POLITECNICO DI TORINO

Collegio di Ingegneria Informatica, del Cinema e Meccatronica

Corso di Laurea Magistrale
in Ingegneria Informatica (Computer Engineering)

Tesi di Laurea Magistrale

Personal Financial Management: Analisi di Mercato, definizione e sviluppo di una soluzione mobile innovativa

Relatore
prof. Giorgio Bruno

Tutor Aziendale
Luca Perino

Candidato
Emanuele Fricano

Ottobre 2017

Alla mia famiglia.

INDICE

CAPITOLO 1

INTRODUZIONE

Dagli anni Settanta ad oggi lo sviluppo del mondo legato all'informatica è stato e continua a essere impressionante. Nei primi anni Ottanta la tecnologia era giunta a un punto tale da consentire una svolta all'epoca straordinaria: la creazione di un computer personale, con costi accessibili quasi a chiunque.

Questa continua crescita non si è fermata e nell'ultimo decennio vi è stata addirittura un'accelerazione nel processo evolutivo. Se alla fine degli anni Novanta l'utilizzo di un computer era indispensabile per svolgere diversi compiti nella vita di tutti i giorni, oggi l'utilizzo di *tablet* e/o *smarpthone* è per tutti quasi irrinunciabile.

I primi *smartphone* combinavano le funzioni di un elaboratore palmare con quelle di un telefono mobile. I modelli più recenti si sono arricchiti della funzionalità di dispositivi multimediali in grado di riprodurre musica, scattare foto e registrare video. Oggi quasi tutti gli *smartphone* sono inoltre dotati di schermo tattile ad alta risoluzione e web browser tramite i quali sono in grado di caricare sia normali pagine web sia siti web appositamente creati per i dispositivi mobili. Caratteristica diffusa è inoltre quella di poter installare funzionalità aggiuntive attraverso le cosiddette App scaricate dai rispettivi market di vendita. Lo sviluppo di dispositivi di questo tipo sempre più performanti e capaci di svolgere compiti, alla stessa maniera o quasi di un computer, ha fatto sì che gli utenti hanno preferito negli anni accedere ai vari servizi non più tramite l'utilizzo di un computer, ma direttamente dal proprio *smartphone*. I motivi che spingono l'utenza a utilizzare uno *smartphone* invece che un PC per connettersi a internet sono molteplici e non riguardano solamente il livello di conoscenze informatiche degli individui o la loro passione per la tecnologia. Chi infatti utilizzava già il computer per navigare su Internet ha trovato nello *smartphone* un valido sostituto,

da utilizzare in mobilità o nei momenti in cui le possibilità offerte dai PC non sono necessarie. Sono invece gli utenti meno esperti ad aver avuto i maggiori benefici dall'arrivo delle tecnologie *mobile*, molto più semplici e pratiche da usare. La presenza di connessioni veloci anche in aree rurali, grazie al 3G e agli altri standard di rete, e la semplicità di utilizzo di *smartphone* e *tablet* permette oggi anche a bambini ed anziani di accedere al web senza la minima conoscenza informatica. Ovviamente i computer Desktop non vengono sfruttati solo per la navigazione Internet, per cui il rischio di una sostituzione totale è molto basso, ma per quanto riguarda lo specifico campo di utilizzo, tra qualche anno, è probabile che gli *smartphone* cresceranno ancora, trainati dalle economie emergenti, mettendo i computer in secondo piano. In questi anni quindi molti servizi che erano accessibili solo tramite un computer come un pagamento ad esempio, possono anche essere fatti in mobilità. Molti enti erogatori di questi tipi di servizi hanno così investito nella ottimizzazione di ciò che offrivano per dispositivi mobile. Altri nuovi servizi prima impensabili sono invece nati grazie al mobile stesso, basti pensare ad esempio ai pagamenti fatti tramite App senza l'utilizzo continuo di Bancomat e/o Carte di Credito, permettendo un'esperienza con una quantità di comfort sempre crescente all'utente. Ovviamente se da un lato vi sono maggiori *comfort*, dall'altro per gli utenti nascono nuove necessità. Ad esempio, per quanto concerne il mondo finanziario, un utente tramite l'App della propria banca, oltre a monitorare facilmente e costantemente la propria situazione economica, può effettuare dei pagamenti. Negli ultimi anni sempre più istituti bancari hanno dato maggiore importanza al PFM, Personal Financial Management o letteralmente gestione delle finanze personali, che permette una migliore gestione del denaro e categorizzazione delle proprie spese. L'obiettivo di questo elaborato di Laurea è lo sviluppo di un PoC (*Proof of Concept)* di un'applicazione per PFM sviluppata per i *devices* con sistema operativo Android e che permetta di dare una interfaccia confortevole agli utenti e di distinguersi per usabilità e semplicità rispetto ad altre soluzioni già presenti sul mercato. Nel capitolo seguente verrà descritto il processo evolutivo riguardo lo utilizzo degli *smartphone* e dei dispositivi *mobile* e dello sviluppo del *mobile financial*, descrivendo alcune App che danno la possibilità di pagamento tramite dispositivi mobili. Nel capitolo 3 verrà spiegato cosa è il Personal Financial Management e verranno poi presentare le

caratteristiche comuni delle soluzioni presenti sul mercato. Una più attenta analisi sulle soluzioni presenti sul mercato verrà affrontata al capitolo 4, descrivendone di ognuna i punti di forza e le loro particolarità. Nel capitolo 5 verranno analizzate le librerie presenti sul mercato per l'implementazione di grafici utilizzati nelle App che si occupano di PFM. Nel capitolo successivo verrà mostrata in dettaglio l'applicazione sviluppata descrivendone l'architettura e mostrando alcuni esempi del codice implementato. Infine saranno presenti le conclusioni riguardo il lavoro svolto.

CAPITOLO 2

STATO DELL'ARTE

In questo capitolo verrà spiegato più in dettaglio il processo evolutivo riguardo lo utilizzo degli *smartphone* e dei dispositivi *mobile* più in generale, dal loro avvento sul mercato internazionale ai giorni nostri. In seguito nella parte finale del capitolo verrà mostrato in corrispondenza lo sviluppo del *mobile financial*.

2.1 Dalla nascita dei dispositivi mobili agli *Smartphone*

Ai giorni d'oggi il possesso di uno *smartphone* non è cosa alquanto inconsueta quanto il non possederne uno. Attualmente bisogna guardare agli *smartphone* non come solamente a telefoni intelligenti, come suggerisce la traduzione letterale, ma possono essere considerati ormai alla stregua dei personal computer. Per disporre di dispositivi con funzionalità simili ai personal computer come ad esempio la lettura e scrittura su pagine Web ci sono voluti più di 40 anni. Gli antenati degli *smartphone* erano dispositivi grandi, non comodi da usare e sempre bisognosi di carica. Visti gli enormi vantaggi che si potevano sfruttare dall'utilizzo di questi dispositivi si è investito moltissimo nel corso degli anni nello sviluppo di *devices* con sempre maggiori funzionalità e sempre più pratici da utilizzare. Per questo motivo è oggi possibile poter ottenere a costi non eccessivi *devices* sempre più sottili, trasportabili, con display grandi e ad alta risoluzione. I dispositivi mobili hanno cambiato le nostre vite e la loro evoluzione è tale che oggi un loro utilizzo senza funzionalità come il *text messaging* e il mobile Internet sarebbe praticamente impensabile. I dispositivi mobili sono alla loro quinta generazione, l'ultima delle quali ancora in fase di diffusione ed adozione da parte degli utenti e resa già antiquata da nuove tecnologie che premono per affermarsi. L'idea

di creare dei dispositivi che unissero la telefonia all'utilizzo degli elaboratori elettronici risale al 1973, ma le prime vendite di tali dispositivi cominciarono solo nel 1993. Il termine *smartphone* apparve per la prima volta nel 1997 quando la Ericsson descrisse il suo GS 88 Penelope come uno Smart Phone.

Il primo *smartphone* in assoluto, chiamato Simon, fu progettato dalla IBM nel 1992 e commercializzato dalla BellSouth a partire dal 1993.

Incorporava oltre alle comuni funzioni di telefono calendario, rubrica, orologio, *block notes*, funzioni di e-mail e giochi. Per poter scrivere direttamente sullo schermo era disponibile uno specifico pennino.

Figura 1: IBM Simon primo *Smartphone* della storia

Dai primi anni 2000 la nascita ed evoluzione degli *smartphone* è strettamente legata all'evoluzione degli standard di telefonia mobile cellulare, in particolare dall'UMTS fino all'HSPA e all'LTE con capacità di connessione dati superiori ai precedenti

standard GSM/GPRS.

I primi *Smartphone* ad essersi affermati su scala internazionale sono considerati essere i BlackBerry. È nel 2001 che appare il primo Blackberry, concentrato sull'uso della posta elettronica in movimento che può essere considerato come il primo pocket pc. Venne equipaggiato con il sistema operativo Windows Pocket PC, precursore dell'odierno Windows Mobile.

I telefoni cellulari non sono stati i soli predecessori degli *smartphone* ma ovviamente hanno contribuito alle evoluzioni successive. Inizialmente questi cellulari erano dotati di applicativi simili a quelli che si utilizzavano al Pc, come Word o Excel, ma l'utente comune non era attratto da queste funzioni e li trovava particolarmente costosi. Dopo un susseguirsi di cellulari sempre più intelligenti e capaci di offrire sempre maggiori vantaggi agli utenti è nel Gennaio 2007 che si può constatare a una sorta di rivoluzione epocale. Apple presenta l'iPhone, un cellulare rivoluzionario rispetto ai precedenti. L'azienda di Cupertino fonde in un tutt'uno il suo iPod, un cellulare e un dispositivo di comunicazione tramite internet per mail, web e ricerca su mappe.

Dotato di display *touch* e fotocamera, permette inoltre la comunicazione con i profili su Yahoo o con il proprio PC e consente di sincronizzare la lista dei propri contatti.

Nel giugno 2008 viene lanciato il nuovo iPhone 3 che principalmente aumenterà la velocità di connessione rispetto alla versione precedente.

E' con questo modello che nasce l'Apple Store, che sta a dimostrare come gli utenti richiedano in massa questi nuovi dispositivi.

L' iPhone 3 è un oggetto facile da utilizzare e man mano che si evolve vi vengono aggiunti servizi, come ad esempio il GPS nella seconda generazione.

Naviga sul web, visualizza le pagine in HTML, ha lo schermo che si regola automaticamente in base all'orientamento con cui si guarda con le due modalità *landscape* o *portrait.*

Il nuovo concetto di telefono cellulare, ora *smartphone* a tutti gli effetti, viene abbracciato da Google, T-Mobile e HTC che nel 2008 si alleano e lanciano Dream, primo cellulare dotato di sistema operativo Android.

Tutt'oggi questi dispositivi possono essere arricchiti con numerose applicazioni, scaricabili da un sistema basato su Java o successivamente tramite *store* dedicati come

l'App Store di Apple e il Google Play Store per Android. Inoltre molte di queste App aggiuntive sfruttano le funzionalità fornite da questi dispositivi equipaggiati da una quantità sempre crescente di sensori quali accelerometro, giroscopio, sensore fotoelettrico, sensore laser di profondità, GPS e molti altri.
Questa crescita negli ultimi anni non ha mostrato segni di rallentamento ma al contrario continuamente migliorie ed evoluzioni si sono continuamente susseguite.

2.2 Passaggio graduale dai PC agli *Smartphone*

A rendere gli *smartphone* così performanti e funzionali rispetto a telefoni cellulari di precedente generazione sono l'aumento delle prestazioni in termini di processamento e memorizzazione grazie a processori sempre più evoluti e sempre più simili a quelli dei dispositivi fissi o portatili e a memorie sempre più capienti come ad esempio le schede SD. Se da un lato l' evoluzione dal punto di vista Hardware è in continua crescita, a essa può essere affiancato lo sviluppo di sistemi operativi progettati ad hoc per i dispositivi mobile o derivati dai sistemi operativi per personal computer e ad interfacce utente sempre più facili da utilizzare per gli utenti. La possibilità di interfacciarsi ad esempio con uno schermo tattile ha permesso una migliore *user-experience*. Nei dispositivi di ultima generazione vi sono schermi di dimensioni fisiche maggiori a parità di spazio disponibile eliminando in molti casi la necessità di tastierine fisiche. Gli *smartphone* possono essere disponibili anche in più versioni oltre a quella standard, generalmente può essere una versione più potente, oppure una con durata maggiorata dell'autonomia oppure più resistente, ma esistono anche modelli studiati per il business o che incentrano le loro caratteristiche per le esigenze d'impresa. Alcuni aspetti come la semplicità, il comfort e l'immediatezza di utilizzo degli *smartphone* e la possibilità di utilizzarli più facilmente in mobilità o anche nei momenti in cui le possibilità offerte dai PC non sono necessarie, hanno fatto sì che questi dispositivi si sviluppassero in modo incessante dal momento in cui sono apparsi sul mercato. Secondo l'Idc (Information Data Corporation) l'avanzata degli *smartphone* e dei *tablet* mette in seria difficoltà il mercato mondiale dei Pc, tanto che ad esempio nel 2015 esso ha toccato il livello più

basso dal 2007, con un calo del 10,4% rispetto all'anno precedente. Oltre alla perdita relativa alle vendite, grazie a un'analisi di StatCounter viene mostrato come gli *smartphone* e i *tablet* abbiano superato i PC come principale mezzo di fruizione di Internet. L' indagine fatta è stata su oltre 15 miliardi di pagine visualizzate e su oltre 2.5 milioni di portali internet tra la fine del 2009 e l'inizio del 2016. Questi numeri permettono di accrescere l'importanza che occorre dare a queste statistiche. Ad ogni modo, a prescindere dal campione utilizzato il fatto rilevante è che il mondo oggi naviga sempre meno con il PC e sempre di più con i *devices* mobili. Questi ultimi hanno permesso l'accesso ai servizi web a un pubblico molto più ampio, anche a coloro che erano stati tagliati fuori dal *digital divide* o divario digitale.

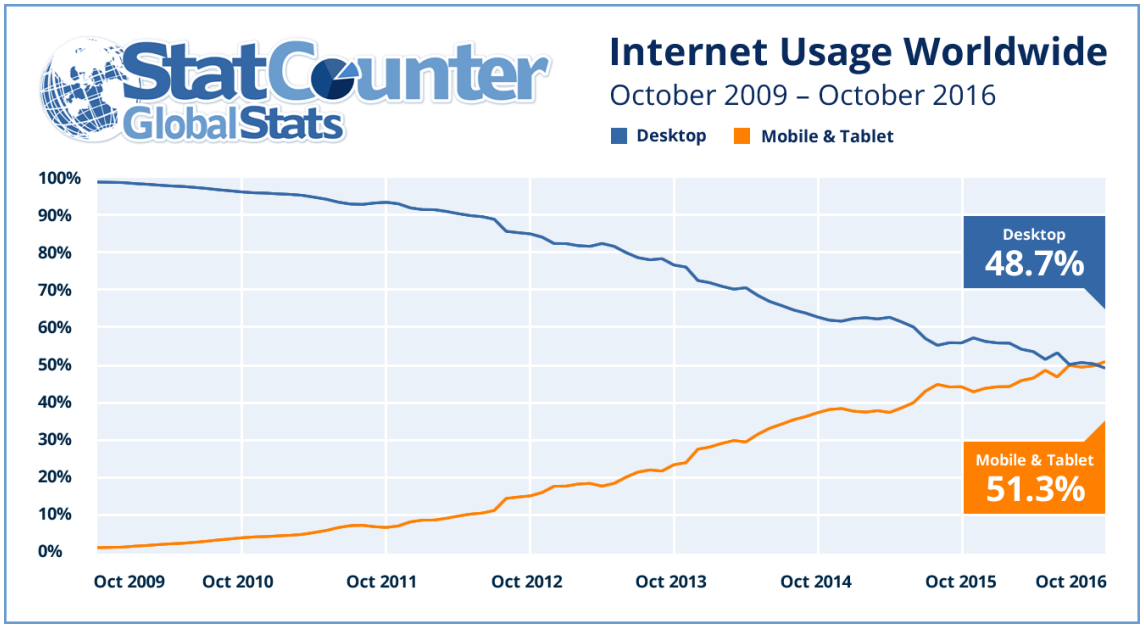

Figura 2: Statistica uso di Internet da Desktop e da Mobile nel periodo 2009-2016

Come viene mostrato dal grafico sulla ricerca svolta da StatCounter l'utilizzo di Internet da *mobile* è in continua crescita dalla fine del 2009, aumentando di più del 50%. Considerando invece il numero di accessi Internet da parte dei computer Desktop è diminuito durante lo stesso periodo di quasi il 49%.

I motivi che spingono l'utenza a utilizzare uno *smartphone* invece che un PC per connettersi a internet sono molteplici e non riguardano solamente il livello di conoscenze informatiche degli individui o la loro passione per la tecnologia.
È soprattutto per gli utenti meno esperti che i benefici connessi all'arrivo delle nuove tecnologie mobili sono maggiori, poiché questi dispositivi risultano essere sempre più semplici e pratici da usare.
Ovviamente bisogna considerare come la situazione, anche se pressoché simile nelle varie parti del mondo, sia accentuata nelle economie emergenti dove i costi di accesso a un PC sono ancora proibitive rispetto a quelle degli *smartphone*, senza contare l'assenza delle infrastrutture di rete fissa, a tutto vantaggio di quelle mobile.
I Computer ovviamente offrono comunque maggiori servizi rispetto ai dispositivi *mobile*, quindi se è chiaro da un lato che il loro utilizzo potrebbe continuare a decrescere, è pur vero che essi non saranno sostituiti completamente da questi nuovi *devices* che per loro natura non sono adatti ad esempio alla programmazione o a un utilizzo più legato a un ambiente professionale.

2.3 Sicurezza informatica negli *Smartphone*

Senza dubbio gli *smartphone* rappresentano così uno degli oggetti che viene utilizzato da un numero di utenti sempre in maggiore crescita. Nel sotto-capitolo precedente si è accennato al fatto che tramite l'installazione di nuove applicazione acquisibili tramite gli *store* (che variano a seconda del sistema operativo dei *devices*) le funzionalità dei dispositivi stessi possono essere accresciuti. Proprio grazie a questi software aggiuntivi è possibile arricchire lo *smartphone* di nuove funzionalità, ma è pure vero che tramite *software* malevolo sono possibili alcuni attacchi informatici.
I primi attacchi agli *smartphone* si hanno nel 2004 con Cabir, che colpiva i sistemi Symbian. Data l'elevata crescita di questi dispositivi e del loro maggiore utilizzo si è registrato un aumento degli attacchi informatici atti a minare la sicurezza sugli *smartphone*, tramite appunto installazioni di applicazioni malevole, sia in modo diretto, sia sfruttando le vulnerabilità del software delle applicazioni popolari. Generalmente, in

un dispositivo di questo genere il rischio maggiore è il furto di dati riservati e personali, che possono essere ad esempio i contatti della propria rubrica fino ad esempio numeri di Carte di Credito utilizzate per pagamenti da *smartphon*e.
Per questi motivi la European Network and Information Security Agency (ENISA), un centro d'eccellenza per gli Stati membri dell'Unione europea e le istituzioni europee in sicurezza delle reti e dell'informazione, ha evidenziato delle linee guida sulla sicurezza e gestione delle applicazioni.

In alcuni ambiti, dove i livelli di sicurezza e segretezza sono particolarmente stringenti, esistono *smartphone* appositamente studiati o modificati, con limitazioni hardware e software, come l'impossibilità dell'accesso ai mercati delle applicazioni e l'assenza della fotocamera. Alcuni dispositivi o programmi aziendali permettono di modificare la modalità di funzionamento del dispositivo, dal più banale lavoro/casa a altre modalità, in modo da tenere i propri dati personali separati dai dati dell'azienda per cui si lavora, conferendo una tutela non solo al lavoratore, ma anche al datore di lavoro. Esistono anche agenzie che forniscono punteggi sulla sicurezza, controlli ed eventuali limitazioni sui dispositivi mobili in ambito professionale. Inoltre è stato dimostrato che anche con la semplice azione di ricarica tramite porta USB si può incorrere in problemi di sicurezza, tanto che il mercato ha risposto con un dispositivo da interporre durante la carica in modo da bloccare le transazioni informatiche durante la ricarica.

2.4 Mobile Financial

L'enorme sviluppo dei dispositivi mobili ha attratto l'attenzione anche degli istituti bancari o altre istituzioni finanziarie che cercano di offrire sempre un maggior numero di servizi ai proprio clienti come il Mobile Banking o ad esempio la possibilità di fare pagamenti per beni o servizi direttamente dal proprio dispositivo mobile.

2.4.1 Mobile Banking

Tramite il servizio di mobile banking i clienti possono così remotamente usare il proprio

dispositivo mobile per effettuare ad esempio transazioni finanziarie per mezzo di apposite App, fornite appunto dall'istituto finanziario stesso.
Generalmente il Mobile Banking è disponibile 24 ore su 24, permettendo agli utenti una esperienza mai avuta.
Ovviamente per ragioni di sicurezza alcuni istituti bancari danno delle limitazioni riguardo alcuni particolari tipi di transazioni che possono avvenire direttamente dai dispositivi *mobile*.
Le transazioni attraverso il mobile banking possono includere ad esempio la visualizzazione del proprio saldo con una lista delle ultime transazioni, pagamenti elettronici e trasferimenti di denaro tra un account e un altro o ancora la possibilità di effettuare un *download* e in seguito una stampa del proprio estratto conto senza doversi recare in filiale.
Dal punto di vista delle banche che offrono questi servizi, il mobile banking riduce i costi di gestione di tutte quelle transazioni non connesse a un prelievo o deposito di denaro per cui i clienti comunque devono recarsi presso gli ATM.
Una tra le funzionalità più recenti di questo tipo di applicazioni è la possibilità di trasmettere digitalmente assegni bancari utilizzando la fotocamera del proprio dispositivo. È bene evidenziare come il Mobile Banking differisca dal Mobile Payments o pagamenti da mobile, i quali invece coinvolgono il dispositivo mobile per pagare sia nei punti vendita o addirittura remotamente in modo analogo all'utilizzo di carta di debito o credito presso un EFTPOS.

2.4.2 Storia del Mobile Banking

Uno dei primissimi servizi di mobile banking non era connesso all'utilizzo di Internet, ma utilizzava la messaggistica basata su SMS, si parlava infatti di SMS banking. È ovvio considerare come i servizi forniti dal Banking basato su SMS, non è minimamente confrontabile con il Banking basato sull'utilizzo di Internet sia dal punto di vista dell'usabilità che per la mole di servizi offerti.
Con l'avvento dei primi *smartphone*, alla fine degli anni Novanta, una Banca norvegese, iniziò a offrire il servizio di Mobile Banking per i loro clienti. Prima del 2010

comunque la maggior parte degli istituti finanziari basava i propri servizi tramite l'utilizzo di SMS piuttosto che utilizzando il Mobile WEB. Solamente dopo il successo di Apple e dei primi *Smartphone* basati su Android vi fu un incremento nell' utilizzo di speciali applicazioni *mobile*. Uno studio del Maggio 2012 intrapreso da Mapa Research suggerì che oltre un terzo delle banche aveva sul proprio sito web la possibilità di individuare se la connessione avveniva da uno dispositivo mobile o meno e nel momento in cui veniva individuata una connessione mobile, poteva esserci la possibilità di ridirigere l'utente verso lo *store* per il *download* dell'applicazione mobile o verso uno specifico sito web strutturato appositamente per i dispositivi mobile. Questo studio mostrava come anche le banche hanno dovuto adeguarsi ai nuovi bisogni e comportamenti dei loro clienti.

2.4.3 Utilizzo del Mobile Banking nel mondo e confronto con Web Banking

Il mobile banking è un fenomeno che si è diffuso così come le nuove tecnologie in molti Paesi del mondo, anche tra quelli in via di sviluppo.

Un' importante ricerca a riguardo è stata condotta da Bain & Company e Research Now tramite diversi sondaggi.

Viene mostrata la percentuale di persone che durante tutto l'arco del 2012 non ha avuto alcuna transazione tramite mobile-banking basato su SMS nei precedenti tre mesi, per sottolineare il fatto che il campione preso in esame presumibilmente utilizzasse il servizio di Mobile Banking offerto dal proprio istituto bancario solo tramite l'applicazione preposta.

Rank	Country/Territory	Usage in 2012
1	South Korea	47%
2	China	42%
3	Hong Kong	41%
4	Singapore	38%
5	India	37%
6	Spain	34%
7	United States	32%
8	Mexico	30%
9	Australia	27%
10	France	26%

Figura 3: Statistica uso Mobile Banking 2012

Come si può evincere dalla figura vediamo che sono presenti tra le prime 10 nazioni col più alto tasso di utilizzatori del Mobile Banking, non basato su SMS, sia Paesi sviluppati come ad esempio Stati Uniti, Francia e Spagna che Paesi in via di sviluppo come ad esempio l'India, che pure essendo uno dei 4 Paesi più importanti del G20, non ha nel suo territorio una infrastruttura di telecomunicazione paragonabili ai principali Paesi Europei, soprattutto nelle sue aree rurali.

In Paesi come il Kenya, Cina, Brasile gli utilizzatori del servizio di Mobile Banking è aumentato di più del 100%. Spesso nei Paesi come quelli sopra citati, le banche sono

presenti solamente nelle grandi città. Questo può spiegare ad esempio l'enorme crescita nell'utilizzo del Mobile Banking da parte dei clienti di questi Paesi, e accresce inoltre l'importanza che il servizio stesso può avere per queste popolazioni. E' bene sottolineare come anche negli Stati Uniti durante l'arco del 2010 si sia registrato un aumento percentuale nel numero di utilizzatori del Mobile Banking di quasi il 100%. Ciò dimostra come i clienti preferiscano sempre di più utilizzare il proprio dispositivo per accedere al proprio profilo bancario non solo nel caso in cui il Mobile Banking sia l'unica soluzione possibile, ma anche quando nonostante la presenza di filiali essi la preferiscano per comodità e utilizzabilità.

Un' altra importante statistica condotta da CACI International Inc, mostra come è cambiato il modo degli utenti di interfacciarsi verso la propria banca. L' azienda con sede ad Arlington, Virginia ha mostrato nella propria ricerca come è cambiato il modo di interfacciarsi nel 2010 e in seguito nel 2015.

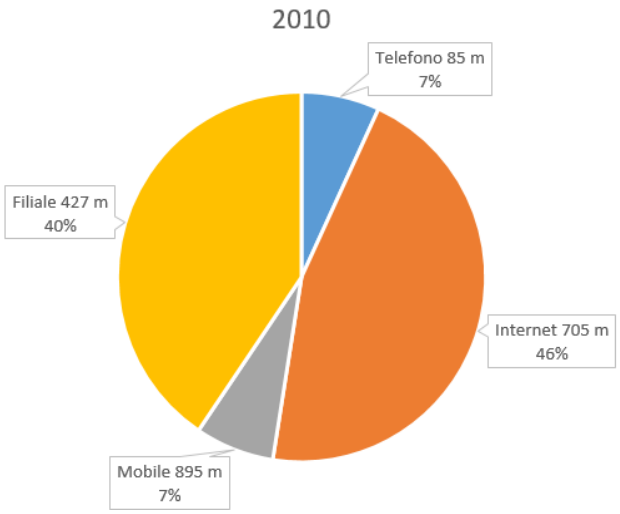

Figura 4: Ricerca CACI su utilizzo servizi di banca dai vari canali 2010

Dalla figura riguardo gli accessi del 2010, si può notare come quelli relativi al mobile siano solamente il 7% del totale ed essendo quindi irrisori rispetto a quelli tramite sito web, che rappresentano quasi il 46 %.

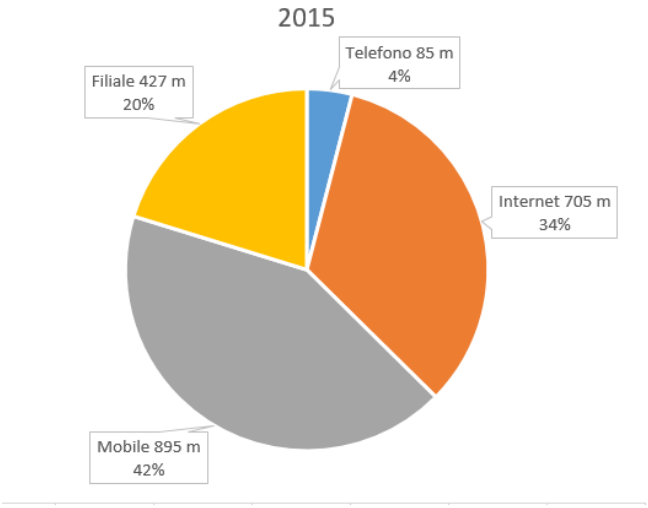

Figura 5:Ricerca CACI su utilizzo servizi di banca dai vari canali 2015

Dal grafico rappresentante i dati relativi al 2015, si nota immediatamente che la percentuale di accessi da mobile siano aumentati del 500% rispetto ai dati raccolti nel 2010. Dal confronto tra i due grafici si nota che gli accessi tramite Web siano leggermente decrementati, mentre il numero di utenti che accede recandosi in filiale è dimezzato. Rimangono invece abbastanza bassi gli accessi tramite chiamate telefoniche. Alcune previsioni relative a questa ricerca indicano che nel 2020 gli accessi tramite *mobile* saranno circa l'80% contro un 10 % di accessi tramite web-site e un 5 % di accessi in filiale. I dati relativi al 2015 fanno comunque riflettere su come si sia evoluta la situazione e quanto importante sia oggi il Mobile Banking.

2.4.4 Possibili evoluzioni del mobile banking

Come accennato nel sotto-capitolo precedente, si è passati dal mobile banking basato su Sms fino all'utilizzo di Internet, con un incremento sui servizi disponibili per i clienti.
L'avvento delle tecnologie *mobile* e l'incremento nell' utilizzo di *tablet* e *smartphone* permettono ai clienti di connettersi da qualsiasi posto e in qualsiasi momento.
Secondo uno studio fatto dal World Business Institute in Australia, alcuni possibili sviluppi del Mobile Banking potrebbero riguardare la connessione dei vari segmenti di

clienti, connettendo le varie generazioni tramite giochi o social network interni alle applicazioni, o ad esempio la personalizzazione dei servizi bancari in base alle proprie preferenze oltre ad esempio a interazioni via video con agenti e consulenti e la capacità di transazioni pervasive.Una considerazione particolare deve essere data nello sviluppo delle applicazioni bancarie alla sicurezza, poiché le informazioni che vengono trasmesse in queste particolari App sono ovviamente molto più confidenziali rispetto a quelle utilizzate dalle altre applicazioni, e per questo motivo nel momento in cui queste vengono inviate si fa ricorso a particolari metodi di *encryption* e *decryption.* Sempre riguardo ciò che concerne la sicurezza, spesso vengono utilizzati con queste applicazioni delle *smart-card* e *one-time password* generate da dispositivi fisici aggiuntivi, o alle volte tramite *push* diretto sull' App stessa da parte degli istituti finanziari, che permettono maggiore sicurezza nella fase di autenticazione, in modo che anche se il cliente dovesse subire il furto dello *smartphone*, non vi è alcun pericolo riguardo l'accesso da parte di utenti non autorizzati, che comunque non riuscirebbero ad utilizzare le funzionalità offerte dall'applicazione.

2.4.5 Mobile Payments

Come accennato in precedenza occorre che l'argomento del *mobile payments* sia distinto dai servizi che può offrire una banca tramite il proprio servizio di mobile banking.

Questi tipi di pagamento hanno la particolarità, come suggerisce il nome, che avvengano tramite dispositivi mobili. Gli utenti possono così pagare senza l'ausilio di denaro *cash,* con assegni o carte di credito, per una enorme quantità di servizi e beni digitali o fisici. Anche se ovviamente il concetto di non usare moneta per portare a termine degli acquisti ha una lunga storia, è solo di recente che gli sviluppi tecnologici hanno fatto sì che questo sistema si sia diffuso e sia diventato disponibile quasi ovunque. Nei Paesi in via di sviluppo le soluzioni di Mobile Payments hanno preso piede come un'estensione dei servizi finanziari che si potevano ottenere con il Mobile Banking. Spesso questi tipi di pagamenti, vengono utilizzati sempre di più per micro-pagamenti.

2.4.6 Principali modelli di Mobile Payments

Le principali metodologie di pagamento differiscono per la tecnologia su cui esse si basano, velocità e per livello di affidabilità.

Alcuni esempi di suddette tecnologie possono essere ad esempio basati su *Mobile wallets*, pagamenti basati su carta, Carrier billing che può essere basato su SMS, pagamenti *Contactless NFC* (Near Field Communication o letteralmente Comunicazione in prossimità) oppure trasferimenti diretti di denaro tra compratori e beneficiari.

I pagamenti tramite carta di credito sono tra i più semplici *mobile payments,* poiché per completare un acquisto l'utente dovrà solamente inserire dettagli relativi alla propria carta.

Il *Mobile Wallet* è molto diffuso e utilizzato ad esempio da PayPal, AmazonPayments e Google Wallet. Per quanto riguarda l'utilizzo esso differisce se l'utente sta effettuando il primo pagamento o i successivi. L'utente deve registrare inizialmente il proprio numero di telefono e tramite il provider invia un SMS con un PIN che verrà utilizzato per autenticare il numero registrato. In seguito l'utente registra la propria carta e può effettuare il primo pagamento. Per i pagamenti successivi invece occorrerà solamente il PIN per validare l'acquisto.

Il *Carrier Billing*, o fatturazione diretta con l'operatore mobile, è la tecnologia che consente agli utenti mobile di effettuare pagamenti attraverso il loro credito telefonico mensile o il credito prepagato utilizzabile anche per i cellulari considerati non *'smart'*. Questo metodo di pagamento, fornisce un collegamento diretto tra le terze parti e il sistema di pagamento di un operatore. Inoltre, il *Carrier Billing* permette un'ottima *user experience*, è flessibile, affidabile e immediato, con la possibilità di avere delle statistiche in tempo reale sulle transazioni.

I pagamenti *contactless* prevedono solitamente l'utilizzo di una Carta Fisica o di un dispositivo mobile che possa essere utilizzato in sostituzione di quest'ultima. Per quanto riguarda i pagamenti tramite trasferimento diretto di denaro può avvenire in diversi modi e consiste nel passaggio di denaro da un *bank-account* a un altro.

2.5 Soluzioni innovative per pagamenti da mobile

In questo sotto-capitolo verranno mostrati due dei metodi di pagamento più innovativi e utilizzati al momento, evidenziandone le differenze. Molte delle banche Italiane e Europee tramite le loro App permettono di fare pagamenti tramite questi servizi.

2.5.1 Satispay

Satispay è un sistema di pagamento indipendente dai circuiti tradizionali, che permette di scambiare denaro con gli amici e di pagare nei negozi convenzionati, fisici e online, tramite un'applicazione disponibile per iPhone, Android e Windows Phone, sviluppata da una start-up italiana capace di attrarre oltre cinque milioni di euro da investitori internazionali. Il servizio, semplice e sicuro, è sempre gratuito per gli utenti privati ed estremamente conveniente per gli esercenti. Come per il servizio mostrato nel capitoletto precedente, Satispay può essere utilizzata da chiunque sia maggiorenne e titolare di un conto corrente bancario o di alcune carte prepagate con codice IBAN. La logica è esattamente quella di un'applicazione di messaggistica ma il dialogo con la controparte è fatto di euro.

Figura 6: SatisPay *screenshots* pagamento tra privati

Figura 7: SatisPay *screenshots* pagamento verso esercenti

Come mostrato nelle figure precedenti, il passaggio di denaro può essere effettuato sia con i propri amici, quanto con negozi convenzionati. Con questi ultimi, quindi, può avvenire un vero e proprio pagamento in formato digitale, il tutto senza alcun costo a carico dell'utente. Ogni somma al di sopra dei 10 euro ha invece un costo fisso pari ad appena 20 centesimi per il venditore. In fase di iscrizione occorre collegare un proprio conto bancario che sarà la fonte del Budget mensile disponibile per i pagamenti tramite Satispay. Ogni utente può mettere a disposizione del proprio account un certo massimale mensile, che può comunque essere modificato in qualsiasi momento. Affinché il pagamento sia effettuato occorre solamente una connessione e l'immissione di una password. A quel punto il denaro verrà trasferito dall' *account* Satispay del pagante all' *account* del beneficiario. Ogni mese la somma che è stata indicata come massimale di spesa sarà trasferita dal proprio conto bancario al *account* Satispay. Se la somma presente sull' *account* Satispay non verrà utilizzata interamente il mese seguente dal conto bancario verrà ricaricata solamente la porzione di denaro tale da permettere al *budget* di essere uguale al massimale deciso in fase di iscrizione. In questo caso se durante un mese non vengono fatti acquisti dal proprio conto Satispay, nel mese seguente non vi sarà nessun passaggio di denaro dal conto bancario verso l'*account* Satispay.

2.5.2 JiffyPay

JiffyPay può essere considerata come la contromossa a Satispay attuata dagli istituti bancari. Sviluppata da Sia, leader europeo che gestisce i servizi finanziari di alcuni istituti finanziari, Jiffypay è un nuovo ed interessante servizio pensato per trasferire denaro tramite *smartphone* in tempo reale. Il servizio permette agli utenti di fare pagamenti in maniera molto semplice ed immediata. Una banca che fa da intermediario permetterà la riuscita del pagamento.

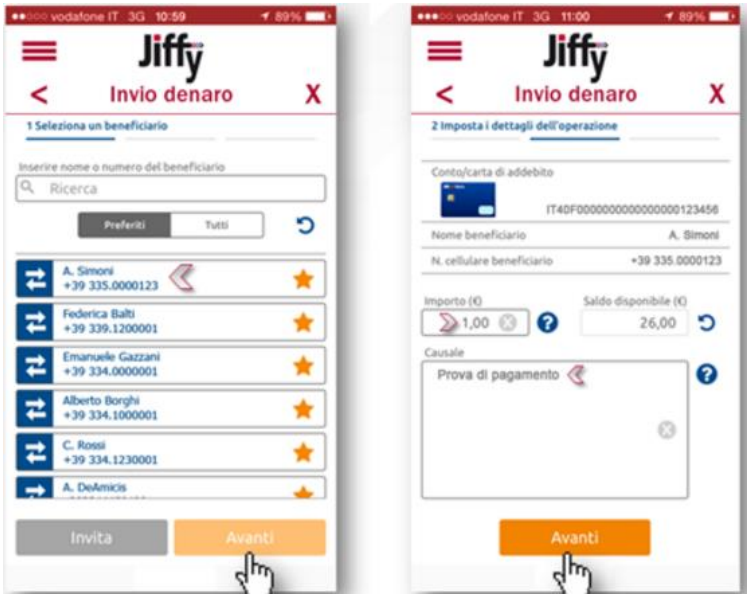

Figura 8: JiffyPay *screenshots* pagamento

Il servizio, ad oggi presente sia su iOS che Android, si concretizza attraverso un'App dedicata in cui sarà sufficiente inserire l'importo da inviare ed il numero di cellulare del beneficiario dell'importo stesso. L'accredito avverrà così in tempo reale direttamente sul conto associato alle coordinate bancarie legate al numero di cellulare della persona interessata. L'IBAN da utilizzare potrà essere, oltre quello classico di conto corrente anche quello associato a una carta prepagata. La più grande differenza con Satispay sta nel fatto che condizione fondamentale per l'utilizzo di JiffyPay, è l'essere titolare di un conto corrente in una delle banche aderenti al progetto. Registrandosi quindi tramite

l'*home banking* e fornendo il proprio numero di telefono si può procedere a effettuare pagamenti tramite JiffyPay. Il numero di Banche che in Italia utilizzano questo servizio di pagamento sono in continua crescita.

CAPITOLO 3

PERSONAL FINANCIAL MANAGEMENT

In questo capitolo verrà rappresentato cosa è il Personal Financial Management o gestione delle finanze personali, descrivendo una breve storia dalle origini dei primi software implementati per rendicontare le spese personali. Nella parte finale del capitolo verranno poi presentare le caratteristiche comuni delle soluzioni presenti ai giorni nostri, le quali presentano maggiori funzionalità, come la possibilità di programmare spese future. Una più attenta analisi sulle soluzioni presenti sul mercato verrà affrontata al capitolo 4.

3.1 PFM o Gestione delle finanze personali

Il Personal Finance Management o gestione delle finanze personali, è la gestione del risorse finanziarie che un individuo o un membro di una famiglia svolge per conservare e/o bilanciare nei vari settori le risorse finanziarie nel tempo, prendendo in considerazione possibili rischi finanziari ed eventi futuri. Quando si fa un piano per la gestione delle proprie finanze, l'individuo dovrebbe considerare sia i propri bisogni sia le proprie spese o i propri investimenti privati in modo equo. Fin dai primi anni del Novecento molte discipline furono altamente correlate al PFM come ad esempio le scienze economiche relative alle spese familiari o alle spese dei consumatori, le quali in molti *college* erano argomenti che facevano parte dell'economia domestica da oltre 100 anni. La primissima ricerca fatta sul *Personal Financial Management* fu fatta nel 1920 da Hazel Kyrk. La sua tesi di laurea presentata all' Università di Chicago pose le

fondamenta e sottolineò l'importanza delle scienze economiche familiari e sulle scienze economiche dei consumatori. Margaret Reid, docente di Economia Domestica nella stessa Università è riconosciuta come uno dei pionieri nello studio del comportamento dei consumatori e dei componenti. Nel 1947, Herbert A. Simon, premio Nobel, dimostrò che un decisore non fa sempre la migliore scelta dal punto di vista finanziario, spesso a causa della sua limitata istruzione formativa e/o a causa di determinate inclinazioni personali. Nel 2009, Dan Ariely ha dimostrato come la crisi finanziaria del 2008 ha reso evidente il fatto che le persone non prendono sempre la decisione finanziaria più razionale, e che il mercato non è necessariamente autoregolamentato e correttivo di ogni squilibrio economico, fattore che alla lunga può causare crisi economiche. Quindi, l'istruzione delle finanze personale è necessaria ad un individuo e ad una famiglia al fine di capire come occorre prendere le decisioni finanziarie migliori dal punto di vista razionale durante tutta la loro vita. Prima del 1990, la maggioranza degli economisti e le Facoltà di Economia non davano molta attenzione alla gestione delle finanze personali. Bisogna sottolineare comunque che negli ultimi 30 anni diverse università Americane come la *Brigham Young University, Iowa State University,* e la *San Francisco State University* hanno offerto diversi corsi riguardo la gestione delle finanze personali. Questi istituti hanno pubblicato diverse ricerche da loro svolte su molte riviste come sul *The Journal of Financial Counseling and Planning* e sul *Journal of Personal Finance*. Enti professionali come la *American Association of Family and Consumer Sciences* e la *American Council on Consumer Interests* iniziarono ad avere un ruolo molto importante nello sviluppo di questo campo tra gli anni Cinquanta e Settanta del Novecento. La fondazione dell'AFCPE *(Association for Financial Counseling and Planning Education*) nel 1984 allo *Iowa State University* e l'AFS *(Academy of Financial Services)* nel 1985 hanno segnato un' importante pietra miliare nella storia del PFM. Conseguenza di ciò è l'incremento delle capacità finanziarie dei consumatori negli ultimi anni, grazie ad un crescente incremento nel numero di corsi disponibili e seguiti in gran numero specialmente da giovani e donne. Questi tipi di corsi formativi sono frequentemente chiamati come alfabetismo finanziario. Dati i numerosi corsi, che sempre di più vengono proposti al pubblico e l'enorme importanza che l'argomento PFM ha riscosso negli ultimi anni,ci si è spinti per sviluppare uno

standard nel campo della formazioni finanziaria.

3.2 Processo di pianificazione del PFM

La componente chiave del Personal Financial Management è la pianificazione finanziaria. Questa fase consiste in un processo dinamico che richiede monitoraggio continuo e eventuali rivalutazioni. Generalmente coinvolge 5 punti differenti:

1. **Valutazione**: La situazione finanziaria di una persona è valutata compilando una versione semplificata di un bilancio di esercizio che include lo stato patrimoniale e il conto economico. Si ha in questo modo una lista che mostra attivi come l'auto, la casa, e i propri conti bancari e passivi come carte di debito, prestiti bancari, mutui. In aggiunta viene inserita una lista di tutte le proprie entrate e le proprie uscite.
2. **Indicazione degli obiettivi**: Avere diversi obiettivi è un fatto abbastanza comune. Gli obiettivi spesso vengono suddivisi, in questa fase, tra obiettivi a lungo termine e obiettivi a breve termine. Impostare gli obiettivi finanziari aiuta a dirigere la pianificazione stessa e viene predisposta principalmente per soddisfare particolari fabbisogni finanziari
3. **Creazione del Programma:** Il programma finanziario indica come raggiungere gli obiettivi prefissati. Potrebbe includere, per esempio, la riduzione delle spese non necessarie, incrementando le entrate, o investendo in borsa.
4. **Esecuzione:** L' Esecuzione del programma spesso richiede disciplina e perseveranza. Molte persone in questa fase ricevono assistenza da professionisti come ad esempio ragionieri, consulenti finanziari e avvocati.
5. **Monitoraggio e rivalutazione**: Col passare del tempo tutto il piano viene monitorato e sono possibili delle variazioni.

Ciò che sta alla base della pianificazione della gestione delle proprie finanze è l'evitare di assumere comportamenti poco corretti. La gente dovrebbe schivarel'attaccamento al denaro, moralmente reprensibile. Per quanto concerne gli investimenti si dovrebbero evitare quelli più rischiosi che promettono ritorni troppo elevati.

3.2.1 Principali aree di interesse durante il processo di pianificazione

Durante tutto il processo di pianificazione viene data maggiore importanza ad alcune aree di interesse. Come suggerito dal *Financial Planning Standards Board* le principali sono:

1. **Posizione finanziaria**: essa è correlata con il prendere nota di quali siano le risorse personali disponibili esaminando il patrimonio netto e il flusso di denaro familiare. Il patrimonio netto può essere visto come la somma del valore di tutti i beni sotto il proprio controllo, meno tutte le passività correlati ai beni stessi. Per quanto riguarda il flusso di denaro personale vengono considerate tutte le forme di guadagno meno tutte le spese attese durante un periodo non molto breve, come ad esempio un anno. Da questa prima analisi, il pianificatore può determinare su cosa essere d'accordo e in quanto tempo gli obiettivi possono essere raggiunti.
2. **Protezione adeguata**: rappresenta l'analisi di come proteggersi da rischi imprevisti. Questi rischi possono essere di diverso tipo, come ad esempio riguardo a debiti, proprietà, morte, disabilità e salute. Alcuni di questi rischi richiedono la sottoscrizione di un assicurazione. La determinazione di quanto assicurare richiede conoscenza del mercato assicurativo. Proprietari di attività, professionisti, atleti e artisti richiedono sempre più spesso particolari assicurazioni. Utilizzando investimenti assicurativi si potrebbe produrre una parte abbastanza grande del totale degli investimenti.
3. **Pianificazione delle tasse**: nel bilancio personale come è noto, le tasse rappresentano la maggior parte delle uscite. Gestire le tasse non è una questione correlata al fatto se le tasse verranno pagate o meno, ma riguarda solo il momento in cui ciò avverrà e l'importo delle stesse. I governi danno diversi incentivi sotto forma di detrazioni e/o crediti, che possono essere utilizzati per alleggerire il carico totale delle tasse. Molti governi usano un sistema di tasse progressive. Tipicamente, man mano che crescono i redditi crescono anche le

tasse a essi correlati. Capire come ottenere vantaggi dalla miriade di agevolazioni fiscali quando si è in fase di pianificazione, può avere un impatto davvero significativo.

4. **Obiettivi di investimento e accumulo**: decidere in che modo accumulare abbastanza denaro e per quali scopi, può essere considerata una parte abbastanza delicata durante il processo di pianificazione. Spesso si tende ad accumulare denaro per grandi acquisti, come ad esempio l'acquisto di un immobile, l'inizio di un'attività di business o il pagamento di tasse per gli studi, o ancora l'organizzazione di eventi molto importanti come potrebbe essere un matrimonio. Spesso viene accumulato denaro, come risparmi da utilizzare nel periodo post-pensionamento. Per raggiungere questi obiettivi occorre proiettare il costo del bene nel momento che il bene stesso sarà acquistato e capire quando occorrerà prelevare questa porzione del fondo. Il rischio maggiore nel raggiungere questi obiettivi di accumulo è il tasso di incremento dei prezzi nel tempo, o l'inflazione. Usando software che calcolano il prezzo dei beni nel futuro, viene suggerita una combinazione di patrimonio che dovrà essere aggiunto ai risparmi che vengono fatti regolarmente e un'altra porzione da investire in una varietà di investimenti con una determinata rendita. Per superare il tasso di inflazione, il totale degli investimenti dovrà avere un grande tasso di rendita, che ovviamente potrebbe portare ad avere rischi anche eccessivi. Per gestire questi rischi spesso si investono porzioni del patrimonio in diversi ambiti, affinché si vadano a diversificare i rischi stessi. Ogni porzione di patrimonio sarà utilizzata per investimenti in azioni, obbligazioni o altri tipi di investimento. La porzione del patrimonio che viene utilizzata nei diversi settori di investimento può variare a seconda del profilo finanziario di ogni investitore e la propria attitudine al rischio.

5. **Piano di pensionamento**: è il processo correlato alla comprensione di quanto potrebbe costare la vita nel momento del ritiro dal lavoro. Occorre cosi avere un piano per distribuire i propri beni e andare incontro alla mancanza o diminuzione del reddito entrante.

6. **Piano di eredità:** esso coinvolge la pianificazione per la disposizione dei propri beni dopo la morte. In alcuni governi c'è una tassa dovuta allo stato nel momento in cui si muore. Evitare queste tasse significa che uno o più dei propri beni siano distribuiti ai propri eredi. Si può così decidere di lasciare dei beni alla propria famiglia, ai propri amici oppure in beneficenza.

3.3 Storia dei primi software PFM

L' enorme quantità di applicazioni *mobile* e *servizi web* che sono utilizzati ogni giorno, permette agli *users* di gestire in modo più semplificato varie situazioni e una quantità di dati enorme se confrontati col passato. Lo stesso vale per la gestione delle finanze personali. Gestire il proprio budget andando a monitorare le proprie spese permette agli utenti di tenere più facilmente sotto controllo le proprie spese e in pochi click riuscire a selezionarle in base alle categorie. Questi software possono essere davvero importanti per gli utenti e possono essere di enorme aiuto per risparmiare denaro e raggiungere specifici obiettivi. Senza l'utilizzo di questi software è davvero difficile tenere traccia di tutti i propri acquisti. Per quanto riguarda la categorizzazione, e il filtraggio delle spese fatte ad esempio in un determinato settore, senza l'utilizzo di questi servizi digitali, diventa quasi impossibile per l'utente. Dal punto di vista dell'utente, tenere traccia delle proprie spese è un atto davvero importante per tenere sotto controllo il proprio *budget*, ma allo stesso tempo è impensabile ai giorni nostri che l'utente stesso tramite carta e penna vada a rendicontare tutte le proprie finanze. Confrontando i due approcci, con e senza l'ausilio di determinati software, non vi è partita in termini di comodità e semplicità di utilizzo. Inoltre, tramite l'utilizzo di software appositi, spesso i pagamenti che avvengono in maniera digitale possono essere automaticamente inseriti nelle App, mentre per quanto riguarda altri tipi di pagamenti vi è la possibilità dell'aggiunzione manuale. Sempre di più ai giorni nostri, come spiegato nel secondo capitolo, per un utente è molto semplice anche in mobilità accedere alle applicazioni presenti sul proprio

smartphone, e visualizzare, modificare o aggiungere una spesa fatta. Questo bisogno per gli utenti ha fatto sì che dagli anni Ottanta ad oggi si sono susseguiti diversi software appositi per la gestione delle finanze personali.

I primi software che permettevano agli utenti di categorizzare le transazioni e aggiungere degli account da diversi istituti bancari dentro una singola veduta, iniziarono ad essere implementati nei primi anni Ottanta. Questi software possono essere considerati come i precursori delle moderne applicazioni che si utilizzano giornalmente tramite *smartphone* ad esempio. Il primo esempio di Personal Financial Management è apparso precisamente nel 1983 con la fondazione di Intuit.

Scott Cook e Tom Proulx, i fondatori dell'azienda, visto l'enorme incremento nello utilizzo del PC durante quegli anni da utenti sempre più eterogenei tra loro, fiutarono l'opportunità di sviluppare del software per la gestione delle finanze personali.

Figura 9: Quicken 8 per DOS

Il loro prodotto, Quicken, di cui è mostrata una schermata nella figura precedente, divenne uno *standart* per molti proprietari di immobili che tramite esso potevano facilmente gestire il denaro proveniente dagli affitti e/o dalle vendite.

Visto l'utilizzo sempre crescente di Quicken, un altro software che fu sviluppato in seguito fu QuickBooks, che aiutava gli imprenditori di piccole o medie imprese a gestire le proprie finanze. Dopo si susseguirono altri software come MYM per Apple e IBM-PC e in seguito nel 1990 Microsoft rilasciò la loro piattaforma per il PFM chiamata Microsoft Money.

Microsoft nel 1997 lavorò con Intuit e CheckFree per creare Opena Financial Exchange(OFX), software che permette alle istituzioni finanziarie di scambiare dati con i propri clienti tramite il web, aprendo la strada per lo sviluppo di software PFM on-line.

Figura 10: Yodlee screenshot versione 2016

Nel 1999 fu fondato Yodlee, originariamente un sito web che permetteva agli utenti di vedere email, *news*, spese personali tutto tramite un'unica piattaforma. Anche eWise che fu fondato nel 2000 come soluzione *open source* per la gestione delle proprie finanze, inventò l'aggregazione lato *client* dei dati, permettendo in modo significativo

alle informazioni finanziarie di essere aggregate su un server finale o su un dispositivo capace di memorizzarle scelto dall' utente come ad esempio uno smartphone, un *tablet* o un personal computer. Ovviamente una soluzione di questo tipo risulta essere più sicura rispetto ad una basata su *cloud* o un server esterno. Oggi molte aziende finanziarie di consulenza come LearnVest, Personal Capital e Credit Karma, integrano il PFM dentro i loro software così come diverse banche tramite la propria Applicazione mobile.

3.4 Caratteristiche generali delle soluzioni PFM

Diverse aziende negli anni hanno sviluppato software per PFM e tra essi vi sono molte caratteristiche comuni. Ogni software connesso alla gestione delle finanze permette agli utenti di aggregare le proprie transazioni finanziarie.

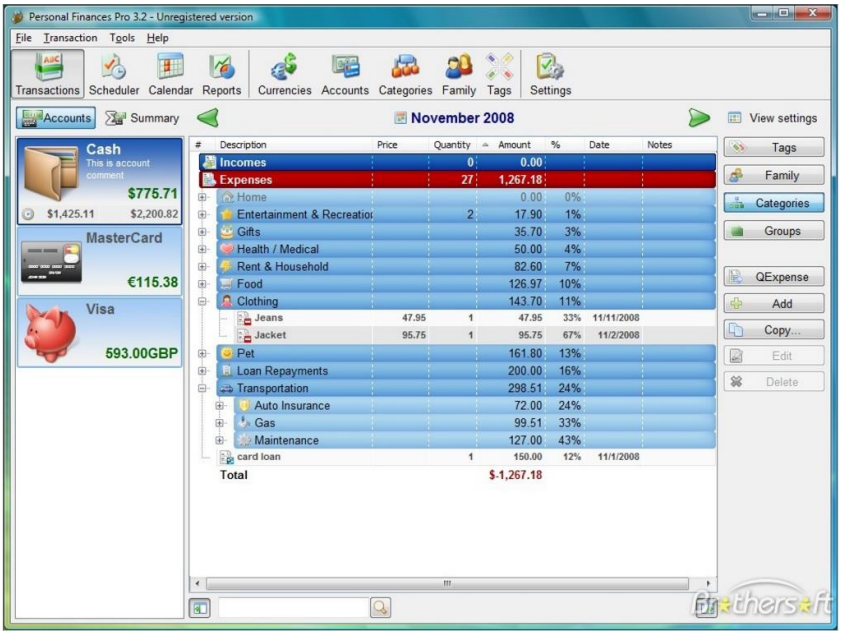

Figura 11: Esempio Software PFM

In alcuni casi, queste transazioni devono essere inseriti in modo manuale ma una percentuale crescente di software presenti sul mercato automatizza questo processo. Questi software, come ad esempio Personal Finances Pro 3.2, mostrato nella figura precedente, mostrano il flusso di denaro, il trend delle spese e la loro categorizzazione.
Inoltre l'utente può definire degli obiettivi, indicando la quantità di denaro da investire

su una certa spesa. In alcuni di essi vi è addirittura la possibilità di gestire debiti e spese future, avendo ad esempio un salvadanaio virtuale, dove si possono annotare i propri risparmi per il raggiungimento di un eventuale obiettivo.

Molti software presenti sul mercato, cercano di mostrare i dati all'utente in modo sempre più chiaro e immediato. Sempre più spesso essi fanno ricorso a grafici a torta e istogrammi per dividere le varie spese per categoria e fornire confronti tra periodi differenti. Un esempio è mostrato nella figura sottostante.

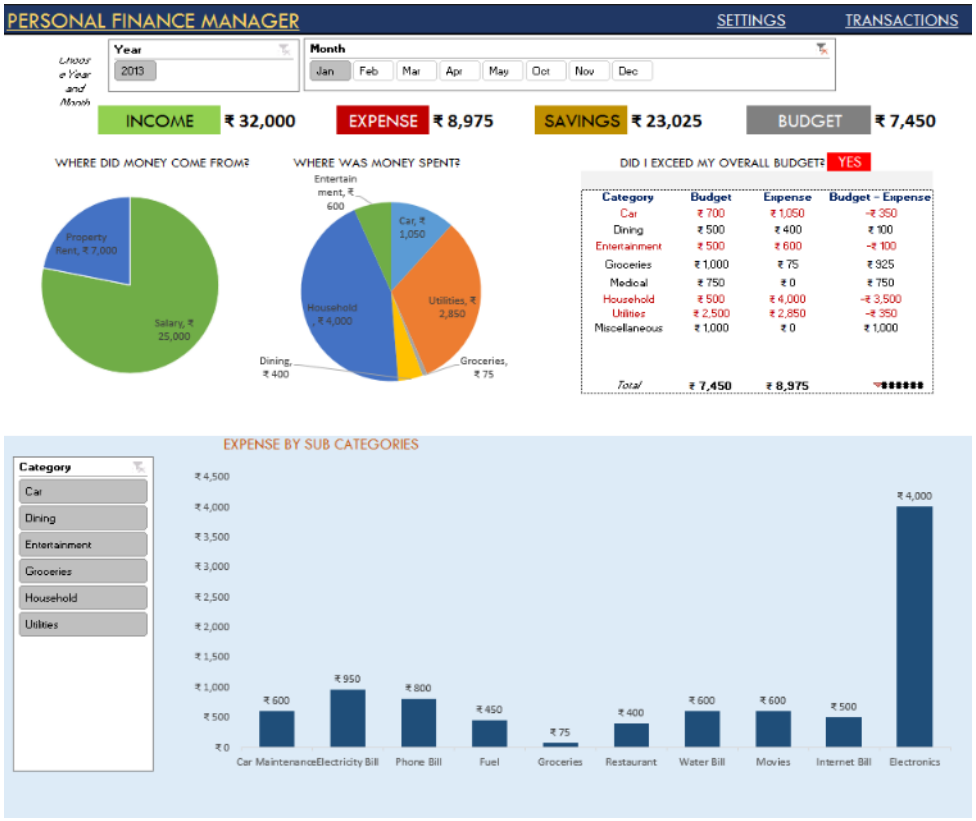

Figura 12: Esempio 2 Software PFM

3.5 Evoluzioni delle soluzioni PFM

Il continuo sviluppo di questo tipo di applicazioni, e il consequenziale aumento sul mercato di questi prodotti, ha fatto sì che alcune delle soluzioni più recenti al fine di distinguersi maggiormente da quelle già esistenti, riescono ad offrire sempre maggiori

funzionalità. Molte di queste soluzioni permettono di gestire non solo le spese fatte a livello personale ma anche quelle fatte all'interno di un certo gruppo di spesa. Alcuni esempi più comuni di gruppi di spesa possono essere ad esempio una famiglia, un gruppo di colleghi o semplicemente un gruppo di amici. Tramite queste applicazioni possono essere rendicontate e consultate non solo le proprie spese, ma anche tutte quelle fatte dai vari membri all'interno del gruppo. Solitamente ognuno di essi può avere accesso alle info relative al gruppo e gestire le spese personali e anche quelle fatte all'interno dei vari gruppi.

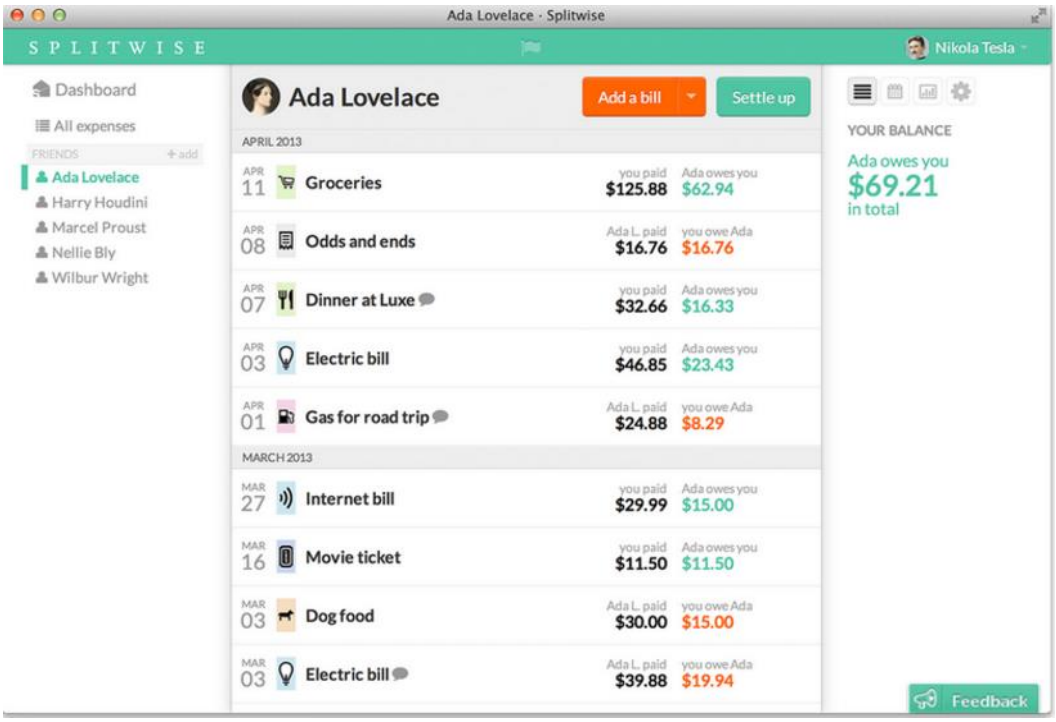

Figura 13: SplitWise screenshot

Una di queste applicazioni può essere ad esempio SplitWise, che permette la categorizzazione delle varie spese fatte in un gruppo tra i membri del gruppi stesso.
Come mostrato nella figura precedente, Splitwise, permette all'utente di avere un rapido resoconto di tutte le spese, evidenziando debiti e crediti che rispettivamente si possono

avere per spese alle quali ancora bisogna contribuire con la propria quota e altre in cui gli altri membri del gruppo devono contribuire per una spesa pagata anticipatamente dall'utente stesso, andando infine a dare l'informazione sul saldo generale dell'utente.

CAPITOLO 4

ANALISI SOLUZIONI PFM ESISTENTI

In questo capitolo verranno mostrate più nel dettaglio alcune delle migliori soluzioni ,presenti oggi sul mercato delle applicazioni mobile, per la gestione delle finanze personali.

4.1 Oval Money

Oval Money è l'App mobile dell'omonima società fondata a Londra nel 2015 da Claudio Bedino, Edoardo Benedetto e Benedetta Arese Lucini, ex general manager in Italia dell'azienda californiana Uber. L'idea dei fondatori che sta alla base dell'applicazione, vincitrice degli Italian Fintech Awards nel 2016, è quella di aiutare i consumatori con degli "Step", ovvero impostazioni tali che in automatico calcolano piccoli importi da accumulare. Per esempio, l'App potrà calcolare piccoli importi da accumulare dopo ogni spesa effettuata, come l'arrotondamento all'euro più vicino. Una piccola spesa di 7,40 € permetterà all'utente di accantonare 0,60 €.

Figura 14: Oval screenshot 1

L'App permetterà cosi a tutti di accumulare del capitale in poco tempo e in modo automatico. Dalla figura si evince facilmente in che modo l'App avvisi ad ogni acquisto l'utente della porzione di capitale raccolta. Con la sua tecnologia, Oval Money traccia ogni transazione effettuata tramite conto e carta, senza bisogno di inserirle manualmente, e fornisce un resoconto chiaro e personalizzato delle abitudini di spesa. Con una consapevolezza maggiore delle proprie finanze gli utenti possono accumulare dei risparmi che vengono spostati tramite Oval in un salvadanaio digitale garantito e personale. Sempre più utilizzata in particolare dai più giovani, che spesso non si sentono rappresentati dai servizi bancari, permette loro oltre ad accumulare, anche di investire in modo semplice e continuo. Alla base del progetto vi è il concetto di economia collaborativa, come lo scambio e la fiducia. Simone Marzola, ingegnere ed esperto di

machine learning, con le sue competenze ha sviluppato un sistema di intelligenza collettiva che alimenta l'applicazione, grazie alla combinazione dei meccanismi presenti nelle community e all'analisi sofisticata dell'intelligenza artificiale. D'altro canto oltre il 75% degli utilizzatori ha evidenziato che essi si affidino all'influenza positiva della propria community, prima di prendere una decisione. Per questo motivo Oval unisce le tecnologie più complesse del *machine learning* alle raccomandazioni della community di utenti che fanno parte dell'App per portare un nuovo tipo di intelligenza collettiva all'interno del settore finanziario.

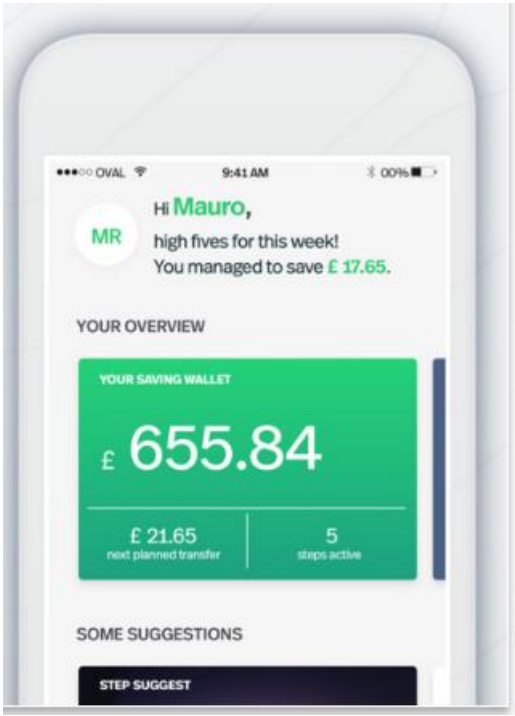

Figura 15: Oval screenshot 2

Come mostrato nella figura precedente si nota in che modo vengano indicati il totale dei risparmi, i prossimi *step* e i prossimi trasferimenti pianificati nel salvadanaio digitale. L'utente può facilmente conoscere la propria situazione e i prossimi obiettivi da raggiungere. Inoltre, come mostrato in precedenza tutte le operazioni registrate vengono anche categorizzate facilmente, permettendo così agli utenti di poter avere una panoramica sui vari acquisti fatti in vari periodi di tempo.

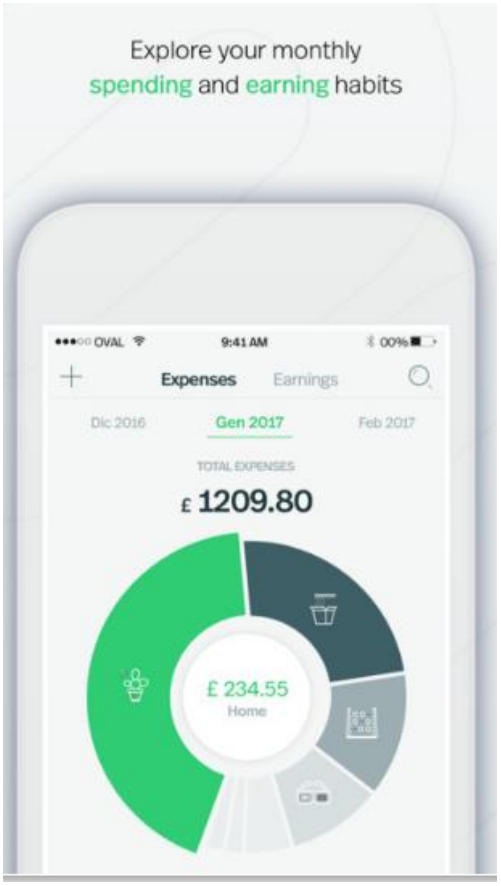

Figura 16: Oval screenshot 3

Se schermate del genere possono essere davvero comuni in molte delle applicazioni che si occupano di Personal Financial Management, esse sono indispensabili poiché permettono un'adeguata *User Experience* e immediatezza, tanto care agli utenti, senza le quali lo sviluppo e l'incremento di utilizzatori dell'applicazione stessa diventa quasi impossibile. Al contrario, una funzionalità abbastanza fuori dal comune che questa app fornisce ai propri utenti è il modo in cui la cronologia delle transazioni, fatte nel passato dall'utente, sono mostrate. Oltre a un elenco di transazioni, infatti, vi sono dei consigli basati proprio sulla cronologia delle transazioni stesse. La figura mostra un consiglio da parte dell'applicazione riguardo le spese fatte nella categoria viaggi dall'utente negli

ultimi sei mesi. Continuando a spendere nello stesso settore con la stessa frequenza del periodo indicato potrebbe risparmiare 12 Sterline alla settimana. La conoscenza delle abitudini finanziarie dell'utente rappresenta il vantaggio di indicare all'utente stesso quale sia il migliore modo in cui continuare a spendere il proprio patrimonio e continuare così ad accumulare risparmi.

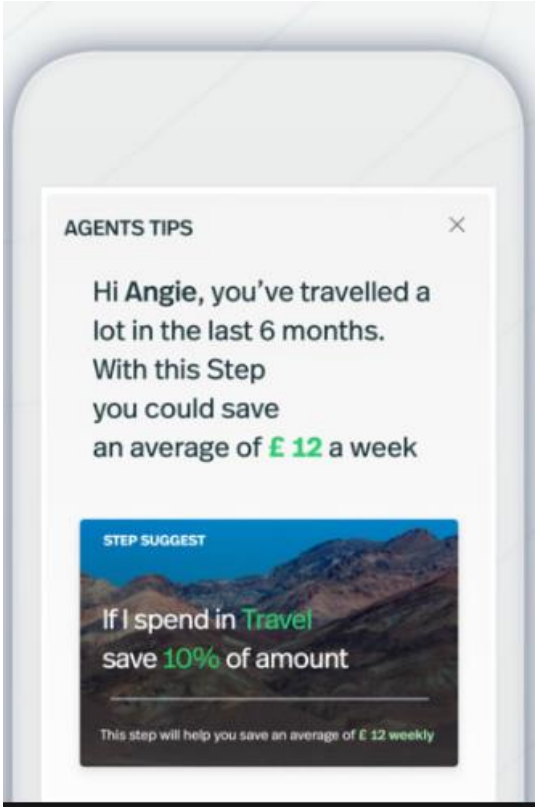

Figura 17: Oval screenshot 4

4.2 N26

N26 è un'app per iPhone o Android che permette di effettuare diverse operazioni bancarie e gestire in tempo reale le proprie finanze. N26 è da considerare come la più giovane banca "mobile-first" con sede a Berlino. Dall'inizio del 2017 anche per i clienti italiani è possibile aprire un conto. Il tempo medio stimato per la procedura di apertura di un nuovo conto è stato stimato essere soltanto 8 minuti per ricevere un IBAN, attivo da subito, più un paio di giorni per la spedizione a casa della carta MasterCard associata. La verifica dell'identità del cliente avviene tramite videochiamata, anche da smartphone. Occorre soltanto avere a portata di mano una carta d'identità oppure un passaporto valido. N26 ha ricevuto l'autorizzazione bancaria europea nel luglio del 2016 ed è già pienamente operativa. In Europa i clienti sono già più di 300.000 e l'azienda prospetta un'ulteriore espansione nel corso del 2017, con un conseguente allargamento del team centrale di Berlino. Dai prodotti analoghi presenti anche nel nostro Paese, (quasi tutti afferenti a istituti di credito più noti, che tuttavia tendono a non associare il marchio storico al nome del servizio) N26 si differenzia per due aspetti principali e correlati: l'indipendenza dai grandi gruppi del settore e l'effettiva assenza di filiali. Esso è un servizio indipendente, grazie all'autorizzazione della BCE ottenuta a luglio 2016. L'apertura di un conto base è gratuita e l'interfaccia di gestione permette di accedere a varie tipologie di servizi bancari digitali: da *Transferwise*, per bonifici internazionali in 19 valute a costo zero, a *Moneybeam*, che permette di inviare e richiedere denaro tramite SMS ai contatti della rubrica. Al conto, che non prevede possibilità di scoperto, si associa una MasterCard da usare come carta di debito in modo che l'addebito avvenga al momento del singolo pagamento e non in un'unica soluzione a fine mese.

Anche i clienti italiani avranno la possibilità di aprire il conto "premium" N26 Black con un costo mensile di 5,90 €, che ha come principale vantaggio l'associazione del conto ad una polizza assicurativa offerta da Allianz. L'offerta comprende un'assicurazione di viaggio e sanitaria con rimborso delle spese mediche, una sul furto dello smartphone, un'estensione di garanzia su prodotti assicurabili acquistati con la carta MasterCard associata al conto e il rimborso del contante rubato fino a 4 ore dal prelievo in Italia o all'estero.

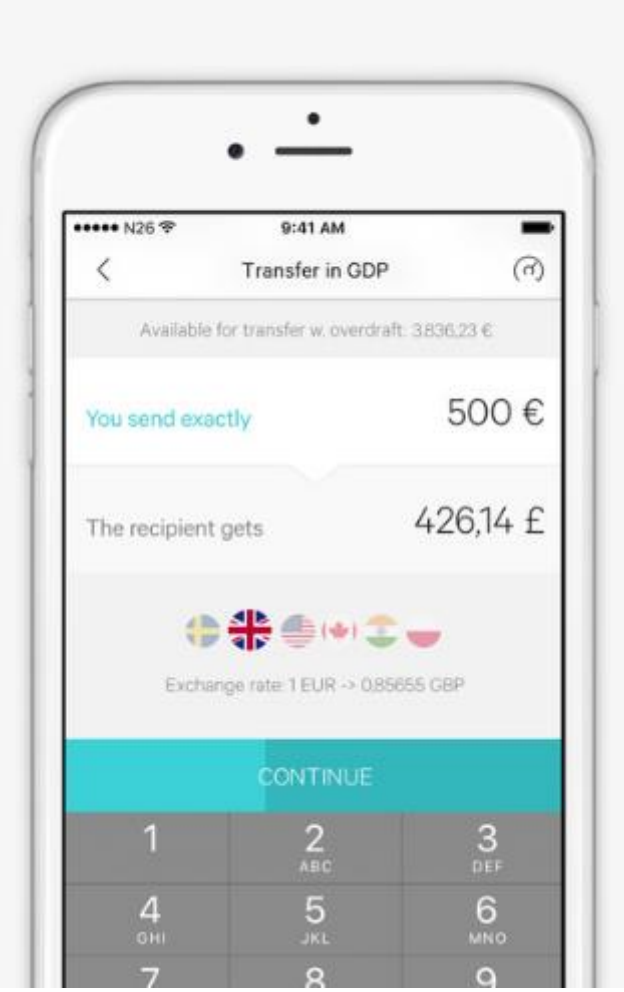

Figura 18: N26 screenshot 1

Come accennato in precedenza l'app permette, quindi, di trasferire denaro verso un numero telefonico associato a un altro conto N26. Come si evince dalla figura l'applicazione permette così di selezionare una quantità di denaro da inviare anche in una differente valuta. Nell'esempio mostrato l'utente ha la possibilità di verificare quanto denaro in Euro sta per inviare e il corrispettivo convertito in Sterline. In basso vi è la possibilità di scegliere fino a 19 valute possibili con il cambio tra la propria valuta e quella selezionata tramite scelta della bandiera dello stato corrispondente.

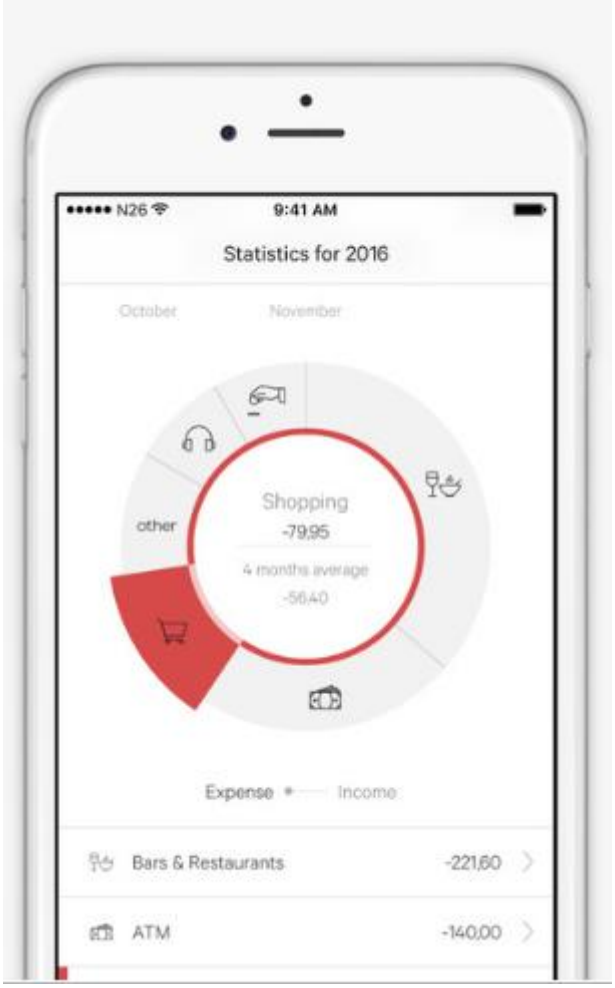

Figura 19: N26 screenshot 2

Similmente a Oval Money e le altre app presenti sul mercato anche con N26 vi è la possibilità di selezione per categoria delle proprie spese. Queste statistiche possono essere suddivise per periodi e per categorie permettendo all'utente di avere immediatamente il quadro della situazione. Tramite selezione di una certa categoria si ha nella parte inferiore dello schermo una lista dei movimenti facente parte della categoria selezionata. Con un *tap* su uno dei movimenti della lista si possono avere delle informazioni aggiuntive, come ad esempio l'esercente presso cui si è effettuato il pagamento, la data e l'ora. Una caratteristica importante di N26 è quella di poter accedere a caratteristiche di sicurezza in semplici passaggi senza dover contattare alcun tipo di assistenza. Tramite accesso alla schermata delle impostazioni della carta

associata si può facilmente resettare il pin o bloccare la carta. Questo permette all'utente di avere un maggiore controllo e di accorciare molto i tempi in situazioni che possono essere molto stressanti e sgradevoli ed evitare possibilmente l'utilizzo della carta dopo ad esempio un eventuale furto o smarrimento.

Figura 20: N26 screenshot 3

Altra caratteristica interessante di N26 è quella di permettere ai propri utenti di investire i propri fondi tramite un'apposita sezione dell'App. All'interno di questa sezione l'utente può anche verificare l'evoluzione dei propri investimenti e avere anche una

previsione degli stessi in un determinato periodo. Inoltre l'applicazione permette di proiettare l'evoluzione della quantità di denaro sul conto stesso anche per un arco di tempo come ad esempio un decennio, con la possibilità di modificare la quantità di denaro accumulata e trasferita sul conto mensilmente o in un sola volta.

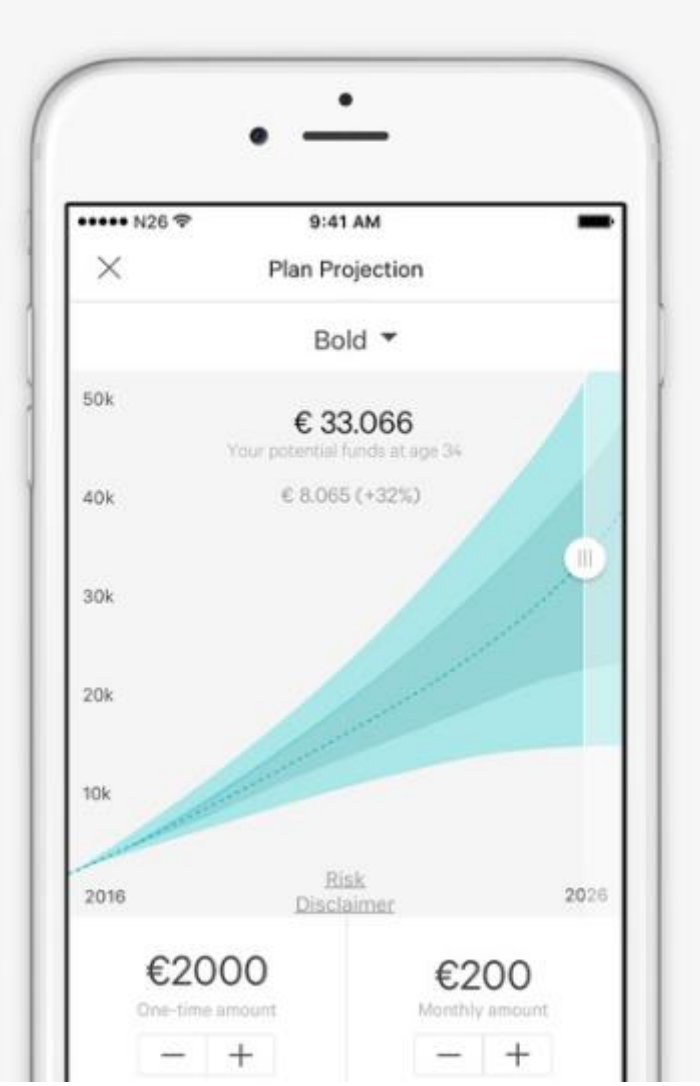

Figura 21: N26 screenshot 4

4.3 Fineco App

Fineco App è l'applicazione ufficiale di FinecoBank S.p.A., la banca diretta multicanale del gruppo UniCredit, con più di un milione di clienti e una delle maggiori reti di consulenza in Italia, con oltre 2.600 Personal Financial Advisor, una raccolta netta di 3,6 miliardi di euro e circa 21 miliardi di patrimonio nel segmento *private* al 30 settembre 2016. L'accesso all'app è permesso tramite il Finger ID, ovvero tramite impronta digitale.

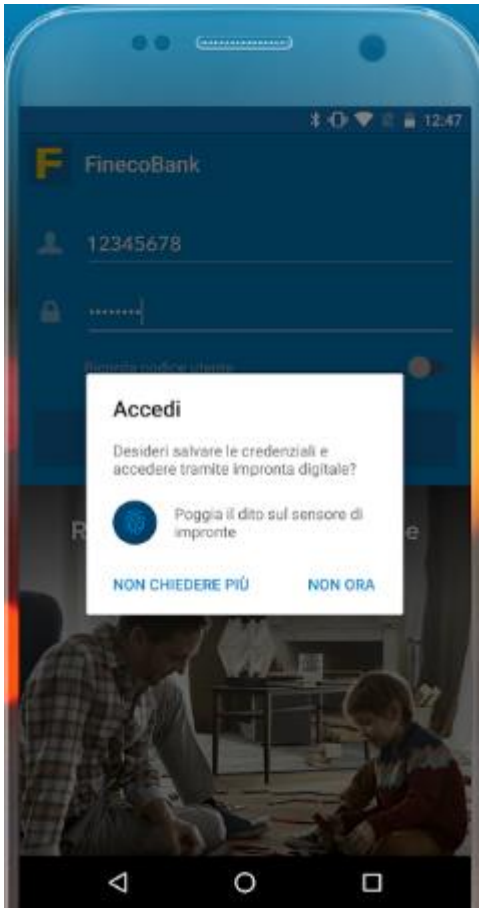

Figura 22: Fineco App screenshot 1

Dalla figura vediamo che dopo aver inserito per la prima volta le credenziali si può accedere tramite impronta digitale, evitando cosi in futuro di dover reinserire le credenziali, ma poggiando solamente il dito sul sensore di impronta. I servizi a disposizione sono quelli di banking, accesso a carte e bancomat, e la possibilità di fare

investimenti e trading professionale. Come le altre applicazioni presenti sul mercato vi è la possibilità di controllare il proprio conto e i movimenti in pochi secondi, e gestire le proprie carte direttamente dallo smartphone. Vi è la possibilità di ricaricarle, impostarne i limiti di spesa, aumentare l'importo limite o sbloccarlo temporaneamente per un maxi-prelievo fino a €3.000 o un maxi acquisto fino a €5.000.

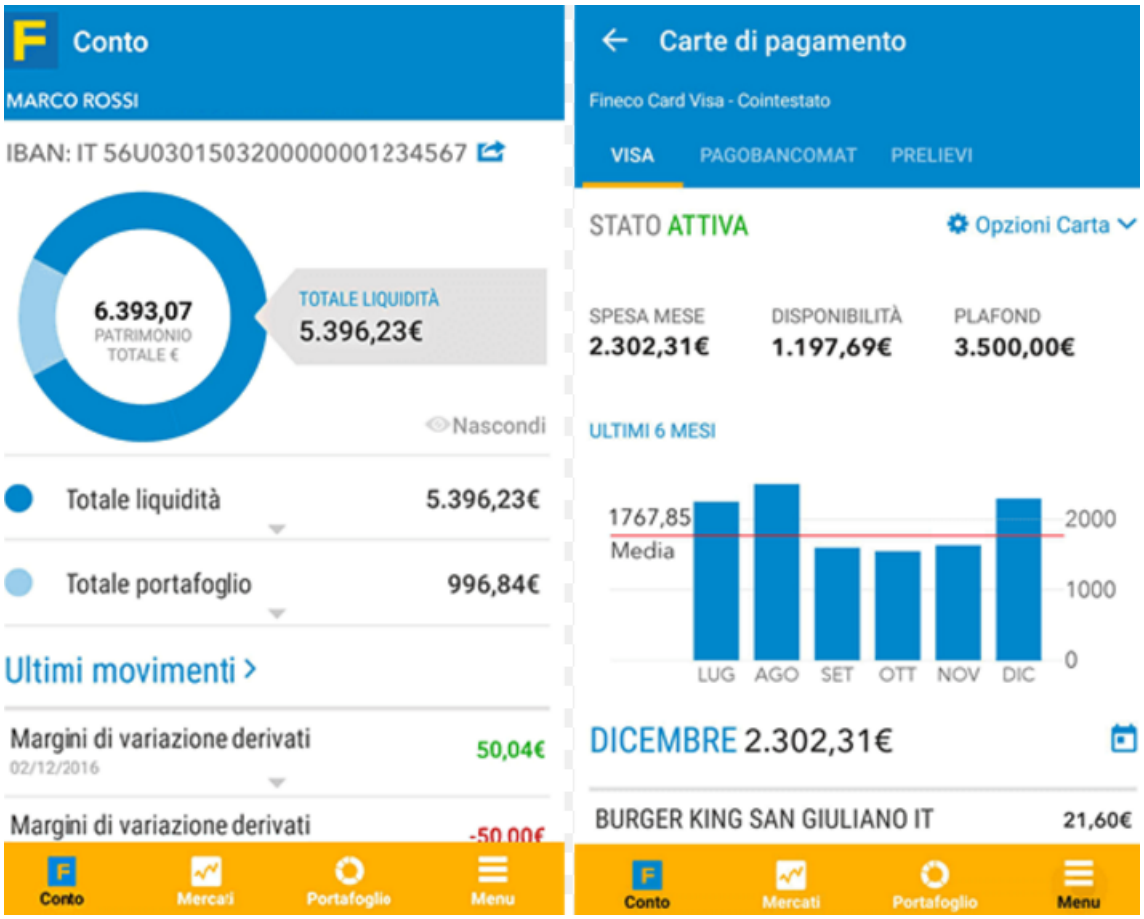

Figura 23: Fineco App screenshot 2

Pensata per migliorare la qualità dei propri servizi per i propri clienti, l'App permette di pagare inoltre i bollettini, Mav e Rav con una semplice foto, effettuare bonifici anche sull'estero, ricaricare il telefono e scambiare valuta in *real-time* grazie al servizio *multicurrency*. Per quanto riguarda gli investimenti, l'App Fineco si trasforma in una

vera piattaforma professionale dando la possibilità di accesso a 26 mercati mondiali con quotazioni in aggiornamento. L'utente viene aggiornato in tempo reale sui mercati con il nuovo Tweets Center e il calendario economico.

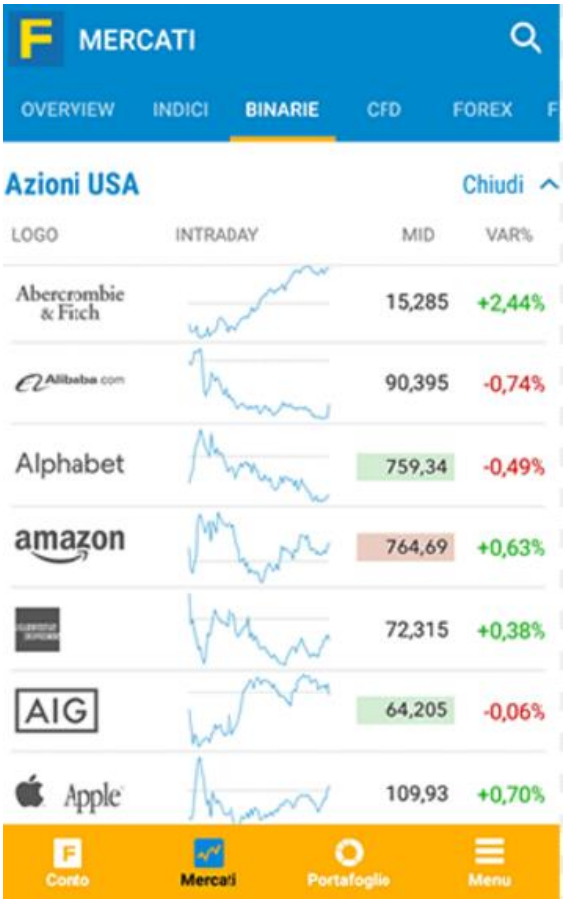

Figura 24: Fineco App screenshot 3

La figura mostra l'andamento delle azioni delle maggiori compagnie mondiali. Vengono mostrati dei grafici riguardo gli andamenti, il MID e VAR.

Figura 25: Fineco App screenshot 2

Come accennato in precedenza gli utenti vengono aggiornati in tempo reale tramite il calendario economico. Esso seleziona i dati principali della giornata da mostrare all'utente e permette l'accesso al calendario completo con la possibilità di filtraggio sulle varie compagnie. Se per quanto riguarda i servizi di banking offerti l'App Fineco non si differenzia molto dalle altre soluzioni presenti sul mercato, per quanto concerne la sfera degli investimenti questa App è una tra le migliori e ne fa uno dei suoi punti di forza per quanto riguarda le funzionalità connesse al Personal Financial Management.

4.4 MoneyFarm

MoneyFarm, è l'applicazione dell'omonima società indipendente di consulenza finanziaria online con sede in Italia e nel Regno Unito. Da una ricerca svolta dall'Istituto Tedesco Qualità e Finanza su oltre 100.000 consumatori, è stata valutata come migliore servizio di consulenza finanziaria indipendente in Italia per il periodo 2015-2016 e per il 2016-2017. MoneyFarm offre consulenza costruendo portafogli di investimenti composti da ETF o Exchange Traded Fund. Questi ultimi rappresentano una tipologia di fondo d'investimento negoziato in borsa che riassume in sé le caratteristiche proprie di un fondo e di un'azione, consentendo agli investitori di sfruttare i punti di forza di entrambi gli strumenti. Riassumere in sé le caratteristiche di un fondo e di un'azione, significa dare la possibilità di diversificazione e riduzione del rischio proprie dei fondi e in più la flessibilità e la trasparenza informativa della negoziazione in tempo reale, tipica delle azioni. Gli ETF replicano passivamente gli indici finanziari a cui si riferiscono, in quanto non c'è un gestore che ribilancia le posizioni all'interno del paniere come accade nei fondi comuni d'investimento. Gli ETF possono consentirci di accedere a tutte le *asset class* disponibili sul mercato finanziario: azioni, bond, commodities, esposizioni in valuta di qualsiasi area geografica e di qualsiasi tipologia. MoneyFarm consiglia quindi, in base al profilo di rischio assegnato al cliente, uno dei 12 portafogli di investimento già elaborati, e provvederà in seguito a comunicare i ribilanciamenti da effettuare per rendere il portafoglio lineare con la *view* di mercato della società. Il compenso che MoneyFarm chiede annualmente è dell'1,25% per investimenti fino a 3.000 euro, dello 0,7% tra 3.000 e 200.000 euro ed oltre questa soglia lo 0,5%. Quindi, su un investimento di 100.000 euro il cliente pagherà a MoneyFarm un compenso di 700 euro l'anno. Moneyfarm nel compenso include anche le commissioni di transazione per l'acquisto degli ETF e non prevede costi aggiuntivi quali commissioni di entrata, uscita o di performance. Inoltre non è richiesto per il cliente nessun vincolo temporale di investimento ed esso è libero di abbandonare il servizio senza costi aggiuntivi. Come le altre applicazioni per la gestione delle finanze personali anche Money Farm permette l'inserimento di tutte le proprie entrate e delle proprie uscite.

Figura 26: Money Farm screenshot 1

Essendo non connessa a un particolare conto o carta, l'utente va a inserire le proprie transazioni che vengono categorizzate e suddivise per periodo. Dalla figura si evince come l'App indichi il totale di tutte le spese effettuate nella parte alta dello schermo e marchi con diversi colori le entrate e le uscite per essere maggiormente intuitiva e fornire all'utente in modo immediato le informazioni più utili. L'App permette inoltre tramite determinate schermate la visualizzazione più compatta delle varie transazioni che permettono facilmente all'utente di avere una migliore visione di insieme di tutti i propri movimenti finanziari. Oltre alle transazioni è possibile andare a definire dei limiti circa determinate categorie di spese e obiettivi di risparmio periodici affinché sia più semplice per l'utente portare a termine questi ultimi.

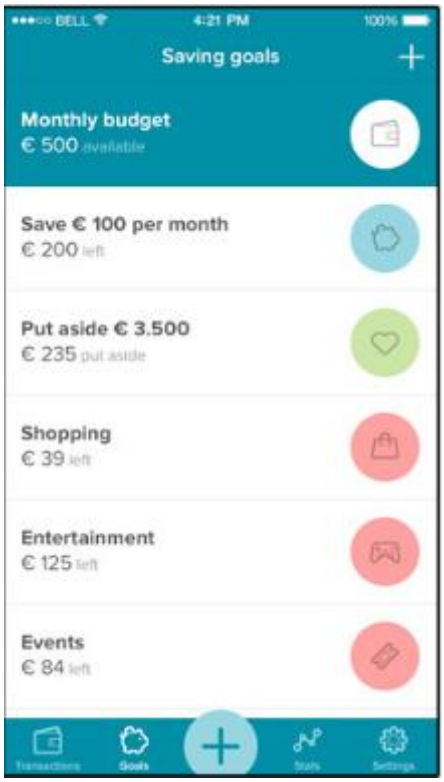

Figura 27: Money Farm screenshot 2

Il punto di forza di Money Farm è sicuramente la scelta del piano di investimento migliore per ogni utente. Una volta che l'utente accede al servizio viene chiesto di compilare un questionario affinché venga definito il profilo di rischio dell'utente stesso e arrivare a una soluzione su misura per lui. In seguito vengono combinate diverse tipologie di investimento e aree geografiche, per massimizzare la diversificazione del portafoglio e garantire un'esposizione globale. Durante tutto il periodo di utilizzo vengono eseguiti dei ribilanciamenti ovvero vengono apportati degli aggiustamenti necessari a proteggere il capitale monitorando i mercati e l'andamento del portafoglio dell'utente.

CAPITOLO 5

ANALISI TECNICA

In questo capitolo verranno analizzate le librerie maggiormente utilizzate dalle applicazioni che si occupano di Personal Financial Management che, come analizzato nel capitolo precedente, fanno largo uso di grafici per rappresentare in modo immediato i dati agli utenti.

5.1 MPAndroidChart

MPAndroidChart è una libreria open-source per Android semplice e intuitiva da utilizzare per la rappresentazione di grafici e istogrammi. Lavora con le API di livello 8 e superiori. Una caratteristica aggiuntiva di questa libreria è quella di permettere lo sviluppo su diverse piattaforme, in particolare Android e iOS. La libreria Charts rappresenta l'equivalente per iOS. MPAndroidChart permette di accedere a diverse Custom View, utili al programmatore per creare i propri layout rappresentanti grafici e/o istogrammi tramite XML oppure in maniera programmatica tramite linguaggio Java. Viene mostrato un esempio di un semplice grafico.

```
<com.github.mikephil.charting.charts.LineChart
    android:id="@+id/chart"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Figura 28: Esempio Codice MPAndroidChart

```
// programmatically create a LineChart
LineChart chart = new LineChart(Context);
```

Figura 29: Esempio Codice MPAndroidChart 1

In questo modo siamo capaci di andare a istanziare un semplice grafico. Per mostrare i dati occorre andare a definire un oggetto LineData. Nel costruttore di quest'ultimo viene definita una lista di coppie di valori, ovvero la posizione che ogni punto della linea avrà nel grafico. Possono eventualmente essere settati dei colori per ogni segmento della linea. Per mostrare il grafico cosi definito occorre invocare sull'oggetto chart il metodo setData, a cui viene passato l'oggetto LineData, e infine invocare il metodo invalidate, per permettere al sistema operativo di Android di poter disegnare la linea sul grafico. In figura viene mostrato un esempio di LineChart definito seguendo i passaggi precedentemente descritti.

Figura 30: Esempio LineChart MPAndroidChart

La libreria permette inoltre di disegnare altri tipi di grafici come ad esempio BarChart e PieChart ovvero istogrammi e grafici a torta.

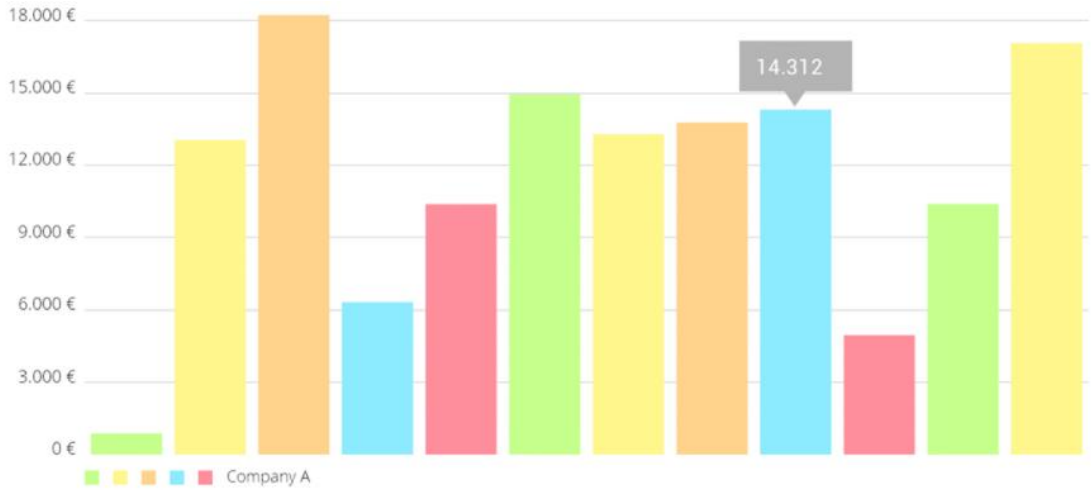

Figura 31: Esempio BarChart MPAndroidChart

Figura 32: Esempio PieChart MPAndroidChart

Inoltre sono forniti vari metodi che permettono di personalizzare la gestione dei tocchi semplici e delle *gesture,* tramite metodi di callback. Vi è la possibilità di abilitare o

meno i tocchi con setTouchEnabled, tramite passaggio di un booleano come parametro che permette di attivare o meno la gestione dei tocchi. Dopo aver abilitato i tocchi vi è la possibilità di gestire lo *Scale* lungo gli assi e lo zoom, rispettivamente tramite setScaleEnabled e setPinchZoom. Un'altra importante caratteristica della libreria MPAndroidChart è quella di poter evidenziare tramite tocco alcuni punti del grafico mentre lo si sta analizzando. Dalla selezione tramite *tap* di un certo punto, si può notare la modifica del colore e della dimensione del punto stesso sul grafico e inoltre si può evidenziare la distanza tra due punti del grafico. Verrà cosi disegnata sul grafico una linea dopo aver tappato sui due punti distinti del grafico e lungo gli assi altre due linee che evidenziano la distanza lungo l'asse x e lungo l'asse y. Per la gestione degli eventi come i vari tocchi sul grafico o le gesture occorrerà utilizzare una classe che implementa l'interfaccia onChartGestureListener e settare una sua istanza come gestore delle varie *callback* per quel grafico, in modo che si possa gestire in modo personalizzato qualsiasi tipo di evento, tramite il metodo setOnChartGestureListener. Vari metodi permettono di personalizzare la visualizzazione degli assi e dei dati, e la possibilità di settare della animazioni tramite l'impostazione di Animations su determinati punti del grafico. Come mostrato in precedenza la libreria permette la realizzazione di diverse tipologie di grafici e ciò comporta l'utilizzo di diverse tipologie di dati. Per esempio, se si vuole utilizzare un BarChart occorre una lista di una List<BarEntry> che conterrà i valori da inserire nel grafico. Tramite questa lista si può istanziare un BarDataSet e in seguito un'istanza della classe BarData che viene settata sul grafico tramite il metodo setData. Come per il caso mostrato all'inizio del sotto-capitolo occorre chiamare sull'oggetto che rappresenta il grafico il metodo invalidate, per permettere il ridisegno del grafico con i dati appena settati. Poiché MPAndroidChart è una libreria open-source, semplice e immediata da utilizzare e soprattutto con un alto numero di grafici a disposizione è sempre più utilizzata dagli sviluppatori, non solo di App nell'ambito del Personal Financial Management, ma di tutte quelle riguardanti la raffigurazione di dati tramite grafici.

5.2 GraphView

GraphView come MPAndroidChart è una libreria open-source con la quale si ha la possibilità di creare facilmente varie tipologie di diagrammi. Alcuni possibili grafici che possono essere creati sono Line Graphs, Bar Graphs, Point Graphs. Vi è inoltre la possibilità di poter implementare proprie tipologie personalizzate di grafici.

```
<com.jjoe64.graphview.GraphView
        android:layout_width="match_parent"
        android:layout_height="200dip"
        android:id="@+id/graph" />
```

Figura 33: Esempio Codice XML GraphView

```
GraphView graph = (GraphView) findViewById(R.id.graph);
LineGraphSeries<DataPoint> series = new LineGraphSeries<>(new DataPoint[] {
          new DataPoint(0, 1),
          new DataPoint(1, 5),
          new DataPoint(2, 3),
          new DataPoint(3, 2),
          new DataPoint(4, 6)
});
graph.addSeries(series);
```

Figura 34: Esempio Codice Java GraphView

Dalle figure precedenti si nota la semplicità con la quale è possibile definire nel layout, tramite XML, un GraphView e popolarlo in maniera programmatica tramite il metodo *addSeries* che accetta una *LineGraphSeries*. Quest'ultima è una collezione di *DataPoint* che viene inizializzata nell'esempio con cinque *DataPoint*, il cui costruttore richiede il valore dell'ascissa e dell'ordinata. Tramite diversi segmenti verranno collegati i vari punti sul grafico. L'effetto finale è di avere una *line* lungo tutto il grafico. In questo modo si avrà un grafico come quello mostrato nella figura successiva.

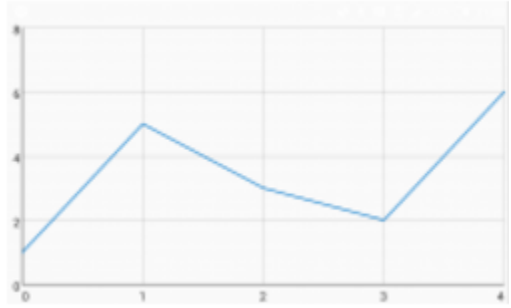

Figura 35: Esempio grafico GraphView

GraphView è una *Custom View* che estende direttamente dalla classe base *View* di Android. Come tale essa può essere utilizzata come tutte le altre *View* in ogni layout creato tramite XML. Oltre all'esempio mostrato in precedenza è possibile anche definire un grafico con dei punti definiti direttamente dal codice XML. Ovviamente l'utilizzo del metodo addSeries, visto nell'esempio, permette di avere una migliore gestione del grafico stesso poiché i dati possono essere più facilmente gestiti programmaticamente. La gestione tramite codice Java a differenza dell'impostazione dei punti tramite XML, permette di poter aggiungere nuovi dati al grafico oppure si può azzerare l'intero insieme dei dati. Una Series, contiene i punti principali di una *line* che possono formare una linea oppure essere mostrati come dei punti o come barre rispettivamente nel Point Graphs e nel BarChart. Vi è inoltre la possibilità di andare a settare per un grafo più di una serie di punti formando cosi più linee che risultano essere indispensabili quando occorre mostrare delle comparazioni. Altre caratteristiche fondamentali sono quelle di poter gestire il tocco dell'utente sui vari punti del grafico tramite impostazioni di vari *listener* e gestione delle *callback.* Per mostrare i dati nella maniera più comprensibile possibile si ha a disposizione una legenda tramite la quale specificare ad esempio il significato dei vari assi e delle varie *line* rappresentate sul grafico. La semplicità di utilizzo è sicuramente uno dei punti di forza di questa libreria ma il numero non elevato di grafici disponibili per lo sviluppatore ne limita la diffusione e l'utilizzo.

5.3 AndroidPlot

Androidplot è una libreria open-source per creare grafici in maniera dinamica o statica. E' stata progettata per Android ed è compatibile dalla versione 1.6 in poi. Oggi è utilizzata da più di 1000 applicazioni disponibili su Google Play. Similmente alle librerie finora utilizzate occorre definire nel XML del layout la *Custom View* tramite la quale verrà raffigurato il grafico. La Custom View *XYPlot* è sicuramente la più semplice da rappresentare.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:ap="http://schemas.android.com/apk/res-auto"
              android:layout_height="match_parent"
              android:layout_width="match_parent">

    <com.androidplot.xy.XYPlot
        style="@style/APDefacto.Dark"
        android:id="@+id/plot"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        ap:title="A Simple XY Plot"
        ap:rangeTitle="range"
        ap:domainTitle="domain"
        ap:lineLabels="left|bottom"
        ap:lineLabelRotationBottom="-45"/>
</LinearLayout>
```

Figura 36: Esempio codice XML Android Plot

Dal frammento di codice precedente si nota come oltre a definire l'altezza e la larghezza possono essere definite altre proprietà come il titolo e altre caratteristiche legate allo stile. Viene mostrato nella figura successiva il codice relativo al recupero dati tramite codice Java del Plot definito nella porzione di codice XML mostrato in precedenza. Il Plot viene così popolato con una o più serie di dati. A differenza delle librerie mostrate finora, dove veniva definito un insieme di dati con relativa ascissa e ordinata, in questo caso occorre andare a definire un array di *Number* per indicare i vari punti del dominio. Per ogni linea che dovrà essere rappresentata sul grafico, occorre definire un array di

Number, dove i valori rappresenteranno le ordinate dei vari punti la cui ascissa avrà il valore definito nell'array del dominio.

```
// initialize our XYPlot reference:
plot = (XYPlot) findViewById(R.id.plot);

// create a couple arrays of y-values to plot:
final Number[] domainLabels = {1, 2, 3, 6, 7, 8, 9, 10, 13, 14};
Number[] series1Numbers = {1, 4, 2, 8, 4, 16, 8, 32, 16, 64};
Number[] series2Numbers = {5, 2, 10, 5, 20, 10, 40, 20, 80, 40};

// turn the above arrays into XYSeries':
// (Y_VALS_ONLY means use the element index as the x value)
XYSeries series1 = new SimpleXYSeries(
        Arrays.asList(series1Numbers), SimpleXYSeries.ArrayFormat.Y_VALS_ONLY, "Series1");
XYSeries series2 = new SimpleXYSeries(
        Arrays.asList(series2Numbers), SimpleXYSeries.ArrayFormat.Y_VALS_ONLY, "Series2");

// add a new series' to the xyplot:
plot.addSeries(series1, series1Format);
plot.addSeries(series2, series2Format);
```

Figura 37: Esempio codice Java Android Plot

Dopo aver definito gli array con i valori, occorre definire una XYSeries tramite il costruttore della classe *SimpleXYSeries* che tra i vari parametri accettati vi è l'array rappresentate i valori delle ordinate. Successivamente come si nota dalla figura, vengono aggiunte le due XYSeries e successivamente tramite il metodo *getGraph* verrà settato l'Array rappresentante il dominio e solo allora le varie linee saranno rappresentate sul grafico.

```
plot.getGraph().getLineLabelStyle(XYGraphWidget.Edge.BOTTOM).setFormat(new Format() {
    @Override
    public StringBuffer format(Object obj, StringBuffer toAppendTo, FieldPosition pos) {
        int i = Math.round(((Number) obj).floatValue());
        return toAppendTo.append(domainLabels[i]);
    }
    @Override
    public Object parseObject(String source, ParsePosition pos) {
        return null;
    }
});
```

Figura 38: Esempio codice Java Android Plot 2

Dalla figura successiva si può avere un esempio finale del grafico prodotto tramite il codice descritto finora, dove si è evitato di analizzare il codice relativo alla personalizzazione del layout delle varie linee definibile tramite XML.

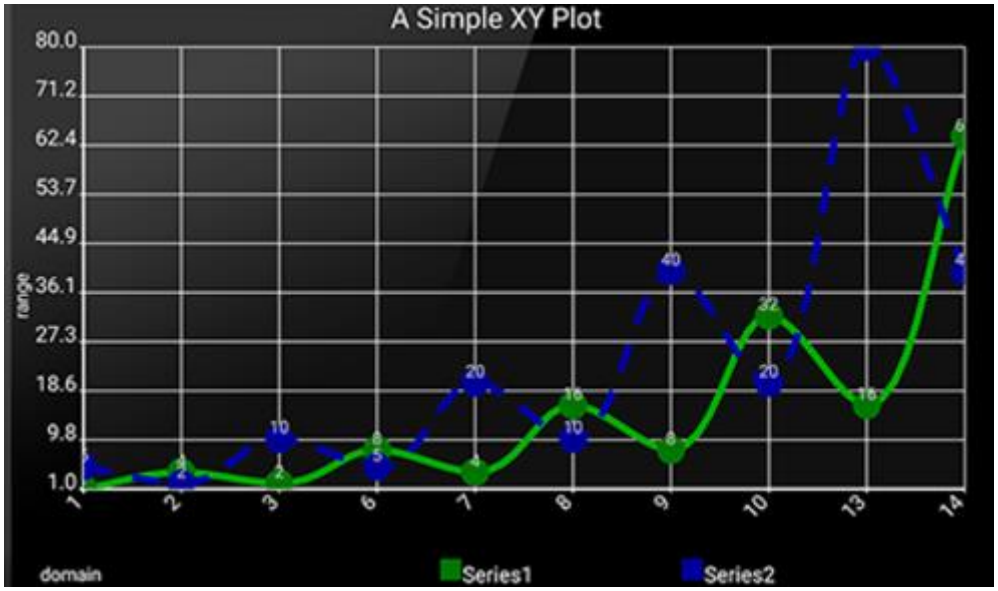

Figura 39: Esempio grafico Android Plot

Dalla descrizione precedente non sono analizzati i metodi utilizzati per modificare lo stile. Infatti, caratteristica chiave di questa libreria è quella di fornire all'utilizzatore un numero abbastanza elevato di metodi per modificare lo stile sia del grafico in termini di leggenda, titolo e delle linee stesse che vengono raffigurate. Come MPAndroidChart questa libreria permette di raffigurare grafici come BarChart o PieChart. Questa caratteristica ha permesso alla libreria di essere molto utilizzata come testimonia l'alto numero di applicazioni che ne fanno uso. Bisogna però sottolineare che se da un lato risulta essere la libreria con più metodi disponibili per la personalizzazione grafica, l'impostazione dei dati che devono essere raffigurati ne aumenta la complessità di utilizzo.

5.4 AchartEngine

AchartEngine è una libreria open-source disponibile per Android. Correntemente supporta diversi tipi di grafici tra cui LineChart, BarChart, PieChart e altri ancora. Ogni tipo di grafico può contenere diverse serie di dati e vi è la possibilità di gestire i vari assi, con la possibilità di invertirli, e moltissimi altre caratteristiche riguardo la personalizzazioni dal punto di vista grafico. La particolarità di questa libreria è quella di poter creare i grafici oltre che come delle comuni View, quindi sia programmaticamente sia tramite dichiarazione all'interno di un layout definito tramite XML, anche tramite dichiarazione di un Intent. In questo modo si avvia una nuova Activity contenente il grafico dichiarato. Il codice riguardo il modello e i grafici stessi è ben ottimizzato cosi che può gestire un enorme numero di valori. Nonostante la grande varietà di grafici disponibili, a causa del continuo aumento del numero di App che utilizzano questa libreria, attualmente alla versione 1.0.0, nuovi grafici saranno aggiunti nei prossimi rilasci. Per quanto concerne la compatibilità AchartEngine supporta tutte le versioni di Android SDK dalla versione 1.6 in poi. Anche se il numero di grafici è abbastanza elevato, essi possono essere suddivisi in tre macro-categorie:

- XY charts, ovvero grafici che mostrano dati su due assi;
- Round charts come ad esempio i grafici a torta;
- Combined charts, che mostrano una combinazioni di diversi XY charts;

Similmente alle librerie descritte precedentemente, per popolare un grafico occorre una serie di dati. Diverse classi possono essere utilizzate per questo scopo, molte delle quali estendono XYSeries. Questa classe gestisce internamente una mappa contenente una coppia di Double per definire il valore dell'ascissa e dell' ordinata di ognuno dei vari punti. Dopo aver quindi creato una serie di dati, essi possono essere utilizzati direttamente nel costruttore del grafico, o tramite appositi metodi dopo aver istanziato la classe che rappresenta il grafico. Ovviamente, in seguito ogni modifica dei dati il grafico viene modificato di conseguenza. Nella figura successiva viene definito il metodo per creare un ChartView.

```
mChartView = ChartFactory.getLineChartView(this, mDataset, mRenderer);
```

Figura 40: Esempio codice AChartEngine

Oltre al contesto corrente in cui definire il grafico e al set di dati che dovranno essere mostrati nel grafico stesso, viene utilizzato come terzo parametro un Render, il quale serve per la gestione grafica dei component principali. Ad esempio tramite il Render possono essere definiti colori, e forme delle varie linee, o il font e la dimensione del titolo del grafico. Un esempio di ChartView con una serie di 10 punti viene mostrata nella figura sottostante.

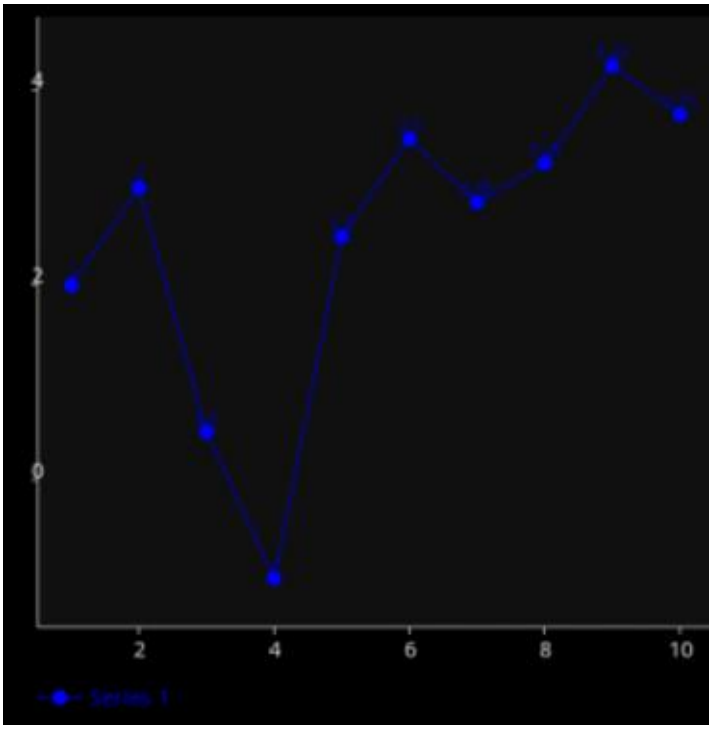

Figura 41: Esempio grafico AChartEngine

Cambiando una singola linea di codice si può facilmente rimpiazzare un lineChart con un barChart.

```
mChartView = ChartFactory.getLineChartView(this, mDataset, mRenderer);
mChartView = ChartFactory.getBarChartView(this, mDataset, mRenderer, Type.DEFAULT);
```

Figura 42: Esempio codice AchartEngine 2

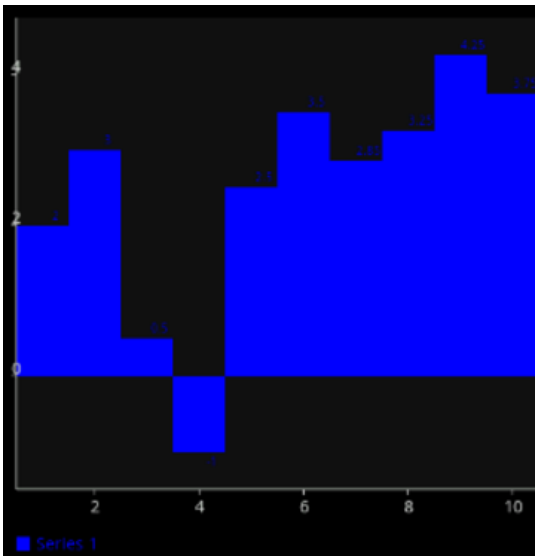

Figura 43: Esempio grafico AchartEngine 2

Per i due grafici mostrati in precedenza deve essere definito nel layout un LinearLayout che ospiterà il grafico. Con pochissime modifiche dal punto di vista del layout è possibile definire un grafico a torta. Occorre una singola TextView per ogni singola porzione di dati da inserire nel grafico e un nuovo tipo di Render, dove vi è la possibilità di definire un certo colore per ogni fetta del grafico.

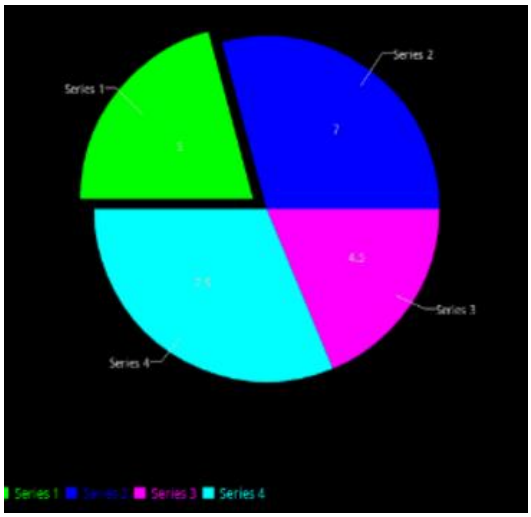

Figura 44: Esempio grafico AchartEngine 3

Se da un lato la libreria permette in maniera abbastanza semplice di definire un numero elevato di grafici, dal punto di vista grafico risulta essere leggermente meno performante rispetto ad altre librerie, come alcune di quelle definite in precedenza.

5.5 Orson Charts for Android

Orson Chart for Android è una libreria non open-source, per grafici 3D capace di generare un'ampia varietà di grafici da utilizzare. La semplicità di utilizzo fa di Orson Charts uno strumento altamente utilizzato soprattutto in ambito professionale. I grafici che vengono generati risultano essere interattivi e allo stesso tempo molto intuitivi per gli utenti. Come molte delle librerie analizzate finora Orson Charts permette la definizione di diversi tipi di grafici come PieChart, BarChart, LineChart e altri ancora. Viene utilizzato un motore 3D basato sulle API Canvas di Android. L'interazione con l'utente è possibile tramite la gestione dei vari *touch* sui vari punti del grafico ed è permessa la rotazione a 360 gradi di ogni tipo di grafico. Occorre sottolineare che tramite API accuratamente documentate è possibile un alto grado di configurabilità e personalizzazione.

```
Chart3D c1 = Chart3DFactory.createPieChart("title", "subtitle",
        createPieDataset());

// background painter
c1.setBackground(new StandardRectanglePainter(Color.RED));
assertFalse(c1.equals(c2));
```

Figura 45: Esempio codice Orson Chart

Nella porzione di codice rappresentato nella figura precedente, viene inizializzato un Chart3D. Per istanziare l'oggetto occorrono come parametri due *String* che indicheranno titolo e sottotitolo e un *PieDataset3D<String>* che conterrà i dati da

rappresentare. Nella porzione di codice viene definito il background di default della torta. Un esempio del metodo che viene richiamato nel *factory* per la creazione del *dataset* è mostrato nella figura sottostante.

```
private PieDataset3D<String> createPieDataset() {
    StandardPieDataset3D<String> dataset
            = new StandardPieDataset3D<String>();
    dataset.add("United States", new Double(30.0));
    dataset.add("France", new Double(20.0));
    return dataset;
}
```

Figura 46: Esempio codice Orson Chart 2

In questo caso sono definiti solamente due porzioni del grafico a torta che rispettivamente rappresenteranno il 30% e il 20% del grafico totale. Dopo aver istanziato il grafico a torta è possibile andare a modificare oltre al titolo e al sottotitolo, anche il dataSet e in seguito definire per ogni porzione un colore diverso, al fine di differenziare le varie porzioni e permettere al grafico di essere altamente intuitivo. Un esempio del grafico potrebbe essere il seguente.

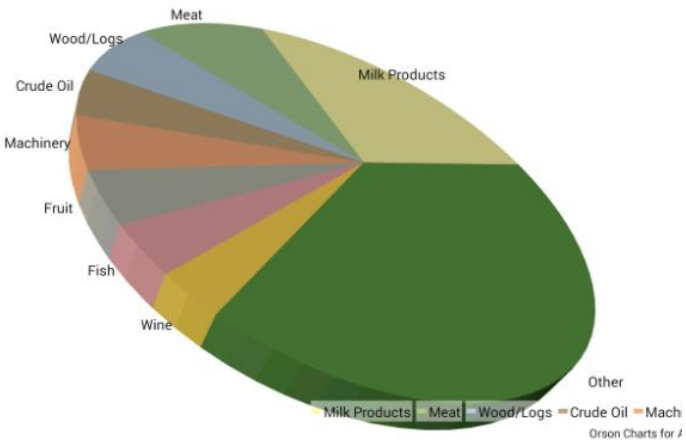

Figura 47: Esempio grafico Orson Chart

Dalla figura precedente si nota la qualità dei grafici ben superiore rispetto alle librerie precedentemente analizzate. Lo stesso vale per i LineCharts, e BarCharts di cui nelle figure seguenti viene mostrato un esempio.

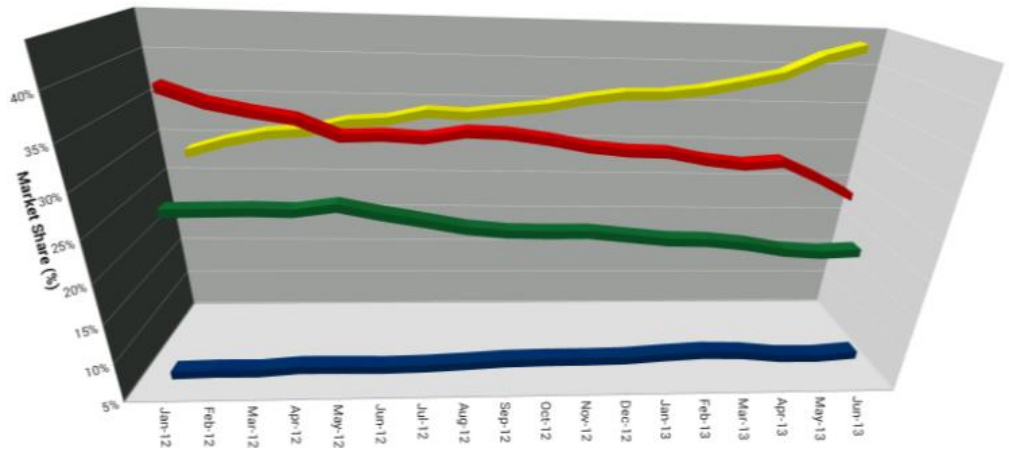

Figura 48: Esempio grafico Orson Chart 2

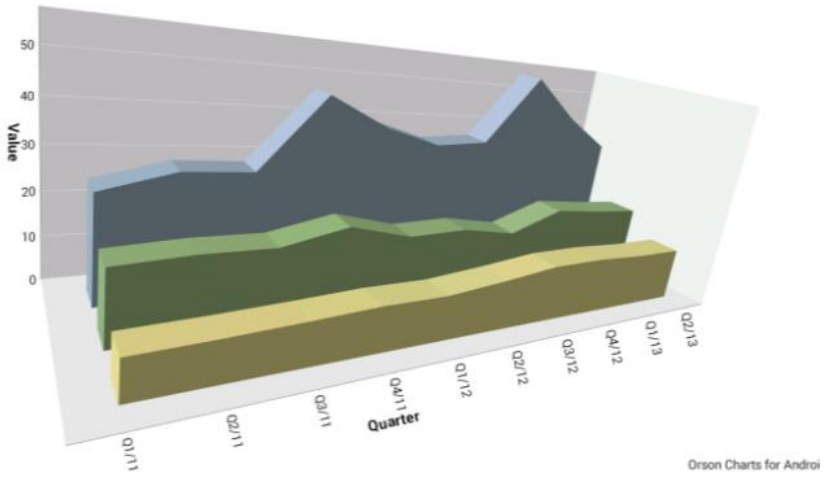

Figura 49: Esempio grafico Orson Chart 3

Il codice per la creazione di questi differenti tipi di grafici non risulta essere molto differente rispetto al PieChart analizzato in precedenza. Se da un lato l'effetto finale è ottimo, l'acquisizione della licenza per l'utilizzo della libreria ne limita il suo utilizzo. Da un minimo di 195 $ per la creazione di una singola App a un massimo di 1495$ per una soluzione Enterprise, fa sì che solamente una porzione ristretta di sviluppatori,start up e aziende ne facciano uso. Un'altra caratteristica che frena l'utilizzo di Orson Chart, che vale la pena evidenziare è quella di fornire solamente grafici 3D e non vi è alcun supporto per la definizione di grafici in sole 2 dimensioni.

5.6 Telerik UI for Android

Telerik UI for Android è una libreria che necessità di licenza di componenti UI nativi con eccellenti capacità grafiche, che permette un'alta definizione e un' elegante interattività. La libreria contiene componenti progettatti appositamente per l'ambiente mobile che offrono grandi performance per quanto riguarda il tempo di caricamento, capacità di disegno e aggiornamenti in real-time. L'intiuitivo *object-model* e le API pubbliche permettono di gestire oggetti complessi e integrarli facilmente nelle App Android. Diversi componenti tra cui RadChartView e RadPieChartView sono disponibili. Essi sono organizzati in gerariche a seconda del loro Sistema di coordinate usato per tracciare i punti. Se nei RadChartView viene utilizzato il sistema cartesiano nei RadPieChartView viene utilizzato un sistema di coordinate apposito per definire le varie porzioni del grafico. Ogni dataset rappresenta una certa collezione di dati, ovvero l'equivalente di un insieme di punti cui viene indicata l'ascissa e l'ordinata. Tramite un processo di data-binding altamente intuitivo vengono trasformati i dati negli appropriati dataPoints a seconda del particolare grafico e quindi della *series* corrispondente. Nelle seguenti figure verrano mostrati i vari step per la rappresentazioni di un RadCartesianChartView che rappresenterà i propri dati tramite un sistema di coordiante cartesiane.

```
RadCartesianChartView chartView = new RadCartesianChartView(this);

ViewGroup rootView = (ViewGroup)findViewById(R.id.container);
rootView.addView(chartView);
```

Figura 50: Esempio codice Telerik UI for Android

Nell'esempio si nota come la View non è recuperata tramite il metodo findViewById dal layout, ma venga inizializzata all'interno del contesto corrente e in seguito aggiunta a un ViewGroup.

```
private List<MonthResult> monthResults;

private void initData() {
    monthResults = new ArrayList<MonthResult>();
    monthResults.add(new MonthResult("Jan", 12));
    monthResults.add(new MonthResult("Feb", 5));
    monthResults.add(new MonthResult("Mar", 10));
    monthResults.add(new MonthResult("Apr", 7));
}
```

Figura 51: Esempio codice Telerik UI for Android 2

I dati utilizzati nell'esempio corrente sono rappresentati tramite un *ArrayList* di oggetti del tipo *MonthResult*, i quali conterranno la String contentente il nome del mese e un valore interno.

```
initData();

LineSeries lineSeries = new LineSeries();
lineSeries.setCategoryBinding(new PropertyNameDataPointBinding("Month"));
lineSeries.setValueBinding(new PropertyNameDataPointBinding("Result"));
lineSeries.setData(this.monthResults);
chartView.getSeries().add(lineSeries);
```

Figura 52: Esempio codice Telerik UI for Android 3

Lo step successivo consiste nel definire a partire dall' ArrayList una LineSeries settando, oltre il nome della categoria, il nome dell'etichetta da presentare per i valori che saranno aggiunti tramite setData. Infine, occorre aggiungere alla lista di Series che il grafico può contenere la LineSeries appena definita. Nella figura sottostante vediamo il risultato finale del codice analizzato.

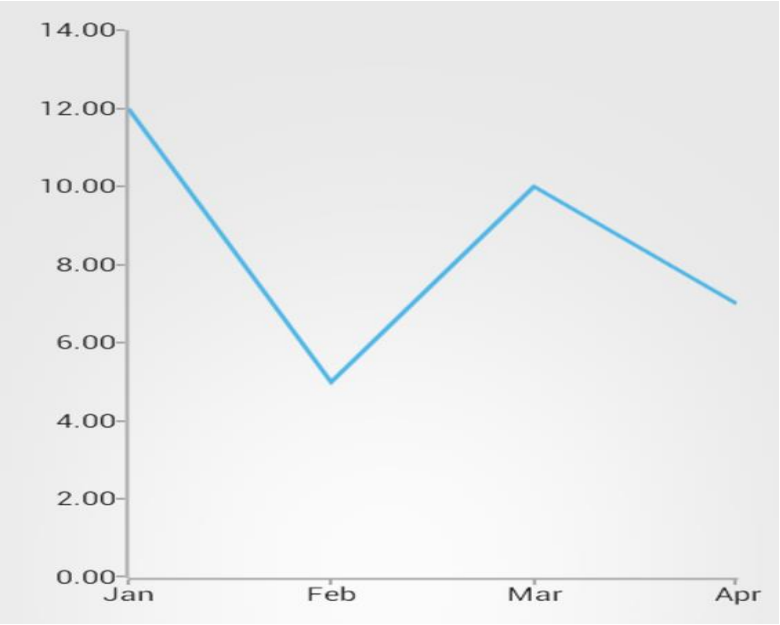

Figura 53: Esempio grafico Telerik UI for Android 3

Il grafico così ottenuto risulta essere minimale e intuitivo. Da notare che la gestione dei valori che verranno mostrati lungo le ordinate viene calcolato automaticamente a seconda dei valori delle Series che verranno mostrate nel grafico. Nelle figure successive sono mostrati alcuni esempi grafici di Bar Chart, PieChart e CartesianChart. L' implementazione di questi grafici non differisce logicamente da quella appena vista per il RadCartesianChartView e dal punto di vista grafico il risultato è altamente intuitivo.

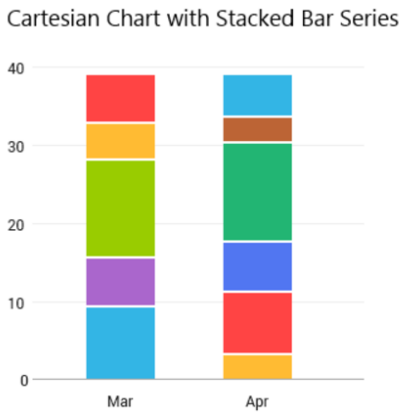

Figura 54: Esempio grafico Telerik UI for Android 2

Figura 55: Esempio grafico Telerik UI for Android 3

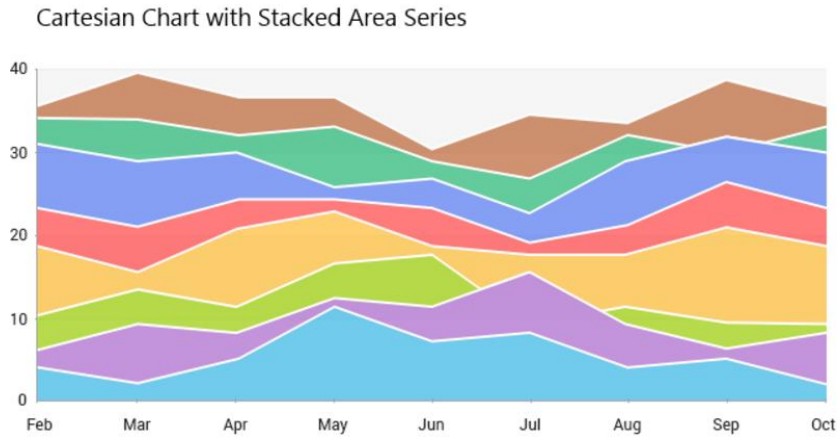

Figura 56: Esempio grafico Telerik UI for Android 4

Come Orson Charts anche Telerik UI for Android registra un uso altamente limitato, nonostante gli ottimi risultati grafici e le API altamente intuitive, a causa dei costi di acquisizione delle licenze. Quest'ultime possono essere acquistate e rinnovate mensilmente da un minimo di 39 $ a un massimo di 149 $ per ogni user a seconda del tipo di licenza. Vi è inoltre la possibilità per le grandi aziende che devono sviluppare grandi progetti di stipulare particolari accordi direttamente con Telerik.

5.7 Comparazione tra le librerie analizzate

Dalle analisi condotte nei sottocapitoli precedenti si evince come la semplicità delle API ricopra un ruolo molto importante nella definizione della qualità di una libreria. Ciò aumenta l'indice di gradimento degli utilizzatori e cosi il numero delle App che ne fanno uso cresce continuamente. Tutto questo permette un continuo miglioramento delle librerie stesse con l'inserimento di nuove tipologie di grafici e nuove funzionalità relative alla personalizzazione di quelli già esistenti. Un'altra caratteristica molto importante è la possibilità di poter usufruire di queste librerie in modalità gratuita. Le soluzioni open-source vengano altamente preferite rispetto a quelle per cui occorre acquistare una licenza. Ovviamente i risultati che si possono ottenere tramite Telerik UI for Android o ad esempio tramite Orson Charts sono superiori in termini grafici. Gli alti costi per le acquisizioni delle licenze fanno si che non vengano altamente utilizzate dalla maggior parte degli sviluppatori. Solo alcune grandi aziende decidono di utilizzarle per i loro progetti. Tra le librerie open-source analizzate la migliore soluzione risulta essere MPAndroidChart. In termini di semplcità di API è comprabile alle soluzioni non open-source e anche l'effetto grafico non si discosta di molto. Come detto precedentemente MPAndroidChart UI for Android dispone di un alto numero di grafici 2D. Quindi, se occorre definire grafici in tre dimensioni questa libreria non può essere utilizzata e occorre orientarsi su Orson Charts. Anche se MPAndroidChart non da la possibilità di poter disegnare grafici 3D, essi risultatno essere comunque abbastanza intuitivi ed esplicativi. Inoltre la semplice gestione delle *callback,* in seguito a determinati eventi e/o touch sul grafico permette di aumentare l'interazione tra il grafico stesso e l'utente. Per questo motivo un sempre maggior numero di sviluppatori decide

di utilizzare questa libreria. Essa verrà utilizzata anche nello sviluppo della soluzione mobile di PFM, la quale verrà descritta nel capitolo successivo.

CAPITOLO 6

PROOF OF CONCEPT

In questo capitolo verrà descritta in maniera approfondita la soluzione *mobile* per Personal Financial Management oggetto di questa Tesi, mostrando porzioni del codice sviluppato. Come accennato nei capitoli precedenti obiettivo principale della tesi è quello di creare un *Proof of Concept* di un App simile a quelle descritte nel capitolo quarto, ma che sia migliore nel *layout,* per una superiore *user-experience* e nelle funzionalità affinchè permetta all'utente di gestire le proprie spese più nel dettaglio e di definire obiettivi finanziari da raggiungere. Dal punto di vista architetturale si è utilizzato il Model-View-Presenter che verrà analizzato nel sottocapitolo seguente.

6.1 Definizione Modello MVP

Il *pattern* architetturale utilizzato nella soluzione proposta è quello del Model-View-Presenter o MVP che è una derivazione di un altro *pattern* architetturale, il model–view–controller o MVC . Questo modello architetturale viene utilizzato maggiromente per l'implementazione di interfacce utente. Nel pattern MVP :

- Il modello è un'interfaccia che definisce i dati che devono essere mostrati all' utente.
- La *view* è un'interfaccia passiva che mostra i dati (ovvero il modello) e instrada i comandi dell'utente al presenter.
- Il presenter agisce sia sui dati e sulla view, gestendone tutta la logica di presentazione.

A seconda delle implementazioni del *pattern* architetturale il livello di logica gestito dalla *view* può variare. Un implementazione estrema potrebbe essere quella in cui la

view risulta essere completamente passiva delegando tutte le operazioni al presenter. In questo caso quando un utente interagisce con la view, ad esempio facendo tap su un *Button,* la view stessa invoca un determinato metodo del presenter. Il presenter quindi recupera i dati che necessita dalla view attraverso dei metodi definiti dall'interfaccia implementata dalla view stessa. Infine, il presenter opera sul modello e aggiorna la view con il risultato delle operazioni eseguite. Solitamente viene definito un file denominato *Contracts* dove si definiscono le interfacce con i vari metodi astratti che dovranno essere implementati. Nella figura seguente viene mostrato un esempio di codice.

```
public interface DettaglioOperazioneContracts {

    interface View extends BaseView{
        Context getContext();
        Payment getTransaction();
        void showProgress();
        void hideProgress();
        void onResponseSplittedTransactions(List<Transaction> splittedTransactions);
    }

    interface ActionsListener extends InterfaceBusPresenter{
        void onEditClicked();
        void getSplittedTransaction(Context context, String transactionId);
    }
}
```

Figura 57: Esempio codice Applicazione Sviluppata

Abbiamo all'interno dell'interfaccia DettaglioOperazoneContracts a sua volta altre due interfacce. La prima ha tutti i metodi che saranno implementati dalla View mentre i metodi della seconda saranno implementati dal Presenter. Dal nome dei metodi è intuibile come i metodi *getSplittedTransaction(...)* e *onResponseSplittedTransaction(...)* siano connessi logicamente. In seguito a un determinato evento sulla View verrà invocato sull'istanza del presenter il metodo che richiede tutte le transazione suddivise. La comunicazione risulta essere asincrona poiché il presenter potrà recuperare i dati o in locale oppure appoggiandosi a un servizio REST. Nel momento in cui i dati sarannno pronti verrà invocato sulla View il metodo *onResponseSplittedTransaction(...)*, con le transazioni richieste che saranno cosi visibili all'utente. Questo esempio tende a

sottolineare come il codice risulti essere organizzato in maniera abbastamza ordinata permettendone una facile gestione .

6.1.1 Recupero dati

Come accennato precedentemente nel modello MVP il presenter è il componente che gestisce la logica dell'applicazione. Risulta, quindi, essere fondamentale il recupero dei dati che andranno ad aggiornare il modello e che verrano presentati nella view. Solitamente i dati vengono forniti da una sorgente locale o possono essere recuprati da una sorgenta esterna. Ovviamente quest'ultima soluzione andrà a incrementare la latenza tra la richiesta dei dati da parte della view e la loro presentazione. Per far sì che l'utente sia conscio del lavoro sottostante dell'App e di questa possibile latenza occorre inserire delle *progress-bar* o altri elementi tipici che indichino un caricamento. Questo permette di dare all'utente un feedback relativo al fatto che l'applicazione abbia ricevuto il comando e lo stia processando.

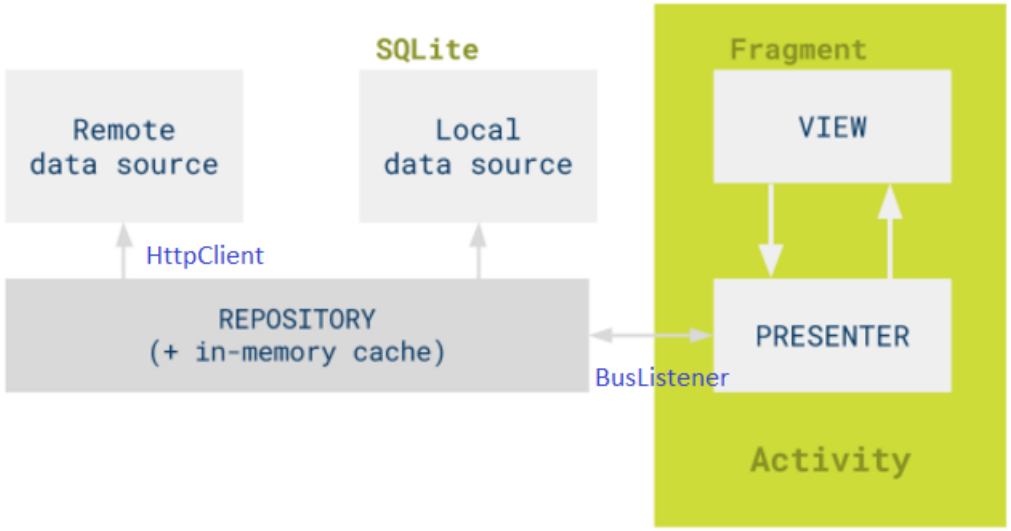

Figura 58: Esempio architettura MVP e *retrieve* dei dati

Dalla figura si nota come i dati potrebbero essere salvati in locale in delle tabelle di un DataBase SQLite oppure recuperati da una sorgente esterna. Nell' applicazione sviluppata si accede ai dati esterni tramite collegamento a un servizio di tipo Rest. Il Repository viene implementato tramite una classe che ha al suo interno una serie di

metodi che possono essere invocati dal presenter. Nel momento in cui il presenter sta per effetturare una richiesta al servizio tramite il Repository, istanzia un BusNetworkListener.

```
@Override
public void getSplittedTransaction(Context context, final String transactionId) {
    LogUtils.d(TAG, "getSplittedTransaction");
    mView.showProgress();
    new BusNetworkListener(this, mBusNetwork) {
        @DebugLog
        public void onEventMainThread(PfmSplitTransactionResponse response) {
            LogUtils.d("manu","eseguo il metodo getSplittedTransaction");
            mBusNetwork.unregister(this);
                   mView.hideProgress();
            PfmSplitTransactionResponse.Payload payload = response.getPayload();
            final List<Transaction> transactions = payload.getmTransactions();

            Runnable temp = () → {
                    mView.onResponseSplittedTransactions(transactions);
            };
            if (!isViewVisibile()) {
                enqueuePendingAction(temp);
            } else {
                temp.run();
            }
        }

        @DebugLog
        public void onEventMainThread(PfmGetTagsErrorNetwork pfmTagErrorNetwork) {
            mView.hideProgress();
            LogUtils.d(TAG, "tag response error network");
            mBusNetwork.unregister(this);
        }
    };
    Repository.getInstance().pfmgetSplittedTransaction(context, transactionId);
}
```

Figura 59: Esempio codice Applicazione sviluppata 2

Questo listener sarà fondamentale per poter ricevere la risposta contenente i dati oppure un eventuale errore. Dalla figura precedente si nota come prima di effettuare la chiamata e instanziare un BusNetoworkListener per la gestione della risposta, viene chiamato sulla view il metodo *showProgress(),* che fa apparrire un progress bar in modo da avvisare l'utente. Nel momento in cui la risposta arriva al Bus, si va a nascondere la progress bar con il metodo corriposnente della view *hideProgress()* e viene richiamato

il metodo *onResponseSplitTransactions(..)*. Quest'ultimo permette alla view di mostrare i dati all'utente. In caso di errore, viene comunque nascosta la progress bar e viene rimosso il listener. Per effettuare la richiesta verso il servizio si è fatto uso della libreria OkHttp. Essa fornisce un efficiente client per la gestione delle richieste. Nella classe OkHttpClient sono definiti i vari parametri per effettuare la connessione come ad esempio la versione del protocollo http usato, la gestione dei cookie e della sicurezza.

6.1.2 Gestione richieste verso il Servizio

Come accennato precedentemente la suddivisione del codice e dei diversi compiti tra presenter e view permette di avere una migliore gestione del codice. Analizzando nel dettaglio il flusso di ogni richiesta vediamo che il principio di divisione dei compiti viene associato anche a tutti gli altri componenti coinvolti nella gestione delle richieste.

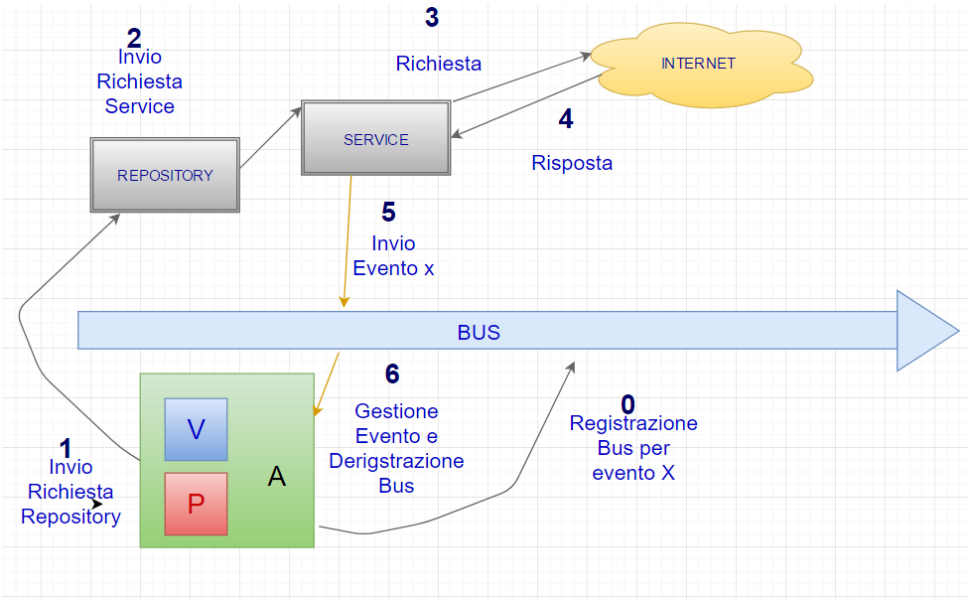

Figura 60: Esempio Richiesta Applicazione sviluppata

Dalla figura precedente si nota che prima di eseguire l'invio della richiesta occorre settare un BusNetworkListener. Occorre sottolineare come in questo modo il Bus si metta in ascolto di un determinato evento connesso alla richiesta. Solitamente il Bus

viene settato sia per poter gestire l'evento connesso a una risposta positiva sia un evento di errore. Vengono cosi settate altrettante *callback.* In questo modo, in seguito al verificarsi di uno o dell'altro evento verrà eseguita l'azione corrispondente. Dopo aver settato il Bus, il presenter invia la richiesta tramite il repository. Quest'utlimo conosce tutte le risorse che il REST Service espone. Tramite un'istanza della classe Service verrà eseguita la richiesta e si otterrà una risposta. Il Service inolterà la risposta direttamente verso il Bus scatenando un determinato evento. Se il Bus è stato configurato per la gestione dell'evento corrispondente il codice della *callback* correlata verrà eseguito. L'ultimo step nella gestone della richiesta consiste nella eventuale modifica del modello e di conseguenza della view da parte del presenter, nel caso la risposta contenga dei nuovi dati. Dopo aver gestito l'evento il Bus viene deregistrato. Ovviamente se si verifica un altro evento per cui il Bus non è stato configurato , nessuna *callback* verrà invocata .

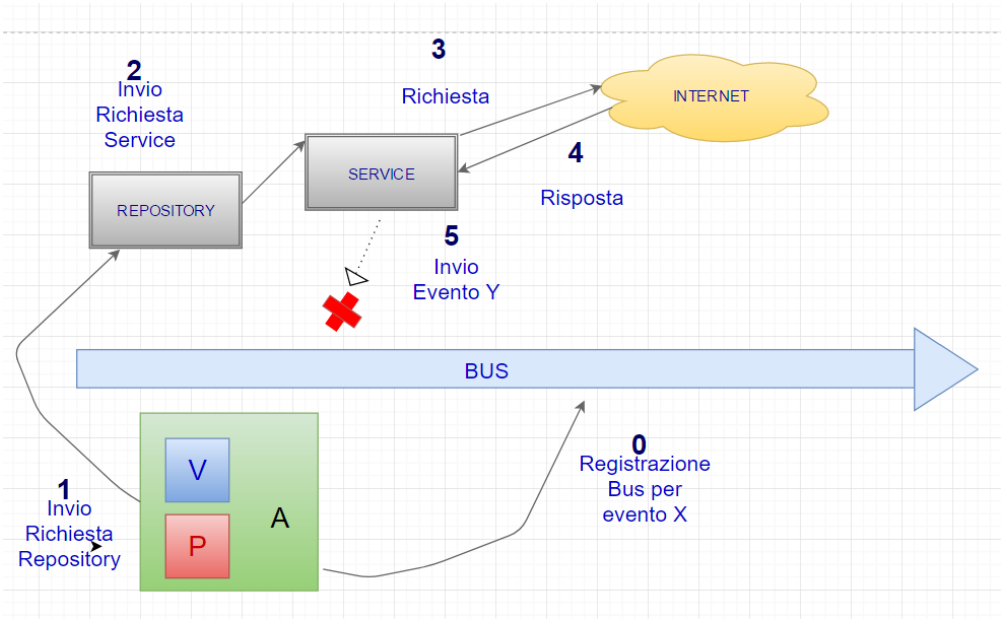

Figura 61: Esempio Richiesta Applicazione sviluppata 2

Nella figura precedente si può notare come il Bus sia stato registrato per la gestione del

solo evento X. In seguito alla richiesta effettuata tramite l'istanza della classe Service si verifica l'evento Y per il quale sul Bus non era stata configurata nessun tipo di gestione. Difatti dal punto di vista del Bus è come se l'evento non si sia mai verificato poiché semplicemente non risulta essere in ascolto dello stesso. Nulla di conseguenza verrà notificato verso il Presenter e la View.

6.2 Macro-Sezioni App

Dopo aver mostrato l'architettura della soluzione sviluppata verrà analizzata in dettaglio la soluzione stessa e come essa appare all'utente. Nei seguenti sotto capitoli verranno analizzate le sezioni tramite le quali l'App è articolata. È bene sottolineare che il REST Service al quale ci si appoggia ogni qualvolta vengono effettuate le chiamate è fornito da un fornitore esterno. I dati che vengono utilizzati per rappresentare ad esempio diversi profili utente con i corrispondenti movimenti sono dei *mock.* Questi ultimi sono dati simulati che riproducono il comportamento dei corrispondenti reali in modo controllato per facilitare il processo di *testing* e produzione.

6.2.1 Categorizzazione movimenti e spese

La prima Activity che si presenta all'utente dopo aver effettuato l'accesso mostra una sorta di resoconto riguardante la situazione dell'utente stesso. Nella parte alta dello schermo vengono mostrate tramite un *ViewPager* le informazioni generali per ognuno dei conti conessi all'utente. Tramite l'utilizzo del ViewPager l'utente tramite uno *scroll* orizzontale riesce ad accedere a un altro conto, se disponibile. Infatti questo componente di Android è utilizzato quando si devono mostrare *View* simili tra loro e passare velocemente dalla visulizzazione di una all'altra tramite semplici *Gesture*. Inizialmente il saldo connesso al conto è nascosto, ma tramite *tap* sull'icona a forma di occhio il saldo viene mostrato. Dalla figura seguente si può notare la sezione appena descritta.

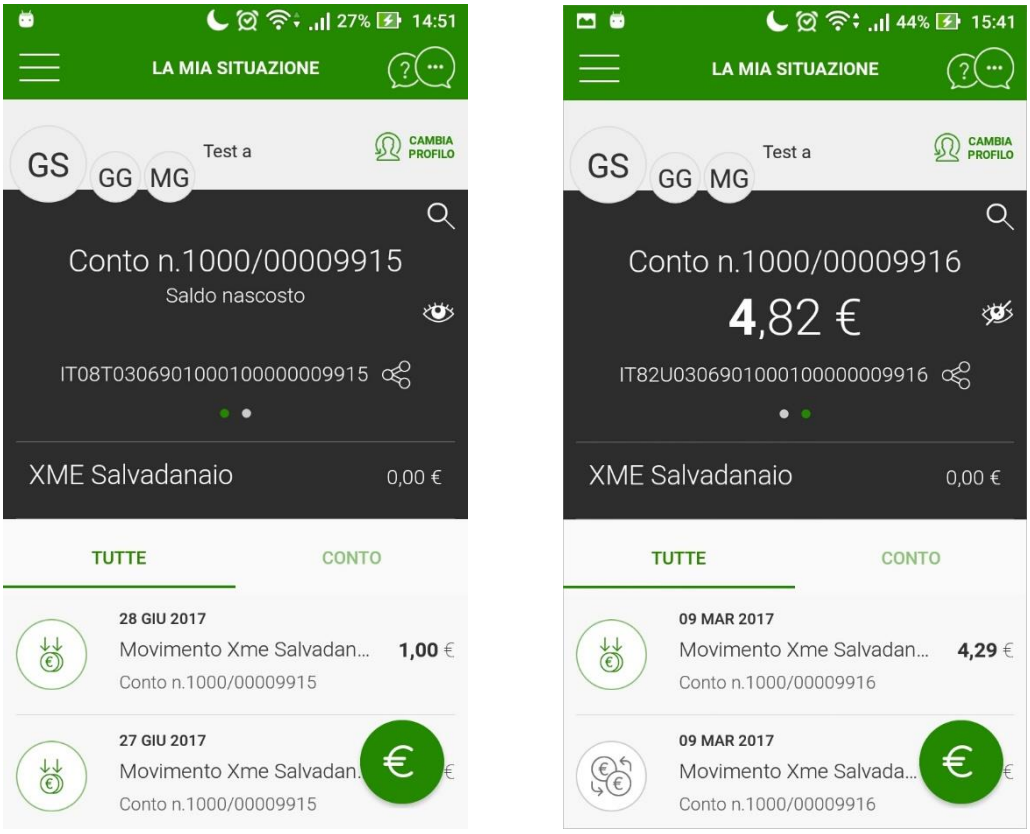

Figura 62: Screenshots Home Activity dettaglio ViewPager

Tramite *tap* sull'icona a forma di lente di ingrandimento si accede a delle informazioni piu dettagliate rigurado il conto che si sta visualizzando attualmente nel ViewPager. Qui il saldo del conto è sempre mostrato a differenza della visualizzazione nella prima Activity e tramite *tap* sul Button in alto a destra si possono condividere le informazioni mostrate. Infatti, dopo aver effettuato il *tap* sul Button si aprirà un menu che mostra tutte le Applicazioni presenti sul dispositivo che permettono di condividere un contenuto definito come *text/plain* ovvero un semplice testo. Android permette in maniera abbastanza semplice di fare ciò tramite descrizione di un Intent. Un Intent in Android è una descrizione astratta di un'operazione che deve essere effettuta. In questo caso viene definito un Intent implicito, ovvero si definisce l'azione tramite una costante. Nel nostro caso l'azione risulta essere ACTION_SEND. Infine occore definire un

Chooser, ovvero il menu che conterrà all'interno tutte le applicazioni che permettono una tale azione. Ad esempio in questo caso verranno mosrate tutte le applicazioni di e-mail, messagistica e alcuni social-network.

Figura 63: Screenshot dettaglio conto

Nella figura sottostante viene mostrata la porzione di codice utile a mostrare il menu che permette la condivisione. Si noti come a seconda della versione di Android, presente sullo smartphone in cui l'App viene eseguita, venga inserito un determinato flag altrimenti un altro. In particolare si ha un flag differenete se la versione risulta essere inferiore alla 5.0, un altro se maggiore o uguale a 5.0.

```
Intent sharingIntent = new Intent(android.content.Intent.ACTION_SEND);
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
    sharingIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_DOCUMENT);
} else {
    sharingIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
    sharingIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
}
sharingIntent.setType("text/plain");
sharingIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, subject);
sharingIntent.putExtra(android.content.Intent.EXTRA_TEXT, body);
startActivity(Intent.createChooser(sharingIntent, "Condividi"));
```

Figura 64: Codice Chooser Intent

Nella seguente figura vediamo invece il risultato finale mostrato all'utente. Con un semplice tap su una delle App proposte, potrà accedere a quella selezionata e condividere cosi il contenuto.

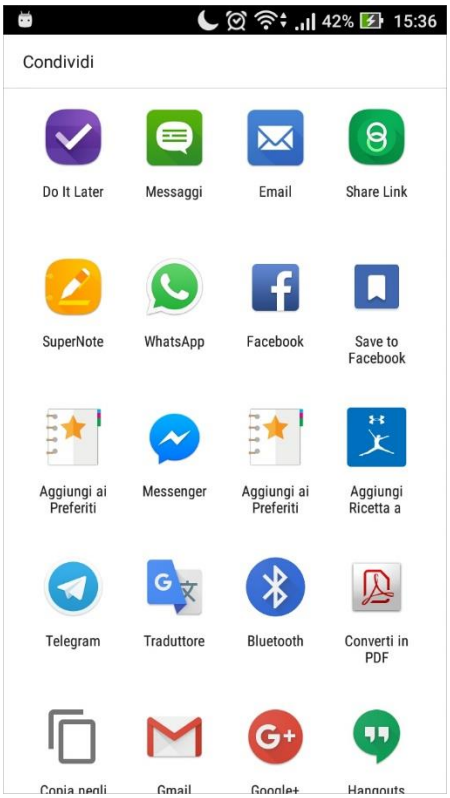

Figura 65: Screenshot Chooser Intent

Tornando all'analisi della prima Activity si nota come vi sia un Button per cambiare profilo. L'utente ha, infatti, la possibilità di avere conti in comune con altri utenti. In alto a sinistra infatti vengono mostrate delle View circolari con le iniziali dei vari utenti con i quali si condivide uno o piu conti all'interno di un determinato profilo.

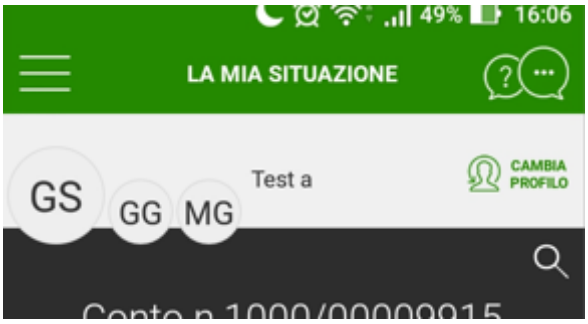

Figura 66: Dettaglio Screenshot Cambia Profilo

Tramite tap sul Button "Cambia Profilo", presente in alto a destra nella prima schermata, si accede così alla lista di tutti i profili disponibili.

Figura 67: Screenshot lista profili

Tramite “scroll” verso l’alto si accede alla sezione seguente al ViewPager. In particolare, vengono mostrate in un altro ViewPager connesso a un TabLayout le operazioni relative al profilo. Vi è così la possibilità di visualizzare solamente le operazioni relative al conto selezionato dal ViewPager precedente, oppure tutte le operazioni a prescindere dal conto al quale le operazioni stesse sono connesse. Una RecyclerView mostrerà tutte le operazioni descrivendole tramite un’icona, la data, il tipo di movimento, il numero di conto e l’importo.

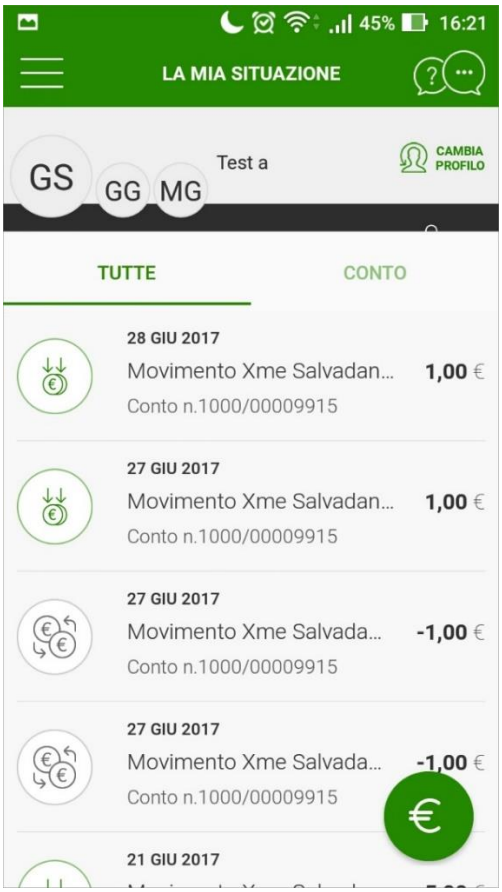

Figura 68: Screenshot lista operazioni

Dopo i primi 15 movimenti vi è la possibilità di visualizzare tutti i movimenti in una nuova Activity, con la possibilità di filtraggio per data e tipologia. Un esempio è mostrato nelle immagini seguenti.

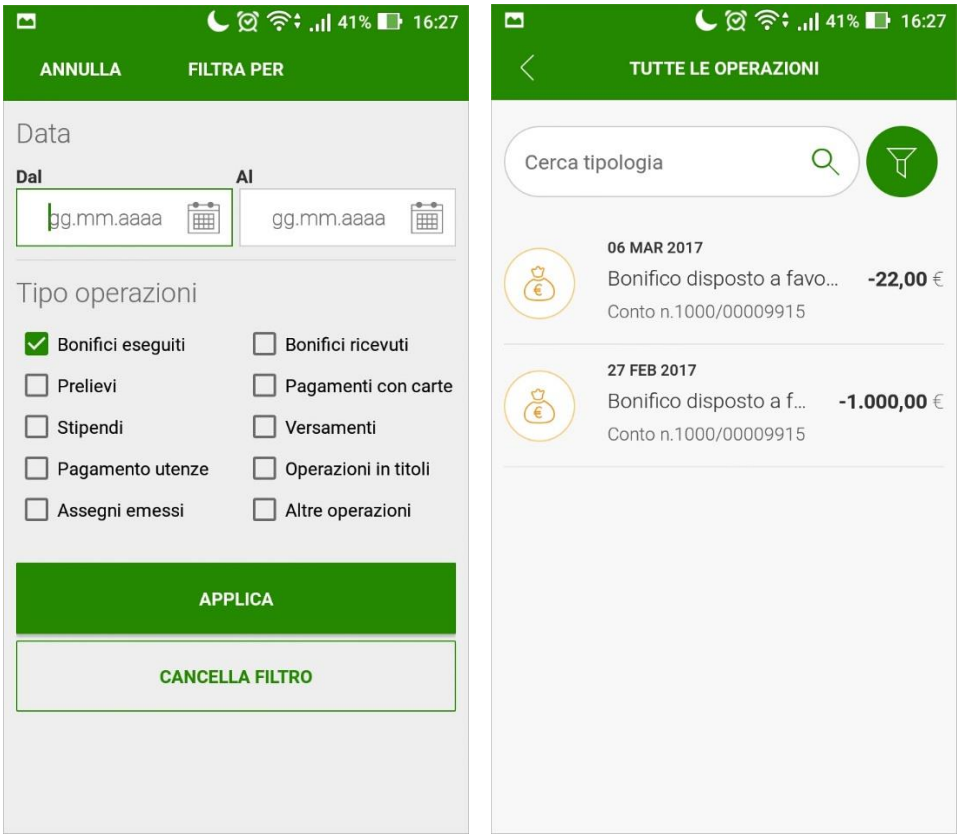

Figura 69: Screenshots filtra operazioni

La parte sottostante della prima Activity mostra informazioni relative alla spese del mese. Qui si farà uso della libreria grafica MPAndroidChart e verranno mostrate inoltre informazioni riguardo alcuni obiettivi, i quali potranno essere creati o modificati. Entrambe le sezioni verranno mostrate nei sotto-capitoli seguenti. Tramite il *Fab-Button,* presente in basso a destra, è possibile effettuare alcune operazioni relative al tipo particolare di conto, come ad esempio un nuovo movimento.

6.2.2 Visualizzazione dettagli e modifica movimenti

Come analizzato in precedenza l'utente ha la possibilità, non appena aver avuto accesso all'applicazione, di visualizzare la lista dei movimenti di un conto in particolare o di un profilo in generale. Effettuando un *tap* su uno degli elementi della lista, si accede alla visualizzazione in dettaglio del movimento. La figura successiva mostra come venga mostrata all'utente ogni operazione.

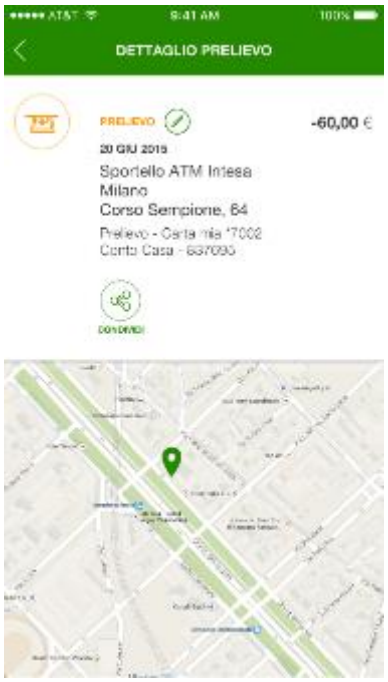

Figura 70: Screenshot dettaglio operazione

In particolare, viene mostrato il tipo di operazione, la data, l'importo e il luogo presso cui è avvenuta l'operazione se disponibile. In questo caso viene mostrato tramite l'ausilio di una MapView il luogo stesso. Come in precedenza anche qui vi è la possibilità di condivisione del contenuto mostrato nella Activity. Tramite *tap* sul Button raffigurante una matita vi è la possibilità di modificare l'operazione. La modifica dell'operazione permette di cambiarne prima di tutto la categoria. In particolare viene data possibilità all'utente di scegliere tra le 3 categorie più probabili o più utilizzate dall'utente, oppure scegliere accedendo a un altro Fragment dove sono elencate tutte la categorie disponibili che vengono suddivise in macro e micro categorie. Inoltre, vi è la possibilità di suddividere il movimento stesso in più sotto-movimenti. In particoalre si accede a un Fragment dove viene mostrato il movimento suddiviso in due movimenti "figli", dove il primo è creato con importo pari al totale mentre il secondo con importo pari a 0, ma con uguale categoria. Del primo movimento "figlio" è modificabile solo la

descrizione. Vi è comunque la possibilità di suddividere in più transazioni "figlie" modificando la categoria e associando una o piu etichette fino a un massimo di 5 a ognuno dei sotto-movimenti. L'importo di ogni sotto-movimento, dal secondo in poi, è modificabile tramite inserimento diretto sull'EditText o tramite *tap* sui Button più e meno che permettono di variare l'importo di un euro alla volta andando a modificare contemporaneamente l'importo padre. Ogni qualvolta si modifica il prezzo le barre laterali arancioni e grigie vengono colorate in proporzione all'importo totale dell'operazione. Come accennato in precedenza viene fatto uso delle etichette. Queste etichette sono delle view personalizzate che estendono un RelativeLayout. In particolare, a differenza dell'approccio standard che presuppone l'accesso al *graphic engine* di Android, tramite un oggetto di tipo Canvas, in questo caso si è scelto di ricorrere all'inflate di un layout descritto tramite XML. Dalla figura sottostante viene definita la modifica di un'operazione "prelievo" che viene suddivisa in due sotto-movimenti.

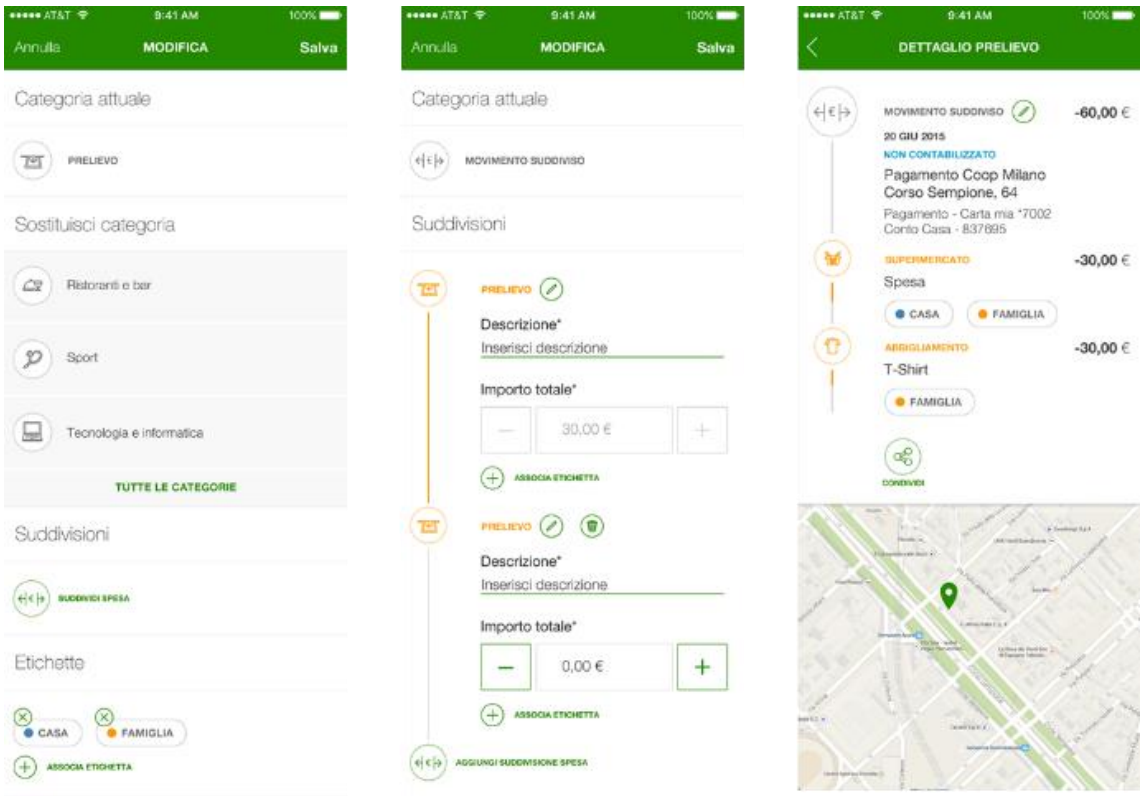

Figura 71: Screenshots modifica operazione

6.2.3 Grafici spese

Come accennato in precedenza, se si analizza nuovamente la Home Activity, si può vedere come nella parte finale della schermata sia possibile verificare il totale delle proprie entrate. Inoltre, si può accedere al dettaglio riguardo le spese mensili. Viene indicato, infatti, di portare il dispositivo in modalità Landscape, ovvero ruotato di 90°.

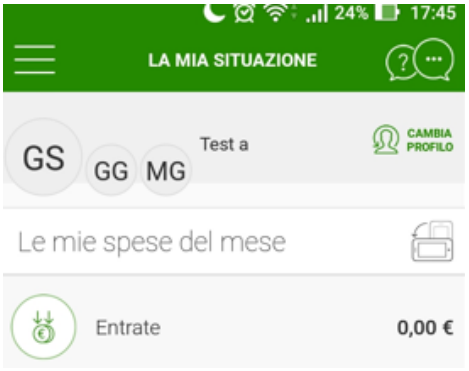

Figura 72: Screenshot dettaglio spese

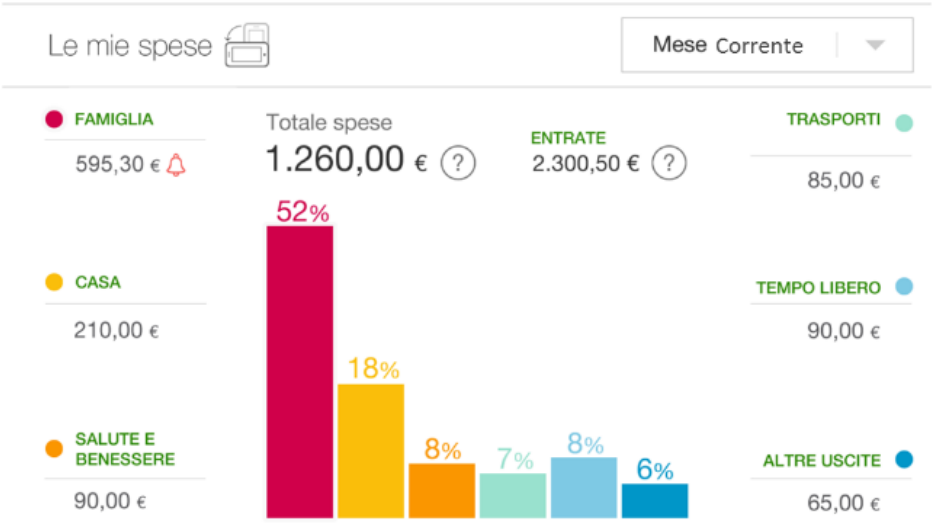

Figura 73: Screenshot grafici spese mensili

Portando, quindi, in Landscape lo smartphone abbiamo dei grafici sviluppati tramite la

libreria MPAndroidChart. Nella parte centrale viene definito un BarChart raffigurante le macro-categorie di spesa con la percentuale corrispondente, mentre ai lati vengono mostrate nel dettaglio con il relativo importo. Vi è la possibilità di accedere alle spese del mese corrente , del mese precedente , degli ultimi 30 giorni oppure degli ultimi 90, tramite uno spinner in alto a destra. Facendo *tap* su una barra del BarChart, si accede al dettaglio della macro-categoria corrispondente. Dalla figura successiva si nota come si abbia sempre, nella parte centrale, il grafico mentre all'esterno i dettagli. In questo caso viene mostrato un grafico a torta colorato in proporzione alle spese effettuate in ogni categoria, attorno la percentuale di ogni micro-categoria e all'interno il nome della macro-categoria e l'importo totale di spesa della macro-categoria stessa. Nella parte esterna si ha invece il nome della micro-categoria , il relativo importo e l'icona corrispondente.

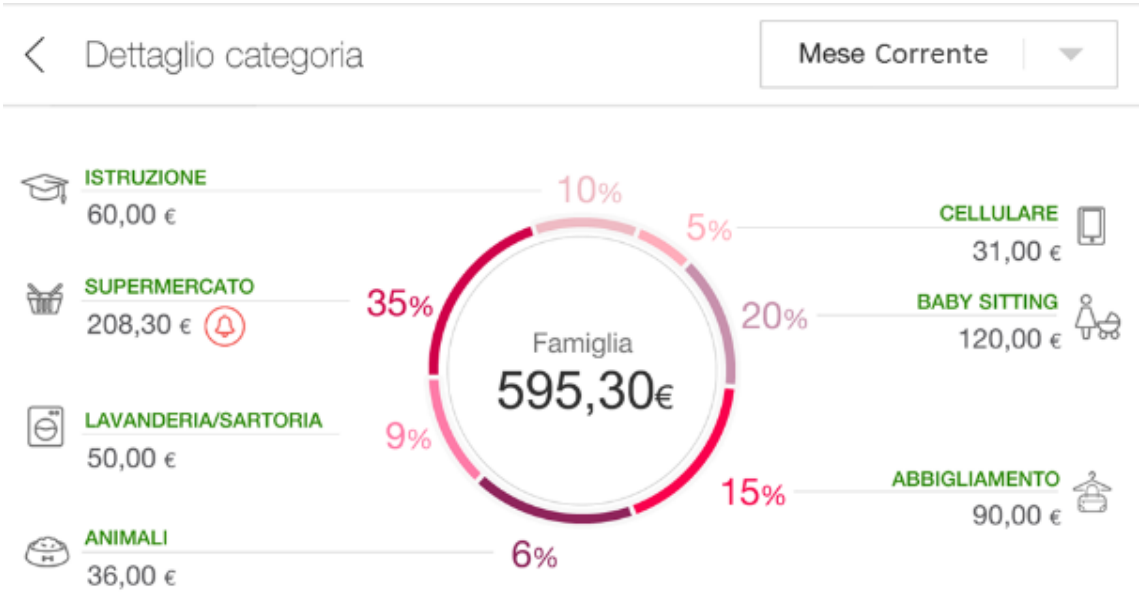

Figura 74: Screenshot grafici spese mensili micro-categoria

6.2.4 Gestione degli obiettivi

Nella parte finale della Home Activity si la sezione relativa agli obiettivi. In particolare, se presenti, vengono visualizzati tre obiettivi e tramite *tap* sulla TextView si accede alla visualizzazione di tutti gli obiettivi.

Figura 75: Screenshot Home Activity sezione obiettivi

Nella figura precedente viene mostrato un nuovo obiettivo per il quale ancora non vi siano stati accantonamenti. Gli obiettivi si differenziano in obiettivi reali e virtuali. I primi vengono creati e collegati a un determinato conto dell'utente. In questo caso gli accantonamenti riguarderanno il denaro "prelevato" dal conto stesso e che è stato collegato nel momento della sua creazione e "depositato" come accantonamento per il raggiungimento di un certo obiettivo. Gli obiettivi, virtuali, invece possono essere considerati come dei veri e propri promemoria per l'utente. Nel momento in cui sono creati non sono connessi a nessun conto dell'utente, e quindi i vari accantonamenti non

vanno a diminuire il saldo disponibile da nessun conto. Infatti si presuppone che questi accontanamenti avvengano tramite denaro non appartenente a nessuno dei conti dell'utente che vengono gestiti dall'applicazione. Per differenziare a livello grafico gli obiettivi reali dagli obiettivi virtuali vengono rappresentati in maniera diversa come si può evincere dalle figure succesive.

Figura 76: Screenshot obiettivo virtuale

Figura 77: Screenshot obiettivo reale

Per la visualizzazione degli obiettivi virtuali viene utilizzata una Custom View. In questo caso si sono dovuti gestire i metodi *onMesure(...)* e *onDraw(...).* In particolare nel primo metodo si va a negoziare col proprio container lo spazio che la view necessita mentre nel secondo tramite un oggetto Canvas si accede al *graphic engine 2D* di Android e si riesce a disegnare i vari componenti della view. Dal codice mostrato nella seguente figura vediamo i due metodi *drawArc(..)* invocati sull'oggetto canvas.

```
@Override
protected void onDraw(Canvas canvas) {
    if (!mClockwise) {
        canvas.scale(-1, 1, mArcRect.centerX(), mArcRect.centerY());
    }

    // Draw the arcs
    final int arcStart = mStartAngle + mAngleOffset + mRotation;
    final int arcSweep = mSweepAngle;
    canvas.drawArc(mArcRect, arcStart, arcSweep, false, mArcPaint);
    canvas.drawArc(mArcRect, arcStart, mProgressSweep, false,
            mProgressPaint);
```

Figura 78: Codice onDraw Custom View obiettivi virtuali

Per indicarne il progresso verrà modificata la lunghezza dell'arco colorato a seconda del rapporto versato rispetto al totale dell'obiettivo. Per quanto concerne gli obiettivi reali, si utilizza un insieme di oggetti di tipo Drawable. Questi oggetti sono una rappresentazione astratta di un qualunque contenuto grafico. Infatti, vengono utilizzati ogni qual volta si ha a che fare ad esempio con file di tipo *png* o *jpeg*. In questo caso sotto la cartella *res/drawable* abbiamo vari elementi di tipo *png* che rappresentano 7 stadi di una piantina. A seconda del progresso dell'obiettivo verrà settata una particolare immagine per indicare all'utente in maniera intuitiva lo stato dell'obiettivo. Accedendo alla scheramta che contiene la lista di tutti gli obiettivi si ha la possibilità di inserire un nuovo obiettivo o tramite *tap* su un obiettivo esistente lo si può modifcare.

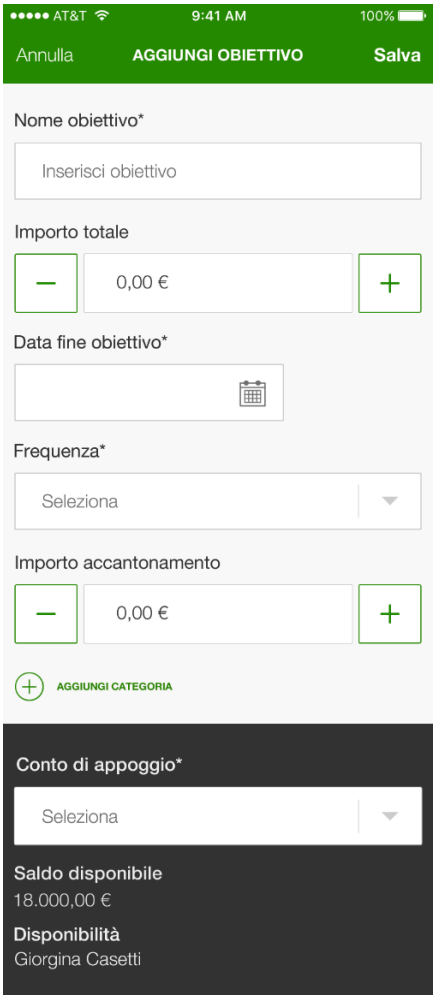

Figura 79: Screenshot Aggiunti Obiettivo

Dalla figura vediamo che si può settare il nome dell'obiettivo, l'importo totale, la data, l'importo dell'accantonamento e la frequenza stessa. Nella parte finale della scheramta vediamo che si può selezioanre uno dei conti dell'utente come conto di appoggio dal quale verrà detratta la quantità accantonata. Nel caso in cui non si selezioni nessun conto l'obiettivo creato è di tipo virtuale. Tramite *tap* su un obiettivo vediamo che si ha la possibilità di avere informazioni dettagliate sull'obiettivo.

Figura 80: Screenshot Dettaglio Obiettivo

Nella parte alta viene mostrata una Label che indica se l'utente è in linea con il suo obiettivo a seconda del rapporto tra gli accantonamenti fatti dal momento dell'inserimento dell'obiettivo fino alla data attuale e del tempo rimanente. Nel caso in cui la media di accantonamenti giornalieri è maggiore di quella necessaria, considerando i giorni disponibili prima della data di fine dell'obiettivo, l'utente risulta essere in linea. Nella parte bassa della schermata abbiamo la possibilità di eliminare l'obiettivo, condividerlo, modificarlo o versare un determinato importo che non incide sugli accantonamenti periodici ma va semplicemente a sommarsi a questi ultimi. Nel momento in cui l'utente voglia effettuare un accantonamento, dopo aver effettuato un *tap* sul Button "versa", vediamo che si aprirà un nuovo *Fragment.* All'interno di questo *Fragment* sarà presente la stessa Custom View utilizzata per indicare un obiettivo virtuale. In questo caso vediamo che l'utente può versare un importo tramite *touch* e/o

trascinamento del pallino verde che può scorrere lungo l'arco. Nella figura seguente si vede un esempio di versamento su un obiettivo.

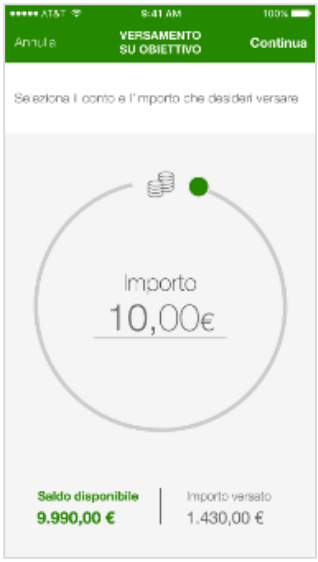

Figura 81: Screenshot Versamento Obiettivo

In questo caso vediamo che sarà gestito l'evento *onTouchEvent* sulla Custom View. In particolare vediamo che l'*onTouchEvent* riceve dal sistema come parametro un oggetto di tipo *MotionEvent* che al suo interno ha tutte le informazioni riguardanti il tocco come ad esempio la posizione espressa tramite X e Y. Accedendo a queste variabili si calcola la nuova posizione corrispondente del pallino verde sull'arco, a partire dall'angolo che il punto toccato va a formare rispetto all'origine dell'arco in grigio e il centro della Custom View.

```
private double getTouchDegrees(float xPos, float yPos) {
    float x = xPos - mTranslateX;
    float y = yPos - mTranslateY;
    //invert the x-coord if we are rotating anti-clockwise
    x = (mClockwise) ? x : -x;
    // convert to arc Angle
    double angle = Math.toDegrees(Math.atan2(y, x) + (Math.PI / 2)
            - Math.toRadians(mRotation));
    if (angle < 0) {
        angle = 360 + angle;
    }
    angle -= mStartAngle;
    return angle;
}
```

Figura 82: Codice calcolo versamento

Nella porzione di codice presente nella figura precedente si mostra in che modo avviene il calcolo dell'angolo. A seconda quindi dell'angolo spaziato si aumenta la lunghezza dell'arco verde e si calcola il valore del versamento tramite il rapporto tra la lunghezza dei due archi. Infine, il valore dell'importo sarà mostrato all'interno della Custom View.

CAPITOLO 7

CONCLUSIONI

A conclusione di questa tesi, in cui si è sviluppato un PoC (*Proof of Concept)* di un'applicazione Android per Personal Financial Management, è bene sottolineare quali siano i punti di forza della soluzione proposta, i risultati ottenuti e quali invece i possibili sviluppi futuri. Gli obiettivi prefissati erano quelli di definire una applicazione intuitiva per l'utente. Tramite l'utilizzo della libreria grafica MPAndroidChart e di diverse Custom View durante il lavoro svolto si è riuscito nel portare a termine questo obiettivo. Infatti, tramite l'utilizzo di questi componenti non nativi di Android, utilizzati insieme ai *widget* che il sistema fornisce di default, l'Applicazione risulta essere molto "espressiva" e facile da utilizzare per l'utente. Specialmente l'utilizzo della libreria esterna ha permesso di creare facilmente dei grafici riassuntivi utilissimi per quanto riguarda ad esempio la categorizzazione delle spese. Tra le varie Custom View utilizzate, come mostrato nel capitolo 6 di questa tesi, quella utilizzata durante i versamenti sugli obiettivi, oltre a essere intuitiva, è anche comoda per l'utente che non deve effettuare un inserimento tramite tastiera numerica, ma semplicemente può decidere l'importo tramite un determinato movimento lungo l'arco che in seguito verrà colorato in proporzione all'importo massimo versabile. L'implementazione di questa Custom View mi ha portato a dover gestire anche a basso livello le *Gesture,* ovvero i tocchi e i movimenti dell'utente sullo schermo. Dal punto di vista architetturale l'utilizzo del *pattern* MVP, mi ha permesso di organizzare il codice in maniera abbastanza semplice, facilitando le correzioni durante lo sviluppo, e di renderlo adatto ad eventuali sviluppi futuri. Infatti, la suddivisione dei compiti tra i vari componenti

permette di modificare ad esempio solamente il lavoro fatto in background dall'applicazione nel *Presenter* senza modificare ciò che viene mostrato all'utente nella *View*. Il PoC sviluppato permette, infine, all'utente di visualizzare informazioni sui propri conti, tutti i movimenti, le spese e i vari obiettivi. Avendo analizzando le altre soluzioni presenti sul mercato, ciò che rende funzionale un'applicazione per Personal Financial Management è proprio la possibilità di poter analizzare in che modo si spenda il proprio denaro. Come mostrato nel capitolo precedente, non appena si fa accesso alla prima Activity, ruotando lo schermo e portandolo in modalità *Landscape,* è possibile visualizzare tutte le spese suddivise per categoria tramite ausilio di grafici. Ovviamente un altro aspetto importante è quello di poter verificare tutti i movimenti effettuati e pianificare eventuali spese future. Per quanto riguarda i movimenti, tramite il PoC sviluppato, l'utente oltre a visualizzare in dettaglio tutti i movimenti può decidere di modificarli e suddividerli in sotto-movimenti. In questo modo l'utente ha il controllo di tutte le proprie spese potendo aumentare il dettaglio e la categorizzazione delle stesse. La gestione degli obiettivi, permette anche di definire eventuali spese future. Ciò può aiutare l'utente nel portare a termine i propri intenti finanziari. L'applicazione, infatti, oltre a permettere di settare un accontamento periodico, indica anche in che stato risiede l'obiettivo rispetto al denaro versato in rapporto al totale dell'obiettivo stesso. Nel caso in cui l'utente sia non in linea con l'obiettivo o comunque voglia versare dell'altro denaro, l'applicazione permette di fare ciò in maniera semplice e diretta come mostrato nel capitolo precedente. Per quanto concerne possibili sviluppi futuri si potrebbe inserire una sezione riguardante in che modo l'utente possa incrementare le proprie entrate, ossia ad esempio una sezione dedicata a dei possibili investimenti. Per concludere, questo lavoro mi ha portato a incrementare le mie conoscenze riguardo il mondo Android e lo sviluppo di un'Applicazione Mobile, facendo sì che mi appassionassi alla risoluzione di problemi tipici dello sviluppo di software Mobile. In particolare, sviluppare codice prendendo in considerazione l'*user-experience,* ovvero tutto ciò che è legato alle sensazioni che l'utente può provare utilizzando un'applicazione, mi ha permesso di sviluppare e utilizzare un metodo di lavoro e analisi molto preciso. Infatti, specialmente nell'ambito Mobile, occorre sviluppare avendo da subito in mente, oltre alle funzionalità che si vogliono fornire, anche in che modo le si

vogliano mostrare all'utente. Di certo risulta molto vantaggioso, capire in che modo facilitare l'interazione tra quest'ultimo e l'Applicazione, affinché, si renda il più piacevole possibile l'utilizzo della stessa.

RINGRAZIAMENTI

Ringrazio anzitutto il Chiar.mo Professor Giorgio Bruno, mio Relatore, Luca Perino, mio Tutor Aziendale, per la loro gentile disponibilità nel seguirmi in questo percorso di Tesi. Un ringraziamento particolare va a tutti coloro che mi hanno incoraggiato e sostenuto durante il mio percorso accademico.

SITOGRAFIA

[1] https://it.wikipedia.org/wiki/Apple

[2] https://it.wikipedia.org/wiki/Smartphone

[3] http://tech.everyeye.it/articoli/speciale-smartphone-vs-pc-internet-più-telefoni-che-computer-fine-un-epoca-31381.html

[4] https://en.wikipedia.org/wiki/Mobile_banking

[5] https://en.wikipedia.org/wiki/Mobile_payment

[6] https://en.wikipedia.org/wiki/Personal_Financial_Management

[7] http://www.lastampa.it/2017/04/04/economia/debutta-in-italia-oval-money-lapp-che-vuole-insegnarci-a-risparmiare-ZOl95SKYAcZ9TlJkH kWa1O/pagina.html

[8] http://www.lastampa.it/2017/01/17/tecnologia/news/n-la-banca-per-smartphone-arriva-anche-in-italia F9IfH1yedvgyfMPj7i6i7O/pagina.html

[9] https://github.com/PhilJay/MPAndroidChart/wiki

[10] http://www.android-graphview.org/

[11] https://github.com/halfhp/androidplot/blob/master/docs/index.md

[12] https://github.com/ddanny/achartengine

[13] http://www.object-refinery.com/orsoncharts/android/index.html

[14] http://docs.telerik.com/devtools/android/introduction

[15] https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93presenter

[16] https://github.com/googlesamples/androi d-architecture/tree/todo-mvp

BIBLIOGRAFIA

[1] *Smartphone* History: Evolution & Revolution - Bruce Taplin

[2] Banking in a digital world: how will customers interact with you in the future - Ian Goodliffe

ELENCO FIGURE