Progettazione e Sviluppo di un'Applicazione Mobile di Marketing ed E-Commerce

Tommaso Berlose

2016-11

Indice

Intro

Il progetto MyGelato nasce nel 2014, da un'idea dell'azienda Carpigiani, come piattaforma di e-commerce e marketing digitale fruibile tramite applicazione mobile. Sviluppato inizialmente da terzi, a causa del cambiamento dei requisiti funzionali e di progettazione, viene poi riportato internamente all'azienda

Il progetto MyGelato, applicazione mobile ideata dall'azienda Carpigiani, nasce nel 2014 come piattaforma di marketing ed e-commerce sviluppata da terzi. Il cambiamento dei requisiti funzionali della piattaforma ha portato a far rinascere il progetto iniziale in un prodotto finale completamente gestito internamente all'azienda. Questa tesi presenta il design e l'implementazione di una piattaforma fruibile da web e dispositivi mobile con lo scopo di supportare un alto numero di utenti mantenendo controllati i costi e le performance. Le tecnologie scelte e la generalità della soluzione consentono di impiegare gli stessi principi per altri progetti con architetture analoghe.

Capitolo 1

Contesto e Finalità

Per comprendere nella sua complessità il progetto elaborato in questa tesi è necessario tenere conto, prima di tutto, del contesto in cui viene sviluppato e delle finalità implicite ed esplicite che si vogliono raggiungere. Si tratta di descrivere lo sviluppo di un applicativo mobile fortemente legato ai sistemi di marketing e alle nuove strategie di business di aziende leader all'interno dei propri mercati, in molti casi saturi o fortemente instabili, scossi dalle innovazioni tecnologiche di questi decenni.

Oltre alle ingerenze esterne sul progetto, come il cambiamento repentino delle tecnologie o la presenza di eventuali competitor, deve essere valutata la posizione del singolo componente all'interno dello sviluppo dell'interno progetto, la quale va tenuta in forte considerazione specialmente durante la prima fase di sviluppo in cui si determinano le prime linee guida da seguire durante tutto il lavoro.

Questo elaborato rientra nel concetto di *Application Economy*, che descrive perfettamente il trend degli ultimi anni. Il cambiamento dei metodi con cui le masse ottengono informazioni e si lasciano influenzare hanno portato a un sostanziale sconvolgimento del sistema di marketing e delle strategie commerciali anche di aziende multinazionali, spostando l'interesse sulla ricerca nel campo delle appli-

cazioni mobile. Proprio in questo contesto è necessario inquadrare le motivazioni che hanno portato Carpigiani, azienda leader nel settore della produzione di macchine per il gelato, a dare vita all'ecosistema MyGelato, di cui questa tesi sviluppa solo una componente, valutando sia le finalità in ambito di marketing sia in ambito commerciale e produttivo.

1.1 Application Economy

Figura 1.1: Application Economy

Non vi è forse modo di descrivere la società attuale e questo periodo storico senza valutare l'importanza dello sviluppo tecnologico che ci ha portato in quella che viene definita l'*Era dell'Informazione*. La rivoluzione tecnologica che sta avvenendo in questi anni, specialmente a partire dagli anni Novanta, ha portato la connessione globale a Internet ad assumere un ruolo essenziale in ogni aspetto della società moderna e della nostra vita.

È cambiato drasticamente il modo con cui le persone accedono alle informazioni, che siano queste di tipo personale o di tipo commerciale. Allo stesso modo si sono dovute adeguare le strategie di tutte quelle aziende che hanno visto cambiare in maniera drastica il proprio mercato, invaso molte volte da tecnologie sempre più diversificate e innovative.

Secondo il giornalista Paul Mason, è stata la dottrina economica degli ultimi decenni della nostra epoca che da una parte ha avuto il merito di promuovere la più grande ondata di sviluppo economico che il mondo abbia mai visto, ma dall'altra ha portato a mercati incontrollati e a vorticosi cambiamenti sociali innescati dalla tecnologia. Si sono diffusi, infatti, concetti come i progetti open-source, la sharing economy e le licenze creative commons che hanno messo in crisi le fondamenta del capitalismo odierno. Questo fenomeno unito alla saturazione di molti mercati ha portando tante aziende a dover rivedere la propria business strategy obbligandole a dirigere i propri investimenti sulla diversificazione e sulle nuove strategie di vendita. Paul Mason. *Postcapitalismo.* 2016

C'è stato un cambio di paradigma per tutti i meccanismi di commercializzazione di un prodotto: sono diventati di fondamentale importanza i concetti di *e-commerce* e *marketing digitale*, fortemente spinti dalle tecnologie e dalla nuova possibilità di accedere a una risorsa comune (Internet) da parte della maggior parte della persone anche di cultura, età e ambienti sociali diversi. L'utente medio, per esempio, viene ormai maggiormente raggiunto e conivolto tramite email pubblicitarie o advertising online rispetto a meccanismi tradizionali ormai meno incisivi di marketing non digitale.

Legato alla diffusione sempre crescente di smartphone, che sono di fatto diventati l'accesso principale degl utenti a Internet, nasce quindi l'*Application Economy*: lo sviluppo e l'utilizzo di applicazioni mobile per raggiungere e dialogare tramite

un collegamento bidirezionale con i consumatori. Tramite un applicativo mobile è quindi possibile svolgere una serie di azioni online in mobilità e con semplicità; potendo confidare, nella maggior parte dei casi, in un'alta sicurezza della propria connessione e nello scambio dei dati.

Se questa strategia può essere applicata, per esempio, in ambito marketing per pubblicizzare un proprio prodotto e fidelizzare il consumatore alla propria azienda, è forse più incisivo osservare come l'e-commerce abbia ormai soppiantato la commercializzazione tradizionale in molti ambiti, specialmente tramite applicativi mobile (una statistica del BI Intelligence riporta che entro il 2020 il commercio mobile supererà il 45% degli acquisti online complessivi). L'abbattimento dei costi di un rivenditore non fisico e la nascita di servizi di pagamento online come PayPal hanno portato questa rivoluzione in ogni ambito commerciale permettendo, inoltre, all'utente di pagare direttamente tramite sistemi bancari online in sicurezza e con semplicità.

Tutti questi concetti sono ormai più che affermati in ambiti come la vendita di prodotti di abbigliamento o tecnologici, ma l'azienda Carpigiani ha scelto di portare questi nuovi meccanismi anche all'interno della vendita di Gelati, in modo innovativo e fortemente avanti nei tempi grazie al progetto *MyGelato*.

1.2 Ecosistema MyGelato

L'ecosistema MyGelato si pone all'interno del contesto descritto nel capitolo precedente, a unire le strategie di marketing dell'azienda produttrice Carpigiani e delle singole gelaterie convenzionate: una piattaforma web e mobile che si rivolge ai gelatieri e ai consumatori di gelato. Gli obiettivi commerciali sono diversi, primi tra i quali quello di incentivare la compravendita di gelati tramite un sistema di coupon

digitali e quello di promuovere il consumo di gelato tramite offerte, fornendo anche ai gelatieri un sistema semplice ed efficace per pubblicizzare il proprio negozio.

Attraverso l'applicativo, l'utente può informarsi sulle gelaterie presenti in zona verificando anche quali facciano parte del circuito MyGelato: insieme delle gelaterie convenzionate alla vendita e alla validazione dei coupon online. È possibile ottenere alcune informazioni utili, come indirizzo e numero di telefono, salvarle nei preferiti e verificare se vi sono promozioni attive. Sono strategie ovviamente di carattere pubblicitario e permettono anche a gelaterie minori, facenti però parte del circuito, di essere pubblicizzate agli utenti che utilizzano la piattaforma.

Questo sistema permette all'azienda Carpigiani di avere un controllo maggiore su un sistema centralizzato di advertasing per le gelaterie di cui è principale fornitore, ottenendo quindi la possibilità di intervenire su realtà locali standardizzandone alcuni meccanismi.

Seconda feature fondamentale dell'applicazione, che riguarda in questo caso solo gli utenti iscritti al sistema, è la possibilità di comprare online dei coupon digitali che permettono l'acquisto di un gelato presso una determinata gelateria. Questi coupon sono visualizzabili in ogni momento e possono essere regalati anche ad altri utenti tramite un semplice metodo di condivisione che permette al destinatario di usufruire dell'oggetto comprato. Ogni gelateria che supporta questo sistema sarà provvista della stessa applicazione con un account specifico per il gelatiere che permetterà di validare i coupon utilizzati e ricevere il proprio compenso tramite il sistema di pagamento online.

Questo sistema di e-commerce sposta la vendita di un prodotto molto semplice come i gelati, online. È infatti possibile comprarli tramite carta di credito e l'azienda Carpigiani, nel frattempo, si inserisce in un meccanismo legato a delle realtà locali sul quale altrimenti non riuscirebbe ad avere informazioni. Questo

sistema, per quanto diffuso in altri mercati come il reselling online di oggettistica è fortemente innovativo in questo campo e rende l'ecosistema MyGelato uno dei primi nel suo settore.

Figura 1.2: Logo MyGelato

Capitolo 2

Specifiche Progettuali

L'ecosistema MyGelato è un progetto ampio che si basa sulla cooperazione tra più elementi fondamentali. Alla base troviamo un backend basato sul framework Ruby on Rails che implementa le funzionalità di creazione, gestione e fruizione di contenuti multimediali affiancate da un sistema e-commerce per la compravendita di coupon digitali. Per poter usufruire di tali servizi è necessario l'utilizzo di un'applicazione mobile, chiamata *MyGelato*, disponibile per i sistemi operativi mobile Apple iOS e Google Android; quest'ultima argomento di questa tesi.

Essendo l'applicazione Android solo una compenente di un più ampio progetto è essenziale fin dalla prima fase di sviluppo valutare correttamente e nella loro completezza tutte le specifiche date dall'azienda che ha commissionato il lavoro e tutte quelle date dalla necessità di far cooperare l'applicazione con le altre parti del sistema. Si valutano inizialmente le specifiche strutturali e logiche che determinano le interazioni principali dell'utente con l'applicativo: navigazione, design e flussi logici. Questi macro argomenti sono presenti in ogni singolo componente del sistema e per questo devono essere valutati prima di iniziare qualsiasi tipo di sviluppo: serve considerare le interazioni tra i flussi logici e valutare come mantenere coerente il design all'interno di tutta l'interfaccia utente.

Fatte queste considerazioni si valuta come implementare una navigazione ch permetta all'utente la scelta delle principali funzioni disponibili all'interno dell'applicazione: *Acquisto/Utilizzo Coupon*, *Lista Gelaterie Preferite*, *Mappa per la Ricerca*, *Mastro Gelatiere* e *Cambio Tema*. Ogni sezione fa parte dei due principali flussi logici presenti all'interno dell'applicativo: quello di marketing digitale che permette l'esplorazione da parte dell'utente delle gelaterie in una determinata zona e delle relative promozioni; e quello di e-commerce che permette l'acquisto online, la gestione, il regalo e l'utilizzo dei coupon gelato resi disponibili dalle gelaterie del circuito MyGelato.

Si valutano infine le specifiche tecnologiche richieste sia dall'azienda che dalla necessaria cooperazione con gli altri elementi del progetto. Vi deve essere la possibilità di avere un'applicazione multilingua, con autenticazione tramite social network e compatibile con la maggior parte dei device Android attualmente sul mercato. Sono di forte impatto anche alcune scelte a livello di comunicazione con il backend che indirizza lo sviluppo in una comunicazione bidirezionale tra applicazione e server tramite API REST e notifiche push. Essenziale poi mantenere la coerenza tra le due applicazioni mobile sviluppate per le piattaforme iOS e Android, rispettando in ognuna i propri pattern tipici.

2.1 Design

Il design dell'applicazione è uno degli elementi di maggior risalto: sviluppato fin nel dettaglio permette di personalizzare anche i componenti più basilari della *User Interface* (*UI*) come icone, testi e bottoni. Sono presenti tre font custom che devono essere utilizzati in situazioni e contesti differenti in modo che possa capire fin dal primo colpo d'occhio quali siano i messaggi informativi e quali siano quelli di contorno.

Altro elemento fondamentale presente in quasi ogni schermata è il tema dell'applicazione. I temi sono principalmente delle colorazioni differenti che devono caratterizzare e modificare il *main color* della UI cambiando anche i testi e le icone in base al fatto di essere temi chiari o scuri fornendo rispettivamente testi e icone neri o bianchi. Ogni tema, che deve essere selezionabile a piacimento dall'utente direttamente da una funzione raggiungibile dalla navigazione principale, ha come nome un gusto differente di gelato e un proprio set di risorse.

Le risorse grafiche rese disponibili all'interno delle specifiche richieste servono a rendere coerente l'interfaccia utente in ogni suo elemento: fin dall'icona per chiudere una sezione a concludere con ogni elemento della navigazione principale. Il fatto di avere un'insieme di risorse ben organizzate è necessario, così da includere di volta in volta in maniera dinamica immagini in base alla lingua dell'applicazione e al tema chiaro/scuro.

In figura 2.1 è possibile vedere alcune delle immagini presentate per la navigazione principale e alcune icone in tutte le versioni disponibili rispettivamente per un tema chiaro ed uno scuro.

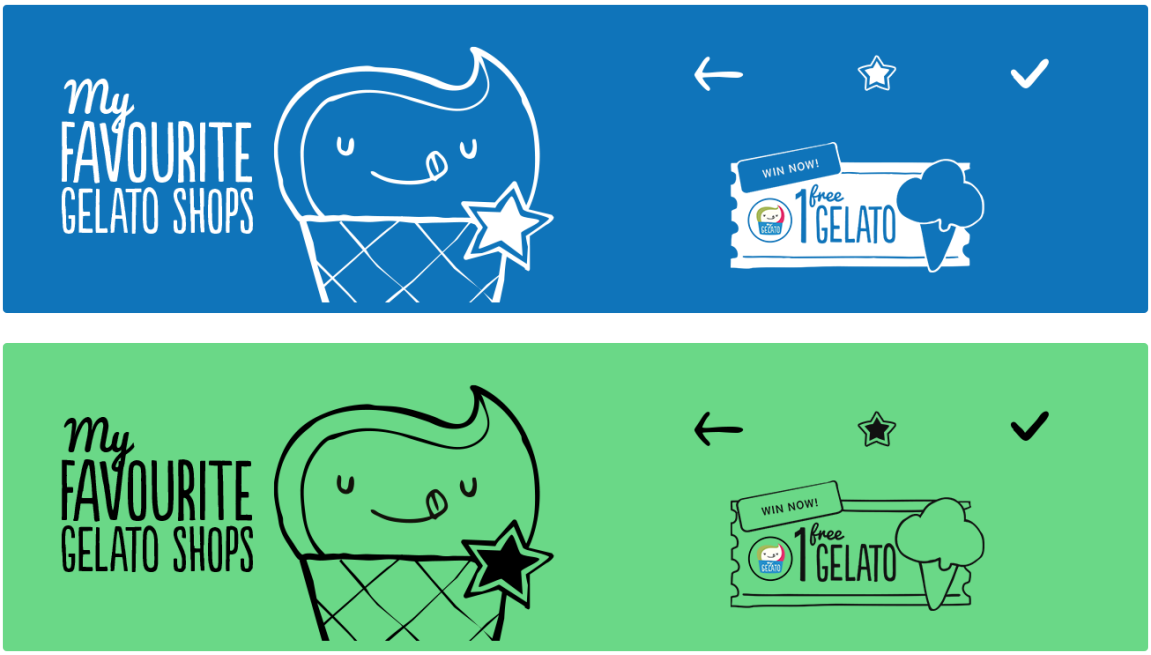

Figura 2.1: Risorse Design

2.2 Ricerca

L'elemento centrale di tutta l'applicazione è la possibilità di effettuare una ricerca delle gelaterie, iscritte al circuito MyGelato, presenti in una determinata zona geografica così da ottenere alcune informazioni essenziali sull'esercizio e le relative promozioni disponibili. Questa funzionalità è infatti l'elemento cardine che lega ogni possibile azione dell'utente all'interno dell'applicazione: è essenziale sia per il sistema di marketing che per il sistema di e-commerce.

Ogni utente, anche se non registrato, deve poter accedere a una mappa che visualizzi tutti gli shop disponibili, sia sotto forma di marker visibili in base al luogo di ricerca sia sotto forma di lista ordinata in base alla distanza dall'utente. Ogni elemento, scaricato e aggiornato in base alle ultime informazioni presenti

sul server, permette di ottenere il nome della gelateria, l'indirizzo e un eventuale recapito telefonico.

Questo tipo di ricerca permette all'utente di esplorare la zona attorno al luogo in cui trova, o anche ad un luogo di suo interesse, così da ottenere informazioni, sia di carattere commerciale sia pubblicitario, riguardo agli esercizi presenti in maniera molto diretta; permettendo anche di salvare lo shop all'interno della lista preferiti. Nel frattempo ogni gelateria presente all'interno del sistema ottiene la possibilità di essere trovata anche da utenti nuovi che non conoscono la zona e di pubblicizzarsi tramite una strategia di marketing unificata e standardizzata: le *carte promozionali*.

Oltre al sistema di marketing la ricerca delle gelaterie è essenziale per il sistema di e-commerce poichè la mappa viene utilizzata per la scelta da parte dell'utente della gelateria per la quale vuole acquistare un determinato coupon.

La ricerca è quindi presente sia nel sistema di navigazione principale dell'applicazione sia raggiungibile all'interno di ogni flusso logico scelto dall'utente e come si può vedere in figura il mockup della mappa permette di capire la necessità di avere un'implementazione semplice e facilmente utilizzabile da qualsiasi utente, dove sia possibile passare dalla visualizzazione a mappa a quella a lista e dove siano già visibili le principali informazioni riguardo a ogni gelateria.

È possibile osservare in figura 2.2 un mockup esplicativo per la schermata della ricerca tramite mappa con dettaglio sulla selezione di un marker informativo.

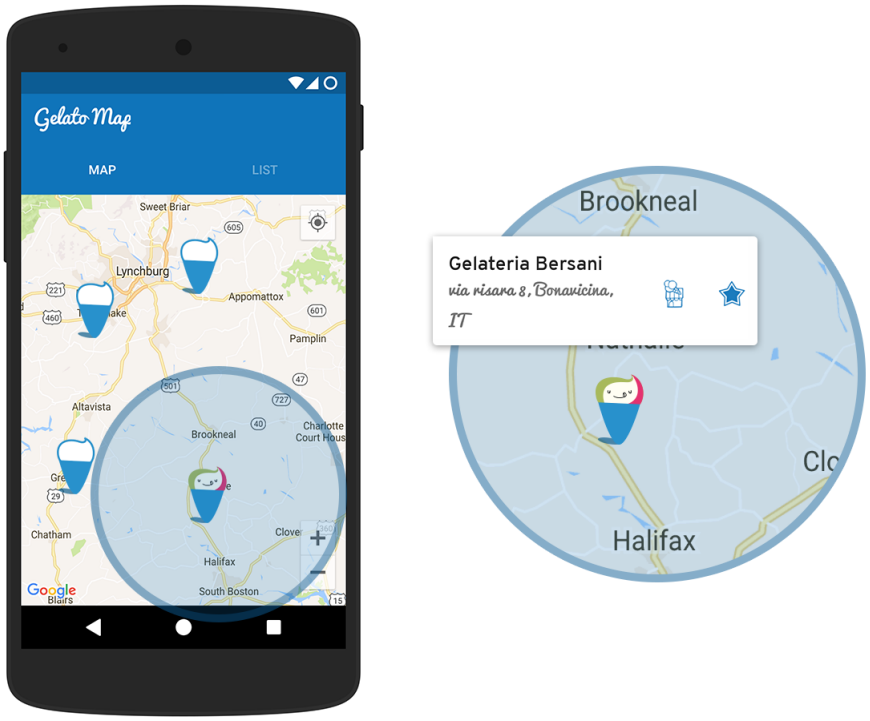

Figura 2.2: Mockup Mappa

2.3 Utente

La piattaforma MyGelato per poter offrire alcune funzionalità personalizzate al singolo utente implementa un sistema di registrazione e di autenticazione. Questa scelta progettuale è ovviamente fondamentale da considerare durante lo sviluppo dell'applicazione mobile poichè si deve considerare un'intera sezione che dia la possibilità all'utente di registrarsi, modificare parte delle proprie informazioni salvate sul server ed eseguire il login sulla piattaforma.

Sarà necessario poter accedere direttamente dalla navigazione principale alle

informazioni riguardanti il proprio stato di accesso alla piattaforma, verificando se si è già autenticati o meno. Le funzionalità che devono essere più facilmente accessibili sono quendi quelle di login e di logout, in modo che l'utente in pochi passaggi possa accedere o rimuovere i propri dati dal device utilizzato.

I nuovi utenti potranno utilizzare un form per iscriversi alla piattaforma MyGelato inserendo solo alcune informazioni essenziali come nome, cognome, email e password. Le funzionalità di registrazione si basano principalmente sull'inserimento manuale di un indirizzo di posta elettronica come metodo identificativo dell'utente, ma deve essere possibile anche eseguire la registrazione al servizio tramite social network. Il sistema prenderà in automatico i dati di cui necessita per inserire l'utente all'interno del proprio database e ogni volta l'accesso all'applicazione sarà legato al login sui social network.

All'interno della schermata di login, nel caso vi sia stata una registrazione manuale, deve essere possibile accedere ad una sezione che permetta di ricevere nuovamente la password legata all'account sulla email inserita durante la registrazione. Infine i dati inseriti che non vengono utilizzati per identificare l'utente potranno essere modificati una volta effettuato l'accesso direttamente dalla sezione utente.

2.4 Marketing Digitale

Uno dei due *workflow* principali presenti all'interno dell'applicazione riguarda il marketing digitale che viene svolto per le gelaterie legate all'ecosistema MyGelato. Ha lo scopo di rendere più facile la ricerca delle gelaterie, sponsorizzarne eventuali promozioni e permettere a ogni utente di salvare gli esercizi preferiti così da rimanere aggiornato sulle nuove promozioni disponibili.

Queste funzionalità, disponibili in buona parte per qualsiasi utente anche non registrato, seguono un flusso logico che parte dalla ricerca degli shop e trova effetto principale nella scoperta e nell'utilizzo delle carte promozionali, della singola gelateria e del Mastro Gelatiere.

2.4.1 Carte Promozionali

Le carte promozionali sono il principale mezzo di advertising all'interno dell'applicazione: sono dei volantini digitali formati da un'immagine o un video, un titolo, un testo descrittivo, un eventuale recapito telefonico e un link per ottenere maggiori informazioni riguardo alla promozione in oggetto. Ci sono due tipologie di carte: le carte specifiche di ogni esercizio, ovvero i volantini informativi della singola gelateria, e le carte del mastro gelatiere che invece si rifanno alle promozioni generiche proposte per l'intero sistema.

Le carte della singola gelateria altro non sono che le ultime promozioni legate all'attività: sconti, novità, messaggi informativi. L'utente in questo modo può facilmente reperire le informazioni pubblicitarie di una determinata gelateria online e in mobilità; considerando che difficilmente in questo campo vi è già una diffusa pubblicizzazione sui social network o sui sistemi di advertising. Ogni shop iscritto al circuito MyGelato ha quindi la possibilità di ottenere una maggior copertura pubblicitaria tramite l'utilizzo di un sistema centralizzato e standardizzato.

Le carte di uno stesso esercizio devono essere raggruppate in modo che l'utente possa scorrerle e visualizzare tutte le promozioni o informazioni disponibili insieme. La schermata per questa visualizzazione deve essere raggiungibile ogni volta che sono mostrate le informazioni della gelateria: sia internamente alla ricerca che nella lista dei preferiti dell'utente.

Oltre alle singole gelaterie l'ecosistema MyGelato prevede anche le carte del Mastro Gelaterie che comprendono promozioni, informazioni e novità pubblicate genericamente dagli amministratori del sistema e che sono valide per tutti gli esercizi facenti parte del circuito. Essendo un insieme di carte totalmente generiche, quindi slegate da qualsiasi flusso logico, e di fondamentale importanza per il sistema di advertising creato, questa sezione deve essere inserita all'interno della navigazione principale del sistema.

Infine, come avverrà poi per le gelaterie preferite, ogni utente registrato dovrà avere la possibilità di essere notificato di eventuali nuove promozioni nel momento stesso in cui diventerano disponibili.

2.4.2 Preferiti

La gestione delle proprie gelaterie preferite è uno degli elementi presenti all'interno della navigazione principale dell'applicazione, accessibile da qualsiasi utente anche non registrato: permette di salvare sul proprio dispositivo gli esercizi di maggiore interesse.

Questo dà modo all'utente di poter accedere in ogni momento, specialmente in mobilità, alle informazioni che più gli interessano di ogni gelateria: indirizzo, recapito telefonico ed eventuali promozioni. La ricerca in questo caso è molto semplicificata poichè viene presentata una sezione a parte con una lista degli shops preferiti, senza dover obbligare l'utente a effettuare una nuova ricerca all'interno della mappa. Il salvataggio all'interno dei preferiti avviene direttamente all'interno della ricerca, sia tramite la mappa che tramite la lista grazie a un'icona esemplificativa.

Questa funzionalità è pensata principalmente per fidelizzare il consumatore: ogni utente avendo la possibilità di salvare una gelateria ha anche la possibilità

di ottenere velocemente informazioni su nuove promozioni, trovare velocemente i contatti dell'esercizio come se li avesse salvati in rubrica e valuatare ogni volta la distanza tra se e lo shop.

Il passo successivo in questo senso è dato dal rendere bidirezionale questo collegamento, rendendo in alcuni casi non necessaria la ricerca da parte dell'utente di nuove informazioni fornendogliele invece a ogni aggiornamento. Per fare questo, solo nel caso di utenti registrati, l'aggiunta di uno shop ai preferiti include le funzionalità di geofencing e notifiche push, quest'ultima presente anche per le carte promozionali del mastro gelatiere.

Il geofencing permette di ricevere una notifica ogni qualvolta l'utente si trovi a meno di 5 km da una delle proprie gelaterie, così da essere informato di essere vicino in termini di localizzazione. Le notifiche push invece sono attivate per avvertire un utente che una delle proprie gelaterie ha pubblicato una nuova promozione tramite l'aggiunta una carta sul sistema MyGelato. L'utente così rimane informato costantemente delle ultime promozioni disponibili e il propietario di una gelateria ha la certezza di effettuare pubblicità diretta tra se e i suoi clienti più affezzionati.

2.5 E-Commerce

Il secondo flusso logico presente all'interno dell'applicazione riguarda l'acquisto e l'utilizzo di coupon gelato online e in completa mobilità. Questo sistema già più che diffuso in tantissimi altri ambiti della vendita online è uno dei cardini su cui maggiormente si vuole puntare sia per funzionalità sia per innovazione.

I *Coupon* gelato sono buoni acquisto che si possono acquistare dirattemente tramite l'applicazione e permettono a chiunque ne sia virtualmente in possesso di utilizzarli nelle gelaterie che li hanno rilasciati tramite validazione. L'acquisto dei

buoni deve poter essere fatto in qualsiasi momento utilizzando le ultime tecnologie disponibili per i pagamenti online; essenziale mantenere un occhio di riguardo alla sicurezza di questo tipo di transizioni.

Tra le specifiche risulta essere presente anche il concetto di condivisione dei coupon tramite un sistema di *share/redeem*: un utente che ha acquistato un coupon ha la possibilità di condividerlo con un altro utente registrato alla piattaforma che può riscuotere poi il buono. La condivisione deve avvenire tramite applicazioni e canali di comunicazioni facilmente utilizzabili all'interno di uno smartphone.

Infine vi deve essere ovviamente la possibilità di validare un coupon una volta che si decide di utilizzarlo all'interno della gelateria che lo ha rilasciato online tramite il circuito MyGelato. Questo procedimento prevede due entità in gioco, colui che possiede il coupon e il gelataio che deve poterlo validare; in entrambi i casi le funzionalità da implementare dovranno essere presenti all'interno dell'applicazione in base al tipo di account utilizzato così da mantenere coerenza negli strumenti utilizzati per collegarsi alla piattaforma.

2.5.1 Coupon

I Coupon sono buoni per l'acquisto di un bene materiale, normalmente gelati, che si può ritirare in qualsiasi momento in sede all'esercizio che ha effettuato la vendita. Hanno un nome e un valore, il prezzo, scelti durante l'inserimento tramite il sistema amministrativo gestionale che può inserire anche informazioni aggiuntive come la scadenza e la valuta. Sono generici per tutto l'ecosistema e sono quindi disponibili per ogni gelateria che li rende disponibili anche se all'acquisto sarà necessario specificare l'esercizio nel quale si desidereranno poi utilizzare.

All'interno della navigazione principale dell'applicazione sarà necessario avere una sezione apposita per poter gestire tutti i propri coupon: una lista di quelli

utilizzati, quelli regalati ad altri utenti e quelli che si possono ancora utilizzare personalmente. Questa sezione sarà disponibile offline ma dovrà ogni volta essere sincronizzata con il server in modo da avere coerenza anche utilizzando la stessa applicazione con lo stesso account da dispositivi differenti.

Oltre alla possibilità di acquistare coupon si dovrà poter accedere direttamente alle funzioni di condivisione, riscatto e utilizzo così da avere una gestione centralizzata di tutto il flusso logico legato all'e-commerce.

2.5.2 Acquisto

L'acquisto di un coupon è permesso a qualsiasi utente che si sia registrato, tramite mail o social network, al circuito MyGelato. Questa funzionalità è raggiungibile direttamente tramite parte della navigazione principale dell'applicazione, anche se deve essere disponibile solo nel caso in cui l'utente abbia eseguito il login, altrimenti dovranno essere richieste le credenziali di accesso.

Le operazioni richieste per l'acquisto di un coupon dovranno essere tutte racchiuse all'interno di una stessa schermata in cui l'utente verrà guidato attraverso le varie fasi fino all'avvenuta transizione. Il primo passo da svolgere è la scelta della gelateria su cui eseguire l'acquisto, come spiegato precedentemente si andrà a sfruttare la mappa e la lista per la ricerca degli shop già utilizzata all'interno di altre funzionalità. In questo caso però verranno visualizzati solo le gelaterie che permettono l'acquisto di coupon e quando verranno selezionati i marker non si otterenno tutte le informazioni sull'esercizio ma sarà presente un'icona per selezionare lo shop.

Una volta selezionata la gelateria dovranno essere scaricati dal server i coupon disponibili e nel frattempo verrà resa disponibile la scelta del metodo di pagamento che si appoggerà al sistema di pagamento online Stripe. Si potrà scegliere

tramite una lista di metodi di pagamento quello che si preferirà usare che sarà successivamente impostato come predefinito.

Scelto il metodo di pagamento, scorrendo la lista dei coupon disponibili ed decidendo il valore che si vuole acquistare si potrà infine scegliere di completare l'acquisto che in caso di successo dovrà poi riportare l'utente sulla sezione di gestione dei coupon e altrimenti presentare un messaggio di errore.

2.5.3 Condivisione e Riscatto

Le funzionalità di condivisone e riscatto che si vogliono implementare all'interno dell'applicazione rendono importante capire come far dialogare due dispositivi differenti in modo che abbiano informazioni coerenti uno con l'altro e in modo che sia semplice l'utilizzo da parte di utenti non esperti. Per rendere questa funzionalità accessibile da chiunque si dovrà legare ogni coupon grazie ad un codice alfanumerico di sei cifre detto PNR (*Product Nr.*) che potrà quindi essere condiviso tramite qualsiasi mezzo disponibile.

L'uso di un codice inviabile facilmente come teso permette di sfrutare altri applicativi presenti sul device dell'utente per la condivisione tra persone differenti. Scegliendo il coupon da regalare si dovrà quindi chiedere all'utente tramite quale canale di comunicazione preferisce inviare il codice PNR per poi inserire il codice in un testo promozionale legato ad un link di reindirizzamento a una pagina web della piattaforma MyGelato.

L'inserimento del link a una pagina web della piattaforma accessibile online da qualsiasi dispositivo è una funzionalità di grande potenza: nel caso in cui la pagina web sia raggiunta da un device che ha installata sopra l'applicazione MyGelato aprirà direttamente l'applicativo stesso gestendo internamente il codice presente;

altrimenti verrà presentata una pagina web che suggerisce all'utente di scaricare l'applicazione dai principali store mobile.

Valutato come scegliere di condividere un coupon è necessario inserire una funzione all'interno dell'applicazione che ne permetta il riscatto tramite il solo inserimento del codice PNR. Raggiungibile dalla navigazione principale vi sarà una sezione che permetta di eseguire questa operazione in maniera molto semplice e che, nel caso venga avviata l'applicazione tramite la gestione del link web di cui si è parlato precedentemente sarà compilata in automatico.

Tramite un semplice controllo sul server verrà quindi controllato se l'utente può riscattare il codice e le informazioni saranno aggiornate per entrambi gli utenti: chi ha regalato il coupon e chi invece può utilizzarlo.

2.5.4 Utilizzo e Validazione

L'ultimo passo all'interno del flusso logico di questo sistema di e-commerce è ovviamente l'utilizzo e la validazione di un coupon per acquistare un bene materiale direttamente in sede all'esercizio che ha venduto il buono. Per semplificare l'utilizzo della piattaforma MyGelato ci si è rifatti nuovamente al fatto che gli smartphone siano diventati dei beni di uso comune e che quindi anche i propetari delle gelaterie possano utilizzare l'applicazione MyGelato come mezzo per la validazione dei coupon.

Allo stesso modo che per le funzionalità di condivisone e riscatto si è scelto di utilizzare i codici PNR dei coupon come mezzo per far dialogare i due dispositivi di chi vuole utilizzare un buono e di chi deve verificare che sia valido. Rispetto ad un sistema di condivisione che prevede la presenza di un canale di comunicazione personale tra i due utenti si dovrà valutare l'utilizzo di mezzi più semplici e slegati.

Si è quindi proposto di generare un QR Code a partire dal codice PNR di ogni coupon, raggiungibile direttamente cliccando sul buono presente nella lista di gestione interna all'applicazione. Il consumatore dovrà quindi mostrare il QR Code al propietario dell'esercizio che, utilizzando la stessa applicazione MyGelato con un account di tipo *gelatiere*, utilizzerà una funzione di validazione tramite lettura del codice mostrato. Sarà infine necessaria una connessione al server da parte di entrambi i dispositivi per verificare l'avvenuta validazione del coupon così da procedere all'acquisto materiale del bene rappresentato dal buono.

Capitolo 3

Progettazione

Definiti gli obiettivi progettuali dell'applicazione si sono valutati gli strumenti di sviluppo da utilizzare durante lo svolgimento della tesi: come parametri si é tenuto conto di tempistiche di aggiornamento, adeguamento alle linee guida del sistema operativo in oggetto, documentazione disponibile e modernità delle tecnlogie utilizzate.

Durante la prima fase di sviluppo strutturale si è dovuto valutare la tecnologia da utilizzare in modo da tenere il passo con le ultime specifiche di Android e tenendo conto che, anche secondo gli ultimi report pubblicati da Google, rimane una forte frammentazione della distribuzione del sistema operativo, dovuta ai molteplici produttori di hardware.[1]

Era quindi da tenere in considerazione la retrocompatibilità delle librerie utilizzate, avendo scelto di supportare fino alla versione 16 del SDK (Android 4.1 Jelly Bean), e la possibilità di utilizzare l'applicazione anche su device con schermi e hardware differenti.

Si sono inoltre adottate alcune strategie implementative, come la creazione di

[1]Google Inc. *Android Developers.* URL: `https://developer.android.com/about/dashboards/index.html`.

custom view, per poter meglio strutturare il progetto favorendone anche eventuali modifiche o ampliamenti.

3.1 Sviluppo Android

Considerato che l'applicazione mobile, oggetto di tesi, era destinata a lavorare unicamente sulla piattaforma Android, si è scelto di utilizzare come principale strumento di sviluppo **Android Studio**, software dedicato alla programmazione nativa del suddetto sistema operativo.[2]

Durante lo sviluppo del progetto si è utilizzato come sistema di versioning il software **Git**, così da poter mantenere uno storico del lavoro svolto e ottenendo tutti i vantaggi del'utilizzo di un Sistema per il Controllo di Versione (*Version Control System - VCS*).

Sono inoltre da citare alcuni software utilizzati durante il lavoro di tesi che, anche se di minore impatto, sono serviti a testare e sviluppare alcune parti fondamentali di codice. Primo fra tanti la web application Postman[3] che permette di testare e sviluppare le APIs con cui due applicativi possono dialogare, in maniera semplice ed intuitiva. Utile per valutare le chiamate al server e per verificare le risposte ottenute. **Altri da citare???**

Questa tesi si inserisce all'interno del lavoro di modifica e miglioramento delle applicazioni mobile dell'ecosistema MyGelato, il quale includeva un progetto sia per piattaforma Android sia per piattaforma iOS. Essendosi creati due differenti progetti, durante la prima fase di progettazione, si è potuto valutare in maniera più libera l'utilizzo o meno della programmazione nativa.

[2]Google Inc. *Android Developers.* URL: `https://developer.android.com/training/basics/firstapp/index.html`.

[3]Inc. Postdot Technologies. *Postman.* URL: `https://www.getpostman.com/`.

Sicuramente lavorare in questo modo o meno ha i propri pro e contro. Mentre da un lato il nativo offre la possibilità di una gestione totale del dispositivo senza la paura di trovare limiti, d'altra parte richiede spesso una programmazione molto professionale e si concentra esclusivamente su una piattaforma impedendo un'agile riciclo dei propri sforzi su altri mercati del mobile. Il non-nativo, invece, offre diversi vantaggi ascrivibili ad una minore necessità di programmare e molto spesso alla possibilità di creare applicazioni cross-platform distribuibili su sistemi operativi diversi.[4]

Non avendo quindi necessità di mantenere alta la portabilità del codice su altre piattaforme, si è preferito sviluppare tramite programmazione natuva; potendo quindi sfruttare pienamente l'architettura del sistema operativo sottostante.

3.1.1 Java

3.1.2 XML

3.1.3 Android Studio

Android Studio[5] è un ambiente di sviluppo integrato (IDE) per lo sviluppo per la piattaforma Android. È stato annunciato il 16 maggio 2013 in occasione della conferenza Google I/O e la prima build stabile fu rilasciata nel dicembre del 2014. Basato sul software della JetBrains IntelliJ IDEA, Android Studio è stato progettato specificamente per lo sviluppo di Android.[4] È disponibile il download su Windows, Mac OS X e Linux,[5][6] e sostituisce gli Android Development Tools (ADT) di Eclipse, diventando l' IDE primario di Google per lo sviluppo nativo di applicazioni Android. Permette di creare un progetto gradle, apk, github,

[4]**HTMLIT:PROGNATIVA**.

[5]Wikipedia. *Android Studio*. URL: `https://it.wikipedia.org/wiki/Android_Studio`.

3.1.4 Git

Git è un software di controllo versione distribuito (*Distributed Version Control Systems - DVCS*) utilizzabile da interfaccia a riga di comando, creato da Linus Torvalds nel 2005.[6] Nacque per essere un semplice strumento per facilitare lo sviluppo del kernel Linux ed è diventato uno degli strumenti di controllo versione più diffusi.

Per quanto riguarda qualsiasi tipo di progetto di IT (*Information Technology*), specialmente se si tratta di un lavoro da dover svolgere in team, Git da la possibilità di mantenere in memoria tutte le versioni del proprio lavoro (sia in locale che online). Questo permette a più persone di poter accedere alla cronologia del lavoro condiviso, avendo anche la possibilità di verificare la presenza di errori e di eliminarli tornando ad una versione precedente del progetto. Git sfrutta alcuni algoritmi avanzati per calcolare le differenze riga per riga tra i file di diverse versioni così da verificare la presenza di errori o conflitti.

Git è fortemente direzionato verso uno sviluppo progettuale non lineare. Supporta diramazione e fusione (*branching and merging*) rapide e comode, e comprende strumenti specifici per visualizzare e navigare l'intero storico delle versioni anche se proposte da sistemi differenti. È veloce e scalabile nella gestione di grandi progetti ed è molto utile nello sviluppo in team, specialmente se affiancato dal modello GitFlow che permette di gestire rami specifici per il developing o per le release.

Infine grazie all'utilizzo di piccoli accorgimenti come il file .gitignore (che permette di non salvare anche i file temporanei e non importanti del progetto) e soluzioni online come GitHub o BitBucket (quest'ultimo utilizzato durante lo sviluppo di questa tesi) il processo di sviluppo software viene drasticamente avvantaggiato.

[6] Wikipedia. *Git (software)*. URL: `https://it.wikipedia.org/wiki/Git_(software)`.

3.2 Database

3.2.1 Realm

3.3 Chiamate Server

Attualmente ogni tipo di piattaforma che voglia permettere al proprio utente di accedere da remoto, e quindi da qualsiasi dispositivo, alle proprie informazioni ha come parte fondamentale lo sviluppo e il mantenimento di un server di beckend (quindi nascosto nelle funzionalità alla vista dell'utente) che mantenga, gestisca e restituisca i dati necessari alle funzioni degli applicativi. Nello stesso modo la piattaforma MyGelato si basa su un server che mantiene ogni informazione riguardante utenti, shop e cards fornendo delle API REST che vengono sfruttate dalle applicazioni mobile e web per poter fornire i propri servizi.

Il passaggio di informazioni tra i due applicativi avviene tramite chiamate internet con protocollo http protette grazie ad una connessione criptata dal Transport Layer Security (TLS) o dal suo predecessore, Secure Sockets Layer (SSL). Grazie alla API REST è possibile sviluppare in maniera strutturata e di seguire alcuni pattern delle tecnologie che permettono di valutare ogni possibile errore delle connessioni: errori nei dati, nell connessione, ecc... In Java le chiamate http sono abbastanza difficili da gestire poichè non sono automatizzate e son stati effettuati dei cambiamenti tra le ultime versioni che non permettono di lavorare correttamente su tutti i dispositivi Android. Per questo motivo è stata utilizzata la libreria open-source OkHttp sviluppata da Square, che permette di automatizzare le chiamate http sfruttando anche una notevole semplificazione del codice utilizzato per programmare.[7]

[7]Square. *OkHttp*. URL: `http://square.github.io/okhttp/`.

3.3.1 OkHttp

La liberia OkHttp nasce dal fatto che il protocollo http è l'attuale mezzo di comunicazione per ogni applicativo moderno. Per questo si pone l'obiettivo di semplificarne l'utilizzo in più di una tecnologia migliorandone le prestazioni e fornendo allo sviluppatore alcuni automatismi comodi durante lo sviluppo e la programmazione.

Supporta ogni tipo di comunicazione http, ne gestisce la latenza, il caching, la riconnessione, l'utilizzo di protocolli sicuri come TLS, l'utilizzo di indirizzi IP multipli ed è molto semplice da integrare e utilizzare all'interno del proprio software. Grazie ad un'interfaccia molto semplice permette di automatizzare le chiamate http utilizzando dei Builder che lasciano specificare solo i parametri necessari e restituiscono un unico oggetto da cui si ottiene la risposta, positiva o negativa, della richiesta.

```
OkHttpClient client = new OkHttpClient();
Request request = new Request.Builder()
            .url(endpoint)
            .addHeader("Accept","application/json")
            .build();

Response response = client.newCall(request).execute();
```

Inizializzato il client OkHttp basta creare un oggetto request a cui vanno passati i parametri necessari; basta poi eseguire la chiamata per ottenere un oggetto response all'interno del quale sono presenti i dati scaricati, il codice di risposta della chiamata e anche eventuali errori.

3.3.2 API REST

3.4 Gestore di Eventi

Un passaggio fondamentale durante lo sviluppo di un software è l'aggiornamento delle view a seguito di un cambiamento nello stato dell'applicazione. In Android è necessario intervenire direttamente sugli elementi dell'interfaccia modificando ogni oggetto in base alle specifiche richieste senza poter sfruttare veri e propri automatismi. Una forte limitazione del sistema operativo è dovuta, per esempio, al fatto che l'unico thread che ha il permesso di accedere e di modificare l'interfaccia utente è il Main Thread; questo complica l'interazione con eventuali sistemi multithreading anche se, ovviamente, fa parte delle specifiche di sistema utili a limitare i conflitti sull'accesso alle risorse condivise (in questo caso le view).

Questo problema necessita di ampia valutazione durante lo sviluppo e anche in questo caso si è dovuto strutturare l'applicazione in modo che non incorresse in errori di inconsistenza tra le view a cui il programma voleva accedere e le view realmente visualizzate al momento. Nel caso si utilizzino script asincroni è molto facile tentare di accedere ad elementi, o anche intere activity, non più presenti nell'interfaccia utente; per questo motivo si è scelto di utilizzare la libreria EventBus sviluppata da GreenRobot[8] e pensata esattamente per risolvere questo tipo di errori.

3.4.1 EventBus

EventBus è una libreria open-source per Android che utilizza il pattern publish/subscribe per far dialogare tutti i componenti di un'applicazione. Disaccoppia le classi che interagiscono sulla stessa interfaccia e le gestisce in maniera centra-

[8]GitHub. *EventBus*. URL: `https://github.com/greenrobot/EventBus`.

lizzata monitorando le performance e gestendo gli errori che potrebbero essere sollevati da uno sviluppo multithreading.[9]

I principali benefici che si riscontrano tramite l'utilizzo di questa libreria sono la semplificazione delle comunicazioni tra componenti, il disaccoppiamento tra mittenti e riceventi degli eventi, miglioramenti nelle perfomance, facile integrazione dei metodi da utilizzare ed infine anche caratteristiche avanzate come priorità e indirizzamento a thread specifici.

Durante lo svolgimento di questa tesi si è definita una struttura abbastanza comune formata da: - AsyncTask, oggetto che permette di elaborare una serie di informazioni in background, utilizzato per eseguire il download delle informazioni dal server; - View specifica dell'activity con determinati componenti da aggiornare in base allo stato dell'applicazione; - Metodo updateUI() che preso in gresso i dati scaricati doveva aggiornare le view rispetto allo stato.

Per far dialogare questi elementi si è utilizzato EventBus così da sfruttare il lavoro di thread in background senza limitare l'utente nell'attesa di un determinato evento. Si procedeva quindi crendo un evento specifico per l'interazione da dover gestire di cui qui viene riportato un esempio:

```
public static class UpdateCouponUiEvent<T> {
  public T data;
  public UpdateCouponUiEvent() {
  }
  public UpdateCouponUiEvent(T data) {
    this.data = data;
  }
}
```

EventBus permette di registrare dei metodi eseguendo una subscribe ad un determinato evento: ogni qual volta un evento di quel tipo viene sollevato, il metodo

[9]GreenRobot. *EventBus*. URL: `http://greenrobot.org/eventbus`.

viene eseguito. La subscribe deve essere definita nel codice nel seguente modo:

```
@Subscribe
public void updateUI(UpdateCouponUiEvent event) {
  // update UI with event.data
}
```

Grazie all'utilizzo di eventi custom è dunque possibile passare anche dei dati tramite l'evento che viene lanciato, nel codice messo ad esempio si tratta della variable data, definito di tipo generico.

Infine all'interno del AsyncTask, a conclusione del download delle informazioni e dell'aggiornamento dello stato, era sollevato un evento grazie ad una publish:

```
EventBus.getDefault().post(new UpdateCouponUiEvent());
```

3.5 E-Commerce

3.5.1 Stripe

Capitolo 4

Implementazione

A seguito della fase di progettazione si è scelto di suddivedere lo sviluppo dell'applicazione in alcune fasi principali che seguono logicamente i workflow presentati nel dettaglio all'interno del capitolo 2. Questo ha permesso di dedicare maggiore attenzione ad ogni componente fino ad un livello di dettaglio molto alto, in modo da poter anche ottenere una buona valutazione sulle prestazioni dei nodi più critici dell'applicativo.

Per poter ottenere uno sviluppo abbastanza lineare è stato necessario considerare che alcune specifiche richieste erano presenti in tutto il sistema e che quindi non potevano essere sviluppate a se stante dagli altri componenti. Prima di tutto il design richiesto doveva essere presente in ogni singola view e per questo si è deciso di implementare fin da subito le interfaccie utenti per non dover replicare ad ogni passaggio le stesse operazioni di personalizzazione della UI.

Strutturato il metodo di sviluppo per la grafica dell'applicazione si è passati a gestire la navigazione principale scegliendo di utilizzare alcuni dei principali pattern Android: il *NavigationDrawer* e le *RecyclerView*. Il menù iniziale è servito a separare anche concettualmente le principali funzioni da dover sviluppare così da poter procedere successivamente con l'impletazione di ogni componente potendovi

accedere anche se altre parti dell'applicazione non erano ancora disponibili.

Ragionando ulteriormente dal generale al dettaglio si è scelto di sviluppare la mappa di ricerca degli shop poichè presente in entrambi i flussi logici principali e quindi nodo cardine dell'applicazione, specialmente in termini di prestazioni. La fase successiva ha fornito l'intera gestione della registrazione e dell'autenticazione di un utente poichè facente parte sia del sistema di marketing sia di quello di e-commerce, ultimi componenti implementati durante la fase di sviluppo.

A conclusione dell'implementazione si sono effettuati dei test per valutare le prestazioni dell'applicazione, in particolar modo sui nodi centrali che avrebbero potuto inficiare l'esperienza utente se con basse prestazioni, come ad esempio la mappa di ricerca.

4.1 Design

Il Design è stato il primo concetto sviluppato poichè avrebbe rappresentato una costante durante l'implementazione di qualsiasi schermata dell'applicazione. In figura 4.1 sono visualizzati rispettivamente la scelta di due temi differenti, uno chiaro ed uno scuro e a fianco è mostarto come cambia l'interfaccia di navigazione in base al tema selezionato. Tutti i componenti sono personalizzati implementando i tre font differenti, utilizzando le risorse messe a disposizione sia per le icone che per le azioni di navigazione.

Per incapsulare tutte le funzioni rigurdanti i temi all'interno di un unico oggetto si è creata la classe *Flavors* che rappresenta il tema attuale, recuperandolo dalle preferenze dell'applicazione, e contiene tutti i dati utili a personalizzare i componenti dell'interfaccia come ad esempio il *main* e il *secondary color*. Questo oggetto viene automaticamente generato unicamente a partire da un contesto di una scherata senza dover eseguire nessa operazione di inizializzazione.

```
public class Flavors {
    Context mContext;
    ...
    public int getPrimaryColor();
    public int getStatusBarColor();
    public int getPrimaryTextColor();
    ...
}
```

Ad ogni inizializzazione e a ogni cambio di tema vi sono elementi dell'UI che sono da aggiornare, in special modo tutte le ImageView e le TextView che rappresentano rispettivamente ogni immagine e ogni testo presenti sulla schermata. Per automatizzare la gestione di questi componenti si è scelto di implementare un'interfaccia comune ai due oggetti chiamata *FlavorObject* che dichiarava un nuovo metodo chiamato *updateFlavor()* da richiamare ad ogni aggiornamento e che modifica gli elementi necessari.

```
public interface FlavorObject {
    public void updateFlavor(Flavors flavor);
    public void updateFlavor(Context context, String taste);
}
```

Da quest'interfaccia si sono ereditate le tre view principali che sono state utilizzate durante tutto lo sviluppo di questa tesi e che hanno permesso di richiamare in metodo ricorsivo su tutti i componenti lo stesso metodo, in modo da aggiornare la schermata senza dover conoscere esattamente come ogni singolo elemento andava modificato. Ogni custom view ha quindi implementato il metodo dell'interfaccia così da gestire in maniera automatica i temi chiari/scuri e il cambio ad ogni aggiornamento.

Per la gestione dei font dei testi si è sfruttato il fatto di aver implementato una versione personalizzata della textView così da implementare un nuovo elemento

all'interno del componente scritto in xml. In questo modo si è potuto rendere il codice molto più pulito indicando il font per ogni testo direttamente nella definizione della view senza dover gestire ogni elemento tramite java alla creazione della schermata.

```
<com.carpigiani.mygelato.custom.CustomTextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   ...
   app:typefaceAsset="fonts/Pacifico.ttf"/>
```

Per eseguire l'aggiornamento automatico di tutte le view si è scelto di creare una custom Activity, che rappresenta il componente di una schermata dell'interfaccia utente, così da implementare anche in questo caso un metodo che venisse richiamato ogni volta che vi erano elementi da modificare.

Questa *FlavorActivity* incapsula la gestione stessa del tema attuale, grazie ad una variabile Flavors, aggiornando in maniera automatica oltre agli elementi dichiarati anche la toolbar e la statusbar visibili come componenti di sistema di Android.

```
public class FlavorActivity extends AppCompatActivity {
    private Flavors flavors;
    ...
    @Override
  public void onResume() {
        super.onResume();
        applyFlavor();
    }

    public void applyFlavor() {
        this.flavors = new Flavors(this);
        updateBar();
    }
```

```
    ...
}
```

Concluso lo sviluppo di ogni componente si è inserita all'interno della navigazione principale la possibilità di accedere alla sezione per la scelta del tema da parte dell'utente che elenca ogni tema identificandolo con un nome di gelato e alcune immagini personalizzate come si può notare in figura 4.1.

Figura 4.1: Temi

4.2 Network

Tutte le funzionalità di collegamento al backend tramite API Rest sono state raccolte e incapsulate all'interno di un unico componente utilizzato ad ogni chiamata server in maniera standardizzata. Sfruttando la libreria OkHttp si è potuto semplificare la gestione delle chiamate http potendo incapsulare il risultato in un classe *Result* contenente i dati sia in caso di errore sia in caso di successo.

La possibilità di utilizzare il *Diamond Operator* ha permesso di utilizzare lo stesso oggetto anche con risultato di tipo differente, come per esempio i dati relativi ad un utente o ad una carta promozionale.

```
public class Result<T, E> {
    public T result = null;
    public E error = null;
    public Result(T result, E error) {
        this.result = result;
        this.error = error;
    }
}
```

Le API Rest rese disponibili dal backend hanno permesso di standardizzare le chiamate richiedendo ad ogni chiamata l'inserimento solo del metodo, dei parametri da inviare e dell'eventuale payload. Anche la gestione degli errori grazie ai codici identificativi Http hanno reso la gestione di tutti gli errori semplificata così da poter visualizzare un messaggio di errore il più specifico possibile e localizzato in base alla lingua utilizzata sul device.

L'utilizzo di un middleware così sviluppato per ogni chiamata al backend ha permesso anche di inserire alcuni controlli su ogni tipo di richiesta, per esempio la possibilità di verificare che le chiamate autenticate non dessero come risultato il codice 401 in caso di utente non autorizzato; situazione che doveva invocare il logout dell'attuale utente dall'applicazione.

4.3 Utente

Il primo passaggio effettuato durante lo sviluppo della sezione legata all'utente è stato quello di realizzare un modello, definito all'interno della classe *User*, che racchiude tutte le informazioni legate ad un account. Utilizzando Realm è stato possibile definire un oggetto che lo rappresenta, mantiene in locale sul dispositivo tutti i dati inseriti e mette a disposizione un insieme di funzioni utili per la gestione del database.

La sezione per la gestione del proprio account e delle funzioni di autenticazione sono inserite in una parte del menù laterale della pagina principale dell'applicazione, permette di verificare se si è effettuato o meno l'accesso rapidamente potendo eseguire anche le azioni di login e logout. Si è inserito un header all'interno del menù laterale in cui sono visibili un'immagine circolare, un messaggio di benvenuto e un bottone per eseguire l'accesso. Una volta che l'utente sarà autenticato saranno invece visibili l'avatar, il nome, un'icona per accedere alla schermata di modifica dell'account e il bottone per eseguire il logout.

Si è poi implementata un'unica schermata dalla quale si può accedere al login tramite credenziali, al login tramite social network, alla registrazione di un nuovo account e si verrà reindirizzati a questa sezione ogni volta che l'utente tenterà di eseguire delle azioni in cui è necessario aver eseguito l'accesso senza essere autenticato.

Ogni azione che si vuole compiere, come per esempio l'accesso tramite credenziali, deve essere gestito all'interno dell'applicazione in modo molto accurato. Ogni campo di input è stato gestito in modo da attuare una validazione anche lato client informando in maniera diretta l'utente nel caso in cui i dati inseriti non siano corretti. A seguito dell'invio di ogni form viene eseguita una chiamata alle API del backend tramite l'interfaccia di supporto *Network* implementata in modo che ritorni eventuali messaggi di errore in maniera standardizzata.

Nel momento in cui l'utente esegue il login vengono aggiornate sul device le informazioni legate all'account, creando un oggetto Realm partendo dal modello, incapsulando ogni informazione scaricata dal server in modo che ogni sezione dell'applicativo possa accedervi localmente. L'accesso permette inoltre di accedere ad alcune sezioni dell'applicazione che altrimenti non si potrebbero visualizzare poichè ogni chiamata alle API deve essere corredata delle informazioni legate all'utente che esegue una determinata azione.

Infine anche le funzioni di logout sono gestite all'interno di un helper che incapsula tutte le operazioni necessarie per far si che nessun dato sensibile rimanga salvato in locale a seguite della disconessione dell'account e non vi siano casi di incosistenza tra le informazioni presenti in locale e sul server.

4.4 Shop

Ogni esercizio inserito all'interno del circutio MyGelato è formalizzato concettualmente come un oggetto *Shop* che incapsula tutte le informazioni che il server rende disponibili al download di volta in volta: identificativo, nome, indirizzo, recapito telefonico, ecc...

La gestione degli Shop all'interno dell'applicazione è stato necessario poichè la visualizzazione delle informazioni necessarie in più di un nodo fondamentale dell'applicazione avrebbe reso impossibile la richiesta dei dati necessari solo in caso di necessità. Il download sul device delle informazioni dei device avviene all'interno della schermata della Ricerca in maniera asincrona, a causa della grossa mole di dati da dover scaricare, e permette all'utente di avere anche offline le informazioni legate agli shop già visualizzati.

È stato quindi creato un modello per l'oggetto Shop sempre derivante da un oggetto Realm così da sfruttare tutta la potenza della libreria per il salvataggio e la gestione dei dati in database. Ad ogni ricerca le gelaterie vengono nuovamente scaricate così da avere sempre tutte le informazioni aggiornate, creando però un collo di bottiglia in termini di prestazioni che verrà spiegato nel dettaglio nel capitolo successivo.

4.5 Ricerca

4.6 Marketing Digitale

Il primo flusso logico principale ad essere stato implementato è stato quello di marketing digitale poichè legato solo in parte alla presenza di un utente registrato e autenticato all'applicazione; sono quindi presenti alcune operazioni comunque disponibili anche senza aver eseguito il login.

La prima sezione ad essere implementata è stata quella legata alla visualizzazioni delle carte del Mastro Gelatiere poichè il numero delle promozioni disponibili era da visualizzare anche all'interno della navigazione principale e quindi la richiesta per scaricarle doveva essere presente anche all'interno della schermata principale dell'applicazione.

Per ottenere un maggiore riutilizzo del codice si è scelto di utilizzare la stessa schermata per visualizzare sia le carte del mastro gelatiere sia quelle delle singole gelaterie ottenendo una visualizzazione differente in base al collegamento che avrebbe portato l'utente all'interno della sezione. Allo stesso modo si è utilizzato uno stesso modello per entrambe le carte definendo un elemento Realm di nome *Card* che incapsula i dati necessari all'utilizzo delle carte nel sistema differenziandole grazie ad un flag.

Avendo quindi già implementato la visualizzazione delle carte, scaricate tramite chiamata alle API, è stato semplice eseguire il passaggio successivo che dalla singola gelateria portava alle proprie carte promozionali. A fianco dell'icona che funge da collegamento alle carte è stato successivamente inserita l'icona per aggiungere ai preferiti il singolo esercizio.

Si è dovuto creare un secondo modello all'interno del database che semplicemente identifica gli esercizi salvati tra i preferiti in modo che queste informazioni rimangano salvate in locale sul dispositivo anche senza un utente che si sia loggato.

L'inserimento di un flag all'interno del modello degli shop non avrebbe funzionato poichè sarebbe stato sovrascritto ad ogni aggiornamento degli shop durante il download e l'aggiornamento con le informazioni sul server.

Una volta che una gelateria viene aggiunta ai preferiti, nel caso l'utente si sia autenticato verrà eseguita una chiamata al server che attiverà la ricezione di notifiche push e in automatico verrà anche aggiunto un controllo per il geofencing sul dispositivo.

4.6.1 Carte Promozionali

L'implementazione delle carte promozionali ha seguito uno sviluppo volto ad utilizzare lo stesso codice e le stesse visualizzazioni sia per la la presentazione delle carte legate al singolo esercizio sia per quelle del mastro gelataio. Prima di tutto si è quindi realizzato il modello in Realm contenente le principali informazioni legate alla carta: identfiicativo, titolo, immagine, descrizione, eventuale link e recapito teleofonico.

È stata creata una schermata unica per la visualizzazione delle carte affiancate con un layout abbastanza grafico dove ogni carta mostra la propria immagine e al click su un'icona ruota su se stessa per permettere di leggere la descrizione ed eventualmente ottenere nuove informazioni tramite i l link proposto.

Il download delle informazioni deve avvenire all'apertura della schermata grazie ad una richiesta alle API in background così da non bloccare il dispositivo dell'utente. Tutti i dati scaricati vengono salvati in locale così da essere disponibili anche offline e nel caso di modifiche saranno aggiornati eliminando anche le promozioni non più disponbili perchè scadute.

Lo stesso procedimento, nel caso delle carte del mastro gelatiere però deve essere eseguito anche all'interno della schermata principale poichè il numero delle carte è da visualizzare all'interno dell'immagine che rimanda alla sezione del mastro

gelatiere. Il collegamento a questa sezione rimane valido comunque da qualsiasi parte dell'applicazione poichè in ogni caso tutti dati vengono anche aggiornati al momento, permettendo così di inserire il collegamento ogni volta che vengono presentate le informazioni legato ad uno shop.

4.6.2 Preferiti

Il primo sviluppo legato alla sezione dei preferiti è stata quella della gestione dell'aggiunta e della rimozione all'interno della visualizzazione della ricerca in cui tramite un'icona l'utente può salvare localmente le sue preferenze. Successivamente si è implementata la sezione dell'applicazione che permette la visualizzazione della lista degli esercizi preferiti con la conseguente gestione, direttamente raggiungibile dalla navigazione principale.

Infine avendo gestito l'aggiunta e la rimozione dei preferiti grazie ad un sistema centralizzato che racchiudeva tutte le operazioni necessarie è stato abbastanza intuitivo poter inserire un middleware che gestisse l'aggiunta e la rimozione di una gelateria dai preferiti anche per quanto riguardava l'iscrizione alle notifiche push da parte del server e del geofencing da parte del dispositivo.

Per l'implementazione della funzinalità di notifiche push si è scelto di utilizzare i servizi resi disponibili da Google stessa utilizzando inizialmente *Google Cloud Messaging* per poi passare ad utilizzare Firebase, nuovo applicativo appena reso disponbile e sicuramente più innovativo e aggiornato. Per far dialogare in maniera bidirezionale l'applicativo e il server ogni volta che un utente esegue il login ad una applicazione viene attivato firebase richiedendo al server un token che verrà poi passato al sistema di gestione delle notifiche che si metterà in ascolto.

```
FirebaseApp.initializeApp(this,
  FirebaseOptions.fromResource(this));
```

Ogni volta che l'utente aggiungerà e rimuoverà un preferito verrà quindi eseguita una chiamata al backend con lo scopo di informarlo dell'aggiunto dello shop a quelli di interesse, in questo modo nel momento in cui sarà disponibile una nuova carta promozionale per quello schop verrà visualizzata una notifica sul device che permetterà all'utente di accedere direttamente alla sezione delle carte promozionali specifiche per quell'esercizio.

Allo stesso tempo, solo a livello applicativo verrà inoltre sottoscritto al sistema una richiesta di geofencing che produrrà un intent ogni qual volta l'utente rimarrà entro l'area di 5 kilometri dalla gelateria per almeno un cinque minuti, il quale verrà gestito per mostrare anche in questo caso una notifica che permetterà di accedere invece alle informazioni della gelateria all'interno della lista dei preferiti. Ovviamente per i dispositivi con le ultime versioni di Android sarà necessario richiedere anche in questo caso la possibilità di accedere alla localizzazione.

```
mGeofenceList.add(new
    Geofence.Builder()
      .setCircularRegion(
          shop.latitude,
          shop.longitude,
          GEOFENCE_RADIUS_IN_METERS
      )
      .setLoiteringDelay(2 * 60 * 1000)
      .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_DWELL)
      .build()
);
```

4.7 E-Commerce

Il sistema di e-commerce è stato l'ultimo tassello dell'applicazione che ha concluso il lavoro di sviluppo diretto di questo elaborato. Si è trattato di unificare par-

te delle componenti già implementate creando un workflow che permetta all'utente di comprare e utilizzare i coupon digitali in sicurezza e con facilità.

La prima fase è stata legata alla rappresentazione formale dei buoni salvati sul server legati ad un determinato account e dei buoni disponibili all'acquisto legati invece ad una determinata gelateria, in particolare è stato importante valutare quali informazioni mantenere sempre in locale per non inficiare la consistenza dei dati presenti sul dispositivo rispetto a quelli online.

Definita la struttura degli elementi che si sarebbero andati ad utilizzare si è creata la sezione dedicata alla gestione dei coupon personali disponibili ad ogni utente, ovviamente legado ogni funzione alla necessaria autenticazione tramite account iscritto all'applicazione MyGelato. Questa sezione permette di avere una visione completa delle proprie informazioni aggiornate poichè vi è un forte legame con i dati presenti sul server online, infatti molte delle funzioni legate al sistema di e-commerce comprendono l'utilizzo di account e dispositivi diversi che interagiscono e che non devono assolutamente creare casi di incosistenza.

Per rendere accessibile all'utente l'accesso alle funzioni di acquisto e di riscatto di un buono in maniera diretta si sono inseriti all'interno della navigazione principale dell'applicazione i collegamenti a queste sezioni insieme ad un bottone flottante che riporta alla gestione dei coupon; nel caso si sia effettuato l'accesso con un account di tipo gelatiere si verrà riportati invece alla sezione per la validazione dei buoni.

4.7.1 Coupon

Per formalizzare il concetto di coupon personale di ogni utente si è scelto di creare una classe *Coupon*, estensione di un modello Realm, che incapsula tutte le informazioni disponibili: identificativo, nome, gelateria di riferimento, acquirente, ecc... Sfruttando Realm le operazioni di gestione e salvataggio sul database sono

Figura 4.2: Coupon

particolamente semplificate ed è essenziale che venga assicurata un'alta sicurezza nella gestione.

Ogni coupon è unico grazie a un identificativo personale, possiede alcune informazioni importanti come l'acquirente e l'attuale propietario, la data di vendita e anche alcune informazioni non direttamente utili all'utilizzo tramite applicativo. Ogni dato viene salvato sul server e richiesto tramite una chiamata alle API quando si utilizza la sezione per la gestione dei buoni.

Dentro questa sezione sono presenti tre elenchi in cui sono visibili i coupon validi (utilizzabili e condivisibili), ricevuti da altri utenti, e la lista completa di tutti buoni legati all'utente. Come si può vedere in figura 4.2, ogni coupon visualizza alcune informazioni di base e sono presenti i collegamenti diretti per l'utilizzo e lo share tramite altri canali di comunicazioni.

4.7.2 Acquisto

4.7.3 Condivisione e Riscatto

Il meccanismo di condivisione e riscatto ha coinvolto funzioni complementari che dovevano sfruttare anche componenti a livello di sistema operativo. Per permettere il dialogo tra due dispositivi differenti si è scelto di sfruttare i canali di comunicazioni più comuni e diffusi anche tra gli utenti medi in modo che il sistema di share sia il più facile possibile e del tutto conforme al sistema di condivisione standard della piattaforma.

Dalla lista di coupon disponibili l'utente ha la possibilità di cliccare sull'icona per la condivisone e verrà inviata una richiesta - *Intent* - a livello di sistema in broadcast per poter selezionare il metodo che si preferisce per inviare un testo

Figura 4.3: Condivisione e Riscatto

precompilato insieme ad un link. Il testo presenta brevemente l'oggetto della condivisione e invita ad utilizzare il link per riscattare il coupon specificato.

Il link riporta a una pagina resa disponibile dal backend che riconosce il sistema operativo dal quale ci sis sta collegando per effettuare un redirect ad un link particolare che verrà gestito in automatico dal device con cui si sta visualizzando il sito. Nel caso in cui vi sia installata sul device l'applicazione MyGelato allorà verrà avviata la schermata di riscatto precompilata con i dati necessari, altrimenti si verrà reindirizzati agli store principali per consigliare il download dell'applicazione.

Il mezzo per permettere l'identificazione del coupon da condividere è un codice alfanumerico a sei cifre, il PNR, che viene condiviso in automatico insieme al resto delle informazioni. All'interno della schermata di riscatto di un buono, raggiungibile dirattamente dalla navigazione princiapale dell'app è necessario inserire il codice del buono che si vuole riscattare. Come detto nel caso si arrivi sulla schermata tramite il link condiviso allora il campo di input del codice sarà precompilato.

Una volta inserito il codice, che verrà validato al momento con dei controlli basilari, sarà effettuata una richiesta al server tramite chiamata API che attuerà lo spostamento del buono da un utente all'altro. Per entrambi gli account vi sarà quindi l'aggiormento delle informazioni sul proprio dispositivo la prima volta che verranno richiesti i coupon disponibili.

L'utente che ha effettuato il riscatto verrà immediatamente reindirizzato sulla schermata di lista dei coupon nel caso in cui il processo sia andato a buon fine, altrimenti verrà notificato con un messaggio di errore. La figura 4.3 mostra tutti i passaggi principali per la condivisione e il riscatto di un buono.

4.7.4 Utilizzo e Validazione

L'ultimo passaggio del workflow di e-commerce è l'utilizzo dei coupon che si sono acquistati o che si hanno riscattato perchè condivisi da altri utenti. In questo processo entrato in gioco due entità che sono utenti che spesso non entrano in contatto se non solo per i pochi momenti che servono all'acquisto materiale del valore del buono utilizzato. Questa situazione pone alcune limitazioni negli strumenti utilizzati per la comunicazione tra i due dispositivi che devono collaborare mantenendo uno stato costantemente consistente sia sui device fisici che sul server.

Per superare questo ostacolo si è scelto uno strumento molto diffuso che è l'utilizzo di QR code che permette di generare un'immagine a partire da un testo, in questo caso il PNR code lagato al coupon che si vuole utilizzare, leggibile ed interpretabile da un qualsiasi cellulare con un fotocamera. Chiunque abbia effettuato l'accesso all'applicazione con account di tipo gelatiere potrà validare il coupon direttamente tramite l'applicazione MyGelato grazie a una chiamata API al backend che confermerà o meno l'avvenuta conclusione del processo senza problemi.

La sezione per generare e visualizzare il codice QR da mostrare in gelateria è raggungibile cliccando sul singolo coupon all'interno della gestione dei propri buoni. Non essendo presente in questo caso una connessione bidirezionale con il backend, per aggiornare la schermata una volta concluso il processo di utilizzo l'applicazione effettua un polling al server ogni cinque secondi per verificare se il coupon sia stato utilizzato o meno.

Chiunque possieda un account di tipo gelatiere avrà nella navigazione principale dell'applicativo il collegamento alla sezione per la convalida. Verrà avviata immediatamente la fotocamera che, tramite una libreria esterna, verificherà la presenza o meno di codici QR leggendoli. Dopo una validazione semplice si effettettuerà la chiamata al server dando poi la possibilità di convalidare altri coupon senza dover ritornare ogni volta alla schermata iniziale. Nel caso in cui la fotocamera sia già

Figura 4.4: Acquisto

occupata da altri processi attivi sul dispositivo o nel caso in cui l'utente abbia rimosso alcuni permessi all'applicativo verrà visualizzato un messaggio di errore che comunicherà al proprietario del device cosa fare per risolvere il problema.

A conclusione del processo di validazione di un coupon entrambi i dispositivi verranno aggiornati per mantenere coerenza tra i dati presenti sul sistema risolvendo così in maniera semplice il problema principale del dialogo tra due dispositivi rendendo l'utilizzo di un buono molto semplice sempre basandosi sul fatto che ormai uno smartphone sia diventato un bene comune. La figura 4.4 permette di visualizzare i passaggi fondamentali per l'utilizzo e la validazione di un buono.

Bibliografia

[1]

Elenco delle figure