

Gestione della vendita dei prodotti dei piccoli negozianti

Titolo del progetto: Gestione della vendita dei prodotti dei piccoli negozianti
Alunno/a: Giairo Mauro
Classe: I4AC
Anno scolastico: 2018/2019
Docente responsabile: Guido Montalbetti

1	Introduzione.....	3
1.1	Informazioni sul progetto.....	3
1.2	Abstract	3
1.3	Scopo	3
2	Analisi.....	4
2.1	Analisi del dominio	4
2.2	Analisi dei costi.....	4
2.3	Analisi e specifica dei requisiti	4
2.4	Use case	9
2.5	Pianificazione	10
2.6	Analisi dei mezzi.....	11
2.6.1	Software	11
2.6.2	Hardware.....	11
3	Progettazione	11
3.1	Design dell'architettura del sistema	11
3.2	Design dei dati e database.....	13
3.3	Design delle interfacce	15
4	Implementazione	24
4.1	Grafica pagine	24
4.1.1	Cliente	24
4.1.2	Gestore	27
4.1.3	Amministratore	29
4.1.4	JavaScript	31
4.2	MVC	32
4.2.1	Htaccess	32
4.2.2	Index file.....	33
4.2.3	application	35
4.2.3.1	Controller	35
4.2.3.2	Model	38
5	Test.....	41
5.1	Protocollo di test.....	41
5.2	Risultati test.....	45
5.2.1	TC-001	45
5.2.2	TC-002	48
5.2.3	TC-003	50
5.2.4	TC-004	51
5.2.5	TC-005	52
5.2.6	TC-006	53
5.2.7	TC-007	54
5.2.8	TC-008	55
5.2.9	TC-009	56
5.2.10	TC-010	58
5.2.11	TC-011	59
5.2.12	TC-012	61
5.2.13	TC-013	62
5.2.14	TC-014	63
5.2.15	TC-015	63
5.2.16	Tabella riassuntiva	63
5.3	Mancanze/limitazioni conosciute.....	63
6	Consuntivo.....	64
7	Conclusioni	65
7.1	Sviluppi futuri.....	65
7.2	Considerazioni personali	65
8	Bibliografia	65
8.1	Sitografia	65
9	Didascalia	67
10	Glossario.....	65
11	Allegati	69

1 Introduzione

1.1 Informazioni sul progetto

Allievo: Giairo Mauro

Docente: Guido Montalbetti

Scuola: SAM Trevano

Sezione: Informatica

Materia: Progetti individuali

Data inizio: 08.01.2019

Data fine: 10.04.2019

1.2 Abstract

My project has the concept of a web site where dealer of small shops can sell their products online and so be more known from most people.

The way to sell things online is used a lot, but my website won't be for just one shop but for more, as it will be about physical shop and not just online, so that a customer can buy from the website or go to the shop at the address written online.

The seller will have the possibility to be registered from the administrator, but they cannot register by themselves. The administrator will register the dealer and the shop with all the information, so that the dealers can log in and work in their space, sell new products or modify the existing ones.

I will use PHP to implement the connection to the database and alter insert or delete the data from it.

To check the data that the user inserts in the website I use JavaScript, so that there is a front-end check to be sure that the user doesn't enter invalid data or make SQL injection, as explained in the "Implementation" chapter.

1.3 Scopo

Lo scopo del progetto è di creare un sito web che permetta ai proprietari di piccoli negozi di mettere in vendita i propri prodotti, e renderli accessibili a un grande numero di possibili clienti.

Il sito darà la possibilità di lavorarci a 3 "tipi" di utenti, i venditori, i clienti che cercano i prodotti e gli amministratori. Tutti avranno la possibilità di fare un login e accedere a "pagine personali".

All'interno della loro pagina i negozianti vedono tutti i prodotti che hanno inserito e possono cercarli tramite la categoria e il nome, se cliccano sul nome di un determinato prodotto hanno la possibilità di modificarne i dati e hanno anche la possibilità di aggiungerne di nuovi.

Una volta fatto il sito gli utenti potranno entrare nello spazio "pubblico" e vedere tutti i prodotti esistenti. C'è la possibilità da parte dei clienti di cercare anche per loro i prodotti tramite categoria e/o nome e inserire i prodotti interessanti nel carrello. Una volta riempito il carrello, ci sarà la possibilità di vederne tutte le informazioni tra cui il prezzo totale, per poter fare gli acquisti c'è bisogno di registrarsi, o accedere se si ha già un account, oppure accedere come utente temporaneo. Inoltre avranno la possibilità di decidere se andare a ritirare la merce in negozio o farsela mandare a casa, questo una volta finita la "spesa".

L'ultimo utente, ma non meno importante, è l'amministratore, che ha la possibilità di vedere tutte le informazioni dei vari negozi, e i negozianti, e modificarle, oppure aggiungerne di nuovi.

Per svolgere questo progetto ho utilizzato una struttura MVC con PHP e JavaScript come linguaggi per implementare il tutto.

2 Analisi

2.1 Analisi del dominio

Il prodotto verrà creato su un PC con installati i software necessari per creare un sito web (Apache, MySQL, PHP, ...).

Attualmente ci sono siti simili ad esso, che gestiscono la vendita e l'acquisto di prodotti ma sono principalmente di negozi singoli, il mio sito andrà a prendere più negozietti.

Può risultare molto utile per tutti i negozianti che hanno un piccolo negozio e vogliono arrivare a raggiungere un pubblico maggiore, essendo online è accessibile da molte persone. Anche il cliente spesso è molto più intenzionato a comprare online, soprattutto se c'è la possibilità di avere la spedizione senza dover andare in negozio.

La cosa bella è che il sito è accessibile a tutti e non c'è bisogno di grandi conoscenze per utilizzarlo, grazie al fatto che è "user friendly" e con una struttura semplice da comprendere per tutti.

Per poter rendere il sito comodo a tutti ci devono essere dei controlli sicuri per evitare che gli utenti creino dei problemi magari navigando in modo incorretto o dove non dovrebbero.

Per creare le varie pagine c'è la possibilità di prendere dei template esistenti, da cui prenderò spunto per fare il mio progetto, essendo che comunque di metodi per creare pagine simili a quella che devo fare io ce ne sono diversi.

2.2 Analisi dei costi

Categoria	Prezzo/h	Ore	Prezzo totale
Mano d'opera	62 Fr./h	155	9'610 Fr.

2.3 Analisi e specifica dei requisiti

ID: REQ-001	
Nome	Gestione dei dati
Priorità	1
Versione	1.0
Note	In un database verranno contenuti tutti i dati relativi agli utenti e alle informazioni che dovranno essere presenti nella pagina
Sotto requisiti	
001	Schema ER per gestire i dati

ID: REQ-002	
Nome	Pagina login commercianti
Priorità	1
Versione	1.0
Note	Nella pagina dei commercianti ci sarà inizialmente un login con registrazione per accedere alle pagine personali.
Sotto requisiti	
001	Pagina di login

ID: REQ-003	
Nome	Pagina messa in vendita oggetti
Priorità	1
Versione	1.0
Note	La pagina da cui un commerciante può mettere in vendita i prodotti inserendo tutte le informazioni relative ad esse
Sotto requisiti	
001	Mockup della pagina
002	Decidere dati da inserire
003	Creare gli input da inserire
004	Creare l'inserimento dei dati all'interno del database

ID: REQ-004	
Nome	Modifica oggetti in vendita
Priorità	1
Versione	1.0
Note	La pagina da cui si possono vedere tutti gli oggetti in vendita
Sotto requisiti	
001	Modal con i dati del prodotto
002	Controlli dei campi
003	Campi obbligatori

ID: REQ-005	
Nome	Pagina mostra oggetti
Priorità	1
Versione	1.0
Note	La pagina da cui si possono vedere tutti gli oggetti in vendita con tutte le informazioni relative e ricercarne alcuni specifici per categoria o per nome.
Sotto requisiti	
001	Visione di tutte le informazioni dei prodotti in vendita (Nome, prezzo, indirizzo, categoria e immagine)
002	Possibilità di ricercare i prodotti tramite per categoria o nome
003	Creare gli input da inserire
004	Creare l'inserimento dei dati all'interno del database
005	Input per la ricerca per nome e/o categoria

ID: REQ-006	
Nome	Pagina del carrello
Priorità	1
Versione	1.0
Note	Pagina in cui vengono mostrati tutti gli oggetti che ci sono nel carrello della sessione corrente
Sotto requisiti	
001	Mostra delle informazioni degli oggetti inseriti nel carrello.
002	Possibilità di stampa di un file pdf in fondo alla pagina.
003	Possibilità di acquisto degli oggetti.

ID: REQ-007	
Nome	Login/registrazione o ospite
Priorità	1
Versione	1.0
Note	Popup in cui l'utente accede o si registra o accede come ospite, si apre all'acquisto da parte dell'utente dopo aver riempito il carrello
Sotto requisiti	
001	Tab di login o accesso come ospite
002	Possibilità di accedere alla pagina di registrazione.

ID: REQ-007.1	
Nome	Pagina di registrazione
Priorità	1
Versione	1.0
Note	Pagina in cui i clienti si registrano al sito
Sotto requisiti	
001	Inserimento dei dati richiesti (Nome, cognome, email, num. Telefonico)
002	Inserimento dell'utente nel database.

ID: REQ-008	
Nome	Pagina di amministrazione
Priorità	1
Versione	1.0
Note	Pagina in cui l'amministratore può visualizzare tutti i negozi e premendo sul nome e visualizza le informazioni, incluso quelli del negoziante e ha la possibilità di modificare, archiviare o aggiungere nuovi corsi
Sotto requisiti	
001	Mostra dei negozi esistenti nel sito
002	Popup con le informazioni al click sul nome del negozio
003	Popup modifica al click del bottone
004	Archiviazione del negozio al click del bottone
005	Pagina di aggiunta negozi al click del bottone

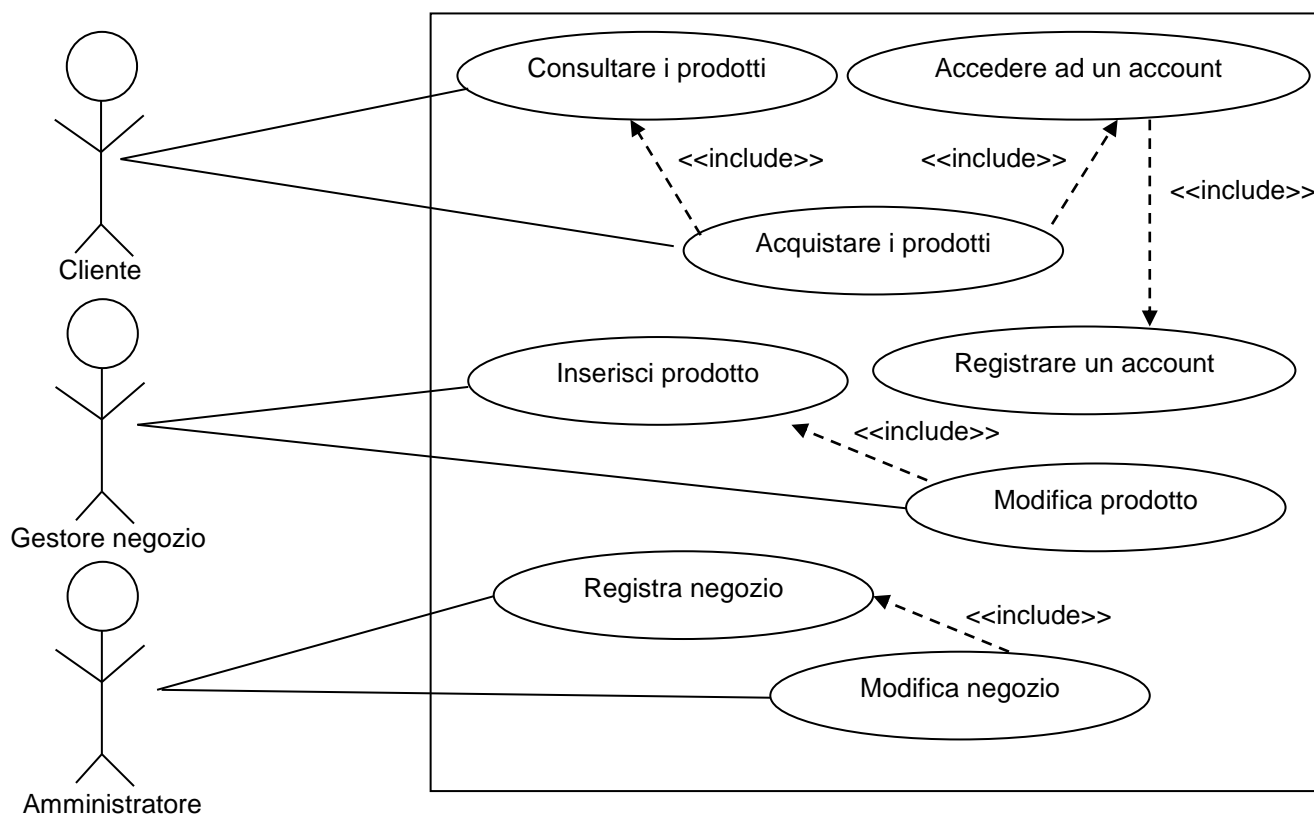
ID: REQ-009	
Nome	Pagina di aggiunta negozi
Priorità	1
Versione	1.0
Note	Pagina che si apre al click dell'amministratore nel bottone apposito e permette di inserire e informazioni del negozio e del negoziante per inserirli nel database.
Sotto requisiti	
001	Campi di inserimento dati negozio
002	Campi di inserimento dati negoziante
003	Inserimento dati nel database.

ID: REQ-010	
Nome	Pagine segnalazione prodotto ritirato
Priorità	2
Versione	1.0
Note	Pagina di gestione della vendita effettiva dei prodotti, pagina che gestiranno i lavoratori dei luoghi di ritiro e segneranno quando un cliente va a ritirare la merce precedentemente comprato e questo viene segnalato nel database
Sotto requisiti	
001	Lista carrelli da ritirare
002	Bottone ritiro avvenuto
003	Inserimento ritiro nel database

ID: REQ-011	
Nome	Carrello desiderate
Priorità	2
Versione	1.0
Note	Gestione del carrello desideri se un utente inserisce nel carrello un prodotto che non è disponibile perché la quantità è 0.
Sotto requisiti	
001	Inserimento nel carrello apposito

ID: REQ-012	
Nome	Pagamento del prodotto
Priorità	2
Versione	1.0
Note	Gestione del processo di pagamento online dei prodotti
Sotto requisiti	
001	Bottone di pagamento
002	Gestione dei dati di pagamento dell'utente
003	Gestione del pagamento online

2.4 Use case



2.5 Pianificazione

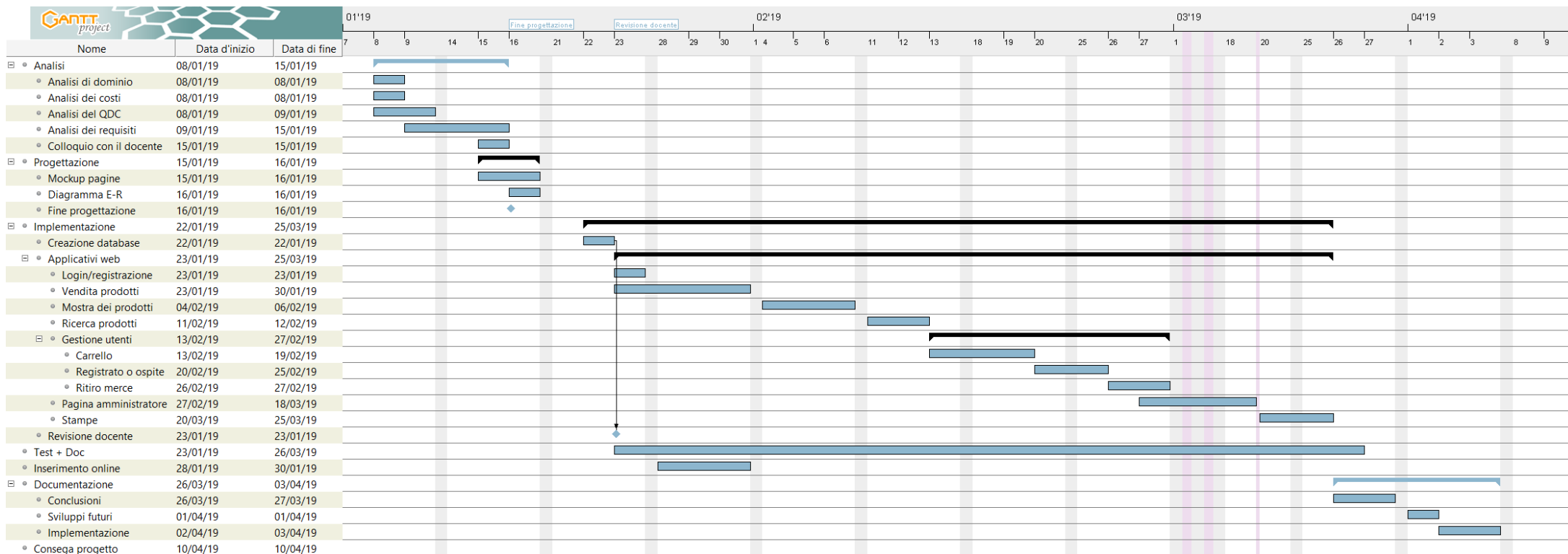


Figura 1 Gantt preventivo

Come si può notare dall'immagine qui sopra La parte dei test non è particolarmente argomentata, questo perché i test vanno di pari passo con l'implementazione, ovvero una volta completata una parte di implementazione essa viene testata direttamente. Un'altra cosa legata ai test è la documentazione perché la parte di documentazione di essi è legata assieme, dato che i test vengono tutti documentati, e di conseguenza il capitolo "Documentazione" contiene solo la parte finale, e nell'implementazione vengono inserite le cose finali e le correzioni ma la parte principale di essa viene fatto durante l'implementazione effettiva.

2.6 Analisi dei mezzi

2.6.1 Software

- MySQL Workbench 63 CE
- Google Chrome
- Apache24
- MySql56
- Php7
- NetBeans IDE 8.2
- TotalCommander 9.12
- Microsoft Visio Professional 2016
- File Zilla 3.4.1.2

2.6.2 Hardware

- 1 PC

3 Progettazione

3.1 Design dell'architettura del sistema

Il sito web utilizzerà una struttura di PHP MVC per riuscire ad utilizzare in modo più comodo tutti i file e le funzioni che ci saranno.

Il programma avrà una parte di login e registrazione, che permetteranno all'utente di entrare nel suo spazio a dipendenza del tipo di utente che è, se cliente, venditore o amministratore.

Ogni utente ha la possibilità di svolgere diverse azioni nel suo spazio:

Cliente:

- Ricerca prodotti
- Inserire prodotto nel carrello
- Acquistare prodotto
- Controllare mappa luoghi di ritiro

Venditore:

- Mettere in vendita prodotto
- Modificare prodotto

Amministratore:

- Registrare nuovo negozio
- Modificare negozio
- Archiviare negozio
- Gestire opzioni sito web

Le varie pagine sono mostrate meglio nel capitolo del [design delle interfacce](#)

La struttura delle cartelle sarà la classica MVC:

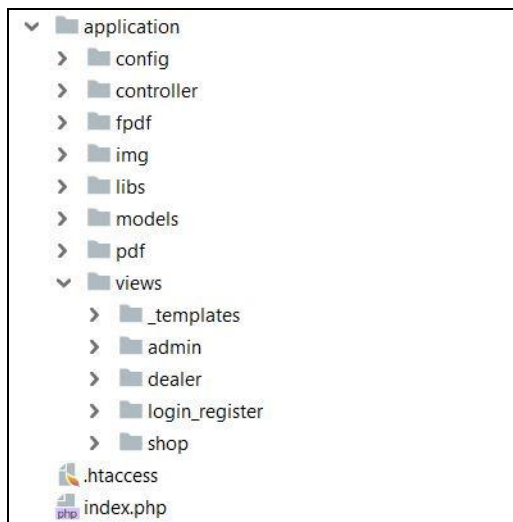


Figura 2 Struttura MVC

Inizialmente si apre il file “index” che poi va a richiamare tutto il resto per aprire le pagine con dei nomi delle funzioni contenute nei controller.

La cartella “models” è quella che va a interfacciarsi con il database, ovvero, contiene diverse classi, una per ogni tabella utilizzata del database, e per ogni classe ci sono delle azioni che vengono eseguite sul database alla determinata tabella.

Tramite le classi contenute nella classe Controller le “views”, ovvero le pagine grafiche e i file JavaScript, quindi la parte front-end del sito, riescono a lavorare sul database, in sintesi le viste comunicano con i Controller che a loro volta comunicano con i Model.

Oltre a queste ci sono cartelle in più. La cartella “img” contiene tutte le immagini dei prodotti utilizzate nel sito. La cartella “fpdf” contiene tutte le informazioni per utilizzare la classe esterna “FPDF” che viene utilizzata per creare i pdf con PHP. La cartella “pdf” invece contiene i pdf creati se vengono salvati.

3.2 Design dei dati e database

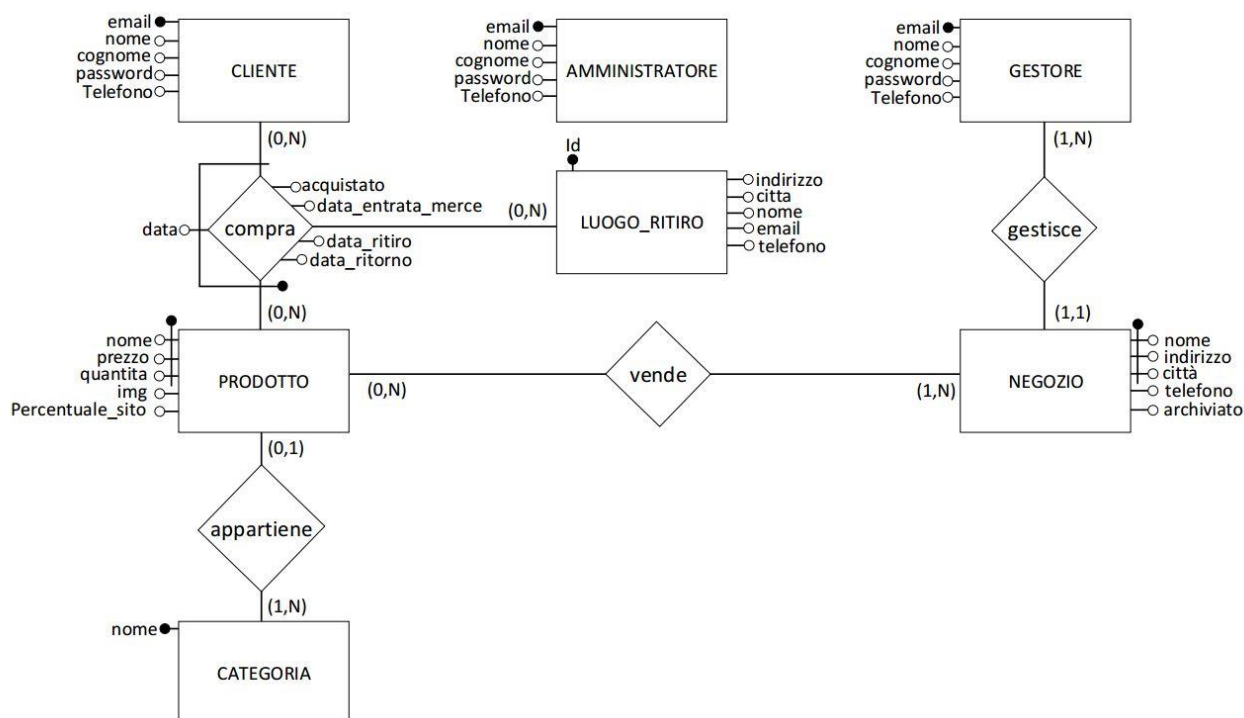


Figura 3 Schema E-R

Il database avrà la collezione di tutti i dati riguardanti gli utenti, di tutti i tipi, come si può notare l'amministratore non è collegato a nulla, questo perché lui non ha l'esigenza di lavorare con qualcos'altro, ma modifica solo i valori, l'importante è che si sappia che è amministratore.

La tabella "CATEGORIA" è stata creata perché le categorie saranno una lista ben definita ed è sempre meglio avere una tabella a parte se bisogna utilizzare una lista già scelta, per evitare che si scrivano i dati sbagliati i prodotti però possono anche non appartenere a nessuna categoria, questo perché magari quando viene aggiunto un prodotto non esiste ancora la categoria a cui appartiene o comunque si può lasciare vuoto.

La tabella "LUOGO_RITIRO" serve a contenere tutti quei posti in cui è possibile ritirare la merce acquistata, come spiega il nome, e servono per evitare che il cliente debba ogni volta andare in negozio, soprattutto perché se compra in negozi diversi poi si ritroverà a viaggiare molto per magari poche cose, quindi un posto in cui poter ritirare tutti risulta molto comodo, ma la sua chiave esterna collegata alla tabella ponte "compra" non è chiave primaria della tabella perché essendo che quella relazione fa anche da carrello, non è sicuro che quello che contiene venga acquistato e ritirato.

All'interno della tabella ponte "compra" alla fine sono inserite tutte le date, come si può notare, questo perché all'interno dell'ultima tabella aggiunta si possono vedere le informazioni del luogo di ritiro mentre nell'altra le informazioni di quali clienti prendono quali prodotti ed è importante sapere anche quando accade e quando un determinato utente va a ritirare quella merce, quando essa arriva e se non viene ritirata quando ritorna indietro, ma queste informazioni bisogna tenerle in quella tabella perché poi si possono ricollegare al luogo di ritiro grazie all'id di esso contenuto e scritto nella tabella ponte.

Quindi in questo modo si possono vedere tutte le informazioni della merce anche in base al luogo di ritiro.

In sintesi il carrello non è una cosa separata, ma semplicemente se il campo "acquistato" è a 0 vuol dire che il prodotto in questione si trova ancora nel carrello del cliente, quindi non c'è ancora una data di ritiro.

CLIENTE (email, nome, cognome, password, telefono)

Contiene tutti i clienti che si registrano al sito.

AMMINISTRATORE (email, nome, cognome, password, telefono)

Contiene tutti gli amministratori che si registrano al sito.

GESTORE (*email, nome, cognome, password, telefono*)

Contiene tutti i gestori che vengono registrati sul sito.

CATEGORIA (*nome*)

Contiene tutte le categorie a cui appartengono i prodotti.

PRODOTTO (*nome, prezzo, quantita, nome_categoria (FK)*, img, percentuale_sito*)

Contiene tutti i prodotti esistenti e la categoria a cui appartengono. Contiene il campo img che non contiene un'immagine ma il percorso in cui andare a prenderla, in modo che non bisogna andare a riempire di dati e appesantire il database. Il campo "percentuale_sito" contiene la percentuale che trattiene il sito su ogni prodotto che i negozi vendono per lo spazio.

La chiave composta è fatta così per fare in modo che lo stesso prodotto possa essere a più prezzi e quantità a dipendenza del negozio.

NEGOZIO (*nome, Indirizzo, città, telefono, archiviato, email_gestore(FK)*)

Contiene tutti i negozi e i gestori di essi.

LUOGO_RITIRO (*id, indirizzo, città, nome, telefono, email*)

Contiene tutte le informazioni dei luoghi di ritiro dove prendere la merce. Le date possono essere impostate a null perché è possibile che vengano impostate dopo la creazione.

COMPRA (*data, nome_prodotto(FK), prezzo_prodotto(FK), quantita_prodotto(FK), email_cliente(FK), id_luogo_ritiro(FK), data_entrata_merce*, data_ritiro*, data_ritorno*, acquistato*)

Contiene tutti i negozi e i gestori di essi, oltre a questo contiene anche il luogo di ritiro dove ritirare gli oggetti.

VENDE (*nome_prodotto(FK), prezzo_prodotto(FK), quantita_prodotto(FK), nome_negozio(FK), indirizzo_negozio(FK), città_negozio(FK)*)

Contiene le informazioni su quali negozi vendono quali prodotti.

3.3 Design delle interfacce

Gestione vendita

[Mappa](#)[Carrello](#)

Categoria ▼	<input type="text" value="Search..."/>	<input type="button" value="Cerca"/>
-------------	--	--------------------------------------



Prodotto XY
Categoria
Prezzo
Quantità

Figura 4 Pagina iniziale

Questa è la pagina iniziale del sito, in cui c'è una lista di tutti il prodotto e la possibilità di ricercarli tramite nome o categoria. All'interno della mostra del prodotto c'è la possibilità, grazie ad un bottone, di inserirlo nel carrello.

Sopra alla lista c'è la barra di navigazione tramite la quale si può accedere alla mappa dei negozi o al carrello.

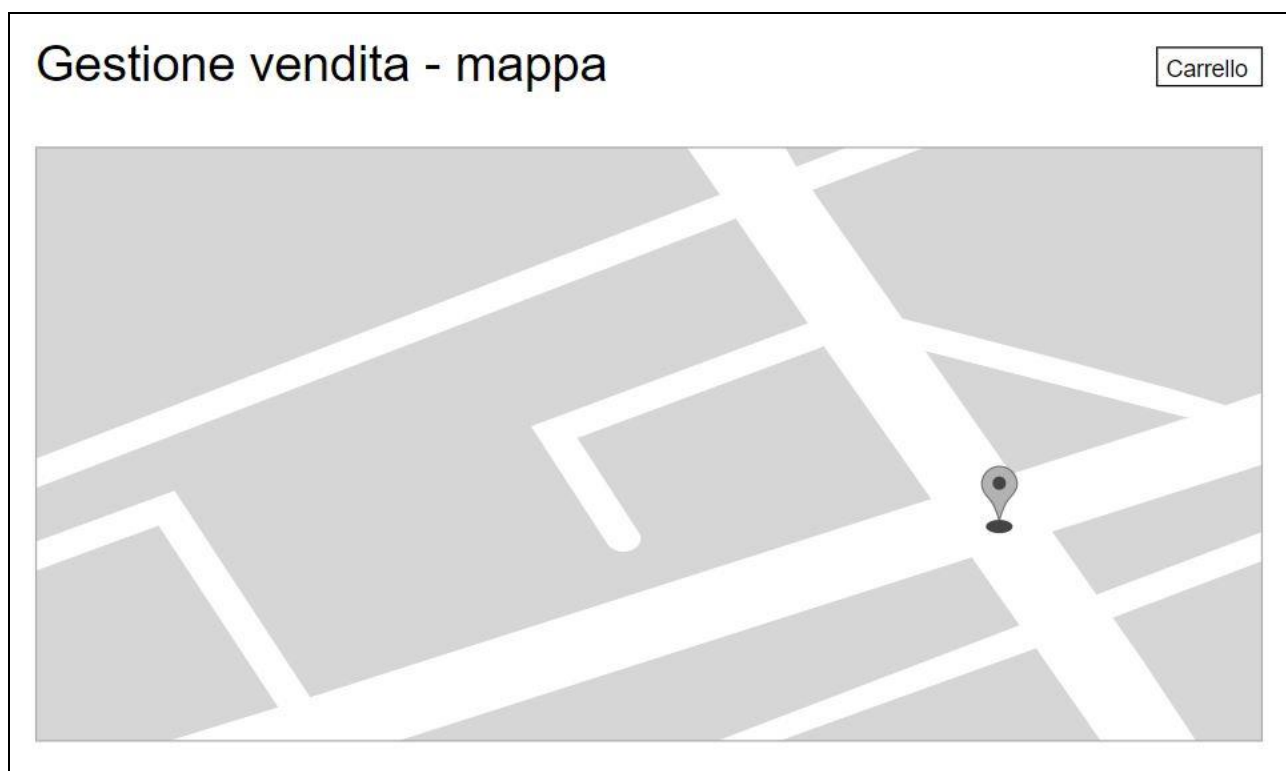


Figura 5 Pagina mappa

Quella mostrata qui sopra è la pagina della mappa, in cui è mostrata una mappa sopra la quale è possibile cliccare dei punti di segnalazione che apriranno un “popup” che mostra le informazioni del negozio che sta lì.

Gestione vendita - carrello

[Mappa](#)

	Nome Stabile
	Indirizzo
	Telefono
	Mail

<p>Prodotto XY</p> <p>Categoria</p> <p>Prezzo</p> <p>Quantità</p> <p>Consegna</p>
--

Prezzo: X Fr.

Figura 6 Pagina carrello

L'ultima pagina a cui i clienti hanno accesso è la pagina del carrello, che viene svuotata ogni volta che si acquista o si esce dal browser, ed è strutturata in questo modo, viene mostrato il prodotto con le relative informazioni e al fianco le informazioni del negozio che l'ha messo in vendita.

Login	Utente ospite
<div>Email Address</div>	<div>Nome</div>
<div>Password</div>	<div>Cognome</div>
<div>Login</div>	<div>Indirizzo email</div>
<div>Registrati</div>	<div>Numero di telefono</div>
	<div><input checked="" type="checkbox"/> Non sono un robot</div>
	<div>Avanti</div>

Figura 7 Pagina login o utente ospite

Una volta che l'utente clicca su "compra" se non ha ancora fatto il login viene mostrata una pagina in cui può fare il login o accedere come ospite, come mostrato nell'immagine, e se non dovesse essere registrato viene data la possibilità di farlo.

Gestione vendita - registrazione

☒ Non sono un robot

Figura 8 Pagina di registrazione

Questa è la pagina in cui l'utente può registrarsi.

Gestione vendita - login

Figura 9 Pagina login Negozianti/Amministratori

Gli utenti amministratori o i negozianti hanno la necessità di accedere alle loro pagine quando entrano nella loro pagina dedicata.

Gestione vendita - negozi

Negozio X [Modifica](#) [Archivia](#)

Negozio Y [Modifica](#) [Archivia](#)



Aggiungi negozio

Figura 10 Pagina negozi amministratore

L'amministratore una volta che accede inizialmente vede una pagina con la lista di tutti i negozi e cliccandoci sopra appare un popup che ne mostra le informazioni. Oltre a questo può lavorare con i negozi o aggiungere di nuove.

Gestione vendita - modifica

Negozio

Nome

Indirizzo

Telefono

Mail

Modifica

Negoziante

Nome

Cognome

Indirizzo email

Numero di telefono

Modifica

Figura 11 Pagina modifica negozi

La pagina di modifica dei negozi permette di modificare le informazioni del negozio stesso o del negoziante che lo tiene, questo è fatto diviso per impedire che bisogna modificare tutto se si vuole cambiare solo un'informazione in uno dei due punti.

Gestione vendita - aggiunta

<h3>Negozio</h3> <div style="margin-bottom: 5px;"><input type="text" value="Nome"/></div> <div style="margin-bottom: 5px;"><input type="text" value="Indirizzo"/></div> <div style="margin-bottom: 5px;"><input type="text" value="Telefono"/></div> <div style="margin-bottom: 5px;"><input type="text" value="Mail"/></div>	<h3>Negoziante</h3> <div style="margin-bottom: 5px;"><input type="text" value="Nome"/></div> <div style="margin-bottom: 5px;"><input type="text" value="Cognome"/></div> <div style="margin-bottom: 5px;"><input type="text" value="Indirizzo email"/></div> <div style="margin-bottom: 5px;"><input type="text" value="Numero di telefono"/></div> <div style="margin-bottom: 5px;"><input type="text" value="Password"/></div>
---	--

Figura 12 pagina aggiunta negozio

La pagina di creazione di nuovi corsi è molto simile quella dell'aggiunta dei negozi ma in questo caso si aggiunge tutto assieme e non si separano negozio e negoziante e in più si può decidere la password dell'utente, che poi gli andrà convocata una volta creato.

Gestione vendita - prodotti

▼



Prodotto XY
 Categoria
 Prezzo
 Quantità
 Consegna



Figura 13 Pagina gestione prodotti

Se il negoziante accede al suo sito invece avrà la possibilità di vedere e cercare tutti i prodotti che ha inserito all'interno del sito, e potrà anche lui decidere se modificare un prodotto o aggiungerne di nuovi.

Figura 14 Popup modifica prodotto

Se si clicca sul bottone “modifica” in un prodotto appare un popup che permette di modificare le informazioni del determinato prodotto, tra cui anche l'immagine mostrata sul sito.

Figura 15 Pagina creazione prodotto

Se si prova ad aggiungere un nuovo prodotto vien visualizzata una nuova pagina, simil al popup ma on la possibilità di aggiungere un nuovo prodotto, quindi con i campi vuoti, e non modificarne uno esistente. Se si lascia l'immagine vuota nel sito verranno mostrati solo i dati ma senza errori.

4 Implementazione

L'implementazione non tratterà tutto il codice, ma solamente le parti fondamentali o un po' particolari, con eventuali spiegazioni, per approfondimenti si possono trovare tutte le informazioni relative al progetto all'interno della repository GitHub al link: <https://github.com/giairomauro/ProgettoVenditaPiccolaNegoziante>.

Nel link si possono trovare:

- Codice Sorgente: <https://github.com/giairomauro/ProgettoVenditaPiccolaNegoziante/tree/master/Implementazione/MVC>
- Diari: <https://github.com/giairomauro/ProgettoVenditaPiccolaNegoziante/tree/master/Diari>
- Molto altro.

4.1 Grafica pagine

Questo primo sotto capitolo mostra la parte grafica delle pagine.

La creazione delle pagine è avvenuta modificando un template già esistente scaricato da Internet al link:

<https://colorlib.com/wp/template/essence/>.

Il template offriva una struttura di file html, css, e anche JavaScript per alcune funzioni speciali, motivo per cui non è mostrato tutto il codice per generare le pagine se non le parti veramente importanti.

4.1.1 Cliente

La prima pagina che si vede se si accede alla pagina dalla parte del cliente è quella mostrata qui sotto



Figura 16 pagina iniziale clienti

Oltre a questa pagina sarà possibile visualizzare una pagina con una lista di tutti i prodotti, che è quasi uguale a quella mostrata più avanti.

La prima pagina importante per il cliente è la pagina di login e registrazione dell'utente.

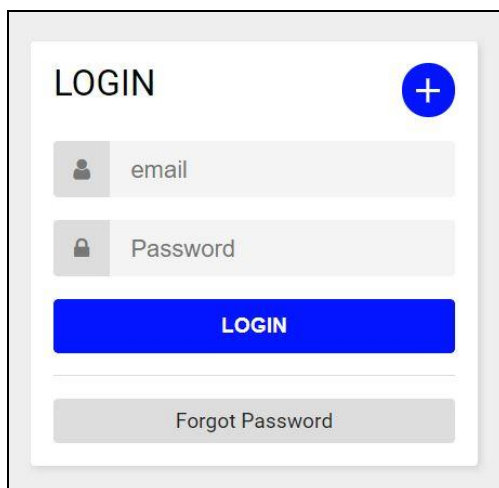


Figura 17 Pagina di login

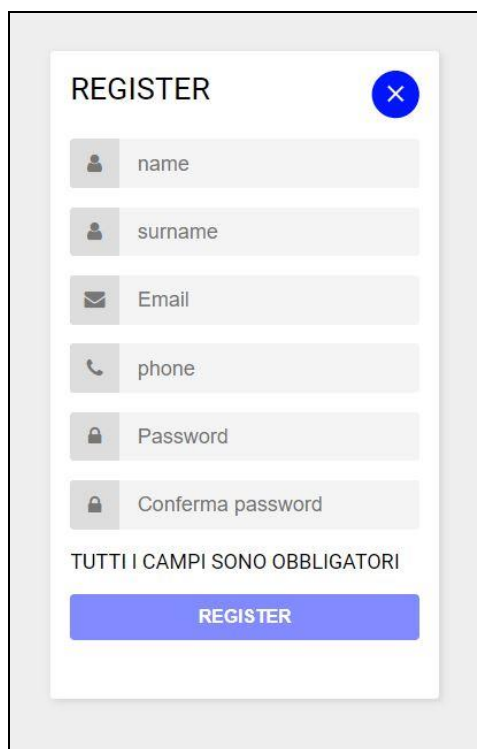


Figura 18 pagina di registrazione

Come si vede dalle immagini, la pagina non mostra direttamente sia login che registrazione, ma è fatto su 2 tab differenti e premendo il bottone blu in alto a destra si può accedere ad entrambe le pagine.

Una volta fatto l'accesso l'utente vedrà la stessa pagina che senza login ma in più c'è l'icona del carrello e non più quella del login, l'omino in alto a destra, ma quella del log out, come mostrato dall'immagine che segue.

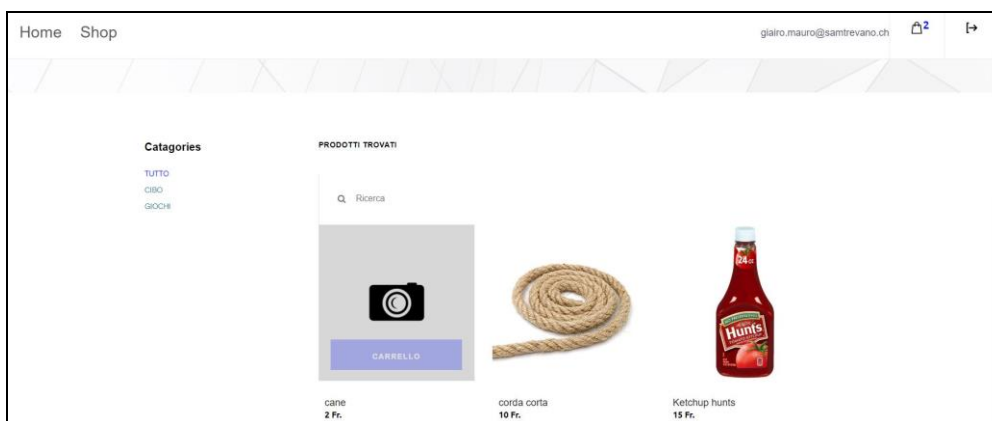


Figura 19 pagina prodotti login cliente

All'interno della pagina si possono cercare i prodotti sia tramite nome che tramite categoria. In alto a destra si può notare una borsa della spesa. Se si clicca su di essa si può vedere il carrello con tutti i dati e le informazioni di essi.

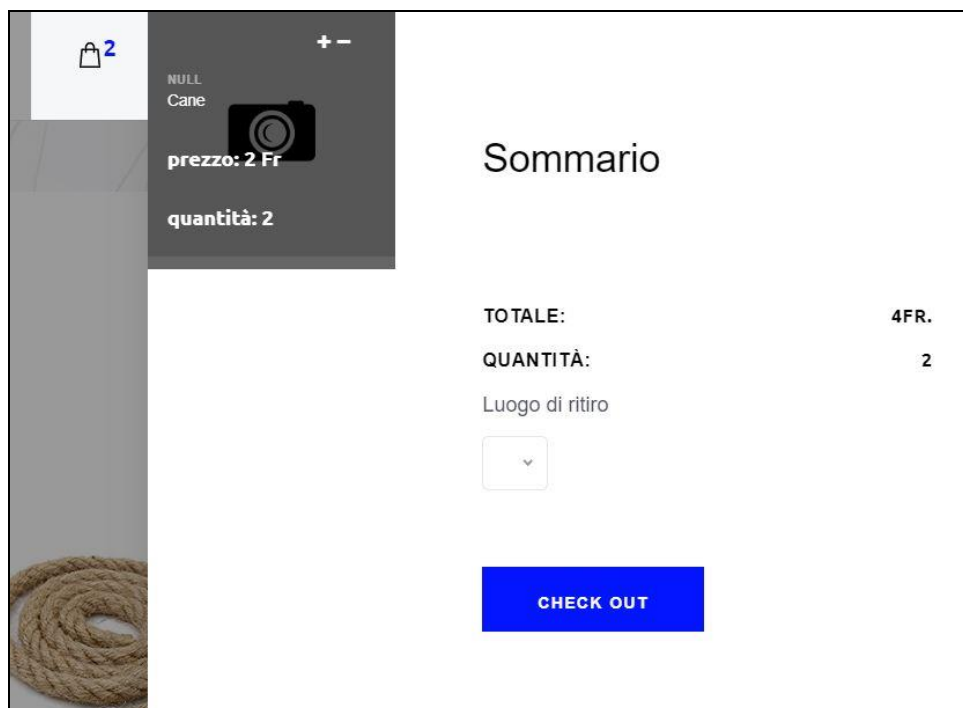


Figura 20 carrello dell'utente

All'interno della pagina si può scegliere un luogo di ritiro e dopodiché verranno mostrati i dettagli in un file pdf.


Figura 21 pdf riassunto acquisto

Il gestore ha anch'esso la pagina di login, che è quasi uguale a quella dei clienti ma con la differenza che può fare solo quello, non ha la possibilità di registrarsi per conto proprio. Una volta fatto il login verrà reindirizzato ad una pagina contenente una lista di tutti i prodotti che ha pubblicato e anche in questo caso ci sarà la possibilità di ricercare tramite la categoria.

Figura 22 pagina iniziale gestore

Versione: 10.04.2019

X
Il sito si prende il 10% del guadagno



Inserimento prodotto

CATEGORIA

NEGOZIO

Nome


Prezzo

Quantità

INSERISCI

Figura 23 modal di inserimento prodotto

X
Il sito si prende il 10% del guadagno



Modifica prodotto

CATEGORIA

NEGOZIO

NOME

PREZZO

QUANTITÀ

MODIFICA

Figura 24 modal modifica prodotto

4.1.3 Amministratore

Anche l'amministratore ha la pagina di login come quella dei venditori, questo perché anche loro non si registrano per conto loro ma vanno registrati hardcoded nel database.

Una volta eseguito il login l'amministratore avrà una pagina simile a quella dei gestori ma con la differenza che non vengono visualizzati i prodotti ma i negozi.



Figura 25 pagina iniziale amministratore

In questa pagina l'utente ha le stesse possibilità che hanno i venditori con i prodotti, aggiungere o modificare ma modificano delle informazioni differenti perché in questo caso si lavora anche sui venditori.

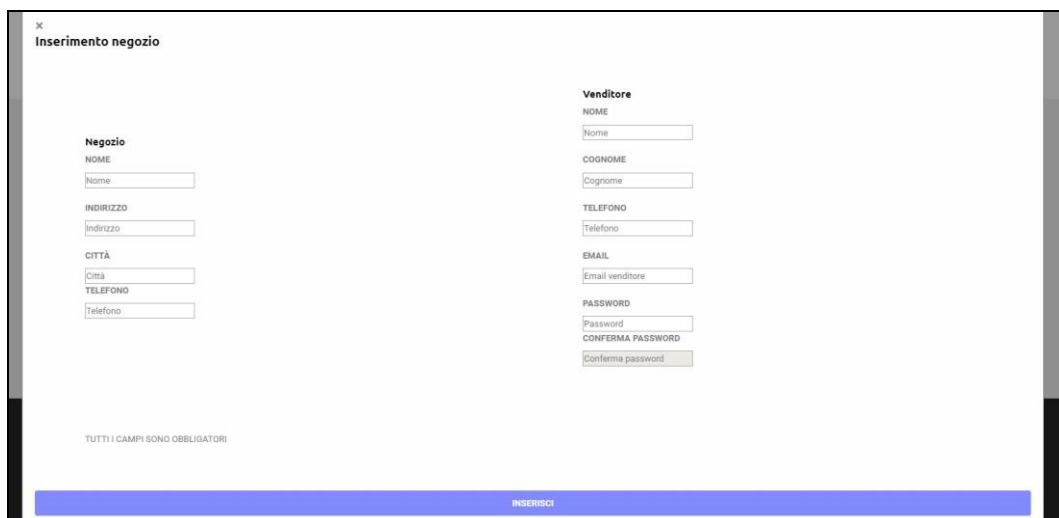


Figura 26 modal inserimento negozi

Negozio	Venditore
NOME <input type="text" value="Moreno dove tutto costa n"/>	NOME <input type="text" value="Salvatore"/>
INDIRIZZO <input type="text" value="Via campana 5"/>	COGNOME <input type="text" value="Pamino"/>
CITTÀ <input type="text" value="Lugano"/>	TELEFONO <input type="text" value="+41792380139"/>
TELEFONO <input type="text" value="+410914567738"/>	EMAIL <input type="text" value="s.pamo@gmail.com"/>
TUTTI I CAMPI SONO OBBLIGATORI	
<input type="button" value="MODIFICA"/>	

Figura 27 modal modifica prodotto

4.1.4 JavaScript

Per i controlli prima di inviare i file o l'inserimento di dati presi dal database viene usato solamente JavaScript, per ogni cartella delle "views" c'è una cartella JavaScript.

Nella foto che segue viene mostrato un esempio di controllo dei campi per assicurarsi il corretto inserimento da parte dell'utente.

```

20  var regLetters = /^[u00c0-\u017E a-zA-Z\']+$;/;
21  var regNumbers = /^[0-9\']+$;/;
22  function convalidate(value, id, regexp) {
23
24      //Variabile del campo
25      var id = document.getElementById(id);
26
27      //Se il campo è vuoto e non rispetta l'espressione regolare
28      if (!regexp.test(value)) {
29
30          //Colora il testo in rosso e segna l'errore.
31          id.style.color = "red";
32          checkInput = false;
33          //Altrimenti
34      } else {
35          //Colora il testo di nero
36          id.style.color = "black";
37          checkInput = true;
38      }
39      confirm();
40  }
41
42  /**
43   * Funzione che conferma e attiva o disattiva il bottone
44   */
45  function confirm() {
46      var submit = document.getElementById( elementId: "insertProduct");
47
48      if(checkInput){
49          submit.disabled = false;
50      }else{
51          submit.disabled = true;
52      }
53  }

```

Figura 28 controllo dati JavaScript

Come si può notare i controlli vengono fatti su un'espressione regolare, questo controllo viene fatto bene o male allo stesso modo in tutti i posti.

Per prendere i dati o comunque andare ad interrogare il database tramite JavaScript invece viene fatto tramite Ajax, richiamando la funzione controller che poi richiamerà un model che interrogherà il database, come l'immagine che segue.

```
295 function fileShop(id){
296     var id = id.split(".");
297     console.log(id);
298     xhttp.onreadystatechange = function () {
299         if (this.readyState === 4 && this.status === 200) {
300             location.reload();
301         }
302     }
303     xhttp.open( method: "POST", url: "/gestionevendita2018/admin/fileShop", async: true);
304     xhttp.setRequestHeader( name: "Content-Type", value: "application/x-www-form-urlencoded");
305     xhttp.send( body: "&name="+ id[0] + "&address="+ id[1] + "&city="+ id[2]);
306 }
```

Figura 29 utilizzo di Ajax

L'esempio nell'immagine è quello che serve per archiviare i negozi, e Ajax si occupa di richiamare il controller che lo fa, dopodiché una volta eseguito e arrivato il ritorno ricarica la pagina.

4.2 MVC

Come spiegato precedentemente il sito utilizza una struttura MVC, questo implica l'utilizzo di diverse classi e cartelle, ma partiamo dall'inizio.

4.2.1 Htaccess

Come si può notare nella foto che mostra la struttura delle cartelle, prima di entrare nei file effettivi che servono a mostrare il file ci sono altri 2 file, tra cui uno denominato ".htaccess", questo file serve a mostrare al sito che struttura deve utilizzare e come ritrasformare i link che gli vengono passati.

```
1 Options -MultiViews
2
3 RewriteEngine On
4
5 Options -Indexes
6
7 RewriteBase /gestionevendita2018/
8
9 RewriteCond %{REQUEST_FILENAME} !-d
10 RewriteCond %{REQUEST_FILENAME} !-f
11 RewriteCond %{REQUEST_FILENAME} !-l
12
13 RewriteRule ^(.+)$ index.php?url=$1 [QSA,L]
```

Figura 30 file .htaccess

Con le prime 3 righe viene impostato il fatto che deve controllare il link che gli viene passato e non i valori standard che andrebbe ad utilizzare.

A riga 7 viene definita la cartella da aprire per prima in cui cercare il file index da aprire per primo, la base della struttura. Poi vengono definite delle condizioni secondo cui non esegue le funzioni per le directory per i file e per i link. Alla fine viene impostata una nuova regola che dice di prendere i link e ritrasformarli a parte il root.

4.2.2 Index file

Un altro file importante prima di accedere alle classi di lavoro è il file “index.php” principale, ovvero quello che si va a richiamare appena si apre il sito.

```
<?php
2
3 // carico il file di configurazione
4 require 'application/config/config.php';
5
6 // carico le classi dell'applicazione
7 require 'application/libs/application.php';
8
9 // faccio partire l'applicazione
$app = new Application();
```

Figura 31 File index

Questo file si occupa di prendere i file di configurazione e di “application”, spiegati di seguito, e aprire il costruttore della classe application, essendo che questa classe viene richiamato appena si apre il sito.

```
<?php
2 session_start();
3 class Application
4 {
5     private $url_controller = null;
6     private $url_action = null;
7     private $url_parameter_1 = null;
8     private $url_parameter_2 = null;
9     private $url_parameter_3 = null;
10
11     public function __construct()
12     {
13
14         $this->splitUrl(); //funzione da creare per dividere l'URL
15         if (file_exists('./application/controller/' . $this->url_controller . '.php')) {
16             require './application/controller/' . $this->url_controller . '.php';
17             $controller = ucfirst($this->url_controller);
18             $this->url_controller = new $controller();
19             if (method_exists($this->url_controller, $this->url_action)) {
20
21                 if (isset($this->url_parameter_3)) {
22                     $this->url_controller->{$this->url_action}($this->url_parameter_1, $this->url_parameter_2,
23                     $this->url_parameter_3);
24                 } elseif (isset($this->url_parameter_2)) {
25                     $this->url_controller->{$this->url_action}($this->url_parameter_1, $this->url_parameter_2);
26                 } elseif (isset($this->url_parameter_1)) {
27                     $this->url_controller->{$this->url_action}($this->url_parameter_1);
28                 } else {
29                     $this->url_controller->{$this->url_action}();
30                 }
31             } else {
32                 $this->url_controller->index();
33             }
34         } else {
35             require './application/controller/login.php';
36             $login = new Login();
37             $login->index();
38         }
39     }
40 }
```

Figura 32 File application

Questo è il file application, che nel costruttore si occupa di prendere l'url passato e dividerlo, grazie alla funzione splitUrl(), in modo che possa controllare tutta la struttura, la struttura del link dev'essere quindi fatta in questo modo, “link iniziale”/“classe da controllare”/“Metodo della classe da aprire”/“Altri eventuali cose da aprire”, ad esempio questa <http://saminfo.ch/gestionevendita2018/home/shop> va ad aprire il metodo “shop”

nella classe “Home” nella classe controller del sito. Il metodo splitUrl() si occupa quindi semplicemente di dividere il sito agli Slash e inseriscono i dati nelle variabili adatte, come mostra l'immagine qui sotto.

```

44 private function splitUrl()
45 {
46     if (isset($_GET['url'])) {
47
48         // tolgo il carattere / dalla fine della stringa
49         $url = rtrim($_GET['url'], '/');
50         //rimuove tutti i caratteri illegali dall'URL
51         $url = filter_var($url, FILTER_SANITIZE_URL);
52         //divido in un array in base al carattere /
53         $url = explode('/', $url);
54
55         // divido le parti dell'url in base a controller, azione e 3 parametri
56         $this->url_controller = (isset($url[0]) ? $url[0] : null);
57         $this->url_action = (isset($url[1]) ? $url[1] : null);
58         $this->url_parameter_1 = (isset($url[2]) ? $url[2] : null);
59         $this->url_parameter_2 = (isset($url[3]) ? $url[3] : null);
60         $this->url_parameter_3 = (isset($url[4]) ? $url[4] : null);
61     }
62 }
63

```

Figura 33 Funzione splitUrl

```

14 error_reporting( level: E_ALL);
15 ini_set( varname: "display_errors", newvalue: 1);
16
17 /**
18  * Configurazione di : URL del progetto
19  */
20 define('URL', 'http://samtinfo.ch/gestionevendita2018/');
21

```

Figura 34 File config

Questo è il contenuto del file “config.php” che semplicemente mostra dei possibili errori e imposta una variabile “URL” che contiene il link iniziale del sito utilizzabile in tutte le classi della struttura.

4.2.3 application

La struttura presenta alla stessa posizione dei file sopra spiegati una cartella “application” in cui sono presenti altre cartelle utili al funzionamento del sito. Il contenuto delle cartelle “config” e “libs” è già spiegato nei sotto capitoli precedenti, come anche la cartella “views”. Anche le cartelle “fpdf”, “pdf” e img sono state spiegate, nel capitolo della Progettazione “Design dell’architettura del sistema”.

4.2.3.1 Controller

La classe controller, come spiegato brevemente in precedenza, si occupa di comunicare sia con le views che con i model. Questo viene fatto grazie a diverse classi create, ognuna creata in base a cosa deve fare.



Figura 35 cartella controller

Come si può notare dall’immagine esiste una classe per ogni “categoria” di lavoro.

La struttura di base delle classi è la stessa, come mostrato nella figura che segue, viene dichiarata la classe e istanziata la funzione iniziale “index”.

```

1  <?php /** @noinspection ALL */
2
3  /**
4   * Class Admin Classe che gestisce il lavoro delle informazioni.
5   */
6  class Admin
7  {
8
9      /**
10     * Apertura della pagina di login
11     */
12     public function index()
13     {
14         //Apro la pagina con il login
15         require_once 'application/controller/login.php';
16         $login = new Login();
17         $login->loginPageA();
18     }

```

Figura 36 inizio classi controller

La funzione iniziale “index” non è uguale per tutti i controller, per ognuno dipende dalla cosa principale che esso deve fare, in questo caso fa riferimento ad un altro controller per aprire la pagina di login, ma in altri potrebbe anche essere vuota, però anche se non ha utilizzi viene creata comunque.

Come detto i controller si occupano anche di aprire le pagine, questo viene fatto, come mostrato nella foto che segue, tramite i require.

```

26         if(isset($_SESSION['dealer'])) {
27             require 'application/views/_templates/header.php';
28             require 'application/views/dealer/static/header.php';
29             require 'application/views/dealer/shop.php';
30             require 'application/views/_templates/footer.php';
31         }else{
32             $this->index();
33         }

```

Figura 37 apertura pagine

Come si può notare prima di eseguire il contenuto importante della funzione viene controllato che si possa, ovvero che l'utente che dovrebbe farlo abbia eseguito il login, in questo caso il venditore, i tipi di utenti sono 3 “customer”, “dealer” e “admin”, questi vengono salvati nelle variabili di sessione. Per aprire le pagine la funzione fa un “require” di tutti i file che deve aprire e scrive il contenuto in sequenza in una pagina a parte, in modo che se ci sono tag html vengono interpretati correttamente e viene aperta una pagina. Questo viene fatto perché in questo modo non si vede il link con l'estensione in alto ma si vedono solo il nome della classe e delle funzioni.

I controller oltre ad occuparsi di aprire le pagine servono anche ad utilizzare funzioni PHP normali, come potrebbe essere quella che crea il file pdf o quella per fare il log out, che lavora solo sulla classe.

```

5     class Logout
6     {
7         public function dealer()
8         {
9             session_destroy();
10            header( string: "location: http://localhost:8042/MVC/dealer");
11        }

```

Figura 38 classe log out

Le classe log out distrugge la sessione, in modo da eliminare le variabili di session, e poi riporta alla pagina iniziale, anche se l'immagine mostra in locale è attuabile anche online inserendo il corretto link a qui reindirizzarla.

Oppure fare riferimento ad un'altra funzione in una classe model se deve lavorare con il database.

Per farlo però non può semplicemente richiamare la funzione che gli serve, ma ha bisogno di controllare altre cose, come mostra quella mostrata qui sotto.

```

40 public function getShops(){
41     //Controllo che la funzione entri in POST
42     if ($_SERVER["REQUEST_METHOD"] == "POST") {
43         //Controllo che sia aperta la sessione e abbia fatto il login un amministratore
44         if (isset($_SESSION['admin'])) {
45             //Prendo la classe model
46             require_once 'application/models/shop.php';
47             $shop = new ShopModel();
48
49             $shops = $shop->getShops();
50
51             //Stampo con json i valori dei coach
52             header( string: 'Content-Type: application/json');
53             echo json_encode($shops);
54         }else{
55             header( string: "location: javascript://history.back()");
56         }
57     }else{
58         header( string: "location: javascript://history.back()");
59     }
60 }

```

Figura 39 funzione che prende dati dal database

Prima di controllare l'utente viene fatto un controllo per assicurarsi che sia stata fatta una chiamata in "POST" perché questo tipo di funzione vengono richiamate solo da dei form o da Ajax.

Viene usato solo POST in tutto il progetto per questione di sicurezza, essendo che nasconde il link completo con gli argomenti.

Nel caso della funzione mostrata sopra viene solo fatto il controllo poi presa la classe e richiamata la funzione, questo perché la funzione deve prendere dei dati quindi non ha bisogno di particolari informazioni, ma in caso dovesse inserire o modificare dei dati la situazione è leggermente diversa.

```

22 public function addToCart()
23 {
24     //Se la sessione è aperta apro le pagine altrimenti no
25     if(isset($_SESSION['customer'])) {
26
27         //Prendo la classe model
28         require_once 'application/models/buy.php';
29         $buy = new BuyModel();
30
31         //Prendo le variabili passate dal POST
32         $name = isset($_POST["name"])? $_POST["name"] : null;
33         $price = isset($_POST["price"])? $_POST["price"] : null;
34         $quantity = isset($_POST["quantity"])? $_POST["quantity"] : null;
35
36         //Se entrambi i campi non sono vuoti
37         if ($name != null && $price != null && $quantity != null) {
38             $buy->insertData($name, $price, $quantity, $_SESSION['customer']);
39         }
40         //Altrimenti
41     }else{
42
43     }
44 }

```

Figura 40 inserimento dati nel database

La funzione si occupa di inserire i dati aggiunti al carrello, di conseguenza riceve dei dati con il "POS", che poi andrà a passare al model, ed essi vengono presi con un controllo di esistenza fatto con un operatore ternario che controlla che la variabile esista.

Queste sono le cose principali delle classi controller, per ulteriori informazioni consultare quanto scritto all'inizio dell'implementazione.

4.2.3.2 Model

Le classi presenti nella cartella model come spiegato si occupano di gestire i collegamenti con il database, ovvero tutte le operazioni che vengono fatte su di esso. Per tutte le classi ho deciso di utilizzare PDO, essendo più sicuro ma anche migliore dato che permette di cambiare in qualsiasi momento database con comodità e sicurezza.

Queste classi sono un po' più semplici di quelle del controller, nel senso che hanno meno possibilità, l'unica cosa che fanno è eseguire delle query sul database, la parte complicata è la logica per poterle utilizzare nel modo corretto.

Ad esempio per fare far funzionare il carrello bastava inserire, modificare o eliminare i dati su una determinata tabella ponte, ma per farlo bisogna anche cambiare il dato del prodotto perché cambia la quantità presente in negozio, e bisogna fare attenzione a cambiare i dati nel modo giusto e prenderli anche nel modo giusto.

Prima di poter accedere alle informazioni del database però bisogna stabilire la connessione, per farlo io ho optato per creare una classe a parte, principale, che si occupi solo di quello, la classe "connection".

```

3  class Connection extends PDO
4  {
5      private $servername = "efof.myd.infomaniak.com";
6      private $username = "efof_gestvend";
7      private $password =
8      private $dbName = "efof_gestvend";
9
10     /**
11      * Costruttore vuoto della classe.
12      */
13     function __construct() {}
14
15     /**
16      * Costruttore della classe che modifica le variabili del database.
17      * @param $servername Nuovo nome del server.
18      * @param $username Nuovo nome utente.
19      * @param $password Nuova password.
20      * @param $dbName Nuovo nome del database.
21      */
22     function __construct1($servername, $username, $password, $dbName)
23     {
24         $this->servername = $servername;
25         $this->username = $username;
26         $this->password = $password;
27         $this->dbName = $dbName;
28     }
29
30     /**
31      * Funzione che stabilisce la connessione al database
32      */
33     public function sqlConnection()
34     {
35         $conn = "";
36         try{
37             //Variabile di connessione con PDO
38             $conn = new PDO( dsn: 'mysql:host='. $this->servername .';dbname='. $this->dbName .';port=3306', $this->username, $this->password);
39         }
40         catch (PDOException $e) {
41             echo $e->getMessage();
42         }
43
44         //Ritorno la variabile
45         return $conn;
46     }
47 }

```

Figura 41 classe connection, connessione al database

La classe ha delle variabili già predefinite che contengono le informazioni per accedere al database, queste sono hardcoded perché questo progetto è utilizzato solamente con un database e risultava inutile renderlo dinamico, ma c'è comunque un costruttore se si volesse cambiarli. L'unica funzione esistente è quella che stabilisce la connessione tramite le variabili globali precedentemente impostate.

Le altre classi sono più o meno simili, per cui non verranno mostrate tutte le funzioni ma solo alcune per mostrare la base, per vedere tutte le funzioni e le informazioni consultare quanto scritto all'inizio dell'implementazione.

L'inizio delle classi è sempre la stessa, come mostrato nella foto che segue, viene dichiarata la classe e creato il costruttore.

Gestione della vendita dei prodotti dei piccoli negozianti

```
1  <?php /** @noinspection ALL */
2
3  class ShopModel extends PDO
4  {
5      private $connection;
6
7      public function __construct()
8      {
9          require_once 'application/models/connection.php';
10         $this->connection = new Connection();
11     }
```

Figura 42 inizio classi model

Come si può notare nel costruttore viene richiamata la classe connessione e istanziata una variabile con il costruttore vuoto, quindi per quella verranno usate le variabili già create.

Dopodiché le altre funzioni andranno a prendere la variabile “\$this->connection” per potersi collegare al database ed eseguire le query.

Per mostrare degli esempi di funzioni mostrerò le 3 principali dalla classe “ShopModel” ovvero quella che va a lavorare principalmente sulla tabella “negozio” del database.

```
76  public function getShops()
77  {
78      //Connetto al database
79      $conn = $this->connection->sqlConnection();
80
81      //Prendo i dati dei negozi
82      $sql = $conn->prepare("SELECT nome, indirizzo, citta FROM negozio");
83
84      //Eseguo la query
85      $sql->execute();
86
87      $dataArray = array();
88      //Se ci sono dei valori
89      if ($sql->rowCount() > 1) {
90          // Ciclo tutti i valori
91          while ($row = $sql->fetch()) {
92              array_push( &array: $dataArray, $row);
93          }
94      } else if ($sql->rowCount() == 1) {
95          array_push( &array: $dataArray, $sql->fetch());
96      }
97      $conn = null;
98      return $dataArray;
99  }
```

Figura 43 funzione che prende i dati dalla tabella

L'esempio sopra mostrato viene utilizzato per prendere nome, cognome e città, ovvero le chiavi, di tutti i negozi esistenti, questo viene fatto creando una prepared Statement con la query da eseguire, e dopodiché eseguita. Essendo una select ritornerà dei dati e questi dati vengono presi, come mostrato nelle righe da 87 a 97, e inserite in un array che verrà ritornato. C'è un controllo perché nel caso in cui dovesse esserci solo un negozio inserisce nell'array solo quel valore mentre se sono di più li passa in un for.

Alla fine della funzione, come in ogni altra, viene chiusa la connessione semplicemente mettendo a null il valore della variabile che la stabiliva.

```

17 public function insertShop($name, $address, $city, $phone, $email){
18     //Connetto al database
19     $conn = $this->connection->sqlConnection();
20
21     //Inserisco i dati del nuovo negozio
22     $sql = $conn->prepare("INSERT INTO negozio(nome, indirizzo, citta, telefono, email_gestore)
23     values(:name, :address, :city, :phone, :email)");
24     $sql->bindParam(':name', $name, PDO::PARAM_STR);
25     $sql->bindParam(':address', $address, PDO::PARAM_STR);
26     $sql->bindParam(':city', $city, PDO::PARAM_STR);
27     $sql->bindParam(':phone', $phone, PDO::PARAM_STR);
28     $sql->bindParam(':email', $email, PDO::PARAM_STR);
29
30     //Se ci sono dei valori
31     if($sql->execute()) {
32         echo "LO";
33     }else {
34         print_r($sql->errorInfo());
35     }
36     $conn = null;
37 }

```

Figura 44 inserimento dati nel database

La funzione sopra mostrata si occupa di andare ad inserire un nuovo valore all'interno del database, come la precedente anche in questo caso c'è una prepared statement che contiene la query, ma essendoci dei dati che devono essere inseriti dall'utente non vanno messi direttamente dentro la query, ma nella query va segnato un "indice" come ":name" e poi sotto con la funzione "bindParam" va segnalato a cosa corrisponde ogni indice, questo metodo permettere di evitare diversi attacchi come le SQLInjection.

La conclusione della funzione è più o meno la stessa ma in questo caso non vengono inseriti dati da nessuna parte ma si controlla che la query vada a buon fine, e in quel caso ritorno un valore altrimenti l'errore.

```

48 public function modifyShop($name, $address, $city, $phone, $email, $oldName, $oldAddress, $oldCity){
49     //Connetto al database
50     $conn = $this->connection->sqlConnection();
51
52     //Prendo i dati dell'utente in base alla mail
53     $sql = $conn->prepare("UPDATE negozio set nome = :name, indirizzo = :address, citta = :city, telefono = :phone, email_gestore = :email
54     WHERE nome LIKE :nameW AND indirizzo LIKE :addressW AND citta LIKE :cityW");
55     $sql->bindParam(':name', $name, PDO::PARAM_STR);
56     $sql->bindParam(':address', $address, PDO::PARAM_STR);
57     $sql->bindParam(':city', $city, PDO::PARAM_STR);
58     $sql->bindParam(':phone', $phone, PDO::PARAM_STR);
59     $sql->bindParam(':email', $email, PDO::PARAM_STR);
60     $sql->bindParam(':nameW', $oldName, PDO::PARAM_STR);
61     $sql->bindParam(':addressW', $oldAddress, PDO::PARAM_STR);
62     $sql->bindParam(':cityW', $oldCity, PDO::PARAM_STR);
63
64     //Se ci sono dei valori
65     if($sql->execute()) {
66         return true;
67     }else {
68         return $sql->errorInfo();
69     }
70     $conn = null;
71 }

```

Figura 45 modifica dati database

L'ultima funzione mostrata contiene la query per modificare i dati nel database, la struttura è più o meno la stessa dell'inserimento ma con una differenza nella query, perché giustamente è diversa, altrimenti i parametri sono inseriti nello stesso modo e anche l'esecuzione e la conclusione avvengono come la precedente.

5 Test

5.1 Protocollo di test

Test Case:	TC-001	Nome:	controlli input
Riferimento:			
Descrizione:	Test per assicurarsi che i controlli degli input nelle varie pagine funzionino correttamente e predano solo i formati adatti al campo richiesto		
Prerequisiti:	Pagina creata, Controlli implementati correttamente.		
Procedura:	<ol style="list-style-type: none"> 1. Inserire dei dati nei campi con il formato corretto. 2. Inserire de dati nei camp con il formato incorretto. 		
Risultati attesi:	Se i dati hanno un formato accettabile non succede niente, in caso contrario viene segnalato che il formato è incorretto scrivendo il testo in rosso e non permettendo di andare avanti.		

Test Case:	TC-002	Nome:	Pagina login – controlli inserimento dati.
Riferimento:	REQ-002		
Descrizione:	Testare che venga fatto correttamente il login con i relativi errori.		
Prerequisiti:	Pagina di login creata e implementazione del codice di login creata		
Procedura:	<ol style="list-style-type: none"> 1. Inserire e-mail e password inesistenti nel database. 2. Inserire e-mail corretta e password errata o il contrario. 3. Inserire i dati corretti di un utente attivo. 		
Risultati attesi:	In caso i dati siano corretti e l'utente attivo fa il login nella sua pagina, in caso contrario segnala con un alert il determinato errore e poi ritorna alla pagina di login.		

Test Case:	TC-003	Nome:	Pagina “messa in vendita oggetti” – controlli inserimento dati.
Riferimento:	REQ-003		
Descrizione:	Test per assicurarsi che i dati vengano inseriti correttamente all'interno del database dopo aver passato i controlli degli input		
Prerequisiti:	Pagina creata, implementazione codice di inserimento dati fatta correttamente		
Procedura:	<ol style="list-style-type: none"> 1. Accedere come venditore 2. Inserire i dati dell'oggetto, almeno quelli obbligatori. (L'immagine non lo è) 3. Premere il bottone di aggiunta. 		
Risultati attesi:	I dati vengono inseriti correttamente nel database nella tabella del prodotto.		

Test Case:	TC-004	Nome:	Pagina “messa in vendita oggetti” – modifica oggetti in vendita
Riferimento:	REQ-004		
Descrizione:	Test per assicurarsi che i dati dei prodotti messi in vendita si possano modificare		
Prerequisiti:	Pagina creata, implementazione codice di inserimento dati fatta correttamente		
Procedura:	<ol style="list-style-type: none"> 1. Cliccare sul nome di un oggetto 2. Modificare un dato 3. Premere il bottone “modifica” 		
Risultati attesi:	I dati vengono modificati nel database e mostrati modificati nella pagina e nel database.		

Test Case:	TC-005	Nome:	Pagina “vendita oggetti” – mostra dei prodotti
Riferimento:	REQ-005		
Descrizione:	Test per assicurarsi che i prodotti esistenti e con una quantità maggiore a 0 vengano mostrati correttamente.		
Prerequisiti:	Pagina creata e formattata in modo corretto.		
Procedura:	<ol style="list-style-type: none"> 1. Aprire la pagina principale del sito. 2. Accedere alla pagina “shop” 3. Scorrere per vedere i vari prodotti e le informazioni. 		
Risultati attesi:	I prodotti vengono mostrati correttamente con le relative informazioni.		

Test Case:	TC-006	Nome:	Pagina “vendita prodotti” – Ricerca
Riferimento:	REQ-005		
Descrizione:	Test per assicurarsi che la ricerca per nome e/o categoria funzioni correttamente.		
Prerequisiti:	Pagina creata, implementazione di ricerca e presa dei dati dal database.		
Procedura:	<ol style="list-style-type: none"> 1. Inserire qualcosa nel campo della ricerca 2. Selezionare una categoria 3. Premere il bottone di ricerca 		
Risultati attesi:	A dipendenza se si riempie uno solo o entrambi i campi alla ricerca vengono mostrati tutti i prodotti che soddisfano quei requisiti.		

Test Case:	TC-007	Nome:	Pagina “vendita oggetti” – inserimento dati nel carrello
Riferimento:	REQ-006		
Descrizione:	Test per assicurarsi che i dati vengano inseriti nel carrello al click nel bottone su un determinato prodotto.		
Prerequisiti:	Pagina creata, implementazione codice di inserimento dati fatta correttamente		
Procedura:	<ol style="list-style-type: none"> 1. Cliccare il bottone “carrello” su un prodotto. 		
Risultati attesi:	Il prodotto viene inserito nel carrello e salvato in modo provvisorio.		

Test Case:	TC-008	Nome:	"Carrello" – aggiunta modifica dati
Riferimento:	REQ-006		
Descrizione:	Test per assicurarsi che i dati modificati quando si aggiungono o tolgono direttamente dal carrello.		
Prerequisiti:	Pagina creata, implementazione dati del carrello non salvati in modo permanente.		
Procedura:	<ol style="list-style-type: none"> 1. Inserire un prodotto nel carrello 2. Aprire il carrello 3. Premere il "+" presente su uno dei prodotti nel carrello. 4. Premere il "-" presente su uno dei prodotti nel carrello. 		
Risultati attesi:	Se si preme il "+" viene aggiunto un prodotto e con il "-" viene tolto.		

Test Case:	TC-009	Nome:	Pagina "Carrello" – checkout
Riferimento:	REQ-007 REQ-008		
Descrizione:	Test per assicurarsi che si possa fare il checkout dei prodotti nel carrello, e viene creato un pdf di riassunto.		
Prerequisiti:	Pagina creata, script che crea il pdf implementato, implementazione dell'apparizione di esso fatta.		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il carrello con i prodotti all'interno. 2. Selezionare un luogo di ritiro. 3. Premere sul bottone "checkout". 		
Risultati attesi:	Una volta cliccato il bottone si apre una nuova pagina con un pdf che mostra il riassunto dell'acquisto.		

Test Case:	TC-010	Nome:	Pagina "registrazione" – registrazione utente
Riferimento:	REQ-008.1		
Descrizione:	Testare che venga registrato correttamente il nuovo utente.		
Prerequisiti:	Pagina di registrazione creata e implementazione del codice di registrazione creata. Test sulla pagina superati correttamente.		
Procedura:	<ol style="list-style-type: none"> 1. Inserire i dati corretti nei campi assicurandosi che nessuno sia non valido. 2. Inserire password che combacino. 3. Controllare la password. 4. Premere il bottone di registrazione. 		
Risultati attesi:	Se c'è qualche campo vuoto viene segnalato, in caso sia tutto corretto, l'utente viene registrato correttamente.		

Gestione della vendita dei prodotti dei piccoli negozianti

Test Case:	TC-011	Nome:	Pagina amministratore – mostra negozi
Riferimento:	REQ-008.1		
Descrizione:	Testare vengano mostrati tutti i negozi e maggiori informazioni al click sul nome di uno.		
Prerequisiti:	Pagina creata, lista di negozi creata, mostra degli oggetti creata, popup con maggiori informazioni sugli oggetti creata.		
Procedura:	<ol style="list-style-type: none"> 1. Aprire la pagina dell'amministrazione 2. Accedere come amministratore. 3. Cliccare sul nome di un negozio nella lista. 		
Risultati attesi:	<p>Al click su uno dei negozi vien aperto un popup che mostra maggiori informazioni su di esso e sul negoziante registrato ed è possibile modificarlo.</p> <p>Se si modificano dei dati e si preme il bottone di submit i dati vengono modificati nel database.</p>		

Test Case:	TC-012	Nome:	Pagina amministratore – archiviazione
Riferimento:	REQ-008.1		
Descrizione:	Testare che vengano archiviati i negozi al click sul bottone.		
Prerequisiti:	Pagina creata, prodotti mostrati correttamente, archiviazione negozi implementata		
Procedura:	<ol style="list-style-type: none"> 1. Premere il bottone “archivia” a fianco ad un negozio. 		
Risultati attesi:	Al click del bottone il negozio sparisce e viene archiviato.		

Test Case:	TC-013	Nome:	Pagina “creazione negozio” – creazione negozio
Riferimento:	REQ-008.1		
Descrizione:	Testare che al click sul bottone di aggiunta si apra una pagina in cui creare un nuovo negozio con un negoziante.		
Prerequisiti:	Pagina creata, prodotti mostrati correttamente, popup creato.		
Procedura:	<ol style="list-style-type: none"> 1. Premere il bottone “aggiungi negozio” in fondo a tutti i negozi. 2. Inserire i dati nella pagina che si apre e cliccare sul bottone. 		
Risultati attesi:	Al click del bottone viene aperta una pagina in cui si può creare un negozio inserendo i dati di esso e del suo negoziante e premendo sul bottone di creazione i dati vengono inseriti nel database e quindi mostrati nella pagina.		

Test Case:	TC-014	Nome:	Pagina “carrello” – Utilizzo carrello senza login
Riferimento:	REQ-008.1		
Descrizione:	Testare che il carrello e le sue funzioni siano utilizzabili senza fare il login		
Prerequisiti:	Pagina creata, prodotti mostrati correttamente, popup creato.		
Procedura:	<ol style="list-style-type: none"> 1. Aprire la pagina senza fare il login. 2. Inserire un prodotto nel carrello. 3. Fare il checkout. 		
Risultati attesi:	Si riesce a riempire il carrello e a quando si fa il checkout richiede di fare l'accesso.		

Test Case:	TC-015	Nome:	Pagina “cartina” – mostra delle informazioni
Riferimento:	REQ-006		
Descrizione:	Testare che venga mostrata la cartina con tutte le informazioni ei vari luoghi di ritiro		
Prerequisiti:	Pagina creata, prodotti mostrati correttamente, popup creato.		
Procedura:	<ol style="list-style-type: none"> 1. Aprire la pagina della cartina 2. Premere su un luogo di ritiro. 		
Risultati attesi:	La pagina mostra una cartina con i vari luoghi di ritiro e al click mostra le relative informazioni.		

5.2 Risultati test

5.2.1 TC-001

Questo test ha funzionato secondo i risultati attesi.

Come si può notare dalle immagini che seguono e le loro didascalie, tutti i controlli sui campi avvengono correttamente.



Figura 46 campo di testo corretto



Figura 47 campo di testo incorretto



Figura 48 campo di email corretto



Figura 49 campo di email incorretto

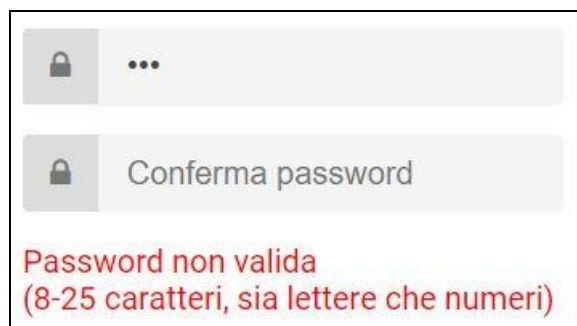


Figura 50 campo telefono corretto



The image shows a single input field for a phone number. It contains a telephone handset icon on the left and a red error icon (a plus sign with a diagonal line) on the right, indicating an invalid entry.

Figura 51 campo telefono incorretto



The image shows two password input fields. The top field has a lock icon and three dots. The bottom field is labeled 'Conferma password' and also has a lock icon. Below the fields, a red error message reads: 'Password non valida (8-25 caratteri, sia lettere che numeri)'.

Figura 52 campo password - password invalida



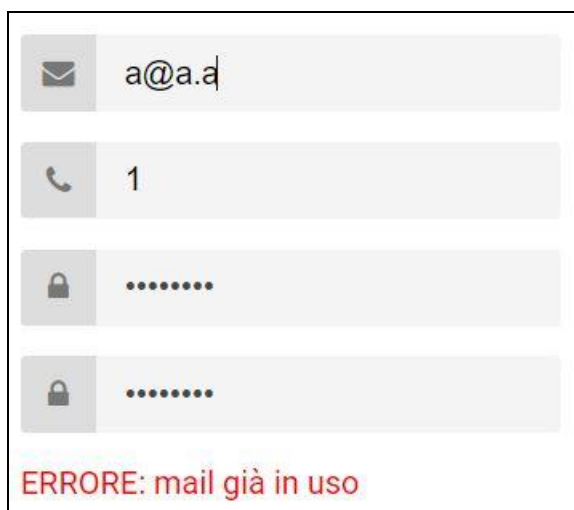
The image shows two password input fields. The top field contains eight dots. The bottom field contains three dots and a cursor. Below the fields, a red error message reads: 'ERRORE: password diverse'.

Figura 53 campo password - password diverse



The image shows two password input fields, both containing eight dots. Below the fields, the text 'TUTTI I CAMPI SONO OBBLIGATORI' is displayed. At the bottom, there is a large blue button with the text 'REGISTER' in white capital letters.

Figura 54 campo password corrette



The screenshot shows a registration form with four input fields. The first field, labeled with an envelope icon, contains the text 'a@a.a'. The second field, labeled with a phone icon, contains the number '1'. The third and fourth fields, both labeled with a lock icon, contain masked text represented by dots. Below the fields, a red error message reads 'ERRORE: mail già in uso'.

Figura 55 campo mail - mail già in uso



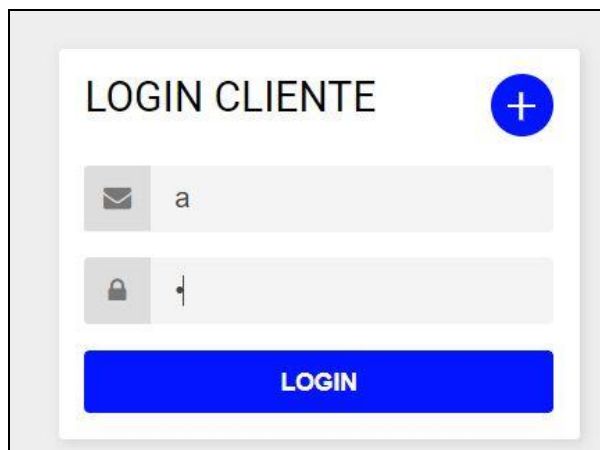
The screenshot shows a 'REGISTER' form with a blue close button in the top right corner. It contains six input fields: two for names (first and last), one for email, one for phone, and two for passwords. The email field contains 'aa@a.a'. Below the fields, a red error message reads 'ERRORE: 1 o più dati inseriti errati o mancanti'. Underneath the error message, the text 'TUTTI I CAMPI SONO OBBLIGATORI' is displayed. At the bottom, there is a blue button labeled 'REGISTER'.

Figura 56 form completo - dati incorretti

5.2.2 TC-002

Questo test ha funzionato secondo i risultati attesi.

Come mostrato dalle immagini che seguono se si prova a fare il login con dei dati non validi viene segnalato altrimenti viene fatto il login.



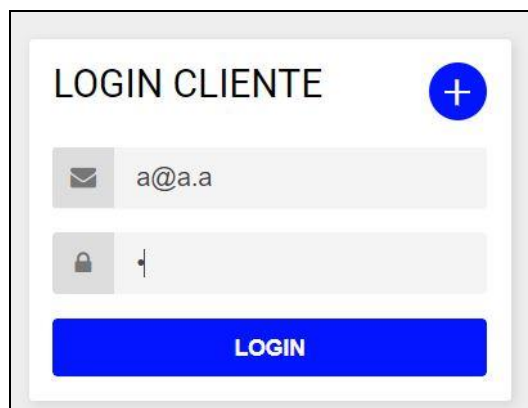
The screenshot shows a login form titled "LOGIN CLIENTE" with a blue plus icon in the top right corner. There are two input fields: the first contains the letter "a" and has an envelope icon on the left; the second contains a single character and has a lock icon on the left. Below the fields is a blue button labeled "LOGIN".

Figura 57 login utente inesistente



The screenshot shows a modal dialog box with a dark blue background. The dialog has a title bar with "LOGIN CLIENTE" and a plus icon. The main content area is white and contains the text "Utente inesistente" in the center. At the bottom of the dialog is a red button labeled "OK". A close button (X) is in the top right corner of the dialog.

Figura 58 login utente inesistente errore



The screenshot shows the same login form as in Figure 57. The first input field now contains "a@a.a" and the second input field contains a single character. The blue "LOGIN" button is at the bottom.

Figura 59 login password errata

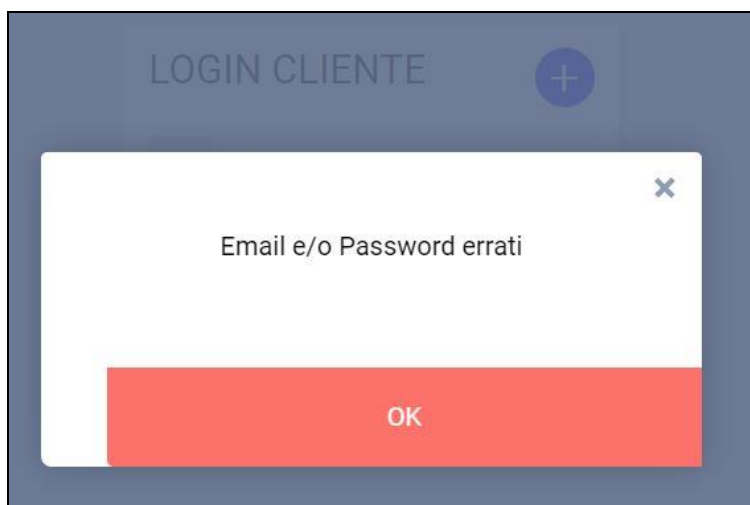


Figura 60 login password errata errore



Figura 61 login corretto



Figura 62 login corretto accesso

5.2.3 TC-003

Questo test ha funzionato secondo i risultati attesi.

Come mostrano le immagini che seguono se si accedere alla pagina dei venditori e si inseriscono i dati essi vengono visualizzati correttamente.

Il sito si prende il 10% del guadagno

Inserimento prodotto

CIBO ▼

NEGOZIETTOBELLO ▼

Lasagne surgelate 4 salti

30

10

INSERISCI

Figura 63 inserimento dati del nuovo prodotto

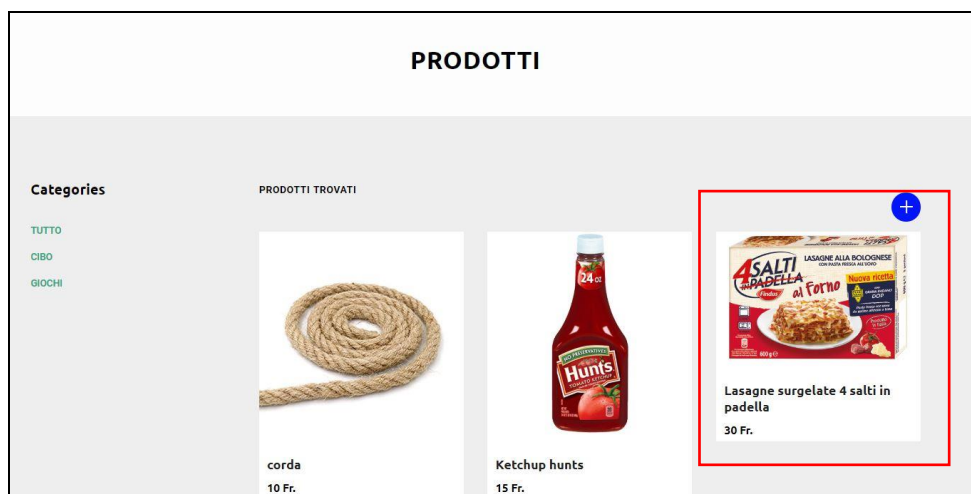


Figura 64 mostra del nuovo prodotto


nome	prezzo	quantita	nome_categoria	img	percentuale_sito
cane	2	100	NULL	application/img/blankImg.png	0
corda	10	115	giochi	application/img/corda.jpg	10
Ketchup hunts	15	104	cibo	application/img/ketchup_hunt_24oz.png	10
Lasagne surgelate 4 salti in padella	30	10	cibo	application/img/LASAGNE congelate 4 salti in padel...	10

Figura 65 mostra prodotto nel database

5.2.4 TC-004

Questo test ha funzionato secondo i risultati attesi.

Se si preme sul nome di un prodotto di può modificarne i dati ed essi vengono applicati al click sul bottone apposito.



Modifica prodotto

GIOCHI

NEGOZIETTABELLO


NOME
corda

PREZZO
10

QUANTITÀ
115

MODIFICA

Figura 66 dati prodotto



Modifica prodotto

GIOCHI

NEGOZIETTABELLO

NOME
corda corta

PREZZO
10

QUANTITÀ
115

MODIFICA

Figura 67 dati modificati

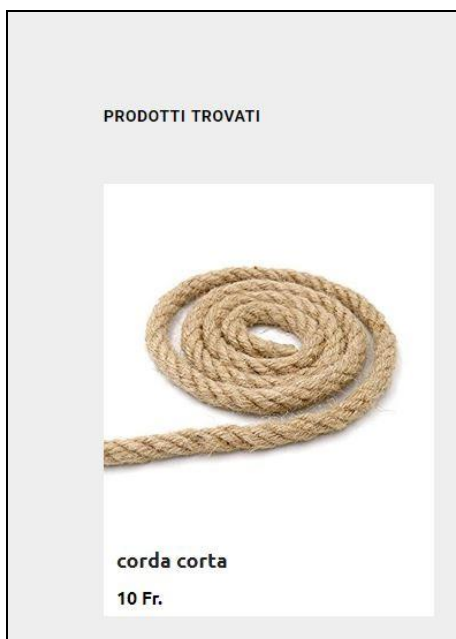


Figura 68 mostra modifica

nome	prezzo	quantita	nome_categoria	img	percentuale_sito
cane	2	100	NULL	application/img/blankimg.png	0
corda corta	10	115	giochi	application/img/corda.jpg	10

Figura 69 mostra modifica nel database

5.2.5 TC-005

Questo test ha funzionato secondo i risultati attesi.

Come mostra l'immagine che segue tutti i prodotti esistenti vengono visualizzati nella pagina "shop" del cliente.

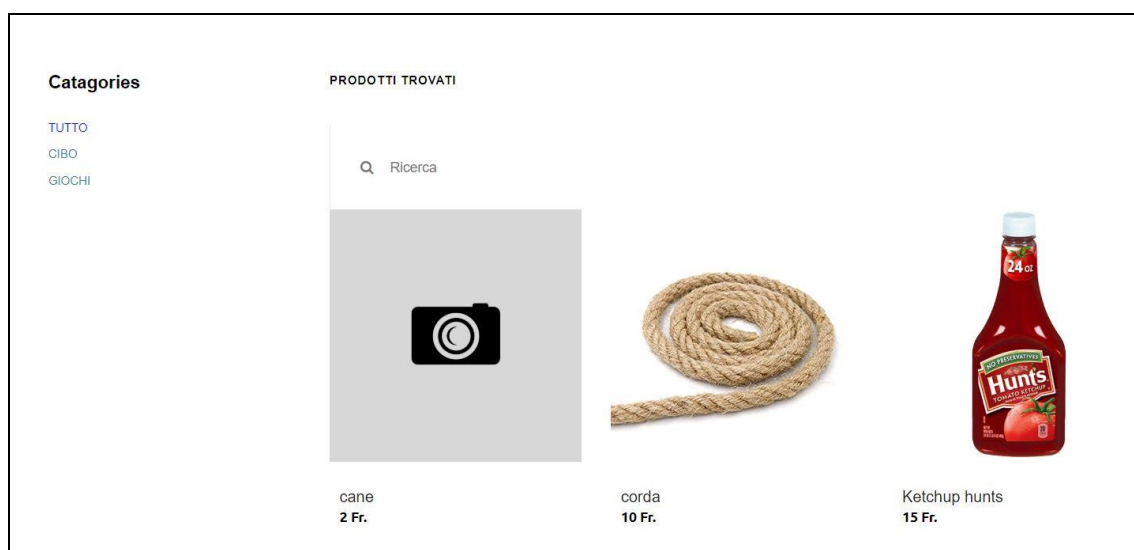


Figura 70 mostra dei prodotti

5.2.6 TC-006

Questo test ha funzionato secondo i risultati attesi.

Come mostrano le immagini che seguono, rispettivamente se si cerca un prodotto per nome o si seleziona una categoria vengono visualizzati i prodotti richiesti.

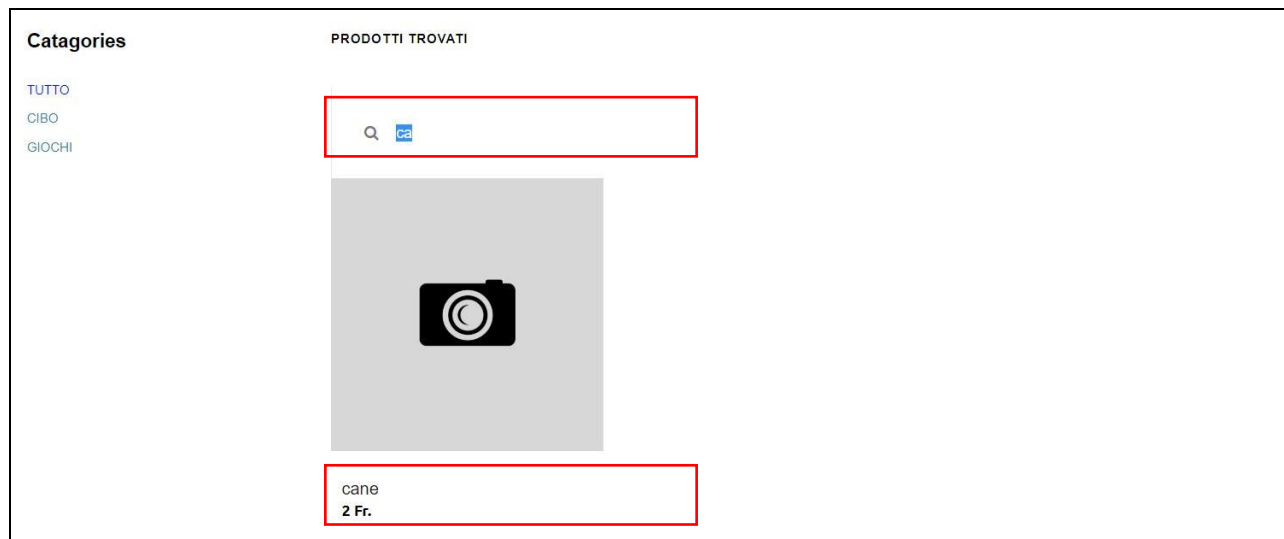


Figura 71 ricerca tramite nome



Figura 72 ricerca tramite categoria

5.2.7 TC-007

Questo test ha funzionato secondo i risultati attesi.

Se si ha fatto il login il prodotto viene inserito correttamente all'interno del carrello, come mostra l'immagine sotto mostrata.

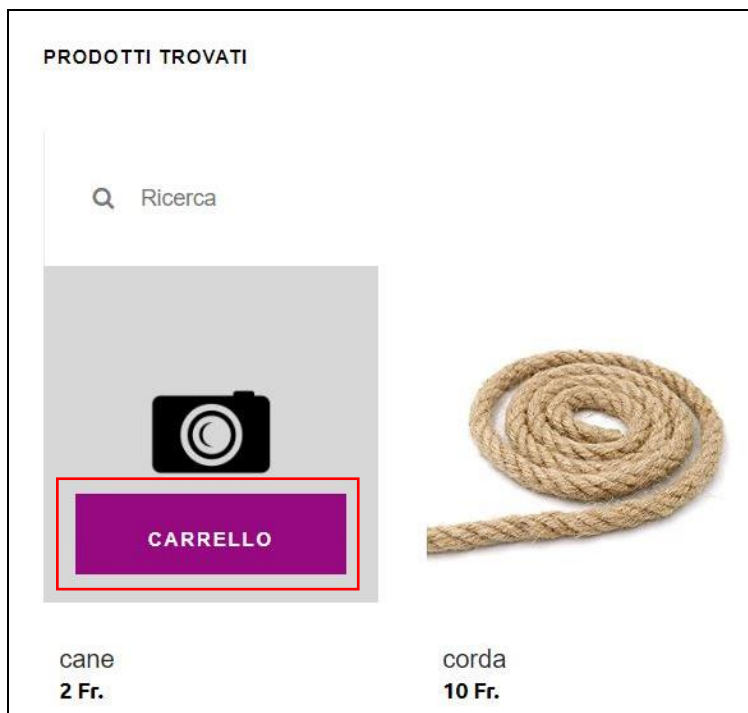


Figura 73 inserimento nel carrello

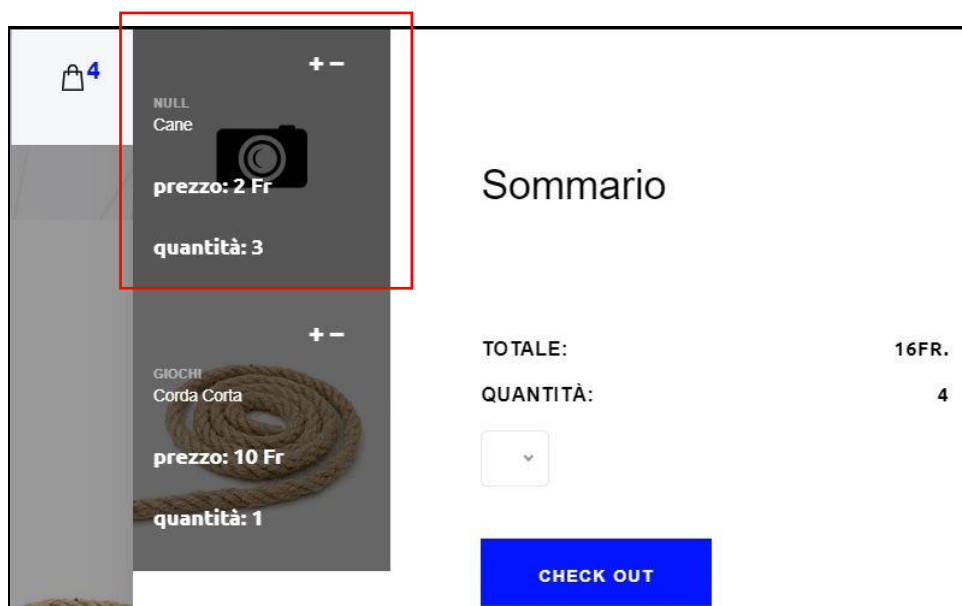


Figura 74 mostra carrello

5.2.8 TC-008

Questo test ha funzionato secondo i risultati attesi.

Come mostrano le immagini che seguono se si preme sul “+” iene aggiunto un prodotto sul “-“ viene tolto.

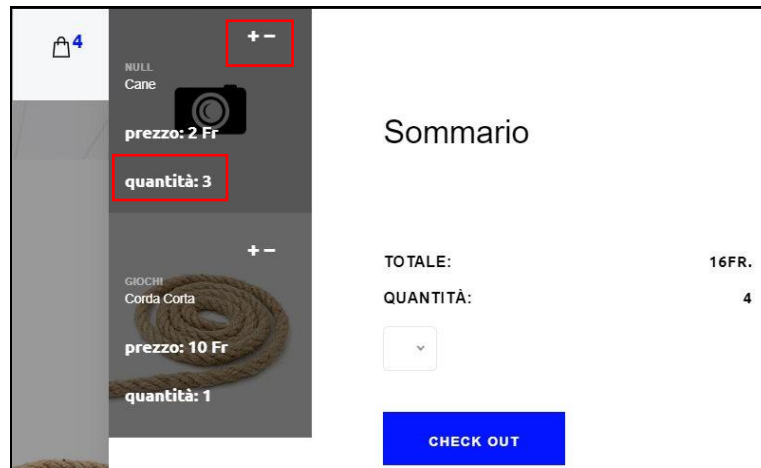


Figura 75 mostra prodotti

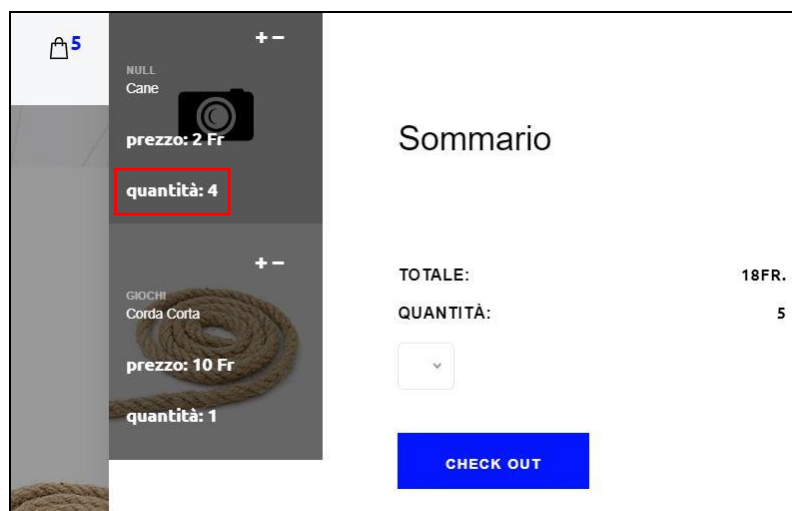


Figura 76 aumento del prodotto

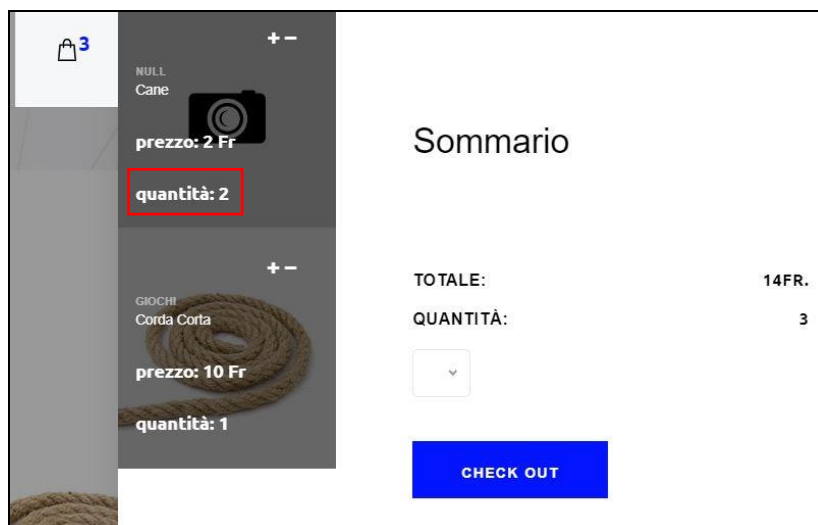


Figura 77 diminuzione del prodotto

5.2.9 TC-009

Questo test ha funzionato secondo i risultati attesi.

Se si preme sul bottone “checkout” viene aperto un file pdf con un riassunto di tutto quanto acquistato.

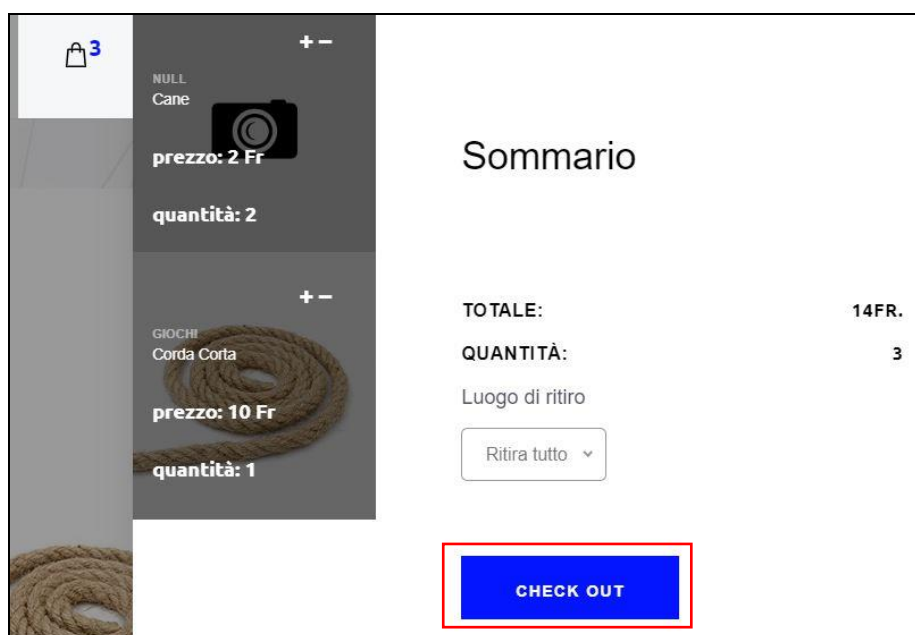


Figura 78 sommario carrello

a@a.a		2019/04/09
Il suo ordine		
Prodotto	Quantita	Prezzo
cané	2	2Fr.
corda corta	1	10Fr.
Totale	3	14Fr.
Negozio	Ritira tutto Via gelsomino 1 ritutto@gmail.com	Lugano 0912839482
<small>Progetto vendita piccoli negozi</small>		

Figura 79 pdf riassuntivo

5.2.10 TC-010

Questo test ha funzionato secondo i risultati attesi.

Come mostrano le immagini che seguono una volta inseriti i dati per la registrazione, se essi son validi e si clicca il bottone di registrazione si viene registrati correttamente.

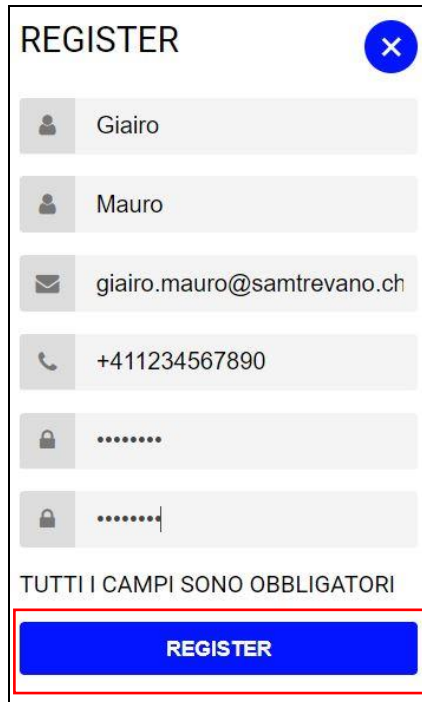


Figura 80 form di registrazione completo



Figura 81 mail conferma registrazione

email	nome	cognome	password	telefono
a@a.a	asd	asd	e54ee7e285fbb0275279143abc4c554e5314e7b417ecac83a5...	123
b@b.b	asd	asd	e54ee7e285fbb0275279143abc4c554e5314e7b417ecac83a5...	123 456 78 90
giairo.mauro@samtrevano.ch	Giairo	Mauro	1d9bbc287b32013d699d4f53351af1c670041d78eacb7ee5a9...	+411234567890

Figura 82 mostra nuovo cliente nel database

5.2.11 TC-011

Questo test ha funzionato secondo i risultati attesi.

Come mostrano le immagini che seguono quando si accede come amministratore si vede una lista di tutti i negozi e premendo sul nome di uno di essi si possono vedere i dettagli e modificarli.



Figura 83 pagina iniziale amministratore - mostra prodotti

Gestione della vendita dei prodotti dei piccoli negozianti

×

Il sito si prende il 10% del guadagno

<p>Negozio</p> <p>NOME</p> <input type="text" value="NegoziettoBello"/> <p>INDIRIZZO</p> <input type="text" value="Via ginevra 32"/> <p>CITTÀ</p> <input type="text" value="Lugano"/> <p>TELEFONO</p> <input type="text" value="+41 91 239 13 43"/>	<p>Venditore</p> <p>NOME</p> <input type="text" value="Gairo"/> <p>COGNOME</p> <input type="text" value="Mauro"/> <p>TELEFONO</p> <input type="text" value="123 456 78 90"/> <p>EMAIL</p> <input type="text" value="gairo.mauro@gmail.com"/>
---	--

TUTTI I CAMPI SONO OBBLIGATORI

MODIFICA

Figura 84 mostra dettagli prodotto e modifica

<p>INDIRIZZO</p> <input type="text" value="Via ginevra 74"/>	<p>COGNOME</p> <input type="text" value="Mauro"/>
<p>CITTÀ</p> <input type="text" value="Lugano"/>	<p>TELEFONO</p> <input type="text" value="123 456 78 90"/>
<p>TELEFONO</p> <input type="text" value="+41 91 239 13 43"/>	<p>EMAIL</p> <input type="text" value="gairo.mauro@gmail.com"/>

TUTTI I CAMPI SONO OBBLIGATORI

MODIFICA

Figura 85 modifica di un dato

nome	indirizzo	citta	telefono	archiviato	email_gestore
NegoziettoBello	Via ginevra 74	Lugano	+41 91 239 13 43 0	0	giairo.mauro@gmail.com

Figura 86 mostra modifica database

5.2.12 TC-012

Questo test ha funzionato secondo i risultati attesi.

Come mostrano le immagini che seguono se si preme sul bottone “archivia” di un prodotto dalla pagina dell'amministratore il prodotto viene archiviato e non più mostrato.



Figura 87 archiviazione negozio



Figura 88 mostra prodotto archiviato

nome	indirizzo	citta	telefono	archiviato	email_gestore
NegoziettoBello	Via ginevra 74	Lugano	+41 91 239 13 43	1	giairo.mauro@gmail.com

Figura 89 Mostra prodotto archiviato database

5.2.13 TC-013

Questo test ha funzionato secondo i risultati attesi.

Come mostrano le immagini che seguono se si preme sul “+” per aggiungere un negozio dopo aver inserito i dati e premuto il bottone di invio i dati vengono salvati e si aggiunge il negozio.

Negozio		Venditore	
NOME	INDIRIZZO	NOME	COGNOME
Moreno dove tutto costa n	Via campana 5	Salvatore	Pamino
CITTÀ	TELEFONO	TELEFONO	EMAIL
Lugano	+410914567738	+41792380139	s.pamo@gmail.com
		PASSWORD	CONFERMA PASSWORD
		*****	*****

TUTTI I CAMPI SONO OBBLIGATORI

INSERISCI

Figura 90 Inserimento nuovo negozio

NEGOZI TROVATI

Moreno dove tutto costa meno
archivia

qwe
archivia

Super negozio
archivia

Vestiti e altro
archivia

+

Figura 91 mostra nuovo negozio

nome	indirizzo	citta	telefono	archiviato	email_gestore
Moreno dove tutto costa meno	Via campana 5	Lugano	+410914567738	0	s.pamo@gmail.com

Figura 92 Nuovo negozio nel database

email	nome	cognome	password	telefono
giairo.mauro@gmail.com	Giairo	Mauro	b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5e...	123 456 78 90
gino.scarpa@gmail.com	Gino	Scarpa	b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5e...	1234567890
q@q.w	qwe	qwe	1d9bbc287b32013d699d4f53351af1c670041d78eacb7ee5a9...	123
s.pamo@gmail.com	Salvatore	Pamino	def9ca24ad7484509daf431e3d100a0026fb57b9481b924cc1...	+41792380139

Figura 93 Nuovo gestore nel database

5.2.14 TC-014

Questo test purtroppo non ha potuto essere stato eseguito perché la funzione non è stata implementata.

5.2.15 TC-015

Questo test purtroppo non ha potuto essere stato eseguito perché la funzione non è stata implementata.

5.2.16 Tabella riassuntiva

5.3 Mancanze/limitazioni conosciute

Le limitazioni conosciute sono molte, purtroppo non sono riuscito a implementare al 100% il sito web ma solo le funzionalità basi e, probabilmente, indispensabili per utilizzare il sito da tutti gli utenti. In sintesi l'unica parte completa è quella che riguarda i venditori, perché tutto quello che devono fare lo fanno.

Per quanto riguarda i clienti mancano diverse parti, a cominciare dalla parte offline. L'utente dovrebbe poter utilizzare il carrello anche senza aver eseguito l'accesso e una volta fatto se prova a fare il checkout dovrebbe venirgli chiesto di fare il login o registrarsi se non ha un account. Un'altra mancanza è la parte della cartina che mostra tutti i luoghi di ritiro possibili con le relative informazioni, informazioni che si possono vedere solo una volta scelto e fatto il checkout, il che rende scomoda la decisione di quello più vicino. In più una volta fatto il login manca la possibilità di scegliere di farsi recapitare i prodotti direttamente a casa con un costo aggiuntivo.

Per la parte dell'amministratore invece manca tutta la parte di amministrazione delle informazioni del sito web, ovvero la modifica della percentuale trattenuta sul prezzo del prodotto da parte del sito, oppure i tempi di consegna, e anche la gestione dei luoghi di ritiro.

Un ulteriore limitazione è la non efficienza del sito nel caso in cui 2 negozi abbiano lo stesso prodotto allo stesso prezzo e nella stessa quantità, perché in quel caso se uno dei due verrebbe modificato non andrebbe a modificarsi solo in uno dei due negozi ma in entrambi.

6 Consuntivo

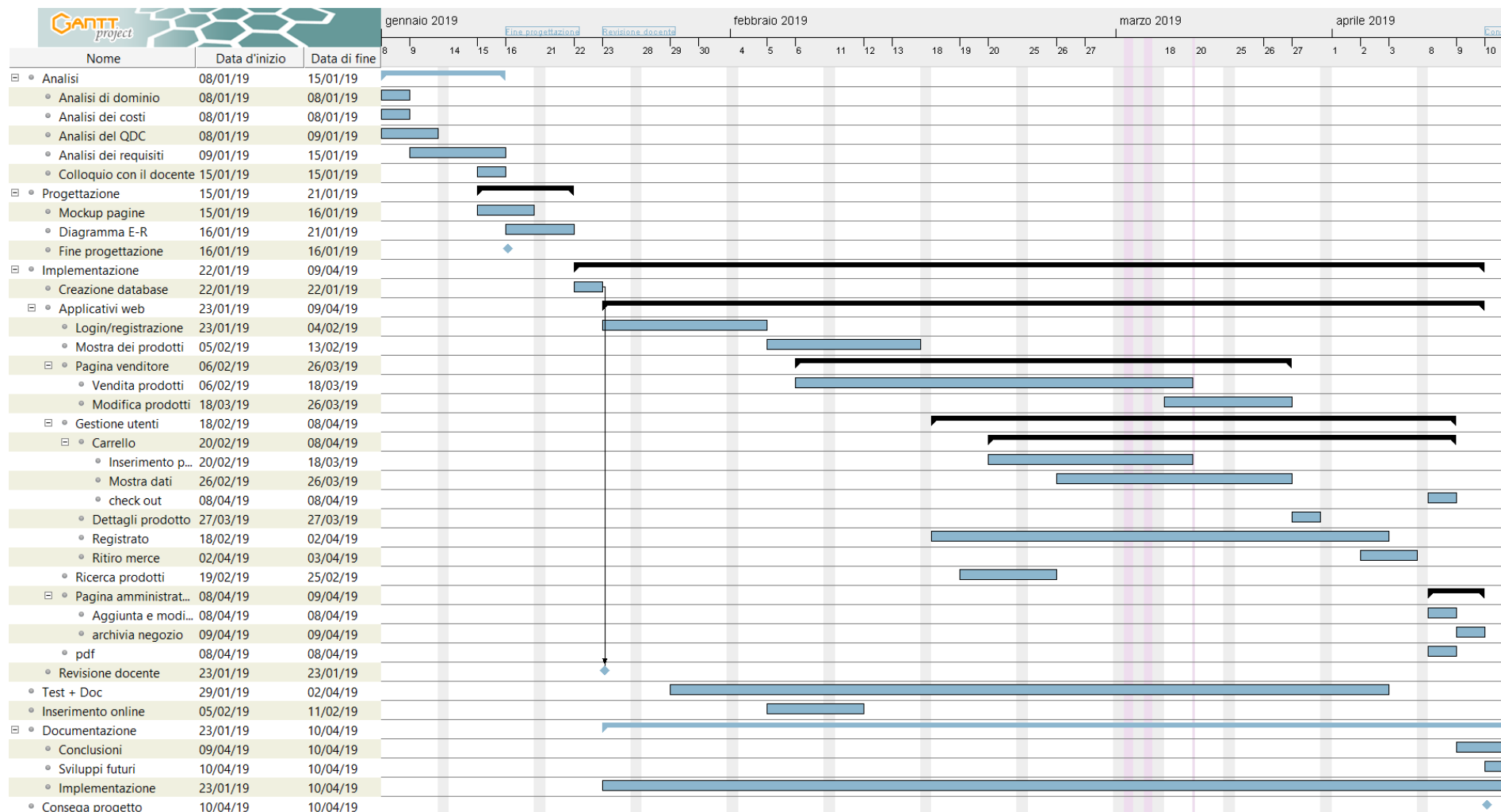


Figura 94 gantt consuntivo

Come si può notare dall'immagine nella pagina precedente il gantt ha subito diverse modifiche. Innanzitutto si può notare uno scalo a livello di tempistiche a partire dall'implementazione, questo è dovuto ai diversi problemi iniziali che ho avuto con il template e con l'adattamento della struttura MVC, ma anche ai diversi errori ricevuti durante l'implementazione di login e registrazione sia per i controlli che per l'inserimento nel database. Dato a questi e altri problemi tutta la parte di implementazione ha subito delle sostanziali modifiche.

Alla fine ho anche dovuto aggiungere nuovi campi per rendere meglio l'idea di cosa dovevo fare, perché mi sono reso conto che il gantt preventivo non era molto comprensibile a livello di compiti.

7 Conclusioni

In conclusione la soluzione finale è utilizzabile, non completa al 100% ma comunque abbastanza funzionale per ciò che è implementato. Dubito il mio prodotto abbia un impatto esagerato sul mercato globale ma in qualunque caso potrebbe risultare comodo se si dovesse fare una leggera modifica al codice, per eliminare il problema del prodotto in 2 negozi spiegato nell'ultimo punto del capitolo "mancanze e limitazioni conosciute". In sé però il sito internet è utilizzabile.

7.1 Sviluppi futuri

Oltre a poter implementare ciò che ancora non è stato inserito, quindi completare il sito, si potrebbero aggiungere diverse funzioni nella parte del login, come ad esempio il bottone "ricordami" o quello per la password dimenticata, o nella sezione di registrazione un controllo "non sono un robot" per evitare spam. Inoltre al sito ci sarebbero i requisiti opzionali che si potrebbero aggiungere, come l'effettivo pagamento, o un carrello "desiderate" se il prodotto richiesto dovesse non essere disponibile.

A proposito del carrello "desiderate" si potrebbe aggiungere anche il classico controllo delle ricerche dell'utente e registrarle in modo da potergli consigliare dei prodotti che potrebbero interessargli.

7.2 Considerazioni personali

Il progetto mi ha insegnato la difficoltà di gestire un template molto complicato e sconosciuto.

In più mi sono reso conto di quanto i controlli devono essere precisi, perché se si lavora in modo superficiale su qualcosa se si dovesse riutilizzare per un altro motivo bisogna ricontrollarlo e ricambiare tutto se le cose non erano state fatte correttamente in precedenza, soprattutto per quanto riguarda la modifica dei dati nel database, se i dati devono poi essere utilizzati per cercare degli oggetti in altre tabelle ponte.

8 Glossario

MVC: Model View Controller, Struttura usata per il sito.

PHP: PHP: Hypertext Preprocessor Linguaggio di programmazione.

JavaScript: Linguaggio di programmazione.

front-end: parte del codice in collegamento con gli utenti finali.

tab: schede diverse nella stessa pagina.

Ajax: Linguaggio di programmazione usato per usare come form JavaScript.

login: Accesso al sito.

log out: Disconnessione del sito.

database: Banca di dati contenente le informazioni del sito.

PDO: PHP Data Objects, interfaccia per utilizzare PHP con un database.

SQLInjection: attacco informatico che a ad inserire delle richieste al database tramite i campi disponibili.

Query: Stringa che segue azioni sul database.

HTML: Hyper Text Markup Language, linguaggio per scrivere pagine web.

9 Bibliografia

9.1 Sitografia

Alcuni dei siti che seguono si ripetono in più giorni quindi la data è ripetuta.

- <https://stackoverflow.com/> (sito aiuti programmazione), diverse volte durante il progetto.
- <https://github.com/giairomauro/ProgettoVenditaPiccolaNegoziante> (repository progetto), 09.01.2019, 05.02.2019, 18.02.2019, 19.02.2019
- https://github.com/giairomauro/ProgettoVenditaPiccolaNegoziante/tree/master/Informazioni_progetto (informazioni progetto in repository), 15.01.2019
- https://github.com/giairomauro/ProgettoVenditaPiccolaNegoziante/blob/master/Documentazione/Documentazione_I4AC_GestioneVenditaProdottiPiccoliNegoziante_Mauro.doc (Documentazione repository), 22.01.2019
- <https://codepen.io/Gibbu/pen/ZKYYZW> (template login), 23.01.2019
- <https://colorlib.com/wp/template/essence/> (template pagina), 23.01.2019
- <https://regexr.com/> (sito espressioni regolari), 28.01.2019
- <https://codyhouse.co/gem/simple-confirmation-popup> (template popup), 29.01.2019
- <https://stackoverflow.com/questions/2855589/replace-input-type-file-by-an-image> (Info rimpiazzo input file con immagine), 06.01.2019
- <http://saminfo.ch/gestionevendita2018/> (sito online), 19.02.2019
- https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_modal (tutorial modal w3schools), 20.03.2019
- https://github.com/giairomauro/ProgettoVenditaPiccolaNegoziante/blob/master/Informazioni_progetto/Annotazioni.docx (Annotazioni progetto), 26.03.2019

10 Didascalia

Figura 1 Gantt preventivo	10
Figura 2 Struttura MVC	12
Figura 3 Schema E-R	13
Figura 4 Pagina iniziale	15
Figura 5 Pagina mappa	16
Figura 6 Pagina carrello.....	17
Figura 7 Pagina login o utente ospite	18
Figura 8 Pagina di registrazione	19
Figura 9 Pagina login Negoziatori/Amministratori	19
Figura 10 Pagina negozi amministratore	20
Figura 11 Pagina modifica negozi	20
Figura 12 pagina aggiunta negozio	21
Figura 13 Pagina gestione prodotti.....	22
Figura 14 PopUp modifica prodotto	23
Figura 15 Pagina creazione prodotto	23
Figura 16 pagina iniziale clienti	24
Figura 17 Pagina di login	25
Figura 18 pagina di registrazione	25
Figura 19 pagina prodotti login cliente.....	26
Figura 20 carrello dell'utente	26
Figura 21 pdf riassunto acquisto	27
Figura 22 pagina iniziale gestore.....	27
Figura 23 modal di inserimento prodotto	28
Figura 24 modal modifica prodotto	28
Figura 25 pagina iniziale amministratore.....	29
Figura 26 modal inserimento negozi	29
Figura 27 modal modifica prodotto	30
Figura 28 controllo dati JavaScript	31
Figura 29 utilizzo di Ajax.....	32
Figura 30 file .htaccess.....	32
Figura 31 File index	33
Figura 32 File application.....	33
Figura 33 Funzione splitUrl.....	34
Figura 34 File config	34
Figura 35 cartella controller	35
Figura 36 inizio classi controller	35
Figura 37 apertura pagine	36
Figura 38 classe logout.....	36
Figura 39 funzione che prende dati dal database	37
Figura 40 inserimento dati nel database	37
Figura 41 classe connection, connessione al database	38
Figura 42 inizio classi model	39
Figura 43 funzione che prende i dati dalla tabella	39
Figura 44 inserimento dati nel database	40
Figura 45 modifica dati database	40
Figura 46 campo di testo corretto	45
Figura 47 campo di testo incorretto	45
Figura 48 campo di email corretto	45
Figura 49 campo di email incorretto	45
Figura 50 campo telefono corretto.....	45
Figura 51 campo telefono incorretto	46
Figura 52 campo password - password invalida	46
Figura 53 campo password - password diverse	46
Figura 54 campo password corrette	46
Figura 55 campo mail - mail già in uso	47

Figura 56 form completo - dati incorretti	47
Figura 57 login utente inesistente.....	48
Figura 58 login utente inesistente errore	48
Figura 59 login password errata	48
Figura 60 login password errata errore	49
Figura 61 login corretto	49
Figura 62 login corretto accesso	49
Figura 63 inserimento dati del nuovo prodotto	50
Figura 64 mostra del nuovo prodotto.....	50
Figura 65 mostra prodotto nel database.....	50
Figura 66 dati prodotto.....	51
Figura 67 dati modificati.....	51
Figura 68 mostra modifica	52
Figura 69 mostra modifica nel database	52
Figura 70 mostra dei prodotti.....	52
Figura 71 ricerca tramite nome.....	53
Figura 72 ricerca tramite categoria	53
Figura 73 inserimento nel carrello	54
Figura 74 mostra carrello	54
Figura 75 mostra prodotti.....	55
Figura 76 aumento del prodotto	55
Figura 77 diminuzione del prodotto	56
Figura 78 sommario carrello.....	56
Figura 79 pdf riassuntivo	57
Figura 80 form di registrazione completo	58
Figura 81 mail conferma registrazione	58
Figura 82 mostra nuovo cliente nel database	58
Figura 83 pagina iniziale amministratore - mostra prodotti	59
Figura 84 mostra dettagli prodotto e modifica	60
Figura 85 modifica di un dato	60
Figura 86 mostra modifica database	61
Figura 87 archiviazione negozio	61
Figura 88 mostra prodotto archiviato.....	61
Figura 89 Mostra prodotto archiviato database	61
Figura 90 Inserimento nuovo negozio	62
Figura 91 mostra nuovo negozio	62
Figura 92 Nuovo negozio nel database.....	62
Figura 93 Nuovo gestore nel database	63
Figura 94 gantt consuntivo	64

11 Allegati

Tutti gli allegati si possono trovare nella repository github o nella cartella all'interno del cd allegato, tra parentesi è segnata la cartella in cui si possono trovare:

- Diari di lavoro e relative immagini (Diari)
- Codici sorgente (Implementazione)
- Quaderno dei compiti (Informazioni_progetto)
- Dati progetto (Informazioni_progetto)
- Annotazioni progetto (Informazioni_progetto)
- Gantt, Mockups e schemi E-R (progettazione)