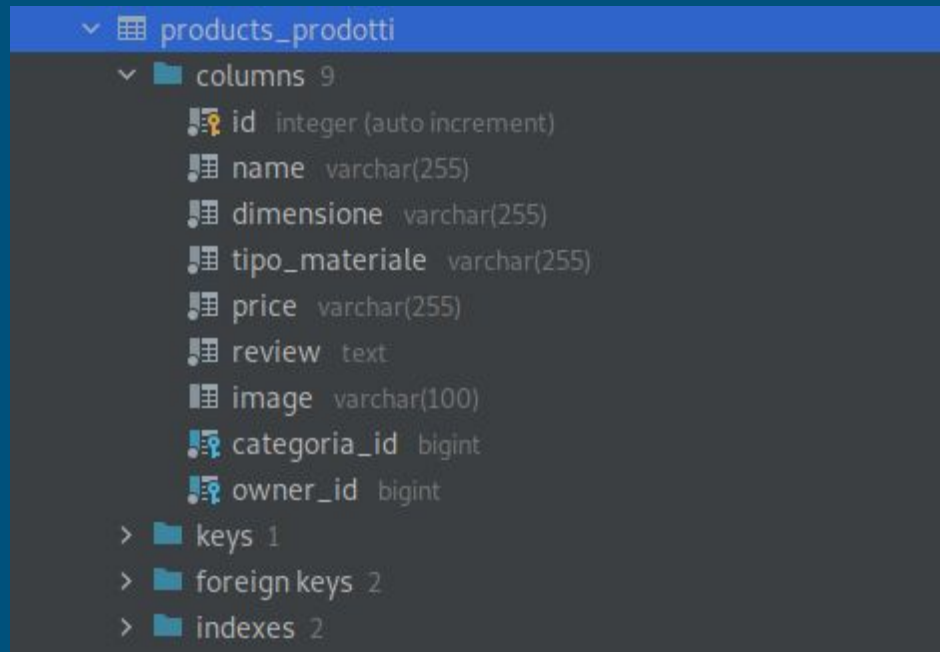
> indexes 1



# Gestione prodotti

Per quanto riguarda l'app products, qui è il vero succo del nostro ecommerce.

All'interno di quest'app ci sono diversi modelli. Il più importante è Prodotti , questo modello rappresenta la caratteristica di un prodotto che un determinato fornitore andrà a caricare. Ha come chiavi esterne , ovviamente l'id del'owner ma anche l'id categoria , proveniente appunto dal modello categoria.



The screenshot shows a database schema viewer for a table named 'products\_prodotti'. The table has 9 columns. The columns are: 'id' (integer, auto increment, primary key), 'name' (varchar(255)), 'dimensione' (varchar(255)), 'tipo\_materiale' (varchar(255)), 'price' (varchar(255)), 'review' (text), 'image' (varchar(100)), 'categoria\_id' (bigint, foreign key), and 'owner\_id' (bigint, foreign key). There is also a 'keys' section showing 1 key, a 'foreign keys' section showing 2 foreign keys, and an 'indexes' section showing 2 indexes.

products_prodotti	
columns	9
id	integer (auto increment)
name	varchar(255)
dimensione	varchar(255)
tipo_materiale	varchar(255)
price	varchar(255)
review	text
image	varchar(100)
categoria_id	bigint
owner_id	bigint
keys	1
foreign keys	2
indexes	2

Il modello Categoria contiene gli attributi semplici richiesti , il nome e l'immagine. La particolarità della categoria è che non può essere introdotta da nessuno se non dall'admin.

Molto importante è il modello Review , che contiene gli attributi necessari per un cliente in caso di recensione. Per un corretto funzionamento vengono assegnate due chiavi esterne che riportano l'id del prodotto che si sta recensendo e quale cliente sta rilasciando la recensione. Un ultima particolarità è data dal modello Report, che permette di salvare le richieste dei fornitori per nuove categorie.

A screenshot of a database schema viewer showing the 'products\_report' table. It has 3 columns: 'id' (integer, auto-increment, primary key), 'author' (varchar(200)), and 'text' (text). There is 1 key defined for this table.

Table	Columns	Keys
products_report	<ul style="list-style-type: none"><li>id integer (auto increment)</li><li>author varchar(200)</li><li>text text</li></ul>	1

A screenshot of a database schema viewer showing the 'products\_categoria' table. It has 3 columns: 'id' (integer, auto-increment, primary key), 'nome' (varchar(255)), and 'image' (varchar(100)). There is 1 key defined for this table.

Table	Columns	Keys
products_categoria	<ul style="list-style-type: none"><li>id integer (auto increment)</li><li>nome varchar(255)</li><li>image varchar(100)</li></ul>	1

A screenshot of a database schema viewer showing the 'products\_review' table. It has 7 columns: 'id' (integer, auto-increment, primary key), 'comment\_name' (varchar(200)), 'comment\_body' (text), 'rating\_fornitore' (integer), 'rating\_prodotto' (integer), 'prodotto\_id' (bigint, foreign key), and 'user\_id' (bigint, foreign key). There is 1 key and 2 foreign keys defined for this table.

Table	Columns	Keys	Foreign Keys
products_review	<ul style="list-style-type: none"><li>id integer (auto increment)</li><li>comment_name varchar(200)</li><li>comment_body text</li><li>rating_fornitore integer</li><li>rating_prodotto integer</li><li>prodotto_id bigint</li><li>user_id bigint</li></ul>	1	2

# Gestione Forum

Per l'app Forum ho previsto due Modelli , uno Forum che permette di salvare i thread e tiene conto di quali clienti creano il thread verso i fornitori.

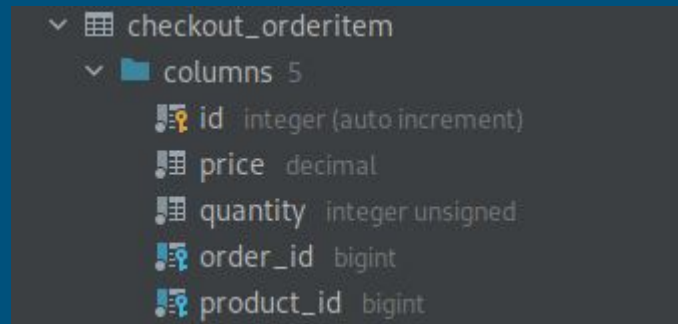
Un modello Risposta che come attributi riposta come foreign key il thread , utilizzato per le risposte che vengono date per un thread

```

v forum_forum
  v columns 6
    id integer (auto increment)
    titolo varchar(255)
    descrizione varchar(255)
    contenuto text
    fornitore_id bigint
    user_id bigint
  > keys 1
  > foreign keys 2
  > indexes 2
v forum_risposta
  v columns 4
    id integer (auto increment)
    user varchar(200)
    commento text
    titolo_id bigint
  > keys 1
  > foreign keys 1
  > indexes 1
```

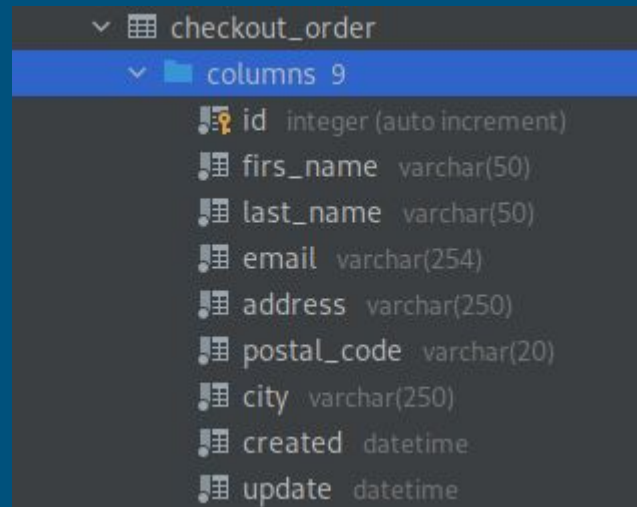
# Gestione Ordini

Nell'app Checkout sono presenti due modelli, il modello Order che salva le informazioni di spedizione di un cliente acquirente. Questo modello è collegato ad OrderItem dove sono presenti invece le informazioni sui prodotti provenienti dal carrello. In foreign key infatti troviamo, `order_id` e per tenere traccia dei prodotti la foreign key `product_id`.



A screenshot of a database schema viewer showing the structure of the `checkout_orderitem` table. The table has 5 columns: `id` (integer, auto-increment), `price` (decimal), `quantity` (integer, unsigned), `order_id` (bigint), and `product_id` (bigint). Each column has a small icon to its left.

checkout_orderitem	
columns 5	
id	integer (auto increment)
price	decimal
quantity	integer unsigned
order_id	bigint
product_id	bigint



A screenshot of a database schema viewer showing the structure of the `checkout_order` table. The table has 9 columns: `id` (integer, auto-increment), `first_name` (varchar(50)), `last_name` (varchar(50)), `email` (varchar(254)), `address` (varchar(250)), `postal_code` (varchar(20)), `city` (varchar(250)), `created` (datetime), and `update` (datetime). Each column has a small icon to its left.

checkout_order	
columns 9	
id	integer (auto increment)
first_name	varchar(50)
last_name	varchar(50)
email	varchar(254)
address	varchar(250)
postal_code	varchar(20)
city	varchar(250)
created	datetime
update	datetime

# Tecnologie utilizzate

---

Per lo sviluppo del seguente progetto è stato utilizzato il web framework Django , visto durante il corso di Tecnologie Web.

# Organizzazione del software e scelte

---

La struttura del progetto prevede a livello di codice l'implementazione di cinque APP

- user\_manage
- products
- forum
- cart
- checkout

# User\_manage

---

L'app user\_manage è implementata per la gestione degli user , nello specifico per la differenziazione tra user-cliente e user-owner. Questa consente ad un utente di registrarsi e contiene i modelli utilizzati per la gestione di tali operazioni:

- User
- Owner
- Cliente

All'interno di quest'app poi sono stati implementati i due decorator , questi due decorator servono per verificare che tipologia di utente sta chiamando un determinato servizio, attraverso i due flag del modello User. In caso positivo abilitano l'user a proseguire , in caso negativo notificano l'errore.

Avendo implementato la parte di Login e Logout utilizzando l'autenticazione di Django , user\_manage ha quindi un ruolo centrale per il rilascio di permessi agli utenti.

# Products

L'app products è implementata per la gestione dei prodotti , ed è un po' il cuore della Web App. All'interno troviamo diversi modelli:

- Prodotti
- Categoria
- Review
- Report

L'app sfrutta i modelli implementati da user\_manage per la gestione dei prodotti . Infatti all'interno di products troviamo la possibilità di creare un nuovo prodotto , la possibilità di recensirlo , e tutte le funzionalità riservate successivamente agli utenti proprietari come update e delete. Tutto questo avviene grazie all'utilizzo di decoratori che verificano l'autenticazione dell'utente e limitano i loro comportamenti. Il modello Categoria mi permette di tenere conto delle varie categorie introdotte dall'admin , ed è chiave esterna in Prodotti con eliminazione a cascata. Il modello Review tiene traccia delle recensioni fatte dai clienti , e il modello Report è utilizzato nel momento in cui un fornitore voglia richiedere l'inserimento di una nuova categoria. Nell'app products viene gestita anche la ricerca filtrata disponibile per gli utenti e in più per gli utenti cliente è disponibile un'area di gestione di tutte le recensioni effettuate , mentre per i fornitori quest'area diventa di gestione dei prodotti caricati. Con gestione si intende la possibilità di update e delete. Infine è gestita anche la parte di recommendation.



# Forum

---

L'ultima app implementata è l'app forum. Un app molto semplice con i seguenti modelli :

- Forum
- Risposta

Questa app in maniera semplice dà la possibilità agli utenti , anche anonimi , di visualizzare l'area forum in cui vengono mostrati i diversi thread. Questi thread però sono creabili solo da clienti , quindi user loggati come tali , e commentati da utenti clienti o dal fornitore del prodotto interpellato . Per semplicità infatti la chiave esterna di Forum è proprio il fornitore , che un cliente può interpellare per qualsiasi problema , anche per questo non è stato reputato necessario inserire altri limiti (come id prodotto). Nel modello Risposta invece sono mantenuti tutti i commenti relativi ad un determinato thread.

# Cart e Checkout

---

L'applicazione Cart dà semplicemente la possibilità ad un utente loggato come cliente di poter inserire il prodotto desiderato nel carrello nella quantità desiderata, prezzo calcolato in automatico. L'app checkout invece registra i prodotti del carrello che vengono ordinati, infatti nel modello `OrderItem` è presente la chiave esterna del prodotto. Sempre dell'app checkout infine abbiamo il modello `Order` che permette al cliente di inserire i dati per la spedizione che verranno poi visualizzati dai vari fornitori dei prodotti.

# Test

---

I test effettuati sono applicati ad alcune form e view e ne testano soprattutto la corretta creazione.

I test sono stati effettuati alle app Forum , User\_manage, Products, Checkout.

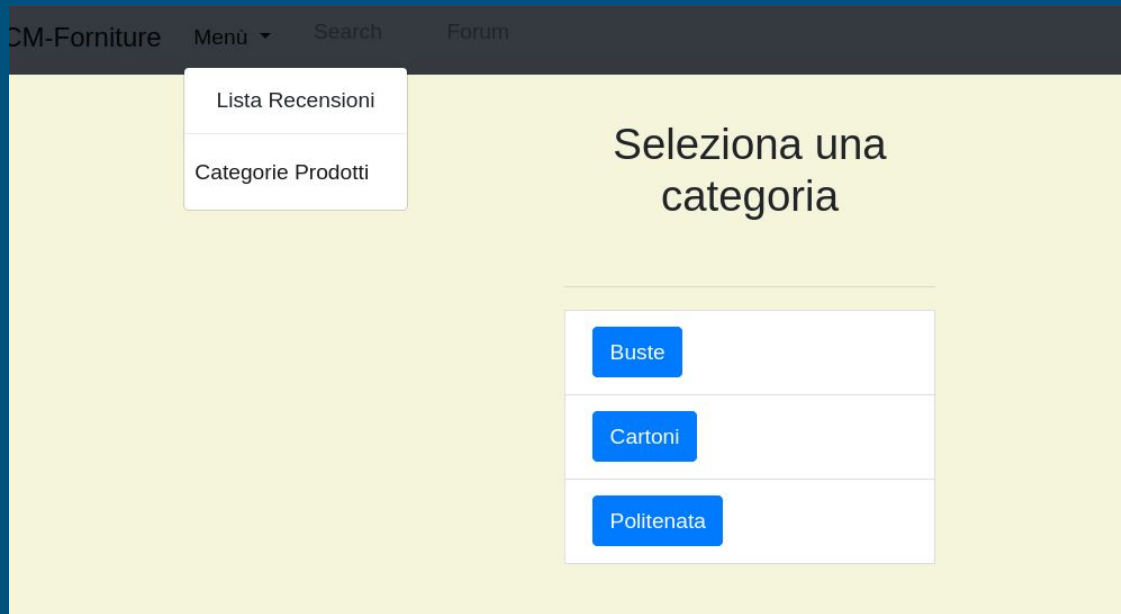
# Risultati ed esempi



Schermata principale da utente cliente 2. Vediamo la barra in alto che ci permette di mantenere a disposizione ogni features

# Risultati ed esempi

Schermata , dal punto di vista cliente , con menù e pagina di categoria prodotto sotto



# Risultati ed esempi

Schermata post dettaglio prodotto , vediamo che il cliente al momento ha la possibilità di recensire il prodotto. A fine pagina troviamo invece i prodotti consigliati. Nella seconda immagine invece vediamo la pagina per recensire

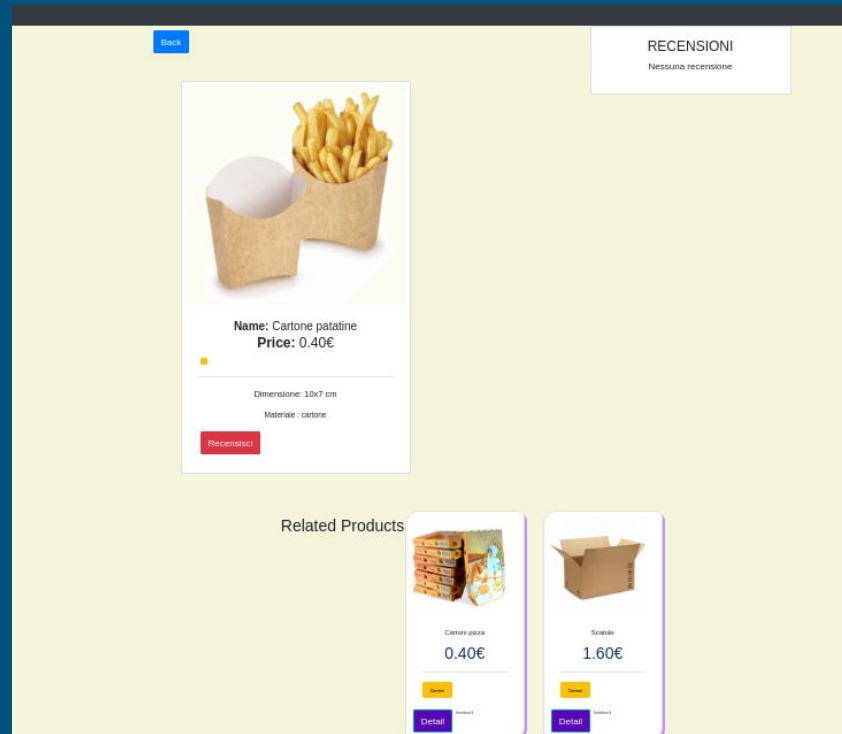
### Recensione

Comment name:

Comment body:

Rating fornitore:

Rating prodotto:



# Risultati ed esempi

FORUM				
Nuovo thread				
Titolo	Fornitore	Utente	Descrizione	Discussione
Politenata	fornitore1	cliente2	Pareri?	Discussione

Schermata principale pagina forum. Dal punto di vista utente vediamo che può creare un nuovo thread oppure partecipare alla discussione di uno esistente

# Risultati ed esempi

## Creazione nuovo prodotto

Categoria\*

-----

Name\*

Dimensione\*

Tipo materiale\*

Price\*

Image

---

Browse

Submit

Pagina creazione prodotto. Per non dilungarmi troppo gli altri screen sono inseriti nella cartella screen.



# Considerazioni finali

---

I problemi sono molto comuni a quelli dei miei colleghi con cui ho parlato , in primis sono dovuti all'inesperienza nello scrivere back-end. Ho dovuto masticare molto velocemente parecchie nozioni di base . Per superare alcune difficoltà ho utilizzato molto la documentazione e stackoverflow. Poi ho seguito il consiglio anche di altri ragazzi e del prof , ovvero vedere subito se qualcuno ha già fatto un qualcosa che potesse darci anche una base grezza da modellare.

Nonostante questo sono molto contento del risultato finale , sia perchè l'applicazione nel suo piccolo funziona correttamente e sia perchè mi ha dato un assaggio di cosa realmente c'è dietro , soprattutto dietro un linguaggio come python ma anche dietro la progettazione web. Sono però soddisfatto soprattutto nel notare come le mie difficoltà in partenza con la creazione di questo ecommerce siano andate pian piano a diminuire , dandomi anche una certa confidenza per la creazione di nuovi progetti.