ALMA MATER STUDIORUM

UNIVERSITÀ DEGLI STUDI DI BOLOGNA

FACOLTÀ DI INGEGNERIA

Corso di Laurea in Ingegneria elettronica e telecomunicazioni

Tesi di Laurea in Software di Telecomunicazioni

Sviluppo di applicazioni Android per promozione e gestione degli ordini in ambito ristorazione

Candidato:

Filippo Fuochi

Relatore:

Prof. Giovanni Emanuele Corazza

Correlatori:

Prof. Alessandro Vanelli-Coralli

Dott. Ing. Matteo Collina

Anno Accademico 2012 / 2013

Indice

Introduzione

Il mercato della telefonia mobile, e in particolare degli smartphone, in Italia è in continua evoluzione, ad oggi circa 27 milioni di persone possiedono uno smartphone e più dell'80% lo utilizzano regolarmente per la navigazione. I ricavi della connettività mobile sono triplicati in tre anni, anche perché gli utenti che utilizzano il proprio dispositivo mobile per accedere alla rete sono ormai pari al 75% degli utenti internet da Pc.
Nel 2012 la navigazione su internet da dispositivi mobili è aumentata del 53% e il mercato delle applicazioni e dei contenuti digitali distribuiti tramite cellulare è incrementato del 20% aumentando i ricavi di tale mercato dell'83% nel 2012 e con una stima di incremento per il 2013 del 50%.
Il mercato complessivo italiano delle applicazioni e dei contenuti digitali distribuiti sui telefoni cellulari vale 623 milioni di euro. Oltre un settimo di tale rendita è dovuta agli investimenti pubblicitari. Non sono solo le applicazioni e i contenuti digitali a trarre beneficio dall'aumento dell'utilizzo degli smartphone, anche i ricavi generati dai siti Mobile sono aumentati del 27%. Sempre più italiani utilizzano il proprio telefono cellulare per accedere a contenuti digitali di qualsiasi natura, questo mercato offre grandi possibilità di business a sviluppatori e startup [ITess].
La disponibilità ad accedere alla rete da una parte così ampia della popolazione dà la possibilità a molti altri settori di aprirsi a questo mondo, utilizzando la rete come strumento pubblicitario o di supporto al proprio lavoro.
Si vuole quindi realizzare un'applicazione, per dispositivi mobili, il cui scopo sia quello di promuovere ed espandere le possibilità commerciali di un'azienda nel settore ristorativo.
Per realizzare tale applicazione sarà necessario fare opportune valutazioni sul target di mercato e l'ambiente di sviluppo.
Nei capitoli 1 e 2 verranno descritti i sistemi operativi mobili più utilizzati in questo momento e, successivamente, illustrate le scelte riguardo quale sia più adatto allo sviluppo di questo tipo di applicazione. Nei capitoli 3 e 4 verrà descritta

l’applicazione stessa ed i suoi requisiti, soffermandosi poi sulle soluzioni da adattare e le varie opportunità.
Infine, nel capitolo 5, verrà descritto lo sviluppo dell’applicazione e degli strumenti utilizzati per realizzarla.

Capitolo 1
Sistemi operativi mobili

La distribuzione di smartphone è molto ampia ed in continua fase di crescita ma non tutti gli smartphone sono uguali e, in particolare, ogni casa produttrice di smartphone sceglie se utilizzare un sistema operativo per dispositivi mobili sviluppato da altri, come Google o Microsoft, o se svilupparselo internamente, come Apple o BlackBerry. Questi sistemi operativi hanno strutture e ambienti di sviluppo dedicati molto diversi.
Prima di introdursi in un mercato così ampio come quello delle applicazioni mobili, è necessario conoscerne gli aspetti e comprendere le differenze esistenti tra i vari sistemi operativi, in modo da poter scegliere per quale piattaforma, o quali piattaforme, sviluppare le proprie applicazioni.

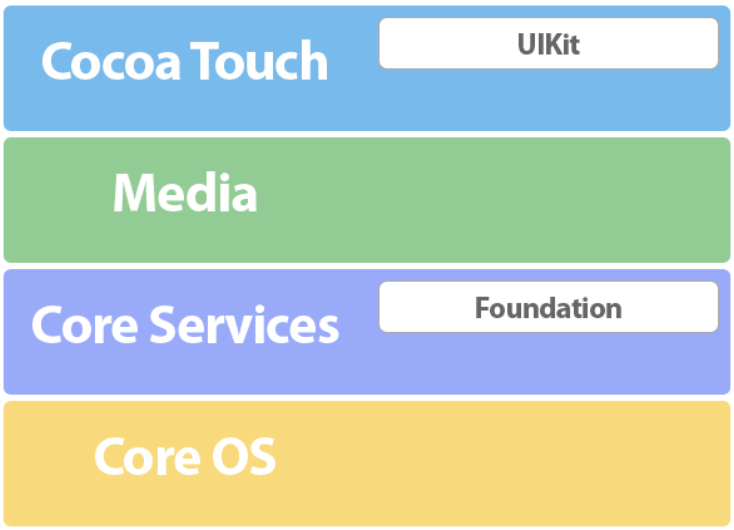
1.1 iOS

I dispositivi Apple, in particolare con iPhone e iPad, sono entrati in possesso di una buona fetta di mercato mondiale in pochi anni ed entrambi utilizzano come sistema operativo Apple iOS.
iOS è utilizzato solo dai prodotti Apple ed è quindi implementato al meglio per funzionare su tali dispositivi.
iOS garantisce ottime prestazioni ed è stato preso come punto di riferimento da altri sistemi operativi mobili. La Apple continua a migliorare le prestazioni di Unix con le ultime versioni del sistema operativo, migliorando, in particolar modo, le prestazioni del proprio browser.
La gestione della batteria d'altro canto non sembra in grado di soddisfare l'aumento di richiesta, è soprattutto con l'inserimento di numerosi servizi online, come dell'assistente personale virtuale (Siri), che consumano molta energia.

iOS è un sistema operativo chiuso ma dà la possibilità agli sviluppatori di programmare nel loro ambiente di sviluppo e fornisce documentazione completa di supporto Il linguaggio di sviluppo di iOS è Objective-C [MOSAT].

1.1.1 Architettura

La struttura del sistema è di tipo stratificato ed è possibile dunque immaginarla come una serie di strati, uno sopra l'altro, che implementeranno determinate funzioni. Dunque ogni strato sarà delegato a gestire ed implementare determinate funzionalità e in sostanza lo strato superiore beneficia dei servizi offerti dallo strato inferiore senza però saperne la logica implementativa (vedi fig. 1.1).



(Fig. 1.1) Architettura iOS [Apple].

Il Core Os è lo strato che lavora più a basso livello, il cuore del sistema operativo. Infatti, in questo strato sono gestiti i file system, le funzioni per la sicurezza del dispositivo e il risparmio energetico. Quest'ultimo, in particolare, consente di gestire nella maniera più efficace possibile l'energia messa a disposizione dalla batteria del dispositivo, minimizzando gli sprechi. Nei dispositivi come iPhone e iPod Touch dove la quantità di energia disponibile è molto limitata risulta di vitale importanza una gestione ottimale.

Il Core Services implementa le utilità di sistema come la gestione del networking, la lista dei contatti e le preferenze di sistema inserite dall'utente. Da questo strato vengono inoltre gestite le operazioni svolte sui database, dunque un database *SQLite*

implementato in un’applicazione lavora a questo livello e dunque l'accesso e le interrogazioni fatte al database sono gestite proprio da questo strato.
Media rappresenta lo strato che contiene tutte le funzionalità e le librerie per la gestione di video, audio, animazioni e tanto altro.
In questo strato, sono ubicate le librerie *OpenAL* per la gestione e la manipolazione di flussi audio e le librerie *OpenGL ES* per produrre grafica 2D con animazioni anche molto complesse.
Cocoa Touch è lo strato che lavora più ad alto livello. Si occupa della gestione della gerarchia delle view, della fotocamera del dispositivo, e delle alert, della gestione e del riconoscimento del touch e del multi-touch dell'utente ed è in grado di interpretare, in maniera corretta, le *gesture* compiute dall'utente.
Per gesture si intendono quei movimenti composti da una o più dita come l’avvicinamento e l’allontanamento di due per rimpicciolire o ingrandire un’immagine o lo scorrimento verticale o orizzontale. Quando questo strato interpreta uno di questi movimenti, richiamerà una routine di gestione specifica per quel movimento.
Questo è quindi il livello più a contatto con gli sviluppatori software.
Oltre a gestire il touch dell'utente, questo strato si occupa di gestire funzionalità come l'accelerometro ed il giroscopio riuscendo dunque a capire in che modo è orientato il dispositivo rispetto ad un asse orizzontale.
[Apple].

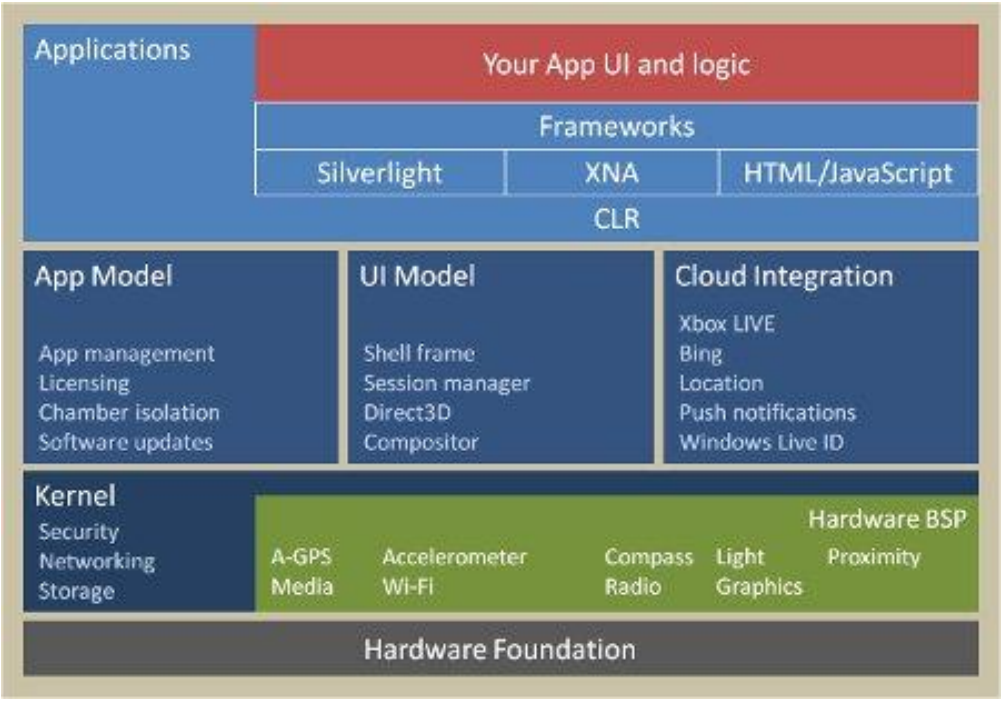
1.2 Windows Phone

L’ultimo sistema operativo rilasciato da Microsoft è Windows Phone. Questo sistema utilizza un design innovativo e differente da quello di tutti gli altri sistemi operativi mobili.
L’interazione con l’utente è touchscreen e, di conseguenza, le applicazioni scritte per Windows Phone non sono compatibili con Windows Mobile, la precedente versione sul mercato.
Windows Phone inventa il concetto di Live Tiles nella home screen, concetto che poi verrà utilizzato anche per Windows 8. Le Live Tiles rappresentano dei riquadri simili

ad icone ma altamente personalizzabili, esse possono rappresentare collegamenti ad applicazioni, a contatti, a siti web o altro ancora.
Dalle precedenti versioni, Windows Mobile e Windows CE, è migliorata di molto anche la gestione del consumo della batteria. Per risparmiare in tale senso è stato anche scelto come tema di default il colore nero in modo da consumere meno su schermi OLED.
I linguaggi di programmazione utilizzati sono C# e Visual Basic [MOSAT].

1.2.1 Architettura

Windows Phone 8, l'ultima versione del sistema operativo rilasciata, possiede un'architettura a strati e utilizza una variante kernel NT di Windows 8 adattata alle esigenze della telefonia mobile, quindi con meno funzioni rispetto al kernel di Windows.



(Fig. 1.2) Architettura Windows Phone [WiPDC].

Il kernel si occupa della gestione della rete, della sicurezza e di altro. Quindi i due sistemi condividono il Windows Core System, il livello più basso dell'architettura di entrambi. Il Mobile core è il livello successivo dell'architettura, esso comprende aree dove le API sono le stesse di Windows anche se possono essere implementate differentemente, a questo livello viene gestita soprattutto l'interazione con i file

multimediali. Salendo di livello si trova il sistema di Windows Phone che si compone del Connection Management, dei Platform Services, delle System Applications e della Windows Phone Shell.
I Platform Services che vengono forniti a tutte le applicazioni sono quattro:

- Il Package Manager che gestisce il ciclo di vita di un'applicazione dall'istallazione alla disinstallazione.
- L'Execution Manager che gestisce le applicazioni e i processi in background, è il servizio che gestisce la creazione di processi host e di messaggi di stato.
- Il Navigation Server che gestisce l'interazione tra le applicazioni in primo piano sul telefono, lo fa comunicando all'Execution Manager quale applicazione lanciare e tenendo traccia della navigazione.
- Il Resource Manager che monitora le risorse di sistema di tutti i processi attivi e impone loro i vincoli necessari, soprattutto sull'utilizzo di CPU e memoria, se l'applicazione smette di funzionare il Resource Manager può terminarla in modo da garantire la stabilità del dispositivo.

Il livello superiore al kernel è diviso in tre parti: l'App Model, l'UI Model e il Cloud Integration. Queste tre componenti si occupano rispettivamente: della gestione delle applicazioni e del software nativo, della gestione dell'interfaccia grafica, e dell'interazione con i servizi cloud.
Il linguaggio di programmazione scelto da Windows per il proprio sistema operativo è C# e gli sviluppatori software dovranno utilizzare Sirverlight, XNA e/o HTML/Javascript per la programmazione per questa piattaforma [WiPDC].

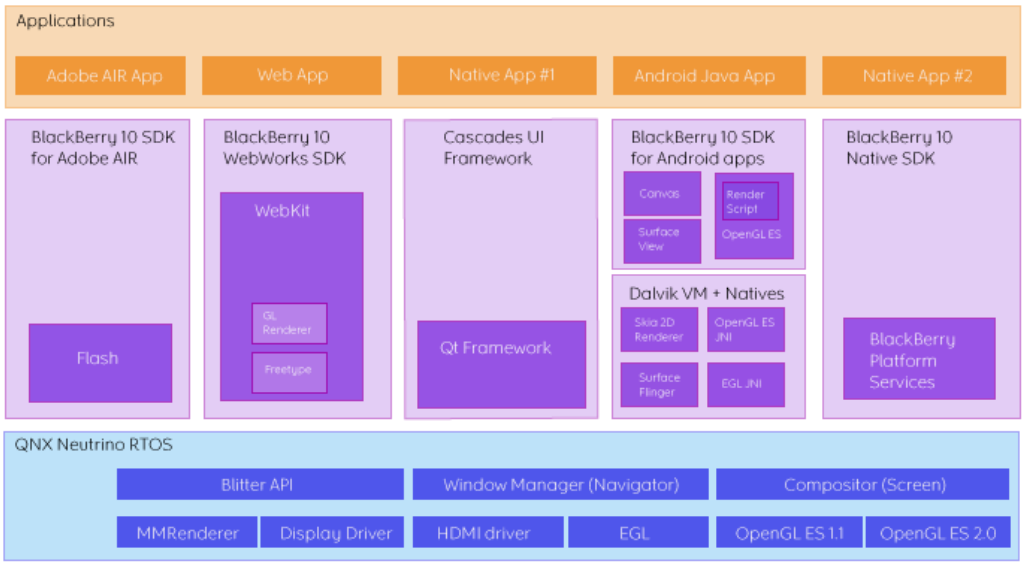
1.3 BB

BlackBerry ha subito molti cambiamenti negli ultimi anni rendendo l'ultimo sistema operativo molto differente dalle versioni precedenti, soprattutto perché si è scelto di abbandonare la classica tastiera che contraddistingueva i propri dispositivi e di passare al touchscreen, in particolare l'ultima versione presenta numerose somiglianze con Android, soprattutto nella gestione dei Widget sulla home.
Ciononostante il target del prodotto non è cambiato, BlackBerry continua a progettare telefoni e sistemi operativi con l'idea di fare poche cose ma fatte bene,

come la gestione della posta o del calendario, rendendo tale telefono adatto a chi lo utilizza soprattutto come agenda di lavoro. Sono presenti anche dei giochi ma è chiaro che tale sistema operativo non è pensato come oggetto ludico.
Le applicazioni sono scritte in Java tramite l'SDK fornito da BlackBarry da integrare con Eclipse, anche se i recenti Widget possono essere creati utilizzando html, css e javascript.
Dato che l'ambiente di sviluppo ed il linguaggio di programmazione sono gli stessi di Android è possibile convertire, semplicemente e velocemente, applicazioni scritte per Android in applicazioni per BB

1.3.1 Architettura

BlackBerry 10 viene reinventata la struttura del sistema operativo BlackBerry per dare vita ad una piattaforma completamente nuova (vedi fig. 1.3).



(Fig. 1.3) Architettura BB 10 [BlaBa].

La nuova ossatura portante del sistema è composta dal QNX Neutrino, che rappresenta il cuore vero e proprio dell'architettura BB10. Si tratta di un sistema operativo real time ad altissime prestazioni, usato spesso in applicazioni mission critical per la sua capacità di reagire immediatamente agli stimoli provenienti dall'esterno con efficienza e precisione. Grazie all'utilizzo del QNX Neutrino BlackBerry garantisce un'esperienza estremamente fluida. QNX è efficiente

soprattutto nella gestione del multitasking e del risparmio energetico. Ma più importante e significativo è il supporto alle tecnologie di mobile computing esistenti, come le OpenGL ES 2.0.
BlackBerry 10 è stato progettato per essere il sistema più flessibile concepibile sul mercato e può vantare il supporto a molte librerie di programmazione attualmente in circolazione come: Adobe AIR / Flash; HTML5 + CSS + Javascript, per cui mette a disposizione degli sviluppatori delle librerie ottimizzate; C e C++ a basso livello anche utilizzando il Cascades Framework per l'interfaccia grafica; e, infine è pienamente compatibile, BlackBerry 10 integra al suo interno la stessa macchina virtuale Java di Android ed è quindi in grado di eseguire le applicazioni Android scritte in Java [BlaBa].

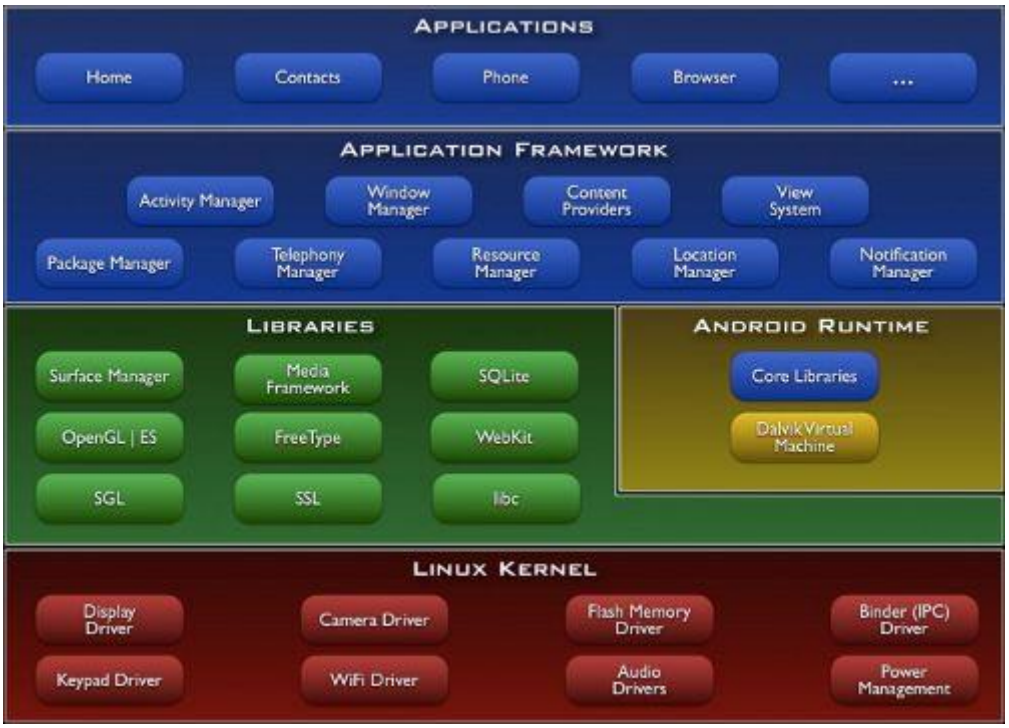
1.4 Android

Android un progetto Open Source che ha come obbiettivo quello di creare un prodotto che possa migliorare l'esperienza della telefonia all'utente finale. Per realizzare tale scopo il team che gestisce il design di Android basa il proprio lavoro su tre principi "Enchant me", "Simplify my Life" e "Make me amazing". Questa gestione del progetto ha portato Android ad essere il sistema operativo che ha avuto una crescita maggiore negli ultimi anni.
La Google ha rilasciato il codice di Android come open source sotto licenza Apache, anche se Google, tipicamente, collabora con una o più aziende selezionate per avere un telefono di bandiera per ogni nuova versione di Android e pubblica il codice solo dopo l'uscita in commercio del dispositivo.
La realizzazione delle nuove applicazioni viene effettuata tramite l'Android SDK integrabile con Eclipse utilizzando Java come linguaggio di programmazione [MOSAT].

1.4.1 Architettura

La piattaforma Android è ben strutturata e stratificata (vedi fig. 1.4).

Il cuore del sistema è basato sul kernel di Linux (versione 2.6) che costituisce il livello di astrazione di tutto l'hardware sottostante che può includere Wi-Fi, Bluetooth, GPS, fotocamera, touchscreen… I produttori di telefoni possono quindi intervenire già a questo livello per personalizzare i driver di comunicazione con i propri dispositivi. Grazie all'astrazione dell'hardware, infatti, i livelli soprastanti non si accorgono dei cambiamenti hardware, permettendo una programmazione ad alto livello omogenea ed una *user experience* indipendente dal dispositivo.
Il livello superiore riguarda l'equipaggiamento software costituito dalle librerie fondamentali che gestiscono per esempio la grafica 2D e 3D come OpenGL ES, browser con engine Web Kit o il supporto al database come SQLite.
L'ambiente di runtime è costituito invece da una libreria *core* e da una macchina virtuale (VM): insieme costituiscono la piattaforma di sviluppo per Android.



(Fig. 1.4) Architettura Android [AndDe].

La VM di Android è versione particolare della Java Virtual Machine, chiamata Dalvik, DVM, progettata e ottimizzata appositamente per essere eseguita su hardware non performanti, come quelli dei cellulari appunto. In realtà gli ultimi modelli che montano Android vantano processori dual core con 1 Gb di RAM per cui questa affermazione comincia già ad essere obsoleta.

Il kernel di Linux di Android è un sistema multi-utente nel quale *ogni applicazione è un utente differente*. Il sistema, infatti, crea un unico user ID per ogni applicazione, sconosciuto ad essa, e imposta i permessi dei file dell'applicazione stessa in modo che solo quell'ID possa averne accesso. Inoltre, ogni applicazione sul telefono è lanciata in un processo Linux a sé stante all'interno della propria istanza della Java VM: questa architettura garantisce la stabilità del telefono nel caso in qui qualche applicazione crei problemi. In caso alcune applicazioni abbiano necessità di comunicare tra loro, il sistema non lo impedisce: permette anzi loro di condividere lo stesso user ID e la stessa Java VM in modo da preservare la coerenza delle risorse di sistema.
La gestione della batteria di Android ha risultati simili a quelli di iOS anche se Android sospende il dispositivo e, in alcuni casi, i processi, quando non usati, per poter risparmiare energia.
Al penultimo livello è possibile rintracciare i gestori e le applicazioni di base del sistema. Ci sono gestori per le risorse, per le applicazioni installate, per le telefonate, per i file system e altro ancora.
Al livello più alto risiedono le applicazioni utente. Le funzionalità base del sistema, non sono altro che applicazioni utente scritte in Java e che girano ognuna nella sua Java VM. Da notare che viene eseguita una versione di Java appositamente sviluppata per Dalvik: il codice Java infatti viene compilato e tradotto in byte code, dopodiché subisce una ulteriore trasformazione in un formato chiamato *dex file* [AndDe].

1.5 Considerazioni

Tutti questi sistemi operativi forniscono gli sviluppatori di strumenti per la realizzazione di applicazioni adeguati alla piattaforma e il più flessibile è sicuramente l'ambiente di sviluppo di BB che permette di utilizzare diversi linguaggi di programmazione.
Il mercato non è però altrettanto clemente con tutti, di fatti Android al momento è il sistema operativo per dispositivi mobili più diffuso, è in continua crescita ed è disponibile ad un numero maggiore di terminali, dato che, secondo una ricerca di

Kantar, una compagnia gestita da WPP Plc, multinazionale inglese che studia l'economia mondiale in vari settori, Android, in Italia, registra un aumento delle vendite in percentuale, rispetto agli altri sistemi operativi mobili, di quasi 7 punti rispetto al trimestre precedente e di quasi 19 punti rispetto allo stesso periodo dell'anno precedente (vedi fig. 1.5).

Smartphone OS Sales Share (%)

Germany	3 m/e Apr 2012	3 m/e Apr 2013	% pt. Change
iOS	18.6	16.6	-2.0
Android	68.1	74.3	6.2
BlackBerry	1.6	0.4	-1.2
Symbian	6.1	2.0	-4.1
Windows	4.1	6.4	2.3
Other	1.5	0.3	-1.2

GB	3 m/e Apr 2012	3 m/e Apr 2013	% pt. Change
iOS	30.3	28.7	-1.6
Android	50.1	56.9	6.8
BlackBerry	13.6	5.6	-8.0
Symbian	1.5	0.3	-1.2
Windows	4.0	8.4	4.4
Other	0.5	0.1	-0.4

France	3 m/e Apr 2012	3 m/e Apr 2013	% pt. Change
iOS	18.9	20.1	1.2
Android	57.9	64.2	6.3
BlackBerry	6.4	3.7	-2.7
Symbian	5.7	1.4	-4.3
Windows	2.5	6.2	3.7
Other	8.5	4.4	-4.1

Italy	3 m/e Apr 2012	3 m/e Apr 2013	% pt. Change
iOS	22.2	16.6	-5.6
Android	47.9	66.7	18.8
BlackBerry	5.2	2.0	-3.2
Symbian	15.8	3.5	-12.3
Windows	6.7	10.5	3.8
Other	2.1	0.7	-1.4

Spain	3 m/e Apr 2012	3 m/e Apr 2013	% pt. Change
iOS	3.9	4.1	0.2
Android	81.1	92.8	11.7
BlackBerry	6.3	0.1	-6.2
Symbian	6.1	1.5	-4.6
Windows	2.5	1.5	-1.0
Other	0.2	0.0	-0.2

USA	3 m/e Apr 2012	3 m/e Apr 2013	% pt. Change
iOS	39.1	41.4	2.3
Android	50.3	51.7	1.4
BlackBerry	5.3	0.7	-4.6
Symbian	0.6	0.3	-0.3
Windows	3.8	5.6	1.8
Other	1.0	0.3	-0.7

China	3 m/e Apr 2012	3 m/e Apr 2013	% pt. Change
iOS		25.1	
Android		69.4	
BlackBerry		0.3	
Symbian		2.2	
Windows		2.4	
Other		0.7	

Australia	3 m/e Apr 2012	3 m/e Apr 2013	% pt. Change
iOS	30.6	28.1	-2.5
Android	57.5	63.6	6.1
BlackBerry	0.7	0.3	-0.4
Symbian	5.0	1.5	-3.5
Windows	4.1	5.1	1.0
Other	2.1	1.5	-0.6

Japan	3 m/e Apr 2012	3 m/e Apr 2013	% pt. Change
iOS		50.2	
Android		44.0	
BlackBerry		0.8	
Symbian		0.4	
Windows		0.6	
Other		3.9	

EU5	3 m/e Apr 2012	3 m/e Apr 2013	% pt. Change
iOS	20.2	18.4	-1.8
Android	60.1	69.6	9.5
BlackBerry	7.1	2.5	-4.6
Symbian	6.0	1.6	-4.3
Windows	3.7	6.7	3.0
Other	2.9	1.1	-1.7

(Fig. 1.5) Variazioni del mercato dei sistemi operativi [Kanta].

Per quanto riguarda la semplicità di diffusione è ancora Android la soluzione migliore dato che le applicazioni possono essere installate anche al di fuori dello store di Google, basterebbe, per esempio, inviare via mail l'applicazione da installare, ma esistono anche store paralleli a quello ufficiale.

Nel caso si voglia acquistare comunque la licenza di sviluppatore per poter pubblicare le proprie applicazioni sullo store ufficiale, la licenza fornita da Google per Android costa 25 $, quella fornita da Apple per iOS costa 99 $, quella fornita da Microsoft per Windows Phone costa 99 $ e quella fornita dalla BlackBerry per BB costa 20 $.

In questo caso è BB ad offrire la migliore tariffa ma i costi sono comunque contenuti anche per gli altri sistemi operativi.
Un altro vantaggio che ha la piattaforma Android è quello di essere Open Source e quindi è possibile visualizzare la maggior parte del codice sorgente e di intervenire su di esso dando molta più flessibilità agli sviluppatori.

Capitolo 2
Applicazione

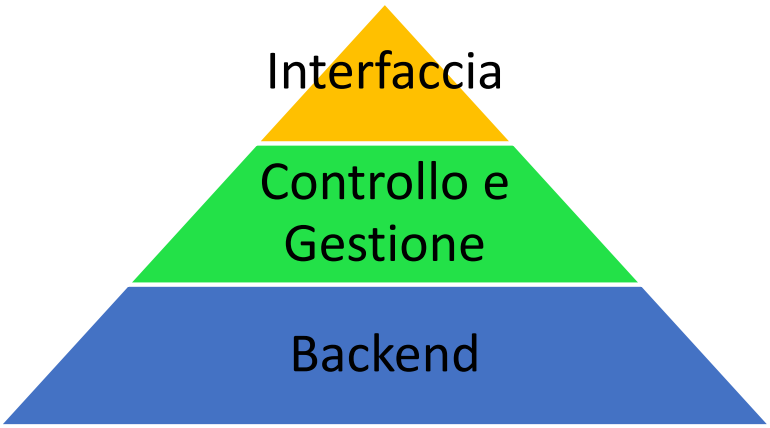
Dato lo sviluppo nel settore delle applicazioni mobili descritto nelle sezioni precedenti si vuole mostrare come ciò offra la possibilità ad attività in diversi settori, che poco hanno a che fare con la telefonia mobile, di aprirsi a tale mercato per trarne benefici, sia a livello pubblicitario, dato l'enorme quantità di utenti coinvolti, sia come supporto pratico nella gestione del lavoro delle attività stesse.
Come esempio, si vuole quindi realizzare un'applicazione che sia di supporto a un'attività di ristorazione, sia nella pubblicizzazione del locale, sia nella gestione delle ordinazioni e delle prenotazioni.

2.1 Requisiti

L'applicazione in questione, per adempire alle esigenze dell'attività dovrà necessariamente soddisfare alcuni requisiti in fatto di portabilità, interfaccia e backend.
La prima tra tutte le esigenze a dover essere analizzata, poiché incide sulla scelta dell'ambiente di sviluppo, è la portabilità dell'applicazione che deve poter raggiungere il maggior numero di utenti possibili nel modo più semplice ed economico possibile.
Nella sezione precedente sono stati presentati i sistemi operativi mobili più utilizzati in questo momento, e in fatto di portabilità e semplicità di distribuzione Android non ha rivali, sia per il numero di terminali disponibili sia perché le applicazioni possono essere distribuite anche su store paralleli o privatamente. Inoltre Android possiede anche il vantaggio di avere licenza Open Source e di utilizzare come ambiente di sviluppo Eclipse, anch'esso Open Source, così da poter utilizzare il codice sorgente in fase di programmazione e impostare al meglio l'ambiente di sviluppo.

Scelto l'ambiente di sviluppo, è necessario impostare l'applicazione in modo che risulti semplice nell'utilizzo tramite l'interfaccia con l'utente, appetibile dal punto di vista grafico e utile nella sue funzioni.
L'interfaccia dovrà essere semplice ed intuitiva in modo che l'utente possa navigare all'interno dell'applicazione sapendo sempre cosa stia facendo e senza essere sovraccaricato da funzioni complesse che possono far perdere l'interesse per l'applicazione. Si vuole separare la parte pubblicitaria a quella di gestione delle prenotazioni e ordinazioni così che l'utente possa scegliere da se il motivo di utilizzo dell'applicazione. Tali principi saranno portati avanti nella progettazione dell'interfaccia utente dell'intera applicazione trattata nella sezione seguente.
Infine, siccome l'applicazione che si vuole realizzare si vuole proporre anche come strumento per semplificare l'accesso alle prenotazioni e ordinazioni da parte dei clienti, per dare più sicurezza in tale ambito e non permettere a chiunque di poter fare un'ordinazione o una prenotazione, vi è la necessità di inserire un sistema di accesso per accedere a tale parte dell'applicazione. Inoltre, una volta effettuato l'accesso con le proprie credenziali, per prenotare posti a sedere o ordinare dei piatti, l'applicazione dovrà avere un sistema di gestione remoto degli ordini.
Per tali motivi tale applicazione necessita di un sistema di backend. Tale backend non richiede grandi necessità di spazio o capacità di calcolo, comunque la scelta verrà trattata in una prossima sezione (Capitolo 4).

2.2 Architettura



(Fig. 2.1) Architettura applicazione.

L’applicazione viene perciò strutturata su tre livelli (Vedi fig. 2.1):

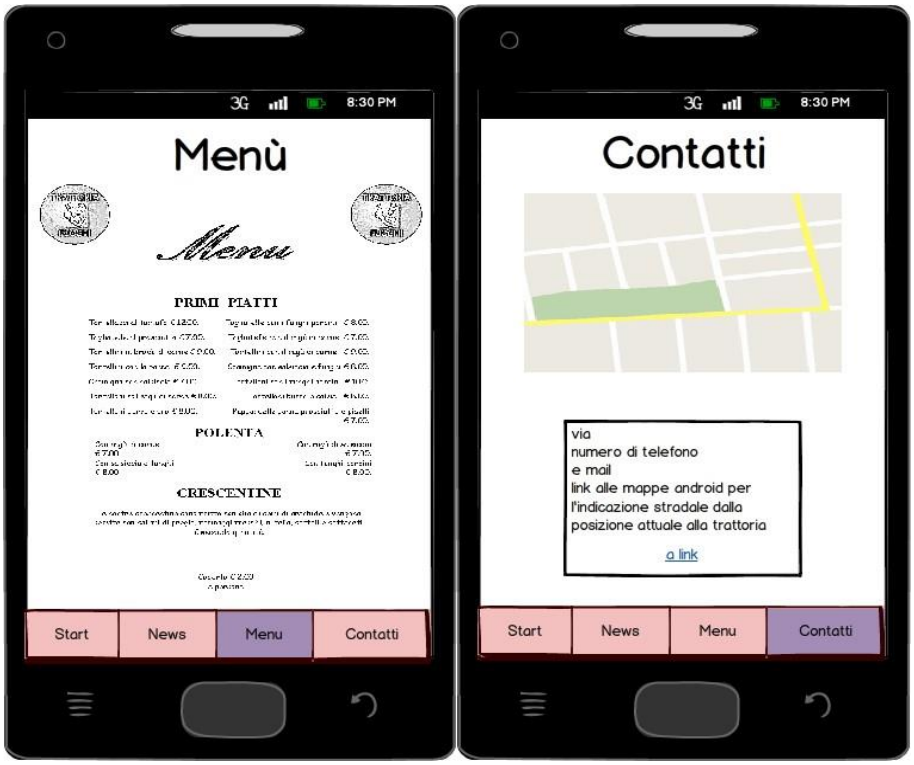
- Interfaccia.
- Controllo e gestione.
- Backend.

Tali livelli implementano i concetti di semplicità ed intuitività sopra descritti.
Al livello più alto vi è l’interfaccia grafica, essa è l’unica parte che sarà visibile all’utente finale e deve quindi essere quella che meglio implementa tali principi, si è quindi deciso di dividere la parte relativa alla presentazione del locale da quella della gestione delle ordinazioni accessibili entrambi da una schermata iniziale (Vedi fig. 2.2).

(Fig. 2.2a) Mockup pagina iniziale. (Fig. 2.2b) Mockup pagina News.

La parte di presentazione, navigabile attraverso un menu, è composta da tre pagine, una che mostra le novità in evidenza caricate dal backend (vedi fig. 2.2b), una che mostra il menu del ristorante e una relativa ai contatti con una mappa centrata su di esso (Vedi figg. 2.3a e 2.3b). Questa parte dell’applicazione è per lo più statica, e solo le novità saranno caricate all’interno del backend.
La parte delle ordinazione è invece più dinamica, vi si può accedere tramite un login ed è anch’essa navigabile attraverso un menu. Questa è composta da una pagina

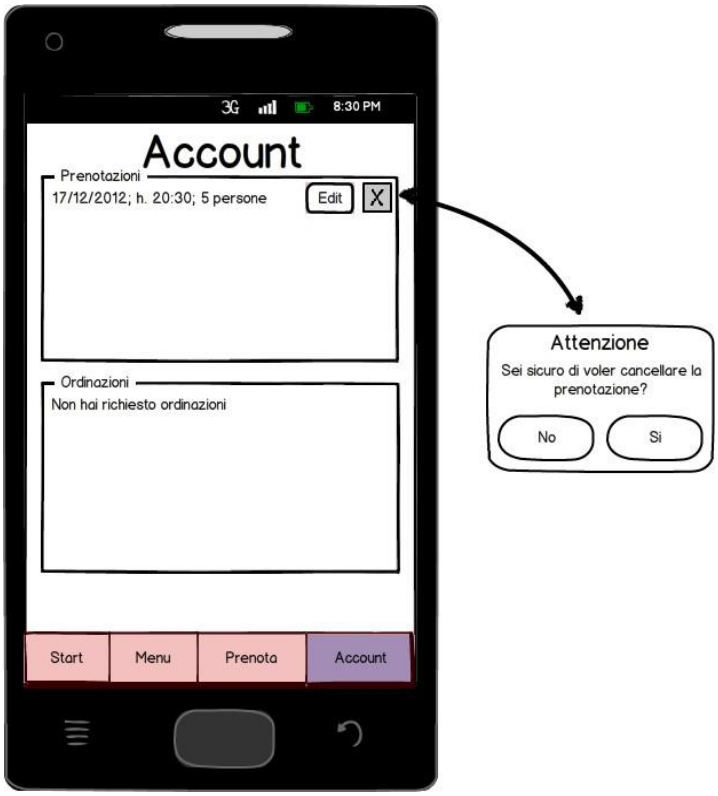
relativa al proprio account da dove è possibile visualizzare le proprie ordinazioni e prenotazioni (vedi figg. 2.4 e 2.5), da una dove è possibile ordinare e da una dove è possibile prenotare.



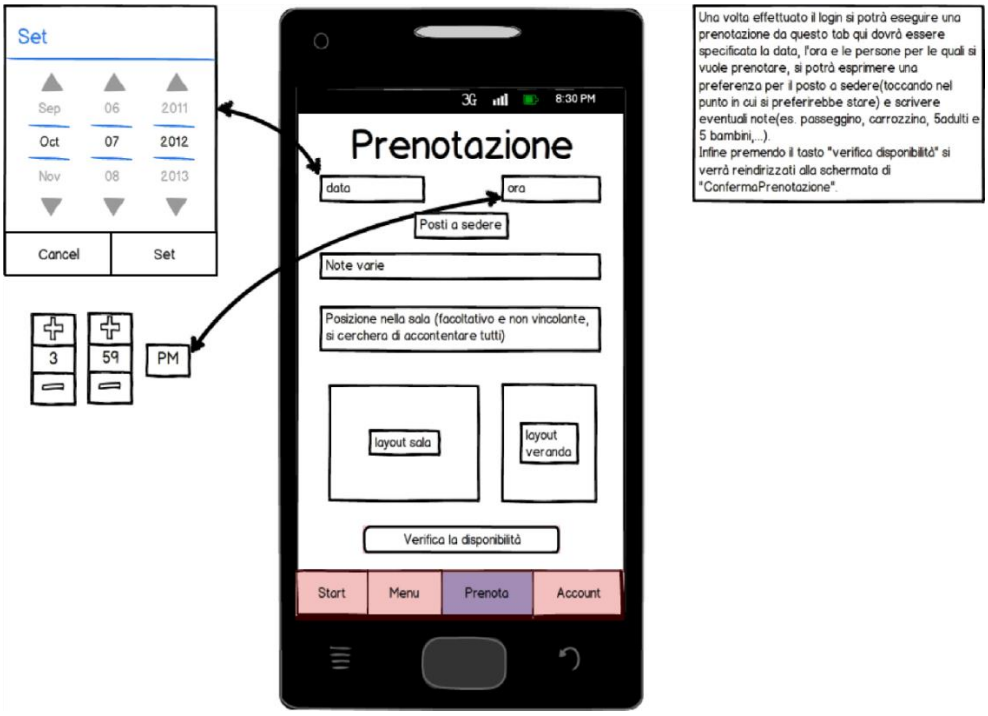
(Fig. 2.3a) Mockup pagina Menù. (Fig. 2.3a) Mockup pagina Contatti.

(Fig. 2.4) Mockup pagina Login.

Per eseguire una prenotazione è sufficiente specificare i campi richiesti (data, ora, posizione preferita, ecc.) e, dopo una verifica sul backend, è possibile passare ad una pagina di riepilogo della prenotazione per poi completarla (vedi figg. 2.6 e 2.7).

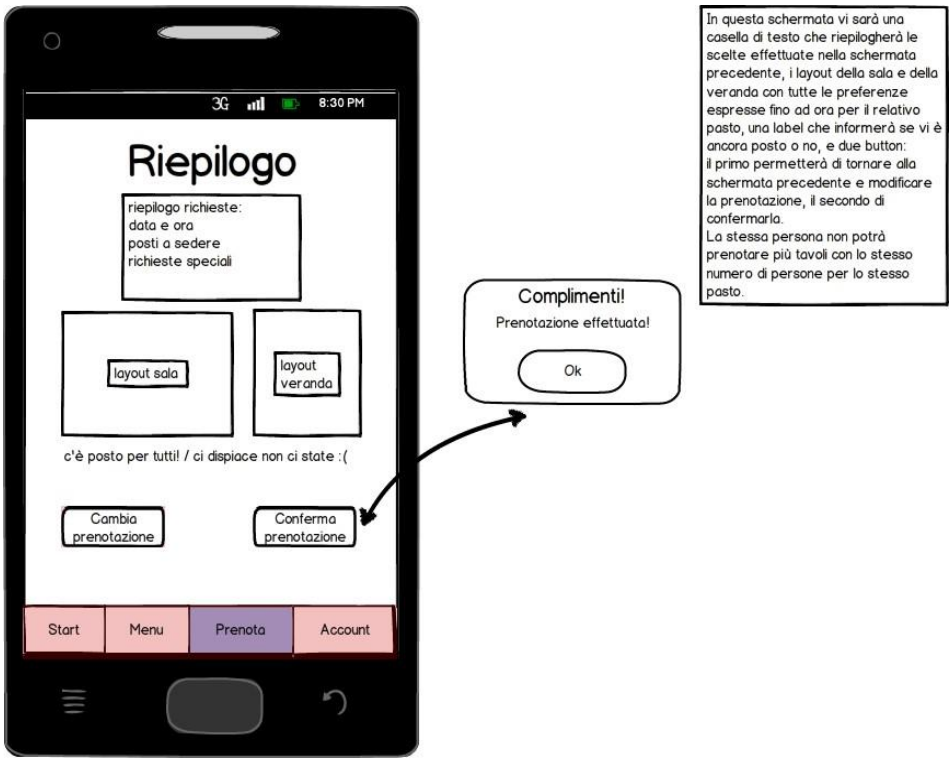


(Fig. 2.5) Mockup pagina Account.



(Fig. 2.6) Mockup pagina Prenota.

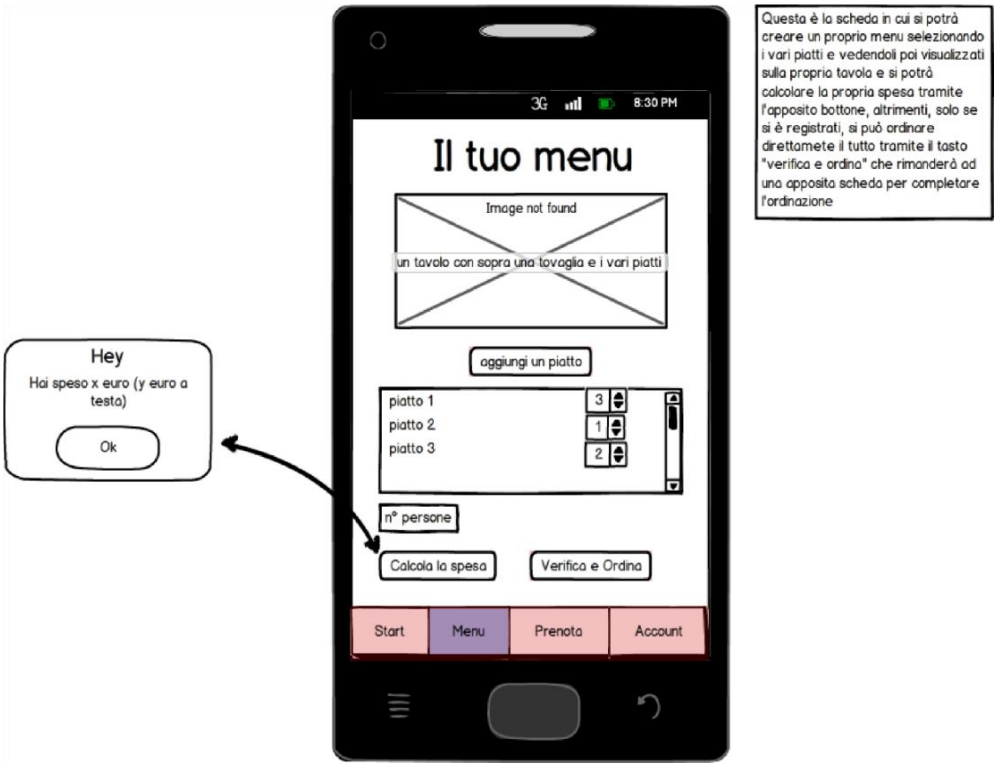
Un’ordinazione è invece composta da una serie di piatti, selezionabili attraverso una relativa pagina (Vedi fig. 2.8), e dalla loro quantità.



(Fig. 2.7) Mockup pagina Riepilogo Prenotazione.

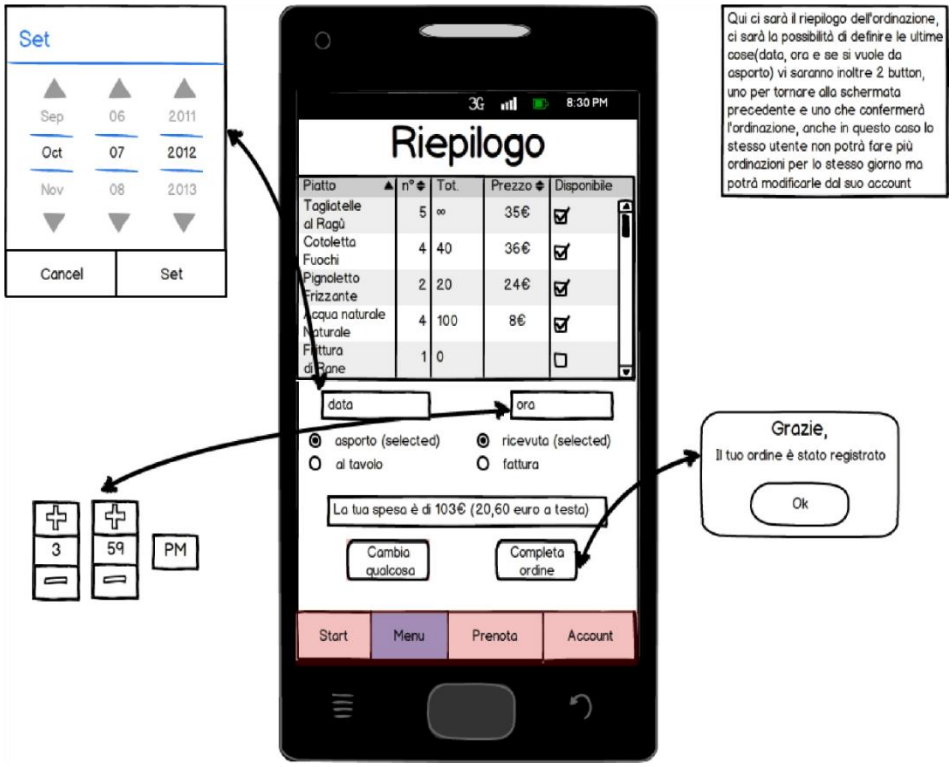
(Fig. 2.8) Mockup pagina Piatti.

Per rendere l'interfaccia più coinvolgente tali piatti vengono visualizzati nella propria "tavola degli ordini" e riepilogati in una tabella sottostate (Vedi fig.2.9). Una volta selezionati piatti e quantità si potrà accedere ad una schermata di riepilogo che, dopo l'inserimento della data, dell'ora, del tipo di ordine che si desidera eseguire e del metodo di fatturazione, offre l'opportunità di completare l'ordinazione (Vedi fig. 2.10).



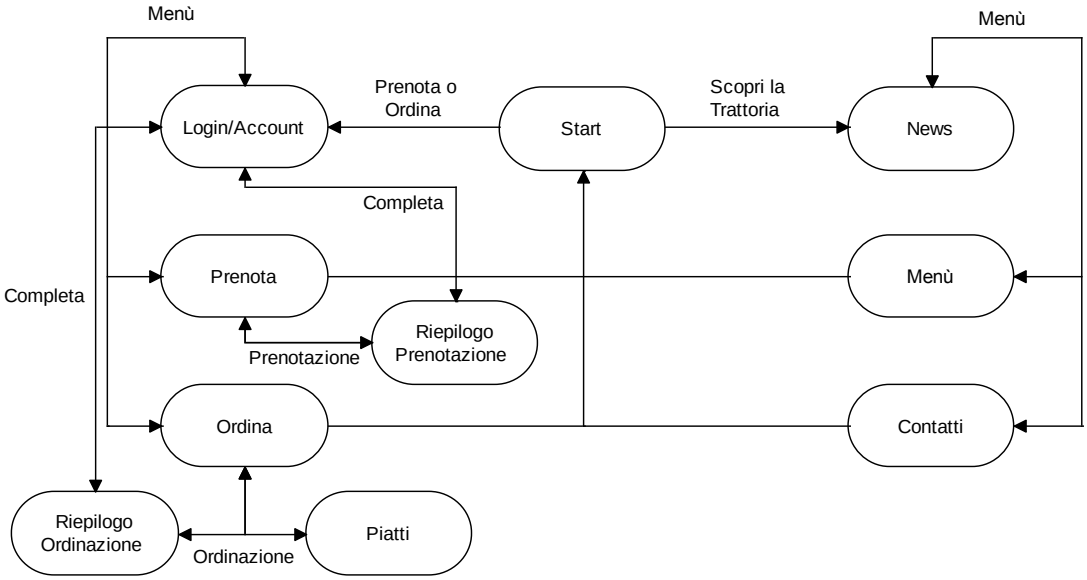
(Fig. 2.9) Mockup pagina Ordina.

Il collegamento tra tutte le pagine viene mostrato schematicamente in figura 2.11
Il livello intermedio funge da mediatore tra gli altri due livelli, questo venendo utilizzato nella creazione di layout dinamici, dove le informazioni vengono prese dal backend, come nella pagine della news, in fase di ordinazione e prenotazione e nella gestione dell'account. In fase di ordinazione e prenotazione gestisce inoltre la trasmissione dei dati tra schede diverse e il salvataggio dei dati sul backend. Per le ordinazioni, in particolare, dove le quantità di dati da passare sono diverse e di diverso tipo a questo livello viene creato un oggetto Piatto per semplificare il passaggio dei dati.



(Fig. 2.10) Mockup pagina Riepilogo Ordinazione.

Infine al livello più basso vi è il backend, questo gestisce il login e mantiene salvate le news, le prenotazioni e le ordinazioni ricevute, in modo da poter essere gestite e visualizzate dai livelli superiori.



(Fig. 2.11) Diagramma di flusso delle pagine dell'applicazione.

Capitolo 3
La piattaforma Android

Come detto nel capitolo precedente, Android risulta essere l'ambiente di sviluppo più adatto alle esigenze dell'applicazione che si vuole realizzare, ma, prima di cimentarsi nello sviluppo vero e proprio di tale applicazione, è necessario studiare più a fondo la piattaforma Android, e l'ambiente di sviluppo ad essa associato, per conoscerne le meccaniche e gli strumenti a disposizione per la programmazione.

3.1 Android SDK e integrazione con Eclipse

Android distribuisce gratuitamente agli sviluppatori software il proprio SDK che può essere integrato nell'ambiente di sviluppo Eclipse, anch'esso di licenza Open Source, per la programmazione per tale piattaforma.
Attraverso questo kit di sviluppo, qualsiasi utente potrà procurarsi il codice sorgente di ogni versione Android per poterlo utilizzare all'interno dei propri progetti e modificarlo o migliorarlo autonomamente. All'interno di esso è possibile trovare, oltre che le sorgenti di tutte le versioni Android precedenti, le librerie fornite da Google, Android e Intel per l'integrazione dei loro servizi all'interno della piattaforma.
Distribuita assieme all'SDK vi è anche l'Android Virtual Device Manager (AVD Manager), che dà la possibilità di emulare macchine virtuali Android nel proprio ambiente di sviluppo, di istallarvi applicazioni, proprie e non, in modo da verificare il corretto funzionamento del proprio codice.
Integrati con Eclipse l'Android SDK e l'Android AVD, esso darà la possibilità di creare progetti strutturati per applicazioni Android, che saranno quindi composte da parti dinamiche scritte in Java, che si occupano della gestione degli eventi e parti statiche scritte in XML che definiscono le caratteristiche che, generalmente, non

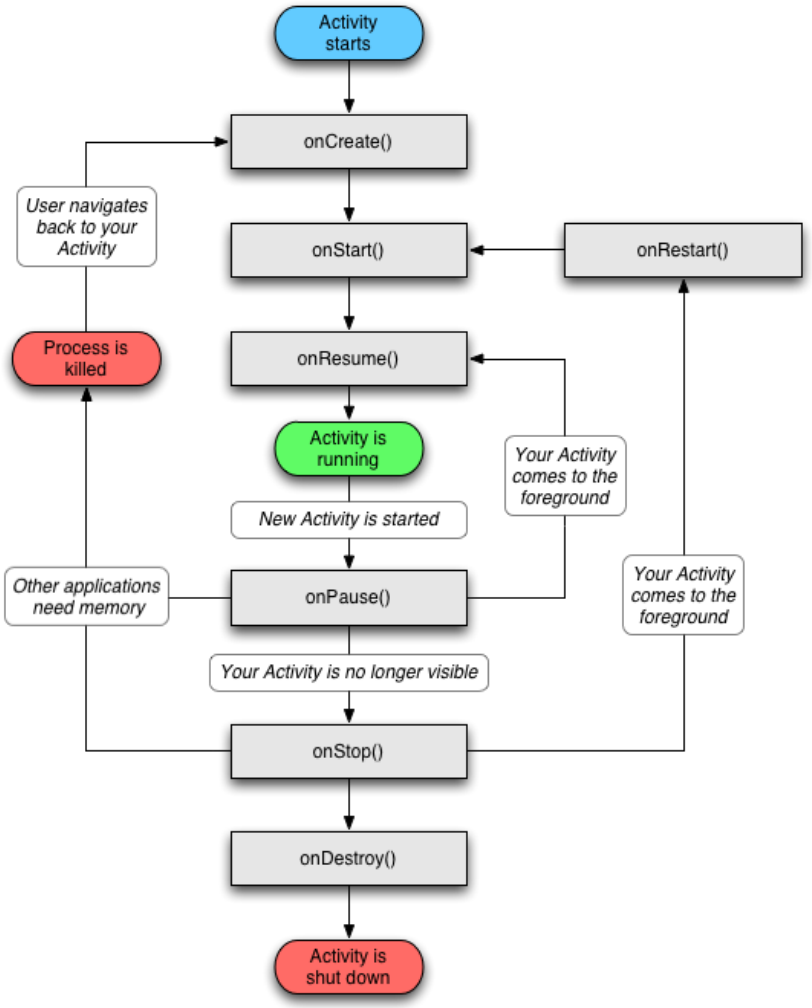
cambiano durante l'esecuzione dell'applicazione, come la disposizione del testo o il colore di sfondo.
L'utilizzo di Java comporta inoltre la necessità di installare l'SDK di Java nel proprio computer prima di poter usufruire di tali servizi.

3.2 Componenti principali

Prima di andare a illustrare i vari oggetti utili ed utilizzabili all'interno dell'applicazione è bene comprendere cosa sono e come vengono gestite le Activity e gli Intent di Android.
La classe Activity è uno degli elementi centrali di ogni applicazione Android, un concetto legato allo sviluppo delle interfacce grafiche. Normalmente una Activity rappresenta una singola schermata di un'applicazione e le applicazioni possono definire una o più Activity per trattare diverse fasi del software.
Dato che l'Activity una delle componenti principali nel ciclo di vita di ogni applicazione Android, il modo in cui esse vengono lanciate e in cui interagiscono tra loro è una parte fondamentale nello sviluppo con la piattaforma. Dal momento che una sola Activity può essere in "primo piano", o foreground, nello stesso tempo, le altre Activity che in un determinato momento non si trovano nello stato attivo e quindi si trovano in background potrebbero essere terminate dal sistema operativo ad esempio perché la memoria diventa insufficiente. Questo significa che ogni applicazione Android può cambiare lo stato in ogni momento e deve essere pronta ad essere interrotta e chiusa in ogni istante.
Il ciclo di vita delle Activity (Vedi fig. 3.2) è composto da:

- onCreate(Bundle): questo metodo è invocato dalla DVM quando l'Activity viene creata per la prima volta ed è necessario alla sua inizializzazione. Il metodo svolge le operazioni di definizione dell'interfaccia grafica attraverso un file .xml, l'associazione di ogni elemento dell'interfaccia alla azione desiderata corrispondente ed inizializza le strutture dati utilizzate come liste e data base. Il parametro passato (tipo Bundle) è essenziale al passaggio d'informazioni riguardanti lo stato dell'applicazione.

- onPause(): quando un l'utente abbandona un'Activity la DVM invoca questo metodo. Il programmatore deve specificare le operazioni che si riferiscono al salvataggio dello stato dell'applicazione.
- onStart(): questo metodo è invocato quando l'Activity è visualizzata.



(Fig. 3.2) Ciclo di vita di un'Activity [AndDe].

- onRestart(): il metodo è usato quando l'Activity diventa visibile all'utente dopo che è era stata fermata nello stato di paused. Alla fine dell'esecuzione del metodo è invocato onStart().
- onResume(): il metodo è invocato subito prima che l'Activity sia posta sulla cima dello stack e che quindi l'utente vi possa interagire.
- onStop(): questo metodo è invocato se un'Activity oscura un'altra.
- onDestroy(): il metodo è lanciato prima della distruzione finale dell'Activity.

L'altro elemento fondamentale è l'Intent, un meccanismo che descrive un'azione specifica.
Schematizzando, un Intent può essere utilizzato per:

- Trasmettere l'informazione per cui un particolare evento (o azione) si è verificato;
- Esplicitare un'intenzione che una Activity o un Service possono sfruttare per eseguire un determinato compito o azione, solitamente con o su un particolare insieme di dati;
- Lanciare una particolare Activity o Service;
- Supportare l'interazione tra qualsiasi applicazione installata sul dispositivo Android, senza doversi preoccupare di che tipo di applicazione sia o di quale componente software gli Intent facciano parte.

3.2.1 Oggetti di Base

Dato che le applicazioni per Android vengono scritte in Java e XML, ne vengono ereditati gli di base, da Java, come linguaggio ad oggetti, il polimorfismo e l'ereditarietà delle classi e degli oggetti, e dall'XML, come linguaggio di markup, la gestione dell'interfaccia con l'utente. Tramite l'utilizzo di Java è possibile accedere a tutti gli oggetti e le classi per la gestione delle variabili e degli oggetti stessi nativi di tale linguaggio.
Oggetti e classi più utilizzati sono [OJava]:

- Primitive Data Types, sono byte, short, int, long, float, double, boolean e char, in un linguaggio come Java le variabili devono essere pima dichiarate per poter essere utilizzate, e il tipo della variabile definisce lo spazio che andrà ad occupare.

  Nello specifico, possono assumere valore di interi con segno: i byte, che occupano 8 bit, gli short che ne occupano 16, gli int da 32 bit e i long da 64. A float e double vengono associati valori decimali rispettivamente a 32 e 64 bit. Le variabili di tipo boolean possono assumere solo i valori "true" o "false" e ad un char viene associate un singolo carattere Unicode a 16 bit.

- String, La classe String rappresenta stringhe di caratteri. Tutte le stringhe di lettere nella programmazione Java, come “abc”, vengono implementate come istanze di questa classe.
- Array, un contenitore di altri oggetti, ne contiene un valore fissato e di un solo tipo.
- ArrayList, si tratta di un array ridimensionabile, un’implementazione dell’interfaccia List, può contenere qualsiasi oggetto, compreso null, e fornisce metodi per gestire la dimensione dell’array.

3.2.2 Oggetti grafici

Android mette a disposizione degli sviluppatori una serie di oggetti grafici, tutti come estensioni della classe android.widget che deriva dalla classe Java java.lang.Object.
Gli oggetti più utilizzati, sia staticamente che dinamicamente sono [AndDe]:

- LinearLayout, un layout che posiziona gli oggetti inseritivi in una singola colonna o in una singola riga. Utilizzando il metodo setOrientation() è possibile selezionare una delle due opzioni. È inoltre possibile specificare la gravity, ovvero l'allineamento di tutti gli oggetti utilizzando setGravity() o specificando nei singoli oggetti quale spazio devono occupare del layout. L'orientamento predefinito è orizzontale.
- RelativeLayout, un layout in cui le posizioni degli oggetti possono essere descritte in relazione tra loro o al layout stesso.
  Non si può avere una dipendenza circolare tra la dimensione della RelativeLayout e la posizione degli oggetti al suo interno. Ad esempio, non si può avere un RelativeLayout la cui altezza è impostata su WRAP_CONTENT e un oggetto viene impostato su ALIGN_PARENT_BOTTOM.
- TaleLayout, un layout che organizza gli oggetti al suo interno in righe e colonne, estende la classe LinearLayout strutturandola per costruire una tabella. Un TableLayout è, in genere, costituito di un certo numero di oggetti TableRow, ciascuno che definisce una riga. I TableLayout non possono visualizzare le linee di confine per le loro righe, colonne o celle. Ogni riga ha

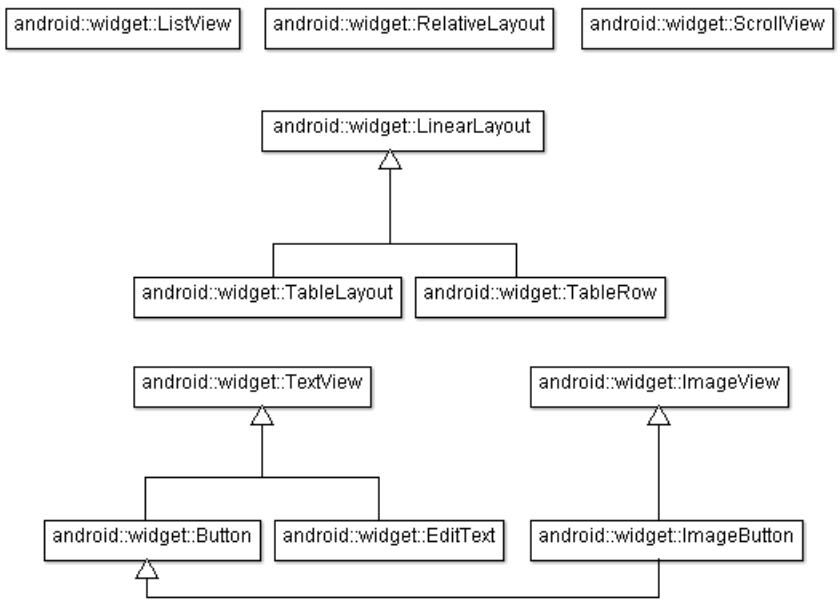
un numero maggiore o uguale a zero di celle ed ogni cella può contenere un oggetto View.

La larghezza di una colonna è definita dalla riga con la più ampia cella della colonna. Tuttavia, un TableLayout può specificare il layout di alcune colonne chiamando setColumnShrinkable () o setColumnStretchable (). Infine, è possibile nascondere una colonna chiamando setColumnCollapsed ().

Gli oggetti in un TableLayout non possono specificare l'attributo layout_width. La larghezza è sempre MATCH_PARENT. Le celle possono essere aggiunte ad una riga in ordine crescente, sia nel codice che nell'XML. I TableRow possono essere sostituiti da una qualsiasi sottoclasse di android.View, tale View verrà visualizzata come una singola riga che si estende su tutte le colonne della tabella.

- TableRow, un layout che organizza gli oggetti all'interno di essi in senso orizzontale, estende la classe LinearLayout per realizzare una riga da inserire in una tabella. Un TableRow dovrebbe sempre essere usato all'interno di un TableLayout, in caso contrario il TableRow si comporta come un LinearLayout orizzontale.
- ScrollView, è un contenitore di layout che permette al Layout interno di venire visualizzato dall'utente attraverso uno scorrimento, permettendo così di utilizzare più spazio rispetto a quello che fisicamente impone la dimensione dello schermo. Uno ScrollView è un FrameLayout, significa che è necessario posizionare un solo Layout al suo interno che contiene l'intero contenuto da scorrere; tale Layout può essere a sua volta un gestore di altri layout ed avere una complessa gerarchia di oggetti. ScrollView supporta solo lo scorrimento verticale. Per lo scorrimento orizzontale, è necessario utilizzare la classe HorizontalScrollView.
- ListView, che consente di visualizzare gli elementi in un elenco a scorrimento verticale. Gli oggetti vengono caricati dalla List Adapter associata a questa View.
- TextView, è un visualizzatore di testo per l'utente e, opzionalmente, permette di essere modificato dall'utente. Un TextView è un editor di testo, e la classe di base è configurata per non consentire la modifica.

- EiditText, si tratta di un'estensione della classe TextView e viene utilizzato come un contenitore di TextView che ne abilita la modifica.
- ImageView, un contenitore di immagini.
- Button, rappresenta, estendendo la classe TextView, un pulsante che può essere premuto o selezionato dall'utente per eseguire delle azioni.
- ImageButton, un pulsante con un'immagine che può essere premuto o selezionato dall'utente. Come impostazione predefinita, un ImageButton appare come un tasto normale con lo sfondo del pulsante standard che cambia colore durante diversi stati. Sulla superficie del tasto vi è posizionata un'immagine, definita mediante l'attributo android:src nell'XML o attraverso il metodo setImageResource (int). Tale oggetto eredita i metodi e lo stile grafico sia dalla classe Button che dalla classe ImageView.
- Menu, un'interfaccia per la gestione delle elementi di un menu. Di default, ogni attività supporta un menu. È possibile aggiungere elementi a questo menu e gestire le operazioni su tali elementi agendo prima sull'XML tramite il metodo MenuInflater, poi nel codice specificando l'azione che si vuole monitorare.



(Fig. 3.3) Diagramma dei principali oggetti utilizzati.

3.2.3 Mappe

La gestione delle mappe in Android merita una sezione a parte, poiché queste sono gestite da Google e non direttamente da Android come gli altri oggetti in presi in esame. Con le Google Maps Api V2 si ha la possibilità di integrare le mappe tramite un motore "vector-based", dunque potenziato dalla stessa tecnologia che sta alla base dell'applicazione ufficiale. Questo permetterà un caricamento immediato delle immagini satellitari, 2D o 3D, mettendo a disposizione degli utenti l'esecuzione di comandi come la rotazione o l'inclinazione del punto di vista senza rallentamenti. Il tutto è integrato con Google Play Services, l'insieme di strumenti attraverso i quali Google facilita la pubblicazione e distribuzione degli aggiornamenti sullo store Play.
Le mappe distribuite prima dell'implementazione del motore "vector-based" erano costituite da un reticolo di "tasselli" che venivano trasmessi per mostrare la mappa. Questo approccio fornisce la possibilità di trasferire molte informazioni, tale metodo è anche molto performante dato che la quantità di informazione visualizzabile dipende alla quantità di pixel del proprio dispositivo.
Le mappe "vector-based", al contrario, trasmettono i dati per costruire la mappa direttamente al dispositivo in modo che la mappa venga assemblata dal lato client. Questo approccio necessita che il dispositivo soddisfi i requisiti per la creazione della mappa ma fornisce molti altri vantaggi, come:

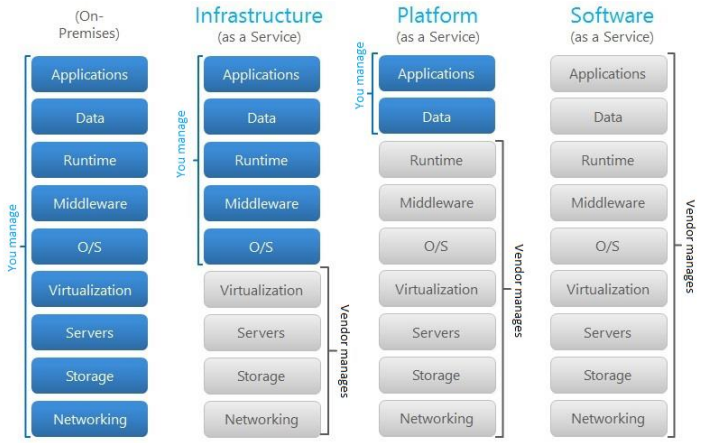
- Riprogettazione e restyling in costruzione: Assemblando una mappa basata su vettori, vengono solo forniti i dati della mappa di base e viene quindi permesso all'applicazione locale di personalizzare la costruzione della mappa.
- Costruzione della mappa graduale: è possibile disegnare la mappa, mentre l'utente osserva, invece di caricarla piastrella per piastrella.
- Modifiche di prospettiva: è possibile inclinare mappe, creare mappe 2.5D o 3D in base ai dati ricevuti.
- Transizioni fluide: Invece di passare tra livelli di zoom predefiniti, è possibile avere transizioni più uniformi durante la navigazione una mappa.

L'implementazione delle mappe di Google in un'applicazione Android deve avvenire importando le librerie di Google Play Services, richiedendo una chiave da

sviluppatore inedita per la propria applicazione a Google e operando alcuni passaggi sul Manifest in modo da permettere all'applicazione di accedere al GPS e alla chiave di Google.
La classe che andrà ad implementare le mappe nell'applicazione è la MapFragment, che permette di inserire una mappa all'interno di un Fragment di Android, della quale è sottoclasse. L'integrazione di tali mappe attraverso la classe MapFragment viene supportata solamente dalle versioni di Google API 12 o superiori, per versioni precedenti è comunque disponibile la classe SupportMapFragment che esegue le stesse operazioni della precedente.

Capitolo 4
Backend

Il backend di un'applicazione può essere realizzato in diverse tecnologie che possono essere più o meno adatte in base alle esigenze dell'applicazione stessa per questo è importante scegliere quella che più si adatta alla portabilità, alle prestazioni e alle spese richieste dall'applicazione stessa. Tale backend, per essere accessibile, verrà appoggiato ad un cloud, una struttura hardware formata da più nodi di calcolo e di storage che lavorano in maniera sincronizzata per offrire dei servizi. I servizi che possono essere offerti attraverso una struttura cloud possono essere di diversi tipi, come Infrastructure-as-a-service, Platform-as-a-service e Software-as-a-service. Normalmente tali servizi vengono tutti offerti da chi possiede la struttura cloud e, assieme ad essi, vengono offerte strutture differenti su cui appoggiarsi, più la struttura è completa minore sarà la flessibilità del sistema (Vedi Fig. 4.1).



(Fig. 4.1) Servizi cloud.

4.1 Infrastructure-as-a-Service

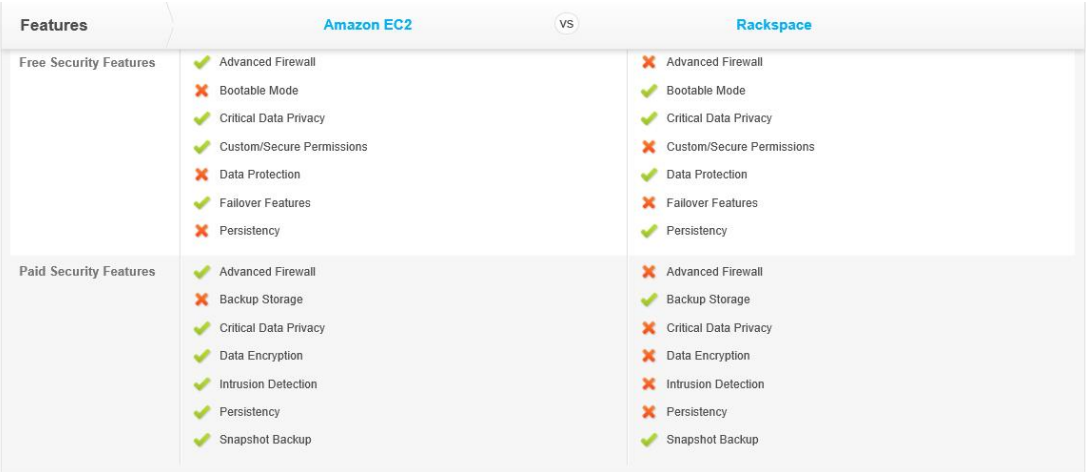
L'Infrastructure-as-a-Service (IaaS) ed è il servizio più flessibile di tutti. Acquistando un servizio IaaS, si ha a disposizione una virtualizzazione dell'hardware

di un computer tradizionale: CPU, RAM, Storage e schede di rete con connettività. Attraverso questo sistema si hanno a disposizione tutte le potenzialità e la flessibilità di un computer fisico [Zai11], senza dover pensare all'hardware o alla continuità di servizio in caso di guasto hardware, in quanto è compito del fornitore di servizi occuparsi di questi aspetti di livello basso. D'altro canto ci si dovrà occupare di tutti gli aspetti legati al bilanciamento di carico su più istanze, a strutturare una base dati adatta, a scegliere il software ecc. Per quanto riguarda i costi di un simile servizio, i servizi IaaS, in genere, permettono la fatturazione "a consumo" in quanto una istanza IaaS è un costo per il fornitore solo quando lavora e non quando è spenta.

4.1.1 Caso di studio: AWS/Rackspace

In questo capitolo vengono presi in esame Amazon web services (AWS) e Rackspace, due IaaS, confrontandoli sotto diversi aspetti.
La maggiore differenza tra i due provider consiste nello modo in cui viene offerto il servizio. AWS è basato sulla pura virtualizzazione, ovvero Amazon è proprietaria di tutto l'hardware e controlla l'intera struttura della rete. L'utente gestisce tutto dal sistema operativo installato e le istanze virtuali richieste on-demand per le operazioni vengono poi rilasciate una volta completato il loro scopo.
Rackspace è invece un venditore di Hybrid computing offrendo quindi virtualizzazioni dove supportate da hardware dedicato questa politica si basa sull'idea che gli utenti vogliano dei classici data-center, semplicemente li vogliono nel cloud [Ree09].
I piani tariffari dei due provider presi in oggetto sono molto vari e permettono di selezionare il tipo di macchina virtuale che si sta acquistando selezionandone vari parametri come il sistema operativo o le specifiche tecniche della macchina.
Il piano tariffario di Amazon è più completo e dà maggiori possibilità di personalizzazione in base alle esigenze, dando anche la possibilità di scegliere la regione dei server da utilizzare [AWSer], inoltre, a parità di macchina richiesta, i costi di Amazon sono inferiori a quelli di Rackspace che, però è più semplice da impostare e fornisce un servizio di supporto alla scelta del piano che più si adatta alle esigenze del consumatore [Racks].

Entrambi i provider forniscono diversi servizi, gratuiti o a pagamento, come supporto agli sviluppatori (vedi fig. 4.2) ed anche in questo caso Amazon fornisce maggiore possibilità di scelta.



Features	Amazon EC2	vs	Rackspace
Free Security Features	✓ Advanced Firewall ✗ Bootable Mode ✓ Critical Data Privacy ✓ Custom/Secure Permissions ✗ Data Protection ✓ Failover Features ✗ Persistency		✗ Advanced Firewall ✓ Bootable Mode ✓ Critical Data Privacy ✗ Custom/Secure Permissions ✓ Data Protection ✗ Failover Features ✓ Persistency
Paid Security Features	✓ Advanced Firewall ✗ Backup Storage ✓ Critical Data Privacy ✓ Data Encryption ✓ Intrusion Detection ✓ Persistency ✓ Snapshot Backup		✗ Advanced Firewall ✓ Backup Storage ✗ Critical Data Privacy ✗ Data Encryption ✗ Intrusion Detection ✗ Persistency ✓ Snapshot Backup

(Fig. 4.2) Servizi di Amazon [AWSer] e Rackspace [Racks] a confronto.

Amazon inoltre fornisce molto più spazio che Rackspace, anche dinamicamente tramite l'EBS (Elastic Block Service), rendendolo più adatto per grandi progetti o per start-up che necessitano di file già utilizzate da altre istanze e che possono essere utilizzate tramite l'EBS.

In conclusione Amazon è più versatile e fornisce più servizi di supporto ma fornisce una soluzione di pura virtualizzazione che può non adattarsi alle esigenze dell'utente nel caso si abbia la necessità di interazioni con hardware dedicato come nell'interfacciamento con determinate periferiche di I/O o l'integrazione di sistemi che possono non essere progettati per un'integrazione web.

4.2 Platform-as-a-Service

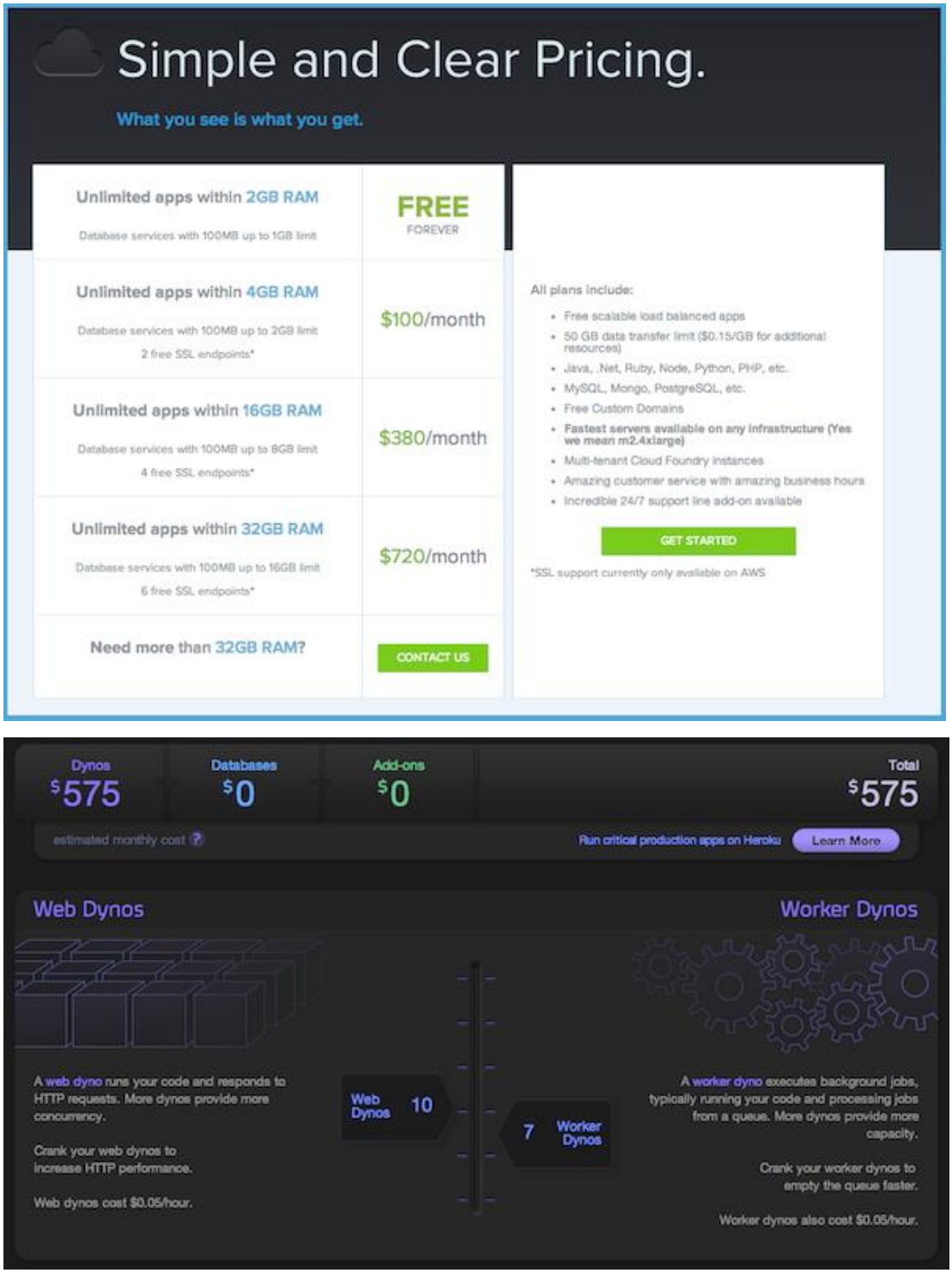
Per Platform-as-a-Service (PaaS) si intende la virtualizzazione di una piattaforma, ovvero di una struttura che comprende sia l'hardware che il software tale da fornire un ambiente di lavoro completo per gli sviluppatori. In questo caso l'infrastruttura attraverso la quale è realizzata una piattaforma è già gestita dal rivenditore. Da qui ci si deve solo occupare di creare l'applicazione per poter offrire un dato servizio [Zai11]. In questo caso, si hanno meno oneri ma anche meno flessibilità: difatti il fornitore di servizi ha sicuramente optato per delle scelte riguardo alla piattaforma su

cui offrire il servizio PaaS a cui adattarsi per lo sviluppo dell'applicazione. Questo tipo di servizio normalmente viene fatturato scegliendo un "taglio" (quanti MB di spazio su disco, la quantità di banda, ecc.).
Offerte PaaS possono comprendere:

- Distribuzione di applicazioni complete, incluse le piattaforme su misura.
- Gestione di hardware e software di base e capacità di hosting.
- Strumenti per la progettazione, sviluppo, test, implementazione e hosting relative ad una applicazione.

4.2.1 Caso di studio: Heroku/AppFog

In questo capitolo vengono presi in esame Heroku e AppFog, due PaaS che affrontano lo stesso problema in modo differente sotto diversi aspetti.
Le tariffe di AppFog dipendono dalla RAM. Si possono creare app e istanze finché non si supera il limite di RAM definite dal proprio piano tariffario. Il piano gratuito di AppFog fornisce 2GB di RAM, il successivo 4GB. La RAM viene condivisa da tutte le applicazioni del proprio account ed ogni istanza deve avere un minimo di 128MB di RAM [AppFo].
Heroku, d'altra parte, si fa pagare per il tempo di utilizzo della CPU. Le proprie applicazioni non condividono le ore di utilizzo tra loro, e Heroku fornisce 750 ore gratis per mese per applicazione, dando la possibilità di eseguire un'istanza senza spendere niente, (30 giorni per 24 ore = 720). Il costo per ogni ora in più è di $0.05 per ora e il tempo viene scalato ogni secondo [Herok].
Con AppFog si deve creare una "app" separata per ogni processo in background, rendendo le istanze disponibili per tutte le applicazioni dell'account.
Per quanto riguarda l'integrazione AppFog supporta MongoDB, MySQL, e PostgreSQL, anche se gli ultimi due solo in versione beta. Heroku, invece, fornisce PostgreSQL direttamente e tutti gli altri attraverso plug-in.
I due servizi offrono add-on simili, Heroku ne ha molti di più, più di cento, anche perché è sul mercato da più tempo di AppFog che ne conta poco più di una dozzina.



(Fig. 4.3) Tariffe di AppFog [AppFo] e Heroku [Herok].

Infine, per quanto riguarda l'assistenza si ha che per trovare un'informazione nello spazio dedicato da Heroku è necessario prima cercare tra le informazioni di base fornite e solo dopo piò essere chiesto il problema al loro centro assistenza. Tale

ricerca può essere evitata comprando l'assistenza tecnica 24h su 24h, 7 giorni su 7. Al contrario AppFog offre assistenza tramite una live chat.
In conclusione, confrontando i due provider si nota che AppFog fornisce un servizio migliore sulla gestione della RAM e sul sistema di assistenza, mentre Heroku è più flessibile grazie ai numerosi add-on e gestisce le applicazioni separatamente.

4.3 Backend-as-a-service

Backend-as-a-service (BaaS) è un'altra categoria di servizi cloud computing dedita alla fornitura di backend per applicazioni mobili e web. I BaaS inoltre forniscono gli utenti di altre funzioni per le loro applicazioni come strumenti per la gestione degli utenti, delle notifiche push e l'integrazione con i social network. Tali servizi vengono forniti attraverso l'utilizzo di relativi SDK e API. I BaaS sono servizi cloud relativamente recenti, tant'è che la maggior parte delle start-up sono posteriori al 2011, anche per questo il mercato relativo ad esse è in continuo sviluppo. I costi di questo servizio sono, in alcuni casi, legati a quante funzioni vengono rese disponibili per le applicazioni mentre, in altri, legati allo spazio reso disponibile nel cloud e al numero di accessi al servizio, in altri ancora ad entrambi, comunque a tariffazione periodica.

4.3.1 Caso di studio: Parse/StackMob

In questa sezione vengono presi in esame Parse e StackMob analizzando i servizi offerti da ciascuno dei due.
La prima differenza tra i due provider sta nei piani tariffari. Parse imposta le proprie tariffe in base al numero di richieste effettuate dalle applicazioni in un mese, al numero di notifiche inviate in un mese e alle operazioni al secondo richieste [Parse]. StackMob, al contrario, non impone limiti per le richieste dalle applicazioni o per le notifiche ma imposta i propri piani tariffari in base ai servizi offerti [StaMo] (Vedi fig. 4.6).
Sia Parse che StackMob forniscono la possibilità di usufruire di vari servizi, come l'integrazione con i social network, il login, la geolocalizzazione o la possibilità di

utilizzare del cloud code, in particolare questo servizio offerto da Parse risulta meno flessibile poiché il codice può essere scritto solo in Javascript e utilizzando esclusivamente l'apposito SDK mentre StackMob permette di utilizzare Java o Scala sui loro server.

	Basic	Pro	Enterprise
API Requests	1,000,000	15,000,000	Need more? Contact us about long term enterprise contracts! (877) 747-8568 enterprise@parse.com
Pushes	1,000,000	5,000,000	
File Storage	1GB	10GB	
App Collaborators	0	2	
App open metrics	✓	✓	✓
Burst Limit	20 per second	40 per second	400 per second
Compare metrics		✓	✓
Global Push opens metrics		✓	✓
Advanced Push Targeting		✓	✓
Push Scheduling		✓	✓
End-User Branding		✓	✓
Multiple Push Certificates		✓	✓
Enterprise-grade Infrastructure			✓
Account Management			✓
Phone Support			✓
Enteprise SLA			✓
Custom Deployment			✓
Individual Push opens metrics			✓

Features	Basic	Pro	Enterprise
Core API	✓	✓	✓
StackMob Marketplace Access	✓	✓	✓
Access Control	✓	✓	✓
Server Analytics	✓	✓	✓
Collaboration Free	✓	✓	✓
CORS Support	✓	✓	✓
Dev & Prod Environments	✓	✓	✓
Social Integration	✓	✓	✓
GeoQueries	✓	✓	✓
Push Notification Basic	✓	✓	✓
Push Notifications Premium	Available	✓	✓
Custom Code	Available	✓	✓
Collaboration Premium	Available	✓	✓
API Versions	Available	Available	✓
Custom Domains	Available	Available	✓
HTML5 Hosting	Available	Available	✓
Dedicated Database		✓	✓
Dedicated Custom Code		✓	✓
SLA		✓	✓
Extra Environments			✓
Dedicated StackMob Platform			✓
Custom Marketplace			✓

(Fig. 4.6) Piani tariffari di Parse [Parse] e StackMob [StaMo].

Entrambi forniscono SDK compatibili con Eclipse, sia quello fornito da Parse che quello fornito da StackMob sono di installazione immediata, è sufficiente importare le librerie necessarie all'interno del proprio progetto e procurarsi una chiave da sviluppatore da inserire nell'activity principale dell'applicazione. La documentazione di Parse è molto più scorrevole e comprensibile, adatta a chi si avvicina per la prima volta a questo tipo di servizi ed entrambi forniscono esempi e tutorial per l'implementazione dei loro servizi, i tutorial di StackMob per Android sono maggiori in numero ma la documentazione di parse è integrata con numerosi esempi.
Per esempio, si noti la gestione di un nuovo utente da parte di StackMob e da Parse: StackMob fornisce la classe StackMobUser e per la registrazione di un nuovo utente sono sufficienti due righe di codice [StaMo]:

```
1    StackMobUser user = new StackMobUser ("bob", "mypassword");
2    user.save();
```

Questa però può contenere solo i campi "username" e "password", per inserire altri dati sarebbe necessario estendere la classe [StaMo]:

```
1    import com.stackmob.sdk.model.StackMobUser;
2
3    public class User extends StackMobUser {
4
```

```
5       private List<TaskList> taskLists;
6       private String email;
7
8       protected User(String username, String password) {
9          super(User.class, username, password);
10      }
11
12
13      public List<TaskList> getTaskLists() {
14         return taskLists;
15      }
16
17      public void setTasks(List<TaskList> taskLists) {
18         this.taskLists = taskLists;
19      }
20   }
```

Parse, contrariamente, fornisce una classe ParseUser che permette di eseguire una registrazione in poche righe e di aggiungere qualsiasi campo si desideri aggiungere semplicemente specificandone il nome [Parse]:

```
1    ParseUser user = new ParseUser();
2    user.setUsername("my name");
3    user.setPassword("my pass");
4    user.setEmail("email@example.com");
5    // other fields can be set just like with ParseObject
6    user.put("phone", "650-253-0000");
7    user.signUpInBackground(new SignUpCallback() {
8      public void done(ParseException e) {
9        if (e == null) {
10         // Hooray! Let them use the app now.
11       } else {
12         // Sign up didn't succeed. Look at the ParseException
13         // to figure out what went wrong
14       }
15     }
16   });
17
18
19
20
```

Inoltre nella documentazione di Parse è ben spiegato, nella stessa sezione, perché si utilizza il metodo “signUpInBackground” e non “saveInBackground” e qual è la struttura dell’oggetto ParseUser, StackMob, invece, riporta solo il codice di esempio, raramente supportato da opportuni commenti. Nella documentazione relativa alla gestione degli utenti di Parse, al contrario che in quella di StackMob, vengono anche spiegati: Il funzionamento della verifica tramite mail, l’eventuale accesso tramite utente anonimo e la gestione dell’accesso ai file da parte degli utenti.

Entrambi hanno un blog per comunicare le novità con gli utenti. L'assistenza tecnica fornita da Parse è più diretta dando la possibilità di porre il proprio problema direttamente agli altri utenti e di navigare tra i problemi già risolti da altri utenti, anche se non tutti i quesiti posti ottengono risposta, in alternativa è possibile contattare il servizio di assistenza via mail. StackMob fornisce un servizio simile ma non si ha la possibilità di scrivere direttamente ai loro operatori.
In conclusione Parse è più adatto a piccoli progetti vista la semplicità di implementazione e i limiti imposti dalle tariffe, per applicazioni rivolte ad un numero molto elevato di utenti è invece preferibile utilizzare StackMob che non impone limiti sulle richieste mensili.

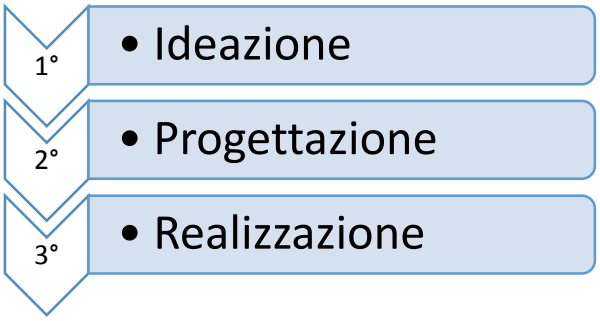
4.4 Conclusioni

La realizzazione di un backend di un'applicazione può essere implementata da zero scegliendo uno stack e un hosting provider, scrivendo il server code testandolo e dimensionando il server in base al traffico previsto, ciò consiste in una grossa mole di lavoro anche per realizzare backend semplici. Per alleggerire tale mole di lavoro si possono utilizzare alcune delle soluzioni viste precedentemente per "appaltare" parte del lavoro a servizi nel cloud.
La soluzione implementata utilizzando i PaaS i cui elementi permettono di sviluppare, testare, implementare e gestire le applicazioni senza i costi e la complessità associati all'acquisto, la configurazione, l'ottimizzazione e la gestione dell'hardware e del software di base, dà la possibilità di risparmiare molto tempo agli sviluppatori che devono però comunque gestire il server code e testarlo in modo da comunicare con l'applicazione. Questo dà la possibilità allo sviluppatore di realizzare in modo completo il proprio backend in base alle proprie esigenze.
La soluzione implementata tramite BaaS che forniscono direttamente agli sviluppatori un sistema completo di backend, gestione di notifiche push e integrazione con social network e, ultimamente, anche la possibilità di scrivere cloud code, da, invece, la possibilità di risparmiare ancora più lavoro utilizzando un sistema di backend precostruito ma, d'altro canto, è meno flessibile a esigenze particolari sulla gestione di tale backend.

L’applicazione da realizzare non ha particolari esigenze di traffico. Dato che la parte dell’applicazione che usufruirà del servizio di backend sarà accessibile solo a chi avrà effettuato il login, tale bacino di utenza è di fatto inferiore a quello dei clienti della trattoria e, anche se le ordinazioni venissero gestite tutte utilizzando l’applicazione, le iterazioni sarebbero comunque talmente poche e poco frequenti. Di conseguenza, anche replicando il sistema per altre applicazioni dello stesso tipo il traffico di dati sarebbe comunque molto basso, questo ragionamento è però valido solo per singoli ristoranti e non può essere applicato a catene di ristorazione. I servizi offerti da un BaaS consentono di realizzare login, di inviare notifiche push e di integrare i social network in modo semplice e veloce attraverso i loro SDK, facendo risparmiare molto tempo in fase di programmazione. Mentre utilizzando un custom backend si ha una flessibilità molto maggiore e la gestione degli ordini potrebbe essere implementata più efficientemente, e sarebbe poi adattabile anche al caso di catene di ristorazione.
Per concludere è necessario impostare delle priorità nel progetto, in questo caso si è scelto di privilegiare il minor tempo di sviluppo utilizzando un BaaS rispetto alla maggiore flessibilità fornita da un PaaS e, nello specifico, di utilizzare Parse piuttosto che StackMob per la completezza della propria documentazione e il sistema di assistenza più diretto.

Capitolo 5
Applicazione

La costruzione di un qualsiasi progetto, e quindi anche dell'applicazione in oggetto, necessita lo svolgersi di tre fasi che ne determinano la struttura di lavoro per permetterne il corretto sviluppo nel tempo (Vedi fig. 5.1).



(Fig.5.1) Processi per lo sviluppo di un progetto.

La prima fase è quella dell'ideazione, dove nasce il progetto e ne viene valutata l'utilità e la realizzabilità.
La seconda è quella della progettazione, trattata nelle sezioni precedenti, si compone di una valutazione dei requisiti del progetto e di una scelta calibrata su come soddisfarli al meglio, impostando le linee guida per la fase successiva.
Infine vi è la fase della realizzazione, trattata in questa sezione, dove viene implementato il progetto ideato nella prima fase utilizzando i principi e gli strumenti impostati dalla seconda.

5.1 Strumenti per lo sviluppo

Effettuate le scelte progettuali per la piattaforma di sviluppo, l'interfaccia, l'architettura e il backend dell'applicazione è possibile passare alla fase di realizzazione, dove esse saranno implementate secondo i concetti di semplicità ed intuitività specificati in fase di progettazione.

Prima di procedere alla programmazione vera e propria dell'applicazione è necessario conoscere l'ambiente di sviluppo e comprendere quali programmi e strumenti siano utilizzati nelle varie fasi.

5.1.1 Programmi

Il primo programma utilizzato in fase di progettazione è stato "Balsamiq Mockups", di Balsamiq S.R.L., un software per sviluppatori, designer, progettisti ed altri creativi, disegnato per creare *wireframe*, ovvero design di interfacce utenti, utilizzato per creare schizzi di schermate per applicazioni di tutti i tipi: siti web, applicazioni web, desktop e mobile [BaMoc]. Grazie a questo software è stato possibile disegnare gli schizzi delle pagine dell'applicazione e collegarli tra loro, in modo da strutturare l'interfaccia utente e la navigazione tra le pagine in fase di progettazione.

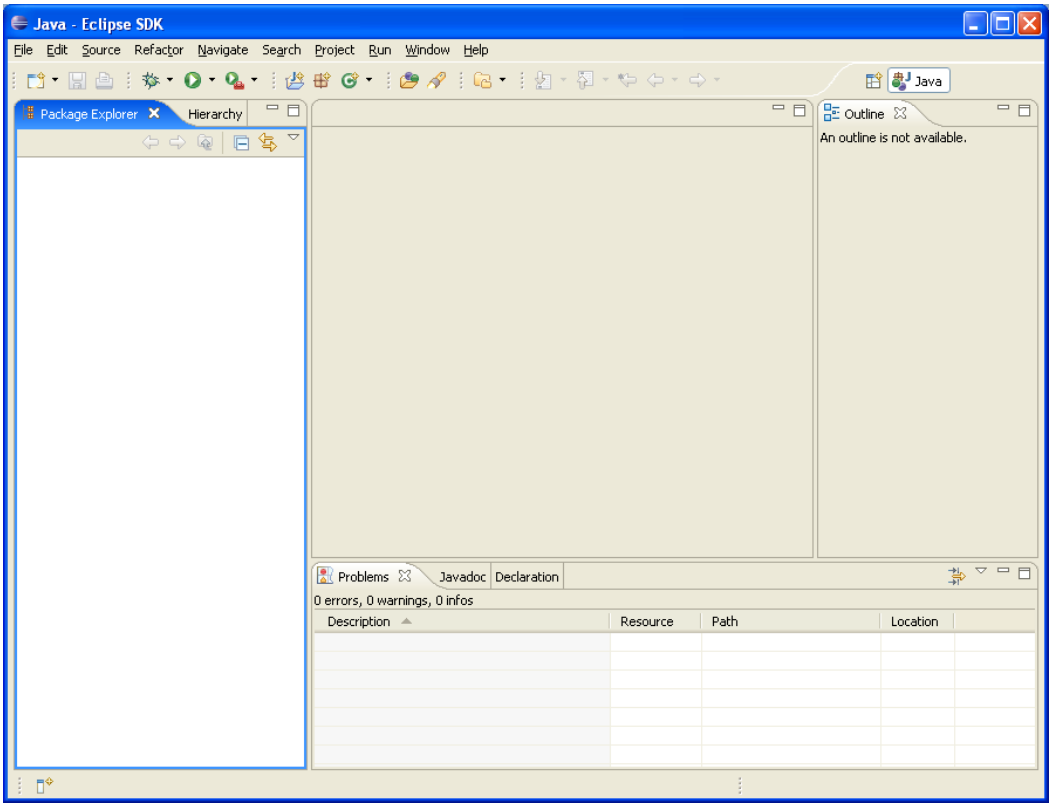
(Fig. 5.2) Balsamiq Mockups.

L'ambiente di lavoro utilizzato da questo software è essenziale, tutto è raggiungibile dalla schermata principale e non vi sono schede di strumenti aggiuntivi da utilizzare

ma ogni componente viene supportato da una propria scheda per la modifica (Vedi fig. 5.2).

Effettuata poi la scelta di sviluppare l'applicazione per Android si utilizza l'ambiente di sviluppo Eclipse, un ambiente di sviluppo integrato multi-linguaggio e multipiattaforma. Ideato da un consorzio di grandi società sullo stile dell'open source.

La piattaforma di sviluppo è incentrata sull'uso di plug-in, delle componenti software ideate per uno specifico scopo, per esempio la generazione di diagrammi UML, ed in effetti tutta la piattaforma è un insieme di plug-in, versione base compresa, e chiunque può sviluppare e modificare i vari plug-in. Nella versione base è possibile programmare in Java, usufruendo di comode funzioni di aiuto quali: completamento automatico, suggerimento dei tipi di parametri dei metodi, possibilità di accesso diretto a CVS, un sistema che controlla la versione del programma installato, e riscrittura automatica del codice in caso di cambiamenti nelle classi.



(Fig. 5.3) Eclipse.

L'ambiente di sviluppo proposto da Eclipse viene presentato, in versione base, come l'insieme di quattro finestre, una per la navigazione tra i progetti a sinistra, uno per i messaggi ricevuti in basso, uno per la visualizzazione gerarchica dell'oggetto

selezionato a destra e uno centrale nel quale viene visualizzato il codice (Vedi fig. 5.3).
Tale ambiente va però integrato con appositi SDKs e plug-in necessari o utili allo sviluppo dell'applicazione.

5.1.2 SDKs e plug-in

L'ambiente di sviluppo base di Eclipse non è sufficiente per la programmazione della nostra applicazione e va integrato con vari strumenti.
Primo tra tutti vi è l'Android SDK, esso contiene le librerie API e strumenti di sviluppo necessari per la costruzione, prova, e applicazioni di debug per Android.
In particolare il bundle ADT fornito da Android comprende i componenti essenziali di Android SDK e una versione dell'IDE di Eclipse integrando Android Developer Tools (ADT).
Il pacchetto ADT include tutto il necessario per iniziare a sviluppare applicazioni per [AndDe]:

- Eclipse + ADT plug-in.
- Android SDK Tools.
- Android Platform-tools.
- La più recente piattaforma Android.
- L'a più recente versione del sistema Android rilasciata per l'emulatore.

L'installazione di questo renderà disponibili per il download le varie versioni di Android e, attraverso l'emulatore, la simulazione di dispositivi Android sui quali poter provare le applicazioni.
Dato che si è scelto di utilizzare Parse come sistema di backend è necessario anche installare il Parse SDK, all'interno del progetto, contenente le librerie di Parse per la gestione del backend e inizializzare l'applicazione inserendo nella prima Activity la chiave ottenuta da Parse:

```
1       Parse.initialize(this, "ID_APPLICAZIONE", "CHIAVE_PARSE");
```

E inserendo all'interno del manifest gli opportuni permessi ad accedere alla rete.
Per la gestione della mappa si sono importate le librerie Google Play Services, disponibili all'interno dell'Android SDK, anche in questo caso è necessario

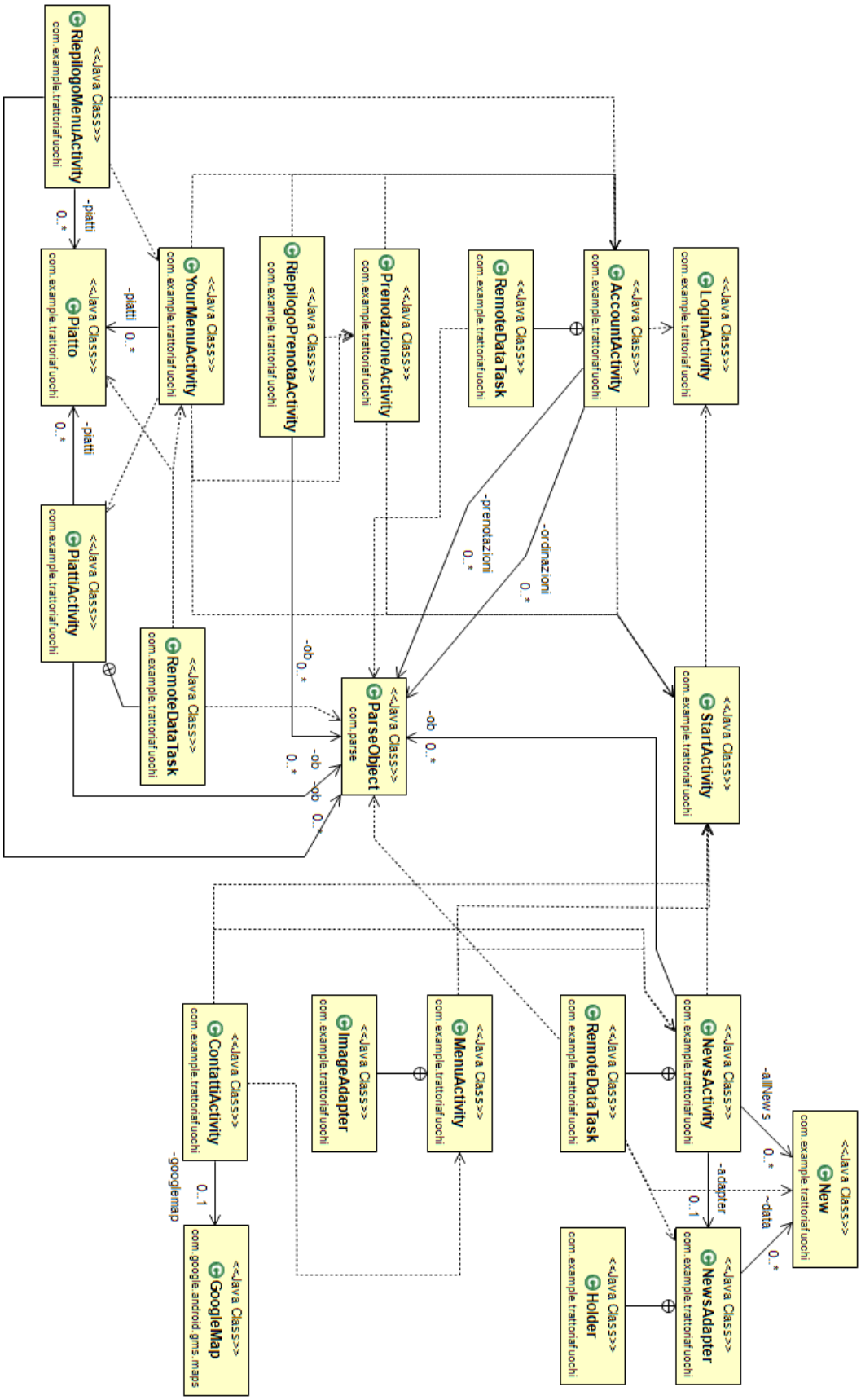
inizializzare l'applicazione per la gestione delle mappe e modificare il manifest gestendo gli opportuni permessi dell'applicazione.
Per semplificare l'accesso alle chiavi da utilizzare per Parse e per le mappe si è scelto di integrare Eclipse con il keytool plug-in, un plug-in che gestisce i keystores e i certificati dando l'opportunità di:

- Esportare certificati, con o senza la chiave privata.
- Creare di certificati.
- Creare di chiavi.
- Aprire esistenti keystore.
- Visualizzare tutte le informazioni disponibili su ogni certificato.
- Aprire il certificati esistenti con l'estensione. cer.
- Mostrare la lista dei certificati in un keystore.
- Mostrare il certificato in un keystore dove ci sono chiavi private.
- Monitorare i cambiamenti nei keystore.
- Aprire un keystore di default all'avvio di Eclipse.
- Generare CSR.

Esso consente di creare certificati da inserire in un keystore.
L'ultimo plug-in istallato come supporto in fase di progettazione, in particolare per la realizzazione di grafici UML per la progettazione delle varie classi, è Papyrus plug-in, che mira a fornire un ambiente integrato per la modifica ogni tipo di modello EMF e in particolare sostenendo UML e linguaggi di modellazione correlati quali SysML e MARTE. Papyrus fornisce editor di diagrammi per linguaggi di modellazione EMF-based, come UML 2 e SysML.

5.2 Realizzazione

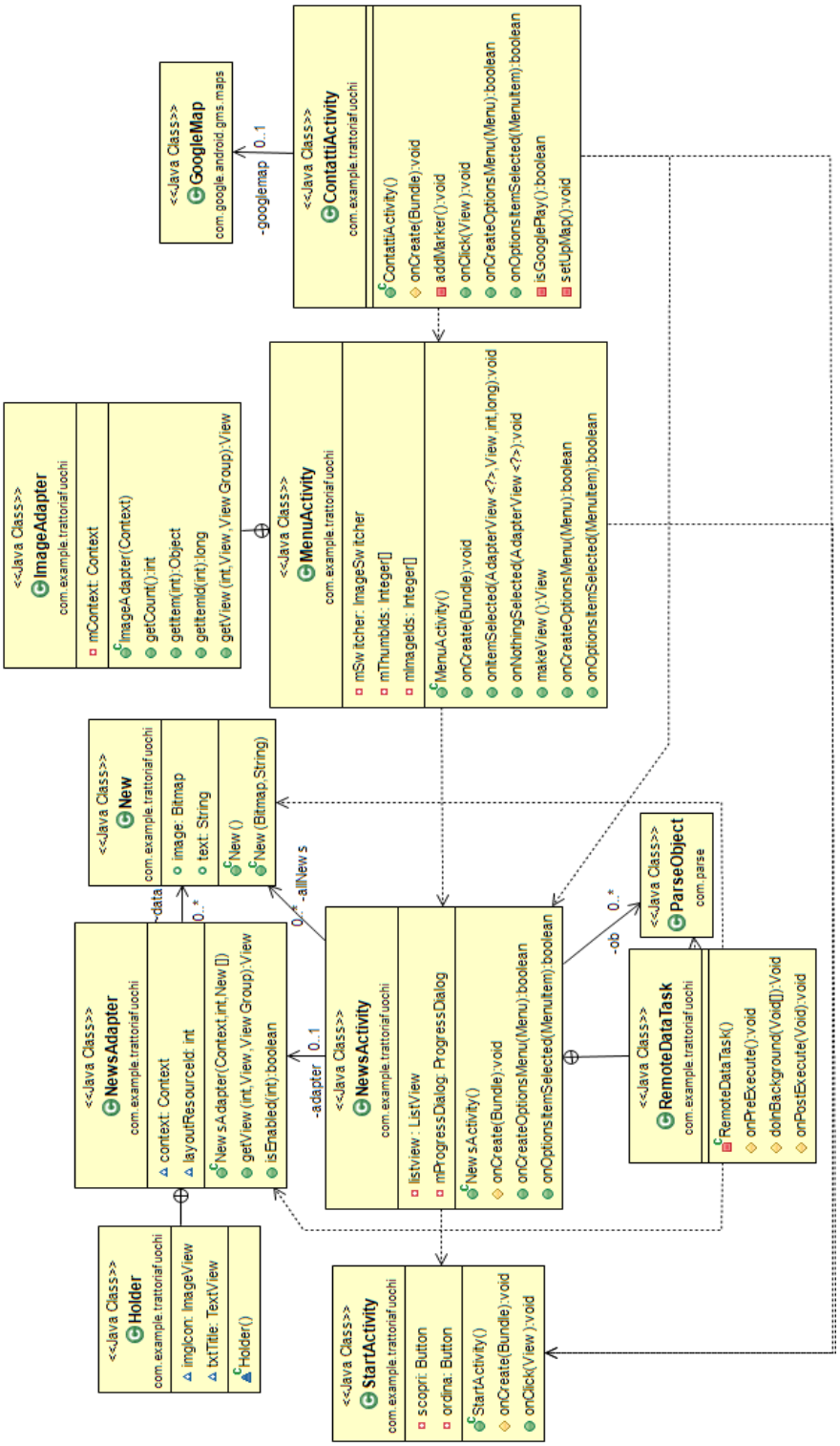
Impostata la struttura che deve avere l'applicazione e adattato l'ambiente di sviluppo per la programmazione di essa è possibile realizzare l'applicazione, prima di tutto vengono impostate le Activity e le relazioni tra di esse, gli oggetti di supporto alle prenotazioni o ordinazioni e i punti di contatto col backend.
In figura 5.4 è mostrato come siano collegate le varie classi create, e che relazione abbiano con gli oggetti ParseObject e GoogleMaps, appartenenti alle relative librerie.



(Fig. 5.4) Diagramma UML dell'applicazione.

Ad ogni pagina dell'applicazione è associata un'Activity, questo per implementare la propria funzione:

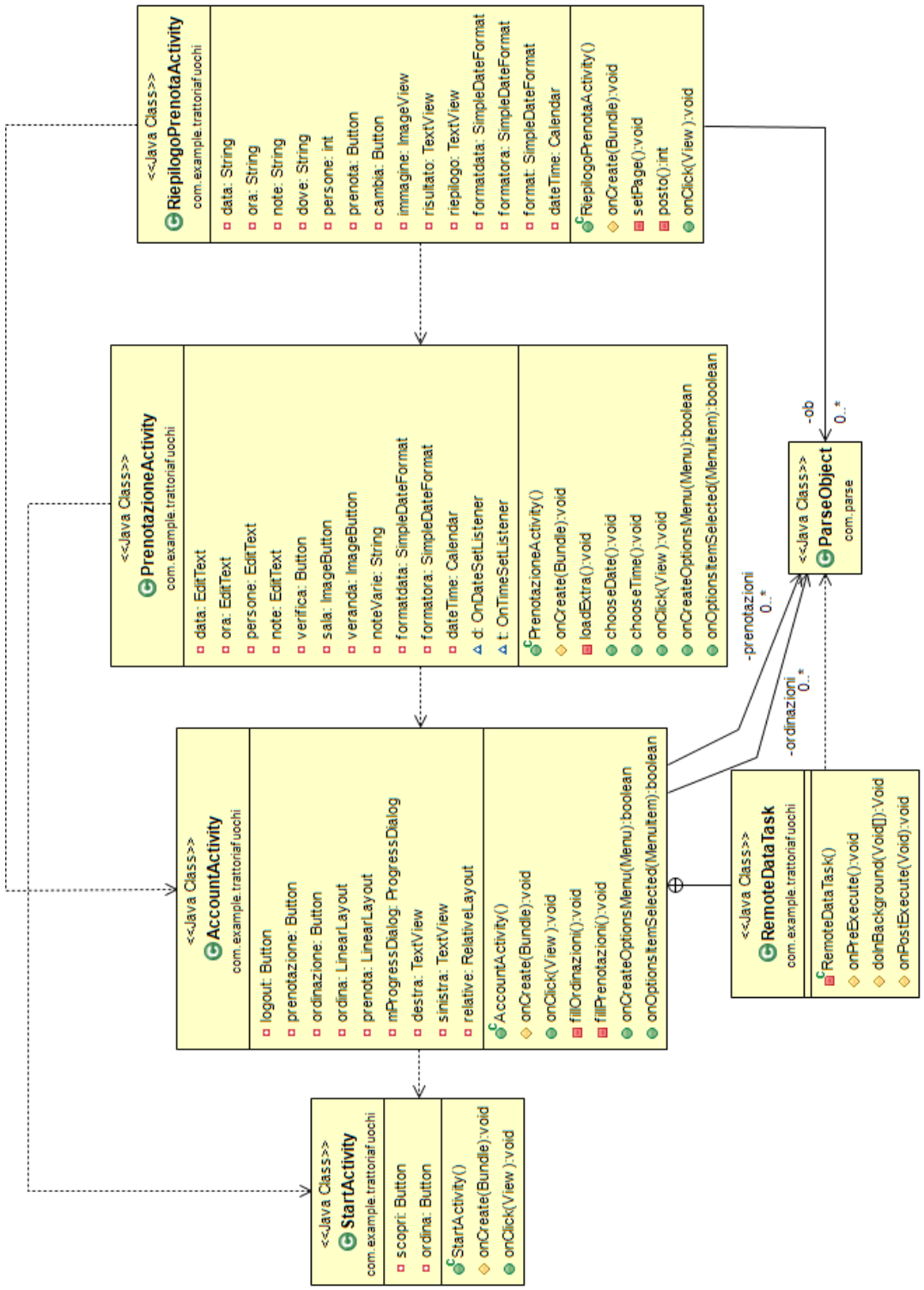
- Alla pagina d'inizio viene associata l'Activity "StartActivity", il cui scopo è solo quello di indirizzare l'utente verso la sezione dell'applicazione cercata.



(Fig. 5.5) Diagramma UML della Presentazione.

- Alla pagina delle news viene associata l'Activity "NewsActivity", essa si collega al backend (Vedi fig. 5.5) e costruisce una lista delle news presenti.
- Alla pagina del Menù viene associata l'Activity "MenuActivity", il cui scopo è quello di visualizzare le pagine del menù svolto attraverso una galleria di immagini.
- Alla pagina dei contatti viene associata l'Activity "ContattiActivity", essa mostra la posizione del locale su una mappa.
- Le pagine di registrazione/login e dell'account vengono implementate rispettivamente dalle Activity "LoginActivity" e "AccountActivity", la prima compare solamente nel caso non sia stato effettuato un login e, una volta effettuato, rimanda alla seconda che fornisce l'accesso alla parte dell'applicazione relativa alla gestione delle prenotazioni e delle ordinazioni.
- La pagina per le prenotazioni viene associata all'Activity "PrenotazioneActivity", attraverso essa vengono richiesti i dati relativi alla prenotazione, poi trasmessi all'Activity "RiepilogoPrenotazioneActivity", implementazione della pagina di riepilogo delle prenotazioni, che verifica col backend la disponibilità (Vedi fig. 5.6) e, se vi è, completa la prenotazione.
- Alla pagina delle ordinazioni viene associata l'Activity "YourMenuActivity", essa permettere di accedere alla pagina dei piatti, implementata attraverso "PiattiActivity", in modo da aggiungerli alla propria ordinazione, tali dati sono poi da trasmettere all'Activity "RiepilogoMenuActivity", che ne verifica la disponibilità col backend (Vedi fig. 5.7) e, una volta verificata, fornisce l'opportunità di scegliere i dati relativi al tipo di ordinazione e completarla.

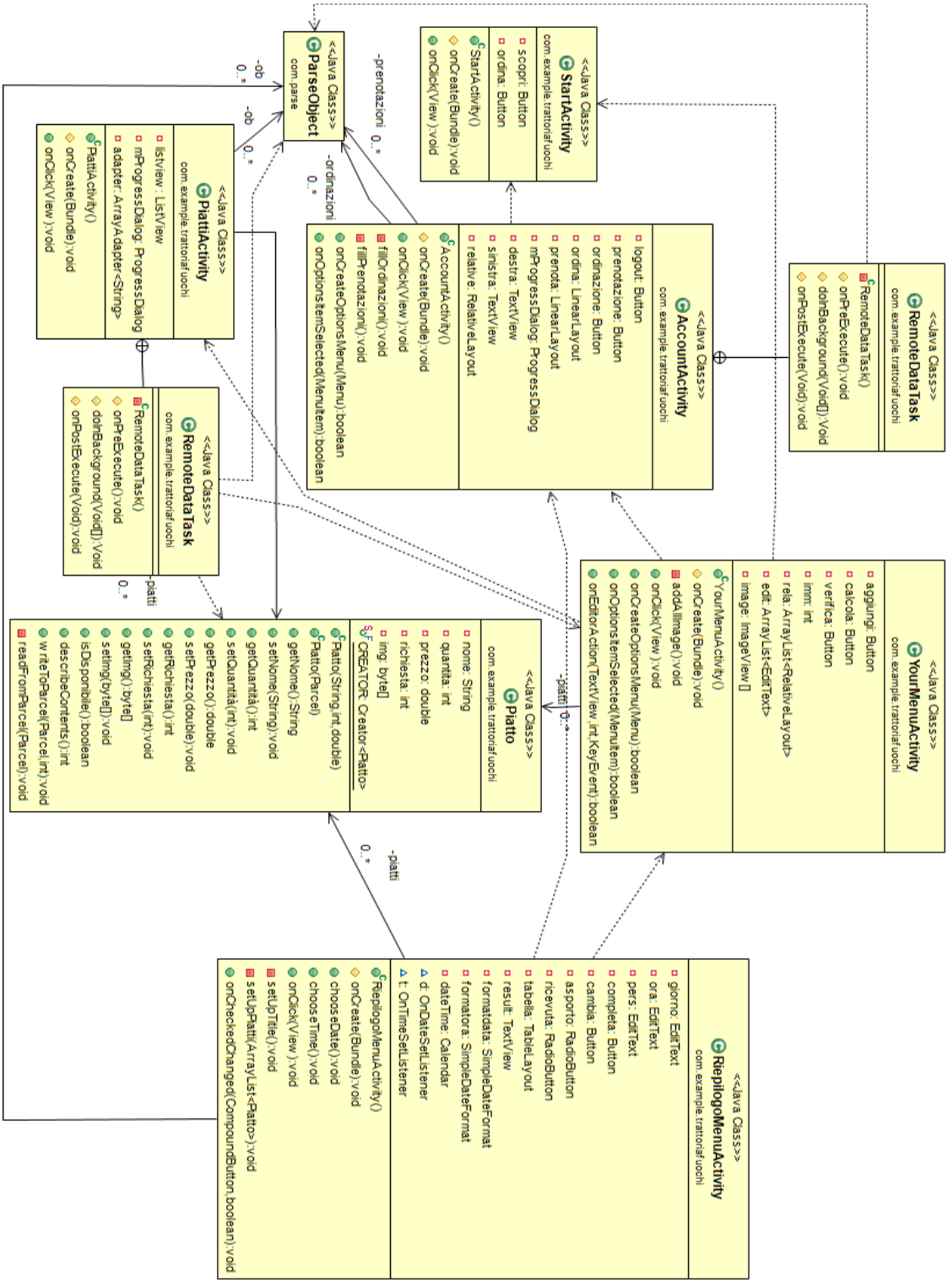
La creazione della lista delle news nella relativa pagina è affidata ad una estensione della classe Adapter, "NewsAdapter", che utilizza degli appositi oggetti di tipo "New", costituiti da un'immagine ed una descrizione (Vedi fig.5.4).
La gestione delle ordinazioni è implementata attraverso l'utilizzo di un oggetto "Piatto" che estende la classe Parcelable per poter essere passato tra un'Activity e l'altra, così viene creato un ArrayList<Piatto> che viene rimbalzato tra le Activity relative all'ordinazione fino al completamento di essa, dove i dati vengono poi salvati sul backend (Vedi fig. 5.7).



(Fig. 5.6) Diagramma UML delle Prenotazioni.

Dato che i dati vanno caricati ogni volta da Parse, gli algoritmi più utilizzati in queste activity sono quelli per la creazione dinamica di tabelle e, più in generale, di layout. Il resto del codice è atto per lo più ai collegamenti tra le activity e all'interazione col

backend o la mappa. Solo nel caso delle prenotazioni vi è un controllo su il numero delle persone richieste per verificarne la disponibilità e verificare che non superino il tetto massimo di capacità del locale.



(Fig. 5.7) Diagramma UML delle Ordinazioni.

Conclusioni

Il progetto realizzato consiste in un'applicazione ideata e realizzata per un'attività commerciale d'esempio, nel caso specifico di una trattoria. Per realizzarla, una volta valutati i requisiti strutturali e di portabilità di tale applicazione, si è mostrato quale fosse la piattaforma di distribuzione, e quindi l'ambiente di sviluppo, più adatto a soddisfarli, trattando una panoramica dei sistemi operativi mobili, al momento, più utilizzati. Allo stesso modo, riconosciuta la necessità di utilizzare un backend dell'applicazione, si sono confrontate diverse strutture di backend, osservando parte della grande varietà di servizi di cloud computing disponibili e distribuiti, per poter scegliere quella più adatta ad un'applicazione per piccole o medie imprese.
Nello specifico attraverso l'utilizzo di questa applicazione la trattoria si presenta alla clientela in modo decisamente innovativo offrendole un contatto diretto e la possibilità di verificare in tempo reale i piatti disponibili, il loro costo e una loro foto, oltre che la disponibilità per un'eventuale prenotazione ancor prima di effettuarla.
L'applicazione realizzata, oltre ad essere strumento pubblicitario e informativo per un locale, è anche in grado di aprire alla propria clientela, attraverso un sistema informatizzato, la gestione delle ordinazioni e delle prenotazioni, aumentando l'accessibilità e la portabilità del locale stesso.
Questo è stato fatto con lo scopo di mostrare come l'evoluzione tecnologica avvenuta negli ultimi anni degli smartphone e la portabilità della rete, sempre più vasta grazie alla distribuzione sempre maggiore di tali dispositivi, possa fornire un utile servizio anche ad attività commerciali che non hanno a che fare con tale settore.
È auspicabile che in futuro sempre più attività, commerciali e non, si aprano al settore delle applicazioni mobili sfruttandolo non solo come strumento pubblicitario ma anche come supporto vero e proprio al lavoro svolto.

Ringraziamenti

Desidero innanzitutto ringraziare il Professor Corazza Giovanni Emanuele per gli insegnamenti durante il terzo anno di laurea e per la disponibilità e il tempo dedicatomi. Inoltre, ringrazio sentitamente il Prof. Alessandro Vanelli-Coralli e il Dott. Ing. Matteo Collina che sono stati sempre disponibili a dirimere i miei dubbi durante la stesura di questo lavoro ed hanno passato numerose ore a controllarlo. Intendo poi ringraziare la mia famiglia che, in ambito lavorativo, mi ha fornito gli strumenti e le conoscenze necessari per poter proporre un servizio migliore col progetto realizzato. Infine, ho desiderio di ringraziare con affetto mia madre per il sostegno ed il grande aiuto che mi hanno dato ed in particolare Fiorenza per essermi stata vicino ogni momento durante questi mesi di lavoro.

Bibliografia

[AWSer] Amazon Web Services, Inc., "*Amazon Web Services*", http://aws.amazon.com.

[Racks] Rackspace, US Inc., *"Rackspace*", http://rackspace.com.

[AppFo] AppFog, Inc., "*AppFog*", http://appfog.com.

[Herok] Heroku, Inc., "*Heroku*", http://heroku.com.

[Parse] Parse, Inc., "*Parse*", http://parse.com.

[StaMo] StackMob, Inc., "StackMob", http://stackmob.com.

[Ree09] George Reese, *"Cloud Application Architectures"*, O'Really, 2009.

[Zai11] Zaigham Mahmood, Richard Hill, *"Cloud Computing for Enterprise* Architectures", Springer, 2011.

[ITess] NetMediaEurope Italia S.r.l. ©, *"ITespresso",* http://itespresso.it.

[MOSAT] Xiao-Feng Li, Yong Wang, Jackie Wu, Kerry Jiang, Bing Wei Liu, *"Mobile OS* Architecture Trends", Intel Technology Journal, Vol. 16, 2012.

[Apple] Apple Inc., *"Apple*", http://apple.com.

[AndDe] Google Inc., *"Android Developers",* http://developer.android.com.

[BlaBa] Research In Motion Limited, *"BlackBarry",* http://us.blackberry.com.

[WiPDC] Microsoft Corporation, **"***Windows Phone Dev Center",* http://msdn.microsoft.com.

[Kanta] Kantar Group, *"Kantar world panel",* http://kantarworldpanel.com.

[GooMa] Google Inc., *"Google Maps",* http://developers.google.com/maps.

[OJava] Oracle Inc., *"Java SE",* http://docs.oracle.com/javase.

[BaMoc] Balsamiq S.R.L., *"Balsamiq Mockups",* http://www.balsamiq.com.