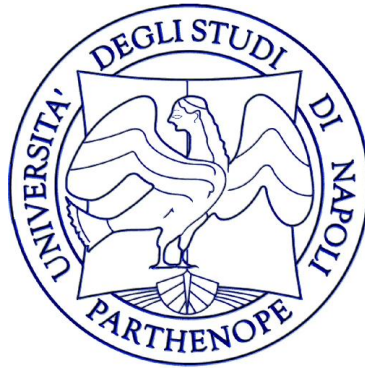


UNIVERSITÀ DEGLI STUDI DI NAPOLI PARTHENOPE

DIPARTIMENTO DI SCIENZE E TECNOLOGIE



CORSO DI LAUREA IN INFORMATICA

TESI DI LAUREA

ICT-SUPPORTED COGNITIVE TRAINING FOR
CHILDREN WITH COGNITIVE PATHOLOGIES

Relatori

Prof.ssa Mariacarla Staffa

Candidato

Riccardo Andrea Spinosa

Matr. 0124002253

ANNO ACCADEMICO 2024/2025

Contents

List of Tables	5
1 Introduzione	7
1.1 Contesto e motivazione	7
1.2 Obiettivi della tesi	8
2 Contesto applicativo	9
2.1 Disturbi dello spettro autistico e tecnologie assistive	9
2.2 Il robot Pepper: caratteristiche e potenzialità	10
2.3 I test cognitivi scelti: TROG, Cloze, Sally-Anne	11
3 Raccolta delle esigenze utente	12
3.1 Colloquio con Psicologa	12
3.2 Colloquio con Clinica Specializzata	13
4 Sistema proposto	15
4.1 Panoramica	15
4.2 Analisi dei requisiti	16
4.3 Architettura generale del sistema	17
4.4 UX per medico e bambino	18
5 Modelli di sistema	19
5.1 Scenari	20
5.2 Casi d'uso	25

6 Data persistent management	31
7 Principi SOLID	34
8 Design Pattern	36
9 Gestione degli Errori	44
10 Parti rilevanti del codice	46
10.1 playAnimation	46
10.2 riconoscereParole	47
10.3 soundAttention	48
10.4 speak	49
10.5 incoraggiaUtente	50
10.6 gestisciTocco	51
10.7 change	52
10.8 configuraGrafico	54
11 Bozza Interfaccia	55
12 Mock up	56
13 Animazioni del robot Pepper	67
14 Risultati e Valutazione	70
14.1 Valutazioni	70
15 Obiettivi raggiunti	73
16 Verifica funzionale	74
17 Feedback qualitativo	75
18 Limiti e miglioramenti futuri	76

19	Metodologia di Testing	78
20	Test di Performance	81
21	Analisi di Mercato (3.0)	82

List of Tables

5.1	Scenario di successo per la somministrazione del test T.R.O.G.	20
5.2	Scenario di annotazione di risposta da parte del medico . . .	21
5.3	Scenario di visualizzazione dei risultati grafici	22
5.4	Scenario alternativo: il bambino non risponde e Pepper attira l'attenzione	23
5.5	Scenario alternativo: il bambino si agita dopo un errore e Pepper lo tranquillizza	24
5.6	Caso d'Uso – Somministrazione T.R.O.G.	25
5.7	Caso d'Uso – Somministrazione “Sally e Annie”	26
5.8	Caso d'Uso – Somministrazione Cloze Test	27
5.9	Caso d'Uso – Visualizzazione Risultati	28
5.10	Caso d'Uso – Aggiunta Nuovo Paziente	29
5.11	Caso d'Uso – Storico Test Paziente	30

Bibliography

- [1] Android Developers. Build your first app – get started with android. <https://developer.android.com/get-started/overview>. [Ultimo accesso: giugno 2025]. 7
- [2] SoftBank Robotics. Qi sdk for android – pepper sdk documentation. <https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/index.html>. [Ultimo accesso: giugno 2025]. 7

Chapter 1

Introduzione

In questo lavoro, la progettazione e lo sviluppo dell'applicazione sono stati supportati dalle linee guida ufficiali fornite da Android Developers [1] e dalla documentazione tecnica del SoftBank Robotics Pepper SDK [2]. Questi riferimenti sono stati fondamentali per comprendere le API e implementare le funzionalità richieste.

1.1 Contesto e motivazione

Negli ultimi anni si è assistito a una crescente integrazione delle tecnologie digitali in ambito educativo e terapeutico, con particolare attenzione verso i disturbi del neurosviluppo. Tra questi, il disturbo dello spettro autistico (ASD) rappresenta una condizione che impatta significativamente sulla capacità di comunicare, interagire e apprendere.

Numerosi studi hanno evidenziato come l'impiego di strumenti tecnologici possa facilitare l'apprendimento e migliorare l'interazione dei bambini con ASD. In particolare, l'utilizzo di robot sociali rappresenta una frontiera promettente: questi dispositivi, grazie alla loro prevedibilità e interattività controllata, riescono a coinvolgere i bambini in modo efficace e non invasivo.

All'interno di questo contesto si inserisce il presente lavoro, che prevede lo sviluppo di un'applicazione Android integrata con Pepper, un robot umanoide progettato per l'interazione sociale. L'obiettivo è creare uno strumento che consenta l'erogazione di test cognitivi in forma interattiva, supportando le attività di diagnosi e monitoraggio delle competenze cognitive nei bambini con ASD.

1.2 Obiettivi della tesi

Scopo principale di questa tesi è la realizzazione di un'applicazione mobile in grado di interfacciarsi con il robot Pepper per somministrare test cognitivi validati, registrare le risposte dei bambini e assistere il medico nel processo di analisi.

Gli obiettivi specifici sono:

- Sviluppare un'interfaccia per il medico per la gestione dei pazienti e dei test;
- Integrare l'interazione vocale e visiva di Pepper nella somministrazione dei test;
- Implementare tre test cognitivi: TROG, Cloze Test e Sally-Anne;
- Raccogliere, registrare e visualizzare i risultati in modo utile per il personale sanitario;
- Gestire l'attenzione e la motivazione del bambino durante la somministrazione dei test.

Chapter 2

Contesto applicativo

2.1 Disturbi dello spettro autistico e tecnologie assistive

I disturbi dello spettro autistico (ASD) sono una serie di disordini neuroevolutivi caratterizzati principalmente da difficoltà nelle aree della comunicazione sociale e da comportamenti ripetitivi o ristretti. Le manifestazioni dei disturbi autistici variano significativamente da individuo a individuo, tanto che i sintomi e le problematiche associate possono apparire a livelli differenti di gravità.

Una delle difficoltà principali che i bambini con ASD affrontano riguarda la gestione delle interazioni sociali. Mentre alcuni bambini autistici possono avere difficoltà nel comprendere le emozioni altrui o nell'impegnarsi in giochi di gruppo, altri possono mostrare difficoltà nell'acquisire competenze comunicative verbali o non verbali.

Nonostante queste difficoltà, molti studi suggeriscono che i bambini con ASD possano beneficiare notevolmente dell'uso di tecnologie assistive, che offrono un ambiente di apprendimento personalizzato, meno stressante e facilmente controllabile.

Le tecnologie assistive comprendono una vasta gamma di dispositivi

progettati per aiutare le persone con disabilità a svolgere attività quotidiane o a migliorare le loro capacità cognitive. In particolare, nel campo dell'autismo, i robot sociali sono stati utilizzati per migliorare le competenze sociali e cognitive dei bambini, grazie alla loro capacità di interagire in modo prevedibile e rassicurante. Tra questi, il robot Pepper può diventare un importante strumento per il supporto e la valutazione dei bambini con ASD.

2.2 Il robot Pepper: caratteristiche e potenzialità

Pepper è un robot umanoide sviluppato da SoftBank Robotics. La sua caratteristica principale è quella di essere un robot progettato per interagire con gli esseri umani in modo naturale, con capacità di percepire emozioni, rispondere a comandi vocali e compiere movimenti articolati. Pepper è dotato di numerosi sensori, tra cui microfoni, fotocamere, un display touchscreen e un sistema di motori che gli permettono di muoversi e interagire fisicamente con le persone.

Le principali caratteristiche di Pepper che lo rendono ideale per il contesto educativo e terapeutico nei bambini con ASD sono:

- **Interazione sociale:** grazie alla sua capacità di rilevare l'umore e rispondere in modo adattivo, Pepper può stimolare l'interazione sociale nei bambini, motivandoli a rispondere a domande o a compiere azioni;
- **Feedback immediato:** Pepper è in grado di fornire un feedback immediato, sia verbale che visivo, utile per rinforzare comportamenti positivi e correggere eventuali errori;
- **Ripetibilità:** le interazioni con il robot sono ripetibili senza che il

robot mostri segni di stanchezza o disinteresse;

- **Personalizzazione:** il comportamento di Pepper può essere adattato alle esigenze specifiche di ciascun bambino.

Queste caratteristiche rendono Pepper uno strumento unico per l'apprendimento e la valutazione nei bambini con ASD.

2.3 I test cognitivi scelti: TROG, Cloze, Sally-Anne

Per valutare le capacità cognitive e sociali dei bambini, sono stati scelti tre test standardizzati:

- **TROG** (Test for Reception of Grammar): misura la comprensione grammaticale del bambino attraverso la scelta di immagini corrispondenti a frasi udite;
- **Cloze Test:** test di completamento delle frasi per valutare la comprensione del linguaggio e le capacità di ragionamento linguistico;
- **Sally-Anne Test** (Theory of Mind): test basato su false credenze per valutare la comprensione degli stati mentali altrui.

Questi test permettono una valutazione completa e diversificata delle competenze cognitive nei bambini con ASD.

Chapter 3

Raccolta delle esigenze utente

3.1 Colloquio con Psicologa

Durante la fase di progettazione dell'interfaccia utente, sono state considerate con attenzione le esigenze di bambini nello spettro autistico. A tal fine è stato condotto un colloquio con una psicologa. Da questo confronto sono emerse le seguenti indicazioni progettuali:

- **Incentivazione tramite suono:** se il bambino non risponde entro 8 secondi dalla richiesta, è stato suggerito di riprodurre un suono gradevole per attirarne l'attenzione prima di ripetere la richiesta.
- **Correlazione semantica delle immagini:** le immagini mostrate all'interno dell'applicazione dovrebbero essere logicamente correlate tra loro, per facilitare la comprensione da parte del bambino.
- **Uso di immagini cartonesche e colorate:** le immagini devono essere realizzate in stile cartonesco, con colori accesi e ben distinguibili, al fine di stimolare l'attenzione e la comprensione del bambino.
- **Reset dell'attenzione:** in caso di risposta troppo rapida o automatica da parte del bambino, è stato consigliato di inserire un'azione per

ristabilire l'attenzione, ad esempio un comando per alzare le braccia seguito da una domanda come *"Sei pronto per ricominciare?"*.

- **Gestione della risposta corretta:** vi è stata una proposta di mostrare la risposta corretta al bambino dopo un errore, per favorire l'apprendimento.

Pareri discordanti: alcuni professionisti consultati hanno espresso opinioni differenti rispetto a quanto sopra riportato:

- Per l'incentivazione sonora, è stato suggerito da alcuni l'uso di un suono neutro invece che gradevole, accompagnato eventualmente da una frase del tipo: *"Ti sei distratto? Perché non pensi al test?"*.
- Per il reset dell'attenzione, sono emerse opinioni divergenti sulla formula verbale da utilizzare.
- Mostrare la risposta corretta è stato sconsigliato da alcuni, poiché potrebbe influenzare negativamente l'efficacia del test in caso di ripetizione.

Tuttavia, per garantire coerenza progettuale e semplificare l'esperienza utente, i pareri discordanti sono stati scartati a priori.

I nomi e i dati identificativi dei professionisti coinvolti sono stati volontariamente omessi nel rispetto delle normative vigenti sulla protezione dei dati personali.

3.2 Colloquio con Clinica Specializzata

Durante un incontro con un neuropsicologo operante in una clinica specializzata nello spettro autistico, sono state raccolte informazioni fondamentali sui test cognitivi attualmente in uso. I due principali strumenti segnalati sono:

- **T.R.O.G. (Test for Reception of Grammar):** test utilizzato con bambini tra i 5 e gli 11 anni. Composto da 18 o 80 domande, ciascuna accompagnata da 4 immagini tra cui il bambino deve selezionare quella corretta. Il test consente la valutazione delle capacità di comprensione grammaticale e discriminazione visiva.
- **Torre di Londra:** test che misura la capacità di pianificazione e problem solving. Non sono stati forniti dettagli tecnici aggiuntivi durante l'incontro.

Anche in questa sezione, i nomi e i dati personali sono stati esclusi per motivi di riservatezza.

Chapter 4

Sistema proposto

4.1 Panoramica

I test si dividono in due parti:

La parte del test sul lessico consiste nel mostrare immagini di oggetti o situazioni comuni e chiedere al bambino di selezionare l'immagine corrispondente a una parola specifica detta dall'osservatore (in questo caso Pepper). Ad esempio, Pepper potrebbe dire "mostrami l'immagine del cane" e il bambino dovrebbe quindi selezionare l'immagine del cane tra un insieme di immagini.

La parte del test sulla grammatica, invece, potrebbe coinvolgere l'uso di immagini per presentare situazioni o azioni che richiedono l'uso corretto della grammatica. Ad esempio, Pepper potrebbe mostrare un'immagine di una persona che cammina e chiedere al bambino di completare la frase dicendo "Il ragazzo ___ per la strada". Il bambino dovrebbe quindi scegliere la parola corretta (*cammina*) per completare la frase in modo grammaticalmente corretto.

Obiettivo:

Realizzazione di un'interfaccia semplice ed intuitiva che somministra test specifici ai bambini che soffrono del disturbo dello spettro autistico.

4.2 Analisi dei requisiti

Requisiti funzionali

- L'applicazione deve presentare al bambino una serie di figure e richiedere di toccare la figura corrispondente in base alle istruzioni fornite dal robot Pepper.
- L'applicazione permette la scelta del test da effettuare prima di avviare l'applicazione.
- L'applicazione deve registrare le risposte date dal bambino e salvarle per l'analisi successiva.
- L'applicazione deve leggere le istruzioni in modo chiaro.
- L'applicazione deve verificare se il bambino risponde entro un determinato periodo di tempo, e in caso contrario deve catturare l'attenzione del bambino mediante un suono apposito.
- L'applicazione deve motivare il bambino o invitarlo a calmarsi se risponde male.
- Il medico deve poter aggiungere i dati relativi ai pazienti, come nome, età, diagnosi e note cliniche.
- L'applicazione deve tenere traccia delle risposte corrette date dal bambino e fornire un punteggio finale.
- L'applicazione deve permettere al medico di selezionare il test da somministrare al bambino (TROG, Cloze, Sally-Anne) e configurare il test in base al bambino (ad esempio, durata del test, tipo di domande).
- Durante la somministrazione del test, il robot Pepper deve interagire con il bambino, eseguendo azioni fisiche e verbali.

- Dopo la somministrazione di un test, l'app deve generare una reportistica per il medico, con una valutazione dei risultati, grafici di performance, risposte corrette ed errate.
- I dati raccolti devono rispettare la privacy attenendosi alle norme locali in vigore.

Requisiti non funzionali

- L'applicazione deve essere facile da usare per i bambini con disturbo dello spettro autistico.
- L'applicazione deve avere un'interfaccia utente chiara e intuitiva.
- L'applicazione deve essere compatibile con il tablet Pepper.
- L'applicazione deve essere sicura per i bambini.
- L'applicazione non deve raccogliere informazioni personali o sensibili dei bambini.
- L'applicazione deve consentire al medico di visualizzare la cronologia dei test somministrati, per monitorare i progressi del bambino nel tempo.

4.3 Architettura generale del sistema

L'architettura dell'applicazione è completamente locale. Non si basa su un modello client-server, ma sfrutta la memoria interna del tablet di Pepper per la gestione dei dati. L'applicazione salva i dati in un file system strutturato, senza la necessità di connessioni di rete.

- **Applicazione Android:** L'applicazione Android è responsabile dell'interfaccia utente, della gestione dei test e della comunicazione con il robot. Uti-

lizza Java/Kotlin per lo sviluppo e Android Studio come ambiente di sviluppo integrato.

- **Comunicazione con Pepper:** La comunicazione tra l'app e il robot Pepper avviene tramite l'SDK NaoQi, che fornisce una libreria di funzioni per interagire con Pepper.
- **Gestione dei dati:** I dati relativi a pazienti, test e risultati sono memorizzati localmente nel file system.

4.4 UX per medico e bambino

La progettazione dell'interfaccia utente si concentra su un'esperienza intuitiva e accessibile per entrambi gli utenti: il medico e il bambino.

Per il medico:

L'interfaccia è orientata a una gestione rapida dei pazienti e dei test. È presente un pannello di controllo chiaro, con sezioni per l'inserimento dei dati del paziente, la selezione del test da somministrare e la visualizzazione dei risultati. L'interazione con l'app è veloce e funzionale, senza distrazioni.

Per il bambino:

L'interfaccia è progettata per essere colorata, semplice e interattiva. Le domande sono presentate in modo visivo, accompagnate da immagini o animazioni. Pepper guida il bambino durante tutta la sessione, mantenendo un tono rassicurante e motivante.

Chapter 5

Modelli di sistema

In questo capitolo si esplorano i modelli di sistema attraverso l'analisi di vari scenari e casi d'uso specifici. L'obiettivo è fornire una rappresentazione dettagliata e strutturata del funzionamento del sistema in diverse condizioni operative, per comprenderne meglio le dinamiche interne e le interazioni tra i componenti. Attraverso l'uso di scenari realistici, si illustreranno le applicazioni pratiche dei modelli, evidenziando come questi possano supportare la progettazione, la validazione e l'ottimizzazione del sistema stesso. Verranno quindi analizzati i casi d'uso principali, che descrivono le modalità con cui gli utenti interagiscono con il sistema, le risposte attese e le possibili varianti di comportamento. Questa trattazione aiuterà a mettere in luce le potenzialità offerte dai modelli di sistema nel contesto del progetto, oltre a identificare eventuali criticità e punti di miglioramento per le fasi successive dello sviluppo.

5.1 Scenari

Titolo scenario	Scenario di successo – Somministrazione del test T.R.O.G.
Attori	Bambino, Medico, Robot Pepper
Condizioni iniziali	<ul style="list-style-type: none">• Il medico è autenticato nel sistema.• Il paziente “Luca Verdi” è selezionato.• Il test T.R.O.G. è pronto per la somministrazione.
Flusso dello scenario	<ol style="list-style-type: none">1. Pepper saluta il medico e il bambino.2. Pepper mostra la prima immagine e legge la frase associata.3. Il bambino osserva quattro immagini e seleziona quella corretta.4. Pepper registra la risposta e fornisce un feedback motivante.5. Alla fine, Pepper mostra il punteggio ottenuto dal bambino.
Risultato atteso	<ul style="list-style-type: none">• Il sistema registra tutte le risposte corrette e errate salvando il punteggio.

Table 5.1: Scenario di successo per la somministrazione del test T.R.O.G.

Titolo scenario	Scenario di annotazione – Medico scrive una nota sulle risposte del bambino
Attori	Medico, Robot Pepper
Condizioni iniziali	<ul style="list-style-type: none"> • Il medico ha effettuato il login nel sistema. • Il paziente e la sessione di test sono stati selezionati. • La registrazione della risposta del bambino è disponibile.
Flusso dello scenario	<ol style="list-style-type: none"> 1. Il medico ascolta la registrazione della risposta del bambino. 2. Il medico seleziona l'opzione "Aggiungi nota". 3. Il medico scrive la nota descrivendo osservazioni e commenti. 4. Il medico salva la nota, che viene associata alla risposta specifica. 5. Pepper conferma il salvataggio con un messaggio di conferma.
Risultato atteso	<ul style="list-style-type: none"> • La nota viene salvata correttamente e collegata alla risposta. • Il medico può consultare in seguito la nota associata. • Le osservazioni sono disponibili per analisi e report futuri.

Table 5.2: Scenario di annotazione di risposta da parte del medico

Titolo scenario	Scenario di visualizzazione – Medico visualizza i grafici delle performance del paziente
Attori	Medico, Robot Pepper
Condizioni iniziali	<ul style="list-style-type: none"> • Il medico è autenticato nel sistema. • Il paziente è stato selezionato. • Sono presenti risultati di test precedenti per il paziente.
Flusso dello scenario	<ol style="list-style-type: none"> 1. Il medico accede alla dashboard dei pazienti. 2. Pepper guida il medico alla sezione “Visualizza risultati”. 3. Il medico seleziona il paziente desiderato. 4. Il sistema mostra i grafici con andamento delle risposte corrette, errate e distrazioni nel tempo. 5. Il medico analizza i grafici per valutare l’andamento cognitivo del paziente.
Risultato atteso	<ul style="list-style-type: none"> • Il medico ottiene una panoramica chiara e dettagliata dell’andamento del paziente. • I grafici sono aggiornati con gli ultimi dati. • Il medico può decidere eventuali azioni o approfondimenti sulla base dei risultati.

Table 5.3: Scenario di visualizzazione dei risultati grafici

Titolo scenario	Scenario alternativo – Il bambino non risponde e Pepper attira la sua attenzione
Attori	Bambino, Medico, Robot Pepper
Condizioni iniziali	<ul style="list-style-type: none"> • È in corso la somministrazione del test. • Il bambino non fornisce risposta per alcuni secondi.
Flusso dello scenario	<ol style="list-style-type: none"> 1. Pepper attende una risposta. 2. Dopo un breve timeout, Pepper rileva l'assenza di interazione. 3. Pepper emette un suono delicato e accende una luce LED colorata. 4. Pepper pronuncia una frase motivante come: “Va tutto bene! Sei pronto per rispondere?” 5. Il bambino riprende l'interazione con il robot. 6. Il test riprende normalmente.
Risultato atteso	<ul style="list-style-type: none"> • Il bambino viene richiamato gentilmente all'attenzione. • L'interazione riprende senza frustrazione o pressione.

Table 5.4: Scenario alternativo: il bambino non risponde e Pepper attira l'attenzione

Titolo scenario	Scenario alternativo – Il bambino risponde male e Pepper lo invita alla calma
Attori	Bambino, Medico, Robot Pepper
Condizioni iniziali	<ul style="list-style-type: none"> • Il bambino fornisce una risposta errata al test. • Il bambino mostra segni di frustrazione o agitazione.
Flusso dello scenario	<ol style="list-style-type: none"> 1. Pepper registra una risposta errata. 2. Il robot rileva agitazione tramite movimenti, voce o espressione facciale. 3. Pepper dice con tono rassicurante: “Non preoccuparti, capita a tutti!” 4. Pepper esegue l’animazione del respiro guidato (luci pulsanti e movimento calmo del torace). 5. Il bambino imita il respiro e si tranquillizza. 6. Il test riprende con una nuova domanda.
Risultato atteso	<ul style="list-style-type: none"> • Il bambino si calma e riprende serenamente il test. • L’esperienza resta positiva e non stressante.

Table 5.5: Scenario alternativo: il bambino si agita dopo un errore e Pepper lo tranquillizza

5.2 Casi d'uso

Nome	Somministrazione del test di comprensione didattica (T.R.O.G.)
Attori	Bambino con disturbi dello spettro autistico, Collaboratore, Robot Pepper
Descrizione	Pepper somministra il test T.R.O.G., mostrando immagini e leggendo le frasi. Il bambino sceglie la figura corretta.
Pre-condizioni	Il collaboratore ha effettuato l'accesso e selezionato il paziente e il test.
Flusso Principale	<ol style="list-style-type: none">1. Il collaboratore avvia l'app ed effettua il login.2. Pepper saluta e accoglie il medico.3. Il collaboratore seleziona il test T.R.O.G.4. Pepper mostra una figura e legge la frase.5. Il bambino sceglie una delle quattro immagini.6. Se distratto, Pepper ripete e attira attenzione.7. Le risposte corrette sono registrate.8. Alla fine, Pepper mostra il punteggio.
Post-condizioni	Il punteggio è memorizzato, pronto per l'analisi grafica.

Table 5.6: Caso d'Uso – Somministrazione T.R.O.G.

Nome	Somministrazione del test “Sally e Annie”
Attori	Bambino con disturbi dello spettro autistico, Collaboratore, Robot Pepper
Descrizione	Pepper mostra un breve video animato per valutare la teoria della mente e registra la risposta del bambino.
Pre-condizioni	Il collaboratore ha effettuato l’accesso e selezionato il paziente e il test.
Flusso Principale	<ol style="list-style-type: none"> 1. Il collaboratore accede all’applicazione. 2. Pepper saluta ed esegue un inchino. 3. Il test “Sally e Annie” viene selezionato. 4. Il video viene riprodotto sul tablet. 5. Pepper pone la domanda: “Dove cercherà Sally la palla?” 6. Il bambino risponde vocalmente. 7. La risposta viene registrata. 8. Il medico ascolta la registrazione e può aggiungere note.
Post-condizioni	La registrazione della risposta è salvata per analisi futura.

Table 5.7: Caso d’Uso – Somministrazione “Sally e Annie”

Nome	Somministrazione del Cloze Test
Attori	Bambino con disturbi dello spettro autistico, Collaboratore, Robot Pepper
Descrizione	Pepper legge frasi con parole mancanti e chiede al bambino di completarle. Le risposte vengono confrontate con quelle attese.
Pre-condizioni	Il collaboratore ha effettuato l'accesso e selezionato il paziente e il test.
Flusso Principale	<ol style="list-style-type: none"> 1. Il collaboratore seleziona il Cloze Test. 2. Pepper saluta e introduce il test. 3. Legge una frase con una parola mancante. 4. Il bambino completa la frase oralmente. 5. La risposta è confrontata con la parola attesa. 6. Le risposte corrette vengono annotate. 7. Pepper esegue un'animazione di rinforzo positivo. 8. Al termine comunica il punteggio finale.
Post-condizioni	Il test è registrato localmente con punteggio e risposte.

Table 5.8: Caso d'Uso – Somministrazione Cloze Test

Nome	Visualizzazione dei risultati in forma grafica
Attori	Medico, Robot Pepper
Descrizione	Il medico accede ai grafici di performance del bambino, suddivisi per test.
Pre-condizioni	Il medico ha effettuato il login e ha selezionato il paziente.
Flusso Principale	<ol style="list-style-type: none"> 1. Il medico accede alla dashboard. 2. Pepper saluta e guida alla sezione “Visualizza pazienti”. 3. Seleziona il paziente e apre la sezione “Grafici”. 4. Visualizza: risposte corrette, errate, mancanti e distrazioni. 5. Il medico può confrontare test nel tempo.
Post-condizioni	Il medico ottiene una visione globale delle performance cognitive del paziente.

Table 5.9: Caso d’Uso – Visualizzazione Risultati

Nome	Aggiunta di un nuovo paziente e fase di registrazione
Attori	Medico, Robot Pepper
Descrizione	Il medico crea un nuovo account e registra un nuovo paziente nel sistema.
Pre-condizioni	Il medico ha aperto l'app per la prima volta o è disconnesso.
Flusso Principale	<ol style="list-style-type: none"> 1. Il medico seleziona “Registrati” e crea un account. 2. Dopo il login, Pepper esegue un inchino. 3. Il medico seleziona “Aggiungi paziente”. 4. Inserisce nome, cognome ed età del bambino. 5. Il paziente è ora disponibile nella lista.
Post-condizioni	Il nuovo paziente è stato registrato correttamente nel sistema.

Table 5.10: Caso d’Uso – Aggiunta Nuovo Paziente

Nome	Visualizzazione dello storico dei test del paziente
Attori	Medico, Robot Pepper
Descrizione	Il medico visualizza l'elenco cronologico dei test svolti da un paziente.
Pre-condizioni	Il medico ha effettuato il login e ha selezionato un paziente.
Flusso Principale	<ol style="list-style-type: none"> 1. Il medico accede all'applicazione. 2. Pepper saluta e guida all'interfaccia pazienti. 3. Il medico seleziona il paziente. 4. Visualizza lo storico test: tipo, data e punteggio. 5. Può selezionare un test per visualizzare i dettagli.
Post-condizioni	Il medico ottiene una panoramica completa del percorso del paziente.

Table 5.11: Caso d'Uso – Storico Test Paziente

Chapter 6

Data persistent management

La pianificazione prevede la sostituzione dei caratteri speciali nelle email con trattini bassi (_). Questo processo rende l'indirizzo email compatibile con il sistema di file del sistema operativo, senza compromettere l'unicità dell'identificativo.

Ad esempio, l'email

`mario.rossi@example.com`

viene trasformata in

`mario_rossi_example_com.`

All'interno della cartella principale del medico (identificata dall'email trasformata), vengono create delle sottocartelle per ogni paziente. Ogni cartella del paziente è denominata con nome e cognome separati da trattino basso:

- `luca_verdi` per “Luca Verdi”;
- in caso di omonimia, si aggiunge un timestamp (es. `luca_verdi_20250620T1030`).

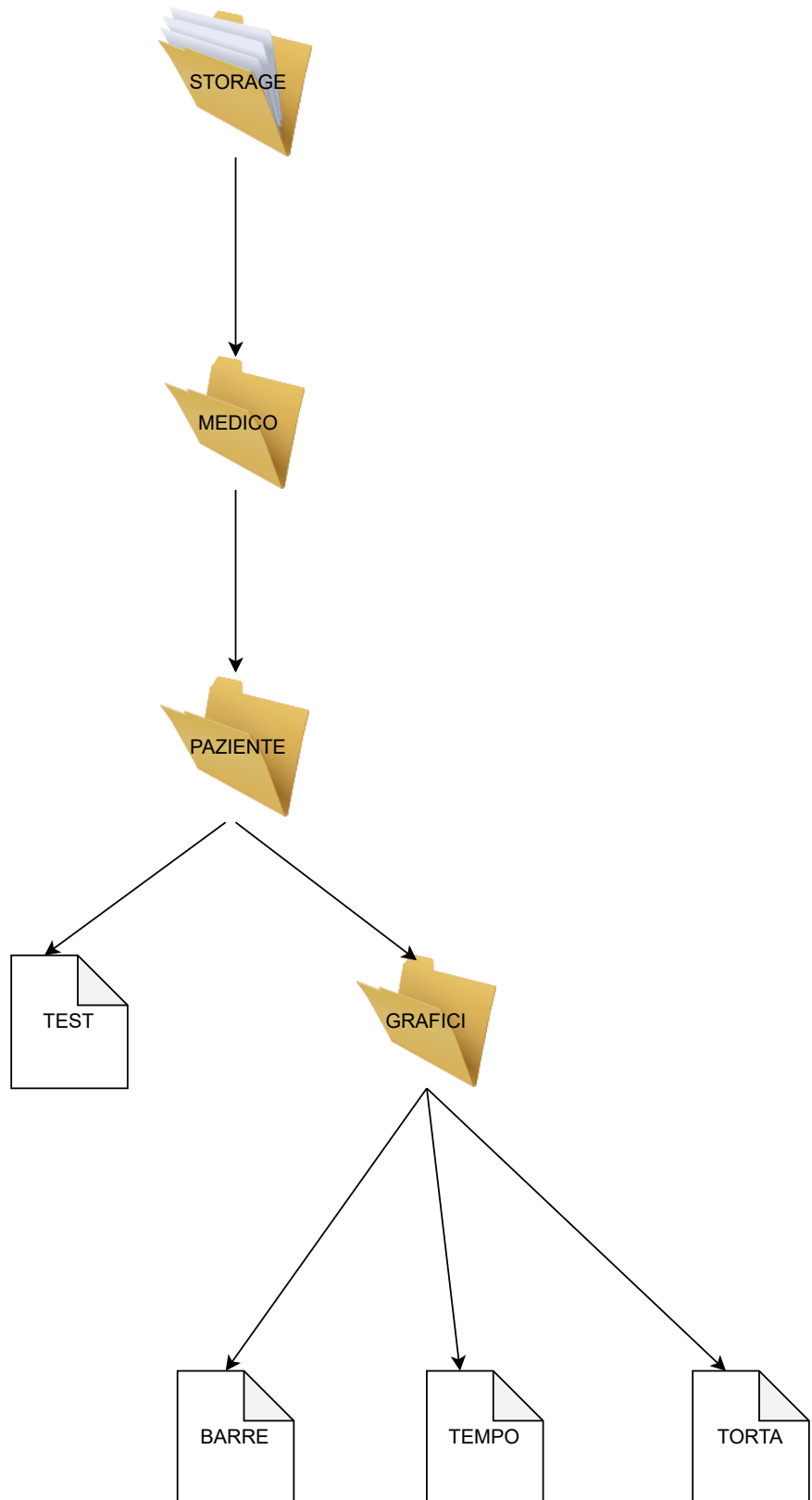
Ogni cartella paziente contiene:

- i file di risultato dei test cognitivi eseguiti;

- le informazioni anagrafiche del bambino;
- i dati grezzi per la generazione dei grafici di performance.

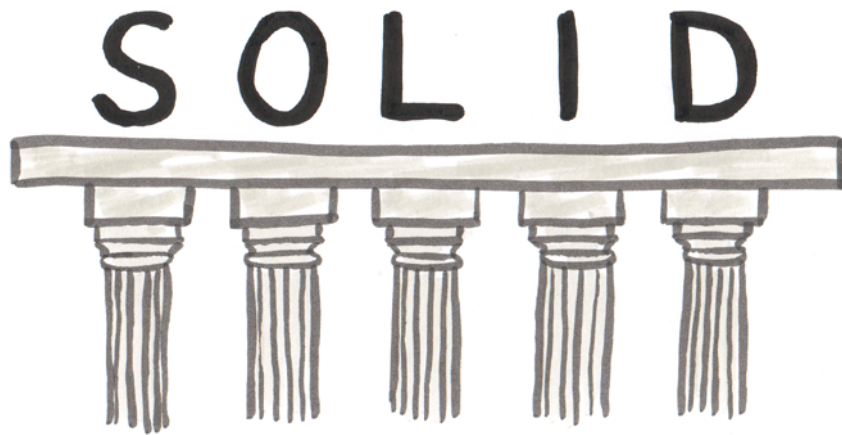
Questa struttura gerarchica consente di organizzare i dati in modo ordinato e di garantire che:

1. ogni medico acceda solamente alla propria cartella, identificata dall'email;
2. i dati sensibili restino isolati, poiché l'accesso alle sottocartelle è ristretto al solo medico titolare.



Chapter 7

Principi SOLID



I principi SOLID sono un insieme di regole di progettazione software che mirano a creare codice leggibile, manutenibile ed estendibile. I principi SOLID sono stati definiti da Robert C. Martin e sono composti dalle seguenti regole:

Single Responsibility Principle (SRP): ogni classe dovrebbe avere una sola responsabilità e un solo motivo per essere modificata. Nel nostro caso questo principio non è rispettato, poiché alcune classi gestiscono più funzionalità contemporaneamente.

Open/Closed Principle (OCP): le classi dovrebbero essere aperte per estensione ma chiuse per modifica. Nel nostro caso questo principio è rispettato, in quanto è possibile aggiungere funzionalità senza modificare

il codice esistente.

Liskov Substitution Principle (LSP): le sottoclassi dovrebbero poter sostituire le classi base senza causare problemi. Nel nostro caso questo principio è rispettato, grazie all'uso dell'ereditarietà.

Interface Segregation Principle (ISP): le interfacce dovrebbero essere piccole e specifiche per evitare implementazioni inutilizzate. Nel nostro caso questo principio è rispettato, in quanto le interfacce sono progettate specificamente e con pochi metodi.

Dependency Inversion Principle (DIP): i moduli di alto livello non dovrebbero dipendere da moduli di basso livello, ma entrambi dovrebbero dipendere da astrazioni. Nel nostro caso questo principio non è rispettato, poiché non si utilizza una chiara separazione tramite DAO o astrazioni.

Chapter 8

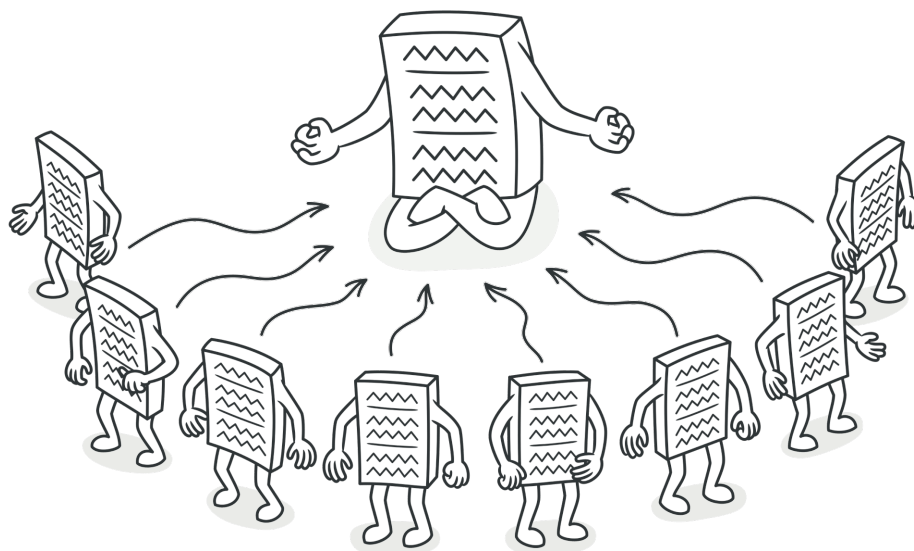
Design Pattern

Si è scelto di utilizzare diversi design pattern in quanto si aveva necessità di risolvere problemi di progettazione comuni in modo efficiente e riutilizzabile. I design pattern infatti forniscono soluzioni ben testate e collaudate a problemi di progettazione specifici che possono verificarsi in diversi contesti. Utilizzando i design pattern, si evita di dover reinventare la ruota ogni volta che si presenta un problema simile e ci si può invece concentrare sull'implementazione della soluzione specifica per il progetto. Inoltre, l'utilizzo dei design pattern rende il codice più leggibile e comprensibile per gli altri sviluppatori, poiché i pattern sono ben noti e documentati nella comunità dello sviluppo software.

Pattern utilizzati:

- Singleton
- Model View Controller (MVC)
- Auth Proxy

Singleton



L'utilizzo del pattern Singleton è utile per la creazione di una sola istanza di una classe sia presente in tutta l'applicazione e che sia possibile accedere a questa istanza da qualsiasi punto dell'applicazione. Ad esempio, per poter cambiare le varie immagini del test, ci salviamo il valore del contatore per poi concatenare il nome dell'immagine per poter sostituire le varie immagini del test, inoltre utilizziamo tale pattern per spostarci informazioni utili.

```

public class CounterSingleton {
    // questa classe salva il valore del contatore delle chiamate di onclick.

    private static CounterSingleton instance;
    private int cont=1;
    private int point=0;
    private int tempoImpiegatoFineTest = 0;
    private int numeroRisposteErrate = 0;
    private int numeroSoundAttention = 0;
    private String testAvviato;
    private String bambinoSelezionato;
    private CounterSingleton() {
        // Costruttore privato per impedire la creazione diretta dell'istanza
    }

    // definisco l'oggetto ed evito race condition grazie a synchronized
    public static synchronized CounterSingleton getInstance(){
        if (instance==null) {
            instance=new CounterSingleton();
        }
        return instance;
    }

    public int getNumeroSoundAttention(){ return numeroSoundAttention; }

    public void setNumeroSoundAttention(int numeroSoundAttention) {
        this.numeroSoundAttention = numeroSoundAttention;
    }

    int getNumeroRisposteErrate(){return this.numeroRisposteErrate;}

    public void setNumeroRisposteErrate(int numeroRisposteErrate) {
        this.numeroRisposteErrate = numeroRisposteErrate;
    }

    int getTempoImpiegatoFineTest(){return this.tempoImpiegatoFineTest;}
    void setTempoImpiegatoFineTest(int tempoImpiegatoFineTest){this.tempoImpiegatoFineTest=tempoImpiegatoFineTest;}

    String getTestAvviato() { return this.testAvviato; }
    void setTestAvviato(String testAvviato) { this.testAvviato = testAvviato; }

    void incrementaNumeroRisposteErrate(){this.numeroRisposteErrate++;}
    void incrementaNumeroSoundAttention(){this.numeroSoundAttention++;}

    void incrementaCont(){
        cont+=1;
    }
}

```

```

void incrementaCont(){
    cont+=1;
}

void incrementaPunteggio() { point+=1; }
int getCont(){
    return cont;
}

public int getPoint() { return point; }

void setCount(int x) { cont=x; }

public void setPoint(int point) { this.point = point; }

public String getBambinoSelezionato() { return bambinoSelezionato; }

public void setBambinoSelezionato(String bambinoSelezionato) {
    this.bambinoSelezionato = bambinoSelezionato;
}

public void reset(){
    setNumeroRisposteErrate(0);
    setNumeroSoundAttention(0);
    setTestAvviato(null);
    setPoint(0);
    setCount(1);
}
}

```

```

public class UserSessionSingleton {
    private boolean isLoggedIn = false;
    private String email;
    private static UserSessionSingleton instance;

    private UserSessionSingleton(){

    }

    public static synchronized UserSessionSingleton getInstance(){
        if(instance == null)
            instance = new UserSessionSingleton();
        return instance;
    }

    public void setIsLogged(boolean isLoggedIn){this.isLoggedIn = isLoggedIn;}

    public boolean getIsLogged() { return isLoggedIn; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }
}

```

```

public class PepperSingleton {

    private static PepperSingleton instance;
    private QiContext qiContext;

    // Costruttore privato
    private PepperSingleton() {}

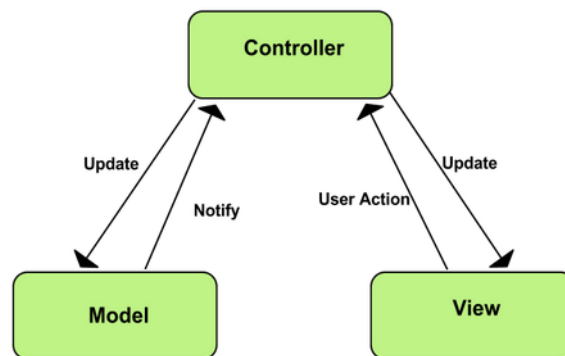
    // Metodo per ottenere l'istanza del singleton (senza richiedere QiContext)
    public static synchronized PepperSingleton getInstance() {
        if (instance == null) {
            instance = new PepperSingleton();
        }
        return instance;
    }

    // Metodo per ottenere il QiContext
    public QiContext getQiContext() { return qiContext; }

    // Metodo per impostare il QiContext
    public void setQiContext(QiContext qiContext) { this.qiContext = qiContext; }
}

```

Model View Controller (MVC)



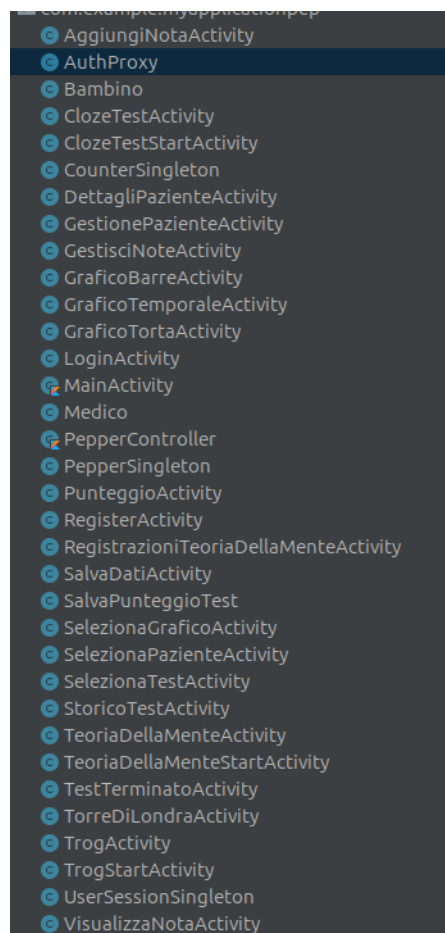
MVC (Model-View-Controller) è un pattern di progettazione software che consente di separare la logica dell'applicazione in tre componenti distinti: il Modello, la Vista e il Controller.

È stato utilizzato questo pattern per due motivi principali: mantenere il codice pulito e separare nettamente le diverse sezioni logiche del lavoro. Il pattern richiede di separare la logica dei moduli in un'applicazione Java e noi abbiamo suddiviso i documenti e le classi in tre sezioni principali, come indicato dal pattern.

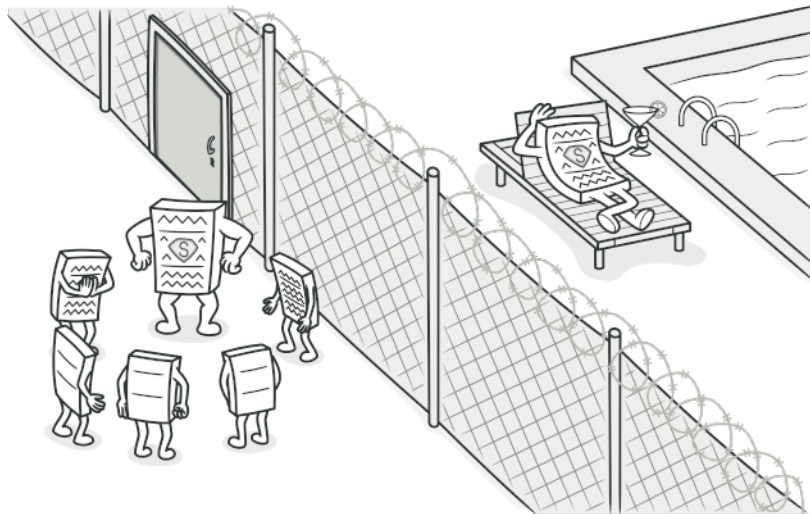
La sezione *Activity* contiene l'interazione e l'elaborazione della sezione View e di quella presente nel Modello. Ciò permette ai controller di essere isolati dietro la logica di funzionamento dei metodi delle classi Modello che

creano il nostro ambiente e permette loro di restare in ascolto e reagire in maniera dinamica agli input ricevuti dalla sezione View. Questa sezione è composta dalle classi "Controller".

La sezione View contiene ciò che l'utente vede a schermo, quindi la GUI con cui esso si rapporta. Nel nostro caso, è definita dai file FXML presenti nella sezione Resources. Questi file sono privi di qualsiasi tipo di logica di implementazione e contengono solo ed esclusivamente la GUI mostrata all'utente.



Auth Proxy



Nel progetto è stato adottato un Auth Proxy, ovvero un pattern di progettazione che si basa sul concetto di proxy per l'autenticazione. Questo approccio permette di centralizzare il controllo degli accessi, migliorando l'organizzazione del codice e separando le responsabilità.

La classe **AuthProxy** ha il compito di verificare se l'utente è autenticato ogni volta che si accede a una sezione dell'applicazione che richiede il login. Nel caso in cui l'utente non sia loggato, viene effettuato automaticamente il reindirizzamento alla schermata di login.

Questo proxy riceve un riferimento al *Context* e un valore booleano che rappresenta lo stato di autenticazione dell'utente. Se l'utente non è autenticato (**false**), viene creato un *Intent* per lanciare la **LoginActivity**, impedendo l'accesso non autorizzato alle funzionalità riservate.

Questo pattern offre i seguenti vantaggi:

- Centralizzazione della logica di accesso, evitando ridondanza di codice.
- Manutenibilità, grazie alla separazione tra la logica di autenticazione e le attività dell'interfaccia utente.

- Sicurezza, poiché impedisce l'accesso a sezioni riservate se non si è autenticati.

In sintesi, l'Auth Proxy agisce come un guardiano tra l'utente e l'interfaccia, garantendo che l'accesso avvenga solo se sono rispettate le condizioni di autenticazione.

```
public class AuthProxy {  
  
    private Context context;  
  
    public AuthProxy(Context context) { this.context = context; }  
    // Verifica se l'utente è loggato  
    public void isLoggedIn(boolean isLoggedIn) {  
        if(!isLoggedIn){  
            Intent intent = new Intent(context, LoginActivity.class);  
            // Avvia l'Activity  
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);  
            context.startActivity(intent);  
        }  
    }  
}
```

Chapter 9

Gestione degli Errori

Nel nostro progetto è stata posta particolare attenzione alla gestione degli errori, al fine di garantire la stabilità e la robustezza del sistema durante l'esecuzione.

Sono stati individuati i punti critici in cui potrebbero verificarsi eccezioni (come accessi al file system, operazioni su oggetti nulli o input errati) e sono stati inseriti costrutti `try-catch` per intercettare e gestire tali eccezioni in modo appropriato.

L'obiettivo è prevenire il crash dell'applicazione, fornendo messaggi di errore chiari e significativi per l'utente, oppure registrando l'errore in modo tale da poter intervenire facilmente in fase di debugging.

Inoltre, è stata adottata una logica difensiva nei confronti di input non validi o comportamenti inattesi, al fine di aumentare la tolleranza agli errori e migliorare l'esperienza utente.

Esempi di casi gestiti includono:

- Tentativi di accesso a file o cartelle inesistenti.
- Errori nel parsing di dati o formati non previsti.
- Connessioni non riuscite o mancanza di risposta da parte di componenti hardware (es. robot Pepper).

- Operazioni su variabili non inizializzate o oggetti nulli.

Questa strategia ha permesso di sviluppare un'applicazione più affidabile, in grado di affrontare situazioni impreviste senza compromettere la continuità del servizio. di seguito alcuni esempi:

```
try (FileOutputStream fos = new FileOutputStream(file)) {
    // Salva i dati del medico nel file
    String userData = "Email: " + medico.getEmail() + "\nPassword: " + medico.getPassword() +
        "\nNome: " + medico.getNome() + "\nCognome: " + medico.getCognome();

    fos.write(userData.getBytes());

    // Mostra un messaggio di successo
    Toast.makeText( context: this, text: "Registrazione completata!", Toast.LENGTH_LONG).show();

    // Torna alla MainActivity
    Intent intent = new Intent( packageContext: this, MainActivity.class);
    startActivity(intent);
    finish();
} catch (IOException e) {
    e.printStackTrace();
    Toast.makeText( context: this, text: "Errore nel salvataggio dei dati.", Toast.LENGTH_LONG).show();
}
```

```
private void riproduciAudio(File file) {
    try {
        if (mediaPlayer.isPlaying()) {
            mediaPlayer.stop();
            mediaPlayer.reset();
        }
        mediaPlayer.setDataSource(file.getAbsolutePath());
        mediaPlayer.prepareAsync(); // Usa prepareAsync per non bloccare il thread principale
        mediaPlayer.setOnPreparedListener(mp -> {
            mp.start();
            Toast.makeText( context: this, text: "Riproduzione avviata", Toast.LENGTH_SHORT).show();
        });
    } catch (IOException e) {
        Toast.makeText( context: this, text: "Errore nella riproduzione", Toast.LENGTH_SHORT).show();
        e.printStackTrace();
    }
}
```

```
// Nome del file con nome e cognome
String fileName = "dati_test.txt";
File file = new File(bambinoDir, fileName); // Percorso completo del file

FileOutputStream fos = null;
try {
    fos = new FileOutputStream(file, append: true); // true = modalità append
    fos.write(data.getBytes());
} catch (IOException e) {
    e.printStackTrace();
} finally {
    try {
        if (fos != null) fos.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Chapter 10

Parti rilevanti del codice

10.1 `playAnimation`

Il metodo `playAnimation` ha la funzione di far eseguire un'animazione sul robot Pepper utilizzando una risorsa locale specificata come parametro.

Il metodo esegue la logica all'interno di un thread separato per non bloccare il thread principale dell'applicazione.

All'interno del blocco `try`, viene prima creata un'istanza di `Animation` utilizzando il `qiContext` e la risorsa di animazione passata, che rappresenta l'animazione da riprodurre.

Successivamente, viene costruito un oggetto `Animate` associato all'animazione creata, il quale gestisce l'esecuzione del movimento del robot.

Infine, viene avviata in maniera asincrona l'animazione tramite il metodo `run()` di `animate.async()`.

In caso di eccezioni o errori durante la creazione o l'esecuzione dell'animazione, questi vengono intercettati e loggati con un messaggio di errore.

Questo metodo consente quindi di far animare Pepper in modo fluido senza interrompere l'esecuzione principale dell'applicazione.

```

fun playAnimation(animationResource: Int) {
    // Esegui in un thread separato
    AsyncTask.execute {
        try {
            // Crea l'animazione usando la risorsa locale (qianim)
            val animation: Animation = AnimationBuilder.with(qiContext)
                .withResources(animationResource) // ID della risorsa dell'animazione
                .build()

            // Crea il comportamento Animate
            val animate: Animate = AnimateBuilder.with(qiContext)
                .withAnimation(animation)
                .build()

            // Esegui l'animazione
            animate.async().run()
        } catch (e: Exception) {
            Log.e( tag: "ANIMATION", msg: "Errore durante l'animazione", e)
        }
    }
}

```

10.2 riconoscereParole

Il metodo `riconoscereParole` ha lo scopo di riconoscere vocalmente una parola specifica passata come parametro, eseguendo l'operazione in modo asincrono per non bloccare il thread principale.

All'inizio, il metodo verifica se il contesto `qiContext` è disponibile; in caso contrario, termina immediatamente.

Successivamente, viene avviato un task in background tramite `AsyncTask.execute`, che esegue il riconoscimento vocale.

Nel dettaglio, si costruisce un `PhraseSet` contenente la parola da riconoscere e si configura un oggetto `Listen` per ascoltare la parola pronunciata dall'utente.

Dopo l'avvio dell'ascolto con `listen.run()`, si ottiene il risultato, ossia la parola effettivamente captata dal robot.

Il metodo confronta quindi la parola riconosciuta con quella attesa, ignorando le differenze di maiuscole/minuscole.

Se la parola è corretta, viene incrementato il punteggio tramite il `CounterSingleton` in caso contrario, non viene effettuata alcuna azione aggiuntiva.

Eventuali errori nel processo di riconoscimento vengono catturati e reg-

istrati tramite il Log.

In questo modo, il metodo consente a Pepper di ascoltare, riconoscere e valutare la risposta vocale dell'utente, mantenendo fluida l'interazione senza blocchi dell'interfaccia.

```
// Metodo per riconoscere parole (asincrono)
fun riconoscereParole(parolaDaRiconoscere: String) {
    val context = qiContext ?: return // Restituisce se qiContext non è disponibile
    Log.d( tag: "PAROLA DA RICONOSCERE", parolaDaRiconoscere);

    // Esegui in un thread separato usando AsyncTask
    AsyncTask.execute {
        try {
            val phraseSet = PhraseSetBuilder.with(context)
                .withTexts(parolaDaRiconoscere)
                .build()

            val listen = ListenBuilder.with(context)
                .withPhraseSet(phraseSet)
                .build()
            Log.d( tag: "RICONOSCIMENTO", msg: "Sta ascoltando per le parole...")
            // Ascolta la parola
            val result = listen.run()
            Log.d( tag: "RICONOSCIMENTO", msg: "Ascolto completato. Parola riconosciuta: ${result.heardPhrase?.text ?: "Nessuna parola"}")
            val parolaRiconosciuta = result.heardPhrase?.text ?: ""

            println("Pepper ha riconosciuto: $parolaRiconosciuta")
            // Confronta la parola riconosciuta con la parola da riconoscere
            val isCorrect = parolaRiconosciuta.equals(parolaDaRiconoscere, ignoreCase = true)

            // Se necessario, puoi comunicare il risultato al thread principale
            // per esempio usando un callback, o anche aggiornare la UI
            val contatore = CounterSingleton.getInstance()
            if (isCorrect) {
                Log.d( tag: "RICONOSCIMENTO", msg: "Parola corretta riconosciuta")
                contatore.incrementaPunteggio()
            } else {
                Log.d( tag: "RICONOSCIMENTO", msg: "Parola errata riconosciuta")
            }
        } catch (e: Exception) {
            Log.e( tag: "RICONOSCIMENTO", msg: "Errore durante il riconoscimento", e)
        }
    }
}
```

10.3 soundAttention

Il metodo `soundAttention` serve a generare un suono specifico per attirare l'attenzione dei bambini durante l'interazione con il robot.

In dettaglio, il metodo verifica innanzitutto se un'istanza di `MediaPlayer` è già attiva e in riproduzione. In tal caso, ferma la riproduzione, rilascia

le risorse associate e azzera il riferimento per evitare perdite di memoria.

Successivamente, viene creato un nuovo `MediaPlayer` utilizzando una risorsa audio specifica (nel caso, un file chiamato `attention` presente nella cartella `raw`).

Dopo aver notificato l'utente tramite un messaggio `Toast`, il suono viene avviato.

Infine, al termine della riproduzione, il `MediaPlayer` viene rilasciato e il riferimento interno azzerato, garantendo una corretta gestione delle risorse.

```
// metodo che genera un suono per attirare l'attenzione dei bambini
fun soundAttention() {
    mediaPlayer?.let { it: MediaPlayer
        if (it.isPlaying) {
            it.stop()
            it.release()
            mediaPlayer = null
        }
    }

    mediaPlayer = MediaPlayer.create(context, R.raw.attention)
    mediaPlayer?.let { it: MediaPlayer
        Toast.makeText(context, text: "Play sound on Robot", Toast.LENGTH_LONG).show()
        it.start()

        it.setOnCompletionListener { mp ->
            mp.release()
            mediaPlayer = null
        }
    }
}
```

10.4 speak

Il metodo `speak` prende in input una stringa `frase` e utilizza il contesto `qiContext` per far pronunciare al robot Pepper il testo passato.

Innanzitutto, verifica che il contesto non sia nullo. Se disponibile, esegue l'operazione in un thread separato tramite `AsyncTask`, per evitare di bloccare il thread principale dell'applicazione.

All'interno del thread secondario, il metodo costruisce un oggetto `Say` utilizzando il `SayBuilder`, configurandolo con il testo da pronunciare. Quindi avvia l'esecuzione asincrona della sintesi vocale tramite `say.async().run()` che restituisce un `Future`.

A completamento dell'operazione, attraverso `thenConsume` controlla se si è verificato un errore durante il parlato. In caso positivo, lo registra nei log; altrimenti conferma il successo con un messaggio di log.

Se il contesto non è disponibile, viene mostrato all'utente un messaggio di errore tramite `Toast`.

In questo modo, il metodo garantisce una sintesi vocale non bloccante e gestisce eventuali errori di comunicazione con il robot.

```
// Metodo per far parlare Pepper usato per guidare utente nell'applicazione
fun speak(frase: String) {
    val context = qiContext
    if (context != null) {
        // Esegui l'operazione in un thread separato usando AsyncTask
        AsyncTask.execute {
            try {
                // Crea l'oggetto Say
                val say = SayBuilder.with(context)
                    .withText(frase)
                    .build()

                // Esegui l'operazione di parlato in background
                val future: Future<Void> = say.async().run()

                // Gestisci il risultato quando l'operazione è completata
                future.thenConsume { result ->
                    if (result.hasError()) {
                        Log.e( tag: "SPEAK", msg: "Errore durante il parlato", result.error)
                    } else {
                        Log.d( tag: "SPEAK", msg: "Pepper ha parlato con successo")
                    }
                }
            } catch (e: Exception) {
                Log.e( tag: "SPEAK", msg: "Errore", e)
            }
        }
    } else {
        // Mostra un messaggio se QiContext non è disponibile
        Toast.makeText(context, text: "QiContext non disponibile", Toast.LENGTH_SHORT).show()
    }
}
```

10.5 incoraggiaUtente

Il metodo `incoraggiaUtente` ha lo scopo di motivare i bambini a proseguire con il test.

All'interno del metodo viene selezionata in modo casuale una frase da un array di frasi motivazionali (`frasiMotivazionali`).

Successivamente, la frase scelta viene passata al metodo `speak`, che si occupa di far pronunciare a Pepper la frase motivazionale.

In questo modo, il robot interagisce con il bambino fornendo incoraggiamenti vocali per mantenere alta l'attenzione e la motivazione durante la somministrazione del test.

```
// la funzione incoraggia i bambini a continuare il test
fun incoraggiaUtente() {
    // Scegli una frase casuale dall'array di frasi motivazionali
    val fraseMotivazionale = frasiMotivazionali.random()

    // Fai parlare Pepper con la frase scelta
    speak(fraseMotivazionale)
}
```

10.6 gestisciTocco

Il metodo `gestisciTocco` gestisce l'interazione dell'utente con un'immagine all'interno dell'applicazione, rilevando il punto esatto in cui è stato effettuato il tocco. L'immagine è suddivisa idealmente in quattro aree quadranti, e il metodo individua quale area è stata toccata in base alle coordinate x e y rilevate dall'evento di tocco. Successivamente verifica se l'area selezionata corrisponde alla risposta corretta per la domanda corrente, utilizzando un array di risposte corrette indicizzato dalla domanda stessa.

Nel caso di risposta esatta, viene incrementato un contatore del punteggio tramite un singleton, viene eseguita la gestione della risposta corretta e, se presente, il robot Pepper viene attivato per incoraggiare l'utente con feedback vocali. Se la risposta è errata, viene invece richiamata una procedura per gestire la risposta sbagliata.

Per evitare tocchi multipli, il metodo rimuove il listener dell'immagine una volta gestito il tocco e aggiorna uno stato interno che segnala l'interazione. Inoltre, se era stato impostato un timeout per la risposta, questo viene cancellato per evitare che si attivi dopo l'interazione. Infine, il metodo restituisce `true` per indicare che l'evento è stato gestito correttamente.

```

private boolean gestisciTocco(MotionEvent event, int domandaIndex) {
    // Ottieni le coordinate del tocco (x, y)
    float x = event.getX();
    float y = event.getY();

    ImageView imageView = findViewById(R.id.imageView2);
    int width = imageView.getWidth();
    int height = imageView.getHeight();

    int areaToccata = -1;
    if (x < width / 2 && y < height / 2) {
        areaToccata = 1; // Area 1 (in alto a sinistra)
    } else if (x >= width / 2 && y < height / 2) {
        areaToccata = 2; // Area 2 (in alto a destra)
    } else if (x < width / 2 && y >= height / 2) {
        areaToccata = 3; // Area 3 (in basso a sinistra)
    } else if (x >= width / 2 && y >= height / 2) {
        areaToccata = 4; // Area 4 (in basso a destra)
    }

    Log.d("TrogActivity", "msg: " + areaToccata);

    // Controlla se la risposta è corretta
    if (areaToccata == risposteCorrette[domandaIndex]) {
        CounterSingleton.getInstance().incrementaPunteggio(); // Incrementa il punteggio

        gestisciRisposteCorrette();
        // Fai parlare Pepper per incoraggiare l'utente
        if (pepperController != null) {
            pepperController.incoraggiaUtente();
        }
    } else {
        gestisciRisposteSbagliate();
    }

    // Rimuove il listener per evitare tocchi multipli
    imageView.setOnTouchListener(null);
    // Segna che l'utente ha interagito con l'immagine
    immagineToccata = true;

    // Annulla il timeout se c'è stata interazione
    if (timeoutRunnable != null) {
        handler.removeCallbacks(timeoutRunnable); // Rimuove il runnable che suonerebbe
    }

    return true;
}

```

10.7 change

Il metodo `change` gestisce il passaggio da una domanda all'altra all'interno del test cognitivo. All'inizio, resetta la variabile che indica se l'immagine è stata toccata e avvia un timer di 30 secondi per la risposta tramite la funzione `attendi30Secondi()`.

Successivamente, ottiene l'istanza del singleton che tiene traccia del conteggio delle domande e ne incrementa il valore. In base al numero corrente della domanda, costruisce dinamicamente il nome della risorsa immagine da caricare, tenendo conto di un formato che prevede un prefisso e un

numero con eventuale zero iniziale.

Se il contatore è inferiore o uguale a 30 (numero totale delle domande), il metodo aggiorna l'immagine visualizzata nell'`ImageView` corrispondente, recuperando la risorsa appropriata dalle risorse drawable. Poi, tramite Text-to-Speech, fa pronunciare al robot Pepper la domanda associata all'immagine corrente. Infine, imposta un listener per gestire il tocco sull'immagine, collegandolo al metodo `gestisciTocco` per verificare la risposta.

Se invece il contatore supera 30, significa che il test è terminato: il metodo azzera il contatore, cancella eventuali timer in corso, calcola il tempo totale impiegato dall'inizio alla fine del test, lo salva nel singleton e avvia una nuova attività per segnalare la conclusione del test.

```
// Metodo chiamato per cambiare immagine e leggere la domanda successiva
public void change(View view) {
    immagineToccata = false;
    attendi30Secondi();

    CounterSingleton contatore = CounterSingleton.getInstance();
    contatore.incrementaCont();
    int cont = contatore.getCont();
    String stringacont = Integer.toString(cont);
    String temp;
    if (cont < 10) {
        temp = "pic0" + stringacont;
    } else {
        temp = "pic" + stringacont;
    }

    if (contatore.getCont() <= 30) {
        ImageView imageView = findViewById(R.id.imageView2);
        int idImmagine = getResources().getIdentifier(temp, "drawable", getPackageName());
        imageView.setImageResource(idImmagine);
        // Legge la domanda corrispondente tramite TTS
        String domanda = domande[cont - 1]; // le domande partono da indice 0
        if (pepperController != null) {
            pepperController.speak(domanda); // Fa parlare Pepper
        }
        // Imposta il listener per il tocco
        imageView.setOnTouchListener((v, event) -> gestisciTocco(event, domandaIndex: cont - 1));
    } else {
        contatore.setCount(0);

        if (timeoutRunnable != null) {
            handler.removeCallbacks(timeoutRunnable);
            timeoutRunnable = null;
        }
        long endTime = System.currentTimeMillis();
        int tempoImpiegato = (int) (endTime - startTime); // Tempo in millisecondi
        contatore.setTempoImpiegatoFineTest(tempoImpiegato);
        Intent intent = new Intent(packageContext, TrogActivity.this, TestTerminatoActivity.class);
        startActivity(intent);
    }
}
```

10.8 configuraGrafico

Un esempio di configurazione di uno dei grafici presenti

```
private void configuraGrafico(ArrayList<BarEntry> barEntries) {
    BarDataSet dataSet = new BarDataSet(barEntries, label: "Risultati Test");
    dataSet.setColors(ColorTemplate.MATERIAL_COLORS);
    dataSet.setValueTextSize(10f);

    BarData barData = new BarData(dataSet);
    barData.setBarWidth(0.5f);

    barChart.setData(barData);
    barChart.setFitBars(true);
    barChart.setViewportOffsets(left: 20f, top: 20f, right: 20f, bottom: 20f);
    barChart.invalidate();

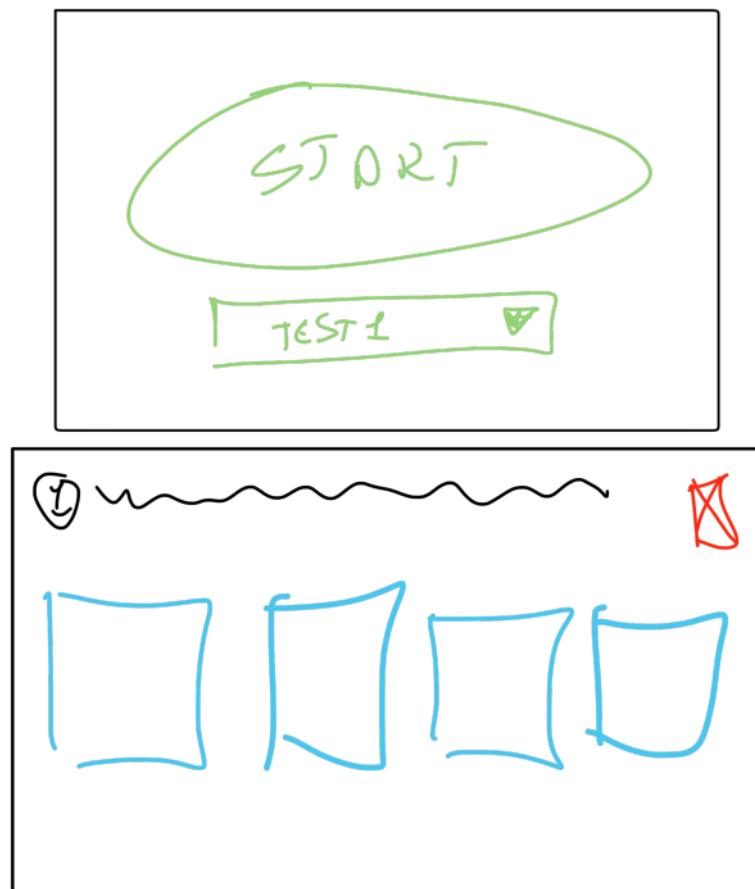
    // Configura l'asse X con etichette
    final String[] xLabels = {"Risposte Esatte", "Risposte Sbagliate", "Distrazioni"};
    XAxis xAxis = barChart.getXAxis();
    xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);
    xAxis.setGranularity(1f);
    xAxis.setTextSize(10f);
    xAxis.setValueFormatter(getFormattedValue(value) -> {
        int index = (int) value;
        if (index >= 0 && index < xLabels.length) {
            return xLabels[index];
        } else {
            return "";
        }
    });

    // Configura l'asse Y
    YAxis yAxis = barChart.getAxisLeft();
    yAxis.setAxisMinimum(0f);
    yAxis.setTextSize(10f);
    barChart.getAxisRight().setEnabled(false);
}
```

Chapter 11

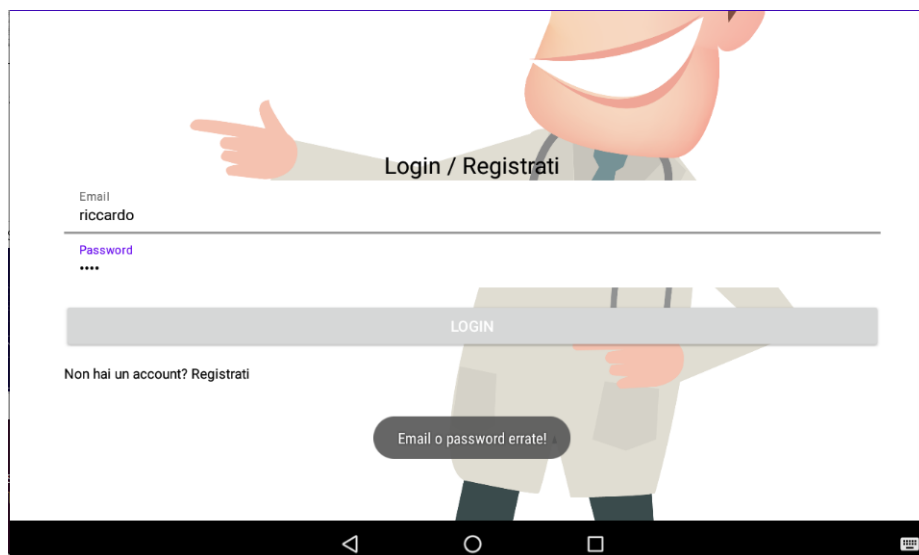
Bozza Interfaccia

Questo processo ha portato alla realizzazione di una prima bozza dell'interfaccia. L'obiettivo del prototipo è quello di offrire un'interfaccia utente intuitiva e di facile comprensione. Di seguito si riporta la versione preliminare dell'interfaccia.



Chapter 12

Mock up



A mobile app screen mockup for a login/registration page. At the top, a large cartoon character with a wide smile and a pointing hand is partially visible. The title "Login / Registrati" is centered. Below it, there are two input fields: "Email" with the text "riccardo" and "Password" with four dots. A grey "LOGIN" button is positioned below the password field. To the left of the button, the text "Non hai un account? Registrati" is displayed. A dark grey error message bubble at the bottom center says "Email o password errate!". The Android navigation bar is at the bottom.

Login / Registrati

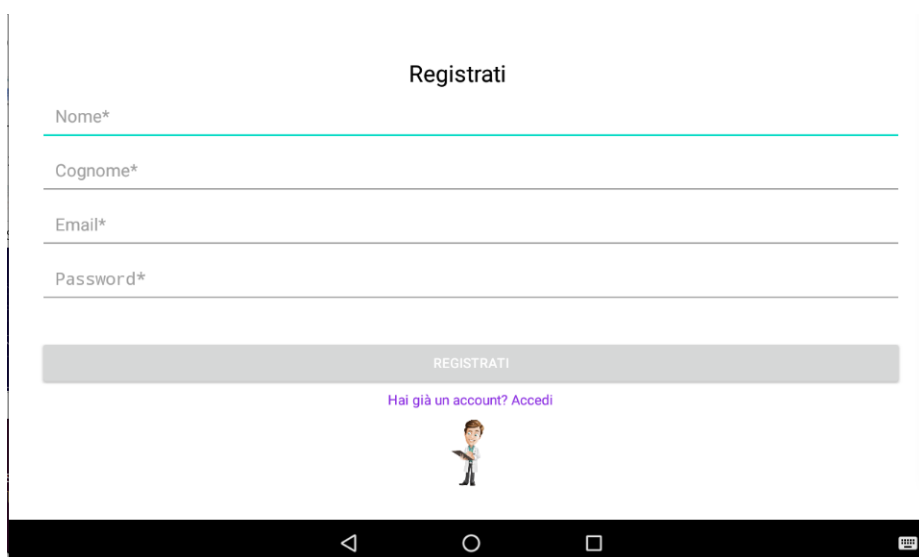
Email
riccardo

Password
....

LOGIN

Non hai un account? Registrati

Email o password errate!



A mobile app screen mockup for a registration page. The title "Registrati" is centered at the top. Below it are four input fields: "Nome*", "Cognome*", "Email*", and "Password*". A grey "REGISTRATI" button is centered below the fields. Below the button, the text "Hai già un account? Accedi" is displayed in purple. At the bottom center, there is a small cartoon character holding a smartphone. The Android navigation bar is at the bottom.

Registrati

Nome*

Cognome*

Email*

Password*

REGISTRATI

Hai già un account? Accedi

Registrati

riccardo

Cognome*

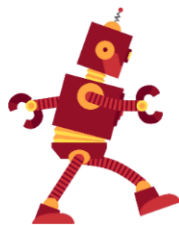
Email*

Password*

REGISTRATI

[Hai già un account? Accedi](#)

Tutti i campi sono obbligatori!



START

GESTIONE PAZIENTE



Aggiungi Paziente



Visualizza Paziente



dati del bambino

NOME*

COGNOME*

ETA*

SESSO*

SALVA



Scegli bambino



Riccardo Spinosa



Alessandro Massadoro



Seleziona un'opzione



Diagramma



Storico Test



Registrazioni Audio



Gestisci Note





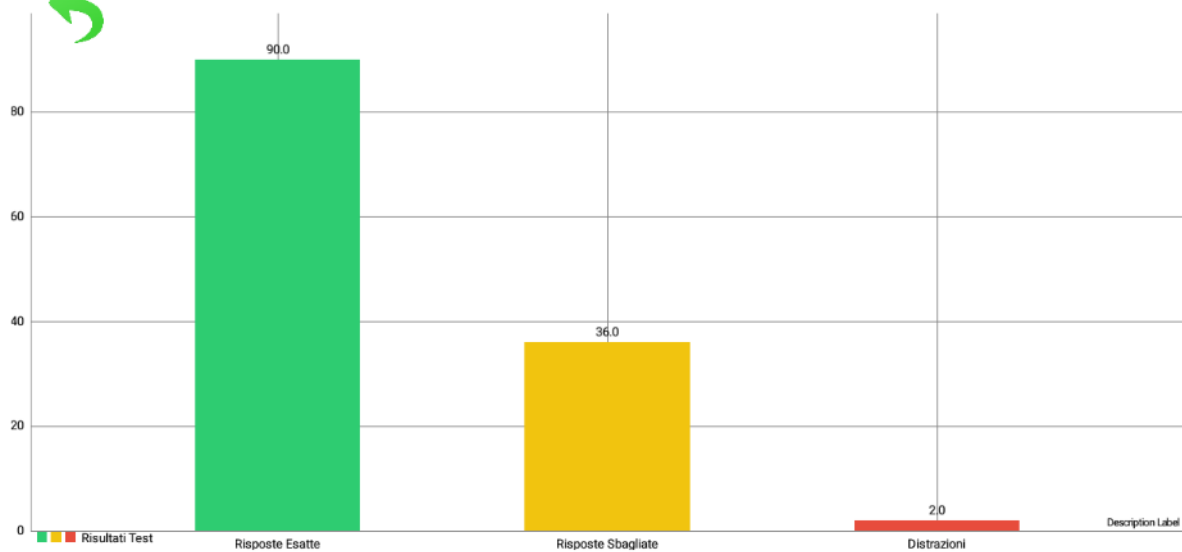
Grafico Barre

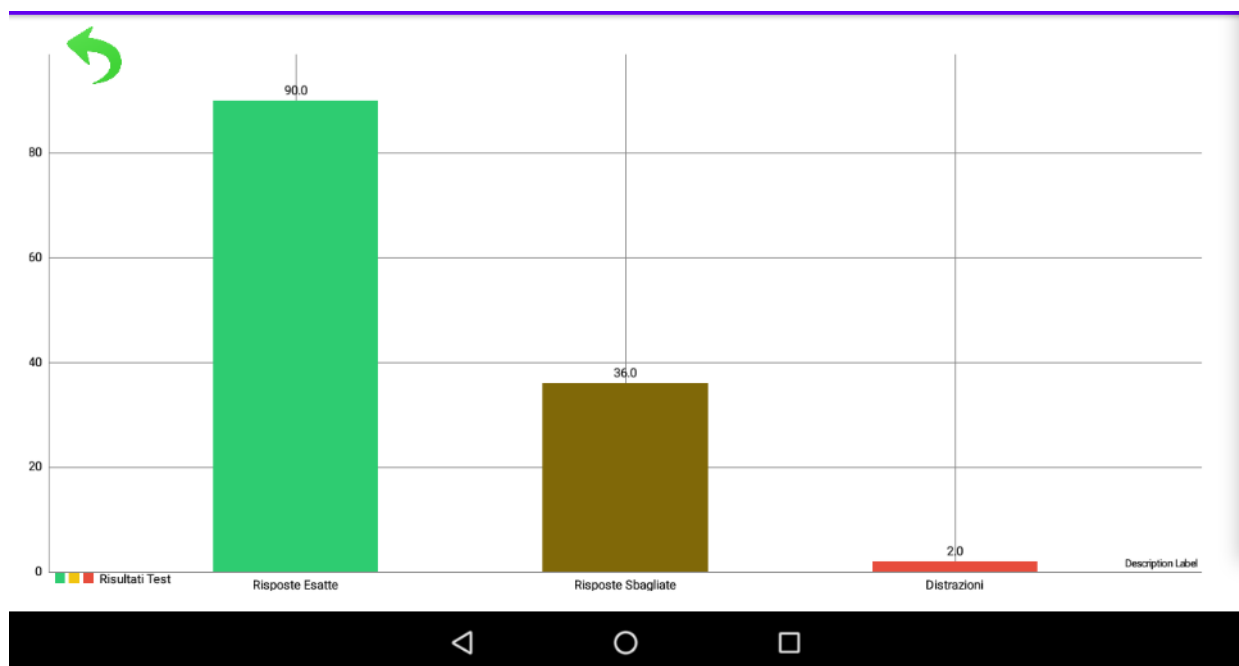
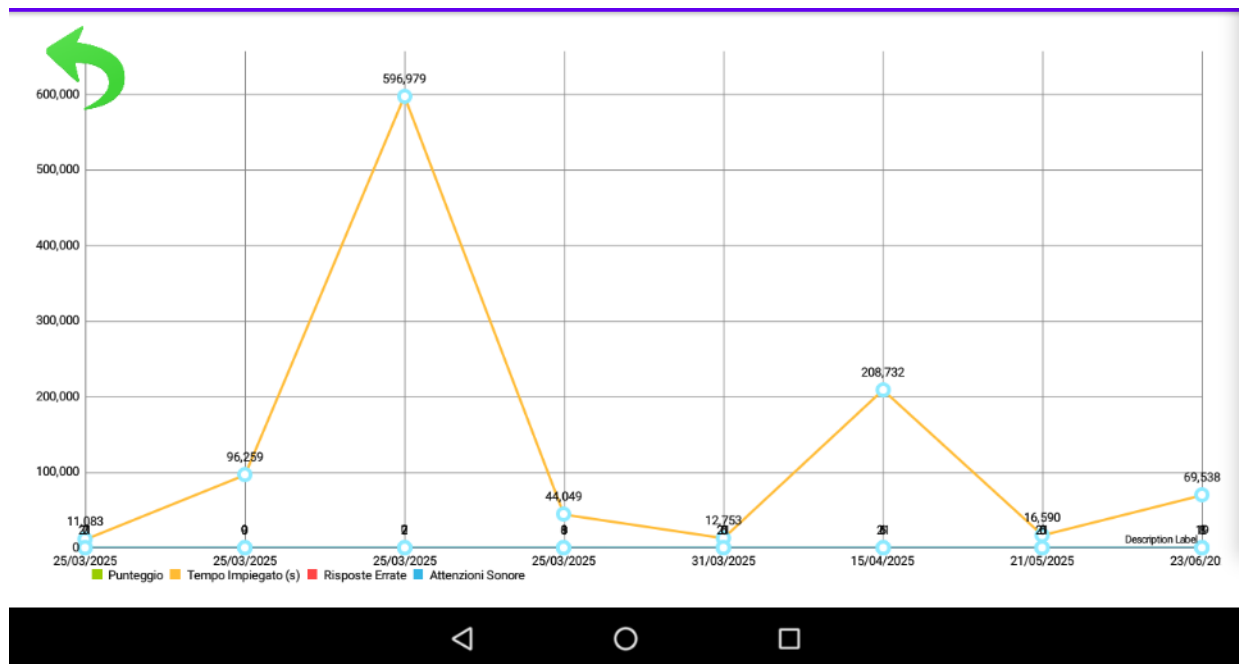



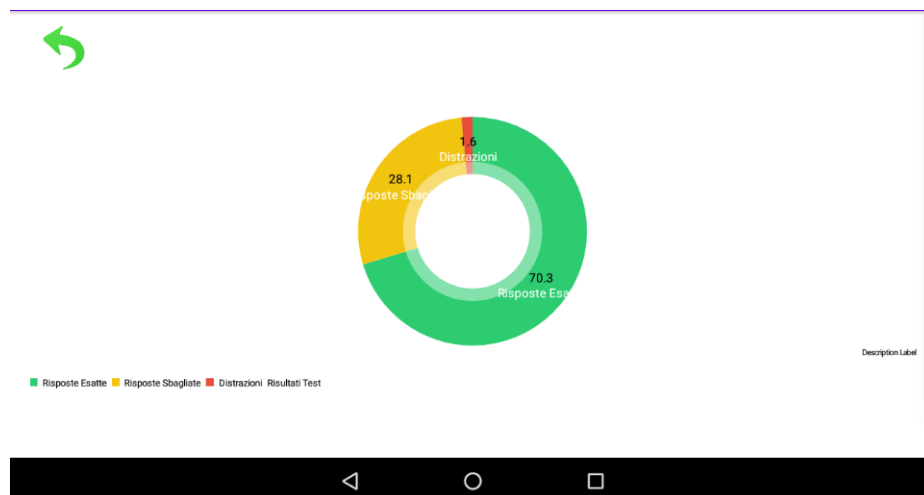
Grafico Temporale







Grafico Torta







Registrazioni delle Risposte

	registrazione_1742866761670.3gp	
	registrazione_1742867115094.3gp	

Dati del Paziente

Età: 25Test Avviato: T.R.O.G
Nome: Riccardo
Cognome: Spinosa
Data: 25/03/2025
Punteggio: 21
Tempo Impiegato (s): 11083
Numero Risposte Errate: 7
Numero Attenzioni Sonore: 0

Test Avviato: CLOZE TEST
Nome: Riccardo
Cognome: Spinosa
Data: 25/03/2025
Punteggio: 0
Tempo Impiegato (s): 96259
Numero Risposte Errate: 9
Numero Attenzioni Sonore: 0

Test Avviato: CLOZE TEST
Nome: Riccardo

CHIUDI



Scegli un'opzione

SCRIVI UNA NOTA

VISUALIZZA LE NOTE



Dati del Paziente

Data: 24/03/2025

Nota: Il paziente ha migliorato nella comprensione delle domande. Tuttavia, ha avuto difficoltà a mantenere l'attenzione per tutta la durata del test. Si necessitano esami approfonditi e la ripetizione dei test in un ambiente con meno distrazioni.

Data: 24/03/2025

Nota: Questa Nota solo un esempio quindi non carci valore

CHIUDI

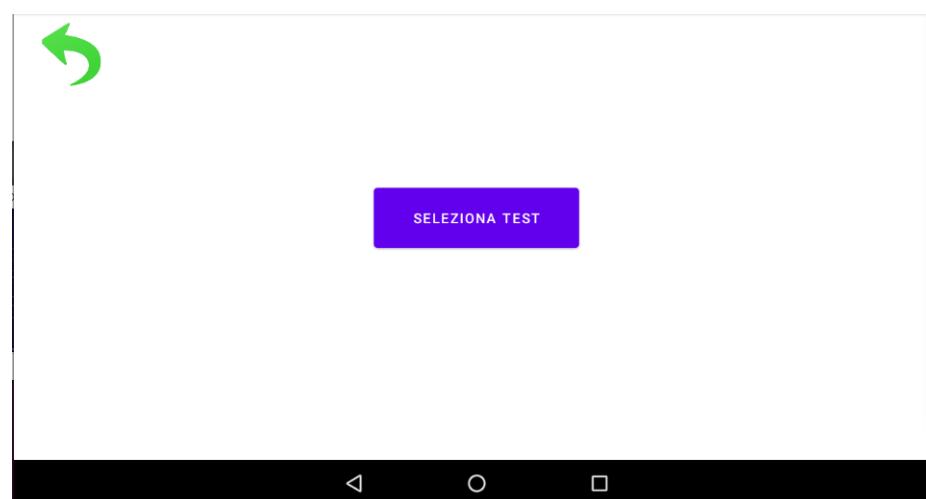
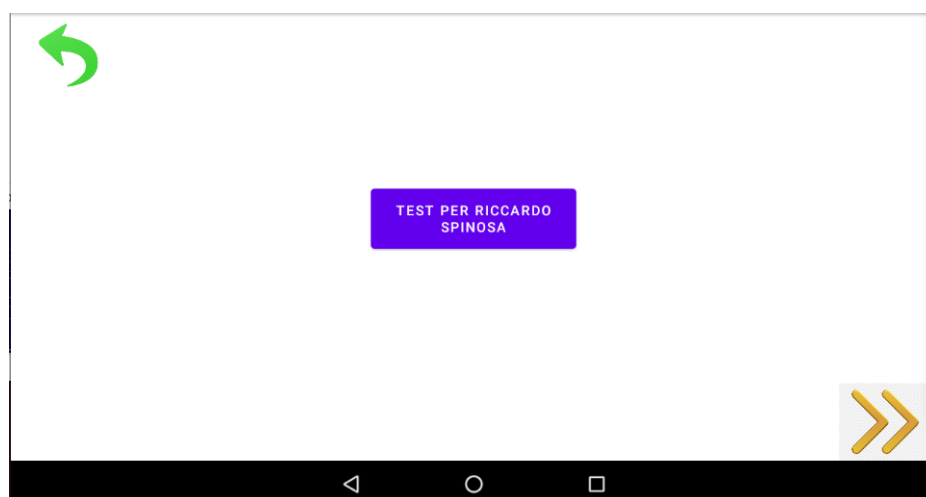
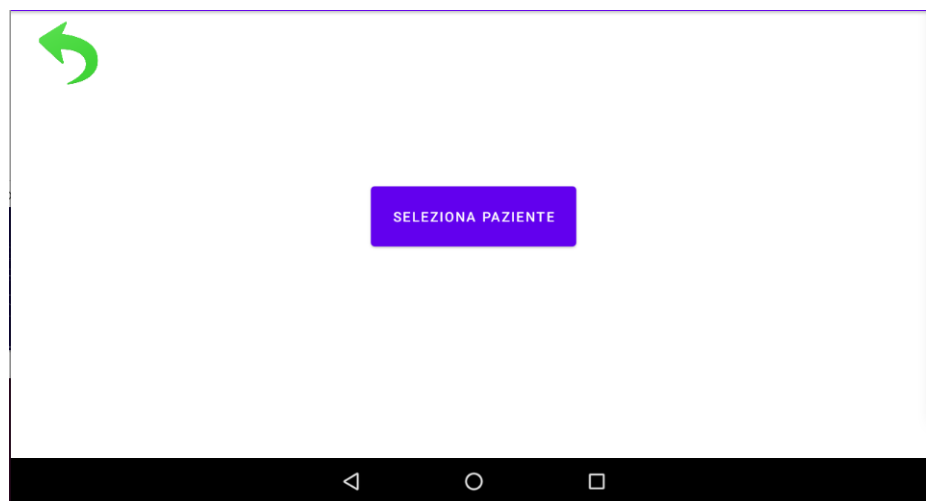


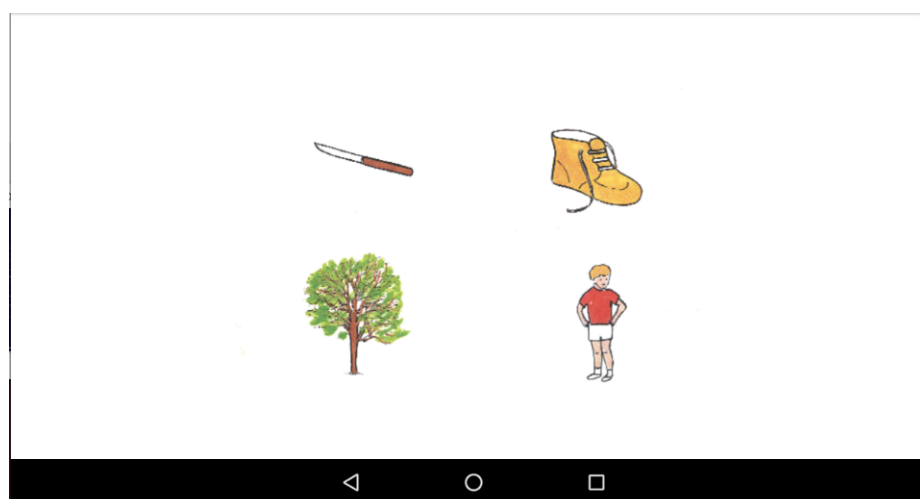
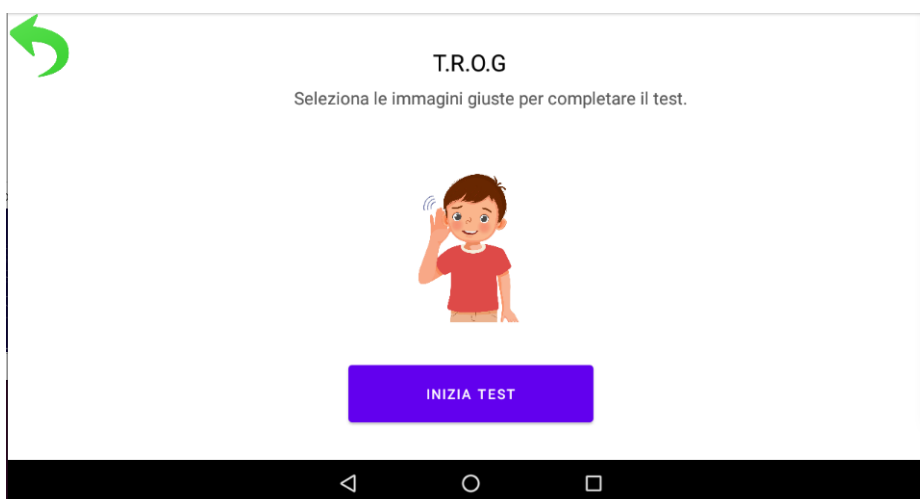
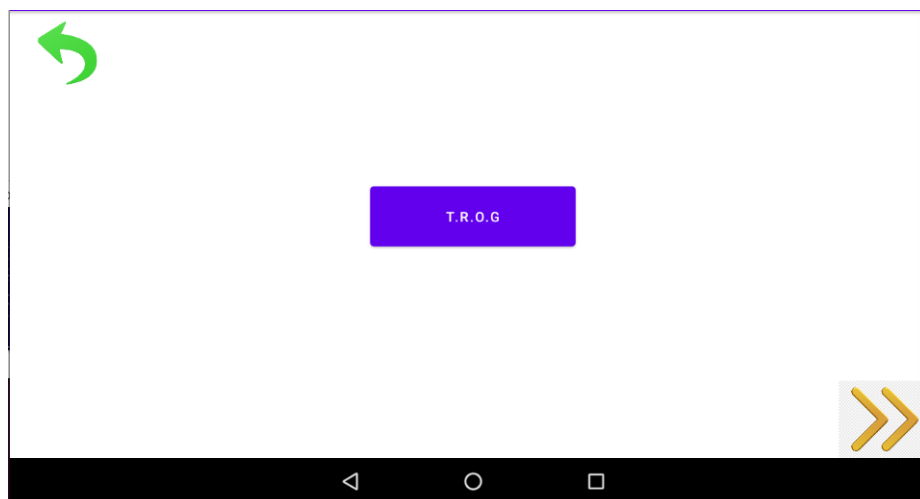
Scrivi Nota

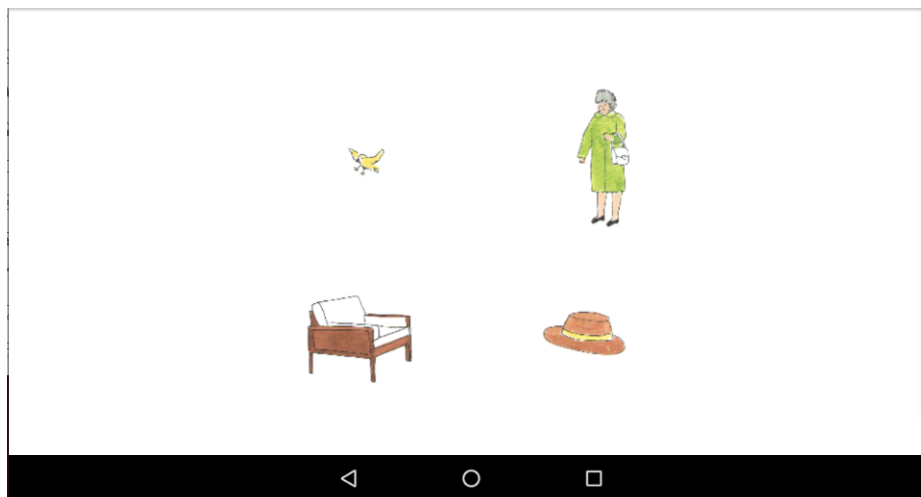
Scrivi una nota

SALVA NOTA









FUNTESSIO

19

FINE



test di completamento del testo

Ascolta e completa le frasi.



INIZIA TEST



AVANTI



Teoria della mente

ascolta il video sul tablet e rispondi alla domanda



INIZIA TEST

DOVE CERCHERÀ SALLY LA SUA PALLA?



InShot

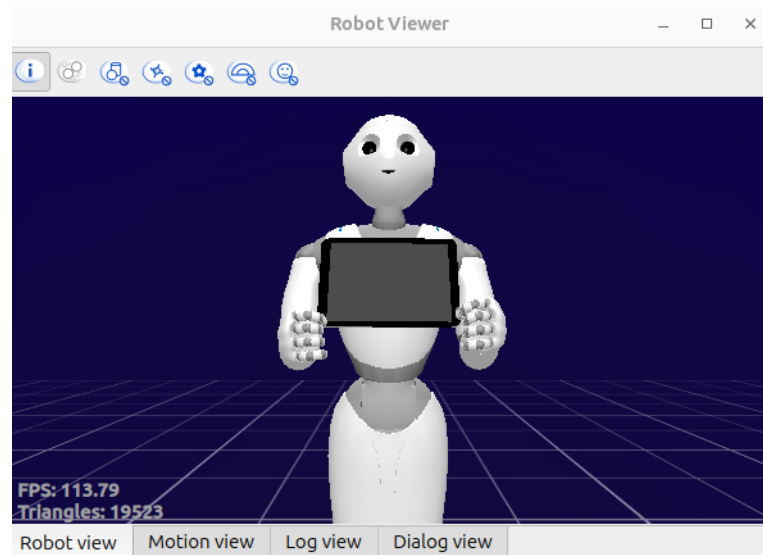
Chapter 13

Animazioni del robot Pepper

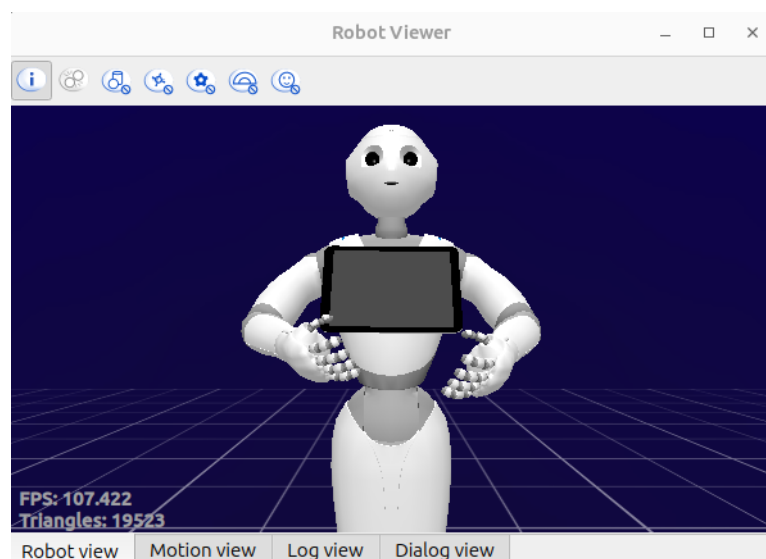
Il robot Pepper è dotato della possibilità di implementare animazioni corporee espressive, fondamentali per rafforzare l'interazione uomo-macchina e migliorare l'ingaggio emotivo del paziente. Tali animazioni sono utilizzate strategicamente all'interno del sistema per comunicare stati emotivi, reagire ai comportamenti del paziente e interagire con il medico/paziente in modo naturale. Le animazioni, sincronizzate con i dialoghi e le risposte del robot, permettono a Pepper di apparire più umano e coinvolgente, contribuendo al successo del protocollo terapeutico o diagnostico.

Di seguito si riportano tre esempi rappresentativi:

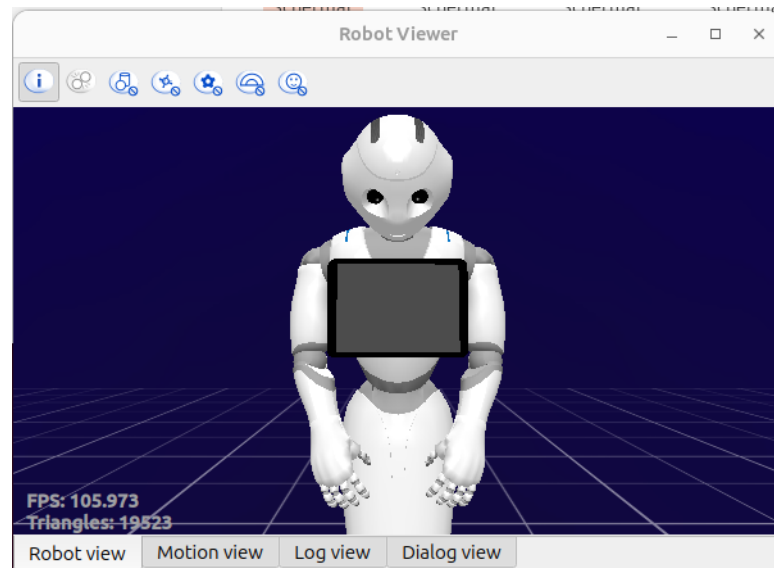
- **Eccitazione:** quando il paziente fornisce una serie di risposte corrette, Pepper esegue un'animazione di gioia ed entusiasmo, sollevando le braccia e mostrando un comportamento vivace e incoraggiante.



- **Respiro:** nel caso in cui il paziente commetta diversi errori consecutivi, Pepper esegue un'animazione di respirazione profonda, simulando calma e pazienza, con movimenti regolari del busto e delle braccia, per trasmettere tranquillità.



- **Saluto:** all'inizio della sessione, Pepper esegue un inchino cerimoniale per salutare il medico, pronunciandone il nome e instaurando un contatto rispettoso e personalizzato.



Chapter 14

Risultati e Valutazione

14.1 Valutazioni

Euristiche di usabilità

Visibilità dello stato del sistema L'applicazione fornisce un feedback costante all'utente: durante l'erogazione dei test, Pepper comunica l'avanzamento delle domande, mentre l'interfaccia Android mostra chiaramente lo stato corrente (test in corso, in attesa di risposta, fine test). Nella dashboard del medico, ogni azione (aggiunta paziente, salvataggio nota, avvio test) restituisce un messaggio visivo o auditivo.

Corrispondenza fra il mondo reale e il sistema Il linguaggio utilizzato nell'interfaccia è semplice, orientato all'utente clinico. I test sono chiamati con il loro nome scientifico (TROG, Sally-Anne, Cloze), ma accompagnati da descrizioni comprensibili. Le interazioni vocali di Pepper sono progettate per essere comprensibili da bambini con disturbi dello spettro autistico.

Controllo e libertà dell'utente I medici possono tornare indietro in ogni sezione, modificare i dati inseriti o annullare operazioni (es. rimozione nota, selezione test). Le azioni critiche richiedono conferma esplicita da

parte dell'utente.

Consistenza e standard L'interfaccia segue gli standard delle app Android: uso di pulsanti, icone, colori e navigazione a tab. L'organizzazione delle funzionalità è coerente in tutta l'app, con uno stile visivo uniforme e comprensibile.

Prevenzione degli errori I campi di input (nome, email, note) sono controllati per evitare caratteri non validi. Le scelte disponibili sono sempre limitate al contesto corrente (es. solo test disponibili, solo pazienti già registrati), riducendo la possibilità di errore.

Riconoscere piuttosto che ricordare Il medico non deve memorizzare nulla: i nomi dei pazienti, test, registrazioni e note sono sempre accessibili. Icone e descrizioni intuitive guidano l'uso dell'interfaccia. I dati sono organizzati per paziente e facilmente esplorabili.

Flessibilità ed efficienza d'uso Nonostante il sistema sia progettato per semplicità, sono presenti funzioni che migliorano l'efficienza: ad esempio, la possibilità di ascoltare direttamente le registrazioni, visualizzare lo storico o accedere rapidamente alle note cliniche.

Design estetico e minimalista L'interfaccia è priva di elementi superflui. L'attenzione è posta su ciò che è essenziale: nome del paziente, test selezionato, risultati. Anche le interazioni di Pepper sono progettate per essere semplici, dirette e coerenti con l'obiettivo terapeutico.

Aiuto al riconoscimento, diagnosi e recupero dagli errori In caso di errore (file mancante, risposta non registrata, input non valido), l'app mostra un messaggio chiaro e indica l'azione correttiva da compiere. Ad esempio,

se il medico dimentica di selezionare un paziente, viene avvisato con un messaggio e un suggerimento.

Guida e documentazione L'interfaccia è progettata per essere intuitiva, senza bisogno di manuali. Tuttavia, alcune sezioni presentano una breve spiegazione dello scopo della funzionalità

Chapter 15

Obiettivi raggiunti

L'obiettivo principale del progetto era realizzare un sistema interattivo basato su robot per la somministrazione di test cognitivi a bambini con disturbo dello spettro autistico. A tal fine, è stata sviluppata un'applicazione Android che comunica con il robot umanoide Pepper, guidando il bambino nello svolgimento di test linguistici e di teoria della mente, raccogliendo le risposte e fornendo uno strumento di analisi utile per il personale medico.

Gli obiettivi sono stati pienamente raggiunti:

- È stato realizzato un sistema funzionante e stabile, capace di avviare i test, raccogliere dati, e coinvolgere il bambino attraverso interazioni vocali e animazioni;
- I risultati dei test vengono salvati in file di testo all'interno della memoria interna del tablet Android, senza l'utilizzo di un database;
- L'interfaccia utente per il medico è risultata semplice e funzionale;

Chapter 16

Verifica funzionale

Il sistema è stato sottoposto a test funzionali con simulazioni di sessioni reali. Ogni funzionalità è stata verificata singolarmente e in contesto:

- **Comunicazione robot–app:** Il collegamento tra Pepper e l'app è avvenuto correttamente tramite la piattaforma NAOqi. I comandi vocali e le animazioni sono stati eseguiti secondo la logica definita.
- **Somministrazione dei test:** I tre test implementati (TROG, Cloze Test, Sally-Anne) sono stati somministrati in modo fluido. Le interazioni risultano comprensibili e il robot riesce a mantenere l'attenzione del bambino.
- **Registrazione e archiviazione dei dati:** Le risposte vengono salvate correttamente in file di testo strutturati nella memoria del tablet. Ogni file è identificato dal nome del paziente e dal tipo di test effettuato.
- **Gestione dell'esperienza utente:** In caso di risposta errata o mancata risposta, Pepper reagisce con animazioni di supporto, suoni o suggerimenti vocali per calmare e motivare il bambino.

Chapter 17

Feedback qualitativo

Non potendo coinvolgere bambini reali per ragioni etiche, l'applicazione è stata mostrata ad un pubblico variegato. I feedback raccolti sono stati positivi:

- L'utilizzo di Pepper come mediatore ha suscitato interesse per il suo potenziale nel migliorare l'engagement dei bambini;
- La semplicità della gestione dei test e la possibilità di visualizzare le registrazioni sono stati considerati molto utili in ambito terapeutico;
- L'approccio basato su file locali è stato apprezzato per la sua trasparenza e per la facilità di accesso ai dati senza infrastrutture complesse.

Sono emersi anche alcuni spunti di miglioramento:

- L'assenza di un sistema di backup automatico espone i dati al rischio di perdita in caso di danneggiamento del dispositivo;
- L'interazione vocale, pur essendo efficace, è suscettibile a interferenze da rumori ambientali o a variazioni nella voce dei bambini.

Chapter 18

Limiti e miglioramenti futuri

Limiti attuali

- Il salvataggio in file locali non consente una gestione centralizzata o multi-dispositivo dei dati;
- Il sistema non è ancora in grado di effettuare analisi automatiche avanzate sulle risposte raccolte;
- Le funzionalità vocali dipendono dalla qualità del microfono e dalle condizioni ambientali.

Miglioramenti futuri possibili

- Integrazione di una funzionalità di sincronizzazione remota, ad esempio tramite cloud o server locale, con autenticazione e crittografia;
- Aggiunta di una dashboard web per la visualizzazione e analisi dei dati da parte del medico;
- Estensione dell'app con altri tipi di test, anche adattivi, e con una maggiore personalizzazione;

- Introduzione di un sistema di logging audio/video per le sessioni, utile per la valutazione clinica.
- Sebbene l'interfaccia per la selezione del paziente e del test risulti semplice e intuitiva, potrebbe essere resa più accattivante e coinvolgente dal punto di vista visivo.

Sono state inoltre integrate strategie per la gestione degli errori, delle distrazioni e della motivazione del bambino.

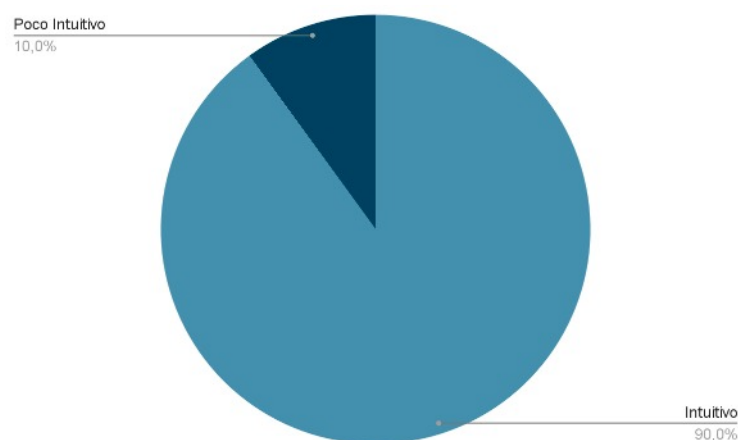
Chapter 19

Metodologia di Testing

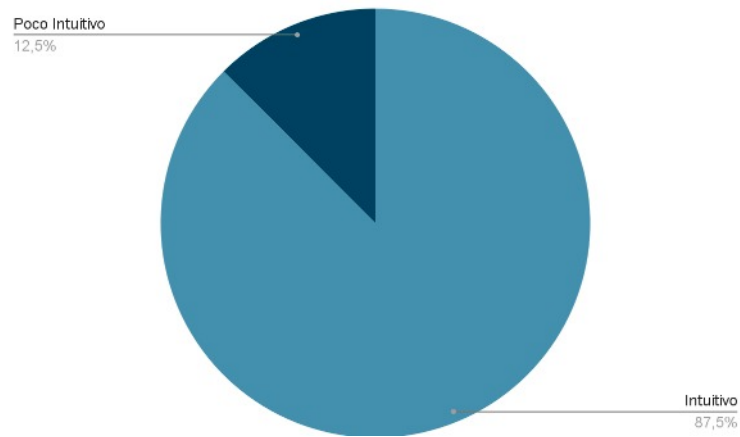
Per la fase di testing dell'applicazione, abbiamo coinvolto un campione di 4 utenti. Ad ognuno di essi sono stati assegnati 5 task specifici da eseguire. Prima di iniziare, ogni partecipante è stato brevemente informato sull'obiettivo e sul funzionamento generale dell'applicazione. Successivamente, gli è stato chiesto di valutare la difficoltà percepita per ciascun task.

Task Assegnati

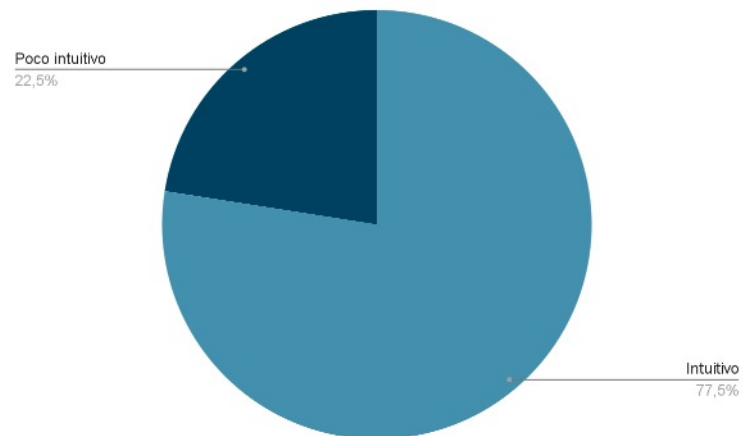
1. Prova a registrarti all'applicazione



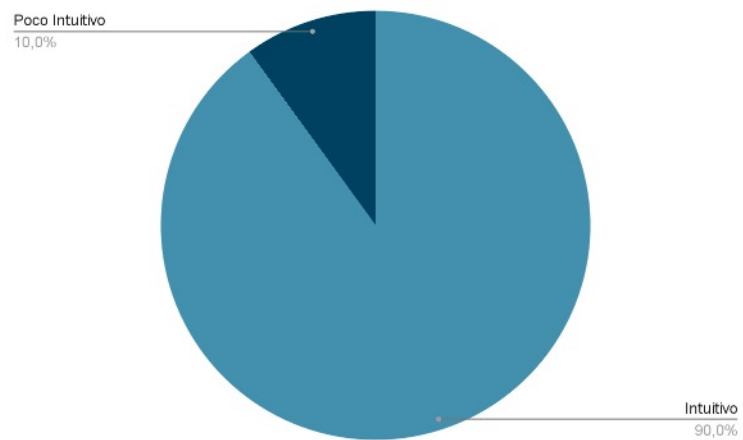
2. Inserisci una nota



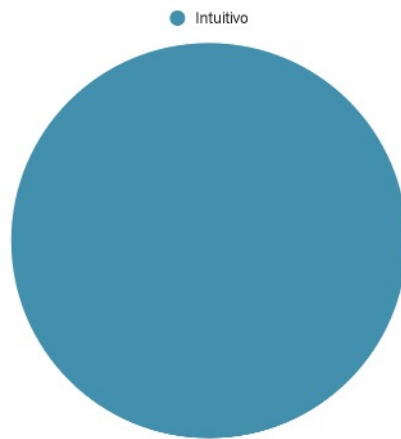
3. Prova ad ascoltare una registrazione



4. Inserisci un paziente o visualizzalo



5. Fai partire un test



I test effettuati ci hanno permesso di verificare concretamente l'usabilità del sistema, assicurandoci che tutte le funzionalità siano accessibili e funzionino correttamente nel contesto previsto. Grazie a questi controlli, abbiamo potuto identificare eventuali criticità o difficoltà d'uso, intervenendo tempestivamente per migliorare l'esperienza dell'utente. In definitiva, i test hanno confermato che il sistema è stabile, intuitivo e pronto per essere utilizzato efficacemente nel suo ambito di applicazione.

Chapter 20

Test di Performance

Di seguito si riportano i tempi di risposta misurati per le principali azioni dell'applicazione. Si indica con **TA** il tempo di accesso al filesystem, variabile in base al sistema e alle condizioni di carico:

- Apertura dell'applicazione: **2 secondi**
- Registrazione al sistema: **0.5 secondi + TA**
- Login: **0.5 secondi + TA**
- Restituzione dati errati: **0.5 secondi + TA**
- Accesso alla vista “Home”: **0.5 secondi**
- Switch tra le varie viste: **0.2 secondi**
- Visualizzazione grafici: **0.3 secondi + TA**
- Inserimento paziente: **0.3 secondi + TA**

Chapter 21

Analisi di Mercato (3.0)

Attualmente non risulta presente alcun competitor diretto. Per questo motivo l'analisi di mercato non è stata approfondita ulteriormente. Tuttavia, si evidenzia che il nostro prototipo risulta essere **innovativo**, trattandosi di una soluzione robotica per l'erogazione di test diagnostici e cognitivi a bambini con disturbi dello spettro autistico, un settore ancora in fase di evoluzione.

Ringraziamenti

Voglio ora ringraziare tutti coloro che mi hanno supportato durante questo cammino.

Ringrazio intensamente la mia ragazza, Chiara, che è stata linfa vitale e un sostegno prezioso. La ringrazio per la pazienza avuta con me in questi tanti anni: ha reso i momenti difficili superabili nonostante il mio caratteraccio.

Ringrazio mio fratello Alessandro e Luigi: Alessandro per aver affrontato insieme i problemi avuti; Luigi per avermi supportato lungo il mio percorso e iscritto al primo anno.

Ringrazio mia sorella, per avermi dato forza e coraggio.

Ringrazio mio fratello Lorenzo, per la sua capacità di trasformare qualsiasi cosa in qualcosa di allegro.

Ringrazio mia madre, per avermi aiutato alleggerendo le pressioni in quest'ultimo anno universitario.

Ringrazio il mio collega ed amico Alessandro: affrontare la maggior parte degli esami insieme è stato entusiasmante. La sua presenza è stata di fondamentale supporto.

Ringrazio Gabriele, che condividendo la stessa passione per l'informatica è stato fonte di sicurezza.

Grazie alla prof.ssa Mariacarla Staffa, che ha guidato e supportato il mio lavoro. Il suo corso ha indirizzato il mio prossimo futuro.

Un grazie speciale ai miei amici Vincenzo Tafuri, Vincenzo Curcio, Emanuele De Rosa, Salvatore Marrano.

Ringrazio inoltre tutti coloro che, in modo diretto o indiretto, hanno contribuito al mio percorso formativo.