

# Università Politecnica delle Marche

Facoltà di Ingegneria

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica e dell'Automazione

---



**Tesi di Laurea**

**Progettazione e implementazione di un'app Android per la gestione del baratto**

**Design and implementation of an Android app for barter management**

Relatore

Prof. Domenico Ursino

Candidato

Matteo Vitellozzi

---

**Anno Accademico 2020-2021**



---

# Indice

|          |   |    |
|----------|---|----|
| <b>1</b> | <b>Android</b>  | 3  |
| 1.1      | Introduzione  | 3  |
| 1.2      | Storia  | 3  |
| 1.3      | Sistemi Operativi                                     | 5  |
| 1.3.1    | Versioni  | 7  |
| 1.4      | Funzionalità di Android                               | 7  |
| 1.4.1    | Interfaccia   | 7  |
| 1.4.2    | Kernel Linux  | 8  |
| 1.5      | Sviluppo  | 8  |
| 1.5.1    | Android Studio  | 8  |
| 1.5.2    | Architettura Android                                  | 10 |
| 1.5.3    | App Android   | 12 |
| <b>2</b> | <b>Il Baratto nella storia</b>                        | 15 |
| 2.1      | La Storia   | 15 |
| 2.2      | Il Concetto del Baratto                               | 16 |
| 2.2.1    | La Moneta   | 17 |
| 2.2.2    | La Moneta Fisica                                      | 18 |
| 2.2.3    | La Moneta Virtuale                                    | 19 |
| 2.3      | Lo Scambio tra Produttore e Consumatore               | 22 |
| <b>3</b> | <b>Analisi dei requisiti e progettazione dell'App</b> | 25 |
| 3.1      | Descrizione   | 25 |
| 3.1.1    | Requisiti Funzionali                                  | 26 |
| 3.1.2    | Requisiti non Funzionali                              | 27 |
| 3.1.3    | Diagrammi   | 28 |
| 3.2      | Struttura dei Dati                                    | 31 |
| 3.2.1    | Dettaglio dei Dati                                    | 32 |
| 3.3      | I Database  | 33 |
| 3.3.1    | Firestore   | 34 |

|          |   |    |
|----------|---|----|
| <b>4</b> | <b>Implementazione</b> .....                  | 37 |
| 4.1      | Premessa dell'applicazione .....              | 37 |
| 4.2      | Login e Registrazione .....                   | 37 |
| 4.3      | Home page .....                               | 39 |
| 4.4      | Aggiunta dei prodotti .....                   | 41 |
| 4.5      | Profilo .....                                 | 42 |
| 4.6      | Offerte inviate e ricevute .....              | 45 |
| 4.7      | Riepilogo scambi .....                        | 48 |
| 4.8      | Contatti .....                                | 49 |
| 4.9      | Statistiche .....                             | 49 |
| 4.10     | Logout .....                                  | 51 |
| <b>5</b> | <b>Manuale Utente</b> .....                   | 53 |
| 5.1      | Avvio dell'applicazione .....                 | 53 |
| 5.2      | Login e Registrazione .....                   | 54 |
| 5.3      | Catalogo e Menù .....                         | 55 |
| 5.4      | Visualizzazione prodotto .....                | 57 |
| 5.5      | Formula Offerta .....                         | 57 |
| 5.6      | Miei oggetti .....                            | 58 |
| 5.7      | Inserisci prodotto .....                      | 59 |
| 5.8      | Profilo .....                                 | 59 |
| 5.9      | Offerte inviate e ricevute .....              | 61 |
| 5.10     | Riepilogo scambi .....                        | 63 |
| 5.11     | Statistiche .....                             | 65 |
| 5.12     | Contattaci e Logout .....                     | 66 |
| <b>6</b> | <b>Confronto con approcci correlati</b> ..... | 67 |
| 6.1      | Lo shopping online .....                      | 67 |
| 6.1.1    | Vinted .....                                  | 68 |
| 6.1.2    | Ebay .....                                    | 71 |
| 6.1.3    | Shpock .....                                  | 73 |
| 6.1.4    | Subito.it .....                               | 75 |
| 6.1.5    | Amazon .....                                  | 77 |
| <b>7</b> | <b>Conclusioni</b> .....                      | 81 |
|          | <b>Riferimenti bibliografici</b> .....        | 83 |

---

## Elenco delle figure

|     |   |    |
|-----|---|----|
| 1.1 | Primo telefono Android lanciato sul mercato . . . . .                       | 4  |
| 1.2 | Struttura del Sistema Operativo . . . . .                                   | 6  |
| 1.3 | Struttura del Kernel Linux con i suoi componenti . . . . .                  | 8  |
| 1.4 | Panoramica di Android Studio . . . . .                                      | 10 |
| 1.5 | Architettura interna di Android . . . . .                                   | 11 |
| 1.6 | File APK di un'App Android . . . . .  | 12 |
|     |   |    |
| 2.1 | Baratto tra indiano e cacciatore . . . . .                                  | 16 |
| 2.2 | Rappresentazione schematica di uno scambio di beni . . . . .                | 17 |
| 2.3 | Moneta Attica con la Dea Atena e la civetta . . . . .                       | 17 |
| 2.4 | Banconota cinese della dinastia Song . . . . .                              | 18 |
| 2.5 | Processo di generazione di una criptovaluta . . . . .                       | 20 |
| 2.6 | Esempio di valuta virtuale: Il Bitcoin . . . . .                            | 20 |
| 2.7 | Portafoglio virtuale (Wallet) . . . . .                                     | 21 |
| 2.8 | Statistiche delle principali criptovalute del 04/08/2021 16:46 . . . . .    | 22 |
| 2.9 | Schema del problema di sincronizzazione: produttore e consumatore . . . . . | 23 |
|     |   |    |
| 3.1 | Diagramma dei casi d'uso BartApp . . . . .                                  | 29 |
| 3.2 | Diagramma di flusso BartApp . . . . .                                       | 30 |
| 3.3 | Diagramma di flusso delle Statistiche di BartApp . . . . .                  | 31 |
| 3.4 | Struttura JSON dei dati . . . . .   | 32 |
| 3.5 | Modello Client-Server . . . . .   | 34 |
| 3.6 | Console di Firebase . . . . .   | 35 |
|     |   |    |
| 5.1 | Icona di BartApp . . . . .  | 53 |
| 5.2 | Schermata di login . . . . .  | 54 |
| 5.3 | Schermata di registrazione . . . . .  | 55 |
| 5.4 | Catalogo dei prodotti . . . . .   | 56 |
| 5.5 | Menù a tendina . . . . .  | 56 |
| 5.6 | Dettagli del prodotto . . . . .   | 57 |
| 5.7 | Schermata di scelta del prodotto da offrire . . . . .                       | 58 |
| 5.8 | Schermata dei Miei oggetti . . . . .  | 59 |
| 5.9 | Schermata di aggiunta dei prodotti . . . . .                                | 60 |

|      |   |    |
|------|---|----|
| 5.10 | Profilo dell'utente .....                                     | 60 |
| 5.11 | Schermata per la modifica del nome utente .....               | 61 |
| 5.12 | Schermata per la modifica della password.....                 | 62 |
| 5.13 | Schermata delle offerte inviate dall'utente .....             | 62 |
| 5.14 | Schermata delle offerte ricevute .....                        | 63 |
| 5.15 | Avviso di conferma .....                                      | 64 |
| 5.16 | Schermata del riepilogo delle offerte accettate .....         | 64 |
| 5.17 | Schermata delle statistiche relative all'applicazione .....   | 65 |
| 5.18 | Schermata che visualizza le voci di Logout e Contattaci ..... | 66 |
|      |   |    |
| 6.1  | Figura rappresentativa dello shopping online .....            | 68 |
| 6.2  | Icona di Vinted .....   | 69 |
| 6.3  | Home page di Vinted .....                                     | 69 |
| 6.4  | Registrazione di Vinted .....                                 | 71 |
| 6.5  | Home page di eBay .....                                       | 72 |
| 6.6  | Vista prodotto di eBay.....                                   | 72 |
| 6.7  | Rappresentazione illustrativa del concetto di vendita .....   | 73 |
| 6.8  | Vista di Shpock .....   | 74 |
| 6.9  | Tollbar per la vendita su Shpock .....                        | 74 |
| 6.10 | Home page di Subito.it.....                                   | 76 |
| 6.11 | Logo di Amazon .....  | 77 |
| 6.12 | Home page di Amazon .....                                     | 78 |
| 6.13 | Pagina di registrazione di Amazon.....                        | 79 |

---

## Introduzione

Sin dagli inizi della preistoria, dalle prime civiltà, l'uomo ha avuto costantemente bisogno di effettuare scambi. Il bisogno primario di barattare un proprio bene per riceverne uno altrettanto importante è un processo primitivo che, seppur involontario, è di fondamentale importanza. La definizione del concetto di "baratto" si è diffusa molto rapidamente, a partire dagli Egizi, i quali barattavano beni per accaparrarsene altri, per arrivare al periodo dell'Alto Medioevo, dove le piccole società, in assenza di una moneta che assegnasse un valore alle cose, scambiavano i propri prodotti. Lo scambio veniva in seguito ad una valutazione puramente qualitativa, tale da stimare i prodotti coinvolti con un valore pressochè identico e consono per entrambe le parti. L'evoluzione della storia e l'attuale modernizzazione non ha, di fatto, cambiato i principi dello scambio. Ancora oggi, infatti, l'operazione di scambio è una pratica valida per il processo di vendita, con ovviamente aspetti più moderni. Con l'avvento della moneta, invece, la stima che viene fatta ad un oggetto è prettamente quantitativa. Ad ogni prodotto è associato un valore monetario ben preciso. La tecnologia, per di più, è riuscita a combinare i processi di vendita offrendo agli acquirenti e ai venditori dei servizi che permettono loro di completare uno scambio con un semplice click. L'avvento delle piattaforme di shopping online, di fatto, ha rivoluzionato completamente il mondo del mercato. Al giorno d'oggi, con il progresso della modernizzazione e della tecnologia, conta più un sito web ben curato che una vetrina di un negozio. L'uomo apprezza più la comodità di fare shopping online da casa che la corsa al negozio. Proprio su queste basi, il nostro progetto si pone come obiettivo quello di realizzare un'applicazione che rispecchi un pò la nostra storia e la modernizzazione. Il lato moderno è realizzato dalla stessa applicazione che rappresenta, di fatto, un sito di e-commerce. Il lato storico, più tradizionale, invece, è espresso dai principi del baratto, mediante i quali ogni persona che intende disfarsi di oggetti in disuso può ottenerne altrettanti più utili.

L'applicazione oggetto della tesi è stata sviluppata su un sistema operativo che, al giorno d'oggi, conta gran parte degli utenti con dispositivi mobili. Per queste ragioni si è deciso di operare in Android, più precisamente sull'ambiente Android Studio. Prima di operare direttamente nello sviluppo dell'app, però, è stata definita una fase di raccolta di requisiti, informazioni e funzionalità che l'applicazione deve presentare. Perciò, sono state definite tutte le funzioni che l'app offre, corredate da descrizioni che spiegano le caratteristiche di ciascuna funzionalità. In questa fase,

poi, è stata progettata la struttura dell'applicazione mediante schermate e diagrammi che definiscono il suo comportamento al susseguirsi di input dell'utente, quali click, swipe o inserimenti di caratteri della tastiera. Conclusa la fase prettamente descrittiva e illustrativa dell'analisi dei requisiti, è stata, poi, definita ed esposta l'implementazione dell'app. Ogni funzionalità, descritta nella fase precedente, è stata implementata e ad essa sono state associate opportune schermate. Per garantire all'utente un impatto visivo che facilitasse ulteriormente la lettura delle medesime funzioni, inoltre, è stata definita una sezione che, di fatto, fornisce un manuale su come navigare all'interno dell'app. Terminata la fase progettuale, per concludere, sono state esposti diverse analogie e confronti con le realtà più vicine alla nostra applicazione di compravendita online. Sono state descritte le analogie e differenze fra le varie applicazioni per consentire, a futuri lettori interessati, di proseguire il progetto apportando le opportune migliorie.

La presente tesi è strutturata come di seguito specificato:

1. Nel Capitolo 1 viene presentato il sistema operativo Android. In questo capitolo, vengono raccontate la storia della sua nascita e la sua struttura dal punto di vista operativo. Vengono, poi, presentate le sue caratteristiche e le sue funzionalità, seguite dalla descrizione dell'ambiente di sviluppo Android Studio.
2. Nel Capitolo 2 vengono presentati la storia e lo sviluppo del baratto dagli albori della civiltà al mondo odierno. Partendo dal concetto di baratto, si è risaliti, poi, all'avvento della moneta e al suo processo evolutivo.
3. Nel Capitolo 3 viene definita la fase progettuale, che descrive i vari requisiti che devono essere soddisfatti dalla nostra applicazione, definendo le singole funzioni realizzate, corredate da diagrammi. In seguito viene definita la struttura dei dati e i database su cui essi sono memorizzati.
4. Nel Capitolo 4 viene descritta l'implementazione dell'applicazione. In questo capitolo sono presenti dei listati di codice che definiscono le singole funzioni che l'app propone agli utenti, corredate da descrizioni dettagliate.
5. Nel Capitolo 5 viene presentato un manuale utente che guida i lettori nella navigazione all'interno dell'applicazione. Essi, per mezzo di schermate illustrative dell'applicazione, sono guidati in tutte le funzionalità che l'applicazione stessa propone.
6. Nel Capitolo 6 sono esposte delle realtà simili alla nostra applicazione e vengono descritte eventuali analogie e differenze.



# Android

*In questo capitolo introduciamo il mondo Android, partendo dalla storia, quindi da come è nato e come è diventato così influente e diffuso nel mondo, a come si struttura e come è organizzato al suo interno. Successivamente daremo uno sguardo al mondo della programmazione Android e a ciò su cui oramai gira la tecnologia odierna, ovvero le App.*

## 1.1 Introduzione

Android è un sistema operativo per dispositivi mobili sviluppato da Google, progettato principalmente per sistemi embedded quali smartphone, tablet, dispositivi per televisori (Android TV), automobili (Android Auto), occhiali (Google Glass) e orologi.

Nel corso del tempo è diventato il sistema operativo più diffuso al mondo, facendo anche una grande concorrenza al mercato Apple. Tant'è che il 62,94% del mercato mondiale è marchiato Android, contro il 37,16% che comprende iOS e tutti gli altri sistemi operativi [12]. Il sistema Android è basato sul Kernel di Linux, esso segue una distribuzione embedded Linux sotto i termini della licenza libera *Apache 2.0*. Viene concesso in licenza da Google, ma per poter essere al passo con i tempi deve essere aggiornato con regolarità.

## 1.2 Storia

Andy Rubin, informatico statunitense, nell'Ottobre del 2003, insieme ad altri due soci, fondarono l'Android Inc., una società per lo sviluppo di dispositivi cellulari.

Nell'estate del 2005 Google acquista la società con l'intento di entrare nel mercato dei dispositivi mobili. Fu allora che il team dei soci incominciò a sviluppare un sistema operativo per questo tipo di dispositivi basato sul kernel Linux.

Dopo due anni di sviluppo Android venne presentato al pubblico alla OHA (Open Handset Alliance). Questa è un'organizzazione di diverse compagnie operanti nel settore dei dispositivi mobili con l'intento di creare standard aperti. Attualmente

è composta da 35 membri, come Google, Vodafone, NVIDIA Corporation, Acer ecc.) [12].

Il primo dispositivo mobile lanciato dalla compagnia fu l'HTC Dream. Esso fu presentato il 23 settembre 2008 a New York al prezzo di 179 dollari negli Stati Uniti e 450€ in Italia (Figura 1.1).



**Figura 1.1.** Primo telefono Android lanciato sul mercato

Il dispositivo presentava nuove caratteristiche innovative quali:

- schermo tattile da 3,2 pollici con risoluzione di 320x480 pixel;
- supporto per la connettività 3G UMTS/HSDPA a 7,2 Mbit/s;
- tastiera QWERTY.

Parallelamente al mercato Apple, Android divenne sempre più influente e diffuso nel mondo. Esso, apportando migliorie a livello prestazionale e risolvendo problematiche di sicurezza (con i successivi aggiornamenti), incrementò le vendite dei prodotti esponenzialmente. Altro passo importante per la crescita della famiglia Android fu il lancio del Samsung Galaxy S e del Nexus S (anno 2010). Proprio grazie a questi ultimi dispositivi, secondo Wikimedia Foundation, nell'agosto 2011 il sistema operativo Android aveva una diffusione tra tutti i dispositivi mobili pari al 22,94% [3].

Una caratteristica delle release o aggiornamenti delle versioni dalla 1.5 in poi è quella che la successione delle stesse ha seguito una convenzione alfabetica di nomi di dolci. La prima versione venne chiamata Petit Four (Versione 1.0-1.1), la 1.5 venne chiamata Cupcake, la 1.6 Donut, la 2.1 Eclair, la 2.2 Froyo (yogurt gelato), la 2.3 Gingerbread (pan di zenzero), la 3.0 Honeycomb, la 4.0 Ice Cream Sandwich, la 4.1 Jelly Bean, la 4.4 Kit Kat in seguito a un accordo con la Nestlé poi la 5.0 Lollipop. Il 5 ottobre 2015 toccò alla 6.0, con il nome di Marshmallow. Il 30 giugno 2016 venne annunciato ufficialmente il nome della versione successiva che il 22 agosto apparve sugli smartphone Android con il nome di Nougat (torrone). Il 22 agosto 2017 venne rilasciato Android 8.0 Oreo, previa concessione di Mondelēz, mentre il

6 agosto 2018 è stato pubblicato Android 9.0 Pie. Il 3 settembre 2019 è stata la volta di Android 10, conosciuto, in fase di sviluppo, come Q: la prima versione di Android a non portare il nome di un dolce, mentre android R, cioè Android 11, è stato pubblicato l'8 settembre 2020 [12].

## 1.3 Sistemi Operativi

In informatica, un sistema operativo (abbreviato comunemente con i termini SO oppure OS, dall'inglese operating system) è un software di base adibito a gestire le risorse hardware e software di un computer. In altre parole, esso fa da intermediario tra l'utente e il computer e, al tempo stesso, permette al computer di gestire più applicazioni software specifiche. Un sistema operativo, quindi crea una interfaccia che si frappone tra le due parti in modo che le richieste dall'utente vengano poi soddisfatte [15].

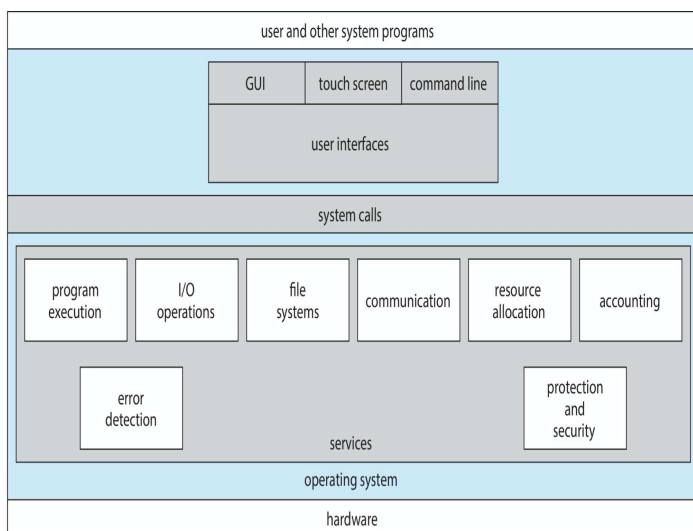
Esso è costituito da una serie di elementi quali:

- *Interfaccia*: svolge un lavoro da intermediario tra utente e hardware.
- *Allocatore di risorse*: gestisce ed alloca in modo efficiente le risorse.
- *Programma di controllo* (*Program Counter*, abbreviato *PC*): si occupa del controllo dell'esecuzione dei programmi utente e delle operazioni dei dispositivi di Input/Output (I/O).
- *Kernel*: Insieme di funzioni che gestiscono le funzionalità fondamentali e permettono un accesso controllato all'hardware.

Possiamo suddividere i sistemi operativi in due grandi categorie, ovvero *Batch* e *Multitasking*. I sistemi *Batch* hanno la caratteristica che possono eseguire un solo lavoro (job). In questo caso, la CPU è interamente dedicata al job. Inoltre il sistema operativo è situato in memoria.

I sistemi *Multitasking*, invece, hanno la caratteristica che possono eseguire più job contemporaneamente, quindi, questa volta, la CPU viene suddivisa tra di essi. Anche in questo caso, però, il sistema è situato in memoria.

Il sistema operativo, quindi, nel suo complesso, ha una struttura generale come quella in Figura 1.2.



**Figura 1.2.** Struttura del Sistema Operativo

Il Kernel è il software che ha il compito di fornire ai moduli che compongono il sistema operativo e ai programmi in esecuzione le funzioni fondamentali ed un accesso controllato all'hardware. Lo Scheduler dei processi, invece, è quel componente del sistema operativo che si occupa di decidere quale processo va mandato in esecuzione. Esso, quindi, si occupa di far avanzare un processo e, contemporaneamente di interromperne un altro, realizzando, così, un cambiamento di contesto (context switch).

Il concetto di multiprogrammazione è pensato per mantenere la CPU occupata il più possibile, ad esempio avviando un processo mentre un altro è in attesa del completamento di un'operazione di I/O. Questo perché generalmente i processi non hanno continuamente bisogno della CPU, spesso si mettono in attesa di risorse o file per poi proseguire. Essi, quindi si mettono in stato di "attesa" per poi riprendere una volta in possesso delle risorse. Perciò lo scheduler gestisce e coordina la lista dei processi che intendono utilizzare la CPU.

Esistono vari *algoritmi di scheduling* che permettono di scegliere nella maniera più efficiente possibile quale processo far proseguire. I migliori scheduler hanno complessità  $O(1)$  (2).

Il File system è lo strumento tramite cui i dati vengono memorizzati ed archiviati all'interno di una memoria, sia essa un hard disk o un CD-ROM. Esistono diversi file system, creati per vari sistemi operativi o dischi di memorizzazione. In linea di massima, però, le tipologie di file system sono due: una per le unità locali, che si occupano dell'organizzazione dei dati all'interno di un disco, e una per i file che devono essere condivisi tra più computer. Perciò possiamo evidenziare 4 strutture del sistema operativo:

- *Gestore della memoria*: il quale si occupa di gestire la memoria e di distribuirla tra i processi attivi. Esso, appunto, è importante per sapere come suddividere e distribuire la memoria evitando sovrapposizioni tra i processi.
- *Gestore delle periferiche*: esso si occupa di gestire tutte quelle funzioni di Input e di Output adibite al controllo e all'esecuzione di dispositivi di Input/Output, quali mouse, stampanti, fax, tastiere, monitor, dispositivi USB.
- *Protezione della memoria*: esso è un sistema che impedisce ad un processo di corrompere la memoria di un altro processo in esecuzione contemporaneamente sullo stesso computer.
- *Spooler di stampa*: gestisce il flusso di documenti inviati alla stampa

### 1.3.1 Versioni

I sistemi operativi più diffusi sono Windows di Microsoft (che ha soppiantato l'MS-DOS), MacOS di Macintosh, Linux e Unix, utilizzati soprattutto per i server o in ambito di ricerca. Esiste una sostanziale differenza tra i vari sistemi: Linux, Unix e MS-DOS di Microsoft sono sistemi operativi a linea di comando. Essi sono privi di grafica e tutti i comandi vengono digitati con la tastiera. Ovviamente sono sistemi complessi e dotati di scarsa usabilità. Ecco perché sono state create delle interfacce grafiche su cui essi si appoggiano. Esistono interfacce user-friendly per Unix e Linux e per tutte le versioni di Windows precedenti al 98.

Il primo sistema Unix nasce nel 1969 [11]; nel corso dei decenni divenne una delle piattaforme più utilizzate dai ricercatori statunitensi. Dopo diverse “personalizzazioni” nel 1983 nasce il progetto GNU seguito poi, da Linux. Linux nasce con l'intento di essere un sistema completamente libero ed alternativo a Unix. Per questo motivo i sistemi GNU/Linux sono ispirati ad Unix ma, di fatto, non sono di sua derivazione diretta.

Windows 98 (e i successivi, dal 2000 all'XP, a Windows Vista) e MacOS di Macintosh sono, invece, sistemi operativi ad interfaccia grafica. Tutte le operazioni si svolgono con il mouse, tramite una visualizzazione a finestre. Si tratta di sistemi che, a differenza di quelli a linea di comando, sono molto più facili da usare, e quindi accessibili a tutti. Mentre il sistema operativo sviluppato dalla Apple Inc. è nato nel 2001 con l'architettura di un sistema operativo di derivazione Unix della famiglia BSD, il sistema Windows è frutto della Microsoft Corporation, azienda che lo lanciò nel 1985 come ambiente grafico per i sistemi operativi MS-DOS e PC-DOS (per personal computer compatibili IBM).

## 1.4 Funzionalità di Android

### 1.4.1 Interfaccia

Android utilizza il concetto di manipolazione diretta per implementare la sua interfaccia.

Il canale di ingresso è mono e multi-touch. Esso utilizza tocchi, pizzichi e swipe sullo schermo per manipolare gli oggetti visibili. L'interfaccia, quindi, per mezzo

di sensori hardware manipola questi ultimi e produce una risposta immediata. Dei sensori come giroscopi, accelerometri o sensori di prossimità vengono utilizzati da applicazioni per rispondere alle interazioni dell'utente. Uno dei componenti classici di Android è il Launcher, il quale si occupa di gestire la schermata principale, le shortcut, le app drawe, la barra inferiore e la barra di stato.

### 1.4.2 Kernel Linux

Sviluppato nel 1991 da un gruppo di studenti finlandesi, il kernel di Linux fu uno dei più importanti software open source della famiglia Linux. Interamente free e disponibile a tutti, venne rilasciato sotto la licenza libera GNU ed è ampiamente personalizzabile. Come gli altri sistemi operativi, anche esso si occupa di amministrare e gestire gran parte dello sviluppo. La caratteristica che, però, lo contraddistingue è che tutti i driver devono avere una parte eseguita in kernel mode. Questi lavorano nel kernel space e hanno accesso all'hardware. Il kernel Linux di Android, inoltre, ha un'architettura creata da Google e permette modifiche al di fuori del ciclo di sviluppo del kernel. In Figura 1.3 viene riportata una struttura base del kernel Linux con i suoi componenti.

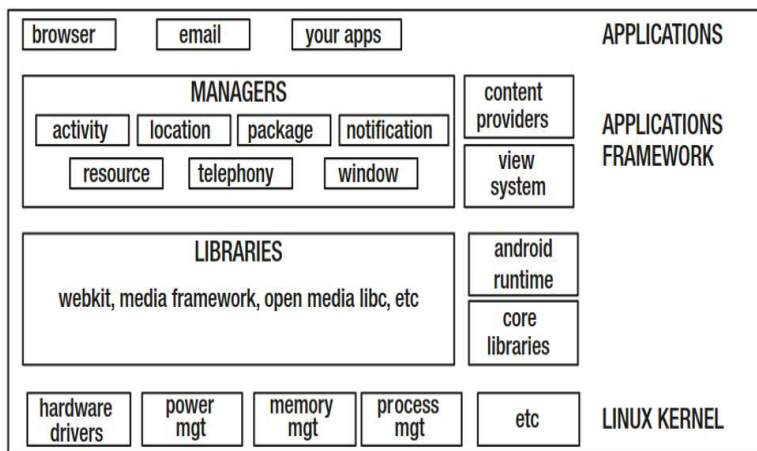


Figura 1.3. Struttura del Kernel Linux con i suoi componenti

## 1.5 Sviluppo

### 1.5.1 Android Studio

Android Studio è un ambiente di sviluppo integrato, più nello specifico un IDE per la programmazione di progetti Android. Percorrendo a ritroso negli anni, ancor prima del lancio di Android Studio, gli sviluppatori usavano strumenti a riga di comando con i quali non era affatto semplice programmare.

Ciò che fornisce un ambiente di sviluppo integrato (IDE), come le evidenziazioni del codice, il setup dei progetti e il debugging, sono funzionalità fondamentali. Infatti, nel 2008, fu rilasciato l'ambiente Eclipse, ambiente dominante per lo sviluppo di applicazioni Java, ma usato anche dagli sviluppatori Android. Eclipse, infatti, è stato l'ambiente predominante per Android fino all'anno 2012.

Nel 2013 venne rilasciato Android Studio, che diventò l'ambiente ufficiale di Android. Completamente gratuito, esso è caratterizzato dall'Android SDK Platform Tools, utile per il debugging e per applicazioni che utilizzano database, l'Android SDK Tools, l'Android Emulator, per simulare l'avvio e il running delle applicazioni sviluppate, il Support Repository, che consente di scrivere codice per Android Wear, TV o Google Cast, e un HAXM Installer, ovvero un'acceleratore per l'emulatore.

Entrando più all'interno dell'IDE possiamo evidenziare diversi componenti fondamentali (Figura 1.4). Tra questi citiamo:

- *Editor Window*: la finestra di lavoro più importante. Tramite essa è possibile creare e modificare i file di progetto.
- *Navigation Bar*: permette di navigare tra i vari file di progetto.
- *Toolbar*: barra per salvare i file, eseguire l'applicazione, aprire il gestore AVD (Android Virtual Device), aprire l'SDK Manager, annullare/ripristinare le azioni.
- *Tool Windows*: strumento per accedere a task specifici, come logcat, annotazioni TODO.
- *Tool Windows Bar*: contiene i singoli pulsanti per attivare specifiche tool windows.
- *Status Bar*: barra che visualizza lo stato del progetto, con i relativi messaggi per lo sviluppatore.

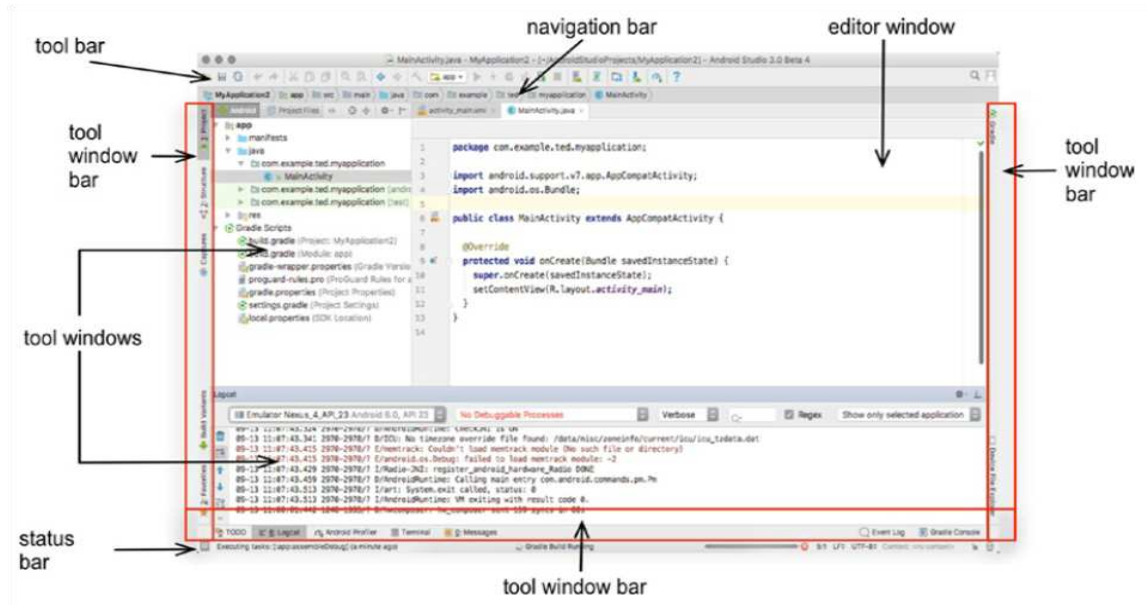


Figura 1.4. Panoramica di Android Studio

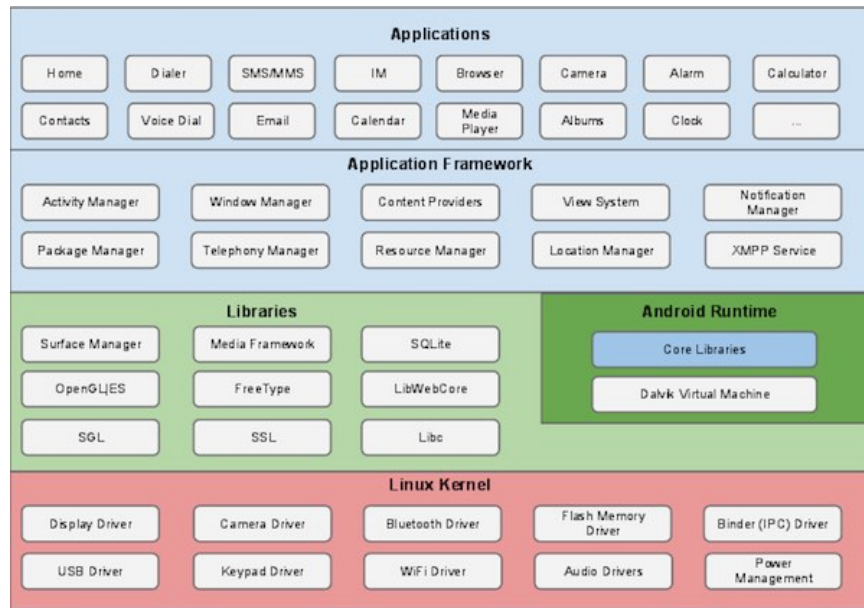
## 1.5.2 Architettura Android

L'architettura di Android (Figura 1.5) si presenta come uno stack con più livelli. Nel primo livello troviamo le “Applications”, ovvero le applicazioni che verranno poi installate. Esse includono i contatti, il browser, i giochi etc. Sotto al livello delle Applications troviamo l'Application Framework di Android, ovvero un'insieme di API e altri componenti che si occupano dell'esecuzione di ciascuna applicazione Android. Le librerie, infatti, vengono utilizzate da questo livello, o più nello specifico da una serie di componenti che formano l'Application Framework.

I componenti fondamentali che formano l'Application Framework sono:

- *Activity Manager*: si occupa di gestire e organizzare le varie schermate di un'applicazione in uno stack a seconda dell'ordine di visualizzazione delle stesse sullo schermo dei diversi dispositivi. Più genericamente, l'Activity Manager è lo strumento fondamentale attraverso il quale l'utente interagisce con l'applicazione.
- *Package Manager*: gestisce il ciclo di vita delle applicazioni nei dispositivi, come il processo d'installazione, aspetti grafici dell'applicazione, diverse activity o aspetti di sicurezza.
- *Window Manager*: permette la gestione delle finestre delle varie applicazioni sullo schermo del dispositivo.
- *Telephony Manager*: permette l'interazione con le funzionalità di un telefono, come chiamate, videocall, etc.
- *Content Provider*: gestisce la condivisione di informazioni tra i vari processi.
- *Resource Manager*: ottimizza le risorse per mezzo di API.
- *View System*: gestisce la renderizzazione dei componenti e degli eventi associati.





**Figura 1.5.** Architettura interna di Android

- *Location Manager*: permette l'utilizzo di servizi di localizzazione, come Mappe e GPS.
- *Notification Manager*: consente l'utilizzo di notifiche.

Un altro elemento fondamentale per l'architettura di Android è l'Android Runtime. Esso fornisce il componente chiave, denominato Dalvik Virtual Machine, simile alla Java Virtual Machine, ma progettato e ottimizzato specificamente per Android. Dalvik Virtual Machine permette, infatti, di utilizzare le funzioni del kernel di Linux in Java, come la gestione della memoria e il multi-threading.

Un altro livello fondamentale per la struttura di Android è quello contenente un set di librerie native. Tra le principali librerie abbiamo:

- *Media Framework*: libreria per la gestione dei diversi CODEC per i diversi formati di riproduzione audio e video.
- *OpenCORE*: libreria che supporta la riproduzione e la registrazione di formati audio e video, compresi file di immagini.
- *Surface Manager*: libreria per l'accesso alle funzionalità del display e il coordinamento delle diverse finestre che le applicazioni desiderano visualizzare sullo schermo.
- *Open GL ES*: libreria usata da Android per la grafica 3D, che permette anche l'accesso a funzionalità di un eventuale acceleratore grafico hardware.
- *Scalable Graphics Library (SGL)*: libreria in C++ che viene utilizzata da Window Manager per la visualizzazione grafica 2D.
- *FreeType*: libreria per il rendering di bitmap e vector font. Essa, utilizza un insieme di API per ciascun tipo di font.

- *SQLite*: libreria utilizzata per la gestione dei dati. Per mezzo di un DBMS relazionale permette di memorizzare e operare sui dati con semplici istruzioni.
- *WebKit*: libreria per il browser engine integrato, open source e fondato sulle tecnologie HTML, CSS, JavaScript e DOM.
- Libc (System C library): è un'implementazione della libreria standard C system (libc).
- Secure Socket Layer (SSL): libreria dedicata alla sicurezza di tutto l'ambiente Android. Essa consiste in un insieme di protocolli crittografati che consentono la comunicazione sicura.

In fondo ai livelli, alla base dell'architettura di Android, troviamo il Kernel di Linux nella Versione 2.6. Esso viene utilizzato da Android per i servizi del sistema centrale, come sicurezza, esecuzione, gestione della memoria e driver model. Oltre alle funzionalità da sistema operativo, il Kernel Linux offre la possibilità di gestire le periferiche multimediali del display, della connessione Wi-Fi e dell'alimentazione.

### 1.5.3 App Android

Le App in Android e le applicazioni scritte per il desktop non sono esattamente uguali. Esse potrebbero avere alcune somiglianze per quanto riguarda le apparenze, ma strutturalmente sono molto diverse. Le applicazioni desktop sono praticamente autonome.

I file `.exe` contengono tutte le routine e le subroutine necessarie al loro interno. Di tanto in tanto può essere necessaria una qualche libreria caricata dinamicamente, ma praticamente il file `.exe` è autosufficiente. Un'applicazione Android non è un pacchetto monolitico come un file `.exe` in Windows. Essa è un pacchetto di componenti e altre risorse liberamente correlati o accoppiati in modo lasco, tenuti insieme all'interno di un file APK (file del pacchetto Android). Come si può vedere dalla Figura 1.6 l'APK file contiene diversi elementi.



Figura 1.6. File APK di un'App Android

Esso, inoltre, ha estensione `.apk`; perciò è un particolare tipo di archivio Jar. L'APK file costituisce l'intera applicazione Android ed è ciò che l'utente scarica nel momento dell'installazione. Si possono individuare 4 componenti fondamentali

all'interno del file: *Resources*, *Manifest*, *Intents* e *Component*. Le *Resources* contengono le varie risorse incluse nell'applicazione, come audio, video o immagini. Il *Manifest* è un file XML che definisce cosa può fare l'applicazione. Esso stabilisce quali richieste essa può effettuare. Inoltre, esso contiene anche informazioni legate al progetto, come il nome dell'App, quale view verrà visualizzata al lancio e che tipo di componenti ha. I *Component* sono gli elementi che rendono possibile l'accoppiamento lasco in Android. Essi sono gli elementi chiave per un'applicazione Android. Essi si suddividono in *Activity*, *Service*, *Broadcast Receiver* e *Content Provider*. Più specificamente abbiamo:

1. *Activity*: elemento figlio dell'Application Framework che consente di comporre l'interfaccia utente dell'applicazione. Lo sviluppatore, quindi, per mezzo di essa costruisce l'interfaccia grafica dell'App.
2. *Service*: elementi che operano in background.
3. *Content Provider*: permettono la condivisione sicura tra le varie applicazioni.
4. *Broadcast Receiver*: vengono utilizzati se si desidera eseguire una qualche logica di programma nella propria applicazione come risposta ad eventi generati dal sistema Android o da altre applicazioni.

Infine gli *Intent* sono elementi che collegano più component. Essi mettono in relazione più component e permettono lo scambio di informazioni tra questi ultimi.



## Il Baratto nella storia

*In questo capitolo introduciamo il concetto di baratto. Nella prima parte verrà descritta la storia del baratto, quindi, verrà illustrato da dove è nato questo concetto e come si è evoluto nel corso del tempo. Nella seconda parte, invece, verranno illustrati gli elementi che caratterizzano il baratto e come esso si presenta al giorno d'oggi.*

### 2.1 La Storia

Sin dalle fasi primordiali della civiltà, il baratto è stata la prima forma storica di scambio commerciale di beni. Esso ebbe le sue origini nell'antico Egitto, dove i venditori e i compratori si accordavano sul prodotto da vendere e su quello da ricevere.

Anche nella prima fase dell'Alto Medioevo, quando ancora l'economia non era ben sviluppata e industrializzata, il baratto era la forma di mercato utilizzata dai proprietari terrieri e dalle piccole "società" dell'epoca. Esse, infatti, operavano secondo il principio dell'economia del dono e del debito, ovvero, un sistema economico fondato sul valore d'uso degli oggetti e dei beni (economia, dunque, non-monetaria). L'economia del dono si contrappone all'economia tradizionalmente intesa, definita "economia di mercato" o "economia mercantile", che si basa, invece, sul valore di scambio (o valore commerciale).

Il "dono" è uno scambio reciproco che ha alcune caratteristiche (per quanto esse siano delle convenzioni e non delle leggi scritte):

- obbligo di dare;
- obbligo di ricevere;
- obbligo di restituire di più di quanto si è ricevuto.

Sono stati rinvenuti documenti simili ad elaborati sistemi di credito, come quello dei Sumeri (3500 a.C.), in periodi molto anteriori alla prima coniazione di monete (800 a.C.), i quali riepilogavano tutti gli scambi avvenuti. Il baratto, quindi, era il mezzo con cui ogni organizzazione sociale faceva mercato.

Le organizzazioni meno evolute eseguivano la produzione e lo scambio di prodotti naturali o di animali cacciati per sostenere le proprie famiglie. Esso, perciò, era il

fulcro di un'economia di sussistenza. Ciò che veniva scambiato, erano prodotti frutto di raccolta, dell'agricoltura rudimentale e della caccia. In Figura 2.1 è illustrata una rappresentazione di baratto tra un indiano e un cacciatore.



**Figura 2.1.** Baratto tra indiano e cacciatore

“Gli economisti hanno ipotizzato che la moneta sia subentrata al baratto per superare il principale inconveniente di tale sistema di scambio, ossia la necessità di una doppia coincidenza dei bisogni: chi desidera un prodotto deve trovare una persona la quale sia possessa quel prodotto, sia abbia interesse a qualche prodotto che egli stesso possiede.” [4].

Il baratto, inoltre, richiede che le preferenze e le necessità di scambio siano simili tra le parti coinvolte e siano, al tempo stesso, compatibili con i beni disponibili per lo scambio. Fu per questo motivo che, attorno attorno alla metà del VII secolo a.C., per superare questi limiti, venne individuato un unico bene da utilizzare per acquistare altri beni e che per sua natura sia generalmente accettato da tutti: la moneta.

## 2.2 Il Concetto del Baratto

In economia, il baratto è “un’operazione di scambio bilaterale o multilaterale di beni o servizi fra due o più soggetti economici (individui, imprese, enti, governi, ecc.) senza uso di moneta” [2]. Dal punto di vista del diritto civile, il baratto viene classificato sotto la denominazione di permuta.

Il valore dei beni messi a disposizione nello scambio viene considerato equivalentemente fra le parti, senza ricorrere ad unità monetarie. Il valore di equivalenza, infatti, si raggiunge attraverso una valutazione delle merci scambiate. In Figura 2.2 viene illustrata una rappresentazione del concetto di scambio.



**Figura 2.2.** Rappresentazione schematica di uno scambio di beni

Dal punto di vista economico, il valore delle merci di scambio, corrisponde al punto di incontro fra la domanda e l’offerta.

Esistono due tipi di baratto: *baratto semplice* e *baratto multiplo*. Il baratto viene definito *semplice* quando entrambe le parti desiderano procurarsi il bene o il servizio che ricevono in cambio del bene o del servizio ceduto. Mentre esso viene definito *multiplo* quando un soggetto cede un bene o un servizio ricevendone in cambio un altro bene o servizio che non desidera avere, ma che scambia per ottenere quanto desiderato. Generalmente, nel caso di baratto multiplo, lo scambio viene definito “desiderato” per il suo valore di scambio e non per il suo valore d’uso.

### 2.2.1 La Moneta

La prima moneta metallica risale al VI secolo a.C. . Essa fu coniata da Creso, re di Lidia. Dopodichè, l’Impero Persiano e le città greche, attraverso Creso, appresero l’utilizzo della moneta. Furono proprio i greci a introdurre l’uso della moneta nel Mediterraneo Occidentale. Un esempio di moneta viene rappresentato nella Figura 2.3.



**Figura 2.3.** Moneta Attica con la Dea Atena e la civetta

Nell’Europa Occidentale, durante il Medioevo, si affermò il bimetallismo. Secondo questo sistema le monete auree erano quelle più di valore. Le monete d’argento, invece, venivano utilizzate per le transazioni commerciali. Per ultime, le monete di rame, venivano utilizzate per la vendita quotidiana di merci al dettaglio. In seguito alla scoperta delle miniere di oro e di argento, la moneta divenne un lingotto recante la firma del Re o della Repubblica.

In questi anni, proprio grazie a queste scoperte, la “zecca” divenne via via un organo fondamentale per la gestione di queste risorse. Essa, quindi, gestiva il flusso e la valuta della moneta.

Al giorno d’oggi, percorrendo a ritroso i secoli di storia, possiamo classificare diversi tipi di moneta:

1. *Metallica*;
2. *Cartacea*;
3. *Bancaria*;
4. *Elettronica*;
5. *Commerciale*;
6. *Scritturale*.

La moneta, in qualsiasi sua forma, assolve tre diverse funzioni. Essa, in primis, è un mezzo di scambio perchè è lo strumento attraverso il quale si può acquisire un bene o un servizio. Al tempo stesso, però, la moneta è misura di valore perchè associa ad ogni bene o servizio, un preciso valore quantitativo. Infine, essa ha funzione di riserva di valore perchè è un bene che tende a conservare il suo valore [10] a discapito di qualsiasi altra risorsa che, con il tempo, perde il suo valore.

### 2.2.2 La Moneta Fisica

Per più di duemila anni, le monete metalliche sono state fuse con metalli preziosi. Le banconote (introdotte in Cina nel XVII secolo) furono utilizzate in Europa solo nel XVIII secolo. In Figura 2.4 vi è la prima banconota cinese che è stata pervenuta ed emessa sotto la dinastia Ming (1368-1398).



**Figura 2.4.** Banconota cinese della dinastia Song

Sebbene il valore intrinseco delle monete sia determinato dal valore dei metalli, il valore intrinseco della cartamoneta è nullo: esse rappresentano solo una semplice “promessa di pagamento”.

Proprio per dare il valore (reale) della cartamoneta (che è più facile da gestire rispetto al metallo), nel XVIII secolo, molti paesi introdussero il gold standard della moneta: l’emissione della cartamoneta viene effettuata dalla banca centrale (con il monopolio di emissione) e il valore nominale delle monete è pari al valore delle sue riserve auree. Questo sistema terminò ufficialmente il 15 agosto 1971, quando Richard Nixon, presidente degli Stati Uniti d’America, annullò la possibilità di convertire i dollari statunitensi in oro.

Se la valuta fisica fosse l’unico mezzo di pagamento, sorgerebbero enormi problemi nel trasferire le grandi quantità di fondi necessarie per operazioni commerciali



a lunga distanza e di alto valore. In effetti, la valuta fisica è significativa per le transazioni di piccole e brevi distanze. Anche per superare questi limiti, già nel Medioevo, nacquero in Italia le prime banche di credito, seguite dalla prima forma di registrazione contabile dei saldi in valuta (versamenti e prelievi) e, soprattutto, la cartolarizzazione dei crediti. Essi sono i cosiddetti “vaglia”, che rappresentano i crediti in oro rivendicati dai depositanti, che possono essere convertiti in modalità di pagamento per transazioni commerciali da parte dei depositanti, evitando, così, i costi e i rischi associati al trasporto fisico della valuta.

Da quelle prime modalità di gestione dei depositi monetari, si sono sviluppate le attuali banche. Per introdurre il concetto di settore bancario, bisogna prima introdurre 4 elementi fondamentali:

- *Moneta legale (circolante)*: insieme di banconote e monete metalliche in circolazione in un determinato periodo nel sistema economico. Esse, devono essere accettate in pagamento e sottoposte a supervisione di Autorità che ne controllano il valore.
- *Moneta bancaria*: insieme di strumenti di pagamento creati utilizzando depositi bancari o crediti bancari. Essi sono rappresentati da forme tipiche: assegno bancario, assegno circolare, giroconto, disposizione di pagamento (bonifico), disposizione permanente (RID), carta di credito, carta di debito, moneta elettronica.
- *Base monetaria*: aggregato monetario che rappresenta lo stock di valuta (monete e banconote) e di depositi caratterizzati da diversi gradi di liquidità presenti in un sistema finanziario.
- *Offerta di moneta*: rappresenta la quantità di moneta in circolazione in un sistema economico in un determinato periodo di tempo.

### 2.2.3 La Moneta Virtuale

Il progresso della crittografia, insieme allo sviluppo di nuove tecnologie, hanno determinato un cambiamento radicale nell’economia. Specialmente nel settore finanziario, più precisamente nello scambio di beni e servizi, esso ha sviluppato una modalità del tutto digitale per le operazioni finanziarie.

Tra le più significative applicazioni della tecnologia digitale al settore finanziario, spicca la nascita e la diffusione delle “criptovalute” (o “valuta virtuale”).

Il termine di “criptovaluta” è composto da due parole: Cripto e Valuta. La moneta virtuale, quindi, è una valuta “nascosta”, ovvero visualizzabile/utilizzabile solo conoscendo un preciso codice informatico. La Figura 2.5 rappresenta una schematizzazione base del processo di creazione della criptovaluta.



**Figura 2.5.** Processo di generazione di una criptovaluta

La criptovaluta non esiste in forma fisica (anche per questo viene denominata “moneta virtuale”), ma si genera e scambia esclusivamente in modalità telematica. Essa ha la particolarità di essere gestita su database, quindi essa è completamente virtuale.

Il lavoro della rete informatica, su cui esse si appoggiano è attribuito a un sistema chiamato peer-to-peer. La valuta virtuale è gestita nel database e può essere scambiata solo online. Infatti, tutte le transazioni, ovvero il flusso di tutte le valute virtuali e tutte le funzioni della valuta, sono tracciate nel database. Una rappresentazione grafica di una valuta virtuale è rappresentato in Figura 2.6.



**Figura 2.6.** Esempio di valuta virtuale: Il Bitcoin

Queste monete sono, quindi, interamente digitali e utilizzate mediante il solo canale del web. Esse, solitamente, sono situate e gestite in portafogli virtuali, o scambiate tra utenti. Al giorno d’oggi, questi portafogli virtuali (Figura 2.7) prendono il nome di Wallet. Essi sono identificati da codici numerici univoci [5]. Ogni trasferimento è tracciato da appositi registri, denominati “blockchain”. Questi registri, mediante sistemi di crittografia, servono a garantire la sicurezza dei dati.



**Figura 2.7.** Portafoglio virtuale (Wallet)

Le principali criptovalute che si sono sviluppate nel corso degli anni sono:

1. *Bitcoin*: esso è una moneta virtuale creata nel 2009, sviluppata da un'organizzazione con lo pseudonimo di Satoshi Nakamoto. Diversamente dalle altre valute, il Bitcoin non ha dietro una Banca centrale, ma si basa fundamentalmente su due principi: un network di nodi, cioè di personal computer, che la gestiscono in modalità peer-to-peer e l'uso di una forte crittografia per validare e rendere affidabili le transazioni. Attualmente, i bitcoin disponibili in rete sono 21 milioni e dal 2009 il suo valore di mercato è passato da 0 a 1200 dollari. Secondo il Financial Times, nell'anno 2019 le transazioni totali avvenute hanno raggiunto i 10 miliardi di dollari, a discapito dei 150 milioni dell'anno precedente [13]. Attualmente, possedere 1 bitcoin equivale a possedere 31.931,17 €.
2. *Ethereum*: essa è una moneta virtuale che si è affermata come alternativa al Bitcoin. La sua blockchain è un sistema di gestione che permette di sviluppare applicazioni decentralizzate. La caratteristica principale di Ether è che esse sono luogo di scambio di valuta di qualsiasi tipo di asset. Attualmente 1 Ether corrisponde a 2.210,85€.
3. *Litecoin*: è una moneta simile al Bitcoin che, però, nel corso degli anni ha riscontrato una forte decrescita dovuta allo sviluppo di altre criptovalute concorrenziali. Attualmente 1 Litecoin corrisponde a 119,53€.
4. *Monero*: essa è una criptovaluta creata nell'aprile 2014 che si sviluppa su 4 caratteristiche: privacy, decentralizzazione, scalabilità e fungibilità. Al contrario di altre criptovalute, Monero si basa sul protocollo CryptoNight, un derivato dell'algoritmo CryptoNote. Attualmente 1 Monero equivale a 194,360 €.
5. *Ripple*: essa è una moneta che, contrariamente a molte altre, permette lo scambio di "token" senza il costo di commissioni. Esse, perciò, garantiscono un flusso di transazioni in modo completamente gratuito. Essendo una moneta creata nel 2004, attualmente il suo prezzo di mercato è sceso notevolmente.

La Figura 5.17 presenta una serie di statistiche relative alle principali criptovalute virtuali attualmente in commercio.

[Att. 1]

Principali criptovalute »











| Nome   | Simbolo | Prezzo (USD) | Cap. mercato | Vol. (24h) | Vol. totale | Var. (24 ore) | Var. (7 giorni) |
|--|---------|--------------|--------------|------------|-------------|---------------|-----------------|
|  Bitcoin      | BTC     | 39.039,3     | \$733,55B    | \$24,75B   | 33,84%      | +2,06%        | -1,51%          |
|  Ethereum     | ETH     | 2.631,53     | \$306,83B    | \$19,97B   | 27,31%      | +4,73%        | +14,93%         |
|  Tether       | USDT    | 1,0001       | \$62,00B     | \$50,38B   | 68,88%      | -0,02%        | -0,01%          |
|  Binance Coin | BNB     | 329,98       | \$55,37B     | \$1,22B    | 1,67%       | +2,40%        | +5,19%          |
|  Cardano      | ADA     | 1,360833     | \$43,64B     | \$1,90B    | 2,60%       | +1,46%        | +6,04%          |
|  XRP          | XRP     | 0,72185      | \$33,33B     | \$2,70B    | 3,69%       | +0,92%        | +1,86%          |
|  USD Coin     | USDC    | 1,000073     | \$27,46B     | \$2,78B    | 3,80%       | +0,02%        | +0,01%          |
|  Dogecoin     | DOGE    | 0,198011     | \$25,97B     | \$1,01B    | 1,38%       | +1,02%        | -3,32%          |
|  Polkadot     | DOT     | 18,513       | \$18,10B     | \$1,28B    | 1,75%       | +6,25%        | +28,30%         |
|  Uniswap      | UNI     | 22,5070      | \$13,15B     | \$466,40M  | 0,64%       | +5,11%        | +19,11%         |

Figura 2.8. Statistiche delle principali criptovalute del 04/08/2021 16:46

## 2.3 Lo Scambio tra Produttore e Consumatore

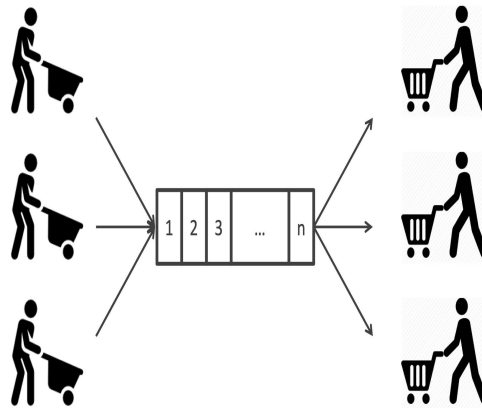
In qualsiasi momento del giorno, qualsiasi persona utilizza e consuma beni e servizi. Ciascun prodotto è il risultato dell'assemblaggio di pezzi ideati e prodotti in luoghi diversi, utilizzando materie prime e servizi provenienti da diversi posti nel mondo. In parole povere, gli scambi tra individui, imprese e organi rappresentano la quotidianità del mercato odierno.

Lo scambio tra due consumatori avviene se è vantaggioso per entrambi. Esso, quindi, avviene quando la distribuzione iniziale dei beni tra i due consumatori non è efficiente. Secondo Pareto, al contrario, un mercato nel suo complesso si può definire efficiente quando si arriva ad un'allocazione finale delle risorse per cui nessun individuo può migliorare la sua situazione (efficienza paretiana detta, anche, allocativa). Più in particolare, questa condizione viene raggiunta quando, utilizzando i fattori produttivi, si riesce a raggiungere un livello di output ottimale sia da un punto di vista quantitativo che qualitativo, con il minor costo possibile (efficienza produttiva), che rispecchi esigenze dei consumatori (efficienza nella composizione del prodotto) e quando le risorse sono allocate in maniera tale che nessun individuo vorrebbe cambiare la sua situazione perché ciò diminuirebbe la sua utilità (efficienza nello scambio) [15].

In economia, l'equilibrio generale della produzione e dello scambio si ottiene quando sia l'allocazione produttiva dei beni e sia lo scambio dei beni sul mercato sono in equilibrio economico-generale.

Dal punto di vista informatico, il problema del produttore-consumatore, rappresentato in Figura 2.9 è uno degli esempi classici di sincronizzazione tra i processi. Esso definisce due processi:

- *Produttore*: processo che genera e deposita i dati nel buffer continuamente.
- *Consumatore*: processo che utilizza e rimuove i dati dal buffer una volta che sono stati usati.



**Figura 2.9.** Schema del problema di sincronizzazione: produttore e consumatore

Il problema di sincronizzazione nasce, quindi, per assicurare che il produttore non crei nuovi dati quando il buffer è pieno, e che il consumatore non prenda dati quando il buffer è vuoto.

Esso è uno dei classici problemi di sincronizzazione tra processi, che si può perfettamente comparare e applicare ai problemi attuali.



## Analisi dei requisiti e progettazione dell'App

*In questo capitolo verranno presentati ed analizzati i dati usati nella presente tesi. Nella prima parte del capitolo, verranno descritti il funzionamento e la struttura dell'applicazione, presentando, quindi, le sue funzionalità, corredate dai diagrammi di flusso e dai diagrammi dei casi d'uso. Nella seconda parte, invece, si illustreranno i dati oggetto dell'applicazione e i database dove essi risiedono.*

### 3.1 Descrizione

L'applicazione propone di realizzare il concetto del “baratto”, il quale, come illustrato nel capitolo precedente, consiste nello scambio di 2 (o più) oggetti, a seguito di un accordo tra i proprietari. Essa si presenta inizialmente con una splash screen a seguito del quale l'utente si trova di fronte ad una classica form di Login. L'utente, inserendo i campi email e password corretti, avrà, poi, accesso al sito; altrimenti, nel caso in cui i dati non siano validi, egli riceverà un avviso di errore e dovrà ripopolare i campi. Nel caso in cui l'utente non sia in possesso di dati e che, quindi, non sia registrato, egli potrà procedere con la registrazione al sito tramite un'ancora presente nella finestra. Tale registrazione prevede l'inserimento di:

1. Nome;
2. Cognome;
3. Email;
4. Password (di almeno 6 caratteri).

Dopo aver effettuato il login, quindi, l'utente avrà accesso alla home page dell'applicazione. Essa è costituita da una lista di oggetti scambiabili proposti da altri venditori. Gli oggetti in questione saranno caratterizzati da:

1. Nome;
2. Venditore;
3. Prezzo.

Il “prezzo” viene mantenuto per una coerenza con una eventuale distribuzione dell'applicazione nello store, ma, non essendo presente nessun metodo di pagamento, è una voce inutilizzata, e rappresenta metaforicamente il valore dell'oggetto in vendita.

Interagendo con gli elementi della lista, l'utente potrà visualizzare i dettagli del prodotto e, nel caso in cui fosse interessato, formulare un'offerta di scambio attingendo alla lista di oggetti che egli ha proposto in vendita.

Come anticipato, l'utente potrà, quindi, inserire prodotti nel catalogo, compilando tutti i campi richiesti, quali: descrizione, foto, prezzo di vendita, posizione. Egli, inoltre, potrà poi visualizzare tutti i suoi oggetti attualmente in vendita nella finestra “Miei Oggetti”. L'utente interessato ad un particolare oggetto può inviare la richiesta di scambio interagendo con lo schermo e cliccando in “Formula Offerta”. Al click, l'utente che possiede l'oggetto in questione, riceverà una notifica, che può declinare o accettare nella sezione “Offerte Ricevute”. In equal modo, l'utente che ha proposto lo scambio può rivedere o annullare la sua proposta nella sezione “Offerte Inviare”.

Ad ogni offerta accettata, nella sezione “Riepilogo Offerte”, l'utente avrà la possibilità di visualizzare lo storico aggiornato degli scambi. Tale riepilogo permette, inoltre, di contattare il venditore e di cancellarlo dalla cronologia.

Nell'App, inoltre, è presente l'ancora “Logout”, la quale permette all'utente di scollegarsi ed essere riportato nella schermata di login, nel caso in cui avesse bisogno di effettuare nuovamente l'accesso. Per quanto riguarda il profilo dell'utente, quest'ultimo, tramite il click nell'item “Profilo”, avrà accesso alle informazioni del proprio profilo con il quale si è collegato. Egli, inoltre, potrà vedere nella finestra: l'immagine (modificabile), il nome Utente (modificabile cliccando sulla matita) e l'email. L'utente, qualora abbia necessità di cambiare la propria password di registrazione, può aggiornarla con un click sulla voce “Modifica Password”. Infine, egli ha la possibilità di cancellare il proprio account con la voce “Elimina Account”.

Nel caso in cui faccia accesso all'App un utente *amministratore*, egli avrà la possibilità di vedere statistiche sulla compravendita. Queste ultime saranno presenti nella sezione “Statistiche” e visualizzeranno:

- il numero di Accessi;
- il numero di oggetti scambiati;
- il numero di oggetti esposti;
- il numero di oggetti esposti al giorno;
- il flusso di “prezzo” giornaliero;
- il flusso totale degli oggetti scambiati.

### 3.1.1 Requisiti Funzionali

I requisiti funzionali sono presentati sotto forma di elenco di funzioni o servizi che il sistema deve fornire. Essi, inoltre, descrivono il comportamento del sistema di fronte a determinati input, e come quest'ultimo dovrebbe reagire in determinate situazioni. Indipendentemente dal fatto che si tratti di un utente o di un sistema, i requisiti funzionali possono essere formulati a diversi livelli di dettaglio; l'accuratezza delle specifiche, però, deve naturalmente essere sempre mantenuta.



Due caratteristiche importanti della specifica dei requisiti sono: *completezza* e *coerenza*. Un documento di specifica dei requisiti viene definito completo quando tutti i requisiti richiesti dagli utenti sono definiti. Esso, invece, viene definito consistente quando non vi sono requisiti in conflitto tra loro. Sebbene questi risultati siano facili da ottenere per i piccoli sistemi, è richiesto uno sforzo maggiore per i sistemi con un gran numero di requisiti. Infatti, in questo secondo caso, la difficoltà è dovuta alla maggiore occorrenza di errori ed omissioni, nonché alla presenza di un numero elevato di stakeholder con esigenze differenti, a volte contrastanti, che possono non essere valutate nella prima fase di scrittura delle specifiche.

In seguito riportiamo le principali funzionalità della nostra applicazione:

- effettuare il login o la registrazione;
- visualizzare tutti gli oggetti in vendita con aggiornamento in tempo reale;
- cercare gli oggetti nel catalogo per nome;
- formulare un'offerta di scambio, selezionando un oggetto dal catalogo ed un oggetto che l'utente ha inserito nello stesso;
- inserire un oggetto da vendere;
- ritirare dal catalogo un proprio prodotto;
- visualizzare i dettagli del proprio profilo con possibilità di modificare alcuni campi;
- visualizzare la lista dei propri oggetti in vendita;
- rappresentare tutte le offerte inviate e ricevute;
- accettare o rifiutare offerte ricevute;
- eliminare offerte inviate;
- contattare il venditore dopo l'accordo;
- eliminare dalla cronologia gli scambi conclusi;
- effettuare la statistica dei prodotti venduti;
- effettuare il logout.

### 3.1.2 Requisiti non Funzionali

I requisiti non funzionali rappresentano i vincoli e le proprietà/caratteristiche relative ad sistema, quali:

1. vincoli di natura temporale;
2. vincoli sul processo di sviluppo;
3. vincoli sugli standard da adottare.

I requisiti non funzionali, oltre al sistema software adottato, riguardano alcuni vincoli sul processo usato per sviluppare il sistema. Esempi di requisiti non funzionali sono: le specifiche degli standard di qualità da usare, la specifica sull'uso di un particolare strumento CASE, una descrizione della lavorazione che deve essere seguita. Essi, generalmente, non vengono applicati a singole funzioni o servizi, bensì all'intero sistema. Essi, inoltre, non fanno riferimento direttamente a specifiche funzioni fornite dal sistema, ma possono riguardare caratteristiche che si vuole applicare al sistema o vincoli ai quali quest'ultimo deve sottostare.

Le caratteristiche generalmente sono:

- affidabilità;

- tempi di risposta;
- occupazione in memoria.

I vincoli, invece, tipicamente sono:

- capacità dei dispositivi di I/O;
- rappresentazione dei dati utilizzata nelle interfacce del sistema.

I requisiti non funzionali sono strettamente vincolati alle esigenze degli utenti, alle politiche organizzative adottate, agli standard adoperati e alla necessaria modalità di interazione del relativo sistema con altri componenti. Più specificatamente, essi vengono classificati in:

1. *Requisiti di prodotto*: essi descrivono le caratteristiche del prodotto, quali usabilità, efficienza, affidabilità e portabilità,
2. *Requisiti organizzativi (o di processo)*: essi derivano dalle politiche e dalle procedure organizzative relative al cliente e allo sviluppatore,
3. *Requisiti esterni*: essi fanno riferimento a fattori esterni al sistema e al processo di sviluppo. Esempi di requisiti esterni sono requisiti legislativi e requisiti di interoperabilità.

I principali requisiti non funzionali dell'applicazione sono:

- la password deve avere almeno 6 caratteri, pena il reinserimento;
- l'inserimento di un prodotto viene effettuato solo dopo che l'utente ha inserito tutti i campi richiesti;
- nella visualizzazione dei dettagli di un prodotto nel catalogo, se l'oggetto è dello stesso utilizzatore dell'app, il "formula offerta" è sostituito da un "elimina prodotto".

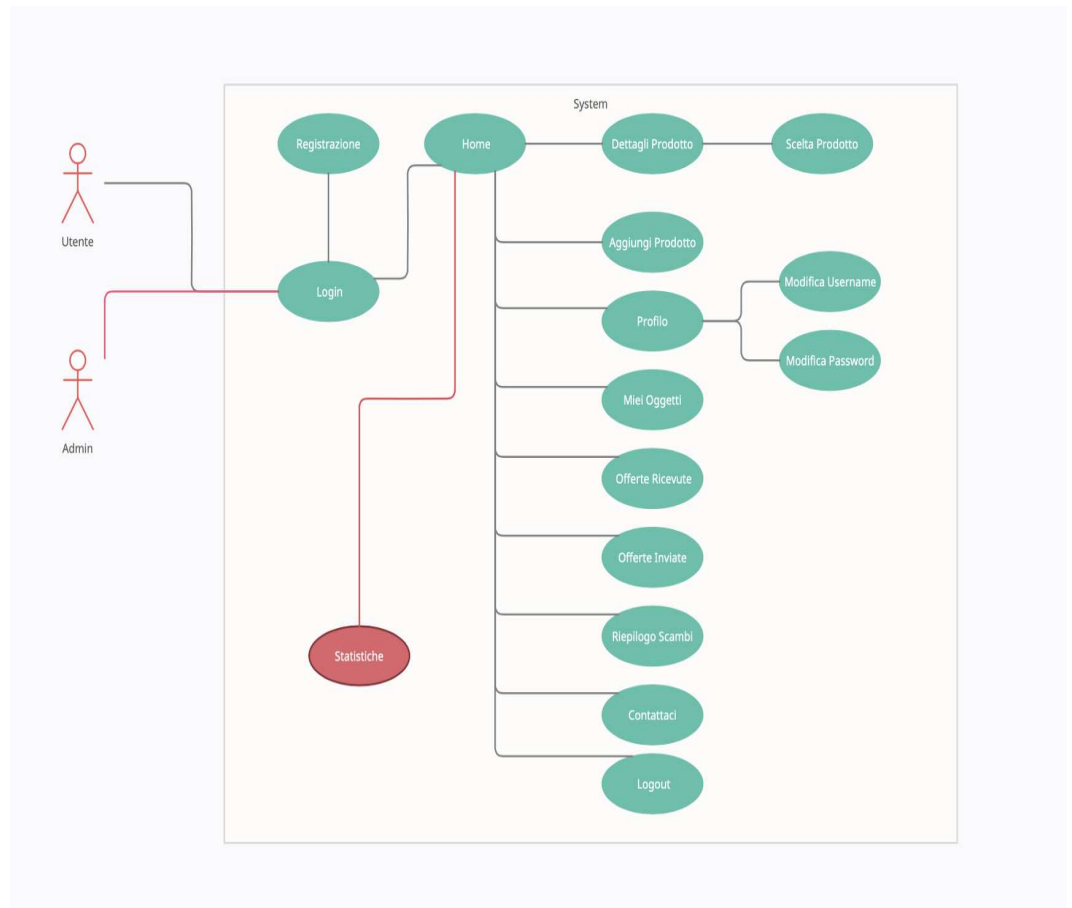
### 3.1.3 Diagrammi

In questa sezione verranno illustrati i diagrammi progettati per lo sviluppo e l'elaborazione dell'App.

#### Diagrammi dei Casi d'Uso

Sulla base dei requisiti funzionali, è stato progettato il diagramma dei casi d'uso. Questo diagramma, presentato in Figura 3.1, illustra come i due attori (utente e admin) possono operare sull'applicazione.

Al momento del Login, entrambi gli attori visualizzano una LoginScreen, la quale, poi, con le credenziali corrette, visualizzerà una Home Screen identica per i due attori. L'utente, quindi, da questa Home Screen può accedere ad una serie di funzionalità, citate nei requisiti funzionali. Tra queste ultime, citiamo, ad esempio, la visualizzazione dei prodotti, la loro scelta, il riepilogo delle offerte inviate, il riepilogo degli scambi, la modifica del profilo, la possibilità di contattare ed il Logout. Nel caso in cui abbia fatto accesso un utente "admin", però, egli ha anche la possibilità di visualizzare una sezione contenente statistiche legate al baratto degli oggetti.



**Figura 3.1.** Diagramma dei casi d'uso BartApp

### Diagrammi di Flusso

Il diagramma di flusso in Figura 3.2 presenta la struttura generale dell'applicazione. Esso, quindi, illustra come essa si pone di fronte ad ogni iterazione e richiesta dell'utente. Gli elementi da evidenziare nel diagramma sono *Home* e la sezione *Statistiche*. La *Home* è l'oggetto chiave su cui è stata implementata l'applicazione. Essa visualizza tutte le funzionalità e caratteristiche dell'App. La sezione *Statistiche*, invece, è una funzionalità extra per utenti che hanno gli opportuni permessi.

Il diagramma in Figura 3.3, invece, rappresenta la struttura base della funzionalità *Statistiche*. Esso è caratterizzato da elementi che operano con database per l'estrapolazione dei dati. Infatti, per la raccolta dei dati dell'applicazione, viene utilizzata una piattaforma denominata Firebase.

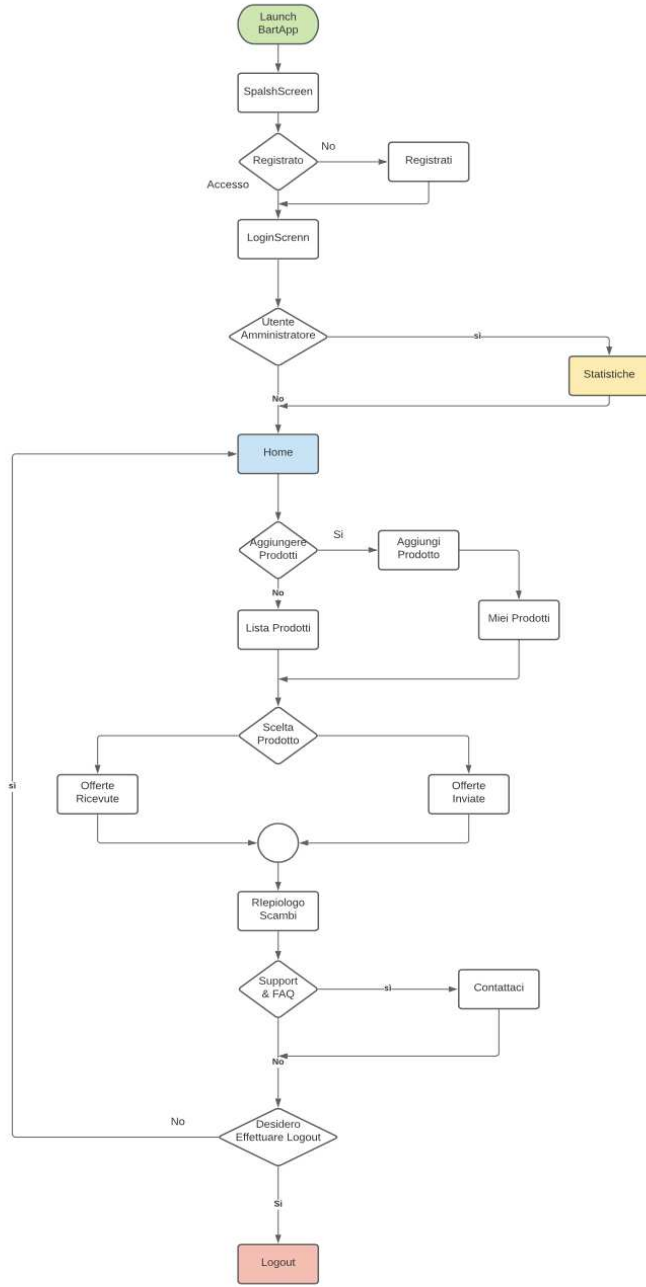


Figura 3.2. Diagramma di flusso BartApp

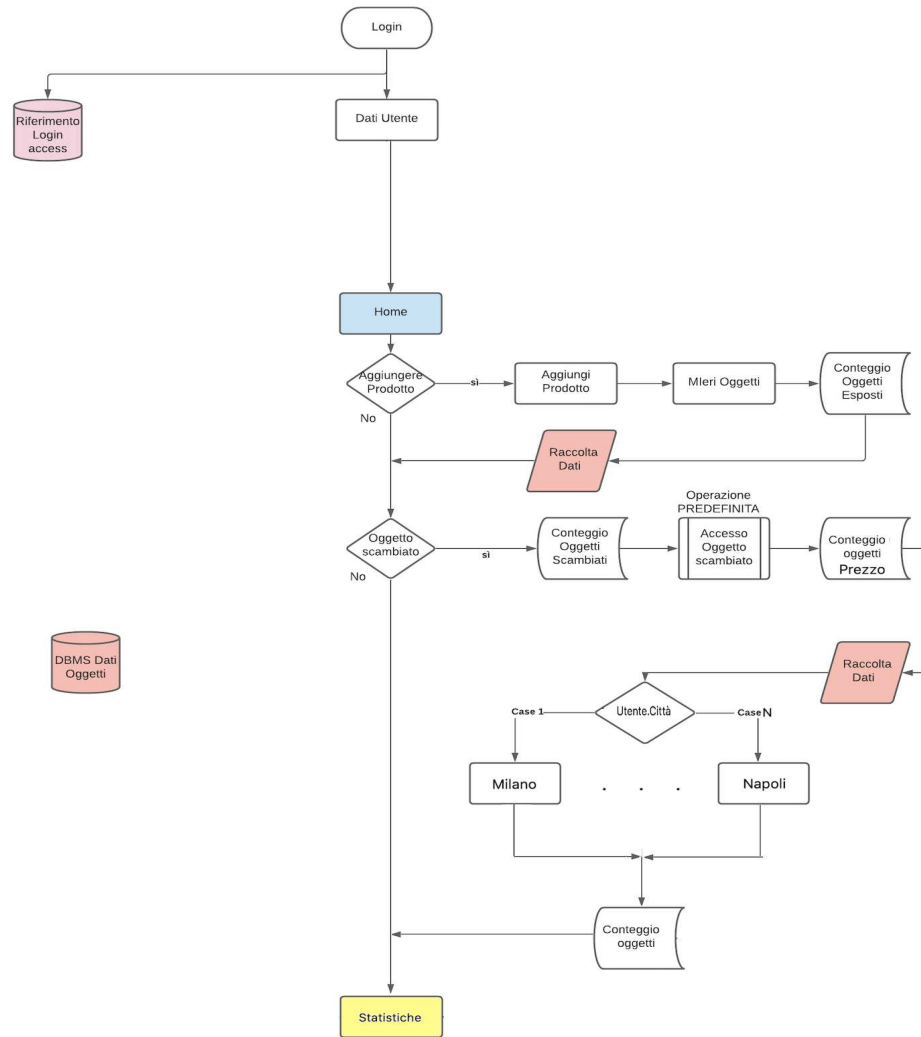


Figura 3.3. Diagramma di flusso delle Statistiche di BartApp

### 3.2 Struttura dei Dati

Le informazioni e i dati utilizzati dall'applicazione sono acquisiti da un DBMS denominato Firebase. Esso è una piattaforma per la creazione di applicazioni per dispositivi mobili e web. In particolare, Firebase è una struttura che offre la possibilità di memorizzare dati. Nel caso dell'applicazione a cui facciamo riferimento, questi ultimi, memorizzati in Firebase, presentano una struttura gerarchica rappresentata dal file json in Figura 3.4 .

```

{
  "oggetti" : {
    "-MN5gs5Fod7Kd1SV0vp5" : {
      "categoria" : 6,
      "dataCreazione" : "08/06/2021",
      "descrizione" : "il nuovo laptop della mela",
      "idUser" : "bsBa11jLhgeiLtBhUoDec4RsFg82",
      "nome" : "Macbook Air 2019",
      "nomeVenditore" : "a b",
      "posizione" : "Verona",
      "prezzo" : 1189
    },
    "-MN986vi3BSw_KUUXKda" : {
      "categoria" : 1,
      "dataCreazione" : "08/06/2021",
      "descrizione" : "Una Fiat Panda del 2006, 250000 km",
      "idUser" : "OHGCcnwJkMY8zyrs6V3BevF11EI3",
      "nome" : "Panda",
      "nomeVenditore" : "bbbbbb ccccc",
      "posizione" : "Roma",
      "prezzo" : 4225
    },
    "-MN98zngP_gKLy5nR_ac" : {
      "categoria" : 9,
      "dataCreazione" : "08/06/2021",
      "descrizione" : "una tazza da caffè",
      "idUser" : "tFosyvtBp0cceaabdAYTovXrN4X2",
      "nome" : "Tazza",
      "nomeVenditore" : "c d",
      "posizione" : "Napoli",
      "prezzo" : 3
    },
    "-MN99Wza7t7anbUqfbas" : {
      "categoria" : 6,
      "dataCreazione" : "09/06/2021",
      "descrizione" : "uno smartphone proveniente dalla Cina, ottimo",
      "idUser" : "0z0BAedqffT0pvpTaXi0E04b1NA3",
      "nome" : "Poco m3 64gb",
      "nomeVenditore" : "d e",
      "posizione" : "Milano",
      "prezzo" : 149
    }
  }
}

```

Figura 3.4. Struttura JSON dei dati

Il listato, inoltre, definisce la struttura (sotto forma di file `.json`) degli oggetti esposti dagli utenti. Esso ha la caratteristica di essere aggiornato in tempo reale; esso, quindi, permette di visualizzare gli oggetti con aggiornamenti live.

### 3.2.1 Dettaglio dei Dati

Analizziamo, ora, in dettaglio, la struttura del file `.json` che contiene i dati di nostro interesse.

- *"oggetti"*: esso è il riferimento a tutti gli oggetti. Rappresenta la radice degli oggetti figli.

- *"-MN5gs5FodZKdlSVOp5"*: rappresenta il codice univoco dell'oggetto.
- *"categoria"*: rappresenta la tipologia di oggetto. Esso viene utilizzato dallo sviluppatore per i meccanismi di ricerca.
- *"dataCreazione"*: rappresenta la data in cui l'oggetto è stato reso disponibile sull'App. Anch'esso è un attributo utilizzato dallo sviluppatore per l'analisi delle statistiche.
- *"descrizione"*: definito dall'utente, esso rappresenta una descrizione minima dell'oggetto proposto.
- *"idUser"*: è il codice identificativo dell'utente; anche esso è univoco ed associato ad un solo utente.
- *"nome"*: definisce il nome dell'oggetto esposto; Esso viene dichiarato dall'utente che propone l'oggetto.
- *"nomeVenditore"*: definisce il nome e il cognome del venditore associato all'oggetto esposto.
- *"posizione"*: rappresenta la città riferita alla posizione dell'utente che espone l'oggetto.
- *"prezzo"*: rappresenta il prezzo simbolico associato all'oggetto.

### 3.3 I Database

In informatica, il termine “database” fa riferimento ad un file di dati strutturato e archiviato in un computer. Esso ha lo scopo di semplificare l'aggiornamento e la gestione delle informazioni e di consentire ricerche complesse. Se queste ultime fossero condotte su archivi analogici tradizionali, impiegherebbero molto tempo e risorse. Perciò, il database è fondamentalmente sviluppato per l'archiviazione, il recupero e l'elaborazione delle informazioni ed è un archivio di dati organizzato [9]. Esso crea una struttura che consente di inserire e modificare rapidamente le informazioni richieste. Tale insieme di dati, una volta archiviato nel computer elettronico, può essere interrogato da qualsiasi terminale mediante l'apposita chiave di accesso. Le informazioni vengono memorizzate all'interno e possono essere gestite e interrogate dalle stesse applicazioni.

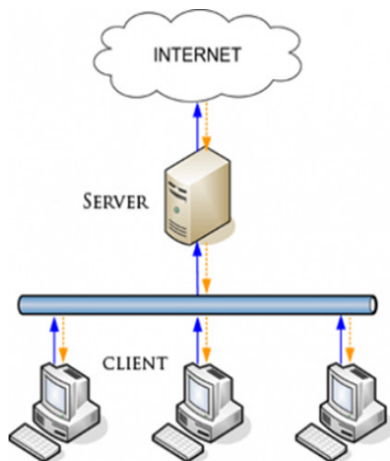
Per gestire i dati e le informazioni, i database utilizzano supporti di memorizzazione per garantire la persistenza dei dati. Il processore viene, invece, impiegato per l'elaborazione di questi ultimi. Fondamentalmente i componenti che strutturano tale meccanismo sono due: *client* e *server*.

Dal punto di vista informatico, un *client* rappresenta un componente hardware o software specifico che accede a una risorsa o a un servizio fornito da un altro componente (denominato *server*). Ad esempio, un computer che utilizza uno o più protocolli di rete per richiedere uno o più servizi da un server attraverso una rete di computer è un client hardware. In breve, il client può essere visto come un dispositivo hardware (come un computer, uno smartphone o un tablet) o come un software classico (come un browser o un programma di posta elettronica).

Con il termine *server* si indica una componente hardware o software che fornisce i dati richiesti da una o più altre componenti, dette *client*. In altre parole, un server non è altro che un computer e/o un programma in grado di rispondere alle richieste effettuate da altri computer e/o da altri programmi. A differenza di un client, un

server deve, però, essere capace di gestire tutti gli accessi, le risorse e i dati richiesti, per cui deve sia avere la potenza necessaria per assolvere a questi compiti, sia essere sempre in funzione, in modo tale da poter soddisfare, di volta in volta, le varie richieste dei client.

In Figura 3.5 è illustrato un classico esempio di modello Client-Server.



**Figura 3.5.** Modello Client-Server

### 3.3.1 Firebase

Firebase è una piattaforma “serverless” per lo sviluppo di applicazioni mobili e web. Esso è open source e, attraverso l’infrastruttura di Google e il suo cloud, fornisce una suite di strumenti per scrivere, analizzare e mantenere applicazioni cross-platform. Firebase, infatti, offre funzionalità come analisi, database (usando strutture noSQL), messaggistica e segnalazione di arresti anomali per la gestione di applicazioni web, iOS e Android [1]. Esso presenta un piano di lavoro chiamato “Console” dal quale è possibile gestire ogni applicazione associata.

Qualsiasi operazione dell’utente/amministratore sul database utilizza un linguaggio di programmazione specifico (nel nostro esempio, il linguaggio utilizzato da Android Studio). Questo meccanismo viene implementato tramite un’interfaccia grafica o un gestore DBMS da riga di comando. Il server è un componente del DBMS ed è responsabile della fornitura di servizi di utilizzo del database ad altri programmi e computer in un modello client/server. Esso, inoltre, archivia i dati, riceve le richieste dai client ed elabora le risposte corrispondenti.

Fondamentalmente, perciò, mediante delle righe di codice implementate sull’IDE di Android, si possono estrapolare e modificare i dati memorizzati nel DBMS. La Figura 3.6 illustra la console che presenta Firebase per lo sviluppo dei progetti.



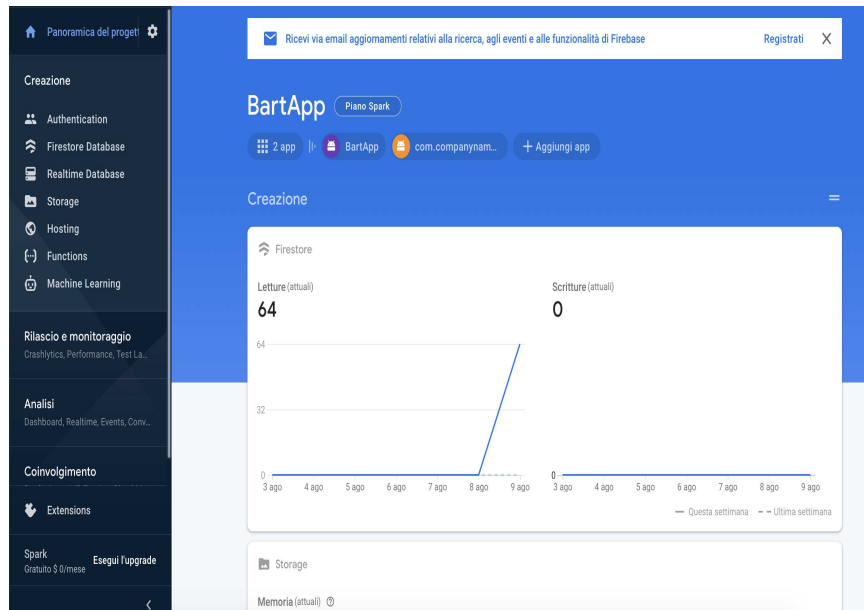


Figura 3.6. Console di Firebase





```

        .build();
        user.updateProfile(profileChangeRequest).addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                writeuserToDb(nome, cognome, user.getId(), email);
                Intent intent= new Intent();
                intent.putExtra("nome", textNome.getText().toString());
                intent.putExtra("cognome", textCognome.getText().toString());
                intent.putExtra("email", textEmail.getText().toString());
                intent.putExtra("password",textPassword.getText().toString());
                Intent intent1 = new Intent(Registrazione.this, Login_Screen.class);
                startActivity(intent1);
            }
        });
    }
    else{
        task.getException().printStackTrace();
        Toast.makeText(Registrazione.this, getString(R.string.errorsignup), Toast.
            LENGTH_SHORT).show();
    }
}
});
}
catch (NullPointerException e) {
    Toast.makeText(Registrazione.this, getString(R.string.inforequired), Toast.LENGTH_SHORT).show();
} catch (IllegalArgumentException e){
    Toast.makeText(Registrazione.this, getString(R.string.inforequired), Toast.LENGTH_SHORT).show();
} /* oppure semplicemente un unico catch con Exception e*/
}
});
getSupportActionBar().setTitle(getString(R.string.Registrati));
}
}

```

**Listing 4.1.** Codice per la registrazione di nuovi utenti

La seconda funzione da menzionare, per la fase di registrazione, è l’inserimento dei nuovi utenti nel database di Firebase.

Il Listato 4.2 illustra il metodo `writeuserToDb`, il quale permette di caricare, in Firebase, i dati dell’utente, quali nome, cognome ed email. Questi, poi, verranno utilizzati da altre funzioni presenti nell’applicazione.

```

private void writeuserToDb(String nome, String cognome, String uid, String email){
    Map<String, Object> user= new HashMap<>();
    user.put("nome", nome);
    user.put("cognome", cognome);
    user.put("email", email);
    FirebaseFirestore db= FirebaseFirestore.getInstance();
    db.collection("utenti").document(uid).set(user);
}

```

**Listing 4.2.** Codice per il caricamento dei nuovi utenti nel database

Per l’operazione di login, è stato utilizzato il metodo predefinito `signInWithEmailAndPassword(email,password)` (Listato 4.3), il quale, mediante il click sul pulsante “Login”, nel caso di autenticazione corretta, permette l’accesso alla home page.

```

btnLogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String email = inputEmail.getText().toString();
        final String password = inputPassword.getText().toString();
        //Metodo predefinito per effettuare il login su firebase
        auth.signInWithEmailAndPassword(email,password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()){
                    Intent intent = new Intent(Login_Screen.this, HomeScreen.class);
                    startActivity(intent);
                }
            }
        });
    }
}

```

```

        else{
            task.getException();
            Toast.makeText(Login_Screen.this, "Email/Password errati!", Toast.LENGTH_SHORT).show();
        }
    }
}
});
});
});

```

Listing 4.3. Codice per il login

## 4.3 Home page

Successivamente all'operazione di login, l'utente avrà accesso alla schermata principale dell'applicazione, la quale mostra il catalogo degli oggetti proposti ed un menù, che visualizza tutte le funzioni dell'applicazione.

All'interno di questo listato (Listato 4.4) sono presenti metodi per la gestione del menù laterale (`ActionBarDrawerToggle`), per la gestione delle altre schermate che verranno visualizzate in seguito ad azioni dell'utente (`setNavigationItemSelectedListener`), e per la visualizzazione del nome dell'utente mostrato nell'header del menù: (`nomeHeader.setText()`). Nel listato, inoltre, sono riportati i metodi per l'inserimento dell'immagine del profilo dell'utente, la quale verrà visualizzata direttamente sopra il suo nome. Essi sono inseriti nella classe chiamata `RecyclerViewFragment`, che costituisce il vero e proprio catalogo.

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_home_screen);

    final String adminuse ;
    final String adminpass= "admin";

    firebaseDatabase = FirebaseDatabase.getInstance().getReference().child("utenti");

    mAuth = FirebaseAuth.getInstance();
    currentUser = mAuth.getCurrentUser();
    utente = mAuth.getUid();

    activityOfApplication = this;
    Preferences preferences = new Preferences();
    int login = preferences.readLoginNumber(this);
    preferences.updateLoginNumber(this, login);

    menuItem = (MenuItem) findViewById(R.id.Contattaci);
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    drawerLayout = findViewById(R.id.drawer_layout);
    actionBarDrawerToggle = new ActionBarDrawerToggle(this, drawerLayout, toolbar, R.string.open, R.string.close);
    drawerLayout.addDrawerListener(actionBarDrawerToggle);
    actionBarDrawerToggle.syncState();
    NavigationView navView = (NavigationView) findViewById(R.id.navigation);
    navView.setNavigationItemSelectedListener(this);
    firebaseAuth = FirebaseAuth.getInstance();
    firebaseUser = firebaseAuth.getCurrentUser();
    View headerView = navView.getHeaderView(0);
    nomeHeader = (TextView) headerView.findViewById(R.id.nomeHeader);
    nomeHeader.setText(firebaseUser.getDisplayName());
    imageHeader = (ImageView) headerView.findViewById(R.id.button_add);
    firebaseStorage = FirebaseStorage.getInstance();
    storageReference = firebaseStorage.getReference();

    //inserisco l'immagine del profilo nell'header della navigation view
    storageReference.child("Image").child("Profile Pic").child(firebaseAuth.getUid()).getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
        @Override
        public void onSuccess(Uri uri) {
            Picasso.get().load(uri).fit().centerCrop().into(imageHeader);
        }
    });
    //inserimento fragment della recycler per la lista dei prodotti che sono sul mercato
    RecyclerViewFragment recyclerViewFragment = new RecyclerViewFragment();
}

```

```

FragmentManager fm = getSupportFragmentManager();
FragmentTransaction ft = fm.beginTransaction();
ft.add(R.id.fragment_container_visualizza, recyclerViewFragment, "");
ft.commit();
}

```

Listing 4.4. Codice per la home page

Altro metodo fondamentale per questa classe è `onCreateView`, che permette la corretta visualizzazione dei prodotti. Essi sono inseriti in un array tramite l'oggetto `options`, nella classe denominata `FirestoreRecyclerOptions`, che rappresenta l'insieme dei prodotti visibili dagli utenti (Listato 4.5).

```

//Connessione con il database "oggetti" di Firebase
databaseReference = FirebaseDatabase.getInstance().getReference().child("oggetti");
databaseReference.keepSynced(true);

//Estraggo tutti gli oggetti che verranno inseriti nella RecyclerView
FirestoreRecyclerOptions<Oggetto> options = new FirestoreRecyclerOptions.Builder<Oggetto>()
    .setQuery(databaseReference, Oggetto.class)
    .build();

adapter = new MyAdapter(options);
recyclerView.setAdapter(adapter);
return view;

```

Listing 4.5. Codice per la visualizzazione dei prodotti

Infine, poi, la classe utilizza dei metodi per filtrare i prodotti. Nel Listato 4.6 viene illustrato il procedimento per cercare un prodotto per nome. Esso utilizza una semplice query che visualizza sullo schermo i prodotti compatibili con il filtro apportato.

```

protected void firebaseSearch(String searchText) {
    final Query query = databaseReference.orderByChild("nome").equalTo(searchText.toLowerCase());

    options = new FirestoreRecyclerOptions.Builder<Oggetto>().setQuery(query, Oggetto.class).build();

    firestoreRecyclerViewAdapter = new FirestoreRecyclerAdapter<Oggetto, MyAdapter.FirebaseViewHolder>(options) {
        @Override
        protected void onBindViewHolder(@NonNull final MyAdapter.FirebaseViewHolder firebaseViewHolder, final
            int i, @NonNull Oggetto oggetto) {
            firebaseViewHolder.nome.setText(oggetto.getNome());
            firebaseViewHolder.nomeVenditore.setText(oggetto.getNomeVenditore());
            firebaseViewHolder.prezzo.setText(String.valueOf(oggetto.getPrezzo()));
            firebaseViewHolder.idUser.setText(oggetto.getIdUser());
            final FirebaseStorage firebaseStorage = FirebaseStorage.getInstance();
            StorageReference storageReference = firebaseStorage.getReference();
            final String keyId = this.getRef(i).getKey();
            final String descrizione = oggetto.getDescrizione();

            String idUser = oggetto.getIdUser();
            String nome = oggetto.getNome();
            storageReference.child("Image").child("ImmagineOggetti").child(idUser).child(nome).getDownloadUrl().
                addOnSuccessListener(new OnSuccessListener<Uri>() {
                    @Override
                    public void onSuccess(Uri uri) {
                        Picasso.get().load(uri).fit().centerCrop().into(firebaseViewHolder.imagineOggetto);
                    }
                });
            firebaseViewHolder.itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                    TextView nome1 = v.findViewById(R.id.nome_oggetto);
                    String Nome1 = nome1.getText().toString();
                    TextView nomeVend = v.findViewById(R.id.nome_venditore);
                    String NomeVend = nomeVend.getText().toString();
                    TextView prezzo1 = v.findViewById(R.id.prezzo);
                    String Prezzo1 = prezzo1.getText().toString();
                    TextView idUser1 = v.findViewById(R.id.id_venditore);
                    String idUser1 = idUser1.getText().toString();

                    Bundle bundle = new Bundle();
                    bundle.putString("IdOggetto", keyId);
                    bundle.putString("Nome1", Nome1);
                    bundle.putString("NomeVend", NomeVend);

```

```

        bundle.putString("Prezzo1", Prezzo1);
        bundle.putString("idUser", IdUser1);
        bundle.putString("descrizione", descrizione);
        AppCompatActivity abc = (AppCompatActivity) v.getContext();
        VisualizzaProdottoFragment visualizzaProdottoFragment = new VisualizzaProdottoFragment();
        visualizzaProdottoFragment.setArguments(bundle);
        firebaseRecyclerViewAdapter.stopListening();
        abc.getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container_visualizza
            , visualizzaProdottoFragment, "").addToBackStack(null).commit();
    }
}
}
}

```

Listing 4.6. Codice per la ricerca dei prodotti

## 4.4 Aggiunta dei prodotti

Per l'aggiunta di prodotti si fa riferimento alla classe `Inserimento`, la quale presenta i metodi per l'inserimento di un nuovo prodotto nel catalogo. Una volta compilati i campi richiesti dalle label presenti nella schermata, il metodo `OnSuccessListener()` caricherà il prodotto nel database di riferimento. Il Listato 4.7, inoltre, descrive il metodo `OnFailureListener()`, che opererà nel caso in cui l'utente non avrà compilato tutti i campi richiesti. In questo caso, l'utente visualizzerà un avviso di errore.

```

databaseReference = FirebaseDatabase.getInstance().getReference().child("oggetti");
inviodati.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (imagePath != null) {
            try {
                //Invio dei dati di carattere prettamente
                final String nome = nomeProdotto.getText().toString();
                final int prezzo = Integer.parseInt(input_prezzo.getText().toString());
                final String nome_venditore = currentUser.getDisplayName();
                final String descrizione = input_descrizione.getText().toString();
                final String posizione = posizioneUggetto.getText().toString();

                //inserimento dei dati dell'oggetto
                oggetto.setPrezzo(prezzo);
                oggetto.setNomeVenditore(nome_venditore);
                oggetto.setNome(nome);
                oggetto.setDescrizione(descrizione);
                oggetto.setPosizione(posizione);
                oggetto.setIdUser(currentUser.getId());
                oggetto.setDataCreazione();

                databaseReference.push().setValue(oggetto);

                //inserimento in firebase dell'immagine dell'oggetto
                firebaseStorage = FirebaseStorage.getInstance();
                storageReference = firebaseStorage.getReference();

                FirebaseDatabase firebaseDatabase = FirebaseDatabase.getInstance();
                DatabaseReference databaseReference = firebaseDatabase.getReference(mAuth.getId());
                StorageReference imageReference = storageReference.child("Image").child("ImmaginiOggetti").
                    child(mAuth.getId()).child(nome); //User id/Images/Profile Pic.jpg
                UploadTask uploadTask = imageReference.putFile(imagePath); //uri dell'immagine
                uploadTask.addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(Inserimento.this, "Error: Uploading profile picture", Toast.
                            LENGTH_SHORT).show();
                    }
                }).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                        Toast.makeText(Inserimento.this, "Immagine Inserita", Toast.LENGTH_SHORT).show();
                    }
                });

                Toast.makeText(Inserimento.this, "Oggetto inserito", Toast.LENGTH_LONG).show();
                Intent intent = new Intent(Inserimento.this, HomeScreen.class);
                startActivity(intent);
                finish();
            }
        }
    }
});

```

```

        } catch (NullPointerException e) {
            Toast.makeText(Inserimento.this, getString(R.string.inforequired), Toast.LENGTH_SHORT).show()
            ;
        } catch (IllegalArgumentException e) {
            Toast.makeText(Inserimento.this, getString(R.string.inforequired), Toast.LENGTH_SHORT).show()
            ;
        }
    } else {
        Toast.makeText(Inserimento.this, "Devi inserire anche l'immagine", Toast.LENGTH_SHORT).show();
    }
}
});

```

**Listing 4.7.** Codice per l'inserimento dei prodotti

La funzione di inserimento di un nuovo prodotto nel catalogo, però, si serve di una libreria denominata **Glide**. Come mostrato nel Listato 4.8, questa libreria permette di visualizzare sullo schermo l'immagine che abbiamo selezionato per il prodotto.

```

public void onActivityResult(int requestCode, int resultCode, Intent intent) {
    super.onActivityResult(requestCode, resultCode, intent);
    if (requestCode == PICK_IMAGE) {
        Bitmap bitmap = null;
        if (resultCode == RESULT_OK) {
            if (getPickImageResultUri(intent) != null) {
                imagePath = getPickImageResultUri(intent);
                try {
                    bitmap = MediaStore.Images.Media.getBitmap(this.getContentResolver(), imagePath);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            } else {
                bitmap = (Bitmap) intent.getExtras().get("data");
            }
        }

        Glide.with(this)
            .load(bitmap)
            .centerCrop()
            .into(imageView);
    }
}

```

**Listing 4.8.** Codice per la visualizzazione dell'immagine

## 4.5 Profilo

Per accedere all'area riservata dell'utente bisogna entrare nel menù laterale e cliccare sull'immagine del suo profilo. La successiva schermata visualizzerà tutti i dati dell'utente, quali l'immagine del profilo con dimensioni più grandi e sensibile al click (per la sua gestione abbiamo utilizzato la libreria **Glide**, come mostrato nel Listato 4.9), l'email, il nome utente e la password. Inoltre, in fondo alla schermata, è presente un pulsante che permette l'eliminazione dell'account. Essa è implementata mediante il metodo `delete()` della libreria **FirebaseAuth**.

```

public void onActivityResult(int requestCode, int resultCode, Intent intent) {
    super.onActivityResult(requestCode, resultCode, intent);
    if (requestCode == PICK_IMAGE) {
        Bitmap bitmap = null;
        if (resultCode == RESULT_OK) {
            if (getPickImageResultUri(intent) != null) {
                imagePath = getPickImageResultUri(intent);
                try {
                    bitmap = MediaStore.Images.Media.getBitmap(this.getContentResolver(), imagePath);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}

```



```

        } else {
            bitmap = (Bitmap) intent.getExtras().get("data");
        }
    }

    Glide.with(this)
        .load(bitmap)
        .centerCrop()
        .into(proPic);
    }
}

```

**Listing 4.9.** Codice per l'immagine del profilo

Il Listato 4.10 definisce tutti i permessi che l'applicazione richiede per le varie operazioni. Infatti, per l'operazione di modifica e aggiunta dell'immagine del profilo, prima di poter accedere alle immagini del dispositivo ed alla fotocamera, l'applicazione richiede i permessi necessari. Nel caso in cui non verranno dati tali permessi, non saranno consentite l'aggiunta e la modifica dell'immagine.

```

@Override
public void onClick(View v) {
    permissions.add(Manifest.permission.CAMERA);
    permissions.add(Manifest.permission.WRITE_EXTERNAL_STORAGE);
    permissions.add(Manifest.permission.READ_EXTERNAL_STORAGE);
    permissionsToRequest = findUnaskedPermissions(permissions);
    if(permissionsToRequest.size() > 0) {
        requestPermissions(permissionsToRequest.toArray(new String[permissionsToRequest.size()]),
            ALL_PERMISSIONS_RESULT);
    } else {
        startActivityForResult(getPickImageChooserIntent(), PICK_IMAGE);
    }
    btnSalvaModifica.setVisibility(View.VISIBLE);
    btnSalvaModifica.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            StorageReference imageReference = storageReference.child("Image").child("Profile Pic").child(
                mAuth.getUid());
            if (imagePath != null) {
                sendUserData(); // metodo per caricare l'immagine nello storage di Firebase
            }
            else{
                Toast.makeText(Profilo.this, "Non hai inserito alcuna immagine!", Toast.LENGTH_SHORT).
                    show();
            }
        }
    });
}

private Uri getPickImageResultUri(Intent data) {
    boolean isCamera = true;
    if (data != null) {
        String action = data.getAction();
        isCamera = action != null && action.equals(MediaStore.ACTION_IMAGE_CAPTURE);
    }
    return isCamera ? getCaptureImageOutputUri() : data.getData();
}

private Intent getPickImageChooserIntent() {

    Uri outputFileUri = getCaptureImageOutputUri();

    List<Intent> allIntents = new ArrayList<>();
    PackageManager packageManager = this.getPackageManager();

    Intent captureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    List<ResolveInfo> listCam = packageManager.queryIntentActivities(captureIntent, 0);
    for (ResolveInfo res : listCam) {
        Intent intent = new Intent(captureIntent);
        intent.setComponent(new ComponentName(res.activityInfo.packageName, res.activityInfo.name));
        intent.setPackage(res.activityInfo.packageName);
        if (outputFileUri != null) {
            intent.putExtra(MediaStore.EXTRA_OUTPUT, outputFileUri);
        }
        allIntents.add(intent);
    }

    Intent galleryIntent = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    List<ResolveInfo> listGallery = packageManager.queryIntentActivities(galleryIntent, 0);
    for (ResolveInfo res : listGallery) {
        Intent intent = new Intent(galleryIntent);

```

```

        intent.setComponent(new ComponentName(res.activityInfo.packageName, res.activityInfo.name));
        intent.setPackage(res.activityInfo.packageName);
        allIntents.add(intent);
    }

    Intent mainIntent = allIntents.get(allIntents.size()-1);
    for(Intent intent : allIntents) {
        if(intent.getComponent().getClassName().equals("com.android.documentsui.DocumentsActivity")) {
            mainIntent = intent;
            break;
        }
    }
    allIntents.remove(mainIntent);

    Intent chooserIntent = Intent.createChooser(mainIntent, getString(R.string.selsorgente));

    chooserIntent.putExtra(Intent.EXTRA_INITIAL_INTENTS, allIntents.toArray(new Parcelable[allIntents.size()]));

    return chooserIntent;
}

public Uri getCaptureImageOutputUri() {
    Uri outputFileUri = null;
    File getImage = this.getExternalCacheDir();
    if(getImage != null) {
        outputFileUri = Uri.fromFile(new File(getImage.getPath(), "propic.png"));
    }
    return outputFileUri;
}

private ArrayList findUnaskedPermissions(ArrayList<String> wanted) {
    ArrayList<String> result = new ArrayList<>();

    for(String perm : wanted) {
        if(!this.checkSelfPermission(perm) == PackageManager.PERMISSION_GRANTED) {
            result.add(perm);
        }
    }

    return result;
}

public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
    if(requestCode == ALL_PERMISSIONS_RESULT) {
        for(String perm: permissionsToRequest) {
            if(!this.checkSelfPermission(perm)==PackageManager.PERMISSION_GRANTED) {
                permissionsRejected.add(perm);
            }
        }
        if(permissionsRejected.size() > 0) {
            if(shouldShowRequestPermissionRationale(permissionsRejected.get(0))) {
                Toast.makeText(this,"Approva tutto", Toast.LENGTH_SHORT).show();
            }
            else {
                startActivityForResult(getPickImageChooserIntent(), PICK_IMAGE);
            }
        }
    }
}

private void sendUserData() { //carico l'immagine nello storage di Firebase
    FirebaseDatabase firebaseDatabase = FirebaseDatabase.getInstance();
    DatabaseReference databaseReference = firebaseDatabase.getReference(mAuth.getUid());
    //Scelgo il percorso adatto per caricare l'immagine nello storage
    StorageReference imageReference = storageReference.child("Image").child("Profile Pic").child(mAuth.getUid());
    //User id/Images/Profile Pic.jpg
    UploadTask uploadTask = imageReference.putFile(imagePath); //caricamento dell'immagine nello storage
    uploadTask.addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(Profilo.this, "Error: Uploading profile picture", Toast.LENGTH_SHORT).show();
        }
    }).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
            Toast.makeText(Profilo.this, "Profile picture uploaded", Toast.LENGTH_SHORT).show();
        }
    });
}
}

```

**Listing 4.10.** Codice per la richiesta di permessi e per la modifica dell'immagine del profilo

Più nel dettaglio, il codice mostrato nel Listato 4.10 definisce un array, `permissionsToRequest` tramite il metodo `findUnaskedPermission()`, che verifica se ci sono permessi da richiedere. Sarà poi il metodo `checkSelfPermission()`,

chiamato dal metodo `findUnaskedPermission()`, che verificherà lo stato di ogni elemento dell'array, e, quindi, di ogni permesso.

La gestione della fotocamera e della galleria, invece, è garantita dai metodi `getPickImageResultUri()`, `getPickImageChooserIntent()` e `getCaptureImageOutputUri()`. Infine, il metodo `sendUserData()` permette di caricare nel database dell'utente l'immagine del profilo selezionata.

## 4.6 Offerte inviate e ricevute

Per visualizzare le offerte inviate, è stata applicata una query, che filtra il database `scambi` per estrarre tutte le offerte che hanno per ID quelle del venditore che intende visualizzare le offerte inviate. Il Listato 4.11 mostra, infatti, come si effettua l'accesso al database e come viene applicato il filtro relativo all'offerta. Nel caso in cui l'utente non ha proposto nessuno scambio, e, quindi, l'offerta inviata è nulla, è stato implementato un codice che visualizza una riga di testo che comunica all'utente che al momento non ci sono dati da visualizzare.

```

@Nullable
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
    View view = inflater.inflate(R.layout.recycler_nosearch, container, false);
    RecyclerView recyclerView = (RecyclerView) view.findViewById(R.id.recycler_view);
    layoutManager = new LinearLayoutManager(this.getContext());
    recyclerView.setLayoutManager(layoutManager);

    String mAuth = FirebaseAuth.getInstance().getCurrentUser().getUid();
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    Query query = db.collection("scambi").whereEqualTo("idVend", mAuth);

    FirestoreRecyclerOptions<Offerta> options = new FirestoreRecyclerOptions.Builder<Offerta>()
        .setQuery(query, Offerta.class)
        .build();
    adapter = new MyAdapterOfferteInviata(options, this.getContext());
    recyclerView.setAdapter(adapter);
    return view;
}

```

**Listing 4.11.** Codice relativo al metodo `onCreate` delle offerte inviate

Per quanto riguarda le offerte ricevute, il codice è speculare a quello relativo alle offerte inviate. Vi è una query che accede al database `scambi` e filtra le offerte con l'ID, che, in questo caso, deve coincidere con quello dell'utente correntemente autenticato.

```

public void rifiuta(final String keyId) {

    AlertDialog.Builder dialog = new AlertDialog.Builder(context);
    dialog.setTitle("Attenzione!");
    dialog.setCancelable(false);
    dialog.setMessage("Sei sicuro di voler rifiutare l'offerta?");
    dialog.setPositiveButton("Rifiuta", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            FirebaseFirestore db = FirebaseFirestore.getInstance();

            db.collection("scambi").document(keyId).delete();
        }
    });
    dialog.setNegativeButton("Annulla", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
        }
    });
}

```

```

        AlertDialog alertDialog = dialog.create();
        alertDialog.show();
    }
    //Funzione per accettare lo scambio in cui creo un database in firestore per registrare un riepilogo dello
    //scambio.
    public void accetta(final String keyId, final String idProdAcq, final String idProdVend, final String idAcq,
        final
        String idVend, final String nomeOggettoAcq, final String nomeOggettoVend, final String
        nomeVend, final String emailVend, final String prezzoAcq, final String
        prezzoOggettoVend, final String dataCreazione) {
        AlertDialog.Builder dialog = new AlertDialog.Builder(context);
        dialog.setTitle("Attenzione!");
        dialog.setCancelable(false);
        dialog.setMessage("Sei sicuro di voler accettare l'offerta?");
        dialog.setPositiveButton("Accetta", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

                FirebaseFirestore firebaseFirestore= FirebaseFirestore.getInstance();
                final Map<String, Object> map= new HashMap<>();

                map.put("emailAcq", FirebaseAuth.getInstance().getCurrentUser().getEmail());
                map.put("emailVend", emailVend);
                map.put("nomeOggettoAcq", nomeOggettoAcq);
                map.put("nomeOggettoVend", nomeOggettoVend);
                map.put("nomeUtenteVend", nomeVend);
                map.put("idAcq", idAcq);
                map.put("idVend", idVend);
                map.put("prezzoAcq", prezzoAcq);
                map.put("prezzoOggettoVend", prezzoOggettoVend);
                map.put("dataCreazione", dataCreazione);

                firebaseFirestore.collection("riepilogoscambi").document().set(map);

                operazione1(idProdAcq);
                operazione2(idProdAcq);
                operazione3(idProdVend);
                operazione4(idProdVend, idProdAcq, idAcq, idVend, nomeOggettoAcq, nomeOggettoVend);

            }
        });
        dialog.setNegativeButton("Annulla", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        });
    }

    AlertDialog alertDialog = dialog.create();
    alertDialog.show();
}

//Elimino tutte le offerte che mi sono state fatte sullo stesso oggetto
public void operazione1 (String idProdAcq) {
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    db.collection("scambi").whereEqualTo("idProdAcq", idProdAcq)
        .get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    for (DocumentSnapshot document : task.getResult()) {
                        document.getReference().delete();
                        Toast.makeText(context, "Scambio effettuato", Toast.LENGTH_SHORT).show();
                    }
                } else {
                }
            }
        });
}

//Elimino tutte le offerte che IO ho inviato ad altri dell'oggetto che scambio con l'utente.
public void operazione2 (String idProdAcq) {
    FirebaseFirestore db0 = FirebaseFirestore.getInstance();
    db0.collection("scambi").whereEqualTo("idProdVend", idProdAcq).get().addOnCompleteListener(new
        OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    for (DocumentSnapshot document : task.getResult()) {
                        document.getReference().delete();
                        Toast.makeText(context, "Scambio effettuato", Toast.LENGTH_SHORT).show();
                    }
                } else {
                }
            }
        });
}

//Elimino tutte le offerte in cui l'oggetto che mi offre l'altro utente e' stato richiesto da altri utenti.
public void operazione3 (String idProdVend) {
    FirebaseFirestore db1 = FirebaseFirestore.getInstance();
    db1.collection("scambi").whereEqualTo("idProdAcq", idProdVend).get().addOnCompleteListener(new
        OnCompleteListener<QuerySnapshot>() {
            @Override

```

```

        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                for (DocumentSnapshot document : task.getResult()) {
                    document.getReference().delete();
                }
            } else {
            }
        }
    });
}
}
//Elimino tutte le offerte in cui l'oggetto che mi offre l'altro utente e' stato proposto ad altri utenti.
public void operazione4 (final String idProdVend, final String idProdAcq, final String idAcq, final String
idVend, final String nomeOggettoAcq, final String nomeOggettoVend) { //oggetto offerto
    FirebaseFirestore db2 = FirebaseFirestore.getInstance();
    db2.collection("scambi").whereEqualTo("idProdVend", idProdVend).get().addOnCompleteListener(new
    OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                for (DocumentSnapshot document : task.getResult()) {
                    document.getReference().delete();
                    Toast.makeText(context, "Scambio effettuato", Toast.LENGTH_SHORT).show();
                    eliminaProdotti(idProdVend, idProdAcq, idAcq, idVend, nomeOggettoAcq, nomeOggettoVend);
                }
            } else {
            }
        }
    });
}
//Elimino dal catalogo tutti i prodotti insieme alle immagini di cui accetto lo scambio.
public void eliminaProdotti (String idProdVend, String idProdAcq, String idAcq, String idVend, String
nomeOggettoAcq, String nomeOggettoVend){
    FirebaseDatabase firebaseDatabase = FirebaseDatabase.getInstance();
    firebaseDatabase.getReference("oggetti").child(idProdVend).removeValue();
    firebaseDatabase.getReference("oggetti").child(idProdAcq).removeValue();
    StorageReference storageReference = FirebaseStorage.getInstance().getReference().child("Image").child("
    ImmaginiOggetti").child(idAcq);
    storageReference.child(nomeOggettoAcq).delete();
    StorageReference storageReference1 = FirebaseStorage.getInstance().getReference().child("Image").child("
    ImmaginiOggetti").child(idVend);
    storageReference1.child(nomeOggettoVend).delete();
}
}
}

```

Listing 4.12. Codice per la gestione delle offerte

Una volta che è stata proposta l'offerta, l'utente che riceve quest'ultima avrà la possibilità di accettare o rifiutare la proposta. Nel caso in cui l'utente accetti, verrà creata una  $\text{Map}\langle\text{String}, \text{Object}\rangle$  che sarà poi popolata con i campi necessari per il riepilogo degli scambi. La mappa, quindi, avrà i seguenti campi:

- emailAcq;
- emailVend;
- nomeOggettoAcq;
- nomeOggettoVend;
- nomeUtenteVend;
- idAcq;
- idVend;
- posizioneAcq;
- posizioneVend;
- dataCreazione.

Nel caso in cui l'offerta venga accettata, la mappa verrà inserita nella tabella riepilogoscambi.

In questo caso, poi, verranno eseguite quattro operazioni: `operazione1()`, `operazione2()`, `operazione3()`, `operazione4()`, le quali permettono di eliminare tutte le altre proposte con lo stesso oggetto scambiato, evitando qualche truffa. Infine, il Listing 4.12 mostra l'operazione `eliminaProdotti()`, che permette di eliminare lo storico dello scambio e, quindi, i due prodotti barattati e le loro immagini.

## 4.7 Riepilogo scambi

Per quanto riguarda lo storico degli scambi effettuati dall'utente correntemente autenticato, sono stati presi in considerazione due layout che si differenziano a seconda se l'utente che desidera visualizzare il riepilogo è il venditore o l'acquirente. Per convenzione è stato utilizzato il colore verde a simboleggiare l'oggetto acquistato e il colore rosso per quello venduto. Infatti, il Listato 4.13 illustra, nelle sue prime righe di codice, un controllo che definisce subito il layout da associare.

```

@Override
protected void onBindViewHolder(@NonNull MyAdapterRiepilogo.FirestoreViewHolder holder, int position, @NonNull
    final Riepilogo model) {
    final Riepilogo model) {
    if (model.getIdAcq().equals mAuth.getUserId()) {
        holder.relativeLayout2.setVisibility(View.INVISIBLE);
        holder.nomeOggettoAcq.setText(model.getNomeOggettoAcq());
        holder.nomeOggettoVend.setText(model.getNomeOggettoVend());
        holder.nomeVend.setText(model.getNomeUtenteVend());
        holder.emailVend.setText(model.getEmailVend());
        DocumentSnapshot documentSnapshot = getSnapshots().getSnapshot(position);
        final String id = documentSnapshot.getId();
        holder.btnCancella.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                eliminaRiepilogo(id);
            }
        });

        holder.btnContatta.setOnClickListener(new View.OnClickListener() {
            final String email = model.getEmailVend();
            final String nomeOggetto = model.getNomeOggettoVend();
            final String nomeOggetto2 = model.getNomeOggettoAcq();

            @Override
            public void onClick(View v) {
                contatta(email, nomeOggetto, nomeOggetto2, context);
            }
        });
    } else if (model.getIdVend().equals(mAuth.getUserId())) {
        holder.relativeLayout1.setVisibility(View.INVISIBLE);
        holder.email2.setText("Email: " + model.getEmailAcq());
        holder.nomeOggettoProposto.setText("Oggetto proposto: " + model.getNomeOggettoVend());
        holder.nomeOggettoRichiesto.setText("Oggetto richiesto: " + model.getNomeOggettoAcq());
        holder.btnContatta2.setOnClickListener(new View.OnClickListener() {
            final String email = model.getEmailAcq();
            final String nomeOggetto = model.getNomeOggettoVend();
            final String nomeOggetto2 = model.getNomeOggettoAcq();

            @Override
            public void onClick(View v) {
                contatta(email, nomeOggetto, nomeOggetto2, context);
            }
        });
    } else {
        holder.itemView.setVisibility(View.INVISIBLE);
    }
}

```

**Listing 4.13.** Codice per la gestione dei layout del riepilogo scambi

Il Listato 4.14, invece, mostra come sia possibile completare la compravendita. Per ogni scambio avvenuto, infatti, è presente una casella contenente l'indirizzo di posta elettronica dell'altro utente, in modo tale che entrambi possano accordarsi per via telematica. I metodi `contatta()` e `contatta2()`, presenti nel Listato, si occupano di aprire il client di posta elettronica con un messaggio che invita a completare definitivamente lo scambio, con destinatario l'altro utente. Nel Listato 4.14, inoltre, è presente il metodo `eliminaRiepilogo()`, che permette di pulire la cronologia dell'utente, cancellando, quindi, dal registro gli scambi avvenuti.

```

public void eliminaRiepilogo(final String id) {
    AlertDialog.Builder dialog = new AlertDialog.Builder(context);
    dialog.setTitle("Attenzione!");
    dialog.setCancelable(false);
    dialog.setMessage("Sei sicuro di voler il riepilogo dell'offerta? Perderai tutti i dettagli relativi ad essa!");
}

```

```

        dialog.setPositiveButton("Elimina", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                FirebaseFirestore db = FirebaseFirestore.getInstance();
                db.collection("riepilogoscambi").document(id).delete();
            }
        });
        dialog.setNegativeButton("Annulla", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        });
        AlertDialog alertDialog = dialog.create();
        alertDialog.show();
    }

    public void contatta(String email, String nomeOggetto, String nomeOggetto2, Context context) {
        Intent intent = new Intent(Intent.ACTION_SENDTO, Uri.fromParts("mailto", email, null));
        intent.putExtra(Intent.EXTRA_SUBJECT, "Scambio Oggetto");
        intent.putExtra(Intent.EXTRA_TEXT, "Sono interessato all'oggetto " +
            nomeOggetto + " che ha inserito, in cambio del mio " + nomeOggetto2 + " possiamo effettuare lo
            scambio!");
        intent.putExtra(Intent.EXTRA_EMAIL, email);
        context.startActivity(Intent.createChooser(intent, "Invia email.."));
    }
}

```

**Listing 4.14.** Codice per la gestione della cronologia delle offerte

## 4.8 Contatti

Come citato nella sezione precedente, l'applicazione prevede una serie di metodi che fanno uso della posta elettronica per lo scambio di informazioni e dettagli. In questo caso, il Listato 4.15 illustra come sia possibile, per l'utente, mettersi in contatto con gli sviluppatori della medesima app, tramite l'ancora presente sul menù a tendina **Contattaci**. Il metodo `invioMail()` risulta piuttosto semplice e permette all'utente di notificare eventuali anomalie o malfunzionamenti.

```

//metodo per il 'contattaci' e contattare gli sviluppatori in caso di problemi
public void invioMail() {
    Intent intent = new Intent(Intent.ACTION_SENDTO);
    intent.putExtra(Intent.EXTRA_SUBJECT, "Help Request");

    intent.setData(Uri.parse("mailto:EartApp@developers.com"));
    startActivity(intent);
}

```

**Listing 4.15.** Codice per l'invio di mail agli sviluppatori

## 4.9 Statistiche

Per quanto riguarda la sezione delle statistiche, la quale è accessibile dai soli utenti amministratori, sono state utilizzate due query che popolano due `Map<String, Object>` differenti.

Il Listato 4.16 illustra le città da cui sono stati effettuati il maggior numero di scambi, il numero di oggetti totali esposti e la quantità di oggetti proposti per giorno. Altro dato fornito da tale listato è il numero di accessi, che viene incrementato di volta in volta a seguito di un accesso da parte di ogni utente.

```

// Recupero numero di login
Activity applicationActivity = HomeScreen.getContextOfApplication();
int loginNumber = applicationActivity.getPreferences(MODE_PRIVATE).getInt("login", -1);
accessigiornalieri.setText(String.valueOf(loginNumber));
System.out.println(String.format("Il numero di login e': %d", loginNumber));

// Recupero numero di oggetti con conseguente statistica su numero di oggetti al giorno
firebaseDatabase = FirebaseDatabase.getInstance().getReference().child("oggetti");
Query query = firebaseDatabase.orderByChild("dataCreazione");

query.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        long numeroOggettiEsposti = dataSnapshot.getChildrenCount();
        Map <String, Integer> mapOggettiGiornalieri = new HashMap<>();
        Map<String, Integer> mapTopCitta = new HashMap<>();

        for(DataSnapshot postsnapshot : dataSnapshot.getChildren()) {
            String dataCreazione = String.valueOf(postsnapshot.child("dataCreazione").getValue());
            String citta =String.valueOf(postsnapshot.child("posizione").getValue()).toLowerCase();

            mapOggettiGiornalieri.put(dataCreazione, mapOggettiGiornalieri.containsKey(dataCreazione) ?
                mapOggettiGiornalieri.get(dataCreazione) + 1 : 1);
            mapTopCitta.put(citta, mapTopCitta.containsKey(citta) ? mapTopCitta.get(citta) + 1 : 1);
        }

        oggesposti.setText(String.valueOf(numeroOggettiEsposti));
        for (Map.Entry<String,Integer> entry : mapOggettiGiornalieri.entrySet()) {
            String key = entry.getKey();
            Integer value = entry.getValue();
            espostialgiorno.append(key + "-->" + value + "\n");
        }
        for (Map.Entry<String,Integer> entry :mapTopCitta.entrySet()) {
            String key = entry.getKey();
            Integer value = entry.getValue();
            topcitta.append(key + " " + value + " \n");
        }

        System.out.println(String.format("Il numero di oggettiesposti e': %d", numeroOggettiEsposti));
        System.out.println(String.format("La mappa oggetti giornalieri e': %s", mapOggettiGiornalieri));
        System.out.println(String.format("La mappa top citta e': %s", mapTopCitta));
    }
}

```

Listing 4.16. Codice per la sezione delle statistiche

Il Listato 4.17, invece, filtra gli oggetti secondo la data con cui sono stati esposti e mostra la quantità di flusso di “denaro” che è avvenuta per giorno. Inoltre, esso visualizza la quantità di oggetti scambiati.

```

// Recupero scambi con conseguente statistiche su prezzi totali e prezzi giornalieri
firebaseFirestore.collection("riepilogoscambi")
    .get()
    .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                int prezzoTotaleOggettiVenduti = 0;
                int prezzoTotaleOggettiAcquistati = 0;
                int prezzoTotale = 0;
                int numeroOggettiScambiati = task.getResult().size();
                Map<String, Integer> mapDataPrezzo = new HashMap<>();

                for (QueryDocumentSnapshot document : task.getResult()) {
                    Map<String, Object> map = (Map<String, Object>) document.getData();

                    int prezzoOggettoVenduto = Integer.parseInt(String.valueOf(map.get("prezzoOggettoVend
                        ")));
                    int prezzoAcquisto = Integer.parseInt(String.valueOf(map.get("prezzoAcq")));
                    int sommaPrezzi = prezzoOggettoVenduto + prezzoAcquisto;

                    prezzoTotaleOggettiVenduti += prezzoOggettoVenduto;
                    prezzoTotaleOggettiAcquistati += prezzoAcquisto;

                    String dataCreazione = String.valueOf(map.get("dataCreazione"));

                    mapDataPrezzo.put(dataCreazione, mapDataPrezzo.containsKey(dataCreazione) ?
                        mapDataPrezzo.get(dataCreazione) + sommaPrezzi : sommaPrezzi);
                }
                prezzoTotale = prezzoTotaleOggettiAcquistati + prezzoTotaleOggettiVenduti;
                valoretotale.setText(String.valueOf(prezzoTotale));

                oggscambiati.setText(String.valueOf(numeroOggettiScambiati));
            }
        }
    });

```



```

        for (Map.Entry<String,Integer> entry : mapDataPrezzo.entrySet()) {
            String key = entry.getKey();
            Integer value = entry.getValue();
            valoreTotalealgiorno.append(key + " " + value + " "+"\\n");
        }

        System.out.println(String.format("Il prezzo totale e': %d", prezzoTotale));
        System.out.println(String.format("Il prezzo totale ogg vend e': %d",
            prezzoTotaleOggettiVenduti));
        System.out.println(String.format("Il prezzo totale ogg acq e': %d",
            prezzoTotaleOggettiAcquistati));
        System.out.println(String.format("Il numero di oggetti scambiati e': %d",
            numeroOggettiScambiati));
        System.out.println(String.format("La mappa giorno-prezzo e': %s", mapDataPrezzo));
    } else {
        Log.d(TAG, "Error getting documents: ", task.getException());
    }
}
});

```

Listing 4.17. Codice per la sezione delle statistiche

## 4.10 Logout

Terminiamo la sezione con il Listato 4.18 relativo al logout. Esso permette la disconnessione dell'utente dall'applicazione, e, quindi, il reindirizzamento alla schermata di login.

```

protected void logout() {
    AlertDialog.Builder dialog = new AlertDialog.Builder(HomeScreen.this);
    dialog.setTitle("Attenzione");
    dialog.setCancelable(false);
    dialog.setMessage("Sei sicuro di voler effettuare il Logout?");
    dialog.setPositiveButton("Conferma", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            FirebaseAuth mAuth = FirebaseAuth.getInstance();
            mAuth.signOut();
            Intent intent = new Intent(HomeScreen.this, Login_Screen.class);
            startActivity(intent);
            finish();
        }
    });
    dialog.setNegativeButton("Annulla", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
        }
    });
}

AlertDialog alertDialog = dialog.create();
alertDialog.show();
}

```

Listing 4.18. Codice per il logout



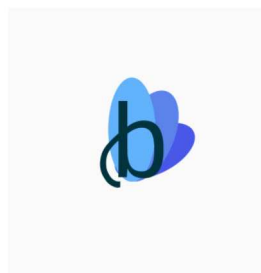
## Manuale Utente

*In questa sezione illustreremo una spiegazione dettagliata di come l'applicazione funziona, secondo le intenzioni dell'utente. Verranno esposti degli screenshot relativi ad ogni funzionalità che l'applicazione propone. Indicheremo, quindi, i passaggi da eseguire per operare e navigare nell'applicazione, guidando passo passo il lettore con una spiegazione delle varie funzionalità che essa offre.*

### 5.1 Avvio dell'applicazione

Come citato nell'introduzione al capitolo, è necessario fornire all'utente un manuale per la navigazione all'interno dell'applicazione.

La prima cosa da fare è avviare l'applicazione, mediante il click sull'icona illustrata in Figura 5.1.

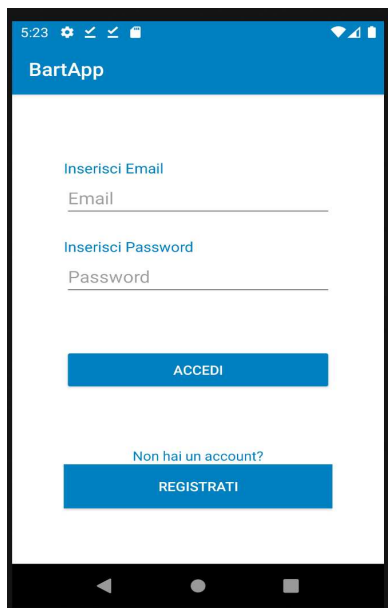


**Figura 5.1.** Icona di BartApp

Subito dopo il click, verrà visualizzata una schermata che chiederà all'utente di interagire con lo schermo per proseguire alla schermata di login. L'utente, quindi, deve toccare lo schermo e visualizzerà poi una schermata con dei campi da popolare.

## 5.2 Login e Registrazione

Una volta che l'utente ha avviato l'applicazione ed ha interagito con essa, visualizzerà una schermata con la formattazione illustrata in Figura 5.2. L'utente, in questa schermata, è invitato ad inserire l'email ed una password relative ad un account preesistente.

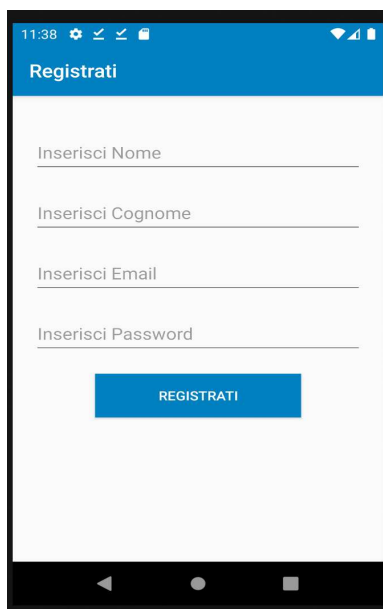


**Figura 5.2.** Schermata di login

Nel caso di un utente che avvia l'applicazione per la prima volta, esso non disporrà dei dati necessari per l'accesso, perciò, egli dovrà recarsi nella sezione "Registrazione" per creare un account. Tale sezione è composta da diverse righe che suggeriscono all'utente quali informazioni devono essere inserite per compilare correttamente il proprio account e completare la fase di registrazione. In Figura 5.3, infatti, sono illustrati i 4 campi richiesti, quali:

- Nome;
- Cognome;
- Email;
- Password.

Una volta che l'utente avrà compilato tutti i campi richiesti, premendo sul tasto "Registrati", l'applicazione procederà con la verifica di questi ultimi. Se, ad esempio, la password inserita non contiene almeno 6 caratteri, la registrazione non andrà a buon fine e l'utente visualizzerà un avviso di errore. Esso sarà avvisato ogni qual volta i campi inseriti non sono validi.



**Figura 5.3.** Schermata di registrazione

Nel caso in cui, invece, tutti i campi inseriti sono corretti e validi, l'utente verrà reindirizzato direttamente alla login screen, dalla quale, inserendo i nuovi dati, egli potrà accedere alla home page dell'applicazione, visualizzando il catalogo dei prodotti. In questo caso, inoltre, verrà popolato il database di riferimento con i dati utente inseriti.

## 5.3 Catalogo e Menù

Una volta effettuato il login, quindi, l'utente avrà accesso all'applicazione e a tutte le sue funzionalità. La prima finestra che egli visualizzerà sarà una lista di prodotti caratterizzati da un'immagine, un nome, il nome del proprietario, il prezzo e la posizione, come mostrato in Figura 5.4.

Nel caso in cui l'utente cercasse dei prodotti specifici, egli può filtrare questi ultimi tramite la lente d'ingrandimento posta immediatamente sopra il catalogo, mediante l'inserimento di un nome di un prodotto nell'apposita barra di ricerca. Inoltre, in alto a sinistra, è presente un bottone che visualizza il menù a tendina dell'applicazione. Esso, rappresentato in Figura 5.5, illustra tutte le funzionalità che l'app propone.

Questo menù propone all'utente una lista di operazioni che esso può compiere, fatta eccezione della sezione "Statistiche", riservata ai soli utenti amministratori.

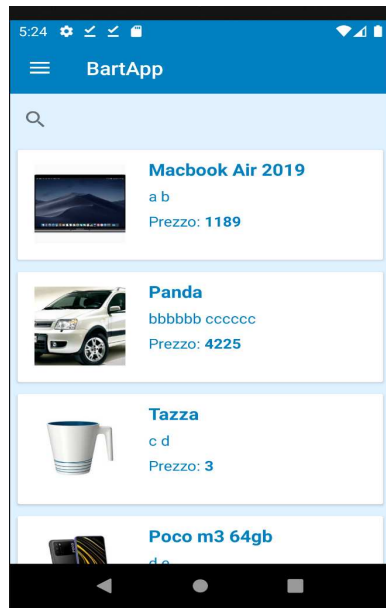


Figura 5.4. Catalogo dei prodotti

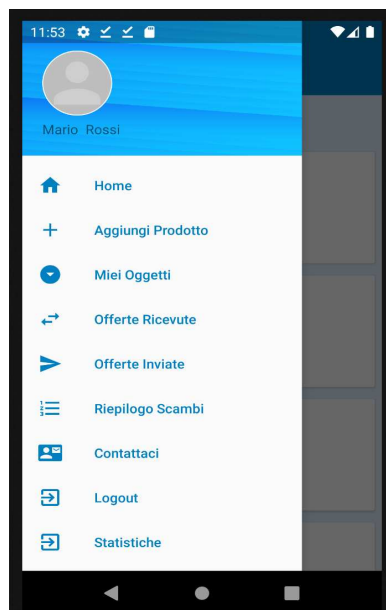


Figura 5.5. Menù a tendina

## 5.4 Visualizzazione prodotto

Qualora un particolare oggetto fosse desiderio dell'utente correntemente autenticato, egli può premere sulla riga corrispondente all'oggetto, per accedere ad una schermata che descrive i dettagli del prodotto. Oltre alle informazioni visibili direttamente dal catalogo, in questa schermata (Figura 5.6), sono presenti:

- Nome dell'oggetto;
- Nome del venditore;
- Descrizione;
- Prezzo;
- Posizione;
- Tasto Formula Offerta.



Figura 5.6. Dettagli del prodotto

L'applicazione prevede una sezione che offre la possibilità di inserire propri prodotti; nel caso in cui l'utente autenticato premesse la riga del proprio oggetto, il tasto del "Formula offerta" è sostituito da quello per l'eliminazione dello stesso.

## 5.5 Formula Offerta

Come ultimo elemento della pagina "Visualizza prodotto", è presente un tasto che permette di formulare l'offerta. Come suggerisce il nome del tasto, esso permette di effettuare una proposta di scambio al venditore che ha messo a disposizione l'oggetto

di nostro interesse. Al click sul tasto, l'utente viene reindirizzato ad una pagina contenente la lista dei prodotti, da lui attualmente esposti. La Figura 5.7, infatti, mostra la lista dei prodotti che l'utente autenticato ha messo a disposizione, ed egli dovrà sceglierne uno dalla tale, per poter inviare l'offerta. Seguirà, poi, un avviso sullo schermo che chiederà conferma all'utente di formulare l'offerta; se quest'ultimo darà il consenso, l'applicazione procederà con l'inoltro della richiesta al venditore.

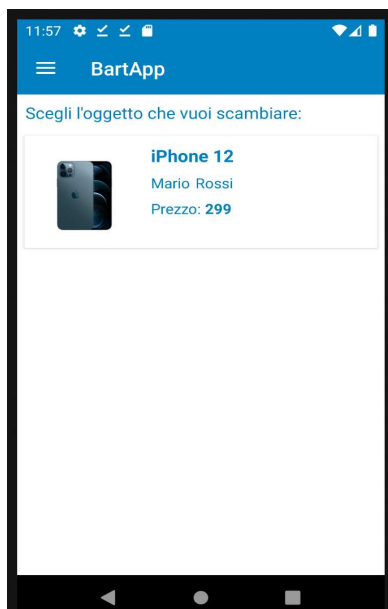


Figura 5.7. Schermata di scelta del prodotto da offrire

## 5.6 Miei oggetti

Qualora l'utente avesse intenzione di vedere la lista degli oggetti da lui inseriti, dal menù a tendina, esso può accedere alla pagina che li visualizza mediante la voce "Miei Oggetti".

La lista presentata in questa schermata è simile al catalogo presente nella home page; ogni oggetto è caratterizzato dalle voci nome, nome del venditore e prezzo, come mostrato in Figura 5.8. Come nel caso del catalogo, il click sull'oggetto permette di visualizzare una panoramica più ampia dei dettagli del prodotto; l'unica eccezione è la presenza del tasto "elimina prodotto". Esso permette di eliminare dal database il medesimo oggetto e, quindi, renderlo non più disponibile allo scambio.



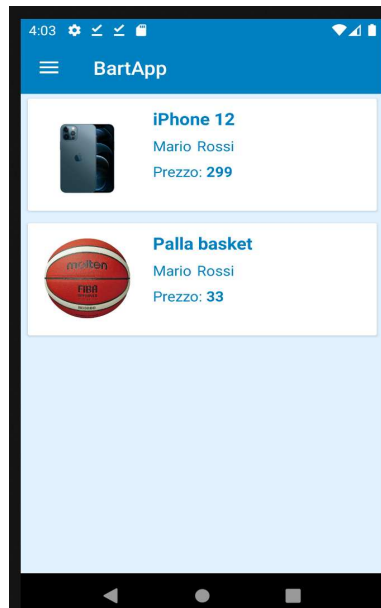


Figura 5.8. Schermata dei Miei oggetti

## 5.7 Inserisci prodotto

Di fondamentale importanza è la sezione che riguarda l’aggiunta dei prodotti. Dal menù laterale è possibile inserire un nuovo prodotto mediante la voce “Aggiungi Prodotto”. L’utente che intende inserire un nuovo prodotto nel catalogo, al click sull’ancora sopra citata, visualizzerà una pagina come quella mostrata in Figura 5.9. Questa pagina richiede una serie di campi obbligatori, quali l’immagine del prodotto, presa dal rullino o scattata sul momento, un nome, una descrizione ed il prezzo virtuale ad esso associato. Il tasto in fondo allo schermo permette l’inserimento del prodotto nel database e, quindi, lo rende visualizzabile nel catalogo della home page. Nel caso in cui qualche campo richiesto non fosse stato validato, l’utente verrà avvisato da un messaggio di errore. Al contrario, se invece tutti i campi sono corretti e validi, egli verrà notificato da un avviso di successo dell’operazione.

## 5.8 Profilo

Facendo swipe a destra dello schermo, accedendo al menù (Figura 5.5), in alto, è presente l’immagine del profilo e il nome dell’utente. Qualora l’utente avesse necessità di visualizzare le informazioni personali e, quindi, di accedere nella proprio area riservata, potrebbe farlo facendo click sulla propria immagine. La Figura 5.10 illustra la pagina che descrive l’area riservata dell’utente, nella quale è presente l’immagine del profilo(ingrandita), il nome utente, l’email e la password.

Facendo click sull’immagine del profilo, l’utente ha la possibilità di cambiare la propria immagine con una foto presa dal rullino o scattata all’istante. Una volta

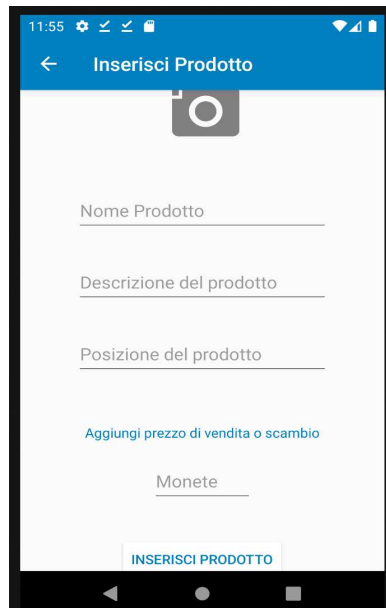


Figura 5.9. Schermata di aggiunta dei prodotti

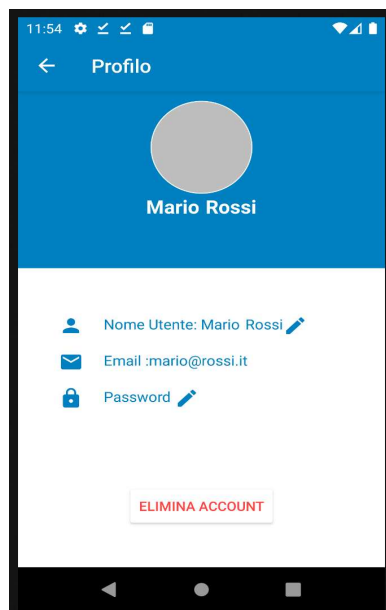


Figura 5.10. Profilo dell'utente

cambiata, è possibile confermare e salvare le modifiche facendo click sul tasto “Salva Modifiche”. Le matite stilizzate presenti sulle voci “nome utente” e “password”, permettono la modifica di questi ultimi. Infatti, se l’utente avesse intenzione di cambiare il nome utente, la matita lo reindirizzerà ad una pagina (Figura 5.11) in cui sarà presente un campo per l’inserimento del nuovo nome utente e un tasto per la conferma. Infine, per la modifica della password, l’operazione da effettuare è simile a quella del nome utente, come mostrato in Figura 5.12. Unica particolarità di questa finestra è la presenza di un ulteriore campo per la conferma della password, il quale verifica se la password inserita rispetta le norme di validazione (lunghezza minima di 6 caratteri). Il tasto in fondo permette la conferma delle modifiche e il reindirizzamento alla pagina del profilo.

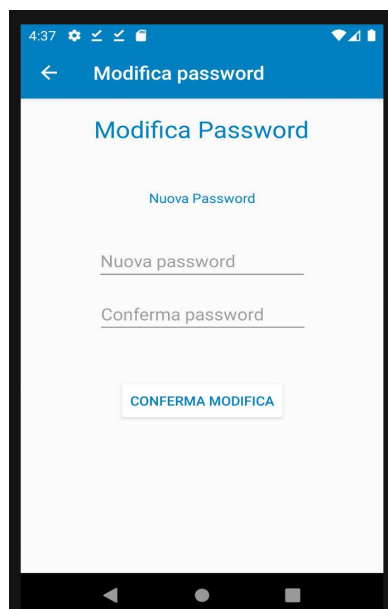


**Figura 5.11.** Schermata per la modifica del nome utente

## 5.9 Offerte inviate e ricevute

Per quanto riguarda la sezione delle offerte che un utente ha proposto e ricevuto, sono presenti due ancore sul menù laterale che indirizzano quest’ultimo in due finestre (Figura 5.13 e Figura 5.14) che visualizzano la lista delle offerte.

Nella voce “Offerte inviate” (Figura 5.13), a differenza del catalogo, questa volta, la riga visualizza ciascun elemento secondo una formattazione differente. Tale riga, quindi, evidenzia i due oggetti (quello proposto e quello da scambiare) con due colori diversi. In particolare, l’oggetto proposto è evidenziato dal colore rosso e quello da scambiare dal colore verde. Come nel catalogo, però, ogni oggetto è caratterizzato anche da un’immagine, un nome, un prezzo ed una posizione.



**Figura 5.12.** Schermata per la modifica della password

Per ogni riga è presente un tasto di “Annulla offerta”, che, come suggerisce esso stesso, sarà preposto all’annullamento dell’offerta, con la conseguente eliminazione di questo elemento dalla lista, nonchè dalla lista dell’utente ricevente.



**Figura 5.13.** Schermata delle offerte inviate dall’utente

Allo stesso modo, per la voce delle “Offerte ricevute”, la finestra che l’utente visualizza è simile. Ogniqualvolta un altro utente è interessato ad un prodotto dell’utente e formula un’offerta, la proposta comparirà nella lista mostrata in Figura 5.14. Anche in questo caso, ogni riga visualizza il nome del prodotto, l’immagine, il prezzo e la posizione. I due colori (rosso e verde) che evidenziano i medesimi prodotti saranno però invertiti. Per ogni elemento di riga, in fondo, sono presenti due tasti che permettono all’utente di accettare o rifiutare l’offerta. In entrambi i casi, al click su di essi, l’utente visualizzerà un avviso sullo schermo che chiederà conferma, come mostrato in Figura 5.15. Fornita tale conferma, si procederà con l’eliminazione di questo scambio dalla lista e da quelle delle offerte inviate da parte del mittente.



Figura 5.14. Schermata delle offerte ricevute

## 5.10 Riepilogo scambi

Qualora l’utente avesse bisogno o necessità di controllare lo storico dei suoi scambi, con la voce “Riepilogo scambi”, egli ha la possibilità di visualizzare tutte le offerte che ha accettato e tutte quelle che sono state da lui inviate ed accettate dal destinatario.

Il riepilogo, come mostrato in Figura 5.16, descrive la lista degli scambi avvenuti, evidenziando i due oggetti scambiati secondo gli stessi colori utilizzati per le offerte. Oltre ad una descrizione minima dei prodotti in questione, l’applicazione mette a disposizione un tasto per accordarsi con l’altro utente. Questo tasto, quindi, permette la comunicazione tra i due utenti, in modo tale che questi ultimi possano



Figura 5.15. Avviso di conferma

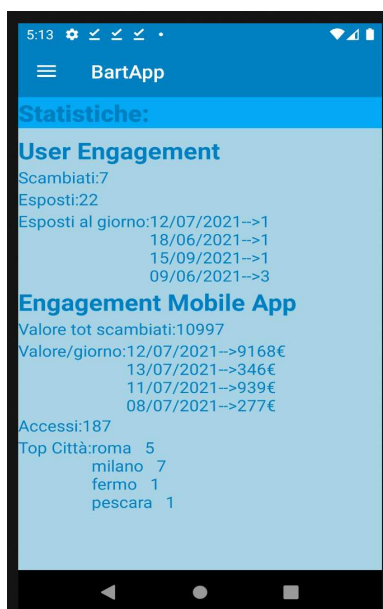


Figura 5.16. Schermata del riepilogo delle offerte accettate

raggiungere un accordo e completare così lo scambio. A corredo del riepilogo, inoltre, è presente un cestino che permette la cancellazione dell'elemento dalla lista, e, quindi, la pulizia dello scambio dallo storico.

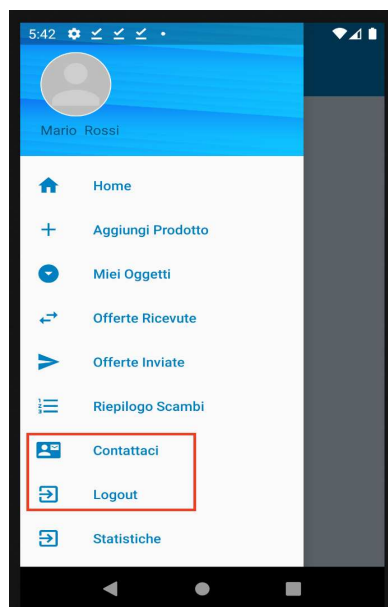
## 5.11 Statistiche

Per i soli utenti admin e sviluppatori, è presente una sezione dedicata alle statistiche relative all'applicazione. Facendo click sulla voce "Statistiche", essi visualizzano una schermata che mostra i dati relativi agli scambi avvenuti nella medesima App. Tale sezione, risulta utile qualora si volesse conoscere e valutare il traffico e il flusso dei dati, per progettare, in futuro, delle offerte o delle promozioni per gli utenti clienti.



**Figura 5.17.** Schermata delle statistiche relative all'applicazione

Questa finestra, come mostrato in Figura 5.17, visualizza, in primo luogo, una sezione in cui sono riportati il numero degli oggetti scambiati, il numero dei prodotti esposti e il numero degli oggetti esposti dai clienti per giorno. La seconda parte, relativa all'engagement dell'applicazione, mostra il flusso totale di denaro accumulato dall'applicazione per mezzo degli scambi avvenuti, il flusso di denaro ordinato per giorno, il numero totale di accessi all'applicazione e una lista che mostra da dove sono avvenuti tutti gli scambi.



**Figura 5.18.** Schermata che visualizza le voci di Logout e Contattaci

## 5.12 Contattaci e Logout

Per finire, sul menù laterale sono presenti le due voci “Contattaci” e “Logout” (Figura 5.18). La prima voce permette di contattare via posta elettronica gli sviluppatori dell’app, nel caso in cui ci fossero delle anomalie o eventuali problematiche. La seconda voce, invece, permette la disconnessione del proprio account dall’applicazione e, quindi, l’utente correttamente autenticato viene riportato alla schermata di Login.



## Confronto con approcci correlati

*In questo capitolo finale, verranno illustrate le analogie dell'applicazione, oggetto della presente tesi, con le altre varie forme di baratto e scambio presenti attualmente. Introduciamo, quindi, il concetto di shopping online per poi scendere nei dettagli con le principali piattaforme di shopping utilizzate al giorno d'oggi. Illustreremo le funzionalità che ogni piattaforma presenta e proporremo un breve confronto con la nostra applicazione.*

### 6.1 Lo shopping online

Al giorno d'oggi, digitando su un qualunque motore di ricerca le parole “shopping online”, verranno proposti circa 4.650.000.000 (caso con motore di ricerca Google) risultati che possono compararsi con ciò che può essere definito “piattaforma di scambio”. Infatti, facendo click su un qualunque link visualizzato, verremo reindirizzati ad una pagina contenente un portale online di vendita. La pandemia globale Covid-19 ha dato una forte scossa al mondo online. Infatti, in poco tempo le abitudini sono cambiate radicalmente e il mercato dell'online è cresciuto esponenzialmente. L'amministratore unico di ITLAB Srl, una web agency che si occupa di progettazione di siti web, dichiarò:

“Stiamo assistendo ad un notevole processo di crescita nell'utilizzo di sistemi informatici da parte di tutti e, al contempo, alla consapevolezza, da parte dell'imprenditore, di non poter più rinunciare ad una presenza online. Per un professionista o un'impresa non essere oggi presente online vuol dire essere fuori dal mondo. L'emergenza in atto ha cambiato abitudini e stili di vita di tutti. Oggi gli acquisti si fanno principalmente online.

Qualsiasi utente, attraverso il suo telefono cellulare, ha sempre Internet in tasca, ci dorme al fianco e, non potendosi spostare liberamente, ha iniziato a comprare online scoprendo vantaggi inimmaginabili. Tutto questo si traduce in una fortissima richiesta di siti web e di sistemi di e-commerce. Gli imprenditori che hanno creduto in passato in queste tecnologie sono stati premiati. Chi non lo ha fatto, cerca di adeguarsi al cambiamento in atto” [8].



**Figura 6.1.** Figura rappresentativa dello shopping online

In termini semplici, quindi, lo shopping online è semplicemente l'acquisto e la vendita di prodotti tramite mezzi come le applicazioni mobili e Internet. Esso ci consente di acquistare e vendere prodotti a livello globale, 24 ore su 24, senza lo stesso sovraccarico che si avrebbe con la corsa al negozio, come rappresentato dalla Figura 6.1.

Esistono varie categorie di e-commerce:

- *E-commerce business-to-business (B2B)*: l'e-commerce B2B rappresenta una transazione che avviene tra due aziende.
- *E-commerce business-to-consumer (B2C)*: il B2C rappresenta la vendita, da parte delle aziende, a categorie di consumatori come le persone.
- *E-commerce consumer-to-consumer (C2C)*: il commercio elettronico C2C avviene quando il consumatore vende direttamente ai consumatori. Più comunemente, questa categoria rappresenta la vendita dell'usato.
- *E-commerce consumer-to-business (C2B)*: il commercio elettronico C2B si materializza quando un consumatore vende o contribuisce denaro ad una società.
- *E-commerce business-to-administration (B2A)*: in questo particolare caso, le transazioni vengono effettuate online tra aziende e Pubblica Amministrazione.
- *E-commerce consumer-to-administration (C2A)*: il commercio elettronico C2A rappresenta tutte le transazioni online tra singoli consumatori e Pubblica Amministrazione o enti governativi. Esempi comuni di questa tipologia di e-commerce sono Istruzione, Tasse e Salute.

### 6.1.1 Vinted

Vinted è un servizio di vendita e scambio online, focalizzato prevalentemente sulla compravendita di abbigliamento e accessori di seconda mano. Esso, inoltre, consente di effettuare transazioni in modo sicuro e senza costo aggiuntivo. Esistono, essenzialmente, due modalità di utilizzo: da browser o tramite l'applicazione pratica e funzionale per cellulari. La Figura 6.2 rappresenta l'icona dell'applicazione per dispositivi mobili.

L'applicazione è stata creata in Lituania nel 2008 da Milda Mitkute e Justas Janauskas ed è stata, poi, lanciata negli Stati Uniti nel 2010. Vinted ora, a distanza di 10 anni, è disponibile in più di 20 paesi [7].



Figura 6.2. Icona di Vinted

Dal punto di vista pratico, utilizzare Vinted è davvero molto semplice. Una volta scelta la modalità di utilizzo (da browser o applicazione gratuita), bisogna procedere con la sottoscrizione di un account. Una volta registrati, si potrà procedere con gli acquisti e la vendita. La vista (da dispositivo browser) che Vinted presenta agli utenti è rappresentata nella Figura 6.3.

The screenshot shows the Vinted home page interface. At the top, there is a teal header with the Vinted logo, a search bar with the text 'Cerca articoli', and buttons for 'Iscriviti | Accedi' and 'Vendi subito'. Below the header, there are navigation links for 'Donna', 'Uomo', 'Bambini', 'Casa', 'Informazioni', and 'La nostra piattaforma'. The main section is titled 'Articoli più richiesti' and features a grid of product listings. Each listing includes a product image, a price tag, a brand name, and a heart icon indicating the number of likes. A 'Venduto' badge is visible on some items. To the right of the grid is a button labeled 'Vedi tutti'. Below the grid, there is a section titled 'Acquista per brand' with a horizontal scrollable list of brand names: Jules, Tape à l'Oeil, Miss Captain, Petit Bateau, Jordan, Pimkie, Decathlon, Timberland, Géo, Nike, Tommy Hilfiger, Pull & Bear, Sergent Major, Cyrillus, Maje, Promod, Esprit, and Kenzo.

Figura 6.3. Home page di Vinted

L'utente che intende acquistare su Vinted può utilizzare la barra di ricerca per

trovare il capo d'abbigliamento di suo interesse, scegliere il capo che vorrebbe comprare e, poi, procedere con l'acquisto utilizzando una fra le varie modalità di pagamento. Più dettagliatamente, nella schermata principale, si può visualizzare un feed con gli articoli consigliati o in evidenza. Qualora si desidera effettuare una ricerca più precisa, l'utente deve fare click sulla voce "Cerca" per poi scegliere gli articoli suddivisi per categorie, quali Donne, Uomini, Bambini o casa. Per ogni categoria, seguiranno una serie di sottocategorie contenenti gli articoli correlati alle medesime. Una volta individuato il prodotto, l'utente può acquistarlo facendo click sulla voce "Acquista".

Qualora, invece, un utente intenda vendere un capo o accessorio, il processo è altrettanto semplice. Vinted segue una procedura lineare e facile da compilare. Durante il processo di vendita sono richieste all'utente tutte le informazioni riguardanti l'oggetto da proporre, in modo tale da renderle poi disponibili e leggibili da un possibile futuro acquirente. Vinted, per dare una migliore visualizzazione agli utenti, chiede diverse informazioni, quali:

1. *Foto*: foto che rappresentano l'articolo proposto;
2. *Titolo dell'articolo*: titolo che rappresenta l'oggetto;
3. *Descrizione dell'articolo*: breve descrizione che caratterizza l'articolo;
4. *Categoria*: categoria a cui l'articolo fa riferimento;
5. *Brand*: descrive la firma dell'oggetto proposto;
6. *Condizioni*: rappresenta lo stato dell'oggetto;
7. *Prezzo*: è il prezzo indicato dal venditore per gli acquirenti;
8. *Voglio scambiarlo*: offre la possibilità di permettere uno scambio con altri venditori.

### Analogie e confronti

Un'analogia individuabile con la nostra applicazione di interesse è quella legata alla registrazione e al suo processo. Come mostrato in Figura 6.4, il format presentato da Vinted è altrettanto semplice e comprensibile da ogni utente. Il processo di registrazione, infatti, prevede pochi campi da compilare, quali:

- Nome utente;
- Email;
- Password.

Inoltre, è altrettanto evidente come l'home page di BartApp abbia varie similitudini con la home page di Vinted. Entrambe le viste visualizzano un feed contenente una serie di articoli in primo piano. Una differenza, però, si nota nella suddivisione che fa Vinted per ciascuna categoria di prodotto. Tale suddivisione è un aspetto molto importante per la catalogazione di dati numerosi. Oltre al filtro di ricerca, il quale risulta indispensabile per le piattaforme di vendita di questo tipo, anche il sistema di assistenza è presente in entrambe le applicazioni. Infatti, in entrambe le piattaforme, sono presenti delle sezioni di assistenza che aiutano l'utente nella compravendita, qualora egli ne avesse bisogno.

The image shows a registration form for Vinted. At the top, the title 'Iscriviti' is centered. Below it are three input fields: 'Nome utente', 'E-mail', and 'Password'. The 'Password' field has a small eye icon to its right. Underneath the fields is a checkbox labeled 'Iscriviti alla nostra newsletter'. Below the checkbox is a line of small text: 'Effettuando la registrazione, confermo di accettare i Termini e condizioni di Vinted, di aver letto la Informativa sulla privacy e di avere almeno 18 anni.' At the bottom of the form is a large teal button labeled 'Iscriviti'. Below the button is a link that says 'C'è qualche problema?'.

**Figura 6.4.** Registrazione di Vinted

### 6.1.2 Ebay

Nel 1995 Pierre Omidyar, laureato in informatica, sviluppò, all'interno del proprio sito, una sezione di aste a distanza. Il sito, che originariamente si chiamava Auction Web, ha riscosso un successo inaspettato. Tant'è, che nel giro di due anni, nel 1997, il sito di Omidyar visualizzò più di 2 miliardi di aste e il nome fu cambiato definitivamente in eBay. eBay, quindi, è una piattaforma online molto simile ad un sito di e-commerce. Esso offre la possibilità di vendere e comprare oggetti agli utenti di ogni regione geografica, a qualsiasi ora e con diverse modalità. La vendita si basa principalmente nell'offerta di un bene o servizio da parte di venditori professionisti e non. Gli acquirenti, d'altra parte, fanno un'offerta per vincere la merce.

La Figura 6.5 illustra la vista visualizzata dalla piattaforma eBay. Essa evidenzia come vengono strutturati e catalogati i prodotti all'interno del sito. La barra di ricerca offre la possibilità agli utenti di cercare il prodotto desiderato per poi procedere con l'acquisto, o l'eventuale offerta, nel caso di un'asta. La Figura 6.6, invece, mostra la vista dettagliata di un prodotto. Un dato rilevante della suddetta vista è la presenza dei dati legati alla spedizione, al tempo di consegna e alle modalità di pagamento. Una volta scelto il prodotto da acquistare, eBay prevede anche la selezione di una tra le varie tipologie di consegna. L'utente, infatti, può decidere se ritirare l'articolo in una delle varie sedi, o riceverlo direttamente a casa.

Tra le varie funzionalità della piattaforma, eBay ha una sezione dedicata alla vendita. Gli utenti, facendo click sulla voce "Vendi", visualizzeranno una pagina dedicata alla vendita di articoli. A differenza dell'applicazione di nostro interesse BartApp, eBay guida l'utente con una serie di domande per inserire l'articolo nella categoria associata. Infatti, l'utente, prima di caricare eventuali foto, descrizioni ed informazioni, è tenuto a scegliere la categoria di appartenenza dell'articolo che intende vendere. Una volta completata tale procedura, egli proseguirà con l'inserimento di tutte le altre informazioni che potrebbero aiutare l'acquirente nella scelta. Una particolarità di questa procedura è che eBay aiuta il venditore nel caricamento del prodotto con domande precompilate e di facile lettura. Esso, inoltre, non prevede

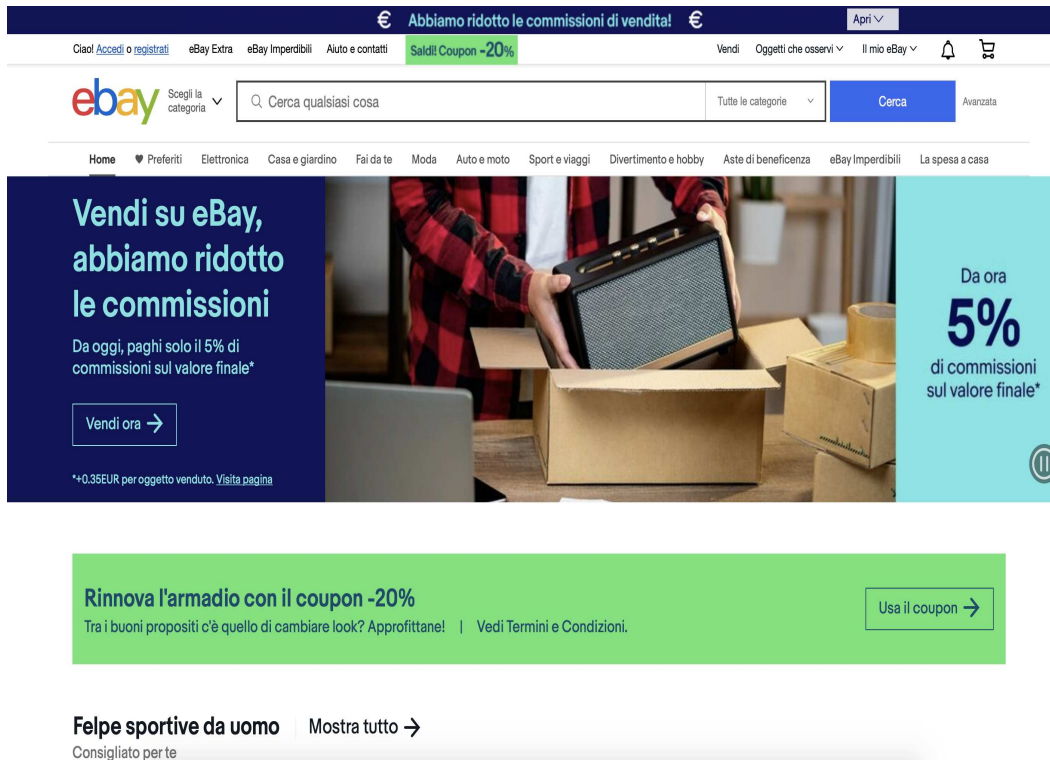


Figura 6.5. Home page di eBay

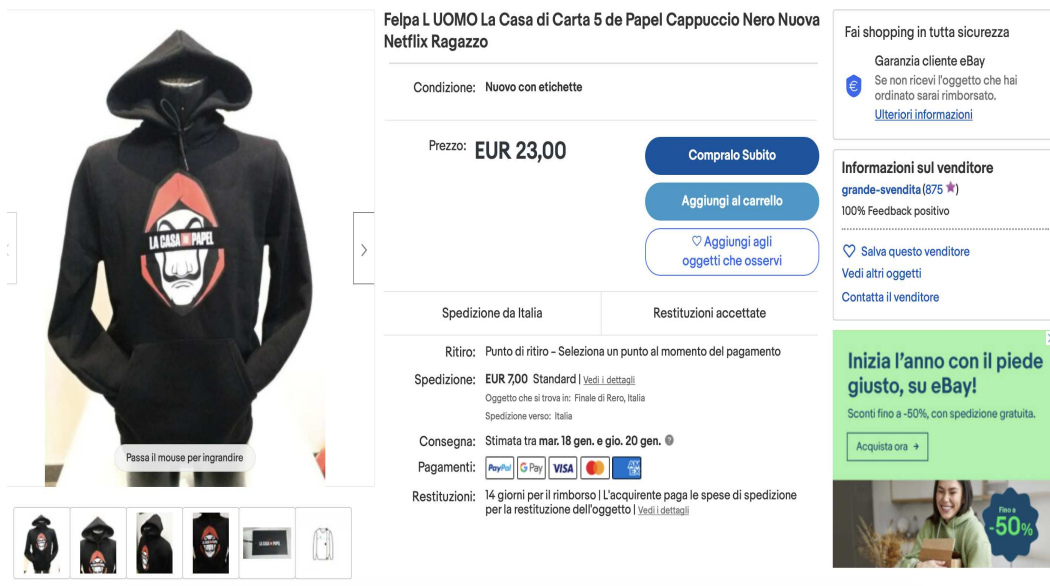


Figura 6.6. Vista prodotto di eBay

alcuna limitazione sulla tipologia e categoria di articolo o bene proposto. Il sito, per di più, nel giro di qualche anno, ha incrementato notevolmente i servizi di sicurezza e assistenza per il cliente. Ogni prodotto, infatti, è soggetto ad una “garanzia cliente eBay” che, in caso di articolo difettoso o non ricevuto, prevede un rimborso per l’acquirente.

### La differenza fra scambio e vendita

Contrariamente all’applicazione oggetto della presente tesi, eBay non prevede il servizio di scambio. Gli utenti della piattaforma non possono, quindi, proporre un proprio articolo per aggiudicarsene uno già disponibile sul sito. L’attività di scambio, per definizione, consiste nel trasferimento di un bene o servizio da una persona all’altra in cambio di un altro bene. L’app BartApp, come già analizzato nei capitoli precedenti, è incentrata completamente sul concetto di scambio. Contrariamente, la vendita, come mostrato nella Figura 6.7, è un contratto attraverso il quale la proprietà di un bene è ceduta ad altri sotto un determinato prezzo.

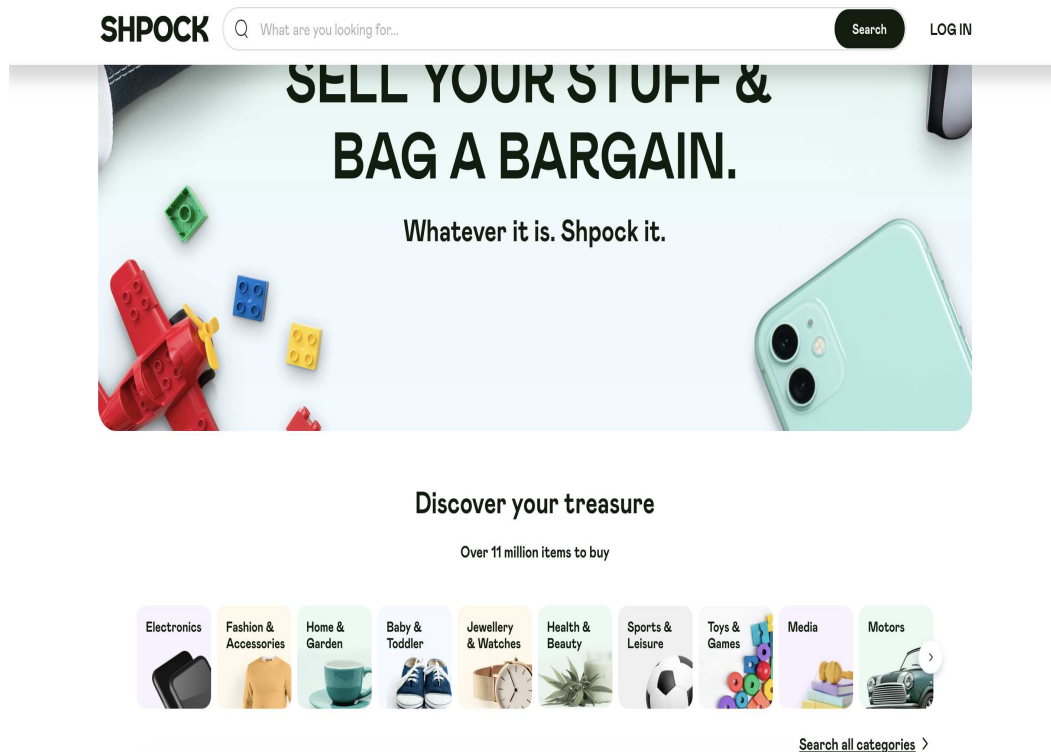


**Figura 6.7.** Rappresentazione illustrativa del concetto di vendita

### 6.1.3 Shpock

Con oltre 10 miliardi di utenti, lanciata nel Settembre del 2012, Shpock è diventata una delle app più popolari in Europa. Germania, Regno Unito, Austria e Italia sono le nazioni con il maggior numero di download ed utenti iscritti. Shpock mostra ai visitatori un feed con articoli in base alla localizzazione geografica. In questo modo, questi ultimi possono accedere a numerosi articoli in offerta nelle vicinanze e accaparrarsi la merce. Per la maggior parte dei casi, si tratta di articoli vintage o di oggetti usati. Secondo Filippo Vendrame, come citato nel suo articolo legato alle nuove applicazioni di vendita, Shpock funziona come il vecchio mercatino delle pulci [16]. Con qualche info, una foto ed una breve descrizione, gli utenti possono pubblicare e rendere disponibile qualsiasi articolo desiderano vendere. Inoltre, tale piattaforma consente loro di accordarsi per un’eventuale negoziazione. Essi, infatti, per mezzo di chat private, possono concordare il prezzo di un articolo e organizzarsi per la vendita. Come rappresentano in Figura 6.8, Shpock non presenta direttamente gli articoli disponibili sul sito, ma bensì lascia scegliere all’utente la categoria che

intende scandagliare. Una volta selezionata la categoria, la vista è del tutto simile alla home page dell'app BartApp. Infatti, in essa, è presente una griglia contenente tutti gli articoli disponibili per la medesima categoria. Una volta deciso il prodotto, l'utente può procedere con la proposta di un'offerta.



**Figura 6.8.** Vista di Shpock

L'accordo diventa legalmente vincolante quando entrambe le parti accettano l'affare. Questo sistema di proposte rispecchia, di fatto, pienamente i principi su cui è stata sviluppata la nostra applicazione. Il venditore, che riceve la proposta da parte dell'acquirente, è tenuto ad accettare l'offerta per concludere l'affare.

La Figura 6.9 mostra dove l'utente deve fare click per procedere alla vendita di un articolo che intende pubblicare.



**Figura 6.9.** Tollbar per la vendita su Shpock

La pagina che il sistema visualizzerà sarà composta da una serie di campi ed una serie di form che l'utente deve compilare per procedere con il caricamento



dell'inserzione. Durante la fase di caricamento, Shpock chiede di inserire:

- Foto (per un massimo di 5 foto);
- Titolo;
- Descrizione;
- Prezzo (con la relativa scelta di valuta);
- Categoria;
- Località.

Terminata tale procedura, e completati correttamente tutti i campi, l'articolo sarà immediatamente disponibile e visibile sul sito.

#### 6.1.4 Subito.it

Subito.it è nato nel 2007 a Milano e, al giorno d'oggi, è la piattaforma italiana di compravendita più utilizzata nel nostro paese. Più precisamente, esso rappresenta un servizio di annunci web che conta, al suo interno, più di 5 milioni di annunci disponibili. I dati forniti dal settore di marketing di Notizie.it ci indicano che il sito conta più di 9 milioni di utenti visitatori e circa 150 mila nuovi annunci al giorno [6]. Inizialmente nata come piattaforma web, essa si è sviluppata anche in forma mobile, come App.

Seppur facendo parte di una multinazionale norvegese che, ad oggi, opera in 29 Paesi del mondo, Subito.it ha caratteristiche ben definite in ambito italiano. Infatti, come rappresentato nella Figura 6.10, la home page ha la caratteristica non comune di chiedere all'utente la propria regione geografica. Oltre all'articolo desiderato e alla categoria di appartenenza, l'utente, infatti, deve selezionare la regione d'Italia su cui fare la ricerca. Questa scelta, come spiegato dai fondatori del sito in un'intervista al Businesscommunity, è stata fatta per venire incontro alle necessità degli utenti italiani che sono legati al territorio.

Dal punto di vista pratico, l'inserimento di un annuncio sul sito risulta più complesso rispetto al caricamento di un prodotto sulla nostra applicazione BartApp. Infatti, Subito.it chiede all'utente il nome dell'articolo e la categoria di appartenenza. Una volta selezionata la giusta categoria, l'utente deve riempire i campi precompilati dal sito, quali:

1. *Categoria*: rappresenta la categoria di appartenenza dell'articolo;
2. *Tipo di annuncio*: indica se l'utente intende vendere, cercare o regalare il prodotto;
3. *Sottocategoria di appartenenza*: rappresenta la sottocategoria dell'articolo;
4. *Condizione*: indica lo stato del prodotto;
5. *Immagine (facoltativa)*: Subito.it chiede se l'utente intende caricare foto o immagini dell'articolo per facilitarne la vendita;
6. *Titolo dell'annuncio*: rappresenta il nome del prodotto che l'utente vuole caricare sul sito;
7. *Testo dell'annuncio*: rappresenta la descrizione dettagliata dell'articolo;
8. *Prezzo (facoltativo)*: indica il prezzo facoltativo dell'articolo;
9. *Modalità di vendita*: indica se l'utente intendere vendere l'oggetto a distanza;

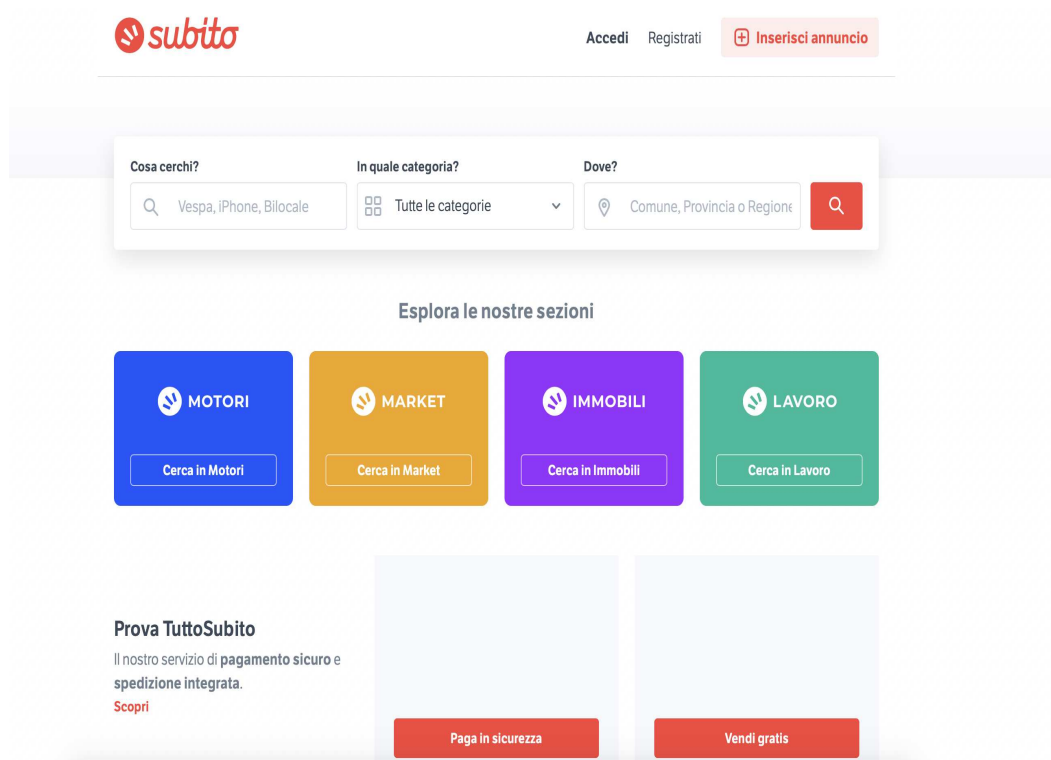


Figura 6.10. Home page di Subito.it

10. *Metodo di spedizione*: consente all'utente di scegliere la spedizione offrendo un servizio gestito da Subito.it oppure organizzarsi autonomamente;
11. *Dimensioni del pacco*: l'utente è obbligato a selezionare una delle voci presenti tra Piccolo, Medio e Grande, le quali rappresentano la dimensione totale del pacco;
12. *Comune*: indica il comune di residenza dell'utente;
13. *Indirizzo*: rappresenta l'indirizzo civico dell'utente;
14. *Sono un/una*: indica se l'utente è una persona fisica o un'azienda;
15. *Telefono*: rappresenta il numero di telefono dell'utente.

Una volta compilati tutti i campi, l'utente, cliccando sul pulsante "Continua", può ricontrollare i dati per poi procedere con la pubblicazione nel sito. Una differenza evidente con la nostra applicazione di interesse, riguarda le varie voci dell'inserimento di un annuncio/prodotto. Mentre Subito.it presenta dei format ben precisi che guidano e gestiscono a priori l'eventuale vendita dell'articolo, la nostra applicazione lascia la libertà ai due utenti (venditore e acquirente) di accordarsi tra di loro. Inoltre, come in BartApp, pubblicare un annuncio su Subito.it è gratuito. Tuttavia, Subito.it offre in più, agli utenti, la possibilità di aumentarne la visibilità con delle opzioni a pagamento. Con poche decine di euro, infatti, l'utente può posizionare il proprio articolo in cima alla lista degli annunci. L'acquisto, invece, segue una procedura molto più semplice. Una volta che l'utente ha ricercato l'articolo desiderato,

inserendo opportunamente la categoria del prodotto e la regione di ricerca, egli può procedere con l'acquisto cliccando sul pulsante "Acquista". Dopo il click, egli può decidere se acquistare l'articolo al prezzo già definito o se inviare una proposta al venditore. In tal caso, sarà, poi, compito del venditore valutare se accettare o meno l'offerta ricevuta.

### 6.1.5 Amazon

Amazon è stato sviluppato nel 1994 a Seattle da Jeff Bezos e, inizialmente, era poco più di una libreria online. Con il progresso ed il passare degli anni, esso ebbe una crescita esponenziale, diventando oggi il più grande rivenditore di e-commerce online. Amazon, per mezzo di innovazioni che hanno ampliato la propria offerta di mercato a beni e servizi, grazie a migliorie al sito e partecipando al mercato dei video in streaming, è riuscito ad operare anche nel settore bancario, diventando, così, al giorno d'oggi, una vera e propria multinazionale. Infatti, secondo la rivista di "Ecommerce Platforms", essa è la quarta azienda per valore al mondo, superata solo da Alphabet, Apple e Microsoft [14].

Il marchio di Amazon, rappresentato in Figura 6.11, è conosciuto in quasi tutti i Paesi del mondo. Il sito, infatti, è accessibile da utenti di ogni regione geografica e spedisce ad ogni utente di qualsiasi Stato ad eccezione di:

- Corea del Nord;
- Cuba;
- Iran;
- Iraq;
- Siria;
- Sudan.



**Figura 6.11.** Logo di Amazon

La vista che propone Amazon agli utenti visualizzatori, come rappresentato in Figura 6.12, è una tra le più articolate fra i vari siti di e-commerce. L'utente può sfogliare il catalogo delle categorie presenti sulla piattaforma e selezionare quella che desidera esaminare. Inoltre, nella home page, sono presenti le offerte maggiormente in sconto e tutti gli articoli più venduti o visualizzati negli ultimi periodi.

Per accedere a tutte le funzionalità e ai servizi che Amazon offre, gli utenti, però, devono creare un account e registrarsi al sito.

Come illustrato nella Figura 6.13, il processo di registrazione su Amazon è del tutto analogo a quello utilizzato nella nostra applicazione BartApp. Gli utenti, infatti, per registrarsi al sito, devono compilare pochi campi, quali: nome, numero

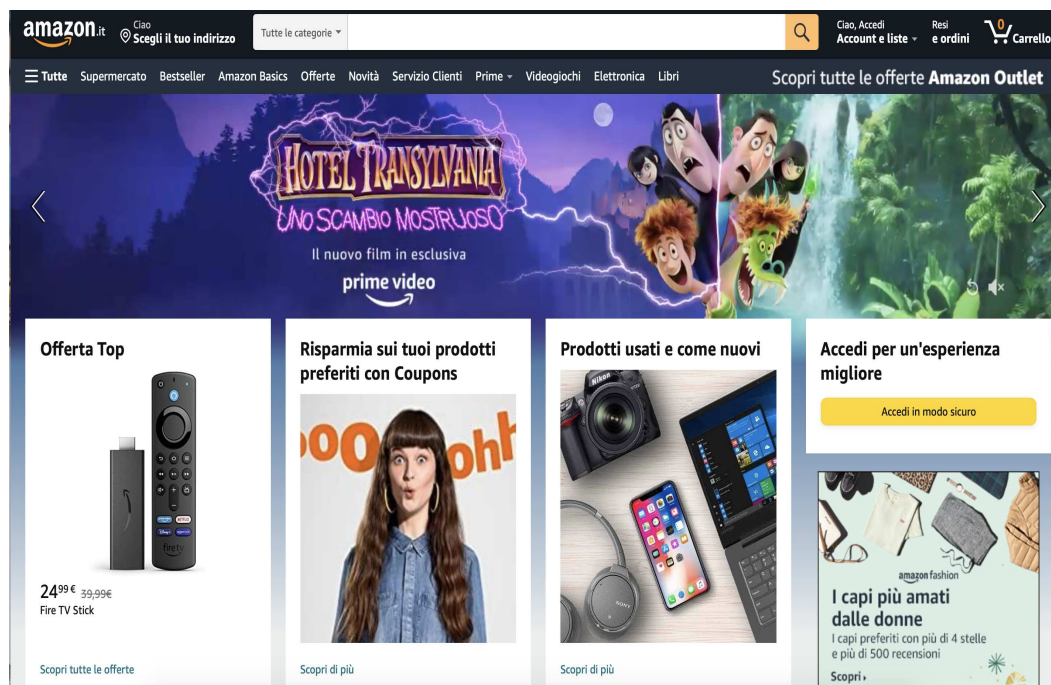


Figura 6.12. Home page di Amazon

di cellulare o email ed una password. Una volta creato l'account, questi ultimi hanno la possibilità di acquistare e vendere prodotti e di accedere a tutti i servizi che la piattaforma propone. Come nell'applicazione BartApp, anche in Amazon è presente una vista dettagliata dei prodotti. Infatti, una volta individuato l'articolo tra la lista visualizzata, l'utente può leggerne i dettagli facendo click sul prodotto stesso. La pagina che egli visualizzerà, conterrà, oltre alle specifiche sull'articolo e ai dettagli sulla spedizione, una serie di prodotti simili a quello visualizzato. In Amazon, infatti, sono presenti algoritmi e sistemi di ricerca automatica che, come in questo caso, non sono presenti sulla nostra applicazione BartApp, e permettono di mostrare all'utente tutti gli articoli simili a quello visualizzato.

Più dettagliatamente, per fare acquisti su Amazon, l'utente, una volta collegato al sito, deve inserire il nome del prodotto che desidera nella barra di ricerca. Egli visualizzerà una serie di risultati sotto forma di lista. Una volta selezionato l'articolo, l'utente può acquistarlo direttamente facendo click su "Acquista ora". Se, invece, egli vuole aggiungere il prodotto in questione al carrello e proseguire con lo shopping, può farlo facendo click sulla voce "Aggiungi al carrello". Completato l'ordine, egli può terminare l'acquisto procedendo con l'inserimento delle informazioni sulla spedizione e sul metodo di pagamento. Amazon non prevede la possibilità, da parte dell'acquirente, di fare offerte o proposte al venditore, contrariamente a quanto avviene con BartApp.

Per quanto riguarda la vendita, invece, Amazon adotta tutt'altra procedura. Come spiegato nella guida offerta dalla stessa piattaforma, la prima cosa da fare è quella di scegliere il piano di vendita. Amazon, infatti, permette la vendita sulla

**amazon**.it

## Creare account

**Il tuo nome**

**Numero di cellulare o e-mail**

**Password**

Almeno 6 caratteri

*i* Le password devono essere composte almeno da 6 caratteri.

**Verifica password**

**Continua**

Registrandoti dichiari di aver letto e accetti integralmente le nostre [Condizioni generali di uso e vendita](#). Prendi visione della nostra [Informativa sulla privacy](#), della nostra [Informativa sui Cookie](#) e della nostra [Informativa sulla Pubblicità definita in base agli interessi](#).

Disponi già di un account [Accedi >](#)  
 Acquisti per lavoro? [Crea un business account gratuito](#)

**Figura 6.13.** Pagina di registrazione di Amazon

propria piattaforma a tutti i clienti, ma ad un costo che varia a seconda della modalità scelta. Una volta che l'utente ha selezionato la modalità di vendita, egli è tenuto a valutare la strategia di vendita. Amazon offre all'utente una serie di informazioni riguardanti marchi, tipologia di utenza e margini di guadagno. Dopo aver valutato la propria strategia di vendita, l'utente è tenuto a creare un account venditore. Egli, può decidere se utilizzare il proprio account Amazon, o se creare un nuovo account venditore con la propria email aziendale. I campi richiesti, in questo caso, per il processo di registrazione sono:

- Indirizzo postale aziendale o account cliente Amazon;
- Carta di credito addebitabile (sono accettate anche carte di credito internazionali);
- Passaporto valido o documento d'identità nazionale;
- Dati di registrazione dell'azienda (con numero di partita IVA);
- Numero di telefono;
- Conto bancario per ricevere i guadagni delle vendite.

Una volta terminata la registrazione, Amazon mostra all'utente una pagina contenente i dettagli sull'IVA. A seconda del tipo di attività che l'utente svolge, Amazon richiederà a quest'ultimo di registrarsi per ottenere un numero di partita IVA in un Paese europeo.

Seppur alquanto articolato, il processo di vendita offerto da Amazon è uno dei migliori previsti dai siti di e-commerce. A differenza dell'applicazione BartApp, oggetto della presente tesi, dove l'utente che intende pubblicare un articolo è tenuto a riempire dei semplici campi, il venditore Amazon, pur avendo una procedura molto più complessa, è guidato passo passo nella procedura con informazioni dettagliate per ogni punto.

## Conclusioni

In questa tesi è stata esposta un'applicazione Android, per dispositivi mobili, che consente di scambiare oggetti tra utenti. Nella presentazione della tesi sono stati presentati gli sviluppi della progettazione e tutte le funzionalità che l'applicazione propone. Partendo dal sistema su cui è stata progettata l'applicazione, il lettore può ripercorrere la storia dell'evoluzione di Android e la sua struttura. Il lettore, infatti, può ricercare tra le pagine per comprendere al meglio il sistema operativo e quali sono i vantaggi e le motivazioni che ci hanno spinti a sviluppare in Android, piuttosto che in altri ambienti. Partendo dalle funzionalità che il sistema operativo offre ai clienti, abbiamo, poi, introdotto gli strumenti che esso stesso propone, come Firebase e Android Studio. Successivamente è stato presentato il principio su cui è stata sviluppata l'applicazione, ovvero il baratto. Il lettore, infatti, può ripercorrere tutta l'evoluzione dei sistemi di scambio, dai più antichi a quelli più moderni, come, appunto, le applicazioni, per poi proseguire la lettura nella nascita della moneta ed il concetto di scambio tra produttore e consumatore. Il capitolo successivo permette al lettore di comprendere più dettagliatamente com'è stata sviluppata e strutturata la fase di progettazione dell'applicazione. Questo capitolo, infatti, ci ha permesso di mostrare quali sono state le motivazioni che ci hanno spinto ad operare secondo una determinata impostazione. Infatti, partendo dalla descrizione dell'applicazione, siamo poi scesi nei dettagli introducendo i diagrammi di flusso e dei casi d'uso nonché i requisiti funzionali e non funzionali. Una particolare attenzione, poi, è stata data alla struttura dei dati nonché alla sua formattazione, per permetterci, in seguito, di introdurre il concetto di base di dati (database) ed il software Firebase, utilizzato in questo caso.

Terminata la spiegazione della fase progettuale, ci siamo soffermati sui dettagli pratici dello sviluppo. Quindi, partendo dall'implementazione dell'app BartApp, correlata con i rispettivi codici e descrizioni, abbiamo poi spiegato più dettagliatamente il funzionamento delle singole istruzioni. In cascata a questo capitolo, abbiamo inserito il manuale utente. Il lettore, infatti, può comprendere punto per punto le singole funzioni e le funzionalità offerte dall'app. Il capitolo, per facilitare la guida e la comprensione al lettore, è correlato di immagini che rappresentano l'applicazione stessa ed una descrizione dettagliata della medesima. Infine, la tesi si conclude con un confronto con app e piattaforme similari alla nostra applicazione, come Vinted, Ebay, Shpock, Subito.it ed Amazon. Infatti, nel capitolo finale, dopo una

breve descrizione del concetto di shopping online, sono state presentate 5 piattaforme/applicazioni che si basano sui principi del baratto e la compravendita. Inoltre, in questo capitolo, sono state illustrate le analogie e le differenze delle medesime piattaforme con la nostra applicazione. Tra le varie differenze che si possono notare, una importante è legata al sistema di pagamento. Una miglioria apportabile a questo progetto, infatti, è lo sviluppo di un sistema che permette la gestione dei pagamenti e dei soldi “virtuali” degli utenti. Possono essere ultimati, anche, i meccanismi di ricerca, offrendo agli utenti una ricerca più precisa con l’inserimento delle categorie associate ai prodotti, in modo tale da semplificare la scoperta degli articoli desiderati. In aggiunta, come in quasi tutte le applicazioni e siti web, lo sviluppatore che intende proseguire il progetto con delle migliorie, può inserire delle facciate e degli spazi adibiti a locandine pubblicitarie, le quali, poi, permetterebbero una remunerazione economica. Essendo un’applicazione sviluppata in ambito scolastico, e, quindi, puramente istruttiva, abbiamo deciso di non incentrare lo sviluppo con queste accortezze, che, però, sono di fondamentale importanza per una piattaforma di compravendita.

Un progetto di questo tipo, corredato di tutte le informazioni necessarie alla progettazione e alla realizzazione, inoltre, può offrire ai lettori la possibilità di realizzarne altrettanti nuovi, specifici per aziende o per chiunque voglia implementare una propria piattaforma di vendita.



---

## Riferimenti bibliografici

1. Cos'è firebase. <https://www.geekandjob.com/wiki/firebase>, 2019.
2. Baratto. <https://www.zerorelativo.it/il-baratto.php>, 2021.
3. Category:mobile. <https://wikimediafoundation.org/news/category/technology/mobile/>, 2021.
4. Il baratto. [https://www.wiki.it-it.nina.az/Economia\\_del\\_baratto.html](https://www.wiki.it-it.nina.az/Economia_del_baratto.html), 2021.
5. Le criptovalute. <https://www.consob.it/web/investor-education/criptovalute>, 2021.
6. Subito.it: come funziona. [https://www.notizie.it/subito-it-come-funziona-come-vendere-e-acquistare/?refresh\\_ce](https://www.notizie.it/subito-it-come-funziona-come-vendere-e-acquistare/?refresh_ce), 2022.
7. Vinted. <https://www.vinted.it/about>, 2022.
8. Ansa. Boom degli e-commerce e delle vendite online. [https://www.ansa.it/pressrelease/tecnologia/2021/01/25/boom-degli-e-commerce-e-delle-vendite-online.come-sono-cambiati-gli-acquisti-online-durante-covid-19\\_8600928d-6513-454a-ba8b-2f4a856a3598.html](https://www.ansa.it/pressrelease/tecnologia/2021/01/25/boom-degli-e-commerce-e-delle-vendite-online.come-sono-cambiati-gli-acquisti-online-durante-covid-19_8600928d-6513-454a-ba8b-2f4a856a3598.html), 2021.
9. P. Atzeni, S. Ceri, P. Fraternali, S. Paraboschi, and R. Torlone. *Basi di dati*. McGraw-Hill, 2019.
10. BCE. Cos'è la moneta? [https://www.ecb.europa.eu/ecb/educational/explainers/tell-me-more/html/what\\_is\\_money.it.html](https://www.ecb.europa.eu/ecb/educational/explainers/tell-me-more/html/what_is_money.it.html), 2021.
11. F. Gadducci. Introduzione a Unix. <http://pages.di.unipi.it/gadducci/PR1L-13/shell/introUnix.pdf>, 2021. Storia di Unix.
12. T. Hagos. *Learn Android Studio 3 – Efficient Android Development*. Appress, 2018.
13. Borsa Italiana. Bitcoin. <https://www.borsaitaliana.it/notizie/sotto-la-lente/bitcoin-172.htm>, 2021.
14. Ecommerce Platforms. Amazon, 2022.
15. A. Silberschatz and P. B. Galvin. *Sistemi Operativi*. Pearson, 2019.
16. Filippo Vendrame. shpock, come funziona, commissioni e acquisti. <https://download.html.it/tutorial/shpock-funziona-commissioni-acquisti/>, 2022.



---

## Ringraziamenti

Arrivare qui, a scrivere questo capitolo finale della tesi, è un'emozione che non si può nemmeno spiegare. Proprio per questo, in primis, voglio ringraziare me stesso. Voglio fare i complimenti a me stesso, perchè nonostante tutti i bassi avuti in questo percorso, sono riuscito a chiudere questo capitolo della vita. Nonostante il Covid-19, nonostante la depressione, nonostante tutti i momenti in cui io abbia fatto fatica anche solo ad aprire i libri, sono riuscito a raggiungere questo traguardo. E proprio perchè è stato un percorso difficile, pieno di insidie, dolori e sacrifici, che ora, arrivare a questo punto, sembra ancora più bello. Un ringraziamento speciale va anche ai miei migliori amici. Ringrazio Andrea, per le giornate in biblioteca a studiare, anche se poi, lo studio era veramente poco. Ringrazio il mio amico Danny, introverso ma, in cuor suo, sempre presente ogni qual volta avessi bisogno di una mano. Non può mancare all'appello Marco, che nonostante la sua vita altalenante, era sempre il primo a capire quando c'era qualcosa che non andava e chiedere spiegazioni. Ringrazio anche Chiara, che seppure entrata nella mia vita da pochi mesi, mi è sempre stata accanto dandomi la forza necessaria per non mollare e terminare i miei ultimi e faticosi esami.

E' stato un percorso lungo, ma in questi anni, ho assunto una consapevolezza delle mie capacità che non pensavo di avere. La scuola, sin da piccoli, è uno strumento che oltre insegnare, aiuta a crescere e prendere conoscenza di se stessi. Affrontare gli esami significa anche comprendere le proprie capacità, saper dialogare in pubblico, saper gestire emozioni ed ansia. Non posso non citare il mio Professore, nonché relatore, Domenico Ursino, che ringrazio per la sua infinità disponibilità offerta, pazienza e tranquillità nell'affrontare i problemi. Ringrazio anche il suo collaboratore Enrico Corradini che si è mostrato sempre disponibile nel darci una mano per lo sviluppo di quest'applicazione. Devo citare anche i miei compagni di corso Nicola e Mattia, con i quali appunto, abbiamo sviluppato questa applicazione.

Ringrazio, infine, la mia famiglia, che mi è stata sempre accanto. Ringrazio, quindi, i miei genitori e mio fratello Gianluca, che ogni qual volta stavo per mollare, mi ridavano la forza necessaria per continuare. Vorrei spendere due parole anche per la mia psicologa Sara, che in questi ultimi mesi, è stata in grado di aiutarmi a gestire le mie preoccupazioni e farmi portare a termine questo percorso. Finalmente ce l'ho fatta! Non posso che essere fiero di me stesso! E lo devo anche a tutti voi. Grazie di cuore a tutti, grazie davvero.