COMUNICATORE PECS

Sommario

CENNI TEORICI *4*

METODO PECS **4**

CAA Comunicazione Aumentativa Alternativa **5**

ABA Applied Behaviour Analysis **6**

ANDROID STUDIO **7**

IL PROGETTO *10*

COS'È **10**

IL CODICE **10**

MAIN ACTIVITY 10
CUSTOM ADAPTER 19
GRID ACTIVITY 22
GRID ADAPTER 29
CLASSE PECS 32
MAIN ACTIVITY LAYOUT (CUSTOM_LAYOUT + ACTIVITY_MAIN) 32
GRID ACTIVITY LAYOUT (CUCSTOM_LAYGRID + ACTIVITY_GRID) 35
LAYOUT CONDIVISO (SHARED_BOX_BUTTONS) 38

CENNI TEORICI

METODO PECS

PECS è l'acronimo di Picture Exchange Communication System. Si tratta di un metodo che ha l'obiettivo di permettere alle persone con scarse capacità di comunicazione di poter comunicare grazie a delle immagini.

Alle persone che usano PECS viene insegnato ad avvicinarsi ad un'altra persona e a dare un'immagine di un oggetto desiderato in cambio dell'oggetto stesso. In questo modo, la persona è spronata ad iniziare delle comunicazioni. Si inizia con lo scambio di icone semplici ma si punta a costruire rapidamente la struttura della "frase".

Un bambino con disturbi dello spettro autistico può utilizzare il metodo PECS a casa o in classe per comunicare una richiesta, un pensiero o qualsiasi altra cosa possa essere trasformata in un simbolo su cartolina illustrata.

Il sistema PECS è considerato un tipo di CAA Comunicazione Aumentativa Alternativa a bassa tecnologia: è così utilizzato che spesso la CAA viene ricondotta al solo PECS.

Le strategie didattiche utilizzate sono interamente basate su principi dell'ABA (Applied Behavior Analysis), che vanno dall'uso di rinforzi motivanti per lo studente a strategie efficaci per la correzione di errori, a formati di lezione vari (discreti, sequenziali) legati al tipo di abilita' che viene insegnata nelle varie fasi del protocollo.

Il metodo PECS è stato sviluppato nel 1984 da Lori Frost e dal Dr. Andrew Bondy. Venne usato per la prima volta nel programma autistico del Delaware. Inizialmente alcune persone si opposero all'uso del PECS e del linguaggio dei segni per insegnare ai bambini con autismo a comunicare. Sostenevano che questi metodi avrebbero danneggiato lo sviluppo della lingua parlata. Tuttavia diversi studi hanno dimostrato che aiuta a sviluppare un linguaggio verbale.

Chi può comunicare tramite metodo PECS?

La formazione PECS non è limitata dall'età ma piuttosto da un piccolo gruppo di criteri:

Il candidato alla formazione PECS dovrebbe essere un comunicatore intenzionale. Deve essere consapevole della necessità di comunicare il proprio messaggio a qualcuno, anche se in modo limitato. Ad esempio un bambino che trascina qualcuno attraverso la stanza fino alla posizione di un oggetto che desidera avere, ha almeno una nozione di intenzionalità.

In secondo luogo, l'individuo dovrebbe avere alcune preferenze personali. Se non si hanno preferenze o sono deboli, allora potrebbe essere più difficile capire il potere di una comunicazione alternativa efficace. Potrebbe essere necessario sviluppare un repertorio di preferenze e antipatie attraverso tentativi ed errori a vari tipi di cibo, oggetti o attività.

La capacità di discriminazione delle immagini non è un prerequisito. Certo, coloro che hanno capacità di discriminazione possono fare progressi più rapidi nelle fasi iniziali del programma.

Inotre alcuni individui potrebbero dimostrare spontaneamente di sapere già come utilizzare le immagini per comunicare.

Sebbene la strategia PECS sia utilizzata principalmente con individui non verbali, potrebbe essere utilizzata con individui che hanno solo un piccolo insieme di parole o segni significativi nel loro repertorio.

Come funziona PECS

Il metodo PECS si basa su sei fasi di apprendimento:

1. Scambio di immagini: L'utente impara a scambiare un'immagine con un oggetto desiderato. Ad esempio, se un bambino vuole un giocattolo, consegna l'immagine del giocattolo all'insegnante per ottenerlo.

2. Incremento della distanza e della persistenza: L'utente impara a cercare e portare l'immagine all'interlocutore, aumentando la distanza tra l'immagine e l'insegnante.

3. Discriminazione delle immagini: L'utente impara a scegliere tra due o più immagini per richiedere l'oggetto desiderato. Ad esempio, deve selezionare l'immagine del giocattolo preferito tra più immagini disponibili.

4. Struttura della frase: L'utente inizia a costruire semplici frasi utilizzando una striscia di frase (un supporto su cui vengono posizionate le immagini in sequenza). Ad esempio, potrebbe comporre la frase "Io voglio + immagine del giocattolo".

5. Rispondere a domande: L'utente impara a rispondere a domande come "Cosa vuoi?" utilizzando le immagini per costruire la risposta appropriata.

6. Commentare: L'utente impara a fare commenti utilizzando frasi come "Io vedo + immagine" o "Io sento + immagine", espandendo così la propria capacità comunicativa oltre le richieste.

CAA Comunicazione Aumentativa Alternativa

La Comunicazione Aumentativa Alternativa (CAA) è un approccio volto a fornire a persone con difficoltà comunicative complesse la possibilità di comunicare utilizzando canali alternativi al linguaggio orale.

La CAA comprende tutte le strategie, gli strumenti e le tecniche usate in ambito clinico e domestico per garantire la comunicazione a chi non può esprimersi verbalmente. La CAA non mira a sostituire il linguaggio verbale, ma a integrarlo. Si basa sulla presenza simultanea di uno strumento alternativo e del linguaggio verbale orale standard, supportato visivamente e oralmente dal partner comunicativo. Il simbolo utilizzato diventa un supporto visivo che accompagna lo stimolo verbale in entrata e, quando possibile, sostiene la produzione verbale in uscita. In questo modo, la Comunicazione Aumentativa non ostacola l’emergere del linguaggio verbale, ma lo potenzia.

La CAA non si limita all’uso di strumenti comunicativi. Uno degli strumenti più noti è il sistema di scrittura in simboli. Tuttavia, l’utilizzo di questi strumenti deve essere supportato da una

competenza comunicativa di base, che spesso non si sviluppa spontaneamente a causa di disturbi linguistici e cognitivi. Pertanto, la CAA non solo fornisce strumenti comunicativi alternativi, ma mira a sviluppare le abilità di comunicazione, come il desiderio di comunicare, la possibilità di comunicare con partner capaci e informati, e l'uso di strumenti adatti.

Per questo motivo, l'intervento di CAA non richiede prerequisiti, ma necessita di creare opportunità di comunicazione. Queste opportunità sono la base per sviluppare vari livelli di comunicazione, non solo per esprimere bisogni primari, ma anche per fare scelte, esprimere pensieri e desideri, e interagire socialmente. La CAA permette così di autodeterminarsi e agire sull'ambiente.

Esprimere il proprio pensiero o desiderio, anche in modo limitato, riduce l'angoscia e la frustrazione legate all'incapacità di farsi capire, diminuendo lo stress e i comportamenti problematici nelle persone con bisogni comunicativi complessi.

È essenziale che l'ambiente e i partner comunicativi siano accoglienti e informati, aderendo alla CAA. La famiglia, in particolare, deve passare da un ruolo marginale a uno centrale nel processo di riabilitazione comunicativa, come previsto dal Modello Family Centered di Rosenbaum (2004). Questo modello promuove un rapporto di reciproco scambio tra specialisti e famiglia, con compiti specifici e necessari da entrambe le parti. La CAA deve essere integrata nella vita quotidiana, coinvolgendo la famiglia, la scuola e altri ambienti sociali.

Per essere efficace, la CAA deve coinvolgere l'intero ambiente circostante, inclusi gli ambienti meno familiari come la scuola, i luoghi ricreativi e pubblici. Solo con la collaborazione di tutta la rete sociale la CAA può esprimere appieno il suo potenziale, migliorando la qualità della vita delle persone con bisogni comunicativi complessi.

ABA Applied Behaviour Analysis

L'ABA (Analisi Comportamentale Applicata) è una branca dell'analisi del comportamento che studia le relazioni tra il comportamento degli organismi e gli eventi che lo influenzano. In altre parole, come affermato da Cooper, Heron e Heward, l'ABA applica al comportamento umano i principi dell'analisi del comportamento per affrontare problemi rilevanti nella vita quotidiana. Uno degli obiettivi principali dell'ABA è dimostrare l'efficacia delle procedure usate per generare cambiamenti attraverso il metodo scientifico.

La prima applicazione dell'ABA con soggetti autistici risale al 1960 grazie a Lovaas, che avviò interventi per ridurre comportamenti problematici e migliorare la comunicazione. Da lì, si sono sviluppate numerose ricerche che hanno portato a un'applicazione sistematica e intensiva dei principi comportamentali, creando un modello di intervento molto efficace, noto come intervento comportamentale intensivo precoce.

I principi fondamentali dell'ABA si basano sulla teoria dell'apprendimento e del condizionamento operante. Il comportamento viene considerato operante perché agisce sull'ambiente per produrre conseguenze specifiche. Queste conseguenze modellano il comportamento influenzandone forma e frequenza. Il comportamento è analizzato in base agli stimoli ambientali precedenti (antecedenti) e alle risposte agli stimoli (conseguenze).

Concetti chiave collegati a questi principi includono rinforzo, estinzione, controllo degli stimoli e generalizzazione. Il rinforzo è qualsiasi conseguenza che aumenta la frequenza di un comportamento. Può essere negativo (evitare uno stimolo avversivo) o positivo (ottenere attenzione o accesso a un'attività). Quando il rinforzo non è più applicato, la probabilità di quel

comportamento diminuisce, un fenomeno noto come estinzione. Il controllo degli stimoli avviene quando un comportamento si verifica solo in presenza di uno specifico stimolo antecedente. La generalizzazione permette di trasferire l'apprendimento in diversi contesti e ambienti.

Questi concetti sono applicati attraverso quattro procedure principali):

Prompting: Fornire indizi o aiuti per ottenere un comportamento non ancora presente nel repertorio del bambino.

- Fading: Ridurre gradualmente e poi eliminare gli aiuti man mano che il bambino diventa più autonomo nel comportamento desiderato.
- Shaping: Rinforzare progressivamente le approssimazioni del comportamento desiderato.
- Chaining: Insegnare sequenze comportamentali complesse spezzandole in piccoli comportamenti.

Queste procedure, chiamate di cambiamento graduale, differiscono nei loro passi: nello shaping si rinforzano approssimazioni successive, nel fading si rinforza la risposta finale con approssimazioni sempre più vicine allo stimolo desiderato, nel chaining si rinforzano le connessioni stimolo-risposta.

Per raggiungere i comportamenti desiderati, si possono utilizzare due tipi di setting: prove discrete (DTT) e ambiente naturale (NET). Il DTT è strutturato per massimizzare le opportunità di apprendimento con ripetizione e rinforzo delle risposte corrette, ma può risultare difficile generalizzare il comportamento appreso. Il NET, invece, sfrutta situazioni quotidiane e interessi del bambino per favorire l'apprendimento, risultando utile per la generalizzazione ma limitato dalla motivazione del bambino.

L'ABA è particolarmente efficace nel trattare comportamenti problematici come stereotipie, autolesionismo, aggressività e capricci, che ostacolano l'apprendimento e il funzionamento quotidiano.

ANDROID STUDIO

Cos'è Android Studio?

Android Studio è l'ambiente di sviluppo integrato (IDE) ufficiale per lo sviluppo di applicazioni Android, supportato e sviluppato da Google. Lanciato ufficialmente nel 2013, Android Studio è basato su IntelliJ IDEA di JetBrains, un potente IDE Java, ed è progettato specificamente per fornire strumenti completi e specifici per la creazione di applicazioni Android.

Caratteristiche Principali

- Editor di Codice Avanzato:
    - Completamento del Codice: Android Studio offre un completamento del codice intelligente che suggerisce parole chiave, variabili e metodi appropriati mentre si digita.
    - Refactoring: Strumenti per modificare il codice senza cambiare il comportamento del programma.
    - Analisi del Codice: Include strumenti di analisi per individuare e correggere errori di codice, migliorare le prestazioni e assicurare conformità agli standard.

- Layout Editor:
    - Design Visuale: Permette agli sviluppatori di trascinare e rilasciare componenti dell'interfaccia utente per creare layout visivamente.
    - Anteprima in Tempo Reale: Mostra come appariranno i layout su una varietà di dispositivi e configurazioni.
    - Supporto per ConstraintLayout: Consente la creazione di layout complessi con una minore profondità della gerarchia di visualizzazione.

- Emulatore Android:
    - Test Rapidi: Permette di eseguire e testare le applicazioni su un dispositivo virtuale senza bisogno di un dispositivo fisico.
    - Varie Configurazioni: Gli sviluppatori possono configurare l'emulatore per imitare diverse versioni di Android e diversi tipi di dispositivi.

- Strumenti di Debugging:
    - Debug in Tempo Reale: Supporta il debug del codice in tempo reale, consentendo di mettere in pausa l'esecuzione, esaminare variabili e il flusso del programma.
    - Logcat: Strumento di log che mostra i messaggi di log del sistema Android, utili per individuare errori e problemi di prestazioni.

- Supporto per le Ultime Tecnologie:
    - Jetpack: Integrazione con le librerie Jetpack di Google, che forniscono componenti predefiniti per un rapido sviluppo.
    - Kotlin: Supporto nativo per il linguaggio di programmazione Kotlin, considerato il linguaggio preferito per lo sviluppo Android.

- Strumenti di Build e Distribuzione:
    - Gradle: Sistema di build automatizzato che gestisce la compilazione, il testing e la distribuzione delle applicazioni.
    - APK Analyzer: Strumento che aiuta a esaminare il contenuto di un APK per ottimizzazioni e riduzioni delle dimensioni.

Vantaggi di Android Studio

- Integrazione Completa: Fornisce un ambiente unificato che combina strumenti di sviluppo, testing e distribuzione.
- Aggiornamenti Frequenti: Google aggiorna regolarmente Android Studio con nuove funzionalità e miglioramenti per mantenere gli sviluppatori aggiornati con le ultime tendenze e tecnologie.
- Comunità e Supporto: Essendo l'IDE ufficiale, gode di un vasto supporto dalla comunità di sviluppatori e da Google, con ampia documentazione e risorse di apprendimento.

Svantaggi di Android Studio

- Consumo di Risorse:
    - Pesantezza: Android Studio è noto per essere molto esigente in termini di risorse di sistema. Richiede molta RAM e potenza di elaborazione, il che può rendere difficile l'utilizzo su computer meno potenti.
    - Lentezza: Su macchine con hardware meno avanzato, Android Studio può risultare lento, con tempi di avvio lunghi e prestazioni non ottimali durante il debugging e la compilazione.
- Curva di Apprendimento Ripida:
    - Complessità: Le numerose funzionalità e opzioni disponibili possono essere travolgenti per i nuovi sviluppatori. La curva di apprendimento può essere ripida, specialmente per chi è nuovo allo sviluppo Android.
- Problemi di Compatibilità:
    - Aggiornamenti: Occasionalmente, gli aggiornamenti di Android Studio possono introdurre bug o problemi di compatibilità con plugin o progetti esistenti, costringendo gli sviluppatori a risolvere questi problemi o a fare il downgrade alla versione precedente.
- Configurazione Iniziale:
    - Setup: La configurazione iniziale di Android Studio e dell'ambiente di sviluppo può essere complessa e richiedere tempo, specialmente per chi non ha familiarità con gli strumenti e le librerie necessarie.

Android Studio è uno strumento fondamentale per chiunque voglia sviluppare applicazioni Android. Con le sue potenti funzionalità, la facilità d'uso e il supporto per le ultime tecnologie, Android Studio facilita il processo di sviluppo, rendendolo più efficiente e accessibile. Tuttavia, gli sviluppatori devono essere consapevoli dei suoi svantaggi, come il consumo elevato di risorse e la complessità iniziale. Con il giusto hardware e una buona base di conoscenze, Android Studio rimane uno degli strumenti più efficaci per lo sviluppo di applicazioni Android di alta qualità.

IL PROGETTO

COS'È

Si tratta di un'applicazione Android innovativa progettata per assistere le persone che hanno difficoltà nella comunicazione verbale. Con questa app, gli utenti possono esprimersi in modo efficace e autonomo, selezionando le PECS (Picture Exchange Communication System) delle parole o frasi che desiderano utilizzare. L'app offre un'ampia gamma di immagini rappresentative, ciascuna associata a una parola o a un concetto specifico, che gli utenti possono facilmente selezionare tramite un'interfaccia intuitiva e user-friendly. Una volta effettuata la selezione, l'applicazione riproduce l'audio corrispondente, permettendo all'utente di comunicare verbalmente attraverso la voce sintetizzata dell'app.

Questa soluzione è particolarmente utile per persone con disturbi dello spettro autistico, afasia, o altre condizioni che compromettono la capacità di parlare. Grazie alla tecnologia di sintesi vocale integrata, l'app non solo facilita la comunicazione in tempo reale, ma contribuisce anche a migliorare la qualità della vita degli utenti, favorendo l'inclusione sociale e l'autonomia personale. L'applicazione è altamente personalizzabile, consentendo agli utenti o ai loro caregiver di aggiungere nuove immagini e parole, adattandosi così alle esigenze specifiche di ciascun individuo. Inoltre, è compatibile con diversi dispositivi Android, rendendola accessibile a un'ampia gamma di utenti.

IL CODICE

MAIN ACTIVITY

```
package com.example.provamaturita

import android.app.Activity
import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.graphics.drawable.BitmapDrawable
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.widget.Button
import android.widget.ImageView
import android.widget.LinearLayout
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.result.contract.ActivityResultContracts
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
```

```
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.provamaturita.ui.theme.ProvaMaturitaTheme
import java.util.concurrent.Executors
import com.example.provamaturita.SharedPreferencesHelper
import com.google.gson.Gson
import com.google.gson.reflect.TypeToken
import android.speech.tts.TextToSpeech
import android.util.Log
import java.util.*


class MainActivity : ComponentActivity(),
CustomAdapter.OnItemClickListener, TextToSpeech.OnInitListener {


    private lateinit var boxLayout: LinearLayout
    private lateinit var recyclerView: RecyclerView
    private val colors = arrayOf("#d2b491", "#e95b55", "#b485f8",
"#94cafa", "#83c972", "#f5cecc")
    private lateinit var sharedPreferences: SharedPreferences
    private lateinit var tts: TextToSpeech
    private var imagesnames = mutableListOf<String>()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        tts = TextToSpeech(this, this)
        boxLayout = findViewById(R.id.boxLayout)
        sharedPreferences = getSharedPreferences("ClickedImages",
Context.MODE_PRIVATE)

        clearStoredImages()

        recyclerView =
findViewById<RecyclerView>(R.id.recycler_view)
        recyclerView.layoutManager = LinearLayoutManager(this)

        val listofPECS = ArrayList<PECS>()

        listofPECS.add(
            PECS(
                R.drawable.azioni,
                "Azioni",
```

```
                listOf(R.drawable.andare, R.drawable.desiderare,
R.drawable.nonvoglio, R.drawable.bere, R.drawable.mangiare,
R.drawable.vestirsi),
                listOf("Voglio andare", "Voglio", "Non voglio",
"Bere", "Mangiare", "Vestirsi")
            )
        )

        /*audio/scritta -> andare = voglio andare
        */


        listofPECS.add(
            PECS(
                R.drawable.cibo,
                "Cibo",
                listOf(R.drawable.acqua,
R.drawable.arancia,R.drawable.aranciata, R.drawable.banana,
R.drawable.biscotti,
                    R.drawable.caramelle, R.drawable.carne,
R.drawable.carota, R.drawable.cereali, R.drawable.ciliegie,
R.drawable.cioccolato,
                    R.drawable.coca, R.drawable.croissant,
R.drawable.formaggio, R.drawable.fragola, R.drawable.gelato,
R.drawable.hamburger,
                    R.drawable.latte, R.drawable.maccheroni,
R.drawable.mela, R.drawable.melone, R.drawable.muffin,
R.drawable.pandoro, R.drawable.pane,
                    R.drawable.panettone, R.drawable.panino,
R.drawable.patatine, R.drawable.pera, R.drawable.pesca,
R.drawable.pesce, R.drawable.patata, R.drawable.pizza,
                    R.drawable.pollo, R.drawable.pomodoro,
R.drawable.prosciutto, R.drawable.salame, R.drawable.succo,
R.drawable.torta, R.drawable.uovo,
                    R.drawable.yogurt, R.drawable.zucchina),
                listOf("Acqua", "Arancia", "Aranciata", "Banana",
"Biscotti", "Caramelle", "Carne", "Carota", "Cereali", "Ciliegie",
"Cioccolato",
                    "Coca Cola", "Merendina", "Formaggio",
"Fragola", "Gelato", "Hamburger", "Latte", "Pasta", "Mela",
"Melone", "Muffin", "Pandoro",
                    "Pane", "Panettone", "Panino", "Patatine",
"Pera", "Pesca", "Pesce", "Patata", "Pizza", "Pollo", "Pomodoro",
"Prosciutto",
                    "Salame", "Succo", "Torta", "Uovo", "Yogurt",
"Zucchina")
            )
        )

        listofPECS.add(
            PECS(
                R.drawable.oggetti,
```

```
                "Oggetti",
                listOf(R.drawable.bambola, R.drawable.bicchiere,
R.drawable.cartaig, R.drawable.colori, R.drawable.cellulare,
R.drawable.computer, R.drawable.cucchiaio,
                    R.drawable.dentifricio, R.drawable.foglio,
R.drawable.forchetta, R.drawable.libro, R.drawable.matita,
R.drawable.mattoncini,
                    R.drawable.palla, R.drawable.pettine,
R.drawable.piatto, R.drawable.puzzle, R.drawable.sapone,
R.drawable.seccpalet,
                    R.drawable.tablet, R.drawable.televisione,
R.drawable.zaino),
                listOf("Bambola", "Bicchiere", "Carta Igienica",
"Colori", "Cellulare", "Computer", "Cucchiaio", "Dentifricio",
"Foglio", "Forchetta",
                    "Libro", "Matita", "Costruzioni", "Palla",
"Pettine", "Piatto", "Puzzle", "Sapone", "Secchiello Paletta",
"Tablet", "TV", "Zaino")
            )
        )

        listofPECS.add(
            PECS(
                R.drawable.vestiti,
                "Vestiti",
                listOf(R.drawable.calzini, R.drawable.cappello,
R.drawable.cappotto, R.drawable.gonna, R.drawable.maglietta,
R.drawable.pantalone,
                    R.drawable.pigiama, R.drawable.scarpe,
R.drawable.sciarpa, R.drawable.mutande),
                listOf("Calzini", "Cappello", "Cappotto", "Gonna",
"Maglietta", "Pantalone", "Pigiama", "Scarpe", "Sciarpa",
"Mutande")

            )
        )

        listofPECS.add(
            PECS(
                R.drawable.luoghi,
                "Luoghi",
                listOf(R.drawable.bagno, R.drawable.camera,
R.drawable.casa, R.drawable.cinema, R.drawable.parco,
R.drawable.spiaggia,
                    R.drawable.cucina, R.drawable.salone,
R.drawable.montagna),
                listOf("Bagno", "Camera", "Casa", "Cinema",
"Parco", "Spiaggia", "Cucina", "Salone", "Montagna")
            )
        )

        listofPECS.add(
```

```
            PECS(
                R.drawable.persone,
                "Persone",
                listOf(R.drawable.bambina, R.drawable.bambino,
R.drawable.famiglia, R.drawable.mamma, R.drawable.padre,
R.drawable.nonna, R.drawable.nonno,
                    R.drawable.fratello, R.drawable.sorella),
                listOf("Bambina", "Bambino", "Famiglia", "Mamma",
"Papà", "Nonna", "Nonno", "Fratello", "Sorella")
            )
        )

        val customAdapter = CustomAdapter(listofPECS, colors)
        customAdapter.setOnItemClickListener(this)
        recyclerView.adapter = customAdapter

        val savedImages = getSavedImages()
        savedImages.forEach { imageResource ->
            addImageToBox(imageResource)
        }

        val speakButton: Button = findViewById(R.id.playaudio)
        speakButton.setOnClickListener {
            speakClickedImageNames()
        }

        val deleteButton: Button = findViewById(R.id.deleteall)
        deleteButton.setOnClickListener{
            clearStoredImages()
            imagesnames.clear()
        }

    }

    override fun OnItemClick(position: Int, item: PECS) {
        // Start a new activity (or fragment) to display grid of
images
        val intent = Intent(this, GridActivity::class.java)
        val backgroundColor = colors[position % colors.size]
        intent.putIntegerArrayListExtra("imageList",
ArrayList(item.imageList))
        intent.putStringArrayListExtra("textList",
ArrayList(item.text))
        intent.putExtra("backgroundColor", backgroundColor)
        gridActivityLauncher.launch(intent)
    }



    private val gridActivityLauncher =
registerForActivityResult(ActivityResultContracts.StartActivityFor
Result()) { result ->
```

```
        if (result.resultCode == Activity.RESULT_OK) {
            val data: Intent? = result.data
            val selectedImages =
data?.getIntegerArrayListExtra("ClickedImages")
            selectedImages?.forEach { imageResource ->
                addImageToBox(imageResource)
            }
        }
    }

    override fun onInit(status: Int) {
        if (status == TextToSpeech.SUCCESS) {
            // Set language for Text-to-Speech engine
            val result = tts.setLanguage(Locale.ITALIAN)// Check
if the language is supported
            if (result == TextToSpeech.LANG_MISSING_DATA || result
== TextToSpeech.LANG_NOT_SUPPORTED) {
                Log.e("TTS", "The Language specified is not
supported!")
            }
        } else {
            Log.e("TTS", "Initialization failed!")
        }
    }

    private fun speakClickedImageNames() {
        if (imagesnames.isNotEmpty()) {
            // Convert the list of clicked image names to a single
string
            val textToSpeak = imagesnames.joinToString(" ")
            // Speak the text
            tts.speak(textToSpeak, TextToSpeech.QUEUE_FLUSH, null,
"")
        } else {
            // If the list is empty, inform the user
            Toast.makeText(this, "No images clicked yet",
Toast.LENGTH_SHORT).show()
        }
    }

    private fun getClickedImageNames(): List<String> {
        val sharedPref =
getSharedPreferences("MySharedPreferences", Context.MODE_PRIVATE)
        val json = sharedPref.getString("clickedImageNames", null)
        return Gson().fromJson(json, object :
TypeToken<List<String>>() {}.type) ?: emptyList()
    }

    private fun getSavedImages(): List<Int> {
        val json = sharedPreferences.getString("imageList", null)
        return Gson().fromJson(json, object :
TypeToken<List<Int>>() {}.type) ?: emptyList()
```

```
    }

    private fun addImageToBox(imageResource: Int) {
        val imageView = ImageView(this).apply {
            setImageResource(imageResource)
            val clickedImageNames = getClickedImageNames()
            imagesnames.addAll(clickedImageNames.filter {
!imagesnames.contains(it) })
            val desiredWidth =
resources.getDimensionPixelSize(R.dimen.desired_image_width)
            val desiredHeight =
resources.getDimensionPixelSize(R.dimen.desired_image_height)

            // Scalare l'immagine alle dimensioni desiderate
            val drawable = this.drawable
            val scaledDrawable = BitmapDrawable(resources,
Bitmap.createScaledBitmap((drawable as BitmapDrawable).bitmap,
desiredWidth, desiredHeight, true))

            // Impostare l'immagine scalata
            setImageDrawable(scaledDrawable)

            layoutParams = LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.WRAP_CONTENT,
                LinearLayout.LayoutParams.WRAP_CONTENT
            ).apply {
            }
        }

        boxLayout.addView(imageView)
        boxLayout.requestLayout()
    }


    override fun onDestroy() {
        if (::tts.isInitialized) {
            tts.stop()
            tts.shutdown()
        }
        super.onDestroy()
        clearStoredImages()
    }

    private fun clearStoredImages() {
        sharedPreferences.edit().clear().apply()
        boxLayout.removeAllViews()
    }

}
```

Cosa fa il codice:

Il codice è una classe MainActivity che estende ComponentActivity. Questa classe implementa un'interfaccia CustomAdapter.OnItemClickListener per gestire i click sugli elementi del RecyclerView e TextToSpeech.OnInitListener per gestire la sintesi vocale.

Variabili

- boxLayout: Un LinearLayout per contenere le immagini cliccate.
- recyclerView: Un RecyclerView per visualizzare una lista di elementi PECS.
- colors: Un array di colori utilizzati per gli elementi.
- sharedPreferences: Utilizzato per salvare le immagini cliccate.
- tts: Un'istanza di TextToSpeech per la sintesi vocale.
- imagesnames: Una lista mutabile di nomi delle immagini cliccate.

Metodi Principali

onCreate

Questo metodo è chiamato quando l'attività viene creata.

- Inizializzazione di TextToSpeech.
- Configurazione di boxLayout e sharedPreferences.
- Chiamata al metodo clearStoredImages per pulire eventuali immagini salvate in precedenza.
- Configurazione del RecyclerView.
- Creazione di una lista di oggetti PECS con immagini e testi.
- Configurazione di CustomAdapter e impostazione dell'adattatore per il RecyclerView.
- Recupero delle immagini salvate e aggiunta al boxLayout.
- Configurazione dei pulsanti speakButton e deleteButton con i relativi listener.

OnItemClick

Questo metodo è chiamato quando un elemento del RecyclerView viene cliccato.

- Avvia una nuova attività GridActivity passando la lista delle immagini e dei testi dell'elemento cliccato e il colore di sfondo.

onInit

Questo metodo è chiamato quando TextToSpeech è inizializzato.

- Imposta la lingua della sintesi vocale su italiano.
- Gestisce gli errori di inizializzazione.

speakClickedImageNames

Questo metodo converte i nomi delle immagini cliccate in una stringa e li pronuncia tramite TextToSpeech.

getClickedImageNames

Questo metodo recupera i nomi delle immagini cliccate da SharedPreferences.

getSavedImages

Questo metodo recupera le immagini salvate da SharedPreferences.

addImageToBox

Questo metodo aggiunge un'immagine al boxLayout e scala l'immagine alle dimensioni desiderate.

onDestroy

Questo metodo è chiamato quando l'attività viene distrutta.

- Ferma e chiude TextToSpeech.
- Chiama clearStoredImages per pulire le immagini salvate.

clearStoredImages

Questo metodo pulisce le immagini salvate e rimuove tutte le viste da boxLayout.

Processo Complessivo

- Creazione della Vista: onCreate configura l'attività, inizializza TextToSpeech, configura il RecyclerView con una lista di oggetti PECS, recupera le immagini salvate e configura i pulsanti.
- Gestione dei Click: OnItemClick avvia una nuova attività per visualizzare una griglia di immagini.
- Sintesi Vocale: onInit configura TextToSpeech e speakClickedImageNames pronuncia i nomi delle immagini cliccate.
- Gestione delle Immagini: getClickedImageNames e getSavedImages recuperano i dati da SharedPreferences, addImageToBox aggiunge un'immagine al layout e clearStoredImages pulisce le immagini salvate.
- Chiusura dell'Attività: onDestroy gestisce la chiusura di TextToSpeech e pulisce le immagini salvate.

CUSTOM ADAPTER

```
package com.example.provamaturita

import android.graphics.Color
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.view.animation.Animation
import android.view.animation.ScaleAnimation
import android.widget.ImageView
import android.widget.TextView
import androidx.cardview.widget.CardView
import androidx.recyclerview.widget.RecyclerView

class CustomAdapter(private val courseList: List<PECS>, private
val backgroundColors: Array<String>):
RecyclerView.Adapter<CustomAdapter.ViewHolder>(){

    interface OnItemClickListener{
        fun OnItemClick(position: Int, item: PECS)
    }

    private var itemClickListener: OnItemClickListener? = null

    fun setOnItemClickListener(listener: OnItemClickListener){
        this.itemClickListener = listener
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): ViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.custom_layout
,parent, false)
        return ViewHolder(view)
    }

    override fun getItemCount(): Int {
        return courseList.size
    }

    override fun onBindViewHolder(holder: ViewHolder, position:
Int) {
        val currentPECS = courseList[position]
        val backgroundColor = backgroundColors[position %
backgroundColors.size]
        val image = currentPECS.image
        val head = currentPECS.catImg

        holder.imageView.setImageResource(image)
        holder.headTextView.text = head
```

```
holder.cardView.setBackgroundColor(Color.parseColor(backgroundColo
r))

        setAnimation(holder.itemView)

        holder.itemView.setOnClickListener {
            itemClickListener?.OnItemClick(position, currentPECS)
        }
    }

    private fun setAnimation(view : View){
        val anim = ScaleAnimation(
            0.0f,
            1.0f,
            0.0f,
            1.0f,
            Animation.RELATIVE_TO_SELF,
            0.5f,
            Animation.RELATIVE_TO_SELF,
            0.5f
        )
        anim.duration = 900
        view.startAnimation(anim)
    }

    class ViewHolder(itemView: View):
RecyclerView.ViewHolder(itemView){
        val imageView =
itemView.findViewById<ImageView>(R.id.image)
        val headTextView =
itemView.findViewById<TextView>(R.id.head)
        val cardView = itemView.findViewById<CardView>(R.id.cibo)
    }
}
```

Cosa fa il codice:

Il codice implementa un adattatore personalizzato per un RecyclerView. L'adattatore, chiamato CustomAdapter, è responsabile di legare i dati (lista di oggetti PECS) alle viste (view).

Variabili

- courseList: Una lista di oggetti PECS da visualizzare.
- backgroundColors: Un array di stringhe che rappresentano i colori di sfondo da applicare alle card view.

Interfaccia OnItemClickListener

- Questa interfaccia definisce un metodo OnItemClick per gestire i click sugli elementi del RecyclerView.

- itemClickListener: Una variabile privata di tipo OnItemClickListener per memorizzare il listener che sarà impostato esternamente.

Metodi Principali

setOnItemClickListener

Imposta il listener per gli eventi di click sugli elementi del RecyclerView.

onCreateViewHolder

Crea una nuova vista per un elemento del RecyclerView utilizzando il layout custom_layout.

getItemCount

Restituisce il numero di elementi nella lista courseList.

onBindViewHolder

- Lega i dati di un elemento specifico (currentPECS) alla vista holder.
- Imposta l'immagine, il testo e il colore di sfondo della CardView.
- Aggiunge un'animazione di scala (ScaleAnimation) alla vista dell'elemento.
- Imposta un listener di click sulla vista dell'elemento per notificare l'itemClickListener quando un elemento viene cliccato.

setAnimation

Applica un'animazione di scala alla vista dell'elemento con una durata di 900 millisecondi.

Classe Interna ViewHolder

La classe ViewHolder estende RecyclerView.ViewHolder e contiene i riferimenti alle viste che compongono l'elemento del RecyclerView.

- imageView: Un ImageView per visualizzare l'immagine associata a PECS.
- headTextView: Un TextView per visualizzare il testo associato a PECS.
- cardView: Un CardView che contiene l'intera vista dell'elemento e a cui viene applicato il colore di sfondo.

Processo Complessivo

- Creazione della Vista: onCreateViewHolder è chiamato per creare una nuova vista per ogni elemento, utilizzando il layout custom_layout.
- Conteggio degli Elementi: getItemCount restituisce il numero di elementi nella lista courseList.
- Associazione dei Dati alla Vista: onBindViewHolder è chiamato per ogni elemento visibile per associare i dati dell'elemento alla vista.
- Animazione: Ogni volta che un elemento viene associato a una vista, viene applicata un'animazione di scala.

- Gestione dei Click: Quando un elemento viene cliccato, viene notificato il listener impostato tramite setOnItemClickListener.

GRID ACTIVITY

```
package com.example.provamaturita

import android.app.Activity
import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.graphics.Bitmap
import android.graphics.Color
import android.graphics.drawable.BitmapDrawable
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.GridLayout
import android.widget.GridView
import android.widget.ImageView
import android.widget.LinearLayout
import android.widget.RelativeLayout
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.gson.Gson
import com.google.gson.reflect.TypeToken


class GridActivity : AppCompatActivity(),
GridAdapter.OnItemClickListener {
    private var clickedImages = mutableListOf<Int>()
    private lateinit var boxLayout: LinearLayout
    private lateinit var sharedPreferences: SharedPreferences

    private val imageNamesMap = mapOf(
        R.drawable.andare to "Voglio andare",
        R.drawable.desiderare to "Voglio",
        R.drawable.nonvoglio to "Non voglio",
        R.drawable.bere to "Bere",
        R.drawable.mangiare to "Margiare",
        R.drawable.vestirsi to "Vestirmi",
        R.drawable.acqua to "Acqua",
        R.drawable.arancia to "Arancia",
        R.drawable.aranciata to "Aranciata",
        R.drawable.banana to "Banana",
        R.drawable.biscotti to "Biscotti",
        R.drawable.caramelle to "Caramelle",
        R.drawable.carne to "Carne",
        R.drawable.carota to "Carote",
```

```
R.drawable.cereali to "Cereali",
R.drawable.ciliegie to "Ciliegie",
R.drawable.cioccolato to "Cioccolato",
R.drawable.coca to "Coca Cola",
R.drawable.croissant to "Merendina",
R.drawable.formaggio to "Formaggio",
R.drawable.fragola to "Fragola",
R.drawable.gelato to "Gelato",
R.drawable.hamburger to "Hamburger",
R.drawable.latte to "Latte",
R.drawable.maccheroni to "Pasta",
R.drawable.mela to "Mela",
R.drawable.melone to "Melone",
R.drawable.muffin to "Muffin",
R.drawable.pandoro to "Pandoro",
R.drawable.pane to "Pane",
R.drawable.panettone to "Panettone",
R.drawable.panino to "Panino",
R.drawable.patatine to "Patatine",
R.drawable.pera to "Pera",
R.drawable.pesca to "Pesca",
R.drawable.pesce to "Pesce",
R.drawable.patata to "Patata",
R.drawable.pizza to "Pizza",
R.drawable.pollo to "Pollo",
R.drawable.pomodoro to "Pomodoro",
R.drawable.prosciutto to "Prosciutto",
R.drawable.salame to "Salame",
R.drawable.succo to "Succo",
R.drawable.torta to "Torta",
R.drawable.uovo to "Uovo",
R.drawable.yogurt to "Yogurt",
R.drawable.zucchina to "Zucchina",
R.drawable.bambola to "Bambola",
R.drawable.bicchiere to "Bicchiere",
R.drawable.cartaig to "Carta Igienica",
R.drawable.colori to "Colori",
R.drawable.cellulare to "Cellulare",
R.drawable.computer to "Computer",
R.drawable.cucchiaio to "Cucchiaio",
R.drawable.dentifricio to "Dentifricio",
R.drawable.foglio to "Foglio",
R.drawable.forchetta to "Forchetta",
R.drawable.libro to "Libro",
R.drawable.matita to "Matita",
R.drawable.mattoncini to "Costruzioni",
R.drawable.palla to "Palla",
R.drawable.pettine to "Pettine",
R.drawable.piatto to "Piatto",
R.drawable.puzzle to "Puzzle",
R.drawable.sapone to "Sapone",
R.drawable.seccpalet to "Secchiello e paletta",
```

```
        R.drawable.tablet to "Tablet",
        R.drawable.televisione to "TV",
        R.drawable.zaino to "Zaino",
        R.drawable.calzini to "Calzini",
        R.drawable.cappello to "Cappello",
        R.drawable.cappotto to "Cappotto",
        R.drawable.gonna to "Gonna",
        R.drawable.maglietta to "Maglietta",
        R.drawable.pantalone to "Pantalone",
        R.drawable.pigiama to "Pigiama",
        R.drawable.scarpe to "Scarpe",
        R.drawable.sciarpa to "Sciarpa",
        R.drawable.mutande to "Mutande",
        R.drawable.bagno to "Bagno",
        R.drawable.camera to "Camera",
        R.drawable.casa to "Casa",
        R.drawable.cinema to "Cinema",
        R.drawable.parco to "Parco",
        R.drawable.spiaggia to "Spiaggia",
        R.drawable.cucina to "Cucina",
        R.drawable.salone to "Salone",
        R.drawable.montagna to "Montagna",
        R.drawable.bambina to "Bambina",
        R.drawable.bambino to "Bambino",
        R.drawable.famiglia to "Famiglia",
        R.drawable.mamma to "Mamma",
        R.drawable.padre to "Papà",
        R.drawable.nonna to "Nonna",
        R.drawable.nonno to "Nonno",
        R.drawable.fratello to "Fratello",
        R.drawable.sorella to "Sorella"
    )

    private val clickedImageNames = mutableListOf<String>()


    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_grid)

        val sharedLayout = findViewById<View>(R.id.sharedLayout)
        val button1 =
sharedLayout.findViewById<Button>(R.id.deleteall)
        val button2 =
sharedLayout.findViewById<Button>(R.id.playaudio)

        button1.visibility = View.GONE
        button2.visibility = View.GONE

        boxLayout = findViewById(R.id.boxLayout)
        sharedPreferences = getSharedPreferences("ClickedImages",
Context.MODE_PRIVATE)
```

```
        val imageList =
intent.getIntegerArrayListExtra("imageList") ?: ArrayList()
        val textList = intent.getStringArrayListExtra("textList")
?: ArrayList()

        val recyclerView =
findViewById<RecyclerView>(R.id.gridRecyclerView)
        recyclerView.layoutManager = GridLayoutManager(this, 3)

        val adapter = GridAdapter(imageList, textList, this)
        recyclerView.adapter = adapter

        val savedImages = getClickedImages()
        savedImages.forEach { imageId ->
            addImageToBox(imageId)
        }

        val backgroundColor =
intent.getStringExtra("backgroundColor")

findViewById<RelativeLayout>(R.id.rootgrid).setBackgroundColor(Col
or.parseColor(backgroundColor))

        val backButton = findViewById<Button>(R.id.backButton)

        // Set an OnClickListener for the back button
        backButton.setOnClickListener {

            clickedImages = getClickedImages().toMutableList()
            saveClickedImages(clickedImages)
            val clickedNames =
getClickedImageName(clickedImageNames).toMutableList()
            saveClickedImageNames(clickedNames)
            val resultIntent = Intent()
            resultIntent.putIntegerArrayListExtra("ClickedImages",
ArrayList(clickedImages)) // Use "selectedImages" key
            setResult(Activity.RESULT_OK, resultIntent)
            finish()

        }

    }

    override fun onItemClick(imageId: Int) {
        clickedImages = getClickedImages().toMutableList()
        clickedImages.add(imageId)
        saveClickedImages(clickedImages)

        val imageName = imageNamesMap[imageId]
        imageName?.let {
            clickedImageNames.add(it)
```

```
        }
        saveClickedImageNames(clickedImageNames)

        // Creare un nuovo ImageView
        val imageView = ImageView(this).apply {
            // Impostare l'immagine
            setImageResource(imageId)

            // Calcolare le dimensioni desiderate per l'immagine
            val desiredWidth =
resources.getDimensionPixelSize(R.dimen.desired_image_width)
            val desiredHeight =
resources.getDimensionPixelSize(R.dimen.desired_image_height)

            // Scalare l'immagine alle dimensioni desiderate
            val drawable = this.drawable
            val scaledDrawable = BitmapDrawable(resources,
Bitmap.createScaledBitmap((drawable as BitmapDrawable).bitmap,
desiredWidth, desiredHeight, true))

            // Impostare l'immagine scalata
            setImageDrawable(scaledDrawable)

            layoutParams = LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.WRAP_CONTENT,
                LinearLayout.LayoutParams.WRAP_CONTENT
            ).apply {
            }

            // Impostare l'azione onClick per rimuovere l'immagine
            setOnClickListener {
                boxLayout.removeView(this)
                clickedImages.remove(imageId)
                clickedImageNames.remove(imageNamesMap[imageId])
                saveClickedImages(clickedImages)
                saveClickedImageNames(clickedImageNames)

            }
        }

        // Aggiungere l'ImageView al layout della box
        boxLayout.addView(imageView)
        boxLayout.requestLayout()


    }
```

```
    private fun saveClickedImages(clickedImages: List<Int>) {
        val editor = sharedPreferences.edit()
        val json = Gson().toJson(clickedImages)
        editor.putString("imageList", json)
        editor.apply()
    }

    private fun saveClickedImageNames(imageNames: List<String>) {
        val sharedPref =
getSharedPreferences("MySharedPreferences", Context.MODE_PRIVATE)
        val editor = sharedPref.edit()
        val json = Gson().toJson(imageNames)
        editor.putString("clickedImageNames", json)
        editor.apply()
    }

    private fun getClickedImages(): List<Int> {
        val json = sharedPreferences.getString("imageList", null)
        return Gson().fromJson(json, object :
TypeToken<List<Int>>() {}.type) ?: emptyList()
    }

    private fun getClickedImageName(clickedImageNames:
List<String>): List<String>{
        return clickedImageNames
    }


    private fun addImageToBox(imageId: Int) {
        val imageView = ImageView(this).apply {
            setImageResource(imageId)
            layoutParams = LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.WRAP_CONTENT,
                LinearLayout.LayoutParams.WRAP_CONTENT
            )
        }
        boxLayout.addView(imageView)
        boxLayout.requestLayout()
    }

    override fun onDestroy() {
        super.onDestroy()
        clearStoredImages()
    }

    private fun clearStoredImages() {
        sharedPreferences.edit().clear().apply()
    }
}
```

Cosa fa il codice:

Il codice implementa un'attività chiamata GridActivity. Questa attività visualizza una griglia di immagini (utilizzando un RecyclerView con layout a griglia) e consente agli utenti di selezionare immagini, che vengono poi visualizzate in un layout aggiuntivo (boxLayout). Le immagini selezionate vengono salvate utilizzando SharedPreferences per mantenere la selezione anche dopo che l'attività è stata chiusa.

Variabili e Costanti

- clickedImages: Una lista di ID di immagini selezionate dagli utenti.
- boxLayout: Un layout lineare dove vengono aggiunte le immagini selezionate.
- sharedPreferences: Per salvare e recuperare le immagini selezionate.
- imageNamesMap: Una mappa che associa gli ID delle immagini alle stringhe di descrizione.

Metodi principali

Metodo onCreate

- Layout e Bottoni: Imposta il layout dell'attività (R.layout.activity_grid) e inizializza i bottoni per eliminare tutte le selezioni (deleteall) e per riprodurre audio (playaudio), che vengono nascosti all'inizio.
- RecyclerView: Configura un RecyclerView con un layout manager a griglia (GridLayoutManager) e un adattatore (GridAdapter).
- Recupero Immagini Salvate: Recupera le immagini salvate dalle SharedPreferences e le aggiunge a boxLayout.
- Colore di Sfondo: Imposta il colore di sfondo dell'attività.
- Bottone Indietro: Configura il bottone indietro per salvare le immagini selezionate e ritornare all'attività precedente con i risultati.

Metodo onItemClick

- Aggiunta di Immagini: Quando un'immagine nella griglia viene cliccata, viene aggiunta a clickedImages e clickedImageNames, e viene visualizzata in boxLayout.
- Salvataggio di Immagini: Le immagini selezionate e i loro nomi vengono salvati nelle SharedPreferences.
- Impostazione e Dimensionamento di ImageView: Crea un nuovo ImageView per l'immagine cliccata, la scala alle dimensioni desiderate e la aggiunge a boxLayout. Aggiunge un listener di click per rimuovere l'immagine quando viene cliccata.

Metodi di Salvataggio e Recupero

- saveClickedImages: Salva la lista di immagini selezionate nelle SharedPreferences come stringa JSON.
- saveClickedImageNames: Salva la lista dei nomi delle immagini selezionate nelle SharedPreferences.
- getClickedImages: Recupera la lista di immagini selezionate dalle SharedPreferences.
- getClickedImageName: Restituisce la lista dei nomi delle immagini selezionate.

- addImageToBox: Aggiunge un'immagine a boxLayout.

Metodi di Pulizia

- onDestroy: Cancella le immagini salvate quando l'attività viene distrutta.
- clearStoredImages: Rimuove tutte le immagini salvate dalle SharedPreferences.

Processo complessivo

- Visualizzazione Griglia: Visualizza una griglia di immagini da cui l'utente può selezionare.
- Selezione e Salvataggio: Le immagini selezionate vengono visualizzate in boxLayout e salvate in SharedPreferences.
- Recupero Selezioni: Le selezioni salvate vengono recuperate e visualizzate quando l'attività viene ricreata.
- Rimozione di Immagini: Gli utenti possono rimuovere le immagini selezionate cliccando su di esse in boxLayout.
- Persistenza dei Dati: Mantiene le selezioni anche dopo la chiusura dell'attività.

GRID ADAPTER

```
package com.example.provamaturita

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView


class GridAdapter(private val imageList: List<Int>, private val
textList: List<String>, private val itemClickListener:
OnItemClickListener) :
    RecyclerView.Adapter<GridAdapter.ViewHolder>() {

    private val clickedImages = mutableListOf<Int>()


    interface OnItemClickListener{
        fun onItemClick(imageId: Int)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): ViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.cucstom_laygr
id, parent, false)
        return ViewHolder(view)
    }
```

```
    override fun getItemCount() = imageList.size

    override fun onBindViewHolder(holder: ViewHolder, position:
Int) {
        val imageResource = imageList[position]
        val text = textList[position]
        holder.imageView.setImageResource(imageResource)
        holder.textView.text = text

        holder.itemView.setOnClickListener{
            itemClickListener.onItemClick(imageResource)
            clickedImages.add(imageResource)
        }
    }


    fun getClickedImages(): List<Int> {
        return clickedImages.toList()
    }

    class ViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        val imageView: ImageView =
itemView.findViewById(R.id.imageView)
        val textView: TextView = itemView.findViewById(R.id.nome)
    }
}
```

Cosa fa il codice:

Il codice implementa un adattatore chiamato GridAdapter per un RecyclerView. Questo adattatore è progettato per visualizzare una griglia di immagini con testo associato, permettendo all'utente di cliccare sulle immagini per selezionarle. Ecco una spiegazione dettagliata delle componenti del codice:

Classe GridAdapter

La classe GridAdapter estende RecyclerView.Adapter e utilizza un ViewHolder interno per gestire le viste degli elementi.

Variabili

- imageList: Una lista di ID delle risorse delle immagini da visualizzare.
- textList: Una lista di stringhe di testo associate alle immagini.
- itemClickListener: Un listener per gestire gli eventi di click sugli elementi.
- clickedImages: Una lista di immagini cliccate.

Interfaccia OnItemClickListener

Questa interfaccia definisce un metodo onItemClick per gestire i click sugli elementi del RecyclerView.

Metodi Principali

onCreateViewHolder

Crea una nuova vista per un elemento del RecyclerView utilizzando il layout cucstom_laygrid.

getItemCount

Restituisce il numero di elementi nella lista imageList.

onBindViewHolder

- Associa i dati di un elemento specifico (immagine e testo) alla vista holder.
- Imposta l'immagine e il testo per l'elemento corrente.
- Aggiunge un listener di click alla vista dell'elemento per notificare l'itemClickListener quando l'elemento viene cliccato, aggiungendo l'immagine cliccata a clickedImages.

getClickedImages

Restituisce una lista delle immagini cliccate.

Classe Interna ViewHolder

La classe ViewHolder estende RecyclerView.ViewHolder e contiene i riferimenti alle viste che compongono l'elemento del RecyclerView.

- imageView: Un ImageView per visualizzare l'immagine associata all'elemento.
- textView: Un TextView per visualizzare il testo associato all'elemento.

Processo Complessivo

- Creazione della Vista: onCreateViewHolder è chiamato per creare una nuova vista per ogni elemento, utilizzando il layout cucstom_laygrid.
- Conteggio degli Elementi: getItemCount restituisce il numero di elementi nella lista imageList.
- Associazione dei Dati alla Vista: onBindViewHolder è chiamato per ogni elemento visibile per associare i dati dell'elemento alla vista.
- Gestione dei Click: Quando un elemento viene cliccato, viene notificato il listener impostato tramite itemClickListener, e l'ID dell'immagine cliccata viene aggiunto a clickedImages.

CLASSE PECS

```
package com.example.provamaturita

data class PECS (
    val image: Int,
    val catImg: String,
    val imageList: List<Int>,
    val text: List<String>
)
```

Cosa fa il codice:

La classe PECS è una data class in Kotlin che rappresenta un oggetto con le seguenti proprietà:

- image: Un Int che rappresenta l'immagine della categoria.
- catImg: Una String che rappresenta il nome della categoria.
- imageList: Una lista di Int che rappresenta le immagini degli elementi nella categoria.
- text: Una lista di String che rappresenta i testi degli elementi nella categoria.

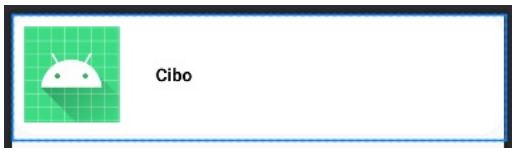
MAIN ACTIVITY LAYOUT (CUSTOM_LAYOUT + ACTIVITY_MAIN)

CUSTOM_LAYOUT

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/mainLayout"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

        <androidx.cardview.widget.CardView
            android:id="@+id/cibo"
            app:cardCornerRadius="20dp"
            android:layout_width="match_parent"
            android:layout_height="100dp"
            android:layout_marginBottom="5dp">

            <RelativeLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content">

            <ImageView
                android:id="@+id/image"
                android:layout_width="80dp"
                android:layout_height="100dp"
                android:layout_marginLeft="10dp"
```

```
                android:src="@mipmap/ic_launcher"/>

            <TextView
                android:id="@+id/head"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Cibo"
                android:layout_marginLeft="30dp"
                android:layout_marginVertical="40dp"
                android:textSize="15sp"
                android:textStyle="bold"
                android:textColor="#000000"
                android:layout_toRightOf="@id/image"
                />
            </RelativeLayout>
        </androidx.cardview.widget.CardView>

</RelativeLayout>
```

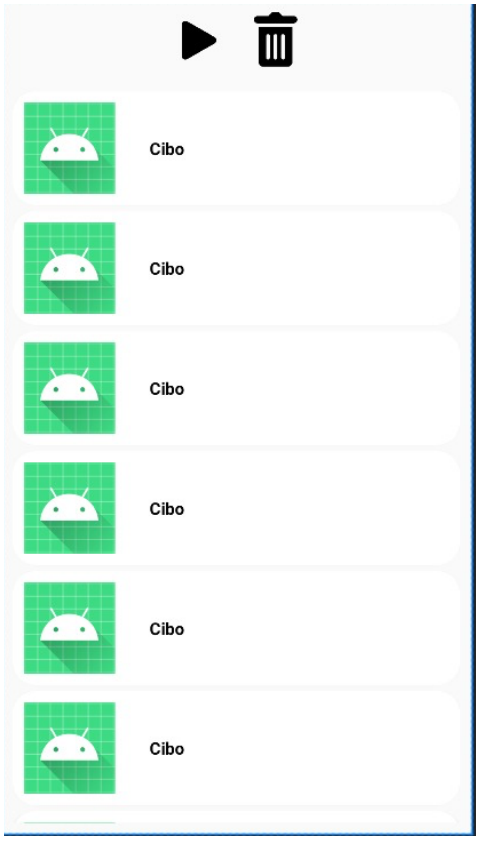
RISULTATO:



MAIN_ACTIVITY

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <include
        layout="@layout/shared_box_buttons"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycler_view"
        android:padding="10dp"
        tools:listitem="@layout/custom_layout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/sharedLayout"/>

</RelativeLayout>
```

RISULTATO:



SCHERMATA PRINCIPALE MAIN ACTIVITY

GRID ACTIVITY LAYOUT (CUCSTOM_LAYGRID + ACTIVITY_GRID)

CUCSTOM_LAYGRID

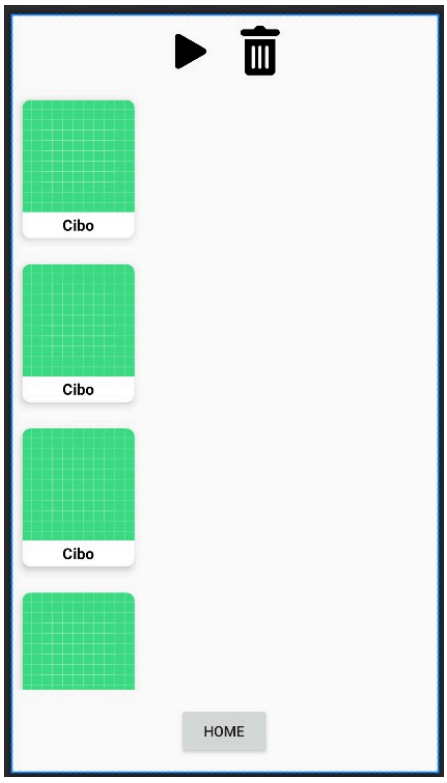
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/cardView"
    android:layout_width="wrap_content"
    android:layout_height="150dp"
    android:layout_margin="4dp"
    app:cardCornerRadius="8dp"
    app:cardElevation="4dp"
    app:cardUseCompatPadding="true">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:adjustViewBounds="true"
        android:scaleType="fitCenter"
        android:src="@drawable/ic_launcher_background" />
    <TextView
        android:id="@+id/nome"
        android:layout_width="wrap_content"
        android:layout_height="20dp"
        android:text="Cibo"
        android:layout_gravity="center|top"
        android:layout_marginVertical="110dp"
        android:textSize="15sp"
        android:textStyle="bold"
        android:textColor="#000000"
        />

</androidx.cardview.widget.CardView>
```

RISULTATO:

ACTIVITY_GRID

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/rootgrid"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <include
        layout="@layout/shared_box_buttons"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_above="@+id/rootgrid"/>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/gridRecyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@id/backButton"
        android:layout_below="@+id/sharedLayout"
        tools:listitem="@layout/cucstom_laygrid"/>

    <Button
        android:id="@+id/backButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Home"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_margin="16dp"/>

</RelativeLayout>
```

RISULTATO:



SCHERMATE PRINCIPALI GRID_ACTIVITY (AZIONI, CIBO, OGGETTI,VESTITI, LUOGHI, PERSONE)

LAYOUT CONDIVISO (SHARED_BOX_BUTTONS)

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/sharedLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <!-- Box -->
    <LinearLayout
        android:id="@+id/boxLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:gravity="center"
        android:orientation="horizontal">

        <!-- ImageView per l'immagine -->
        <ImageView
            android:id="@+id/boxImageView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:adjustViewBounds="true"
            android:scaleType="fitCenter"
            android:visibility="gone" />


    </LinearLayout>
```

```
    <!-- Button -->
    <Button
        android:id="@+id/playaudio"
        android:layout_width="70dp"
        android:layout_height="70dp"
        android:layout_below="@id/boxLayout"
        android:layout_marginLeft="135dp"
        android:background="@drawable/play"/>

    <Button
        android:id="@+id/deleteall"
        android:layout_width="70dp"
        android:layout_height="70dp"
        android:layout_below="@id/boxLayout"
        android:layout_toRightOf="@id/playaudio"
        android:background="@drawable/bin"/>

</RelativeLayout>
```

RISULTATO: