



Dipartimento di ingegneria Informatica,
Modellistica, Elettronica e Sistemistica (DIMES)
Corso di Laurea Magistrale in Ingegneria
Informatica

**Relazione del PROGETTO DI SISTEMI DISTRIBUITI E
CLOUD COMPUTING**

**Creazione di un e-commerce, lato web e lato
mobile per Android**

Studente:
Mattia Gatto, 216649

Docenti:
Domenico Talia

Esercitatore:
Loris Belcastro

Sommario

1. Introduzione	3
1.1. Descrizione e finalità del progetto	3
2. Applicazione lato web	4
2.1. Persistence	4
2.2. Back-End	5
2.3. Front-End	5
3. Applicazione lato mobile	12
3.1. Architetture e servizi utilizzati: Firestore di Firebase	12
3.2. APACHE Cordova	13
3.3. Applicazione Android	13

1. Introduzione

Si vuole realizzare un'applicazione web e un'applicazione Android il cui scopo sarà quello di creare un "e-commerce" che possa essere utile al pubblico dell'attività commerciale che possiede mio padre.

L'attività consiste in una eliografia, ossia un luogo in cui è possibile sia rifornirsi di elementi di cartoleria, cancellaria e copisteria, sia avere a disposizione molti altri tipi di servizi, che vanno dalla definizione di progetti a larga scala fino alle relative stampe di essi.

Per poter far ciò un buon incremento della visibilità dell'attività commerciale ed una migliore possibilità in termini di acquisto di prodotti e di commissione di lavori da effettuare, risiede nel possedere un sito web che mostri ciò che dispone e si offre, ma soprattutto un'applicazione mobile, poiché ad oggi, ognuno di noi preferisce effettuare acquisti online, particolarmente se si tratta di prodotti d'uso quotidiano a basso prezzo.

1.1. Descrizione e finalità del progetto

Il processo di sviluppo del progetto è stato affrontato seguendo due fasi:

1. La definizione di un'applicazione che utilizza SPRING per una migliore definizione della gestione delle informazioni, e quindi una migliore struttura di back-end che si interfaccia con un modello entità relazione definito in mysql, seguita da una struttura di front-end sviluppata in angular per avere una migliore visibilità per l'utente che usa il programma e di conseguenza una migliore interazione.
2. Successivamente grazie a Firebase, la piattaforma per la creazione di applicazioni per dispositivi mobili e web sviluppata da Google, è stata definita la gestione di tutte le informazioni sfruttando esclusivamente angular, il quale grazie alle sue peculiarità riesce a prendere e ricevere informazioni dai servizi offerti da Firebase.

Nello specifico utilizzando Firestore ho definito il modello entità relazione che precedentemente veniva utilizzato tramite MySQL. Qui a causa del linguaggio nosql usato da Firestore che memorizza i dati in file json, sono state apportate diverse modifiche e infine è stato utilizzato un servizio chiamato Apache Cordova che attraverso una selezione di comandi predefiniti su console, riesce a inserire nel progetto angular quei file che mancano al fine di poter generare l'eseguibile per dispositivi mobili. Personalmente ho preferito per il momento aggiungere esclusivamente la piattaforma Android e quindi generare un file eseguibile APK, ma comunque è possibile in ogni caso aggiungere una qualsiasi altra piattaforma.

2. Applicazione lato web

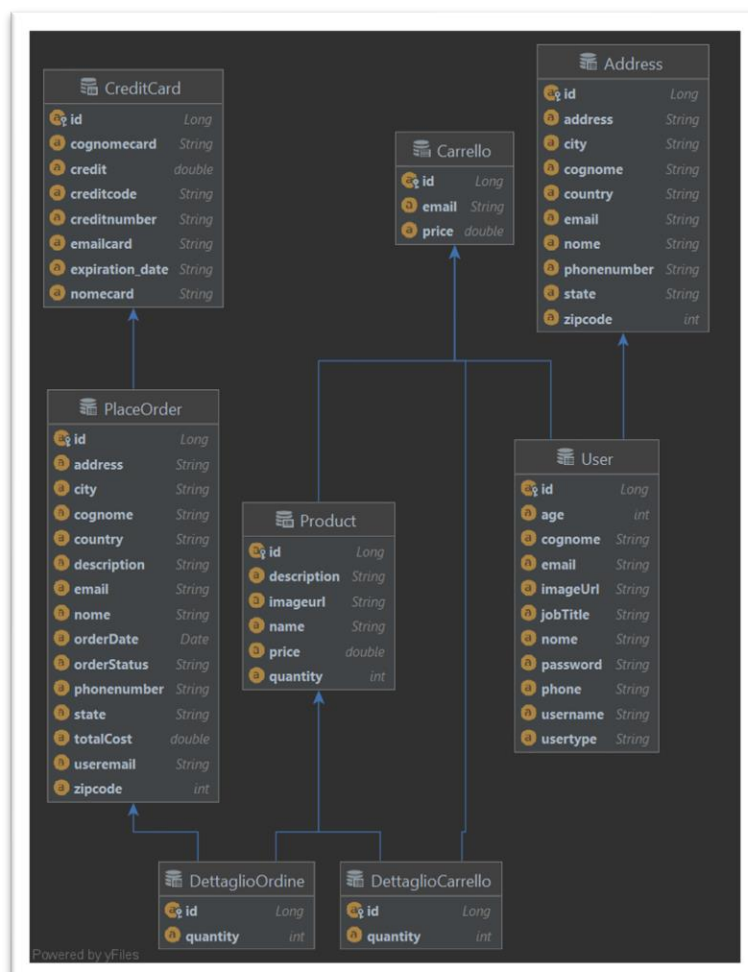
Come accennato nell'introduzione inizialmente è stata definita l'applicazione lato back-end attraverso l'utilizzo di SPRING che è stato implementato per gestire richieste di tipo http da eseguire con il database che si trova in locale su MySQL.

Per quanto riguarda la parte di front-end invece sono state generati dei metodi all'interno di un service, il quale fa delle richieste http a Spring seguendo la metodologia CRUD e quindi definendo le 4 operazioni principali:

- Create, INSERT
- Read (Retrieve), SELECT
- Update, UPDATE
- Delete (Destroy), DELETE.

2.1. Persistence

Il modello entità relazione definito per gestire al meglio la struttura che ho voluto dare ai miei dati viene esposto meglio attraverso questo diagramma generato automaticamente tramite il framework messo a disposizione dal software intellij-idea sul quale risiede tutta la parte di back-end della nostra web app.



2.2. Back-End

Dopo aver definito il modello che dovrà assumere il nostro database, ed aver impostato le giuste e opportune configurazioni nel pom.xml affinché il codice possa interagire il meglio possibile con il database gestito da MySQL, si è proceduti con la definizione di:

1. File di tipo repository che sono interfacce, precisamente una per ogni entità definita. Esse risultano molto utili in quanto abbiamo la possibilità di avviare tramite semplici comandi del tipo "findAll()" o "findById(id)" interazioni con il nostro database, il quale interpreta il significato vero e proprio di questi comandi e aggiunge, rimuove, aggiorna o restituisce rispettivamente ciò che semanticamente il comando vuole.
2. Nella fase successiva sono state definite le classi Service, anche questa volta una per ogni entità, che attraverso una variabile privata che identifica il rispettivo repository, va a definire funzioni java per l'interazione e la raccolta di informazioni con il nostro database.
3. Infine, nell'ultima fase è stata sviluppata una classe che ha la funzione di Controller, ossia fornisce una serie di metodi che sono del tipo CRUD per ognuna delle entità nel modello. Questi metodi lanciano richieste http di tipo:
 - a. get per prendere informazioni tramite parametri passati tramite l'url delle richieste http, e sfruttando le opportune classi di tipo service;
 - b. post per ricevere un corpo di informazioni, che saranno disponibili in formato json e andranno a costituire le variabili che caratterizzeranno le singole entità in modo tale da poter aggiungere nuovi campi.
 - c. Put per ricevere un corpo di informazioni, che serviranno per aggiornare campi precedentemente creati e quindi modificare i singoli attributi di essi.
 - d. Delete per rimuovere dal nostro database attraverso l'identificazione di un particolare attributo o dell'id di un campo, le informazioni che non ci servono più, ovviamente i parametri saranno passati tramite l'url delle richieste http.

Arrivati alla conclusione delle seguenti operazioni siamo passati alla configurazione dell'ambiente per la parte di front-end.

2.3. Front-End

Inizialmente è stato configurato l'ambiente opportuno al fine di riuscire ad interagire con IntelliJ IDEA connettendosi all'url "localhost://8080" per poter sfruttare tramite successivi comandi l'interazione con la parte di backend attraverso metodi che lanciano richieste http di tipo get, put, post e delete e come risposta riceveranno entità di tipo Observable utili per effettuare operazioni con il linguaggio funzionale.

Dopo aver configurato opportunamente Angular, è stato definito il route che ha il ruolo importante di riuscire a gestire la navigazione tra pagine web.

Il fulcro della programmazione lato front-end è rappresentato dall'Apiservice, il servizio che gestisce tutti i metodi che saranno utili per la gestione delle richieste put, post, get e delete verso l'applicazione lato backend.

Inoltre è stato introdotto l'utilizzo di un service che implementa CanActivate, una particolare componente in angular che consente l'utilizzo di un meccanismo utilissimo nell'ambito della

gestione della sessione del singolo utente.

Infatti l'utente loggato può navigare fra le pagine solo fino a quando non richiede il logout, in tal caso gli è impossibile andare verso le pagine della nostra applicazione eccetto la pagina di login o di logup.

Nella parte di front-end vediamo differenti componenti:

1. LoginComponent: è stata definita per il lancio del sito, infatti appena viene lanciato il comando "ng serve" per avviare Angular è la prima pagina a essere visualizzata, e in essa vengono gestite e controllate le informazioni per l'accesso. Ecco come appare:

The screenshot displays the LoginComponent interface. On the left, there is a white background with the title "Sign in". Below the title are two input fields: "Email" and "Password". A blue button labeled "SIGN IN" is positioned below the password field. On the right, there is a dark gray background with the text "Hello, Friend!" and a smaller line of text "Enter your personal details and start journey with us". A white button labeled "SIGN UP" is centered at the bottom of the dark gray area.

2. RegisterComponent: è stata definita per la registrazione di un utente, in essa sono state effettuate tutte le opportune operazioni dedite alla possibilità di aggiungere il nuovo user solo nel caso in cui tale e-mail non sia stata mai registrata nel sito. Ecco come appare:

The screenshot displays the RegisterComponent interface. On the left, there is a white background with the title "Create Account". Below the title are seven input fields: "Email", "Password", "Name", "Cognome", "Username", "JobTitle", and "Phone". A blue button labeled "SIGN UP" is positioned below the phone field. On the right, there is a dark gray background with the text "Welcome Back!" and a smaller line of text "To keep connected with us please login with your personal info". A white button labeled "SIGN IN" is centered at the bottom of the dark gray area.

- AdminComponent, rappresenta quella componente lato amministratore, per l'aggiunta di nuovi prodotti, la rimozione di vecchi, o la modifica di essi. Al suo interno vi risiede una componente per la visualizzazione degli ordini effettuati dagli utenti, con le relative informazioni di spedizione, o pagamento. Ecco come appaiono:

Admin Manager
Esci
ordini
Add Product
Search Products...

evidenzia tore
buono

€ 3.5

Quantità : 69

penna bic

€ 2

Quantità : 799

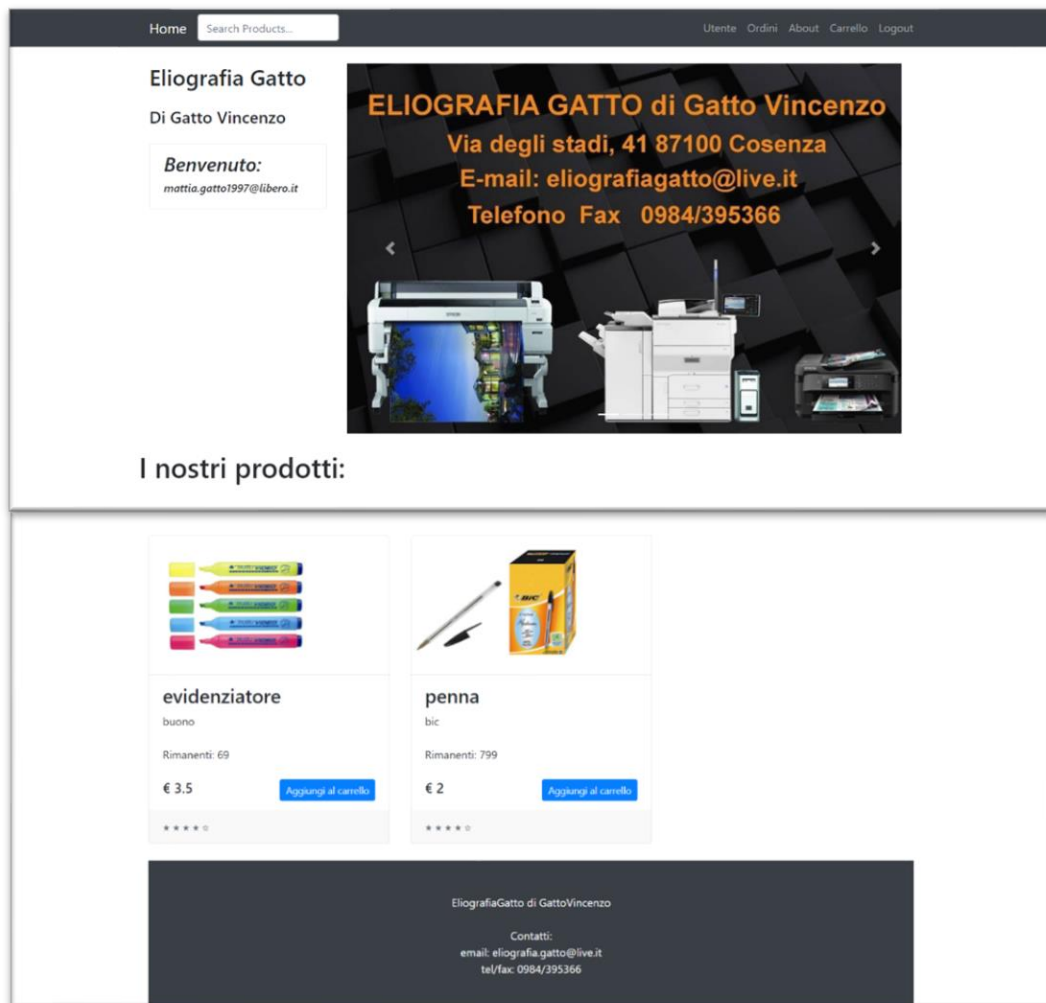
Admin

Tabella ordini

Id	Data	Ora	name	email	Indirizzo	Numero	Info	CreditCard	totalCost	Status	Prodotti(quantità, Id, Name, subTot)
1	2021-05-31	20:25:57	Mattia Gatto	mattia.gatto1997@libero.it	Contrada Fontanesi P.S., Castrolibero, 87040, Cosenza, Italy	3452775099	Chiamare prima se possibile	1234123412341234	€ 5	Completato	1 1 evidenziatore €1.5 1 3 righello €3.5
2	2021-05-31	20:29:09	Mattia Gatto	marco.gatto1997@libero.it	Via Roma n°4, Castrolibero, 87040, Cosenza, Italy	3452775030		1222144412341222	€ 13.5	Completato	1 3 righello €3.5 10 2 penna €10

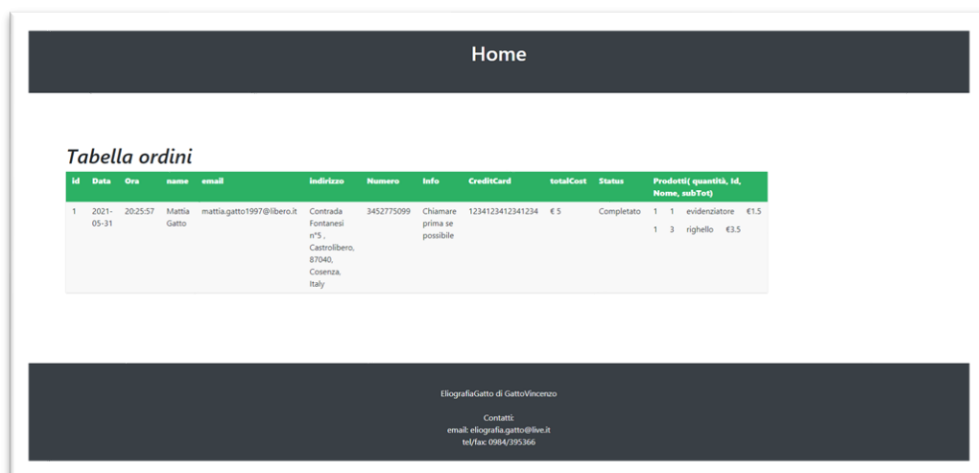
Elicografacatto di GattoVincenzo
Contatti:
email: elicografacatto@live.it
telefono: 0984/205100

4. HomeComponent, è invece la componente dedicata alla pagina iniziale post login o logup per il cliente, in essa vengono fornite varie informazioni più la possibilità di ricerca e aggiunta dei prodotti che si desidera.



Essa al suo interno dispone della possibilità di:

- a. Verificare lo storico degli ordini effettuati, definiti grazie ad ordercomponent:



- b. Verificare le informazioni dell'utente attualmente loggato, e inserire delle informazioni di default per l'indirizzo di consegna, grazie al usercomponent.

The image displays three sequential screenshots of a web application interface, likely for user profile management and address entry. The first screenshot shows the 'Informazioni utente' (User Information) section, which includes fields for 'Full Name' (split into 'Nome' and 'Cognome'), 'User email', and 'password'. The second screenshot shows the 'Address info per consegna' (Delivery Address Information) section, which includes fields for 'jobTitle', 'phone', and a 'Submit' button. The third screenshot shows the 'Address info per consegna' section with more detailed address fields, including 'Street Address', 'Country', 'City', 'Province', 'Postal / Zip Code', and 'Phone Number', along with a 'Submit' button.

Modifica Utente [Go to Home](#)

Informazioni utente

Full Name *

Mattia Gatto

Nome Cognome

mattia.gatto1997@libero.it *****

User email password

Mattia Gatto 8

username age

ingegnere informatico

jobTitle

3452775099

phone

Submit

Address info per consegna

Mattia Gatto

Nome Cognome

Contrada Fontanesi n°5

Street Address

Mozambique

Country

Castrolibero Italia

City Province

87040

Postal / Zip Code

Phone Number *

3452775099

Submit

- c. Infine vi è la possibilità di verificare gli ordini del proprio carrello , modificare la quantità dei prodotti, rimuoverne uno di essi, o procedere all'acquisto dei prodotti presenti nel carrello, questo procedimento sarà possibile grazie al cartcomponent



dedito alla visualizzazione del carrello e dei suoi prodotti e all'inserimento di informazioni per generare un Placeorder, ossia un ordine vero e proprio.

L'ordine verrà effettuato se verrà inserita una carta di Credito.

Per il momento la carta di credito è stata gestita in modo tale che se accetta i canoni del formato delle carte e se dovesse essere una carta già inserita da un utente, allora in tal caso la somma dei prodotti viene scalata dal credito della carta, altrimenti se il credito dovesse essere insufficiente l'acquisto non verrà effettuato. Per una questione di comodità a fine progettuale ogni qual volta che l'utente inserisce le informazioni di una carta che hanno formato valido ma che ancora non esiste nel db, viene generata una nuova carta con credito pari a €1000.

[Go to Home](#)

Carrello

 penna € 2	 evidenziatore € 3.5
<input type="text" value="1"/>	<input type="text" value="1"/>
Update Delete	Update Delete
SubTot: € 2	SubTot: € 3.5

TOT € 5.5

Informazioni consegna e pagamento

Full Name

First Name

Last Name

E-mail

example@example.com

Contact Number

Billing Address

Street Address

City

Country / Province

Postal / Zip Code

Country

Special Instructions

Payment Methods

First Name

Last Name

Credit Card Number

Security Code

Card Expiration

[Submit Order](#)

- d. Avere più informazioni riguardo l'attività commerciale attraverso il tasto ABOUT che si ricollega ad una componente che ha come unico scopo quello grafico, ossia aboutcomponent.

Eliografia Gatto di Gatto Vincenzo

Qualcosa sulla nostra attività

L'eliografia Gatto è un attività commerciale fondata circa 25 anni fa.

Qui è possibile effettuare qualsiasi forma di lavoro di tipo eliografico a prezzi bassissimi.

[Go to home](#)



Vincenzo Gatto
Proprietario dell'attività commerciale

Nato : 16-06-1965
email: eliografiagatto@live.it

[Informazioni](#)

DOVE SIAMO?

Via degli stadi 41, Cosenza(CS)



3. Applicazione lato mobile

In questa fase progettuale si è annullata la parte di back-end per utilizzare esclusivamente una copia della parte sviluppata di front-end la quale è stata sottoposta a numerose modifiche al fine di riuscire ad utilizzare sempre sfruttando l'apiscervice, i servizi forniti da Firebase.

Definita la configurazione lato Google e importata nell'environment la variabile firebase per lo sviluppo di applicazioni mobile web si è definito un insieme di metodi e si sono effettuate modifiche tra le varie component per poter utilizzare le potenzialità fornite da Firestore.

A livello grafico l'applicazione web risulta essere identica alla visualizzazione delle componenti sopra esposta, il vantaggio che si ha qui e che non vi è alcun bisogno di utilizzare né spring e né MySQL in quanto è esclusivamente angular a far funzionare il tutto. Questi vantaggi sono dovuti all'utilizzo di Firestore.

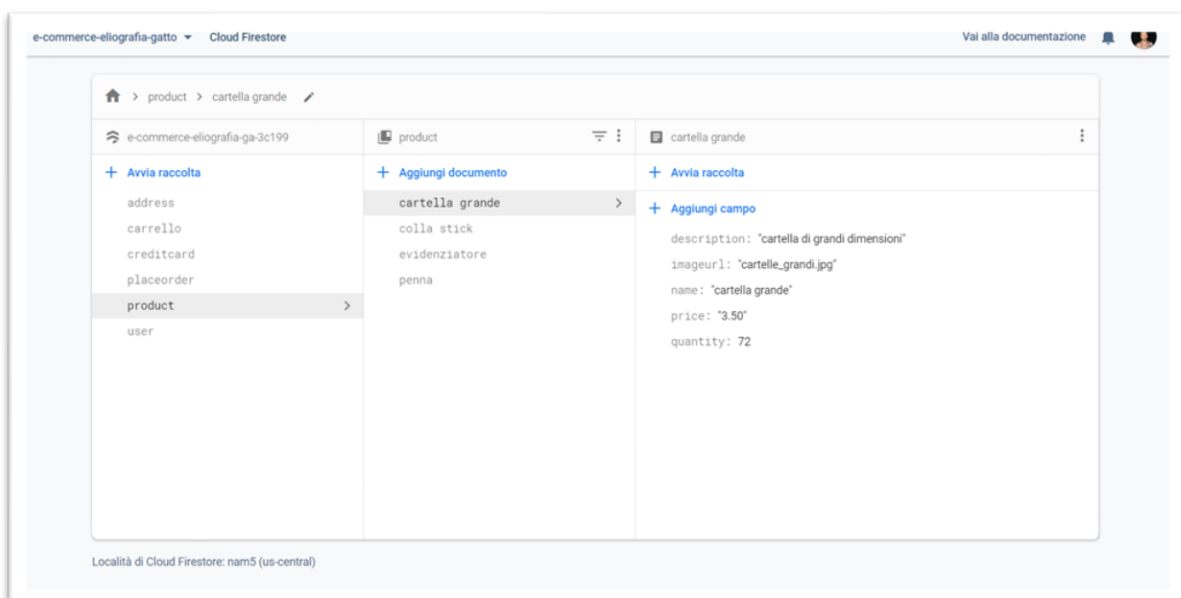
Un ulteriore vantaggio è quello di poter utilizzare gli stessi dati di archiviazione tra app web e applicazione mobile, poiché entrambe utilizzano lo stesso Firestore.

3.1. Architetture e servizi utilizzati: Firestore di Firebase

Come implementazione di servizi si è preferito utilizzare Firestore di Google poiché risulta essere molto potente in termini di archiviazione di dati. Grazie alle peculiarità sopra indicate, inoltre il poter visualizzare dati in real time risulta un ulteriore vantaggio.

In termini di archiviazione come accennato nell'introduzione, si è avuta la necessità di dover reimpostare il modello entità relazione, in quanto Firebase non possedendo un linguaggio SQL ma uno NOSQL, non può sfruttare le potenzialità definite dall'utilizzo di chiavi primarie o esterne, e quindi di relazioni oneToone, oneTomany, manyToone, manyTomany.

Per queste motivazioni le informazioni sono state inserite sotto forma di raccolte con la maggior parte di esse indicizzate dall'email dell'utente al fine di simulare una sorta di chiave per ricercare le informazioni, mentre per altri dati come per i prodotti sono stati utilizzati come indicizzatori i nomi specifici.



3.2. APACHE Cordova

Con questo potente framework è stato possibile generare un'applicazione per dispositivi mobili da un'applicazione web scritta in angular. Infatti, con ciò riusciamo ad aggiungere tutti quei file mancanti che servono per riuscire a far avviare il Gradle, per la generazione di un APK file che possa essere successivamente eseguito su qualsivoglia dispositivo.

Le operazioni da eseguire sono le seguenti e sono molto semplici, anche se come requisiti ha bisogno che siano segnalate le variabili d'ambiente relative ad:

- Gradle
- Android SDK
- JAVA_HOME con versione 1.8 e non superiore.

Del resto, basta avere npm per la gestione dei pacchetti di nodejs ed eseguire i seguenti passi:

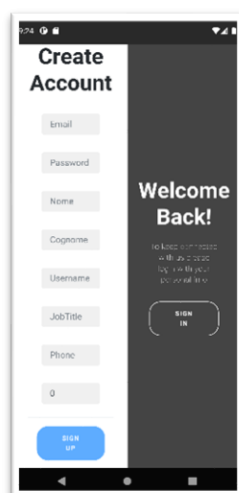
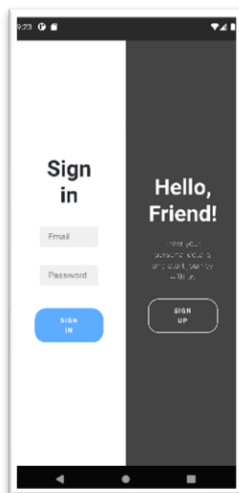
1. Installazione di Cordova, la riga di comando di Cordova viene eseguita su Node.js ed è disponibile su NPM. Basta aprire un prompt dei comandi o Terminale e digitare, ***npm install -g cordova***.
2. Creare un progetto Cordova vuoto utilizzando lo strumento da riga di comando e digitando, ***cordova crea MyAppcordova create MyApp***.
3. Aggiungere una piattaforma a piacere all'interno del progetto Cordova, digitando ***cordova platform add <platform name>***.
4. Spostare tutti i file nel nostro progetto lato web sovrascrivendoli, tranne il file angular.json che deve essere inglobato all'interno di quello già esistente, cambiare base di index.html e impostarlo su www.
5. Esegui ***cordova build <platform name>***.
6. Esegui ***cordova run <platform name>***

In questo modo avvieremo la nostra applicazione, nel nostro caso essa verrà avviata su un device di Android studio.

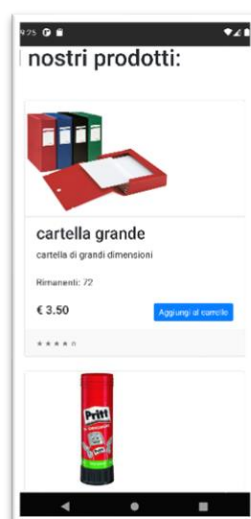
3.3. Applicazione Android

Per generare l'applicazione è stata utilizzata la versione creata tramite Angular e Firebase, e sfruttando questo framework della famiglia APACHE chiamato CORDOVA il risultato ottenuto è stato il seguente:

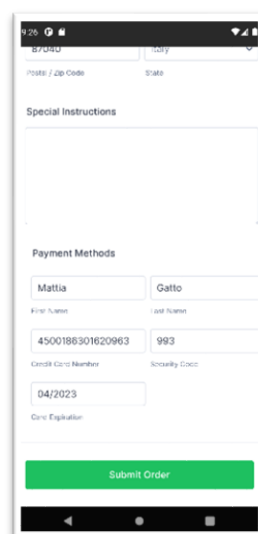
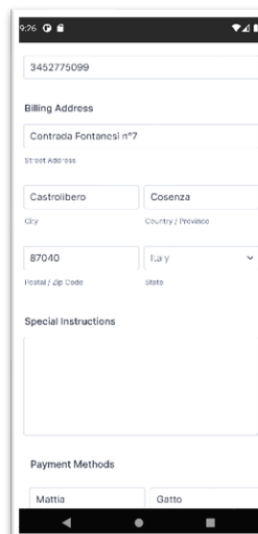
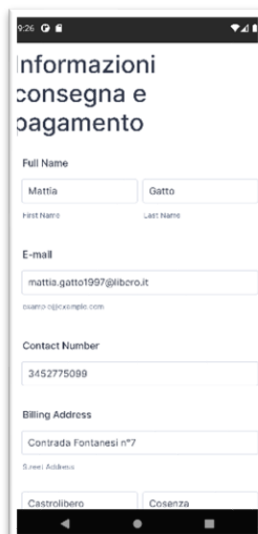
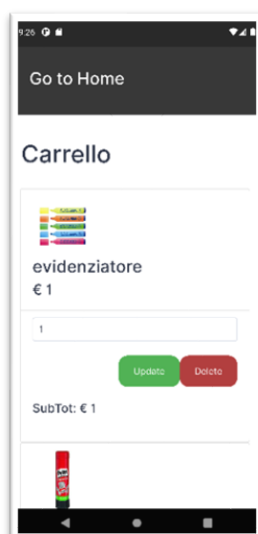
- Fase di login e di logup:



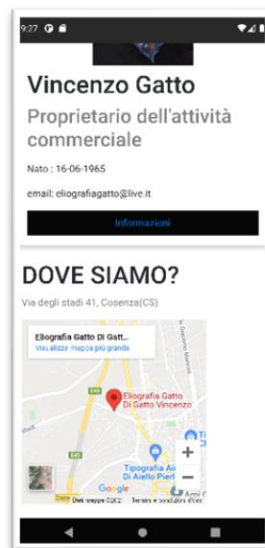
- Home page:



- Carrello, definizione ordine e pagamento.



- Pagina di informazioni:



- Tabella ordini:



- Admin Page:

