

UNIVERSITÀ DEGLI STUDI DI TORINO
DIPARTIMENTO DI INFORMATICA

CORSI DI LAUREA IN INFORMATICA



TESI DI LAUREA IN

INFORMATICA

GestApp: un gestionale aziendale per la piattaforma Android

Relatore:

Prof. GIOVANNA PETRONE

Candidato:

MOHAMED EDDAAKOURI

Sessione

a.a. 07/2016

Ringraziamenti

Un ringraziamento speciale va alla mia famiglia che mi ha sempre sostenuto in tutti questi anni, a mia madre che mi ha sempre motivato a dare il meglio in ogni cosa, ai miei fratelli che sono sempre stati presenti, ai miei amici e colleghi che mi sono sempre stati di ispirazione, a Piero per essere un grande mentore per me, ai membri della MC Team s.a.s per la loro pazienza, alla mia relatrice per essere sempre stata disponibile e in ultimo, ma non per importanza, a Arianna per essere sempre stata al mio fianco ed avermi sostenuto in ogni momento.

Indice

1	Introduzione	5
1.1	Obiettivo	5
1.2	Contenuto	6
2	Android	7
2.1	Stroria	7
2.2	Il primo hardware	8
2.3	Evoluzione	9
2.4	Diffusione	11
2.5	La frammentazione	12
2.6	“Android is about freedom...”	13
2.7	Android Core	14
2.7.1	Linux Kernel	15
2.7.2	Libraries	16
2.7.3	Android Runtime	17
2.7.4	Application Framework	17
2.7.5	Applications	18
2.8	Application Anatomy	18
2.8.1	Componenti principali	18
3	Architettura	27
3.1	Architettura del sistema	27
3.1.1	REST	28
4	GestApp overview	33
4.1	GesTeam	33
4.1.1	Scopo dell'applicazione	34
4.2	GestApp	35
4.3	Ruoli e funzioni	35
4.4	Diagramma casi d'uso	37
4.5	Funzionalità e design	38
4.5.1	Login	38
4.5.2	Registrazione	41

4.5.3	Home	43
4.5.4	Tools	43
5	Tecnologie utilizzate	54
5.1	Lato server	54
5.1.1	Slim Framework	54
5.1.2	IDIORM	55
5.2	Lato client	56
5.2.1	Librerie	56

Elenco delle figure

2.1	HTC Dream il primo hardware con sistema operativo Android	8
2.2	Evoluzione Android	10
2.3	Market share delle piattaforme mobile dal 2007 al 2013	12
2.4	Gap di prezzo tra un dispositivo Android e IOS	13
2.5	Controfoto frammentazione Android vs IOS	14
2.6	Android core stack	15
2.7	Le componenti Android	19
2.8	Ciclo di vita di un'Activity	21
2.9	Le principali componenti View	23
3.1	Architettura di una RESTful API	27
3.2	Operazione di visualizzazione della representation della risorsa login	30
3.3	Creazione di una nuova risorsa Societa con il metodo HTTP PUT	31
3.4	Eliminazione di una risorsa società con il metodo HTTP DELETE	32
4.1	GesTeam applicazione web based home page	33
4.2	Funzioni di un amministratore	36
4.3	Funzionalità commerciale	37
4.4	Funzionalità gestionale	37
4.5	Funzionalità consulente	38
4.6	Diagramma casi d'uso	39
4.7	Login GestApp	40
4.8	UML Sequence Diagramm Login	41
4.11	Suddivisione dei moduli	44
4.14	Fasi avanzamento commesse	46
4.15	Ricerca Avanzata sulle commesse	47
4.16	Creazione nuova associazione con scelta multipla	47
4.17	Strumento Allegati	48
4.18	Suddivisione in mesi	48
4.19	Fragment relativo al mese di Luglio	50
4.21	Report mensile su una commessa nel mese di Luglio	53

Capitolo 1

Introduzione

La tesi tratta principalmente dello sviluppo di un'applicazione per la piattaforma Android presso la MCTeam s.a.s. L'applicazione sviluppata è una versione mobile della già esistente versione web based denominata GestTeam.

1.1 Obiettivo

La MC Team s.a.s è un'azienda che offre servizi di consulenza nel settore dell'Information Technology. La necessità di avere una versione mobile nasce dal bisogno di interagire costantemente, in qualsiasi luogo e momento, con i dati aziendali dei collaboratori esterni offrendo la possibilità di gestire i propri dati e le proprie attività, come la possibilità di gestire le commesse che sono state assegnate o la possibilità di consuntivare le attività svolte su una commessa nel momento in cui essa viene svolta. L'applicazione mobile nello specifico deve permettere da remoto:

- La gestione dei dati relativi alle società, agli utenti e alle commesse;
- La gestione dei allegati aziendali;
- La gestione delle offerte su commesse, degli ordini aziendali e i relativi stadi di avanzamento;
- La gestire della documentazione amministrativa interna e la creazione dei relativi report;
- La creazione di consuntivi giornalieri;
- Il monitoraggio l'avanzamento delle commesse;
- La gestione dei cicli passivi ed attivi aziendali.

1.2 Contenuto

Il **capitolo 1 - Android** è un'introduzione al mondo di Android, dalla sua nascita alla sua diffusione a livello globale, senza tralasciare la parte tecnica su questo sistema operativo.

Nel **capitolo 2 - REST** si parla dell'architettura REST, di come la comunicazione avviene tra il client(GestApp) e il server(MCTeam Web Service).

Nel **capitolo 3 - GestApp Overview** si fa una panoramica (overview appunto) sull'applicazione sviluppata, le funzionalità implementate e le scelte di design.

Nel **capitolo 4 - Tecnologie utilizzate** si parla delle tecnologie utilizzate nella realizzazione dell'applicazione e delle librerie usate per lo sviluppo del software lato client.

Capitolo 2

Android

Per comprendere meglio il cuore della tesi, è quasi doveroso fare una piccola introduzione su Android. Questa introduzione ha il preciso scopo di illustrare il contesto in cui nasce questo sistema operativo, cosa ha reso possibile il suo successo e la sua successiva diffusione fino a renderlo il sistema operativo per smartphone più diffuso al mondo con più di un miliardo di dispositivi attivi.

2.1 Stroria

Secondo gli standard odierni, gli smartphone della metà degli anni 2000, risultano lenti e “brutti”. Il mercato era dominato da Symbian, Windows Mobile e BlackBerry. Gli smartphone non erano soltanto basic da un punto di vista tecnologico, cioè con le funzionalità di un feature phone¹, ma erano anche un campo minato per gli sviluppatori, e in molti mercati incombevano le restrizioni da parte delle compagnie telefoniche. L’User Experience, una semplificazione e supporto nello sviluppo da terze parti erano di secondaria importanza per le case produttrici concorrenti.

All’epoca lo sviluppo di software e applicazioni per dispositivi mobile non era molto attraente per i programmati. Uno delle ragioni principali era la sua scarsa redditività, anzi spesso si incorreva più in perdite che in guadagni. Non esisteva una standardizzazione sul mercato. Virtualmente ogni smartphone eseguiva un software e applicazioni differenti: un software scritto per uno smartphone *Samsung* molto spesso non veniva eseguito su un *Motorola* o *Nokia*. Le piattaforme software erano incompatibili perfino tra dispositivi della stessa compagnia, per esempio esistevano differenti versioni di *Symbian*².

¹cellulare evoluto, ma privo delle funzionalità avanzate degli smartphone

²**Symbian** OS è un sistema operativo per dispositivi mobili, prodotto da Symbian Foundation.

Android prende vita nell'ottobre del 2003 con il nome di Android Inc., fondata da Andy Rubin, Rich Minerva, Nick Sears e Chris White con l'intento di creare ciò che Rubin definì “dispositivi cellulari più consapevoli della posizione e delle preferenze del loro proprietario”. Con un piccolo team di ingegneri ed un piano per fondare la prossima generazione di software per smartphone, la società si è focalizzata su un’evoluzione *open source* di alcune idee concepite da Rubin nell’epoca in cui lavorava per la Danger (DangerOS era un sistema operativo per gli smartphone sviluppati da Danger ed erano largamente basato su linguaggio Java).

Focalizzandosi sulla migliore esperienza di connessione web possibile e sulla creazione di un ambiente di sviluppo con cui qualunque sviluppatore potesse lavorare senza incorrere in difficoltà e restrizioni, Android aveva un visione ed un piano solido. La società ha operato in segreto fino al 2005, quando il progetto venne presentato ai possibili investitori.

Page e Brin, fondatori di Google, avevano una visione per il futuro: avere più dispositivi mobili con i servizi di Google e un ’*open platform* come Android offriva esattamente questo. Alla fine del 2005 Android Inc. venne ufficialmente acquisita da Google.

2.2 Il primo hardware



Figure 2.1: HTC Dream il primo hardware con sistema operativo Android

Mentre molti ricordano l'**HTC Dream** (in USA e Canada conosciuto come **T-Mobile G1**) come il primo smartphone con il robottino verde, con una tastiera QUERTY e uno schermo touchscreen, questo era solo uno dei tanti design che Google prese in considerazione con il partner HTC.

Tra i vari prototipi che Google scartò prima del HTC Dream, il più conosciuto era senza dubbio il **Sooner**. **Sooner** era il risultato della visione combinata di Google e Andy Rubin di quello che un dispositivo Android poteva essere. Seguiva la moderna idea di quello che uno smartphone doveva somigliare. **Sooner** fu costruito in collaborazione con HTC, il suo design era molto somigliante ai dispositivi BlackBerry dell'epoca (i dispositivi BlackBerry erano considerati il top della gamma sia per la produttività che per il design).

Mentre Google ed il team di Rubin lavoravano incessantemente al conseguimento di ciò che essi credevano essere il dispositivo rivoluzionario che avrebbe potuto cambiare il mondo dei dispositivi mobile per sempre, nello stesso periodo stava nascendo il dispositivo che avrebbe rivoluzionato per sempre il modo in cui percepiamo ed interagiamo con i smartphone: l'**Iphone**. Nel Gennaio del 2007 Steve Jobs sale sullo stage per svelare l'Iphone. La reazione istantanea e viscerale di Chris DeSalvo, uno degli ingegnere di Google, fu: “Come consumatore ero meravigliato. Ne volevo uno immediatamente. Ma come ingegnere Google, ho pensato ‘Dobbiamo ricominciare.’”

Il rilascio inaspettato dell'**Iphone** ha ritardato di ben due anni il lancio del primo dispositivo Android. Il prototipo Sooner era stato accantonato perché Google voleva un prodotto che potesse competere con l'**Iphone**. Il “robotino verde” fu ufficialmente presentato il 5 novembre 2007 dalla neonata **OHA** (*Open Handset Alliance*): un accordo di differenti compagnie con Google come capofila, ASUS, HTC, Intel, Motorola, Qualcomm, T-Mobile, Samsung e NVIDIA il cui obiettivo è sviluppare “standard aperti” per dispositivi mobili.

Diversi prototipi furono progettati e rigettati prima che il HTC Dream fosse ultimato e commercializzato nel 2008. Il software del **HTC Dream** apportò alcune feature agli smartphone che non erano mai state realizzate, o almeno non così bene. Un vero *multitasking*, un *copy and paste* funzionante e un sistema di notifiche a *push-down* furono molto apprezzati dai consumatori. La prima versione di Android prometteva molte sorprese future, indipendentemente dal produttore del dispositivo. Ciò che ha reso “unico” Android è la sua natura open source, in cui gli utenti possono dare un contributo sul codice per aggiungere idee innovative.

2.3 Evoluzione

A partire dal primo rilascio ufficiale con la **versione 1.0**, Android ha fatto molta strada. Gli aggiornamenti per migliorare le prestazioni, l'*user experience* e per eliminare i bug delle precedenti versioni sono stati molti:



Figure 2.2: Evoluzione Android

Android 1.5 Cupcake ha introdotti molti miglioramenti, uno dei quali è stato l'aggiornamento del **kernel** alla versione 2.6.27, che rese il sistema più stabile. Furono introdotti *widgets* che sono largamente usati anche ora e per la prima volta gli utenti potevano installare una tastiera virtuale *custom*.

Android 1.6 Donut, rilasciato 4 mesi dopo, non apportò miglioramenti tranne l'aggiunta di screenshots nell'Android Market e la scelta multipla di photo nella galleria.

Android 2.0 Eclair è stato un grande passo in avanti nello sviluppo di Android. Il **kernel** era stato nuovamente aggiornato, questa volta alla versione 2.6.29. È stato introdotto un sistema di sincronizzazione dei contatti che permetteva l'aggiunta di indirizzi email. Sono state introdotte le applicazioni Email e un supporto bluetooth, l'UI è stata ottimizzata incrementando in tal modo la velocità di *scrolling*. Si è anche resa possibile la scelta del Live Wallpaper come background.

Android 2.2 Froyo (un abbreviazione per *frozen yogurt*) è stato presentato nel Maggio del 2010. L'obiettivo principale di questo aggiornamento era quello di migliorare la velocità del sistema, con l'introduzione di un compilatore **JIT** (Just-in-Time) nella **Dalvik Virtual Machine**. La connettività è stata rinnovata, permettendo il *tethering* (cioè l'utilizzo del dispositivo mobile come gateway per offrire accesso alla rete) sia via USB che WIFI. La browser app è stata aggiornata per supportare le GIFs e Flash player.

Il Google Nexus S è stato il primo smartphone fornito con **Android 2.3 Gingerbread**. Il Gingerbread è stato il più diffuso sistema operativo per gli smartphone. Ha offerto un supporto nativo per i nuovi sensori come l'NFC, giroscopio e barometro, e per la prima volta, Android offriva un API di supporto per la fotocamera anteriore e posteriore.

Android 3.0 Honeycomb è la prima versione di Android dedicata ai tablets. I miglioramenti sono stati molteplici tra cui lo trasferimento del rendering grafico nella **GPU**(Graphic Processing Unit), che ha reso il sis-

tema molto più responsive con un incremento delle performance grafiche con l'aggiunta di un supporto della grafica 3D.

La versione successiva introdotta da Google fu **Android 4.0 Ice Cream Sandwich**. Rivolto sia agli smartphones che ai tablets, Google ha rilasciato questa versione in contemporanea a un dispositivo estremamente popolare: il Samsung Galaxy Nexus. Per la prima volta fu adottato il **kernel 3.0.1 di Linux**.

Pochi mesi dopo fu la volta della versione **Android 4.1 Jelly Bean**. Google lo rilasciò con il Google Nexus 7, prodotto dalla ASUS. Jelly Bean, con i suoi tre principali aggiornamenti (versione 4.1, 4.2 e 4.3), divenne la versione più diffusa tra i dispositivi Android.

Alla fine dell'ottobre del 2013, precisamente ad *Halloween*, fu rilasciata la versione **Android 4.4 KitKat**. Fu introdotto un numero considerevole di funzionalità, ma l'obiettivo principale era rendere il sistema operativo più fluido sui dispositivi di bassa potenza, come i dispositivi con solo 512 megabyte di RAM. Fu introdotto il nuovo compilatore **ART** (approfondimento Android Runtime 2.7.3).

Un anno dopo Google rivelò la nuova versione del sistema operativo: **Android 5.0 Lollipop**. Google fece un drastico cambiamento per la UI introducendo il **Material Design** per dare un risposta tattile fluida agli elementi grafici. Si rese più efficiente il sistema di notifiche e si apposero miglioramenti alla durata della batteria e alla sicurezza.

Android 6.0 Marshmallow è l'ultima versione ufficialmente rilasciata da Google. Questa versione ha introdotto un nuovo sistema di richieste dei permessi da parte delle applicazioni all'avvio dell'applicazione invece che alla prima installazione.

2.4 Diffusione

La diffusione di Android a livello globale è stata velocissima. Dalla sua nascita nel 2007 ad oggi la sua crescita (come si può vedere nel grafico 2.3) è stata a dir poco fuori dal comune. Nel primo trimestre (Q1) del 2016, sono stati venduti globalmente più di un miliardo di smartphones di cui 84.1% erano dispositivi Android (14.8% iOS, 0.7% Windows Phone e il rimanente 0.4% altre piattaforme).

Tra i vari fattori che hanno reso Android la piattaforma più diffusa al mondo, quello che a mio parere è stato il più determinante è la sua accessibilità. La chiave della sua accessibilità è il prezzo. Secondo Statista, medi-

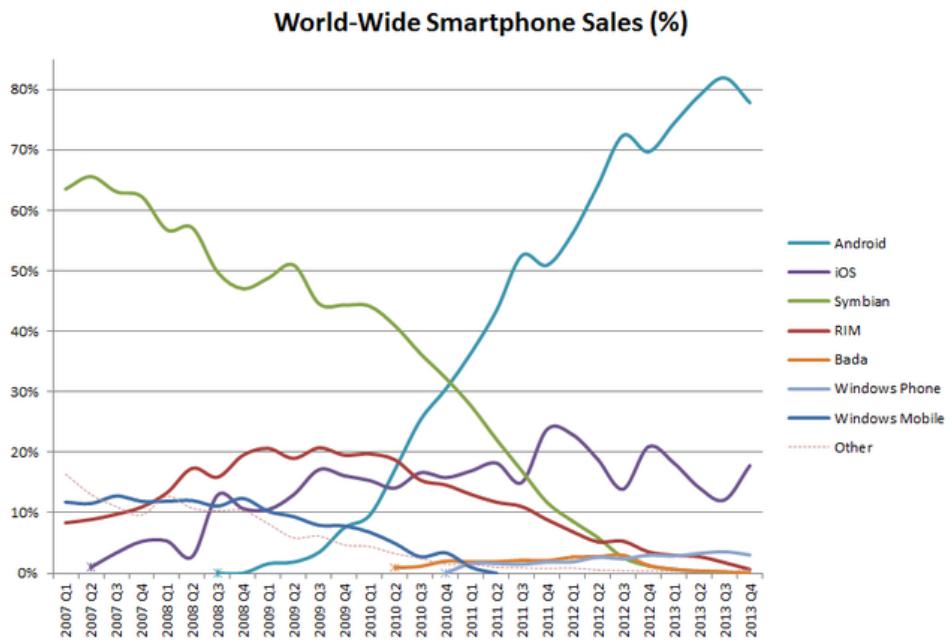


Figure 2.3: Market share delle piattaforme mobile dal 2007 al 2013

amente uno smartphone con piattaforma Android costa meno della metà di un IOS. Questo *gap* tra le due piattaforme si sta progressivamente allargando grazie soprattutto agli smartphone di produzione cinese.

2.5 La frammentazione

All'interno dell'ecosistema Android esiste una enorme frammentazione tra le versioni distribuite. La frammentazione è data dall'uso contemporaneo di differenti versioni del sistema operativo Android. Quest'ultima sta diventando un enorme problema sia per i sviluppatori che devono rendere il software compatibile con i dispositivi, che per Google perché le nuove versioni introdotte prevedono migliori misure di sicurezza e offrono migliori performance e *user experience*. La causa principale di questa frammentazione sono i produttori dei dispositivi mobile, che devono rendere compatibile le versioni rilasciate da AOSP con i vari dispositivi realizzati e questo può richiedere molto tempo prima che avvenga.

La versione di Android 5.0 Lollipop rimane, fino a oggi, la versione più diffusa sul mercato con circa il 36% dei dispositivi Android a livello globale.

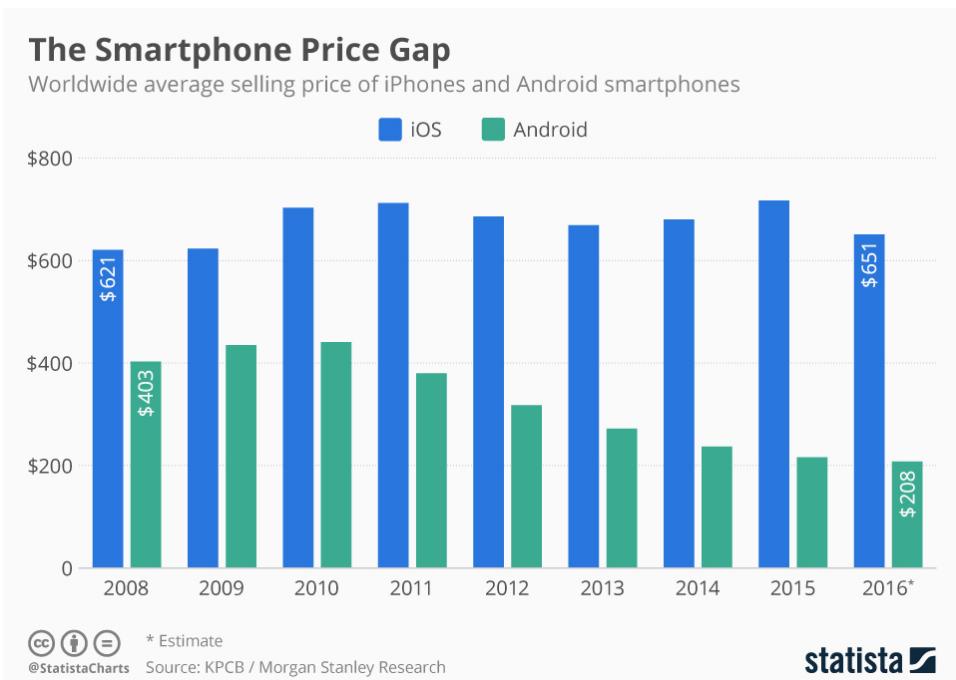


Figure 2.4: Gap di prezzo tra un dispositivo Android e IOS

2.6 “Android is about freedom...”

Android è conosciuto come un sistema operativo *open-source*, vale a dire che chiunque può scaricare il codice sorgente di Android e può sviluppare la propria versione del sistema operativo. *Android Open Source Project (AOSP)* è il canale ufficiale di distribuzione del codice sorgente. Ad ogni nuova *release* da parte di Google, il codice viene pubblicamente rilasciato tramite **AOSP**. L'**AOSP** è stato progettato con la convinzione che il mondo avesse bisogno di una piattaforma *open source* che rendesse lo sviluppo per le piattaforme mobile molto più semplice, aiutando così i programmatore nel processo di sviluppo. La scelta di Google di distribuire liberamente Android sotto la licenza *Apache Software License* versione 2.0 (“Apache 2.0”) ha garantito sia una distribuzione di contenuti gratuiti per tutti, sia un uso distribuito su una varia gamma di dispositivi differenti.

I grandi produttori come Samsung e HTC utilizzano le loro versioni modificate di questo sistema operativo rispettando il programma di compatibilità di Android (*Android Compatibility Program*). Il programma di compatibilità, come suggerisce il nome, mira a garantire una compatibilità tra i vari dispositivi. Verifica che alcune condizioni siano rispettate al fine di ottenere una certificazione per la pubblicazione sui vari canali di distribuzione di Google come il Play Store, Google Play Services e altre piattaforme. Ogni

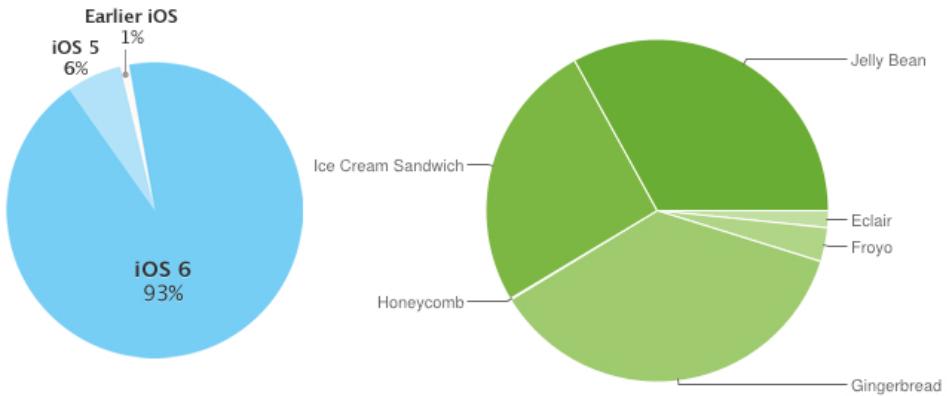


Figure 2.5: Contrasto frammentazione Android vs iOS

build (versione) di Android è sottoposto a questo test di compatibilità. Come si può leggere nel manifesto della AOSP:

Android is about freedom and choice. The purpose of Android is promote openness in the mobile world, and we don't believe it's possible to predict or dictate all the uses to which people will want to put our software. So, while we encourage everyone to make devices that are open and modifiable, we don't believe it is our place to force them to do so. Using LGPL libraries would often force them to do just that.³

AOSP rende anche possibile l'utilizzo del sistema operativo Android senza le applicazioni e i servizi di Google. La linea “Fire” di **Amazon** utilizza il sistema operativo Android, cambiando radicalmente quasi tutto il sistema operativo senza tenere conto del programma di compatibilità. Mentre il *core* di Android è open source, molto di ciò che compone Android non lo è. Le applicazioni di Google, che lentamente hanno sostituito quelle native e *open source* di Android, sono software proprietario.

2.7 Android Core

Il core di Android ha un'architettura a stack divisa in quattro livelli principali che interagiscono tra di loro. Di seguito verrà fatta una breve analisi di tutti i livelli utilizzando un approccio *bottom-up*, cioè dal basso verso l'alto.

³<http://source.android.com/source/licenses.html>

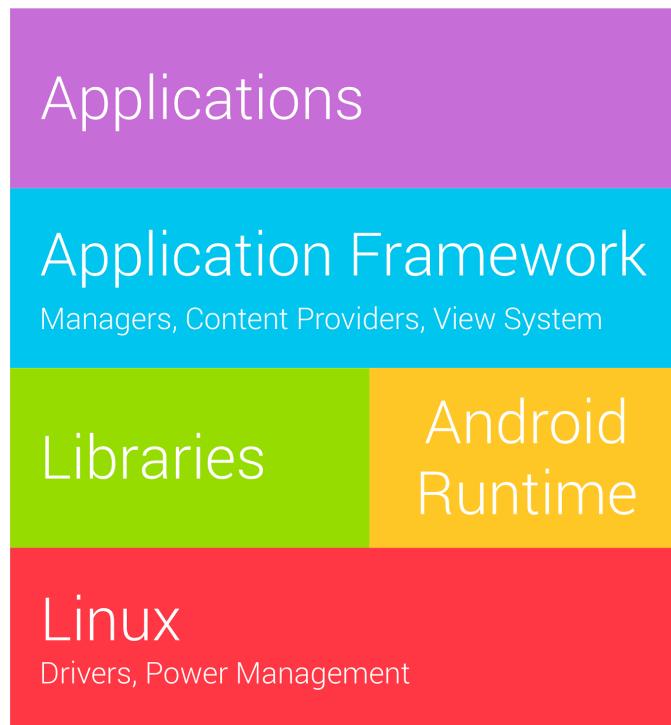


Figure 2.6: Android core stack

2.7.1 Linux Kernel

Android utilizza un'versione specializzata di **Linux Kernel** (Linux versione 3.6) con alcune modifiche come il **wake locks** (uno strumento che evita il consumo eccessivo della batteria), un sistema di gestione della memoria molto aggressivo che serve a preservarla, un **Binder IPC driver** e altri strumenti importanti per un sistema operativo per dispositivi mobili. Il **Kernel Linux** permette un'ottima gestione della memoria e dei processi. La sicurezza è un modello basato sui permessi (*Permissions-based security model*). Supporta la condivisione di librerie ed è open source.

Binder IPC Driver

Il **Binder IPC Driver** risolve il problema di come far comunicare le applicazioni e i servizi che vengono eseguiti in differenti processi e che devono comunicare e condividere i dati. Un noto problema della IPC(Inter Process Communication) è che può introdurre un eccessivo *overhead* e problemi di sicurezza. La soluzione è stata la realizzazione di un Driver per facilitare la comunicazione tra processi, garantendo una elevata prestazione attraverso

la memoria condivisa, con chiamate asincrone tra i processi.

Power Management

I dispositivi mobili funzionano a batteria e quest'ultima ha una capacità limitata. La **Power Management (PM)** di Android è costruita sulla **Linux Power Management** standard, con l'implementazione di una politica molto più aggressiva. Le componenti effettuano la richiesta tramite il **wake locks** per mantenere il dispositivo acceso.

2.7.2 Libraries

Sopra il livello del Kernel risiede un set di librerie native in C/C++. Sono per la maggior parte librerie esterne come OpenSSL, WebKit e bzip2 con alcune modifiche.

- **Bionic libc:** Le librerie standard C, chiamate Bionic, sono importate dalle librerie sotto la licenza BSD riscritte e ottimizzate per supportare l'hardware ARM.
- **WebKit:** Basate sulle librerie open source WebKit, offrono un supporto per i linguaggi web come il CSS, Javascript, DOM e AJAX.
- **Media Framework:** Componente in grado di gestire il contenuto multimediale, basato su PacketVideo OpenCORE paltform, supporta sia video standard, audio che immagini. Offre un supporto sia lato hardware che per il software per il codec dei vari formati multimediali.
- **SQLite:** Un database transazionale di piccole dimensioni e di enorme efficienza. Offre una persistenza dei dati alle varie applicazioni.
- **Surface Flinger:** Componente fondamentale, gestisce tutte le schermate delle applicazioni (le **Views**) e può gestire sia schermate 2D sia 3D o schermate multiple di applicazioni differenti. Utilizza OpenGL ES e 2D hardware accelerator.
- **Audio Flinger:** Componente che gestisce i dispositivi di output audio. Capace di gestire il routing dell'audio a dispositivi output diversi.
- **Hardware Abstraction Layer (HAL):** Sono librerie in C/C++, separano la logica delle applicazioni dalle interface hardware del dispositivo. Sono composte per lo più da driver di cui Android ha bisogno per comunicare con il hardware.

2.7.3 Android Runtime

Questa sezione offre una componente chiave denominata **Dalvik Virtual Machine**(DVM) che è una versione ottimizzata per Android di *Java Virtual Machine*. La **DVM** è una macchina virtuale che esegue il proprio bytecode, che a differenza del byte code standard è molto più compatto e genera .dex file più piccoli.

Con la nuova versione di *Android 4.4 Kitkat*, la DVM è stata rimpiazzata dalla **ART**(Android Runtime). Quest'ultima introduce molte novità che migliorano le performance e la smoothness delle applicazioni in esecuzione. La principale differenza tra le due macchine virtuali è che mentre la DVM è un compilatore basato sulla tecnologia *Just-In-Time(JIT)* cioè compila ed esegue il codice in tempo reale ad ogni esecuzione dell'applicazione, la **ART** introduce la compilazione *Ahead-Of-Time(AOT)* che, detta in breve, il codice viene interamente compilato in un linguaggio intermedio (bytecode) durante l'installazione dell'applicazione e non all'esecuzione. Questo, a detta degli ingegneri di Android, dovrebbe migliorare notevolmente il tempo di esecuzione.

2.7.4 Application Framework

Questo livello offre servizi essenziali alla piattaforma Android. È composto da un insieme di classi con cui le applicazioni interagiscono direttamente. Queste librerie gestiscono le funzioni base del dispositivo mobile (risorse, applicazioni installate, chiamate in uscita e in entrata, il file system ecc..) e offrono allo sviluppatore uno strumento di base per la creazione delle applicazioni.

Alcuni blocchi di questo livello sono:

- **Activity Manager:** Gestisce il ciclo di vita delle activities di ogni applicazione (vedi **Activity** 2.8.1).
- **Window Manager:** Componente che permette di gestire le finestre delle diverse applicazioni sullo schermo del dispositivo.
- **Content Provider:** Modulo che gestisce la condivisione di informazioni tra vari processi e le applicazioni.
- **View System:** Gestisce l'insieme delle Views(bottoni, liste, editor di testo ecc..) con cui vengono costruite le interfacce (il layout) delle applicazioni.
- **Package Manager:** Gestisce il ciclo di vita delle applicazioni, dall'installazione fino alla rimozione.

- **Telephony Manager:** Gestisce l'interazione delle funzioni tipiche di un dispositivo mobile (chiamate, sms, mms ecc..).
- **Resource Manager:** Modulo che fornisce l'accesso alle risorse interne del dispositivo che non sono codice come, il layout di un'applicazione con le relative stringhe e colori.
- **Location Manager:** Modulo che mette a disposizione dello sviluppatore una serie di API che si occupano della geo-localizzazione utilizzando il **GPS** o tramite la rete(*cell towers*)
- **Notification Manager:** Offre la possibilità di mostrare le notifiche all'user per un particolare evento, mettendo a disposizione una serie di meccanismi per la loro gestione.

2.7.5 Applications

In cima al framework troviamo il livello in cui risiedono tutte le applicazioni native e quelle esterne (le applicazioni di terze parti). Un utente medio di un dispositivo Android interagisce per lo più con questo livello. Tutte le applicazioni che risiedono in questo livello hanno gli stessi privilegi per accedere alle altre risorse dello stack.

2.8 Application Anatomy

Le applicazioni Android sono composte da un insieme di componenti che sono richiamabili all'occorrenza. Un'applicazione, a grandi linee, è composta da una serie di finestre chiamate Activities collegate insieme da **Intent** per passare da un **Activity** all'altra. Le Activities sono composte da **Views** e **Fragments**, composti da altre **Views**. Le Activities rappresentano i principali blocchi costitutivi delle applicazioni e sono le schermate con le quali l'utente interagisce.

2.8.1 Componenti principali

Qui di seguito verranno analizzati i componenti principali con i quali è stata costruita la nostra applicazione:

Activity

Un'**Activity** è una componente che offre una schermata con un interfaccia con cui l'utente può interagire. Per esempio, la nostra applicazione è composta da diverse Activities: un'**Activity** che mostra la schermata di login, un'altra che mostra il menù principale, un'altra **Activity** che mostra la rubrica dei clienti ecc.. Ogni **Activity** mostra una UI che risponde

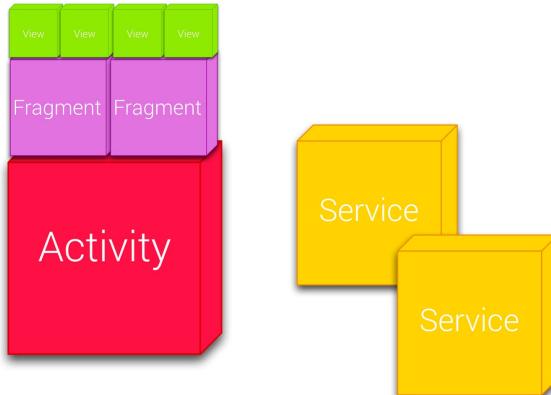


Figure 2.7: Le componenti Android

ad eventi generati sia dal sistema sia dell’utente. Mentre le Activities lavorano insieme per creare un’esperienza utente coesiva nella nostra applicazione, tuttavia ognuna di esse lavora in modo indipendente dalle altre. Applicazioni differenti possono avviare Activities di altre applicazioni (se queste lo permettono): per esempio la nostra applicazione avvia l’**Activity** dell’applicazione Dropbox per la visualizzazione dei documenti PDF. Il passaggio da un’**Activity** ad un’altra è compiuto con i metodi **startActivity()** oppure **startActivityForResult()** quando si desidera una chiamata sincrona oppure un risultato da un’altra **Activity**. Questi metodi richiedono un **Intent** come argomento. Tendenzialmente, all’interno dell’applicazione un’**Activity** viene specificata come la “main”**Activity** (attività principale), cioè la prima **Activity** che viene mostrata all’utente all’avvio dell’applicazione, nella nostra applicazione è l’**Activity** di login che deve essere specificata all’interno del **Manifest**. Ogni volta che una nuova **Activity** viene avviata viene messa in cima allo stack e acquisisce il focus dell’utente mentre la precedente **Activity** viene stoppata, ma viene mantenuta dal sistema in uno stack (il “back stack”). Ogni **Activity** ha un ciclo di vita indipendente dalle altre, che inizia dal momento in cui Android ne crea un’istanza con lo scopo di rispondere ad un **Intent** e terminare quando l’istanza viene distrutta. Ogni cambiamento di stato viene notificato dai metodi di callback.

```

1 import android.app.Activity;
...
3 //A login screen that offers login via email/password.
5 public class LoginActivity extends Activity{
...
7 @Override
    protected void onCreate(Bundle savedInstanceState) {

```

```

9      super.onCreate(savedInstanceState);
10     setContentView(R.layout.activity_login);
11     ...
12 } //End of onCreate method
13     ...

15     @Override
16     protected void onStart() {
17         super.onStart();
18         // The activity is about to become visible.
19     }

21     @Override
22     protected void onResume() {
23         super.onResume();
24         // The activity has become visible (it is now "resumed").
25     }

27     @Override
28     protected void onPause() {
29         super.onPause();
30         // Another activity is taking focus (this activity is
31         // about to be "paused").
32     }

33     @Override
34     protected void onStop() {
35         super.onStop();
36         // The activity is no longer visible (it is now "stopped")
37     }

39     @Override
40     protected void onDestroy() {
41         super.onDestroy();
42         // The activity is about to be destroyed.
43     }
44 } //End of LoginActivity class
45

```

Listing 2.1: Metodi di callback di una Activity

La classe `LoginActivity` estende la classe `Activity`. Una delle prime operazioni che un'Activity svolge è di mostrare la UI con il metodo `setContentView()` (se la UI è stata implementata in un file Layout XML). Un'Activity passa da uno stato ad un altro in risposta a eventi esterni: il ciclo di vita è direttamente influenzato dal fatto che questa sia associata ad altre Activities. In un determinato momento un'Activity si può trovare in uno di questi stati:

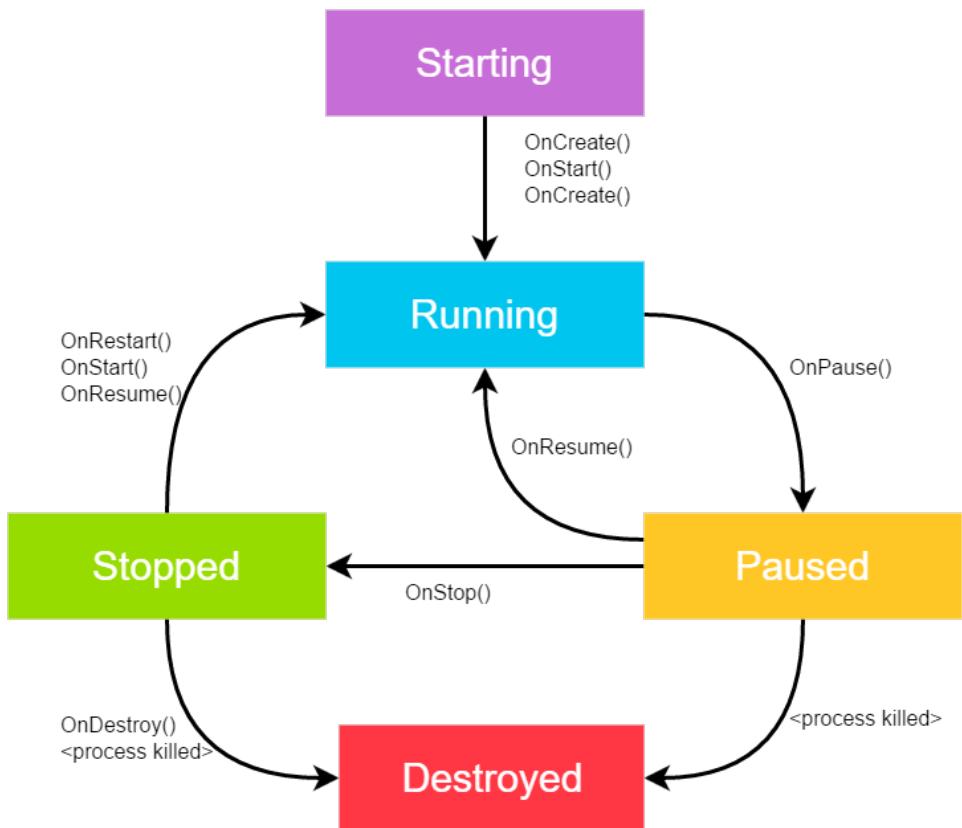


Figure 2.8: Ciclo di vita di un'Activity

- *Created:* quando il metodo `onCreate()` viene chiamato, l'Activity è in fase di creazione, ma non è ancora visibile all'utente.
- *Starting:* in questo stato l'Activity è stata creata e può essere visibile all'utente ma non può interagire con essa. Il metodo che viene generalmente invocato è `onStart()`.
- *Resumed:* Quando l'Activity viene richiamata e riposizionata in cima allo stack, essa riprende il focus in foreground. Viene invocato il metodo `onResume()`.
- *Paused:* l'Activity ha perso il focus e non è più in foreground ma è ancora visibile. Quando si trova in questo stato, l'Activity è ancora “viva” ma può essere distrutta dal sistema per mancanza di memoria in questo caso viene invocato il metodo `onPause()`.
- *Stopped:* l'Activity non è più visibile, ma è ancora conservata in memoria. Il metodo `onStop()` è invocato ogni volta che entra in questo stato.

- *Destroyed*: l'**Activity**, non più necessaria, viene rimossa dalla memoria. La distruzione delle Activitis è una decisione presa dall'**Activity Manager** quando decide che un **Activity** non è più utilizzata.

Fragment

I Fragments furono introdotti solo a partire dalla versione **HoneyComb**, per sfruttare l'enorme spazio offerto dagli schermi dei tablet. Un **Fragment** va pensato come una parte modulare di un'**Activity**, con il proprio ciclo di vita e il proprio layout, che riceve eventi in input e può essere aggiunto o rimosso da un **Activity** mentre quest'ultima è in esecuzione; essa è un componente modulare e riutilizzabile.

Il ciclo di vita di un **Fragment** è direttamente influenzato da quello dell'**Activity** ospitante: se l'**Activity** ospitante è nello stato **paused**, anche i **Fragment** che contiene sono nello stesso stato, quando l'**Activity** è **resumed**, i **Fragment** possono essere manipolati in modo indipendente l'uno dall'altro, aggiunti, rimossi, nascosti o mostrati.

Content Provider

Il **Content Provider** è un meccanismo che permette alle applicazioni di comunicare tra di loro o semplicemente offre una persistenza memorizzando alcune informazioni. Si tratta principalmente di un'interfaccia che offre dei metodi standard per accedere ai dati come query, insert, update, delete. Non esistono restrizioni da parte del sistema per il tipo di struttura dati utilizzata, che può essere un file singolo, un database **SQLite** oppure tramite la rete. Esistono differenti **Content Provider** all'interno del sistema che offrono supporto per differenti dati. Ogni **Content Provider** è identificato da un `<authority>`; il principale modo per accedere ai dati è quello di interrogare tramite query il **Content Provider** utilizzando il **Content URI** che ha la seguente forma:

```
content://<authority>/..<type path>..<id>/
```

²

Listing 2.2: Formato Content URI

Shared Preferences

Le **SharedPreferences** (preferenze condivise) sono una classe che permette di immagazzinare delle coppie di chiave-valore in un file all'interno del sistema. Solitamente viene utilizzata per salvare dati riguardanti l'applicazione come la preferenza della lingua, un ID utente o altro. Nel nostro caso è stata utilizzata per memorizzare le credenziali di accesso degli utenti.

Views

Le **Views** sono il mattoni con cui si costruisce una UI di un'applicazione Android. Le Activities, come abbiamo visto, contengono le Views e gli oggetti **View** rappresentano elementi sullo schermo che sono responsabili dell'interazione con l'utente tramite eventi(click, long click, inserimento input, drag ecc...). Ogni schermata Android è composta da un albero gerarchico di elementi View. La piattaforma Android offre un set di **View** all'interno del package `android.view` necessarie per realizzare una schermata base. Le Views possono essere inserite in un'Activity (sulla schermata) sia direttamente in codice sia tramite l'uso di un file Layout XML conservato nella directory `res/` che viene “inflated” a runtime. La figura 4.8 illustra le Views incluse nel package `android.view`.

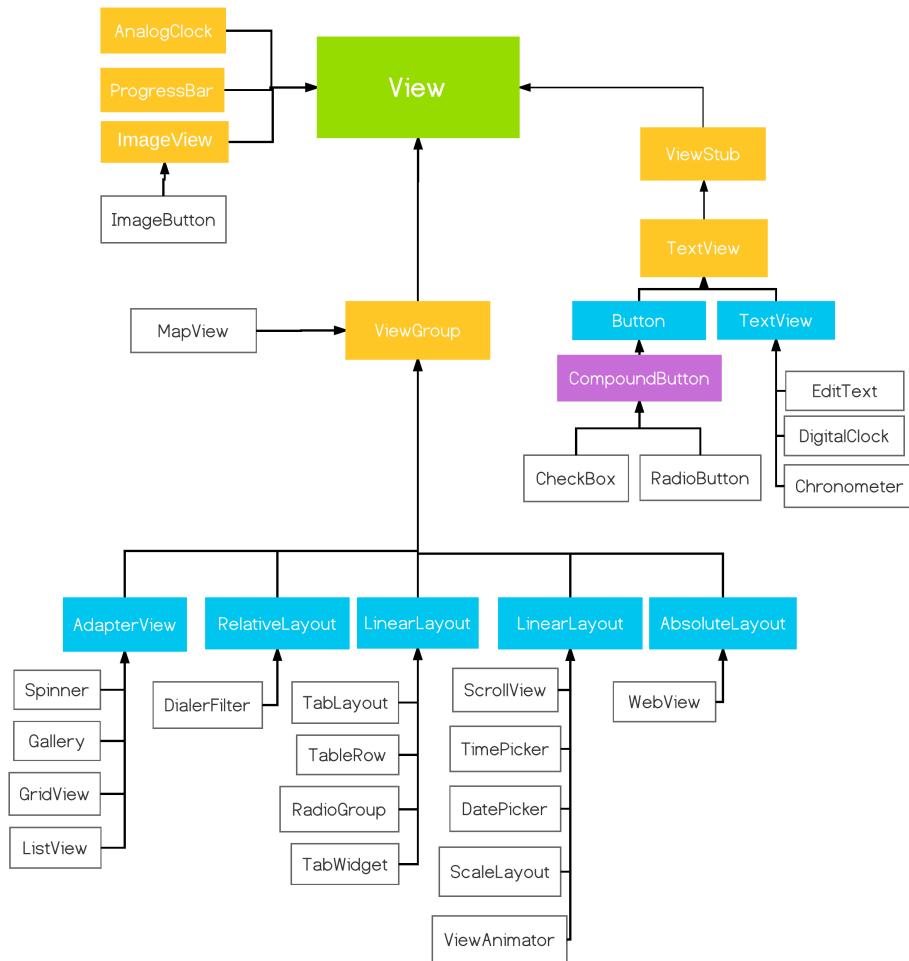


Figure 2.9: Le pincipali componenti View

Gli ambienti di sviluppo per la piattaforma Android come Android Studio e Eclipse offrono un Layout Editor che permette di creare in maniera intuitiva una UI trascinando le componenti nella posizione che ci interessa. Il risultato è un file XML di layout che solitamente è così composto:

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout      xmlns:android="http://schemas.android.
3   com/apk/res/android"
4       android:layout_width="fill_parent"
5       android:layout_height="fill_parent"
6       android:orientation="vertical" >
7
7  <TextView
8      android:layout_width="fill_parent"
9      android:layout_height="wrap_content"
10     android:text="@string/hello" />
11
11</LinearLayout>
```

Listing 2.3: Layout XML con componente TextView contenente Hello

Il Manifest

In ogni progetto Android è presente un file `AndroidManifest.xml`, conosciuto anche come il **Manifest**. Questo file si trova nella directory principale e presenta informazioni essenziali circa l'applicazione all sistema, informazioni che il sistema deve avere prima di poter eseguire il codice. Alcune informazioni che vale la pena nominare, come il *package name* e l'*application version* sono le prime da settare nel progetto; il primo è il nome del package nel quale sono conservate le classi sorgenti. Mentre l'informazione relativa alla *version* è distinta da due informazioni, cioè dalla `versionCode` che definisce la versione corrente dell'applicazione e la `versionName`, che specifica la versione visibile agli utenti.

```
1  <manifest
2      xmlns:android="http://schemas.android.com/apk/res/
3       android"
4         package="mcteamgestapp.momo.com.mcteamgestapp"
5         android:versionCode="1"
6         android:versionName="1.0">
7
7  ...
```

Il **Manifest** descrive le componenti dell'applicazione (le Activities, i Services, i Broadcast receivers di cui l'applicazione è composta), attribuisce un nome alle classi che implementano ognuna di queste componenti e definisce le loro

capabilities (per esempio quale messaggio Intent possono gestire). Queste dichiarazioni permettono al sistema Android di capire che comportamento adottare per ogni componente dichiarata.

```
2   <application
3       android:name=".MyApplication"
4           android:icon="@mipmap/ic_launcher"
5           android:label="@string/app_name"
6           android:theme="@style/AppTheme">
7       ...
8           <activity
9               android:name=".Moduli.Login.LoginActivity"
10              android:label="@string/title_activity_login">
11                  <intent-filter>
12                      <action android:name="android.intent.action.MAIN"
13                      />
14                      <category android:name="android.intent.category.LAUNCHER" />
15                  </intent-filter>
16          </activity>
17      ...
18  </application>
```

Tutte le Activities di un'applicazione Android devono essere dichiarate nel **Manifest** sotto l'elemento **<application>**. Le Activities non dichiarate all'interno del **Manifest** non vengono viste dal sistema e non vengono eseguite e sono rappresentate dall'elemento **<activity>**. L'elemento **<intent-filter>** dichiara le *capabilities* del componente padre e specifica quale Intent può eseguire l'Activity. Nel nostro caso l'Activity **LoginActivity** viene dichiarata come la MAIN Activity ed è quella che viene lanciata al momento dell'avvio dell'applicazione.

Il **Manifest** inoltre dichiara quali permessi sono richiesti dall'applicazione per poter accedere a API protette o per interagire con altre applicazioni. L'elemento **<uses-permission>** richiede un permesso che deve essere garantito per poter funzionare. Nel nostro caso il sistema deve garantirci un accesso alla rete internet per poter funzionare.

```
// Requesting permission for internet access
2  <uses-permission android:name="android.permission.
3      INTERNET" />
4  <uses-permission android:name="android.permission.
5      ACCESS_NETWORK_STATE" />
```

4

I permessi, nelle versioni Android precedenti la **Marshmallow 6.0** vengono richiesti all’utente al momento dell’installazione. Mentre in quelle successive vengono richieste a runtime.

Capitolo 3

Architettura

Lo scopo principale del progetto GestApp è quello di offrire alle piccole e medie imprese un sostegno per la gestione remota delle attività aziendali, specialmente per la gestione del *workflow*. In questo capitolo si discute principalmente dell'architettura adottata per lo sviluppo dell'applicativo. La prima architettura è quella dell'intero sistema; le scelte progettuali fatte; la modalità di comunicazione tra il client e il server.

3.1 Architettura del sistema

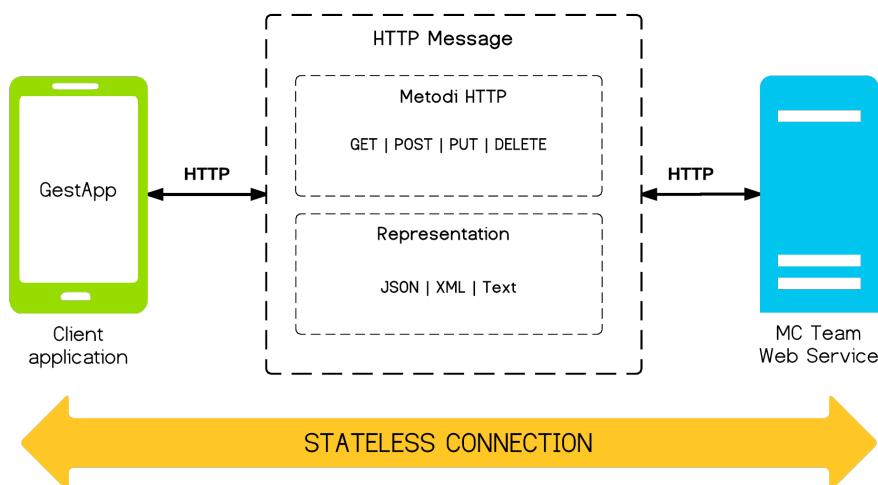


Figure 3.1: Architettura di una RESTful API

L'applicazione GestApp è stata progettata per avere le stesse funzionalità della versione web GestTeam. L'applicazione comunica costantemente con

il server per inviare e ricevere i dati. La comunicazione tra il dispositivo (client) e il web server utilizza il sistema **RESTful** basato sulle richieste HTTP e scambio di dati in formato JSON.

3.1.1 REST

Il termine **REST** è un acronimo per **REpresentational State Transfer**. Il concetto di REST fu introdotto da Roy T. Fielding nella sua tesi di dottorato nel 2000[1]. Fielding aveva contribuito allo sviluppo del protocollo HTTP 1.0 ed era il principale sviluppatore della versione HTTP 1.1 e l'autore della sintassi Uniform Resource Identifier (**URI**). Sebbene REST non sia un'architettura di per sé, ma è un insieme di vincoli che, applicati, conducono a un design simile ad un'architettura software. Questi vincoli, quando applicati nel loro complesso, enfatizzano una maggiore scalabilità di interazione tra le componenti e una maggiore sicurezza.

Invece quando ci si riferisce ad una **RESTful** web service, spesso ci si riferisce ad una **API** (Application Programming Interface), che è conforme alle specifiche REST e che comunica utilizzando dati in formato XML o JSON tramite il protocollo HTTP.

I vincoli che definiscono un'architettura **RESTful** sono i seguenti:

1. Modello client-server
2. Comunicazione stateless
3. Meccanismi di Caching
4. Uniform Interface
5. Resources (tutto viene trattato come risorse)

Per capire meglio il sistema REST, è fondamentale comprendere i concetti di risorsa (*resource*), *representation* e stato all'interno del contesto di REST.

Resources

Le risorse sono le componenti chiave su cui ruota l'intero paradigma REST. Ogni informazione può diventare una risorsa. Per esempio: un documento, un'immagine, un oggetto lato server ecc. Fielding definisce la risorsa come “*a conceptual mapping to a set of entities, not the entity that corresponds to the mapping at any particular point in time.*” Vale a dire che non solo la rappresentazione (*representation*) della risorsa cambia con tempo, ma anche che i dati su cui è basata la risorsa potrebbero mutare. Le risorse devono essere indirizzabili per essere accessibili e trasferibili tra i clients e i servers. Le risorse in REST sono univocamente identificate dai loro **URIs**. **URI** è un acronimo per *Universale Resource Identifier*, che può essere considerato una combinazione di *Uniform Resource Locators* (URL) e *Uniform Resource*

Names (URN). REST richiede che l'URI ad una risorsa non cambi nel tempo. www.mcteam.it/mobile/societa/10 questo è un esempio di URI che identifica la risorsa società con l'identificato 10.

Representation

La *representation* di una risorsa sono i dati o i contenuti scambiati tra il server e il clienti. Se vogliamo una definizione precisa allora possiamo definire una representation come un'istanza temporanea dei dati che sono effettivamente memorizzati oppure trasferiti in un determinato momento. Perciò una representation, di solito, è costituita da dati, metadati che descrivono i dati e, a volte, metadati che descrivono i metadati (per verificare l'intergrità del messaggio). Per esempio una pagina HTML non è una risorsa ma una tra le tante representation della risorsa.

Una representation di una risorsa può avere diverse forme come HTML, immagine, file, plain text, XML e JSON ecc. Il formato dei dati di una representation è conosciuta come **media type**, ma generalmente viene indicato come **MIME type**. In base alla richiesta del client, alle preferenze nei formati o alla natura delle risorse, una risorsa può avere un'appropriata representation generata dinamicamente (HTML, XML, JSON ecc.). Ma tutti questi differenti representation della medesima risorsa devono essere raggiungibili tramite lo stesso URI.

Stateless interactions

In base ai principi di REST, tutte le comunicazioni tra il client e il server devono essere di natura **stateless**. Questo implica che ogni richiesta da parte del client al server debba contenere tutte le informazioni necessarie per gestire correttamente la richiesta. Sul server non sono conservati informazioni sullo stato delle richieste. Invece le infomrationi sullo stato della sessione, sono conservati interamente dal client.

È importante distinguere tra stato della sessione o dell'applicazione e stato della risorsa. Lo stato della sessione è conservato dal client mentre lo stato della risorsa è conservato sul server.

Uniform interface

Nelle moderne applicazioni web based, le operazioni base possono essere categorizzate nelle seguenti quattro tipologie: creazione, visualizzazione, modifica e eliminazione. Queste operazioni sono generalmente conosciute con l'acronimo **CRUD** che deriva dalle quattro operazioni in inglese **Create, Retreave, Update and Delete**. Le API **RESTful** permettono una manipolazione degli stati delle risorse tramite i *web services* e queste manipolazioni consistono nella creazione, modifica, aggiornamento e eliminazione delle risorse.

Operazione	Metodo HTTP
CREATE	PUT
RETRIEVE	GET
UPDATE	POST
DELETE	DELETE

Table 3.1: Tabella CRUD e metodi HTTP

Il protocollo HTTP offre una varietà di metodi (*verbs*) tra i quali GET, POST, PUT, DELETE, OPTIONS, HEAD, TRACE, CONNECT e PATCH; solo un sottoinsieme di questi metodi sono necessari a soddisfare le esigenze di numerose web application. Le operazioni **CRUD**, prendendo in considerazione REST e i suoi principi di risorsa e representation, possono essere mappate con metodi HTTP PUT, GET, POST e DELETE. In breve, HTTP GET è utilizzata per acquisire una representation di una risorsa, HTTP DELETE per rimuovere una risorsa, HTTP PUT per creare una risorsa e HTTP POST per aggiornare una risorsa. La risorsa è identificata dal suo URI e l'operazione da eseguire su di essa è indicata dal metodo HTTP usato.

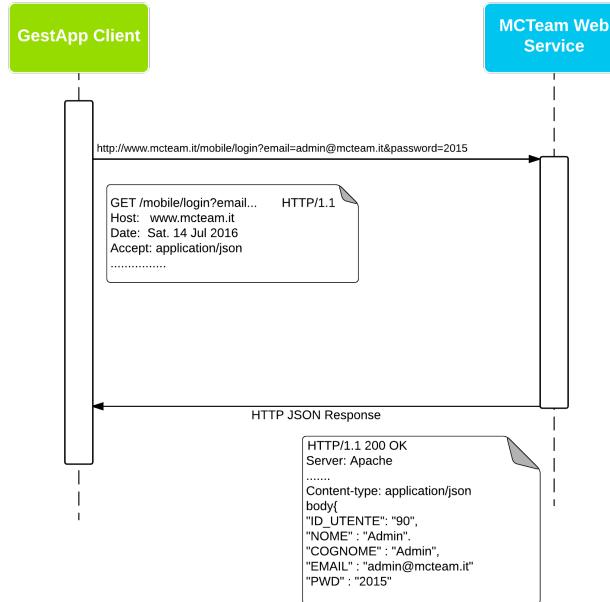


Figure 3.2: Operazione di visualizzazione della representation

Operazione GET/RETRIEVE: Il metodo HTTP GET è utilizzato per ottenere una risorsa. È un'operazione *read-only*(di solo lettura) e non permette nessuna modifica alla risorsa. La figura 3.2 è un diagramma di

sequenza al alto livello preso dal progetto, che illustra il meccanismo. La nostra client application invia una richiesta con HTTP GET al web service per ottenere la representation della risorsa `/login` identificata con i parametri `login=admin@mcteam.it` e `password=2015`. Il client inoltre informa il server sul formato della representation tramite il campo `Accept` nel header della richiesta HTTP; nel nostro caso si tratta del formato JSON. Il server elabora i dati in back-end ed invia una risposta HTTP contenente la representation risultante nel formato richiesto. In questo caso l'la risorsa identificata con le credenziali email e password, serializzati in formato JSON.

Operazione PUT/CREATE e POST/UPDATE: Il metodo HTTP PUT è utilizzato per creare una nuova entità. La figura 3.3 illustara il meccanismo: In questo caso il client tenta di creare una nuova entità sulla risorsa

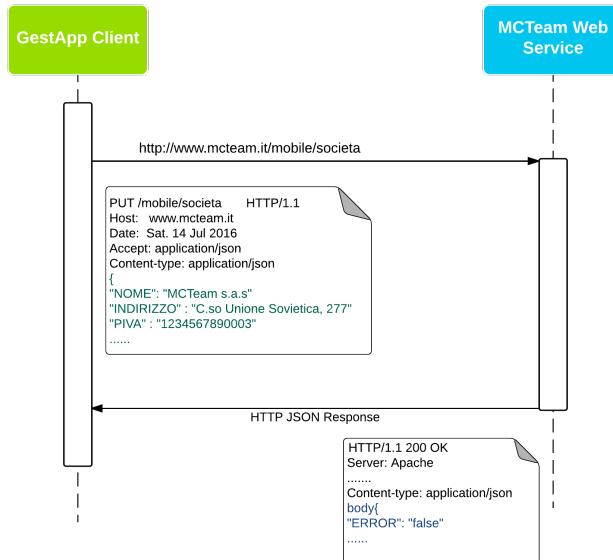


Figure 3.3: Creazione di una nuova risorsa Societa con il metodo HTTP PUT

`/societa`, inviando una richiesta HTTP PUT al server contenente, nel body della richiesta, le informazioni per creare una nuova risorsa serializzati in JSON. Una volta ricevuta la richiesta, il server identifica l'operazione dal metodo HTTP ed il formato del body dal campo `Content-type` presente nel header della richiesta. Il server deserializza le informazioni in JSON e chiama la funzione in back-end corrispondente per eseguire l'operazione di creazione. Invece il metodo HTTP POST è utilizzato per aggiornare una risorsa. Il meccanismo è simile a quello della PUT l'unica differenza sta nella funzione di back-end invocata. Mentre il POST crea una nuova entità, il PUT aggiorna i

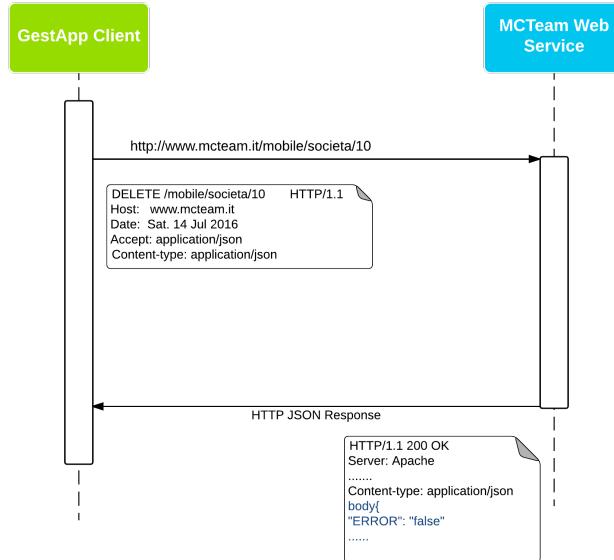


Figure 3.4: Eliminazione di una risorsa società con il metodo HTTP DELETE

dati contenuti in essa.

Operazione DELETE: Il metodo HTTP **DELETE** viene utilizzato per l’eliminazione delle risorse. Il client invia la richiesta **DELETE** al server per eliminare la risorsa univocamente identificata come `/societa/1`. Il server elabora la richiesta e chiama la funzione desiderata per completare l’operazione di eliminazione. L’URI `www.mcteam.it/mobile/societa/10` identifica la risorsa da eliminare. Il server restituisce un feedback al client in base all’esito dell’operazione.

Capitolo 4

GestApp overview

4.1 GesTeam

GesTeam è una piattaforma Web-based ideata e sviluppata dalla MC Team al preciso scopo di offrire un supporto alle piccole e medie imprese nella gestione del *workflow* aziendale. **GesTeam** è una applicazione web

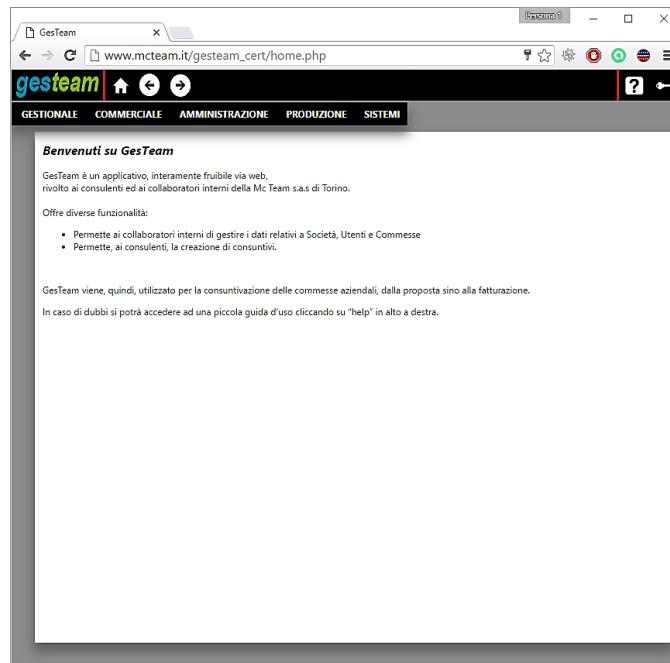


Figure 4.1: GesTeam applicazione web based home page

based sviluppata interamente in linguaggio PHP, HTML, CSS e JavaScript. Essa si presenta in cinque moduli separati: gestionale, commerciale, amministrazione, produzione e sistemi. Ogni modulo è accessibile solo agli utenti che hanno determinati privilegi. Ogni modulo presenta funzionalità o tools

differenti che vanno dalle rubriche clienti e società fino alla gestione delle commesse e gestione dei calendari di lavoro.

4.1.1 Scopo dell'applicazione

L'applicativo **GesTeam** è rivolto principalmente alle piccole e medie imprese e offre supporto per gestire in modo semplice e intuitivo il ciclo di vita delle attività aziendali. Una tra le principali attività svolte all'interno di un'azienda è l'interazione tra i vari ruoli aziendali (dal Direttore Commerciale all'Amministratore Aziendale e dal Project Manager al Consulente), ovvero un continuo comunicare informazioni, conferme e risposte, le quali, nella maggior parte dei casi, tendono a diventare ridondanti e inefficienti. Per fare un esempio, prendiamo in considerazione la gestione delle risorse impiegate sulle commesse e la successiva consuntivazione, un'operazione che coinvolge quotidianamente del personale addetto alla gestione della documentazione e che comporta un elevato ammontare cartaceo e dei tempi di consegna eccessivi. L'idea di base di **GesTeam** nasce dalla necessità da parte dell'azienda di ottimizzare i tempi e diminuire i costi di elaborazione, rendendo l'intero sistema aziendale più efficiente. Essa possiede svariate caratteristiche che risultano vantaggiose allo scopo, tra cui:

- unificazione del sistema informatico in un unico applicativo;
- agevolazione del lavoro con la possibilità di accedere al sistema da una qualsiasi postazione dove sia presente una connessione Internet;
- rubrica clienti e documentazione aziendale condivisa con supporto alla ricerca dei dati ed eventuale stampa;
- report statistici dei clienti e delle commesse;
- monitoraggio e controllo della documentazione amministrativa;
- immediato feedback sullo stato di avanzamento delle commesse e dei consuntivi;
- diminuzione del materiale cartaceo prodotto.

GesTeam è stato strutturato in modo tale da permettere l'accesso a differenti tipologie di utenti, assegnando ad ognuno di essi dei privilegi, in base ai quali viene assegnato un servizio personalizzato determinando così la possibilità di accedere alle diverse funzioni all'interno dell'applicativo. Ad esempio, un consulente può visionare le proprie commesse e i propri consuntivi; un Project Manager può esaminare i consuntivi sia delle risorse sia delle commesse di sua competenza e analizzarne lo stato di avanzamento delle medesime; il dipendente amministrativo ha una visibilità globale dei dati, in modo tale da rendere più agevole il lavoro contabile.

Tutto questo ha lo scopo di creare infine una documentazione suddivisa in base alla diversa tipologia di utenti, interni e consulenti, tramite la generazione, su richiesta, dei report su un singolo documento (o su un insieme) in base agli input di ricerca inseriti. Ad esempio, la ricerca dei report su tutte le commesse associate a una particolare società, l'avanzamento delle commesse in un particolare mese dell'anno, la consuntivazione di un determinato consulente oppure la stampa della rubrica dei clienti.

La creazione dei report è stata realizzata, a video, in formato PDF, per poter salvare i dati in formato elettronico per le successive attività che ne richiederanno l'utilizzo, come ad esempio la fatturazione verso il cliente.

4.2 GestApp

L'applicazione mobile GestApp, progettata e realizzata durante lo stage, è un'applicazione nativa per dispositivi Android, che presenta gli stessi moduli e gli strumenti o *tools* presenti nella versione web based. GestApp è un'applicazione che si appoggia su un web service e l'intera comunicazione avviene utilizzando l'architettura REST(vedi 3.1.1).

4.3 Ruoli e funzioni

Possiamo definire all'interno dell'azienda cinque ruoli importanti:

- **Amministratore:** è l'addetto alla contabilità industriale, si occupa di attribuire ai diversi settori aziendali i costi e ricavi che la contabilità generale rileva per l'intera impresa. Lo scopo della contabilità industriale (o analitica) è di rendere esplicite la gestione interna delle risorse e le modalità di utilizzo dei fattori produttivi nelle aree di attività dell'azienda.
- **Direttore Commerciale:** è il responsabile dell'attuazione delle politiche commerciali dell'azienda. Il suo ruolo dipende da molti fattori ma, soprattutto, dalla struttura organizzativa dell'azienda: nelle realtà più piccole, a questa figura è spesso richiesto di occuparsi anche della promozione dei prodotti, dello studio dei mercati di riferimento e della definizione delle strategie commerciali. Al pari di altri dirigenti, il Direttore commerciale ha il compito di raggiungere gli obiettivi strategici dell'impresa con le risorse finanziarie, umane e strumentali di cui dispone ed elabora i piani d'azione, con cui stabilisce come impiegare le risorse a sua disposizione, dà indicazioni operative e fissa i traguardi definendo le politiche d'incentivazione.

- **Gestionale:** è colui che si occupa della gestione delle risorse umane all'interno dell'azienda, individuando un insieme di politiche che possono essere applicate in tutte le circostanze e che avranno un effetto positivo sulle performance aziendali.
- **Capo Progetto:** la sua missione consiste nel pilotare un progetto (nel nostro caso si tratta di una Commessa), dalla sua idea di partenza all'attuazione generalizzata. Alcuni dei compiti del capo progetto sono:
 - Definizione del progetto e individuazione dei bisogni;
 - Quantificazione del costo del progetto;
 - Inquadramento dell'équipe di realizzazione;
 - Controllo e monitoraggio sull'avanzamento del progetto, in termini di qualità, costi e scadenze;
- **Consulente:** si tratta solitamente di una persona assunta in una società di servizi ma che presta servizio presso un cliente; esso ricopre un ruolo di analisi, valutazione dei bisogni, di consiglio e di proposta di soluzioni.

Una volta definiti i ruoli principali, è possibile definire le funzionalità che gli sono state attribuite all'interno del sistema.

Amministratore Alcune delle funzionalità dell'amministratore sono la visualizzazione o la stampa dei consuntivi associati ad una commessa o ad un consulente relativi ad un mese; la manipolazione completa (creazione, visualizzazione, modifica, eliminazione e stampa) della rubrica banche.

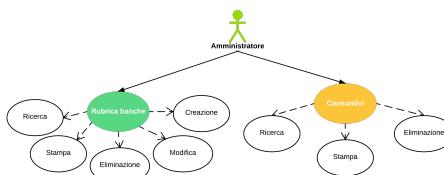


Figure 4.2: Funzioni di un amministratore

Commerciale Il commerciale ha la completa gestione sulle commesse. Chi ha i privilegi da commerciale può creare, modificare, eliminare una commessa oppure effettuare la ricerca o la stampa. Oltre a queste funzionalità il commerciale può mantere sotto controllo le fasi d'avanzamento delle commesse. Combinando queste due funzionalità, il direttore commerciale ha sotto controllo l'andamento delle commesse, dei consulenti allocati su di esse e il quantitativo di ore/giorni di lavoro.

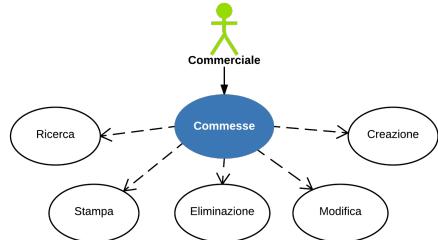


Figure 4.3: Funzionalità commerciale

Gestionale Questo ruolo di occupa principalmente della gestione del personale: creazione modifica eliminazione e stampa dei dati (in formato PDF) delle anagrafiche dei clienti, utenti(risorse umane), commesse e associazioni dei Consulenti e Capi progetto sulle commesse.

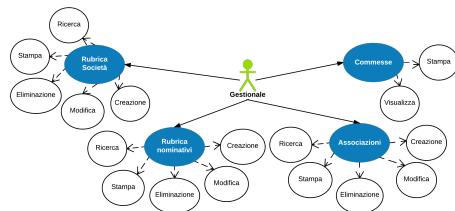


Figure 4.4: Funzionalità gestionale

Capo Progetto Le funzionalità accessibili agli utenti a cui sono stati assegnati i privilegi da Capo Progetto sono le stesse definite per il Direttore Commerciale, con l'unico vincolo che si permette la visualizzazione/stampa dei report ai soli utenti allocati sulle commesse di sua responsabilità. In questo modo il Capo Progetto tiene sotto controllo solo quello che è sotto la sua responsabilità e riferisce dello stato di avanzamento delle commesse e impiego delle risorse al superiore, in base ai costi (preventivati e sostenuti) e al raggiungimento degli obiettivi.

Consulente Il consulente ha due funzionalità essenziali: la compilazione e modifica dei report giornalieri o mensili, e la visualizzazione e stampa di report per commessa.

4.4 Diagramma casi d'uso

Il seguente diagramma dei casi d'uso riporta le principali funzionalità che sono state sviluppate all'interno dell'applicazione client. Il principale scopo

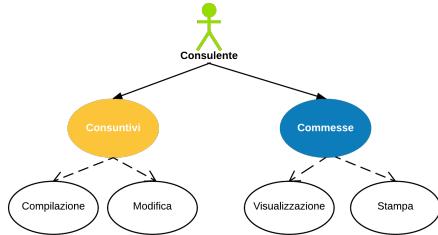


Figure 4.5: Funzionalità consulente

di questo diagramma è quello di dare un quadro generale del sistema da sviluppare. Gli attori principali all'interno del sistema sono i collaboratori che possono essere collaboratori interni o collaboratori esterni(consulenti), e il web service della MCTeam.

4.5 Funzionalità e design

Gli utenti primari dell'applicazione GestApp sono principalmente i consulenti che lavorano presso i clienti. Mentre i collaboratori interni utilizzano principalmente la versione web based. Di seguito vedremo le parti implementate.

4.5.1 Login

La schermata di login è la prima schermata che viene mostrata all'utente all'avvio dell'applicazione Figura 4.7.

Per accedere all'applicazione è necessario possedere le credenziali che vengono fornite da chi possiede i privilegi da sistemi tramite il tool Accessi. Ad ogni login l'utente può scegliere se permettere al sistema di memorizzare i dati o meno per facilitare un futuro accesso all'applicazione. Nel caso in cui l'utente decida di memorizzare i dati d'accessi i dati vengono memorizzati all'interno della `SharedPreferences` in `MODE_PRIVATE` e vengono recuperati ad ogni avvio dell'applicazione.

```

1   public class LoginActivity extends Activity{
2       private SharedPreferences mSharedPreferences;
3       private SharedPreferences.Editor mLoginEditor;
4       ...
5       private void rememberMeFirst() {
6           String email = mEmailView.getText().toString();
7           String password = mPasswordView.getText().toString()
8       ;
9           if(mCheckRememberMe.isChecked()){
10               mLoginEditor.putBoolean("rememberMe", true);
11           }
12       }
13   
```

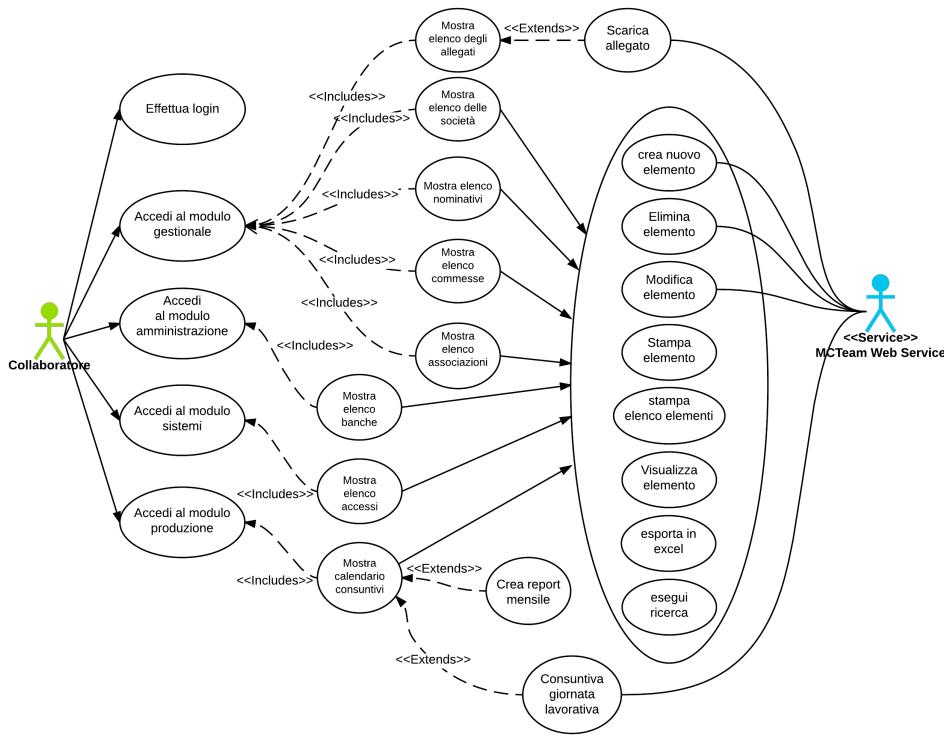


Figure 4.6: Diagramma casi d'uso

```

11     mLoginEditor.putString("email", email);
12     mLoginEditor.putString("password", password);
13     mLoginEditor.commit();
14 }
15 else {
16     mLoginEditor.clear();
17     mLoginEditor.commit();
18 }
19 ...
20 } //end LoginActivity
21

```

Listing 4.1: Memorizzare i dati di login in SharedPreferences

Alla pressione del tasto ACCEDI l'applicazione effettua una validazione dell'*email* e in caso positivo invia i dati tramite una richiesta HTTP GET alla risorsa `/login` al web service per confermare il login ed ottenere le informazioni relative all'utente ed ai suoi privilegi. Le informazioni utente, in caso risulti registrato, sono restituite dal server in formato JSON. Una



Figure 4.7: Login GestApp

volta ricevuti i dati si verifica che non si sia verificato un errore e in caso negativo i dati vengono deserializzati in un nuovo oggetto Java `UserInfo` e si viene rimandati alla `HomeActivity` (L'`Activity` principale) tramite un `Intent` memorizzando per la durata della sessione i dati dell'utente attuale.

```

1   private void goToHome( UserInfo user ) {
2       Intent registrationIntent = new Intent( getContext() ,
3           HomeActivity.class );
4       registrationIntent.putExtra( "actualUser" , user );
5       ((MyApp) this.getApplication() ).setCurrentUser( user );
6       startActivity( registrationIntent );
7       finish();
8   }

```

Listing 4.2: Intent che riporta alla HomeActivity

Il diagramma di sequenza illustrato nella Figura 4.8 mostra come avviene la comunicazione tra le componenti e come le informazioni vengono scambiate durante l'operazione di login. La `View` tra le componenti rappresenta lo schermata con cui l'utente interagisce ed è responsabile della gestione delle azioni dell'utente (vedi **Views** 2.8.1).

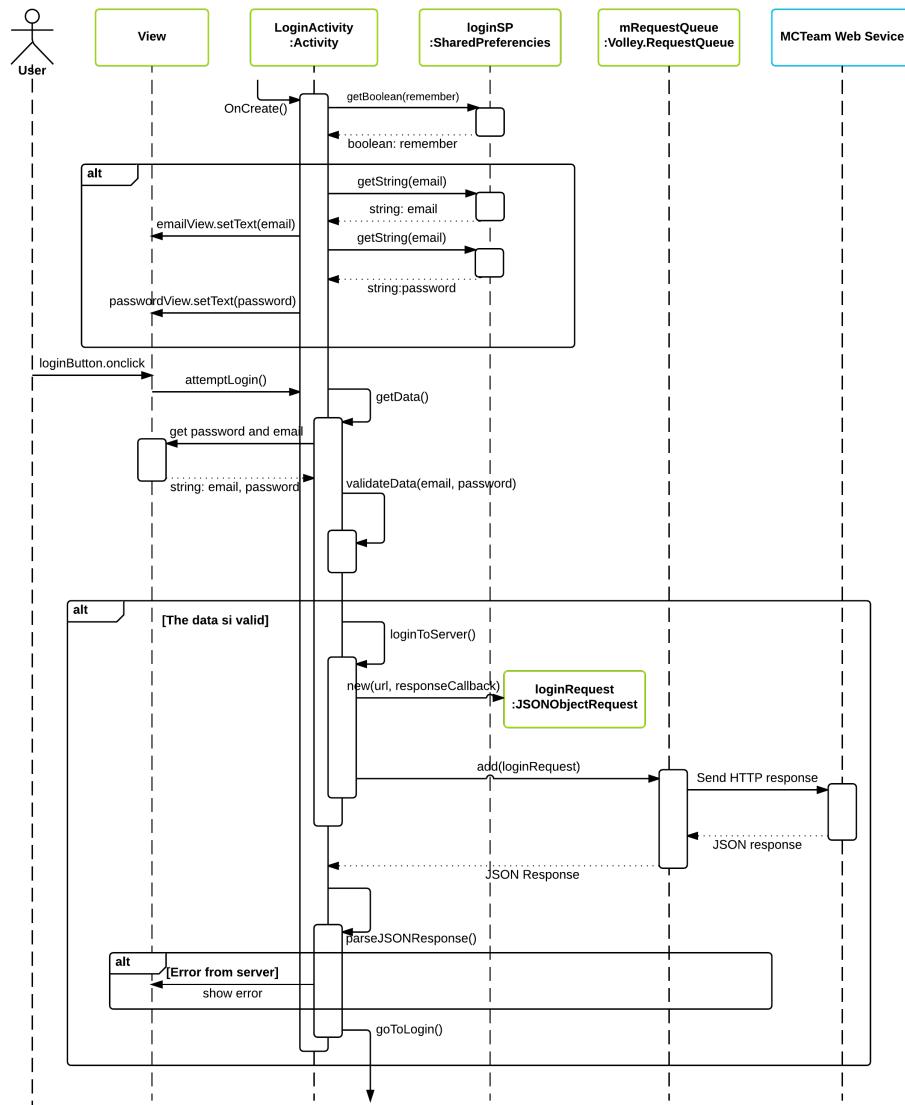
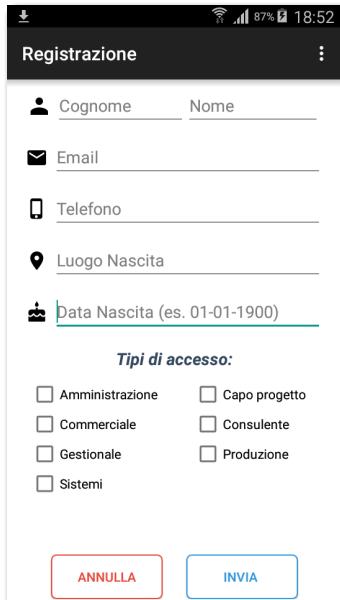


Figure 4.8: UML Login Sequence Diagramm

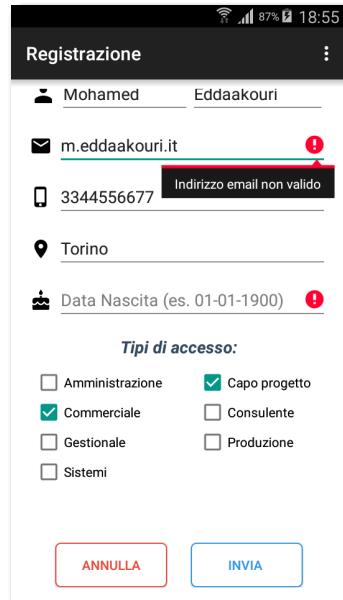
4.5.2 Registrazione

Nel caso in cui un collaboratore sia privo di credenziali, può richiederle all'amministrazione tramite la schermata di registrazione.

La schermata di registrazione è un form da compilare in cui, al momento della registrazione, il collaboratore inserisce i dati anagrafici ed i tipi di accesso (privilegi) di cui ha bisogno. Una volta inseriti tutti i dati e premuto il



(a) Schermata di registrazione



(b) Insuccesso validazione email

tasto INVIO verrà effettuata una validazione dei dati inseriti per verificare che tutti i dati siano corretti e in un formato accettabile. Un esempio è la validazione dell'indirizzo email:

```

1   // Check for a valid email address.
2   if (TextUtils.isEmpty(email)) {
3       mEmailView.setError(getString(R.string.
4           error_field_required));
5       focusView = mEmailView;
6       cancel = true;
7   } else if (!isValidEmail(email)) {
8       mEmailView.setError(getString(R.string.
9           error_invalid_email));
10      focusView = mEmailView;
11      cancel = true;
12  }
13  ...
14  ...
15  private boolean isValidEmail(String email) {
16      if (TextUtils.isEmpty(email)) {
17          return false;
18      } else {
19          return android.util.Patterns.EMAIL_ADDRESS.
20              matcher(email).matches();
21      }
22  }

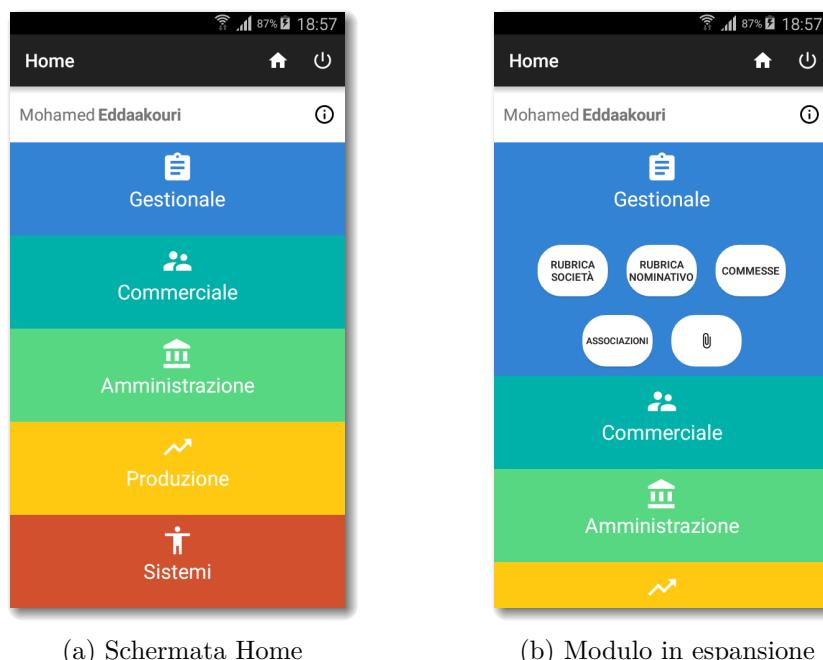
```

Listing 4.3: Validazione dati

Una volta validati i dati, verranno inviati via email all'amministrazione che provvederà a creare le credenziali per effettuare l'accesso.

4.5.3 Home

Una volta effettuato l'accesso il collaboratore/utente viene indirizzato alla schermata principale o la home (si ricorda che il passaggio tra le Activity avviene tramite Intent).



La schermata principale è composta da un menù suddiviso in sei sezioni o moduli. Ogni modulo è visibile solo a chi possiede i privilegi necessari per potervi accedere, questi privilegi sono assegnati al momento della registrazione. Ogni modulo raggruppa tools che possono essere utilizzati dai collaboratori. I vari *tools* risultano così raggruppati:

4.5.4 Tools

Tutti i tools, tranne il tool Consuntivi, presentano la stessa struttura e operazioni. Possiamo distinguere due categorie di operazioni: le operazioni su un'unico elemento e le operazioni su tutti gli elementi. Nella prima categoria troviamo la creazione, modifica, eliminazione e stampa, mentre nella

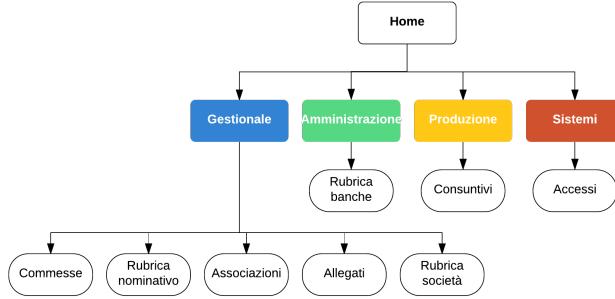


Figure 4.11: Suddivisione dei moduli

seconda troviamo la ricerca semplice e avanzata, la stampa di report in formato PDF e l'esportazione in formato Excel. Di seguito verranno analizzate solo le funzionalità rilevanti all'interno dell'applicazione.

Rubriche

Le rubriche sono uno strumento molto importante per i collaboratori. Esse contengono le informazioni relative a tutte le società (clienti e fornitori), ai nominativi (i collaboratori interni ed esterni) e alle banche. Le varie rubriche presentano la medesima interfaccia e operazioni. Le rubriche principali sono: la rubrica dei nominativi, rubrica delle società e rubrica delle banche.

(a) Rubrica nominativo

(b) Rubrica società

(c) Rubrica banca

La schermata principale delle rubriche (e anche gli altri principali tools) è formata da una lista di entità ottenute dal web service in formato JSON (Array JSON per l'esattezza) e poi deserializzate in oggetti Java utilizzando la libreria Gson.

```

1   // Callback class for the Volley Request
2   public class RubricaResponse implements Response.
3   Listener<JSONArray>{
4       ...
5       Gson gson = new Gson();
6       ...
7       @Override
8       public void onResponse(JSONArray responseArray) {
9           try {
10               ArrayList<Nominativo> nominativi = new ArrayList
11               <>();
12
13               for (int i = 0; i < responseArray.length(); i++)
14               {
15                   JSONObject response = responseArray.
16                   getJSONObject(i);
17                   // Deserializzazione
18                   Nominativo nominativo = gson.fromJson(response.
19                   toString(), Nominativo.class);
20
21                   nominativi.add(nominativo);
22               }
23           } catch (JSONException e) {
24               e.printStackTrace();
25           }
26       }
27   }

```

Listing 4.4: Metodo di callback per le Volley HTTP request e uso di Gson

Il Layout è composto da una `ListView` e ogni riga o 'item' è una risorsa (o un record delle relative tabelle nel database) ottenuta dal web service. Se si ha i privilegi da gestionale è possibile effettuare l'operazione di creazione di un nuovo elemento:

Commesse

Il tool commesse, permette di creare, modificare, visualizzare ed eliminare le informazioni relative ad una commessa. Permette anche il monitoraggio dell'avanzamento di una commessa nelle sue fasi.

Questo tool è accessibile solo a chi ha i privilegi di gestionale e presenta la stessa struttura di tutti gli altri tools sviluppati in precedenza con le stesse operazioni che si possono effettuare sulla singola entità ma anche sull'intero set di entità.

The figure consists of three side-by-side screenshots of a mobile application interface.
 (a) **Dati nominativo:** A form for creating a personal record. It includes fields for Title (Titolo: Sig.), Surname (Cognome), Name (Nome), Company (Società: dropdown with 'Scegliere...'), Address (Indirizzo), ZIP code (CAP), Province (Provincia: AG), City (Città), Nationality/Country (Nazionalità/Paese), Phone (Telefono), Fax, and Cellphone (Cellulare).
 (b) **Nuova società:** A form for creating a company record. It includes fields for Company Type (Tipologia: Cliente), Company Code (Codice società), Company Name (Nome società), Address (Indirizzo), City (Città), Province (Provincia: AG, CAP), Country (Stato), and Additional Information (Altre informazioni: Phone, Cellphone, VAT number - P. IVA).
 (c) **Nuovo banca:** A form for creating a bank record. It includes fields for Name (Nome), Address (Indirizzo), IBAN, Reference (Referente), and buttons for ANNULLA (red) and CREA (blue).

(a) Creazione nuovo ele- (b) Creazione nuovo ele- (c) Creazione nuovo ele-
mento nominativo mento società mento banca

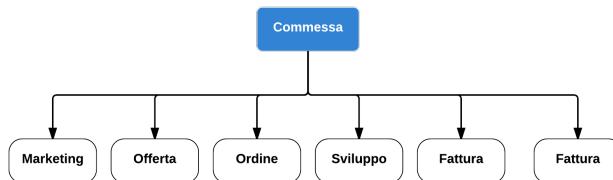


Figure 4.14: Fasi avanzamento di una commessa

Associazioni

Il tool Associazioni è lo strumento con il quale, chi si occupa del ruolo di gestionale (o gestione delle risorse umane o anche HR), utilizza per assegnare ad una commessa una o più risorse (generalmente consulenti o collaboratori). Alla creazione di un associazione è previsto anche l'inserimento della data di inizio e fine dell'associazione (utili durante consuntivazione). Durante la creazione di una nuova associazione è possibile scegliere contemporaneamente più risorse nel caso in cui vi siano più risorse assegnate alla commessa (Figura 4.16).

Le associazioni sono necessarie nella fase relativa ai consuntivi in quanto un consulente può consuntivare un periodo lavorativo su una commessa solo nel momento in cui viene associato a tale commessa.

Allegati

Il tool allegati è lo strumento con il quale il team del MCTeam può condividere documenti aziendali o report interni. Esso ha la stessa funzione di

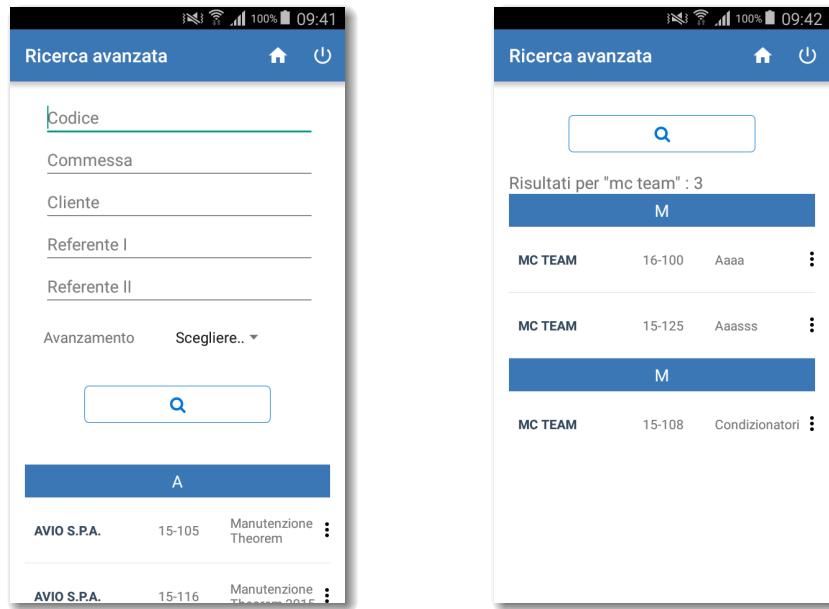


Figure 4.15: Ricerca Avanzata sulle commesse

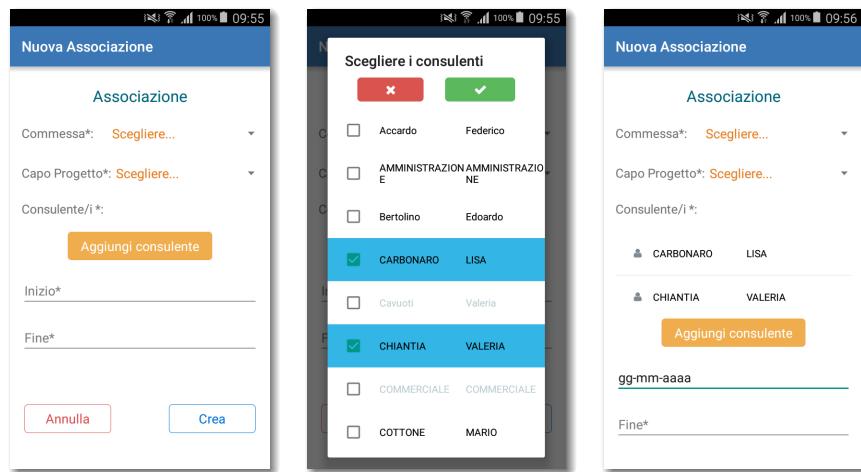


Figure 4.16: Creazione nuova associazione con scelta multipla

una cartella condivisa sul cloud (all'interno dei server dell'MC Team) a cui tutti i collaboratori vi possono accedere.

Questo tool inoltre permette la creazione un nuovo allegato effettuando l'upload del file sul server direttamente dal dispositivo mobile. E anche la possibilità di scaricare il file o di eliminarlo definitivamente.

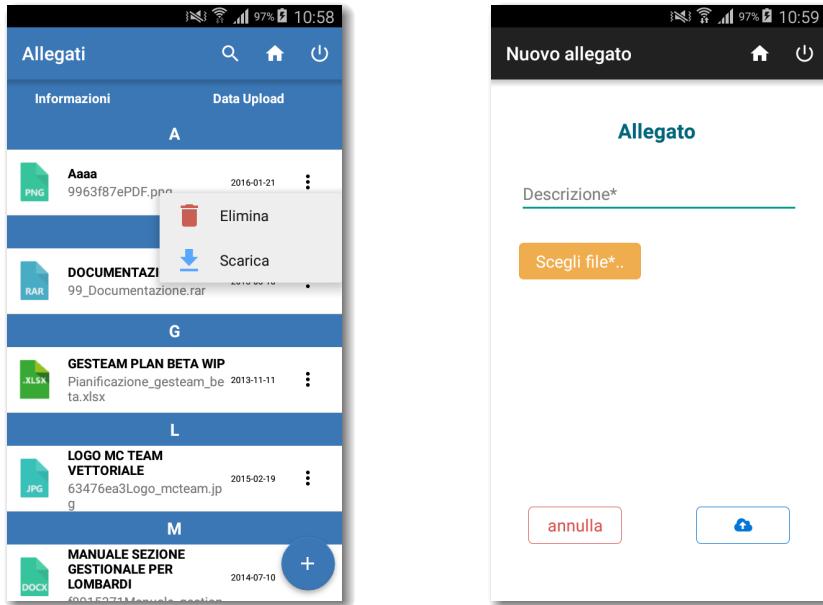


Figure 4.17: Tool Allegati

Consuntivi

Questo è il principale tool per cui è stato pensata la versione mobile: cioè uno strumento che permette la consuntivazione giornaliera immediata delle ore di lavoro di un consulente su una determinata commessa su cui è stato associato in precedenza tramite il tool associazioni. Questa è una delle funzionalità che rende efficiente questo applicativo in quanto permette la reperibilità istantanea della consuntivazione associata ai dipendenti; in fatti l'amministrazione non dovrà attendere di ricevere i consuntivi dai consulenti, poiché questi sono sempre accessibili in rete. Questo tool si differenzia dagli



Figure 4.18: Suddivisione in mesi

altri per la sua struttura grafica e le operazioni che si possono svolgere. Esso si presenta come un calendario diviso in mesi. Il numero di mesi in cui è suddiviso il calendario è ottenuto dalle associazioni in cui l'utente compare come consulente. Perciò il numero di mesi varia per ogni utente.

```
2     private void getCalendarRange() {
4
5         DateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
6
7         Date inizio = null;
8         Date fine = null;
9
10        for(Associazione associazione: mAssociazioni){
11            Date tempInizio;
12            Date tempFine;
13            try {
14                tempInizio = sdf.parse(ToolUtils.getFormattedDate(
15                    associazione.getData_inizio()));
16                tempFine = sdf.parse(ToolUtils.getFormattedDate(
17                    associazione.getData_fine()));
18            } catch (ParseException e) {
19                e.printStackTrace();
20                return;
21            }
22            if(fine == null && inizio == null){
23                inizio = tempInizio;
24                fine = tempFine;
25            }
26            else{
27                if(tempInizio.before(inizio)){
28                    inizio = tempInizio;
29                }
30                if(tempFine.after(fine)){
31                    fine = tempFine;
32                }
33            }
34        }
35    }
36
37    setupViewPager(mViewPager, inizio, fine);
38
39    //Fine metodo getCalendarRange
```

Listing 4.5: Ottiene il numero dei mesi dai range

Per permettere una più facile navigabilità all'interno del calendario ogni mese è stato rappresentato da `MonthCalendarFragment` (una estensione di `Fragment`) e quest'ultimo è composto da una `ListView` e ogni item o elemento della `ListView` rappresenta un giorno o meglio un oggetto `OrarioAttività` (il nome di questo oggetto deriva dalla denominazione della tabella nel database) (Figura 4.19). La transizione da un `Fragment`(mese) all'altro è gestita da un `PageViewer` che ci permette di passare da un `Fragment` ad un'altro.

```
private void setupViewPager(ViewPager vp, Date start, Date end) {
```

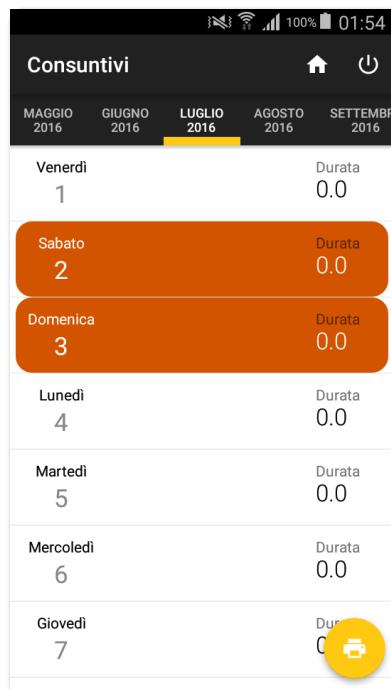


Figure 4.19: Fragment relativo al mese di Luglio

```
4     Calendar actuaDateFromStart = Calendar.getInstance();
5
6     actuaDateFromStart.setTime(start);
7     actuaDateFromStart.set(Calendar.DATE, 1);
8
9
10    SimpleDateFormat month_date = new SimpleDateFormat("MMM");
11
12    //Parse date to local date
13    LocalDate startDate = LocalDate.fromDateFields(start);
14
15    LocalDate endDate = LocalDate.fromDateFields(end);
16
17    //using JOBA Date library to get the difference in
18    months between two dates
19    Period period = new Period(startDate, endDate);
20    monthsNumber = (period.getYears() * 12) + period.
21    getMonths();
22
23    if(period.getDays() > 0){
24        monthsNumber++;
25    }
26
```

```

24         for( int i = 0; i < monthsNumber; i++){
25             ArrayList<OrariAttivita> attivitaPerMonth = new
26             ArrayList<>();
27
28             String month_name = month_date.format(
29                 actuaDateFromStart.getTime());
30
31             for( OrariAttivita attivita : mOrariAttivitaList){
32                 if( attivita.getMonth() == actuaDateFromStart.get(
33                     Calendar.MONTH) && attivita.getYear() ==
34                     actuaDateFromStart.get( Calendar.YEAR)){
35                     attivitaPerMonth.add( attivita);
36                 }
37             }
38
39             //Creating Fragment for each month
40             mSectionsPagerAdapter.addFragment(
41                 MonthCalendarFragment.newInstance(i, actuaDateFromStart.
42                     getTime(), mAssociazioni, actualUser), month_name + "\n"
43                     + actuaDateFromStart.get( Calendar.YEAR));
44
45             //incrementing month
46             actuaDateFromStart.add( Calendar.MONTH, 1);
47         }
48         vp.setAdapter( mSectionsPagerAdapter);
49         tabs.setupWithViewPager( mViewPager);
50     }

```

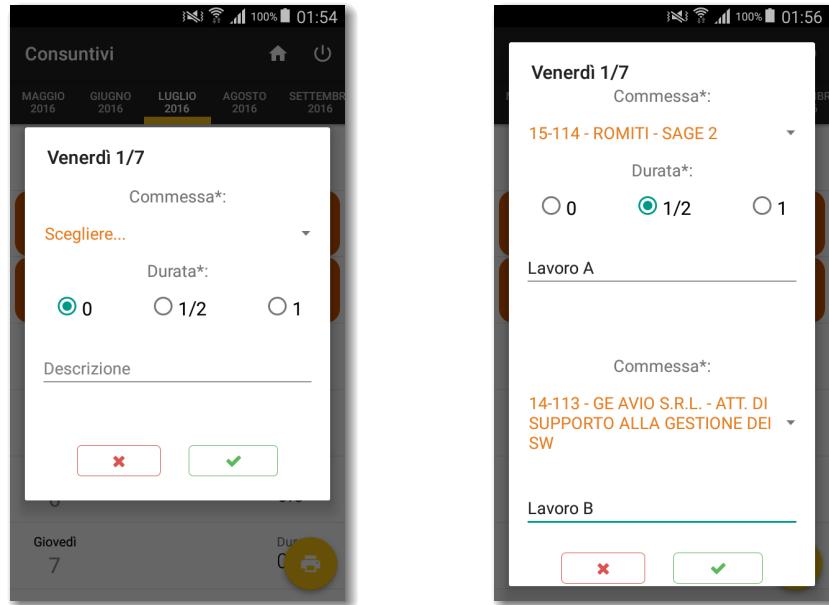
Listing 4.6: Ottiene il numero dei mesi dai range

I giorni in rosso identificano le giornate festive o ferie. I giorni festivi sono ottenuti da un semplice algoritmo che utilizza la classe `java Calendar` per ottenere i weekend, mentre le ferie sono ottenute dal database remoto.

È possibile compilare un consuntivo relativo ad una giornata premendo direttamente sulla riga relativa al giorno.

Alla pressione sul un item o elemento della `ListView` si apre un `Dialog` che ci permette di inserire le informazioni relative alla giornata. Il `Dialog` è composto da un form e tra gli input da inserire vi è la commessa, la durata e la descrizione come si vede nelle figure ???. Le commesse che un consulente può scegliere e consuntivare sono solo quelle in cui compare come risorsa in quel determinato periodo. La durata può essere:

- 0 : nessuna attività svolta(nella descrizione di deve indicare la motivazione).
- $\frac{1}{2}$: mezza giornata lavorativa.



(a) Compilazione consuntivo giornaliero

(b) Compilazione mezza giornata

- 1 : intera giornata lavorativa.

Nel caso in cui il consulente selezioni solo mezza giornata($\frac{1}{2}$) sarà per lui possibile compilare una parte relativa alla rimanente mezza giornata nel caso in cui abbia svolto un’ulteriore attività su un’altra commessa (nell’arco della stessa giornata). Al salvataggio le informazioni vengono inviate al web service per essere memorizzate all’interno del database.

Questo tool permette anche la creazione di un report in PDF per ogni mese su ogni commessa assegnata al consulente. Questo report viene generato utilizzando la libreria iTextPDF.

Accessi

Gli accessi è uno strumento con il quale si creano le credenziali di accesso per poter accedere all’applicazione e si assegnano eventuali privilegi per poter accedere ai tools. I privilegi possono essere assegnati a più figure aziendali contemporaneamente. Chi ha accesso a questo tool può creare un nuovo utente oppure modificare, eliminare o disabilitare qualunque utente registrato all’interno del sistema. Presenta le stesse operazioni delle rubriche di cui ho discusso precedentemente.

Gestteam

REPORT CONSUNTIVO COMMESSA

ID	Commessa	Cliente	Risposta
15-114	SAGE 2	RCMTH	Mohamed

Data	Ora/Prezzo	Descrizione	Prezzo
Venerdì 11 Lug.	0.0		0.0
Sabato 12 Lug.	0.0		0.0
Domenica 13 Lug.	0.0		0.0
Lunedì 14 Lug.	0.0		0.0
Martedì 15 Lug.	0.0		0.0
Mercoledì 16 Lug.	0.0		0.0
Giovedì 17 Lug.	0.0		0.0
Venerdì 18 Lug.	0.0		0.0
Sabato 19 Lug.	0.0		0.0
Domenica 20 Lug.	0.0		0.0
Lunedì 21 Lug.	0.0		0.0
Martedì 22 Lug.	0.0		0.0
Mercoledì 23 Lug.	0.0		0.0
Giovedì 24 Lug.	0.0		0.0
Venerdì 25 Lug.	0.0		0.0
Sabato 26 Lug.	0.0		0.0
Domenica 27 Lug.	0.0		0.0
Lunedì 28 Lug.	0.0		0.0
Martedì 29 Lug.	0.0		0.0
Mercoledì 30 Lug.	0.0		0.0
Giovedì 31 Lug.	0.0		0.0

TOTALE MESE: 0.0 | 0.0 | 0.0

Figure 4.21: Report mensile su una commessa nel mese di Luglio

Accessi

TIPOLOGIA COGNOME NOME

(a)

- GES, PRO. Accardo Federico
- AMM, COMM. AMMINISTRAZIONE AMMINISTRAZIONE
- GES, PRO. Bertolino
- AMM, COMM. CARBONARO LISA
- AMM, COMM. Cavuoti Valeria
- AMM, COMM. CHIANTIA VALERIA

B

- Modifica
- Elimina
- Stampa

C

Accessi

TIPOLOGIA COGNOME NOME

A

- GES, PRO. Accardo Federico
- AMM, COMM. AMMINISTRAZIONE AMMINISTRAZIONE

B

- Ricerca Avanzata Edoardo
- Stampa tutto

C

- Esporta in Excel
- Aggiungi nuovo

(a) Tool Accessi

(b) Menu Accessi

Capitolo 5

Tecnologie utilizzate

In questo capitolo verranno analizzati gli strumenti tecnologici utilizzati durante l'implementazione dell'applicativo.

5.1 Lato server

A lato server sono stati utilizzati principalmente strumenti largamente utilizzati per il *rapid development*. In questo progetto è stato utilizzato Slim Framework: un Micro Framework¹ PHP molto leggero (slim per l'appunto) ma che offre alcune features che sono estremamente utili alla realizzazione di una web application e API, IDIORM una libreria lightweight di pochi KiloByte utile nella gestione del database e nella generazione di query SQL.

5.1.1 Slim Framework

Slim Framework è un Mirco Framework, interamente in linguaggio PHP, pensato per velocizzare lo sviluppo di web applications e API. Slim si presta ad essere il miglior candidato per la realizzazione una piccola/media RESTful API, in quanto funge da dispatcher che riceve le richieste HTTP, richiama la funzione di callback appropriata, e restituisce una risposta HTTP. Come ogni Mirco Framework Slim offre un set minimo di tools o strumenti necessari allo sviluppo di una web application. Uno di questi strumenti, che è stato utilizzato principalmente durante lo sviluppo del web service, è il HTTP Routing. Il HTTP Routing è un tool molto sofisticato che ci permette di mappare per ogni metodo HTTP una funzione di callback che verrà invocata al momento della ricezione della richiesta. Questo stralcio di codice illustra tutta la sua semplicità:

¹ <?php

¹A microframework is a term used to refer to minimalistic web application frameworks. It is contrasted with full-stack framework also called enterprise frameworks.
<https://en.wikipedia.org/wiki/Microframework>

Operazione	Metodo HTTP	Metodo Slim
CREATE	PUT	put()
RETRIEVE	GET	get()
UPDATE	POST	post()
DELETE	DELETE	delete()

Table 5.1: Tabella operazioni CRUD associate ai metodi HTTP e metodi offerti da Slim

```

3 $app = new \Slim\App;
5 $app->get('/hello/{name}', function ($args) {
    echo "Hello " . $args['name'];
7 }
9 $app->run();
?>
```

Listing 5.1: Slim Framework primo approccio

Questo è un esempio di un'applicazione Slim molto semplice, che mappa per il metodo HTTP GET sul URL `http://dominio/hello/[nome]` una funzione di callback che stampa Hello [nome].

Slim e CRUD

Per ogni metodo HTTP, Slim offre un metodo appropriato per la sua gestione (Tabella 5.1).

Questo ci è di fondamentale aiuto in varie gamme di operazioni che bisogna effettuare. Di seguito vedremo le quattro operazioni CRUD come sono state implementate nel caso della rubrica banca.

5.1.2 IDIORM

IDIORM è un altro strumento che è stato largamente utilizzato in questo progetto per velocizzare lo sviluppo della web service. Esso consiste in una libreria estremamente leggera, che ci aiuta nella gestione del database con un'interfaccia semplice per la creazione di query SQL in modo quasi automatico ”senza la necessità di scrivere una riga di codice SQL” come sottolinea lo sviluppatore di IDIORM.

```

$allegati = ORM::for_table('allegati')->find_array();
2
```

Dove:

- `ORM`: la classe ORM precedentemente configurata.
- `for_table()`: il metodo che specifica la tabella su cui si opera. 'allegati' rappresenta la tabella allegati nel nostro database.
- `find_array()`: esso trasforma il risultato della query in un array.

Questa riga di codice è un ottimo esempio di come IDIORM sia estremamente compatto e semplice da utilizzare. Inoltre offre un supporto per le famose operazioni CRUD.

5.2 Lato client

Gli strumenti utilizzati per sviluppare l'applicazione client sono stati principalmente Android Studio IDE: di seguito si farà una descrizione delle librerie che sono state utilizzate per lo sviluppo.

5.2.1 Librerie

Per sviluppare l'applicazione client sono state utilizzate molte librerie Java. Di seguito verranno illustrate solo quelle principali e più utilizzate.

Volley

Volley è una libreria HTTP molto potente rilasciata da Google. Questa libreria è stata pensata per facilitare e velocizzare il networking per le applicazioni Android offrendo i seguenti vantaggi:

- Uno scheduling automatico per le richieste HTTP;
- Connessioni multiple concorrenti;
- Un caching delle risposte HTTP;
- Supporto per la prioritizzazione delle richieste HTTP;
- API per la cancellazione delle richieste;
- Strumenti di debug e tracing.

Per iniziare ad usare Volley è necessario creare una `RequestQueue` e passargli gli oggetti Request come argomento. La `RequestQueue` gestisce i threads responsabili dell'esecuzione delle operazioni di networking e parsifica le risposte in entrata. Volley offre il metodo `Volley.newRequestQueue` che genera in automatico una `RequestQueue` inizializzata con valori di default.

```

1      // Request queue declaration
3      private RequestQueue mRequestQueue;
5
5      protected void onCreate(Bundle savedInstanceState) {
6          ...
7          //RequestQueue initialization
8          mRequestQueue = Volley.newRequestQueue(this);
9          ...
10     }
11
11     private void tryLoginOnline(String username, String
12         password){
13         ...
14         //Add the new request to queue
15         mRequestQueue.add(loginRequest);
16         ...
17     }
18
19

```

Listing 5.2: Volley in action

Per aggiungere una nuova richiesta alla coda cioè alla `RequestQueue`, si utilizza il metodo `add()`. Una volta aggiunta la richiesta, essa viene prioritizzata (per l'ordine di entrata in coda per default).

Gson

Gson è una libreria Java, rilasciata da Google, che può essere usata per convertire Java Objects nella loro rappresentazione JSON e per convertire stringhe JSON in un equivalente Java Objects. Per effettuare queste operazioni di conversione si utilizzano rispettivamente due metodi: `toJSON()` e `fromJSON()`.

```

//Deserialization from JSON to Commessa Java Object
2     Commessa commessa = gson.fromJson(response.toString(),
3                                         Commessa.class);

4     //Serialization from Java Object Commessa to JSON
5     String json = gson.toJson(commessa);
6

```

Bibliography

- [1] R. Fielding. 2000. Architectural Styles and the Design of Network-based Software Architectures, PhD thesis, University of California, Irvine, USA
- [2] W. Frank Ableson, Charlie Collins, Robi Sen. 2009. Unlocking Android - A Developer's Guide. Manning.
- [3] Wikipedia (2016). Android. <https://it.wikipedia.org/wiki/Android>
- [4] Nicola D'Ambrosio. 2010/2011. Relazione di stage: Sviluppo ed integrazione di un'applicazione Web-based per la gestione delle risorse aziendali.
- [5] Alex Dobie, Russell Holly, Jerry Hildenbrand and Andrew Martonik. Last access 05/07/2016. Android Story. <http://www.androidcentral.com/android-pre-history>