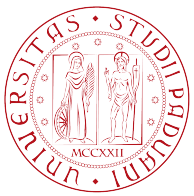


**Università degli Studi di Padova**

DIPARTIMENTO DI MATEMATICA

Corso di Mobile Programming and Multimedia

# Relazione del progetto SmartOrder

Studenti:
**Alberto Bezzon 1211016**
**Tommaso Carraro 1210937**

**Anno Accademico 2018-2019**

# Indice


1 Introduzione . . . . . 4
2 La scelta del framework . . . . . 4
3 Scelta delle tecnologie web . . . . . 5
4 Informazioni utili all'utilizzo dell'applicazione . . . . . 5
5 Progettazione frontend . . . . . 7
5.1 Login e tutorial . . . . . 8
5.2 Carrello . . . . . 8
5.2.1 Aggiunta di un articolo al carrello . . . . . 8
5.3 Visualizzazione ordini . . . . . 9
5.4 Inventario . . . . . 9
6 Mobile design . . . . . 10
6.1 Bottoni . . . . . 10
6.1.1 Bottone indietro . . . . . 11
6.2 Menu . . . . . 13
6.3 Progressive disclosure . . . . . 13
6.4 Uso della tastiera . . . . . 14
6.5 Aiuti per l'utente . . . . . 14
6.6 Elementi nativi . . . . . 14
6.7 Gesture . . . . . 14
6.8 Considerazioni finali . . . . . 14
7 Backend . . . . . 15
7.1 Architettura . . . . . 15
7.2 Web service . . . . . 15
7.2.1 Servlet . . . . . 15
7.2.2 Utility . . . . . 18
7.3 Database . . . . . 18
7.3.1 Database per l'autenticazione . . . . . 18
7.3.2 Database aziendale . . . . . 19
8 Conclusioni . . . . . 20

# Elenco delle figure


1 Tipologia di *e-mail* inviate dall'applicazione . . . . . 6
2 *Barcode* per il test dello scanner . . . . . 7
3 Bottoni nel *footer* . . . . . 10
4 Bottoni nella parte centrale delle pagine . . . . . 10
5 Dimensione vs. posizione dei bottoni . . . . . 11
6 Bottone indietro su *iOS* . . . . . 12
7 Menu dell'applicazione . . . . . 13

# 1 Introduzione

Il progetto consiste nello sviluppo di un'applicazione *mobile*, chiamata *SmartOrder*, progettata per permettere agli utenti registrati di effettuare ordini presso il proprio fornitore. L'applicazione è stata sviluppata esclusivamente per *smartphone*, al momento soltanto per l'ambiente *Android*, utilizzando il *framework cross-platform PhoneGap*. Per l'ambiente *iOS* non è stato possibile realizzare l'applicazione, non essendo il gruppo in possesso di un *Mac*. L'applicazione non prevede una registrazione esplicita, infatti le aziende che intendono acquistare il servizio devono comunicarlo al *team* di *SmartOrder*. Al momento della richiesta devono essere fornite le anagrafiche degli utenti che il fornitore intende registrare e, in questo modo verranno creati gli *account* per tutti gli utenti, compresi di credenziali. Inoltre, devono essere forniti i dati relativi a tutti gli articoli che il fornitore desidera rendere disponibili per gli utenti. Una volta ricevute le credenziali via *e-mail*, gli utenti possono iniziare ad usufruire del servizio. Il tutto sarà perfettamente configurato per fare in modo che ogni utente possa visionare solamente gli articoli venduti dal proprio fornitore. Tramite l'applicazione sarà possibile:

- ricercare prodotti e aggiungerli al carrello;
- scansionare il codice a barre di un prodotto che si ha già acquistato per aggiungerlo al carrello più velocemente;
- modificare o eliminare i prodotti nel carrello;
- inviare un ordine presso il proprio fornitore;
- visualizzare le informazioni relative a tutti gli ordini effettuati presso il proprio fornitore.

L'idea è nata per cercare di soddisfare un bisogno reale. È stato richiesto se fosse possibile velocizzare il processo di rifornimento di libri di una libreria una volta terminati. L'idea è fornire la possibilità di ordinare un libro scansionandone il codice a barre quando esso sta per terminare. La possibilità di aggiungere gli articoli dall'inventario è stata implementata per permettere all'applicazione di essere robusta rispetto all'aggiunta di nuovi clienti, in quanto essi non avendo ancora acquistato prodotti, non hanno la possibilità di scansionare il codice a barre per ordinarli.

L'applicazione realizzata non presenta la gestione dei pagamenti, in quanto si assume che l'utente abbia un contratto con il proprio fornitore, quindi il denaro viene estratto dal conto corrente al momento dell'ordine.

# 2 La scelta del framework

Per lo sviluppo del progetto si è scelto il *framework cross-platform PhoneGap*, in quanto un membro del gruppo aveva già familiarità con le tecnologie offerte dallo stesso e il suo utilizzo non avrebbe richiesto l'apprendimento di nuove tecnologie. Altri *framework* presi in considerazione, come *Xamarin* e *React Native*, avrebbero richiesto l'apprendimento di *C#* e *JSX*, rispettivamente, linguaggi poco conosciuti da entrambi i membri del gruppo e il quale apprendimento non avrebbe permesso di concentrarsi maggiormente sulle funzionalità offerte dall'applicazione. Il *framework* scelto ha permesso di:

- implementare l'interfaccia grafica dell'applicazione una sola volta;
- accedere ad alcune funzionalità native del dispositivo.

# 3 Scelta delle tecnologie web

Siccome è stato scelto il *framework PhoneGap*, il gruppo ha utilizzato i linguaggi *HTML5*, *CSS3* e *JavaScript*. In particolare, *HTML5* è stato scelto perché include un insieme di funzionalità che permettono di valorizzare le interfacce *mobile*. Alcune di queste evidenziano come *HTML5* sia già per sua natura orientato al *mobile*. In particolare, *HTML5* fornisce *API* per la gestione degli eventi *touch*. Mentre i meccanismi di *input* nei PC consistono per lo più nella tastiera e nel *mouse*, nei dispositivi mobili quasi tutto passa per il *touch screen*, e avere funzionalità comode per gestire questo strumento consente un'interazione più ricca e senza limitazioni per l'utente. Le gestualità da attuare su un *display*, nel mondo *mobile*, costituiscono un vero e proprio linguaggio fondamentale nella *user experience*.

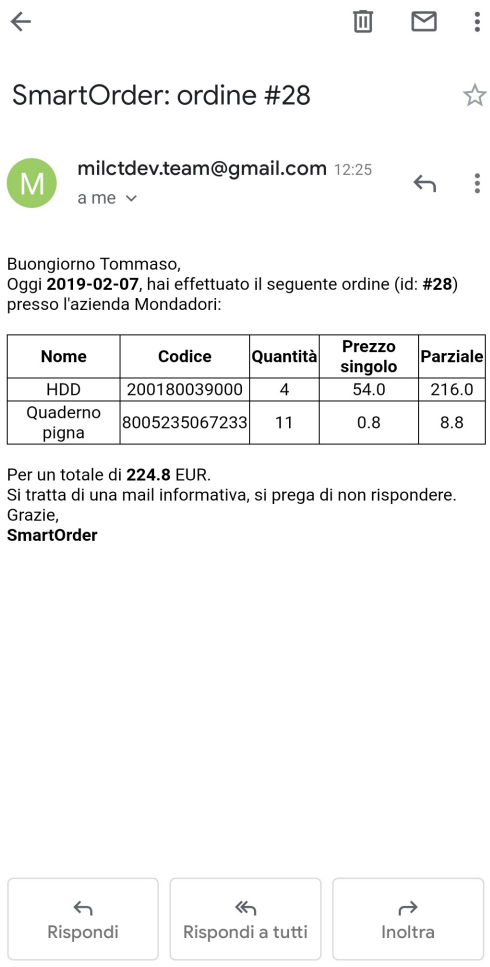
Ciò che ha favorito la scelta di *CSS3* sono state le *media queries*. Esse permettono di definire regole stilistiche in base alla tipologia del mezzo di visualizzazione, delle sue dimensioni e della sua attuale disposizione (*portrait* o *landscape*). Ciò influisce non solo sull'aspetto esteriore degli elementi ma anche sul loro posizionamento e quindi sulla struttura stessa dell'interfaccia.

Per quanto riguarda il linguaggio *JavaScript* si è utilizzato *JavaScript* puro, senza l'utilizzo di *framework* o *JQuery*. Una particolarità del linguaggio, detta *AJAX*, ha reso possibile eseguire chiamate all'*API* del servizio *web* di *SmartOrder*. *AJAX*, acronimo di *Asynchronous JavaScript and XML*, è una tecnica di sviluppo *software* per la realizzazione di applicazioni *web* interattive (*Rich Internet Application*). Lo sviluppo di applicazioni *HTML* con *AJAX* si basa su uno scambio di dati in *background* fra *web browser* e *server*, che consente l'aggiornamento dinamico di una pagina *web* senza esplicito ricaricamento da parte dell'utente.

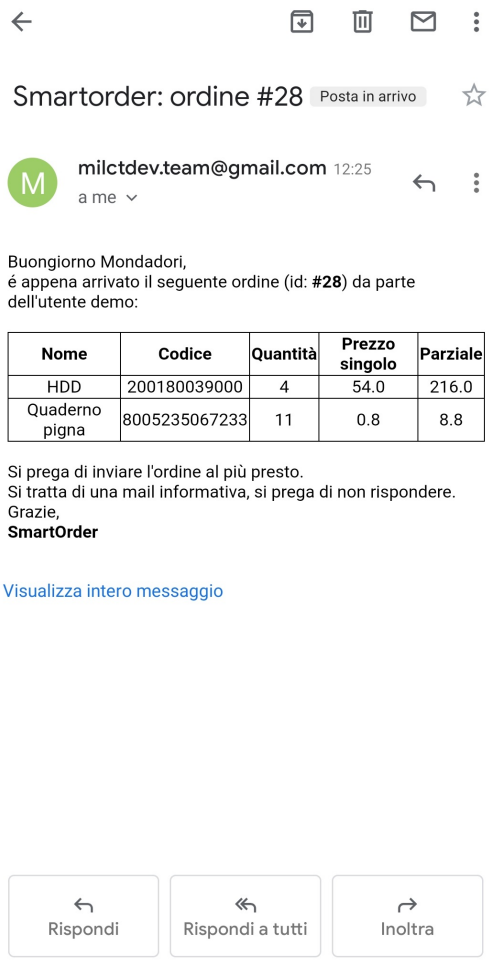
# 4 Informazioni utili all'utilizzo dell'applicazione

Per poter utilizzare l'applicazione è necessario essere connessi ad *Internet*, in quanto essa scarica dati da un *server*. Si fa presente che la connessione viene bloccata dalla rete *Eduroam* e *Studenti.math.unipd.it* dell'Università, quindi si consiglia di utilizzare altre reti. Nel caso in cui l'applicazione venisse aperta in assenza di connessione, verrà visualizzato un messaggio d'errore ed essa verrà successivamente chiusa. Per ricevere le *e-mail* al momento dell'invio dell'ordine, è necessario che il proprio indirizzo *e-mail* sia certificato Amazon[1]. Poiché non sarà possibile quindi ricevere l'*e-mail*, viene di seguito fornito uno *screenshot* della stessa. L'*e-mail* presenta due versioni, a seconda che debba essere inviata all'azienda o all'utente che ha effettuato l'ordine, e contiene un riepilogo dell'ordine effettuato.

[1]certificazione Amazon: i *server SMTP* di Amazon permettono la spedizione gratuita di *e-mail* ai soli indirizzi certificati. Per poter inviare *e-mail* a tutti gli indirizzi è necessario effettuare l'*upgrade* dell'*account*




(a) *Mail* ricevuta dall'utente




(b) *Mail* ricevuta dall'azienda

Figura 1: Tipologia di *e-mail* inviate dall'applicazione

Per testare lo scanner di codici a barre si consiglia di scansionare il seguente codice a barre. Tentando di scansionare altri codici si visualizzerà un messaggio d'errore, in quanto ogni nuovo *barcode* deve essere registrato all'interno del *database*.


Figura 2: *Barcode* per il test dello scanner

Infine si fa notare che il *loader* dell'applicazione indica che *SmartOrder* è in fase di scaricamento dati dal *server* e sta momentaneamente attendendo una risposta per la costruzione della pagina. Il *loader* è stato implementato proprio per fare in modo che l'utente non interagisca con l'applicazione prima della visualizzazione della pagina. Quindi, se si premesse qualche bottone prima dell'avvenuto caricamento, potrebbero accadere problemi di *rendering*.

# 5 Progettazione frontend

Il *frontend* è stato realizzato mediante utilizzo di tecnologie *web*. *HTML* e *CSS* hanno predisposto l'interfaccia grafica, mentre *JavaScript* ha permesso di implementare la logica applicativa. Le pagine realizzate sono le seguenti:

- *index.html*;
- *login.html*;
- *homepage.html*;
- *order.html*;
- *article.html*;
- *articles.html*;
- *newpage.html*;
- *inventory.html*.

Ad ognuna di queste pagine è associato un foglio di stile e un *file Javascript*; inoltre sono presenti un *file CSS* e un *file Javascript* generali. Nelle prossime sezioni vengono descritte le funzionalità di *SmartOrder*.

## 5.1 Login e tutorial

L'applicazione, una volta installata, è utilizzabile. All'avvio della stessa viene mostrata la pagina di *login* per accedere al servizio. In questa schermata devono essere inserite le credenziali fornite dal proprio fornitore in fase di registrazione. La pagina presenta una *checkbox* "Resta collegato" per evitare che ad ogni apertura dell'applicazione venga richiesto il *login*. In seguito al *login*, soltanto al primo avvio, viene mostrato un breve tutorial sul funzionamento dell'applicazione. In fondo al tutorial è presente un bottone che permette di iniziare ad utilizzare l'app. Nel caso in cui un utente si trovasse in difficoltà durante l'utilizzo dell'applicazione, è possibile trovare il tutorial nel menu della stessa. Completato il tutorial si viene reindirizzati al *Carrello*.

## 5.2 Carrello

La pagina *Carrello* è la pagina principale dell'applicazione e visualizza per ogni articolo in carrello la quantità decisa dall'utente, il prezzo e il prezzo parziale (prezzo unitario di un articolo x quantità dell'articolo stesso). Ogni articolo in carrello presenta due bottoni:

- **MODIFICA**: permette di modificare l'articolo. Alla sua pressione si viene reindirizzati alla pagina per la modifica dell'articolo, dove è possibile modificarne la quantità. Una volta effettuata la modifica è necessario confermarla per apportarla definitivamente, premendo sul bottone "Conferma", oppure annullarla, tramite il bottone "Annulla";
- **ELIMINA**: permette di rimuovere l'articolo dal carrello. Se tale bottone viene premuto, verrà chiesta conferma prima di effettuare l'operazione. I motivi di questa scelta sono spiegati in sezione §6.1).

Inoltre, in questa pagina è presente un *footer* contenente:

- il prezzo totale di tutti gli articoli inseriti nel carrello;
- un pannello di bottoni utilizzabili per la gestione del carrello:
  - bottone *Scan*: la spiegazione di questo pulsante è presente nella sezione successiva;
  - bottone *Svuota*: permette di svuotare il carrello. In seguito alla pressione viene chiesta conferma dell'azione;
  - bottone *Invia*: permette di inviare un ordine composto dagli articoli nel carrello. In seguito alla pressione viene chiesta conferma dell'azione. Dopo aver confermato si riceverà un'*e-mail* contenente le informazioni relative all'ordine effettuato.

### 5.2.1 Aggiunta di un articolo al carrello

È possibile aggiungere un articolo al carrello in due modi:

1. tramite scansione del codice a barre in seguito alla pressione del bottone *Scan* nella pagina *Carrello*, accedibile dal menu dell'applicazione;
2. dalla pagina *Inventario*, accedibile tramite il menu dell'applicazione, selezionando l'articolo che si desidera aggiungere e inserendone la quantità nella pagina di aggiunta dell'articolo. Una volta immessa la quantità desiderata, è necessario premere su "Conferma" affinché l'articolo venga inserito nel carrello, oppure su "Annulla" per annullare l'inserimento del nuovo articolo. Nella pagina *Inventario* è inoltre possibile ricercare un articolo, per nome o per codice, digitando la chiave di ricerca nella *textbox* disposta in alto.

La pagina per l'aggiunta di un articolo, che presenta la stessa interfaccia grafica della pagina per la modifica di un articolo, contiene le seguenti informazioni:

- nome dell'articolo: visualizzato in alto alla pagina;
- descrizione dell'articolo: visualizzata subito dopo il nome;
- pannello per l'inserimento della quantità: le modalità di interazione con questo pannello sono spiegate in sezione §6.4;
- prezzo dell'articolo: visualizzato alla fine della pagina. Si tratta di un campo dinamico in quanto varia a seconda della quantità inserita.

**Attenzione**: nel caso in cui si cercasse di aggiungere al carrello un articolo già presente in esso, il sistema mostrerà un messaggio nel quale chiederà se si vuole modificare la quantità dello stesso.

## 5.3 Visualizzazione ordini

Dal menu dell'applicazione è possibile accedere alla lista di tutti gli ordini effettuati premendo sulla voce *Ordini*. Alla sua pressione si verrà reindirizzati alla pagina *Ordini*, dove per ogni ordine, vengono mostrate le seguenti informazioni:

- codice: è il codice che identifica univocamente l'ordine;
- data: è la data in cui è stato effettuato l'ordine;
- totale: è il totale in euro dell'ordine.

La pagina permette di modificare la visualizzazione degli ordini, ordinandoli per data, dal più recente al meno recente, oppure per prezzo crescente. Per selezionare la tipologia di ordinamento è presente una *select* in fondo alla pagina. Tramite il bottone "DETTAGLI", situato sulla destra di ogni ordine, è possibile consultare la lista degli articoli acquistati in quell'ordine. Alla sua pressione si verrà reindirizzati ad una pagina contenente le seguenti informazioni per ogni articolo acquistato:

- nome;
- quantità;
- prezzo;
- prezzo parziale.

## 5.4 Inventario

Dal menu dell'applicazione è possibile accedere alla lista di tutti i prodotti venduti dalla propria azienda, premendo sulla voce *Inventario*. Alla sua pressione si verrà reindirizzati alla pagina *Inventario*, dove per ogni articolo, vengono mostrate le seguenti informazioni:

- codice: è il codice che identifica univocamente l'articolo;
- nome: è il nome dell'articolo;
- prezzo: è il prezzo unitario dell'articolo.

Infine, per ogni articolo, è possibile premere sul bottore "APRI", che permette di visualizzare informazioni dettagliate (descrizione) sullo stesso e di aggiungerlo al carrello. La pagina che viene aperta in seguito alla pressione del bottone è la pagina di aggiunta o modifica di un articolo, di cui si è già parlato.

# 6 Mobile design

In questa sezione vengono descritte le scelte implementative effettuate ai fini di una corretta progettazione dell'interfaccia grafica. Questa sezione è suddivisa in varie sottosezioni e ognuna di esse descrive un elemento dell'interfaccia e le motivazioni alla base della sua progettazione.

La maggior parte delle pagine dell'applicazione sono progettate nel seguente modo:

- la parte superiore presenta il pulsante menu sulla sinistra e il titolo della pagina al centro;
- la parte centrale presenta il contenuto informativo;
- la parte inferiore presenta un *footer* che varia a seconda della pagina visualizzata.

## 6.1 Bottoni

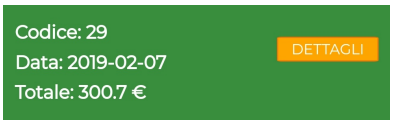

(a) Bottoni presenti nel *footer* della pagina *Carrello*


(b) Bottoni presenti nel *footer* delle pagine *Aggiungi* e *Modifica* articolo

Figura 3: Bottoni nel *footer*

Nelle pagine *Carrello*, *Modifica* e *Aggiungi* sono presenti dei bottoni nella parte bassa dello schermo (Figura 3). L'applicazione richiede una frequente modifica dei dati, quindi i controlli sono stati posizionati in una zona semplice da raggiungere e, seguendo la regola "*Content always on top*", si è preferito inserire questi bottoni in basso, infrangendo così la convenzione che su *Android* i controlli devono essere posizionati nella parte alta dello schermo.


(a) Bottoni nella pagina *Carrello*




(b) Bottoni nella pagina *Ordini*


(c) Bottoni nella pagina *Inventario*

Figura 4: Bottoni nella parte centrale delle pagine

I bottoni presenti in Figura 4 sono situati nella parte centrale dello schermo. Essendo disposti nella *comfort zone*, risultano facilmente raggiungibili e più piccoli rispetto ai bottoni del *footer*, in quanto si è seguita l'immagine in Figura 5. La dimensione è stata opportunamente scelta per fare in modo che l'utente non prema accidentalmente su di essi, in quanto possono cambiare lo stato del carrello oppure dare luogo all'apertura di nuove pagine. Nonostante la dimensione ridotta, sono comunque sufficientemente distanti da evitare *tap* accidentali sui bottoni vicini.

Infine, in caso si premesse sul bottone "ELIMINA" affianco ad un articolo nella pagina *Carrello*, viene chiesta la conferma di eliminazione. Si tratta di un modo per rendere l'operazione reversibile.

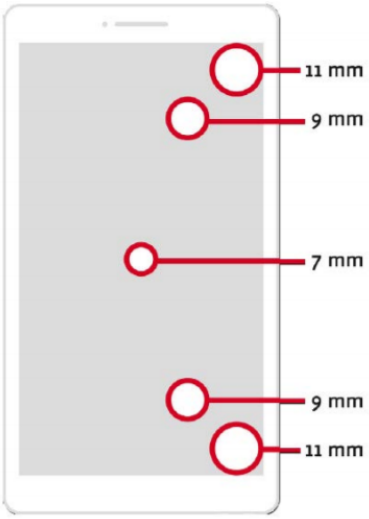



Figura 5: Dimensione vs. posizione dei bottoni

### 6.1.1 Bottone indietro

Poiché negli *smartphone Apple* non è presente il tasto indietro, si è dovuto aggiungere lo stesso nella versione *iOS* di *SmartOrder*. La visualizzazione del bottone viene mostrata nell'immagine sottostante. Per lo stile del bottone e la sua posizione, il gruppo ha preso ispirazione dall'app *mobile* Amazon per *iOS*.


Figura 6: Bottone indietro su *iOS*

## 6.2 Menu


Figura 7: Menu dell'applicazione

Il menu dell'applicazione è rappresentato in Figura 7. Per accederci è presente il classico bottone "*hamburger*" in alto a sinistra di ogni pagina. Il menu è suddiviso in due parti:

- informazioni relative all'utente che ha effettuato il *login* nell'applicazione. In particolare, vengono visualizzati il nome dell'utente e il codice azienda del fornitore dello stesso. Queste informazioni non sono modificabili e pertanto sono state disposte nella parte alta del menu;
- insieme di *link* che permettono di navigare tra le pagine dell'applicazione. In particolare, sono disponibili i *link* al carrello, alla lista degli ordini e all'inventario. Inoltre, è possibile effettuare il *logout* dall'applicazione o visionare nuovamente il tutorial. Questi bottoni sono stati posizionati nella parte centrale del menu (*comfort zone*), in modo da essere facilmente accessibili quando il menu risulta aperto.

## 6.3 Progressive disclosure

In generale, è buona prassi non affollare le interfacce. Al fine di perseguire questo obiettivo si è utilizzato il principio della *progressive disclosure* (interfacce *just-in-time*):

- nella pagina *Ordini* è possibile visualizzare le informazioni essenziali per poter identificare un ordine. Per accedere ai dettagli risulta necessario un ulteriore *tap*;
- nella pagina *Inventario* è possibile visualizzare le informazioni essenziali per poter identificare un articolo. Per accedere ai dettagli dello stesso, o per aggiungerlo al carrello, risulta necessario un ulteriore *tap*.

## 6.4 Uso della tastiera

L'utilizzo della tastiera è stato limitato a tre casi:

- inserimento delle credenziali in fase di *login*;
- ricerca di un articolo;
- inserimento o modifica della quantità nelle pagine *Aggiungi* e *Modifica*, rispettivamente. In questo caso la tastiera permette un veloce inserimento di grandi quantità.

Per l'inserimento della quantità è stata predisposta una seconda modalità. Sono stati messi a disposizione due pulsanti (+/-), i quali permettono di aumentare o diminuire la quantità, rispettivamente. Tramite questa modalità è possibile inserire velocemente piccole quantità.

## 6.5 Aiuti per l'utente

Al primo accesso di un utente all'applicazione, viene visualizzato il tutorial della stessa. Esso fornisce istruzioni rilevanti per l'utilizzo dell'applicazione e risulta breve, in quanto l'applicazione è stata progettata per essere massimamente intuitiva. Alla fine del tutorial è presente un bottone che permette di iniziare ad utilizzare l'applicazione. Il tutorial sarà comunque disponibile dal menu, nel caso in cui l'utente non ricordasse come utilizzare l'app.

## 6.6 Elementi nativi

Gli elementi nativi dell'applicazione sono la fotocamera, per la scansione del codice a barre, e i *dialog*, per la visualizzazione di messaggi informativi o di conferma. Inoltre, si è dovuta monitorare la rete del dispositivo. Per rendere questi elementi nativi si sono utilizzati i seguenti *plugin* di *PhoneGap*:

- *dialogs*: fornisce un'interfaccia per convertire gli *alert* e i *confirm* delle pagine web in *dialog box* nativi. In questo modo è stato possibile customizzare i titoli e le etichette dei pulsanti nei *dialog*;
- *barcodescanner*: fornisce un'interfaccia per accedere nativamente alla fotocamera del dispositivo ed effettuare la scansione di un codice a barre;
- *network information*: fornisce un'interfaccia per accedere alle informazioni relative alle connessioni *Wi-Fi* e cellulare del dispositivo.

## 6.7 Gesture

Per l'applicazione è stata implementata una sola tipologia di *gesture*, ossia lo *scroll* per scorrere all'interno delle varie pagine. Questa *gesture* è presente nella pagine *Carrello*, *Ordini*, *Inventario* e per la descrizione nella pagina di aggiunta o modifica di un articolo in carrello. Il gruppo non è riuscito ad implementare la *gesture swipe right* per l'apertura del menu. Quest'ultima risulta difficilmente realizzabile tramite il solo utilizzo di *JavaScript* puro.

## 6.8 Considerazioni finali

Si ritiene che l'applicazione soddisfi i primi tre livelli della piramide di *Maslow* rimappata sui bisogni degli utenti, ovvero che sia funzionale, affidabile e usabile.

# 7 Backend

Per l'implementazione del *backend* si è deciso di appoggiarsi ad *Amazon Web Services (AWS)*, in quanto tramite l'*account* universitario è possibile usufruire dei servizi offerti gratuitamente.

## 7.1 Architettura

Il *backend* della piattaforma presenta la seguente architettura:

- un *server cloud AWS EC2* configurato con *Windows Server 2017*;
- un'istanza *AWS RDS* contenente un *database Microsoft SQL Server*.

All'interno dell'istanza *AWS EC2* sono stati installati:

- un *server web Apache Tomcat*: esso permette di eseguire un *web service* scritto in linguaggio *Java*, il quale si occupa di captare le richieste *HTTP* che provengono dall'applicazione, di elaborarle e di rispondere ad esse tramite stringhe *JSON* costruite sulla base di *query* effettuate su *database*;
- il linguaggio *Java*: necessario per il funzionamento di *Apache Tomcat* e per l'esecuzione del *web service*.

## 7.2 Web service

Per lo sviluppo del *web service* si sono utilizzati gli oggetti *servlet Java*. Essi permettono di captare richieste *HTTP* e di rispondere ad esse tramite stringhe in formato *JSON*. Il servizio web è costituito dai seguenti *package*:

- *dbConnection*: *package* contenente una classe utilizzabile per connettersi ed interagire con un *database SQL Server*;
- *servlet*: *package* contenente le classi *servlet* del servizio web;
- *utility*: *package* contenente le classi utilità del servizio web.

### 7.2.1 Servlet

In questa sezione vengono descritte le classi *servlet* e per ognuna di esse vengono indicate le seguenti caratteristiche:

- **End-point**: *end-point* che identifica il *servlet*. L'*end-point* è utilizzabile per comunicare con lo specifico *servlet* all'interno del *web service*;
- **Parametri**: lista dei parametri che devono essere inseriti nella richiesta *HTTP* al fine di comunicare correttamente con il *servlet*;
- **Funzionalità**: una breve descrizione delle operazioni eseguite dal *servlet* e della tipologia di risposta che restituisce.

1. **AggiuntaModificaArticolo.java**
    - **End-point**: */AggiuntaModificaArticolo*
    - **Parametri**:

  - codice azienda: è il codice che identifica univocamente il *database* del fornitore corrispondente all'utente loggato;
  - *query*: è la *query* di inserimento o modifica di un articolo nel carrello dell'utente loggato.
- **Funzionalità**: il *servlet* si connette al database identificato dal codice azienda fornito ed effettua la *query* fornita su di esso. Se la *query* va a buon fine restituisce una stringa *JSON* contenente codice 1 (operazione andata a buon fine), altrimenti codice 0 (operazione fallita).

2. **Autenticazione.java**

- **End-point**: */Autenticazione*
- **Parametri**:
  - *username*: è la *username* che l'utente ha inserito in fase di *login*;
  - *password*: è la *password* che l'utente ha inserito in fase di *login*.
- **Funzionalità**: il *servlet* si connette al *database* ideato per la gestione dell'autenticazione e cerca all'interno di esso le credenziali fornite. Se le credenziali vengono identificate viene restituita una stringa *JSON* contenente codice 0, altrimenti ci possono essere le seguenti possibilità:
  - codice 1: la *password* inserita dall'utente non è quella corretta;
  - codice 2: la *username* inserita dall'utente è inesistente.

3. **EliminazioneArticoli.java**

- **End-point**: */EliminazioneArticoli*
- **Parametri**:
  - *username*: è la *username* dell'utente loggato;
  - codice azienda: è il codice che identifica univocamente il *database* del fornitore corrispondente all'utente loggato;
  - lista codici: è la lista dei codici a barre dei prodotti che devono essere rimossi dal carrello dell'utente loggato.
- **Funzionalità**: il *servlet* si connette al *database* identificato dal codice azienda fornito ed elimina dalla tabella *contenutoCarrelli* gli articoli corrispondenti ai codici e alla *username* forniti. Infine restituisce una stringa *JSON* contenente codice 1 se l'eliminazione è andata a buon fine, altrimenti codice 0.

4. **InvioOrdine.java**

- **End-point**: */InvioOrdine*
- **Parametri**:
  - *username*: è la *username* dell'utente loggato;
  - codice azienda: è il codice che identifica univocamente il *database* del fornitore corrispondente all'utente loggato;
  - totale: è il totale dell'ordine che l'utente ha deciso di inviare.

- **Funzionalità**: il *servlet* si connette al *database* identificato dal codice azienda fornito e crea in *database* un nuovo ordine contenente gli articoli presenti nel carrello dell'utente identificato dalla *username* fornita. Infine invia un'*e-mail* di notifica dell'ordine sia all'utente identificato dalla *username* fornita che all'azienda identificata dal codice azienda fornito.

5. **PrelevaArticoliAzienda.java**
   - **End-point**: */PrelevaArticoliAzienda*
   - **Parametri**:
     - codice azienda: è il codice che identifica univocamente il *database* del fornitore corrispondente all'utente loggato;
   - **Funzionalità**: il *servlet* si connette al *database* identificato dal codice azienda fornito e restituisce una stringa *JSON* contenente le informazioni sugli articoli venduti dall'azienda.
6. **PrelevaDatiOrdine.java**
   - **End-point**: */PrelevaDatiOrdine*
   - **Parametri**:
     - codice azienda: è il codice che identifica univocamente il *database* del fornitore corrispondente all'utente loggato;
     - codice ordine: è il codice dell'ordine di cui si vogliono conoscere i dettagli.
   - **Funzionalità**: il *servlet* si connette al database identificato dal codice azienda fornito e restituisce una stringa *JSON* contenente le informazioni relative all'ordine corrispondente al codice ordine fornito.
7. **PrelevaOrdini.java**
   - **End-point**: */PrelevaOrdini*
   - **Parametri**:
     - *username*: è la *username* dell'utente loggato;
     - codice azienda: è il codice che identifica univocamente il *database* del fornitore corrispondente all'utente loggato;
   - **Funzionalità**: il *servlet* si connette al *database* identificato dal codice azienda fornito e restituisce una stringa *JSON* contenente le informazioni relative agli ordini effettuati dall'utente corrispondente alla *username* fornita.
8. **PrelievoInfoArticoli.java**
   - **End-point**: */PrelievoInfoArticoli*
   - **Parametri**:
     - *username*: è la *username* dell'utente loggato;
     - codice azienda: è il codice che identifica univocamente il *database* del fornitore corrispondente all'utente loggato;
   - **Funzionalità**: il *servlet* si connette al *database* identificato dal codice azienda fornito e restituisce una stringa *JSON* contenente le informazioni relative agli articoli nel carrello dell'utente identificato dalla *username* fornita.

9. **PrelievoInfoArticolo.java**

   - **End-point**: */PrelievoInfoArticolo*
   - **Parametri**:
     - codice articolo: è il codice dell'articolo di cui si vogliono conoscere i dettagli;
     - codice azienda: è il codice che identifica univocamente il *database* del fornitore corrispondente all'utente loggato;
   - **Funzionalità**: il *servlet* si connette al *database* identificato dal codice azienda fornito e restituisce una stringa *JSON* contenente le informazioni relative all'articolo corrispondente al codice articolo fornito.

### 7.2.2 Utility

Il *package utility* contiene le seguenti classi utilità:

- ***GetDb***: classe contenente un metodo statico che restituisce la stringa di connessione dal *database* di autenticazione;
- ***SendMail***: classe che fornisce un'interfaccia per l'invio di *e-mail* tramite un *server SMTP* di *AWS*.

## 7.3 Database

Per la gestione di ordini, utenti e articoli venduti dai vari fornitori sono stati utilizzati dei *database Microsoft SQL Server*. In questa sezione vengono descritti i seguenti *database*:

- *database* di autenticazione;
- *database* aziendale.

### 7.3.1 Database per l'autenticazione

Il *database* per l'autenticazione contiene le credenziali e le anagrafiche degli utenti registrati e le informazioni dei fornitori registrati all'applicazione. Questo *database* contiene le seguenti tabelle:

- aziende: contiene le informazioni delle aziende registrate a *SmartOrder*. La tabella contiene i seguenti attributi:
  - codAzienda (chiave primaria): è un codice che identifica univocamente l'azienda. Esso viene fornito all'azienda in fase di registrazione all'applicazione;
  - nome: è il nome dell'azienda;
  - via: è la via dove è situata la sede principale dell'azienda;
  - civico: è il civico dove è situtata la sede principale dell'azienda;
  - comune: è il comune dove è situata la sede principale dell'azienda;
  - provincia: è la provincia dove è situata la sede principale dell'azienda;
  - CAP: è il codice postale dove è situata la sede principale dell'azienda;
  - *mail*: è la *mail* che l'azienda ha fornito in fase di registrazione;
  - telefono: è il numero di telefono che l'azienda ha fornito in fase di registrazione.

- *users*: contiene le informazioni relative agli utenti registrati a *SmartOrder*. La tabella contiene i seguenti attributi:
  - *username* (chiave primaria): è il nome utente che è stato fornito all'utente in fase di registrazione;
  - *password*: è la *password* che è stata fornita all'utente in fase di registrazione;
  - *mail*: è la *mail* che l'utente ha fornito in fase di registrazione;
  - codAzienda: è il codice dell'azienda presso cui l'utente è cliente;
  - cellulare: è il numero di cellulare che l'utente ha fornito in fase di registrazione;
  - nome: è il nome dell'utente;
  - cognome: è il cognome dell'utente;
  - via: è la via di residenza dell'utente;
  - civico: è il civico di residenza dell'utente;
  - comune: è il comune di residenza dell'utente;
  - provincia: è la provincia di residenza dell'utente;
  - CAP: è il codice postale di residenza dell'utente.

### 7.3.2 Database aziendale

Il *database* aziendale contiene le informazioni relative agli articoli venduti e agli ordini effettuati presso uno specifico fornitore. Per cui, nel *server* è presente un *database* di questa tipologia per ogni fornitore registrato all'applicazione. Questo *database* contiene le seguenti tabelle:

- articoli: contiene le informazioni relative agli articoli venduti dall'azienda. La tabella contiene i seguenti attributi:
  - nome: è il nome dell'articolo;
  - *barCode* (chiave primaria): è il codice a barre dell'articolo;
  - prezzo: è il prezzo dell'articolo;
  - descrizione: è la descrizione dell'articolo.
- contenutoCarrelli: contiene le informazioni relative al contenuto dei carrelli degli utenti dell'azienda. La tabella contiene i seguenti attributi:
  - idRiga (chiave primaria): è un identificativo autoincrementante che funge unicamente da chiave primaria;
  - *barCode*: è il codice a barre dell'articolo in carrello;
  - quantità: è la quantità scelta dall'utente per l'articolo in carrello;
  - *username*: è la *username* dell'utente che ha inserito in carrello l'articolo.
- contenutoOrdini: contiene le informazioni relative al contenuto degli ordini effettuati presso l'azienda. La tabella contiene i seguenti campi:
  - idRiga (chiave primaria): è un identificativo autoincrementante che funge unicamente da chiave primaria;
  - codiceOrdine: è il codice dell'ordine in cui é presente l'articolo;

  - *barCode*: è il codice a barre dell'articolo all'interno dell'ordine;
  - quantità: è la quantità ordinata per l'articolo.

- ordini: contiene le informazioni relative agli ordini effettuati presso l'azienda. La tabella contiene i seguenti campi:
  - codiceOrdine (chiave primaria): è il codice che identifica univocamente l'ordine presso l'azienda;
  - data: è la data in cui l'ordine è stato effettuato;
  - *username*: è la *username* dell'utente che ha effettuato l'ordine;
  - totale: è il totale in euro dell'ordine.

# 8 Conclusioni

Vi è la consapevolezza che l'interfaccia grafica possa essere resa più accattivante. L'implementazione della stessa poteva risultare più semplice tramite l'utilizzo di un altro *framework*, come ad esempio *React Native*. Il *framework* scelto ha comunque permesso di realizzare un'interfaccia usabile in tempi ristretti. Il gruppo ritiene di aver realizzato una buona applicazione, che potrebbe risultare utile ad aziende ed utenti nel contesto per cui è stata progettata e implementata. Tra gli sviluppi futuri vi sono:

1. utilizzo di icone personalizzate;
2. utilizzo di una *palette* di colori definita a priori;
3. implementazione di *gesture swipe right* e *swipe left* per l'eliminazione di articoli dal carrello;
4. implementazione della *gesture long press* per la modifica di un articolo in carrello;
5. implementazione della *gesture swipe right* per l'apertura del menu.