Capitolo 1

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

Considerato infine che alcuni nodi centrali dell'elaborato necessitano di specifiche minime in termini di affidabilità e sicurezza si è scelto di utilizzare librerie esterne anche per le funzioni di utilizzo del database e di connessione al server, rispettivamente Realm e OkHttp, che oltre a semplificare lo sviluppo garantiscono un'alta gestione degli errori.

1.1 Sviluppo Android

Questa tesi si inserisce all'interno del lavoro di creazione della piattaforma MyGelato, il quale include il supporto sia alla piattaforma Android sia alla piattaforma iOS. Essendosi però creati due differenti progetti durante la prima fase di progettazione, si è potuto scegliere come metodo implementativo l'utilizzo della programmazione nativa Android che si basa su Java per la programmazione e su XML per la creazione delle risorse.

Sicuramente lavorare in questo modo 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 alla necessità di creare applicazioni cross-platform distribuibili su sistemi operativi diversi.HTMLIT:PROGNATIVA Non avendo quindi necessità di mantenere alta la portabilità del codice su altre piattaforme, si è preferito sviluppare tramite programmazione nativa; potendo quindi sfruttare pienamente l'architettura del sistema operativo sottostante e gli strumenti di sviluppo forniti ufficialmente.

Si è scelto quindi di utilizzare come principale strumento *Android Studio*, Integrated Development Environment (IDE) dedicato alla programmazione Android distruibuito da JetBrains IntelliJ IDEA e Google che è ormai il miglior software per la creazione dei applicazioni su questa piattaforma. ANDROIDDEVELOPERS:FIRSTAPP

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*), mentre per il test delle chiamate al backend della piattaforma si è utilizzata la web application Postman POSTMAN che permette di testare e sviluppare le APIs con cui due applicativi possono dialogare.

1.1.1 Programmazione Nativa

Le applicazioni Android sono sviluppate utilizzando il linguaggio Java. È un linguaggio di programmazione molto popolare sviluppato da Sun Microsystems (ora di proprietà di Oracle) orientato agli oggetti a tipizzazione statica, specificatamente progettato per essere il più possibile indipendente dalla piattaforma di esecuzione. //https://it.wikipedia.org/wiki/Java_(linguaggio_di_programmazione)

Per lo sviluppo di un'applicativo si devono sfruttare le principali caratteristiche di Java andando ad estendere, per ogni schermata che si vuole creare, la classe *Activity* implementata come standard all'interno delle libreria Android. Qualsiasi progetto sarà quindi formato da un insieme di classi che rappresenteranno logicamente ogni singola schermata unite ad altre classi che incapsuleranno invece le funzioni, i modelli e ogni altro componente standard di Android (Broadcast Receiver, Service, ecc...).All'interno dell'ultima distribuzione di Android sono state incluse alcune delle funzionalità di Java 8, ma l'alta frammentazione della piattaforma non ne permette correttamente l'utilizzo. Questo si ripercuote su alcune funzionalità come le chiamate http che non supportano alcune delle semplificazioni che sono state fatte ad esempio nelle connessioni sicure grazie a TLS.

Come si può vedere all'interno dello snippet di codice la classe *Main* estende la classe *AppCompatActivity* e dopo aver richiamato il costruttore della propria superclasse esegue la creazione dell'internfaccia utente grazie al file di layout *activity_main* che ne specifica la struttura e le impostazioni, scritto in XML.

```
public class Main extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    ...
}
```

In informatica XML (sigla di eXtensible Markup Language) è un metalinguaggio per la definizione di linguaggi di markup, ovvero un linguaggio marcatore basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo. All'interno dello sviluppo di applicazioni Android rappresenta il linguaggio per la creazione delle risorse: layout, icone, manifest, ecc...

L'esempio più ecclatante è l'utilizzo dell'XML per produrre il manifesto dell'applicazione, che ne rappresenta ed espone la struttura principale dichiarando ogni singola schermata o servizio presenti specificando anche il tema da utilizzare, il nome e alcune opzioni di avvio. Sono scritte in XML anche tutte le risorse interne all'applicazione come si può notare nell'esempio di codice sottostante che descrive una schermata con una singola immagine.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        src="drawable/icon" />
</LinearLayout>
```

1.1.2 Android Studio

Android Studio WIKIPEDIA:ANDROIDSTUDIO 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. Sostituisce gli Android Development Tools (ADT) di Eclipse, diventando l' IDE primario di Google per lo sviluppo nativo di applicazioni Android offre ancora più funzioni che migliorano la produttività durante la creazione di applicazioni Android, come ad esempio: un sistema di compilazione Gradle, un ambiente unificato dove è possibile sviluppare per tutti i dispositivi, esecuzione e costruzione di un file APK per la pubblicazione di una applicazione sui canali di distribuzione più importanti e l'integrazione GitHub.

Ogni progetto creato all'interno del software contiene alcuni moduli principali che riguardano l'applicazione vera e propria, le librerie incluse e i moduli proprietari di Google che permettono l'utilizzo dei suoi servizi principali come Firebase e le Google Maps API.

A livello di build del sistema si trova uno script Gradle che a partire da un manifesto, che dichiara la struttura dell'applicazione, i file Java contenenti il codice sorgente (inclusi i test) e l'insieme delle risorse e degli asset può generare un file apk firmato installabile su qualsiasi device Android.

Per la generazione delle risorse Android Studio mette a disposizione alcuni tool, tra cui alcuni grafici, che permettono l'implementazione e il test dell'interfaccia utente senza dover di volta in volta avviare l'applicativo su un emulatore, messo a disposizione dal sofwtare stesso, o su un device fisico.

Ovviamente AS incapsula tutti i tool utili alla programmazione come la formattazione, lo stile e la correzione in realtime del codice scritto effettuando anche il controllo sintattico e un basilare confronto semantico seguendo i pattern standard della piattaforma.

1.1.3 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. WIKIPEDIA:GIT 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 al-

goritmi 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.

1.2 Database

La piattaforma Android fornisce diversi metodi e strumenti per salvare i dati delle applicazioni in modo persistente. Per quanto riguarda le preferenze relative alle impostazioni dell'applicazione si possono implementare le *Shared Preferences* per salvare le scelte dell'utente; mentre per quanto riguarda la memorizzazione persistente di una grossa mole di dati si possono utilizzare strumenti come l'*Internal Storage* o un *Database SQLite*. //cit http://www.html.it/articoli/la-gestione-dei-database-in-android-1/

Esistono però anche alternativa agli strumenti standard forniti ufficialmente per la piattaforma e uno di questi è la libreria *Realm*: database object-oriented che non incapsula SQLite ma lo sostituisce con un sistema di memorizzazione scritto in C++ che permette l'accesso ai dati anche tramite l'utilizzo di linguaggi di programmazione differenti. Modifica inoltre il metodo di accesso ai dati salvati non richiedendo query esplicite ma semplici richieste in Java utilizzabili direttamente all'interno del codice implementato, semplificando di gran lunga l'utilizzo del database durante lo sviluppo di qualsiasi applicazione.

1.2.1 Realm

La prima configurazione per l'utilizzo di Realm è rintracciabile nella creazione di una semplice dipendenza all'interno dei file di assetto di *Gradle* (tool di Andriod Studio per il compile, il deploy e la gestione delle dipendenze) dove va inserita l'ultima versione disponibile dell'applicativo. È inoltre necessario inserire una prima inizializzazione all'interno dell'avvio dell'applicazione, andando quindi ad estendere la classe *Application* esposta dalle libreria dell'sdk di Android.

Una volta configurato il plugin si possono creare i modelli degli oggetti che saranno da salvare all'interno della base di dati, come per esempio un oggetto *Utente* rappresentato ad esempio dalla classe *User* come si può vedere nello snippet di codice CODICE dove sono descritte le principali variabili che lo compongono, tra cui la chiave primaria denotata dalla keyword *PrimaryKey.*

```
import io.realm.RealmObject;
import io.realm.annotations.PrimaryKey;
public class User extends RealmObject {
    @PrimaryKey
    public String uuid;
    public String firstName;
    public String lastName;
    public String email;
    public User() {
    }
}
```

Dalla documentazione ufficiale è anche richiesto un costruttore vuoto, una serie di annotazioni per descrivere quali oggetti sono da ignorare durante il salvataggio e vi è infine la possibilità di aggiungere funzioni setter/getter.

L'utilizzo di questi modelli è molto semplificato poichè Realm permette di creare delle transizioni sui singoli oggetti creati così da poter aggiornare la loro versione contenuta all'interno del database, assicurando in ogni istante le proprie ACID delle operazioni. Si può vedere nel seguente listato un esempio di creazione o aggiornamento di un oggetto user in base al fatto che sia o non sia già presente all'interno della base di dati.

```
Realm realm = Realm.getDefaultInstance();
User user = new User(Params...);
realm.executeTransaction(new Realm.Transaction() {
        @Override
        public void execute(Realm realm) {
            realm.copyToRealmOrUpdate(user);
        }
});
realm.close();
```

Allo stesso modo con cui vengono semplificati gli inserimenti di dati all'interno del db è forse fondamentale capire come vengono gestite anche le query sulla base di dati; Un esempio molto semplice visibile nello snippet di codice CODICE mostra come sia possibile richiedere un utente con email non nulla all'interno dei dati sal-

vati ottenendoli in un *ArrayList* utilizzabile direttamente per scorrere ogni singolo oggetto ottenuto dalla richiesta.

```
RealmResults<User> results =
      Realm.getDefaultInstance()
            .where(User.class)
            .notNull("email")
            .findAll();
```

Ultimo elemento da presentare della libreria Realm sono i *listener* che si possono attivare sia sull'intero database sia su una singola porzione in modo che eseguano una determinata callback nel momento in cui avviene una modificati dei dati monitorati. Un breve esempio è visualizzato nel codice seguente.

```
Realm realm = Realm.getDefaultInstance();
realm.addChangeListener(new RealmChangeListener<Realm>() {
      @Override
      public void onChange(Realm element) {
            // do something...
      }
});
```

1.3 Network Connection

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 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 difficilmente gestibili poichè non sono automatizzate e son stati effettuati dei cambiamenti nelle 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. SQUARE:OKHTTP

1.3.1 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 serve creare un oggetto *Request* a cui vanno passati i parametri necessari per poi eseguire la chiamata ottenendo infine un oggetto *Response* all'interno del quale sono presenti i dati scaricati, il codice di risposta della chiamata e anche eventuali errori.

1.4 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 schermate, non più presenti a video; per questo motivo si è scelto di utilizzare la libreria EventBus sviluppata da GreenRobot GITHUB:EVENTBUS e pensata esattamente per gestire e risolvere questo tipo di problemi.

1.4.1 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 centralizzata monitorando le performance e gestendo gli errori che potrebbero essere sollevati da uno sviluppo multithreading.GREENROBOT:EVENTBUS

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 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());
```

1.5 E-Commerce

Ogni sistema di e-commerce presente sugli applicativi moderni è normalmente basato su alcune delle librerie più diffuse nell'ambito dei pagamenti online. La sicurezza e l'affidabilità che questo tipo di sistemi devono assolutamente garantire rende necessario l'uso di tools testati e specifici per l'utilizzo. Questi sistemi ovviamente devono garantire l'utilizzo sia da parte dell'applicativo di frontend sia da parte del server di backend in modo che l'intero sistema sia consistente.

In questo specifico caso si è scelto di utilizzare il tool Stripe, piattaforma per i pagamenti online disponibile anche come plugin da poter importare all'interno del proprio progetto che permette di avere una gestione dei metodi di pagamento di un singolo utente e di eseguire transizioni monetarie tra un account user ed un account commerciant.

1.5.1 Stripe

L'implementazione di Stripe richiede un'inizializzazione da parte del sistema di backend della piattaforma, il quale espone poi per ogni account un token identificativo che sarà necessario per le richieste effettuate tramite la piattaforma Stripe.

Ogni token viene inserito all'interno dei dati del singolo account in modo che grazie all'utilizzo del plugin importato come dipendenza all'interno del progetto Android sia possibile richiedere i metodi di pagamento già collegati all'account selezionato e nel caso aggiungerne di nuovi. Questo sistema non espoerrà mai le informazioni sensibili e riservate date dall'utente all'aggiunta di una carta poichè

saranno gestite internamente alla libreria e salvate solo sul sistema di e-commerce. Gli unici dati, utili all'identificazione e alla scelta di volta in volta del metodo di pagamento saranno solo le ultime quattro cifre della carta e la data di scadenza.

Nel momento in cui si vorrà eseguire una transizione monetaria tra due account sarà necessario avere il token identificativo dell'utente e quello dell'esercizio su cui si vorrà acquistare, in questo caso le gelateria inserite all'interno del circuito MyGelato. Interverrà poi il sistema di pagamento interno alla piattaforma Stripe per effettuare il passaggio e concludere la transizione in sicurezza e con l'affidabilità di un alto controllo sugli errori.

Oltre alla semplicità di utilizzo di un tool di questo tipo che permette l'inserimento di pagamenti online all'interno dei propri progetti anche senza una specifica esperienza nel campo, è fondamentale la parte di test resa disponibile da Stripe che permette di aggiungere dei metodi di pagamento fittizzi e di eseguire delle transizioni senza che vi sia un reale spostamento monetario ai fine del testing dei proprio applicativi.