



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica
Corso di Laurea triennale in Informatica

TESI DI LAUREA

SimplyText: una tecnica per l'editing di testo su dispositivi Android

Relatori

Prof. Vittorio Fuccella
Dott. Mattia De Rosa

Candidato

Albanese Maria Giovanna
Matr.: 0512105770

Anno Accademico 2020/2021

A chi ha sempre creduto in me.

Ringraziamenti

Desidero dedicare questo spazio a tutti coloro che mi hanno sostenuto e mi sono stati accanto durante questo periodo di profondo apprendimento, non solo a livello scientifico, ma anche personale.

Vorrei ringraziare, innanzitutto, il professore Vittorio Fuccella, relatore di questa tesi di laurea, per l'aiuto che mi ha fornito e la conoscenza che mi ha donato.

Ringrazio il dottore Mattia De Rosa, il quale mi ha accompagnato lungo tutto questo percorso, insegnato tanto e aiutato nei momenti di difficoltà, con la massima disponibilità e cordialità.

Ringrazio infinitamente la mia famiglia: Mamma, Papà e Davide che mi ha sempre sostenuto e incoraggiato, appoggiando ogni mia decisione, permettendomi di raggiungere questo traguardo. Ringrazio i miei nonni, anche chi non è qui, ma sono sicuro stia facendo il tifo per me, per essere sempre stati orgogliosi di me e dei miei traguardi.

Per ultimi, ma non meno importanti, ringrazio i miei amici, che mi hanno sempre fornito tutto il sostegno necessario per affrontare ogni ostacolo di questo percorso.

Abstract

In un'era in cui il tempo sembra scorrere sempre più velocemente, la tecnologia ed in particolare gli smartphone, ci permettono di velocizzare alcune azioni che prima risultavano lente e tediose.

Fra le pratiche più comuni troviamo la necessità di scrivere lunghi testi o di modificarne il contenuto.

Occorre, quindi, cercare nuovi metodi per rendere questa pratica sempre più veloce ed efficiente.

Questa tesi propone una nuova tecnica di modifica del testo alternativa a quella predefinita in Android con l'obiettivo di consentire all'utente l'esecuzione di operazioni di editing quali selezione, taglia, copia, incolla in maniera più efficiente. La tecnica si basa sull'utilizzo di softkey rappresentanti le frecce direzionali e le operazioni di editing. La tecnica è stata inoltre confrontata con la tecnica di editing predefinita di Android in uno studio sperimentale.

Indice dei contenuti

CAPITOLO 1 INTRODUZIONE	1
1.1 CONTESTO APPLICATIVO.....	1
1.2 MOTIVAZIONI E OBIETTIVI.....	2
1.3 SIMPLYTEXT	2
1.4 STRUTTURA DELLA TESI.....	3
CAPITOLO 2 STUDIO DEI SISTEMI OPERATIVI E DELLO STATO DELL'ARTE ..4	
2.1 SISTEMI OPERATIVI MOBILI	4
2.1.1 <i>iOS</i>	5
2.1.2 <i>Android</i>	5
2.1.3 <i>Considerazioni</i>	7
2.2 EDITING DEL TESTO PREDEFINITA IN ANDROID	8
2.3 SVANTAGGI	8
2.4 LAVORI CORRELATI.....	9
2.4.1 <i>TouchTap</i>	9
2.4.2 <i>Tecnica proposta da S.Finelli et al.</i>	10
CAPITOLO 3 PROGETTAZIONE DI SIMPLYTEXT	12
3.1 SOLUZIONE AL POSIZIONAMENTO NEL TESTO.....	12
3.2 SOLUZIONE AL RECUPERO DI VECCHIE MODIFICHE	14
3.3 VALUTAZIONE EURISTICA.....	14
CAPITOLO 4 IMPLEMENTAZIONE DI SIMPLYTEXT	16
4.1 STRUMENTI DI SVILUPPO	16
4.2 STRUTTURA LOGICA	17
4.3 STRUTTURA DEL PROGETTO	18
4.3.1 <i>AndroidManifest.xml</i>	19
4.3.2 <i>Cartella res</i>	19
4.3.3 <i>Gradle</i>	20
4.4 MODULO SIMPLYTEXT	21
4.4.1 <i>note_Editor.xml</i>	21

4.4.2 <i>NoteEditor.java</i>	22
CAPITOLO 5 ESPERIMENTO	26
5.1 PARTECIPANTI.....	26
5.2 APPARECCHIATURA	26
5.3 PROCEDURA.....	27
5.4 RISULTATI E DISCUSSIONE	31
5.5 SODDISFAZIONE DEGLI UTENTI E COMMENTI.....	33
CAPITOLO 6 CONCLUSIONI	34

Indice delle figure

Figura 1 Architettura Android (Prisco, 2021)	6
<i>Figura 2 Previsione della quota di mercato del sistema operativo</i>	7
<i>Figura 3 SimplyText step by step</i>	13
<i>Figura 4 Icone Move e Ghost</i>	13
Figura 5 Bottone backands	14
<i>Figura 6 Ciclo di vita dell'Activity</i> (Prisco, 2021)	17
Figura 7 Struttura del progetto	19
Figura 8 Cartella res	20
<i>Figura 9 Layout Grafico SimplyText</i>	22
<i>Figura 10 Button Backward</i>	23
<i>Figura 11 ButtonMove</i>	24
<i>Figura 12 Elenco dei task</i>	27
<i>Figura 13 Task completato</i>	30
<i>Figura 14 Tempo medio di completamento dell'attività per le due tecniche di editing</i>	32
<i>Figura 15 Tempo di completamento dei task</i>	32

Indice delle tabelle

<i>Tabella 1 Mappatura fra gesti e operazioni di editing (Fuccella, s.d.)</i>	10
<i>Tabella 2 Approfondimento dei task (per ridurre le dimensioni della tabella, i testi delle attività vengono visualizzati solo in parte).</i>	28
<i>Tabella 3 Bilanciamento dei partecipanti utilizzato durante l'esperimento. Dimensione font abbreviata (s - piccolo, m- medio)</i>	31

CAPITOLO 1

INTRODUZIONE

Questo capitolo illustra il contenuto della tesi mostrando motivazioni e obiettivi fornendo una breve sintesi del contenuto dell'elaborato.

1.1 Contesto applicativo

Al giorno d'oggi, le persone hanno sempre a disposizione un cellulare, lo smartphone fa parte delle nostre vite ed il suo utilizzo è in continua crescita.

Una ricerca dell'Istat rileva che tra gli utenti di 14 anni e più, il 91,8% utilizza lo smartphone per fare ricerche sul web, il 43,3% accede tramite PC da tavolo, il 27,2% utilizza il laptop o il netbook, seguono quelli che si avvalgono del tablet (25,7%) mentre il 6,1% utilizza e-book, smartwatch o altri dispositivi. (AGI, 2019)

Computer e internet hanno cambiato il modo di stare insieme e di comunicare, studiare, divertirsi e lavorare, ma l'uso dello smartphone va ancora oltre. I cellulari non servono solo per telefonare, ascoltare musica, inviare messaggi o fotografare, l'utilizzo di questi dispositivi riguarda oramai tutte le attività umane.

Il principale vantaggio di questo strumento è proprio quello di utilizzare le sue funzionalità in diversi contesti e in diversi momenti della giornata. Favorisce, inoltre, la produttività perché consente di svolgere attività nei momenti morti della giornata, come durante l'attesa del treno o durante la pausa pranzo. (Binetti, 2021)

1.2 Motivazioni e obiettivi

Ci ritroviamo spesso a dover fare operazioni semplici come inviare un messaggio o modificare la lista della spesa nei luoghi meno opportuni, dove il livello di concentrazione è molto basso e la probabilità di sbagliare estremamente alta.

Operazioni come selezione del testo, taglia, copia o incolla su dispositivi molto piccoli richiedono molto tempo ed un elevato livello di concentrazione. Il problema principale riguarda la dimensione delle dita ed il poco spazio sullo schermo. Esse, infatti, tendono a coprire parzialmente l'area d'interesse rendendo la pratica di editing poco piacevole e noiosa.

Sebbene l'approccio adottato attualmente da Android sia abbastanza valido, è stato deciso di sperimentare un nuovo approccio alla modifica del testo che velocizzi sempre di più le pratiche sopraccitate.

1.3 SimplyText

In questo lavoro di tesi viene presentata una nuova tecnica di editing del testo su dispositivi Android che prevede l'utilizzo di softkey per spostarsi nel testo e per svolgere operazioni come copia, taglia e incolla.

Successivamente, è stata svolta un'indagine su sette persone per capire le prime impressioni e la fattibilità del metodo. Per svolgere l'esperimento, il modulo implementato è stato inserito in un'applicazione la quale, attraverso dei task, si presta a testare i seguenti comandi:

- **Selezione:** toccare approssimativamente l'inizio e la fine della selezione sul testo e correggere la selezione attraverso le frecce di posizionamento fornite dall'app;
- **Copia/Taglia:** dopo aver selezionato il testo utilizzare il bottone copia o taglia per svolgere l'operazione richiesta dal task;

- **Incolla:** Dopo aver copiato o tagliato il testo con l'apposito bottone incollare il testo nel punto indicato dal task aiutandosi con le frecce di posizionamento.

1.4 Struttura della tesi

Gli argomenti introdotti sono trattati in dettaglio nei seguenti capitoli che compongono la tesi:

- **Capitolo 2:** tratta dello stato dell'arte. Saranno sottoposti ad un'attenta e accurata valutazione i sistemi operativi mobili al fine di comprendere le motivazioni che ci sono dietro la scelta di Android. Verranno analizzate tecniche e soluzioni già esistenti con l'obiettivo di migliorarne l'efficienza;
- **Capitolo 3:** ha lo scopo di illustrare l'idea proposta per risolvere il problema, mostrando una possibile tecnica per l'editing di testo. L'analisi di tale tecnica sarà l'oggetto fondamentale di questo lavoro;
- **Capitolo 4:** mostra nel dettaglio le specifiche del software realizzato, le tecnologie utilizzate e le scelte implementative;
- **Capitolo 5:** illustra il confronto fra SimplyText e la tecnica di editing predefinita di Android analizzando i risultati. Verrà descritto l'intero esperimento e le tecnologie utilizzate;
- **Capitolo 6:** tiene le conclusioni finali del lavoro descritto, proponendo spunti e riflessioni per eventuali sviluppi futuri e approfondimenti sul tema.

CAPITOLO 2

STUDIO DEI SISTEMI OPERATIVI E DELLO STATO DELL'ARTE

Questo capitolo illustra un approfondimento sullo stato dell'arte, i lavori già presenti in letteratura e gli aspetti di ricerca trattati in questo lavoro di tesi.

2.1 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 se svilupparselo internamente, come Apple.

Questi sistemi operativi hanno strutture e ambienti di sviluppo dedicati molto diversi, a tal proposito è necessario conoscere gli aspetti e comprendere le differenze esistenti tra i vari sistemi operativi, in modo da poter scegliere per quale piattaforma concentrare le proprie ricerche.

Escludendo piccoli sistemi emergenti ancora poco conosciuti ed altri che ormai non vengono più utilizzati (come Windows Phone o BlackBerry), sul mercato attualmente troviamo due alternative valide: iOS e Android.

2.1.1 iOS

I dispositivi Apple sono entrati in possesso di una buona fetta di mercato ed utilizzano il sistema operativo Apple iOS.

Esso è utilizzato solo da questa azienda ed è quindi ottimizzato al meglio per funzionare su tali dispositivi. iOS garantisce ottime prestazioni e viene spesso preso come punto di riferimento da altri sistemi operativi mobili.

L'architettura di questo sistema è di tipo stratificato, dove troviamo una serie di strati uno sopra l'altro, ognuno di essi implementa una determinata funzione. Ogni strato superiore beneficia dei servizi dello strato sottostante senza conoscerne la logica.

Per creare una applicazione per i dispositivi Apple è necessario avere a disposizione un Mac ed installare l'ambiente di sviluppo Xcode. Per quanto riguarda il linguaggio di programmazione attualmente viene utilizzato Swift, un linguaggio di programmazione orientato agli oggetti per sistemi macOS, iOS, watchOS, tvOS e Linux.

Sviluppare app per iOS è, quindi, molto costoso e richiede non solo un'attrezzatura adatta ma anche un abbonamento ad Apple Developer Program pagando un canone annuale. (Apple Inc., “Apple”, s.d.)

2.1.2 Android

Android è un sistema operativo per dispositivi mobile sviluppato da Google. Esso è un progetto OpenSource che ha come obiettivo migliorare l'esperienza della telefonia quanto più possibile.

A tal proposito il design di Android si basa su tre principi fondamentali “*Enchant me*” “*Simply my life*” e “*Make me amazing*”. La gestione “libera” del progetto ha portato Android ad essere il sistema operativo con maggiore crescita negli ultimi anni.

La piattaforma Android è ben strutturata e viene suddivisa in 5 livelli come possiamo vedere nella Figura 1.

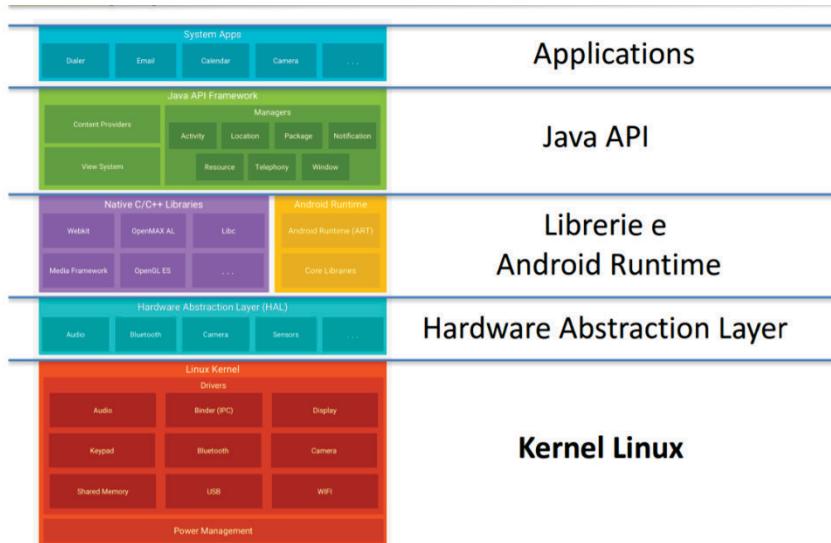


Figura 1 Architettura Android (Prisco, 2021)

Il kernel utilizzato è un kernel Linux esso rappresenta il cuore del sistema operativo e costituisce il livello di astrazione di tutto l'hardware sottostante.

I programmatore possono, quindi, intervenire già a questo livello per personalizzare i driver di comunicazione con i propri dispositivi. Attraverso l'astrazione riusciamo ad ottenere un'ottima separazione fra i livelli che ci permette di avere una *user experience* indipendente dal dispositivo e una programmazione ad alto livello omogenea.

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 Blink 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.

Al penultimo livello è possibile rintracciare i gestori e le applicazioni di base del sistema. 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. (Fuochi F. S., 2013)

2.1.3 Considerazioni

Le ragioni che hanno portato a concentrare questo lavoro di tesi sul sistema Android sono molteplici. In primis la sua enorme diffusione a livello mondiale, recenti studi svolti dalla IDC hanno rilevato che il market share controllato da Android è ora al 83,8% e mantiene previsioni alte anche per gli anni successivi come mostrato in Figura 2. (NEEDHAM, Worldwide Smartphone Shipment OS Market Share Forecast, 2021)

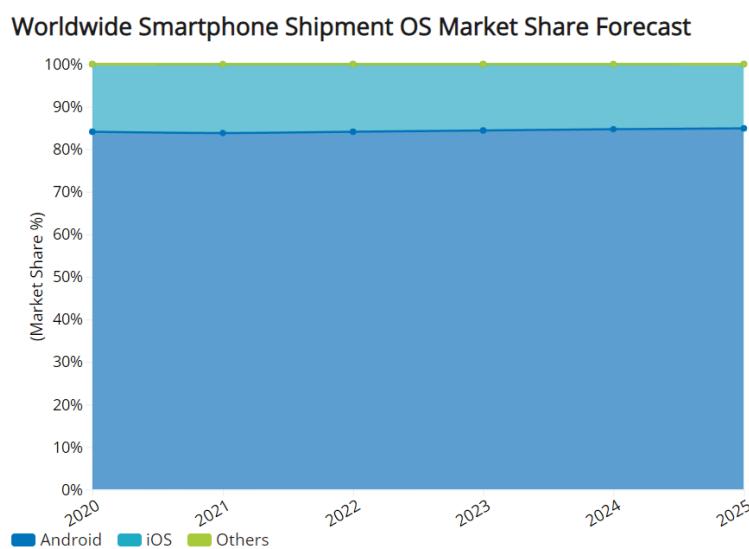


Figura 2 Previsione della quota di mercato del sistema operativo per la spedizione di smartphone in tutto il mondo (NEEDHAM, Worldwide Smartphone Shipment OS Market Share Forecast, 2021)

La seconda ragione che ha spinto questa scelta è la licenza Apache che rende Android completamente open-source, oltre alle innumerevoli API Android facilmente reperibili e le ottime guide e tutorial online che agevolano la programmazione nel mondo Android.

Importante fattore da considerare, sono i costi contenuti rispetto alla concorrenza (ad esempio Apple) che danno la possibilità a chiunque di entrare nel mondo della programmazione e di confrontarsi con esso. Android è un ambiente libero e aperto a tutti e grazie alla solidità dell'azienda Google il sistema è indubbiamente stabile e in continuo sviluppo.

2.2 Editing del testo predefinita in Android

Data la larga diffusione, i dispositivi mobile vengono utilizzati per eseguire un'ampia gamma di operazioni: navigazione web, chat, lettura di documenti, ecc...

Con la proliferazione dei touchscreen, la ricerca si è concentrata sul miglioramento dell'interazione dell'utente con essi e, in particolare, sul poter garantire un'usabilità quanto più fluida possibile.

La modifica del testo su questi dispositivi è attualmente eseguita tramite una serie di Widget. Tipicamente, l'utente inserisce il testo utilizzando una tastiera virtuale e sposta il cursore semplicemente toccando con il dito il punto desiderato nel testo.

Oltre a inserire un nuovo carattere, l'utente può effettuare la selezione tenendo premuto sul testo e trascinando fino a raggiungere il punto desiderato. Altre operazioni di modifica possono essere effettuate utilizzando i widget che appaiono in un menu sul testo dopo un'interazione dell'utente (ad esempio un tocco lungo sullo schermo).

2.3 Svantaggi

A partire dalle prime versioni di Android, fino ad arrivare ad oggi, Android ha sempre cercato di migliorare la fluidità del touchscreen e ha continuato negli anni la ricerca di nuove soluzioni per migliorare l'usabilità. Nonostante questo, le piccole dimensioni dello schermo degli smartphone restano un problema per le operazioni che richiedono una certa precisione.

Parlando di testo e della modifica di esso, uno degli svantaggi della tecnica predefinita di Android è la difficoltà nel posizionare le dita nel punto esatto del testo. La dimensione delle dita copre quasi sempre il testo, soprattutto quando il font dei caratteri è molto piccolo.

Spesso utilizziamo il cellulare in momenti in cui non possiamo dedicare una piena attenzione alle operazioni da svolgere, questo comporta un numero crescente di errori. La tecnica predefinita di Android non ci dà la possibilità di recuperare le vecchie modifiche aumentando, in questo modo, il tempo richiesto per completare

l'operazione e scoraggiando l'utente ad utilizzare lo smartphone per le operazioni di editing del testo.

2.4 Lavori Correlati

Prima di sviluppare una nuova tecnica è stato utile analizzare altre soluzioni che hanno cercato di risolvere il problema. In letteratura molti lavori si concentrano sul processo di editing testuali su dispositivi mobili di varia natura.

Diversi studi mostrano come, in presenza di distrazioni, l'uso dei gesti per impartire comandi migliori le prestazioni rispetto all'uso di pulsanti ma aumentano gli errori da parte degli utenti, come mostrato da Bragdon et al. (A. Bragdon, 2011)

Alcuni dei lavori si concentrano solo su una singola funzione di editing, altri invece si concentrano sulle operazioni di copia/incolla che non sono facili da eseguire su smartphone. Una tecnica di questo tipo è BezelCopy. (C. Chen, 2014)

Scheibel et al. (J.-B. Scheibel, 2013) si concentrano sul problema del puntamento preciso sui touch screen, proponendo una tecnica che mira a facilitare il posizionamento del cursore quando la dimensione del carattere è piccola attraverso l'uso di un virtual stick.

Di seguito verranno mostrate due delle tecniche che hanno suscitato maggiore interesse e che sono state fonte di ispirazione per SimplyText.

2.4.1 TouchTap

Una prima ricerca approfondita (Fuccella, 2017) presenta la progettazione e la valutazione di una tecnica di interazione basata sui gesti per l'editing testuale. La tecnica proposta da V.Fuccella e B.Martin prevede sia tocchi sul touchscreen che gesti da disegnare sulla parte superiore della tastiera software per eseguire azioni di base.

La tecnica proposta, chiamata TouchTap, risulta essere indiretta in quanto tutte le azioni che hanno effetto in un campo testuale possono essere eseguite dalla tastiera, differenziandosi così dalle tecniche del passato che prevedevano anche comandi diretti. Per eseguire qualsiasi comando, l'utente ha bisogno di due puntatori:

- Un puntatore primario, necessario per attivare la modalità di modifica;
- Un puntatore secondario che esegue il comando attraverso i gesti.

Ad ogni gesto, quindi, corrisponde un'operazione di modifica. (*Vedi Tabella 1*)

I risultati ottenuti hanno mostrato un aumento delle prestazioni per piccoli font, tuttavia, alcuni partecipanti allo studio hanno avuto problemi a ricordare ed eseguire correttamente i gesti.

Tabella 1 Mappatura fra gesti e operazioni di editing (Fuccella, 2017)

Name	Category	Actions	Shape
Tap	CP/Sel	Moves the caret/selection endpoint one char left or right	•
Left	Sel	Moves the selection endpoint one word left	←
Right	Sel	Moves the selection endpoint one word right	→
Up	Sel	Moves the selection endpoint one row up	↑
Down	Sel	Moves the selection endpoint one row down	↓
Reset	Sel	Reset the selection	Ƨ
Copy	CC	Copies the selected text to the clipboard	ʌ
Cut	CC	Cuts the selected text and copies it to the clipboard	↘
Paste	CC	Pastes the text from the clipboard	∨

2.4.2 Tecnica proposta da S.Finelli et al.

L'obiettivo di questo studio (S.Finelli, 2018) consiste nell'eseguire attività di editing in maniera più rapida introducendo, un nuovo sistema di selezione e menu contestuale.

La fondamentale differenza fra questa tecnica e quella descritta nella sezione precedente sta nel fatto che in questo approccio una continua pressione sulla tastiera consente di accedere alla modalità di editing, la quale permette la selezione del testo con lo scopo di tagliare o copiare. Se viene selezionato il testo in questa modalità, un menu contestuale consente all'utente di eseguire le azioni fondamentali.

Un tocco sul testo in modalità di editing, inoltre, può essere utilizzato per incollare il testo.

I risultati mostrano che nonostante tutti i partecipanti non abbiano avuto problemi a completare i task, i dati ottenuti dall'esperimento hanno mostrato una velocità maggiore con il classico approccio.

CAPITOLO 3

PROGETTAZIONE DI SIMPLYTEXT

In questo capitolo illustreremo una possibile soluzione ai problemi sollevati nel capitolo precedente. Verranno mostrate le idee alla base di queste scelte e le motivazioni che hanno spinto ad implementare questa nuova tecnica e a testarla.

3.1 Soluzione al posizionamento nel testo

L'obiettivo della ricerca è quello di realizzare un metodo per posizionare il cursore all'interno del testo senza coprire con le dita i caratteri. Per questo motivo sono state realizzate delle softkey utilizzate per spostare il cursore e per effettuare le modifiche.

Esse si compongono di quattro frecce direzionali che spostano il cursore nel punto giusto del testo e dei bottoni per copiare, incollare e tagliare.

Il funzionamento di questo insieme di bottoni è molto semplice ed è illustrato in *Figura 3*:

- L'utente clicca sul testo ed appare la griglia con i bottoni per i movimenti;
- L'utente si posiziona nel punto richiesto attraverso le frecce direzionali e cliccando sull'icona di selezione fa partire la selezione del testo;
- Se si ha la necessità di selezionare una frase o un intero paragrafo, si può cliccare nel punto di fine selezione e regolarlo tramite le frecce;

- Si possono utilizzare direttamente le frecce, se abbiamo la necessità di selezionare pochi caratteri;
- Dopo aver deciso se copiare o tagliare il testo selezionato, cliccando sull'apposita icona posso incollare il testo nel punto scelto attraverso il bottone.

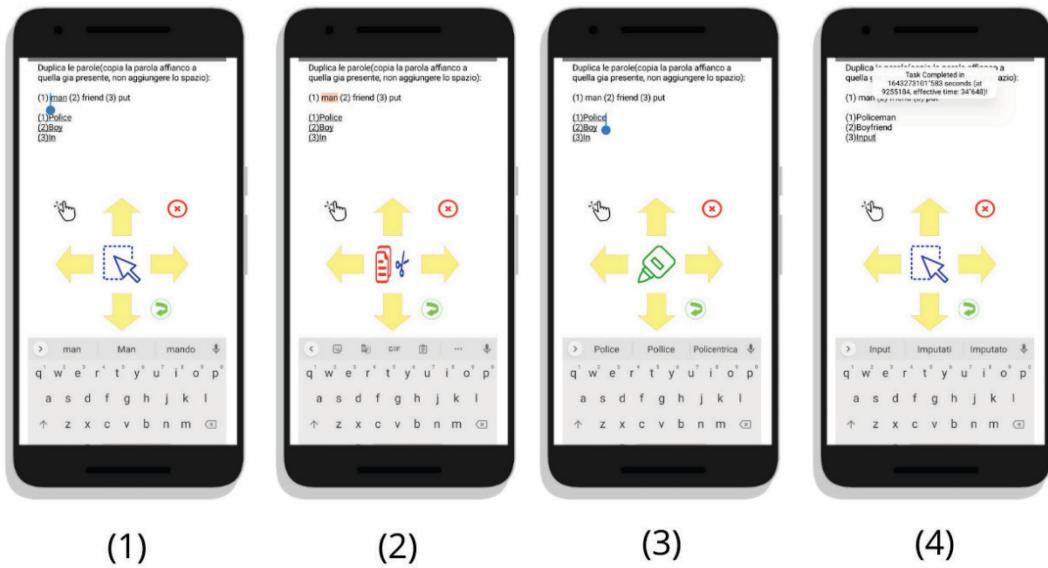


Figura 3 SimplyText step by step

La griglia compare con un tocco sul testo e può essere nascosta attraverso il bottone in alto a destra.

È presente, inoltre la possibilità di trascinare le softkey nel punto dello schermo desiderato, in modo da poter operare sul testo senza che essa ostacoli la lettura. (Vedi Figura 4)



Figura 4 Icônes Move e Ghost

3.2 Soluzione al recupero di vecchie modifiche

È sembrato opportuno aggiungere un bottone per tornare in dietro. La scelta nasce dal fatto che spesso gli utenti durante la digitazione del testo da tastiera oppure cliccando sui bottoni di modifica compiono errori oppure cambiano facilmente idea sulle operazioni da svolgere.

Per questo motivo è stato aggiunto un bottone che riporta allo stato precedente ripristinando il testo, se la modifica effettuata è errata, oppure facendo ricomparire i bottoni usati nell'azione precedente per ripetere l'operazione.

Il bottone, inoltre, ci “avvisa” se sono state fatte delle modifiche e se possono essere ripristinate attraverso un cambio di colore. Se l'icona è grigia allora non ci sono modifiche da recuperare, se il bottone è verde possiamo tornare all'azione precedente. (Vedi Figura 5)



Figura 5 Bottone backands

3.3 Valutazione euristica

La progettazione dell'interfaccia di SimplyText si basa sui principi fondamentali dell'usabilità. L'obiettivo di questa tecnica è quello di far in modo che l'utente possa raggiungere specifici obiettivi con efficacia, efficienza e soddisfazione.

Essa, infatti, si promette di soddisfare:

- **Coerenza:** Il sistema garantisce la coerenza attraverso l'uso di icone che mostrano familiarità con applicazioni già esistenti. Ogni elemento all'interno del modulo ha una terminologia identica durante tutta la sequenza di azioni.

- **Usabilità universale:** il sistema si rivolge a diverse tipologie di utilizzatori, per questo motivo è stato scelto un background tecnologico intuitivo e adatto a qualsiasi fascia d'età.
- **Offrire Feedback informativi:** ad ogni azione dell'utente corrisponde una risposta dell'interfaccia in modo che ciascun utilizzatore sappia cosa sta succedendo in ogni momento in modo chiaro, appropriato e leggibile.
- **Chiusura:** le sequenze di azioni sono organizzate con un inizio, un punto intermedio e una conclusione.
- **Prevenire gli errori:** l'interfaccia è progettata in modo tale che l'utente non sia indotto a commettere errori.
- **Reversibilità:** il sistema incoraggia l'utente a esplorare le varie opzioni dando la sensazione di poter tornare indietro senza fare danni.
- **Controllo:** il sistema garantisce agli utenti tempi di risposta ridotti e velocità di visualizzazione in modo da dare all'utilizzatore un senso di pieno controllo dell'applicazione.
- **Ridurre il Carico di Memoria:** è stato evitato l'uso di gesture complicate per ridurre quanto più possibile le informazioni che l'utente deve ricordare durante l'utilizzo del sistema.

Attraverso questa valutazione è stato possibile farsi un'idea preliminare dell'usabilità del sistema e capire se ne vale la pena procedere con l'implementazione.

CAPITOLO 4

IMPLEMENTAZIONE DI

SIMPLYTEXT

A seguito di quanto illustrato, in questo capitolo verrà approfondita la fase di implementazione illustrando gli strumenti di sviluppo e la struttura di un'applicazione Android su cui risiede il modulo di editing del testo realizzato.

4.1 Strumenti di sviluppo

Per realizzare l'applicazione è stato utilizzato **Android Studio**. Esso è un ambiente di sviluppo integrato per app Android.

Android studio ci dà la possibilità di interfacciarsi con l'SDK (Software Development Kit) di Android in modo facile ed intuitivo in quanto il kit è già presente in Android Studio durante l'installazione. In questo modo possiamo usufruire di tutte le funzionalità che il kit ci mette a disposizione, come per esempio emulatori, debugger ecc.

Importante fonte di ispirazione e di conoscenza è stato <https://developer.android.com/docs> in cui possiamo trovare la documentazione per sviluppatori di app Android.

4.2 Struttura Logica

In un'applicazione è possibile identificare quattro componenti principali: Activity, Services, Broadcast Receiver e Content Provider. Ogni componente è fondamentale per l'applicazione e diversifica il tipo di accesso che ha il sistema con essa. Ognuno di essi svolge un determinato ruolo e contribuisce a definire il comportamento dell'applicazione.

L'**Activity**, componente indispensabile per una applicazione, rappresenta la finestra contenente l'interfaccia grafica dell'applicazione e ha lo scopo di gestire l'interazione con l'utente. In un'applicazione possono esserci numerose Activity con lo scopo di trattare diverse parti del software. Viene tipicamente specificata quale fra le tante sarà presentata all'utente in fase di avvio dell'app, essa sarà l'Activity "Principale".

Da ogni activity è possibile avviarne un'altra e conservare uno stack (back stack) di tutte le Activity visitate e l'ordine. Grazie ad esso, quando l'utente preme il tasto indietro, viene riattivata l'Activity precedentemente visitata.

Ogni Activity, durante tutta l'attività dell'applicazione, segue diverse fasi che costituiscono il **ciclo di vita dell'Activity** riassunto in Figura 6.

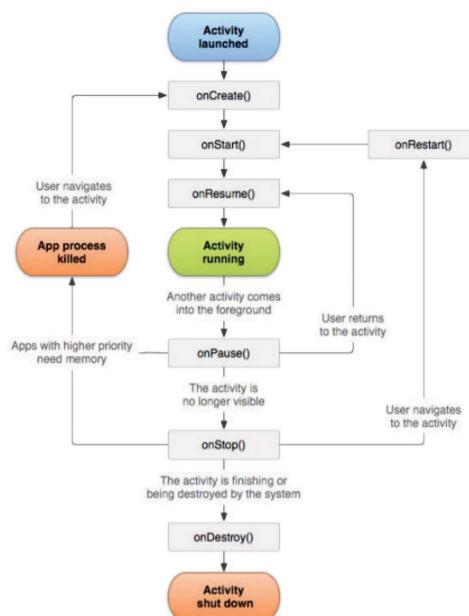


Figura 6 Ciclo di vita dell'Activity (Prisco, 2021)

L'activity può trovarsi in sei stati diversi: Created, Started, Resumed, Paused, Stopped, Destroyed.

Il primo metodo che viene eseguito quando l'Activity viene invocata è **onCreate()** e definisce gli elementi di base per il funzionamento. Successivamente, l'Activity viene resa visibile all'utente attraverso il listener **onStart()** e quando esso inizia ad interagire con l'Activity viene richiamato **onResume()**.

Quando da un'Activity si passa ad un'altra dobbiamo mettere la corrente in uno stato di pausa attraverso **onPause()** che si occuperà di salvare il contesto dell'Activity nel “back stack”.

Se essa, poi, dovesse essere richiamata, riprenderebbe la sua esecuzione attraverso **onResume()**, mentre se non fosse più visibile all'utente verrebbe chiamato il metodo **onStop()**. Il listener **onRestart()**, invece, viene utilizzato se abbiamo la necessità di riaprire l'Activity dopo **onStop()**.

In fine è presente **OnDestroy()** che permette di svolgere le ultime operazioni prima che l'applicazione venga terminata. (Google, s.d.) (AndroidGeek., s.d.)

4.3 Struttura del progetto

Il progetto è composto da varie parti:

- Il manifest file (Android Manifest.xml)
- La cartella java contenente le varie classi java
- La cartella res
- Una sezione “Gradle Scripts” contenente i file Gradle

La struttura del progetto è rappresentata in Figura 7.

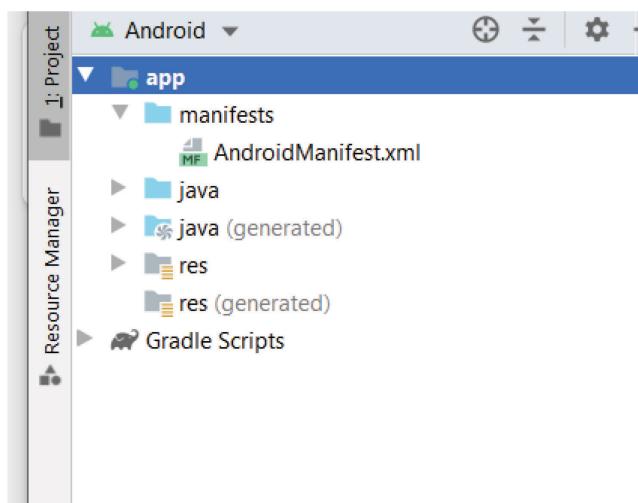


Figura 7 Struttura del progetto

4.3.1 AndroidManifest.xml

Questo file viene creato automaticamente da Android Studio durante la creazione del nuovo progetto. Questo file è fondamentale perché contiene le informazioni essenziali del progetto ed è indispensabile per la dichiarazione dei componenti, per il nome del pacchetto e per stabilire i permessi.

È proprio qui che vengono decisi i dettagli implementativi di ogni singola activity e viene definita l'icona e il nome che verranno mostrati sulla home del dispositivo.

4.3.2 Cartella res

In questa cartella troviamo tutti quegli elementi messi a disposizione dell'applicazione, questi elementi sono interni al progetto come ad esempio i file XML, immagini e altro.

Nel caso in questione abbiamo le seguenti sottocartelle: (Figura 8)

- **drawable**: contiene le immagini utilizzate nell'applicazione. Sia l'immagine usata come icona dell'app che le immagini utilizzate per le softKey.
- **layout**: è l'insieme di tutti i file XML in cui vengono definiti i componenti per costruire l'interfaccia utente. Ogni file XML definisce il layout grafico di un'Activity, cioè quello che l'utente visualizza sullo schermo.

- **values:** è la cartella contenente stringhe e attributi definite tramite l'uso dei tag XML appropriati. Questi valori vengono utilizzati all'interno dell'applicazione.
- **xml:** contiene un file XML in cui vengono definiti i vari task. Questo file contiene per ogni task il testo iniziale ed il testo desiderato come risultato per il successo del task.

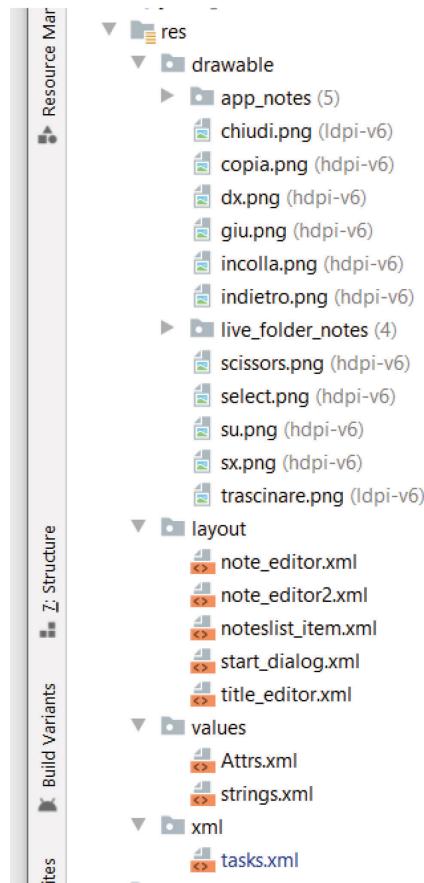


Figura 8 Cartella res

4.3.3 Gradle

In questa sezione sono contenuti gli script, necessari per automatizzare la build dell'applicazione. Essi, infatti, hanno lo scopo di scaricare le dipendenze necessarie (ad esempio le librerie), compilare il codice sorgente e fare il packaging degli eseguibili per realizzare il file .apk.

In Android Studio la sezione è chiamata Gradle Scripts e contiene due file build.gradle. Il primo è associato all'intero progetto, mentre il secondo è associato allo specifico modulo di default "app" su cui possiamo effettuare le modifiche.

4.4 Modulo SimplyText

Dopo aver scelto gli strumenti di sviluppo e illustrato la struttura di un'applicazione Android possiamo implementare la tecnica pensata e progettata nel capitolo precedente.

La tecnica di editing del testo realizzata può essere vista come un modulo contenente le componenti necessarie per svolgere le varie operazioni.

Questo modulo può essere inserito all'interno di una qualsiasi applicazione Android che abbia uno o più editText all'interno delle Activity.

L'**EditText** è un componente grafico con il quale è possibile inserire del testo e gestire gli eventi relativi al suo contenuto. Esso è il protagonista del nostro modulo perché è su questo componente che si basano tutte le nostre operazioni.

I componenti principali di SimplyText sono suddivisi in una parte grafica contenuta nel file XML dell'activity principale e da una classe contenente tutti i metodi che gestiscono i comportamenti degli elementi nel layout.

4.4.1 note_Editor.xml

Nel file XML è stato realizzato un **RelativeLayout** avente come scopo quello di raggruppare i bottoni contenuti in esso e posizionarli in modo "relativo" rispetto agli altri elementi presenti nel layout.

Per la realizzazione dei bottoni è stato usato il componente **ImageButton**. Esso permette all'utente di comunicare con l'applicazione attraverso un tocco, a differenza del comune Button questa classe ci permette di applicare come sfondo del bottone un'immagine che rende esteticamente più piacevole la grafica dell'Activity.

Di seguito è mostrato un frammento di codice di uno dei vari ImageButton presenti nel progetto.

```
<ImageButton
    android:layout_width="37dp"
    android:layout_height="74dp"
    android:id="@+id/buttonCopia"
    android:backgroundTint="#F44336"
    android:visibility="invisible"
    android:background="@drawable/copia"
    android:layout_below="@+id/buttonSu"
/>

```

Gli ImageButton sono stati posizionati in una griglia, come è possibile vedere nella *Figura 9* sottostante.

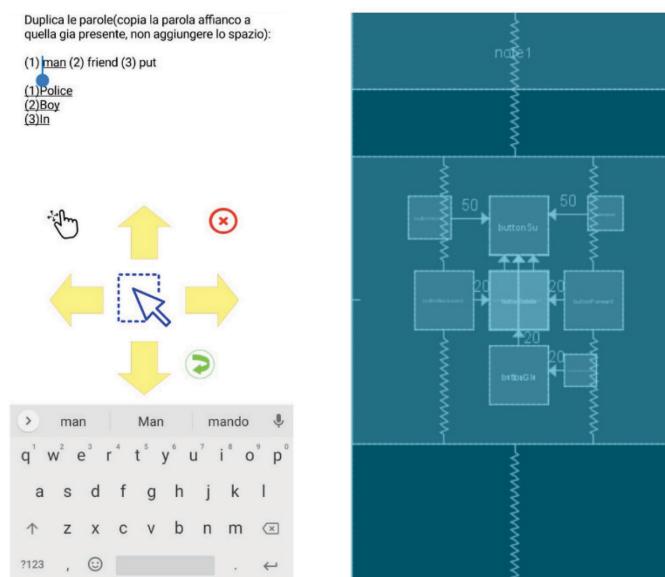


Figura 9 Layout Grafico SimplyText

4.4.2 NoteEditor.java

Nella classe NoteEditor sono stati gestiti gli eventi legati al click sui componenti. Attraverso un **Listener** (gestore di eventi) è stato possibile intercettare la pressione avvenuta sul tasto, settando il bottone nello stato OnClickListener.

Per gestire le parti statiche dell'applicazione, cioè i bottoni precedentemente mostrati, è stato necessario utilizzare la classe R. Essa ci permette di identificare in modo

dinamico tutte le risorse dichiarate via via nell'XML, in modo da poterle utilizzare all'interno della classe java.

```
buttonBackward = (ImageButton) findViewById(R.id.buttonBackward);
arrows = (RelativeLayout) findViewById(R.id.arrows);
global = (ViewGroup) findViewById(R.id.sample_main_layout);
```

Ogni bottone assume un comportamento diverso ed è stato gestito singolarmente. Esaminiamo di seguito la gestione di un singolo componente come esempio.

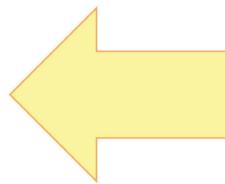


Figura 10 Button Backward

```
buttonBackward.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (mText.getSelectionStart() > 0) {
            if (buttonSelect.getVisibility() == View.INVISIBLE && flag ==
true) {
                mText.setSelection(start, mText.getSelectionEnd() - 1);
            } else {
                mText.setSelection(mText.getSelectionEnd() - 1);
            }
        } else {
            //fine stringa non posso muovere il cursore
        }
    }
});
```

Come mostrato nel codice, al tocco del bottone Backward (*Figura 10*) il cursore si sposta di una posizione a sinistra.

Per fare questo, però, bisogna rispettare una serie di condizioni. Innanzitutto, bisogna verificare che la posizione del cursore sia valida attraverso il metodo `.getSelectionStart()`. Successivamente, bisogna capire se il tocco del bottone Backward è stato effettuato prima dell'inizio della selezione o dopo. Si fa uso di una variabile booleana `flag` che, impostata a `true`, indica che è partita la selezione e al tocco del bottone non sposta solamente il cursore ma seleziona o deselectiona il carattere a

sinistra. In caso contrario, il cursore si muove semplicemente nel testo, senza selezionare alcun carattere.

È stato ritenuto importante dare la possibilità all'utente di trascinare i componenti del modulo nella parte desiderata dello schermo. Le motivazioni alla base di questa scelta sono state illustrate nel capitolo precedente e di seguito è mostrato il frammento di codice che realizza questa funzionalità.



Figura 11 ButtonMove

```
float dX, dY;  
@Override  
public boolean onTouch(View view, MotionEvent event) {  
  
    switch (event.getAction()) {  
  
        case MotionEvent.ACTION_DOWN:  
            dX = arrows.getX() - event.getRawX();  
            dY = arrows.getY() - event.getRawY();  
            break;  
  
        case MotionEvent.ACTION_MOVE:  
            arrows.animate()  
                .x(event.getRawX() + dX)  
                .y(event.getRawY() + dY)  
                .setDuration(0)  
                .start();  
            break;  
        default:  
            return false;  
    }  
    return true;  
}  
buttonMove.setOnTouchListener(this);
```

Per trascinare i componenti del modulo è necessario implementare View.OnTouchListener e sovrascrivere il metodo onTouch per gestire i comportamenti al tocco dell'elemento ButtonMove (*Figura 11*).

Questo metodo ci permette di gestire i cambiamenti avvenuti. Nel nostro caso, attraverso l'uso del costrutto switch è stato possibile intercettare il cambiamento e in caso di ACTION_DOWN (tocco dello schermo con un dito) vengono salvate le informazioni riguardanti la posizione dei componenti in due variabili float. In caso di ACTION_MOVE (movimento delle dita sullo schermo) vengono animati i componenti sommando la posizione salvata durante il tocco con la posizione assunta durante il trascinamento.

CAPITOLO 5

ESPERIMENTO

Dopo aver discusso nei capitoli precedenti della progettazione e della realizzazione di SimplyText è opportuno confrontare la tecnica appena realizzata con la tecnica classica presente sui dispositivi Android. È stato condotto, quindi, un esperimento al fine di mostrare vantaggi e svantaggi della nuova tecnica in termini di efficienza.

5.1 Partecipanti

Sono stati reclutati 7 partecipanti (4 femmine e 3 maschi) tra i 20 e i 54 anni (in media 30 anni). I partecipanti hanno accettato gratuitamente e sono stati supervisionati durante tutta la fase di test. Tutti i partecipanti hanno avuto esperienza con i dispositivi touchscreen e hanno dimostrato padronanza con l'editing di testo su smartphone.

5.2 Apparecchiatura

L'esperimento è stato effettuato su uno smartphone Xiaomi Mi 10T Pro con versione Android 11. Il dispositivo dispone di un display da 6.67 pollici con una risoluzione di 2340X1080 pixel. Il modulo è stato integrato in un'applicazione già esistente, la quale fornisce dei file di log consultabili a fine esperimento.

5.3 Procedura

Prima di iniziare l'esperimento, ai partecipanti è stata spiegata brevemente la tecnica sperimentale e sono stati mostrati i gesti consentiti per la modalità SimplyText. È stato spiegato loro il funzionamento dell'app e i singoli task da portare a termine.

Dopo questa fase introduttiva, sono stati sottoposti ad un questionario preliminare in cui sono state chieste informazioni di carattere generale come: dati anagrafici (età, sesso), manualità (destro, mancino) e livello di esperienza con smartphone.

L'esperimento consiste in otto task, in cui l'utente deve correggere il testo fornito.
(*Vedi Figura 12*)



Figura 12 Elenco dei task

Il funzionamento di ogni task è approfondito della *Tabella 1* di seguito mostrata.

Tabella 2 Approfondimento dei task (per ridurre le dimensioni della tabella, i testi delle attività vengono visualizzati solo in parte).

Task	Titolo	Descrizione	Testo iniziale	Testo finale
1	Cancella Carattere	Elimina il carattere X nel testo	<p>It was a lovely night, so warm that he threw his coat over his arm and did not even put his silk scarf round his throat. As he stroked home, smoking...</p>	<p>It was a lovely night, so warm that he threw his coat over his arm and did not even put his silk scarf round his throat. As he strolled home, smoking...</p>
2	Cancella parola		<p>It was a lovely night, so warm that he threw his coat over his arm and did not even put his silk scarf XXXXXX round his throat. As he strolled home, smoking...</p>	<p>It was a lovely night, so warm that he threw his coat over his arm and did not even put his silk scarf round his throat. As he strolled home, smoking...</p>
3	Cancella frase		<p>It was a lovely night, so warm that he threw his coat over his arm and did not even put his silk scarf round his throat. As he XXXXX XXXXX strolled home, smoking...</p>	<p>It was a lovely night, so warm that he threw his coat over his arm and did not even put his silk scarf round his throat. As he strolled home, smoking...</p>
4	Sposta parola (taglia-incolla)	Taglia la parola e spostala nella posizione indicata	<p>It was a lovely night, so warm that he threw his coat over his arm and did not even put his silk scarf round his throat. As he strolled home...</p>	<p>It was a lovely night, so warm that he threw his coat over his arm and did not put even his silk scarf round his throat. As he strolled home...</p>
5	Sposta frase	Taglia la frase e spostala nella posizione indicata	<p>...He of them heard one whisper to the other, that is Dorian Gray. He used to be when remembered how pleased he he was pointed out...</p>	<p>...He heard one of them whisper to the other, that is Dorian Gray. He remembered how pleased he used to be when he was pointed out...</p>

6	Correggi testo	Cancella e sposta le parole	<p>It was a lovely night, so warm that he threw his coat over his arm and did not even put his silk scarf round his throat. As he strolled home, smoking his cigarette, two young men in evening dress passed him. He heard one of them whisper to the other, That XXXX is Dorian Gray. He remembered how pleased he used to be when he was pointed out, or stared at, or talked about. He was tired of hearing his own name now. Half the charm of the village where he had little been so often lately was that no one knew who he was.</p>	<p>It was a lovely night, so warm that he threw his coat over his arm and did not even put his silk scarf round his throat. As he strolled home, smoking his cigarette, two young men in evening dress passed him. He heard one of them whisper to the other, That is Dorian Gray. He remembered how pleased he used to be when he was pointed out, or stared at, or talked about. He was tired of hearing his own name now. Half the charm of the little village where he had been so often lately was that no one knew who he was.</p>
7	Duplica parola (copia-incolla)	Copia la parola affianco a quella già presente senza aggiungere lo spazio	<p>(1) man (2) friend (3) put (1) Police (2) Boy (3) In</p>	<p>(1) man (2) friend (3) put (1) Policeman (2) Boyfriend (3) Input</p>
8	Duplica frase	Copia la frase fra le XXX in fondo al testo	<p>XXX It was a lovely night XXX, so warm... Half the charm of the little village where he had been so often lately was that no one knew who he was.</p>	<p>XXX It was a lovely night XXX, so warm... Half the charm of the little village where he had been so often lately was that no one knew who he was. It was a lovely night</p>

Alla fine di ogni task, all'utente viene notificato il successo del test. In caso di insuccesso, sul task compare la scritta “failed”, il partecipante può ripetere il task tenendo premuto su di esso e resettandolo. (*Figura 13*)

Al partecipante viene richiesto di ripetere i task falliti fino al completamento con successo degli stessi, verrà registrato e tenuto in considerazione il numero di volte in cui si è verificato il fallimento.

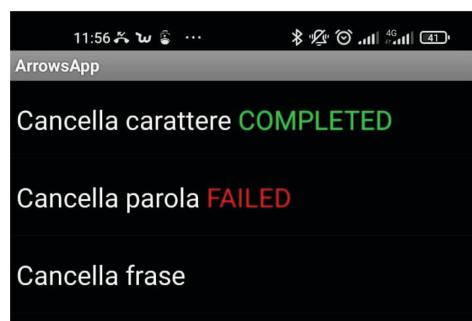


Figura 13 Task completato

L'esperimento è stato diviso in quattro fasi. La prima fase è stata di pura formazione per i partecipanti. È stato chiesto a ciascuno di essi di eseguire i task proposti dall'applicazione per prendere confidenza con l'app. In questa fase preliminare non vengono registrate le performance e i risultati, ma ci si assicura che il partecipante abbia capito come svolgere ogni task. Per l'esecuzione di questa fase sono stati utilizzati i caratteri con dimensione 3.0.

La metà degli utenti ha utilizzato prima la tecnica sperimentale e poi la tecnica classica, la restante parte ha svolto il test con ordine invertito.

Le altre tre fasi, invece, rappresentano l'esperimento vero e proprio. I task vengono eseguiti dall'utente per due volte utilizzando una diversa dimensione del carattere (piccola= 2.5, media= 3.0). L'ordine in cui i compiti sono stati eseguiti è illustrato nella seguente tabella. (*Tabella 3*)

Tabella 3 Bilanciamento dei partecipanti utilizzato durante l'esperimento. Dimensione font abbreviata (s - piccolo, m- medio)

Partecipante	Odine grandezza font		Ordine tecniche di editing
	Training	Test	
1	m	s/m	new/classic
2	m	s/m	classic/new
3	m	s/m	new/classic
4	m	s/m	classic/new
5	m	m/s	classic/new
6	m	m/s	new/classic
7	m	m/s	classic/new

A fine esperimento, è stato chiesto ai partecipanti di compilare un questionario System Usability Scale (SUS) per ciascuna delle due tecniche. Il questionario SUS si compone di 10 affermazioni a cui l'utente assegna un punteggio da 1(fortemente in disaccordo) a 5 (fortemente in accordo). Il punteggio finale più alto indica una maggiore usabilità da parte dell'utente. Inoltre, dopo l'esperimento, sono stati raccolti commenti e suggerimenti in forma libera in merito alla tecnica. È stato infine chiesto un parere sulla tecnica sperimentale e quale fra le due è stata più gradita.

5.4 Risultati e discussione

Tutti i partecipanti hanno completato l'esperimento con successo e hanno impiegato circa mezz'ora per completare i test e i questionari. I tempi complessivi di completamento dell'attività sono stati raggruppati in base alla dimensione del carattere. (*Figura 14*)

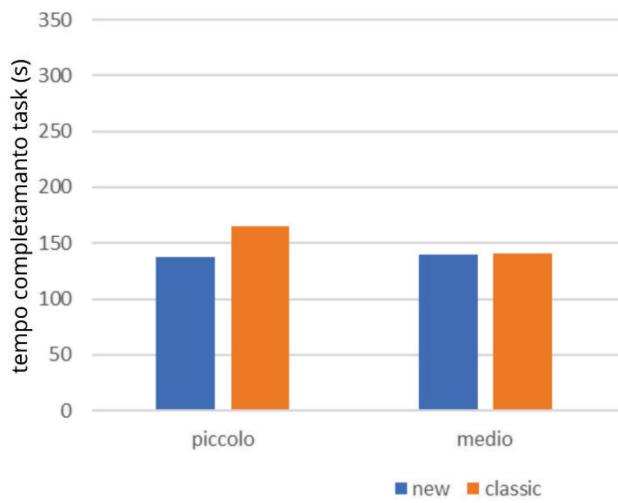


Figura 14 Tempo medio di completamento dell'attività per le due tecniche di editing

Come si evince dal grafico, con caratteri di dimensione 2.5 (piccolo) la tecnica sperimentale è risultata più veloce per gli utenti (126 vs 162"). Per quanto riguarda i caratteri di dimensione 3.0 (medio) risulta quasi impercettibile la differenza (138" vs 140") .

Ciò è dovuto al fatto che la tecnica predefinita in Android richiede estrema precisione, soprattutto per la selezione del testo e per il posizionamento del cursore. Con il font di piccole dimensioni questa precisione richiede tempo e concentrazione. A livello globale, il metodo proposto è più veloce (131" vs 151")

I tempi medi di completamento per ciascun task sono mostrati in *Figura 15*.

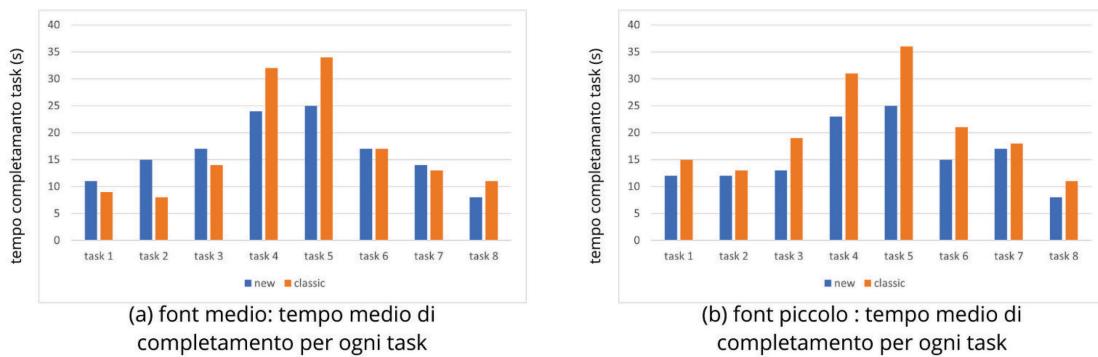


Figura 15 Tempo di completamento dei task

I dati sono stati suddivisi in 2 grafici; uno per il font 2.5 (a) e l'altro per il font 3.0 (b).

In merito ai test svolti con il font 2.5, risultata più veloce la nuova tecnica nei confronti di quella classica per tutti i task.

Per quanto riguarda i primi tre task, che hanno l'obiettivo di cancellare frasi, parole o singoli caratteri, la tecnica tradizionale per il font 3.0 risulta più veloce di quella innovativa. Diversi, sono i risultati per quanto riguarda i task che includono operazioni di copia, taglia e incolla. Dal grafico spicca in modo particolare che i task 4 e 5 sono stati completati in meno tempo con la nuova tecnica.

Infine, si è scoperto che i partecipanti hanno fallito meno compiti con la tecnica SimplyText rispetto alla tecnica tradizionale grazie alla possibilità di ritornare all'azione precedente per recuperare i dati persi durante l'errore. In particolare, si è verificato un solo task fallito per la tecnica sperimentale e sei task falliti per la tecnica tradizionale.

Il test è stato eseguito da partecipanti di diverse fasce d'età, il tempo impiegato dai partecipanti con età superiore ai 50 anni è simile a quella degli altri partecipanti.

I risultati hanno dimostrato che la nuova tecnica può essere utilizzata facilmente a prescindere dal soggetto poiché risulta essere intuitiva e lineare.

5.5 Soddisfazione degli utenti e commenti

Il punteggio medio SUS riscontrato durante il test è 70.5 per la tecnica di editing classico e 79.2 per la nuova tecnica di editing. Di fronte alla scelta della tecnica preferita fra le due, quasi tutti gli utenti hanno scelto la nuova tecnica evidenziando forte interesse in particolare per i caratteri di piccole dimensioni. I problemi riscontrati, invece, sono quasi tutti legati alle dimensioni delle softKey. Alcuni utenti hanno suggerito di ridurre le dimensioni dei bottoni o di farli sparire automaticamente quando non vengono utilizzati.

CAPITOLO 6

CONCLUSIONI

In questa tesi è stata presentata una nuova tecnica di modifica del testo basata sull'utilizzo di softkey, chiamata SimplyText, che consente all'utente di eseguire operazioni come copia, taglia, incolla in modo più efficiente.

La tecnica è stata progettata tenendo conto dei problemi rilevati durante una serie di indagini sullo stato dell'arte ed è stata sottoposta ad un'analisi euristica per verificarne la fattibilità. È stato, inoltre, condotto uno studio sugli utenti per confrontare la tecnica proposta con quella classica.

I risultati di tale studio mostrano che la nuova tecnica supera sempre quella tradizionale in termini di velocità quando il font è piccolo, mentre con font più grande nel caso di taglia-incolla. Il feedback su SimplyText è positivo ed i partecipanti hanno dimostrato forte interesse e sono risultati molto collaborativi.

In definitiva, i contributi principali offerti da questo lavoro di tesi sono: un tool eseguibile, che fornisce migliori prestazioni durante l'editing del testo; Una descrizione dettagliata del lavoro svolto, che supporta i futuri fruitori del sistema ed eventuali sviluppatori interessati a modificare e migliorare la tecnica; Un approfondimento riguardo approcci innovativi all'editing del testo.

I risultati ottenuti attraverso questo lavoro posso essere utilizzati come base per ulteriori sviluppi. I lavori futuro includono un miglioramento del design grafico in modo da rendere il componente più piacevole alla vista e meno ingombrante. Potrà essere implementata e valutata anche la proposta dei partecipanti di integrare una

funzionalità per la cancellazione, in modo da ridurre quanto più possibile anche l'utilizzo della tastiera.

Verranno inoltre svolti ulteriori studi con un maggior numero di partecipanti e con dispositivi diversi al fine di mitigare le minacce alla validità dello studio già svolto.

Riferimenti bibliografici

- [1] Apple Inc., "Apple". (s.d.). Tratto da <http://apple.com>
- [2] A. Bragdon, E. N. (2011). *Experimental analysis of touch-screen gesture designs in mobile environments*. Tratto da Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '11, ACM, New York, NY, USA, 978-1-4503-0228-9: <http://dx.doi.org/10.1145/1978942.1979000>.
- [3] AGI. (2019, Dicembre 18). *Solo l'8% degli italiani non usa lo smartphone per andare su Internet*. Tratto da Agenzia Giornalistica italiana: https://www.agi.it/economia/istat_smartphone-6761751/news/2019-12-18/
- [4] AndroidGeek. (s.d.). *Activity*. Tratto da <http://www.androidgeek.it/tutorials/>.
- [5] Binetti, F. (2021, giugno 12). *Uso smartphone statistiche audience*. Tratto da Bitmusic.it: <https://www.bintmusic.it/uso-smartphone-statistiche-audience/>
- [6] C. Chen, S. P. (2014, Maggio). *W.T. Ooi, Bezelcopy: an efficient cross-application copy-paste technique for touchscreen smartphones*. (Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces, AVI '14, ACM New York, NY, USA, 978-1-4503-2775-6, 2014, pp. 185–192) Tratto da <http://dx.doi.org/10.1145/2598153.2598162>
- [7] Fuccella, V. &. (2017, Settembre). *B. TouchTap: A gestural technique to edit text on multi-touch capable mobile devices*. In *Proceedings of the 12th Biannual Conference on Italian SIGCHI Chapter* (pp. 1-6).
- [8] Fuochi, F. (2013). *Sviluppo di applicazioni Android per promozione e gestione degli ordini in ambito ristorazione*. Tratto da <https://core.ac.uk/download/pdf/31145145.pdf>
- [9] Google. (s.d.). *Activities*. Tratto da <http://developer.android.com/guide/components/>
- [10] J.-B. Scheibel, C. P. (2013). *Virtual stick in caret positioning on touch screens*. (Proceedings of the 25th Conference on L'Interaction Homme-Machine, IHM '13, ACM, New York, NY, USA, 2013, pp.107:107–107:114) Tratto da <http://dx.doi.org/10.1145/2534903.2534918>.
- [11] NEEDHAM, M. (2021, Ottobre 28). *Worldwide Smartphone Shipment OS Market Share Forecast*. Tratto da IDC: <https://www.idc.com/promo/smartphone-market-share>
- [12] Prisco, R. d. (2021). *MP.* Tratto da robdep: <https://intranet.di.unisa.it/~robdep/MP/slides/slides2021.pdf>

-
- [13] S.Finelli, N. (2018). An experimental approach for text editing with software keyboard in Androird devices.