



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

# Sviluppo di un'applicazione per la gestione di un B2B

RELATORE

Prof. **Roberto De Prisco**

Università degli studi di Salerno

CANDIDATO

**Raffaele Sais**

Matricola: 0512107082

Anno Accademico 2021-2022

*Il computer è la bicicletta della nostra mente.*

## Sommario

Esiste un divario molto ampio tra l'utilizzo delle tecnologie nel B2B e il B2C di almeno 10 anni. In particolare, i B2B si servono di tecnologie obsolete, poco adatte al giorno d'oggi a tal punto che le aziende riscontrano diversi problemi, quali: organizzazione, rapporto con la clientela, comunicazione rapida e problemi concerni sia la presentazione di interfaccia grafica che l'esperienza dell'utente. L'applicazione si pone come obiettivo quello di mostrare alcuni miglioramenti nel mondo del B2B, ottenuti mediante l'utilizzo delle nuove tecnologie informatiche, colmando il gap tra il B2C e il B2B.

Le funzionalità sviluppate riguardano la UI/UX, la corretta e veloce gestione degli ordini, l'organizzazione della scaffalatura, la fidelizzazione degli utenti e l'utilizzo di ML per la ricerca dei prodotti. Tali miglioramenti giovano al contempo sia l'utente finale, che il titolare dell'azienda, garantendo al cliente un'esperienza confortevole ed al proprietario un'efficace gestione, riducendo le spese e massimizzando la forza lavoro.

<b>Indice</b>	<b>ii</b>
<b>1 Il B2B</b>	<b>1</b>
1.1 Concetti generali . . . . .	1
1.2 I problemi . . . . .	2
1.3 Obiettivo della tesi . . . . .	2
<b>2 Il progetto</b>	<b>3</b>
2.1 Architettura del sistema . . . . .	3
2.1.1 Server . . . . .	3
2.1.2 Applicazione mobile . . . . .	4
2.2 Analisi delle funzionalità . . . . .	4
<b>3 Sviluppo progetto</b>	<b>5</b>
3.1 I Linguaggi . . . . .	5
3.1.1 NodeJS . . . . .	5
3.1.2 Kotlin . . . . .	6
3.2 Programmazione nativa . . . . .	6
3.3 Funzionalità principali . . . . .	7
3.3.1 Registrazione . . . . .	7
3.3.2 Ricerca prodotti . . . . .	7
3.3.3 Creazione ordine . . . . .	7
3.3.4 Evasione ordini . . . . .	7

3.3.5	Visualizza statistiche . . . . .	8
3.3.6	Prevendita . . . . .	8
3.3.7	Quotazione prodotti . . . . .	8
3.4	Progettazione dei dati . . . . .	8
<b>4</b>	<b>Implementazione soluzione</b>	<b>11</b>
4.1	Architettura . . . . .	11
4.2	Librerie . . . . .	12
4.2.1	Libreria Volley . . . . .	12
4.2.2	Libreria Lottie Animation . . . . .	14
4.2.3	Libreria ML Firebase . . . . .	14
4.2.4	Libreria Gson . . . . .	15
4.2.5	Libreria OneSignal . . . . .	16
4.3	Git . . . . .	16
4.4	Gestione dei permessi . . . . .	16
4.5	Funzionalità principali realizzate . . . . .	18
4.5.1	Registrazione . . . . .	18
4.5.2	Ricerca prodotti . . . . .	19
4.5.3	Creazione ordine . . . . .	20
4.5.4	Evasione ordini . . . . .	21
4.5.5	Visualizza statistiche . . . . .	22
4.5.6	Supporto multilingua . . . . .	23
4.5.7	Assistenza Whatsapp . . . . .	24
<b>5</b>	<b>Prospettive future</b>	<b>26</b>
5.1	Trello . . . . .	26
5.2	Funzioni da implementare . . . . .	27
5.2.1	Gestione prevendite . . . . .	27
5.2.2	Gestione quotazioni . . . . .	28
5.2.3	Ordine senza fila . . . . .	28
5.2.4	Integrazioni pagamenti . . . . .	28
5.2.5	Lingua cinese . . . . .	28
5.2.6	Gestione valute . . . . .	29
5.3	Espansione target . . . . .	29
5.3.1	Applicazione iOS . . . . .	29

5.3.2	E-Commerce . . . . .	29
5.3.3	API pubbliche . . . . .	30
<b>6</b>	<b>Conclusioni</b>	<b>31</b>
	<b>Bibliografia</b>	<b>33</b>

### 1.1 Concetti generali

Il B2B rappresenta l'acronimo dell'espressione "Business To Business" ove sia il venditore che l'acquirente risultano essere delle aziende, al contrario del B2C (Business To Consumer) dove l'acquirente è il cliente finale. Tali mercati presentano delle analogie, ma soprattutto delle differenze.

In Italia il valore delle transazioni annue generate dal B2B è di circa 2.700 miliardi di euro<sup>1</sup> mentre quello mondiale è di oltre 7 trilioni di dollari.

Si possono individuare tre tipologie di vendita con cui l'azienda propone i suoi prodotti sul mercato<sup>2</sup>:

- **Canale diretto:** l'azienda dispone di punti vendita e non presenta intermediari. Questo permette di ridurre i costi ma anche di diminuire il fatturato (tipicamente queste aziende sono dei Cash & Carry);
- **Canale breve:** l'azienda ha un intermediario che detiene il contatto diretto con i dettaglianti (ad esempio grandi aziende che vendono prodotti proprietari);

---

<sup>1</sup><https://www.economyup.it>

<sup>2</sup>[https://it.wikipedia.org/wiki/Distribuzione\\_commerciale](https://it.wikipedia.org/wiki/Distribuzione_commerciale)

- **Canale lungo:** l'azienda ha due o più intermediari, il che determina un aumento generale dei prezzi. La vendita attraverso tale tipologia viene effettuata direttamente al fornitore e non al dettagliante.

## 1.2 I problemi

Durante gli ultimi anni, con il passaggio di proprietà dei punti vendita al dettaglio, da persone di generazione Z (nati dal 1965 al 1979) a persone di generazione Y e oltre (nati dal 1980 in poi)<sup>3</sup>, il mondo del commercio B2C si è aggiornato costantemente, predisponendo la creazione di e-commerce e gestionali per l'inventario e la realizzazione di un sistema efficace di spedizioni<sup>4</sup>. Al contrario il B2B non è riuscito a compiere un processo di aggiornamento e miglioramento dello sviluppo tecnologico, risultando antiquato e non pienamente efficiente. Oggigiorno, circa il 20% dei B2B adopera ancora l'utilizzo di Excel per l'effettuazione degli ordini, mentre circa l'80% dei B2B possiede un e-commerce<sup>5</sup>. Come si può notare, il problema principale è che il B2B non è al passo coi tempi e spesso presenta una UI datata e, di conseguenza, una UX poco efficiente. Oltre ai clienti, molto spesso anche i dipendenti e gli stessi soci aziendali rilevano delle difficoltà nello svolgimento dei task. I dipendenti riscontrano elevate difficoltà nei controlli per i carichi di magazzino e/o per la gestione degli ordini da parte di clienti, mentre i soci non riescono ad avere le giuste informazioni quali fatturato, guadagno, margine, ricarico ecc...

## 1.3 Obiettivo della tesi

La tesi si pone come finalità la realizzazione di un'applicazione Android per risolvere i problemi sopra elencati. Grazie alla cooperazione con l'azienda New Net SRL, presso la quale ho svolto un'attività di tirocinio, e sulla base di alcuni progetti, idee e miglioramenti (data la mia esperienza lavorativa nel mercato del B2B) abbiamo individuato alcune funzionalità che riducano il divario tra il mondo del B2C e il mondo B2B, permettendo di migliorare la forza lavoro e l'esperienza utente. Pertanto, l'applicazione sarà fruibile non solo dal cliente e dai dipendenti, ma anche dai responsabili aziendali, consentendo la raccolta di dati precisi, accurati e in tempo reale anche da remoto.

---

<sup>3</sup><https://www.argoserv.it/generazione-x-y-z-c>

<sup>4</sup><https://www.nextre.it/e-commerce-b2c-vs-b2b>

<sup>5</sup><https://www.bigcommerce.it/articoli/e-commerce-b2b/tendenze-e-commerce-b2b/>



## 2.1 Architettura del sistema

L'applicazione che New Net SRL ha in mente di sviluppare richiede l'utilizzo di un server per la gestione di una base di dati e delle API necessarie, e lo sviluppo di un'app mobile per usufruire dei servizi. Quest'ultima può essere scaricata su qualsiasi cellulare (al momento solo con SO Android) oppure su qualche terminale portatile noleggiato presso New Net. Sono state individuate due diverse tipologie di offerte per il cliente:

- **Configurazione:** viene installato il server per la gestione del magazzino, mentre l'applicazione viene fornita nei principali mercati di app (PlayStore, App Store e Xiaomi Store);
- **Configurazione + Noleggio:** viene effettuata l'installazione del server insieme al deploy nei principali mercati di app e vengono consegnati anche dei terminalini aziendali per la gestione degli ordini e dei carichi di magazzino.

### 2.1.1 Server

Il server gestisce sia le richieste che arriveranno dai vari client (in questo caso l'applicazione android) sia la base di dati. Il server avrà come componenti un servizio REST API suddiviso in delle macro aree, dove ogni area fa riferimento a funzionalità chiavi per il servizio. Le API sfruttano un modello di autenticazione basato su *token* per identificare

univocamente l'utente che effettua la richiesta. Tale funzione permette anche di avere un file di log completo, riportando le azioni svolte da un utente, seppur queste ultime non rientrino nei suoi permessi. Lo scambio di informazioni (sia richiesta che risposta) viene effettuato nel formato JSON<sup>1</sup> dove ogni risposta seguirà una nomenclatura precisa per far sì che il client che ha effettuato la richiesta sia capace di comprendere se ci siano stati errori o meno durante l'elaborazione della richiesta.

### 2.1.2 Applicazione mobile

Si è pensato di unire tutte le funzionalità (cliente, dipendente e soci) in un'unica applicazione, la quale, in base al tipo di utente, mostrerà solo le operazioni che ne competono. La suddetta app inizialmente è prevista solo per Android anche se l'obiettivo a lungo termine è quello di renderla disponibile per dispositivi iOS, iPadOS e un portale web.

## 2.2 Analisi delle funzionalità

L'identificazione di tutti i possibili fruitori dell'applicativo e dei contesti di utilizzo è stata semplificata e facilitata dalla rispettiva conoscenza dei B2B e dei B2C.

- **Dipendenti:** utilizzano il servizio per visualizzare informazioni sui prodotti e per evadere gli ordini. Sono interessati ad avere la lista degli ordini non evasi per prenderli in carico e alla visualizzazione degli articoli per fornire informazioni ai clienti;
- **Clienti:** utilizzano il servizio per visualizzare i prodotti ed eventuali offerte, effettuare ordini e richiedere anche delle quotazioni.
- **Soci:** utilizzano il servizio per visualizzare informazioni e/o documenti, carichi, ordini e statistiche. Quindi, sono interessati a visualizzare i carichi effettuati e i carichi in arrivo, gli ordini dei clienti e il loro stato di elaborazione, ma soprattutto le statistiche per analizzare l'andamento aziendale e della forza lavoro.

---

<sup>1</sup>[https://it.wikipedia.org/wiki/JavaScript\\_Object\\_Notation](https://it.wikipedia.org/wiki/JavaScript_Object_Notation)

## 3.1 I Linguaggi

In questo paragrafo vengono descritti i linguaggi utilizzati per lo sviluppo del progetto.

### 3.1.1 NodeJS

NodeJS rappresenta uno dei framework di JavaScript più utilizzati e innovativi degli ultimi tempi, in quanto permette di utilizzare JavaScript non solo lato client ma anche lato server.

Il framework si basa sulla modalità *event-driven*, dove ogni evento corrisponde ad una reazione/azione, garantendo una certa efficienza delle applicazioni utilizzando un sistema di *callback*.

NodeJS può essere utilizzato per la realizzazione di backend, applicazioni desktop, gestione del SO e la gestione della domotica. Inoltre è presente un importante gestore di pacchetti, chiamato **npm** che viene installato insieme a NodeJS che presenta numerose librerie da utilizzare per semplificare il lavoro.

Infine, è giusto specificare che NodeJS non è un linguaggio di programmazione, bensì un *runtime environment* per l'esecuzione di JavaScript al di fuori del browser. Riscontriamo tra le caratteristiche principali di Node sicuramente la sua leggerezza; questo permette il suo

utilizzo anche su dispositivi poco potenti e con poca memoria (per questo viene utilizzato molto nel mondo della domotica e del IOT<sup>1</sup>)

### 3.1.2 Kotlin

Kotlin è un linguaggio di programmazione creato da JetBrains focalizzato all'interoperabilità di Java che permette di realizzare applicazioni in Android e in JVM<sup>2</sup>. Vista la stretta correlazione tra i due linguaggi spesso è possibile far coesistere entrambi all'interno di un singolo progetto, oppure convertire un intero file Kotlin in Java e viceversa.

Divenuto famoso grazie alla sua facilità d'uso e alle sue notevoli performance, il 17 maggio del 2017, Google ha annunciato pieno supporto al linguaggio Kotlin, con la possibilità di rilasciare sul PlayStore applicazioni sviluppate con esso.

Kotlin, così come Java, è un linguaggio a tipizzazione forte, ovvero che per ogni variabile si deve specificare il suo tipo. Una delle funzionalità più importanti di Kotlin è sicuramente la sicurezza della sua architettura, eliminando alcuni errori di Java 'noiosi' come ad esempio il **NullPointerException**, visto che per impostazione predefinita nessun oggetto o classe è annullabile<sup>3</sup>.

## 3.2 Programmazione nativa

Durante le fasi iniziali del progetto, la scelta del linguaggio di programmazione da utilizzare ricadeva su due alternative: linguaggio di programmazione nativo o ibrido. I linguaggi nativi sono i linguaggi standard per quel sistema operativo, ossia Java/Kotlin per Android e Swift per iOS mentre quelli ibridi, scrivendo il codice una sola volta, permettono di avere l'applicazione sia in Android che in iOS. Si è deciso di utilizzare i linguaggi nativi (in questo caso Kotlin) dati i numerosi vantaggi circa le performance e la fruibilità.

Nonostante i vantaggi citati è opportuno segnalare lo svantaggio di dover scrivere due volte l'applicazione, sia per i sistemi Android che per quelli iOS. Per quanto concerne i linguaggi ibridi oggi offrono molte funzionalità anche se è netta la differenza di prestazione confrontandola con una app nativa. Pertanto, questa scelta è stata valutata in base agli obiettivi che l'applicazione deve rispettare.

---

<sup>1</sup>Internet of Things

<sup>2</sup>Java Virtual Machine

<sup>3</sup><https://www.nextre.it/java-o-kotlin/>

### **3.3 Funzionalità principali**

In questo paragrafo si definiscono le funzionalità più importanti del sistema e come l'utente interagisce con l'applicazione.

#### **3.3.1 Registrazione**

Qualsiasi utente per utilizzare la piattaforma necessita della registrazione, durante la quale vengono richiesti i dati aziendali che verranno inviati in modo sicuro al server. Nel caso l'esito della registrazione risulti essere positivo, il sistema effettua l'accesso con l'account appena creato e porterà l'utente nella schermata principale dell'applicazione.

#### **3.3.2 Ricerca prodotti**

L'utente può visualizzare tutti i prodotti disponibili e filtrarli in base alla descrizione, alla categoria oppure ordinarli in base a dei parametri quale il prezzo dell'articolo, che verrà visualizzato solo dagli utenti che hanno effettuato l'accesso. Si ha inoltre la possibilità di visualizzare le informazioni dell'articolo e la possibilità di aggiungerlo al carrello cliccando semplicemente sul prodotto.

#### **3.3.3 Creazione ordine**

Una volta inseriti gli articoli nel carrello, l'utente ha la possibilità di visualizzare i prodotti presenti nella pagina e confermare la scelta. Successivamente, il server creerà l'ordine associato. Dopo aver compiuto tale operazione i dipendenti riceveranno una notifica di un nuovo ordine, mentre l'utente riceverà un riepilogo dell'ordine.

#### **3.3.4 Evasione ordini**

Quando viene creato un nuovo ordine, i dipendenti lo visualizzano nella loro schermata principale ed una volta selezionato si procederà con l'evasione dell'ordine che può avvenire anche in più fasi ed essere effettuata da più dipendenti. Durante questa funzione verranno salvate tutte le operazioni effettuate dal dipendente (lettura prodotto errata, corretta ecc...) affinché i soci possano visualizzare le statistiche. Infine, il cliente che ha effettuato l'ordine riceverà una notifica per ogni cambiamento di stato dell'ordine (evaso o annullato).

### 3.3.5 Visualizza statistiche

Un socio (admin) può visualizzare tutte le statistiche dei dipendenti sia in modo generale sia quelle inerenti le letture ed errori dei singoli dipendenti.

### 3.3.6 Prevendita

L'utente può visualizzare tutti i prodotti in arrivo nel magazzino con la possibilità di scegliere la quantità di merce che vuole ordinare e confermare. Questo permette di agevolare il cliente attraverso la creazione di un ordine nel momento in cui il carico arriva in sede, acquistandolo molto spesso ad un prezzo inferiore.

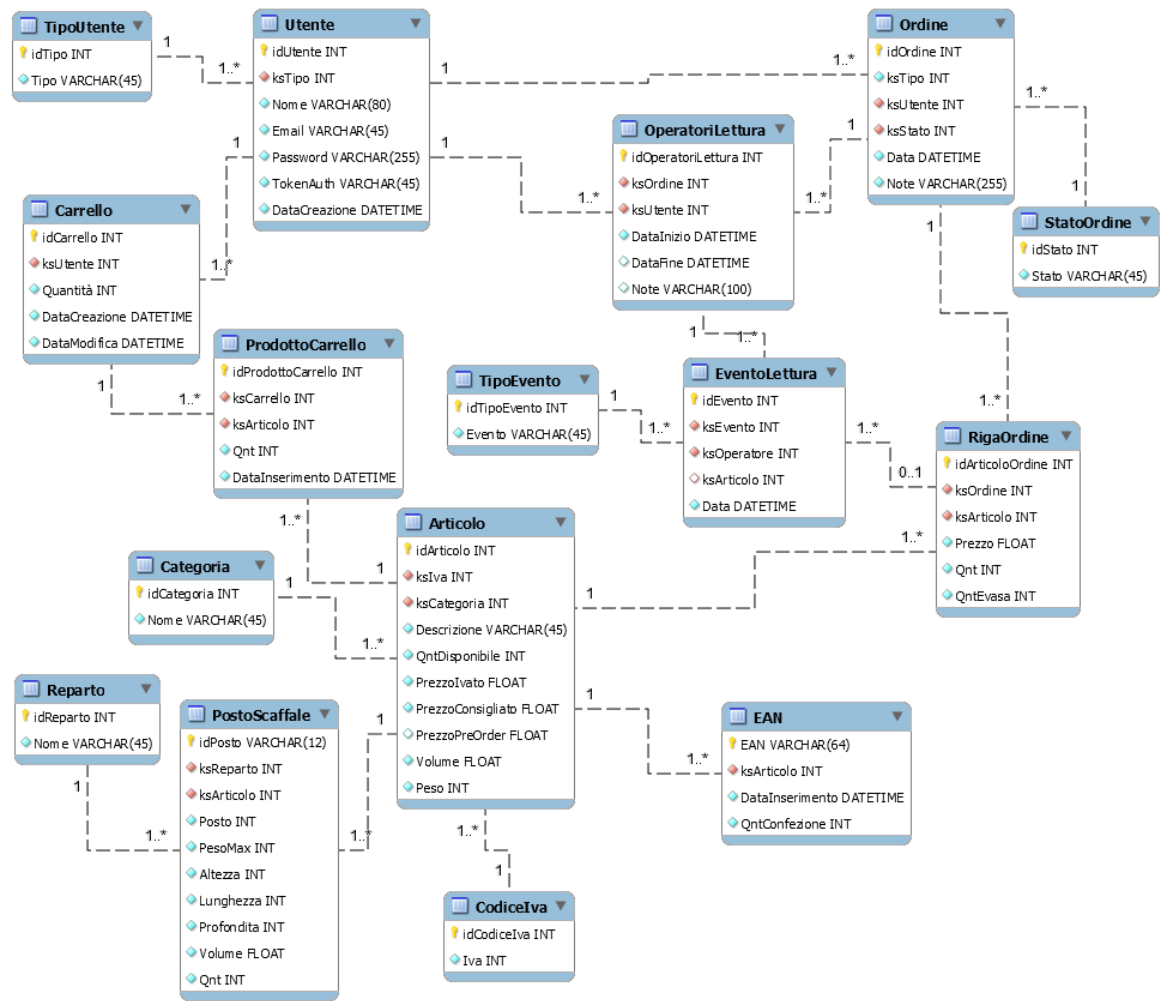
### 3.3.7 Quotazione prodotti

L'utente può visualizzare tutti i prodotti preordinabili e richiedere una quotazione. Dopo aver selezionato i prodotti e averli aggiunti al carrello, i termini di pagamento prevedono un versamento del 30% al momento della richiesta e il restante 70% nel momento in cui giunge la merce in sede. Tale modalità risulta vantaggiosa al cliente in quanto presenta in media un prezzo inferiore al 20% rispetto al prezzo standard. In seguito alla conferma il carrello del preordine verrà inviato al server e creato un ordine al fornitore.

## 3.4 Progettazione dei dati

Tra il client (in questo caso l'applicazione) e il server ci sarà un costante scambio di informazioni che dovranno essere ben organizzate su entrambi i lati.

La scelta è ricaduta sull'utilizzo di un database relazionale, diversamente dall'app che invece utilizza le *classi* per incapsulare le informazioni e schematizzarle correttamente.



Schema logico database

Ogni *entità* del database viene trasformata in una *classe Kotlin* insieme ai suoi attributi.

Di seguito vengono elencate alcune delle entità più importanti:

- **Utente:** sono presenti le informazioni anagrafiche, il tipo di utente, il token per effettuare le operazioni e la data di creazione dell'account;
- **Ordine:** sono presenti le informazioni del cliente che ha effettuato l'ordine, lo stato dell'ordine, la data ed eventuali note aggiuntive;
- **Articolo:** sono presenti le informazioni degli articoli, l'iva, la categoria, il nome, il prezzo e le informazioni sulla dimensione del prodotto (altezza, lunghezza, larghezza, peso e volume);
- **Carrello:** sono presenti le informazioni dell'utente proprietario del carrello, la data di creazione del carrello e l'ultima data di modifica (la data dell'ultima operazione).

di aggiunta/rimozione di un prodotto), particolarmente utile per inviare email di promemoria o promozionali.

.



#### 4.1 Architettura

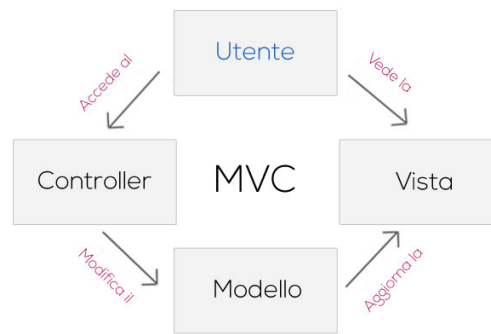
Per ottenere una buona fase sia di sviluppo che di manutenzione è importante scegliere con attenzione l'architettura. Non si riscontrano architetture migliori delle altre, si osserva solo una variazione del sistema, del linguaggio e delle intenzioni future del progetto. Il seguente progetto segue il pattern architetturale MVC<sup>1</sup> in cui la struttura viene suddivisa in tre componenti:

- **Model:** contiene i metodi di accesso ai dati;
- **View:** visualizza i dati elaborati all'utente e gestisce le possibili iterazioni tra l'utente e il sistema;
- **Controller:** gestisce le richieste del client ed esegue operazioni che tipicamente portano ad utilizzare il Model. Infine, i dati vengono ritrasmessi all'utente con la View.

L'obiettivo principale prevede di disaccoppiare e creare quante più indipendenze possibili tra le componenti, permettendo un migliore riutilizzo in futuro e un'efficiente manutenibilità.

---

<sup>1</sup>Model View Controller



*Schema funzionamento pattern MVC*

Considerando il sistema Android, le View verranno rappresentate tramite file *xml*, ovvero i layout, mentre per i Controller, le Activity e il Model verranno create delle classi *Kotlin*, realizzate per contenere informazioni. L'Activity rappresenta il cuore dell'applicazione e si occupa della gestione delle chiamate al server e della creazione degli oggetti che verranno successivamente visualizzati dalle View.

## 4.2 Librerie

Le librerie rappresentano un insieme di funzioni che possono essere utilizzate dal software che le importa. Di seguito sono elencate le librerie più importanti utilizzate durante lo sviluppo del progetto.

### 4.2.1 Libreria Volley

L'applicazione per comunicare e ricevere informazioni con il server necessita di effettuare delle richieste HTTP. Per la realizzazione del progetto è stata utilizzata, per comunicare con il server, la libreria di Google, Volley, in quanto il funzionamento è molto semplice e le prestazioni sono elevate. Ogni richiesta rappresenta un oggetto *JsonObjectRequest* e per inviarlo è sufficiente creare una *RequestQueue* (un pool di richieste); mentre per aggiungere una nuova richiesta alla coda bisogna utilizzare il metodo *addToRequestQueue* che gestirà la richiesta. La risposta della HTTP Request viene gestita nell'oggetto di tipo *JsonObjectRequest* con un listener dove è presente sia l'*onResponse(JSONObject)* in caso in cui il server risponda senza problemi, sia *onErrorResponse(VolleyError)* se il server presenta degli errori durante la gestione della richiesta.

---

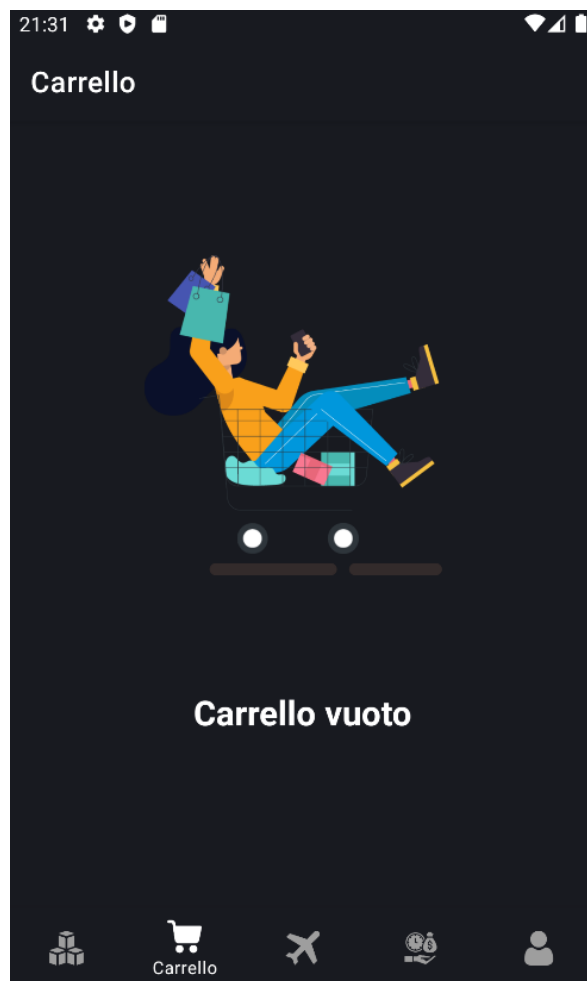
```
fun clearCart(idUtente: Int, token: String, queue:
    RequestQueue, onSuccess: () -> Unit, onError: (mess: String) ->
    Unit)
{
    val url="{Utilita.host}/api/cart/removeCart"
    val json=JSONObject()
    json.put("Token", Utilita.token)
    json.put("idUtente", idUtente)
    json.put("TokenAuth", token)
    val jsonObjectRequest = JsonObjectRequest(
        Request.Method.POST, url, json,
        { response ->
            val ris=response.getInt("Ris")
            if(ris==1)
            {
                onSuccess()
            }
            else
            {
                val mess=response.getString("Mess")
                onError(mess)
            }
        },
        { error ->
            onError(error.toString())
        }
    )
    jsonObjectRequest.retryPolicy =
        DefaultRetryPolicy(
            0,
            DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
            DefaultRetryPolicy.DEFAULT_BACKOFF_MULT)
    queue.add(jsonObjectRequest)
}
```

---

*Esempio di chiamata HTTP con Volley*

### 4.2.2 Libreria Lottie Animation

Lottie Animation è una libreria contenente oltre diecimila animazioni (gratis e non). Per l'installazione è necessario scegliere l'animazione dalla libreria e scaricare il file, creando successivamente all'interno del layout la componente *LottieAnimationView* e indicando il file di animazione con l'attributo *rawRes*. Inoltre, è possibile modificare alcuni attributi come l'auto play (attivo per default) e il loop (attivo per default) sia nel file xml del layout, che programmaticamente.



*Esempio di animazione con Lottie*

### 4.2.3 Libreria ML Firebase

È stata implementata la libreria di Firebase ML (Machine Learning) per utilizzare il *text recognition* e il *barcode reader*. Il text recognition permette di trovare articoli simili in base ad una fotografia data in input dall'utente; ad esempio se il cliente scatta la foto ad un prodotto, vengono lette varie parole (nome del prodotto, grammatura, fragranza ecc...) e grazie ad un sistema di tag è possibile visualizzare gli articoli che contengono le parole riconosciute. Il

barcode reader è stato utilizzato per la lettura dei codici a barre andando a visualizzare il prodotto corrispondente; quest'ultima funzione è stata ideata sia per la ricerca di un prodotto che per l'evasione di un ordine nel momento in cui il dipendente deve prelevare i prodotti ordinati dal cliente. Firebase permette di scegliere di utilizzare la versione offline della libreria o la versione online. L'ultima è più accurata anche se presenta dei costi, a differenza di quella locale, che è gratuita. Inizialmente si è deciso di adottare la locale con la riserva di aderire a quella online nel caso si presentassero dei feedback negativi sul suo corretto funzionamento.

---

```
val uri: Uri = data?.data!!
val recognizer =
    TextRecognition.getClient(TextRecognizerOptions.DEFAULT_OPTIONS)
val image= InputImage.fromFilePath(requireContext(),uri)
recognizer.process(image)
    .addOnSuccessListener { visionText ->
        if(visionText.text.isNotEmpty())
        {
            var text=""
            for(block in visionText.textBlocks)
            {
                for(line in block.lines)
                {
                    if(line.text.length>3)
                        text+="${line.text.trim()} "
                }
            }
            getProdTags(text)
        }
    }
```

---

#### *Utilizzo di ML Firebase per il Text Recognition*

#### **4.2.4 Libreria Gson**

Per la comunicazione tra l'applicazione e il server si è deciso di utilizzare il JSON come formato per lo scambio dei dati.

Inoltre, è stato scelto di implementare la libreria Gson per facilitare la trasformazione tra oggetto e JSON in un'unica semplice chiamata. La libreria presenta varie opzioni nel dettaglio che non sono state utilizzate in quanto futili per le nostre operazioni. I metodi principali sono *toJson()* e *fromJson()*, che permettono, la prima, la trasformazione di un oggetto in una

stringa in formato JSON, mentre la seconda trasforma la stringa JSON in un oggetto di una classe specifica.

#### 4.2.5 Libreria OneSignal

OneSignal permette la gestione delle notifiche in modo intelligente, conferendo la possibilità di identificare l'utente non solo tramite l'UID<sup>2</sup> ma anche tramite attributi alternativi. In questo caso è stata utilizzata l'email del cliente in modo da rendere più facile l'invio di notifiche. La libreria è stata utilizzata sia lato lato server (installando la libreria per NodeJS) sia tramite app (libreria per Kotlin), andando a impostare l'email come identificatore lato client side, a differenza dell'invio delle notifiche che avviene server side durante l'esecuzione di determinate richieste. Per descrivere un esempio concreto, durante la gestione di un ordine viene inviata una notifica al cliente per avvisarlo degli eventuali stati. Un ulteriore possibile utilizzo è l'invio di notifiche promozionali e/o avvisi ai clienti; grazie all'utilizzo di OneSignal è possibile inviare notifiche ad un determinato gruppo di utenti. È inoltre possibile inviare messaggi mirati anche in caso di manutenzione straordinarie o di promozioni valide solo per alcuni dispositivi e/o zone in base al sistema operativo, alla versione dell'app e alla posizione geografica.

### 4.3 Git

Durante lo sviluppo di un progetto, che sia di gruppo o singolo, è fondamentale tenere traccia dell'evoluzione del codice insieme a tutte le modifiche relative al progetto sia per un'organizzazione delle versioni sia per conservare una copia di backup del progetto. Questi strumenti sono chiamati VCS (Version Control System); tra i più famosi spicca Git, un sistema di VCS gratuito e open-source dove ogni progetto viene definito *repository*. Tali strumenti permettono di gestire la possibilità di lavorare parallelamente allo stesso progetto, e di gestire anche casi di conflitto. Per utilizzare Git si fa riferimento a GitHub, un cloud-based hosting di Microsoft che permette di gestire le repository Git.

### 4.4 Gestione dei permessi

Il sistema realizzato presenta un sistema di autenticazione basato sui token. Pertanto, anche se un utente esperto riesca ad identificare alcune chiamate API con dei privilegi

---

<sup>2</sup>Unique identifier

maggiori rispetto a quelli concessi, le operazioni non verranno effettuate in quanto ogni HTTP Request avrà in input due parametri: l'identificativo dell'utente e il suo token di autenticazione che verrà fornito all'utente al momento del login in caso di credenziali corrette. Questo sistema per tale motivo riesce ad identificare il tipo di utente (socio, dipendente, utente o bannato) effettuando un controllo lato server per ogni operazione eseguita.

---

```
Utente.getTipoUtente=(idUtente,tokenAuth,result)=>{
  let query="SELECT ksTipo FROM Utente WHERE idUtente=? AND
    TokenAuth=?";
  sql.query(query,[idUtente,tokenAuth],(errQ,risQ)=>{
    if(errQ)
    {
      result(-1);
    }
    else
    {
      if(risQ.length)
      {
        result(risQ[0].ksTipo)
      }
      else
      {
        result(-1);
      }
    }
  });
};
```

---

#### *Funzione che ritorna il tipo di utente*

In questo caso specifico viene effettuata una query al database che seleziona il tipo di utente dove l'identificativo dell'utente e il suo token combaciano con quelli in input. Viene utilizzato l'operatore "?" per evitare l'SQL Injection<sup>3</sup>.

---

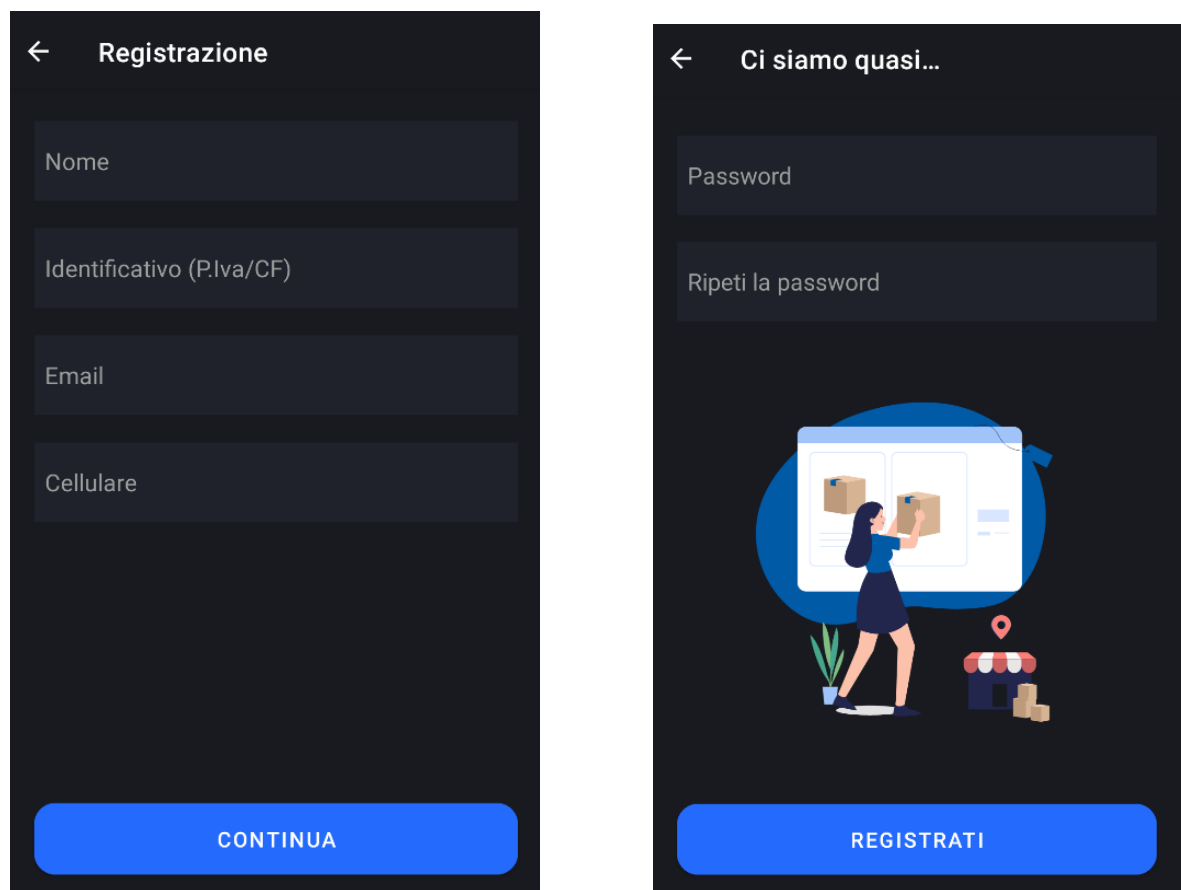
<sup>3</sup><https://portswigger.net/web-security/sql-injection>

## 4.5 Funzionalità principali realizzate

Di seguito sono presentate alcune delle funzionalità implementate nella prima versione dell'applicazione. Per ogni funzionalità verrà presentata l'interfaccia utente con cui l'utente interagisce e lo scopo di essa.

### 4.5.1 Registrazione

Per poter utilizzare tutte le funzionalità, l'utente deve registrarsi. Nella schermata principale è presente un *Button* che porta alla schermata della registrazione che viene suddivisa in due *Fragment*. In particolare, il primo riguarda l'inserimento dei dati personali/aziendali mentre il secondo l'inserimento della password.



The figure consists of two side-by-side screenshots of a mobile application's registration process. Both screens have a dark background and a blue header bar with a back arrow on the left.

Screen (a) is titled 'Registrazione' and contains four input fields stacked vertically: 'Nome', 'Identificativo (P.Iva/CF)', 'Email', and 'Cellulare'. At the bottom is a large blue button labeled 'CONTINUA'.

Screen (b) is titled 'Ci siamo quasi...' and contains two input fields: 'Password' and 'Ripeti la password'. Below these fields is an illustration of a woman in a blue dress interacting with a large screen that displays a form with checkboxes. To the right of the illustration is a small red location pin icon and some cardboard boxes. At the bottom is a large blue button labeled 'REGISTRATI'.

(a) Inserimento dati aziendali

(b) Inserimento password

**Figura 4.1:** Schermata registrazione

Come si può osservare dalle figure, le due schermate presentano prevalentemente degli *EditText* e dei *Button* per confermare la registrazione, insieme ad una simpatica animazione utilizzando la libreria *Lottie*.



Una volta confermata la registrazione viene inviata la richiesta al server che effettua prima dei controlli, come ad esempio se l'email è già presente nel database oppure se la password rispetta i requisiti (quest'ultimo controllo viene effettuato sia client-side che server-side). Dopo aver completato questi passaggi, se la registrazione ha esito positivo l'utente viene riportato alla schermata iniziale; se al contrario si riscontrano errori viene visualizzato il messaggio di errore (email già registrata oppure identificativo non valido), invitando (se possibile) l'utente a risolvere il problema.

#### 4.5.2 Ricerca prodotti

Nella schermata dei prodotti è possibile visualizzare una lista di prodotti utilizzando la componente *RecyclerView*. Prima della lista sono anche presenti alcune componenti per filtrare i prodotti; si evidenziano 3 *Button*. Il primo permette di filtrare i prodotti in base alla categoria e/o alcuni parametri come il prezzo, la descrizione e la disponibilità in magazzino; il secondo permette la ricerca tramite machine learning di prodotti in base ad una foto data in input, infine il terzo permette la ricerca di un prodotto dato il suo codice a barre. È inoltre presente anche un *EditText* per filtrare i prodotti in base alla descrizione.

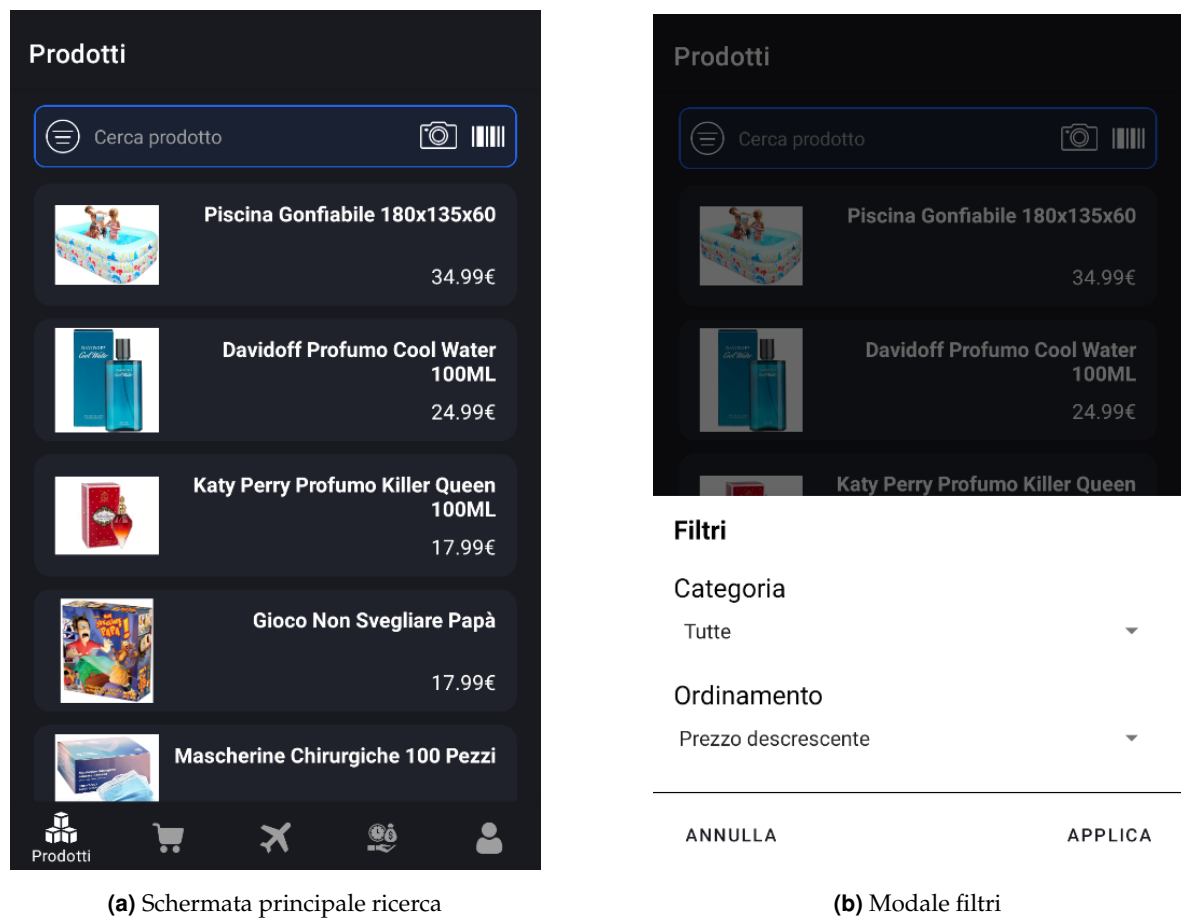
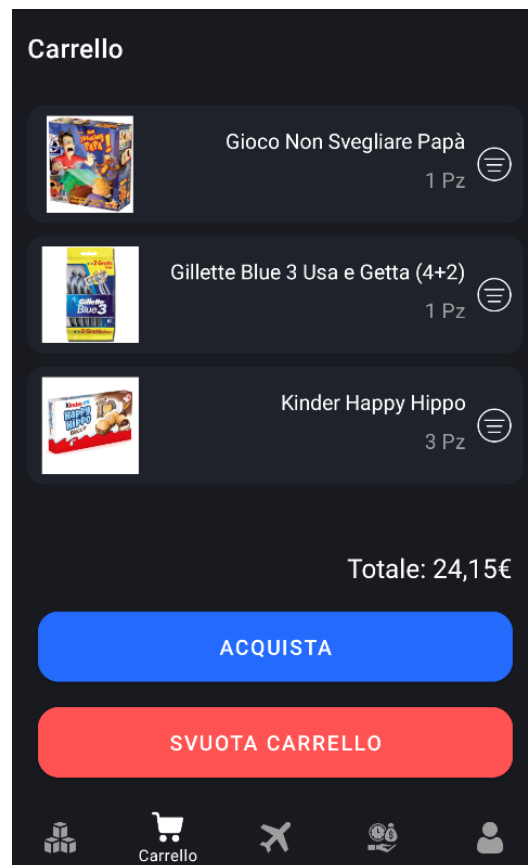


Figura 4.2: Schermata filtri prodotti

Per l’ottimizzazione delle risorse e dei tempi di risposta, i prodotti vengono inviati utilizzando la paginazione, vengono inviate le informazioni sui prodotti solo se è necessario. Cliccando su un prodotto verrà visualizzata una schermata che presenta tutti i dettagli del prodotto e la possibilità di aggiungerlo al carrello per effettuare un ordine.

### 4.5.3 Creazione ordine

Dopo aver inserito tutti i prodotti nel carrello, l’utente può sia visualizzare i prodotti aggiunti e il totale, che effettuare operazioni come eliminare un prodotto dal carrello oppure modificare la quantità. Sono presenti due *Button*; il pulsante ‘Svuota carrello’ che serve per eliminare tutti i prodotti dal carrello, mentre il pulsante ‘Acquista’ permette di effettuare l’ordine ed inviare tutte le informazioni del carrello al server tramite una chiamata HTTP che in caso di successo crea un nuovo ordine visualizzato dai dipendenti e dagli admin con la possibilità di iniziare l’evasione dell’ordine. Il cliente verrà aggiornato tramite notifica per ogni modifica allo stato dell’ordine.



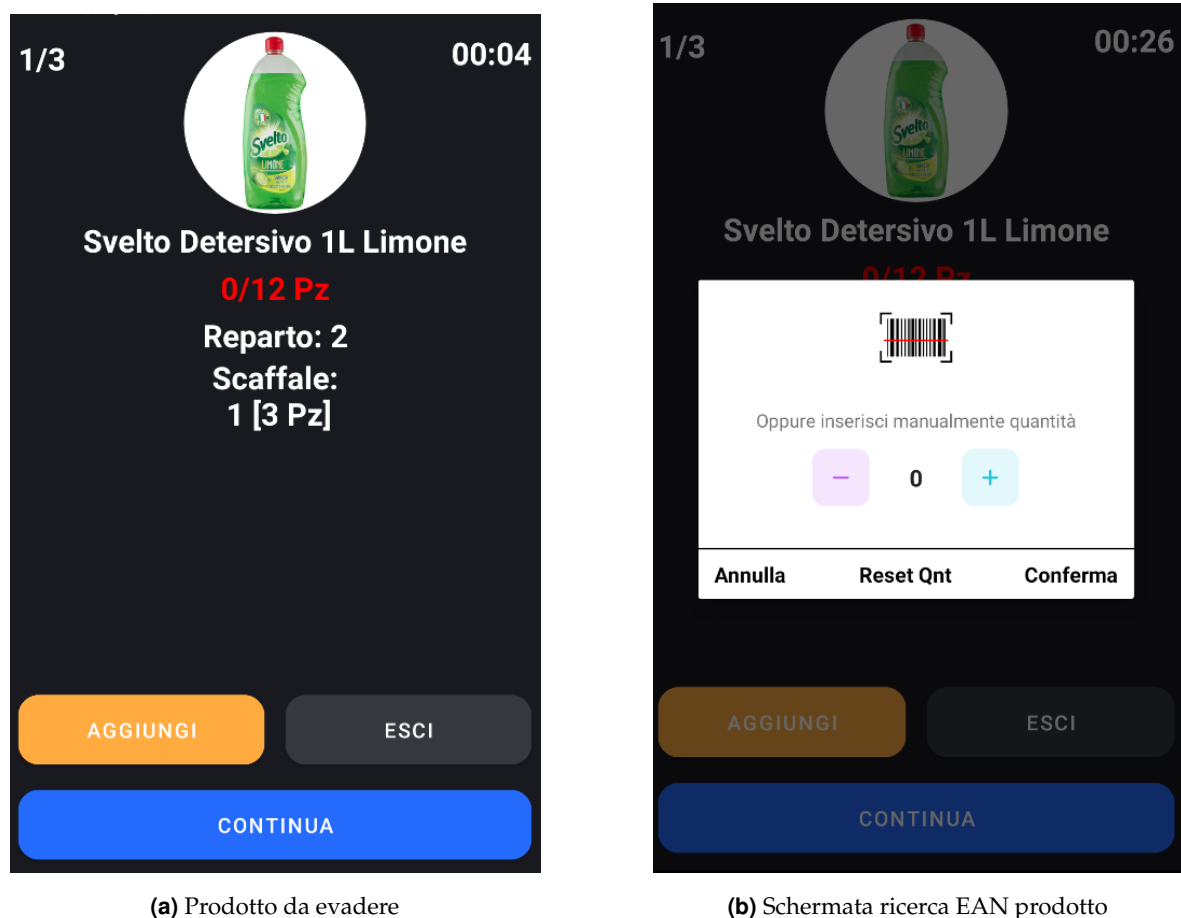
(a) Carrello

**Figura 4.3:** Schermata creazione ordine

#### 4.5.4 Evasione ordini

I dipendenti accedono ad una sezione per visualizzare tutti gli ordini evadibili. Dopo aver selezionato l'ordine sarà possibile visualizzare una schermata che mostrerà il prodotto da prelevare, la quantità e la posizione nel magazzino.

Nella schermata sono presenti tre *Button*: il primo pulsante 'Aggiungi' aprirà un modale dove l'operatore può scansionare l'EAN per prelevare il prodotto richiesto dal cliente, oppure inserire la quantità manualmente. Il secondo pulsante 'Esci' permette al dipendente di effettuare una chiusura motivata dell'ordine nel caso di pausa pranzo, cambi turno o emergenze. Il terzo e ultimo pulsante 'Continua' permette di passare al prodotto successivo e di continuare con l'evasione dell'ordine.



(a) Prodotto da evadere

(b) Schermata ricerca EAN prodotto

Figura 4.4: Schermata evasione ordine

#### 4.5.5 Visualizza statistiche

Gli admin (i soci) hanno una sezione dedicata alle statistiche che è suddivisa in due parti. Nella prima vengono visualizzate le statistiche generali, constatando il dipendente che ha evaso più ordini e il dipendente che ha effettuato più errori, mentre nella seconda zona viene mostrata una lista di tutti i dipendenti tramite una *RecyclerView* che permette, attraverso un click su un utente, di visualizzare tutte le statistiche personali della persona selezionata. Queste funzioni possono essere utili per verificare come sta procedendo l'utilizzo dell'applicazione e per cercare di ridurre i margini di errore dei dipendenti.

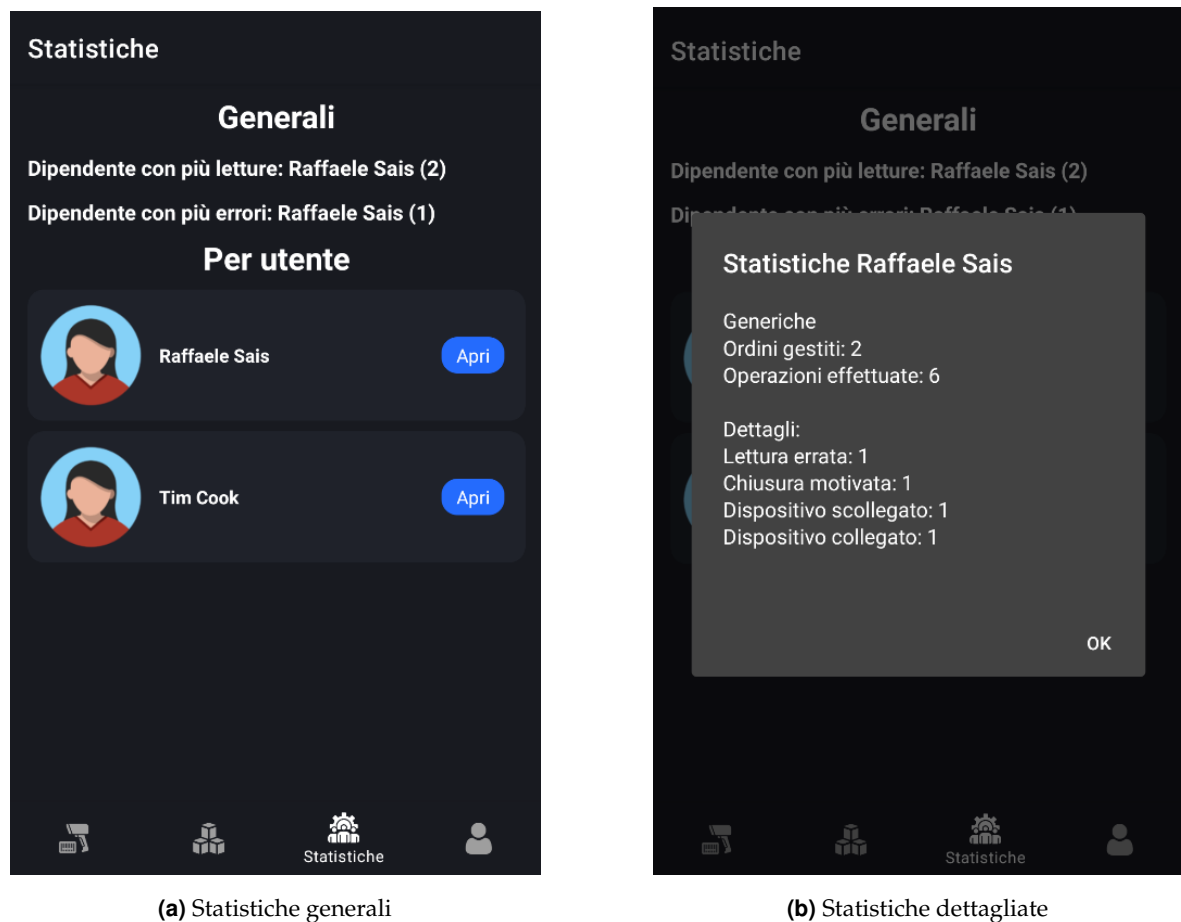


Figura 4.5: Schermata statistiche

#### 4.5.6 Supporto multilingua

Una funzione semplice da implementare molto importante per un processo di espansione aziendale è sicuramente il supporto a varie lingue oltre all'italiano.

Al momento l'applicazione è disponibile sia in lingua italiana che inglese. Il procedimento di traduzione consiste nell'inserire in un file *string.xml* tutte le stringhe associandole alle variabili. Dopodiché per ogni altra lingua che si vuole implementare è necessario creare un altro file.

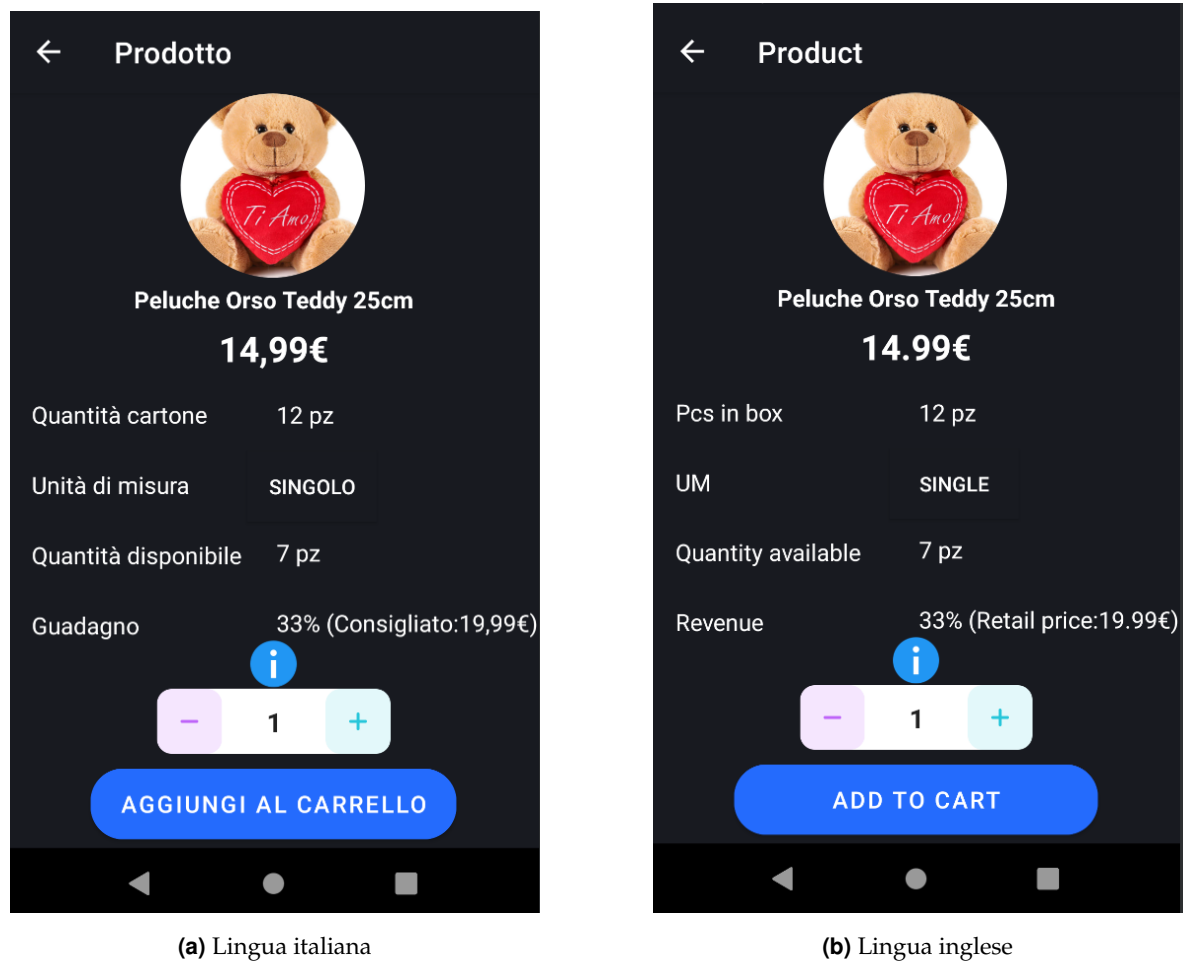
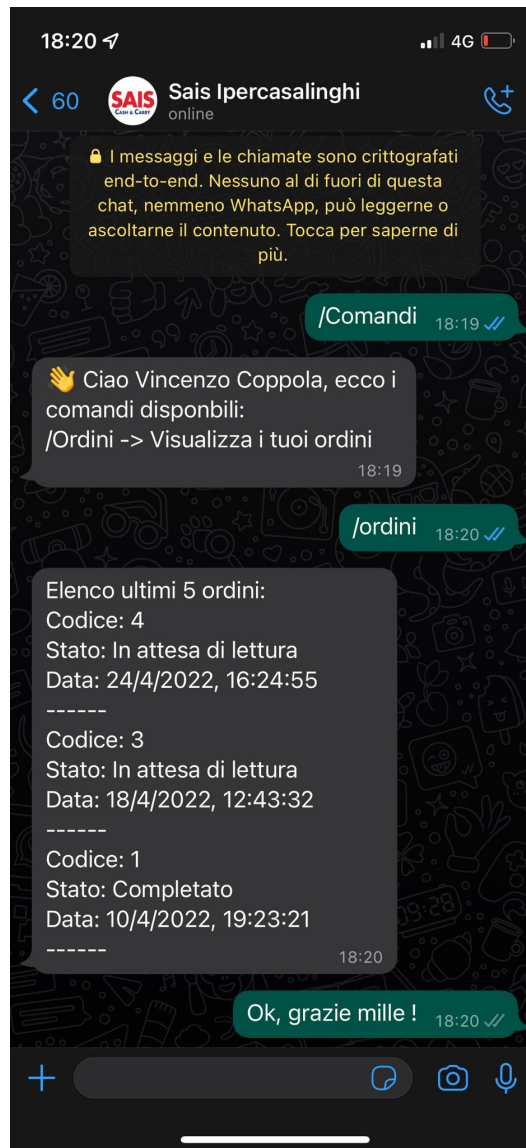


Figura 4.6: Schermata dettaglio prodotto

#### 4.5.7 Assistenza Whatsapp

Tra gli obiettivi del progetto, è prevista una maggiore trasparenza con il cliente e la possibilità di un'assistenza semplice e rapida. Per tale motivo è stato sviluppato un bot, sulla nota piattaforma di messaggistica istantanea Whastapp, ove ogni cliente attraverso dei comandi rapidi può visualizzare le informazioni relative agli ordini. Attualmente sono stati realizzati due comandi:

- **\Comandi:** inviando il seguente comando verrà visualizzata la lista dei comandi disponibili (anche in base al tipo di utente);
- **\Ordini:** inviando il seguente comando verrà visualizzata la lista degli ordini effettuati.



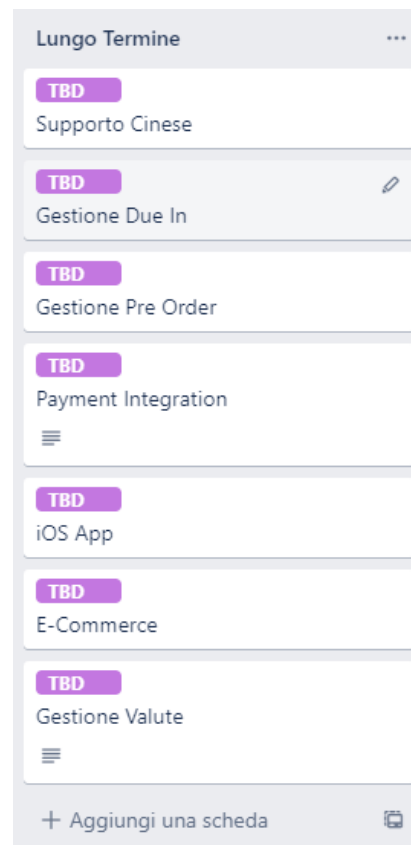
(a) Chat d'esempio utilizzo BOT

#### 5.1 Trello

Trello è uno strumento che permette la gestione/pianificazione di qualsiasi tipo di progetto e la condivisione con i colleghi e/o i clienti. Al giorno d'oggi una caratteristica aziendale importante per il suo successo è la trasparenza; arricchita con dialoghi e/o informazioni informali, permettendo un rapporto più stabile ed emotivo tra due soggetti. New Net SRL si impegna quotidianamente per fornire un'esperienza migliore ai suoi clienti.

In questo caso, tutte le funzionalità a lungo termine, i bug e le informazioni sono sempre visibili e verificabili tramite la pagina Trello.





*Pianificazione funzionalità a lungo termine*

## 5.2 Funzioni da implementare

In questa sezione sono elencate tutte le funzionalità non ancora sviluppate ma prioritarie per migliorare il servizio.

### 5.2.1 Gestione prevendite

Con la gestione prevendita il cliente può vedere in anticipo i prossimi carichi aziendali e richiedere la merce a prezzo scontato. La suddetta funzione presenta vantaggi sia per i venditori che per gli acquirenti, in quanto, permette a chi effettua la vendita di gestire al meglio l’allocazione dei prodotti in magazzino, consentendo di effettuare vendite senza esporre la merce. Il vantaggio principale per il cliente prevede l’acquisto di prodotti ad un prezzo inferiore rispetto a quello esposto a scaffale.

### 5.2.2 Gestione quotazioni

Con la gestione quotazioni il cliente può richiedere dei prodotti che non sono disponibili in magazzino, facilitando il venditore, in quanto molti fornitori hanno un MOQ<sup>1</sup> elevato. Pertanto con le richieste dei clienti può diventare più facile effettuare ordini da multinazionali ed avere sconti maggiori. Anche nel suddetto caso vengono presentati gli stessi benefici della gestione prevendita (elencati precedentemente) in quanto vengono effettuate vendite risparmiando tempo con l'esposizione e aumentando i guadagni ottimizzando la forza lavoro.

### 5.2.3 Ordine senza fila

Uno degli obiettivi principali di questo progetto è sicuramente l'ottimizzazione del lavoro; molti B2C (tipicamente le grandi distribuzioni come Carrefour, LIDL ecc...) hanno iniziato a sperimentare la cosiddetta *cassa veloce*. Si è deciso di realizzarla e renderla disponibile anche nel mondo del B2B, evitando la coda e risparmiando molto tempo sia per il lavoratore che per il cliente. Per effettuare la spesa rapida basterà semplicemente accedere all'applicazione ed essere collegati al WiFi dell'azienda. Questo preposto è il primo livello di spesa rapida, tuttavia, ci sono possibilità potenzialmente infinite; basti pensare al servizio Amazon Go<sup>2</sup> che utilizza le telecamere e il machine learning per una spesa veloce senza neanche toccare il cellulare, dove una volta usciti dal negozio viene effettuato l'addebito per la spesa effettuata.

### 5.2.4 Integrazioni pagamenti

Oggigiorno sono pochi i fornitori che accettano pagamenti online oltre al comunissimo bonifico bancario; il nostro obiettivo prevede l'integrazione dei maggiori gateway di pagamento<sup>3</sup> tali Stripe, Paypal e gestire tramite le API bancarie anche il controllo sui bonifici. Grazie a tale innovazione è possibile ridurre il tempo su controlli di pagamento, scadenze, resi e note credito, automatizzando il processo anche se bisogna considerare delle commissioni da pagare che tipicamente oscillano tra lo 0.4% e l'1.2%.

### 5.2.5 Lingua cinese

L'attivazione di un processo di internazionalizzazione da parte di un'azienda necessita la scelta di selezione della lingua; purtroppo non è sufficiente la lingua italiana come unica

---

<sup>1</sup>Minimum Order Quantity

<sup>2</sup><https://www.aboutamazon.it/notizie/innovazioni/amazon-go>

<sup>3</sup>[https://it.wikipedia.org/wiki/Gateway\\_di\\_pagamento](https://it.wikipedia.org/wiki/Gateway_di_pagamento)

lingua del sistema. Il progetto dispone già l'utilizzo della lingua inglese anche se l'obiettivo preposti prevede l'aggiunta di una terza lingua, il cinese, molto importante per il mercato del B2B in Italia.

### 5.2.6 Gestione valute

La gestione delle valute riguarda le aziende che vogliono avviare un processo di internazionalizzazione ricoprendo aree dove non è presente l'euro; tra le principali ci sono lo yen (Giappone), la sterlina (Regno Unito), il dollaro (USA) e il renminbi (Cina). Pertanto, è necessario gestire le valute dei principali mercati.

## 5.3 Espansione target

Qui sono elencati degli obiettivi a lungo termine per completare lo sviluppo ed espandere il target.

### 5.3.1 Applicazione iOS

Realizzare un'applicazione in iOS (e iPadOS) rappresenta l'obiettivo a lungo termine più importante in quanto gli utenti che utilizzano un iPhone sono circa 2 miliardi<sup>4</sup> e in questo momento viene, quindi, esclusa una consistente clientela. Escludendo le funzionalità riservate ai dipendenti, in quanto tali funzioni verranno svolte su un terminale portatile con un lettore di codici a barre che hanno tutti come SO Android, le funzionalità per gli utenti e per i soci richiedono la creazione di un app per l'ecosistema Apple che presenta anche dei costi fissi annuali, come ad esempio i 99€/anni previsti per l'Apple Developer Account<sup>5</sup>.

### 5.3.2 E-Commerce

Realizzando l'e-commerce si riesce a raggiungere dei nuovi utenti che preferiscono utilizzare un pc anziché l'applicazione. Verranno gestite le funzionalità che riguardano gli utenti (effettuare ordini, visualizzare ordini ecc...) e quelle relative ai soci (statistiche, fatturato ecc.); escludendo le funzionalità per i dipendenti in quanto da una piattaforma web non verranno mai utilizzati.

---

<sup>4</sup><https://tg24.sky.it/tecnologia/2020/01/11/iphone-due-miliardi-apple>

<sup>5</sup><https://developer.apple.com/support/compare-memberships>

### 5.3.3 API pubbliche

Realizzando un servizio pubblico di API, si permette alle aziende la creazione di un'app e/o sito personalizzato secondo le loro esigenze, sfruttando appieno tutte le capacità del sistema. Per la realizzazione verrà installato solo il server API senza la configurazione dell'applicazione mentre al cliente verrà dato un *token* per accedere a tutte le chiamate API.

L'obiettivo preposti prevede la realizzazione di un sistema che possa colmare il gap dell'utilizzo delle tecnologie tra il B2C e il B2B, realizzando un sistema fluido e intuitivo che rispetti gli standard attuali per ottenere una buona applicazione. La versione attuale rappresenta un prototipo che alcune aziende possono iniziare ad utilizzare internamente, senza rilasciarla al pubblico in quanto al momento il progetto è disponibile solo per Android. L'applicazione dispone di tutte le funzionalità primarie anche se non è ancora stata testata da nessuna azienda. Pertanto, il passo successivo prevede la ricezione di feedback iniziali per l'applicazione mentre si realizzano gli altri obiettivi elencati precedentemente, inoltre è importante notare che il sistema comunica in modo sicuro, andando a criptare tutte le informazioni sensibili.

Tuttavia, si possono già individuare alcune modifiche da effettuare per rendere il progetto sempre più solido; tra queste si evidenzia la gestione dell'applicazione nel caso in cui il dispositivo dovesse andare offline.

Tra le altre modifiche da apportare si rileva la localizzazione dell'applicazione in lingua inglese, tradotta al momento in modo superficiale. Necessita, pertanto, di un'accurata revisione da parte di un esperto, che procederà a rendere disponibile l'applicazione anche in altre lingue come ad esempio il cinese.

Inoltre, per quanto concerne la gestione degli utenti da parte degli admin, questa funzionalità è stata trascurata nonostante rivesti una grande importanza per il lancio dell'applicazione.

Pertanto, l'obiettivo del prototipo è stato completato seppur è evidente la mancanza di alcune funzionalità essenziali per il corretto funzionamento del sistema.

## Siti Web consultati

- Android Developers – [www.developer.android.com](http://www.developer.android.com)
- Kotlin Guide – [www.kotlinlang.org](http://www.kotlinlang.org)
- Firebase Docs – [www.firebase.google.com/docs](http://www.firebase.google.com/docs)
- Wikipedia – [www.wikipedia.org](http://www.wikipedia.org)
- NodeJS Docs – [www.nodejs.org](http://www.nodejs.org)
- ExpressJS Guide – [www.expressjs.com](http://www.expressjs.com)