

Gotta Shop 'Em All
Applicazioni e Servizi Web

Oleg Konchenkov

oleg.konchenkov@studio.unibo.it

Elia Pasqualini

elia.pasqualini@studio.unibo.it

Giovanni Poggi

giovanni.poggi2@studio.unibo.it

28 Ottobre 2020

Indice

1	Introduzione	1
2	Requisiti	2
2.1	Requisiti Funzionali	3
2.2	Requisiti Tecnologici	4
3	Design	5
3.1	MockUp Iniziali	6
3.2	Target User Analysis	9
3.3	Story Board	11
4	Tecnologie	29
4.1	Supporto alla programmazione	29
4.2	Autenticazione e Autorizzazione	34
4.3	Comunicazione Client-Server	34
4.4	Geo-Localizzazione	34
5	Codice	36
5.1	Login e Registrazione	36
5.2	Gestione delle Immagini	38
5.3	Gestione della Geo-Localizzazione	38
5.4	GeoJSON	39
5.5	Gestione Aspetti Real-Time	40
5.6	Struttura Rotte Lato Server	42
5.7	Struttura Database	56
6	Test	58
6.1	Euristica di Nielsen	58
6.2	Usability Test	59
7	Deployment	62
7.1	Installazione	62
7.2	Messa in funzione	62
8	Conclusioni	63
8.1	Commenti Finali	63
9	Bibliografia	65

Elenco delle figure

1	Mockup Negozio	6
2	Mockup HomePage	6
3	Mockup Navbar	7
4	Mockup Prodotto	7
5	Mockup Account	8
6	Mockup Ordine	9
7	Schermata Login	11
8	Schermata Registrazione Account	12
9	Schermata Modifica Dati Founder	13
10	Schermata Menù Founder	13
11	Schermata Azione Founder	14
12	Schermata Menù Commerciante	15
13	Schermata Creazione Prodotto Commerciante	15
14	Schermata Modifica Prodotto Commerciante	16
15	Schermata Modifica Prodotto Commerciante Parte 2	17
16	Schermata Crea Annuncio Commerciante	17
17	Schermata Modifica Dati Fattorino	18
18	Schermata Visione Mappa per Consegna	18
19	Schermata Lista Ordini da Accettare	19
20	Schermata Ordine da Consegnare	20
21	Schermata Tragitto per Consegna	20
22	Schermata Menù Cliente	21
23	Schermata Home Offerte del Momento	21
24	Schermata Menù Categorie Prodotti	22
25	Schermata Categoria Abbigliamento	23
26	Schermata Lista Negozi	23
27	Schermata Ricerca per Nome	24
28	Schermata Risultato Ricerca per Nome	24
29	Schermata Descrizione Prodotto	25
30	Schermata Carrello	25
31	Schermata Conferma Ordine	26
32	Schermata Lista Ordini	26
33	Schermata Dettaglio Ordine	27
34	Schermata Notifiche	27
35	Schermata About Us	28
36	Diagramma Attività Change Order State	40
37	Diagramma Attività Product out of stock	41
38	Diagramma Attività New Advertisement	42
39	Rotta Account	43
40	Rotta Vendor	44
41	Rotta Vendor Parte 2	45
42	Rotta Vendor Parte 3	46

43	Rotta Customer	47
44	Rotta Delivery	48
45	Rotta Founder	49
46	Rotta Authentication	50
47	Rotta Order	51
48	Rotta Shopping Cart	52
49	Rotta Product	53
50	Rotta Product Parte 2	54
51	Rotta Product Parte 3	55
52	Schema MongoDB	56

1 Introduzione

Il Progetto nasce dall'idea della realizzazione di una piattaforma di E-Commerce facile, veloce ed intuitiva che dia la possibilità ai commercianti, costretti a chiudere in periodo di pandemia Covid-19, di poter proseguire con la vendita dei propri prodotti Online.

L'obiettivo del sistema messo in esecuzione è quello di permettere agli Utenti di effettuare acquisti di vario genere tramite la piattaforma e tracciare comodamente da casa il pacco in attesa del recapito tramite fattorino.

La Web Application è formata dalle seguenti tipologie di Utenti:

- **Fondatore:** Gestore dell'intero sistema, si occuperà di attivare personalmente e gestire gli Account degli Amministratori/Vendors.
- **Amministratore/Vendor:** Commercante che vuole portare la sua attività online, si occuperà di creare e gestire il proprio negozio ed inserire vari prodotti.
- **Fattorino:** Utente che si propone di lavorare per il sito come Fattorino, il suo ruolo sarà quello di trasportare la merce dal negozio al cliente.
- **Utente:** Cliente del sistema che andrà alla ricerca di vari prodotti all'interno del sito, navigando tra quelli più venduti, i migliori recensiti o quelli consigliati su misura direttamente dall'algoritmo della web application.

L'Applicazione si presenterà come un vero e proprio E-Commerce stile Amazon orientato alla catalogazione dei prodotti posti in vendita dai commercianti. Come tale disponde di:

- Home Page di presentazione veloce ed intuitiva dei principali prodotti in vendita, consigliati sull'utente o apprezzati globalmente.
- Sistema di gestione delle Informazioni Personaliali di ogni singolo Account tramite apposita pagina.
- Sistema personalizzato della gestione del sito lato Fondatore e lato Commercianti con funzioni diversificate.
- Sistema di notifiche che permetterà ai Clienti, ai Vendor ed ai Fattorini di essere aggiornati sulle vendite, sulle ordinazioni, sui pagamenti e sulle consegne effettuate.
- Sistema di Geo-localizzazione semi-ottimizzata che permetterà di comunicare al fattorino più vicino l'ordinazione del Cliente e la sua locazione geografica.

2 Requisiti

Per poter comprendere a pieno i requisiti utili alla realizzazione di un E-Commerce che possa rispecchiare gli interessi di un potenziale Utente e Commercianti, sono state eseguite un paio di interviste con Clienti abituali di shop online e Commercianti di negozi fisici che vorrebbero spostare il loro Business Online in questo periodo di incertezza dovuto alla situazione sanitaria vigente in Italia nel 2020.

Qui di seguito riporteremo un estratto dell'intervista effettuata:

- **Commercianti:** "Vorrei poter registrare una sola volta tutte le informazioni del mio negozio, inserire immagini relative ai miei prodotti e cambiare gli sconti quando necessario"
- **Commercianti:** "Vorrei potermi disinteressare della consegna e degli ordini per lasciarli evadere ad un fattorino non gestito da me"
- **Commercianti:** "Vorrei poter cambiare le informazioni sul mio negozio in maniera semplice, intuitiva e veloce"
- **Commercianti:** "Vorrei non dover competere con mille negozi rivali ma far apparire i miei prodotti migliori e più interessanti solo a chi realmente interessato, senza dover pagare degli extra"
- **Commercianti:** "Quando inserisco un annuncio sugli sconti effettuati nel mio negozio, voglio poter avvisare tutti i clienti che mi seguono"
- **Utente:** "Vorrei potermi registrare facilmente ed inserire successivamente tutti i dati completi e necessari alla fatturazione del mio ordine."
- **Utente:** "Vorrei poter recensire un prodotto e visualizzare quelli ritenuti più affidabili e qualitativamente alti da tutti gli altri utenti"
- **Utente:** "La visualizzazione di un negozio dovrebbe essere veloce e semplice, se voglio vedere i prodotti di un negozio devo farlo velocemente senza perdermi nel sito"
- **Utente:** "Vorrei essere informato sullo stato di un mio ordine effettuato"
- **Utente:** "Vorrei poter seguire un negozio ed essere informato tramite il sito di eventuali sconti"

A seguito dell'intervista, si è optato per memorizzare le richieste più importanti e significative degli intervistati arrivando a stilare i seguenti requisiti.

2.1 Requisiti Funzionali

Fondatore: Deve poter gestire tutti i profili dei Venditori e tener sotto controllo il sito.

1. Pagina personalizzata per Approvare, Bannare o Sospendere un Venditore.
2. Pagina apposita per la modifica dei propri dati.

Commerciale/Amministratore/Vendor: Si potrà registrare nella piattaforma ed aggiungere i prodotti del proprio negozio.

1. Registrandosi dovrà indicare tutte le informazioni importanti e conosciute del proprio negozio (Nome, Locazione, Immagine, etc..).
2. Potrà aggiungere potenzialmente infiniti prodotti relativi al suo negozio.
3. Avrà la possibilità di modificare in qualsiasi momento ogni singolo prodotto in tutte le sue principali informazioni.

Fattorino

1. Potrà iscriversi alla piattaforma ed indicare tutte le informazioni relative alla sua tipologia di Utente per permettergli di effettuare delle consegne.
2. Dovrà inserire facilmente e notificare l'avvenuta consegna del prodotto.
3. Dovrà notificare velocemente ed intuitivamente la presa in gestione di un ordine.

Cliente

1. Potrà iscriversi alla piattaforma fornendo tutti i dati necessari alla possibile fatturazione di un acquisto ed alla sua consegna.
2. Potrà effettuare il pagamento di un prodotto in maniera semplice e veloce.
3. Visualizzerà nella Home Page tutti i prodotti migliori della piattaforma e quelli consigliati appositamente alla sua tipologia di profilo.
4. Potrà seguire i negozi che più gli interesseranno.
5. Riceverà notifiche dai negozi di cui avrà espresso preferenza se questi pubblicheranno annunci di nuovi sconti o altro.

Sistema

1. Produrrà una notifica verso i Clienti che avranno seguito un negozio se quest'ultimo pubblicherà un annuncio.
2. Produrrà notifiche ogni qualvolta ci saranno aggiornamenti relativi agli ordini effettuati notificandole ai Clienti, ai Fattorini ed ai Negozianti.
3. Permetterà la visualizzazione tramite mappa del percorso effettuato da un Fattorino durante la consegna in tempo reale.
4. Permetterà di Navigare la piattaforma sia tramite i prodotti sia tramite i negozianti.

2.2 Requisiti Tecnologici

Il requisito tecnologico di maggior importanza su cui il gruppo si è maggiormente concentrato è la **Reattività**. Quest'aspetto è di fondamentale importanza nella gestione delle notifiche relative agli annunci dei Negozianti, relative agli Ordini in tempo reale ed alla posizione real-time del Fattorino.

Inoltre un altro requisito fondamentale è l'**Estendibilità e Scalabilità** del sistema in quanto abbiamo utilizzato un'architettura a Microservizi con Docker. Quindi ci si aspetterà che la piattaforma risponda bene sotto questi aspetti.

3 Design

Il modello tramite il quale il gruppo ha deciso di realizzare il progetto è stato il modello iterativo basato su **UCD (User Centered Design)** con utenti "Virtualizzati". Ovvero, sono stati individuati i target e su di essi sono state create una serie di **Personas** che sono state utilizzate per capire come le funzionalità dell'applicazione dovessero rispondere alle caratteristiche degli Utenti.

Questo modello classico UCD è stato poi integrato con il modello **AGILE**.

In particolare sono stati svolti dei briefing settimanali per una corretta ed equa distribuzione del lavoro sulla base dello stato di avanzamento del progetto. In questo modo i membri del team hanno potuto partecipare attivamente allo sviluppo delle varie funzionalità. Nonostante non siano state eseguite delle release concrete e periodiche verso il cliente, durante lo sviluppo del progetto ci si è attenuti il più possibile al principio **MVP (Minimum Viable Product)**. Ovvero, si realizzava ogni settimana una versione minimale del sistema alla quale si andava sempre più integrando funzionalità aggiuntive in modo da possedere ad ogni step una piattaforma funzionante e testabile.

Dal punto di vista del design delle interfacce si è cercato di allinearsi il più possibile ai principi **KISS** ovvero "**Less is More**". Il gruppo ha optato per questi principi con l'obiettivo di arrivare a realizzare un'interfaccia funzionale e semplice all'utilizzo da parte anche di Utenti finali non propriamente esperti.

Infine, per rendere migliore la **User Experience** si è optato per rendere le interfacce utilizzabili da qualsiasi dispositivo applicando i principi alla base del **Responsive Design**.

3.1 MockUp Iniziali

Logo Categorie ▾ Negozi About Us Il mio profilo ▾  

Foot Locker



Email footlocker@gmail.com

Numero di telefono 3315467903

Foot Locker Retail, Inc. o Foot Locker, Inc. è un'azienda statunitense specializzata nella vendita di abbigliamento sportivo e di calzature

Dove trovarci:



Figura 1: Mockup Negozio

Logo Categorie ▾ Negozi About Us Il mio profilo ▾  

Gotta Shop 'Em All.

Cerca... 

Offerte del Momento

OFFERTA	OFFERTA	OFFERTA	OFFERTA
			
Iphone 11 ★★★★★ 900 € 1000 €			

Figura 2: Mockup HomePage

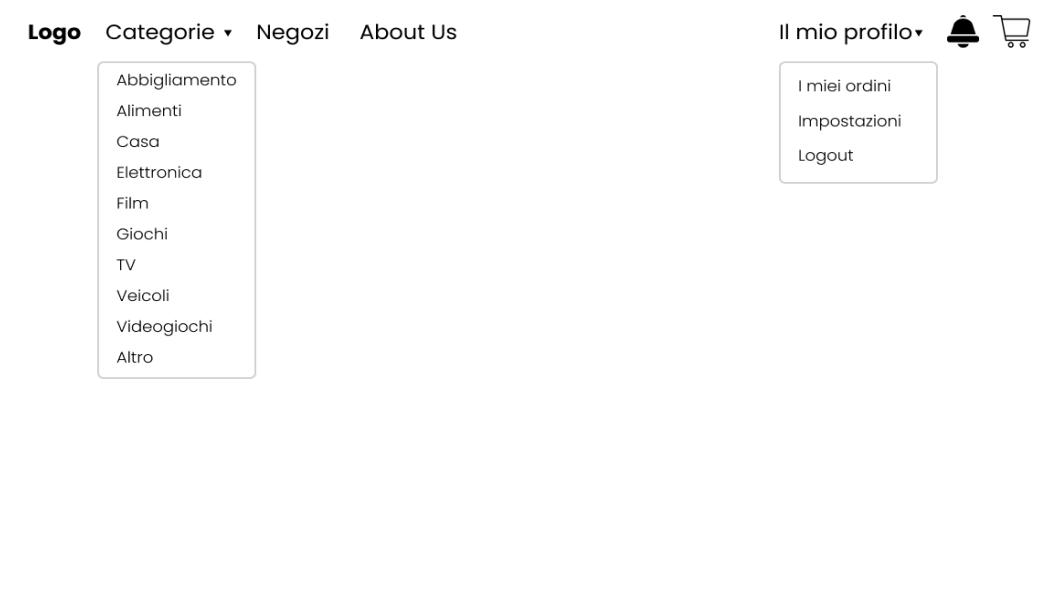


Figura 3: Mockup Navbar

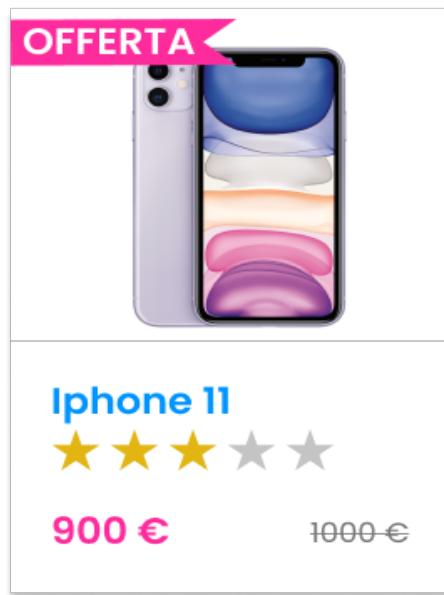


Figura 4: Mockup Prodotto

Benvenuto Cliente !

Il tuo Account:

Nome: Elia
Cognome: Pasqualini
Email: elia.pasqualini.97@gmail.com
Username: pasqualini10

Modifica Account

Cancella Account

Modifica Account :

Nome:

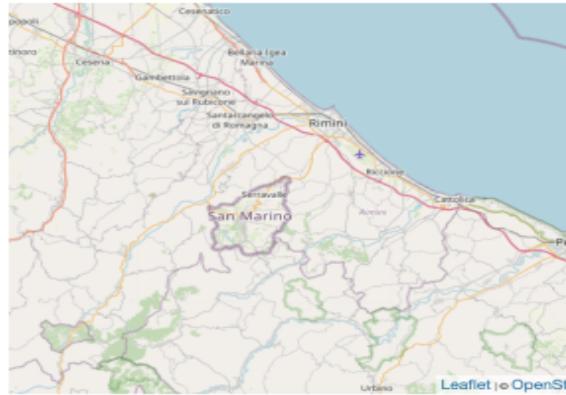
Cognome:

Email:

Figura 5: Mockup Account



Ordini pronti per la spedizione



Ordine del 2020-10-23

Via Romolo Murri 10, San Marino

[Prendi in carico](#)

Ordine del 2020-10-08

Via Napoleone Bonaparte, Rimini

[Prendi in carico](#)

Figura 6: Mockup Ordine

3.2 Target User Analysis

Sono stati individuati i seguenti Target:

- Fondatore
- Amministratore/Venditore
- Fattorino
- Cliente

Personas: Andrea

Andrea è il fondatore del sito che si occuperà della gestione degli Account dei Commercianti.

- Scenario d'uso

1. Andrea accede al sito con le sue credenziali pre-esistenti tramite form di Login
2. Andrea modifica i suoi dati personali tramite apposta sezione Settings.
3. Andrea gestisce e modifica lo Status dei Commercianti iscritti nel sito.

Personas: Elena

Elena è una Commerciante che si è vista chiudere il suo negozio causa Epidemia e vuole vendere i suoi prodotti Online.

- Scenario d'uso: Elena deve trovare il modo di farsi conoscere Online.

1. Elena accede alla piattaforma
2. Elena si iscrive con i suoi Dati personali
3. Elena accede alla sezione Settings del suo account e completa i dati relativi al suo Negozio
4. Elena accede alla sezione Actions del suo account e può gestire i prodotti del suo negozio, aggiungendoli, modificandoli e togliendoli dalla lista.

Personas: Giulio

Giulio è un cittadino che decide di volersi trovare un lavoro come Fattorino.

- Scenario d'uso: Giulio viene a conoscenza delle piattaforma e decide di iscriversi.

1. Giulio accede alla piattaforma
2. Giulio si iscrive con i suoi Dati personali
3. Giulio accede alla sezione Settings del suo account e completa i dati relativi alla posizione di Fattorino
4. Giulio riceverà notifiche direttamente sul sito quando ci sarà un Ordine vicino alla sua zona da evadere.
5. Giulio accetterà l'ordine e lo notificherà
6. Giulio consegnerà la merce al Cliente e lo notificherà tramite la piattaforma.

Personas: Cecilia

Cecilia è una ragazza che vuole effettuare un acquisto online.

- Scenario d'uso: Cecilia viene a conoscenza delle piattaforma e decide di iscriversi.

1. Cecilia accede alla piattaforma
2. Cecilia si iscrive con i suoi Dati personali

3. Cecilia accede alla sezione Settings del suo account e completa i dati relativi alla tipologia del suo Account
4. Cecilia esplora il sito in cerca di prodotti da acquistare
5. Cecilia seleziona un prodotto da acquistare e decide di comprarlo.
6. Cecilia può visionare lo stato del suo ordine.
7. Cecilia verrà notificata dal sistema quando l'ordine sarà in consegna
8. Cecilia potrà visionare in tempo reale la posizione del Fattorino sulla mappa.

3.3 Story Board

Login

Inserisci il tuo Username
Username Necessario

Inserisci la tua Password
Password Necessaria

RegistratiAccedi

Figura 7: Schermata Login

Gotta Shop 'Em All

User Delivery Admin

Nome
Il Nome è Necessario

Cognome
Il Cognome è Necessario

La tua Miglior Email
L'Email è richiesta

Scegli il tuo Username
Username Necessario

Scegli una Password
Password Richiesta

[Registrati](#) [Indietro](#)

Figura 8: Schermata Registrazione Account

Come mostrato nelle sovrastanti figure, l'utente che desidera iscriversi o semplicemente accedere al sito, può raggiungere la schermata di Login direttamente dalla Home. Se, invece, desidera Registrarsi alla piattaforma, può accedere alla schermata di Registrazione direttamente da quella di Login. Scegliere semplicemente tramite un Click del Mouse a quale tipologia di Account vuole far riferimento (Commercante, Fattorino e Cliente), inserire tutti i dati richiesti ed inviare la richiesta. Se i dati inseriti sono validi, l'Utente verrà reindirizzato alla schermata Home (Nel caso sia un Cliente) oppure nella sua apposita pagina personalizzata.

Fondatore

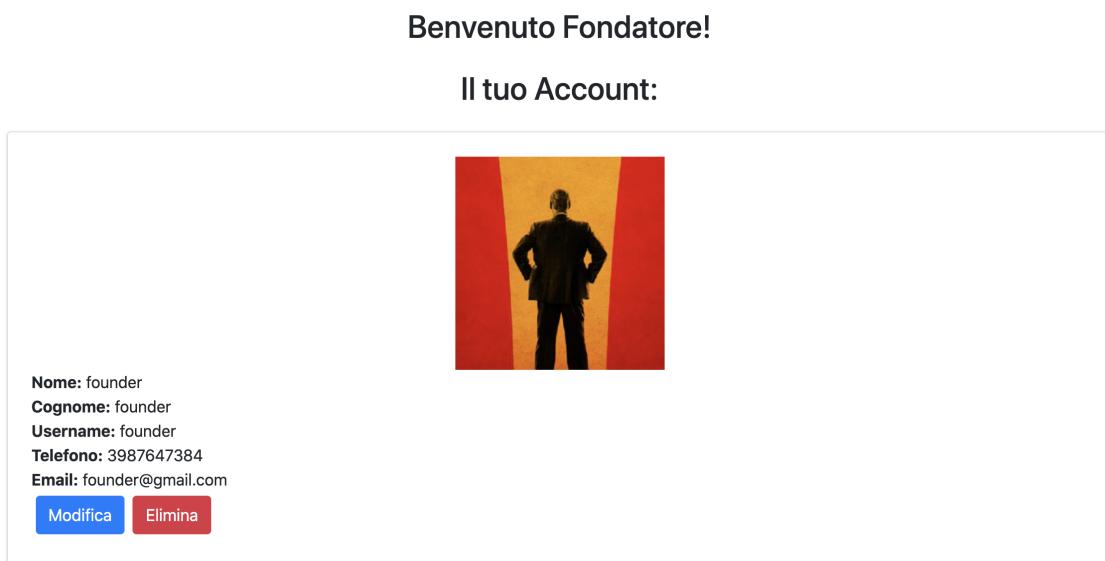


Figura 9: Schermata Modifica Dati Founder

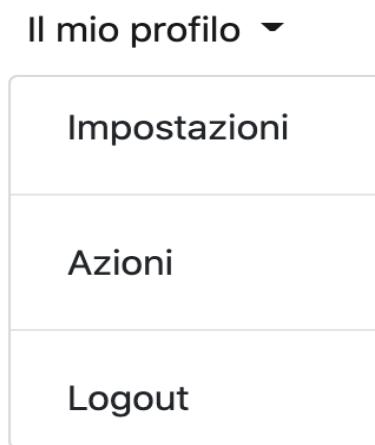


Figura 10: Schermata Menù Founder

Gestisci Commercianti:

Account Commercianti:

Immagine Negozio:



ID:
5f74cb9659ba4c0015dec634
Username: admin
Email: Jonh@gmail.com
Stato Registrazione: Pending

[Modifica](#) [Elimina](#)

Sotto questa Lista trovi il Form per Modificare il Commercante!

Modifica Commercante:

Stato Registrazione: Pending ▾

[Modifica Commercante](#)

Figura 11: Schermata Azione Founder

Come suggerito dalle immagini, la tipologia di Account denominata Fondatore potrà accedere tramite il menù a tendina nella sua pagina di gestione degli Account di tipo Commercante. In questa pagina potrà impostare lo stato degli Account dei Commercianti ed accettarli, bloccarli o sosperderli.

Commercante



Figura 12: Schermata Menù Commercianti

Gestisci Prodotti:

Crea Prodotto:

Immagine: Choose file No file chosen

Nome: Inserisci Nome Prodotto

Tipologia: Abbigliamento ▾

Prezzo: Inserisci Prezzo Prodotto

Sconto: Inserisci Sconto Prodotto

Descrizione: Inserisci una Descrizione del Prodotto

Quantità: Inserisci Quantità Prodotto

Figura 13: Schermata Creazione Prodotto Commercianti



Nome: Nike

Tipologia: Abbigliamento

Prezzo: 80 €

Sconto: 20%

Descrizione: Veramente comode

Quantità: 33 pz.

[Modifica Prodotto](#)

[Elimina Prodotto](#)

Figura 14: Schermata Modifica Prodotto Commercianti

Modifica Prodotto:

Immagine: Choose file No file chosen

Nome: Inserisci Nome Prodotto

Tipologia: Abbigliamento ▾

Prezzo: 80

Sconto: 20

Descrizione: Veramente comode

Quantità: 33

Modifica Prodotto



Figura 15: Schermata Modifica Prodotto Commercianti Parte 2

Gestisci Annuncio:

Crea Annuncio:

Titolo: Inserisci Titolo Annuncio

Descrizione: Inserisci Descrizione Annuncio

Crea Annuncio

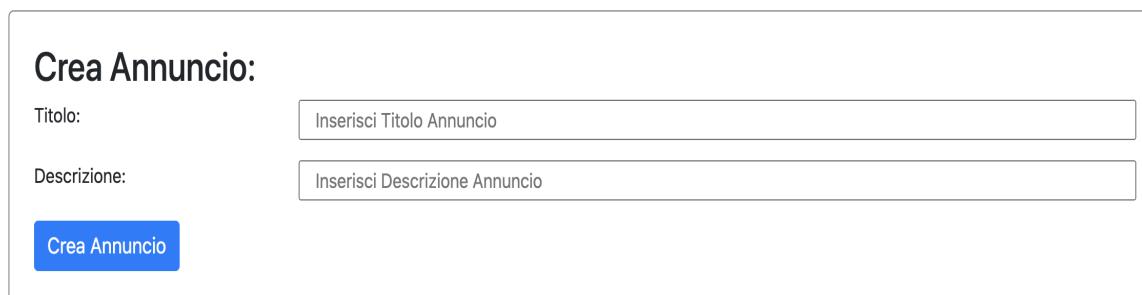


Figura 16: Schermata Crea Annuncio Commercianti

In questa sequenza di immagini vediamo come l'Account Commercianti venga rein-dirizzato nella sua pagina di Modifiche relative al suo Negozio. Successivamente potrà accedere dal menù a tendina a due pagine personalizzate per la gestione dei prodotti del Negozio e Advertisement. In queste due pagine può accedere a tutte le funzionalità relative ad un prodotto (Creazione, Modifica, Cancellazione) ed alle funzioni di creazione di un Annuncio Pubblicitario da visualizzare nella Home Page.

Fattorino

Benvenuto Fattorino, al lavoro!

Il tuo Account

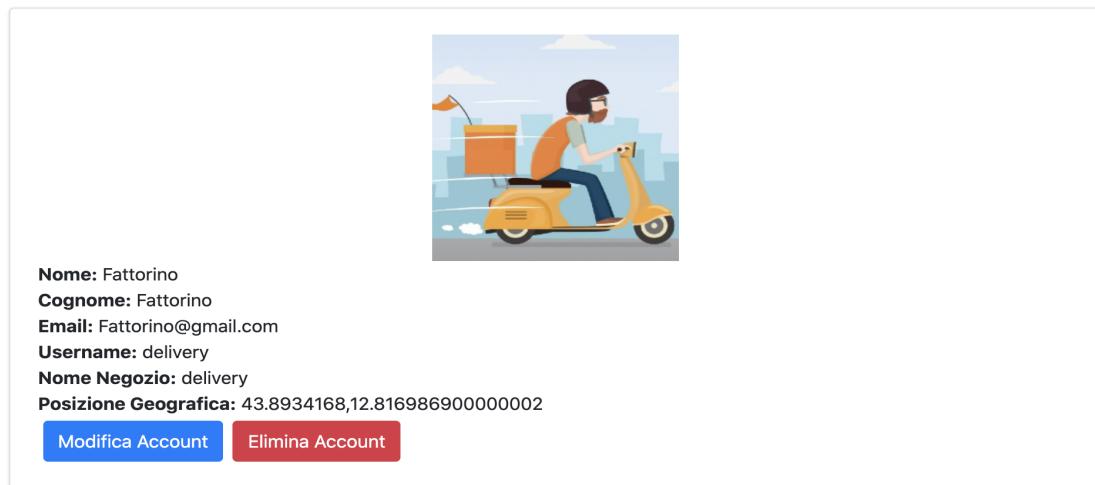


Figura 17: Schermata Modifica Dati Fattorino

Ordini pronti per la spedizione

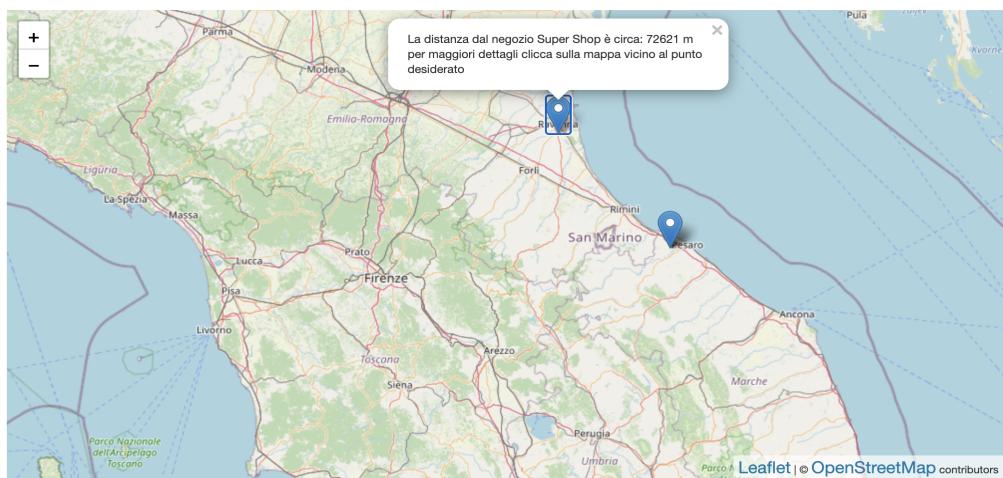


Figura 18: Schermata Visione Mappa per Consegna

Ordine del 2020-11-05

Num. ordine:

5fa46611efa32b001cba07e9

Stato:

Created

Nome negozio:

Super Shop

Indirizzo consegna:

Paese: Italia

Città: Cesena

Via: Sacchi 43

Informazioni: Primo piano

Distanza:

72621m

[Dettagli](#)

Figura 19: Schermata Lista Ordini da Accettare

Ordine del 2020-11-05T20:52:33.712Z

Num. ordine:
5fa46611efa32b001cba07e9

Stato:
Created

Nome negozio:
Super Shop

Prodotto:
Nike

Quantità:
3

Indirizzo consegna:
Country: Italia
City: Cesena
Street: Sacchi 43
Info: Primo piano

Distanza:
72621m

Figura 20: Schermata Ordine da Consegnare

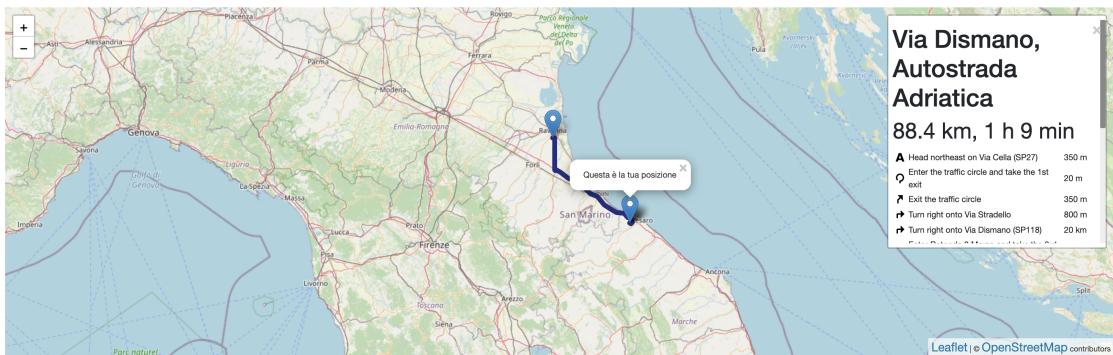


Figura 21: Schermata Tragitto per Consegna

Come mostrato nelle immagini sovrastanti, il Fattorino verrà reindirizzato alla sua pagina di modifiche dei dati personali. A quel punto potrà accedere alla sua pagina personalizzata in cui visualizzerà gli ordini inevasi della sua zona, potrà scegliere quale prendere in carico, visualizzare nella Mappa il tragitto da percorrere e notificare l'avvenuta consegna una volta sopraggiunto dal Cliente con il prodotto ordinato.

Cliente

Il mio profilo ▾



I miei ordini

Impostazioni

Logout

Figura 22: Schermata Menù Cliente

Offerte del Momento

OFFERTA	OFFERTA	OFFERTA	OFFERTA
Nike ★★★★★ 78.3 € 87€	Nike 2 ★★★★★ 78.3 € 87€	Nike 3 ★★★★★ 78.3 € 87€	Nike 4 ★★★★★ 78.3 € 87€

Prev Next

Figura 23: Schermata Home Offerte del Momento



Categorie ▾ Negozi About Us

- Abbigliamento
- Alimenti
- Casa
- Elettronica
- Film
- Giochi
- TV
- Veicoli
- Videogiochi
- Altro

Got
"Un'

Figura 24: Schermata Menù Categorie Prodotti

Abbigliamento

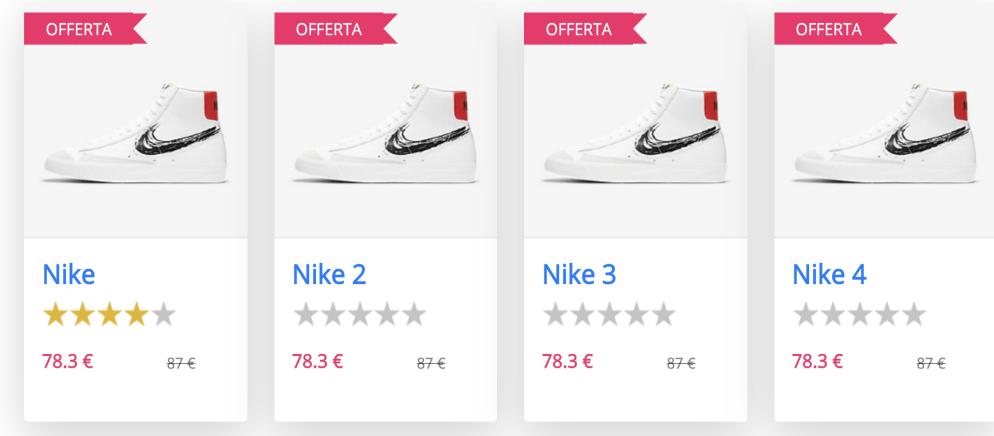


Figura 25: Schermata CATEGORIA Abbigliamento

Negozi



Figura 26: Schermata Lista Negozi

Come mostrato in questa sequenza di immagini, l’Utente loggato potrà effettuare numerose tipologie diverse di ricerche di prodotti all’interno della piattaforma. Effettuare ricerche per CATEGORIA, per VENDITORE, per PRODOTTI PIÙ VENDUTI, per meglio RECENSITI e CONSIGLIATI. L’utente non loggato potrà eseguire le stesse ricerche senza però usufruire della sezione PRODOTTI CONSIGLIATI, essendo basata sulle preferenze espresse dall’Utente nel corso della permanenza nella piattaforma.

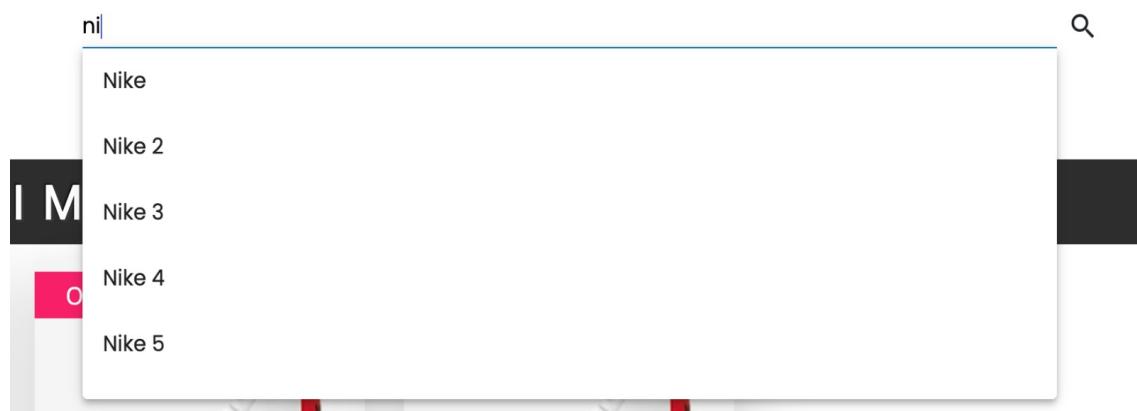


Figura 27: Schermata Ricerca per Nome

Risultati per: Nike

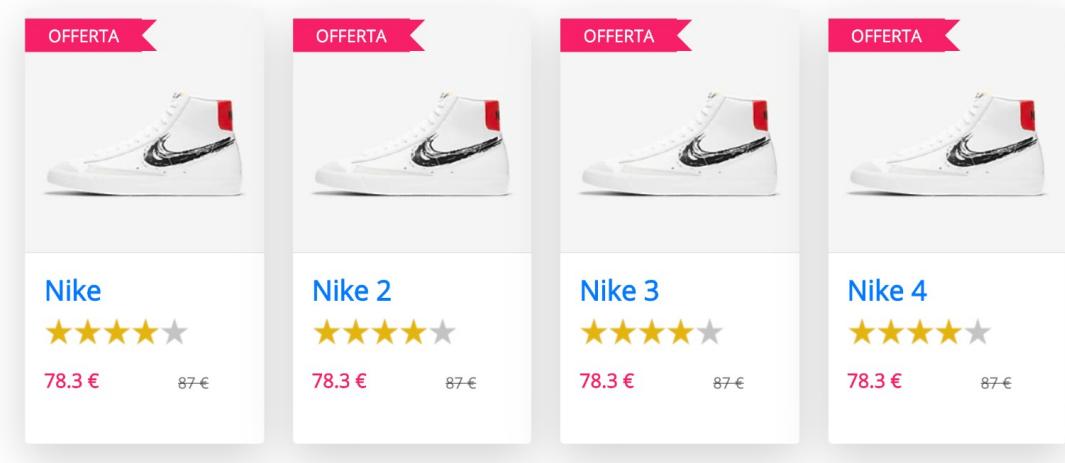


Figura 28: Schermata Risultato Ricerca per Nome



Figura 29: Schermata Descrizione Prodotto

Il tuo carrello

	Prezzo
	EUR 87
Nike	
Quantità	2
Rimuovi	Visualizza articoli simili
Totale provvisorio:	156.6
	Procedi

Figura 30: Schermata Carrello

Nike

EUR 87

Compila queste informazioni per procedere 0 %

Totale pagato

Tipo di pagamento _____

Numero di carta _____

Paese _____

Città _____

Via _____

Ulteriori informazioni _____

Annulla

Paga e concludi

Procedi

Figura 31: Schermata Conferma Ordine

I miei ordini			
Ordine del 2020-11-05			
Stato: Consegnato	Prezzo: EUR 234.89999999999998	Num. prodotti: 1	Dettagli

Figura 32: Schermata Lista Ordini

Dettaglio dell'ordine

Ordine del 05-10-2020	
Indirizzo di spedizione: Sacchi 43 - Italia - Cesena Primo piano	Tipo di pagamento: carta di credito 7845857485493404
Riepilogo ordine: Totale articoli 234.8999999999998 EUR	Stato: Consegnato

Prodotti dell'ordine	
	Nike Quantità : 3 Scrivi una recensione per questo prodotto.
	Prezzo 87 € 10% SCONTO

Figura 33: Schermata Dettaglio Ordine

I miei messaggi

Spedizione	
Cambio stato dell'ordine L'ordine con codice: 5fa46611efa32b001cba07e9 è stato Spedito	<input type="button" value="Segna come letto"/>

Spedizione	
Cambio stato dell'ordine L'ordine con codice: 5fa46611efa32b001cba07e9 è stato Consegnato	<input type="button" value="Segna come letto"/>

Figura 34: Schermata Notifiche

Gotta Shop 'Em All

"Un' esperienza di shopping senza paragoni."



DESIDERIA

Seleziona ciò che desideri e aggiungilo al carrello.



PAGA

Completa l'ordine effettuando il pagamento online.



ATTENDI

Aspetta il nostro fattorino comodamente sul divano.

Figura 35: Schermata About Us

Cliccando sopra uno dei prodotti, si verrà reindirizzati alla scheda tecnica di quel prodotto il più dettagliata possibile. Dentro questa pagina l'Utente potrà effettuare un ordine di quel prodotto, Recensirlo o visionare la scheda del Commerciante che vende l'oggetto ed, eventualmente, visionare tutti i prodotti da egli caricati sulla piattaforma e messi in vendita.

4 Tecnologie

Il Gruppo di Lavoro ha deciso di realizzare l'intero sistema utilizzando il Solution Stack **MEAN**. Esso è stato scelto sia perché da un lato risulta adeguato alla gestione delle funzionalità definite in precedenza e sia perché è lo stack sul quale il team ha avuto modo di fare maggiore esperienza.

4.1 Supporto alla programmazione

Architettura a Microservizi

I microservizi sono un approccio architettonico alla realizzazione di applicazioni. Quello che distingue l'architettura basata su microservizi dagli approcci monolitici tradizionali è la suddivisione dell'applicazione nelle sue funzionalità di base. Ciascuna funzione, denominata servizio, può essere compilata e implementata in modo indipendente. Pertanto, i singoli servizi possono funzionare, o meno, senza compromettere gli altri.

Quindi, un microservizio è una funzione di base di un'applicazione che viene eseguita indipendentemente dagli altri servizi. Tuttavia, l'architettura basata su microservizi, non comporta solo il basso accoppiamento tra le funzioni di base di un'applicazione ma propone una ristrutturazione dei team di sviluppo e del framework comunicativo tra servizi. Questo approccio offre la possibilità di gestire criticità inevitabili, supporta la scalabilità dinamica e agevola l'integrazione di nuove caratteristiche.

Per eseguire il deployment dei microservizi e sfruttare i vantaggi di questo approccio, occorre adattare gli elementi di base di un'architettura SOA (Service-Oriented Architecture) e suddividere un'applicazione nelle relative funzioni di base.

Inoltre, questo metodo permette di compilare, testare e modificare più servizi contemporaneamente, svincolando il team dai cicli di sviluppo monolitici.

I microservizi possono comunicare tra di loro, in genere in modalità stateless, permettendo di realizzare applicazioni con una maggiore tolleranza agli errori e meno dipendenti dalla resto dell'applicazione. Inoltre, comunicano tramite interfacce di programmazione delle applicazioni (API) indipendenti dal linguaggio e ciò ha consentito al team di sviluppo di scegliere i propri strumenti.

Basandosi su un'architettura distribuita, i microservizi consentono di ottenere sviluppo e routine più efficienti. La possibilità di sviluppare più microservizi in contemporanea ha consentito al team di lavorare simultaneamente alla stessa applicazione, riducendo le tempistiche di sviluppo.

Ciascun microservizio, se costruito correttamente, è indipendente e non influisce sugli altri servizi nell'infrastruttura. Di conseguenza, l'eventuale errore di un componente, non determina il blocco dell'intera piattaforma, come avviene con il modello monolitico. Poiché le applicazioni basate su microservizi sono più piccole e modulari delle tradizionali applicazioni monolitiche, tutti i problemi associati a tali deployment vengono automaticamente eliminati. Benché questo approccio richieda un coordinamento superiore, i vantaggi che ne sono derivati sono determinanti.

Poiché le applicazioni più grandi vengono suddivise in parti più piccole, per il team è risultato molto più semplice comprendere, aggiornare e migliorare tali componenti, per-

mettendo di accelerare i cicli di sviluppo, soprattutto in combinazione con le metodologie di sviluppo Agile.

Grazie alle API indipendenti dal linguaggio, il gruppo è stato libero di scegliere il linguaggio e la tecnologia ottimale per la funzione da creare.

Oltre a queste, un'architettura basata su microservizi presenta due criticità principali: la complessità e la produttività.

- **Compilazione:** occorre dedicare tempo all'identificazione delle dipendenze fra i servizi. Inoltre, a causa di tali dipendenze, quando si esegue una compilazione può essere necessario eseguirne anche molte altre. È fondamentale considerare anche gli effetti prodotti dai microservizi sui dati.
- **Test:** i test di integrazione, così come i test end-to-end, possono diventare molto difficili ma più importanti che mai. Occorre ricordarsi che un errore in una parte dell'architettura potrebbe causare un errore in un componente a vari passi di distanza, a seconda del modo in cui i servizi sono strutturati per comunicare a vicenda.
- **Gestione delle versioni:** l'aggiornamento a una nuova versione potrebbe compromettere la retrocompatibilità. Il problema può essere gestito attraverso la logica condizionale ma in breve tempo può diventare difficile da gestire.
- **Deployment:** la configurazione iniziale è una fase critica. Per semplificare il deployment, occorre innanzitutto investire notevolmente in soluzioni di automazione. La complessità dei microservizi renderebbe il deployment manuale estremamente difficile.
- **Registrazione:** nei sistemi distribuiti, per ricollegare tutti i vari componenti, sono necessari registri centralizzati. Senza di essi, gestirli in modo scalabile risulterebbe impossibile.
- **Monitoraggio:** è stato fondamentale per il team avere una visibilità centralizzata del sistema, al fine di identificare l'origine dei problemi.
- **Debugging:** il debugging remoto non è una scelta praticabile con decine o centinaia di servizi. Al momento non esiste un'unica soluzione legata al debugging.
- **Connattività:** occorre valutare quale metodo di rilevamento dei servizi scegliere, se centralizzato o integrato.

Docker

La tecnologia Docker utilizza il kernel di Linux e le sue funzionalità, come Cgroups e namespace, per isolare i processi in modo da poterli eseguire in maniera indipendente. Questa indipendenza è l'obiettivo dei container. Ovvero la capacità di eseguire più processi e applicazioni in modo separato per sfruttare al meglio l'infrastruttura esistente pur conservando il livello di sicurezza che sarebbe garantito dalla presenza di sistemi

separati.

Gli strumenti per la creazione di container, come Docker, consentono il deployment a partire da un'immagine. Ciò semplifica la condivisione di un'applicazione o di un insieme di servizi, con tutte le loro dipendenze, nei vari ambienti. Docker automatizza anche la distribuzione dell'applicazione (o dei processi che compongono un'applicazione) all'interno dell'ambiente containerizzato.

Gli strumenti sviluppati partendo dai container Linux, responsabili dell'unicità e della semplicità di utilizzo di Docker, offrono agli utenti accesso alle applicazioni, la capacità di eseguire un deployment rapido, e il controllo sulla distribuzione di nuove versioni.

L'approccio Docker alla containerizzazione si basa sulla capacità di estrarre i singoli componenti di un'applicazione da aggiornare o riparare. Oltre a questo approccio basato sui microservizi, è possibile condividere i processi tra più applicazioni in modo molto simile a quello usato dalla Service-Oriented Architecture (SOA).

I container basati su Docker possono ridurre il deployment a pochi secondi. Creando un container per ogni processo, puoi condividere con rapidità i processi simili con le nuove applicazioni. Poiché non è necessario riavviare un sistema operativo per aggiungere o spostare un container, i tempi per il deployment sono sostanzialmente più brevi. Inoltre, grazie alla velocità del deployment, attraverso i container è possibile creare ed eliminare dati in modo sicuro, semplice ed economico. Dunque, la tecnologia Docker è caratterizzata da un approccio basato sui microservizi granulare e controllabile, per un ambiente più efficiente.

RabbitMQ

RabbitMQ si basa su un principio di Advanced Message Queuing Protocols (AMQP). Il grande vantaggio dell'AMQP: mittente e destinatario non devono comprendere lo stesso linguaggio di programmazione. Il principio di base è che tra il produttore e il consumatore di un messaggio c'è una coda. E' qui che vengono depositati i messaggi. In questo contesto, il termine "messaggio" viene utilizzato in modo molto ampio. Per messaggi si intendono disposizioni ad altri programmi, ma anche effettivi messaggi di testo. Qualsiasi forma di trasmissione di informazioni può svolgersi tramite RabbitMQ o altri broker di messaggi.

Il vantaggio di RabbitMQ è che il produttore del messaggio non deve occuparsi anche della sua trasmissione. Il broker di messaggistica consegna il messaggio permettendo al produttore di cominciare un nuovo incarico. Il mittente, dunque, non deve attendere la ricezione del messaggio da parte del destinatario. In processi di questo tipo il messaggio si trova in coda e può essere prelevato dal consumatore. Quando ciò avverrà, il mittente sarà già impegnato con un nuovo incarico. Si tratta dunque di un processo asincrono: mittente e destinatario non devono agire allo stesso ritmo.

La trasmissione di un messaggio richiede quattro elementi:

- Producer: crea messaggi
- Exchange: inoltra i messaggi
- Queue: deposita i messaggi

- Consumer: rielabora il messaggio

Il producer pubblica un messaggio ma non lo invia direttamente al consumer, perché lo trasmette all'exchange. Questo elemento è responsabile di inoltrare i messaggi a diverse code. Di queste si servono poi i consumer con i messaggi. Sia l'elemento exchange che la queue sono parti di RabbitMQ e vengono amministrati dal software. Per garantire che i messaggi raggiungano i destinatari corretti, si utilizzano le routing key. Il mittente dà al messaggio una routing key, che funziona come un indirizzo. Con l'ausilio di una chiave, l'exchange è in grado di riconoscere come indirizzare il messaggio.

Tra l'exchange e la queue si trova il cosiddetto binding. Attraverso di esso ogni singola coda viene collegata all'exchange. Inoltre il binding definisce i criteri con cui un messaggio deve essere inoltrato.

I Consumer, e cioè i software destinatari, si registrano in code precise per potersi riferire ai messaggi. Perciò è previsto un Consumer per coda. Se più consumer prendono messaggi dalla stessa coda, non è possibile garantire la loro corretta distribuzione. Si decide in maniera facoltativa per ogni messaggio, se il destinatario deve riconoscerne il contenuto o se questo non è necessario.

Nginx

Nginx (si pronuncia Engine X), è un server web open source ad alte prestazioni disponibile per vari sistemi Unix e Unix-like e Windows. Nginx è stato concepito con lo scopo esplicito di essere più performante del web server Apache. La sua architettura ad eventi, infatti, esibisce prestazioni consistenti in presenza di carichi elevati, con maggiore efficienza rispetto al modello multi-threaded adottato da Apache.

È bene precisare, comunque, che nginx non è soltanto un server web. Esso infatti può agire da reverse proxy, da proxy TCP/UDP, da e-mail proxy e da load balancer. Queste caratteristiche ne hanno favorito una larga diffusione, impiegandolo soprattutto come reverse proxy o load balancer.

Il server nginx è composto principalmente da due tipi di processi: Master e Worker. I processi Worker (tipicamente uno per ogni core o CPU presenti nel sistema) gestiscono le connessioni con i client e le relative richieste. Sono processi che non richiedono l'accesso a elementi privilegiati del sistema. Il processo master è responsabile del coordinamento dei worker. Esso esegue alcuni compiti privilegiati quali il binding delle porte, il caricamento dei file di configurazione, ecc.

I processi worker fanno uso di una sofisticata architettura event-driven (basata su kqueue in ambienti BSD e sull'omologa implementazione su Linux, epoll) per gestire in modo concorrente le richieste provenienti dai client in modo non bloccante. Questo approccio, in radicale contrapposizione con quello multi-threaded, favorisce un migliore sfruttamento delle risorse di calcolo a parità di numero di client connessi. Avere un unico processo non bloccante che gestisce più client permette infatti di evitare frequenti cambi di contesto (o context switch) inevitabili in un ambiente fortemente multiprogrammato.

Nginx dispone di un'architettura modulare, e ciò consente di estendere le funzionalità offerte dal server mediante lo sviluppo di moduli custom. Esistono moduli per il supporto alla compressione (gzip, Brotli ed altri), per l'elaborazione delle immagini servite,

per lo streaming audio/video, ecc.

Angular

Angular è un framework dedito alla realizzazione di applicazioni desktop, web e mobile. È stato realizzato utilizzando Javascript. Angular permetterà la realizzazione di applicazioni Client-Side utilizzando html, scss e typescript.

I componenti (**Components**) sono l'elemento principale di un'applicazione Angular. Un componente contiene la definizione della vista e dei dati che determinano l'aspetto ed il comportamento di quella vista. I componenti sono semplici classi Javascript e sono definiti tramite il decoratore **@Components**.

Il decoratore fornisce al componente la vista per visualizzare i metadati sulla classe.

Il Componente passa i dati alla view utilizzando un processo chiamato **Data Binding**. Tale processo viene messo in atto legando gli elementi del DOM alle proprietà del componente. Il bind può essere utilizzato per visualizzare i valori delle proprietà delle classi dei componenti per l'utente, per modificare gli stili degli elementi, per rispondere ad un evento, ecc.

Le direttive (**Directives**) ci aiutano a manipolare il DOM. Puoi cambiare l'aspetto, il comportamento o il layout di un elemento del DOM utilizzando le direttive. Esse aiutano ad estendere l'HTML. Le direttive sono classificate in tre categorie in base al loro comportamento. Abbiamo quindi, Component, direttive di attributo e direttive strutturali. **NgFor** è una direttiva strutturale, tale direttiva ripete una parte del modello html tante volte quanti sono gli elementi di un elenco (**Collections**).

NgSwitch ci permette di aggiungere/rimuovere elementi dal DOM. Può essere paragonabile all'istruzione switch in javascript.

NgIf ci permette di aggiungere o rimuovere elementi dal DOM.

La direttiva **NgClass** è una direttiva di attributo, tramite ngClass è possibile aggiungere o rimuovere classi css dagli elementi html.

La direttiva **ngStyle** permette di modificare lo stile di un elemento html utilizzando semplici espressioni, con ngStyle si può modificare dinamicamente lo stile di un elemento html.

Le **Pipe** sono utilizzate per trasformare i dati. Per esempio, la pipe dedicata alle date viene utilizzata per formattare le date. Angular inoltre consente di creare pipe personalizzate.

I **Componenti** condividono dati tra di loro. Il componente padre può comunicare con il componente figlio tramite l'annotazione **@Input**. I componenti figli rilevano le modifiche alle proprietà di input utilizzando l'hook del ciclo di vita di **OnChanges** o tramite un setter di proprietà. Il componente figlio può comunicare con il padre generando un evento che il padre può ascoltare.

I **Life Cycle Hook**, sono metodi che Angular invoca su direttive e componenti durante la loro creazione, modifica e distruzione. Usando i Life Cycle Hook possiamo mettere a punto il comportamento dei componenti durante la creazione, l'aggiornamento e la loro distruzione.

I **Servizi** consentono di creare codice riutilizzabile in altri componenti. I servizi possono essere iniettati all'interno di componenti ed altri servizi. Le dipendenze sono dichiarate

nel modulo utilizzando i metadati dei provider.

Il modulo **HttpClient** ci permette di interrogare un'API remota ed ottenere dei dati da utilizzare all'interno dell'applicazione. Tale modulo richiede di sottoscrivere la response restituita utilizzando gli observable RxJs.

Il modulo **Router** gestisce la navigazione nelle applicazioni Angular. Il routing permette di spostarsi da una parte all'altra dell'applicazione o da una vista ad un'altra vista.

Angular utilizza diversi modi per definire lo stile di un'applicazione. Si può personalizzare l'applicazione a livello globale e quindi sovrascrivere gli stili globali per ogni singolo componente molto facilmente. Gli stili dei componenti si riflettono localmente al livello di singolo componente.

Stile

Per realizzare lo stile grafico delle pagine della piattaforma, il gruppo ha optato per l'utilizzo di **Bootstrap** che utilizza **FlexBox** per la realizzazione del Grid System e permettere la realizzazione di pagine con un layout fluido.

In alcune parti di codice di è optato per l'utilizzo del supporto nativo offerto da FlexBox che ha consentito di ottenere un comportamento più predicable e stabile degli elementi della pagina quando essa deve adattarsi a display di vari device con dimensioni differenti.

Persistenza

Per quanto riguarda la persistenza delle infomazioni lato server si è optato per l'utilizzo della libreria di **Mongoose** per interfacciarsi al Database di **MongoDB**.

Il gruppo ha optato per questa libreria poiché risulta essere una delle soluzioni più utilizzare nel suo campo e l'unica presentata a lezione.

4.2 Autenticazione e Autorizzazione

Si è optato per l'utilizzo del modulo **BCrypt** per allinearsi alle prassi di sicurezza previste per la memorizzazione delle password. Ciò ci ha consentito di salvare le password sul Database in maniera sicura e sotto forma di Hash.

Per l'elaborazione delle richieste HTTP provenienti dal client, il server utilizzerà il **JWT (Json Web Token)** per garantire contemporaneamente Autenticità e Autorizzazione. Il Server genera il JWT quando riceve una richiesta di registrazione o login e lo restituisce al Client. Quet'ultimo si occuperà di presentarlo al Server allegandolo ad ogni richiesta futura.

4.3 Comunicazione Client-Server

Quando si è effettuata l'implementazione delle funzionalità real-time come l'invio delle notifiche e la ricezione di queste, si è optato per l'utilizzo della libreria **Socket.io**.

4.4 Geo-Localizzazione

Per quanto riguarda le sezioni della piattaforma in cui si richiedevano coordinate geografiche, il gruppo ha optato per la libreria **Leaflet**. Questa libreria consente di visualizzare

la mappa geografica e fornire supporto di base per la visualizzazione su di essa di layers quali markers, boundaries, etc..

Per quanto riguarda la Geo-Localizzazione, il gruppo ha optato per l'utilizzo del plugin open-source **Esri-Leflet** che fornisce il componente **Esri-Leflet-Geocoder** che è stato usato per fare da intermediario verso funzionalità messe a disposizione dal servizio **Ar-
cGIS**, che si occupa realmente di implementare funzionalità di geo-localizzazione.

Per la gestione della mappa tramite l'Account Fattorino, si è optato per consentire l'utilizzo del **GPS** per identificare il tragitto più breve per raggiungere il Cliente ed identificare la posizione dell'Utente.

5 Codice

5.1 Login e Registrazione

La Web Application possiede alcune funzionalità sbloccabili solamente se si è utenti loggati. In particolare, le funzionalità sono:

- Modifica Dati Personalni
- Accesso Carrello
- Aggiunta prodotti al Carrello
- Gestione prodotti nel Carrello
- Ricezione e Lettura Notifiche
- Aggiunta/Rimozione negozi preferiti
- Visualizzazione prodotti consigliati
- Aggiunta recensioni prodotti

Queste esigenze hanno prodotto le seguenti necessità:

1. **Lato Client:** Deve essere possibile scoprire se l'utente è loggato o meno ed a quale tipologia di Account appartiene (Fondatore, Amministratore/Vendor, Fattorino, Cliente).
2. **Lato Server:** Deve esistere un meccanismo di identificazione ed autorizzazione che permetta di riconoscere il mittente di una richiesta e valutare il suo diritto o meno di accedere a tale risorsa.

Il gruppo aveva optato inizialmente per l'utilizzo dei **Cookie di sessione** ma questa soluzione non consentiva di soddisfare tutte le esigenze sorte durante la realizzazione del progetto, come il poter accedere lato client alle informazioni contenute nel cookie.

Per ovviare a questa problematica, si è pensato di introdurre un cookie aggiuntivo creato direttamente dal client dopo l'autenticazione, che consentisse di immagazzinare informazioni essenziali sull'utente ed accessibili dal frontend.

La soluzione appena citata è efficace ma non tiene in considerazione la probabilità di un eventuale attaccante di creare cookie di sessione per impersonare un possibile utente del sistema. Per risolvere il problema sarebbe necessario:

- Una connessione sicura tramite protocollo HTTPS
- Un token di autenticazione

Il gruppo, riflettendo su questa problematica, ha deciso di ricorrere ad un **Json Web Token** creato lato server al momento del login o della registrazione Utente per poi restituirlo al client. Inoltre, onde evitare attacchi di Cross Site Scripting (XSS) il Json Web Token verrà restituito al client all'interno di un cookie HTTP-Only che andrà a sostituire il cookie di sessione sopracitato.

Questo cookie verrà inviato automaticamente nelle successive richieste inviate dal Client ed il Server potrà recuperare il Json Web Token contenuto e procedere nella verifica della firma.

```
// JWT generation
const token = jwt.sign(payload, process.env.SECRET_KEY, { expiresIn: 30 * 86400 })
// expiresIn expressed in seconds
const cookieConfig = {
    httpOnly: true,
    maxAge: 30 * 86400 * 1000, // 30 days cookie
    signed: true
}
res.cookie('jwt', token, cookieConfig)
res.status(201).send({ _id: user._id })
```

Il sito non utilizza protocollo HTTPS, di conseguenza rimarrà vulnerabile ad attacchi di "Man in the middle" o di "Eaves Dropping".

In caso il gruppo volesse allinearsi alle comuni prassi di sicurezza relative alla memorizzazione delle password, si è deciso di criptarle. Durante la fase di registrazione, quest'ultima viene criptata lato client prima di inviarla dentro una richiesta HTTP. Successivamente criptata per la seconda volta lato server prima di essere memorizzata dentro il database.

```
if(bcrypt.compareSync(req.body.old_password, customer.password)){
    bcrypt.hash(req.body.new_password, 10, (err, hash) => {
        customerData.password = hash
        const query = {$set: customerData}

        Customer.findOneAndUpdate({ _id: client_id }, query)
            .then(cust => {
                res.status(200).send({ customer: cust })
            }).catch(err => {
                res.status(500).json({ error: err })
            }).catch(err => { res.status(500).json({ error: "error performing search of user" })})
    })
} else {
    res.json({ error: "wrong old password"})
}
```

Per la fase di Login, la password viene criptata lato client prima dell'invio. Successivamente ri-criptata lato server e confrontata con quella presente all'interno del database. Le operazioni relative alla cifratura sono eseguire utilizzando il modulo **bcrypt** e **SHA512** come algoritmo di hashing.

Quando un Utente decide di effettuare il logout, il client elimina il cookie precedentemente creato ed esegue un'apposita richiesta al server per eliminare definitivamente il cookie HTTP-Only contenente il Json Web Token.

```
exports.logoutUser = function(req, res) {
    if (req.signedCookies.jwt != null) {
        const token = req.signedCookies.jwt;
        try {
            jwt.verify(token, process.env.SECRET_KEY);
            res.clearCookie("jwt");
            res.json({ description: "Logout succeeded" })
        } catch (error) {
            res.sendStatus(401).json({ error: "unauthorized user, invalid token"});
            // The JWT is not valid - verify method failed
        }
    }
}
```

```

    }
} else {
    res.sendStatus(401).json({error: "unauthorized user, no token"});
    // No JWT specified
}

```

5.2 Gestione delle Immagini

Per la memorizzazione delle immagini dei Negozi, dei Prodotti e dei Profili Utente, si è deciso di salvare le immagini sottomesse dai Client all'interno del Database sotto il formato base64.

Il Client codifica le immagini in base64 e le trasferisce al server all'interno di un oggetto Json con le altre informazioni dell'utente.

Alla ricezione, il server procede alla memorizzazione della stringa che rappresenta l'immagine in base64.

Inoltre, per migliorare la resa visiva delle immagini del profilo, il team ha deciso di uniformare l'aspect-ratio delle immagini memorizzate nel server. Tutto questo è reso possibile effettuando un ritaglio dall'anteprima dell'immagine caricata dal frontend nel momento della selezione da parte dell'utente dal suo computer.

Ovviamente questo ritaglio non verrà effettuato in automatico ma l'Utente o l'Account interessato avrà pieno controllo su di esso con la possibilità di selezionare una porzione di immagine da ritagliare. Per riuscire ad eseguire questi passi appena citati, si è optato per l'utilizzo della libreria **Cropper.js**.

5.3 Gestione della Geo-Localizzazione

Per la realizzazione della geo-localizzazione si è optato per l'utilizzo del plugin open-source **Leaflet**. Le più importanti funzionalità supportate da leaflet che sono state utilizzate sono:

- **Geocoding:** utilizzato per recuperare le coordinate spaziali
- **Reverse-geocoding:** utilizzato per rendere possibile l'utilizzo del GPS per individuare la posizione dell'utente
- **Leaflet root controller**

Tutte le volte che all'interno dell'applicazione si richiede l'utilizzo della mappa, si è fatto utilizzo dello stesso componente "App-Map". Esso contiene la mappa Leaflet ed ogni volta che il componente viene usato viene inizializzato sempre allo stesso modo.

E' possibile personalizzare la configurazione di base della mappa secondo le esigenze specifiche di cui si necessita. Queste personalizzazioni vengono innescate in base al tipo di Utente Loggato alla piattaforma.

Con il seguente codice dichiariamo una mappa Leaflet:

```

var map = L.map('map').fitWorld();

L.tileLayer('https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}'
    ?access_token={accessToken}',
{
    attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">OpenStreetMap</a>'
}

```

```

    contributors , <a href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>,
    Imagery <a href="https://www.mapbox.com/">Mapbox</a>',
    maxZoom: 18,
    tileSize: 512,
    zoomOffset: -1
}).addTo(map);

```

Leaflet ha una scorciatoia molto utile per ingrandire la visualizzazione della mappa sulla posizione voluta. Ovvero il metodo di individuazione con l'opzione setView che sostituisce il solito metodo setView nel codice:

```
map.locate({setView: [44.133333, 12.233333], maxZoom: 8});
```

Qui specifichiamo 8 come zoom massimo quando si imposta automaticamente la visualizzazione della mappa. Non appena l'utente accede al componente, la mappa imposterà la visualizzazione sulla posizione indicata.

La mappa risulta responsive e funziona bene anche per dispositivi mobile.

5.4 GeoJSON

Per permettere la memorizzazione sul database delle coordinate spaziali relative alla posizione dei Negozi, dei Fattorini e dei Clienti si è deciso di utilizzare il formato GeoJSON. Questa operazione ci ha portato ad una serie di vantaggi notevoli perché sia Leaflet sia MongoDB offrono supporto diretto a questo tipo di formato.

Leaflet fornisce supporto per la creazione di layers, anche complessi, senza il bisogno di particolari elaborazioni.

Su MongoDB esiste la possibilità di utilizzare query geospaziali per restituire risultati sulla base delle coordinate memorizzate.

```

const location = { // GeoJSON
  type: 'Feature',
  geometry: {
    type: 'Point',
    coordinates: [44.127286, 12.251577]
  },
  properties: {
    shop_name: req.body.shop_name,
    profile_picture: req.body.shop_picture
  }
}

```

Quando un Utente Cliente o Fattorino o Commerciante specifica la propria posizione nella piattaforma, il client invia direttamente le coordinate geospaziali al server. Sarà quindi il server ad utilizzare tali coordinate per creare l'oggetto GeoJSON da salvare nel Database prima di memorizzare il nuovo utente.

5.5 Gestione Aspetti Real-Time

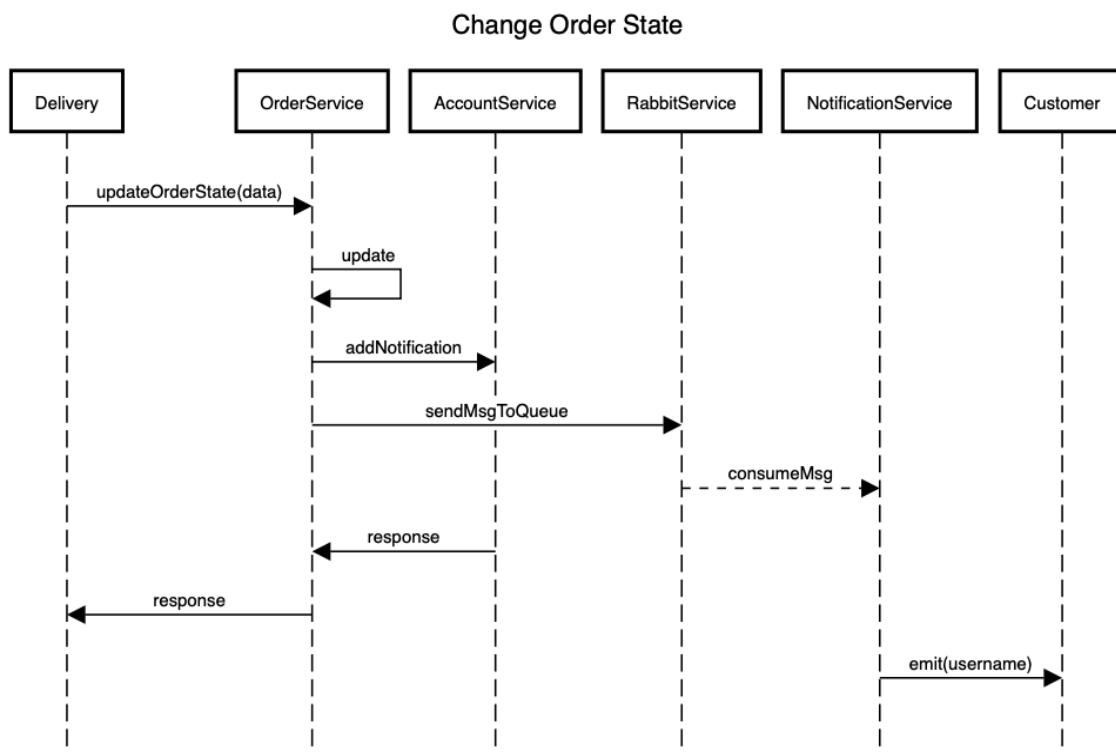


Figura 36: Diagramma Attività Change Order State

Lato Frontend, quando il Fattorino premerà il pulsante per cambiare lo stato dell'ordine, verrà effettuata una chiamata al Backend per aggiornare lo stato dell'ordine. Successivamente verrà chiamata in "AccountService" una API per aggiungere la notifica al Database e notificare al Customer l'avvenuto cambiamento. In risposta si riceverà un semplice ACK dell'avvenuto inserimento della notifica. Infine l'"OrderService" genererà un messaggio inviandolo alla queue di RabbitMQ (Coda di messaggi). Il "NotificationService", che sarà in ascolto in attesa di un messaggio, riceverà una notifica da RabbitMQ con le informazioni necessarie per la generazione della notifica. A quel punto emetterà un evento indirizzato al Customer che verrà notificato dell'avvenuto cambiamento dello stato dell'ordine del Prodotto Acquistato.

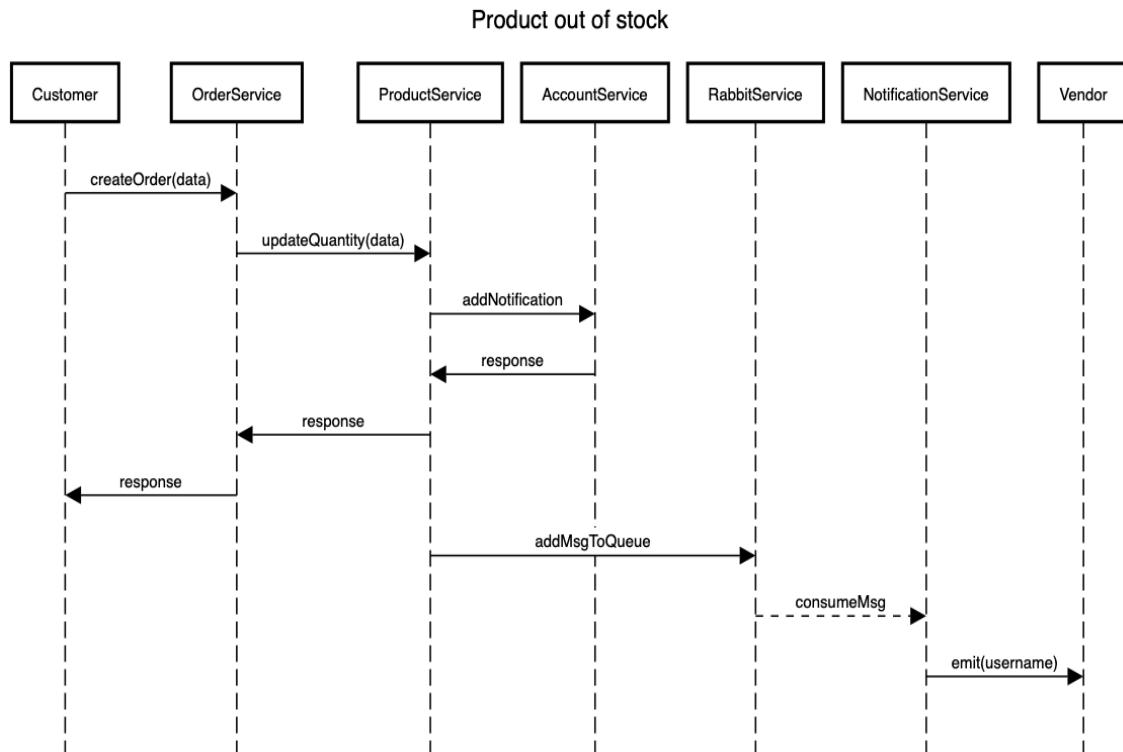


Figura 37: Diagramma Attività Product out of stock

Quando l’utente effettuerà un nuovo Ordine, il Servizio degli Ordini andrà a visualizzare la quantità dei vari ordini relativi a quel determinato prodotto nel carrello. Aggiorerà la quantità del Prodotto anche nel Magazzino e, in caso la quantità sia pari a 0 o inferiore, andrà ad inserire un messaggio nel Database del Venditore andando a notificare al Venditore che il prodotto è esaurito, generando un nuovo messaggio nella coda di RabbitMQ. Il ”NotificationService”, che sarà in ascolto in attesa di un messaggio, riceverà una notifica da RabbitMQ con le informazioni necessario per la generazione della notifica. A quel punto emetterà un evento indirizzato al Commerciante che verrà notificato della quantità insufficiente per la vendita di un determinante Prodotto.

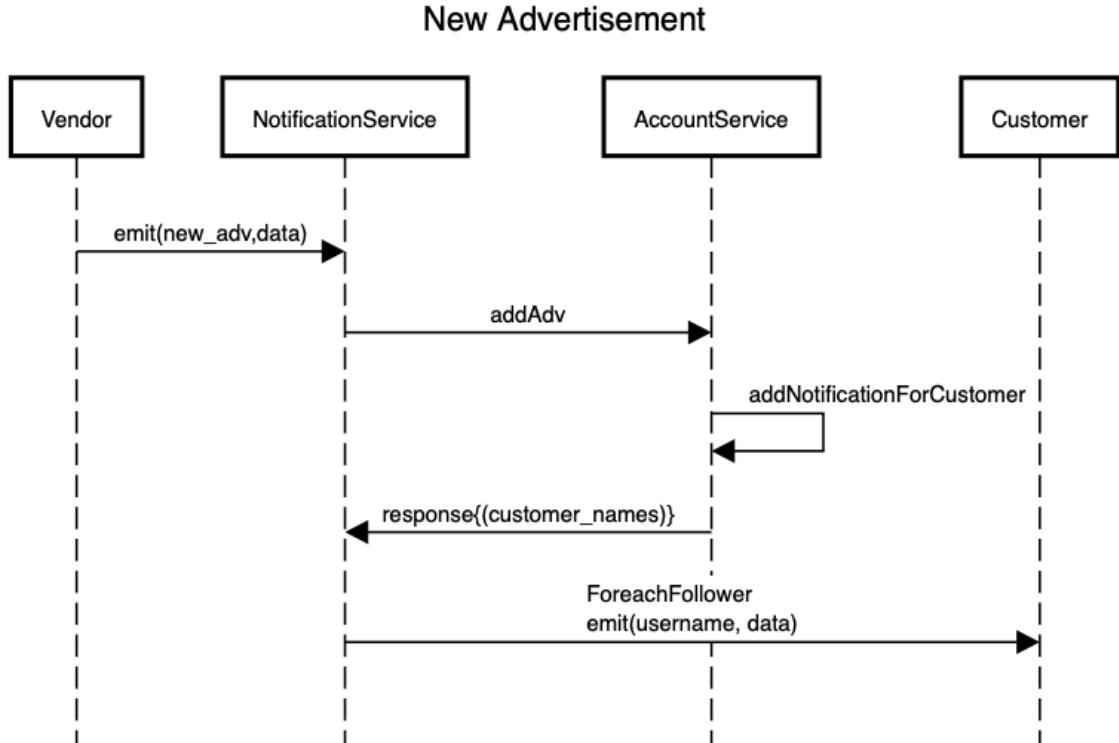


Figura 38: Diagramma Attività New Advertisement

Per la gestione della Creazione degli Annunci, il Commerciante inserisce "Title" e "Body" dell'Annuncio. A quel punto, nel Server, verrà inserito un evento sulla Socket con Target "NotificationService" che possiederà una listen per l'evento di tipo "New Advertisement". Alla sua ricezione, verrà eseguita una chiamata REST all'"AccountService" dell'Account Commercianti in cui vi sarà una API che gestirà la creazione di un nuovo annuncio. Questo annuncio verrà inserito nel Database del Venditore che lo ha generato e, successivamente, andrà a recuperare la lista dei suoi Followers. Per ciascuno di essi andrà ad aggiungere nella loro Collection del database la notifica. Infine prende lo Username dei Followers del Negozio ed andrà a creare una Lista di Username che verrà ritornata al "NotificationService". Il "NotificationService", rimasto in attesa della risposta della suddetta API, riceve la lista degli Username e per ciascuno di essi esegue un emit generando un evento sulla Socket inviandogli i dati della Notifica dell'Annuncio appena creato. Lato Frontend sarà presente una listen che si occuperà di catturare i nuovi eventi sulla WebSocket.

5.6 Struttura Rotte Lato Server

Le varie API esposte dal server sono organizzate sulla base del dominio su cui agiscono. Per questo motivo sono state suddivise in file differenti.

Qui di seguito le specifiche tecniche delle API:

Account Routes				
Metodo	Rotta	Parametri	Risposta	Descrizione
GET	/account/get_user_by_username	username: String	{ profile_type: String, usr: oggetto contenente i dati dell'utente }	Restituisce un oggetto contenente i dati dell'utente e il tipo di profilo dato il suo username.
GET	/account/get_user_by_id	_id: String	{ profile_type: String, usr: oggetto contenente i dati dell'utente }	Restituisce un oggetto contenente i dati dell'utente e il tipo di profilo dato il suo id.

Figura 39: Rotta Account

Admin Routes				
Metodo	Rotta	Parametri	Risposta	Descrizione
POST	/account/admin	{ first_name: String, last_name: String, phone_number: String, email: String, username: String, password: String, shop_name: String, shop_picture: String, description: String, }	{ _id: String }	Registra un nuovo account di tipo amministratore (venditore), restituisce l'id se l'operazione è andata a buon fine, setta il jwt.
GET	/account/admin	tipo_id: String (Ricavato dal JWT)	Oggetto contenente informazioni sull'amministratore	Restituisce un oggetto contenente i dati dell'amministratore.
PUT	/account/admin	{ first_name: String, last_name: String, phone_number: String, email: String, username: String, password: String, shop_name: String, shop_picture: String, description: String, }	{ _id: String }	Modifica i dati di un amministratore e restituisce l'id se l'operazione è andata a buon fine
PUT	/account/admin/position	{ new_coordinates: [Number] }	Oggetto contenente nuove informazioni sull'amministratore	Modifica le coordinate di un amministratore
POST	/account/admin/follower	{ follower_id: String, follower_type: String, }	{ _id: String }	Aggiunge un follower alla lista dei follower di un amministratore

Figura 40: Rotta Vendor

		shop_name: String }		
GET	/account/admin/follows	{ shop_name: String }	{ is_follower: boolean }	Verifica se l'utente con un dato id segue l'amministratore
GET	/account/admin/follower	{ id: String }	{ admin_id: String, followers: lista di follower }	Restituisce la lista di followers di un amministratore
DELETE	/account/admin/follower	{ follower_id: String, follower_type: String, shop_name: String }		Rimuove un follower dalla lista di follower di un amministratore
DELETE	/account/admin	tipo_id: String (Ricavato dal JWT)		Rimuove tutti i dati di un dato amministratore dal database
POST	/account/admin/advertisement	{ adv_title: String, adv_body: String, admin_id: String }		Aggiunge un nuovo annuncio alla lista degli annunci dell'amministratore e genera un notifica per tutti i clienti che seguono questo amministratore
GET	/account/admin_customer	{ shop_name: String, }	Oggetto contenente informazioni sull'amministratore per il cliente	Ritorna un oggetto contenente i dati di un amministratore per un cliente
GET	/account/admins		Oggetto contenente la lista di informazioni sugli amministratori	Ritorna la lista contenente i dati di tutti gli amministratori
POST	/account/admin/notification	{ sender_id: String, sender_type: String, title: String, body: String, admin_id: String }	{ id: String }	Aggiunge una nuova notifica alla lista di notifiche di un amministratore

GET	/account/admin/notification	{ notification_id: String, }	{ _id: String }	Ritorna la lista di tutte le notifiche di un dato amministratore
-----	-----------------------------	------------------------------------	-----------------------	--

PUT	/account/admin/notification	{ notification_id : String }	Oggetto contenente la i notifica modificata dell'amministratore	Modifica lo stato di una notifica
-----	-----------------------------	------------------------------------	---	-----------------------------------

Figura 42: Rotta Vendor Parte 3

Customer Routes				
Metodo	Rotta	Parametri	Risposta	Descrizione
POST	/account/customer	<pre>{ first_name: String, last_name: String, profile_picture: String, email: String, username: String, password: String, date: String, notification: String, }</pre>	_id: String	Aggiunge un nuovo Customer alla Collezione e restituisce l'ID se l'operazione è andata a buon fine, setta il jwt.
GET	/account/customer	_id: String	Oggetto contenente tutte le informazioni sul Customer.	Restituisce tutti i dati del Customer.
GET	/account/customer_id	_id: String		
PUT	/account/customer	<pre>{ first_name: String, last_name: String, profile_picture: String, email: String, username: String, password: String, }</pre>	_id: String	Modifica i Dati del Customer.
DELETE	/account/customer	_id: String		Elimina l'Account del Customer.
PUT	/account/customer/notification	_id: String	_id: String	Modifica lo stato di una notifica
POST	/account/customer/notification	<pre>{ sender_id: String sender_type: String title: String body: String, visualized: Boolean, }</pre>	_id: String	Aggiunge una nuova notifica alla lista di notifiche di un Customer
GET	/account/customer/not_visualized_notification	_id: String	Oggetto contenente la lista di notifiche non visualizzate	Ritorna la lista di notifiche non visualizzate di un Customer

Figura 43: Rotta Customer

Delivery Routes				
Metodo	Rotta	Parametri	Risposta	Descrizione
POST	/account/delivery	{ first_name: String, last_name: String, profile_picture: String, email: String, username: String, password: String, date: String, location: String, }	_id: String	Aggiunge un nuovo Delivery alla Collezione e restituisce l'ID se l'operazione è andata a buon fine, setta il jwt.
GET	/account/delivery	_id: String	Oggetto contenente tutte le informazioni sul Delivery.	Restituisce tutti i dati del Delivery.
GET	/account/delivery_id	_id: String	Oggetto contenente tutte le informazioni sul Delivery	Restituisce un oggetto contenente i dati di un Delivery dato il suo id
PUT	/account/delivery	{ first_name: String, last_name: String, profile_picture: String, email: String, username: String, password: String, }	_id: String	Modifica i Dati del Delivery.
PUT	/account/delivery/position	new_coordinates: [Number]	_id: String	Modifica le coordinate di un Delivery
DELETE	/account/delivery	_id: String		Elimina l'Account del Delivery.

Figura 44: Rotta Delivery

Founder Routes				
Metodo	Rotta	Parametri	Risposta	Descrizione
POST	/account/founder	{ first_name: String, last_name: String, phone_number: String, email: String, username: String, password: String, profile_picture: String, date: String, }	_id: String	Aggiunge un nuovo Founder alla Collezione e restituisce l'ID se l'operazione è andata a buon fine, setta il jwt.
GET	/account/founder	_id: String	Oggetto contenente tutte le informazioni sul Founder.	Restituisce tutti i dati del Founder.
PUT	/account/founder	{ first_name: String, last_name: String, phone_number: String, email: String, username: String, password: String, profile_picture: String, }	_id: String	Modifica i Dati del Founder.
GET	/account/founder/vendor		Oggetto contenente tutte le informazioni sui Vendors del sito.	Restituisce una Lista in formato Json di tutti i Vendor presenti sulla piattaforma.
PUT	/account/founder/vendor	_id: String	_id: String	Modifica lo Stato del Vendor con id specificato
DELETE	/account/founder/vendor	username: String		Elimina l'Account del vendor con username uguale a quello specificato.
DELETE	/account/founder	_id: String		Elimina l'Account del Founder.

Figura 45: Rotta Founder

Authentication Routes				
Metodo	Rotta	Parametri	Risposta	Descrizione
POST	/authentication/login	_id: String	{ profile_type: String, _id: String, }	Permette ad un utente (Customer, Admin, Delivery, Founder) di effettuare il login nel sistema, setta il jwt.
GET	/authentication/logout		description: String	Rimuove il cookie http-only contenente il JWT permettendo il logout dell'utente identificato dall'id passato come parametro nel JWT

Figura 46: Rotta Authentication

Order Routes				
Metodo	Rotta	Parametri	Risposta	Descrizione
POST	/order	{ client_id: String, payment_type: String, card_number: String, country: String, city: String, street: String, info: String, }	_id: String	Crea un nuovo ordine e verifica se tutti i prodotti presenti nel carrello sono disponibili nei vari magazzini, in caso contrario ritorna una lista di prodotti che non sono attualmente disponibili e che verranno spediti non appena riforniti.
GET	/order	_id: String	Oggetto contenente la lista degli ordini di un dato utente.	Restituisce la lista di tutti gli ordini effettuati da un utente .
GET	/orders	{ orderState: String, }	Oggetto contenente la lista degli ordini che sono in un determinato stato	Restituisce la lista degli ordini che sono in un determinato stato specificato dall'utente
GET	/order/info	{ _id: String }	Oggetto contenente le informazioni di un ordine	Restituisce tutte le informazioni di un ordine con un id specifico
GET	/order/customer_id	{ client_id: String }	_id: String	Restituisce la lista degli ordini effettuati da un utente con uno specifico id
PUT	/order/delivery	{ delivery_id: String, delivery_name: String, orderStatus: String, }	Oggetto contenente le informazioni dell'ordine modificato	Modifica lo stato di un ordine.

Figura 47: Rotta Order

Shopping Cart Routes				
Metodo	Rotta	Parametri	Risposta	Descrizione
POST	/shopping_cart	{ client_id: String, vendor_id: String, product_name: String, product_price: String, product_quantity: String, product_sale: String, }	_id: String	Aggiunge un prodotto al carrello se questo non era presente altrimenti ne aggiorna la quantità presente
PUT	/shopping_cart	{ client_id: String, product_name: String, product_quantity: String, }	_id: String	Modifica la quantità di un prodotto presente nel carrello, se la nuova quantità è uguale a zero il prodotto viene rimosso dal carrello. Inoltre ricalcola il nuovo prezzo totale
DELETE	/shopping_cart	{ client_id: String, name: String, vendor_id: String, }	Oggetto contenente le informazioni del carrello	Rimuove un prodotto dal carrello e ricalcola il nuovo prezzo totale
GET	/shopping_cart_by_id	{ _id: String }	Oggetto contenente le informazioni di un carrello	Restituisce le informazioni sullo stato del carrello
GET	/shopping_cart		Oggetto contenente le informazioni del carrello	Restituisce le informazioni sullo stato del carrello
PUT	/shopping_cart/sale	{ name: String, new_name: String, vendor_id: String, sale: String, price: String, }	shopping_cart: String	Modifica lo sconto, il nome e il prezzo di un prodotto nel carrello e ricalcola il prezzo totale

Figura 48: Rotta Shopping Cart

Product Routes				
Metodo	Rotta	Parametri	Risposta	Descrizione
POST	/product	{ vendor_id: String, vendor_name: String, shop_name: String, name: String, type: String, price: String, description: String, rating: String, sale: String, quantity: String, product_picture: String, }	_id: String	Crea un nuovo prodotto e lo aggiunge alla Collezione
PUT	/product	{ vendor_id: String, vendor_name: String, shop_name: String, name: String, type: String, price: String, description: String, rating: String, sale: String, quantity: String, product_picture: String, }	_id: String	Modifica lo stato di un prodotto e propaga la modifica ai prodotti presenti nei carrelli dei clienti
GET	/product		Oggetto contenente la lista di tutti i prodotti di un utente	Ritorna l'oggetto contenente la lista di tutti i prodotti di un utente
GET	/product/info	{ vendor_id: String, name: String }	Oggetto contenente le informazioni su un prodotto con id venditore e nome del prodotto specifico	Ritorna l'oggetto contenente le informazioni su un prodotto dato il nome del prodotto e l'id del venditore
GET	/product/info_id	_id: String	Oggetto contenente le informazioni su un prodotto con id specifico	Ritorna l'oggetto contenente le informazioni su un prodotto dato il suo id
GET	/product/sale		Oggetto contenente la lista dei Prodotti in sconto.	Ritorna la lista dei prodotti attualmente in sconto

Figura 49: Rotta Product

GET	/product/shop_name	shop_name: String	Oggetto contenente la lista dei Prodotti di un Negozio.	Ritorna la lista dei prodotti dato il nome di un negozio
GET	/product/rating		Oggetto contenente la lista dei Prodotti più votati nelle Recensioni.	Ritorna la lista dei prodotti che hanno un rating maggiore di una certa soglia prefissata
GET	/product/recommended	_id: String	Oggetto contenente la lista dei prodotti consigliati per	Ritorna la lista di prodotti consigliati per un determinato utente

			I'Utente.	
GET	/product/vendor	_id: String	Oggetto contenente la lista dei Prodotti di un Venditore.	Ritorna la lista dei prodotti di un certo venditore dato il suo id
GET	/product/type	type: String	Oggetto contenente la lista dei prodotti di una determinata CATEGORIA	Ritorna la lista dei prodotti di una certa categoria specificata
GET	/product/most_sold		Oggetto contenente la lista dei prodotti più venduti.	Ritorna la lista dei prodotti più venduti
GET	/product/names/similar_name	name: String	Oggetto contenente la lista di nomi simili alla stringa specificata	Ritorna una lista di nomi di prodotti che sono più simili alla stringa specificata
GET	/product/similar_name	name: String	Oggetto contenente la lista dei Prodotti con il nome simile alla stringa specificata.	Ritorna una lista di prodotti che hanno il nome più simile alla stringa specificata
POST	/product/most_sold	{ <u>vendor_id</u> : String, name: String, }	Oggetto contenente la lista dei prodotti più venduti.	Aggiunge un prodotto alla lista dei prodotti più venduti se è già presente ne aggiorna la quantità

54
Figura 50: Rotta Product Parte 2

POST	/product/review	<pre>{ username: String, text: String, rating: String, }</pre>	_id: String	Aggiunge una recensione ad un determinato prodotto
PUT	/product/quantity	<pre>{ vendor_id: String, name: String, }</pre>	Oggetto contenente il prodotto aggiornato.	Modifica la quantità di un prodotto
PUT	/product/sale	<pre>{ name: String, vendor_id: String, sale: Integer, }</pre>	_id: String	Modifica lo sconto di un prodotto
DELETE	/product	_id: String		Elimina un determinato prodotto dato il suo id

Figura 51: Rotta Product Parte 3

5.7 Struttura Database

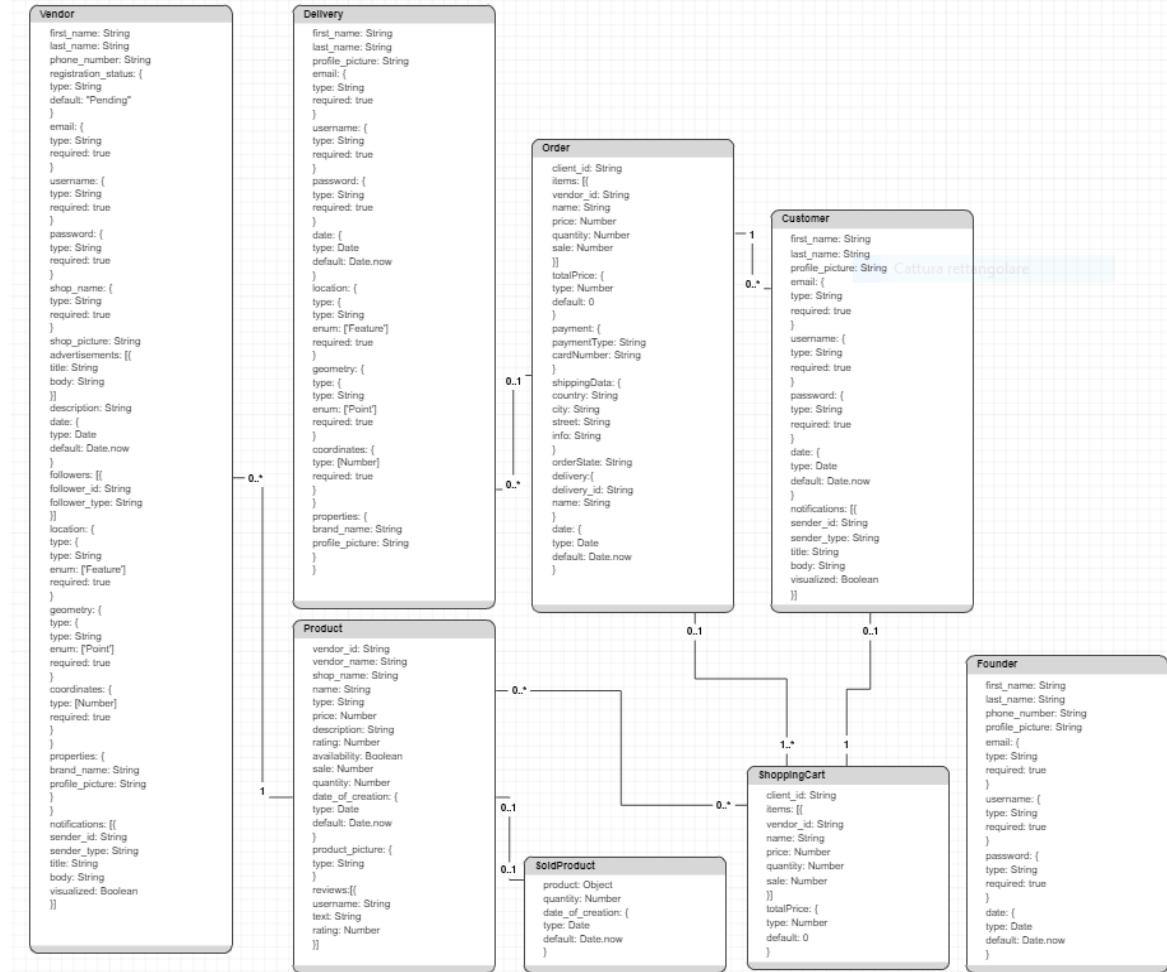


Figura 52: Schema MongoDB

Il gruppo, dopo un'attenta riflessione, ha deciso di memorizzare tutte le informazioni della piattaforma suddividendo i dati in varie collezioni.

Le informazioni relative ai vari Utenti sono state suddivise per Tipologia Utente in Collezioni distinte. (Founder, Customer, Delivery, Vendor)

Lo storico degli Ordini è stato salvato in una Collezione a parte, così da permettere al server di operare più velocemente e riorganizzare gli ordini nella maniera più "pulita" possibile.

La Collezione Carrello si occuperà di salvare tutti i dati relativi ai Prodotti selezionati da ogni singolo Cliente prima dell'effettivo ordine.

Le Collezioni Vendor e Product sono strettamente correlate ma si è deciso di separarle

perché si è pensato che per ogni Venditore ci possano essere centinaia di prodotti messi in vendita. Di conseguenza quella Collezione tenderà a crescere esponenzialmente e separandola si andranno ad ottimizzare le altre operazioni sul Vendor che non influenzano i Prodotti.

6 Test

Le funzionalità del sistema sono state testate dal team su vari Browser (**Opera, Chrome e Firefox**) con l'obiettivo di verificare e garantire la portabilità.

In particolare, i test sono stati atti a verificare che funzionalità complesse come quelle real-time funzionassero allo stesso modo su tutti i Browser principali presenti in commercio.

Le API sviluppate lato server sono state opportunatamente testate utilizzando lo strumento **Postman** che ha permesso al gruppo di verificare che tali API avessero il comportamento desiderato prima di procedere con l'integrazione di esse nel front-end e quindi prima di effettuare il test della funzionalità completa.

6.1 Euristiche di Nielsen

Per quanto riguarda l'usabilità dell'interfaccia, il sistema è stato sottoposto preventivamente ad **Euristiche di Nielsen**. Si elencano di seguito le considerazioni emerse:

- **Controllo e Libertà:** si è tentato di ridurre al minimo il numero di operazioni necessarie da parte dell'utente per portare a termine un task.
- **Aiuto Utente:** Nella piattaforma sono presenti messaggi di errore che vengono espressi e visualizzati in un linguaggio comprensibile all'utente.
- **Riconoscimento e non Ricordo:** Le immagini presenti sulla piattaforma cercano di essere il più autoeplicative possibili e lo stile di ogni pagina è uniforme in tutto il sistema.
- **Visibilità Stato Sistema:** tutte le latenze presenti nel sistema danno un feedback visivo all'utente che lo rendono consapevole del fatto che il sistema sta lavorando per lui.
- **Prevenzione Errori:** Il sistema cerca il più possibile di evitare situazioni ambigue all'utente che lo possano portare a commettere errori. L'utente è guidato da procedure che abilitano l'esecuzione di un task attraverso label, suggerimenti e messaggi di errore.
- **Documentazione:** Data la banale semplicità di utilizzo del sistema, il gruppo non ha ritenuto utile fornire una guida all'utente.
- **Corrispondenza Sistema e Mondo Reale:** Tutti i termini e le icone utilizzate nella piattaforma sono in linea con quelle che caratterizzano i più famosi social network.
- **Consistenza e Standard:** Tutti i pulsanti che abilitano l'avvio di un task presentano le stesse caratteristiche visive. Definita inizialmente una gamma di colori da utilizzare, si è fatto uso uniforme di questi all'interno del sistema in modo da rivolgere l'attenzione dell'utente verso gli elementi principali delle varie schermate.

- **Design ed Estetica Minimalista:** Il gruppo ha utilizzato il principio chiave KISS. Lo si nota fin da subito nella schermata di Home Page. Essa rivolge l'attenzione dell'utente verso i prodotti principali che il sistema mette a disposizione per attirare l'attenzione del consultatore.
- **Flessibilità ed Efficienza d'uso:** Il sistema offre funzionalità che sono già semplici per cui non si è ritenuto necessario introdurre scorciatoie o modalità di utilizzo alternative.

6.2 Usability Test

La piattaforma, una volta realizzata, è stata sottoposta all'attenzione di tutte le tipologie di Utente coinvolte in modo da valutare tramite feedback attivi le loro reazioni e comprendere cosa poteva essere migliorato per le evoluzioni future.

A tutte e 4 le tipologie di utenti sono stati assegnati un elenco di Task da portare a termine.

Il team in questa fase ha ricoperto il ruolo di osservatore cercando di intervenire quanto meno possibile.

Qui di seguito riportiamo il riassunto dei risultati ottenuti:

Fondatore

Il Fondatore intervistato in fase di stesura dei requisiti è stato poi coinvolto nuovamente in fase di testing. I task individuati per quest'ultimo sono:

1. Il Fondatore deve eseguire e completare la registrazione completa del proprio Account.
2. Deve cambiare lo stato di un Account di un Negoziante

Il Fondatore è riuscito a compiere tutti i task assegnatogli con successo e senza la necessità del supporto del gruppo.

Amministratore/Commerciano

Il Commerciano coinvolto anch'esso sia in fase di stesura dei requisiti sia in fase di testing, ha eseguito i seguenti Task:

1. Il Commerciano deve eseguire e completare la registrazione completa del proprio Account.
2. Deve creare un nuovo prodotto
3. Deve, inoltre, modificare e cancellare un prodotto esistente
4. Infine deve creare un annuncio di sconto

Il Commerciano è riuscito a compiere quasi tutti i task assegnatogli senza la necessità dell'intervento del gruppo. L'unica operazione nella quale si sono riscontrate difficoltà è stata la sezione di creazione di un nuovo annuncio. Difficoltà superata grazie al supporto del team. Il pulsante che porta alla sezione di creazione annunci non risulta immediato. Pertanto il Commerciano ha suggerito di rivedere quella parte e renderla più accessibile. Inoltre sono state evidenziate le seguenti necessità:

- Permettere al Commerciante la selezione e la gestione di determinati Fattorini della propria Zona
- Aggiunta di una Chat di Supporto in caso di problemi per ricevere aiuto dal Fondatore

Fattorino

Il Fattorino è stato coinvolto anch'esso sia per la fase di stesura dei requisiti sia per la fase di testing. Al termine del Testing il Fattorino ha eseguito i seguenti Task:

- Il Fattorino deve eseguire e completare la registrazione completa del proprio Account.
- Deve riuscire ad impostare la propria posizione tramite la mappa.
- Deve riuscire ad accettare ordini.
- Infine deve riuscire a notificare l'avvenuta consegna al Cliente del prodotto ordinato.

Il Fattorino è riuscito a compiere tutti i task assegnatogli con successo e senza la necessità del supporto del gruppo.

Cliente

Per la parte relativa al Cliente si è voluto valutare la reazione al sistema da parte di utenti di diverse fasce d'età. Sono stati coinvolti:

- **Cliente 1:** Persona giovane (24 anni) molto pratico di Applicazioni Web di vario genere.
- **Cliente 2:** Persona adulta (63 anni) che fa un uso limitato del Web, tendenzialmente Facebook.

Ad entrambi i possibili Clienti sono stati sottoposti i seguenti Task:

- Il Cliente deve eseguire e completare la registrazione completa del proprio Account.
- Deve riuscire a ricercare agevolmente un prodotto all'interno della piattaforma.
- Deve riuscire a gestire il proprio Carrello
- Inoltre deve riuscire ad effettuare un ordine.
- Deve saper visionare la posizione in real-time del fattorino e comprenderla.
- Infine deve saper giostrarsi tra le notifiche varie di stato del proprio ordine.

Il Cliente 1 è riuscito a compiere tutti i task assegnatogli con successo e senza la necessità del supporto del gruppo.

Ha trovato però scomodo l'utilizzo delle notifiche ed avrebbe preferito delle Email o delle notifiche direttamente tramite SMS, senza dover ogni volta effettuare l'accesso alla piattaforma.

Il Cliente 2 ha trovato difficoltà nell'eseguire i task a lui assegnati, soprattutto quelli di notifica e tracking del fattorino, per i quali ha chiesto supporto al gruppo per portare a termine i Task.

Inoltre il Cliente non ha prodotto dei veri e propri suggerimenti ma durante l'esperienza ha commentato il fatto di trovarsi meglio chiamando direttamente il venditore per l'acquisto e farsi dare suggerimenti direttamente al telefono evitando la procedura online.

7 Deployment

7.1 Installazione

- **Si cloni il Repository:** git clone <https://github.com/GiovanniPoggi/Gotta-Shop-Em-All-Pasqualini-Poggi-Konchenkov.git>
- **Si apra una Shell Bash**
- **Spostarsi nella Cartella del Progetto ed eseguire i seguenti comandi:**
 - **docker-compose build:** Per compilare il progetto.
 - **docker-compose up:** Per lanciare il progetto.
- **Se si desidera ripristinare i dati del database si apra una nuova Shell Bash e si esegua i seguenti comandi:**
 - **docker ps -a:** Per reperire le informazioni dai Container Docker presenti e quindi per poter visualizzare il nome del container del Database.
 - **docker exec -it ”NOMECONTAINERDB” bash:** Per aprire una Shell Bash all'interno del Container target.
 - **cd /backup/:** Per spostarsi nella directory target.
 - **mongorestore /backup/:** Per caricare i dati del database presenti nella directory /backend/backup.

7.2 Messa in funzione

- **Si colleghi alla piattaforma tramite un Browser:** <http://localhost:8080/>

8 Conclusioni

Il gruppo, dopo i risultati ottenuti dai test effettuati ed i consigli espressi dagli utenti coinvolti in fase di test, ha tratto le seguenti considerazioni.

Per quanto riguarda le funzionalità interessate dalla tipologia di account "Commercante", ci si potrà sicuramente focalizzare sul rendere più intuitiva e fluida l'interfaccia dedicata alla creazione di un nuovo annuncio. Inoltre, come suggerito dal Commercante coinvolto nella fase di testing, si potrebbe migliorare la piattaforma aggiungendo la possibilità di selezione di determinati Fattorini nella propria Zona e l'aggiunta di un'opportuna Chat di Supporto in caso si riscontrino problemi nell'utilizzo della piattaforma. Oltre a queste funzionalità suggerite, si potrebbe integrare il sistema con una serie di funzionalità statistiche su misura del Commercante riguardante i vari prodotti messi in vendita come:

- Prodotto più venduto
- Prodotto meglio recensito
- Prodotto meno venduto
- Prodotto con più resi
- Statistiche ed andamento Visualizzazione Prodotti

Per risolvere i problemi evidenziati dai Clienti in fase di testing, si potrebbe aggiungere una funzionalità aggiuntiva che permette di contattare direttamente il Venditore e prendere accordi con esso evitando l'ordine e l'acquisto tramite piattaforma. Ad esempio aggiungendo e rendendo più visibile il contatto con il Commercante tramite chiamata telefonica.

8.1 Commenti Finali

Il gruppo è consapevole del fatto che, in fase di testing, sarebbe stato molto più efficace coinvolgere gruppi di persone più numerosi e diversificati sotto l'aspetto dell'età per ogni tipologia di Account in modo da ottenere un numero di Feedback maggiore e più rilevante. Tuttavia il team ha optato per un numero esiguo di persone sia perché il progetto in sé non conteneva un numero elevato di funzionalità, sia perché si disponeva di un limite di tempo imposto per la realizzazione della piattaforma. Nonostante queste condizioni, si è cercato di selezionare le tipologie di Utenti coinvolti il più rappresentative dei target finali.

La realizzazione di questo progetto ha contribuito ad arricchire il bagaglio culturale delle tecniche acquisite fino ad ora a Lezione, dandoci la possibilità di porci nelle condizioni di mettere alla prova le nostre conoscenze in campo pratico nella realizzazione di funzionalità real-time e coinvolgendo tecnologie diversificate nella stessa soluzione.

Nella realizzazione della piattaforma ha giocato un ruolo importante l'utilizzo di Angular, che ha contribuito nel migliorare l'organizzazione del codice e ci ha permesso di

apprendere l'utilizzo di un nuovo Tool di Supporto alla programmazione lato Client. In aggiunta, questo progetto è stato utile anche nello sviluppo di skill ulteriori di programmazione Javascript (Lato Backend) pura e Typescript (Lato Frontend), cosa che non ci è stata possibile nel corso della Triennale a causa dell'utilizzo massiccio di framework come JQuery.

Javascript è l'unico linguaggio interpretato visto fino ad ora nella nostra carriera universitaria e ci è stato utile poterlo approfondire ai fini di confronto con i linguaggi compilati. L'idea di questo progetto è nata dalla conoscenza di un Commerciale che, a causa della recente Pandemia con annesso Lockdown, si è ritrovato a dover chiudere la sua attività senza possibilità di avere introiti dalla vendita di prodotti online. Questo ci ha dato modo di sperimentare sul campo il rapporto con un vero committente, dal sorgere della problematica alla stesura di una soluzione condivisa.

Le difficoltà nella realizzazione del progetto non sono mancate, prima fra tutte la stesura di un elenco dei requisiti che rappresentino concretamente l'idea reale del committente. In fase di testing, questo ha portato ad evidenziare aspetti che il committente pensava sarebbero stati implementati ma che di fatto non erano stati espressi da nessun requisito iniziale.

9 Bibliografia

1. Socket.io <https://socket.io/>
2. Cropper.js <https://github.com/fengyuanchen/cropperjs>
3. Bootstrap <https://getbootstrap.com/>
4. BCrypt <https://github.com/kelektiv/node.bcrypt.js>
5. Leaflet <https://leafletjs.com/>
6. Json Web Token <https://github.com/auth0/node-jsonwebtoken>
7. Mongoose <https://mongoosejs.com/>
8. RabbitMQ <https://www.rabbitmq.com/>